Bachelor's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Vegard Johansen
Nicolai Andre Olsen
Joachim Olerud Milward

Enhancing Chat Moderation with Soft Biometric Keystroke Dynamics

Bachelor's thesis in BIDATA
Supervisor: Sony George
May 2024

NTNU
Norwegian University of
Science and Technology

Vegard Johansen
Nicolai Andre Olsen
Joachim Olerud Milward

Enhancing Chat Moderation with Soft Biometric Keystroke Dynamics

Bachelor's thesis in BIDATA
Supervisor: Sony George
May 2024

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Enhancing Chat Moderation with Soft Biometric Keystroke Dynamics

Joachim Olerud Milward Vegard Johansen Nicolai Andre Olsen

May 21, 2024

Abstract

Supporting chat moderators by reducing their workload in removing unwanted behavior and eliminating cyber grooming in game communities is essential for maintaining user-friendliness and welcoming online gaming environments.

Through our work, we propose a method for extracting Soft Biometric Keystroke Dynamics data from third-party chat applications in Unity. This data could then processed using AIBA's AI algorithm to assess whether individuals are the age and gender they claim to be online.

In this project, we have developed an Software Development Kit (SDK) that integrates seamlessly into most Unity chat applications. We also completed an overview of upcoming regulations, including the EU Digital Services Act (DSA), Online Safety Act 2023 (OSA), US Kids Online Safety Act (KOSA), and EU Artificial Intelligence Act (EU AI Act), and assessed their implications for service providers. Additionally, we conducted a market scan to analyze the current age verification methods used by other enterprises.

The report will also discuss the measures taken to efficiently transfer data from the SDK to a server, minimizing the bandwidth usage when transferring this data

Overall, the thesis addresses the importance of exploring new ways to help combat unwanted behavior online.

Sammendrag

Å støtte chattemoderatorer ved å redusere arbeidsmengden deres når det gjelder å fjerne uønsket innhold og eliminere nettgrooming i spillsamfunn er et essentielt tiltak for å opprettholde brukervennlige og innbydende spillmiljøer på nettet.

Gjennom vårt prosjekt foreslår vi en metode for å hente ut Soft Biometric Keystroke Dynamics-data fra tredjeparts chat-applikasjoner i Unity. Disse dataene vil deretter behandles ved hjelp av AIBAs KI-algoritme for å vurdere om enkeltpersoner er den alderen og det kjønnet de hevder å være på nettet.

I dette prosjektet har vi utviklet en Software Development Kit (SDK) som integreres sømløst i de fleste Unity-chatapplikasjoner. Vi har også utarbeidet en oversikt over kommende reguleringer, inklusivt EU Digital Services Act (DSA), Online Safety Act 2023 (OSA), US Kids Online Safety Act (KOSA) og EU Artificial Intelligence Act (EU AI Act), og vurdert hvilke konsekvenser de vil få for tjenesteleverandører. I tillegg har vi gjennomført en markedsanalyse for å kartlegge hvilke verifiserings-metoder for alder som brukes av i dagens marked av andre virksomheter.

Rapporten vil også diskutere tiltakene som er iverksatt for å overføre data fra SDKen til en server på en effektiv måte, og minimere båndbreddebruken ved overføring av disse dataene.

Samlet sett tar avhandlingen for seg viktigheten av å utforske nye måter å bekjempe uønsket atferd på nettet.

Acknowledgements

We would like to thank our supervisor Sony George for his guidance with this thesis. We also extend our gratitude to Gard Støe and Patrick Bours for their assistance with the topic of Soft Biometric Keystroke Dynamics.

Additionally, we appreciate the Unity Developer forum for their providing guidance where the documentation did not. Finally, we thank our girlfriends for their patience and support in these trying times.

Contents

Figures

Tables

Code Listings

Acronyms

.NET .NET Framework. 66, 67, 93

ADL Anti-defamation League. 40–42

AI Artificial Intelligence. iii, 1, 37, 50, 52–55, 95, 100, 101, 103

API Application Programming Interface. xvii, 2, 6, 13, 14, 19–22, 26, 36, 61, 63–65, 67, 73–78, 85, 87, 89, 90, 94, 95, 99, 102, 125

AWS Amazon Web Services. 89, 90

COPPA Children's Online Privacy Protection Act. 5, 33, 47, 95, 101

CPU Central Processing Unit. 31, 68

CSS Cascading Style Sheets. 70

CSV Comma-Separated Values. 26–29, 31, 32

DSA EU Digital Services Act. iii, v, 2, 3, 5, 47–49, 61, 87, 95, 100, 101, 103

EU European Union. 3, 48, 52, 101

EU AI Act EU Artificial Intelligence Act. iii, v, 2, 5, 13, 47, 52–60, 95, 100, 101, 103

FAR False Acceptance Rate. 24

FRR False Rejection Rate. 24

GDPR General Data Protection Regulation. 5, 47, 101

git https://en.wikipedia.org/wiki/Git#Naming. 67

GPU Graphics Processing Unit. 68

HTML HyperText Markup Language. 70

HTTP Hypertext Transfer Protocol. 2, 19, 64, 65, 75, 89, 90, 99

HTTPS Hypertext Transfer Protocol Secure. 21, 90, 97, 99, 102

IaaS Infrastructure as a Service. 89

IDE Integrated Development Environment. 6, 68, 97

IMGUI Immediate Mode GUI. 69, 98

ISP Internet Service Providers. 51

JSON JavaScript Object Notation. 26, 28, 29, 31, 32, 64, 65, 73–75, 77, 90

KDE KeyDownEvent. 73, 81–83, 85

KOSA US Kids Online Safety Act. iii, v, 2, 5, 13, 47, 51, 95, 100, 101, 103

KPM Keystrokes per minute. 82, 83

KUE KeyUpEvent. 73, 81–83, 85

LINQ Language Integrated Query. 66, 82

LLMS Large Language Models. 39

MMS Multimedia Messaging Service. 49

MOOC Massive Open Online Courses. 24

NLP Natural Language Processing. 39

NoSQL not only Structured Query Language. 19, 75, 77

NTNU Norwegian University of Science and Technology. 1, 93

OFCOM Office of Communications. 50

OS Operating System. 22, 51, 66

OSA Online Safety Act 2023. iii, v, 2, 5, 47, 49, 50, 95, 100, 101

PaaS Platform as a Service. 89

POC Proof of concept. 4, 14, 17, 66, 85

REST Representational State Transfer. 64, 75, 89, 90, 99

SaaS Software as a Service. 89

SBKD Soft Biometric Keystroke Dynamics. xvii, 1–7, 10, 13, 14, 17, 19–21, 23, 25, 26, 28, 29, 31, 33, 38, 44, 50–52, 54, 55, 57, 58, 60, 62, 63, 65, 71, 73, 78, 81, 89, 95, 96, 98, 100, 101, 103, 133

SCP Secure Copy. 125

SDK Software Development Kit. iii, v, 1–6, 17–23, 25, 33, 36, 37, 47–52, 61–65, 73, 75, 76, 78, 87, 89–97, 99, 101–103

SMS Short Message Service. 49

SQL Structured Query Language. 77

UGS Unity Gaming Services. 78

uGUI Unity User Interface package. 64, 69, 84, 85, 97, 98

UI User Interface. 64, 65, 68–71, 74, 79, 81, 84–86, 90, 96–99

UK United Kingdom. 49, 50, 101

UN United Nations. 3

USA United States of America. 51, 101

USS Unity Style Sheet. 70, 79, 80

UTF-8 Unicode Transformation Format – 8-bit. 28, 66, 75, 80

UUID Universal Unique Identifier. 21, 77

UXML Unity Extensible Markup Language. 70, 84

VM Virtual Machine. 77, 125

VPC Virtual Private Cloud. 90

YAML YAML Aint a Markup Language. 125

Glossary

cyber grooming Cyber grooming is when someone (often an adult) befriends a child online and builds an emotional connection with future intentions of sexual abuse, sexual exploitation or trafficking.[1]. 1, 3, 100

part 3 service A service that is deemed regulated according to the terms of the Online Safety Act. 49

penalty-and-reward model A model that defines an initial confidence value that is increased/decreased as actions are taken. 24, 37

physiological biometrics Refers to physical features in biometrics such as fingerprints, hand shape, iris, retina and face[2]. 24

The European Patients' Academy on Therapeutic Innovation (EUPATI) Organization which focuses on educating patients and patient representives on reserach and development processes within pharmacutical and therapautic innovations. Website: `https://eupati.eu/`.
All use of EUPATi's work is licensed under CC BY-NC-SA 4.0 (`https://creativecommons.org/licenses/by-nc-sa/4.0/`). 56

Unity prefab Prefabs are a special type of component that allows fully configured GameObjects to be saved in the Project for reuse[3]. 65, 72

[1] `https://www.csa.gov.gh/cyber_grooming.php` - fetched 20.04

[2] `https://www.thalesgroup.com/en/markets/digital-identity-and-security/government/inspired/biometrics` - fetched 03.05

[3] `https://learn.unity.com/tutorial/prefabs-e` - fetched 15.03

1 Introduction

1.1 Background

Today, the realm of video games boasts unprecedented popularity[1], yet this surge brings with it a number of challenges. Among these challenges is the occurrence of predatory behavior of minors within gaming communities. Our thesis aims to address this concern, specifically in terms of implementing strategies for verifying user ages and genders.

This project is made in collaboration with our client, AIBA, a spin-off company from NTNU research focusing on combating cyber grooming and other unwanted behavior in game chats. The company's main goal is to prevent bullying, harassment, and cyber grooming in online communities. For this purpose, they are assessing the efficacy of using Soft Biometric Keystroke Dynamics (SBKD) on their Amanda platform for profiling age and gender of users, as this is perceived to facilitate more accurate age and gender predictions.

The client currently use their Amanda child protection platform for moderation and analysis of chat messages, but lack a way of extracting keystroke dynamics data from Unity based games as no existing solution seem to address this need. In response to this, we have undertaken the development of a Software Development Kit (SDK) tailored for Unity, which is proficient at capturing user keystrokes and deriving the required metadata. AIBA can integrate this SDK with its existing moderation tools to identify conversations with a presumed age disparity, provisioning moderators with an indication of conversations that could exhibit signs of predatory behavior.

1.2 Problem description

AIBA considers the capability of extracting Soft Biometric Keystroke Dynamics data essential, as their AI model is intended to be used for predicting a users age and gender based on this information. This in turn could prove critical to identify users sending messages with a predatory intent. Additionally, upcoming and newly implemented regulations may require significant changes in how Soft Biometric Keystroke Dynamics is captured and used with AI.

While the initial assignment outlined these issues, client meetings revealed a desire for a market scan to better understand how large corporations currently handle age verification. In addition, a request for research into the most efficient methods of transmitting SBKD data to minimize network traffic.

1.3 Goals and frames

In this section, we have outlined a set of result goals for the project, alongside effect goals that we expect to stem from these objectives. Furthermore, we have

[1] `https://www.statista.com/topics/1680/gaming/#topicOverview` - Fetched 11.05.2024

defined the scope and constraints.

1.3.1 Frames

Discussions with the client have yielded a set of criteria that the project is required to adhere to:

- Must work with Unity
- Must work on Windows and preferably macOS
- The chat program should be able to connect directly to AIBA's servers
- The SDK will exclusively focus on collecting SBKD data and extracting features by processing them on the client-side
- Will limit the regulatory review to mainly include more recent western regulations from the EU, UK and US

1.3.2 Result Goals

Our objective is to deliver a fully developed SDK which is easily modifiable by the company to facilitate integration with their software. After discussions with the client, we have also created additional goals beyond the minimum requirements set forth in the original assignment description. We aim to complete a market scan of current age verification methods and a review of existing and upcoming regulations. Furthermore, We plan to conduct a study to gauge which methods proves most efficient for transmitting SBKD to an API, aiming to minimize network costs.
The overall result goals are:

- **Software Development Kit:** Develop an SDK capable of extracting SBKD data.
- **Chat Application:** Create a chat application utilizing Vivox and the SDK as an example to facilitate further testing for the client.
- **Legal Compliance:** Determine the SDK's adherence to various regulations in relation to AIBA's final solution, including the EU AI Act, DSA, KOSA, and the OSA, ensuring legal compliance.
- **SBKD transfer:** Develop an HTTP handler that can transmit Soft Biometric Keystroke Dynamics data to AIBA's servers, enhancing user age profiling accuracy.
- **Market Study:** Conduct a comprehensive market analysis to identify existing solutions and determine the SDK's unique value proposition.
- **Efficient Data Transfer:** Examine methods for efficient transmitting of keystroke data to the API to minimize network costs associated with large data packets.
- **Documentation:** Produce detailed documentation for integrating the program into existing Unity projects, complete with code samples and essential information to facilitate implementation by developers.

1.3.3 Effect Goals

By finalizing our project, we anticipate that the solution will notably enhance online video game moderation where it is employed, enabling chat moderators to better concentrate their moderation effort. This could result in a significant decrease in grooming and other unwanted behavior within these video game chats.

- **Grooming Detection:** Enhancing the capabilities of chat moderators in detecting cyber grooming by giving them warning when users might be involved in cyber-grooming.
- **Grooming Reduction:** Reducing the prevalence of children falling victim to cyber-grooming by giving chat moderators the ability to step in before a potential escalation has occurred.
- **Moderation Cost Decrease:** Lowering the costs associated with chat moderation in Unity-based video games by flagging the game chats that should be more closely looked at.
- **SBKD Contribution:** Contributing to the field of Soft Biometric Keystroke Dynamics (SBKD) by enabling its capture in Unity game chats as well as optimizing the data transfer to minimize overhead as much as possible.

1.4 Societal contribution

In this section we describe how optimization of data transmission packet sizes and how contributing in the field of keeping children safe online facilitate for progressing towards United Nations Sustainable Development Goals.

Through the EU Digital Services Act (DSA) introduced in late 2022, the EU wanted to create a safer digital space where the rights of all users of digital services are protected (Section 6.1). With an increasing trend of children using online gaming services [1] and social media[1], the need to implement measures to protect children is becoming more and more vital. Doing so would also contribute to UN Sustainability Goal 16, which specifies Peace, Justice and Strong Institutions, and more specifically sub-goal 16.2:

> *End abuse, exploitation, trafficking and all forms of violence against and torture of children*[2]

The SDK we are developing in this thesis contributes to this sub-goal, in reducing abuse and exploitation of children on games using the Unity gaming platform. More accurately, we provide an improvement in manual moderation of content which in turn will keep more children safe from harassment and grooming online.

Additionally, the SDK aids in the UN Sustainability goal number 12 regarding responsible consumption and production[3]. Through the data transfer research we

[2] `https://sdgs.un.org/goals/goal16#targets_and_indicators` - Fetched 13.05

[3] `https://sdgs.un.org/goals/goal12` - Fetched 13.05

have conducted and its implementation in the SDK, we have given AIBA the necessary information to make an educated decision (Section 4.2) and thus minimize the amount of computational resources required to utilize the SDK to its full extent.

1.5 Scope and limitations

There are certain factors that should be taken into considerations when reading the thesis. Notably, the client already possesses an established algorithm for profiling users' age and gender, which this project will not replicate or modify. Our SDK's primary function is capture and transfer SBKD data to the client's servers, allowing their algorithm to process and profile the input.

The time constraints, alongside the necessity to conduct a market study and regulatory research, impose limitations on our ability to develop a comprehensive SDK. Given the extensive nature typically associated with an SDK, encompassing a wide array of content and features, our project will aim to deliver a foundational SDK with a limited set of features. Similarly, while we will identify obligations that result from the regulations that we will review in Chapter 6, we will not prioritize implementation of these regulations. This is due to the substantial time investment that would be required to meet these obligations.

A critical decision within the project's scope involved determining the approach towards chat SDK integration — whether to develop a new complete Chat SDK, enhance an existing one, or engineer a solution compatible with multiple chat SDKs available for Unity, if possible. With over a dozen chat services available within the Unity ecosystem, tailoring our SDK to each service individually exceeds our project's limitations due to time constraints. Since the main goal of the project is to investigate the viability of extracting SBKD data and produce a Proof of concept SDK the choice was made to develop a chat application utilizing Unity's own chat service, Vivox. This approach enables the client to leverage an existing application as a reference point for further research into their specific needs.

1.6 Legal and Ethical Considerations

While our implementation of the product is primarily a prototype to demonstrate a proof of concept, it is important to address the ethical considerations associated with a final product based on our work.

One significant ethical consideration is the profiling of children. Although AIBA does not intend to profile children, profiling all users' ages may inadvertently include minors. The potential for inadvertently profiling these minors based on their biometrics raises the ethical question of whether the ends justify the means. To mitigate this, it may be necessary for the client to limit data collection to the absolute minimum, with a heightened focus on safeguarding the privacy of minors. One potential solution could be to stop Soft Biometric Keystroke Dynamics based profiling of a certain user when there is a high likelihood that the user is a minor.

Fortunately, there are numerous regulations that impose obligations on companies to enforce more ethical measures. For instance, GDPR mandates informing the user, requiring consent for storing data, and minimizing data storage to the absolute necessary. While regulations such as the GDPR and COPPA have been in effect for several years, new regulations have recently been implemented or are on the horizon.

In light of this, we have chosen to examine four key regulations: the EU Digital Services Act (DSA), the US Kids Online Safety Act (KOSA), the EU Artificial Intelligence Act (EU AI Act), and the Online Safety Act 2023 (OSA) (Chapter 6). The objective of this analysis is to determine the actions that AIBA must take to comply with these regulations and to assess the extent to which these regulations enhance the ethical integrity of the final product.

It is our responsibility to ensure that the client is informed about existing and forthcoming regulations. Our aim is to guide them on what to focus on in their further work on our project and how to make their final product as ethically responsible as is feasible.

1.7 Structure of the Thesis

Chapter 2 - Development Process: Outlines the methodologies and frameworks used throughout the project, with a particular focus on the Scrum methodology. It highlights how Scrum enabled a flexible and iterative development process, facilitating dynamic adjustments based on evolving project requirements and client feedback.

Chapter 3 - Software Requirements: Provides a concrete description of both the functional and non-functional requirements of the SDK. Moreover, this section encompasses the user stories developed in the sprint planning meetings.

Chapter 4 - Soft Biometric Keystroke Dynamics (SBKD): Gives an overview of SBKD, defining its key concepts and common applications. It examines various experiments on keystroke dynamics, emphasizing their use in authentication and profiling. The chapter also highlights our findings on efficient methods for transmitting SBKD data, aiming to reduce network costs.

Chapter 5 - State of the Art: This chapter provides a state of the art market scan regarding existing age verification solutions in social media and games. It also includes findings with regards to challenges associated with online game moderation, particularly in text chats. It examines the prevalence of toxic behavior in game chats and explores existing solutions to help moderators manage these issues, highlighting gaps in current approaches.

Chapter 6 - Legal Frameworks: Overview of the relevant legal frameworks that impact the development of AIBA's solution, particularly focusing on new EU regulations concerning age verification for children in games and social media. It discusses the EU Digital Services Act (DSA), US Kids Online Safety Act (KOSA), Online Safety Act 2023 (OSA), and the upcoming EU Artificial Intelligence Act (EU AI Act), explaining their relevance and implications for AIBA's compliance

and operational strategies. The chapter prioritizes recent and upcoming legislation to ensure the SDK meets modern regulatory requirements, excluding well-established frameworks like COPPA and GDPR-K due to their widespread existing compliance in the industry.

Chapter 7 - Technical Design and Implementation: Provides an in-depth look at the technical aspects of the project. It starts by breaking down the problem into manageable sub-problems and describing the iterative approach used to solve them. Detailed sections cover the system architecture, the tools, and libraries used, including programming languages, IDEs, APIs, and frameworks. Code examples, UML diagrams, and database models are included to illustrate key points. Additionally, the chapter explains the processes of data collection, culminating in how data was gathered and analyzed as outlined in the subsequent chapter.

Chapter 8 - Deployment: Detailed instructions on how to deploy the solution to the company's systems. It includes technical steps for setting up the solution, along with discussions on scalability, maintenance, and other technical decisions. Despite not having direct access to AIBA's API during development, this chapter outlines how AIBA can configure their API gateway for private integration with the SDK.

Chapter 9 - Quality Assurance: Here we explain the strategies employed to ensure the quality of the SDK. Detailing how the testing procedures were implemented, including manual verification and automated tests using scripts. Further, we highlighting the the SDKs compliance with the performance requirements. Then, we discuss the implementation of industry standards, the challenges faced across different operating systems, and the limitations encountered due to external dependencies like Vivox.

Chapter 10 - Discussion: This chapter presents the key findings, project process, and SDK implementation from our work. It also includes reflections on deviations from the original plan and our adaptive approach to the task. The SDK implementation section covers technical challenges, solutions, and the integration of the SDK with the chat application. Additionally, we discuss the viability of SBKD for age detection. Furthermore, we address legal and ethical considerations to ensure compliance with relevant regulations and discuss the ethical implications of profiling users.

Chapter 11 - Conclusion: The conclusion gives a recapitulation if the objectives, states the results and concludes the thesis.

2 Development Process

This section outlines the overall framework and methodologies employed in the development of our project, focusing particularly on our choice of the Scrum methodology. We discuss how Scrum facilitated a flexible and iterative development process, supporting dynamic adjustments of goals and tasks in response to evolving project requirements and client feedback.

2.1 Process model

A process model is a framework of guidelines and tools designed to streamline organizational processes[1][2]. Its goal is to ensure tasks are completed efficiently and consistently, with a focus on continuous improvement. The framework outlines roles, procedures, and best practices for managing and enhancing processes.

Process frameworks vary, ranging from rigid models like the waterfall method, suitable for well-defined projects, to agile methodologies such as Scrum and extreme programming, which allow for rapid adjustments and fast delivery of early product versions for feedback.

Adopting a process framework enhances work standardization, reduces variation, and improves efficiency and product quality.

2.2 Choice of model

Given the scope of the project and the frequent use of process frameworks in professional settings, as well as the fact that it was encouraged for the bachelor thesis, we recognized the need to implement one for this project. This decision was driven by a desire for better structure, to deliver the best possible product, and to gain valuable experience with a tool that would be essential in our future careers.

2.2.1 Requirements

Given none of us had prior experience with Unity or C#, it introduced significant uncertainty regarding time requirements for various development phases. Some tasks might be more challenging or simpler than initially anticipated. Additionally, the feasibility of completing the project was uncertain, particularly concerning the possibility of capturing Soft Biometric Keystroke Dynamics data within Unity. These uncertainties necessitated an agile framework that would allow for rapid adjustments. Should capturing keystrokes in Unity prove impossible, the project's focus would shift towards research rather than implementation. We aimed for an iterative approach, easily enabling feedback from AIBA based on our progress, which also facilitated starting with a "proof of concept" to quickly ascertain the

[1] `https://www.institutedata.com/us/blog/understand-software-process-models/` - Fected 03.04

[2] `https://www.ibm.com/blog/business-process-modeling/` - Fetched 03.04

feasibility of keystroke capture in Unity.

Furthermore, we wanted a framework popular in the industry, enhancing the applicability in future professional settings. These requirements led us to select a framework that meets the following criteria:

- Agile, with the flexibility to quickly shift focus.
- Iterative, enabling a start with "proof of concept" for early validation.
- Iterative for frequent feedback from AIBA.
- Popular among businesses.

2.2.2 Framework Options

After conducting research, we evaluated the following potential frameworks:

- **Scrum:** Offers short sprints allowing rapid focus shifts and iterative development, with frameworks for showcasing work to clients. While it lacks formalized intra-sprint communication structures, its popularity in the industry is notable.
- **Kanban:** Features continuous flow and flexible task management, supporting ongoing evaluation and adjustments. The Kanban board provides client insight into progress, fostering feedback.
- **Lean:** Prioritizes eliminating resource waste to quickly remove non-value activities. It is non-iterative but emphasizes frequent feedback by continuous communication, making it popular among development frameworks.
- **Crystal Clear:** Minimal rules for quick focus adjustments and emphasizes rapid delivery of a functioning prototype. It encourages frequent deliveries for client feedback but is less widely used in industry.
- **Extreme Programming (XP):** Includes an iteration plan for adaptable focus, with a strong emphasis on customer collaboration. Its popularity has waned as its principles have been integrated into other frameworks like Scrum.

Based on our analysis, we created the matrix in figure 2.1 to visualize how well each framework matched our project requirements.

Criteria	Scrum	Kanban	Lean	Crystal Clear	Extreme Programming
Agile and adaptable	5	4	4	4	5
Iterative for PoC	5	4	3	4	5
Iterative for feedback	4	3	2	4	5
Popularity	5	4	4	3	4
Average score	4.75	3.75	3.25	3.75	4.25

Figure 2.1: Comparison of Agile Methodologies as perceived by group members

2.2.3 Choice of framework

We ultimately chose Scrum as our framework, as it best matched our needs according to the criteria matrix. Sprints offer opportunities to update the client on our progress during sprint reviews, allowing for timely feedback on their thoughts. While Scrum traditionally does not advocate contacting the client outside of sprint reviews, together with AIBA we determined that our project had limited need for such interaction between bi-weekly client meetings or sprint reviews. Scrum also facilitates adjustments to the backlog based on the outcomes of the current sprint, enabling us to quickly pivot if capturing keystrokes in Unity proved infeasible.

Additionally, AIBA recommended the use of Jira, a popular tool for managing Scrum projects in the professional sphere. After some research we concluded that expertise in both Jira and Scrum could prove beneficial for our future professional opportunities due to their extensive use in the industry.

2.3 Implementation of framework

While Scrum provides structured rules and practices for teamwork and project management, it's designed with flexibility in mind to accommodate the unique needs of different teams.

2.3.1 Rituals and Meetings

Scrum incorporates various "rituals" designed to facilitate optimal communication among developers, the Scrum Master, and the Product Owner.

2.3.1.1 Sprint planning

During the sprint planning process, the group collaboratively engaged in shaping the upcoming sprint by examining and updating the backlog. This initial phase involved identifying any missing items and assigning story points to those that had not yet been estimated.

Following the backlog refinement, we proceeded to define the sprint's goals and its duration. We then selected stories that aligned with these goals, prioritizing tasks that would best steer us towards achieving these goals.

To maintain a sustainable workload and ensure the feasibility of our sprint commitments, we aimed at around 50 story points for each week of the sprint to make the goals for each sprint achievable.

2.3.1.2 Daily Stand Up

At the beginning of each day, we conducted a 15-minute daily stand-up meeting. Initially, the implementation of these meetings was somewhat inconsistent, but by the second sprint, we had established a more stable routine, with fixed procedures in place by the third sprint. Meetings were scheduled every day at 10:30 AM unless all team members agreed and there was a specific reason to cancel.

During the daily stand-up, each member addressed three key points: priorities, progress, and issues. This structure allowed team members to update each

other on their current work focus and achievements since the last meeting. It also enabled the Scrum Master to identify any issues early in the process, facilitating quick resolution. Responses to these three questions were recorded by the Scrum Master in Confluence and shared with the group. An example of these notes can be found in Appendix B. This practice provided team members with a resource to revisit the day's objectives and assess which group member could help with which issues.

2.3.1.3 Supervisor meetings

Every Thursday at 1:00 PM, we held a meeting with our supervisor. An agenda was distributed before each meeting in order to allow all participants to prepare. The Scrum Master took minutes during these meetings, enabling team members to review discussions and decisions made on various topics. These meetings provided an opportunity to clarify uncertainties and receive feedback on our completed work and on our prioritization strategies relative to our project's progress.

2.3.1.4 Client meetings

The biweekly meetings with AIBA were important for receiving feedback on our focus areas and AIBA's vision for the product. This also provided an opportunity to showcase our progress, allowing for ongoing input throughout the product development process. Additionally, these meetings enhanced our understanding of Soft Biometric Keystroke Dynamics, benefiting from the client's expertise in this area. The meeting minutes from these meetings can be found in Appendix D.

2.3.1.5 Sprint Reviews

At the end of each sprint, we held a sprint review meeting to present our work to the client and receive feedback. This is a crucial component of the Scrum methodology, as it maximizes the likelihood of delivering a product that meets AIBA's expectations. Primarily, these meetings involved Gard Støe, although Patrick Bours was also typically present. The transparency fostered by these meetings ensured that AIBA, was well-informed of our progress and had realistic expectations for the project's outcomes.

2.3.1.6 Sprint retrospective

At the end of each sprint, we conducted a sprint retrospective to discuss our perceptions of the sprint's progression. Each team member was tasked with contributing items to the retrospective board, which was divided into three sections: "Start Doing," "Stop Doing," and "Keep Doing." This categorization was designed to encourage the team to identify successful practices to continue, highlight ineffective actions to cease and recognize ongoing effective strategies worth maintaining. The underlying theory is that subsequent sprints would improve upon earlier ones through increased feedback and the elimination of less productive practices.

At the end of the retrospective, the team developed one or two actionable items to be addressed in the next sprint. These actions provided specific goals to focus on, ensuring continual improvement and progress throughout the pro-

ject's life-cycle. One example of the sprint retrospective board can be found in Appendix E

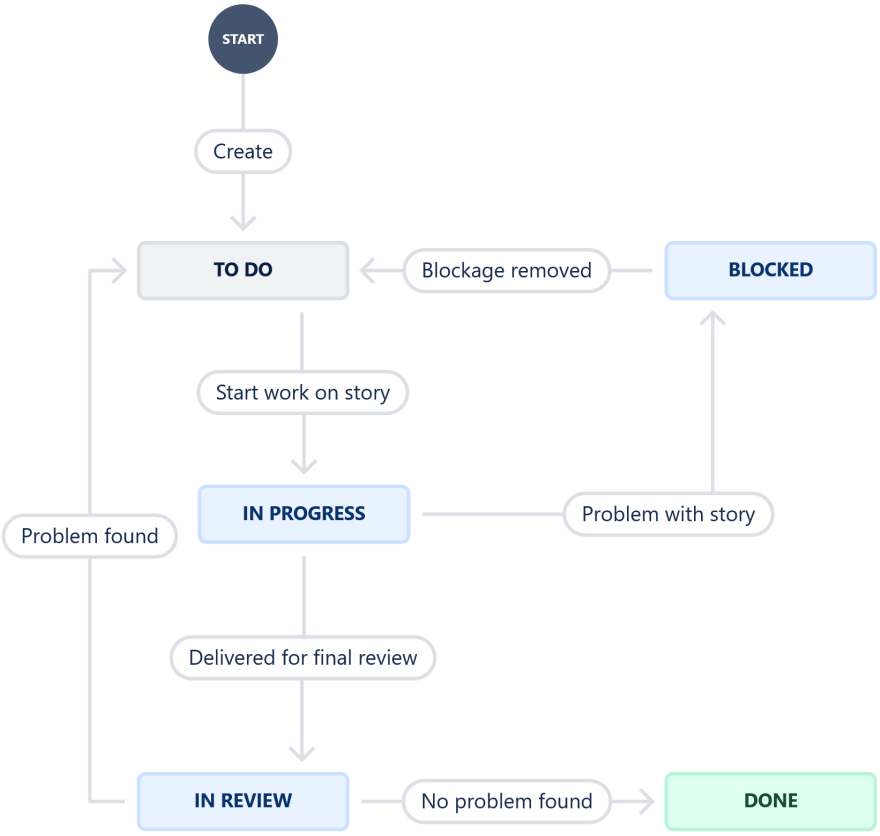


Figure 2.2: Jira workflow

2.3.2 Scrum Board

To manage issues effectively, we utilized Scrum boards, comprising five different stages:

- **To-Do:** Contains tasks that are pending commencement. Developers select stories to work on from this board.
- **Doing:** Indicates that a developer has started working on a task. Tasks remain here until they are completed and ready for review or encounter a blockage requiring intervention.
- **In Review:** Once tasks are completed, they move to this board for quality assurance. All other team members will then review this story and provide feedback.
- **Done:** Represents tasks that have been completed and approved by the team. Placement in this board signifies that a story has met all criteria and objectives, marking its conclusion.

- **Blocked:** Used for tasks that cannot progress due to external dependencies or obstacles. Tasks remain in this status until the issue is resolved, at which point they either return to the To-Do board for reevaluation or proceed as determined by the resolution.

See Figure 2.3 for an example of how the board could look, and figure 2.2 for a workflow diagram on how the boards were to be used.

This decision for collective review aims to provide all members with insight into the work accomplished and its methodology, while also leveraging multiple perspectives for identifying potential errors.

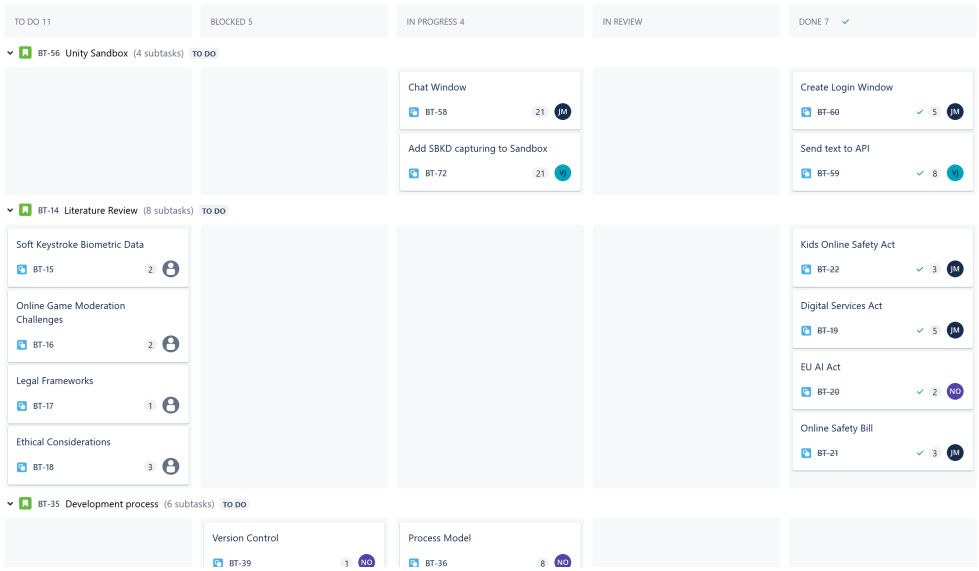


Figure 2.3: Jira Boards

2.3.3 Story points

To estimate the complexity of user stories, we utilized story points. Story points indicate the relative scale of effort required for each "story" based on its complexity rather than the time it would take to complete. This approach takes into account that developers may spend varying amounts of time on the same task and aims to mitigate discrepancies in time estimation that have led to disagreements in the past. By using common scrum practises and focusing on complexity, we found it easier to reach a consensus.

To establish a baseline for these estimates, we identified the simplest task as a reference point and assigned it a value of 1. Then, we determined the most challenging task at hand and assigned it a value of 55. The sizes for all other tasks

were estimated based on where they fell within this spectrum of complexity.

For the numerical values, we adopted the Fibonacci sequence: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89. This choice reflects the common Scrum practice rooted in the difficulty humans have in estimating with more than 60% accuracy; the Fibonacci numbers increase by just under 60% from one to the next.

The use of story points provided developers with a clearer understanding of the workload they were committing to, based on complexity, and offered an effective overview of the remaining work during each sprint.

2.3.4 Stories

Each user story encapsulates a unique narrative, typically formulated as: "As a [role], I need [requirement], so that [benefit]."

An example of this could be: "As an AIBA developer, I need the SBKD data from chats to be sent to our API, so that we can process data for further use."

This narrative framework is designed to provide developers with a clearer understanding of the problem that the story aims to solve. By framing the task in terms of who requires what and why, it becomes easier for the development team to grasp the purpose behind each task and approach their work with a solution-oriented mindset.

2.4 Sprint Overview

At the outset of our project, the sprints commenced with a lengthy duration. This was primarily due to the fact that other courses were taking up time, leaving us with a more limited time to work on the thesis. Once the other courses were finished, full focus was directed towards the thesis, and the sprint duration shortened as more work could be accomplished in less time. We went from four-week sprints in the beginning to one-week sprints by the end. The reasoning for these short sprints at the end was for frequent evaluation of the team's position in relation to the delivery date, and adjust our strategy to accommodate the the work that had been accomplished thus far.

2.4.1 Sprint summaries

Under is a compilation of what was the main focus areas, the length, and what was the goals for each sprint during the thesis.

2.4.1.1 Sprint 1 (07.02 - 06.03)

The focus of this sprint was on setting up the report structure, writing the first chapter, and start working on the literature research related to it. Significant time was devoted to understanding and writing about regulations such as the EU AI Act and KOSA. Additional research on SBKD was conducted to enhance our understanding of its functionality and general application. Efforts were also made to make a console application for capturing keystrokes in C# as a preliminary step towards integration with Unity, and an API was developed for receiving captured

data.

Due to a misunderstanding of what is considered a best practice in Scrum when sprint goals are not met, this sprint duration was extended, which should not be done. It was later recognized that a better Scrum practice would have been to address unmet goals during the retrospective, and discussing strategies for improvement in the next sprint.

2.4.1.2 Sprint 2 (07.03 - 26.03)

Following the C# Proof of concept (POC) application, our focus shifted to creating a similar POC within a minimalist chat service in Unity. This effort was to verify the feasibility of implementation in Unity and assess sending data from the Unity environment to a REST API. It would also be useful for a showcase for presenting the program and as a "fallback" solution if it would not be viable to capture keystrokes with Vivox or other chat services. Concurrently, we commenced work on the "Development Process" chapter of the report.

The progress this sprint was impeded due to the group's decision to allocate additional focus on the ING2300 course, which had a project submission and examination during this period.

2.4.1.3 Sprint 3 (27.03-05.04)

As we reached further into the semester, the group decided to adopt shorter sprints to accelerate iterations, now fully directing all attention to the project. The primary focus was to complete as much as possible of the first four chapters of the report, aiming to catch up on our lack of report progress.

2.4.1.4 Sprint 4 (05.04-12.04)

In sprint 4 we stared work on making a Vivox application, and our data transfer research. We also had a large focus on report writing this sprint.

2.4.1.5 Sprint 5 (12.04-19.04)

After facing bigger challenges on getting a Vivox project going, this work was continued this sprint. We also continued work on the data transfer, and starting work on the implementation chapter.

2.4.1.6 Sprint 6 (19.04-30.04)

During sprint 6 we had the goal of completing the most crucial work that was in progress. The Implementing SBKD into the Vivox application, finishing the data transfer research, and completing work on the API.

2.4.1.7 Sprint 7 (30.04-07.05)

This sprints focus went into report writing with finishing chapter 1-4 in the report, most of the implementation chapter, and reworking the report structure based on supervisor feedback. We also had a goal of completing all tests on the Vivox application during this sprint.

2.4.1.8 Sprint 8 (07.05-21.06)

The last sprint we had full focus on report writing with a goal of completing the rest of the report. In addition to that as a way to vary the work, we implemented finishing touches to the code and provided missing documentation.

3 Software Requirements

This chapter covers the essential aspects of requirements engineering for our project. We start with Requirements Elicitation, detailing how stakeholder needs were gathered. Functional Requirements follow, featuring a Use Case Diagram and both High-Level and Detailed Use Cases.

Next, we explore Non-Functional Requirements, discussing Performance, Scalability, Security, Usability, Maintainability, and Compatibility. We then address Requirement Management, focusing on tracking and controlling requirement changes. Finally, we discuss Requirement Validation, ensuring the requirements align with stakeholder needs and are feasible to implement.

As the projects initial main goal was to investigate the feasibility of extracting keystroke information from a Unity application, the requirements are less comprehensive than if it would be a production ready application. Through conversations with AIBA, they wanted our focus to be on making a Proof of concept (POC) that can extract keystroke data and outside of that goal we were free to make our own requirements.

3.1 Requirements Elicitation

Requirements were elicited from a number of sources, such as priorities of the client, software with similar areas of applicability that we found online and through a market scan (Section 5.1). The undergone research ranging from legal frameworks to SBKD also helped define the requirements (Chapter 4, Chapter 6).

In general, these requirements were defined through the thesis description (Chapter 11) and biweekly meetings. The requirements we have set for the task internally within the group also had an impact on the SDK.

The methods we used to gather requirements varied depending on where they came from. For instance, the requirements we received from AIBA were mainly defined through the thesis description and through meetings during the semester. One such example of a requirement not present in the task description, was the study of legal frameworks. This task, which was an additional assignment that went beyond the task description, was both prioritized due to a wish from AIBA and has enabled us to reflect on what would be required to deploy the SDK for use by real providers of Unity games. The task description also defines a market scan on age verification methods employed by various actors, which can be found in Section 5.1. Lastly, discussion among the group members evoked the self-defined goals we wanted to achieve through the thesis.

3.2 Functional Requirements

The functional requirements detail the core functionality that the SDK provides.

3.2.1 Use Case Diagram

For depicting the system interaction in the project, a general use case diagram was created.

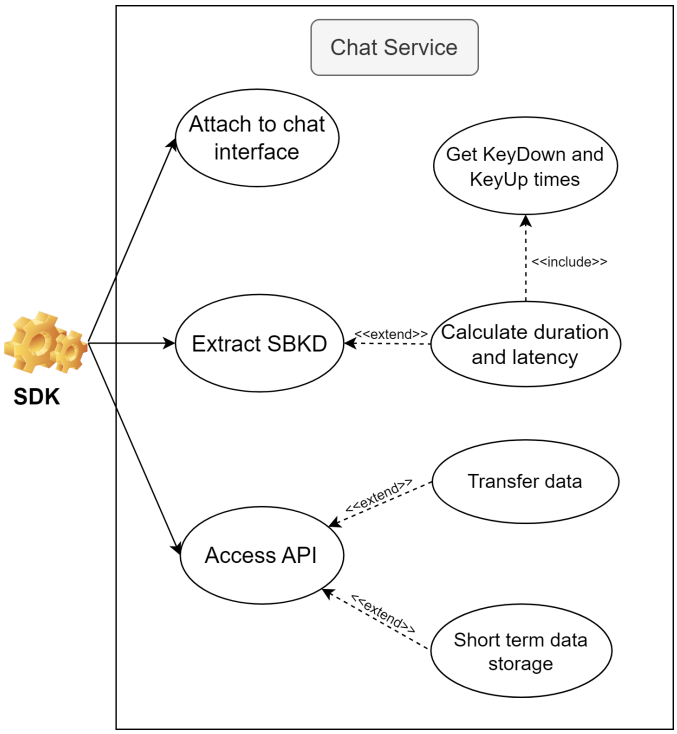


Figure 3.1: Use Case Diagram

3.2.2 High-Level Use Cases

The use cases that are mentioned in this section were created to get a general overview of the functionality we would have to incorporate to comply with the functionality requirements and to provide a satisfactory final product. It is also something the group members can look back at during development to ensure compliance with functionality requirements.

The high level use cases created for the purpose of this project abstract away user interaction and moderator responsibilities commonly found in online game chats. In Table 3.1, 3.2, 3.3, 3.4, and 3.5 are the primary use cases focusing on SDK functionality.

3.2.3 Detailed Use Case

In Table 3.6 we have extended one of the main use cases of the application after a detailed use case model template[1]. This was done to provide more details on

[1] `https://www.rose-hulman.edu/class/csse/csse497/Process/detailed_use_case_model_template.html` - Fetched 01.04

Use case: Attach to a chat interface **Actor(s):** SDK **Goal:** Provide a well-documented, user-friendly way of integrating the SDK in different chat interfaces **Description:** Supporting integration with different chat services that are supported by Unity

Table 3.1: Use case 1: Attaching to a chat interface

Use case: Extract Soft Biometric Keystroke Dynamics data **Actor(s):** SDK **Goal:** Capture Soft Biometric Keystroke Dynamics in chat session **Description:** The SDK attaches onto the active chat session and captures the SBKD data.

Table 3.2: Use case 2: Capture SBKD data

Use case: Transfer of SBKD data to API **Actor(s):** SDK **Goal:** Transmission of data to API included in SDK. **Description:** Ensure proper transmission of the collected SBKD data to the API using HTTP.

Table 3.3: Use case 3: Transfer of SBKD data to API

Use case: Calculate SBKD characteristics **Actor(s):** SDK **Goal:** Calculate duration and latency from the derived KeyUp and KeyDown timing information sampled **Description:** Client-side calculation of useful characteristics for profiling of individuals

Table 3.4: Use case 4: Calculate SBKD metrics

Use case: Storage of SBKD data **Actor(s):** SDK **Goal:** Storing user SBKD data to increase sample size on actors and to provide data in a simple way **Description:** API with a NoSQL database for storing data on users of chat service and possibility of data lookup on said user

Table 3.5: Use case 5: Storage of SBKD data

this specific use case, the arguably most important one, and to picture how an extended use case could be structured if providing more details is desirable.

Use case: Capture SBKD data
Description: The SDK attaches onto the active chat session and captures the keys pressed by the user and timing information of the key press
Actor(s) involved: User of service, chat service, SDK
Basic flow of events:

1. User starts typing a message to another user
2. Whilst user is typing, the SDK captures SBKD data.
3. User sends message
4. SDK arranges data in the correct order and format

Alternate flow of events:

1. User starts typing a message to another user
2. Whilst user is typing, the SDK captures SBKD data.
3. User clicks out of chat window
4. SDK temporarily saves data for session and temporarily stops capture
5. User focuses chat windows once more
6. Capture is restarted
7. User sends message
8. SDK arranges data in the correct order and format, ensuring inclusion of all sessions

Pre-conditions: The SDK is attached to the chat interface, and the chat window is focused or active
Post-conditions: Captured data is formatted correctly, and may at any time be sent to API
Special requirements: These are detailed in section 3.3

Table 3.6: Detailed use case: Capture SBKD data

3.3 Non-Functional Requirements

Non-functional requirements detail how a system as a whole should operate and can be described as the emergent properties of the system or quality attributes of a system[2].

In this section we will describe the non-functional requirements that we have set for the system based on our meetings with AIBA and discussions within the team.

[2] `http://users.csc.calpoly.edu/~jdalbey/SWE/QA/nonfunctional.html` - Fetched 02.04

3.3.1 Performance

- The system should not add a significant perceivable delay for the logging and processing of the keystroke data. Significantly perceivable is defined as 78 ms. Threshold perceivable click to visual response is 78 ±12 ms according to [2].
- The Soft Biometric Keystroke Dynamics capturing should not have a discrepancy of more than 0.1 ms between a key press occurring and it being registered, as AIBA indicated a preference for at least 0.1 ms accuracy
- The system should handle capturing keystroke data above the upper end of normal writing speed without problems. Upper end of normal writing speed for programmers can be estimated as 90 words per minute[3] which roughly equates to 441 characters per minute if there are 4.9 characters per word [4].
- The size of transferred data should be minimized. Tests should be conducted to determine the size of data using the most common file and data types.

3.3.2 Scalability

- The system should be designed in such a way that adding or removing keystroke metrics would require little extra work.
- If the service has a considerable amount of users, the API component of the SDK needs to undergo vertical scaling to handle the increased demand.

3.3.3 Security

- The entry-id in the API should be provided through Googles UUID to ensure all ids are unique [5].
- Each transaction sent to the API should utilize the HTTPS protocol to ensure confidentiality and integrity of the data.
- Credentials should not be stored or transmitted in plain text to ensure confidentiality. Instead, environment variables should be defined and used

3.3.4 Usability

- The system should not add a significant perceivable delay for the logging and processing of the keystroke data. Significantly perceivable is defined as two times 78 ms. Threshold perceivable click to visual response is 78 ±12 ms according to [2].
- The user interface should be modeled on the behavior of common chat services, e.g. Discord, Messenger and so on.

3.3.5 Maintainability

- The code of the project should be well documented. Each line of code in the SDK should have an explanation unless the variable and function names

[3] `https://onlinetyping.org/blog/average-typing-speed.php` - fetched 23.02
[4] `https://norvig.com/mayzner.html` - fetched 23.02
[5] `https://pkg.go.dev/github.com/google/uuid` - Fetched 02.04

describes it sufficiently.
- The project should follow established coding standards and best practices as specified in the project plan.
- Efforts should be made to write modular code with clear separation of concerns. This will ease the workload if any modifications or updates needs to be done.

3.3.6 Compatibility

- Tests should be conducted to determine the best file formats and data types for us to use which minimizes payload size.
- The transferred data should be uniformly structured throughout life-cycle of the SDK.
- Little effort should be required to route the data from the SDK to the client's API.

3.4 Requirement Management

Having been given a lot of freedom in the development process, requirements in the early phases were under constant refinement. One such change was the addition of a research requirement on legal frameworks that was proposed in the meeting held the 7th of February. Another example of one such change was after scrum review 3 where a focus shift happened due to amount of time remaining. Here, AIBA suggested prioritizing ensuring integrity over confidentiality on transmitted data.

For management of requirements we utilized stories and boards in Jira. More information regarding requirement management can be found in section 2.3.1 and 2.3.2.

3.5 Requirement Validation

Requirements were mainly validated through scrum reviews with the client present. These were held after most the sprints had concluded. The different prototypes (Section 7.2), starting with the console application, were showcased to make sure we were on the right track. The feedback received in these meetings helped shape the requirements for upcoming sprints and the final SDK, and gave us concrete goals to work towards.

Quality assurance of the SDK was completed, where the application was tested on different OSs, making sure that the attributes were being calculated correctly and documenting the SDK Chapter 9.

A final review on whether the requirements were met or not was held the 24th of April, where the client told us that they were satisfied with what we had created. They also seemed to appreciate the chat application we had developed, as they saw it as an opportunity to use for testing themselves.

4 Soft Biometric Keystroke Dynamics (SBKD)

This chapter aims to provide an overview of Soft Biometric Keystroke Dynamics (SBKD) and how it works. It will provide a definition, it's most common applications, and results with these uses. Additionally, we will present our research findings on efficient methods for transmitting keystroke dynamics to the API, aiming to minimize network costs associated with large data packets.

4.1 Soft Biometric Keystroke Dynamics (SBKD) data

To understand the results and to define the data that is captured in the SDK, the research conducted by the group on Soft Biometric Keystroke Dynamics has been summarized in this section.

4.1.1 Definitions

Soft biometrics may be defined as:

> *Characteristics that provide some information about an individual but lacks the distinctiveness to sufficiently differentiate any two individuals* [3]

The SDK proposed in this thesis focuses on capturing keystroke dynamics, a type of behavioural biometric, in a Unity environment. In this context, keystroke dynamics are habitual patterns or rhythms an individual exhibits while typing on a keyboard [4]. There are currently no other solution that captures keystroke dynamics for biometric purposes in Unity (Section 5.1.3).

4.1.1.1 Different keystroke dynamics experiments

Most research publications conducting keystroke dynamics experiments, may use one or more of these different kinds of observations [5]:

- Free-text where a user has no restrictions when typing a paragraph [4]
- Fixed-text where the text to type is predetermined and stays constant throughout the experiment [6]
- Semi fixed-text, a middle ground between free and fixed text. An example being commands in a command line

We have found keystroke dynamics to have most commonly been studied in authentication systems. Two common approaches in authentication, are the fixed-test approach that analyzes a single input string where the goal is to grant a genuine person access to a system, or a continuous approach that aims to lock an impostor out from a system [6].

Another utilization of keystroke dynamics are in profiling of soft biometric traits, which is of most interest in this thesis.

4.1.2 SBKD Performance

4.1.2.1 Proven results

The performance of keystroke dynamics as a metric for authentication or profiling are lower than that of physiological biometrics modalities based authentication systems [7].

In a study on continuous authentication by Bours and Barghouthi, a suggested method involved compiling a template on keystroke dynamics after longer periods of use, and continuously compute a confidence level through a penalty-and-reward model that determines if a user change has happened or not [8]. In these research articles they commonly utilize False Acceptance Rate (FAR) and False Rejection Rate (FRR) as measurable values on whether the user is who they say they are or not. Another article on the same topic achieves both low FAR(1.89%) and FRR(1.45%), which is ideal [9].

In terms of soft biometrics, one study has been successful in identifying traits such as gender (70%-86%), age (67%-78%) and handedness (76%-88%) using a fixed-text analysis achieving a recognition rate between 67%-88% across these categories [7]. The article is based on another previous study, which achieved up to 90% recognition rate across these same categories with a database of 110 users [10].

4.1.2.2 Challenges

Even though the results that these methods yield are promising, the application of keystroke dynamics for either profiling or authentication in the real world has been limited [5].

Keystroke dynamics does run into challenges when analyzing user behaviour. A persons typing can be affected by a number of factors, and may even change throughout the day. A persons fatigue, mood, posture or whether the person is sitting or standing up may affect results of such soft biometrics [11]. Additionally, if a user discerns that their typing metadata is tracked, they may change their typing pattern and rhythm to dupe the profiling algorithm.

4.1.3 Applications

In 2014, an article was published detailing the use of keystroke dynamics as a verification method in Massive Open Online Courses, where Coursera[1] was the targeted platform [12]. After a task had been completed, the course attendee could choose between verification through photo or through typing a short phrase. More applications have been found through a market scan we have completed that looks into common state of the art age verification software utilized on different platforms (Section 5.1)

[1] `https://about.coursera.org/how-coursera-works/` - Fetched 06.03

4.1.4 Common features

Common Soft Biometric Keystroke Dynamics features applied are figuratively described in Figure 4.1, where the captured ones are listed below and visualized in Figure 7.4.

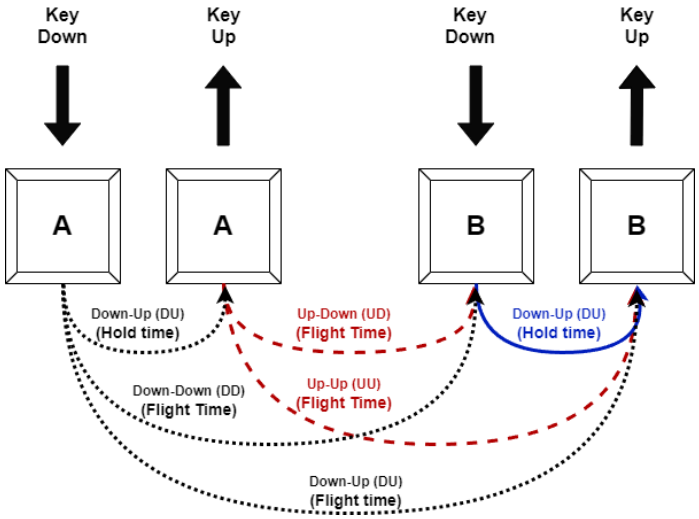


Figure 4.1: SBKD metrics (inspired by figure in [13])

- Key down time - the time of pressing down a key
- Key up time - the time of releasing a key that was previously pressed
- Down-up time - the time between pressing down a key and releasing the same key (also known as duration, dwell time or hold time)
- Up-down time - the time of releasing one key to pressing down another (also known as latency or flight time)

Between the research publications studied, these definitions overlap in all of them, but they may be termed differently.

4.2 Efficient SBKD Transmission Research

As part of our research into Soft Biometric Keystroke Dynamics, we have conducted a study into how to efficiently transmit Soft Biometric Keystroke Dynamics data with minimal amount of bandwidth usage. The client indicated at an early stage that there was a desire to ascertain the most efficient method of transmitting SBKD, with the objective of minimising the data packet size. It was recognised that sending a message with Soft Biometric Keystroke Dynamics data would be costly for those who had implemented the SDK, given that the data would occupy a significant increase in space compared to just sending the plain message. Professor Bours from AIBA had found that it would take eight times the bandwidth than if it were to only transmit the message text. We did not find research that explored the size difference between a file containing only the message text versus the mes-

sage text with it's SBKD data. Similarly, no research was found on how to send the SBKD as efficiently as possible. While the problem was not included in the task description, we found the problem interesting, and the lack of existing research on this made us see an opportunity to contribute to this field.

4.2.1 Implementation strategies

We began with the understanding that different file types would have varying sizes. Four distinct file types were thus selected for analysis: text (txt), Comma-Separated Values (CSV), JavaScript Object Notation (JSON), and Protocol Buffers (protobuf). The txt and CSV formats were chosen because they can be created with minimum extraneous information, thus preserving only the most critical data. The JSON and Protobuf formats were selected for their prevalence in API implementations, scalability and their robust serialization capabilities[2,3], which can streamline data handling once it has been transferred.

We also explored the possibility of file compression before transmission to reduce file size. Several compression methods exist, but we opted for Gzip due to its widespread use and the fact that the compression is considered lossless. Other compression methods might have provided a better compression ratio or greater efficiency, but since the objective here is to explore possibilities for saving data during transmission rather than expending significant resources to find the absolutely best method, Gzip was deemed sufficient.

4.2.2 Data Transfer Results

In order to obtain the relevant data for analysis, we utilized the dataset provided by [13]. Specifically, we extracted the initial 10,000 keystrokes from this dataset and converted the pertinent information into various data types to generate four distinct file types. Consequently, we created 10,000 files each in CSV, TXT, Protobuf, and JSON formats, allowing us to analyze the relationship between file size and the number of keystroke entries, up to 10,000 keystrokes.

Additionally, we employed the Gzip compression utility with a compression level of 9 (which provides the best compression ratio, at the cost of compression speed) to compress all these files, resulting in another 40,000 files. In total, 80,000 files were generated for further analysis.

Subsequently, we utilized the Matplotlib graphing library to visualize the file size differences in relation to the number of keystroke entries. Furthermore, tables 1 and 2 were generated to provide a representation of the file size development.

A more detailed description on how this data was created, the variables used to create the files, and how the testing how done can be found in Appendix G.

4.2.2.1 Graph Intervals and Their Relevance

In the report, we have chosen to present the graphs across four distinct ranges: [1, 50], [50, 1000], [1000, 10000], and [50, 10000]. The rationale for these ranges

[2] `https://infinum.com/blog/json-vs-protocol-buffers/` - Fetched 08.04

[3] `https://jsoneditoronline.org/indepth/compare/json-vs-csv/` - Fetched 08.04

is as follows:

- The [1, 50] range: Represents file sizes when data is sent with each message. This range helps in understanding the overhead associated with each transmission.
- The [50, 1000] range: Represents scenarios with smaller batch sizes or longer messages.
- The [1000, 10000] range: Demonstrates potential file sizes when opting for larger batch sizes.
- The [50, 10000] range: Provides a comprehensive overview of data, excluding the [0, 50] range to ensure clarity. This exclusion prevents extreme values from disproportionately enlarging the y-axis and obscuring insights.

4.2.2.2 Graphs

Total size of file types

The graphs in Figure 4.2 shows a linear relationship between the number of key entries and the file size. Of particular interest is the observation that the size of Protobuf files is very close to that of CSV and TXT files. As can be calculated from the values in Table 4.1, when the files are uncompressed, Protobuf uses only 6% more space for the first character. However, the difference increases in favor of the CSV file, but it stabilizes at larger number of key entries with Protobuf using 33% more space than CSV. Considering the speed with which Protobuf can be parsed and its ease of use, these are very favorable results, especially if the data were to be sent uncompressed.

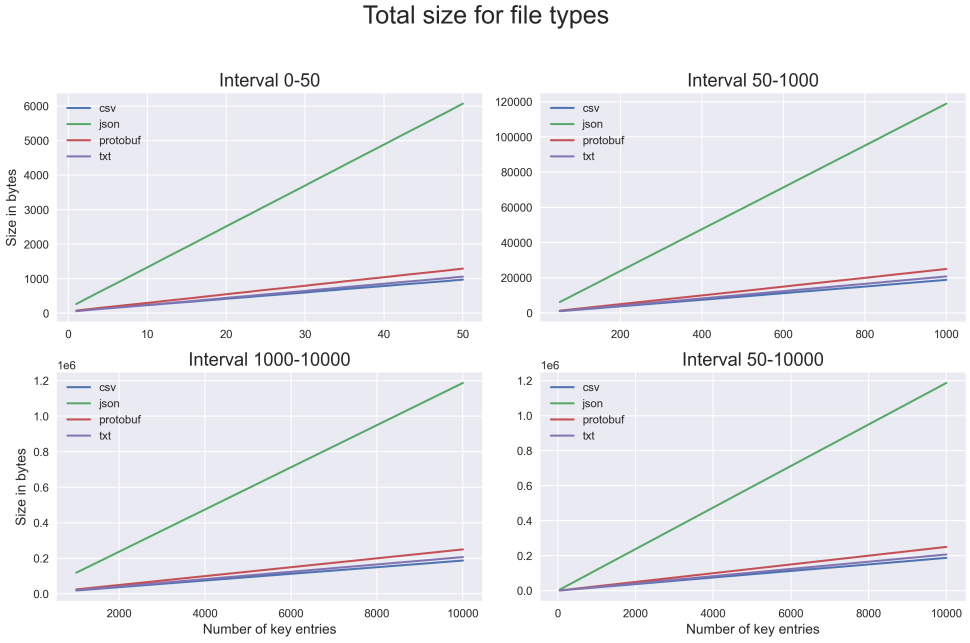


Figure 4.2: Total Size for file

Total size for compressed file types

A number of noteworthy findings emerge from the graphs depicted in Figure 4.3. The data in Table 4.1 show significant space saving, ranging from 76.7% for CSV files to 95.2% for JSON files. Of interest is the observation that, for approximately 9 to 1,900 key entries, the total file size of JSON is smaller than that of Protobuf. This can likely be attributed to the fact that Protobuf is stored as binary files, which could have fewer duplicated characters for Gzip to effectively compress. Alternatively, it may be that Gzip is simply just not as effective at compressing binary files.

Furthermore, the graphs indicate that both TXT and CSV files consistently maintain a significantly smaller size than both JSON and Protobuf. The reduction in file size is initially around 30% for 10 key entries, but this difference decreases to around 20% for 10,000 key entries. The differences in file size are less pronounced with compressed files than with uncompressed ones, meaning that there is a bit more freedom in which files to use as the relational differences between then, bandwidth wise, will be smaller should they be compressed

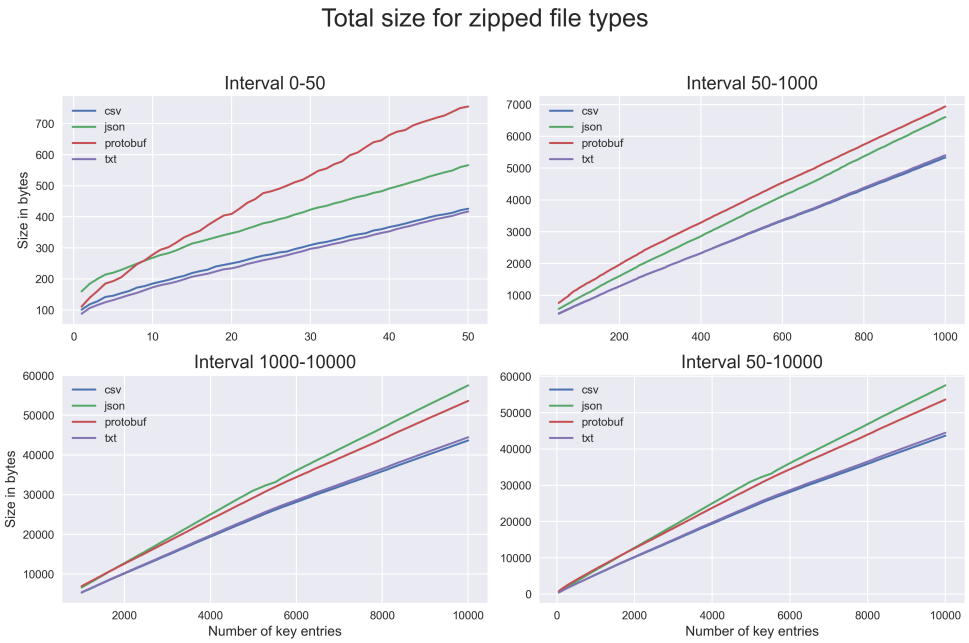


Figure 4.3: Total Size for zipped file

Total size per key entry for file types

In many ways, the most relevant factor for AIBA is the amount of space each keystroke occupies. An English text formatted in UTF-8 typically requires one byte per character. However, significantly more space is needed when transmitting SBKD. This underlines the importance of using short variable names to minimize the amount of extra data sent. In Figure 4.4's graphs for size per key entry of uncom-

pressed files, we observe that the size quickly stabilizes. This is consistent with Figure 4.2 which had a linear correlation between the number of key entries. Table 4.2 shows that CSV quickly reaches 18.82 bytes per key entry when there is 250 key entries in the file. TXT rises to 20.76 bytes, Protobuf to 25.06 bytes, and JSON stands at 119.20 bytes per key entry. From this data, we can see that the structure of the files and their corresponding SBKD data result in file sizes that are at least 18 times larger than if only the messages were sent, and up to 119 times larger if the data is transmitted as JSON files.

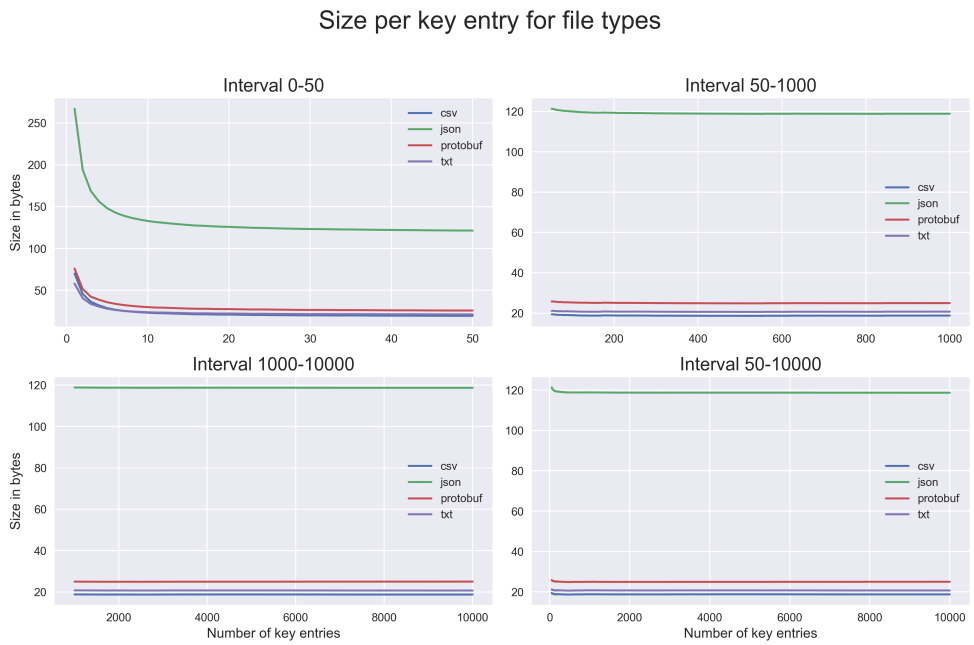


Figure 4.4: Size per key entry

Total size per key entry for compressed file types

However, the size difference becomes much more acceptable when data is compressed prior to transmission, which is shown in Figure 4.5, significantly improving the results for larger files. This makes this graph particularly noteworthy. At 250 key entries, the CSV file is already reduced to 6.267 bytes per key entry and declines further to 4.36 bytes per key entry at 10,000 entries. The TXT format is almost the same, with 6.296 bytes at 250 key entries and 4.4454 bytes at 10,000. As previously stated, Protobuf exhibits weaker results in the 10-1900 key entries range. Nevertheless, it still significantly outperforms its uncompressed version at 250 key entries, with 9.33 bytes per entry, and drops to 5.36 bytes at 10,000 key entries, which is almost exactly 1 byte more than CSV, the most cost-effective file format. The JSON format demonstrates the most significant improvement in comparison to its uncompressed counterpart. It reduces from size per key entry from 119 bytes at 250 key entries to 7.8 bytes, and 5.75 bytes at 10,000 key entries.

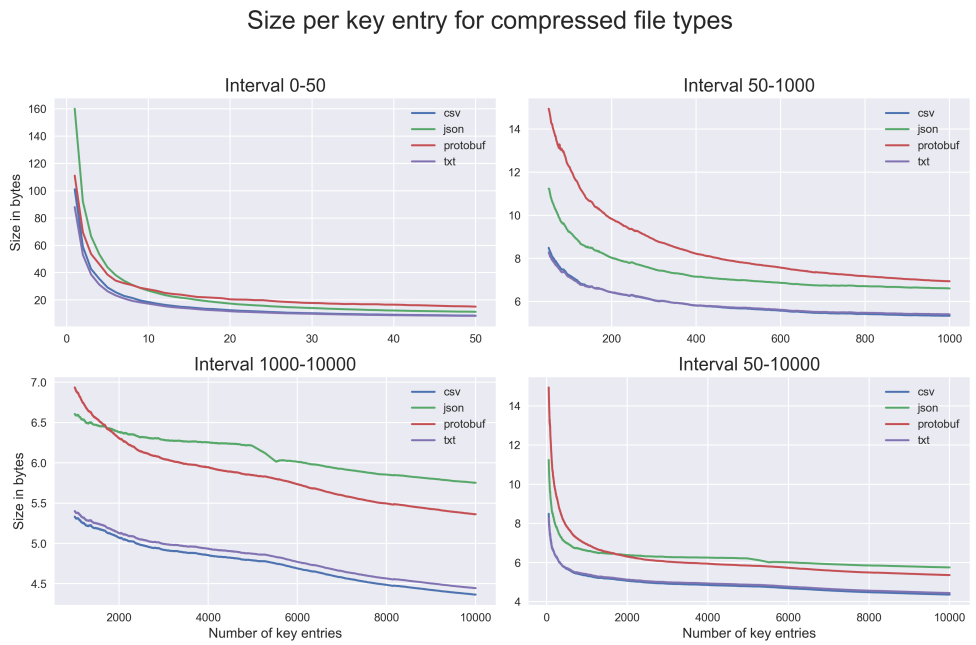


Figure 4.5: Compressed size per key entry

Number of key entries:	Un-compressed files				Compressed files			
	csv	txt	protobuf	json	csv	txt	protobuf	json
1	70	58	76	267	101	88	111	160
10	232	238	299	1,329	185	173	278	268
50	973	1,059	1,291	6,070	426	417	755	566
250	4,704	5,190	6,265	29,801	1,569	1,574	2,333	1,951
1,000	18,792	20,748	24,989	118,859	5,332	5,402	6,937	6,604
5,000	93,913	103,591	124,713	593,702	23,952	24,364	29,241	31,037
10,000	187,381	206,748	249,992	1,186,860	43,651	44,454	53,615	57,526

Table 4.1: Total size in Bytes per file type

Number of key entries:	Un-compressed files				Compressed files			
	csv	txt	protobuf	json	csv	txt	protobuf	json
1	70.00	58.00	76.00	267.00	101.00	88.00	111.00	160.00
10	23.20	23.80	29.90	132.90	18.50	17.30	27.80	26.80
50	19.46	21.18	25.82	121.40	8.52	8.34	15.10	11.32
250	18.82	20.76	25.06	119.20	6.28	6.30	9.33	7.80
1,000	18.79	20.75	24.99	118.86	5.33	5.40	6.94	6.60
5,000	18.78	20,72	24.94	118.74	4.79	4.87	5.85	6.20
10,000	18.74	20.67	24.99	118.69	4.37	4.45	5.36	5.75

Table 4.2: file size per key entry in Bytes per file type

4.2.2.3 Size Differences in Data Formats Relative to Key Entries

We can see that the compression of data prior to transmission can result in a significant reduction in the data volume that needs to be transmitted. Compression reduces the size of data from 77% for CSV files and up to 95% for JSON. This large decrease for JSON is likely due to JSON's structural data such as brackets and quotations stored in plain text. In contrast, Protobuf is stored in binary format, which eliminates the need for human readability and offers benefits similar to those of JSON. By compressing data, particularly JSON, the size reduction can reach 95%. CSV files are shown to have the least data per key entry, while Protobuf surpasses JSON in efficiency after 2,000 key entries.

4.2.2.4 Progressive SBKD Transmission Strategy

The transmission of SBKD data would undoubtedly result in a significant increase in network costs due to the larger data transfer. Additionally, to quickly map critical data, it would be impractical for AIBA to batch all messages until reaching 10,000 characters each time they are sent, as potential issues may have already occurred by the time the first 10,000 keystrokes are sent. This results in significant delays for moderators to assess potential age discrepancies among speakers. Therefore, a more gradual approach to transmission may be beneficial for AIBA. For instance, the initial 10 messages could be sent individually for immediate analysis and a rough age estimate. As the conversation progresses, more messages can be batched until a certain key entry sum is reached, or a specific time has elapsed since the first batched message. This method would allow AIBA to quickly obtain information about the SBKD of users, while subsequently reducing cost per user as more data is collected about the specific user to confirm that the initial estimates remain unchanged. While this approach make new users expensive as their baseline is calculated and smaller messages batches are sent, it would be a significantly lower overall data cost than if all messages are sent individually, and provide a minimal delay for the moderator to obtain an age prediction for the users.

4.2.3 Final thoughts on compression and file sizes

While the compression of files undoubtedly improves file sizes, it is not without costs. For example, files must be compressed on the users devices. This process consumes relatively few data resources, typically during less intense sections of games where messaging is feasible, and the computational demand is lower, it might be imperceptible if one of the CPU cores on the device takes 0.1 seconds (the longest duration recorded to compress a file with 10,000 key entries in our tests). Nevertheless, many video game players are sensitive to any performance degradation, regardless of the reason. While many might accept this trade-off to protect children in video games, it risks criticism from some players.

More efficient compression options exist. while Gzip is designed to operate on a single processor core, alternatives like Pigz can utilize multiple cores, thereby speeding up the compression process. An alternative method, Brotli, may take

longer to compress data but often achieves a better compression ratio than Gzip. We have not tested Brotli or Pigz, so we are unable to provide specific insights on how they would perform with our data. Nevertheless, they provide interesting fields for further testing.

If AIBA can manage the additional overhead of parsing CSV data compared to JSON or Protobuf, it is clear that this would be the most cost-effective method for receiving data. If neither CSV nor TXT is viable, the choice then depends on whether the data can be compressed. If compression is not feasible, Protobuf is the most appropriate choice due to its efficiency in both size and parsing. In the event that compression is feasible and the choice is between JSON or Protobuf, it becomes significantly more challenging to determine the most economical option. If the majority of communications consists of shorter messages, then it is likely that JSON will be the most cost-effective option. However, if messages are batched into larger collections more frequently, then Protobuf may be the more cost-effective option.

5 State of the Art

This chapter will firstly present a market scan into how other enterprises conduct age verification of it's users, and what other enterprises employ Soft Biometric Keystroke Dynamics in one way or another. Furthermore, it will look into the challenges that are faced with online game moderation in relation to text chats. It will examine the prevalence of toxic behavior in game chats, the existing solutions to help moderators address these issues, and the functional gaps in these current solutions.

5.1 Market scan

In order to explore the problem space, a market scan was conducted on currently existing age verification for children in social media and games. This section aims to summarize the results of the data found on each social media and game platform in Appendix H. It shows what age verification solutions are available in popular social media and games. Additionally, it indicates whether Soft Biometric Keystroke Dynamics are used when verifying age. Furthermore, looking for existing solutions that occupy a similar domain to the SDK could help expand or improve the field of SBKD

Due to the duties imposed by COPPA and GDPR-K and ensuring compliance with these regulatory frameworks, many of the age verification methods utilized among all actors are similar. Table 5.1 and Table 5.2 show the available age verification methods that are used by different social media and game platforms.

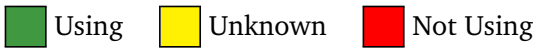
The following is a description of what methods may be supported on the different platforms:

1. **Age specification**: Users are asked for their age upon account creation
2. **Community reporting**: Users of the platform may report other users they believe violate guidelines
3. **Moderator reviewing**: Platforms that have moderators that supervise the content that is published or exists
4. **Parental controls**: Platforms that allow parents to supervise the content their child can access
5. **ID verification**: Users having to provide legal documents to prove their age
6. **Video verification**: Sending a video doing a defined set of tasks in front of a camera which is then analyzed
7. **Face age estimation**: Similar to video verification, a user takes a selfie which is analysed to determine age
8. **Age vouching**: Enabling friends or parents of users to vouch for a persons age
9. **AI review**: Employing an AI algorithm that looks at content, analyses and attempts to draw conclusion based on user behaviour
10. **Keystroke dynamics**: Using either a fixed-text or continuous analysis of user typing dynamics to authenticate users

5.1.1 Social Media

Given the popularity of social media among children and young people, it is particularly useful to explore. It is estimated that around 40% of American children between the ages of 8 and 12 years old use social media. Furthermore, 95% of children between 13 and 17 years old in the US use social media[1].

The social media we have explored were chosen for their active monthly users and popularity[2]; Facebook, Instagram, Youtube, TikTok and Snapchat.



Methods\Social Media	Facebook	Instagram	TikTok	Youtube	Snapchat
Age specification					
Community reporting					
Moderator reviewing					
Parental Controls					
ID verification					
Video verification					
Face age estimation					
Age vouching					
AI reviewing					
Keystroke dynamics					

Table 5.1: Age verification methods used in popular social media

Users that specify their age to be below 13 or younger than their respective minimal age of digital consent in their country, are locked out during the user creation process altogether (Appendix H). To combat attempting to specify a false age, all social media examined employ community reporting and moderator reviewing of reported users. Instagram, Facebook, TikTok and YouTube may also require ID verification to access certain content.

Age estimation software solutions such as face age estimation or video verification are adapted by Facebook, Instagram and TikTok. Facebook and Instagram use Yoti[3] for this purpose. It was previously stated that TikTok also used the same company for face age estimation, but the article is no longer accessible (Appendix H).

Facebook and Instagram, in some cases, use AI for scanning posts, birthday messages, other linked accounts across Meta or other behavioural patterns to find and remove underage accounts.

When looking into whether these services utilized keystroke dynamics for authentication, TikTok was the only actor that specified that they captured key-

[1] `https://health.clevelandclinic.org/dangers-of-social-media-for-youth` - Fetched 06.03

[2] `https://en.wikipedia.org/wiki/List_of_social_platforms_with_at_least_100_million_active_users` - Fetched 06.03

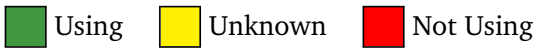
[3] `https://www.yoti.com/` - Fetched 07.03

strokes in their privacy policy. TikTok currently capture keystroke information and patterns to: "Verify the authenticity of an account" or as they specified in their terms of service: "for risk control, debugging, troubleshooting, and monitoring for proper performance"[4]. It is likely that they use keystroke dynamics for some form of authentication, but the purpose is not clear and was not found during this analysis.

Different media outlets such as The Guardian and The Register have claimed that Facebook and Instagram captured these events on their in-app browsers when accessing external websites on iOS mobile phones. This was based on a report from Felix Krause[5,6] but these claims are mostly unsustained and are not conclusive.

5.1.2 Games

An article published in the national library of medicine in 2023 estimates that more than 90% of children above the age of 2 years old play video games [1]. Looking at how age is verified in game services is therefore also an important factor to consider, as they in the same article estimate that children between the ages of 8 and 17 spend 1.5-2 hours playing video games each day.



Methods\Game platforms	Roblox	Epic games	Riot games	Steam
Age specification				
Community reporting				
Moderator reviewing				
Parental Controls				
ID verification				
Video verification				
Face age estimation				
Age vouching				
AI reviewing				
Keystroke dynamics				

Table 5.2: Age verification methods used on popular game platforms

In their terms of service, the game providers, Roblox, Epic games, Riot games and Steam, urge parents to take a more active role in supervising their childs online presence rather than incorporate comprehensive age verification solutions (Appendix H). From Table 5.2 we can, however, see that Roblox and Epic games

[4] `https://www.tiktok.com/legal/page/eea/privacy-policy/en` - Fetched 13.03

[5] `https://krausefx.com/blog/ios-privacy-instagram-and-facebook-can-track-anything-you-do-on-any-website-in-their-in-app-browser` - Fetched 06.03

[6] `https://krausefx.com/blog/announcing-inappbrowsercom-see-what-javascript-commands-get-executed-in-an-in-app-browser` - Fetched 07.03

apply age authentication methods on par with some of the social media surveyed (Table 5.1)

All gaming platforms taken into consideration have solutions for underage accounts. Steam adds functionality for *Junior mode* accounts and Epic games have *Cabined* accounts. These accounts allow underage accounts to play games, but restrict communication with other users and account visibility. Roblox and Riot games grant parents the right to supervise and limit what the accounts of their children can access.

One of the potential reasons that Roblox have a wider variety of authentication methods, may be because of its user base. In 2022, 47% of all Roblox users were under the age of 13[7]. This creates an extra responsibility and underlines the need for separation between the accounts.

Roblox also closely monitors content posted in chats or on their platform. Furthermore, they "train models" that detect such content. This way they can use a method that detect breaches in their terms of conditions quickly and efficiently that becomes more sophisticated as time goes on (Appendix H).

5.1.3 SBKD data capturing

There is currently no software that captures keystroke dynamics metadata in Unity, neither on the Unity asset store nor as downloadable add-ons for Unity. The closest asset found was, Dishook[8], an asset that enables sending a Discord message from a Unity game.

During this market scan, the group found no tools or providers that focused on profiling using keystroke dynamics. There are, however, several other third party actors that provide software that capture keystroke metadata for use in authentication on customer platforms and services.

5.1.3.1 TypingDNA

One such service is TypingDNA, who provide a means for continuous authentication, 2 factor authentication and an authentication API using keystroke dynamics authentication services[9]. The authentication API alongside its typing biometrics recorder are the components they provide that are most likely to be comparable to the SDK we are developing.

The API and capture software is also easy to use and can easily be merged into a technology stack without much trouble. The frontend should include the typing biometrics recorder, while the backend should communicate with their REST API to get the results[10] (Figure 5.1).

When looking at their public JavaScript source code, latency (*seek time*) and

[7] `https://create.roblox.com/docs/production/roblox-user-base` - Fetched 07.03

[8] `https://assetstore.unity.com/packages/tools/network/dishooks-send-discord-messages-from-your-game-171381` - Fetched 07.03

[9] `https://www.typingdna.com/` - Fetched 06.03

[10] `https://api.typingdna.com/#api-overview-how` - Fetched 07.03

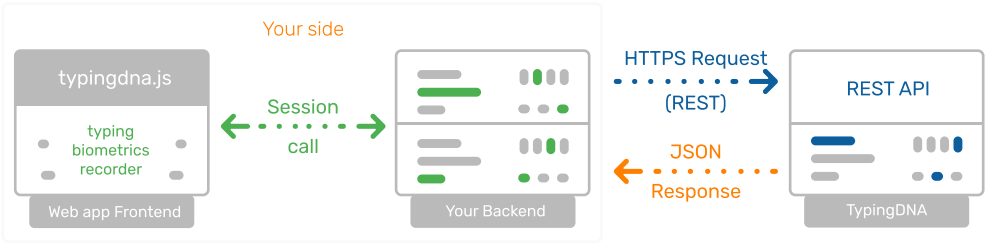


Figure 5.1: TypingDNA worflow[10]

duration (*press time*) are the attributes they capture[11]. Additionally, their software supports fetching keystroke dynamics for mobile devices, a number of operation systems and different browsers.

TypingDNA only provides authentication rather than profiling of individuals. Its purpose does therefore not align with AIBA's in terms of profiling based on keystroke dynamics. Its typing biometrics recorder does, however, incorporate most of the things we would like the SDK to also do.

5.1.3.2 Other solutions

LexisNexis' BehavioSec[12] provides an analysis tool for continuous authentication using AI. In short, the tool detects non-human behaviour on the deployed service and flags such behaviour. It employs a penalty-and-reward model for this purpose.

Plurilock provides a continuous authentication solution called Defend which uses keystroke dynamics[13]. Defend notices moderators or staff when user behaviour exceeds their defined *considerable risk threshold*.

5.1.4 Summarization

Across many of the game platforms and social media studied, there is no single best way of authenticating age of users. For many services, figuring out the age of an account without overstepping the boundaries of the data they are allowed to collect is troublesome (Appendix H).

- ALL applications allow users to specify age, community reporting of accounts and moderator reviewing of content
- 6 of 9 services studied employ identity verification through IDs
- 5 of 9 services use face age estimation or video verification to authenticate
- 3 of 9 services use AI or train models to detect content and figure out information and estimate the age of an account
- Potentially 3 of 9 services utilize keystroke dynamics in some way or another, where 1 of these services explicitly specify doing so to authenticate users

This analysis describes the current state of age verification in popular social

[11] `https://github.com/TypingDNA/TypingDnaRecorder-JavaScript/blob/master/typingdna.js#L266` - Fetched 08.03

[12] `https://risk.lexisnexis.com/products/behaviosec` - Fetched 09.03

[13] `https://plurilock.com/products/defend/` - Fetched 09.03

media and games. An addition to the already established ways of authenticating users could be through the use of Soft Biometric Keystroke Dynamics. There are already software solutions that provide authentication through keystroke dynamics as a service, but no applications of verifying age through keystroke dynamics was found during the market scan.

5.2 Online Game Moderation Challenges

This section aims to provide an overview of current moderation practices in game chats, the extent of content that needs moderation, available solutions to enhance moderation tools and the common challenges in moderating text-based game chats.

5.2.1 Current moderation practices

Various issues with user-generated content often demand a distinct moderation approach. These practices could be divided into two main categories: manual and automated moderation[14], each with its own strengths and weaknesses. Automated moderation is generally viewed as a supplement to manual moderation rather than a potential replacement.

5.2.1.1 Manual moderation

Manual moderation relies on human moderators, either paid staff or volunteers, who review messages. This process can involve reviewing each message sent, but more commonly, it prioritizes messages flagged for violating the game's rules[15]

Advantages of Manual Moderation:

- **Precision:** Humans generally understand context and nuance better than automated systems, reducing errors.
- **Complex Situations:** Humans are capable of handling complex situations that automated services are not trained for or capable of resolving.
- **Percieved Fairness:** Humans may be perceived as more fair than automated systems when moderating content.

Disadvantages of Manual Moderation

- **Resource intensive:** Resource-Intensive: Requires a significant investment of time and labor, making it challenging to expand.
- **Human Error:** Moderators are susceptible to error and inconsistency, which may impact the outcomes of moderation.
- **Longer Reaction Time:** Messages are typically reported before they can be deleted or addressed, which allows harmful messages to impact other players before action is taken.

[14] `https://getstream.io/blog/chat-moderation/` - Fetched 10.04

[15] `https://www.checkstep.com/content-moderation-a-comprehensive-guide/` - Fetched 10.04

- **Emotional Burden:** The process of manual moderation has been found to place an emotional burden on moderators [14].

5.2.1.2 Automated Moderation

The complexity of automated moderation can vary considerably. It can be as simple as blocking specific words by utilizing a block list or as advanced as using Natural Language Processing (NLP) and machine learning to better understand the intentions, context, and meanings behind messages.[16] These moderation tools rely on predefined rules to make decisions.

Advantages of Automated Moderation:

- **Scalability:** Unlike manual moderation, which is inherently resource-intensive, automated systems are capable of reviewing every message before it reaches other players. Consequently, harmful messages can be prevented from causing any form of emotional distress to other players.
- **Consistency:** Automated systems apply rules in a uniform manner, thereby avoiding the inconsistencies that can arise when different human moderators have varying standards.
- **Efficiency:** Machine learning and NLP can analyze each message, categorizing its content and identifying those that violate established rules or that require further review by human moderators.

Disadvantages of Automated Moderation:

- **Lack of Context Understanding:** Although AI models such as GPT-4 or similar LLMS have made significant advancements, they remain susceptible to limitations in their ability to comprehend nuances and context. This can result in false positives or missed detections when not adequately trained.
- **Handling Complex Situations:** Automated systems may encounter difficulties in managing complex and unusual events that are not well-defined in their rule sets. This may result in the failure to flag potentially dangerous conversations.
- **Dependence on Training Data:** The quality of the training data utilized by automated systems is of vital importance. In the case of games such as MovieStarPlanet, where users frequently insert typos other techniques with the intention of circumventing filters, these systems may encounter difficulties if not trained on data that includes such instances. [15]

5.2.2 Scope of the problem

This text will primarily focus on moderation in text-based chats in video games. However, it will also include studies on voice-based chats to provide a more comprehensive understanding of the scope of toxic behavior in game chats, due to the lack of research exclusively focusing on text-based conversations in video games.

[16] `https://www.cometchat.com/blog/what-is-chat-moderation` - Fetched 10.04

5.2.2.1 Prevalence of Harassment

A 2022 study [16] involving 15,000 Discord users who used voice chat for communication identified 25,291 instances of "offensive" behavior. Among the users, 26.43% had one or more instances of offensive outbursts. Of these, 21.39% were classified as minor, while 5.03% of users had at least one severely offensive outburst. The distribution of outbursts was as follows:

- 53% racial/cultural hate speech
- 33% sexual vulgarity
- 12% gender/sexual orientation hate speech
- 1% other offensive speech

The study revealed that 16.58% of minors had an offensive outburst, compared to 36.28% of adults. For severe outbursts, adults were almost three times more likely to have at least one compared to minors (7.29% for adults vs 2.77% for minors). Adults were more likely to engage in racial and cultural hate speech (55.63%), whereas minors exhibited higher rates of sexual vulgarity (38.17%) and gender/-sexual orientation hate speech (18.23%), compared to adults (32.22% and 11.25%, respectively). The study used the tool "ToxMod" to collect data. More details on ToxMod will be provided in Section 5.2.4.1

Another 2022 survey conducted by the Anti-defamation League (ADL) [17] interviewed 2,134 Americans who play games on PCs, consoles, and mobiles, with 1,931 of them playing online games. The survey has a margin of error of 2-3%. This survey revealed that 66% of teenagers (13-17) and 70% of pre-teens (aged 10-12) had experienced harassment in video games. Additionally, 15% of young people (aged 10-17) had been exposed to white supremacist ideologies. The percentage of adults experiencing harassment increased from 65% in 2019 to 77% in 2022. Rates of harassment among teenagers (aged 13-17) vary by game, with 46% in Minecraft and 80% in Valorant reporting harassment. Similarly, exposure to white supremacist ideologies also varies, with the lowest prevalence observed in Destiny 2 (0%) and the highest in PUBG: Battlegrounds (32%).

The most comprehensive study we identified was conducted by the London School of Economics and Political Science in 2012 [18], which surveyed 25,142 children aged 9-16 from 25 different countries. This study focused more on children's exposure and behavior on the internet in general rather than specifically in video games. The study revealed that 12% of European children aged 9-16 had been bothered or upset by something online, with most not reporting it to parents or others. The study noted that children do not necessarily perceive risks as upsetting or harmful. For example, 1 in 8 children have seen sexual images or received sexual messages online, but they generally do not find these experiences inappropriate or harmful. The probability of encountering a risk increases with age: 14% of children aged 9-10, 33% of 11-12-year-olds, 49% of 13-14-year-olds, and 63% of 15-16-year-olds reported having experienced scenarios that the London School of Economics and Political Science labeled as risk. Additionally, 15% of children aged 11-16 have received sexual messages or images directly from other users,

with 3% of these children reporting that they had sent such messages. Of those who received sexual messages, a quarter reported being bothered by them.

The study also highlighted significant gaps in parents' awareness of their children's online experiences. which landed on the following numbers:

- 40% of parents whose children had seen sexual images online believed their children had not seen such content.
- 56% of parents whose children had received messages that were hurtful or unpleasant were unaware of this fact.
- 52% of parents whose children had received sexual messages believed that their children had not received such content.
- 61% of parents were unaware that their children had met someone in person with whom they had initially communicated online.

5.2.2.2 Impact of Toxic behaviour

The harm caused to players exposed to toxic behavior from other players is noteworthy. The ADL's 2022 survey [17] provides the graphs in Figure 5.2 representing how toxic game chats affect youth.

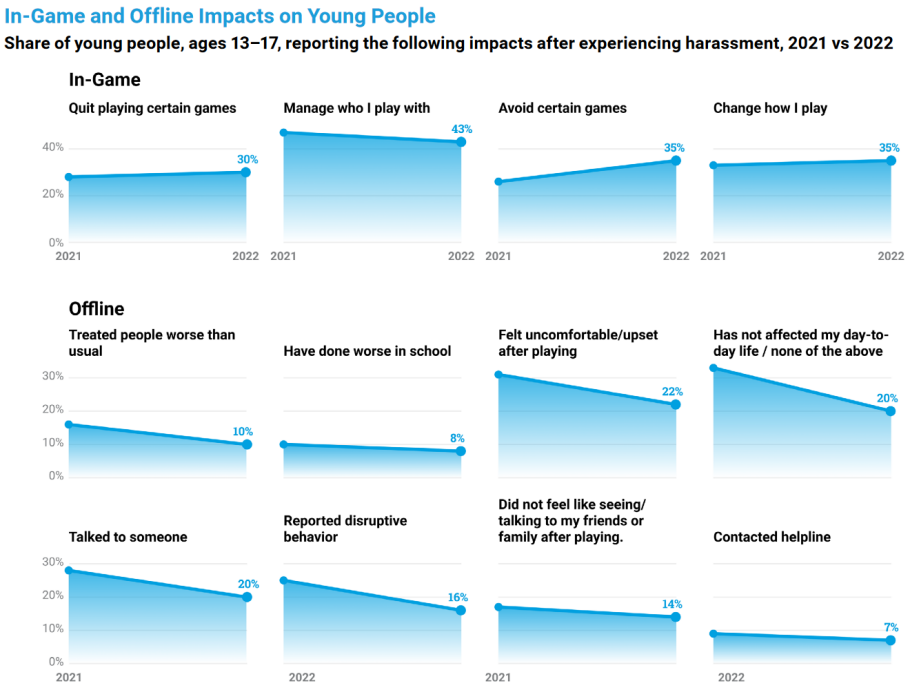


Figure 5.2: Graphs from ADL about impact of harassment on young people [17]

From these graphs, we observe that nearly one third of young players stop playing certain games due to toxic behavior. The negative impacts extend to academic performance and interpersonal relationships. Only 20% of players report that toxic behavior does not affect them. The most common responses to toxic behavior include limiting the games they play, altering how they play, and choos-

ing whom they play with. Additionally, only 16% of young players report such behavior.

For adults, the ADL provides the graphs in Figure 5.3 representing how they are affected by toxic behavior.

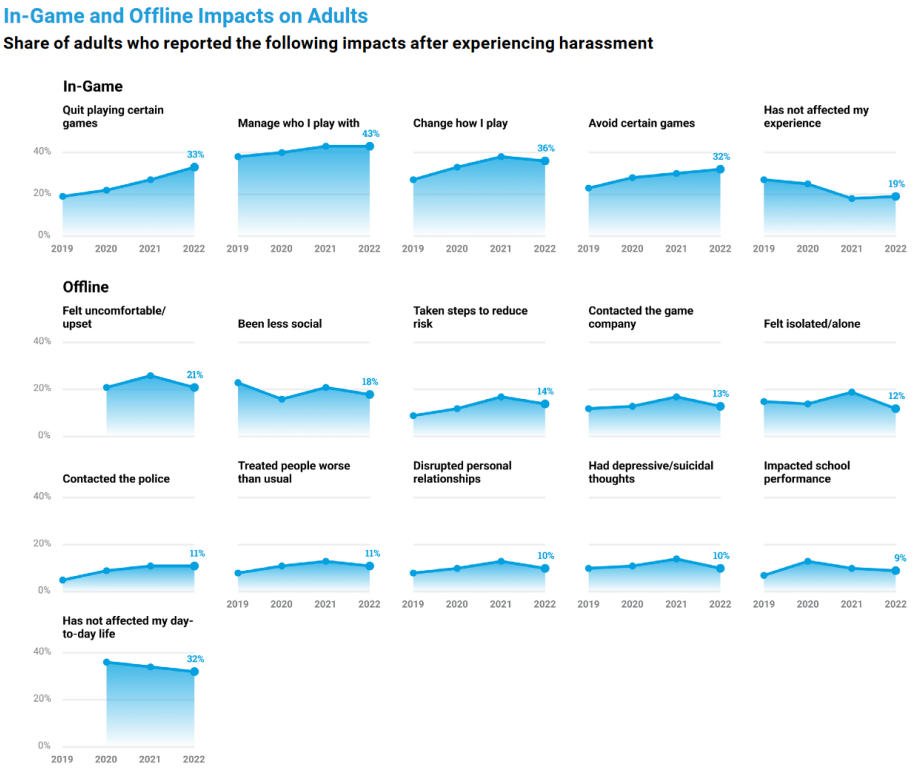


Figure 5.3: Graphs from ADL about impact of harassment on young people [17]

The data indicates a general increase in the impact of toxic players on adults. Toxic behavior commonly influences what games are played, who players choose to play with, and their playing style. Nearly one-fifth of adults report becoming less social, and 12% feel isolated and lonely. Alarmingly, 11% of adults have contacted the police over something said in game chats, highlighting the severity of these interactions. Moreover, adults also report that toxic behavior causes them to treat others poorly and affects their personal relationships. Furthermore, 10% have experienced depressive or suicidal thoughts as a result of harassment in game chats.

5.2.3 Grooming

On the topic of grooming. In the UK there have been, on 150 different application, 34,000 cases of online grooming in the last 6 years, with an 82% increase from 2017/18 to 2022/23 [https://www.nspcc.org.uk/about-us/news-opinion/2023/2023-08-14-82-rise-in-online-grooming-crimes-against-children-in-the-last-5-years/]. In 2009 the UK police receive more than 100 alerts every month from child internet

users who are in immediate danger of sexual abuse or violence from online predators [https://www.theguardian.com/uk/2009/feb/25/internet-children-sexual-predators-abuse].

It is Estimated that there are 500,000 predators online each day, and that an estimated 89 percent of sexual advances directed at children occur in Internet chatrooms or through instant messaging [https://childsafety.losangelescriminallawyer.pro/children-and-grooming-online-predators.html] Of these, One out of every four reported cases of child exploitation involves a sexual predator requesting sexual images from the child. [https://susiesplace.org/prevention/gaming-safety/]

5.2.4 Existing Solution

Given the widespread issues of harassment and toxic behavior in video games, it is evident that purely manual moderation is not a scalable solution. This is especially true since many players do not report being subjected to such behavior. Therefore, automated moderation tools are necessary to handle obvious rule violations and identify unreported incidents. These tools can significantly alleviate the burden on human moderators by addressing clear infractions and uncovering cases that go unreported.

5.2.4.1 ToxMod and Unity Safe Voice

ToxMod[17] is a proactive voice moderation tool and, according to its developers, the only voice moderation tool built on advanced machine learning models. It goes beyond keyword matching to interpret the tone, emotions, and context of conversations. Available in 18 different languages, ToxMod can accurately detect whether users are minors or adults based on their voice chats. It can also identify and flag sexual vulgarity. However, ToxMod's limitation is that it currently only supports voice chat, but not text-based conversations.

Unity Safe Voice[18] appears to be an alternative to ToxMod. It is an automated tool for classifying conversations and is also designed for voice chat. However, it does not have the capability to determine the age of the speakers, which may make it difficult to identify when adults are talking to children.

5.2.4.2 Vivox and WebPurify

Vivox offers a native solution for text moderation[19], but appears to be limited to filtering specific words or phrases. WebPurify[20] is slightly more comprehensive; it can categorize the topics of conversations and flag content that moderators should review. Like Vivox, WebPurify also allows blocking specific words or phrases. However, neither WebPurify nor Vivox can predict the age of users.

[17] `https://www.modulate.ai/tox-mod` - Fetched 12.04

[18] `https://www.google.com/search?client=firefox-b-d&q=unity+safe+voice` - Fetched 12.04

[19] `https://unity.com/products/vivox-text-chat` - Fetched 12.04

[20] `https://www.webpurify.com/` - Fetched 12.04

5.2.5 Remaining Challenges

While machine learning is employed by some moderation tools, it appears that standard word filters are still commonly used to moderate text chats, with manual moderation typically reserved for messages flagged by other users. For instance, MovieStarPlanet is a popular game that operates using extensive lists of alert and blacklisted words [15]. Here the blacklisted words would be blocked from being sent, whereas alert words accumulate for individual users as their use does not necessarily indicate malicious intent but should lead closer scrutiny for those who frequently use them.

5.2.5.1 Technical Challenges

The study from Cheong et al. [15] highlights the challenges associated with games like MovieStarPlanet, where many users frequently misspell words, use abbreviations, or employ alternative terms to circumvent the filters. These misspellings and abbreviations can also hinder machine learning models from keeping up with language evolution. The study observed that their model could achieve up to 98% accuracy in detecting predatory users, but this required significant preprocessing. It is also uncertain whether the model would remain effective over time as player language evolves away from the data it was trained on.

Additionally, there is a noticeable lack of text-based moderation tools that map the age of users. Estimating age based solely on text analysis appears to be more challenging. None of the tools we identified could predict age based purely on users' text data alone.

5.2.5.2 Reporting Issues

Another issue is the under-reporting of incidents in games. As previously noted, only a small number of players report harassment. Many players choose to leave the game rather than report the behavior. This may be due to a widespread belief that the video game industry does not effectively combat abuse and harassment in games [19], leading players to see no point in reporting incidents.

5.2.5.3 Cultural and Linguistic Variations

The most effective tool for the analysis of multiple languages was ToxMod, which is capable of handling 18 different languages. This highlights a market gap in the availability of automated moderation tools for less commonly spoken languages.

5.2.6 Conclusion

While there are several analysis tools for both text- and voice-based conversations, the extent of abusive and harassing behavior in games is significant enough that human moderators could benefit from all available automated moderation tools. Reports indicate that players are greatly affected by harassment in video games. Additionally, we found a lack of text-based services capable of predicting players' age and gender. This is an area where Soft Biometric Keystroke Dynamics could be beneficial.

Given that children rarely report incidents and that approximately half of parents are unaware of the content their children are exposed to or the individuals with whom they interact online before meeting in person, this could be an effective method of automatically flagging suspicious conversations for closer examination by moderators to detect unwanted behavior.

6 Legal Frameworks

A significant part of our project involved learning about and analyzing four different regulatory frameworks that have the potential to impact the development of a final product that incorporates the SDK. Our focus was not on how these regulations would affect the SDK itself, as it has limited functionality that would be affected by these regulations. Instead, by focusing on the impact of the regulations on the final product, we can better understand how they will affect a complete software product and identify key compliance points that AIBA should adhere to in order to ensure compliance with the regulations and to address the ethical aspects that arise from these obligations.

While frameworks such as General Data Protection Regulation-K and Children's Online Privacy Protection Act are important when dealing with the collection of data from children, we have prioritized the EU Digital Services Act (DSA), US Kids Online Safety Act (KOSA), EU Artificial Intelligence Act (EU AI Act), and the Online Safety Act 2023 (OSA). These are regulatory frameworks that have recently been implemented or are about to be implemented. Both GDPR-K, the part of GDPR that covers children's privacy, and COPPA were enacted in 2018 and 2000, respectively, meaning that most companies have had ample time to ensure compliance with these frameworks.

The oldest framework we are covering is the DSA, which came into force in the EU on 09.09.2022. The OSA came into force on 26.10.2023. As of 21.05.2024, neither the KOSA nor the EU AI Act has been enacted. While the KOSA is currently in the proposal stage, the EU AI Act was finalized and endorsed by all 27 EU member states on 02.02.2024 and by the European Parliament on 13.03.2024.[1]

6.1 Digital Services Act (DSA)

The EU Digital Services Act is an active regulation which entered into force 16.09.2022, while its rules became applicable on 17.02.2024. The act aims to fulfill the following two goals: [2]:

> *1. To create a safer digital space in which the fundamental rights of all users of digital services are protected*
>
> *2. To establish a level playing field to foster innovation, growth, and competitiveness, both in the European Single Market and globally*

The act affects by definition different "online intermediaries" which are providers service functionality aligns with one or more of these [20] (Chapter 1, Article 3.g):

[1] `https://www.whitecase.com/insight-our-thinking/ai-watch-global-regulatory-tracker-european-union` - Fetched 02.05

[2] `https://digital-strategy.ec.europa.eu/en/policies/digital-services-act-package` - Fetched 03.02

- Facilitates transmission of data over a communication network
- Provides information or data to a recipient of a service
- Stores data provided by a recipient of the service

Moreover, actors who target their services specifically towards countries of the Union and have a "substantial connection" with a member country are the ones concerned by this act [20] (Preliminary, Point 8). The act also defines stricter obligations which are to be followed by "very large online platforms" which are platforms that have more than 45 millions users monthly in the EU [3].

One specific such duty mentioned in the act is to explain terms and conditions in a manner that is understandable to minors [20] (Chapter 3, Section 1, Article 14.3). Furthermore:

> *Providers of online platforms accessible to minors shall put in place appropriate and proportionate measures to ensure a high level of privacy, safety, and security of minors, on their service.*

In more detail, this entails that such a service is obligated to limit collection of personal data and not present advertisements based on profiling for an individual that *they are aware with reasonable certainty* is a minor [20] (Chapter 3, Section 3, Article 28). This obligation does not require providers to take measures in assessing whether a recipient is a minor.

Another essential obligation found in the EU Digital Services Act states that providers of hosting services and online platforms are required to give reasoning for their content moderation and the measures taken [20](Chapter 3, Section 1, Article 15.1c). This would mean that the providers of services that employ the SDK are required to inform the recipient of the measures employed. For this reason, the SDK must be documented well, so that the reasoning behind the matter can be clearly explained to the recipient. The act also specifies a *notice and actions mechanism* requirement which should allow recipients to flag content [20] (Preliminary, Point 50). Recipients who show success in reporting inappropriate or illegal content are defined as *trusted flaggers* (Preliminary, Point 62). The SDK that has been developed will serve as an important tool and facilitate for detection of inappropriate content.

It is not clear whether AIBA per definition is an intermediary service or not. The role of AIBA is in many ways a third party service that analyzes data given from providers. One could in this case argue that it does not fit under any of the definitions and therefore exempt from this act. On the other hand, due to the nature of this act it would be unrealistic to deem oneself unaffected by this act. AIBA processes chat data and could thus be defined as a "hosting service" as per the definition section of the act [20] (Chapter 1, Article 3.g.iii). Because of this uncertainty, more thorough insight into the act is needed to conclude whether AIBA is affected or not.

[3] `https://digital-strategy.ec.europa.eu/en/policies/dsa-vlops` - Fetched 04.02

It is important to note that the DSA eases the responsibility on startups and moderately sized companies, and tailors the different obligations to the size of the company in question [20]. AIBA is a growing actor, but would not be labeled a "very large online platform" due to their services being provided as a tool for companies and thus not fitting the requirements of the definition.

There are other articles that aim to protect minors as recipients of intermediary services. These are, however, directed toward "very large platforms" and small businesses are exempt from these. Thus, they are at this time not vital for the sake of the SDK. These articles can be read in the following sections of the act:

1. Chapter 3, Section 5, Article 34 [20]
2. Chapter 3, Section 5, Article 35 [20]
3. A lot of the preliminaries also include such duties (p.89, p.102, p.104) [20]

6.2 Online Safety Act 2023

The Online Safety Act 2023 was enacted by the British government on the 26th October 2023[4], and can be read in its entirety online[5]. The act provides a regulatory framework ensuring safer use of internet services in the UK. Furthermore, the act imposes a duty upon providers in - and outside of the UK in Part 1, section 1, point 2a to mitigate and manage risk from content containing illegalities and content that is harmful to children.

The act mentions different types of services. Firstly, user-to-user services, i.e. services where content uploaded by one user can be accessed or viewed by other users. Secondly, a search service is where content provided by the service is limited to SMS, MMS, verbal communication or other related identifying content. These services, by definition in this Act are either regulated or not regulated. In this context, regulated means that a service has ties within the UK, or combines user generated content with pornographic content thus already being regulated in other ways. These types of regulated services will from here on be addressed as part 3 services. All the definitions mentioned here are found in part 2 in section 3 of the act.

Considering this act regulates services that provide services as mentioned, the focus in this particular analysis will lie in how AIBA's service and the SDK may benefit providers that fall under the scope of this act. More specifically, how this act affects services which are mainly utilized by children. This is, for instance, mentioned in Chapter 2 and Chapter 4 of the Online Safety Act 2023.

Services likely to be accessed by children are to follow guidelines which facilitates protection of children using the service. Such services who also fall under the scope of a part 3 service are imposed several duties as mentioned in Chapter 2 section 11 point 6 and section 12 point 8 in the same part, some of which are mentioned below:

[4] `https://bills.parliament.uk/bills/3137` - Fetched 01.02

[5] `https://www.legislation.gov.uk/ukpga/2023/50/enacted` - Fetched 01.02

- Providing an up-to-date children's risk assessment
    - User base - number of children in respective age groups
    - Harmful content children may encounter - either within the service or non-designated
    - Separating such content in different age groups
- Notify Office of Communications (OFCOM) if any unwanted content is found
- Safety duties protecting children
    - Mitigate impact of harm when children encounter harmful content
    - Prevent children from encountering harmful primary content
    - Do so by implementing a way of age verification or age estimation
    - If a service is utilizing age estimation, it must be "highly effective" at correctly determining whether a user is a child or not.
- Actively take measure to prevent encounters of such harmful content
    - Minimize time frame illegal content is reachable
    - Swiftly take down said content and blocking users
    - Include policies for staff, use of service and moderation
    - Include provisions in terms of service of how users are protected from illegal content
    - Design service in such a way that the points above are facilitated
    - Include information about countermeasures that are applied of compliance with duties in the act

Given that this act mainly imposes duties on the services which affect UK citizens, extra care should be taken if the Software Development Kit (SDK) is deployed by providers of services situated in the UK or targeted towards British people. A thorough compliance review of the service and its functionality should be completed if the service falls under the scope as a part 3 service. AIBA as a company does not by definition fall under the scope of a part 3 service nonetheless. This means that the duties imposed by this act does not directly affect their work. Rather, the existing solution AIBA provides in addition to the SDK could be used as a way of complying with the OSA. In a way, the OSA yields more clients for providers such as AIBA, considering navigating the duties sanctioned by this Act could be troublesome.

Regulated services are also required to improve and minimize harmful content shown to children through active use of age validation or age estimation. Given that some age validation methods are less effective and more susceptible to manipulation, the ability to provide effective age estimation functionality by capturing SBKD data and using AI analysis algorithms could be a key component in meeting the obligations. Such an age estimation service must be considered effective enough to comply with the conditions proposed by the Act. The SDK that the Group is developing could therefore make a positive difference and enable more companies to comply.

The SDK is neither the full solution nor the only way to comply to these duties.

It is not realistic for the SDK to disable certain accounts from viewing content in real-time. Comparatively, it will be one of many tools that should be acquired to form a full compliance package.

One consideration worth mentioning, however, is the fact that the user must be notified on what sort of technology is implemented in the service. Because of AIBA's solution regarding age estimation capabilities, this will have to be included in the terms of service. Going forward, the group must remember to include the fact that use of the SDK should be included in the terms of service. This act should also be examined further if such an opportunity to work with regulated services arise.

6.3 Kids Online Safety Act (KOSA)

The US Kids Online Safety Act (KOSA) is a legislation proposed in 2022 in the USA that established guidelines that aim to protect minors from harm on social media platforms. In this legislation "minors" refers to individuals under the age of 13. In the proposition, Internet Service Providers (ISP), email services and educational institutions are exempt from the guidelines. Below is a quote from KOSA reported in the United States senate (13.12.23) which can be found in *point A of section 3* "*Duty of care*" [6]

> "*Covered platforms must take reasonable measures in the design and operation of products or services used by minors to prevent and mitigate certain harms that may arise from that use (e.g., sexual exploitation and online bullying)*."

It is not clear from reading the report what "reasonable measures" entails, and since this is currently only a proposed bill we can not conclude anything about these measures.

The proposition in it's current form does not seem to directly impact neither AIBA's business nor our SDK as the bill only targets data collected for advertising purposes. The bill is also more directed at preventing harm to children through limiting or preventing interaction between users, and to limit information disclosure about minors to other users.

In addition to what is stated above, the bill proposes that there should be conducted independent research on the harms of using social media for minors, and ways to age verify minors at the device or OS level [7,8]. With the increased focus on age verification of minors, it may be an opportunity for AIBA to further expand their market using SBKD. Since KOSA is only a proposition, all of this is subject to change, and any further impact analysis would be guesswork with a low probability of yielding meaningful results.

---

[6] `https://www.congress.gov/bill/118th-congress/senate-bill/1409/text` - Fetched 10.02

[7] `https://www.congress.gov/bill/118th-congress/senate-bill/1409/text#id856540475ea648148031fa3374b7f9f6` - Fetched 10.02

[8] `https://www.congress.gov/bill/118th-congress/senate-bill/1409/text#idcf4f5fdf-6bd7-46c1-ab08-b3a4993ef449` - Fetched 10.02

6.4 EU AI Act

While our project does not directly incorporate Artificial Intelligence (AI) components, the end product is designed to interface with AI. Specifically, it aims to profile users based on age and gender to identify high-risk conversations that may involve predatory behavior. Given this context, we find it important to examine how the EU Artificial Intelligence Act (EU AI Act) would influence further development of our SDK upon completion of the project. This section focuses primarily on categorizing which of the risk categories in the EU AI Act the use of Soft Biometric Keystroke Dynamics with AI for age and gender detection would place in. To our knowledge, this analysis may be among the first to clarify how this use of AI will be classified under the EU AI Act and outline the corresponding regulatory obligations as of April 16, 2024.

The EU AI Act is presently under development, and expected to become the worlds first AI Act[9]. A provisional agreement was reached on 12 December 2023 and subsequently published on 26 January 2024, outlining the European Union (EU) legislative objectives[10]. The EU published a final draft of the EU AI Act on 13 March 2024, followed by a corrigendum [21] on 16 April 2024. All references and citations to the EU AI Act in this document are based on this latest corrigendum. Following the enactment of the EU AI Act, entities using AI will have two years to comply with its obligations.

6.4.1 Applicability of the EU AI Act

First, it is necessary to determine whether the EU Artificial Intelligence Act applies to AIBA's end product. Article 2 of the EU AI Act defines the scope of the legislation. Specifically, Paragraph 1(a) of article 2 states:

> "*providers placing on the market or putting into service AI systems or placing on the market general-purpose AI models in the Union, irrespective of whether those providers are established or located within the Union or in a third country;*"

Given that AIBA qualifies as a 'provider' under the definition stipulated in point (3) of article 3 :

> "*provider' means a natural or legal person, public authority, agency or other body that develops an AI system or a general-purpose AI model or that has an AI system or a general-purpose AI model developed and places it on the market or puts the AI system into service under its own name or trademark, whether for payment or free of charge;*"

it is clear that if AIBA were to develop and release software utilizing AI to profile users based on their Soft Biometric Keystroke Dynamics, such software would

[9] `https://www.wiley.law/alert-EU-Adopts-the-AI-Act-The-Worlds-First-Comprehensive-AI-Regulation` - Fetched 17.05

[10] `https://data.consilium.europa.eu/doc/document/ST-5662-2024-INIT/en/pdf` - Fetched 28.04

indeed fall under the jurisdiction of the EU AI Act, should it operate within the EU.

6.4.2 Risk categories

On the topic of *how* the EU Artificial Intelligence Act will influence AIBA and which obligations apply, we need to examine the risk classifications established by the legislation. The EU AI Act will classify Artificial Intelligence systems according to their risk where minimal risk/general-purpose AI models, limited risk, high risk, and unacceptable risk are the established risk categories. Each of AIBA's AI-driven components carries regulatory obligations, influencing the steps which AIBA needs to take to become compliant with the EU Artificial Intelligence Act

6.4.2.1 Minimal risk and general-purpose AI models

The minimal risk/general-purpose AI category represents the most prevalent uses of AI. The minimal risk category will comprise of AI systems such as AI-enabled video games and spam filters. The term 'general-purpose AI' is defined in point (66) of Article 3 of the legislation as

> "*'general-purpose AI model' means an AI model, including where such an AI model is trained with a large amount of data using self-supervision at scale, that displays significant generality and is capable of competently performing a wide range of distinct tasks regardless of the way the model is placed on the market and that can be integrated into a variety of downstream systems or applications, except AI models that are used for research, development or prototyping activities before they are placed on the market;*"

This category encompasses applications such as Dall-E, ChatGPT, Gemini, and other chatbots and APIs. Given that our project does not align with this category, it is of limited relevance to this report

6.4.2.2 Unacceptable risk category

In the initial press releases concerning the EU AI Act, it was suggested that AI-based products, such as AIBA's, might be classified within the unacceptable risk category[11]. However, subsequent press releases and the final draft of the Act have narrowed the criteria for this category as the legislation neared completion. Article 5 of EU AI Act now specifies that biometric AI applications deemed unacceptable primarily include real-time biometric profiling of individuals in public spaces. This narrowing of criteria makes it so that the unacceptable risk category is also of little interest in this report as the end product will not operate in public spaces, but rather within game chats, where game developers can disclose the use of such biometric profiling in their terms of service.

[11] `https://www.europarl.europa.eu/topics/en/article/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence` - Fetched 14.02

6.4.3 Determining High-Risk Status

Having determined that the minimal and unacceptable risk categories are not applicable to our project, we can now focus on the remaining categories: limited and high risk. Paragraph 2 of Article 6 states that AI systems referred to in Annex III shall be considered High risk. In Annex III, there are two uses of biometrics which is relevant to our case, paragraph 1(a) and 1(b).

> "*High-risk AI systems pursuant to Article 6(2) are the AI systems listed in any of the following areas:*
>
> *1. Biometrics, insofar as their use is permitted under relevant Union or national law:*
>
> *(a) remote biometric identification systems.*
>
> *This shall not include AI systems intended to be used for biometric verification the sole purpose of which is to confirm that a specific natural person is the person he or she claims to be;*
>
> *(b) AI systems intended to be used for biometric categorisation, according to sensitive or protected attributes or characteristics based on the inference of those attributes or characteristics;*"

6.4.3.1 Evaluating Remote Biometric Identification

The EU AI Act defines "remote biometric identification systems" in Article 3, point (41), as:

> "*'remote biometric identification system' means an AI system for the purpose of identifying natural persons, without their active involvement, typically at a distance through the comparison of a person's biometric data with the biometric data contained in a reference database*"

Given that the program involves users interacting in game chats, where they type messages without actively participating in any biometric data provision, there is an argument that the program could be classified under Annex III paragraph 1(a). However, Article 3 point 35 clarifies that 'biometric identifications' involve the *identification of individuals* based on biometric data compared against previously stored data in a database. This definition supports the requirement for a reference database, indicating that 'biometric identification' pertains to identifying specific individuals, rather than extracting limited information about an unknown person.

The application in question, which employs AI to predict an individual's age and gender, does not identify specific individuals. This suggests that using Soft Biometric Keystroke Dynamics (SBKD) to predict users age and gender does not fall under the specified high-risk category of Annex III, paragraph 1(a). However, Recital 54 provides a broader context for consideration:

> "*As biometric data constitutes a special category of personal data, it is appropriate to classify as high-risk several critical-use cases of biometric*

> *systems, insofar as their use is permitted under relevant Union and national law. Technical inaccuracies of AI systems intended for the remote biometric identification of natural persons can lead to biased results and entail discriminatory effects. The risk of such biased results and discriminatory effects is particularly relevant with regard to age, ethnicity, race, sex or disabilities. Remote biometric identification systems should therefore be classified as high-risk in view of the risks that they pose.*"

Recital 54 specifies that the use of Artificial Intelligence to predict age is regarded as high-risk due to the potential for technical inaccuracies and the possibility of discriminatory outcomes due to potential age bias from moderators. Although the use of SBKD for age detection may not precisely meet the criteria for high risk systems set forth in Annex III, the explicit language in Recital 54 indicates that the legislative intent is to categorize such uses of SBKD as high-risk, reflecting concerns about potential bias and discrimination.

6.4.3.2 Biometric Categorization

While we have determined that the end product qualifies as high-risk under Annex III paragraph 1(a) of the EU AI Act, it is wise to also explore paragraph 1(b) as this exploration could provide (valuable) insight into the regulatory stance on the use of SBKD for age and gender prediction. Annex III paragraph 1(b) refers to the use of biometric categorisation which is defined in Article 3 point (40):

> "*'biometric categorisation system' means an AI system for the purpose of assigning natural persons to specific categories on the basis of their biometric data, unless it is ancillary to another commercial service and strictly necessary for objective technical reasons*"

As the program categorises individuals into specific age groups and genders, its use case falls clearly within the definition of 'biometric categorisation'. The data is not ancillary to another commercial service, but is used by AIBA as a core feature. The next consideration is then whether the categorisation parameters are sensitive or protected characteristics as defined by the EU AI Act.

The EU AI Act refers to Article 9(1) of Regulation (EU) 2016/679 for the definition of sensitive or protected attributes:

> "*1. Processing of personal data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, or trade union membership, and the processing of genetic data, biometric data for the purpose of uniquely identifying a natural person, data concerning health or data concerning a natural person's sex life or sexual orientation shall be prohibited.*"

The relevant question here is whether age or gender qualifies as "data concerning health". Article 4(15) of Regulation (EU) 2016/679 defines "data concerning health" as:

> "*'data concerning health' means personal data related to the physical or mental health of a natural person, including the provision of health care services, which reveal information about his or her health status;*"

While there is a lack of definition for physical health in any of the EU regulations, The European Patients' Academy on Therapeutic Innovation (EUPATI) defines physical health as

> "*Physical health is defined as the condition of your body, taking into consideration everything from the absence of disease to fitness level.*"[12]

Therefore, neither age nor gender should be categorized as data concerning health under EU regulations. This indicates that although the application appears to satisfy the criteria for a high-risk system as outlined in paragraph 1(a) of Annex III, it avoids the criteria in paragraph 1(b). Since the age and gender data output by the application is not categorized as protected or sensitive data, it is not covered by paragraph 1(b).

6.4.3.3 Exceptions to Annex III

Having established that the program qualifies as a high-risk system under Annex III, as specified in Recital 54, we proceed to examine potential exceptions outlined in Article 6, paragraph 3 of the EU Artificial Intelligence Act. The complete text of Article 6 paragraph 3 is as follows:

> "*By derogation from paragraph 2, an AI system shall not be considered to be high-risk if it does not pose a significant risk of harm to the health, safety or fundamental rights of natural persons, including by not materially influencing the outcome of decision making. This shall be the case where one or more of the following conditions are fulfilled:*
>
> *(a) the AI system is intended to perform a narrow procedural task;*
>
> *(b) the AI system is intended to improve the result of a previously completed human activity;*
>
> *(c) the AI system is intended to detect decision-making patterns or deviations from prior decision-making patterns and is not meant to replace or influence the previously completed human assessment, without proper human review; or*
>
> *(d) the AI system is intended to perform a preparatory task to an assessment relevant for the purposes of the use cases listed in Annex III.*
>
> *Notwithstanding the first subparagraph, an AI system referred to in Annex III shall always be considered to be high-risk where the AI system performs profiling of natural persons.*"

[12] `https://toolbox.eupati.eu/glossary/physical-health/` is licensed under CC BY-NC-SA 4.0 - fetched 30.04.2024

The paragraph commences with a general exclusion criteria, stipulating that AI systems which do not pose a significant risk to a natural persons health, safety, fundamental, or materially influence decision-making outcomes, may be considered for exemption. Following this, it lists specific conditions under which an AI system could be excluded from being categorised as high-risk. In regards to paragraph 3(a), the EU AI Act recital 53 gives the following examples of procedural tasks:

> "*AI system that transforms unstructured data into structured data, an AI system that classifies incoming documents into categories or an AI system that is used to detect duplicates among a large number of applications.*"

Although the estimation of users' ages and genders is more complex than the examples presented, it is fundamentally similar to the task of classifying documents. The primary function of the end product is to classify users by their estimated age and gender, providing these classifications to moderators who may then take further action based on the users' chat history. Since the application itself does not utilise this information post-classification, it can be categorised as a procedural task.

With regard to point (b), the primary objective of the program is to optimise the allocation of moderators' resources, thereby reducing the number of moderators required in conversations where it is evident that both participants are in the same age group, either adult or minor. This optimisation allows for increased focus and resources directed towards conversations where age discrepancies might pose a risk of predatory behavior occurring. Thus, the program clearly meets the stipulations of condition (b).

For point (c), according to Recital 53 of the EU AI Act, the essence of this condition is to identify anomalies within a dataset that warrant further review. The legislation uses the following example:

> "*AI systems include for instance those that, given a certain grading pattern of a teacher, can be used to check ex post whether the teacher may have deviated from the grading pattern so as to flag potential inconsistencies or anomalies!*"

Although AIBA's application may not appear to be directly analogous, a closer examination reveals that it does align with this condition. The application processes the self-reported ages of users and utilises AI to detect deviations from these ages based on the users' Soft Biometric Keystroke Dynamics. Such anomalies, once flagged, can be scrutinised further by human moderators. Thus, the program also meets the criteria outlined in point (c).

With regard to point (d), the program performs an important preparatory task in assessing users' ages, which moderators subsequently evaluate in greater detail. This preparatory assessment is of great importance in ensuring that moderators can effectively prioritise their review processes, particularly in scenarios where age-related discrepancies are detected. Consequently, the application fulfils the

stipulations of condition (d), as it aids in preparing assessments that are vital for the purposes outlined in Annex III of the EU AI Act.

Upon examination of the conditions set forth in Article 6, Section 3, it becomes evident that the application in question fulfills not only one of the exceptions, but all of them. However, there remains one final sentence to be considered within Article 6, paragraph 3, which states that all AI systems referred to in Annex III shall be considered high-risk if the system performs profiling of a natural person. In order to ascertain whether SBKD for age and gender detection constitutes profiling of natural persons under the EU AI Act, it is necessary to examine the definition of profiling set out in the regulation. The definition of profiling given in point (52) of Article 3 in the regulation is

> "*profiling' means profiling as defined in Article 4, point (4), of Regulation (EU) 2016/679;*"

Additionally, recital 53, which addresses Article 6 section 4, states:

> "*profiling within the meaning of Article 4, point (4) of Regulation (EU) 2016/679 or Article 3, point (4) of Directive (EU) 2016/680 or Article 3, point (5) of Regulation (EU) 2018/1725.*"

All three referenced regulations consistently define profiling as:

> "*'profiling' means any form of automated processing of personal data consisting of the use of personal data to evaluate certain personal aspects relating to a natural person, in particular to analyse or predict aspects concerning that natural person's performance at work, economic situation, health, personal preferences, interests, reliability, behaviour, location or movements;*"[22][23][24]

The definition of personal data under these regulations is also consistent:

> "*'personal data' means any information relating to an identified or identifiable natural person ('data subject');*"

Given these definitions, the application's use of SBKD to predict age and gender can be considered profiling, given that it involves the automated processing of personal data to evaluate specific personal aspects. Consequently, this profiling categorizes the use of SBKD for age and gender prediction as high-risk under the EU AI Act, aligning it with the stringent regulatory requirements intended to manage the risks associated with such AI systems.

6.4.4 Obligations of high risk systems

Given that we have now categorised the application as high-risk, it seems prudent to shortly examine the specific obligations that AIBA would be subject to under this classification.

The EU AI Act delineates two distinct categories of obligations for entities like AIBA operating within the high-risk framework. The first set comprises general obligations applicable to all high-risk systems, as outlined in Chapter III, Section II of the EU AI Act. The second set of obligations relates specifically to providers of high-risk systems, detailed in Article 16 of the Act. both sets of obligations will, according to article 113, become effective 24 months after the EU AI Act has been enacted.

6.4.4.1 General obligations for high risk systems

The following list enumerates the general obligations for high-risk systems as derived from the EU AI Act Compliance Checker, accessed through *https://artificialintelligenceact.eu*[13]:

- Establish and implement risk management processes according to Article 9
- Use high-quality training, validation and testing data according to Article 10.
- Establish documentation and design logging features according to Article 11 and Article 12.
- Ensure an appropriate level of transparency and provide information to users according to Article 13.
- Ensure human oversight measures are built into the system and/or implemented by users according to Article 14.
- Ensure robustness, accuracy and cybersecurity according to Article 15.
- Set up a quality management system according to Article 17.

These obligations collectively aim to mitigate risks associated with the deployment of high-risk AI systems, ensuring they are safe, secure, and transparent in their operations.

6.4.4.2 Obligations for Providers of high risk systems

In conjunction with the general obligations for high-risk systems, AIBA, as a provider of an AI system, must adhere to supplementary obligations outlined primarily in Article 16. This article serves as a comprehensive list detailing additional responsibilities for providers. It includes references to relevant articles that providers must comply with and outlines obligations to meet accessibility standards as set forth in Directives (EU) 2016/2102[25] and (EU) 2019/882[26].

Moreover, there are more stringent obligations enumerated in Section 2 that AIBA is required to follow. These pertain specifically to Articles 9, 10, 13, and 14, which cover the aspects of risk management, data quality, transparency, and human oversight, respectively.

[13] https://artificialintelligenceact.eu/assessment/eu-ai-act-compliance-checker/ - Accessed: 29.04.2024

6.4.5 Final thoughts on the EU AI Act

In this section, we have delineated the risk category under which AIBA's application of SBKD for age and gender prediction falls within the EU AI Act, and discussed the associated regulatory implications. We determined that the application is classified as high-risk. Furthermore, we have identified the relevant sections of the EU AI Act that address the obligations of being a provider of high-risk AI systems. While a more detailed exploration of these obligations is possible, the primary aim of this report section was to ascertain the classification of SBKD for age and gender prediction under the EU AI Act. This objective has been satisfactorily achieved. Delving further into the specific obligations would be beneficial but also exceed the scope of this particular inquiry.

7 Technical Design and Implementation

This chapter outlines the technical design and implementation of the project. It begins with an overview of the overall structure, defining functional layers for the SDK components. Next, it shows how we broke down the main problem into manageable sub-problems, referred to as stages, and describes the iterative approach used to gradually increase complexity as the tools became familiar. The subsequent sections provide a detailed account of the implementation of each stage, along with a mention of the necessity for a DSA notification if the SDK is to be deployed to the Unity Asset Store.

7.1 Technical design

This section describes the overall system architecture, its different components and their interoperability in how they enable diverse subsystems to function as a cohesive unit.

7.1.1 Overall structure

The SDK provides a separate data stream that can send keystroke dynamics data to AIBA's API. The SDK we have developed will operate between a *User* and the *Chatroom server*, extracting keystrokes as portrayed in Figure 7.1.

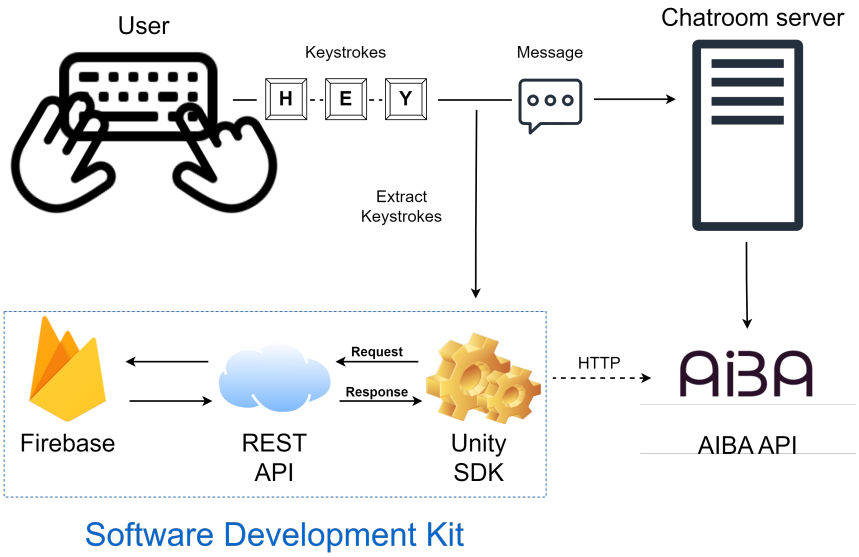


Figure 7.1: System Architecture

We have attempted to follow the layered architecture pattern for software development, dividing components that provide similar functionality into func-

tional layers providing their own purpose[1]. There are no rules as to how many layers a team can define, but generally a set of predefined layers are implemented: presentation, business, persistence, and database (Figure 7.2).

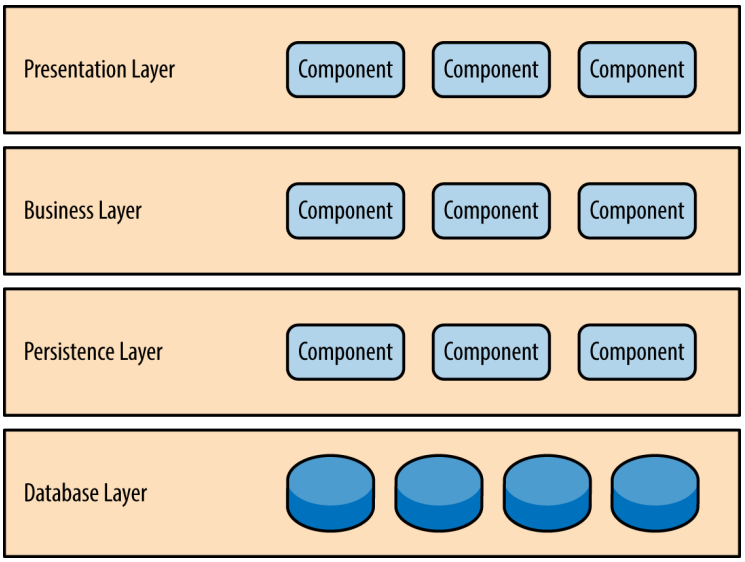


Figure 7.2: Layered architecture pattern [1]

The presentation layer is normally the component that a user can interact with, the business layer consists of all logic and most functionality of the service while the persistence and database layer encompass the means of storing retrievable information in a service[2].

7.1.2 Chat service layer - Presentation

Before sending a message to a chat room server, the user types a message in a chat field using a keyboard. This action is shown as the horizontal line going from *user* to *Chatroom server* (Figure 7.1). Depending on the chat service, this is usually done through transferring data packets between users. They may contain information such as the message string itself, time information and other metadata are sent in the process. The chat service we have developed supports concurrent users talking in channels using Vivox (7.7.2).

7.1.3 Keystroke Dynamics Extraction layer - Business

The lifecycle of a continuous authentication system using Soft Biometric Keystroke Dynamics (SBKD) includes the phases as seen in Figure 7.3 [27]. The SDK lives in the two first phases of the lifecycle; collecting SBKD data in a chat window and extracting features by calculating them on the client-side.

[1] `https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html` - Fetched 18.05

[2] `https://cs.uwaterloo.ca/~m2nagapp/courses/CS446/1195/Arch_Design_Activity/Layered.pdf` - Fetched 18.05

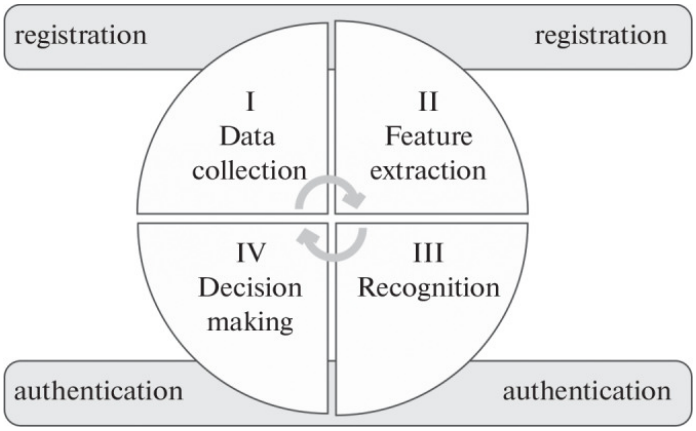


Figure 7.3: SBKD continuous authentication lifecycle (figure from [27])

Although this lifecycle illustrates the process of **continuous authentication** using SBKD without considering biometric applications such as those facilitated by the SDK, the data collection and feature extraction phases (I and II) remain unchanged, while stages III and IV differ.

Not noticeable to the users of the chat service, the SDK will synchronously extract keystroke dynamics such as their typing rhythms, patterns and key-codes given as input while a user types. The features calculated by the SDK are detailed in Figure 7.4. Having Latency and Duration, one has enough information to be able to calculate the rest of the other attributes as mentioned in Figure 4.1.

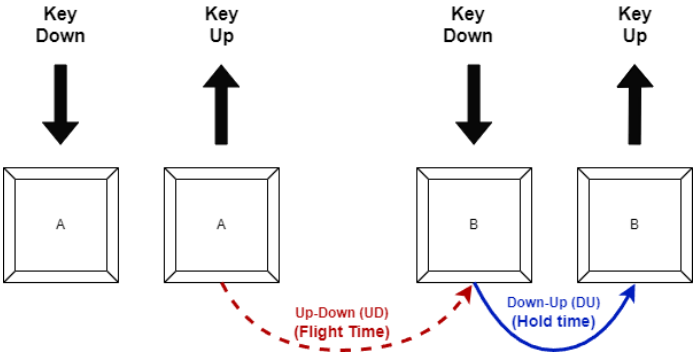


Figure 7.4: SDK computed SBKD metrics (inspired by figure in [13])

When a user presses *Enter* or sends the message, the extraction process will stop. The information gathered during this process is saved in short-term memory in non-persistent Keystroke and Data classes (Figure 7.9).

7.1.4 Data transfer layer - Business

Not having access to AIBA's API, meant the data would not be transferred directly to the client. It was rather transmitted to the API we have developed ourselves

(Section 7.6.2). With a few tweaks such as changing the IP which the Unity SDK transmits data to, it is able to transmit data to the company's API without trouble, as further described in Section 8.1 detailing deployment of SDK on the employers system.

The *APIHandler* and *Gzipper* libraries in the Unity SDK handle transmission of data and compression from Unity SDK to the REST API (Section 7.6.1).

Due to the results from the data transfer research we completed in Section 4.2, gzip compression was something we implemented through a wrapper library using `System.IO.Compression` from the C#s standard library.

The REST API receives compressed JSON data from the *APIHandler*. On the server-side, the endpoint handler receives the compressed data, reads the gzipped request with the gzip package by calling `gzip.NewReader()`[3], and finally decode the JSON struct.

7.1.5 Firebase - Persistence/Database layer

The database is crucial in a layered architecture, as this is where all the data is stored. When a valid POST-request is sent to the endpoint from the REST API, the data will be saved in the database. This is done through a Request-response model using HTTP (Figure 7.11).

7.2 The Three Feasibility Stages

In consultations with AIBA, it was determined that capturing keystrokes in Unity had not been verified prior to defining the project. This situation required an alternative approach to the task, considering the lack of previous experience with Unity and C#. To address this challenge, we developed what was labeled "The Three Feasibility Stages." These stages consisted of three progressively complex prototypes, which were constructed sequentially over the span of the project. This structured approach was important when mitigating the initial uncertainties present when starting development and for facilitating systematic progress during the project.

One of the reason for dividing development into three stages was to learn concepts through prototypes.

- Stage 1 aided in learning C# syntax
- Stage 2 gave insight into developing in Unity, a prior unknown environment for the group
- Stage 3 for integrating key capture in Vivox

Additionally, through stages 2 and 3 the opportunity to explore Unity's two UI systems, uGUI[4] and UI toolkit[5], arised (7.5).

The first prototype was a console chat application in pure C# that captured user keystrokes in a terminal (7.3). Although the application itself was not in-

[3] `https://pkg.go.dev/compress/gzip` - Fetched 18.05

[4] `https://docs.unity3d.com/Manual/com.unity.ugui.html` - Fetched 16.04

[5] `https://docs.unity3d.com/Manual/UIElements.html` - Fetched 16.04

cluded in the SDK, it became a way to learn basic C# syntax, naming conventions for variables and representing data using the JSONSerializer class. Similarly, C# code regarding keystroke capture and the aforementioned C# language understanding was further developed and utilized in the next stages of implementation.

Secondly, a Unity chat window UI with keystroke capture embedded in it was developed (7.5). In this prototype, attaching a key logger script onto a UI text field and sending the SBKD data required for profiling (4.1) to an API was accomplished. From this implementation we had a Unity prefabs consisting of a keylogger and an HTTP handler we could use for the next stage.

The previous iteration solves the task depicted in the task description by enabling extraction of SBKD from a Unity game chat (Chapter 11). However, when learning about Unity Gaming Services and how it provides cloud services similar to that of Google Cloud Platform, the next stage consisted of enabling SDK support for another chat service. The chat service chosen by the group was Vivox [6] (7.7 - Fetched 16.04). The final iteration acts as a middle layer between user keyboard interaction and the Vivox chat service, extracting SBKD data when a user is typing in the chat window.

7.3 Console application

The console application is a program made in the C# programming language. Our main goal when planning and implementing it was to familiarize ourselves with C#, VSCode and Rider. The main goal of the program is to register which key is pressed and when that key press occurred relative to when the program started running. This section presents both a textual description and a sequence diagram (see Figure 7.6) that illustrate the flow of operations in the console application.

7.3.1 Data classes

This functionality was achieved by creating the three classes which is shown in Figure 7.5:

The Keystroke classrepresents a keystroke and contains properties for which key is pressed and the time of the key press in relation to when the session started.

When a user has submitted a message in the console the keystroke data is prepared for transmission. This is done by organizing the the data in the **Data** class. This class has properties for the username of the message sender, a list of keystrokes, the full message and the operating system in which the program is running.

These classes are structuring classes or data classes that move some of the responsibility away from the main class, the **Program** class. This class contains a Main-function that is called on program start. It will prompt the user to enter some text, then start capturing keystrokes by declaring and initializing a list of keystrokes as the return value of the `CaptureKeystrokes()` method.

[6] `https://unity.com/products/vivox-text-chat`

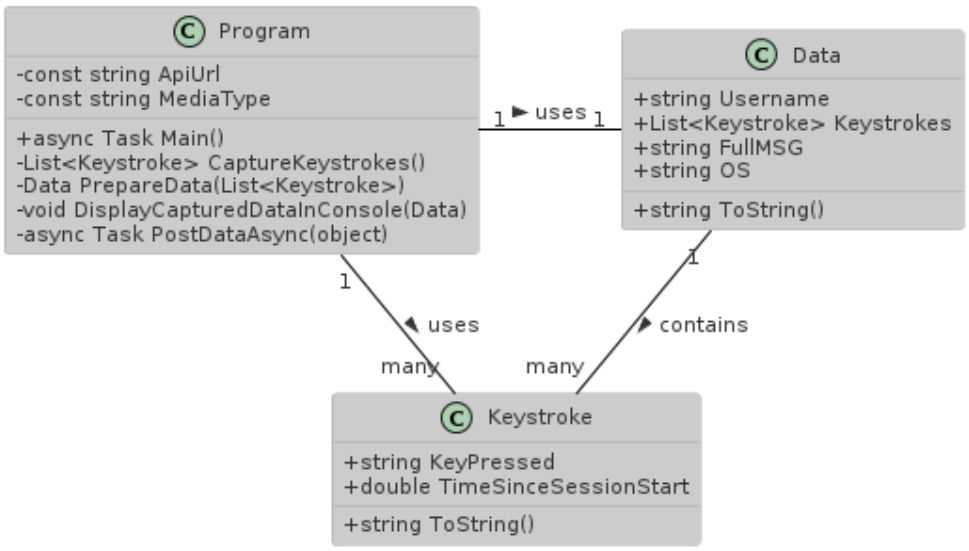


Figure 7.5: Class diagram for the console application

7.3.2 Program class

When the method for capturing keystrokes is called it will start a stopwatch, initialize an empty list of keystrokes, then start a `while` loop that runs until the enter key is pressed. The loop awaits a key input, when a key press occurs, the elapsed time is saved in a variable. A new instance of the Keystroke class is created and the key and elapsed time is passed as arguments to its constructor and added to the list of keystrokes. When the enter key is pressed, the loop stops together with the stopwatch and the list of keystrokes is returned which completes the initialization of the keystroke variable in `Main()` method.

Once the list of keystrokes is initialized we pass it as an argument to the `PrepareData(List<Keystrokes>)` method. In this method an instance of the Data class gets constructed by setting the username to a dummy value. The full message from the console input is concatenated using a Language Integrated Query (LINQ) expression with the `string.join` method to select each character from the list of keystrokes.

To get the OS version we use the `Environment` class of the .NET Framework which contains an OSVersion property which receives an OperatingSystem object which in turn contains details such as a platform identifier and a version number which we deemed good enough for this Proof of concept (POC) application[7].

Once the data is returned from the preparation function it is passed to a `PostDataAsync(Data)` method. This method serializes the Data object into JSON format, sets the encoding to Unicode Transformation Format – 8-bit (UTF-8) and sets the mediatype to "application/json" which is defined as a constant property of

[7] `https://learn.microsoft.com/en-us/dotnet/api/system.environment.osversion?view=net-8.0` - Fetched 23.04

the Program class. The data is then passed to a `PostAsync(Uri, StringContent)` function of the HttpClient .NET class and either a success or failure message is sent to the output stream of the console.

A method to display the prepared data to the console output stream in proper JSON format was also created, this was used for getting data to test the API in the early stage before the POST method was implemented. It can be found in the git repository.

In Figure 7.6 we have included a diagram showing the sequence of the console application. It covers the flow from starting the program, capturing the keystroke dynamics data, preparing the data to be sent and then posting the data to the API.

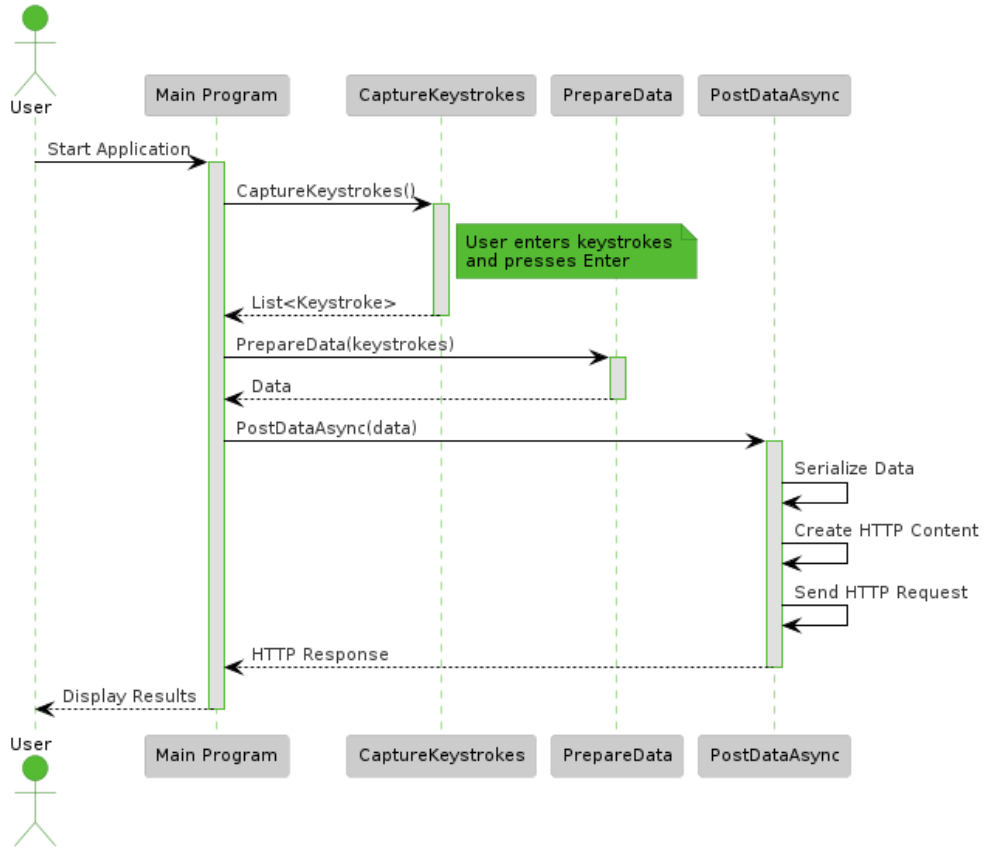


Figure 7.6: Sequence diagram for the console application

7.4 Unity development

In Unity, the development of an application revolves around the concept of a *scene*, which acts as a container for all elements within the application[8]. When creating a new scene it includes a *camera* and depending on the version of Unity it may

[8] `https://docs.unity3d.com/Manual/CreatingScenes.html` - Fetched 01.05

also include a *light* source. The camera will render the visuals that a user sees, while the light is used to illuminate the elements within the scene.

The core architecture of Unity is built around GameObjects[9], which purpose is to act as containers for various functionalities though added components. The previously mentioned camera and light are such GameObjects. Unity has many pre-made objects with commonly used functionality, like objects for handling audio, physics and user interface elements. In our project we primarily utilized the camera and UI objects as the others were not relevant for our purpose.

We used the Unity Editor for Unity specific development. It can be thought of as an IDE for game development. It's interface is largely drag and drop focused, but does also provide scripting functionality directly in the editor. In our project we did not use the built-in scripting functionality as Unity supports integration with external development environments for writing code and we deemed Jetbrains Rider a better fit for the task. The Unity editor also has a "Profiler" for viewing performance metrics such as CPU, GPU and memory usage, rendering stats and an overview of network operations which helped in detecting potential problems at an early stage. There are also dedicated debuggers for physics, network, animations and the one utilized most frequently in our case, the UI Toolkit debugger.

In Figure 7.7 we provide a screenshot of the editor displaying some of the commonly used windows when using Unity. In the upper left corner section A of the screen there is the "Hierarchy", which displays the currently opened scene and all the GameObjects of that scene. The middle part of the screen, section B, has the "Game" window which displays a real-time preview of the currently active scene. This is how the scene appears in the built version of the application. To the right of the screen, section C, there is an "Inspector". When a GameObject is selected, all of it's components are shown in this window, and provides a way to modify or add new behaviors and functionality. At the bottom in section D is the "Project" window which provides an overview of the directory structure of the project and an area to display the files contained in a selected directory. Docked in the same window as the Project window in section D, two other commonly used tabs are also shown, the "Console" and the "Unity Version Control" which shows their respective windows.

7.5 Unity Chat Application

The Unity Chat Application was created because there was a need to get familiar with Unity development in general. In the early phases of learning Unity, a YouTube tutorial on how to make a simple game in Unity was completed by each group member[10]. This was done to make sure everyone was acquainted with the Unity Editor and to eliminate some of the challenges that might get encountered early on. Each member was also assigned the task of exploring and getting an

[9] `https://docs.unity3d.com/Manual/GameObjects.html` - Fetched 01.05

[10] `https://www.youtube.com/watch?v=XtQMytORBmM` - Fetched 01.05

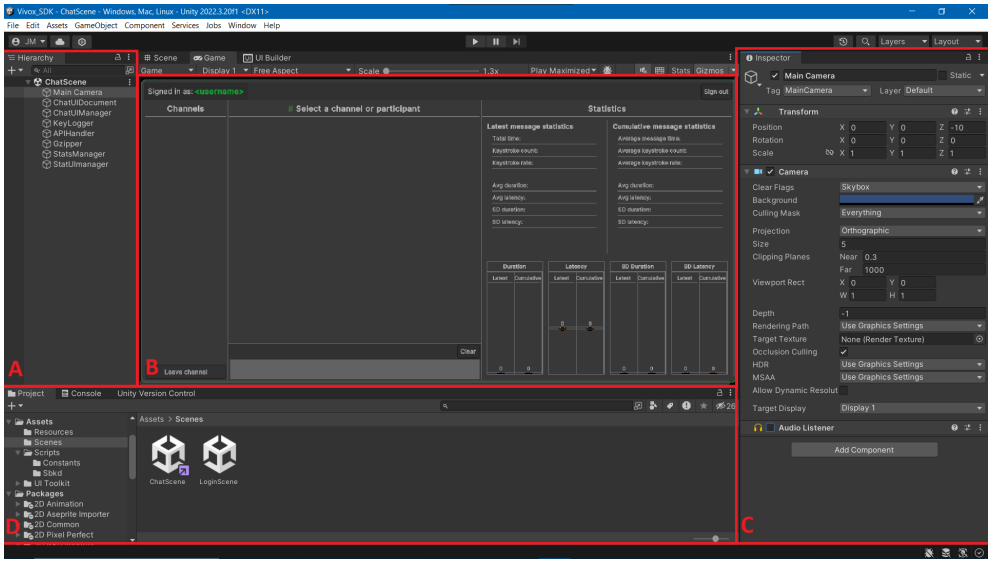


Figure 7.7: An example of how the Unity Editor looks

overview of the Unity Documentation pages[11].

When the development started on the chat application the Unity User Interface package (uGUI) was utilized and we built the complete UI using this. The UI framework is based on creating a hierarchy of GameObjects containing components for handling the arrangement, positioning, styling and behavior of the interface. Getting the layout to behave as expected proved troublesome because of some layout components in uGUI that was not performing as specified in the documentation. When looking for solutions in the Unity developer forums we came across many posts suggesting UI Toolkit as an alternative to uGUI. In Unity docs there is a page comparing the different UI systems[12] and since Unity recommends using UI Toolkit for new UI development projects the group decided to rebuild the UI using the UI Toolkit.

7.5.1 UI Toolkit

The UI Toolkit, formerly known as UI Elements, is a framework that can be used for creating user interfaces in Unity. As stated in the Unity Manual it is the recommended tool when creating new UI development projects and is intended to replace the older uGUI and IMGUI systems[13]. The UI Toolkit has been in development ever since Unity version 2017.1[14] which was released in July 2017, but is still a work in progress and has not yet attained the same level of maturity as the other UI frameworks.

[11] `https://docs.unity.com/` - Fetched 10.05

[12] `https://docs.unity3d.com/Manual/UI-system-compare.html` - Fetched 11.05

[13] `https://docs.unity3d.com/Manual/UIElements.html` - Fetched 28.04

[14] `https://docs.unity3d.com/2017.1/Documentation/Manual/UIElements.html` - Fetched 28.04

UI Toolkit is similar to common web technologies like HyperText Markup Language (HTML) and Cascading Style Sheets (CSS) with their analogs Unity Extensible Markup Language (UXML) and Unity Style Sheet (USS). The UXML is used to structure an interface and USS is used to style the elements. In Code listing 7.1 is an example of how a simple login form would look in UXML.

Code listing 7.1: UXML login form example

```
<?xml version="1.0" encoding="utf-8"?>
<UXML>
    <Box class="login-form">
        <Label text="Username:"/>
        <TextField name="username" />
        <Label text="Password:"/>
        <TextField name="password" type="password" />
        <Button name="submit" text="Login" />
    </Box>
</UXML>
```

The *Box*, *Label*, *TextField* and *Button* defined in the UXML above are all types of VisualElements, which is the most basic object in UI Toolkit and UXML. Since they all inherit from the VisualElement base class they are all able to be layed out, styled and interacted with. A VisualTree is an object graph that defines the UI and contains all the VisualElements of the window or panel. VisualElements themselves can have parents and children.

In the Unity Editor there is a UI Builder which provides a user with a drag and drop interface for building UIs with UI Toolkit. It has a panel for stylesheets where it's possible to define selectors for applying styles to the elements. Below the stylesheet panel there is a hierarchy view of the VisualTree which displays all the VisualElements and their children. If an element is selected, it's attributes and styles will be visible and modifiable in the inspector panel to the right of the screen. There are only a selection of the USS styles that are modifiable through the inspector. There is also a *Library* panel visible in the UI Builder that contains *controls*.

Controls are pre-made VisualElements with functionality, behaviour and styles defined by default. Some examples include: ListView, Button, Label, Dropdown and Toggle. With these pre-made elements only some of the functionality and styles are modifiable directly in the UI Builder, it is however possible to directly modify the applied USS styles either through creating a separate USS file and defining a selector for the element, or through accessing the element in C# code and setting it there. Sufficient documentation on specific VisualElements can be scarce or non-existent, but it's often possible to find the desired information in the Unity forum[15]. Creating custom controls is also an option.

[15] `https://forum.unity.com/`- Fetched 05.05

7.5.2 Layout

The UI layout was inspired by existing chat services such as Discord, Teams or Messenger (3.3.4). The UI can be split into three columns.

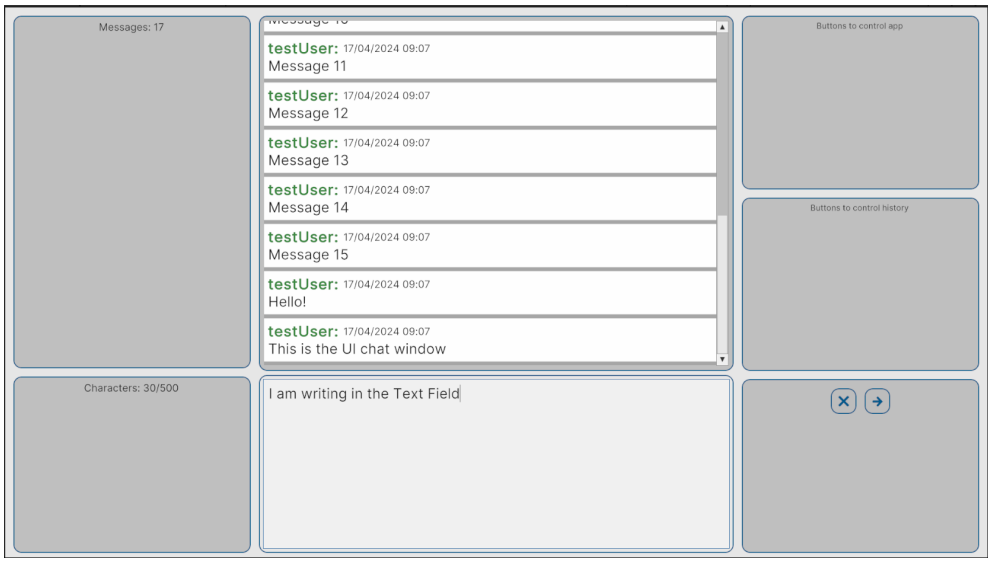


Figure 7.8: Unity chat application screenshot

The left aligned column contains two sections that each contains statistics. The top section contains statistics about the message history and the bottom section contains statistics about the message input field. Currently they have counters. In the top section there are a counter for the number of message and in the bottom, a counter for the number of keystrokes. This was just to provide an example for the layout, and more useful statistics could be added at a later stage. The middle top column contains a message history where posted messages would appear. Each message has a field for the username, timestamp with date and time, and the message itself. The middle bottom column has a text field where a user can input their message. Lastly, the right columns contains sections for controls, where each section has it's own responsibility, like general application controls, chat history controls and chat input controls. We did not populate the general and chat history control panels with any functionality as our focus was on creating an MVP where the chat functionality was in focus. The functionality of the buttons in the bottom section is for clearing the text field and for submitting the message.

7.5.3 Keylogger prefab

Testing of SBKD data capture in an unknown environment was also an essential part of this prototype. While previously only recording the keys typed in a terminal window (7.3), we now had to handle UI as well. We further developed and refined the code from the console application to also work with this implementation.

Initially, we used the `Update()` method that is included in all classes that in-

herit from the `MonoBehaviour` parent class[16]. When the scene the script resides in is active, this function is called every frame. Though technically working in our favor capturing every user's press of keys, it is tied to the application frame rate which could lead to some of the keystrokes being missed or record faulty timing information.

`FixedUpdate()` also seemed like a viable option, as it calls logic inside it in a fixed time interval of 0.02 seconds[17]. This would decouple the capturing of keys from the client frame rate, but does unfortunately not solve the problem of potentially missing keystrokes. If a user presses two or more keys within the fixed interval, two keys would be recorded in the same frame and as a single keystroke element rather than the single one. The data would in this case have to be sorted which would require looping through all of the keystrokes several times.

In the version of the keylogger Unity prefab in this iteration, the decision was made to have the `Update()` function handle the logic. It would check for user input every frame by calling the `DetectKeys()` function which fetched the Keycode for the pressed key by looping through the Keycode Enum used in Unity (Code listing 7.2).

Code listing 7.2: Pseudocode using Update()

```
1  private void Update()
2      {
3          // Check if chatField is active, dont track if not
4          if (chatController.IsTextFieldActive())
5          {
6              // Start stopwatch
7              // Start detecting keys, will run until Enter/return is pressed
8              DetectKeys();
9
10             // Check if return is pressed by checking if it is in the cache
11             if (_keyCache.ContainsKey(KeyCode.Return))
12             {
13                 // Remove return key from cache,
14                 // Send data to API
15                 // Clear keystroke and keycache register
16                 // Restart stopwatch
17             }
18         }
19     }
20
21 private void DetectKeys()
22     {
23         // Loop through each element in the KeyCode Enum that is built into Unity
24         foreach (KeyCode key in Enum.GetValues(typeof(KeyCode)))
25         {
26             // Check if the key is pressed down in this frame and not in cache
27             if ()
28             {
29                 // Add time key was pressed down to key cache
30             }
```

[16] `https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html` - Fetched 11.05

[17] `https://docs.unity3d.com/ScriptReference/MonoBehaviour.FixedUpdate.html` - Fetched 11.05

```
31
32              // Check if the key is released in this frame and was recorded
33              if ()
34              {
35                  // Add keystroke to list
36                  // Remove key from cache, as it has been released
37              }
38          }
39      }
```

The use of the `Update()` function was a good option when wanting to show off the prototype. The goal was not a refined solution, but rather one that AIBA could give specific feedback on while still solving the task specified in the task description (Chapter 11). The code would still have to be optimized and tested, but the ideal feedback we wanted for this stage was on the application workflow rather than how well the code ran.

The key capture data that is saved during a typing sessions is stored in Serializable classes in Unity[18]. By using the UnityEngine JsonUtility class, the class and its data can easily be represented in JSON format. Having the data represented in the correct way, we could send a POST request to the API with the keystrokes and its respective timing information.

Only requiring the injection of a single UI Gameobject where a user can type, this Keylogger prefab was revised and reused when capturing SBKD from the Vivox chat engine (7.7).

7.6 SDK

The following sections introduce all the components of the SDK and how they were implemented in their respective regard.

7.6.1 Capturing SBKD data

The sequence diagram for the Keylogger displayed in Figure 7.10 explains the sequence of interactions between the system's components in response to user-generated events. The process is initiated when a user triggers a KeyDownEvent (KDE) by pressing any key while the text field is focused. The event is captured by the text field through a callback to a KDE handler method in the Keylogger class. Subsequently, when the key is released a KeyUpEvent (KUE) is captured through a similar callback to a KUE handler. The Pseudo code of this interaction can be found in Code listing 7.3.

When a KDE is received, the keylogger class will check with the ChatUIManager class to see if command mode is triggered. If it is, the KDE handler method will reset the keylogger state and return. If the command mode is not triggered the method will proceed with initiating the stopwatch to record the initial KDE time. When a KUE is received the keylogger calculates the total duration of the key-press by referencing the KDE timestamp and current time. This information

[18] `https://docs.unity3d.com/ScriptReference/Serializable.html` - Fetched 16.05

is saved in the *Keystroke* and the *SbkdData* classes (see Figure 7.9).

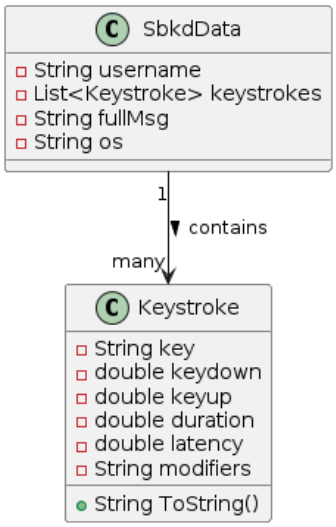


Figure 7.9: Final iteration data classes

After the keystroke has been added to the list of keystrokes the data is prepared for transmission by serializing it to JSON and passing it to the APIHandler. This component is responsible for uploading the keystroke data to the remote API. When all data has been sent and processed the keylogger's internal state is reset.

In parallel with uploading the data to the API, the list of keystrokes is also passed to the KeystrokeStatisticsManager instance which performs statistical analysis on the list and triggers a StatUIManager class to update the values in the statistics panel of the UI.

Code listing 7.3: Pseudocode Key Handlers

```
1    private void KeyDownHandler(KeyDownEvent kde)
2        {
3            // Reset KeyLogger state if command mode, return
4
5            // if key pressed is Return-Key, Send data and reset state, return
6
7            // Start stopwatch if not already running
8
9            // Add keycode and time of key press to cache
10       }
11
12   private void KeyUpHandler(KeyUpEvent kue)
13       {
14           // Assign variable to key pressed and check if cache contains key
15
16           // Assign variables for key down time, up time
17
18           // Add keystroke with information to list of keystrokes
19
20           // Remove key from cache as it has been released
21       }
```

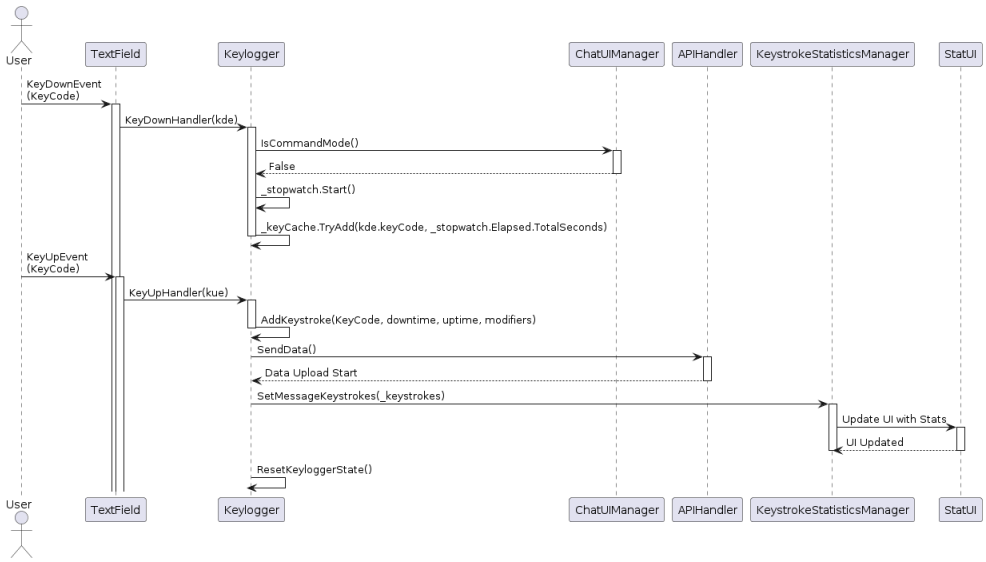


Figure 7.10: Sequence diagram for the keylogger

7.6.2 RESTful API

Initially used for testing data transfer and for facilitating the proof of concept, an API was created as part of the SDK.

The API was created using Golang[19] with a NoSQL Firestore[20] database for storing data captured in the keylogger (7.7).

Golang was used due to its familiarity from the PROG2005 course and it's simplicity for building endpoints. As the API was necessary for proving the feasibility of sending HTTP requests from Unity, the framework we used had to be effectively deployed and functional early on in the development phase.

7.6.2.1 Communication

All communication between Unity keylogger and API is in the JSON file format and is compressed using gzip. Protobuf was also considered a viable candidate for the file format choice due to its efficiency in storing data and speed in serialization and deserialization. However, as the application had not implemented any sort of message batching as suggested in Section 4.2.2.4, we chose to use JSON. Compressed JSON files were found in Section 4.2.2.2 to use less bandwidth when the package had less than 1,900 key entries, which the 320 UTF-8 character limit in the chat application further ensured that all packages were.

The communication is a Request-Response model which in turn follows REST API principles and architectural constraints[21]. The API supports the REST methods GET, POST and DELETE to be able to view, add and delete data stored in the API.

[19] `https://go.dev/` - Fetched 03.05

[20] `https://firebase.google.com/docs/firestore` - Fetched 03.05

[21] `https://www.redhat.com/en/topics/api/what-is-a-rest-api#rest` - Fetched 15.05

When a request is sent to the endpoint, a relevant response body and status code is sent back (Code listing 7.4, Code listing 7.5). This ensures the service that receives gets feedback on whether the transmission was successful or not.

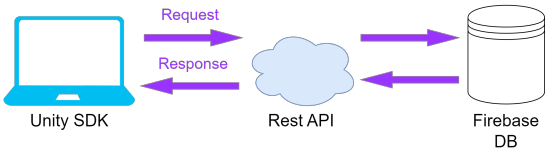


Figure 7.11: Case-specific REST API communication

Code listing 7.4: JSON request when POST-ing data

```
{
    "username": "user",
    "keystrokes": [
        {
            "key": "H",
            "keydown": 0.0811971,
            "keyup": 0.1783721,
            "duration": 0.097175,
            "latency": -0.0960924,
            "modifiers": "None"
        },
        {
            "key": "E",
            "keydown": 0.2418755,
            "keyup": 0.3312188,
            "duration": 0.0893433,
            "latency": 0.0635034,
            "modifiers": "None"
        },
        {
            "key": "Y",
            "keydown": 0.4710933,
            "keyup": 0.5766012,
            "duration": 0.1055079,
            "latency": 0.1398745,
            "modifiers": "None"
        }
    ],
    "fullmsg": "hey",
    "os": "Microsoft Windows NT 10.0.22631.0"
}
```

Code listing 7.5: JSON response when POST-ing data

```
    {
        "id": "9fe48431-3e4e-46cc-9fad-4db2c889a56c"
    }
```

The different status codes, example requests and response as well as how to interact with the API are described in the README that is included as SDK documentation.

7.6.2.2 API Storage

For persistence of data we have used a Google Firebase database. When a valid entry is sent to the API and stored in Firebase, it is given an UUID. In the database, we would ideally be able to separate each entry using a unique identifier to search for single entries. Assigning a UUID to an entry does not make the probability of duplicated values zero, but the chances are considered negligible as they are close to zero[22]. This choice was made due to the need of a NoSQL database, as we believed the JSON structure of the data would be reworked several times in the span of the project. MongoDB[23], which is also a Cloud-based NoSQL database would also fulfill our requirements but fell short due to Firebase's free version offering twice as much storage capacity compared to MongoDB. Other Structured Query Language databases such as SQLite[24] were considered, but were not a viable choice given that the structure of the data for transmission had not been defined this early on in the development phase.

7.6.2.3 Deployment of changes

The API is deployed by utilizing assigned resources on NTNU Gjøvik's Openstack installation, SkyHiGh. This means that the API is dependent on SkyHiGh to function properly, unless deployed elsewhere.

The API is deployed with Docker and simplified using Docker-compose. An installation guide for the initial deployment one can follow on their own can be found in Appendix F.

Using Docker-compose mitigates a lot of manual commands that one has to write. Instead of having to create a new image and then deploy it, we can just run a single command. Furthermore, we facilitated for even less manual interaction with the VM by making a Bash-script that automates deploying changes to the API (Code listing 7.6). This script takes in 2 arguments:

- $1 Absolute or relative Path to the private key used to access the VM
- $2 The directory of files that should be copied

These path parameters should be defined as environment variables or kept in an *.env* file before running the script. Whilst the most common and more secure method used in production environments would be to inject secrets in a pipeline during deployment[25], defining environment variables is a simple measure that makes it considerably more secure. Using environmental variables in this way avoids exposing the file structure of your local machine and is generally considered good practice when dealing with secrets.

The aforementioned bash script runs another bash script on the VM, *update.sh* that sets current directory and runs *docker compose up -d*. A new container with the changes that were applied now runs on the VM.

[22] `https://en.wikipedia.org/wiki/Universally_unique_identifier` - Fetched 02.05

[23] `https://www.mongodb.com/` - Fetched 102.05

[24] `https://www.sqlite.org/` - Fetched 11.05

[25] `https://cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_Cheat_Sheet.html` - Fetched 10.05

Code listing 7.6: Bash script for deploying changes (IP redacted)

```
#!/bin/bash

# Uses scp to copy files securely onto the VM.
scp -i $1 -r $2 ubuntu@[Public IP]:/home/ubuntu/

# SSH into the VM, and run the script on there which deploys the new changes
# to the API + spins up docker container.
ssh -t -i $1 ubuntu@[Public IP] 'sh update.sh'
```

7.7 Unity Chat Application with Vivox

Vivox is a communication service providing voice and text chat technology for online games. In 2019 the company was acquired by Unity and integrated into their developer platform, Unity Gaming Services (UGS). It is currently used in more than 650 games developed using either Unity or Unreal Engine on Steam[26]. It supports most major platforms such as Xbox, PlayStation, PC, Android, and iOS. The service is engine agnostic, and can be integrated with a custom engine using the Unity Core SDK[27].

The reasoning for choosing to implement the chat using Vivox is that we wanted to investigate if there was any unexpected hurdles when capturing SBKD data in an environment likely to be encountered in a Unity game. To further facilitate this we created the chat interface shown in Figure 7.13. A dependency diagram of the final application with the keylogger SDK is also included in Appendix I.

7.7.1 User Authentication

Vivox requires a means of user authentication when using their services. In UGS there is a service called Unity Authentication which lets developers authenticate users through either SDK or API calls. In the project we opted for utilizing the Authentication SDK by creating an AuthManager class acting as a wrapper for the Authentication service methods provided by UGS. This class invokes events on sign up, sign in and sign out which triggers scene changes.

UGS enables users to authenticate anonymously with Unity creating a player ID and associating it with a session token without any input from the user. Additionally, Unity provides functionality for third-party authentication, including Unity Player Accounts, Google accounts, Facebook, Steam, and username and password solutions amongst others[28]. In the project, we opted for the username and password solution. Unity imposes certain requirements for the username and password such as uniqueness of the username, credential length and allowed characters. To comply with these requirements, a regular expression-based validation service class was created, which the credentials were passed though in the sign-up process. In addition, a class for displaying information and visual feedback to the

[26] `https://steamdb.info/tech/SDK/Vivox/` - Fetched 03.05

[27] `https://unity.com/products/vivox-voice-chat` - Fetched 03.05

[28] `https://docs.unity.com/ugs/en-us/manual/authentication/manual/approaches-to-authentication` - Fetched 05.05

user was included, where the border color of the message is based on the validity state of the credentials. As can be observed in Figure 7.12, where a valid username and password are provided, but without a matching confirmation password.

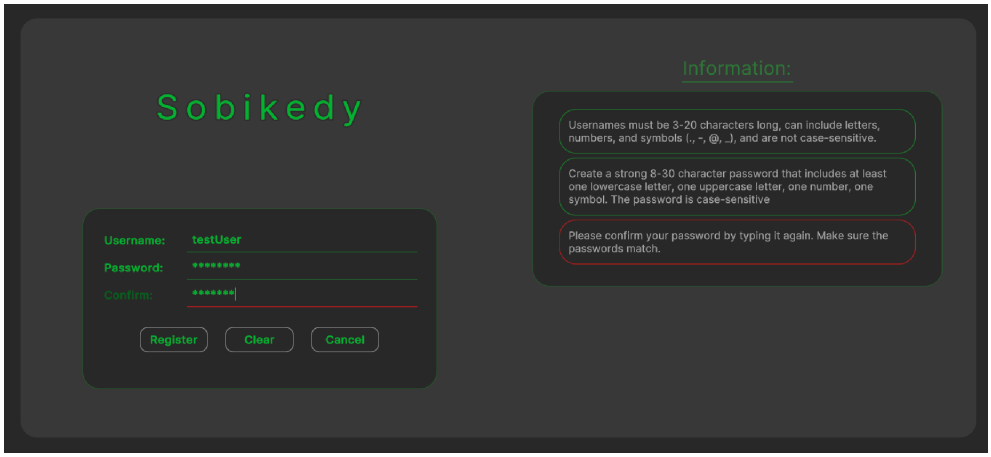


Figure 7.12: Visual feedback on credentials in the sign up process

7.7.2 Concurrent users

It is possible for multiple users to communicate in the chat application simultaneously. A "participants" panel that would be visible whenever multiple users were in the same channel was implemented. This panel would show the usernames of the participants. However, in the final product, it was decided to hide this feature in order to avoid cluttering the UI with unnecessary elements. The majority of the functionality for sending private messages between users instead of in a "public" channel has been implemented, but not utilized, as it would deviate from the core objectives of the project.

Upon posting of a new message to a channel, whether by the user in question or another user, an event is triggered, resulting in the UI being updated by the addition of the new message and a scrolling action to the bottom of the chat history. The username displayed in the message will be coloured green for the currently logged-in user and blue for other users' messages as can be seen in Figure 7.13.

7.7.3 User interface & interaction

In the final iteration of the chat application where Vivox was utilized, an opportunity was identified to enhance the interface and align the visual representation with a more contemporary style. This task proved challenging due to the limitations of UI Toolkit, which is not yet fully developed and lacks certain functionality found in the web technologies it's based on, including z-index, shadows, blurs, and pseudo-classes for USS. The appearance of some default controls, such as ListView and TextField, cannot be modified using the UI Builder. Therefore, the UI Toolkit debugger, a tool similar to Developer Tools or Inspector in a browser,

had to be used to identify the correct USS selectors. The design is inspired by the Discord interface, with modifications made to accommodate the statistics panel. This can be observed in Figure 7.13.

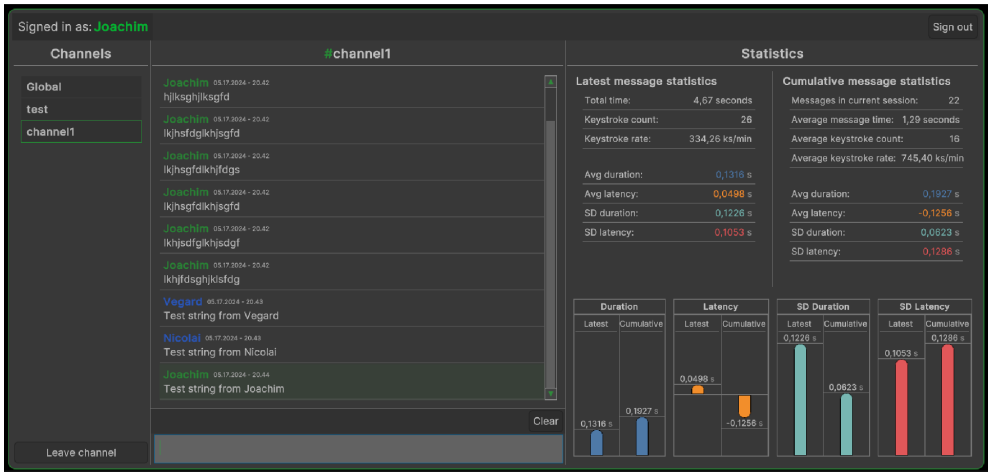


Figure 7.13: The final iteration of the chat interface

The interface can be divided into four sections. The top of the screen displays a tool and information bar, which includes the username of the currently signed-in user. Additionally, a sign-out button is located here.

The first column on the left, with the header "Channels," displays the channels a user has joined. The channels are selectable by clicking them. At the bottom of the channels column, there is a "Leave channel" button, which leaves a selected channel and removes it from the list. The channel with the name "Global" is a channel that all users join by default. It is also not possible to leave this channel.

The middle column displays the chat interface, which contains the message history of a channel and a text field for writing and sending a message to the selected channel. If no channels are selected when a message is written, no actions are taken. However, when a channel is selected, the message history of the channel is displayed. Currently, only the default value of 10 messages are displayed in order to conserve bandwidth and avoid straining the Vivox SDK, as is considered a good practice. However, this limit can be expanded by modifying a configuration constant in the code. A paging mechanism with lazy loading[29] could also be implemented, but this was considered a low priority task. The Scrollbar to the right of the message history is hidden until there are enough messages to warrant scrolling, at which point it will scroll to the bottom of the history when a new message event is received.

At the base of the chat interface is a text field where a user may input a chat message. This text field has a size limit of 300 characters, as the default Vivox configuration has a limit of 320 bytes with the UTF-8 encoding per message. It

[29] `https://www.cloudflare.com/learning/performance/what-is-lazy-loading/` - Fetched 16.05

is possible to alter this limit through a pre-login configuration, as detailed in the Unity manual[30]. However, this was not done in this instance, as the objective was to demonstrate that the system was functioning correctly. The text field will expand and contract in the vertical direction in accordance with the calculated size of the characters. Located immediately above the text field is a button labeled "Clear" that clears the text and resets the Keylogger state.

The Statistics panel, located to the right of the interface, displays statistical information derived from the messages that the user has sent. The top left section of the panel displays the statistics for the most recent message that the user has sent, while the top right section displays the cumulative statistics for all messages in the current session. The lower section of the panel comprises four sections, each containing two bars. The bars provide a visual representation of the statistical values displayed in the upper section of the panel and are color-coded identically to those values. Each section contains a bar for the most recent message on the left and the statistics for the entire session on the right. Since a negative latency is a valid value, the baseline of the latency bars is set to the middle of the container. If the value is less than 0, the bar will reflect about the baseline (to point downwards).

7.7.4 Statistical calculations in UI

Making a statistics panel and displaying the statistics based on the keystroke dynamics data was not part of the initial task. The team still though it would help in showcasing some aspects in the field of SBKD analysis, and to provide some real-time visual feedback to any users of the chat application.

The following list outlines the methodology used to derive the values displayed in the UI. Where applicable, each list entry starts with an explanation for the value of the latest message, then proceeds with an explanation for the sessions cumulative messages. In Figure 7.14 a screenshot of the statistics panel is presented. The messages sent to produce the values consists of a variety of different typing cadences, rhythms and button mashing.

- **Message duration**: is calculated by taking the time from the sessions last KeyUpEvent (KUE) (excluding the enter key press) and subtracting the time of the first KeyDownEvent (KDE) resulting in the total time taken to write the message. The time of the first KDE will be approximately zero in most cases, but some delay in the order of a few microseconds due to processing are factored in. For the cumulative statistics, the durations are summed up and divided by the number of entries to get the average duration of writing a message in the current session.
- **Keystroke count**: in the case of the latest message, the keystrokes are simply counted with a call to the Count() extension method of the IEnumerable interface in C#. For the cumulative statistics, the number of keystrokes in each list is summed up and divided by the total number of messages giving

[30] `https://docs.unity.com/ugs/en-us/manual/vivox-unity/manual/Unity/text-chat-guide/text-chat-guide-overview` - Fetched 17.05

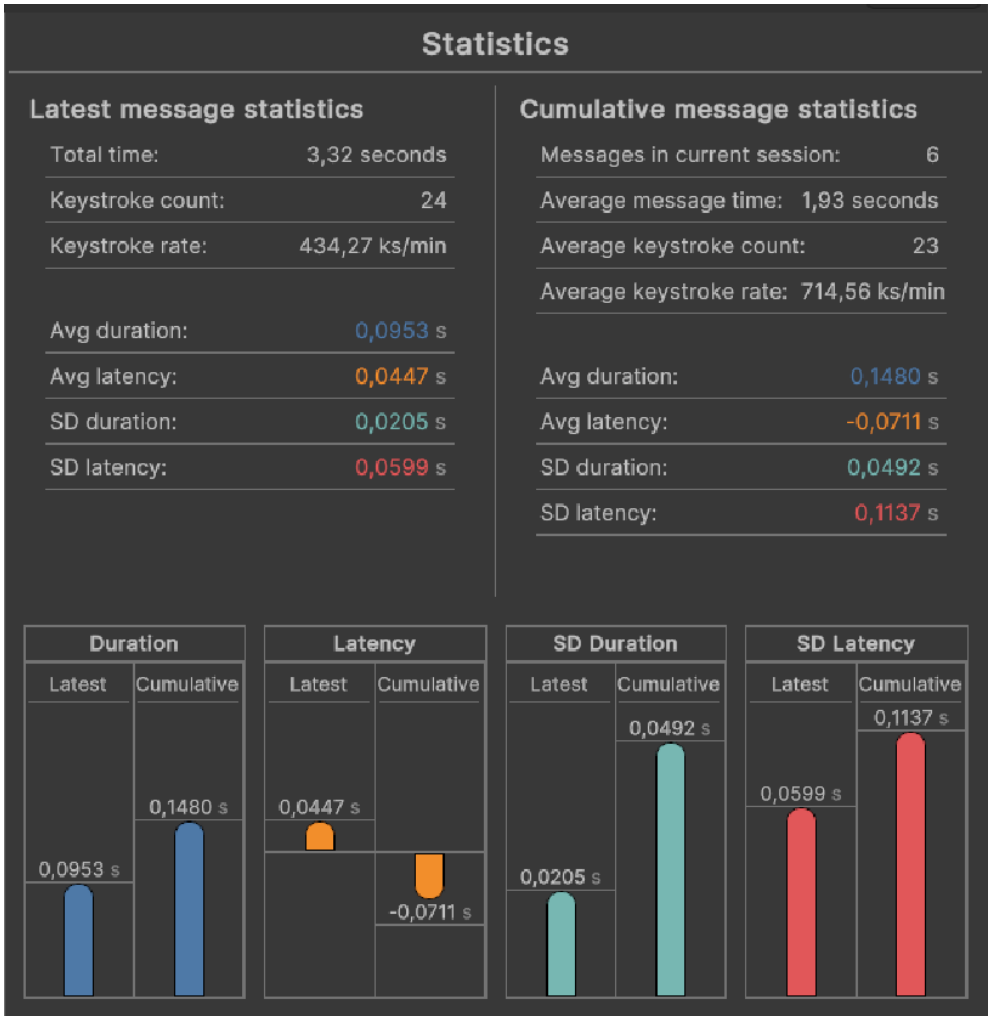


Figure 7.14: The statistics panel of the interface

the average number of keystrokes per message.

For the following calculations there are in essence no other difference between the latest message and cumulative message calculations besides the list that is passed to the method.

- **Keystroke rate**: is calculated by counting the number of keystrokes and divide the sum by the total time resulting in a value with the unit Keystrokes per minute (KPM). Since the keystroke timings are recorded in seconds we convert the time to minutes to get the KPM.
- **Average keystroke duration**: These are the dark blue colored values. They are calculated by using a LINQ expression where the `Average()` method is called on the duration property of the Keystrokes in the respective list. The duration property contains the time value from a KDE to the same keys KUE.

- **Average keystroke latency**: These are the orange colored values. Similar to the duration calculation, the latency also uses the `Average()` method to compute the latency. A notable difference is that the first keystroke's latency is removed from the list before processing as it will always be zero, since the latency itself is calculated by subtracting the previous KUE from the current KDE. Negative values are valid and produced by pressing a different key before releasing the previous key.
- **Standard deviation duration**: These are the cyan colored values. The deviation is calculated by taking the difference between each keystroke's duration and the average duration, squaring this difference, and summing all these squared differences. Finally, the sum of squared differences is divided by the number of keystrokes to get the variance, and the square root of the variance gives the standard deviation. This would show how much variation there is in the keystroke durations relative to the average duration.
- **Standard deviation latency**: These are the red colored values. The latency standard deviation calculation is fundamentally the same as for the duration standard deviation.

7.7.5 Formulas utilized in statistics calculations

We include a general formula for the average of a property in a collection since it is used throughout the calculations. N is the number of elements in the list and x_i would be substituted for the values.

$$\text{Average} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Calculations for keystrokes per minute:

$$\text{Total time in minutes} = \frac{\text{Total time in seconds}}{\text{60 seconds per minute}}$$

$$\text{KPM} = \frac{\text{Total number of keystrokes}}{\text{Total time in minutes}}$$

General formula for standard deviation:

$$\text{Sum of Squared Differences} = \sum_{i=1}^{N}(x_i - \mu)^2$$

$$\text{Variance} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

$$\sigma = \sqrt{\text{Variance}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

where:

- N is the total number of elements in the collection.
- x_i is the value of the i-th element in the collection such as duration or latency of a keystroke.
- μ is the mean (average) value of the elements in the collection.
- σ is the standard deviation.

7.7.6 Keylogger integration with chat application

The integration of the Keylogger class with a chat application is dependent on the implementation of the chat application itself and which frameworks is used. The primary focus when integrating the keylogger with the application we created has been on integration with UI Toolkit because that is what Unity recommend for new UI projects[13]. However, when developing the second iteration of the Keylogger, it was successfully utilized with the old Unity User Interface package (uGUI) before refactoring it to use UI toolkit.

In our implementation of the Keylogger class, two fields are declared: one of the TextField type and another of the UIDocument type. The UIDocument field is marked with the `[SerializeField]` attribute, enabling it to be initialized directly within the Unity Editor. This is achieved by dragging the corresponding UIDocument GameObject onto this field in the Inspector panel, thus linking the UI document to the Keylogger script (see Figure 7.7 section C for a view of the Inspector).

Following the initialization of the UIDocument, the TextField instance is subsequently initialized. This is accomplished by accessing the rootVisualElement property of the UIDocument, which serves as the container for the UI elements defined in the UI document. The TextField is then specifically retrieved by invoking the `Q<TextField>("idOfTextField")` method. Here, "idOfTextField" refers to the unique identifier or name assigned to the TextField element within the Unity Extensible Markup Language (UXML) definition. This method effectively locates and initializes our TextField instance, allowing it to be used within our class for further operations.

Using old UI system

When integrating the Keylogger class using uGUI instead of UI Toolkit, this necessitates a slightly different approach due to the architectural and component system differences between the two. In uGUI, components are generally attached directly to GameObjects in the scene, and these components are managed and accessed through scripts. In the case of uGUI, one would begin by attaching a script to the GameObject that contains the InputField component, which is uGUI's equivalent of UI Toolkit's TextField.

To get a reference to the InputField component the GetComponent

<InputField>() method could be used, which retrieves the component attached to the same GameObject as the Keylogger. The reference to the InputField component allows the Keylogger to subscribe to the onValueChanged and onEndEdit events, which are triggered whenever the user types into the field or completes their input, respectively.

Custom solutions:

Another possibility would be to use the custom input handlers with "Input.GetKeyDown" and "Input.GetKeyUp" from the Unity Scripting API and associate these key events with the active UI elements which would be a lot more complex than using either the built in functionality of UI Toolkit, or the "valueChange" and "onEndEdit" listeners of uGUI.

The last possibility we explored to a significant depth was creating a custom control for a text field. In this option the best suited solution would be to extend the already existing TextField or BaseField of UI Toolkit to override or create new methods with the desired functionality. This would still provide the opportunity to inherit the ability to register callbacks with KDE and KUE. A possible advantage to this option would be to build the Keylogger's functionality directly into the new TextField control.

7.7.7 Command-line mode

As a way to explore one of the group members personal curiosity, and to prevent the UI becoming crowded with elements that could take away focus from the core of the Proof of concept (POC) application, command-line functionality was added. It was implemented directly in the text field analogous to how Discord commands operate.

In the chat interface UIManager class there is a callback to a handler method on the text field for handling message submission when a KDE with the key code for the "enter" key is pressed. A boolean flag, "IsCommandMode", at the top of this function is set to "true" when the first KDE has the key code for the "/" character. This flag is accessed in the Keylogger and stops the recording of keystroke dynamics data and resets the Keylogger's state.

In the figure below (Figure 7.15), a class diagram of the command-mode related entities is shown:

Furthermore, a call to the static method `CmdProcessor.ProcessCommand(string input)` is performed. This method trims the command keyword "/", from the input string, then splits the rest of the string into a command and an argument and passes them to the asynchronous `CmdProcessor.ExecuteCommand(string command, string argument)` method for execution. See (Figure 7.16) for the activity diagram providing a high level overview of the decisions in the process.

A member of the CmdProcessor class is a field of a Dictionary type with the generic type arguments "string" and "ICommand". This dictionary is used to hold a mapping of command keywords to their corresponding ICommand implementa-

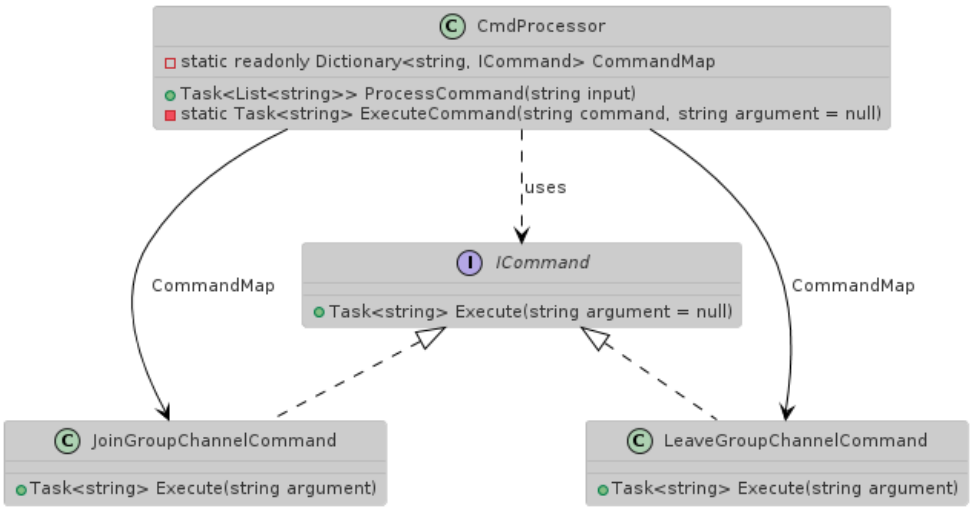


Figure 7.15: A class diagram for the command mode related classes and interface

tions. There are currently two implemented commands, "JoinGroupChannel" and "LeaveGroupChannel".

The ICommand interface was created to promote separation of concerns within the application. Each command implementation can focus solely on its specific functionality without needing to consider the execution logic handled by the CmdProcessor. This design approach simplifies the process of extending the system; new commands can be added without modifying the CmdProcessor class, adhering to the Open/Closed Principle. By not requiring CmdProcessor to be aware of the details of each command, the design achieves loose coupling between classes, enhancing modularity and maintainability.

To expand with additional commands, a command class that implements the ICommand interface needs to be created. It is also recommended to add a constant for the command keyword. Then a new entry in the command dictionary of the CmdProcessor class needs to be created with the command keyword constant and an instance of the newly created implementation of the command class.

Example commands:

- If the command `/joinGroupChannel channel1` is typed in the input field and the enter key is pressed to submit the command. A Vivox group channel with the name "channel1" would either be joined, or created and joined if it did not exist prior to the command submission. The channel would be displayed in the "Channels" panel of the UI
- Similarly the `/leaveGroupChannel channel1` command will leave the channel if it is joined. It is possible to re-join the channel again after leaving.

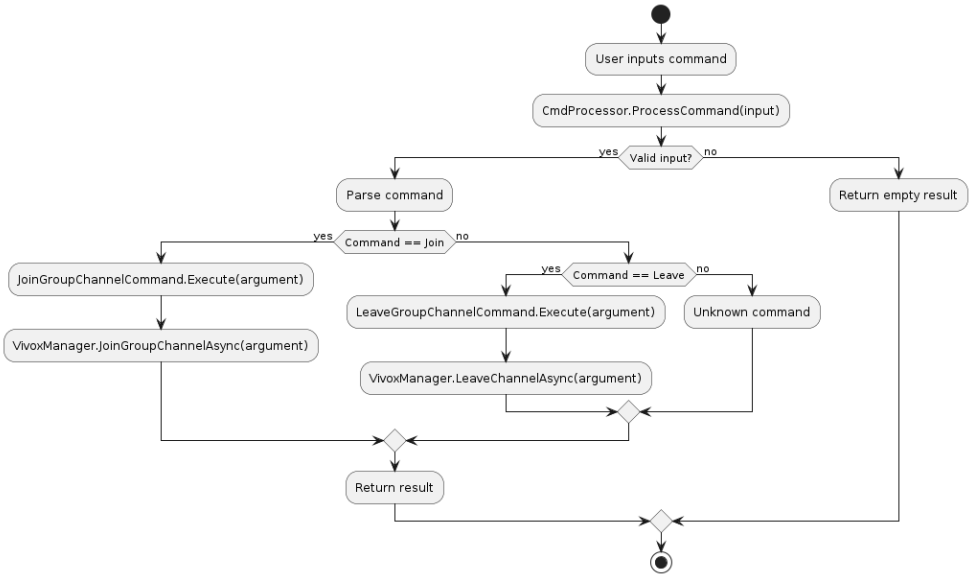


Figure 7.16: Activity diagram illustrating the command handling process in the CmdProcessor

7.7.8 Unity Asset store

If the SDK is to be deployed on the Unity Asset store, there is a requirement imposed by Unity for the implementation of a DSA notification that is shown to a signed in user. Any time a user signs in, a check for notifications should be performed, and if any are found it should be displayed to the user. The same applies for restricted user. To help developers meet this requirement, Unity has created a new notifications API for this purpose[31].

[31] `https://docs.unity.com/ugs/en-us/manual/authentication/manual/dsa-notifications` - Fetched 16.05

8 Deployment

This chapter provides information on how the solution could be deployed on the companies services, with detailed instructions on how to integrating the SDK with AIBA's API gateway.

8.1 Deployment on client service

AIBA provided general instructions on how their API is configured and how their customers connect to it. From this information, we will explore the process of rerouting the Soft Biometric Keystroke Dynamics (SBKD) data from the Unity SDK directly to the client.

8.1.1 AWS API Gateway

Amazon Web Services (AWS) is a cloud computing service providing comprehensive Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) services. Providing a pay-as-you-go model, this makes it a great choice for companies such as AIBA as they can administrate the costs and scale up and down as needed[1]. The client has recently ported their services to AWS, and use one such IaaS service; the RESTful API gateway for transmission of chat data to and from their clients. The API gateway is described in Figure 8.1.

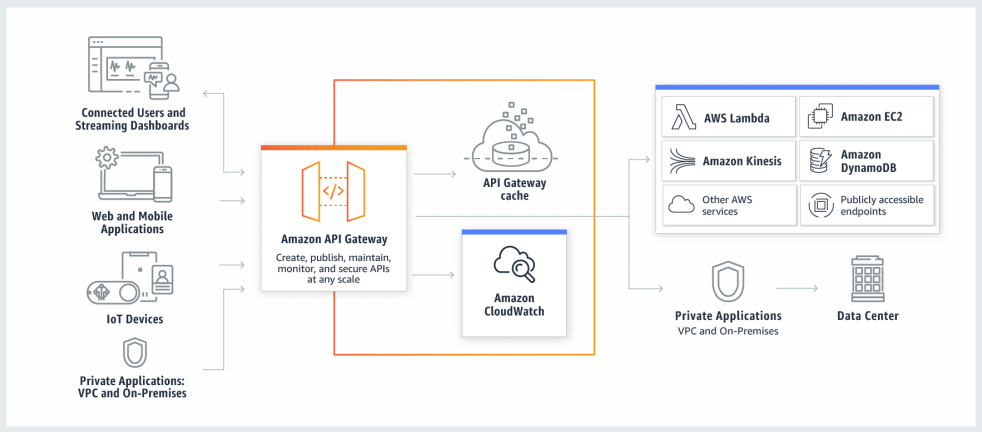


Figure 8.1: Amazon API gateway architecture [1]

Some of the relevant features AWS provides by the AWS API gateway[2] are:

- Support for stateful (WebSocket) and stateless (HTTP and REST) APIs.
- Authentication mechanisms (AWS Access Management policies, Lambda authorizer functions, and Amazon Cognito)

[1] `https://aws.amazon.com/api-gateway/` - Fetched 19.05

[2] `https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html` - fetched 19.05

- Tools for monitoring API usage and costs (CloudTrail and CloudWatch)

AWS provides a graphical UI for creating APIs. Furthermore, in terms of APIs, one is presented with the choice of creating an *HTTP API* or a RESTful API, where the latter provides more functionality and supports more features.

8.1.2 Enabling private integration

Accessing the API gateway can be done through a number of ways[2]:

- AWS Management Console – provides a web interface for creating and managing APIs. Have to create an account, an administrator account and assign users to privilege groups.
- AWS SDKs – Simplifies authentication, integrates easily with development environment, and provide access to API Gateway commands. Can only be used if AWS supports the programming language.
- API Gateway V1 and V2 APIs
- AWS Command Line Interface
- AWS Tools for Windows PowerShell

Ultimately, AIBA uses a template for building an API with private integration to allow customers to access resources through HTTPS [3].

Implementation wise, AIBA must facilitate a private connection by creating a new REST API and turn on VPC proxy integration by enabling VPC link. Furthermore, they must add the VPC id, `stageVariables.vpcLinkId` to the VPC link. By adding the *VpcLink*, approved clients requests will be forwarded to the Virtual Private Cloud (VPC) that the company possesses. Thus, this becomes the integration endpoint.

When the API is configured and eventually called by a client, the network load balancer routes the requests to the intended receiver component of AIBA's Virtual Private Cloud and returns back-end responses to the caller.

8.1.3 Integration with SDK

First and foremost, the Unity keylogger prefab must be attached to a chat in a Unity game to start capturing and sending data (Section 7.7.6). This SDK incorporates all the functionality needed to capture keystroke dynamics data in a Unity game chat and sending it to an API (Section 7.7).

When it comes to the data transmission, the Unity SDK we have developed already interfaces with a self-developed REST API and should therefore have no issues with accessing AIBA's API. REST API interfaces are meant to be uniform[4], and as long as they make the effort to grant access to the SDK, it should be plug-and-play. Additionally, as the *APIHandler* from the Unity SDK sends compressed data, AIBA must ensure that they decompress the zipped file to access the JSON contents.

[3] `https://docs.aws.amazon.com/apigateway/latest/developerguide/getting-started-with-private-integration.html` - fetched 19.05

[4] `https://restfulapi.net/` - fetched 19.05

9 Quality Assurance

In this chapter, we delve into the details of how we ensured the quality of the SDK. This is accomplished by examining and testing its conformity to the requirements outlined in Section 3.3.1, followed by presenting the standards used to streamline the program's code. Lastly, the chapter discusses how our SDK performs across different operating systems.

9.1 Quality assurance of keylogger

To ensure that the keylogger functions as within the requirements set forth in Chapter 3, it is crucial to conduct adequate testing to verify its compliance. Given that our main product is an SDK and not intended for direct user interaction, we determined that user tests would provide limited feedback on the SDK's functionality. Instead, we focused on verifying the precision of the times recorded by the SDK, aiming for accuracy and a low variability from expected values.

9.1.1 Manual Verification and Analysis of an Automated Test Setup

The test setup involved an automated Python script utilizing the pynput.keyboard library to simulate key presses and releases at set intervals. Due to the pynput.keyboard library's key press and key release methods, each of which took approximately 0.9 ms on the test machine, the script also logged the actual duration and latency of each keystroke from the script. We then used the script to log key press times on the SDK within the chat application. The times recorded by the Python script and those provided in the Vivox statistics panel were then manually logged into a separate list to facilitate a comparison between the Python script and the SDK.

9.1.2 Results from tests

Based on 25 rounds of using a Python script to type 320 characters into the application, we calculated the mean differences in latency, duration, and total typing time. Additionally, we computed the standard deviation for these values to assess the stability of the measurements.

- **Latency:** The mean delay from key press to the time logged by the SDK was 0.38 ms, with a standard deviation of 0.12 ms.
- **Duration:** Unexpectedly, the SDK recorded durations 0.37 ms faster than the key presses, with a standard deviation of 0.13 ms.
- **Total Time per Keystroke:** Surprisingly, the SDK registered total times 0.22 ms faster than the Python script, with a standard deviation of 0.23 ms.

9.1.3 Timing Discrepancies and Consistency Analysis

It may appear counterintuitive that the duration times registered by the SDK are shorter than those recorded by the keylogger. However, a closer examination of the libraries used to simulate the key presses provides an explanation. In a test

involving 10,000 key presses and releases, we found that the average times were 0.89 ms for key presses and 0.91 ms for key releases. Further examination of the key release method in the Python pynput.keyboard library revealed that additional checks are performed after the key is released, which adds extra overhead before the method completes. Conversely, the SDK timer stops immediately after registering the key release. The negative total key entry time appears to be a follow-up error from the duration measurement in Python.

Despite these unusual numbers, they are not as critical as the consistency of the time logging. A discrepancy of half a second between the time a key is pressed and the time it is logged in the SDK would render the keylogging functionality nearly useless due to insufficient precision for analysis. Fortunately, the standard deviation between the key press script and the SDK is very low. The standard deviation for total time per key press is under 0.25 ms, while duration and latency are both well under 0.15 ms.

9.1.4 Evaluation of Non-Functional Requirements

The first requirement in the non-functional requirements (Section 3.3.1) regarding performance is that the delay for each keystroke should not exceed 78 ms, as this is perceivable to the average person. Our tests show that the delay for each key press is at most 0.25 ms, meaning that this requirement is satisfactorily met.

In an early client meeting, Professor Bours indicated that the SDK does not require accuracy beyond 0.1 ms. The standard deviations, which show times slightly above 0.1 ms in variation, suggest that the system is either already adequate or very close to being adequate for a final product, thus almost satisfying the second non-functional requirement regarding maintaining discrepancies below 0.1 ms.

Regarding the ability to capture keystroke data at the speed of high-end human typing, the goal was to capture at least 441 characters per minute. In our tests, we were able to capture all keystrokes at speeds in excess of 10,000 keystrokes per minute, far exceeding this requirement.

For the last requirement, we conducted tests as described in Appendix G, where we outlined strategies for minimizing file sizes and implemented the specified approach, as explained in Section 7.6.2.1.

9.1.5 Final thoughts

The low standard deviation observed is a promising indicator of the SDK's consistency. Additionally, the fact that the mean differences between the SDK and the Python script are less than 0.25 ms underscores the SDK's promising accuracy.

Despite identifying that the issue of shorter times in the SDK than in the python script likely resides within the Python pynput.keyboard library's timing for completing the key release method, the fact that the SDK reports faster times for duration raises some doubts about the accuracy of these tests. A variety of approaches were attempted to address the issue, including multi-threading the Python script to decouple the key press from the time logging. However, these approaches proved infeasible. Nevertheless, given the low standard deviation, we

remain confident that the results demonstrate the SDK's ability to capture stable and reliable keystroke times. Therefore, the SDK appears suitable for use in a final product for capturing keystroke data.

Regarding the functional requirements, we satisfactorily met all but one goal. The unmet goal pertained to achieving a standard deviation for the key press of less than 0.1 ms. Our tests showed a standard deviation slightly exceeding 0.2 ms for the total key press duration. While this misses the target, it is reasonable to suspect that this discrepancy may stem from variation in the Python script or computer performance. Thus, it is plausible that the SDK meets the non-functional requirements specifications, but we are simply unable to prove it with our current test setup.

9.2 Use of standards

We have made an effort to follow common development standards for Unity in our project. Together with what we have been taught at NTNU we also used the Unity website's how-to for naming and code style tips as a reference[1]. In addition to this we used the code convention guide in the Microsoft .NET website[2].

In Unity and C# there are some notable differences in naming conventions, serialization, method handling, and property usage. In C#, *PascalCase* is used for public methods and class names, and *camelCase* for private fields, whereas Unity often uses *PascalCase* for public fields to facilitate Inspector access. Serialization in C# relies on standard .NET attributes and mechanisms, while Unity uses its own system, requiring public fields or `[SerializeField]` attributes, which can be limiting for complex types. C# methods are explicitly called by developers, but Unity has the lifecycle methods like Start(), Update(), and Awake() that are implicitly managed by the engine. Additionally, while C# encourages encapsulation through properties with getters and setters, Unity frequently uses public fields for ease of accessing them in the inspector, potentially compromising encapsulation principles.

9.3 Testing for different Operating systems

To ensure keystroke capture is supported on as many devices as possible, we have built the chat application on other operating systems to test whether this is the case.

9.3.1 Windows & macOS

In the instance of Windows and macOS, the SDK successfully captures keystrokes on both operating systems and the chat application works as intended. This is

[1] `https://unity.com/how-to/naming-and-code-style-tips-c-scripting-unity` - fetched 16.05

[2] `https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions` -fetched 16.05

likely because Vivox provides SDK support for both of these operating systems [3], allowing the built applications to run and function properly natively.

9.3.2 Linux

Vivox does not currently have Linux support[4], and it is therefore not possible to log in to the chat application we have created. When attempting to do so, the connection times out and you are unable to get past the login screen. This is likely an error caused by the lack of support from Vivox, as when testing the Unity chat application from the second stage of the feasibility stages (Section 7.2) in Linux, the data was successfully sent to the API.

This means that unless Vivox implements Linux support, a chat application using Vivox with the SDK ran as a native application on Linux will in this specific case hinder the SDK from functioning properly.

[3] https://docs.vivox.com/v5/general/unity/15_1_170000/en-us/Unity/developer-guide/supported-platforms.htm?TocPath=Vivox%20Unity%20SDK%20documentation%7CVivox%20Unity%20Developer%20Guide%7CSupported%20platforms%20and%20versions%7C_____0 - Fetched 19.05

[4] https://support.unity.com/hc/en-us/articles/4780622639636-Vivox-Does-Vivox-offer-Linux-support - fetched 19.05

10 Discussion

This chapter delves into the key findings, the project process and the implementation of the report, providing a comprehensive discussion of the project's outcomes.

The chapter begins with a summary of the primary results, highlighting the creation of a fully functional SDK and the regulatory implications of implementing it in a final product. It then discusses the project's process, emphasizing the use of Scrum and the changes in project scope. The implementation of the SDK is examined, including challenges faced and solutions implemented. The feasibility and viability of using SBKD for age detection in game chats are evaluated, considering both the need for such tools and their efficacy. The chapter also addresses legal and ethical considerations, focusing on the implications of using AI and SBKD for age prediction under various regulations. Finally, the discussion concludes with insights into future work, outlining potential improvements and additional features for the SDK.

10.1 Key Findings

The main results of our work consists of creating a fully functional SDK that has the ability to collect Soft Biometric Keystroke Dynamics (SBKD) data from a Unity application and transmit this data to an API. Another major finding is having established that the use of SBKD for age prediction lies within the high risk category of the EU AI Act. In addition where this concept finds itself within the KOSA, DSA and OSA regulations.

In addition to these primary goals from the original assignment, we also created a chat application in Unity for testing and visualizing the SDK. We conducted a market scan on existing age verification methods, finding that most platforms studied employ similar age verification methods to comply with COPPA. Interestingly, only 1 of 9 services explicitly stated that they use keystroke dynamics for authentication in one way or another. We also established where the final product will fit under the KOSA, COPPA, and OSA regulations. Furthermore, we conducted a study on SBKD transmit sizes and laid forth a strategy in Section 4.2.2.4 for minimizing the bandwidth cost related to transmitting this data.

10.2 Project Process

One of our objectives was to achieve a stable and comprehensive understanding of utilizing an agile process framework. Ultimately, the project team selected Scrum as the framework to be used. While the project initially focused on the creation of an SDK and conducting either a research study on regulations or a market scan of existing solutions, the use of Scrum allowed us to quickly change focus based on discussions during client meetings. These discussions identified exciting ideas that expanded the project scope, allowing us to gain more knowledge and to make the project more our own.

10.2.1 Use of Scrum

Although our adherence to the principles of Scrum was somewhat lacking in the beginning, we improved over time and reinforced the principles. By the latter half of the project, we had established effective routines, including daily stand-ups, productive Scrum reviews with the client, better story creation processes and valuable Scrum retrospectives. Effectively implementing Scrum not only made our workflow more systematic but also enhanced the overall quality of our project outcome.

10.2.2 Change of Scope

The initial project description outlined the creation of an SDK and either a market scan of existing solutions or a minor study on upcoming regulations. After discussions with the client, it was decided that the scope of the project would be expanded. This expansion included conducting both a market scan of existing solutions and upcoming regulations, as well as a study on effectively transferring Soft Biometric Keystroke Dynamics (SBKD) data. Additionally, we decided to develop a chat application and integrate it with our SDK. This facilitated testing of the SDK and provided the client with a tool to test and further develop the SDK. These expansions made the project significantly larger than initially planned, but it enabled us to deliver a more comprehensive product to the client while gaining deeper insights into areas of our interest.

10.2.3 Deviations from Project Plan

While reexamining the project plan, some deviations from the original plan were discovered.

The *black box* of profiling ages and genders that AIBA were responsible for after the data collection and extraction had been completed, was originally something we believed we would take part in. This was likely due to the fact that AIBA mentioned that they had no prior employment of SBKD on their moderation platform and a misunderstanding on our part. This, in the end, had little impact on development, but became an uncertainty until it was clarified shortly after the project plan had been delivered.

Initially, we planned to arrange user and expert evaluations, but the idea was scrapped early on. Testing our SDK on users wouldn't provide sufficient benefits, as its functionality isn't visible to users and they wouldn't focus on the most critical aspects we needed to test. Although the chat application's UI would have made the tests easier to conduct, it wouldn't have provided the correct feedback. The tests would have focused on the UI and chat application functionality rather than the underlying processes of the SDK, which was our main objective.

10.3 SDK implementation

Through the work on the SDK, we got the opportunity to explore new technologies and refine our skills in the field of software development. At times, it was

challenging to work without prior knowledge and experience, as well as not having the training wheels of a typical university assignment. However, these challenges gave us a deeper understanding and adaptability, ultimately enhancing our problem-solving abilities and confidence in taking on real-world projects more independently.

10.3.1 SBKD SDK

Many aspects of this project was unfamiliar to all members of the group. No members of the group had used C# previously, though we found it quite similar to other programming languages encountered. Unexplored nuances such as the syntax, new IDE's, events and delegates made it more challenging to master.

At times, it was difficult to determine what was expected and how to progress. However, by consulting with the client, the supervisor, and conducting independent research, we managed to find solutions to most of our problems. Some elements, such as data transmission using HTTPS in Unity, proved too time-consuming and complex. With encouragement from the client to focus on the core functionality, this aspect was abandoned.

A concern that was up for consideration was to have a complete separation of the SDK and chat application. In the final deliverable we opted for a close integration. This was in part due to the numerous options Unity offers for a control like the text field, including various functionalities and event registration methods that are not interchangeable. By integrating the keylogger functionality of the SDK with the text field provided in the chat application, the client is provided with a comprehensive prototype that can be utilized without having to make any additional modifications.

If the client for any reason needs compatibility with the old UI system, uGUI, we outlined an alternative for how the keylogging functionality could be implemented using this in Section 7.7.6. Here we mentioned creating a custom control implementation that would integrate the recording of keystrokes directly. This option was something we became aware of at a late stage in development, and therefore did not try to implement it. We did however investigate how it could be done and if it could provide any benefits over our chosen implementation. We think a direct implementation in a custom control could better encapsulate the keystroke recording logic thus making it simpler to reuse and maintain.

10.3.2 Unity specific development

Our introduction to Unity and game development presented a valuable learning opportunity. While navigating Unity's extensive Platform roadmap[1], we encountered challenges due to the large backlog of planned features, some dating back several years. We noticed frequent discussions in Unity developer forums about delayed "promised" features, often attributed to shifts in focus or the need to rework existing systems. This experience underscored the importance of thor-

[1] `https://unity.com/roadmap/unity-platform` - Fetched 19.05

oughly researching the chosen technology to avoid assumptions about feature availability and to ensure realistic planning that accounts for potential limitations.

Another problem we encountered in the implementation phase was a lack of comprehensive and up-to-date documentation for Unity related code. Often times, we had to either rely on the hope of finding a post discussing a related problem in the Unity developer forum, or just figuring it out from scratch. One possible reason for this lack of or outdated documentation is that Unity sometimes has multiple concurrent frameworks providing the same functionality, as is the case with the three UI systems: IMGUI, uGUI, and UI Toolkit. In addition to this, the input system, rendering pipeline, networking, physics engine and animation system also has multiple systems each. With that being said, using Unity as a development platform gave us a great deal of flexibility in how things could be implemented, but at the cost of doing lots of research and having to make decisions that had to be reverted at a later point in the development process.

10.3.3 Feasibility stages

By working through feasibility stages, we were able to investigate the problem incrementally. This approach allowed us to pivot the project in conjunction with the client if it became clear that the task was impossible or unlikely to be completed. Partitioning the project into stages with increasing complexity and expanding scope provided us with a great opportunity to become comfortable with the new environments and learn from mistakes and challenges encountered in previous iterations. This process also gave us a better chance to identify what was missing or what could be removed, particularly in the UI part of the development.

On the other side, not incorporating feasibility stages and setting a goal of creating the end-product from the start could have reduced the time needed for the development stage. It would have provided more time for direct work on the final deliverable and other aspects of the thesis. Each option would have its advantages, but overall we think using incremental stages provided the best learning outcomes as the process became more comprehensive.

10.3.4 Vivox Chat Application

In the planning phase of the project, many possible solutions for how to start capturing keystroke data was investigated. A screen overlay was considered, but the idea was abandoned in it's infancy due to lacking relevance for the given task. Another approach would have been to use the console application we initially created for collecting the required data. This option would limit the use of the input systems provided by Unity, and therefore not solve the task of collecting keystrokes in a Unity application. Early on in the process we observed the need of having a natural environment to capture SBKD data from. Because of this reason, we decided to create the chat interface using Unity's own chat service, Vivox. This proved to be a time consuming endeavour and a task that was more comprehensive and complex than initially though.

We spent some time working on features that we later decided against, such

as sending direct messages between users. This was a thing we originally wanted and saw as a necessity, but upon further consideration it was decided that the group channel messages would suffice in providing the needed features. Thus, the necessary functionality for direct messages was implemented, but ultimately remains a disabled feature.

Since the core of the project concerns collecting metadata about keystrokes, which will be further processed and used in statistical analysis, we saw the opportunity to provide visual feedback to a user of the application. This was achieved by displaying some of the captured values and values derived from the data in the statistics panel. We also planned to utilize this statistical display in a demonstration for the presentation of our work.

When work on the implementation of the bar chart in the statistics panel started, we encountered a problem. Unity does not have any built-in functionality for creating charts and graphs. We searched the internet for a solution, but found none that would fit our needs. In the Unity Asset store there are some solutions that claim to work, but as those had a high price tag we did not consider those an option. In our solution we ended up creating the charts ourselves from scratch using plain VisualElements that gets scaled based on values passed to a class for calculating statistical values.

10.3.5 Final thoughts on Chat Application

Since all requirements and goals of the project were met, we are satisfied overall with how the chat application turned out. However, there are still some aspects of the application that could be improved, such as testing and improved error handling, detection of edge cases, more detailed documentation, optimization and other minor improvements. The UI itself and state-handling in the UI also needs more work. Considering the applications purpose was to work as a prototype for proving and visualizing a concept, we are generally happy with the achieved final results.

10.3.6 API

It was initially difficult to determine whether the REST API would be deemed as part of the final Software Development Kit or not. An SDK should include several components and an API is a one such vital addition. Furthermore, the entire SDK on its own would not be complete without it, as there would be nowhere to send the data if it is not present. On the other hand, however, the fact still remained that AIBA already has an API which the SDK should interface towards, making the API redundant when the client ultimately would integrate the SDK into their stack themselves.

The API was in the end kept as a component of the SDK. Due to the concerns addressed above, however, the API ended up not being as extensive as a production-ready API should be. As of its latest iteration, it still utilizes HTTP in favor for HTTPS, and there are no authentication methods present. This would later be labeled as "future work" when we decided we had spent too long attempt-

ing to set this up without much progress being made.

10.4 Viability of SBKD for Age Detection in Game Chats

While we have successfully developed a product that captures Soft Biometric Keystroke Dynamics (SBKD) in Unity-based games and determined that using AI with SBKD to predict user ages is permissible under the reviewed regulations, there remains the question of its viability as a product.

10.4.1 The Need

Game moderation faces significant challenges, particularly concerning the prevalence of toxic behavior and online grooming. Statistics highlight widespread toxicity in online video games, and while specific research on the occurrence of grooming in video games is limited, studies confirm the growing problem of online grooming in general. This information suggests that tools aiding moderators and law enforcement in detecting and preventing cyber grooming would be valuable additions to a game moderator's toolkit.

10.4.2 The Efficacy

With regard to the efficacy of using SBKD to predict user age, studies cited in Section 4.1 indicate an accuracy range of 67% to 78% for age prediction and 70% to 86% for gender prediction. One study achieved up to 90% accuracy in these categories. An accuracy of 67% for age prediction may not be sufficient for moderators, as it could result in a significant number of false positives, increasing their workload. However, discussions with the client revealed that the accuracy of their age prediction AI could be significantly improved by predicting age groups rather than specific ages, potentially yielding much better results.

10.4.3 The Viability

There is a clear need for effective tools for age detection in game chats. The question thus arises as to whether the efficacy of age prediction with SBKD is good enough to be useful. If focusing on age groups rather than specific ages substantially improves accuracy, these tools could be sufficiently effective. Furthermore, considering AIBA's Amanda software, which would use not only SBKD to highlight worrisome conversations, but also chat history analysis. The combined approach could significantly enhance accuracy and viability. This comprehensive tool-set is likely to provide a useful and practical solution for chat moderators, thereby improving their ability to detect and address grooming behaviors effectively.

10.5 Legal and Ethical Discussion

10.5.1 Legal

In our project, we focused on four key legislative frameworks: the EU AI Act, the US Kids Online Safety Act (KOSA), the EU Digital Services Act (DSA), and the On-

line Safety Act 2023 (OSA). The DSA and OSA were enacted in 2022 and 2023, respectively, while KOSA and the EU AI Act have yet to come into force. Among these, particular emphasis was placed on the EU AI Act due to its prominence as the world's first regulation specifically addressing the use of AI[2], which is highly relevant to our SDK's final product which collects SBKD data and uses AI for predicting users' ages.

Through our work on these legislations, we gained a comprehensive understanding of their objectives, potential impacts, and the specific products they affect. We particularly examined how the use of our SDK for age detection aligns with these regulations. Our research on the EU AI Act was noteworthy, both in terms of our analysis and the timeliness of our completion—just one month after the Act's latest version was released.

We believe our work significantly contributes to the understanding of the implications of using SBKD and AI for user information collection.

10.5.2 Ethics

The program profiles not only adult individuals but also children, which raises significant ethical concerns. Determining whether our use case scenario makes it ethical to collect someone's biometric data is challenging and requires careful consideration.

Many remedies for these concerns are already enforced by various regulations, making it crucial to adhere to them. These include the regulations discussed in this report, as well as GDPR, COPPA, and other relevant regulations in the EU, UK, and USA.

Key aspects of these regulations include GDPR's requirements to inform users, obtain consent, and collect only the minimum information necessary. In addition, the upcoming AI Act mandates the use of high-quality training datasets to ensure accurate AI results.

Even when adhering to these regulations, ethical concerns persist. Monitoring users' keystrokes to detect suspicious behavior may infringe on their autonomy and privacy. Users may feel surveilled, altering their natural behavior and negatively impacting their gaming experience. Balancing child safety with user privacy and autonomy is essential. Determining whether keystroke logging for a limited number of messages is sufficient for accurate age prediction requires careful consideration by AIBA.

False positives are another critical issue. If the product incorrectly classifies an adult as a minor, or vice versa, it could lead to unwarranted surveillance. This risk must be mitigated through careful procedures. While the AI Act's obligation to use quality datasets addresses this to some extent, additional measures are needed. Providing a confidence interval score for the age predictions could help, but other safeguards are needed for when there is high-confidence yet incorrect predictions. Given that AIBA intends to utilize the product in conjunction with their Amanda

[2] `https://www.wiley.law/alert-EU-Adopts-the-AI-Act-The-Worlds-First-Comprehensive-AI-Regulation` - Fetched 17.05

platform, which is capable of generating a risk score based on the conversation history, one solution might be to prevent a moderator from further investigating a user based solely on their predicted age.

There is also a broader debate about whether such software contributes to increased societal surveillance and whether the benefits justify the means.

While we acknowledge these serious ethical concerns, we believe it is possible to implement this program ethically. First, careful compliance with all applicable regulations is essential. Second, clear communication with users about what data will be collected, how it will be used, and for what purpose is critical to ensure that users both understand and accept what they are agreeing to when they play video games that implement the final product. Third, robust measures must be in place to minimize unnecessary surveillance. This includes systems to prevent false positives from triggering investigations and to stop keystroke monitoring when a high confidence interval is reached, indicating no further monitoring is necessary. We believe that these measures would greatly improve the ethics of using such software.

10.6 Future work

SDK mobile support: Unity can also be built on many other operating systems than Linux, Windows and macOS; the ones we have tried. The SDK singles in on physical keyboards, but it is also possible to support extracting keystroke dynamics from touch-screen devices, which would greatly extend the potential games that the SDK could be used with.

Unit testing for business layer: A task that was repeatedly postponed, the group would like to incorporate unit tests for each component of the SDK.

Implementing HTTPS for the API: A lack of having implemented HTTPS before and having to do so on an internal network made HTTPS hard to implement. Deploying the SDK on SkyHiGh meant that acquiring a signed certificate would be challenging, and resulted in an attempt to create our own certificate authority and sign a certificate ourselves. Being one step towards the goals means that this feature was close to be implemented.

Token-based authentication for the API: The group also wanted to ensure that only the SDK could access the API. This was not prioritized, but would be implemented had the group had more time.

More robust error handling: Implement more comprehensive error handling and checks for edge cases.

11 Conclusion

In this thesis, we have addressed the pressing issue of predatory behavior in online gaming communities by developing a Software Development Kit (SDK) tailored for Unity-based games to extract and process Soft Biometric Keystroke Dynamics (SBKD) data. Our collaboration with AIBA aimed to enhance the Amanda child protection platform, providing a crucial tool for age and gender profiling to support more effective moderation and intervention strategies.

The primary goal of our project was to create a functional SDK that seamlessly integrates with Unity, facilitating the collection of SBKD data. This SDK serves as a foundational component that AIBA can use to refine their existing moderation tools, thus enabling more accurate detection of conversations with potential predatory intent. By focusing on the development of a keystroke data collection tool, we have laid the groundwork for future advancements in automated moderation and user safety.

Our investigation into the upcoming legal framework revealed that while AIBA has minimal obligations under the DSA, KOSA, and SDK, the EU AI Act will classify the use of SBKD for age profiling as high risk, thus imposing significant responsibilities. Given that the last corrigendum of the EU AI Act is just over a month old as of 21.05.2024, we believe this is the first study examining the placement of SBKD and AI for age profiling usage within the EU AI Act.

We conducted a market analysis to understand the age verification methods employed by major enterprises and gaming services. Our findings indicate that few are utilizing any form of Soft Biometric Keystroke Dynamics. Furthermore, there is a seem to be a major need for enhanced moderation tools in video games, as highlighted by Section 5.2, which found significant levels of toxic behavior towards both minors and adults, where major parts of this goes unreported by the users..

The project's outcomes include not only the SDK itself but also a prototype chat application utilizing Vivox, comprehensive documentation for developers, and detailed guidelines for efficient data transmission to minimize network costs. These deliverables collectively aim to empower Soft Biometric Keystroke Dynamics to enhance it's moderation capabilities, reduce instances of cyber grooming, and ultimately create safer online environments for minors.

In conclusion, our work contributes significantly to the field of online safety and moderation. By providing a robust technical solution and aligning it with ethical and legal standards, we support the ongoing efforts to protect vulnerable users in digital spaces. Future work can build on this foundation to further refine and expand the capabilities of the SDK, ensuring it remains a vital tool in the fight against online predatory behavior.

Bibliography

[1] D. Alanko, 'The health effects of video games in children and adolescents,' en, *Pediatr Rev*, vol. 44, no. 1, pp. 23–32, Jan. 2023.

[2] H.-J. Zuberbühler, H. Krueger and A. Kündig, 'Delay perception thresholds in human-computer interaction. fundamentals for cscw-applications,' en, I. für Hygiene und Arbeitsphysiologie Zürich, Ed., XVII International Annual Occupational Ergonomics and Safety Conference; Conference Location: Munich, Germany; Conference Date: 2003, Zürich: Swiss Federal Institute of Technology Zurich, Institute of Hygiene and Applied Physiology, 2003. DOI: 10.3929/ethz-a-004606991.

[3] A. K. Jain, S. C. Dass and K. Nandakumar, 'Soft biometric traits for personal recognition systems,' in *Biometric Authentication*, D. Zhang and A. K. Jain, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 731–738, ISBN: 978-3-540-25948-0. [Online]. Available: https://biometrics.cse.msu.edu/Publications/SoftBiometrics/JainDassNandakumar_SoftBiometrics_ICBA2004.pdf.

[4] Y. "Deng and Y. Zhong, '"keystroke dynamics user authentication based on gaussian mixture model and deep belief nets",' "*ISRN Signal Processing*", S. Kwong and K. Wang, Eds., 2013. [Online]. Available: https://doi.org/10.1155/2013/565183.

[5] R. Shadman, A. A. Wahab, M. Manno, M. Lukaszewski, D. Hou and F. Hussain, *Keystroke dynamics: Concepts, techniques, and applications*, 2023. arXiv: 2303.04605 [cs.CR].

[6] L. Araújo, L. Sucupira, M. Lizarraga, L. Ling and J. Yabu-uti, 'User authentication through typing biometrics features,' vol. 3072, Jan. 2004, pp. 694–700, ISBN: 978-3-540-22146-3. DOI: 10.1007/978-3-540-25948-0_94. [Online]. Available: https://www.researchgate.net/publication/221215288_User_Authentication_through_Typing_Biometrics_Features.

[7] S. Z. Syed Idrus, E. Cherrier, C. Rosenberger and P. Bours, 'Soft Biometrics for Keystroke Dynamics: Profiling Individuals While Typing Passwords,' *Computers & Security*, p. 1, Jun. 2014. [Online]. Available: https://hal.science/hal-01011801.

[8] P. Bours and H. Barghouthi, 'Continuous authentication with biometric keystroke dynamics,' in *The Norwegian Information Security Conference (NISK)*, vol. 2009, 2009. [Online]. Available: https://sciencegatepub.com/books/gcsr/gcsr_vol2/GCSR_Vol2_Ch3.pdf.

[9] L. C. F. Araújo, L. H. R. Sucupira, M. G. Lizárraga, L. L. Ling and J. B. T. Yabu-uti, 'User authentication through typing biometrics features,' in *Biometric Authentication*, D. Zhang and A. K. Jain, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 694–700, ISBN: 978-3-540-25948-0. [Online].

Available: https://link.springer.com/chapter/10.1007/978-3-540-25948-0_94.

[10] S. Z. S. Idrus, E. Cherrier, C. Rosenberger and P. Bours, 'Soft biometrics for keystroke dynamics,' in *Image Analysis and Recognition*, M. Kamel and A. Campilho, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 11–18, ISBN: 978-3-642-39094-4.

[11] D. Shanmugapriya and G. Padmavathi, 'A survey of biometric keystroke dynamics: Approaches, security and challenges,' *CoRR*, vol. abs/0910.0817, 2009. arXiv: 0910.0817. [Online]. Available: http://arxiv.org/abs/0910.0817.

[12] A. Maas, C. Heather, C. (Do, R. Brandman, D. Koller and A. Ng, 'Offering verified credentials in massive open online courses: Moocs and technology to advance learning and learning research (ubiquity symposium),' *Ubiquity*, vol. 2014, no. May, May 2014. DOI: 10.1145/2591684. [Online]. Available: https://doi.org/10.1145/2591684.

[13] T. Dias, J. Vitorino, E. Maia, O. Sousa and I. Praça, 'Keyrecs: A keystroke dynamics and typing pattern recognition dataset,' *Data in Brief*, vol. 50, p. 109 509, 2023, ISSN: 2352-3409. DOI: https://doi.org/10.1016/j.dib.2023.109509. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352340923006091.

[14] D. Y. Wohn, 'Volunteer moderators in twitch micro communities: How they get involved, the roles they play, and the emotional labor they experience,' in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI '19, Glasgow, Scotland Uk: Association for Computing Machinery, 2019, pp. 1–13, ISBN: 9781450359702. DOI: 10.1145/3290605.3300390. [Online]. Available: https://doi.org/10.1145/3290605.3300390.

[15] Y.-G. Cheong, A. K. Jensen, E. R. Guðnadóttir, B.-C. Bae and J. Togelius, 'Detecting predatory behavior in game chats,' *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 7, no. 3, pp. 220–232, 2015. DOI: 10.1109/TCIAIG.2015.2424932.

[16] R. Kowert and L. Woodwell, *Moderation challenges in digital gaming spaces: Prevalence of offensive behaviors in voice chat*, https://www.takethis.org/wp-content/uploads/2022/12/takethismodulatereport.pdf, A white paper by TakeThis, 2022.

[17] Anti-Defamation League, 'Hate and harassment in online games 2022,' Center for Technology & Society, Tech. Rep., 2022, Accessed: 13.05.2024. [Online]. Available: https://www.adl.org/sites/default/files/documents/2022-12/Hate-and-Harassment-in-Online-Games-120622-v2.pdf.

[18] S. Livingstone, L. Haddon, A. Görzig and K. Ólafsson, 'Risks and safety on the internet: The perspective of european children: Full findings and policy implications from the eu kids online survey of 9-16 year olds and their parents in 25 countries,' EU Kids Online Network, London School of Economics and Political Science, Tech. Rep., 2011. [Online]. Available: https://eprints.lse.ac.uk/33731/1/Risks%20and%20safety%20on%20the%20internet%28lsero%29.pdf.

[19] R. D. Meyer, 'Exploring toxic behaviour in online multiplayer video games,' Ph.D. dissertation, University of York, Department of Computer Science, 2020. [Online]. Available: https://etheses.whiterose.ac.uk/30580/1/Meyer_206059909_CorrectedThesisClean.pdf.

[20] C. o. t. E. U. European Parliament. 'Regulation (eu) 2022/2065 of the european parliament and of the council of 19 october 2022 on a single market for digital services and amending directive 2000/31/ec (digital services act) (text with eea relevance).' (Oct. 2022), [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32022R2065.

[21] European Parliament and Council of the European Union, *REGULATION (EU) 2024/... OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of ... laying down harmonised rules on artificial intelligence and amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act)*, Corrigendum to the position of the European Parliament adopted at first reading on 13 March 2024, with a view to the adoption of Regulation (EU) 2024/... laying down harmonised rules on artificial intelligence and amending various regulations and directives (Artificial Intelligence Act), 2024. [Online]. Available: https://www.europarl.europa.eu/doceo/document/TA-9-2024-0138-FNL-COR01_EN.pdf.

[22] E. Parliament and C. of the European Union. 'Regulation (eu) 2016/679, *on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*.' (Apr. 2016), [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016R0679.

[23] E. Parliament and C. of the European Union. 'Directive (eu) 2016/680, *on the protection of natural persons with regard to the processing of personal data by competent authorities for the purposes of the prevention, investigation, detection or prosecution of criminal offences or the execution of criminal penalties, and on the free movement of such data, and repealing Council Framework Decision 2008/977/JHA*.' (Apr. 2016), [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016L0680.

[24] E. Parliament and C. of the European Union. 'Regulation (eu) 2018/1725, *on the protection of natural persons with regard to the processing of personal data by Union institutions, bodies, offices, and agencies and on the free movement of such data, and repealing Regulation (EC) No 45/2001 and Decision No 1247/2002/EC*.' (Oct. 2018), [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32018R1725.

[25] European Parliament and Council of the European Union. 'Directive (eu) 2016/2102, *on the accessibility of the websites and mobile applications of public sector bodies*.' (Oct. 2016), [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016L2102.

[26] European Parliament and Council of the European Union. 'Directive (eu) 2019/882, *on the accessibility requirements for products and services*.' (Apr. 2019), [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32019L0882.

[27] E. A. Kochegurova and R. P. Zateev, 'Hidden monitoring based on keystroke dynamics in online examination system,' en, *Program. Comput. Softw.*, vol. 48, no. 6, pp. 385–398, Dec. 2022. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9707207/.

A Thesis Description

Oppgave 46 BPROG, BIDATA, DIGSEC 2 stk

Oppgavetittel: Soft Biometic Keystroke Dynamics SKD

Bedrift:	AIBA
Kontaktperson:	Gard Støe
E-post:	gard.aiba.ai
Telefon:	99223535
Lokasjon:	Gjøvik

Beskrivelse av oppgaven

AIBA is a spin-off company from NTNU research and aims at detection of online cybergrooming in game chats as early as possible. This is done both by analyzing the messages in a conversation and by looking at anomalous user behaviour. Besides that, there is a focus on detection of toxic language and other unwanted behaviour, as well as profiling of user's gender and age based on behaviour.

The students should in this project first perform a metastudy or market scan on age verification software in light of the new EU regulations coming (i.e. age verifying kids in games and SoMe). Furtermore, they should develop an SDK that can collect keystroke dynamics data (i.e. typing behaviour information) and send it to a server and a server application where different analysis algorithms can be incorporated for age detection.

For more information you can visit us on Wednesdays in room A108 in the A-building on campus.

B Daily Stand Up Summary from Week 15

🗓️ Wednesday 10.04.2024

	Name	Priorities 🧐	Progress 😊	Problems 😐
1	@Joachim Olerud Milward	• SBKD • Vivox chat	• The password conundrum • Refactored Auth & UI manager	• Tight coupling
2	@nicolai andre olsen	• Data research • Reviews • SBKD	• Data visualizer refactoring	N/A
3	@Vegard Johansen	• SBKD • Reporting • Refactor unity chat application	• Implementation template • SBKD research	N/A

Figure B.1: Sample day from the Daily Stand up notes

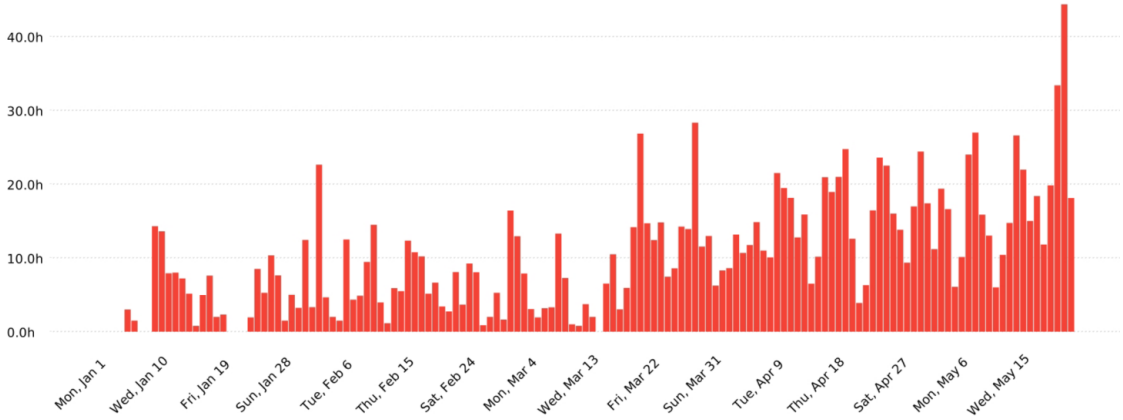
C Clockify summary report

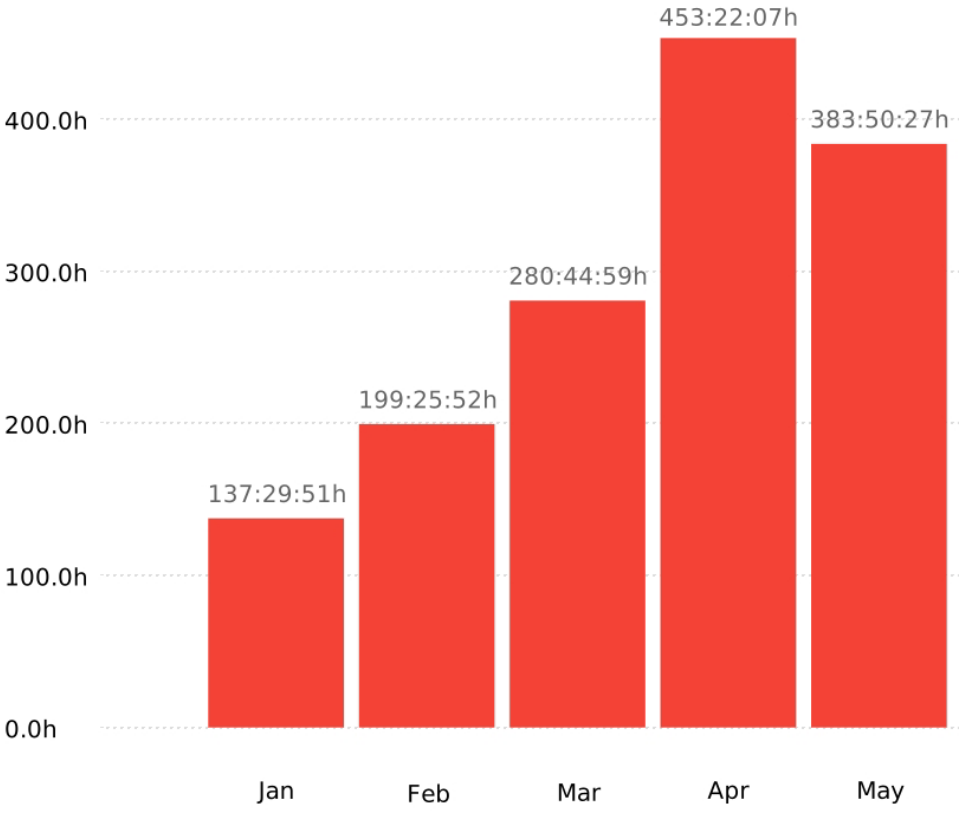
Summary report

clockify

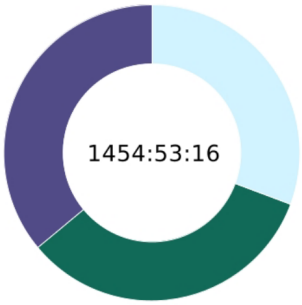
01/01/2024 - 21/05/2024

Total: 1454:53:16



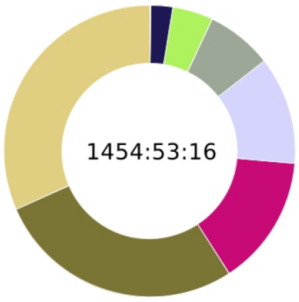


User

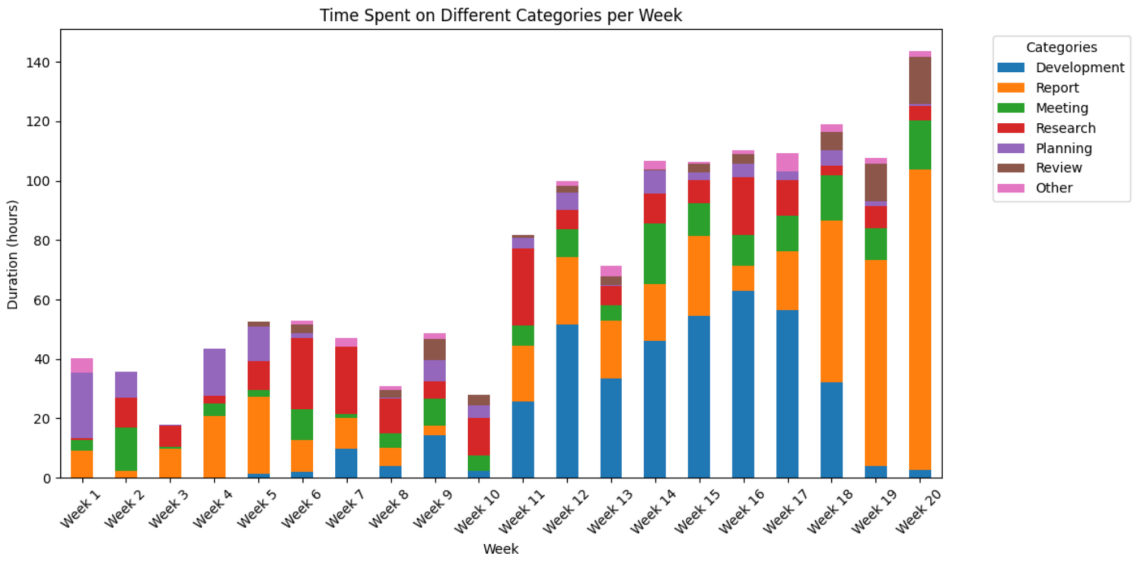


Joa	524:37:24	36.06%
nicolai olsen	484:12:29	33.28%
Vegard Johansen	446:03:23	30.66%

Tag



Report	458:40:11	31.53%
Development	403:07:28	27.71%
Research	209:52:23	14.43%
Meeting	173:14:38	11.91%
Planning	106:54:15	7.35%
Review	65:54:16	4.53%
Other	35:01:46	2.41%
Testing	02:08:19	0.15%



D Client meetings

13.12.2023 | Aiba meeting no. 1

- Aiba is a spinoff company from NTNU that aims to prevent grooming online. They cooperate with police, developers and other entities to achieve this. Aiba are responsible for the analysis part, where they collect text and apply an algorithm to analyze the age of the users. This way, it will be easier to see where there is a large age discrepancy between the users who are chatting together.
- Aiba works mainly with .NET and python.
- As the project will be based on unity, we will probably need to learn C# as that seems to be the most used language in unity
- Our task is to research the possibility of, and possibly implementing software to extract the keystroke biometric data from a chat in unity.
    - This will be used to calculate the age of the users and to search for language that can signal that there is intent to groom a younger person in a game.
- We will need to do research to see what already exists for unity in relation to keystroke dynamics data, and if it is at all possible to extract it from unity
- Identify what devices the program should support.
    - Should phones be included, or just computers?
    - If only computers, should it be limited to only windows, or can we extend it to linux and OSX as well?
- Aiba has old java code we could take a look at to get some inspiration.
- Our task would very simply be; register the time a key is pressed down, then the time before it is released again, and lastly the time to get to the next character. This will be done for the entire message. Afterwards we need to have an efficient stream or batch of data to be sent to the server so that it can be analyzed by Aibas other programs.

12.01.2024 | Aiba meeting no. 2

Deltakere:
Nicolai Andre Olsen
Vegard Johansen
Joachim Olerud Milward
Gard Støe

Notater
Case no. 01 Opening meeting

Case no. 02 Establishing a regular (bi-weekly meeting time)

> 16.01: Holding meetings on Tuesdays every other week from 16:00 to 16:30. Starting the upcoming one (16.01.2024).
> But will look at the possibility of having weekly meetings during the start of the project.

Case no. 03 System development method

> 16.01: Aiba has an agile development method, not religiously adhering to scrum, but runs 2-week sprints.
> Aiba also has a kanban board, implementing 'scrumban'.
> Uses Jira as a tool for task management and documentation

Case no. 04 Documentation standards at Aiba.ai

> 16.01: Aiba has not established any documentation standards.

Case no. 05 “Standard agreement” terms

> 16.01: Gard is arranging a Non-Disclosure Agreement in relation to code and chat data. We will bring the collaboration agreement.

Case no. 06 Side note – NISlab

> 16.01: Gard does not have knowledge about this, but Patrick might know more.

Case no. 07 Curiosity - Possible to look at current processing of chat data?

> 16.01: Possible to gain insight into the chat data.

31.01.2024 | Aiba meeting no. 3

Participants:
Nicolai Andre Olsen
Joachim Olerud Milward
Vegard Johansen
Gard Støe

Notes
Case no. 01 Establishing a regular (bi-weekly meeting time)

> 12.01: Holding meetings on Tuesdays every other week from 16:00 to 16:30. Starting the upcoming one (16.01.2024).
> But will look at the possibility of having weekly meetings during the start of the project.
>
> 16.01: Will try to have physical meetings on wednesdays 10:00.

Case no. 02 “Standard agreement” terms

> 12.01: Gard is arranging a Non-Disclosure Agreement in relation to code and chat data. We will bring the collaboration agreement.
>
> 16.01: will look more at this on wednesday

Case no. 03 Possible to look at the current processing of chat data?

> 12.01: Possible to gain insight into the chat data.
>
> 16.01: Will look more at chat data at the next meeting as that will be physical.

Case no. 04 SDK vs library

> 12.01: Aiba wants the final product to be more of a full package that developers can download from the unity asset store, with limited ability to edit on their own. It should also have more documentation than a regular library. However, also developing a library that could add supplemental elements could be a choice to consider during the end of the development.
> The final product will probably not become a full sdk before the delivery date, but can then supplement with have explanation of what is missing for the product to become an sdk.

Next steps

- Gard will update with potential meeting time on wednesday
- We will take a look at the unity asset store to see what is there of similar assets, and take an initial look on what is needed there to release an asset.
- Send mail to Patrick regarding NISlab
- Gett in touch with the Unity developer for Attensi that was spoken about

31.01.2024 | Aiba meeting no. 4

Participants:
Nicolai Andre Olsen
Joachim Olerud Milward
Vegard Johansen
Patrick Bours

Notes

Case no. 01 “Standard agreement” terms

> 12.01: Gard is arranging a Non-Disclosure Agreement in relation to code and chat data. We will bring the collaboration agreement.
>
> 16.01: Will look more at this on wednesday
>
> 31.01: Will have to send it digitally to Gard as he is in oslo today.

Case no. 02 Possible to look at the current processing of chat data?

> 12.01: Possible to gain insight into the chat data.
>
> 16.01: Will look more at chat data at the next meeting as that will be physical.
>
> 31.01: Patrick did not find the SKBD source code, but will look for it.

Case no. 03 Showing of keystroke data registration

> 31.01: Patrick talked about the different potential solutions on how the SKBD should be saved, and sent, for efficient data transfer.

Case no. 03 AOB

> 31.01: no other buisness

07.02.2024 | Aiba meeting no. 5

Participants:
Nicolai Andre Olsen
Joachim Olerud Milward
Vegard Johansen
Gard Støe

Notes

Case no. 01 Signing of confidentiality agreement

> 07.02: Gared will send the confidentiality agreement after the meeting

Case no. 02 Set a possible date for seeing service in action

> 07.02: Will have a more set date next meeting, with patrick to show both Aiba's current solution, and the datastream they use for the SKBD

Case no. 03 AOB

> 07.02: Gard mentioned that there is a digital service act (DSA) that will go into effect next week, that we will need to do research on.
>
> There is also an AI act for regulating work with AI, which we should take a small look at, and a UK safety bill that should be taken a look at and lastly kids only safety act (KOSA) which relates more to USA.
> Relevant links:
> https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/europe-fit-digital-age/digital-services-act_en
> https://www.gov.uk/government/news/britain-makes-internet-safer-as-online-safety-bill-finished-and-ready-to-become-law
> https://www.europarl.europa.eu/news/en/headlines/society/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence
> https://en.wikipedia.org/wiki/Kids_Online_Safety_Act

Next steps:

- Bring the confidentiality agreement to Sony, to have him take a look at it before signing.
- Look more into the new regulations that are forthcoming

14.02.2024 | Aiba meeting no. 6

Participants:
Nicolai Andre Olsen
Joachim Olerud Milward
Patrick
Gard Støe

Notes

Case no. 01 Signing of confidentiality agreement

07.02: Gared will send the confidentiality agreement after the meeting

Case no. 02 AOB

14.02:

15.03.2024 | Aiba meeting no. 8

Participants:
Nicolai Andre Olsen
Joachim Olerud Milward
Gard Støe

Notes

Case no. 01 confidentiality agreement

07.02: Gared will send the confidentiality agreement after the meeting

15.03: small delays, will be sent to us in not long

Case no. 02 Current progress update/future work. API + calculations, C# keylogger, Unity Sandbox

15.03: Aiba will never create their own chat client, and would realistically try to use existing chat clients to capture the SKBD. Aiba's focus will rather be to focus on the API part.

Case no. 03 Chat service – create own or use one that already exists? Vivox etc. Plans to create your own service in the future?

15.03: Gard thoughts on the implementation would be for the solution to only capture timestamps, duratins, latency and such. Patrick think it might be useful to to calculations on the client side. Also notes that JSON files are big and it might be useful to have it be sent as raw data or such.

Case no. 03 AOB

15.03:

Next steps:

- Preliminary reseach on size difference of different version (csv, json, run in real time, as batches and so on) to send the messages
- Look at output and how that can be handled in relation to the cases generated by Aiba

05.04.2024 | Aiba meeting no. 9

Participants:
Nicolai Andre Olsen
Vegard Johansen
Joachim Olerud Milward
Gard Støe
Patrick Bours

Notes

Case no. 01 Progress update + demo of current prototype

> 05.04: in regards to when to send data: Gard thinks it sounds most logical to send it when the messages are sent to another user.

Case no. 02 Encryption of SBKD captured, any security concerns? Requirement for secure transfer from SDK/API to your servers?

> 05.04: Aiba has not discussed the topic of encryption. Gard thinks that the APi call *should* be encrypted, but it has yet to be discussed. Though it is not seen as necessary for this project, but rather explain it as a *further work* in the report.
>
> Patrick thinks that we should not forget to place a large focus on the *integrity* part of the data, as that might be more relevant for this project
>
> Gard: Focus primarily on getting stuff to work before doing a lot of work in the confidentiality aspect.

Case no. 03 Specifics on metrics that we can calculate in API. Which metrics would be ideal for your case?

> 05.04: The most important fact according to Patrick is the minimization of data sent. e.g., is it possible to calculate duration of press already on user level, before the data is sent to the api. Metric about the data sent is not very relevant for this part of the project.
>
> Gard think a smart step would be to calculate the different payload sizes, to find the smaller possible payload size.
> Patrick: duration accuracy should be ms, with 1 decimal accuracy.

Case no. 04 Would it be possible to use your API for creating a demo video when demonstrating the SDK? I.e. Using a chat service with SDK attached, send data and receive profiling information.

> 05.04: Gard: if it is technically possible, Gard does not see a problem with it.

Case no. 04 AOB

> 05.04: No other comments from Gard.
>
> Gard is satisfied with meetings every other week instead of every week.

Next steps:

- Get Vivox in place
- Talk in the report about how Unity has not been consistent with how the UI is built, and that there are a lot of threads online about how this is a problem for many users.

24.04.2024 | Aiba meeting no. 10

Participants:
Nicolai Andre Olsen
Vegard Johansen
Joachim Olerud Milward
Gard Støe
Patrick Bours

Notes

Case no. 01 Suggestion for plan moving forward. Potential changes?

24.04: Aiba seek no other changes, and are happy with product the way it is now

Case no. 02 Vivox progress

24.04: Aiba very happy with progress in Vivox.

Case no. 03 Data transfer research progress

24.04: Aiba was happy with reserach, and found results very interesting.

Case no. 04 What type of API are you using? REST? GraphQL? Needed for writing deployment section in thesis

24.04: Aiba uses Rest API.

Case no. 04 AOB

24.04:

- Aiba had also experieced sloppy game development documentation
- Aiba very happy with progress up to now, and are willing to read through report to find sections they wish to change

Sprint retrospect 3

📋 Overview

Date	05.04.2024
Team	Group 11
Participants	@Joachim Olerud Milward @Vegard Johansen @nicolai andre olsen

💭 Retrospective

Start doing	Stop doing	Keep doing
• Adding more varied tasks to the sprint • Make more structure around when to send meeting summons and such • Start taking implementation notes	• Nothing of note	• Write report during week • Add story-id to logs and commits

✅ Action items

- [ ] Standardization of meeting summons
- [ ] Sprint story variation
- [ ] Make template for implementation

E Sprint Retrospective

F Initial docker installation guide

1. Make a Dockerfile that copies the project into the image, exposes a port and instantiates server
2. Make a docker-compose YAML file that automates container building
3. Created the Virtual Machine on SkyHiGh where the API would be deployed
4. Install the packages required to run Docker and docker-compose on the VM
5. Transfer the API code using SCP [1].
6. Deploy the API by using the *docker compose up -d* command
7. Make sure to allow TCP traffic for the port specified in the Dockerfile

[1] https://www.commandlinux.com/man-page/man1/scp.1.html

G File Size Analysis: Data Structure and Visualization Methods

G.1 Creation of file types

G.1.1 Data to capture

Figure 4.1 presents the types of information relevant to SBKD. Figure 7.4 illustrates the data selected for transmission. We determined that the two data points shown in Figure 7.4, Up-Down flight time and Down-Up Hold time, are sufficient, as additional data can be derived from these if necessary. This approach ensures that the data transmitted are streamlined, encompassing only the most essential measurements for efficient data processing, and thus transmitting the least amount of data to use as little data as possible.

Thus the data points that we use are as follows:

- Up-down flight time: Stored as milliseconds from when the previous key was released to when the current key was pressed down. For the first key pressed by the in each message, this value is set to 0 milliseconds, as there were no preceding keystrokes.
- Down-up hold time: Recorded as milliseconds from when the current key is pressed down to when it is released.

The remaining data points can be calculated from the two data points that have been used in the following ways:

- Down-up flight time: This is calculated by summing the down-up hold time from the previous key, the up-down flight time from the previous to the current key, and the down-up hold time of the current key.
- Down-down flight time: Calculated by adding the hold time of the previous key to the up-down flight time from the previous key to the current key.
- Up-Up flight time: Calculated by adding together the Up-Down flight time and Down-Up Hold time

Each data point's time was calculated from the start of the arrow to the end of the arrow, as illustrated in Figure 7.4. For instance, the duration of a key entry's down-up hold time is the duration from when the key was pressed down(the start of the blue arrow) to when it was released (the end of the blue arrow). Additionally, each key entry's up-down flight time is recorded as the duration from when the previous key was released (the start of the red arrow) to the pressing of the current key (the end of the red arrow). The rationale for storing each key's data point times in this manner, rather than using a simpler method of recording time from the first key press in a message for all values, is that the chosen method should increase the likelihood of producing duplicate values. This approach enhances the compression benefits, as with a precision of 0.1 ms, these values will rarely exceed five digits (one second). In contrast, using milliseconds from the first key press would cause

values to increase to six digits after just 10 seconds of total typing time, thereby consuming more data space than the chosen method.

G.1.2 CSV

We chose CSV due to its systematic nature and its efficiency in terms of file sizes, which we believed created a good balance between ease of handling and file size. The CSV was structured as in Table G.1.

k	kd	ku	msg	ID
W	0.15	-0.796	We	724d95de-0592-4749-94b6-a339cb002aa5
Shift	0.962	1.148		
e	0.107	0.172		

Table G.1: Example of SBKD in csv format

Here, the first line of data contains information about which user the message belongs to, and what the full message is. The first and subsequent messages also contain key entry data.

G.1.3 txt

The TXT file type was selected because it allows for excluding any information from the file that is not essential. The TXT file was structured in the same manner as Code listing G.1:

Code listing G.1: Example of SBKD in txt format

```
1    724d95de-0592-4749-94b6-a339cb002aa5
2    We
3    W | 0.15 | -0.796
4    Shift | 0.962 | 1.148
5    e | 0.107 | 0.172
```

Here, the first line provides information about the user. The second line presents the full message, while lines 3 and beyond contain the key entry values. The values and keys are separated by "|" to distinguish the values from each other.

G.1.4 JSON

The decision to utilise JSON was based on its extensive utilisation and the simplicity of parsing. The JSON structure was as in Code listing G.2:

Code listing G.2: Example of SBKD in JSON format

```
1    [
2        {
3            "id": "724d95de-0592-4749-94b6-a339cb002aa5",
4            "ks": [
5                {
6                    "k": "W",
7                    "kd": 0.15,
8                    "ku": -0.796
```

```
                },
                {
                    "k": "Shift",
                    "kd": 0.962,
                    "ku": 1.148
                },
                {
                    "k": "e",
                    "kd": 0.107,
                    "ku": 0.172
                }
            ],
            "msg": "We",
            "os": "Windows 10"
        }
    ]
```

In this format, the key entries are stored in a list. This facilitates the extraction of only the key entries and subsequent analysis of their values when required.

G.1.5 Protobuf

Protobuf was selected for its fast serialization and deserialization capabilities, and its ability to significantly reduce the space required in comparison to JSON files. This is due to the fact that it is stored as a binary file instead of a human-readable file. The data structure is given in Code listing G.3:

Code listing G.3: Example of Protobuf datastructure for SBKD capturing

```
message KeyEntry {
    string k = 1;
    double kd = 2;
    double ku = 3;
}

message Session {
    string id = 1;
    string msg = 2;
    repeated KeyEntry entries = 3;
    string os = 4;
}
```

Similar to the JSON file, key entries are stored in a list, which would make it easy to extract these values for further processing.

G.2 Implementation of dataset

To generate key entries for the file size data, we utilized a dataset provided by [13]. To streamline the process and minimise variables, such as special characters (å, ø, æ), we opted for an English dataset to base the data off of.

While the dataset contained over half a million keystrokes, we selected only the first 10,000 for our study. This decision was based on the impracticality and scope limitations associated with processing files containing more than 10,000 keystrokes, which would likely impact user experience due to potentially notice-

able performance implications in application scenarios if the users where to compress files with more than 10,000 keystroke entries. Moreover, accumulating 10,000 keystrokes can be time-consuming, potentially requiring a significant duration for some users to achieve, ranging from half an hour to several days. A review of the Vivox specifications revealed that it has a default limit of 320 bytes per message, implying that collecting 10,000 key entries would necessitate at least 32 messages.

G.2.1 Dataset Parsing Methodology

The dataset was parsed into the various files using the format described above. While the dataset contains all the data depicted in Figure 4.1, we chose to extract only the two instances shown in Figure 7.4. For the parsing process, we utilized Python with the pandas library to read the CSV file of the dataset. The program processed each entry, adding them to different data types—JSON, TXT, CSV, and Protobuf—and subsequently saved a new copy for each format. This process resulted in the creation of a file for every increment of keystrokes up to 10,000 key entries.

G.2.2 Compression Implementation

Once all the files had been generated, they were sent to a specific function within the Python compressor file to create a compressed copy of each file. This was done to facilitate the comparison of space savings achieved by zipping the files across different file types. The compressed files were then stored in a separate folder.

G.2.3 Data Structure Design for Key Entry Analysis

After preparing all the files, we began the next step of creating dictionaries from the lists to facilitate data handling. These dictionaries were stored within another dictionary to in order consolidate all the information. The representation of these dictionaries can be viewed in Code listing G.4. The data necessary for comparing file sizes included the number of inputs each file contained, the space it occupied, and the space it required per key entry. In order to determine the number of key entries, special processing was conducted based on the file type. For JSON, the length of the KS list was checked; in CSV, the number of lines was counted, excluding the header line, similarly for TXT. Since Protobuf is stored as a binary file, the numbers in the file name were used as an indicator of the number of inputs. The file sizes were determined using the 'os' library to check the file sizes. The 'size per entry' was calculated by dividing the total file size by the number of entries, thereby determining the space each entry occupied in each file. The final structure was as follows:

Code listing G.4: Structure for Python dictionary

```
1    data = {
2      'txt': {
3        'num_inputs':         [1, 2, ..., 9999, 10000],
4        'zipped_size':        [58, 81, ..., 164, 173],
5        'size_per_entry':     [88, 106, ..., 24, 24],
```

```
        'zip_size_per_entry': [88.0, 53.0, ..., 18, 17]
      },
      'csv': {
        similar structure as txt
      },
      'json': {
        similar structure as txt
      },
      'protobuf': {
        similar structure as txt
      }
    }
```

This structure allowed for the easy retrieval of the necessary data for the subsequent generation of graphs.

G.2.4 Data Storage with Pickle Module

Once the data variable was fully constructed, it was saved using the pickle module to facilitate future use without the need to reconstruct the lists from scratch. It was decided not to save the entire dataset as a single file. Instead, it was segmented and saved as separate files for each data type, thereby simplifying the selection process for future analyses and allowing for easier choice among the available file types.

G.2.5 Graph Generation Using Matplotlib and Plotly Express

Once all the data had been processed, it was visualized using Matplotlib and Plotly Express. Plotly Express was utilized for live presentations to the client due to its interactivity, including zoom capabilities and the ability to display values at the cursor's nearest graph point. For the report, we employed Matplotlib, as it offered superior adaptability for integration into a PDF file. Four different graphs were selected for display, each presenting all data types.Two sets of graphs were created, each including versions for uncompressed and compressed files. The first set illustrates the total file size in relation to the number of key entries, while the second set shows the file size per key entry, indicating the space each key occupies.

H Market Scan - Social media and games

This document provides the basis of information on social media and games studied in the market scan. The analysis regarding existing SBKD verification solutions are not included in this document but can be found in Section 5.1.3.

H.1 Social media

H.1.1 Meta - Instagram, Facebook

- For both Facebook AND Instagram - age must exceed 13 y/o to create and use account
- On account creation through an age screen, must specify age, will be locked out if age is less than 13[1]. Is however easy to maneuver past, can specify faulty age when signing up.
- Currently apply a report mechanic, allowing users to report underage users and have content reviewers trained to flag possible underage individuals
- May require user to show ID to determine whether a users age exceeds a certain number[2], but not all users have IDs (varies according to nationality)
- Current: Using AI to determine whether someone is above 18 or not. Check birthday messages, posts and other accounts registered under same handle[1].
- Uses the above AI to stop adults from messaging young people that don't follow them. Will not show public posts by young people to adults who have shown suspicious behaviour
- New potential method, work with OS providers to share information regarding age to apps
- Want to create "own platforms" for underage people that has a similar experience, but can be parental controlled
- In 2022: using video selfie and age vouching (currently in test phase), partnering with Yoti[3] that has a dataset of peoples faces of different age. Was planned to be made globally available in 2 months from March 2024.
- As of January 2024: Start to hide more content from teens on Instagram, placing in restrictive content control
- Speculative: Potentially also captures keystrokes on mobile devices
- Other exciting links include:
    - Meta child protection
    - Facebook help center

[1] https://about.fb.com/news/2021/07/age-verification/

[2] https://www.meta.com/nb-no/help/quest/articles/accounts/privacy-information-and-settings/id-verification-meta-accounts/

[3] https://about.fb.com/news/2022/06/new-ways-to-verify-age-on-instagram/

H.1.2 Youtube

- 13 minimum age if country rules do not specify otherwise. Can create an account with parental link to enable underage accounts
- Age restriction of videos that have inappropriate content for children, and ads are 18+[4]
- Launched YT kids to enable underage users to use youtube, also has parental controls[5]
- New verification step after EUs AVMSD directive, some european users may be asked to provide ID/credit card if unable to detect age
- Community content reporting, reviewed by moderators[6]
- Other interesting links:
    - Google age requirements description and compliance
    - Youtube's suggested youth policy

H.1.3 Tiktok

- Must be 13 years or older to utilize service
- Tiktok will ban any account found to be used by users below 13 years of age. A user can then appeal the decision by using any of these methods (depends on where they are located)[7]:
    - Selfie with ID
    - Photo with parent/guardian (age 13-17)
    - Credit card authorization (if above 18)
    - Facial age estimation (uses Yoti most likely)[8] (broken url, but: "Yoti only uses your selfie and face information to estimate your age and to judge whether the image is of a live person or not. Your selfie and face information aren't used to identify you or for any other purpose. Yoti doesn't share your face information with TikTok or third parties.")
- Prior to 2022, recorded keystrokes[9]. Used to **verify the authenticity of an account**, for risk control, debugging, troubleshooting, and monitoring for proper performance. Never tracked which key was pressed, only that a key event happened. Currently collects keystroke information and patterns[10]
- Privacy policy for younger users

[4] https://blog.youtube/news-and-events/using-technology-more-consistently-apply-age-restrictions/

[5] https://www.youtube.com/kids/

[6] https://support.google.com/youtube/answer/2802027?hl=en&co=GENIE.Platform%3DAndroid

[7] https://support.tiktok.com/en/safety-hc/account-and-user-safety/underage-appeals-on-tiktok#2

[8] https://support.tiktok.com/en/account-and-privacy/personalized-ads-and-data/how-we-process-face-and-voice-information

[9] https://newsroom.tiktok.com/en-us/tiktok-truths-a-new-series-on-our-privacy-and-data-security-practices

[10] https://www.tiktok.com/legal/page/eea/privacy-policy/en

H.1.4 Snapchat

- At least 13 years old to use[11], will delete account if familiar that the user is younger than 13 no hesitation
- Limit collection of data of user below 18 years old
- In-app reporting[12]
- Parental controls - family center[13]

H.2 Games

H.2.1 Steam

- Minimum age for collection of user data and to create a steam account is 13[14]
- When viewing games that have an age restriction of 18+, the user has to state their DOB, but can be easily faked and reentered
- Community based reporting
- Junior mode accounts, limit accessibility to certain games, communication with other users and visibility to other users[15]

H.2.2 Epic games

- Users that create an account below the age of 13, get a so-called *Cabined account*. Here they must wait until they are of digital consent age, or until parents grant them access to certain services that collect personal data[16]
- Cabined accounts may still play games created by Epic.
- This only applies to accounts that are SPECIFIED to be Cabined
- Has partnered with KWS (Kids Web Services, owned by Epic) as a verification platform, they provide:
    - Credit card verification - Stripe/KWS
    - Cross referencing social security number - only in US
    - ID card/passport - Veratad/Veriff
    - Face scan - Yoti

H.2.3 Riot

- Age limit is 16 across EU, unless country specifies otherwise[17]
- It is possible to play the game underage if the date of birth specified is

[11] https://values.snap.com/privacy/privacy-policy?lang=en-US

[12] https://values.snap.com/safety/safety-enforcement?utm_source=values_snap_com&utm_medium=referral&utm_campaign=universal_navigation&utm_content=footer_item_link&lang=en-US

[13] https://parents.snapchat.com/parental-controls

[14] https://store.steampowered.com/privacy_agreement/?snr=100601_44_44_

[15] https://store.steampowered.com/eula/471710_eula_0

[16] https://www.epicgames.com/site/en-US/parental-consent

[17] https://support-leagueoflegends.riotgames.com/hc/en-us/articles/20590570974483-Parents-of-Underage-Players-FAQ-and-Contact

wrongly stated by the underage user[18]. There are in other words no mechanisms involved except for community reporting of user that may be underage.
- Suggest parents monitor their child's activities online

H.2.4 Roblox

- Has content for all ages. They define the different age requirements and label each game/part as rated
- They support:
  - ID verification - Persona[19]
  - ID scan + selfie - Veriff, Persona[20]
  - 17+ ONLY AVAILABLE IF AGE IS VERIFIED[21]
- Community reporting of abuse[22]
- Monitor speech and text written when under 13 years old, quickly catch violations and train models that detect this content [23]
- Parental controls[24]

[18] https://support-leagueoflegends.riotgames.com/hc/en-us/articles/360001192567-Parental-Approval-Information-for-Parents

[19] https://en.help.roblox.com/hc/en-us/articles/4407276151188-Age-ID-Verification-FAQs

[20] https://en.help.roblox.com/hc/en-us/articles/4412863575316-Roblox-Biometric-Privacy-Notice

[21] https://en.help.roblox.com/hc/en-us/articles/8862768451604-Experience-Guidelines-and-Age-Recommendations

[22] https://en.help.roblox.com/hc/en-us/articles/203313410-Roblox-Community-Standards

[23] https://en.help.roblox.com/hc/en-us/articles/115004630823-Roblox-Privacy-and-Cookie-Policy

[24] https://en.help.roblox.com/hc/en-us/articles/8863284850196-Allowed-Experiences-Controls

I Type dependency diagram for SDK

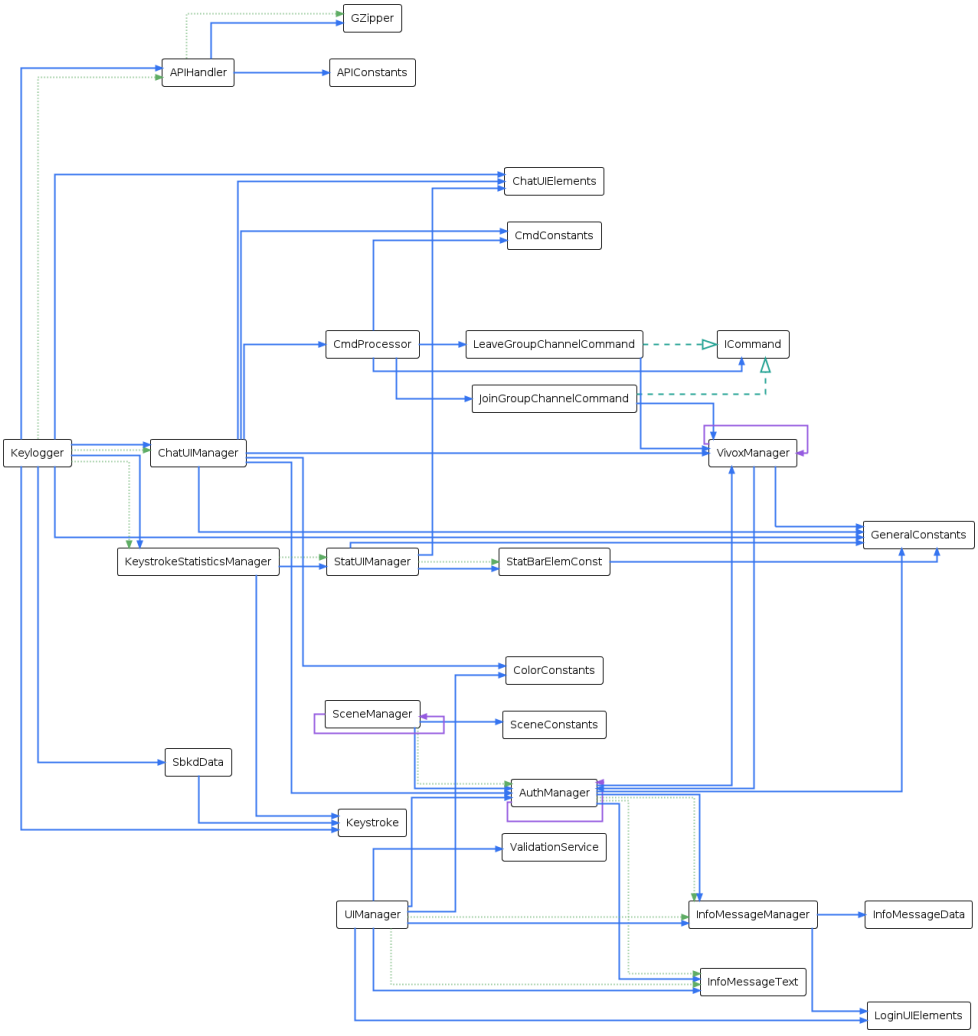


Figure I.1: Autogenerated type dependency diagram for SDK

J Project Plan

Project Plan

Enhancing Chat Moderation with Soft Biometric Keystroke Dynamics

Authors:
Vegrad Jonhansen
Nicolai Andre Olsen
Joachim Olerud Milward

January 2024

Contents

1. Goals and Framework

1.1 Background

AIBA, a spin-off company originating from NTNU research, is dedicated to combating online cyber-grooming and other unwanted behavior. Their primary market target is within game chats, with a focus on early detection of such behaviour. The company's mission encompasses the analysis of conversational messages and the identification of anomalous user behaviors. Moreover, AIBA aims to detect toxic language and other undesirable behaviors while also profiling users based on gender and age-related behaviors.

To align AIBA with the emerging regulatory frameworks, the project mandates an initial meta-study or market scan of age verification software, especially considering the forthcoming EU regulations concerning age verification in gaming and social media platforms. This preparatory phase sets the stage for informed decision-making regarding age verification strategies tailored to the evolving regulatory landscape.

Central to the project's technical objectives is the development of a Software Development Kit (SDK) capable of gathering soft biometric keystroke dynamics data, offering insights into users' typing behaviors. This SDK facilitates the transmission of collected data to a server application, wherein AIBA's existing analysis algorithms can be integrated for age detection purposes. By request from our contact at AIBA, our main focus would be to target the Unity framework as that has widespread use in many of today's popular games.

Given the increasing prevalence of cyber threats and the imperative to safeguard online communities, AIBA's initiative represents a proactive response to the challenges posed by cyber-grooming and related online misconduct. By combining innovative technological solutions with a proactive regulatory approach, AIBA seeks to contribute to the creation of safer digital environments for users of all ages.

1.2 Project Goals

1.2.1 Learning Goals

Throughout this project, our primary objective is to deepen our understanding of the SDLC. We aim to explore the development of SDKs, understanding their usage and the requirements for creating one. A significant focus will be on Soft Biometic Keystroke Dynamics, particularly in learning how it is performed, what data is necessary for effective profiling, how to acquire this data from unity chat, and how to optimize data collection to reduce transmission load without compromising analytical effectiveness.

Working within the Scrum framework, we anticipate gaining valuable insights into this process model. Our specific learning goals include:

- Understanding the nature of SDKs and their practical implementation.
- Capturing keystroke biometric data in unknown environments.
- Acquiring basic skills in programming within the Unity framework, including suitable programming languages and implementation principles.
- Managing chat message data streams in Unity and optimal methods for transmitting this data to external servers.
- Recognizing and addressing the challenges of scaling software projects.
- Effective implementation of the Scrum framework in our project management.

- Enhancing our existing project management skills.

1.2.2 Result Goals

By the conclusion of this project, we hope to deliver a fully functional SDK within the Unity framework. This SDK should efficiently capture keystroke dynamics, send data to AIBA with minimal overhead, and provide comprehensive profiling of users in gaming chat environments. Specific goals include:

- Developing and integrating a functional SDK into Unity to capture keystroke dynamics and analyze chat messages for cyber-grooming indicators.
- Ensuring the SDK adheres to EU regulations and other applicable data privacy laws.
- Successfully completing the assignment and constructing the proposed solution.
- Innovating within the field of Soft Biometic Keystroke Dynamics in the sense of figuring out more ways to make capturing and transfer this data more efficient (reduced data transfer cost).

1.2.3 Effect goals

- Enhancing the capabilities of chat moderators in detecting cyber grooming by giving them warning when users might be involved in cyber grooming.
- Reducing the prevalence of children falling victim to cyber grooming by giving chat moderators the ability to step in before a potential escalation have occured.
- Lowering the costs associated with chat moderation in Unity-based video games by pinpointing the game chats that should be more closely looked at.
- Contributing to the field of Soft Keystroke Biometric Data by enabling its capture in Unity game chats.

2. Project Scope

2.1 Problem Field

One of the concerning aspects of children playing multiplayer games is the difficulty in monitoring their interactions. A seemingly harmless friendship formed over gaming could, in reality, involve a predatory adult masquerading as a peer. This project aims to prevent such scenarios. While it's relatively easy for someone to falsely declare their age, manipulating biometric keystroke data to appear younger is significantly harder. Our program is designed to identify potential grooming situations by analyzing users' biometric keystroke data. Using AIBA's algorithm, it should determine the likelihood of an interaction involving a minor and an adult, and should be able to alert moderators or administrators to potential risks.

2.2 Problem Limitation

This project will investigate the feasibility of implementing an SDK to capture soft keystroke biometric data within the Unity framework. Furthermore, the aim is to integrate this SDK with an existing algorithm developed by AIBA. The main focus lies in establishing communication between our program and the algorithm without modifying its core functionality, as it falls outside the scope of our project.

The group faces a time constraint due to our commitment to the INGG2300 course, which spans three full working days until March 13, 2024. Consequently, we will have only two working days for the bachelor thesis per week for the first two months. However, once the course concludes, we will dedicate our full attention to the project, albeit with a slower start than anticipated.

2.3 Problem Description

A prevalent method today for detecting grooming and unwanted behavior in online platforms with chat functionality involves manual human moderation or the use of algorithms analyzing chat texts for potential grooming signs. These methods have significant drawbacks. Human moderation requires substantial manpower and can be inefficient, while algorithms must be continually updated to keep pace with the evolving nuances of language. What an algorithm could detect a decade ago might no longer be sufficient to identify grooming in today's context. Furthermore, groomers might intentionally alter their typing style to appear younger, or they might employ more covert tactics in their grooming, making detection even more challenging for these algorithms.

Implementing an SDK to capture users' ages offers a significant improvement in this area. It allows moderators to efficiently allocate their time and attention, moving beyond guesswork and imperfect algorithms. Unlike text analysis, which can be fooled by language manipulation, age detection through biometric data is less susceptible to such deception. Even if a user employs language typically used by children, their biometric patterns might not match those of a child. This clearer indicator of potential risk enables moderators to respond promptly to alarms and focus on conversations that genuinely require scrutiny. Consequently, this more streamlined approach reduces the burden of sifting through a large volume of harmless interactions, allowing moderators to concentrate on genuinely concerning cases.

3. Project Organisation

3.1 Project Roles and Responsibility Management

- **Project Manager (Vegard Johansen)**: The Project Manager is the key leader of the project, focusing on efficient execution, strict adherence to schedules, and maintaining budget control. Responsibilities include:
  - *Planning and Coordination*: Coordinates meeting times, topics, and general communication with Sony, and AIBA.
  - *Central point of contact*: acts as a central point of contact for project-related inquiries, providing guidance, support, and leadership to the project team and external actors.
  - *Risk Management*: Identifying and addressing potential risks with strategies to minimize impact and ensure project completion.
  - *Communication*: Ensuring effective communication within the team and with stakeholders, maintaining transparency and clarity.
  - *Quality Assurance*: Upholding high standards throughout the project, with regular evaluations to guarantee that outcomes align with set criteria.
  - *Final Say*: Will have the final say if there are disagreements where the team are split two or three ways, unable to come to an amicable agreement between each other.
- **Scrum Master and Report Product Owner (Nicolai Olsen)**: This role combines Scrum leadership with report management, ensuring smooth sprint operations and comprehensive reporting. Responsibilities include:
  - *Sprint Management*: Overseeing Scrum ceremonies like Sprint Planning, Daily Stand-ups, Reviews, and Retrospectives.
  - *Meeting Notes*: Documenting discussions and decisions in all meetings for future reference and action plans.
  - *Facilitating Product Owner*: Assisting in backlog prioritization and planning backlog enhancements.
  - *Removal of Hindrances*: Proactively addressing issues impeding developers' sprint progress.
  - *Report Backlog*: Developing, maintaining, and prioritizing the report backlog for developer engagement.
- **Program Product Owner (Joachim Olerud Milward)**: The Product Owner is pivotal in final product delivery and backlog management. Key responsibilities are:
  - *Product Backlog*: Developing, maintaining, and prioritizing the product backlog for developer engagement.
- **Developers (All Members)**: Each team member functions as a developer, balancing this with their respective roles. This dual responsibility is feasible due to:
  - The project's scale, which allows for effective management of development tasks alongside other roles without significant time conflicts.

To get a better picture of the different roles, they are depicted as an organizational chart below:

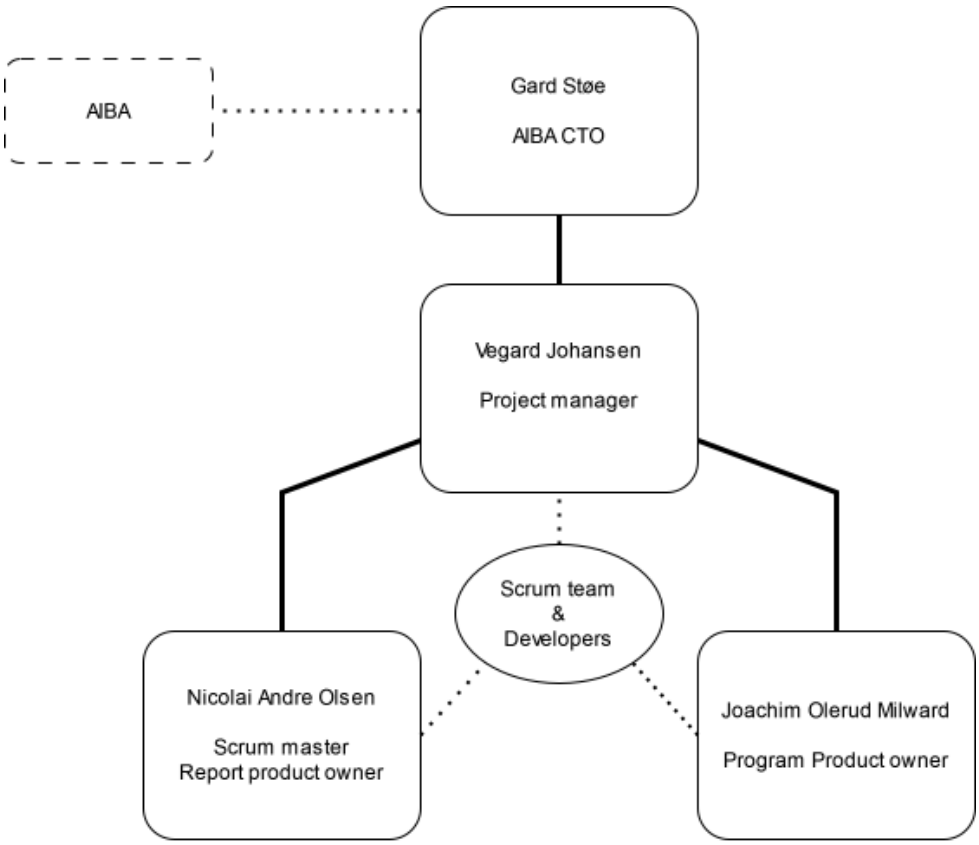


Figure 3.1: Organizational chart for the group

3.2 Routines and Group Rules

For the group rules and routines, the group has created a group contract.

For the sake of simplicity, the group rules and routines are summarized below:

3.2.1 Group rules

Teamwork rules:

- Said work time will be tracked using *Clockify* at the end of every work day.
- Planning of meetings will happen on every Monday the same week they are held.
- Sprints will begin and end each Wednesday the weeks they are scheduled

- Each team member will attend every meeting with supervisor and client, unless a valid reason is given not to do so.
- If for some reason a group member does not perform as defined by the work rules, this would be a breach of conduct. If the breach is reoccurring, the appropriate consequence will be employed (3.2.2)

Work rules:

- Complete the tasks that you have said to complete.
- Don't give up easily. If the tasks requires cooperation, this should be explained to other group members as fast as possible to ensure progress.
- Each group member will prioritize the bachelors project and should give their all.
- Each deadline will be held, and each group members is responsible for completing their part in doing so.
- As the group is aiming for a good grade on the thesis, they should have high expectations to one another and openly communicate it.
- A meeting agenda will be sent out to the respective participants at least 12 hours before the meeting takes place.
- Meetings can be rescheduled upon request or other valid reasons.
- The group will follow the best practices for Unity development defined in organisation and quality assurance 5.
- Complete tasks at minimum 2 days prior to internal deadlines.

3.2.2 Routines

Routine for solving conflicts

Ideally, all conflicts, no matter how major, are to be resolved at the lowest level possible. If the conflict cannot be resolved through discussion alone, the matter will be resolved by bringing in an external person. The following steps will be followed if a conflict breaks out between the group members:

1. If a person does not deliver as expected, the member will be given either a written or an oral warning what the rest of the group felt was done wrong, how it hindered progress and what could be done in the future to avoid similar situations. The group should always, when giving constructive criticism, state it this way.
2. Internal discussion within the group. We should act like professionals and grown-ups, and should thus be able to resolve whatever is hindering teamwork.
3. If the matter is not easily resolved, is reoccurring or not improved upon after internal discussions, a conversation with the group and supervisor will be scheduled.
4. Group discussion with bachelor thesis supervisor Tom Røise if none of the above has any effect.

Development routines

Our development routines are based on the development routines used in Opphus et al., 2017 [1], as we found these to work well for us as well
The development process adheres to the following guidelines to ensure efficiency and clarity:

- All issues must be clearly defined and documented within the Jira backlog.
- Estimation of completion time for all issues will utilize story points.
- Every commit must include the corresponding issue ID from Jira to maintain traceability.

- Developers are encouraged to commit their changes at least once every hour, with a minimum expectation of one commit per day.
- Commit messages must be concise, clear, and informative to assist other developers in understanding the changes.
- Naming conventions for functions, variables, and similar constructs must be descriptive and adhere to the best practices of the specific programming language used.

Workflow in Jira

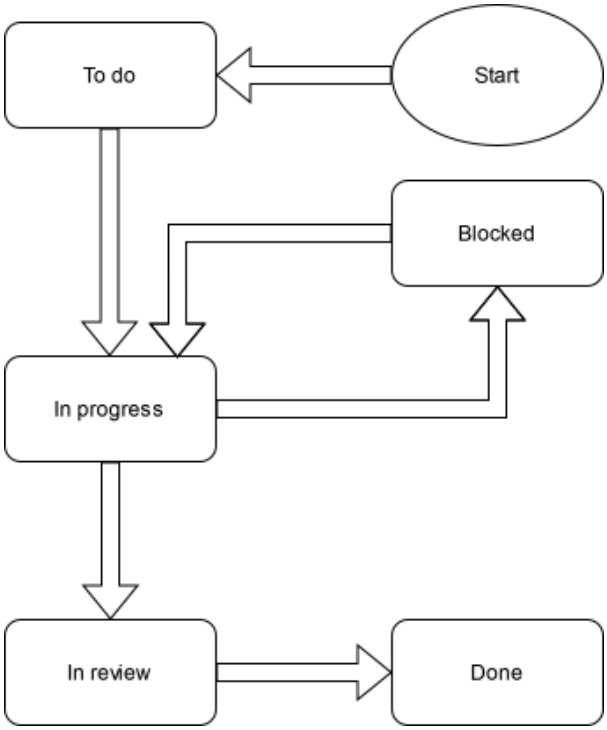


Figure 3.2: Jira backlog board workflow

Table 3.1: Workflow descriptors

Status	Description
To do	Non started issues
In progress	Issues that have been started on, and assigned to at least one team member
Blocked	Issues that have something hindering them from being completed
In review	Issues that are done, and waiting for quality assurance from another team member
Done	Issues that are checked and completed.

4. Planning, Follow-up and Reporting

4.1 Process framework

The group has decided upon the use of the scrum process framework. Scrum is in short a framework that aids in generating value through adaptive solutions for complex problems [2].

4.1.1 Choice of process framework

Our decision to adopt Scrum as our process framework was driven by several compelling reasons, primarily its widespread acceptance in software development. This not only equips us with industry-relevant knowledge and practices, but Scrum also offers the flexibility to adapt our approaches quickly. Scrum's iterative nature enables us to refine our focus with each sprint, allowing for real-time adjustments based on ongoing experiences and findings. Additionally, its accessibility, supported by extensive resources gives the scrum master all resources needed to ensure that the entire team is familiar with the scrum workflow. The Scrum framework simplifies setting tangible goals and milestones, with each sprint serving as a clear target, enhancing visibility on progress, and enabling prompt corrective actions if necessary.

Moreover, Scrum's emphasis on continuous learning and improvement is invaluable. It fosters a culture of reflection and adaptation, ensuring that the team grows more efficient and cohesive over time. The backlog board further streamlines task management, providing clarity on responsibilities and pending tasks. Particularly relevant to our project is Scrum's flexibility in accommodating ongoing research and implementation phases, permitting us to revisit and revise strategies as needed—an advantage not readily available with more rigid methodologies like the waterfall model. This dynamic approach is crucial for navigating the complexities of our thesis work, ensuring that we can pivot effectively when faced with challenges.

4.2 Practical use of model

The Sprint Planning session will be initiated by the Scrum Master with help of the Product Owners. Developers are tasked with organizing their work on the product backlog board, ensuring that tasks they are responsible for are placed on the appropriate board. Once the sprint commences, daily 15-minute standup meetings will be held where each team member discusses their progress, any challenges faced, and any obstacles hindering their work. The Scrum Master will actively address these issues to ensure developers can proceed with minimal disruptions. At the sprint's conclusion, a Sprint Retrospective meeting will take place. Here, team members will reflect on the sprint's successes and failures, providing valuable insights to improve future sprints.

4.3 Meeting Plans

Meetings will take place regularly, respectively with supervisor Sony weekly on Thursdays at 13:00 and client AIBA on Wednesdays 10:00-10:30 biweekly.

For each meeting, an agenda will be proposed and a meeting summon will be sent out to each party attending the meeting. This will enable participants to prepare, thus making the meeting as efficient as possible. Talking points will be prepared and decided amongst the group members on Mondays the same week the meetings are to be conducted.

The meetings with the supervisor of the group, Sony, will mainly be spent updating the supervisor regarding the current standing in the project and receive useful feedback. Concurrently, these meetings will facilitate tracking progress; to make sure that the group does not fall behind schedule.

Furthermore, the client meetings are mainly spent to gain more insight in solving the task. This will be done through the client answering any questions that may arise during the span of the project. Additionally, these meetings will be used to update the client on progression. In doing this, the client can ensure the quality of work is up to par, and that the group stays within the scope of the project.

Meetings may be cancelled if there is just cause 3.2.

4.4 Meeting Minutes

During each meeting, a single group member will note what is discussed. Later, these notes will be revised and formatted more appropriately after the meeting has finished. All group members will be present during the approval of meeting minutes. This will aid in both recalling meetings and giving an insight into the meeting if an individual is absent.

5. Organisation of Quality Assurance

5.1 Documentation, Storage and Source Code

5.1.1 Documentation

Documentation will serve as a way of tracking the project progression. Each meeting will be documented in the form of an agenda and meeting minutes. A meeting in this instance refers to a meeting between the group and either the client or the supervisor.

Documentation and commenting of code will be standardized. The client does not at the time of writing have a documentation standard for their source code. The group members will follow the standards proposed by Microsoft [3] when commenting C# code. *XML Documentation Comments* will be used when commenting source code. This also allows for generation of API documentation through third party libraries similar to JavaDoc [4].

The agendas, meeting minutes, diagrams and other files that the code is not dependent on, will be stored on Google Disk 5.1. This allows for easy access, concurrent editing and sharing of files between the group members. Additionally, the risk of losing files will decrease as the files are stored in more than one place. Other files, such as the README.md which includes a description of what is included in the repository, main aspects of the program and an installation guide is located in the GitHub repository.

5.1.2 Storage

For storage of source code and for version control, GitHub will be utilized. This enables the group to save the code and files of relevance in two places; locally and on the cloud. The group also benefits from this by being able to make changes locally and committing and pushing the changes to a central repository.

5.1.3 Source code

To ensure quality of code, the group will devise a Github actions pipeline which will run every time a group member pushes a change to the repository. This action has the purpose of checking if the code is of the appropriate quality. This also allows for an approach that minimizes human error by automating the linting and testing process the code will undergo after each change. The tools the group will employ for this purpose are:

- Dotnet format [5]
- Standard .NET compiler platform, Roslyn analyzers [6] (included in platform)

For this project, we will program in C# while using the Unity framework. As most, if not all applications in Unity are developed in C#, this was the logical choice. Unity also has extensive support for C#. In terms of the IDE the group will use, this will likely be Visual studio or Rider. Considering NTNU offers student licences for Jetbrains Rider, this is likely the best candidate for the nature of the task. The group has the most experience, and is most familiar with the Jetbrains IDEs. However, the Unity framework does have seamless integration with Microsoft visual studio, so the choice of IDE will be based on preference alone.

The group will follow the best practices for Unity development [7].

5.1.4 Testing

In the starting phase of development, most of the software testing will be testing of functionality, such as unit testing and integration testing. After parts of the SDK are developed and intended functionality is ensured through unit testing, integration tests will be created to ensure cooperation between the SDK and the server receiving data. Finally, user testing or expert evaluations will be conducted to improve efficiency of, for instance, data transfer, program performance, or functionality.

5.2 Data Management

The data sent from the SDK will be stored on a Openstack server, which will be provisioned by NTNU 5.1 through SkyHiGh. This will only be in the starting phase of development, however, as the data transfer will be routed to AIBA's servers at a later stage when the implementation is more refined and thoroughly tested.

5.3 Tools

Purpose	**Name**	**Description**
LaTeX-editor	Overleaf	Online LaTeX-editor used for report writing
Timesheets	Clockify	We used a time tracking applet for recording hours worked
Version control	GitHub	GitHub for source control and storage of source code. Will also use GitHub actions for automating processes when code is pushed to the repository
Communication	Discord, Teams and Slack	Discord for communicating with the bachelor group, Microsoft Teams for the supervisor and Slack for AIBA
File storage	Google drive	We have a Google drive directory with all files
Scrum tool	Jira	Provides backlog board, and insights on how the sprints are going.
Receiver server	OpenStack	Infrastructure for receiving keystroke data
Gantt chart	Instagantt	Online Gantt chart maker

Table 5.1: Project management tools

5.4 Risk Analysis

The risk assessment have a probability, prob. for short, and an impact rating.

Both the impact and probability rating is ranked from 1 to 5.
The probability rating is as follows:
1 - highly Unlikely, 2 - unlikely, 3 - possible, 4 - likely, 5 - very likely.

The impact rating is as follows:
1 - insignificant impact, 2 - minor impact, 3 - moderate impact, 4 - major impact, 5 - catastrophic impact.

Table 5.2: Project Risk Assessment

Risk	Description	prob.	Impact	Consequence	Mitigation
R1	Illness in one group member	3	2	Workload on other team members will be higher	Continue work as normal
R2	Illness in more than one group member	2	4	Progress will probably be slowed considerably	Work on high priority tasks
R3	Project not done by deadline	2	5	Lower grade, possibility of failing the course	Keeping internal deadlines with due dates well ahead of course deadlines
R4	Unable to capture soft keystroke biometrics data (SKBD) in Unity due to it not being technically feasible	2	3	We would have to replan how we structure our report	Pivot report to focus more on the research side of the project
R5	Loss of report or source code	1	5	If it is not recoverable, we would have to start over	Connect Overleaf to GitHub and periodically save a copy locally
R6	Existing solution already exists	2	1	It could be hard to bring innovation with our project	Look into the possibility of adding innovation to the existing solution, or using another approach
R7	Problem with having SDK connect to AIBA's servers	2	3	Unable to properly test that the SDK is working as intended, leading to an unfinished SDK	Have a meeting with AIBA where we discuss solutions. Try to set up our own test environment
R8	Project is in Breach of GDPR	3	2	We are not in control of how AIBA handles their data. This is more of an ethical consideration on our part	Talk with our supervisor on how to move forward. Include a section in our report
R9	Project stretching beyond scope	4	3	Adding too many or unnecessary elements to our project will impact the time we have for producing the report	Follow the plan and only add further work when reaching our goals
R10	Minor Group conflict	5	2	Has the potential to create tension in the group which could hinder the group dynamic and impact our progress	Disputes or disagreements will be voiced when encountered and discussed until a solution is found

Continued on next page

Table 5.2 – *Continued from previous page*

Risk	Description	prob.	Impact	Consequence	Mitigation
R11	Major Group conflict	2	4	Can create serious delays in our progress	If we are not able to solve our disputes in the group we will seek outside guidance. See project plan → Group rules3.2.1
R12	Unexpected amount of time spent on certain parts of project	4	3	Too much time spent on one part could impact our progress in other parts of the project	Use Jira tools to catch it early when some issues have been in the "doing" board for too long. Work to find a solution before too much time has been sunk on these issues

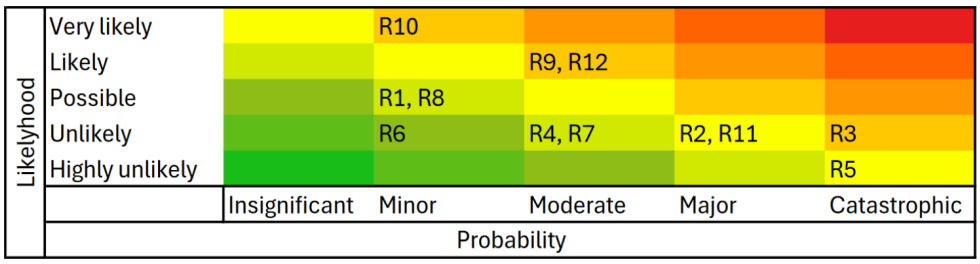


Figure 5.1: Risk factors visualized

6. Plan of Execution

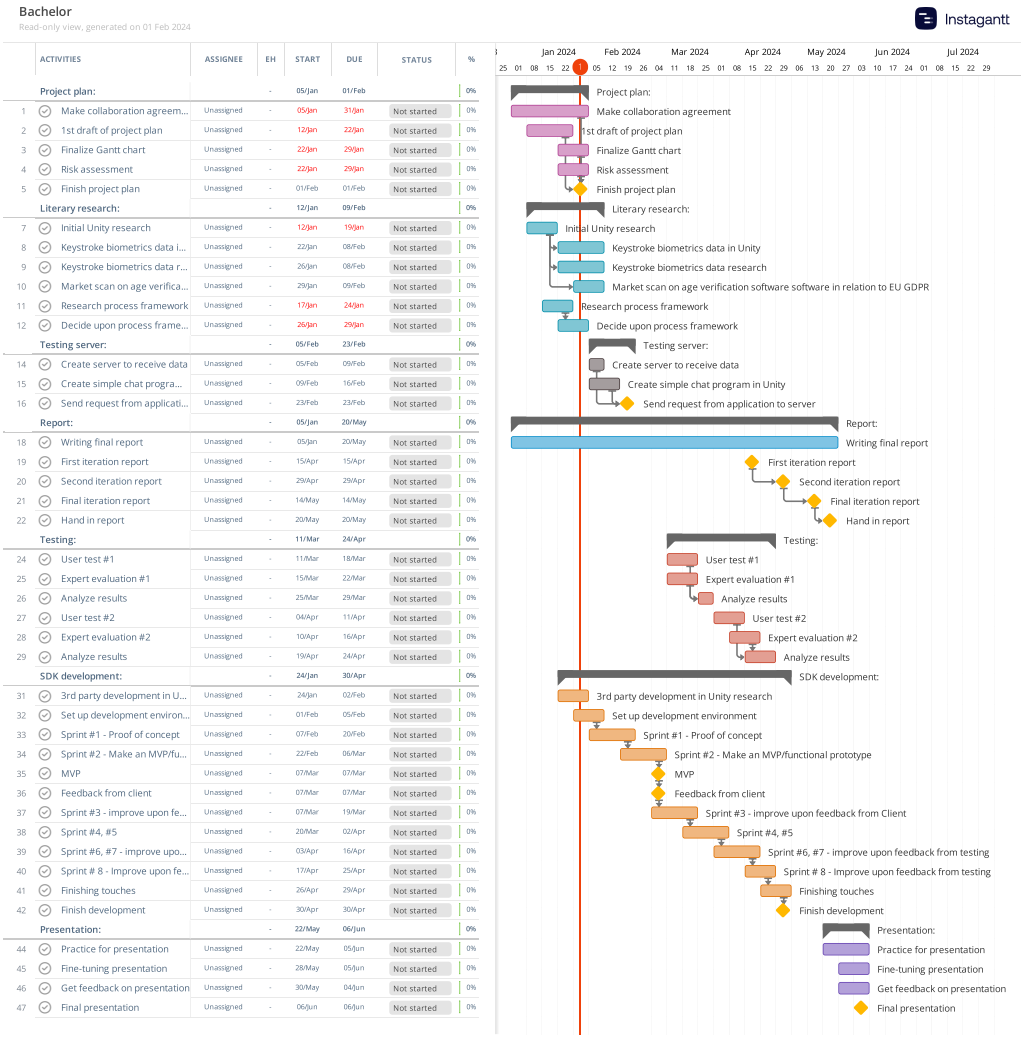
6.1 Time schedule

A detailed time schedule has been created for the purpose of planning the project from start to finish. The Gantt chart can be found in Figure 6.1.

The main milestones are:

1. **Finishing project plan** (31.01.2024)
2. **Send data from mock application to receiver server** (23.02.2024, might become delayed because of SkyHiGh server outage)
3. **Developing an MVP** (17.03.2024)
4. **Getting feedback from AIBA** (17.03.2024)
5. **First iteration report** (15.04.2024)
6. **Second iteration report** (29.04.2024)
7. **Finishing development of SDK** (30.04.2024)
8. **Final iteration report** (13.05.2024)
9. **Handing in final report** (20.05.2024)
10. **Final presentation** (06.06.2024)

Figure 6.1: Gantt chart



Bibliography for resources

[1] S. G. Steinar Opphus Ole Andre Slettum, "Prosjektplan viten i senter," 2017.
[2] *Scrum guides*, https://scrumguides.org/, Accessed: 2024-01-26.
[3] *C# documentation standards*, https://learn.microsoft.com/en-us/dotnet/csharp/, Accessed: 2024-01-26.
[4] *Xml documentation comments (c# guide)*, https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/xmldoc/, Accessed: 2024-01-26.
[5] *Dotnet format*, https://github.com/dotnet/format, Accessed: 2024-01-26.
[6] *Overview of .net source code analysis*, https://learn.microsoft.com/en-us/dotnet/fundamentals/code-analysis/overview?tabs=net-8, Accessed: 2024-01-28.
[7] *Technical deep dive, unity best practices*, https://unity.com/how-to, Accessed: 2024-01-26.
[8] *Integration testing*, https://en.wikipedia.org/wiki/Integration_testing, Accessed: 2024-01-31.
[9] *Unit testing*, https://en.wikipedia.org/wiki/Unit_testing, Accessed: 2024-01-31.

Glossary

A | I | M | S | U

A

agenda

An brief overview of what will be discussed during a meeting. 9, 11

I

integration testing

A way of testing software where several units of a system is tested as a combined unit [8]. 12

M

meeting minutes

An overview of what was discussed during a meeting in detail. 11

S

Soft Biometic Keystroke Dynamics

Soft Biometic Keystroke Dynamics is a way of profiling a person on the way they write. It is based on the users travel time between the pressed keys, the duration of each key press, and the rest time between each key press. 2, 3

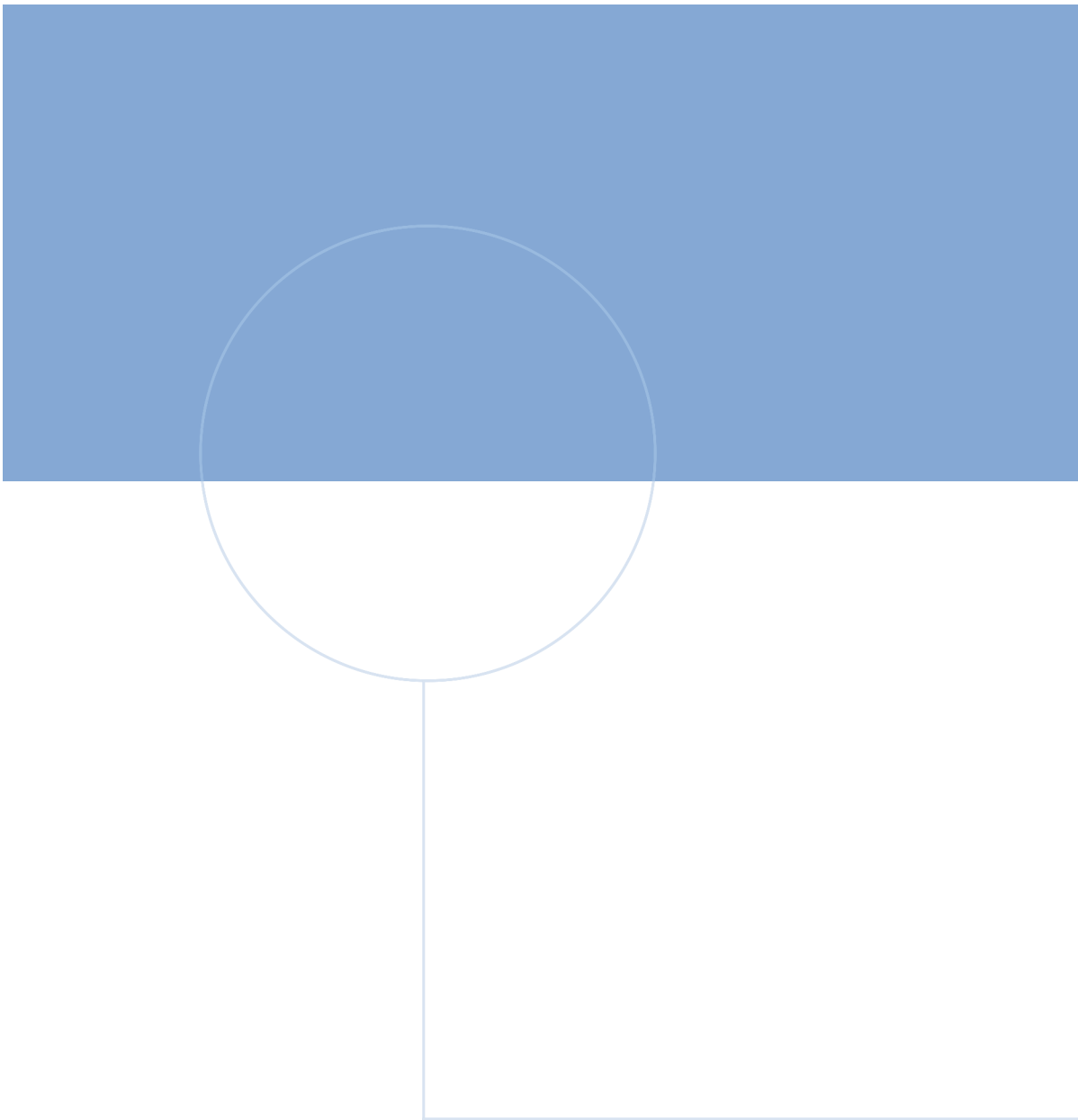
U

unit testing

Testing of isolated components of a system [9]. 12

Unity

A game engine for developing both 2D and 3D games, made by Unity technologies . 2



NTNU

Norwegian University of
Science and Technology