Bachelor’s thesis

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

Amundsen, Trygve Sunde
Helmersen, Martin Dolmen

Microsoft Outlook add-in for VisBook's property management system

A system development project for simplifying workflow between systems

Bachelor’s thesis in Dataingeniør
Supervisor: Ali Alsam
May 2023

NTNU
Norwegian University of
Science and Technology

Amundsen, Trygve Sunde
Helmersen, Martin Dolmen

Microsoft Outlook add-in for VisBook's property management system

A system development project for simplifying workflow between systems

Bachelor's thesis in Dataingeniør
Supervisor: Ali Alsam
May 2023

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

Preface

It is with great pleasure that we present this bachelor project in Computer Engineering, which marks the completion of our three-year study at NTNU. Throughout this journey, we have acquired invaluable friendships, gained a wealth of knowledge, and developed a passion and drive for software development and engineering.

First and foremost, we would like to express our sincere gratitude to our guidance counselor, Ali Alsam, for his invaluable support, guidance, and expertise throughout this project. We want to thank VisBook for their support, trust, guidance and for the welcoming nature of every employee we have crossed paths with throughout this project.

We would also like to give thanks to all our friends, both on and off campus, and also a special thank you to our families, who has motivated and shown support throughout this project.

This project has provided us an invaluable opportunity to apply the knowledge and skills acquired during our academic journey. It has been challenging but also exciting. We have gained profound experience in software development and academic writing. We are both eagerly anticipating the future endeavors and challenges that lie ahead.

abstract

In our study, we addressed the following problem: *"How can the communication between users of a property management system, hotels as an example, and their guests be enhanced through the implementation of a Microsoft Outlook add-in?"*. This report presents an analysis of the findings obtained from interviews, literature, and usability tests conducted during the design and development of the add-in. In our case, the add-in is a tool to search and connect email correspondence and their related data. Our main objective was to streamline the communication process and provide hotel management with easy access to the information necessary for effective correspondence with their guests directly within Outlook. By eliminating the need for users to switch between multiple programs to gather relevant information, our solution enables prompt responses to guest requests.

Sammendrag

I vår studie tok vi for oss følgende problemstilling: *"Hvordan kan kommunikasjonen mellom brukere av et eiendomsadministrasjonssystem, hoteller som et eksempel, og deres gjester forbedres gjennom implementering av et Microsoft Outlook-tillegg?"*. Denne rapporten presenterer en analyse av funnene fra intervjuer, litteratur og brukervennlighetstester utført under design og utvikling av tillegget. I vårt tilfelle er tillegget et verktøy for å søke og koble sammen e-postkorrespondanse og tilhørende data. Vårt hovedmål var å strømlinjeforme kommunikasjonsprosessen, samt gi hotellledelsen enkel tilgang til informasjonen som er nødvendig for effektiv korrespondanse med gjestene deres direkte i Outlook. Ved å eliminere behovet for brukere å bytte mellom flere programmer for å samle relevant informasjon, muliggjør løsningen vår raske svar på gjesteforespørsler.

Table of Contents

List of Figures

1 Introduction

A Property Management System or PMS for short, is a software application that lets accommodation businesses manage their operations and guest related activities. The accommodation businesses could be real estate companies, hotels, resorts, restaurants or camping sites[18].

A PMS is designed to automate tasks such as booking reservations, managing room rates, tracking inventory and supplies, handling billing and accounting, generating reports, and communicating with guests. The system can also provide tools for managing maintenance and repairs, scheduling cleaning services, and tracking property inspections. Overall, a property management system helps property owners and managers to efficiently and effectively manage their properties, reduce costs, increase revenue, and enhance the guest experience.

VisBook is a company that was founded in 1995 and specialises in delivering a property management system that, as of today, serves over a thousand customers in the Nordic countries. The majority of VisBook's customers are hotels and accommodation businesses. In addition to the standard functionalities of a PMS, VisBook provides solutions for integration with third-party software and services, such as payment gateways, accounting software and online travel booking.

To aid the explanation of this project, let us consider an example where the customer of Visbook is a hotel. Further, let us envisage a case where the hotel management sends special offers to their guest list. In this example, the hotel personnel would make use of the property management system to send the offers which are received by the guests as an email. If a guest wishes to make use of the offer, she/he would then respond by sending an email to the hotel. The reply is received by the hotel's email client, without being registered by the property management system. Knowing that VisBooks' PMS and the hotel's email client are two non-connected programs, the hotel's personnel are forced to spend a considerable amount of time connecting guest information across two separate platforms.

To improve the workflow, alleviate the pressure on the hotel's personnel (in our example), and reduce human error it is advantageous to connect the email correspondence, between the hotel and the guests, with the property management system. To this end, VisBook wishes to develop an add-in that retrieves all guest's information, related email correspondence and displays them to the end user in an organised and user friendly fashion.

An add-in, otherwise known as a plug-in, is a software component that adds extra features and functionality to a program. VisBook desires an add-in that is connected to Microsoft Outlook, which is the email client that is typically used by VisBook's customers. Specifically, VisBook desires the functionality to display information from their PMS and Microsoft Outlook, that includes email correspondence, bookings, statistics, sent attachments, and reviews within this add-in.

The task of this Bachelor project is to develop a Microsoft Outlook add-in that includes the functionalities specified by VisBook. To optimize the design of such an add-in, we have researched required tools and conducted usability tests.

2 problem description

2.1 Background

In response to the challenges faced by their customers who were using two separate and non-connected platforms, Microsoft Outlook and the property management system, Visbook developed a Microsoft Outlook add-in as a supplementary feature to their property management system. The add-in whose sole function was archiving email correspondence between the users of the PMS and their guests in Microsoft Outlook was an attempt to simplify the users' experience and workflow. Unfortunately, due to limitations in development-resources, this add-in is no longer available to VisBook customers.

The idea behind the discontinued add-in was to store all communication between the customers and their guests on the digital client-card which is initialized in the PMS for each guest/client the moment they use the service. The digital card includes information such as: email, statistics, preferences, and expenditures.

Customer feedback received by VisBook indicates that they deeply miss the functionality provided by the discontinued add-in. In response to this feedback and their commitment to customer satisfaction, VisBook has embarked on a new project to develop a Microsoft Outlook add-in that not only replaces the old version, but also includes additional features. These new functionalities include visualized statistics, an overview of attachments from email correspondence, and a booking history, all aimed at enhancing the user experience and meeting the evolving needs of VisBook's valued customers.

We undertook this bachelor project, which involves research and system development, with the aim of establishing a Minimal Viable Product (MVP) for VisBook. This entails building the foundational structure, data, and system, as well as conducting relevant research, to provide VisBook with a solid foundation for further development.

The problem description for this bachelor thesis is : "How can the communication between users of a property management system, hotels as an example, and their guests be enhanced through the implementation of a Microsoft Outlook add-in?"

2.2 Structure

3 **Theory and relevant literature:**
An overview of relevant literature and technologies tied to development of a Microsoft Outlook add-in.

4 **Methodology and process:**
Within this section we'll go through our choices of technology, work methodology, and scientific approach.

5 **Results:**
Here we will present the results of this project. Scientific, Engineering and Administrative results.

6 **Discussion:**
The groups implementation and final results from the project will be discussed within this chapter.

7 **Conclusion and further work:**
Finally we will explain if the project answered the problem description and if the solution, or foundation, that we've built will be a sustainable solution. At the same time, we will put this into perspective with regards to further development.

8 **Societal Impact:**
This last section details the societal impact by implementation of our solution.

3 Theory

3.1 Property Management System

Property management systems first appeared in the late 1970s, they then gained traction in the hotel industry throughout the 1980s and 90s. Despite vast improvements in technology over the years, the original intentions behind the PMS remain the same 40 years later: helping hotels run more efficiently by automating manual and paper-based administrative tasks[6].

The core functions of the PMS include managing workflows for both back office and front office day operations, like reservations and guest profiles, room inventory and occupancy, pricing and Revenue Per Available Room (RevPAR), check-inn and check-out, housekeeping, invoicing, and reporting. Over the years, new functions have been either built into the PMS or integrated as third-party applications[6].

3.2 Software Development Life Cycle

A software life cycle model (also termed process model) is a pictorial and diagrammatic representation of the software life cycle. A life cycle model represents all methods required to make a software product transit through its life cycle stages. It also captures the structure in which these methods are to be undertaken.

The Software Development Life Cycle, or SDLC, refers to a methodology with clearly defined processes for ensuring quality software. In detail, the SDLC methodology focuses on following the phases of software development:

- Requirement analysis
- Planning
- Software design
- Software development
- Testing
- Deployment
- Maintenance

The software development process typically consists of seven distinct phases, of which the initial five pertain to planning and developing the software. The final two phases, are dedicated to establishing a connection between the software and its end-users and managing the software over the long term. In the specific case of our project, the development was unable to progress beyond the development phase due to the unavailability of production and deployment pipelines at VisBook. Moreover, certain functionality needed to be incorporated into the core systems of VisBook to facilitate the retrieval of certain data.

The requirement analysis and planning of the development process were drafted through the initial meetings with VisBook, while also setting the boundaries and formulating the specifications from which we were to operate within. This ensured that the SDLC phases following these meetings, see 1, would progress without mishaps.

SDLC's stages:[2]:

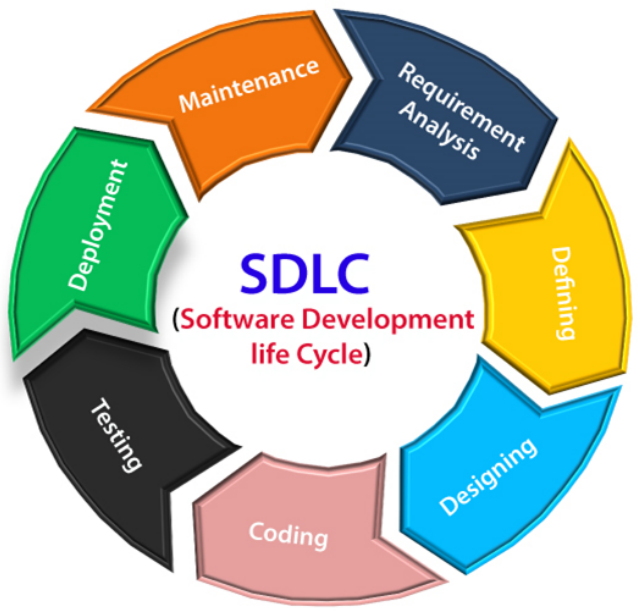


Figure 1: Software Development Life Cycle figure

- Identify the current problems.

This stage involves getting input from all stakeholders, those in the case of our project were customers of VisBook, industry experts, and VisBook employees, which help us learn the strengths and weaknesses of the current system with improvements as the goal.

- Plan

In this stage, the team determines the cost and resources required for implementing the analyzed requirements. It also details the risks involved and provides sub-plans for softening those risks. In other words, the team should determine the feasibility of the project and how they can implement the project successfully with the lowest risk in mind.

- Design

This phase starts by turning the software specifications into a design plan called the Design Specifications. All stakeholders can then review this plan and offer feedback and suggestions. It's crucial to have a plan for collecting and incorporating stakeholders input into this document. Failure at this stage will almost certainly result in cost overruns at best and the total collapse of the project at worst.

- Build

Here is where the actual development starts. It's important that every developer sticks to the agreed blueprint. Also, make sure you have proper guidelines in place about the code style and practices.

For example, define a variable naming style such as ”camelCase”. This will help your team to produce organized and consistent code that is easier to understand but also test during the next phase.

- Code Test

This is where we test for defects and deficiencies. We fix those issues until the product meets the required specifications. In short, we want to verify if the code meets the defined requirements.

3.3 Human-Computer interaction

Human-computer interaction (HCI) is a multidisciplinary field of study which combines research in computer science, psychology and design. Specifically, it is the study of how people interact with technology. A few of HCI main focus areas are user experience (UX), interface and interaction design.

The interaction between the the user and the computer occurs through a process where the user provides instructions to the computer through the interface, and then is subsequently served with the result[7].

In this Paper of 'Usability Heuristics as an Assessment Parameter", Jakob Nielsen has the following definition of usability ''Usability is a quality attribute that assesses how easy user interfaces are to use". Further, Nielsen defined five quality components that measures the usability of a product [14].

- Learnability

How easy is it to learn the application and do different tasks.

- Efficiency

Once you are familiar with the design, how quickly can a task be completed.

- Memorability

Does the application have a intuitive design and is it easy to recall the required steps to perform a task.

- Error

How many and how often do user error occur, and how easily can they recover from the errors.

- Satisfaction

How pleasant the application is to use.

3.4 Development Process

3.4.1 Version Control

A widely employed collaborative tool in the software industry is the Version Control System (VCS). The VCS allows teams to manage and monitor multiple versions of source code by enabling each member to work simultaneously on different segments of the code. The tool facilitates this process by allowing developers to work on different branches, which are essentially copies of the code base that can be modified without affecting the primary codebase. Following the coding phase, developers can merge their work with the original codebase, allowing for the implementation of new functionalities without disrupting the work of other team members.

There are two types of version control systems, namely centralized and distributed. Distributed VCS allow developers to have their copy of the repository (code base). Changes can be made locally before being merged with the central repository. Centralized VCS stores the code in a single repository, requiring developers to check out and check in code in order to make changes. [13]. Distributed VCS is the most commonly used by developer teams; it is also used in this project due to its benefits of backups, fast merging, and flexible branching.

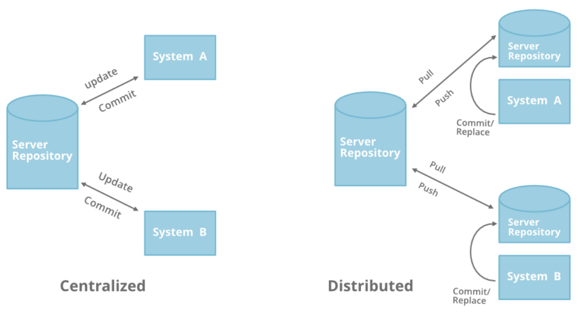


Figure 2: Centralized Version Control System vs Distributed Version Control System

3.4.2 Objectives and Key Results

Objectives and Key Results is a collaborative goal-setting methodology used by teams, companies, and individuals. The approach comprises two components: objectives and key results. The objectives to be attained must be significant, concrete, and measurable. The key results are smaller goals that should contribute to the completion of the objectives. Effective key results are specific, verifiable, and time-bound and should be measurable indicators of progress toward the objective.

The terms ”Monday commit” and ”Friday wins” are often associated with the OKR framework. The former refers to a weekly practice wherein the team identifies the desired key results for the week and sets specific, attainable goals (3-5) aimed at advancing towards those results. On the other hand, ”Friday wins” is a weekly review process that assesses the team’s progress towards the objectives established on Monday. During this process, the team reflects on both successes and shortcomings, evaluates what went as planned and what did not, and discusses potential future complications.[29].

3.5 Kanban Board

The Kanban board is a workflow visualization tool. 'Kanban' is a Japanse term that translates to 'Visual cards'. Kanban's main purpose is to create visual cards that list details about a task and organize them into different columns. The columns represent different stages in a production cycle; the stages could be 'To-do', 'Doing', 'For review', 'Reviewed' and 'Done'. The board could be physical on a whiteboard with different colored sticky notes or digital. There are multiple websites and programs that offer this functionality, such as Github, Gitlab, KanBanize, and Miro.

In the systematic literature review, 'Kanban in Software development' [1], the authors found the following critical features pertaining to the use of Kanban in modern software development.

- Motivations for implementing Kanban were simplicity, focus on workflow, and efficiency.
- The benefits of using Kanban were improved software quality, reduced project delivery time, improved stakeholder communication, and improved customer satisfaction.

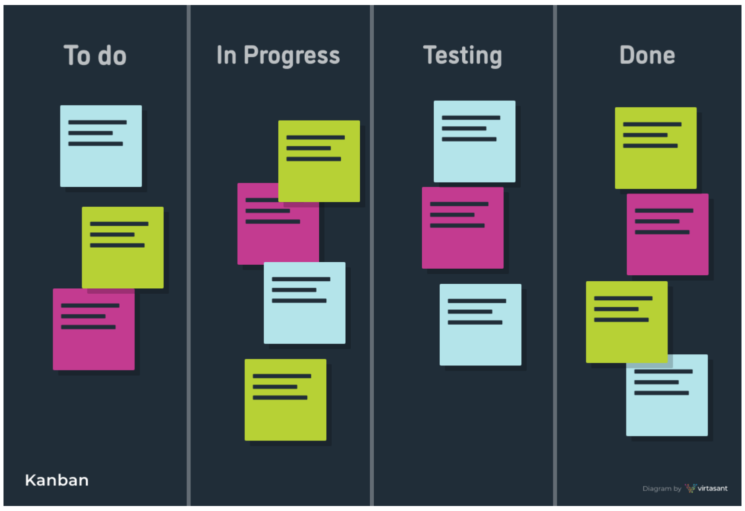


Figure 3: Kanban board[26]

3.6 Usability Testing

Usability testing is a crucial process that seeks to assess how actual users interact with and employ a product. This evaluation provides valuable insights into user performance and the level of acceptance of the product. Through usability testing, potential pitfalls can be identified, and practical solutions can be proposed to address them. Conducting usability tests early in the development process proves immensely beneficial, as it enables the identification of potential roadblocks and challenges that, if left unaddressed until the final stages of production, could be prohibitively expensive to overcome.

A prototype or a wireframe of the product is needed to conduct early testing. A wireframe is a visual blueprint of the designed User Interface (UI) for the product; it focuses on space allocation, layout, functionality, and intended behaviors. Modifying a wireframe is more straightforward and require less time than changing production code at a later stage.

In this project, wireframes were created using a web-based platform called Figma. This platform allows for the generation of clickable prototypes that facilitate user navigation and interaction with the product. Such prototypes help highlight navigation issues and elements that need clarification.

4 Methodology

4.1 Scientific approach

There are two prominent scientific methodologies employed in data collection, namely qualitative and quantitative approaches. These approaches differ in the type of collected data. Qualitative data are typically descriptive, conveying information through language, rather than numerical values. Conversely, quantitative data are a collection of observations that can be quantified, counted, or measured, and thus represented through numerical values [24].

While both quantitative and qualitative approaches offer valuable insights into research questions, the nature of the information sought for this particular project necessitated the use of qualitative methodology. This was determined based on a thorough consideration of the type of knowledge and data that were needed to gain insight into the research question, as well as the significance of the data.

The selection of a qualitative research approach involved recruiting representatives from VisBook-affiliated businesses, utilizing established customer relationships to facilitate data collection through interviews. The study participants consisted of hotel managers and receptionists, possessing extensive knowledge and work experience in using VisBook's property management system and Outlook in their daily operations. During the interviews, the research team explored the technical experiences of the participants, as well as their expectations and desired functionalities of the add-in under development in the project. The data obtained through these interviews provided crucial insights into the users' business and technological expertise.

In this project, we utilized semi-structured interviews which are a qualitative approach that involves a set of open-ended questions designed to facilitate a conversational flow while enabling the researcher to maintain a level of control over the general topic areas under discussion. This approach allows research participants to articulate their thoughts and experiences freely, without the constraints imposed by rigid questioning protocols, providing the researcher with the flexibility to explore unanticipated topics that may emerge during the interview [25]. Semi-structured interviews are typically recorded for subsequent transcription and analysis. In the present project, the data obtained from the interviews, which also incorporated a usability test, were deemed sufficiently informative to be utilized during the second and third phases of the Systems Development Life Cycle.

4.2 Development Methodology

4.2.1 OKR with Kanban

To establish a systematic and professional workflow, we adopted an agile development methodology throughout the project duration [4]. While we had prior experience with Scrum and Kanban, we recognized that these methodologies may not be suitable for our team's size. Scrum and Kanban involve formally defined roles and ceremonies, such as daily stand-ups, sprint planning, and retrospectives, which can be overly time-consuming. Therefore, after careful consideration of our team size and time constraints, we chose to implement Objectives and Key Results (OKR) methodology, complemented by a Kanban board for our development process [22] [12].

As described in the theoretical framework, OKR is a goal-setting methodology that helps users focus on objectives and key results. In our case, the goal was to create a 'Minimal Viable Product of an add-in.' The team's use of a Kanban board in conjunction with OKR facilitated tracking progress towards the stated objective and key results, while also providing a centralized location for monitoring all available tasks and their respective stages of production [19].

In practice, we created the OKRs in an external Word document, along with the key results. Every week, we added commits and wins, which resulted in a complete list of achievements and goals for the entire project duration. Regarding the Kanban board, we utilized Gitlab's built-in issue board to organize tasks and production stages.

4.2.2 Gitlab

The project's source code was managed by GitLab, an open-source software development platform that supports end-to-end development processes. Other optional platforms for hosting were Microsoft Azure and Amazon Web Service (AWS). The decision to use Gitlab was motivated by it being VisBook's main development platform. To access the project repository, the team was required to use a Virtual Private Network (VPN)[28].

In addition to hosting source code, Gitlab was utilized for managing our Kanban board with tasks, creating milestones, and handling branching and merging processes. Additionally, the team adhered to standard protocols for merge requests and reviews, as outlined in [9]. Finally, Gitlab provides digital graphs known as burndown charts, which are commonly used by project managers, for project time estimates, code production, issue completion, and milestone progress. In our project, we utilized these charts as a visual and numerical aid, where progress is shown as a graph with a timeline[8]. Our graph can be seen in figure 4.

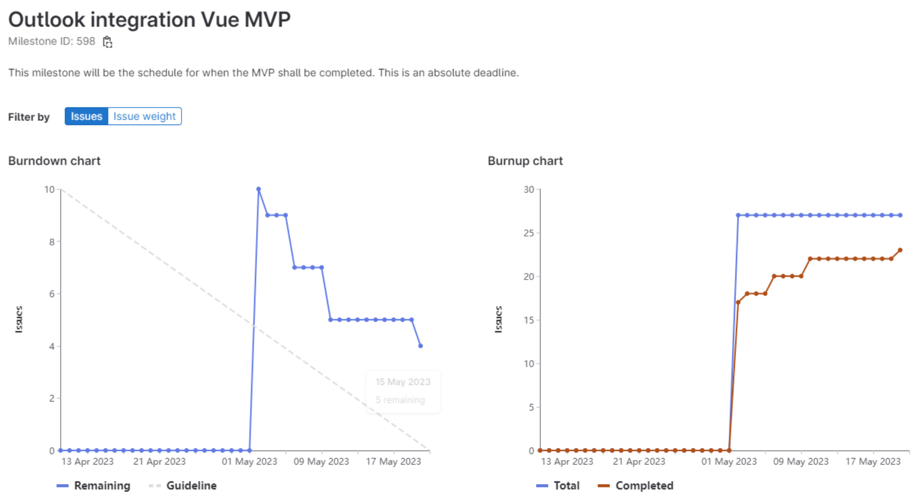


Figure 4: Burndown and burnup charts

4.3 Usability Testing

As stated in the theoretical framework 3, incorporating usability testing early in the development process can yield several advantages in terms of minimizing unnecessary revisions and identifying potential obstacles. Usability testing involves four core elements: the facilitator, the tasks, the participants, and the product.

In our project, the facilitator played two distinct roles. One team member was responsible for monitoring the participant's behavior, while the other member tracked task completion. During one usability interview, a participant was asked to perform specific tasks, such as using the add-in to search for current and past guest reservations. The facilitators split responsibilities for this task, with one reading the task instructions to the participant, and the other carefully observing the process. This included monitoring ease of use, completion time, and the participant's body language.

Prior to commencing the usability tests, it was imperative to identify and recruit participants who were appropriate for the study. Accordingly, individuals who were either interested in a new add-in or had prior experience with the previous version were selected among VisBook's business affiliates. Digital meetings were arranged with personnel from three hotels to participate in the study. To accommodate for geographical restrictions, the usability tests were conducted over Microsoft Teams. Details pertaining to the test participants, along with the associated results, can be found in the usability test document provided in Appendix E.

The usability test was divided into three distinct sections, namely preliminary questions, specific tasks, and post-task review questions. The purpose of the preliminary questions was to establish a baseline understanding of the test subject's knowledge and usage of VisBook's PMS. The specific tasks consisted of a series of activities related to the clickable prototype, which had been created and published in Figma, as previously mentioned in the theoretical framework 3. Test subjects were granted access to the prototype and instructed to share their screen while conducting the tests. Two facilitators were responsible for observing and analyzing the subject's behavior. The post-task questions were employed to summarize the test subjects' overall experience with the product. Both the preliminary and post-task questions comprised semi-structured qualitative interviews. Prior to commencing the interviews, an interview guide was carefully formulated. To ensure the quality of the questions, a guideline from sociology at Harvard[17] was used. A comprehensive list of the preliminary and post-questions can be found in the test documentation provided in Appendix D.

4.4 Choice of Technology

4.4.1 Frontend

Vue
After analyzing the project's demand specifications (Appendix B), it was determined that Vue would be the optimal front-end framework for the development of the application. Vue is an open-source JavaScript-based framework that is widely used in building interfaces and single-page applications. It is built on standard HTML, CSS, and JavaScript and offers a component-based programming model that is similar to other frameworks such as React or Flutter. Since its initial release in 2014, Vue has gained popularity among developers due to its flexibility and rendering performance [27]. Its widespread adoption has also resulted in a vast network of resources and support, which facilitates swift resolution of errors and technical challenges. The decision to select Vue for the add-in development was further reinforced by the team's and Visbook's previous experience with the framework.

Yeomen-generator (Yo-Office)
In order to integrate the Vue application with Microsoft Outlook, it was necessary to create and upload a dedicated manifest file. A Microsoft Outlook add-in manifest file, is an XML file that contains crucial information about the add-in, including its functionality, user interface, settings, and permissions. In this project, the manifest file was generated using the "Yo-office" package, which is a component of the Yeoman-generator tool. The default manifest file that was automatically generated was then customized to align with the specific requirements of the project. [16].

Vuetify
Vuetify is an open-source framework that offers a complete package of reusable User Interface components. Each component is easily customizable to multiple layouts and themes. Vuetify is easily integrated into Vue.js and is built upon the material design principles, making the design user-friendly and modern [15].

Npm
The Node Package Manager (Npm) for Node.js was created in 2009 as an open source project to help JavaScript developers share packaged modules easily. In 2014, Npm was founded as a standalone company, and in 2020, it was acquired by GitHub.

The Npm registry is a public collection of open source packages that cover a wide range of applications, including Node.js, front-end web apps, mobile apps, robots, routers, and more.[20].

Npm is the command line client that allows developers to install and publish packages. In this project we use Npm for installing, and maintaining the packaged modules needed for our development. This is because we use different packages like Vuetify and Material Icon design.

4.4.2 Backend

ASP.NET Core web API
Based on the technical requirements of the project, and an analysis of Visbook's primary tech stack, we decided to use the following backend technologies:

- ASP.NET core
- C#
- SOAP API

- RESTful API

ASP.NET Core web API is a well-known framework for building RESTful web services using the Microsoft .NET Core platform with C#. The decision to adopt ASP.NET Core Web API for the backend application was made based on several technical requirements. Specifically, the backend application needed to connect to a SOAP API - an acronym for Simple Object Access Protocol Application Programming Interface akin to an API. Notably, Microsoft developed SOAP API; therefore, ASP.NET Core Web API provides well-documented features for connecting and utilizing SOAP APIs. In addition, Visbook's primary tech stack includes ASP.NET and C#, which further reinforces the selection of the framework above [11].

Swagger
To facilitate the publication of the endpoints required for the retrieval of data from VisBook's database, a RESTful API with Swagger was utilized for the backend. Swagger is a set of tools and associated specifications that seamlessly integrates into an ASP.NET Core application, generating API documentations and thus enabling developers to build and consume API's more efficiently. Furthermore, Swagger follows standard protocols for security and generates examples of responses and requests, which enabled us to test all required responses securely.

4.4.3 Design Tools

Figma
Figma is a cloud-based design and collaboration tool that facilitates the creation of digital designs among teams, designers, and developers. This browser-based platform enables smooth collaboration and sharing of project design, as well as robust interface design, wireframing, and usability testing.

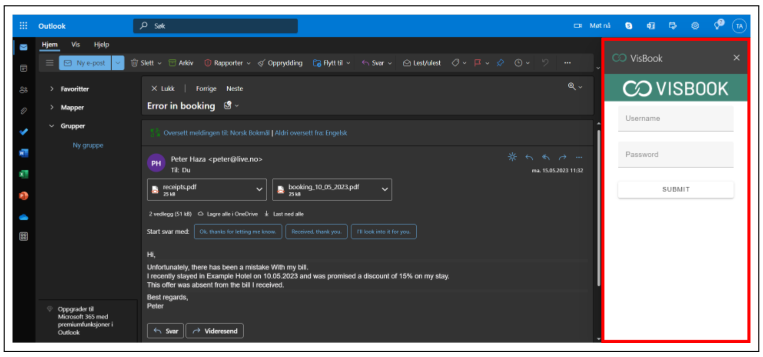
In our project, we found Figma to be highly useful for creating a clickable prototype that was essential for testing and client interviews. The ability to share a clickable prototype with remote users proved to be an efficient way of conducting usability testing, as we were able to share our prototype with VisBook clients while conducting the interviews on Teams.

Additionally, having a mock-up design of the application proved useful as a reference point, helping to unify the vision of the client and team members. Utilizing Figma to create the prototype and explore potential new features prior to implementation in production code helped streamline the development process and minimize the likelihood of unforeseen issues. Overall, Figma proved to be an indispensable tool for enhancing the design and development process of our project.

5 Result

This section is divided into three parts, namely: scientific, engineering, and administrative results. In the scientific results, we review the results gained from the interviews done with VisBook clients. The engineering results include: Functional and design demands. Finally, the administrative results outline: project plan, time management, and development methodology.

Before introducing detailed results, we wish to give the reader a visual overview of the developed add-in the way it appears in Microsoft Outlook. In figure 5, we see the add-in which is located to the right of the user screen. We note the clean design of the add-in with a clear VisBook header. The user interacts with the add-in through the active mail window, specifically, with the option symbol(...) where the add-in is initially activated. Upon activating the add-in, it will appear to the right of the active mail window, as per default positioning of Microsoft Outlook add-ins. In practice, the add-in, which is a taskpane, will only be available for VisBook users who choose to make use of the functionalities offered by the add-in.



Error in booking

Peter Haza <peter@live.no>
Til: Du
ma. 15.05.2023 11:32

Hi,

Unfortunately, there has been a mistake With my bill.
I recently stayed in Example Hotel on 10.05.2023 and was promised a discount of 15% on my stay.
This offer was absent from the bill I received.

Best regards,
Peter

Figure 5: Add-in visible on the right side of the figure, displayed within the Microsoft Outlook email browser

5.1 Scientific Result

5.1.1 Pre-development phase

Prior to commencing the development of the add-in, a pre-development phase was required to acquire the necessary skills, design the add-in, create the prototype, and conduct usability tests. This phase aligns with the 'Planning' and 'Software design' found in the Software development life cycle, as outlined in the theory framework 3.

During the planning phase, the project timeline was divided into smaller segments and visualized using a Gantt chart. This approach facilitated the tracking of important milestones and enabled effective planning of different project phases. Key documents, including the Vision Document and demand specifications, were prioritized as initial deliverables. Subsequently, the team embarked on acquiring the prerequisite skills to fulfill the demands specified in the Vision Document. This phase involved engaging with various learning resources such as tutorials, guides, and documentation on multiple technologies, namely: C#, .NET Core, RESTful API, Vue, Vuetify, and SOAP API.

Once the necessary skills were acquired, a prototype of the add-in was developed using Figma, which enabled the creation of a clickable prototype. This process involved careful analysis of the project requirements, engaging in meetings with employees of Visbook, and adhering to design guidelines provided by Microsoft. Following the design approval received from the product owner at Visbook, the team proceeded to conduct usability testing through interviews with customers of Visbook.

Design
The interface exhibits a minimalistic and user-friendly design. The team has deliberately minimized the presence of extraneous components on the page, aiming to diminish visual clutter and highlight the prominent features. A coherent design methodology has been employed, incorporating consistent shapes and color combinations throughout the entire add-in. However, it should be noted that the color scheme of the top header of the add-in remains subject to the chosen system theme of Microsoft Outlook. The visual representation in Figure 6 portrays our application within the contexts of both dark and light themes inherent to Microsoft Outlook.

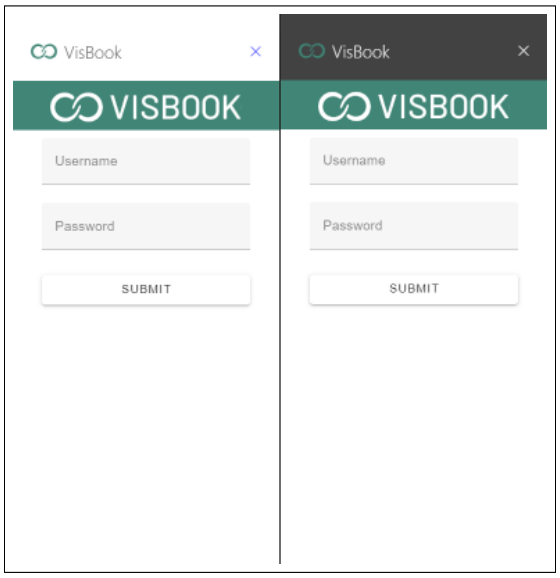
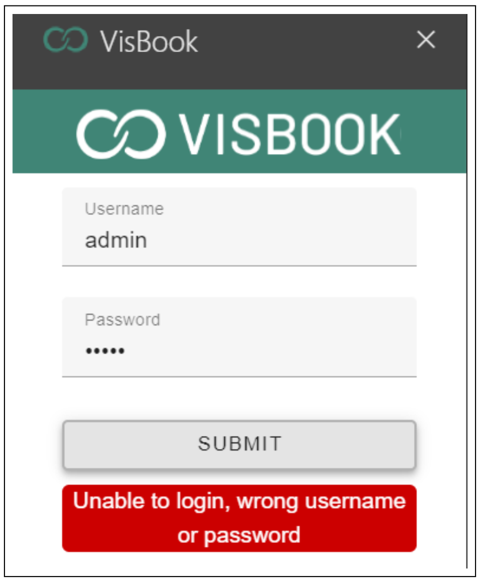


Figure 6: Upon activation of the add-in, users are met with this Login page, which responds to the selected theme of outlook with a dark or light theme

Login
The login page, shown in Figure 6, comprises the presence of the Visbook logo along with two input fields and a button. Upon submission of the provided username and password, the authentication process is triggered. If the credentials are successfully authenticated, the user is subsequently redirected to the home page. However, in the event of a failed authentication request, the user is presented with an error message, as illustrated in Figure 7.



Unable to login, wrong username or password

Figure 7: Upon login, if the credentials input into the fields aren't correct, Failure to authenticate will result in an error message that is displayed

Header
The purpose of the header bar is to give the add-in user concise information pertaining to the relevant guest, whom the user is interacting with. As depicted in figure 8, the header bar contains the name and email address of the guest. Note that for new guests, where no information is available in the user's system, the header will show the email address and the account name. For example, the left side of figure 8 shows an email sent by Adobe Creative Cloud while the right side figure shows the name and email of an existing user.

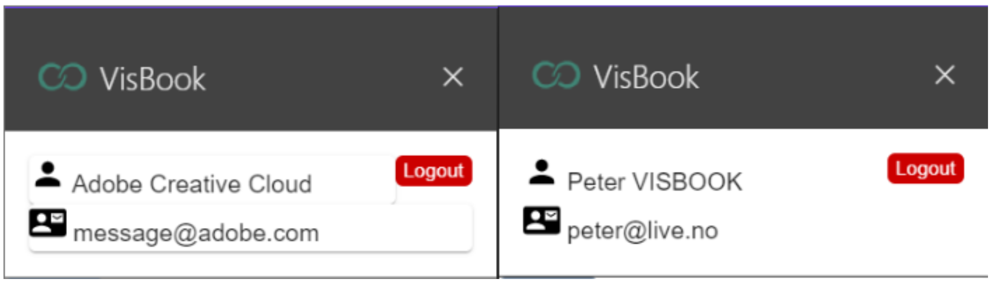
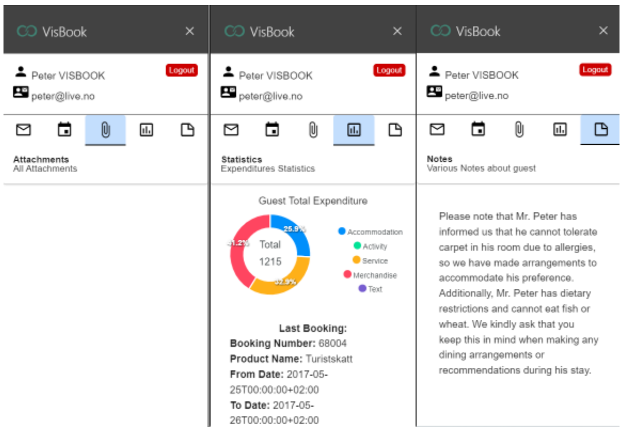


Figure 8: *Header with Information anchored at the top, displaying data from a client-card that is pulled from VisBook. In the case of no recognized client-card existing, data from outlook mail will be displayed.*

Navigation bar The purpose of the navigation bar is to provide the user with a consistent and intuitive way of navigating through our add-in. The active tab is marked with a subtle blue background, see figure 9. Upon selection of a tab, data for the corresponding tab is displayed. Every tab has an icon and color scheme that follows directives for WCAG and VisBooks color palette.



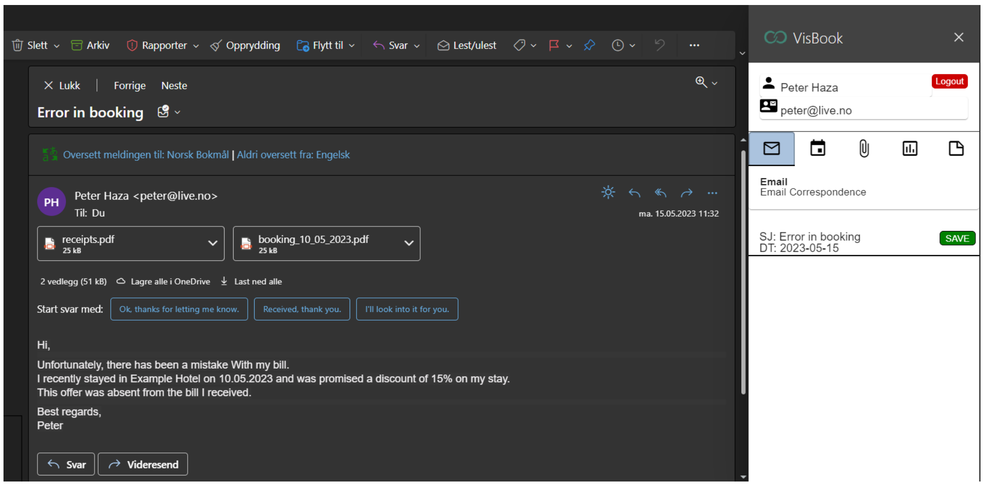
Last Booking:
Booking Number: 68004
Product Name: Turistskatt
From Date: 2017-05-25T00:00:00+02:00
To Date: 2017-05-26T00:00:00+02:00

Please note that Mr. Peter has informed us that he cannot tolerate carpet in his room due to allergies, so we have made arrangements to accommodate his preference. Additionally, Mr. Peter has dietary restrictions and cannot eat fish or wheat. We kindly ask that you keep this in mind when making any dining arrangements or recommendations during his stay.

Figure 9: Navigational tabs routing user throughout the add-in

Email

The email tab highlighted in the add-in, see figure 10, contains a dynamic list, which retrieves its data from the Microsoft Office Outlook API, from which the mail item and it's associated data gets fetched. The email tab is automatically connected to the Outlook's email item currently selected.



Error in booking

Oversett meldingen til: Norsk Bokmål | Aldri oversett fra: Engelsk

PH Peter Haza <peter@live.no>
Til: Du
ma. 15.05.2023 11:32

receipts.pdf 25 kB
booking_10_05_2023.pdf 25 kB
2 vedlegg (51 kB)

Start svar med: Ok, thanks for letting me know. | Received, thank you. | I'll look into it for you.

Hi,

Unfortunately, there has been a mistake With my bill.
I recently stayed in Example Hotel on 10.05.2023 and was promised a discount of 15% on my stay.
This offer was absent from the bill I received.

Best regards,
Peter

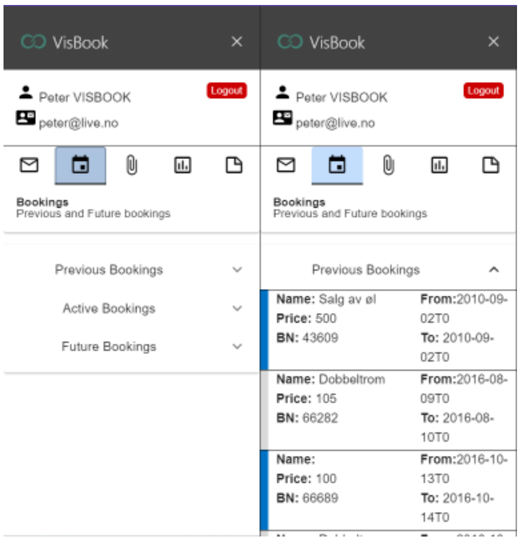
Peter Haza
peter@live.no

SJ: Error in booking
DT: 2023-05-15

Figure 10: The pane displaying information of email is shown when activation and login of the add-in has been completed.

Bookings

Within the booking's tab, highlighted in figure 11, users are offered three expandable menus: Previous, Active, and Future bookings. When expanded, these menus provide users with a detailed listing of information pertaining to each respective booking. As seen in figure 11, the user is provided with structured information about the guest including booking number, name, date, and service cost. To enhance visual separation and facilitate ease of navigation, an alternating color scheme has been implemented for every other booking entry.



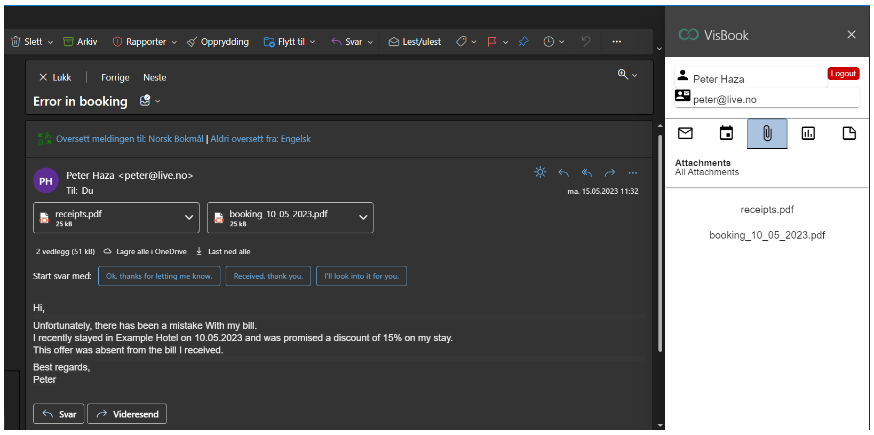
Name: Salg av øl **From:**2010-09-02T0
Price: 500
BN: 43609 **To:** 2010-09-02T0

Name: Dobbeltrom **From:**2016-08-09T0
Price: 105
BN: 66282 **To:** 2016-08-10T0

Name: **From:**2016-10-13T0
Price: 100
BN: 66689 **To:** 2016-10-14T0

Figure 11: Booking with and without previous bookings list expanded, which displays the relevant data.

Attachment
The attachments tab, highlighted in figure 12, displays a dynamic list of all attachments associated with a given mail item. The attachment data are grabbed from the current selected mail item, and displayed using the filename and format.



Error in booking

Oversett meldingen til: Norsk Bokmål | Aldri oversett fra: Engelsk

Peter Haza <peter@live.no>
Til: Du
ma. 15.05.2023 11:32

receipts.pdf 25 kB
booking_10_05_2023.pdf 25 kB

2 vedlegg (51 kB) Lagre alle i OneDrive Last ned alle

Start svar med: Ok, thanks for letting me know. Received, thank you. I'll look into it for you.

Hi,
Unfortunately, there has been a mistake With my bill.
I recently stayed in Example Hotel on 10.05.2023 and was promised a discount of 15% on my stay.
This offer was absent from the bill I received.
Best regards,
Peter

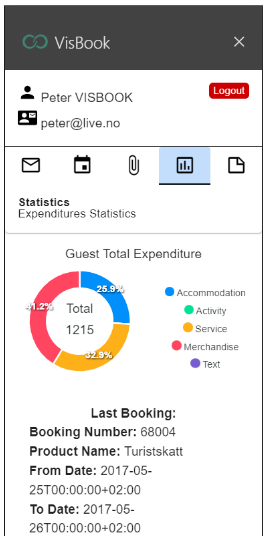
Peter Haza
peter@live.no

Attachments
All Attachments

receipts.pdf
booking_10_05_2023.pdf

Figure 12: In the example, we can see that the file is a .pdf

Statistics
The statistics tab, see 13, is comprised of two elements, namely the guest's total spending during their previous visits, and their latest booking. The guest spending's are visualised in a donut chart with corresponding labels. The purpose of the statistics pertaining to the guests expenditures are to tailor offers in accordance with the guests interests. Furthermore, if no client card for the guest exists, the contents of the statistics page will default to a string informing the user of this, see 14.

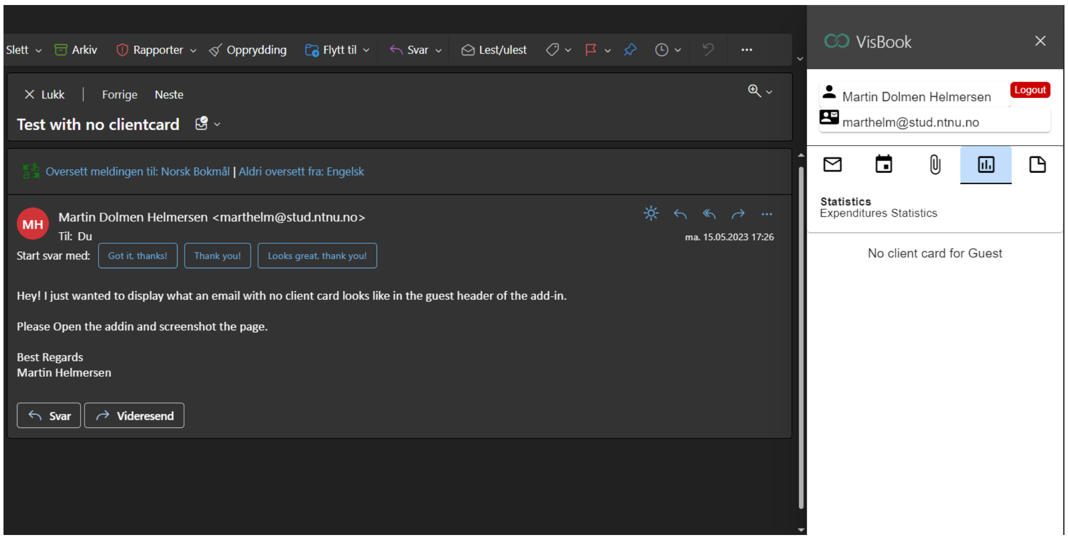


Peter VISBOOK
peter@live.no

Statistics
Expenditures Statistics

Last Booking:
Booking Number: 68004
Product Name: Turistskatt
From Date: 2017-05-25T00:00:00+02:00
To Date: 2017-05-26T00:00:00+02:00

Figure 13: Statistics are displayed both numerical and within a donut chart


Slett Arkiv Rapporter Opprydding Flytt til Svar Lest/ulest

Lukk | Forrige Neste

Test with no clientcard

Oversett meldingen til: Norsk Bokmål | Aldri oversett fra: Engelsk

Martin Dolmen Helmersen <marthelm@stud.ntnu.no>
Til: Du
ma. 15.05.2023 17:26

Start svar med: Got it, thanks! Thank you! Looks great, thank you!

Hey! I just wanted to display what an email with no client card looks like in the guest header of the add-in.

Please Open the addin and screenshot the page.

Best Regards
Martin Helmersen

Svar Videresend

VisBook

Martin Dolmen Helmersen Logout
marthelm@stud.ntnu.no

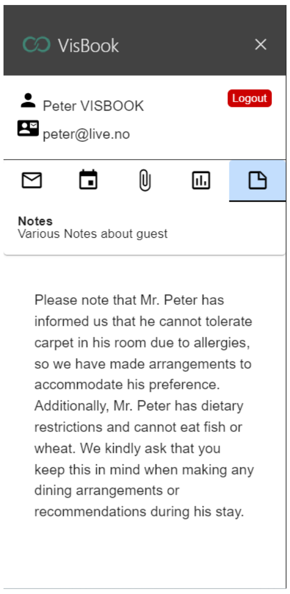
Statistics
Expenditures Statistics

No client card for Guest

Figure 14: No statistics are displayed if no client card data exists

Notes

The notes tab, see figure 15, is a tab that displays note data that originates from within the client-card that exists within VisBooks' system. The note is a text-box, where the user might for example write information about their guest, such as dietary requirements or special needs.


VisBook

Peter VISBOOK Logout
peter@live.no

Notes
Various Notes about guest

Please note that Mr. Peter has informed us that he cannot tolerate carpet in his room due to allergies, so we have made arrangements to accommodate his preference. Additionally, Mr. Peter has dietary restrictions and cannot eat fish or wheat. We kindly ask that you keep this in mind when making any dining arrangements or recommendations during his stay.

Figure 15: Notes tab

5.1.2 Result from Usability Tests

As outlined in the methodology section 4, usability tests were conducted as part of semi-structured interviews. In total we had three subjects for the interviews. For two of the participants, navigating the add-in according to our directions and questions was straightforward. However, the third participant experienced some difficulty, which was attributed to a language barrier between the researchers and the subject, as well as issues with peripheral equipment, on the interview subjects side, such as the headset and microphone. Nevertheless, the instructions were conveyed effectively after some back and forth, resulting in successful navigation.

Constructive feedback was provided by the participants, focusing on their state of mind when using the PMS system during work. As a result of the interviews, we revised the symbols, colors, and text, to align our color scheme with the design of the parent application (VisBook PMS), making it recognizable for new users and clearly indicating that this is an addition to VisBook's PMS.

All test participants expressed a desire for the add-in, particularly an add-in that replicated the functionality of the discontinued version. Interestingly, despite initially expressing this preference, every test participant demonstrated a strong interest in the new add-in after completing their respective interviews, and recognized that the new version offers enhanced functionality and a foundation for growth based on their feedback.

It was observed that some participants encountered difficulty in navigating to specific sections of the add-in, primarily due to their lack of familiarity with add-in's and limited guidance provided beforehand. Furthermore, one subject suggested the option of having the add-in in their own language, which could be considered for future work.

The results gathered from these interviews can also found in its entirety below, see appendix D.

5.2 Engineering Result

5.2.1 Functional Demands

The previous add-in developed by VisBook had a single function, namely: storing email correspondence from Microsoft Outlook into VisBook's PMS. To implement this single function into the new add-in several technical requirements need to be met. Specifically, VisBook must incorporate the functionality into their existing API, create a SOAP element template, and solve the authentication problem. Additionally, these requirements needs to be met in a fashion that does not create a processing time bottleneck.

In table 1, the functional demands of the current add-in are tabulated. We note that some demands have been met in this project, while other remain as future work. Further, we note that some of the requirements are subject to extending the existing VisBook system to include the necessary functionality.

For further details regarding the specific functionality or user story, see Demand Specifications in Appendix B.

As a user, i want to	Achieved	Not Achieved	Reasoning
Enable the add-in.	x		This is done through receiving a Manifest.xml file from VisBook that the end-user will have to install
Disable the add-in.	x		
navigate with navigational tabs.	x		
See guest information in a overview at the top.	x		Will be displayed with information conditionally gotten from VisBooks database
See guest relations.		x	Deemed not important and given low priority
See guest name.	x		
See guest email address.	x		
See if the guest has an existing client card.	x		Displays information about guest from Client card, but does not show "if" a client card exists.
See a guest expenditure tab.	x		
See guest expenditure visualized with graphical statistics.	x		
See an email tab.	x		
See previous email correspondence.		x	Microsoft Outlook API does not support access to all email items. Further implementation with Microsoft Graph API is required.
See a booking tab.	x		
See information about previous bookings.	x		
See information about future bookings.	x		

See attachments tab.	x		
See sent and received attachments.	x		User can see all attachments within the "active" email window (selected email). But here the issue is the same as with the previous email correspondence.
See feedback tab.	x		Changed from feedback window to "notes" window, as per request through usability test feedback.

Table 1: User stories table

5.2.2 Non-Functional Demands

The most prevalent Non-Functional Requirements encompass performance, scalability, compatibility, reliability, availability, maintainability, security, and usability, as well as localization and portability to a lesser extent [3]. Within the scope of this project, we have successfully addressed all of these Non-Functional Requirements except for localization and portability. Moreover, we made concerted efforts to adhere to the Web Content Accessibility Guidelines (WCAG), which provide a framework to ensure inclusive accessibility for all users [5]. The WCAG guidelines emphasize the importance of:

WCAG 2.1 has four main guiding principles (abbreviated as POUR):

- **Perceivable**
  - Users must be able to perceive the information being presented.
- **Operable**
  - Interface forms, controls, and navigation are operable.
- **Understandable**
  - Information and the operation of user interface must be understandable to all users.
- **Robust**
  - Users must be able to access the content as technologies advance.

We followed the WCAG principles, while also incorporating preferences that VisBook desired. Specifically, VisBook wished to have a color and design that fit the general outlook of their PMS. The color palette and contrast between foreground and background elements were controlled using the webAIM [10] contrast checker, seen in figure 16 and 17. Furthermore, we used Vuetify and Vue, which have built in accessibility (a11y), and follows the design guidelines of WCAG.

Vue, as a web development framework, adheres to established guidelines that prioritize accessibility best practices. These guidelines encompass various aspects such as labeling, linking, headers, placeholder usage, and content visibility control, among others. By emphasizing these practices,

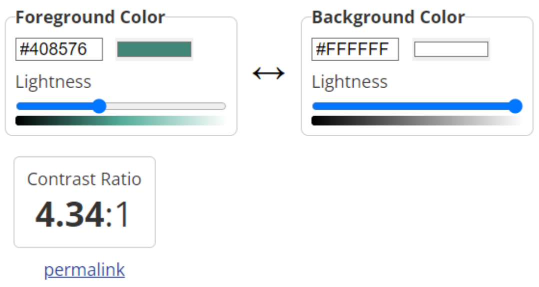


Figure 16: WebAIM contrast score, describes the contrast between foreground and background in a numbered ratio.

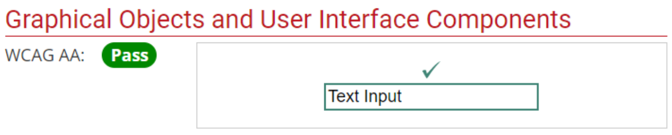


Figure 17: WebAIM contrast check, displays that the two colors conforms to WCAG AA standard for Graphical Components and User Interface.

Vue aims to facilitate the creation of accessible web applications by providing developers with the necessary tools and frameworks. This approach ensures that developers can readily incorporate accessibility features into their web apps, thereby enhancing the overall accessibility and usability of the applications.

As mentioned in the 3, Vuetify is a library of ready made components, that have built in functionality and design and conforms to the standard WCAG guidelines.

5.2.3 Workflow

One of the goals of this project was to determine how the development of the add-in could help users streamline their workflow and increase their effectiveness. Visbook wanted to reduce the need to open the product management systems to find information regarding the guest. In figure 18, we have created a flowchart, visualizing that our solution has removed one key component in Visbook's user workflow. Note that in the figure, we have included 'login'; if the user is already authenticated, this component would also be removed.

5.3 Administrative Result

5.3.1 Project Plan

To ensure adherence to the project plan and effective tracking of significant milestones, a Gantt chart was developed in the early stages of the project. The Gantt chart, provided in Appendix E, is organized into four distinct segments: "Planning," "Research and Development," "Poster," and "Report." Key dates, such as the due date for the "Pre-project Plan" and "Poster Presentation," are clearly indicated. Considering the project's requirements, additional subdivisions within the Gantt chart were deemed unnecessary. Instead, the team adopted an Objectives and Key Results (OKR) system to manage weekly tasks. This approach enabled flexibility in the development process, allowing for changes in direction as needed, aligning with the agile workflow methodology.

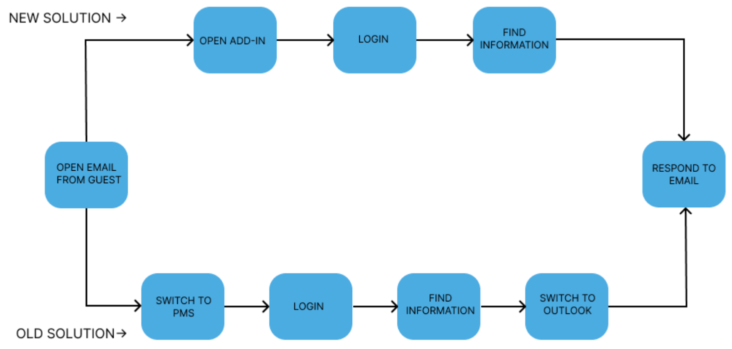


Figure 18: Old and New Workflow

5.3.2 Time management

Efficient time management and optimal task distribution were crucial factors in the successful development of the add-in. The team's goal was to maintain an even workload throughout the project period, limiting workdays to a maximum of eight hours and the total to approximately 500 hours ±50 per team member. However, the team members were aware that additional hours might be necessary during later stages to achieve the desired functionalities in the add-in.

As evident in the timetable summary, see appendix E, the team primarily worked together at school, with an almost identical workload. Appendix E provides a complete overview of the timetables and status report. Although there were instances of travel and illness resulting in remote work, good communication and planning prevented any significant complications.

To document the timetable, the team categorized the activities into 11 groups: documentation, programming, report writing, preparation, review, presentation, meeting, usability testing, reading, and attending lectures. This approach enabled a visual view of the number of hours allocated to each category, as illustrated in Figure 19. The pie chart indicates that programming, report writing, and documentation were the most time-consuming activities. The pie chart in Figure 19 shows a total of 979.5 hours for both team members, well within the margin of 500 hours each ± 50.

Figure 20 shows a graph representing number of hours spent on the project per week. As seen in the figure, there is an increase in the working hours as the team got closer to delivery date.

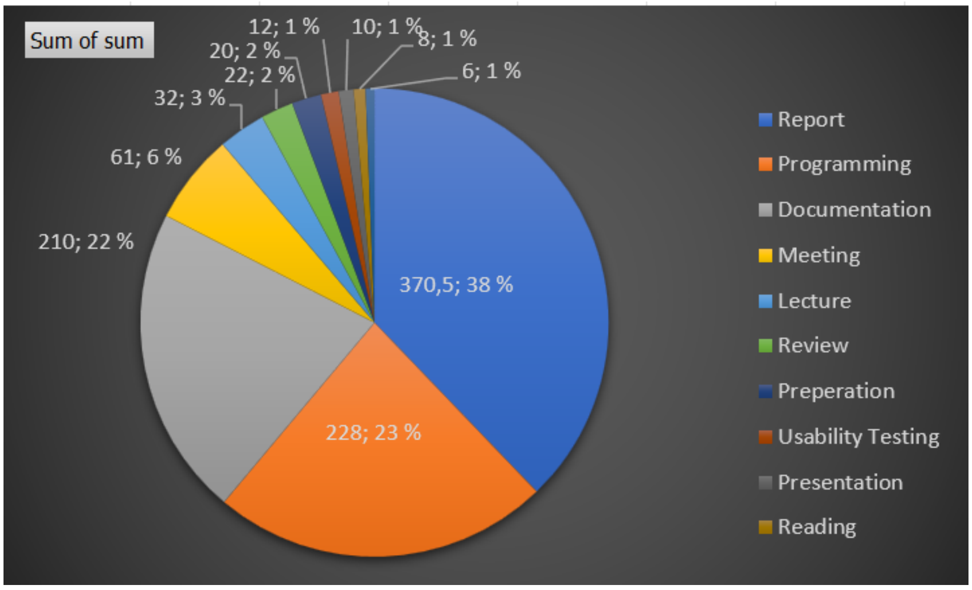


Figure 19: Pie chart over hours on category

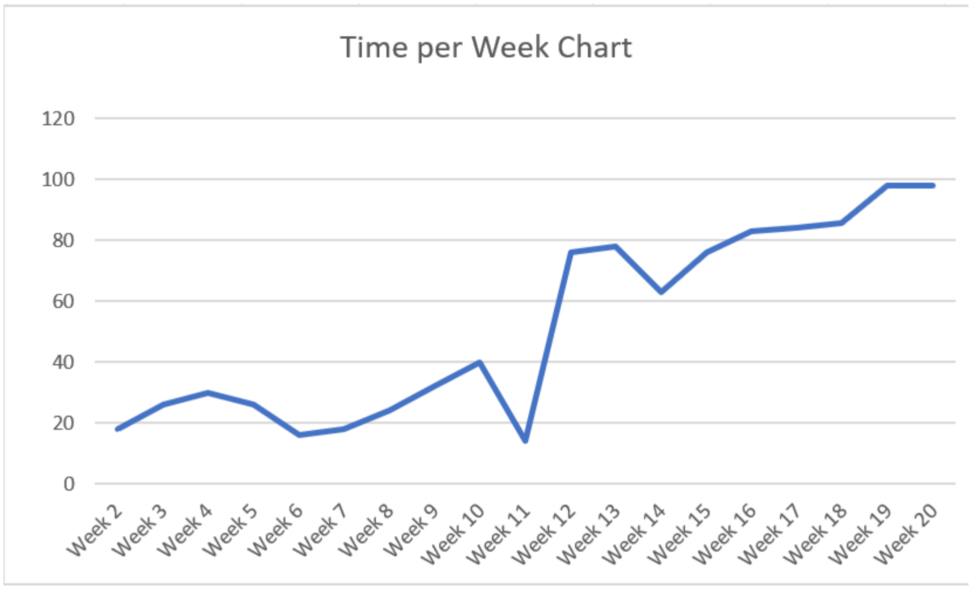


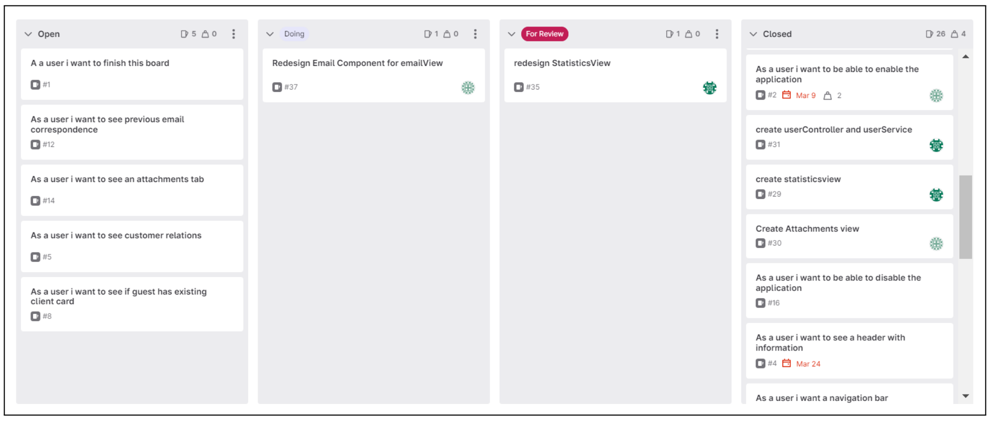
Figure 20: Linechart over hours per week

5.3.3 Development Methodology

The project team utilized a combination of Objectives and Key Results (OKRs) and the Kanban board methodology throughout the entire duration of the project.

Every user story derived from the vision document was incorporated into GitLab's integrated issue board, where it was subsequently transitioned across various production stages, namely 'Open', 'Doing', 'For Review', and 'Done'. In cases where it was deemed suitable, a user story was further subdivided into two distinct tasks. For instance, the user story "As a user, I want to login" was divided into "Implement authentication backend" and "Create login view". This division facilitated concurrent work on a single user story.

Task selection was linked to the team's weekly objectives as determined during the Monday Commit process 4. Each Monday, the team reviewed the project's original objectives and key results, and identified three to five goals to achieve by Friday. The selection of the user story was naturally aligned with the goals established during this process. Any necessary adjustments to objectives or tasks were quickly made, including reordering tasks within the production stages.


Open 5 0

A a user i want to finish this board
#1

As a user i want to see previous email correspondence
#12

As a user i want to see an attachments tab
#14

As a user i want to see customer relations
#5

As a user i want to see if guest has existing client card
#8

Doing 1 0

Redesign Email Component for emailView
#37

For Review 1 0

redesign StatisticsView
#35

Closed 26 4

As a user i want to be able to enable the application
#2 Mar 9 2

create userController and userService
#31

create statisticsview
#29

Create Attachments view
#30

As a user i want to be able to disable the application
#16

As a user i want to see a header with information
#4 Mar 24

As a user i want a navigation bar

Figure 21: Kanban board in week 19

In this section, the objective and key results (OKRs) that were obtained by the team during the early phases of this project and their alignment with the desired accomplishments are outlined. In total we had 28 Monday Commits and 24 Friday Wins, to see complete list of the OKR and commits, see appendix F.

Our Objective and key results were:
Create a user friendly Microsoft Outlook add-in MVP

- Key Result: Implement over 70 percent of the desired user stories

As mentioned in the engineering result, we managed to incorporate 11 of the original 15 user stories, bringing the overall completion rate to 73 percent.

- Key Result: Conduct at least three usability tests with a clickable prototype

Referring to usability test result, see appendix D, we conducted three usability tests with personnel from three different hotels.

- Key Result: Invest at least 200 programming hours on the product

Shown in figure 19 we have well over 200 hours in production of the add-in (detaljer om timer)

- Key Result: Have meetings with Visbook representatives two times a month for feedback and review of progress

The team had regular meetings with Visbook every two weeks. In addition, the team had multiple discussions pertaining to design and functionalities with different individuals in Visbook. See process Appendix E for details about all meetings and the corresponding meeting notices.

6 Discussion

Through this section, we will discuss the results from the previous chapter.

6.1 Scientific Result

6.1.1 Interview

As outlined in the result 5 and methodology 4 sections, our research approach involved conducting interviews and usability tests, which served as the primary means of gathering information during the initial stages of the development process. We engaged in meetings with three willing participants, utilizing Microsoft Teams as the communication platform. While conducting additional interviews could have provided further valuable insights for our project, it was necessary to consider the associated time commitments for post-interview analysis and meeting logistics. Therefore, we concluded that the three interviews conducted were sufficient for our research purposes.

During the review of our pre-interview process, we recognized the potential for further improvement by incorporating a test-interview phase. This would have involved selecting a participant from Visbook or associates to provide feedback on the overall interview process. Such a test-interview could have been beneficial in refining our interview guide and formulating more effective interview questions, leading to enhanced data collection and insights.

6.1.2 Sources of Error

The primary sources of error in the scientific findings undoubtedly stem from the underlying data foundation. Conducting interviews during the early phase of development led us to rely on the results obtained from the clickable Figma model, rather than conducting interviews where respondents directly interacted with the Minimal Viable Product (MVP). Consequently, it would have been more appropriate to initially conduct a more general interview to determine the requirements and then follow up with a mid-project interview phase, where participants would engage with the early-stage MVP to obtain more accurate and comprehensive responses. This iterative approach would have helped mitigate potential errors and yielded more reliable scientific results.

It is important to consider that the selected subjects represented active users of VisBook's system (PMS). This fact introduces the possibility of preconceived opinions, which could potentially affect the users responses and result in biased data. Simultaneously, it is worth noting that the users, whom we interviewed, have valuable insight and experiences based on their usage which mitigates the data bias.

Through the interview process, we have gained valuable insights into the time-consuming and demanding nature of conducting interviews. We acknowledge that preparation plays a vital role in ensuring the effectiveness of semi-structured interviews and the quality of the responses.

Reflecting upon our process, conducting interviews at multiple stages in the software development life cycle (see SDLC 4) could have been beneficial, as it would have generated more feedback and insight into the add-in. However, we did not incorporate multiple interview stages due to time constraints and the potential to delay the development of the add-in.

6.2 Engineering Result

6.2.1 Functional Demands

During the initial meeting, a comprehensive list of functional requirements and their corresponding priorities were meticulously documented in conjunction with VisBook. This process established a baseline that determined the priority order in which tasks needed to be completed to attain the desired state of the add-in.

As described in the 5 section, the primary functional requirement, pertains to the capacity to store the email exchanges between guests and hotel employees. Due to the lack of support for this functionality in VisBook's system at the development stage, its implementation was unfeasible. The issue of incorporating this feature into their system was subsequently addressed with the VisBook team, who would oversee the implementation of the necessary requirements, security considerations, and overhead analysis. Consequently, the MVP of the new add-in does not currently possess the capacity to store email correspondence.

The selection of the technology stack for the project presented us with several challenges. Microsoft's interaction with elements such as email items made it essential to choose a suitable technology stack. Following discussions with VisBook, we initially selected Blazor WebAssembly[23] as the development framework. Blazor, a Single Page Application (SPA) framework, would have been ideal for our project, given its extensive capabilities. However, we encountered difficulties as Blazor WebAssembly did not support the Service connection with the SOAP API that was required for consumption. Consequently, we attempted to implement the project using the Blazor Server solution. Unfortunately, the scarcity of well-documented information on the capacity of the Blazor Server in combination with Microsoft Outlook add-in required us to look into alternate solutions.

Subsequently, the project was revamped, and a Vue frontend was implemented alongside a .NET C backend. To establish a connection with Outlook and consume the SOAP API, we incorporated the Yo-office package from Yeoman-generator, as illustrated in the 4.4.1 section. The adoption of this approach ensured that we could effectively acquire the essential data from VisBook, and access data through Microsoft's Outlook API.

Throughout the entirety of this project we have gained valuable knowledge of software development and integration towards existing solutions. We have encountered first hand, how difficult it is combine old and new technologies. Being dependant on external solutions that effect and dedicate how we develop our solution has proved very educational. We have interacted with both employees of Visbook and their users to develop the add-in, utilizing the domain knowledge and experience of Visbook's employees has been crucial.

6.2.2 Non-Functional Demands

The design and color scheme was selected based on the knowledge gathered from the interviews along with knowledge shared by VisBook pertaining to this subject.

As per the results from the interviews, there are a lot of users of the PMS, which in turn means there are a lot of users of the add-in. Therefore, we had to keep simplicity in mind when designing the visuals, as well as the User Interaction. This was done through familiar icons and colors. Since the users of today are familiar with VisBook and Outlook, we chose similar symbols. Although, a final rendition of the add-in would be more familiar with symbols that are similar to the PMS, and for this to happen, the add-in would first need to be deployed, tested, and then a design on the required icons should be done.

Furthermore, the add-in itself is required to conform to a level of privacy when dealing with VisBook and mail data. This should, under no circumstance, be divulged. Therefore, authentication for each user of the add-in, in accordance with VisBook's standards were implemented.

6.3 Administrative Result

The Administrative results provide an account of the project's progress, a progress timeline, and showcases the team's adaptability in implementing their prescribed work methodology and planned activities.

6.3.1 Project Plan

During the project's initiation, a Gantt chart was constructed to effectively oversee project milestones and temporal phases. As explained in the results section, the team collectively determined that committing to an excessive number of activities was disadvantageous, instead opting to focus on smaller weekly objectives using the OKR (Objectives and Key Results) framework. This strategic choice afforded the team the flexibility to adapt and plan tasks based on their perceived benefits. It is noteworthy that all activities listed in the original Gantt chart were accomplished. While a few milestones were shifted by a slight margin in terms of days, the overall impact on the project remained negligible.

6.3.2 Time management

Recognizing the potential challenges associated with diligently adhering to the Gantt chart, the team acknowledged the importance of devising a method to monitor and track hours spent across different categories. The document titled "Timetables w Report," presented in process-documentation in appendix D, provides a comprehensive overview of all activities undertaken by the team members, accompanied by their corresponding hours and concise descriptions. The document, created in Excel, features individual sheets dedicated to each week with a brief summary of the week. Additionally, a supplementary chart (refer to Figure 19) was developed to visually depict the distribution of hours among various categories.

In this project, an estimated total of 1000 hours, equating to 500 hours per team member, was initially established. These estimations were formulated with limited prior knowledge of the domain of the project, and the project tasks. Consequently, slight disparities arose between the estimated and actual time expended in the different categories. To highlight an example, our estimate on 'programming' and 'report' where 250 and 320 hours, actual time spent on these categories were 228 and 370 hours.

The team focused on solving the assigned tasks, not on attaining the time estimates outline early in the project. However, it is worth noting that the discrepancies between the planning and reality were minor in nature, affirming the team's realistic projection and planning capabilities based on their collective knowledge and experience in software development.

6.3.3 Development Methodology

The objectives and key results combined with the Kanban board suited the team well. We had no significant issues with our development methodology of choice. This development methodology allowed us to focus more on programming by excluding time consuming ceremonies and activities. By having weekly meetings discussing goals and achievements (Monday Commit, and Friday Wins), the team was able to swiftly adapt to new tasks or directions from Visbook. The team managed to create weekly goals in a good fashion, but could sometimes struggle with relating them to the original key results. This process was a good exercise as it reminded us what our original goal was.

During the initial phase of the project (Week 2 to Week 12), an obligatory study course was taken concurrently with the project. As a consequence, the team had limited availability, with only two days per week allocated for project work. In order to accommodate this constrained schedule, the decision was made to defer from the full implementation of Objectives and Key Results (OKR) until Week 13. During the period spanning Week 2 to Week 12, the team effectively managed their time and task by setting goals on a four-week basis. See appendix F to see complete list of OKR's.

The user stories outlined in the vision document were promptly incorporated into our Kanban board, providing a visual representation of the completed user stories and enabling progress monitoring. Primarily the programming tasks derived from the initial user stories and aligned with weekly objectives, the Kanban board served as a valuable tool for task management. The team adhered to established protocols for conducting peer reviews of each other's work prior to merging with the main branch, thus ensuring code quality and minimizing the likelihood of errors.

7 Conclusion and Further Work

In our study (see Section 2), we addressed the following problem: *"How can the communication between users and guests be enhanced through the implementation of a Microsoft Outlook add-in?"*. This report presents an analysis of the findings obtained from interviews, literature, and usability tests conducted during the development of the add-in. Our main objective was to streamline the communication process and provide users with easy access to the information necessary for effective correspondence with guests directly within Outlook. By eliminating the need for users to switch between multiple windows to gather relevant information, our solution enables prompt responses to guest requests.

7.1 Conclusion

The development and successful implementation of our add-in progressed smoothly following the identification of a technology stack that aligned with the requirements of VisBook and our team. The success of the project, seen from the point of view of the different stakeholders, VisBook, supervisor, and the team, can be attributed to several key factors, including our selection of an appropriate work methodology, effective prioritization, a robust research approach, and efficient time management. During the later stages of the project, we conducted a comprehensive review of the Minimal Viable Product (MVP) with the project owners at VisBook, receiving highly valuable feedback that validated our prudent decisions regarding technology choices and design considerations.

The conducted interviews yielded valuable feedback from individuals with extensive experience in the hotel business, providing profound insights into the daily operations of users within the property management system. These interviews not only aroused our interest as developers but also proved to be an invaluable resource during the development of the add-in. Notably, one interviewee expressed that, "*In the span of one month, this (add-in) might actually save up to several days of work*" emphasizing the potential time-saving benefits of our solution.

For the duration of the project, we as a team, have gained knowledge about system design, and team work, as well as integration towards existing solutions. Furthermore, we have experienced that this project has required us to combine a multitude of technologies (see 3) to comply with the requirements and demands from VisBook's system. As such, the team has been through a comprehensive learning process, which has allowed us to develop a functioning Microsoft Outlook add-in.

Upon reflection of the initial problem description, we can assert that a properly implemented Microsoft Outlook add-in, integrated with VisBook's property management system, has the potential to significantly reduce the time spent on each guest interaction through email.

7.2 Further Work

For the further work of this add-in, it would be natural to first implement the functional demands in their entirety before exploring additional functionalities.

7.2.1 Email Correspondence

The user story 'See previous email correspondence' was not implemented during the project duration. It was not possible to gather emails already sent via Visbook's PMS due to limitations in Visbook's API, nor was it possible to send emails from Microsoft Outlook into Visbook's systems. The system limitations regarding Visbook's integration API would have to be fixed before this functionality could be integrated into the add-in. One functionality that could be resolved without updating Visbook's system would be to collect and visualize all email correspondence between users and guests already listed in Microsoft Outlook; this is possible with the use and integration of Microsoft's Graph API.

7.2.2 Sent/Received Attachments

Connected to the previously mentioned missing functionality of email correspondence, it was desired to visualize all sent attachments between the guest and the user; this includes booking confirmations, invoices, receipts, etc. This has also not been included for the same reasons, limitations in Visbook's systems, and no implementation of Microsoft Graph API.

7.2.3 Additional

Multiple additional functionalities have arisen upon developing the add-in; these include the option to choose the desired text language of the application, the possibility to add text to the notes section, and the option to see full details of the relevant email, booking, and attachments in full.

For the project as a whole, it would be desirable to integrate this add-in in an environment that handles continuous integration and continuous development (CI/CD). Incorporating CI/CD would allow Visbook to deploy updates quickly and efficiently.

8 Social Impact

Our developed add-in enhances and streamlines the workflow within Visbook's product management system, offering potential benefits in terms of time efficiency and reduced workload for users. Such an add-in can have valuable impacts on society and the environment.

Visbook serves a substantial customer base of over 1000 customers in the Nordic countries, with multiple employees utilizing their systems on a daily basis. Our usability testing yielded feedback from a hotel manager, who stated, "*Over the course of a month, this add-in could potentially save several days of work*". If this statement holds true for all Visbook users, our add-in has the potential to reduce thousands of working days annually.

By reducing several days of work per month, VisBook's customers gain valuable time, allowing them to serve more guests within the same time frame. The add-in optimizes resource utilization by minimizing repetitive tasks and streamlining workflows. Consequently, the reduced time spent on computers, tablets, or phones leads to decreased energy consumption, resulting in lowered total energy usage and carbon emissions per guest, thereby generating positive environmental implications.

Moreover, the economic impacts of this add-in on various businesses should be considered. Increased efficiency and productivity may result in cost savings that can be reinvested in development, job creation, or research, fostering financial and economic growth within the business and society.

8.1 Sustainability

Analyzing the United Nation's seventeen Sustainable Development Goals (SDG), we want to highlight two of them in alignment with our developed add-in. [21]

SDG number 8, Decent work and Economic growth, by reducing time required for repetitive task, our add-in could help reduce cost and shift focus to more important and innovative work that could lead to financial growth.

SDG number 12, Responsible consumption and Production, by reducing activities and time spent on power-consuming appliances, such as computers, tablets or phones, we could argue that this would be a more responsible use of electricity and electrical hardware. With reduction in usage, we contribute to longer lifespan in the hardware and a reduction in electrical waste.

Bibliography

[1] Muhammad Ovais {Ahmad, Jouni Markkula and Markku Oivo. 'Kanban in software development: A systematic literature review'. In: 2013, pp. 9–16. URL: https://ieeexplore.ieee.org/abstract/document/6619482?casa_token=rC8Gg9naFVwAAAAA:eCjkYkSB2NLs8B4wJQ4BM0SfRqk4wFu1u0VeGd

[2] Alexandra. *What is SDLC? Understand the Software Development Life Cycle*. English. Apr. 2020. URL: https://stackify.com/what-is-sdlc/ (visited on 18th Apr. 2023).

[3] Altexsoft. *Non-functional Requirements: Examples, Types, How to Approach*. English. Blog. July 2022. URL: https://www.altexsoft.com/blog/non-functional-requirements/ (visited on 9th May 2023).

[4] Atlassian. *What is Agile?* URL: atlassian.com/agile (visited on 9th May 2023).

[5] Ben Caldwell et al. *Web Content Accessibility Guidelines (WCAG) 2.0*. English. June 2018. URL: https://www.w3.org/TR/WCAG21/ (visited on 9th May 2023).

[6] Richard Castle. *PMS Technology: The evolution and future of the hospitality platform*. English. Dec. 2021. URL: https://www.cloudbeds.com/articles/hotel-property-management-system-guide/ (visited on 26th Apr. 2023).

[7] Gong Chao. 'Human-Computer Interaction: Process and Principles of Human-Computer Interface Design'. In: IEEE, 2009, pp. 230–233. (Visited on 22nd Mar. 2023).

[8] GitLab. *Burndown and burnup charts*. URL: https://docs.gitlab.com/ee/user/project/milestones/burndown_and_burnup_charts.html (visited on 8th May 2023).

[9] *Gitlab Merge Requests*. URL: https://docs.gitlab.com/ee/user/project/merge_requests/.

[10] Institute for Disability Research, Policy, and Practice. *Contrast checker*. URL: https://webaim.org/resources/contrastchecker (visited on 9th May 2023).

[11] Jedox. *SOAP (WSDL) Connection*. English. Apr. 2023. URL: https://knowledgebase.jedox.com/integration/connections/soap-connection.htm (visited on 3rd May 2023).

[12] Kanbanize. *What is Kanban? Explained for beginners*. English. URL: https://kanbanize.com/kanban-resources/getting-started/what-is-kanban (visited on 9th May 2023).

[13] Nasraldeen Khleel and Károly Nehéz. 'Comparison of version control system tools'. In: 10.Multidiszciplináris Tudományok (2020), pp. 61–69. ISSN: 2062-9737. URL: https://www.researchgate.net/publication/342740384_COMPARISON_OF_VERSION_CONTROL_SYSTEM_TOOLS.

[14] Afifa Lodhi. 'Usability Heuristics as an assessment parameter: For performing Usability Testing'. In: *Usability Heuristics as an assessment parameter*. Vol. V2. IEEE, 2007, pp. 256–259. URL: https://ieeexplore.ieee.org/document/5608809 (visited on 22nd Mar. 2023).

[15] *Material design*. 2014. URL: https://m3.material.io/ (visited on 9th May 2023).

[16] Microsoft. *OffiveDev/generator-office*. Mar. 2023. URL: https://github.com/OfficeDev/generator-office (visited on 27th Apr. 2023).

[17] Timothy Nelson. *Some strategies for qualitative interviews guide*. English. May 2012. URL: https://sociology.fas.harvard.edu/files/sociology/files/interview_strategies.pdf (visited on 13th Apr. 2023).

[18] Oracle. *What is a Hotel Property Management System(PMS) ?* English. URL: https://www.oracle.com/be/industries/hospitality/what-is-hotel-pms/ (visited on 17th Apr. 2023).

[19] ProductPlan. *Minimum Viable Product (MVP)*. URL: https://www.productplan.com/glossary/minimum-viable-product/ (visited on 9th May 2023).

[20] Isaac Schlueter, Laurie Voss and Rod Boothby. *npm*. Jan. 2014. URL: https://www.crunchbase.com/organization/npm (visited on 8th May 2023).

[21] *Sustainable Development Goals*. English. org. URL: https://sdgs.un.org/goals (visited on 18th May 2023).

[22] Tekna. *Hva er OKR, og hvorfor er det viktig for bedrifter?* Oct. 2022. URL: https://www.tekna.no/kurs/innhold/hva-er-okr-og-hvorfor-er-det-sa-viktig-for-bedrifter/ (visited on 9th May 2023).

[23] The .NET Foundation. *Blazor WebAssembly*. URL: https://blazor-university.com/overview/what-is-blazor/ (visited on 14th May 2023).

[24] The fullstory education team. *Qualitative vs. quantitative data: what's the difference?* English. Oct. 2021. URL: https://www.fullstory.com/blog/qualitative-vs-quantitative-data/ (visited on 2nd May 2023).

[25] Aksel Tjora. *Kvalitative forskningsmetoder i praksis*. Norsk. 2nd ed. Gyldendal Norsk Forlag AS, 2021. ISBN: 9788205546530. URL: https://www.akademika.no/realfag/naturvitenskap-filosofi-teori-og-metode/kvalitative-forskningsmetoder-i-praksis/9788205546530?gclid=Cj0KCQjwu-KiBhCsARIsAPztUF3T4mCnGTiIUBpAMSGEit0KfQ__mpKMBgO5FrdBwExOVYcUfXmvsrYaAh37EALw
wcB (visited on 9th May 2023).

[26] Oliver Trunkett. *SDLC methodologies: From waterfall to agile*. Aug. 2020. URL: https://www.virtasant.com/blog/sdlc-methodologies (visited on 1st May 2023).

[27] Evan You. *Vue.js*. 2014. URL: https://vuejs.org/about/faq.html (visited on 10th Apr. 2023).

[28] Dmitriy Zaporozhets and Valeriy Sizov. *GitLab*. Oct. 2011. URL: https://about.gitlab.com/company/history/ (visited on 8th May 2023).

[29] Hao ZHOU and Yu-Ling HE. 'Comparative Study of OKR and KPI'. In: 2018. ISBN: 978-1-60595-552-0. URL: http://u.camdemy.com/sysdata/doc/4/4a6b816a1fb5cebb/pdf.pdf.

Appendix A Vision Document

Kunnskap for en bedre verden

IDATT2900 - Bachelor project

Vision Document
Project 64

Author:
Amundsen, Trygve
Helmersen, Martin

Date 20.05.2023

Revision History			
Date	Version	Description	Author
13.02.2023	1.0	First submission	Martin Helmersen, Trygve Amundsen
20.05.2323	1.1	Revision before delivery	Martin Helmersen, Trygve Amundsen

Table 1: Revision history

Table of Contents

1 Introduction

The intent with this document is to describe the vision regarding the following bachelor assignment for the subject IDATT2900-064. The bachelor assignment is written in the spring period from January to May.

2 Summary of Product and Problem

2.1 Background

Visbook is a PMS (Property Management System). PMS is a system to organize, schedule and handle the day-to-day transactions and events in an accommodation business. They have existed since 1995 and has over 1000 customers in the Nordic countries.

Visbook customers are accommodation businesses such as hotels, camping sites, resorts, bars or spa's. Visbook offers a system to do bookings, check-in and check-outs, Point of sale (POS), Event planning and Food and Beverage sales (Restaurant/Bar). Visbook handles all the services needed for the particular accommodation.

Visbooks customers have information about their guests within the product management system (PMS). Information about the guest get stored in a digital client-card, a digital client-card is an electronic registry of information, which contains specific personal information about the guest, records of bookings, planned events and other transactions made. Additionally all sent attachments, documents, e-mails and followups get stored on the card.

The data is stored to improve the experiences for the guest, but also to improve Visbooks marketing strategies.

When employees of the accommodation business, henceforth referred to as users, communicate with guests by e-mail, it is important for them to have the correct information about the guest available.

2.2 Problem summary

The problem from Visbook was presented such; Our customers (hotels) send offers to their guests through Visbook´s PMS. If a guest is to respond to the offer or send an email to the hotel, the communication is continued through Microsoft Outlook. Since Microsoft Outlook and the PMS are two different programs, the user (manager, receptionist, owner) is forced to switch between Outlook and the PMS to retrieve necessary information. The switching in programs is time consuming and could result in the guest receiving faulty answer or wrong information.

To resolve this problem, visbook desires a Microsoft Outlook add-in that shows guest information from the PMS into Outlook. Relevant information could be (not limited) to name, email, bookings, expenditure statistics and notes. The add-in should also be able to store emails received in Microsoft Outlook to the Visbook´s product management system.

Visbook has an previous developed a Microsoft Office Outlook add-in which they want remade. They want an entirely new and modernised add-in with a simplistic look. The add-in should have en easy to learn User Interface (UI) to ensure that any user of the add-in, young or old, will be able to use it and understand it without being overloaded with buttons and options. For getting information about the guest, this add-in would need to be connected to Visbook´s system via their integration application programming interface (API).

Todays Problem	Is that Visbook don't have a sufficient add-in for displaying information about the guest in Microsoft Office Outlook. The correspondence made in Outlook is also not stored in the PMS.
Touches	Users, guests and developers of the PMS.
As a result	Any user of the platform will have to open several programs/windows and switch between them to ensure that they have up-to-date information to make the necessary decisions.
A successful solution will	Give all users the needed information about any guest within a single platform which will increase and simplify the learning and usage of said platform.

Table 2: Problem summary

2.3 Product summary

For	Users of Visbooks property management system
That	Use the system for day-to-day transactions and events
Visbook mail plugin and api integration	Is a system that handles any and all guest interaction prior to their booking and all interaction needed with the guest under their stay, within the same system.
Which	Enables the users to easily stay within the same program/browser to do any task required by them to while handling a guest.
Unlike	Their system today which does not include the availability of all information in a simple manner within the same system.
Our product	Incorporates the full circle of communication and information, making the platform a more stand alone platform regarding e-mail and information.

Table 3: Product summary

3 Stakeholders and Users

3.1 Stakeholders

Name	Description	Role
Visbook	Company contact Sofie Moene, Otto Meisingset, and Peter Haza	Technical guidance and specifications
NTNU	Ali Alsam	Report guidance

Table 4: Stakeholders

3.2 Users

Name	Description	Role under development	Represented by
The Group	Developers of the add-in	Get familiar with developing within frameworks, and use this to develop the requested tools and software	Martin Helmersen, Trygve Amundsen
Survey subject	Gives feedback concerning existing application and requests for the new concept	Tests the plug-in and gives feedback under development	Employees of Hotels or similar bookable services
Visbook Employees	Visbook representatives	Directs us according to technical and systemical requirements and needs	Sofie Moene, Otto Meisingset, Peter Haza

Table 5: Users

3.3 User Environment

The designated user environment for our product is reception area, office, customer service center and other locations where bookings or management takes place. The system is designed to link two systems together, the outlook email platform and Visbook's PMS.

3.4 Summary of user needs

3.5 Alternative products

As Visbook mentioned in the meeting we had (see appendix for minutes of meeting), there are no complete PMS (Property Management System) which encapsulates all in a "one package" kind of deal. This means that no PMS system can be seen as the same kind of system as they offer. This means that there are no related alternative products, but there are companies that offer solutions that while modular can be made into a product that resembles Visbooks' platform.

Needs	Priority	Influences	Todays Solution	Proposed Solution
Gather feedback	Critical	Design and development of the solution	-	-
Filter duplicates of information cards	High	The plug-in data	Non existent	Handles duplicates through a unique key
Integration with API	Critical	Ensures that everything will run on one platform	Non Existent	Use Visbooks' Integration API to seamlessly automate the e-mail process
Dashboard with information card details	High	User interaction	Non Existent	Side panel with all necessary information regarding the guest

Table 6: Stakeholders

4 Product

4.1 Products role in user environment

The teams product is meant as an addition to Visbooks' system. Where there is a computer running Visbook's product management system, the add-in system should be able to run. The products role is to make the communication between end-user (hotel employee) and their guests good, available and clear.

4.2 Dependencies and prerequisites

To use the product it will be required to have the latest version of Outlook and Visbooks' product management system available. Since the product is closely linked to Visbooks' systems and the integration platform, certain changes in their systems could potentially reduce the functionality of our add-in. This is also the case for Outlook and how they handle add-ins.

5 Product functionality

The Microsoft Outlook Add-in should include the following functionalities:

- The add-in should be placed in a side panel.
- The add-in should display information about the guest.
- The add-in should retrieve information from Visbook´s system.
- The add-in should store email correspondence from Outlook, into the PMS.
- The add-in should store attachments from Outlook, into the PMS.
- The add-in should display (if) multiple client-cards are connected to a guest.

6 Non functional requirements

Our non functional abilities are base on the FURPS model. FURPS stands for Functionality, Usability, Reliability, Performance and Support-ability.

- Functionality

The product should have required features. Parts of the product should be reusable and secure (not prone to security breaches and exploiting)

- Usability

The product should have an consistency in design with focus on responsiveness and documentation.

- Reliability

The product should be durable with low failure frequency. The products cycles should be predictable with high recover ability.

- Performance

The product should a low response time, be efficient with high throughput and capacity.

- Support-ability

The product should be easy to maintain and service. The product should be modifiable and installability should be prioritized.

Appendix B Demand Spesification

NTNU

Kunnskap for en bedre verden

IDATT2900 - Bachelor project

Demand Specifications
Project 064

Author:
Amundsen, Trygve
Helmersen, Martin

Date 20.05.2023

Table of Contents

List of Figures

1 Introduction

The intention of this document is to describe, in depth, the functions and the technological aspects of our product.

2 User Stories

In the following user stories, a user is referred to someone using Visbooks property management system. This could be an receptionist, hotel owner, back office employees, camping site manager or other personnel responsible for day-to-day tasks.

- As a user, i want to enable the application at any given time, so that i can use it when necessary.
  - If not logged in, no email list is shown, and enabling add-in is not applicable
  - If logged in, email list is shown, enabling add-in displays the add-in window
- As a user, i want to disable the application at any given time, so that i can minimize visual disturbance when necessary.
  - The application should be able to close immediately on exit click.
- As a user, i want to have a navigation bar, so that i can efficiently look up information.
  - The navigation bar should include 5 tabs.
  - Each tab should change background color when hovering.
  - Each tab should have a tooltip when hovering.
  - Each tab should change background color when clicked (marked).
- As a user, i want to see a header displaying personnel information in the application, so that i can see stored information about the guest.
  - The header should have a 'show more' button which expands the window.
- As a user, i want to see important customer relations, so that i can accurately provide service.
  - Display customer relations information if exist
- As a user, i want to see the guest name in the application, so that i can accurately provide service.
  - If user not in digital client-card, show stock name.
- As a user, i want to see the guest email address in the application, so that i can accurately provide service.
  - Guest email should always be visible.
  - If email is clicked, new email is drafted in outlook.
- As a user, i want to see if the guest has a digital client card in the application, so that i can accurately provide service.
  - Should display 'yes' or 'no' corresponding to if the designated email has an digital client card.
  - If the email has multiple, should display 'mult'.
  - If 'mult' is clicked, action to change which digital client is displayed should start.
- As a user, i want to see a expenditure tab in the application, so that i can see the expense history for the guest.

- As a user, i want to see graphical statistics on the guests expenditure, so that i can accurately provide service.
  - Show statistics on latest stay and payments if the guest has some.
  - Should include details about receipts and outstanding payments.
- As a user, i want to see a mail tab in the application, so that i view mailing history.
- As a user, i want to see previous email correspondence, so that i can accurately provide service.
  - List over all emails sent between guest and user, should include sender, receiver, date, subject and if-attachments.
- As a user, i want to to see a booking tab in the application, so that i can see bookings from previous and future bookings.
- As a user, i want to see information about previous bookings, so that i can accurately provide service.
  - List over all previous bookings, should include location, type, addition information, date, time and price.
- As a user, i want to see information about future bookings, so that i can accurately provide service.
  - List over all previous bookings, should include location, type, addition information, date, time and price.
- As a user, i want to to see a attachments tab in the application, so that i can sent and receives attachments.
- As a user, i want to see sent and received attachments, so that i can accurately provide service.
  - The attachments should be displayed in chronological order.
  - Each attachments should include information about sender, filename, date and location of attachment.
- As a user, i want to see a feedback tab in the application, so that i can see feedback from earlier stays.

3 Domain Model

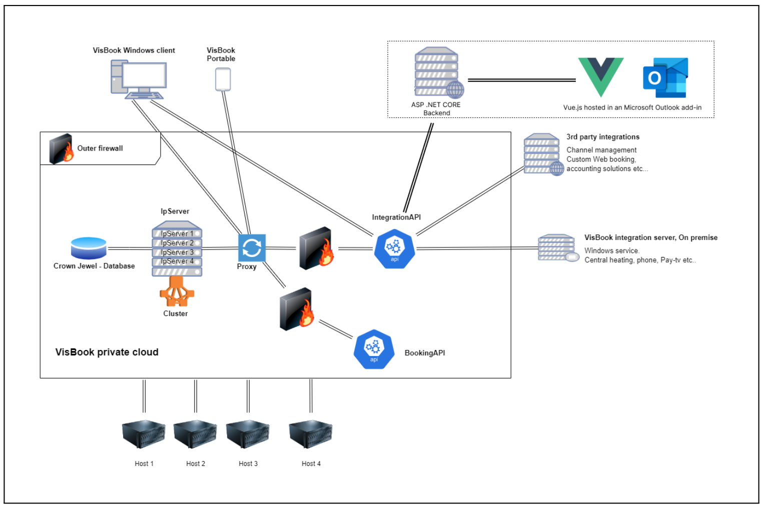


Figure 1: Domain model of Visbook´s architecture, with our implementation in dotted lines.

4 Prototypes

The clickable prototype was created and published in Figma, it can be found HERE.
Complete link below:

https://www.figma.com/proto/b346evIu8eHliQCJkqJI9h/BA64?node-id=88-250&starting-point-node-id=88%3A250

Appendix C System Documentation

IDATT2900 - Bachelor project

System Documentation
Project 064

Author:

Amundsen, Trygve

Helmersen, Martin

Date: 20.05.2023

Table of Contents

List of Figures

1 Introduction

The purpose of this document is to give a comprehensive system documentation for out developed Microsoft Outlook add-in. This includes architecture, project structure, class diagram, REST-server description, security considerations, an installation guide and documentation of source code.

2 Architecture

2.1 Components

The key components in our project are visualised in figure 1. The manifest file is uploaded into Microsoft Outlook, this lets our Vue.js frontend application run inside Outlook. The Vue.js application communicates with our RESTful API delivered by our ASP.NET CORE server application.

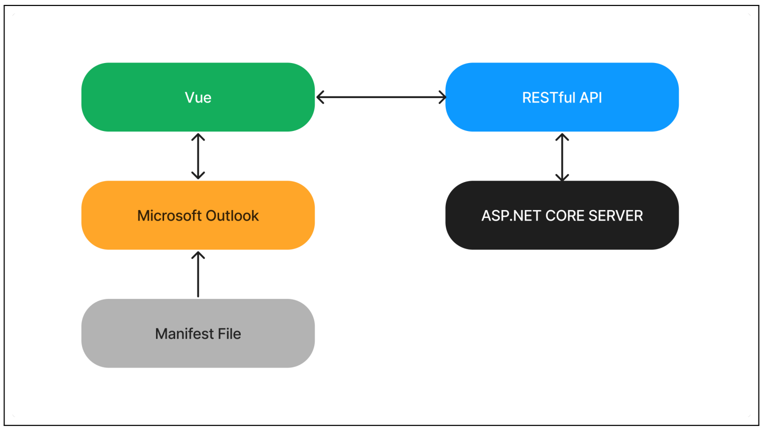


Figure 1: Components in this project

2.2 Workflow

In figure 2, our intended user behaviour is displayed. The diagram is split into three different segments, highlighting where the process is handled / started.

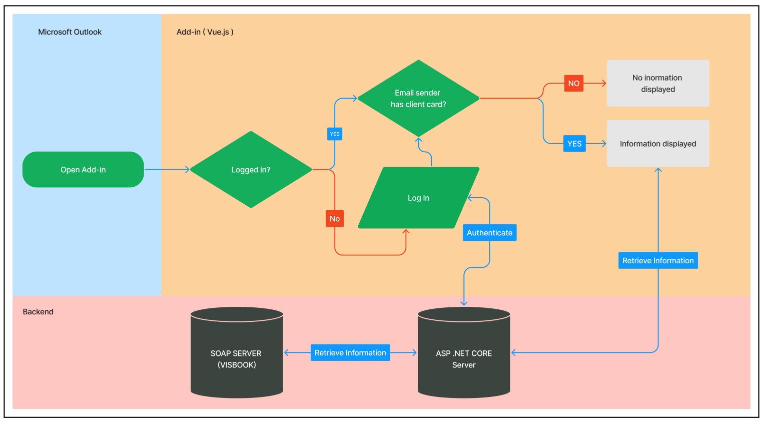
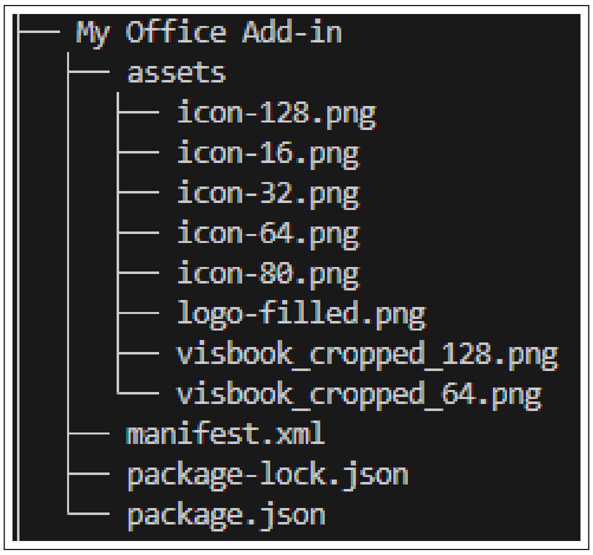


Figure 2: Intended workflow

3 Structure

3.1 Manifest

The Manifest project contains the manifest file, a package.json configure file and an asset folder with icons, see figure 3.

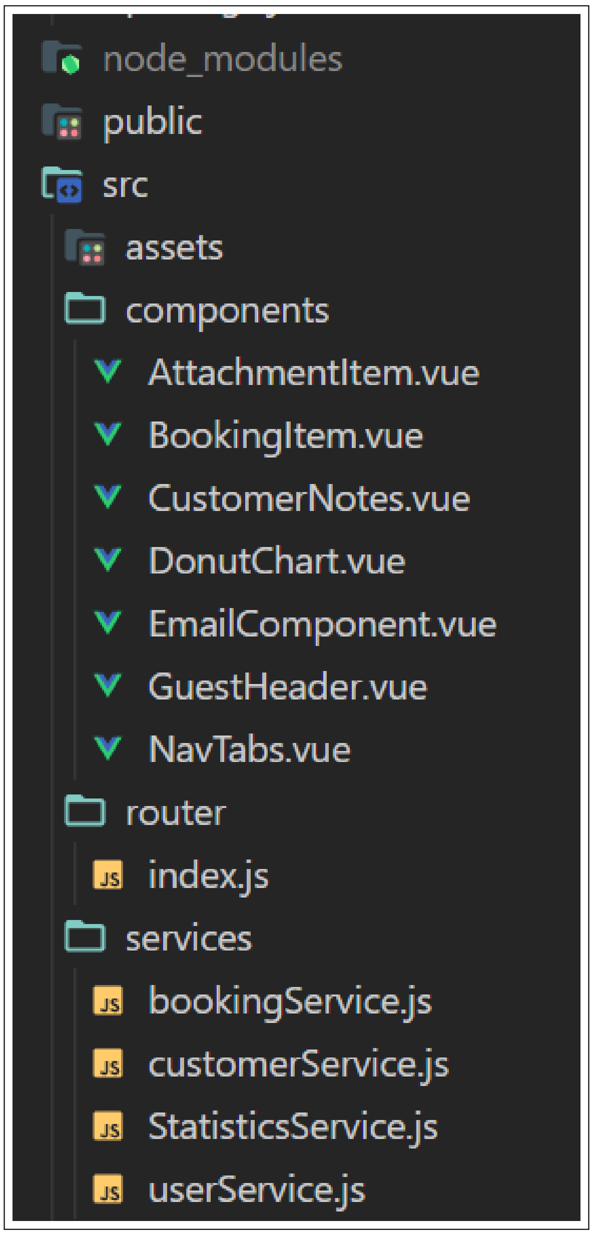


```
├── My Office Add-in
│   ├── assets
│   │   ├── icon-128.png
│   │   ├── icon-16.png
│   │   ├── icon-32.png
│   │   ├── icon-64.png
│   │   ├── icon-80.png
│   │   ├── logo-filled.png
│   │   ├── visbook_cropped_128.png
│   │   └── visbook_cropped_64.png
│   ├── manifest.xml
│   ├── package-lock.json
│   └── package.json
```

Figure 3: Structure for Manifest Project

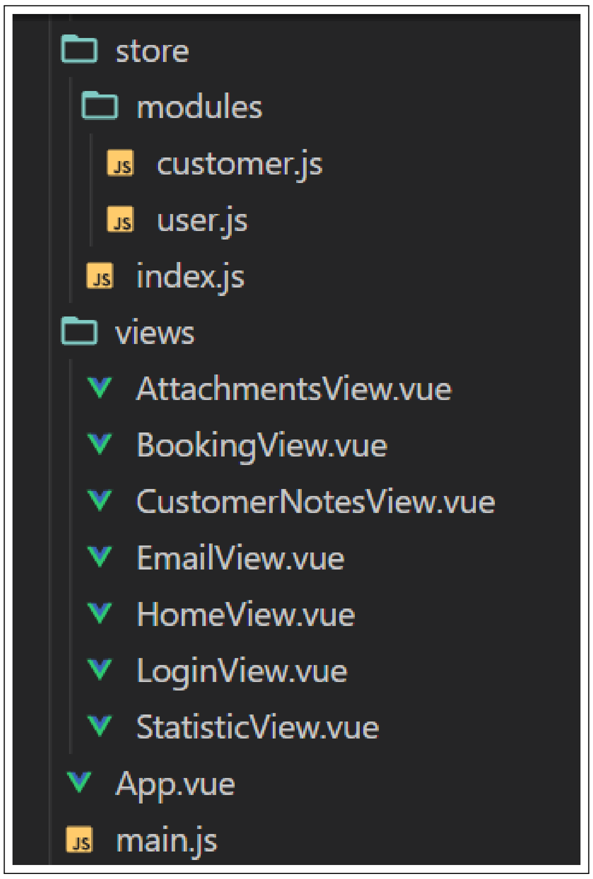
3.2 Vue

In the Vue project we have followed Vue's folder structure guidelines. Including store, views, router, assets, services and components in different folders. This project also includes important configuration files to connect to Microsoft Outlook and the API. See figure 4, 5, and 6.



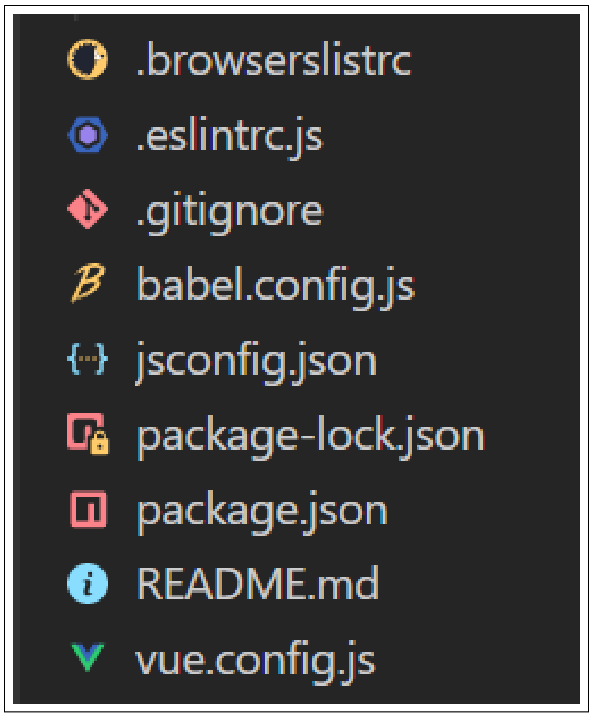
```
node_modules
public
src
  assets
  components
    AttachmentItem.vue
    BookingItem.vue
    CustomerNotes.vue
    DonutChart.vue
    EmailComponent.vue
    GuestHeader.vue
    NavTabs.vue
  router
    index.js
  services
    bookingService.js
    customerService.js
    StatisticsService.js
    userService.js
```

Figure 4: Structure for Vue Project



```
store
  modules
    customer.js
    user.js
  index.js
views
  AttachmentsView.vue
  BookingView.vue
  CustomerNotesView.vue
  EmailView.vue
  HomeView.vue
  LoginView.vue
  StatisticView.vue
App.vue
main.js
```

Figure 5: Structure for Vue Project

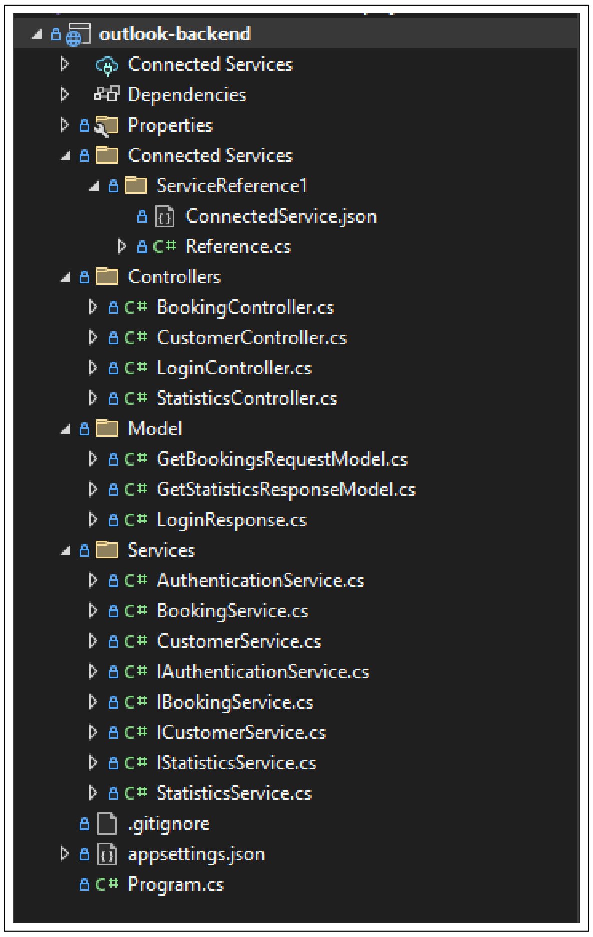


```
.browserslistrc
.eslintrc.js
.gitignore
babel.config.js
jsconfig.json
package-lock.json
package.json
README.md
vue.config.js
```

Figure 6: Structure for Vue Project

3.3 ASP.NET Core

For the backend projection we have divided into model, controllers and service. See figure 7. Every service class has a corresponding interface.



```
outlook-backend
  Connected Services
  Dependencies
  Properties
  Connected Services
    ServiceReference1
      ConnectedService.json
      Reference.cs
  Controllers
    BookingController.cs
    CustomerController.cs
    LoginController.cs
    StatisticsController.cs
  Model
    GetBookingsRequestModel.cs
    GetStatisticsResponseModel.cs
    LoginResponse.cs
  Services
    AuthenticationService.cs
    BookingService.cs
    CustomerService.cs
    IAuthenticationService.cs
    IBookingService.cs
    ICustomerService.cs
    IStatisticsService.cs
    StatisticsService.cs
  .gitignore
  appsettings.json
  Program.cs
```

Figure 7: Structure for ASP.NET Core project

4 Class Diagram

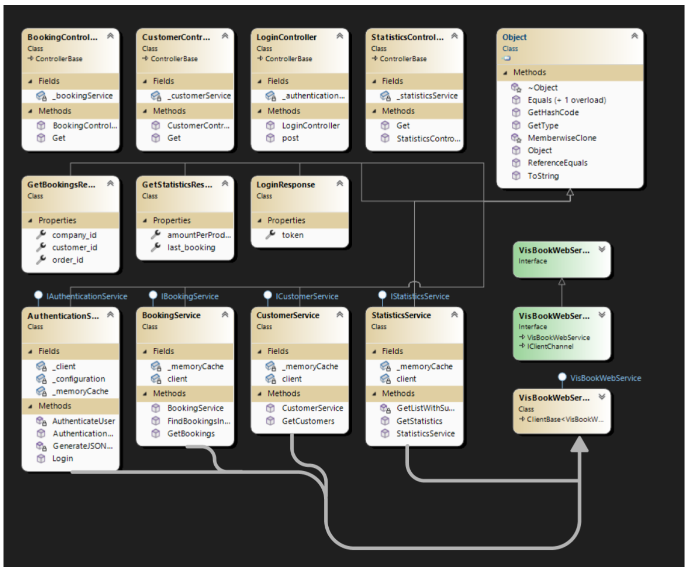
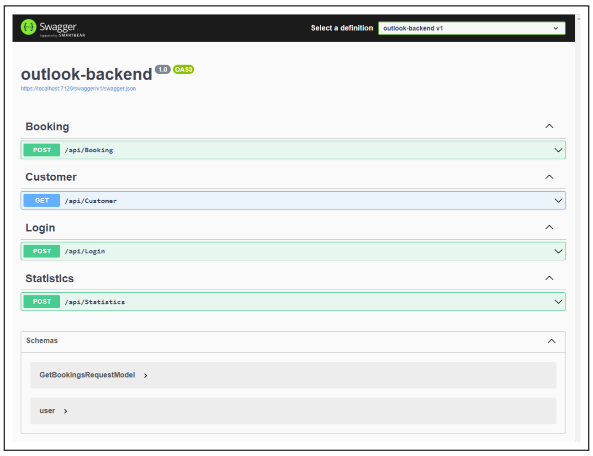

BookingControl... Class ControllerBase Fields _bookingService Methods BookingControl... Get
CustomerContr... Class ControllerBase Fields _customerService Methods CustomerContr... Get
LoginController Class ControllerBase Fields _authentication... Methods LoginController post
StatisticsControl... Class ControllerBase Fields _statisticsService Methods Get StatisticsContro...
Object Class Methods ~Object Equals (+ 1 overload) GetHashCode GetType MemberwiseClone Object ReferenceEquals ToString
GetBookingsRe... Class Properties company_id customer_id order_id
GetStatisticsRes... Class Properties amountPerProd... last_booking
LoginResponse Class Properties token
VisBookWebSer... Interface
IAuthenticationService
IBookingService
ICustomerService
IStatisticsService
AuthenticationS... Class Fields _client _configuration _memoryCache Methods AuthenticateUser Authentication... GenerateJSON... Login
BookingService Class Fields _memoryCache client Methods BookingService FindBookingsIn... GetBookings
CustomerService Class Fields _memoryCache client Methods CustomerService GetCustomers
StatisticsService Class Fields _memoryCache client Methods GetListWithSu... GetStatistics StatisticsService
VisBookWebSer... Interface VisBookWebService IClientChannel
VisBookWebService
VisBookWebSer... Class ClientBase<VisBookW...

Figure 8: Class diagram depicting the connectivity between the different classes within the add-in backend

5 Server Services


Swagger Supported by SMARTBEAR
Select a definition outlook-backend v1

outlook-backend 1.0 OAS3
https://localhost:7120/swagger/v1/swagger.json

Booking
POST /api/Booking

Customer
GET /api/Customer

Login
POST /api/Login

Statistics
POST /api/Statistics

Schemas
GetBookingsRequestModel >
user >

Figure 9: Picture of our swagger endpoints

By the figure 9, we have created and utilized four different endpoints for our add-in. Booking, Customer, Login, and Statistics.

- Booking, to collect all bookings from a guest.
- Customer, to get digial client-card for a specific guest.
- Login, endpoint to login and authenticate into the add-in.

- Statistics, to get various statistics from a guest.

6 Security

Sine the add-in was developed as an MVP, it was not prioritized to implement comprehensive security measures such as SQL-injection or cross site scripting. However, it was desire from Visbook to implement token based authentication with Json Web Token (JWT). This feature has been implemented.

To use our add-in, users are required to be registered in the Visbook system. We do not provide functionality for user registration within the add-in itself. Instead, individuals who wish to utilize our add-in should already have registered credentials in the Visbook System. These credentials will be used for logging in to the add-in.

When the login process is initalised, a request will be send to our backend, the backend application will then (if user credentials are correct) create and store the token. The token is stored in the backend via MemoryCache, the user then gets a token as a response from the backend. This token is further used for any future request towards the server. See figure 2 to see intended user workflow.

7 Installation

7.1 Step-by-Step installation guide for end user

- Receive Manifest file from VisBook

This Manifests file represents the add-in, and all it's connections to Outlook.

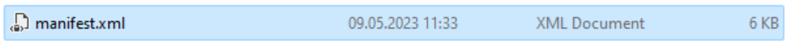

manifest.xml 09.05.2023 11:33 XML Document 6 KB

Figure 10: The manifest file you will receive from VisBook.

- Select an email within your inbox

The selection of the email opens up the email viewer window. You will only be able to open the add-in within this window.

- Upon opening the email view window, select the options "..." dots in the upper right.

The selection of the three "..." dots, will display a dropdown window, see figure 12.

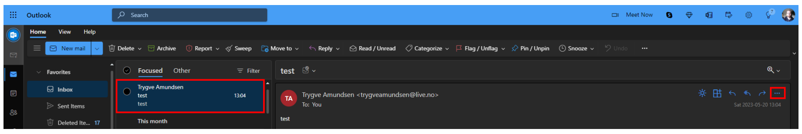


Figure 11: Select a mail from window will initialize the context window where you view the email contents.

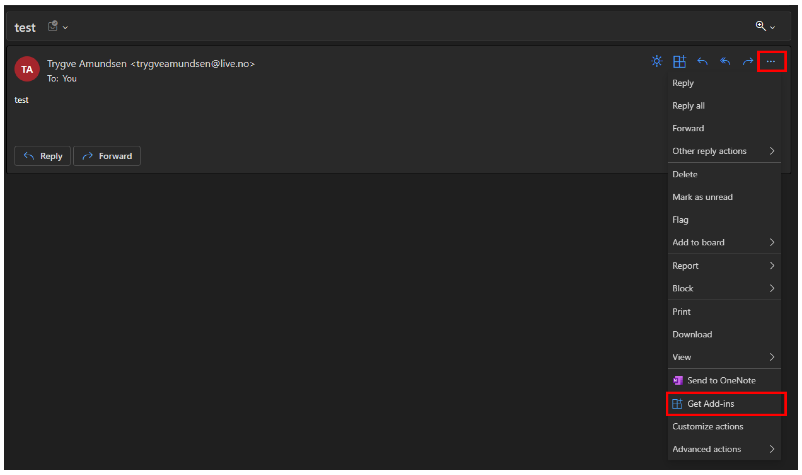
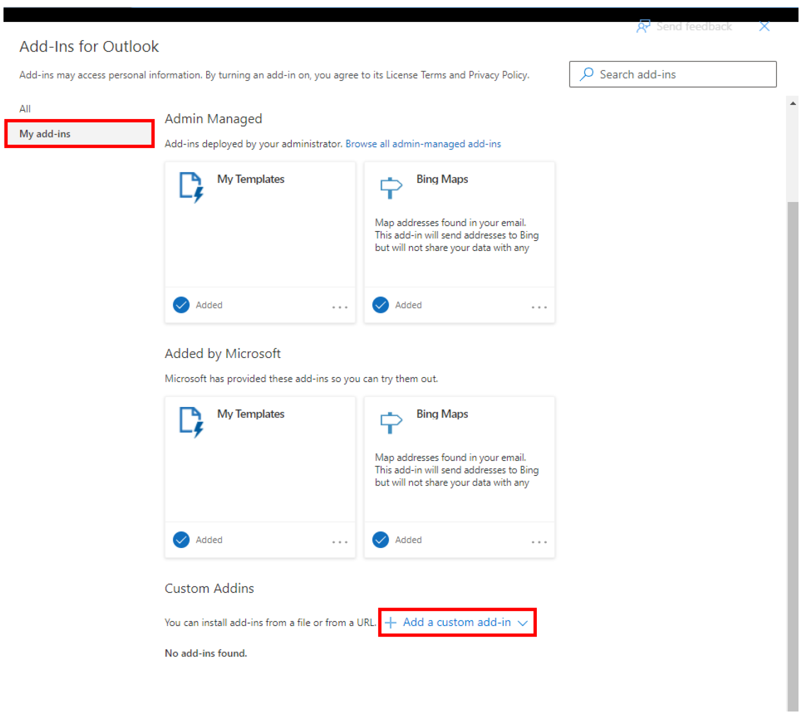


Figure 12: Select the Get add-in option within the drop-down

- Navigate to "Get add-ins"

After locating the add-ins browser, click on "my add-ins", scroll down to you see a "+" icon, and click it.



Add-Ins for Outlook

Add-ins may access personal information. By turning an add-in on, you agree to its License Terms and Privacy Policy.

Admin Managed

Add-ins deployed by your administrator. Browse all admin-managed add-ins

Map addresses found in your email. This add-in will send addresses to Bing but will not share your data with any

Added by Microsoft

Microsoft has provided these add-ins so you can try them out.

Map addresses found in your email. This add-in will send addresses to Bing but will not share your data with any

Custom Addins

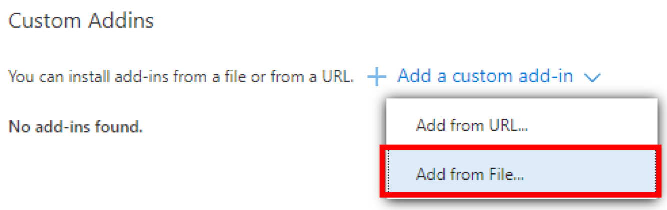
You can install add-ins from a file or from a URL. + Add a custom add-in

No add-ins found.

Figure 13: Window for browsing add-in's, note we're inside the "my add-ins" tab

- **Click** the + icon, then select from file.

Upon selecting from file, the file browser should open. Locate your Manifest.xml file and select it, then press open.

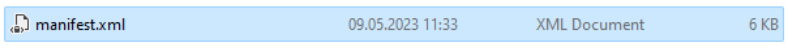


Custom Addins

You can install add-ins from a file or from a URL. + Add a custom add-in

No add-ins found.

Figure 14: Caption

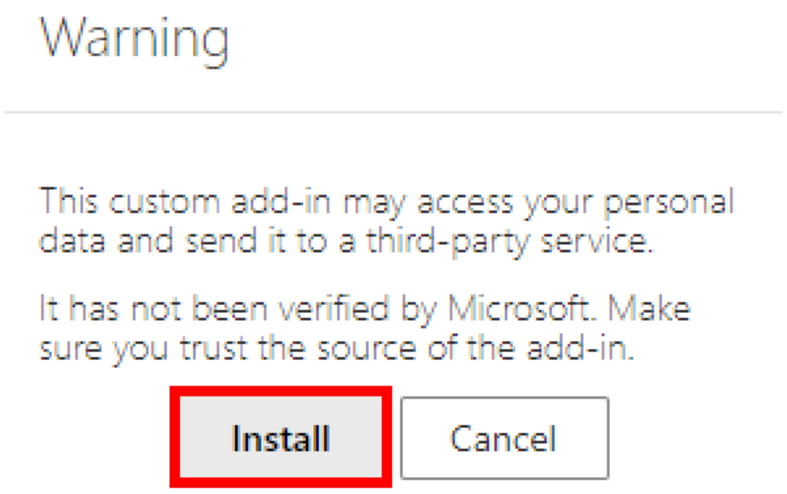


manifest.xml	09.05.2023 11:33	XML Document	6 KB

Figure 15: Select the manifest file where it's located in your directory

- Upon opening the Manifest.xml file, you will receive a pop-up notifying you about trusting files. Click "Install".

Upon selecting that you trust the contents of the Manifest, you should now see the VisBook add-in at the bottom, next to the "+" icon.


Warning

This custom add-in may access your personal data and send it to a third-party service.

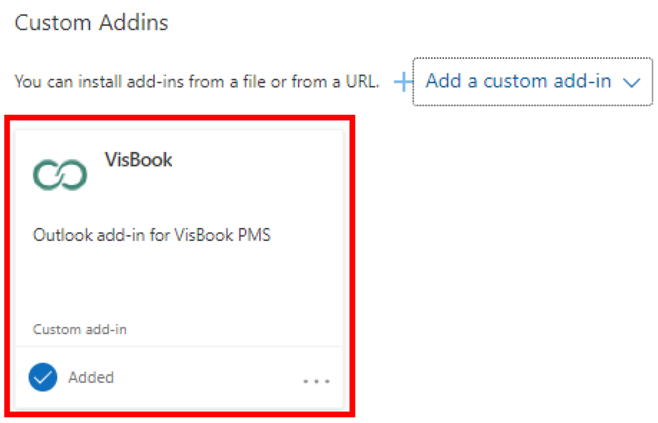
It has not been verified by Microsoft. Make sure you trust the source of the add-in.

Figure 16: Warning window caused by add-in being a custom add-in selected through a Manifest file

To open Add-in after installation After the end-user has installed the add-in. To open the add-in and make use of it, the user needs to click (select) an email, upon the email viewer window will appear. From here, the user will then have to press the three dots "..." in the upper right corner of the email view window, where the a drop-down menu will display.

The add-in, labeled as VisBook, will appear near the bottom of this drop-down.

Upon clicking the VisBook add-in within the menu, you will be presented with the Login page of the add-in.



Custom Addins
You can install add-ins from a file or from a URL.
Add a custom add-in
VisBook
Outlook add-in for VisBook PMS
Custom add-in
Added

Figure 17: The add-in displayed within the add-in browser window. If the "VisBook" add-in is visible, the process was a success.

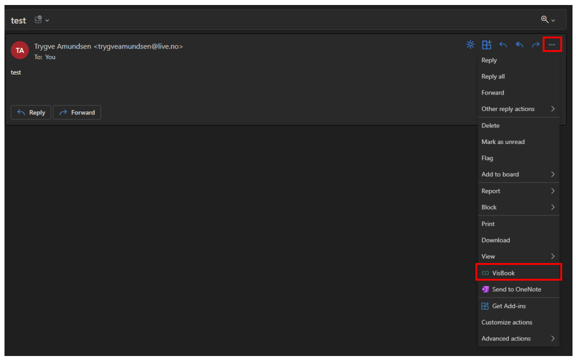


Figure 18: Dropdown from options will now display the add-in "VisBook"

8 Documentation

8.1 Frontend

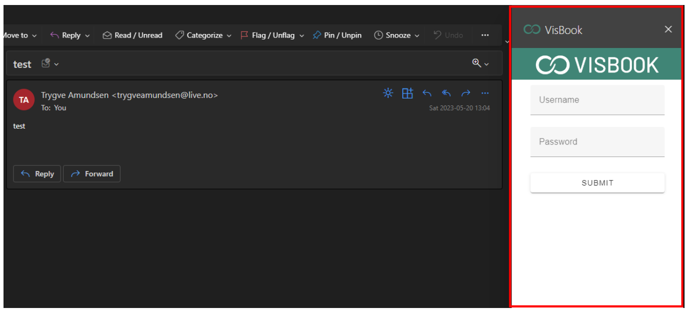
For the frontend application, the team decided to only comment /document vital parts of the code. The team has adhered to best practices and followed consistent naming conventions, making the code easy to read and understand.

8.2 Backend

For the server application, the team used *DocFX* 2023 for the documentation. DocFX enables us to generate and publish the code to a .html file which is viewable in the browser. To generate and view the documentation:

When located inside the outlook-backend project folder, run:

```
docfx docfx_project/docfx.json --serve
```



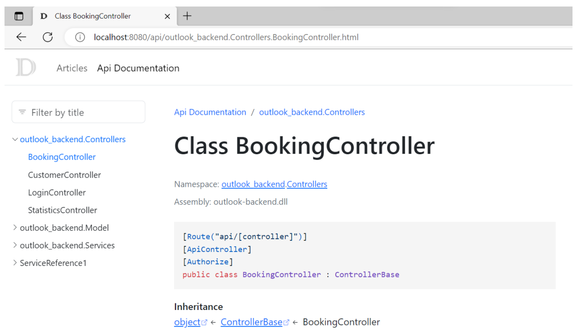
test

TA Trygve Amundsen <trygveamundsen@live.no>
To: You
Sat 2023-05-20 13:04

test

Figure 19: Login page of the VisBook add-in

This command will automatically build and run the documentation, go to http://localhost:8080/ to see result. Reference figure 20 for example.



Class BookingController

localhost:8080/api/outlook_backend.Controllers.BookingController.html

Articles Api Documentation

Filter by title

- outlook_backend.Controllers
  - BookingController
  - CustomerController
  - LoginController
  - StatisticsController
- outlook_backend.Model
- outlook_backend.Services
- ServiceReference1

Api Documentation / outlook_backend.Controllers

Class BookingController

Namespace: outlook_backend.Controllers

Assembly: outlook-backend.dll

```
[Route("api/[controller]")]
[ApiController]
[Authorize]
public class BookingController : ControllerBase
```

Inheritance

object ← ControllerBase ← BookingController

Figure 20: Documentation of BookingController

Disclaimer!

For this project, all related source code is owned by VisBook, and will therefore be disclosed separately to ensure privacy. This, along with other permissions and access to API, results in a source code that will not run outside of the source code owners' environment. As such images from running code will be documented within the main report.

9 CI/CD and Testing

Due to time limitations, the implementation of continuous integration and continuous deployment (CI/CD) was not included in this project.

Unit testing and end-to-end testing were not prioritized in this project, instead the team decided to focus on the desired user stories from Visbook.

Bibliography

DocFX (2023). URL: https://dotnet.github.io/docfx/ (visited on 20th May 2023).

Appendix D Interview Collection

This appendix was not included in the report on the grounds of sensitive information. Can be found in the .zip folder.

Appendix E Process Documentation

This appendix was not included in the report on the grounds of sensitive information. Can be found in the .zip folder.

Appendix F OKR

Can be found in the .zip folder.