NTNU

Det skapende universitet

[Game-based Learning] The Knowledge Challenge Game

Jon Magnus Nielsen
Jonas Olseng

Master i datateknologi
Innlevert: juni 2015
Hovedveileder: Alf Inge Wang, IDI

Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

NTNU – Trondheim
Norwegian University of
Science and Technology

Game-based Learning

The Knowledge Challenge game

Jon Magnus Nielsen

Jonas Olseng

Master of Science in Computer Science

Submission date: June 2015

Supervisor: Alf Inge Wang, IDI

Norwegian University for Science and Technology
Department of Computer and Information Science

Problem Definition

The goal of this thesis is to develop a prototype of a game where you can challenge friends in knowledge battles. The game should produce the experience of playing in real-time, although the game play is asynchronous.

The project consists of a study of state-of-the-art, development of a prototype, and experiments and testing with real users.

The game should be developed in HTML5/JavaScript and appropriate server technology.

Assignment given 16th of January 2015.
Supervisor: Alf Inge Wang, IDI NTNU.

Abstract

This thesis studies what kind of solutions exists for multiplayer mobile games and focuses on a method called time-shifting. We will present a time-shifted quiz application that we developed for Android to test on users, present the results and give suggestions for further work in the field of time-shifting. We will also take a look at how willing the testers are to use time-shifted games in education.

The application was developed using HTML5, JavaScript and Parse for rapid progression. It can be played against other people and/or bots based on other people without the player knowing who is a bot and who is not. The players can also challenge their friends to play the same quiz as they did. We used the results of the tests to see how time-shifted game play impacts the immersion and enjoyment of a game compared to the more standard single and multiplayer games.

The results from the testing shows that people generally are positive to time-shifting and that there is potential to develop it further. The testers also had a hard time distinguishing time-shifted players from real players, which further suggests that time-shifting is something that should be looked into in the future. However, it is important to note that time-shifting is more likely to become a new genre of game than a replacement for the genres that already exists, like single player and real-time multiplayer.

Sammendrag

Denne masteroppgaven ser på hvilke løsninger som finnes for flerspiller mobilspill og fokuserer på en metode kalt "time-shifting". Vi vil presentere en "time-shifted" quiz-applikasjon som vi utviklet til Android for å teste på brukere, presentere resultatene fra testene og foreslå videre arbeid innen feltet "time-shifting". Vi vil også se på hvor villige testerne er til å bruke "time-shifted" spill i utdanning.

Applikasjonen ble utviklet ved hjelp av HTML5, JavaScript og Parse for raskt progresjon. Applikasjonen kan spilles mot andre spillere og/eller mot bots basert på andre spillere uten at spilleren vet hvem som er en bot og hvem som ikke er det. Spillerne kan også utfordre vennene sine til å spille samme quiz som de gjorde. Vi brukte resultatene fra testene for å se hvordan "time-shifted" spilling påvirker innlevelsen og gleden av et spill sammenlignet med de mer vanlige enkelt- og flerspillerspillene.

Resultatene fra testene viser at folk generelt er positive til "time-shifting" og at det er potensiale til å utvikle det videre. Testerne hadde også problemer med å skille "time-shifted" spillere fra ekte spillere, som videre støtter opp om at "time-shifting" er noe som burde utvikles videre i fremtiden. Det er viktig å tenke på at det er mer sannsynlig at "time-shifting" blir en ny sjanger innen spill enn at det skal ta over for sjangre som allerede eksisterer, som enkeltspiller og sanntids flerspiller.

Acknowledgements

We would like to thank Alf Inge Wang for his guidance and help during this project. We would also like to thank everyone that participated in the tests and gave us valuable feedback.

Trondheim, 10th June 2015.

Jon Magnus Nielsen Jonas Olseng

Abbreviations

AI	Artificial Intelligence
APK	Android Application Package
CSS	Cascading Style Sheet
FPS	First-person Shooter
FR	Functional Requirements
G1..2	Goal 1..2
GQM	Goal Question Metric
HTML	HyperText Markup Language
M1..6	Metric 1..6
NFR	Non-functional Requirements
NPC	Non-playing Character
P1..13	Participant 1..13
RQ	Research Question

Contents

List of Figures

List of Tables

Part I

Introduction and Research

1 Introduction

In 2007 the iPhone was introduced to the market, and a year later their massive app store was released, marking a start to a boom in smart phones and multipurpose mobile devices. This created a platform for a new genre of games and made it possible for everyone to be connected to, and play with, their friends no matter where they were. It also changed how we play games. While traditionally you would have to be at your computer to play, now you can take your game with you and play it whenever you have the time. This is often referred to as "dip in, dip out" [1] and creates the need for new solutions in multiplayer games, since the players are not necessarily online at the same time. In this paper we will look at the feasibility of creating asynchronous games that feel like they are played in real time by both players, and how this affects the user experience.

Our work here is based on an earlier project where we successfully created a proof of concept of time-shifting for a quiz game [24]. While the application created in that project illustrated how time-shifting could be achieved, it was never designed for or tested on users. Our goal with this project is to create a new version of the application for Android that we can test on users to see if time-shifting effects the user's immersion and enjoyment of games. We will also test the user's ability to distinguish between real users and AI based on users that played earlier.

1.1 Time-shifting

The term time shifting is today mostly used in reference to storing a television program so that it can be viewed at a more convenient time for the consumer. It was introduced to computer gaming with the 2013 release of the game Real Racing 3 by FireMonkeys Studios where the developers referred to their multiplayer mode as "time-shifted multiplayer". This game mode was designed to give players the opportunity to compete against each other in a real time game, even if they were not playing at the same time.

When we use the term "time-shifting" in this report, what we are referring to is this behavior where you are playing against another person in what appears to be real time, but in reality you are playing the game at different times.

1.2 Background and Motivation

The success of smart phones have made games more popular than ever, with titles like Wordfeud having more than 100.000 daily users on Facebook [2] and Clash of Clans having several million daily users [3]. If we look at the multiplayer games, they are mostly built on synchronous or asynchronous game play. This requires users to either be online at the same time, or wait for their opponent to respond before they can proceed with their turn. We have earlier created a proof of concept for time-shifting of quiz games. Knowing the technology is possible, our next step is to get user feedback to see if it adds something to the user experience. We are particularly interested in its applications as a learning tool. Games like Kahoot! and Buzz the School Quiz have already shown promising results in increasing immersion from the students in the classroom, but they are played synchronously and require all the participants to be ready at the same time. Our goal is to create a supplement to these games by having a asynchronous alternative while containing the benefits of increased immersion that real-time games give. The same way Kahoot! and Buzz are tailored for classrooms, we see our time-shifted quiz as a good choice for homework and as a tool for studying for tests.

While creating and testing this application, we hope to better understand the limitations and possibilities time-shifting brings to the educational and gaming world.

While creating our proof of concept, we discovered that emulating the communication between players in real-time games was difficult in time-shifted games because they are not playing at the same time, and while perfectly replacing the real-time communication seems impossible, we are interested in exploring methods to replicate parts of that experience.

1.3 Research Approach

In literature, four different research methods was presented by V.R Basili in 1993 [27]. These four methods are the Engineering Method, the Scientific Method, the Mathematical Method and the Empirical Method. In section 1.3.1 we will explain all of them with extra emphasis on the ones we use and why we decided to use these methods.

The measurement of our results are based on the Goal, Question, Metric (GQM) approach which we present in section 1.3.2

1.3.1 Research methods

In this chapter we will look at the research methods presented by V.R Basili [27] and how it relates to our work.

Scientific Method

The scientific Method starts with the researchers studying the environment the technology is supposed to exists in. That is, the people, processes, products and everything else the researched technology will influence and be influenced by.

The next step is to come up with a hypothesis about how the technology will work, and then test this hypothesis in its environment. This is usually done several times as you are likely to discover new problems and possibilities which is used to create an improved hypothesis.

The last step is to validate the hypothesis by testing if it fits with the observed world. If it fails the validation, the researchers will have to improve upon the hypothesis until it fits with reality.

Both the Engineering Method and Empirical Method are variations of the Scientific Method.

Engineering Method

The Engineering Method is an evolutionary method where you improve upon existing designs until no further improvements are possible.

The first step is to study and analyze existing models of the environment, product, processes and people involved in the old solutions. Then the researchers improve upon these solutions by developing new and improved solutions and analyzing them. Once you have a new solution, you can use the analysis of it to repeat this step and create an even better solution, and this continues until there is no more room for improvement.

Since this method uses existing methods as a start, this obviously requires some existing methods to exist. If this is not the case the Empirical Method must be used instead.

Empirical Method

In the Empirical Method, the researchers propose a new method that may or may not be based on an old solution. The next step is to measure and analyze how this new method performs. When no old solutions exist to compare to, it is especially important to have good measurement criteria. As Basili states in "The experimental Paradigm of Software Engineering" [27]: "Experimentation alone offers no value if there is no underlying framework where experimental results can be interpreted"

The results from the analysis is then validated and used to improve the hypothesis if the results are not satisfactory. Just like in the other two methods, the steps are repeated with the new hypothesis until it fits the goals.

Mathematical Method

In the Mathematical Method you create a theory and then derive results directly from it without necessarily creating the system. The results are then compared to empirical observations if they are available. Since this method is not used or relevant for our domain, we will not discuss it in any more detail.

Research Methods in this project

In our project, we will use both variations of the scientific method. Our project can be divided into two parts, a quiz application with focus on educational use and time-shifted behavior. Several quiz apps already exists, and many of them, like Buzz The School Quiz and Kahoot!, are used for educational purposes. This means that we can use the engineering method to improve on these.

The time shifting part on the other hand has not been studied to the same extent, and we have few earlier models to look to. The biggest commercial attempt on time shifting is Real Racing 3 which is a racing game, and thus very different from our domain. We have however in a project leading up to this paper [24] created a proof of concept for time-shifted quiz applications that we can use as a model for creating our new and improved design. We will mostly have to rely on the Empirical Method when analyzing the successfulness of the time-shifting part of our project since there is not much earlier work done in the field.

1.3.2 Research measurement

Our method of measurement is based on the GQM[46] approach. Using this approach you first define your goals, then questions regarding the quality of the goal and lastly the metrics for measuring these qualities.

Conceptual Level (Goal)

The goal of our research is given by the problem definition for our thesis which is presented below:

"The goal of this game is to develop a prototype of a game where you can challenge friends in knowledge battles. The game should produce the experience of playing in real-time, although the game play is asynchronous.

The project consist of a study of state-of-the-art, development of a prototype, and experiments and testing with real users.

The game should be developed in HTML5/JavaScript and appropriate server technology."

This was decomposed into two goals for our thesis:

G1: Improve the enjoyment and immersion of the players in quiz games using time-shifted AIs.

G2: Create time-shifted AIs that are indistinguishable from real-time players from the perspective of the other players.

Operational Level (Questions)

The questions defined by decomposing the goal are the research questions presented in Section 1.4 where RQ1 are the questions derived by the decomposition of G1 and RQ2 the same for G2. In addition to these research questions the functionality specified by the Goal were decomposed into several functional requirements presented in Section 6.1.1. The usability and enjoyment of the prototype were further analyzed by using the System Usability Scale which we present in Section 2.5 and GameFlow framework presented in Section 2.4.2.

Quantitative Level (Metrics)

In GQM, metrics are divided into subjective and objective metrics were subjective metrics depend on both the object being measured and the viewpoint of the person measuring and objective measures only on the object. We use a combination of objective and subjective metrics, two different viewpoints with the data source used being the questionnaire given to the participants after finishing the user test. The objective metrics are the precision the participants have when trying to distinguish between real-time and time-shifted players in RQ1. The subjective measures are the rest of the questions in the questionnaire as they all relate to how the participants feel about time-shifting and the application.

The two viewpoints used are the participants viewpoint when answering the questions in the questionnaire and our viewpoint when analyzing these results to answer the research questions.

1.4 Goals and Research Questions

Our overall goal is defined in the problem definition and we decomposed this down to two different goals we wanted answered for our research.

G1: Improve the enjoyment and immersion of the players in quiz games using time-shifted AIs.
The research questions derived from this goal are:

RQ1 *How does time-shifting effect the user experience?*
We already know that that time-shifting can be done for quiz games, but we still don't know if it adds to the immersion and enjoyment of the game or not. In this research question we will try to uncover if, and to what extent, time-shifting effects the user experience.

M1: The metric is our analysis of the participants answers regarding how time-shifting affects their user experience.

RQ1.1 *How does time-shifting affect users immersion and enjoyment compared to single player games?*
We want to discover how the immersion and enjoyment of games that are time-shifted are compared to single player games. We will try to answer this questions by analyzing results from the user tests and come up with a conclusion.

M2: The metric used is our analysis of the participants answers regarding their enjoyment when comparing time-shifted and single-player games.

RQ1.2 *How does time-shifting affect users immersion and enjoyment compared to real time games?*
Multiplayer games is one of the most, if not the most, popular types of games. Challenging this genre of games and comparing it to time-shifted games is something we really want to take a closer look at. We will conduct user tests and analyze the results to come up with an answer to this question.

M3: The metric used to answer these questions is our analysis of the results from the participants in the questionnaire.

G2: Create time-shifted AIs that are indistinguishable from real-time players from the perspective of the other players.
The research question derived from this goal are:

RQ2 *How difficult is it to differentiate between real players and time-shifted players?*
Ideally we want our time-shifted players to be indistinguishable from a real time player, but we don't know how realistic that is. In this research question we want to discover how difficult it is for users to differentiate between them.

M4: The first metric used to answer this question is the precision of the participants when trying to differentiate between real-time and time-shifted players

M5: The second metric is our analysis of the answers given about the difficulty of differentiating.

RQ2.1 *How did players differentiate between time-shifted and real time players?*
If the users are capable of distinguishing between them, understanding how they did it is an important step in improving time-shifting technology.

M6: The metric is our analysis of the answers given by the participants about how they differentiated.

RQ2.2 *What methods are available for solving the communication problem in time-shifting?*
In our earlier work with time-shifting one of the limitations we found was that communication between players is very difficult since the time-shifted one is already finished. In this research question we will see if we can find methods to solve or reduce this problem.

M7: The metric used is our analysis using the experiences we get while working with this problem in this thesis.

1.5 Readers Guide

This report covers everything done in our project, and every part might not be equally relevant every reader. This chapter aims to give our readers a better understanding of what the different parts of the report covers so they are better suited to find the pieces they find interesting.

The thesis is divided into five parts, and a short summary of what you can expect to find in them follows.

Part I - Introduction

The first part of the paper is an introduction to the project and should be relevant to everyone since it explains what the rest of the report will be about.

Chapter 1 - Introduction This chapter explains the term time-shifting, our research goals and motivation for this project and the research methods we intend to use to reach these goals.

Part II - Pre study

The second part is the pre study which is divided into four chapters. This part will be of interest to everyone that wants to learn about the work that has been done in time-shifting and educational games earlier.

Chapter 2 - Background theory Explains terms and theories that we have used in our project.

Chapter 3 - Earlier work This chapter covers the state of the art applications that exists with similar functionality to what we aim to create.

Chapter 4 - Technologies This chapter covers all the technologies we have used to develop our application.

Chapter 5 - Time-shifted games This chapter explains methods of creating time-shifted behavior in games and the opportunities and limitations of time-shifting.

Part III - Own Contribution

The third part covers the work we have done in this project. This Chapter should be of interest to anyone who wants to understand our application and the how`s and whys about its development.

Chapter 6 - Architecture This chapter presents our architecture and explains our reasoning for doing it the way we did.

Chapter 7 - Application This chapter presents the application as it is seen from a users perspective, and our reasoning for design choices. We will also explain how we planned the tests of the application.

Chapter 8 - Testing the Application The last chapter of this part presents the planning and execution of the tests we conducted.

Part IV - Evaluation

This part of the report is presenting our results and the evaluation of them.

Chapter 9 - Evaluation of the Results This chapter presents what we have discovered by evaluating the questionnaire, the System Usability Scale and GameFlow analysis.

Part V - Conclusion

This chapter presents our conclusions and should be of interest to anyone wanting to know what we learned during our project and/or who wants an insight into possibilities that lays in the future of time-shifting.

Chapter 10 - Conclusion This chapter sums up our key findings and the conclusions that can be drawn from them. We also suggest ideas to further development of applications like the one we developed.

Chapter 11 - Further Development In this chapter we suggest how our application can be developed further and how similar applications can be improved.

Chapter 12 - Future of Time-shifting The last chapter in this report presents our thoughts on the future of time-shifting.

Part II

Prestudy

2 Background Theory

In this chapter we will explain different terms that are relevant for our project and give examples of relevant games that already exists. The terms we will look at is gamification, serious games, artificial intelligence in games, game flow and system usability scale, SUS and an introduction to our findings while creating the proof of concept for time-shifted games.

2.1 Gamification

Gamification is an umbrella term for the process of using game mechanics outside of traditional games with the goal of improving user engagement and motivation [26]. This is achieved by trying to tap into our inbuilt desire for mastery and competition. One frequently used method is the use of achievements for reaching certain levels which serves as a "carrot" for the user. Gamification became popular in 2009 because of new applications like FourSquare [21] which used "badges" to stimulate the users to do different actions. Today gamification is used on many popular web pages like Yahoo! answers and the programming discussion site "Stack overflow" where all the content is made from questions and answers posted by the users.

2.1.1 Pros and Cons of Gamification

There are both pros and cons of gamification. The obvious pros is that it gives users incentives to make an effort to answer questions on web pages like "Stack overflow" where the users gets points from other users depending on how good answers they give. By answering questions and getting feedback from the other users the user builds a reputation and gets rewarded with bronze, silver and gold medals dependant on how many views and positive feedback she gets. This systems is self-balancing and hardly requires any moderators to intervene. This system works really good in theory, but there is always possible for users to "buy" a good reputation by using bots or other methods.

There might also be a problem related to balancing the ratio between the effort of answering a question and the reward given. In a blog by Michael Richter [42], a previously active member of Stack overflow, he argues that there is a big problem with the ratio between effort and reward in such a system as "stack overflow". He comes up with examples where he answers easy questions with little effort and gets a lot of positive reputation from it, and complex questions without hardly getting any reputation. This seems like a big problem, and probably is, but in web pages like Stack Overflow the complexity of an answer might not be the most helpful for the community. It is obvious that an easy question asked by a beginner will be seen by more users than a complex answer asked by a coder with 20 years of experience. There are far more beginners that have questions than experienced coders. This leads to the easy questions and answers getting more publicity than the complex ones. And since giving someone a positive reputation only gives them a fixed amount of reputation, independent of the complexity of the question, an easy question and answer will give you a higher reputation to effort ratio.

To handle this ratio problem you could have some moderators which have a good understanding of the area add a complexity multiplier to the reputation given to an answer of a complex question. However, this would require moderators to constantly look for these complex questions and answers which counteracts the purpose of a self-balancing reputation system.

There are also sites that uses gamification to make everyday chores easier. A great example is HabitRPG [37] where the users lists everything they need to do and the websites creates monster for each task that the player must defeat. By completing a chore you defeat the monster in game and your game character progresses and gets new items. If you are slacking and not doing your chores your character will become less powerful. This approach to gamification works pretty good since people like progression and a game is a nice way of showing it. It does not say what ages the game is targeting, but it might be a bit hard to make a grown up use this game. For kids however, it probably works like a charm.

However, the problem with this site is that it is so easy to cheat. The website itself cannot check if you have completed your chores or not and you could in theory list a lot of chores and complete them on the site without really doing anything in real life. That way you progress in the game but not in real life which really works against the purpose. In this case it is important to have a parent overseeing the chores before their child can check it off on the site. This is also the main assumption to saying this game only work on children and not older, lazy, adults.

2.1.2 Relation to this Thesis

The idea for our application is to take a rather boring task, e.g. answering questions from a textbook by writing them down on paper, and make it more fun by implementing elements that gives the player enjoyment. The elements we use are sound, color, dynamic multiplayer and competition. These elements should spark an interest in the players and hopefully make it more interesting to answer questions from a curriculum. Our application can of course be used for answering any kinds of questions, but our goal is to use it in a school environment to answer questions that improves learning. The idea behind the use of AI, which is explained in Section 2.3, is that we want the players to be able to learn wherever and whenever, not only in class at school or at home doing homework but also on the bus, when going to soccer practice and so on.

2.2 Serious Games

Serious game consists of all the basic elements of normal game which are story, art and software. In addition, it also implements pedagogy [5]. The purpose of a serious game is not only to entertain, but also make the players learn something from it. Today, serious games are used in several different fields to improve the player's ability to complete different tasks or learn. In his paper Serious Games: Serious Opportunities [25] A. J. Stapleton presents, among others, the following types of serious games: K-12 Edutainment, higher education, healthcare and military. We will take a closer look at the focus of these, different examples of these types of serious games and their positive and negative sides.

2.2.1 K-12 Education

Educational entertainment, or edutainment, that focuses more on the entertainment than the educational factor. These types of games often targets a young audience where having fun is more important than learning something. These games are often repetitive to make the players learn through repetition and reward them with progression in the game instead of focusing on what they have learned, as stated in[25]. A game that tries to do this is Talljakten [16] shown in Figure 2. 1: The game Josefine Skolehjelp: Talljakten. where the players learn how to count, add and subtract while following a story to keep the children immersed in the game.

Figure 2. 1: The game Josefine Skolehjelp: Talljakten.

2.2.2 Higher Education

According to A. J. Stapleton [25] the higher education sector of serious games is where the greatest potential for serious games lies. Since these types of serious games are for older and more mature people it is easier to cooperate with them and get good feedback on the games to develop and improve them further. One example of such a game is Supercharged! shown in Figure 2. 2: Screenshot from the game Supercharged! which is a game made to teach the physics of electromagnetism and how different charges affects each other according to the distance between them. The purpose of the game is to navigate a space ship to a goal through a 3D space by using what they know about electromagnetism [38].

Figure 2. 2: Screenshot from the game Supercharged!

2.2.3 Health Care

In health care there is a broad variety of usages for serious game. It can be used to motivate and reward patients undergoing treatment and distract patients during procedures [25]. It can also be used to train health care personnel to handle different situations that may arise during a crisis to better prepare them if something were to happen. One example of this is the game vHealthCare which is a simulator of a virtual 3D hospital that focuses on helping health care personnel practice clinical skills in a safe environment to prepare them for catastrophic incidents [39]. A screenshot from the game can be seen in Figure 2. 3: Screenshot from the game vHealthCare..

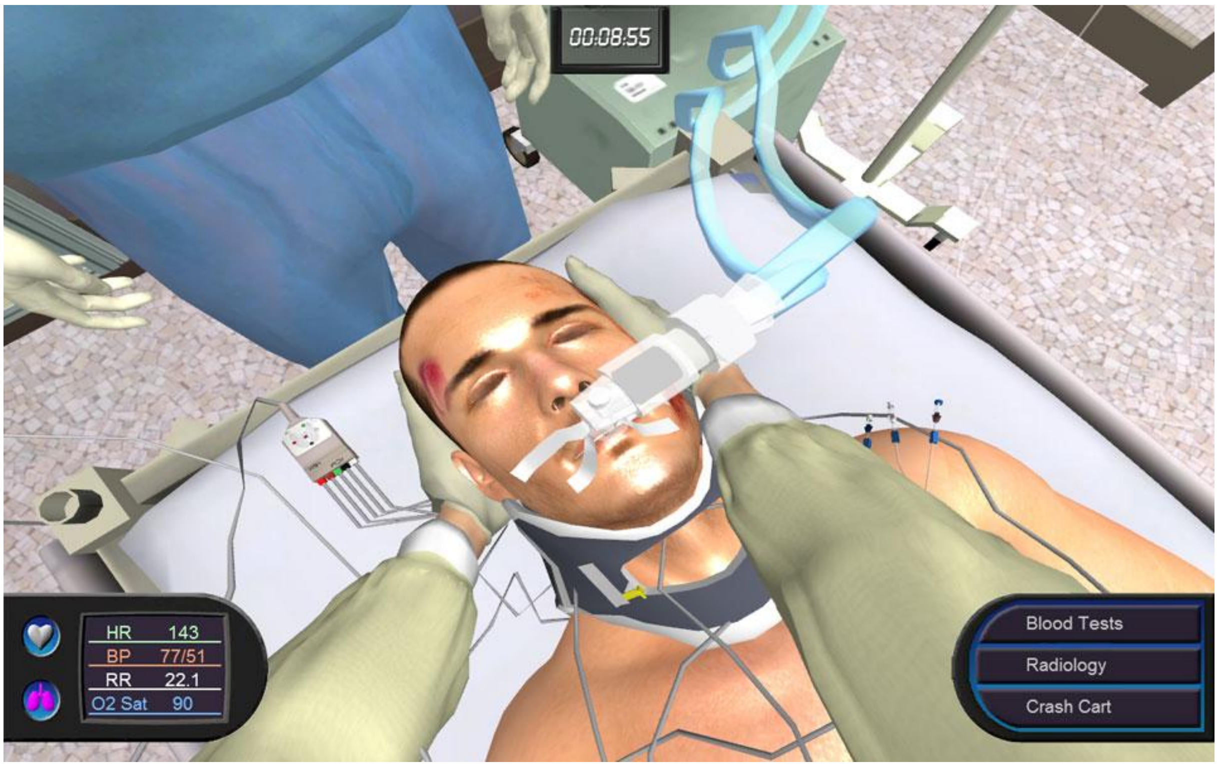


Figure 2. 3: Screenshot from the game vHealthCare.

2.2.4 Military

Military training might not be the first thing you think of when talking about games, but the military does in fact play a big role when it comes to serious games. According to M. Zyda [5] the game America's Army shown in Figure 2. 4: America's Army game play. , which is a FPS, was in 2002 used for training recruits at Fort Benning that had problems at the rifle range or at the obstacle course. These recruits sat down and played specific levels of the game to practice what was needed to improve in these areas. After practicing in the game they often passed the tests they failed before playing.

Figure 2. 4: America's Army game play.

There are also other projects that takes gaming and military to the next level. The Gadget Show is a British television show about new technology and consumer gadgets. This show does not always just review small gadgets, but they also do bigger project, like the one they did in 2011. In this project they wanted to create a first-person-shooter, FPS, simulator that was as close to real life as possible. What they created was a dome where the player could move freely in 360 degrees on top of treadmills that would move the character in the game according to how the player moved in real life. They player also held a gun that was used to shoot in the game. By the help of motion sensors the player screen would also move according to how the player moved his head to look left and right. In addition to all the movement they also added paintball guns around the dome that would fire on the player when he was hit in the game to add even more realism [40]. A display of the setup is shown in Figure 2. 5: First person shooter simulator..

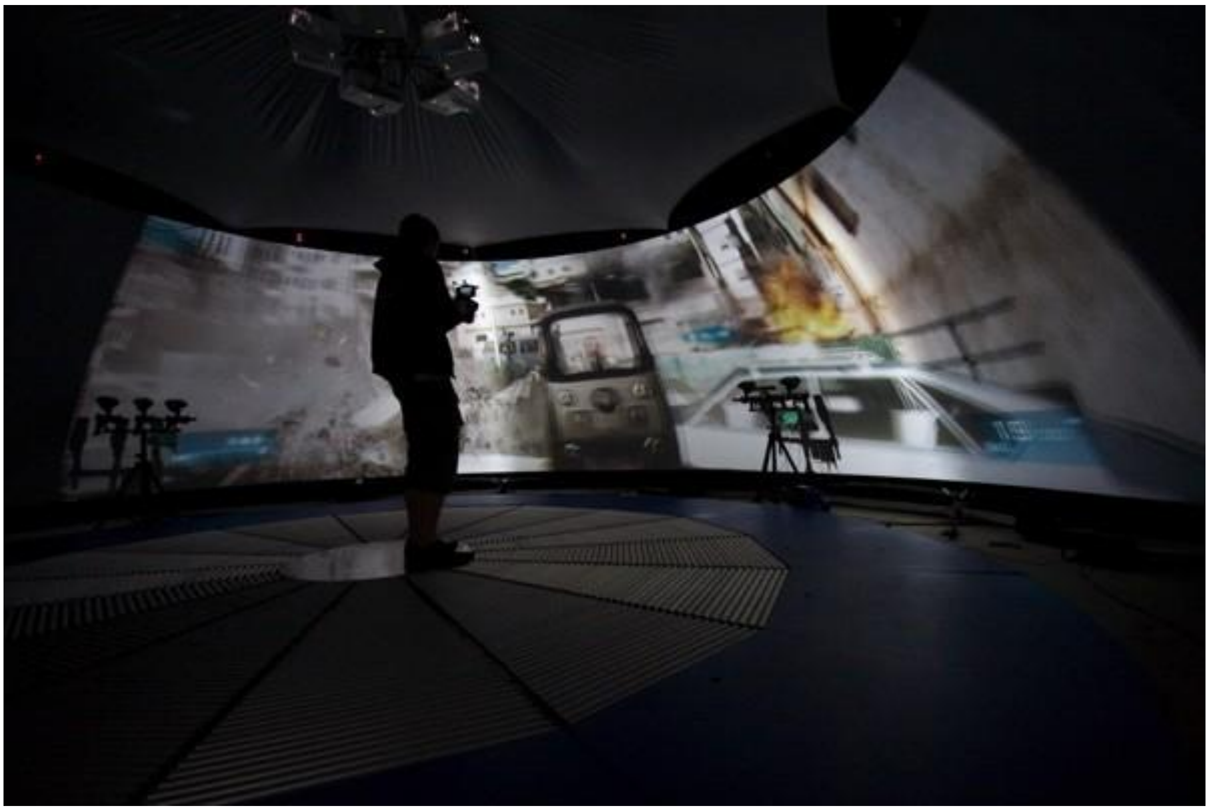


Figure 2. 5: First person shooter simulator.

To really test the simulator they invited a SAS commander to try it out. After testing the simulator he said that he got the same feeling playing the simulator as he did in real life. He also said that being shot by the paintball guns in the simulator was really important so that people would not think they were Superman after playing the simulator.

2.2.5 Pros and Cons of Serious Games

There are a lot of different kinds of serious games and many degrees of seriousness in the games. It is impossible to compare the value of games like Talljakten and the FPS simulator even though they both are serious games. However, they are very valuable in their own area. Making children learn mathematics which requires a lot of repetition through games is brilliant in its own way. If the game is well made and makes children learn while having fun it is truly invaluable. Motivating children to learn while they are having a good time will most likely give them a good association to learning and make it easier for them to learn more complex things later in life. The problem with these kinds of games is that they might not be that good at teaching children mathematics if the focus is shifted too much over on entertainment. It could also go the other way where the game focuses too much on teaching and too little on entertaining the children and make them bored really fast.

When it comes to the FPS simulator made by The Gadget Show, which used the game Battlefield 3 for testing, the learning value is probably not optimal. Battlefield 3 is purely made for entertainment and is not guaranteed to give the players anything more than the feeling of being in combat. If you however made games specifically for the simulator with the purpose of teaching recruits how to handle different situations and learn from that experience, then you would have an amazing resource for learning. The recruits could be tested in different scenarios that are as close to reality as possible and prepare them for those scenarios in real life. The simulator could also be improved by introducing 3D with Oculus Rift or other similar technologies.

The problem of using this simulator is that it is not real. It is impossible to predict what happens in real life and put it into a game to prepare the recruits for everything. There is no risk involved while playing a simulator and the player can get used to the pain from the paintball guns. That way it is impossible to really know if they would act differently in real life or not.

2.2.6 Relation to this Thesis

As we have seen in the previous chapters there are different degrees of how much focus is put on pedagogy and entertainment in serious games. For the younger players it is probably more important that a game is fun while for the older audience there is more focus on learning. With our application we do not focus on children but rather players that are 15 and older. Therefore we have not spent a lot of time making the game very entertaining, but rather focusing on making people learn by playing, while hopefully having fun. We do, however, focus on the immersion in the game by a clever implementation of multiplayer by using artificial intelligence based on how people play and behave. This will be covered in the next section.

2.3 Artificial Intelligence in Games

Artificial Intelligence in games has been around since the first commercial attempts of computer games with the release of arcade machines. In Section 2.3.1 we will present a brief summary of the history of AI in computer games. Section 2.3.2 will look at some of the methods used to create artificial Intelligence in games, Section 2.3.3 will explain and give some insight into the difference between the AIs in games and the academic field of AI. In Section 2.3.4 we will look at how, and for what purpose, AI is used in computer games before we end the chapter with the usage and relevance of computer AI in our project.

2.3.1 History of Artificial Intelligence in Games

The first steps into AI for computer games came with the arcade machines in the 1970s. Early attempts of AI in these games consisted of hard-coded patterns that the computer would control. Examples of this would be enemy spaceships moving towards the player in a predefined pattern.

In 1980 the game Pac-Man as released with an innovative approach to AI. In Pac-Man, the player is chased by 4 different ghost that all have different algorithms giving the impression of some intelligence from the computers part [8].

As games continue to grow in sophistication, so does the AI to keep up with the added complexity of the games. This requires new and improved methods to reach the requirements which we will discuss in the next chapter.

2.3.2 Methods of Artificial Intelligence in Games

This section will give a short overview of some of the methods used to create Artificial Intelligence in computer games. Note that the names are not official, and should be read as descriptions of the methods.

Stored patterns

This is the simplest form of AI found in computer games. With stored patterns the AI will simply move or act based on hard-coded commands without regards to the player's actions or position. Old space shooter games where enemy spaceships move towards the player use this method. In these games the enemies do not react to the player, they just move towards the bottom of the screen following a predefined pattern like following a straight line, moving from side to side etc.

Reactive patterns

This is a variation of the stored pattern where the computer reacts to the player, but still in a hard-coded way. One of the first and best examples of this is in Pac-Man where the ghosts will react to the player's position [44]. Other examples are enemies that will chase the players if they come to close as seen in many role-playing and fighting games.

Player evaluation

Player evaluation is a method where games evaluate the actions of players to create AIs that better mimic them. The goal here is not to create the best possible AI in terms of results but to make them more realistic by acting like a human player would. This method is explained in deeper detail in Section 5.1.1 as it is very relevant for creating time-shifted behavior in games.

2.3.3 Differences Between Game AI and Academic AI

AI in games and the academic field of AI are different in their goals, and here we aim to clarify some of these important differences.

In one of the leading textbooks on artificial Intelligence, it is defined as "the study and design of intelligent agents, in which an intelligent agent is a system that perceives its environment and takes actions that maximize its chances of success"

[4]. This means that AI in academia is concerned with solving problems in the most efficient way.

This is very different from AI in games. Here the AI is there to improve the user experience of the player and that usually means solving problems in a sub optimal way so that the player has a good chance to win. A good example of this would be all the existing computer chess games. Today, chess AIs can beat even the world's best chess players and it would be impossible for an amateur player to win a single game against an AI that played to "maximize its chances of success" like the academic definition states. Always losing would obviously hurt the user experience, and because of this, game AIs tries to create an environment where the challenge is possible to overcome without being trivial.

Another difference between academic and game AI is how they solve their problems. Game AIs will often "cheat" [45] to create a better user experience. To exemplify this, let`s consider a AI controlled player in a first person shooter. To be a "real" AI it would have to be able to notice the player through sight or sound in the game. This is often very difficult to do and instead the game AI will "cheat" by getting the position of the player from the game engine and reacting based on this instead of the inputs like sound and vision that a player has to rely on. From a player's perspective, this and other methods of game AI "cheating" is often indistinguishable from the actions of a real player, and allows the AI to serve its purpose without advanced calculations that an academic AI approach to the problem would require. There is however some issues with it. If the player notices that the AI is using these methods it can hurt the user experience as it can be seen as unfair play.

2.3.4 Usage of Artificial Intelligence in Games

Artificial Intelligence is used for several purposes in computer games and in this section we will discuss some of the most important ones.

NPC

A NPC (non-playing character) in computer games is, as the name implies, a character in the game world that is not controlled by a player but by the game itself. It is widely used in role-playing games where they can be merchants or quest givers that have little intelligence. That is, they generally stay in the same place, and interaction with them is just predefined dialog options. But NPCs can also be more advanced where they are capable to interact with the player and environment in more sophisticated ways. A city guard that chases and possibly attacks a misbehaving player, or a companion that follows the player and has to react to approaching enemies, movement challenges like stairs, different terrain etc. are examples of more advanced NPCs.

AI controlled players

Computer AI is in multiplayer games often used as an opponent if no human challenger is available. In a game like chess you always need to have two participants and here the computer can serve as the second player.

Path Finding

In many games, specifically role-playing and real-time strategy games, players tell their character(s) where they want them to go by issuing a simple command like clicking on the map. Then the AI has to figure out how to move the character from its current position to the one chosen by the player. Doing this in a realistic way has been a big problem in the field of game AI especially when several characters are moving at the same time and they can collide with each other. Methods for dealing with this is constantly being improved and is one of the fields where game AI increasingly rely on methods from academic AI to create good solutions.

Dynamic Difficulty

Since the goal of games is to keep the player entertained it is important that it never becomes too difficult or too easy. Traditionally game designers have solved this by allowing the player to pick between different difficulty settings, but it is becoming increasingly popular to let the game adjust itself to create an ideal challenge level for the player. In Nintendo's popular Mario Kart series, the game balances the difficulty by giving players falling behind better items than the ones that are in the lead so they can catch up.

Another method proposed by Hunicke and Chapman [37] is to use game metrics like remaining health points, shooting accuracy etc, to try and detect when players are about to fail at a task. Examples of failing in this context would be dying or taking too long to solve a puzzle. If the game detects that the player most likely is going to fail, it will try and intervene and decrease the difficulty level. This can be achieved by reducing the accuracy or speed of computer controlled opponents for example.

2.3.5 Usage of Artificial Intelligence in this Thesis

The time-shifted players in our project can be seen as AIs that try to mimic the behavior of the player they are based on. What we are trying to achieve is a variation of an AI created by the player evaluation method discussed in Section 2.3.2 that mimics the player to the extent that it "feels" like you are actually playing against the person and not an AI.

With this, we can create AI controlled players that hopefully gives the same positive effects as playing a real-time multiplayer game.

2.4 Game Flow

Measuring how enjoyable a computer game is has proven to be very difficult as different players often give significantly different answers when asked how much they enjoyed a game or to compare games to each other. Because of this, judging how good a game is has usually been done by popular opinion, expert reviews and web pages like metacritic.com [22] where user and expert reviews are aggregated and used by many fans of computer games to decide if a game is worth playing.

The problem with this approach is that reviews are based on subjective analysis and no framework has existed to give structure to the reviews of users or experts.

Researchers P. Sweetser and P. Wyeth wanted to change this, and in 2005 they presented their attempt of a framework [10] for analyzing the enjoyment in games. They created a framework called GameFlow based on the concept of Flow in positive psychology.

2.4.1 Flow

In positive psychology. Flow is a reference to a state of mind where the user is fully immersed and have a feeling of enjoyment for the task they are doing [9]. It was coined by Prof. M Csíkszentmihályi and through his research where he collected data from interviews, questionnaires and other sources, showed that people regardless of culture, gender, age, social status etc. had similar experiences when asked what they felt when their lives were at their fullest and most enjoyable. He used this to create a framework for understanding enjoyment in humans.

Csíkszentmihályi found that flow experiences consisted of these eight elements:

1. A task that can be completed.
2. The ability to concentrate on a task.
3. That concentration is possible because the task has clear goals.
4. That concentration is possible because the task provides immediate feedback.
5. The ability to exercise a sense of control over actions.
6. A deep but effortless involvement that removes awareness of the frustrations of everyday life.
7. Concern for self disappears, but sense of self emerges stronger afterwards.

8. The sense of the duration of time is altered.

When all of these elements are present, they give a sense of deep enjoyment. In Csíkszentmihályi's own words Flow is an experience "so gratifying that people are willing to do it for its own sake, with little concern for what they will get out of it, even when it is difficult or dangerous" [10].

2.4.2 The GameFlow Framework

From the eight elements presented in Flow, the researchers defined eight elements for GameFlow with several criteria for each of them to properly analyze how well the game fulfills the elements.

A list of the elements and their criteria are listed in Table 2.1.

Element	**Criteria**
Concentration Games should require concentration and the player should be able to concentrate on the game.	Games should provide a lot of stimuli from different sources. Games must provide stimuli that are worth attending to. Games should quickly grab the player`s attention and maintain their focus throughout the game. Games should have a high workload, while still being appropriate for the players perceptual, cognitive, and memory limits. Players should not be distracted from the tasks they need to concentrate on.

Challenge Games should be sufficiently challenging and match the player's skill level.	Challenges in games must match the players skill levels. Games should provide different levels of challenge for different players. The level of challenge should increase as the player progresses through the game, and increases their skill level. Games should provide new challenges at an appropriate pace.
Player Skills Games must support player skill development and mastery.	Players should be able to start playing the game without reading the manual. Learning the game should include online help so players don't need to exit the game. Players should be taught to play the game through tutorials or initial levels that feel like playing the game. Games should increase the player’s skills at an appropriate pace as they progress through the game. Players should be rewarded appropriately for their effort and skill development. Game interfaces and mechanics should be easy to learn and use.
Control Players should feel a sense of control over	Players should feel a sense of control over their characters or units and their movements and interactions in the game world.

their actions in the game.	Players should feel a sense of control over the game interface and input devices. Players should feel a sense of control over the game shell (starting, stopping, saving, etc.). Players should not be able to make errors that are detrimental to the game and should be supported in recovering from errors. Players should feel a sense of control and impact onto the game world (like their actions matter and they are shaping the game world). Players should feel a sense of control over the actions that they take and the strategies that they use and that they are free to play the game the way that they want (not simply discovering actions and strategies planned by the game developers).
Clear Goals Games should provide the player with clear goals at appropriate times.	Overriding goals should be clear and presented early. Intermediate goals should be clear and presented at appropriate times.
Feedback Players must receive appropriate feedback at appropriate times.	Players should receive feedback on progress toward their goals. Players should receive immediate feedback on their actions. Players should always know their status or score.

Immersion Players should experience deep but effortless involvement in the game.	Players should become less aware of their surroundings. Players should become less self-aware and less worried about everyday life or self. Players should experience an altered sense of time. Players should feel emotionally involved in the game. Players should feel viscerally involved in the game.
Social Interaction Games should support and create opportunities for social interaction.	Games should support competition and cooperation between players. Games should support social interaction between players (chat, etc.). Games should support social communities inside and outside the game.

Table 2.1: Table showing the GameFlow Framework.

Each of these criteria should be given a score between 0 - 5 where

0 - N/A

1 - Not at all

2 - Below average

3 - Average

4 - Above average

5 - Well done

The average score within each element is then used as the score for that element and the average score of all the elements are used as the overall score of the game.

2.4.3 Usage of Gameflow in this Thesis

One problem with the GameFlow framework is that scores for different criteria are subjective. As an example one person might feel like the controls in a game are intuitive and give them good control over their character, while another might feel that it is lacking and give it a lower score.

In their original presentation of the framework, the researchers used "expert opinion" to set the scores. In their case that expert opinion was their own, which obviously raises questions about their objectivity since they have an incentive to prove their framework successful.

Unfortunately, flawed as the method of self-scoring is, it appears to be the most practical one. The number and level of technicality of the questions means that asking users to rate it is unpractical and useful feedback might disappear in misinterpretations of the questions.

Because of this our approach will be to self analyze the application based on the feedback we get from our user testing. This does however create the same problem with bias that the researchers faced in their initial presentation of the framework, and require our readers to be critical of our analysis. Despite of these problems, we believe that the GameFlow analysis will present valuable insights into how ours and future games can be improved to increase the level of enjoyment felt by the users.

It should be noted that the GameFlow framework is extensive while our game is quite simple. Therefore several of the criteria and questions regarding the criteria might be almost or completely irrelevant. A question like "players should feel a sense of control over their characters or units and their movements and interactions in the game world" is important for role playing or real-time strategy games, but for our quiz game it doesn't make much sense.

2.5 System Usability Scale

The System Usability Scale (SUS) is a frequently used way of measuring the usability of an application for the end user. It consists of ten questions with five options shown in Appendix C - Questionnaire. Each of these options has a score between zero and four and is multiplied by two and a half so they add up to 100 for all questions. Half of the questions have the rightmost option as the most valuable and the other half have the leftmost as most valuable. The average SUS score is 68 and a score above this would classify the system as good.

The SUS is designed to be used when the user has had time to use the system, but no training or explaining has been done. Within the usability umbrella term, SUS tries to measure both how much users enjoy using the system, and how efficient they can use it to fulfill their goals.

3 Earlier Work

In this chapter we will look at earlier work done in the field of educational games, solutions to make multiplayer games for mobile devices, and the time-shifted game Real Racing 3.

3.1 Synchronous Games - Kahoot!

Kahoot! is a synchronous multiplayer quiz game designed to make lectures more engaging and fun for the students [11]. The game supports many users and can be played by an entire class at the same time, allowing everyone to participate in the learning experience. The game begins with the quiz creator making a multiple-choice quiz with multiple questions that the participants answer. The game lasts until every question has been answered. The game has received good reviews from both experts and users with an average score of 3.9 out of 5 on Google Play [12] and was one of the finalist in the 2013 LAUNCHedu [13].

Since Kahoot!, shown in Figure 3. 1: Screenshot from the game Kahoot!, is such a successful game we will use some of its elements when developing our application. We will also try to find flaws in the design to improve on to make our application as good as possible.

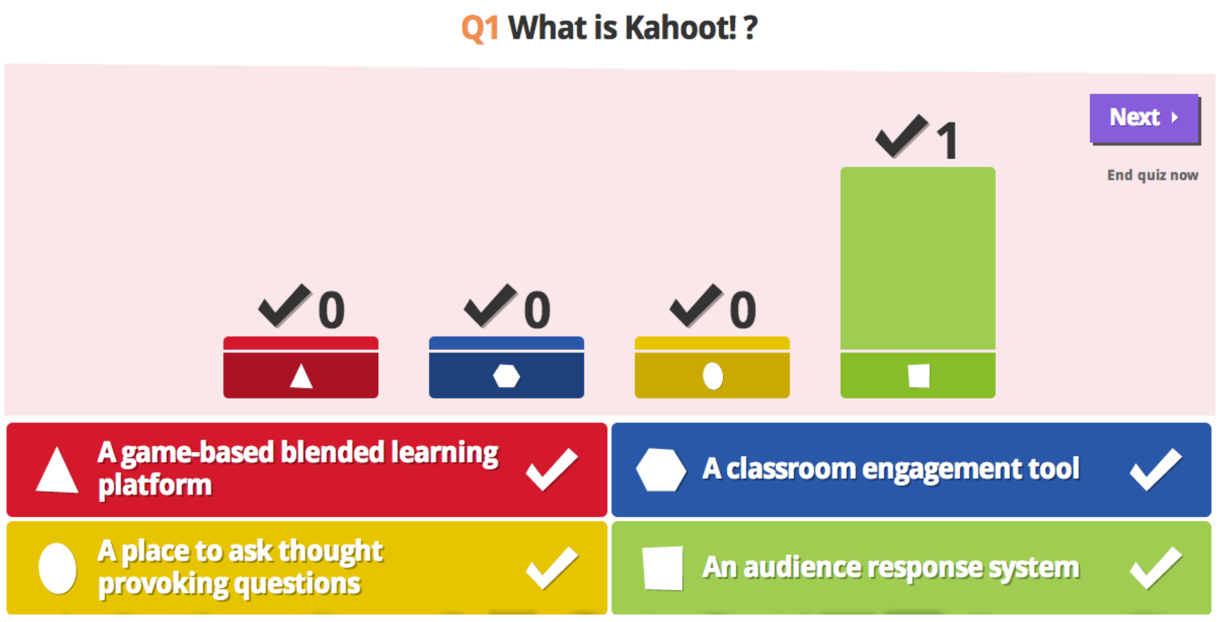

Q1 What is Kahoot! ?
Next
End quiz now
✓0
✓0
✓0
✓1
A game-based blended learning platform
A classroom engagement tool
A place to ask thought provoking questions
An audience response system

Figure 3. 1: Screenshot from the game Kahoot!

3.2 Asynchronous Games - Wordfeud

Wordfeud is an asynchronous crossword game for smartphones, shown in Figure 3. 2: Screenshot of Wordfeud played on a mobile phone., where you play against an opponent on a shared game board [14]. The game is available for Android, iPhone and Windows Phone 7 and has millions of downloads. It lets you challenge someone from your contact list, a Facebook friend or a random person. The game also has an in-game chat which makes it easier for the players to communicate.

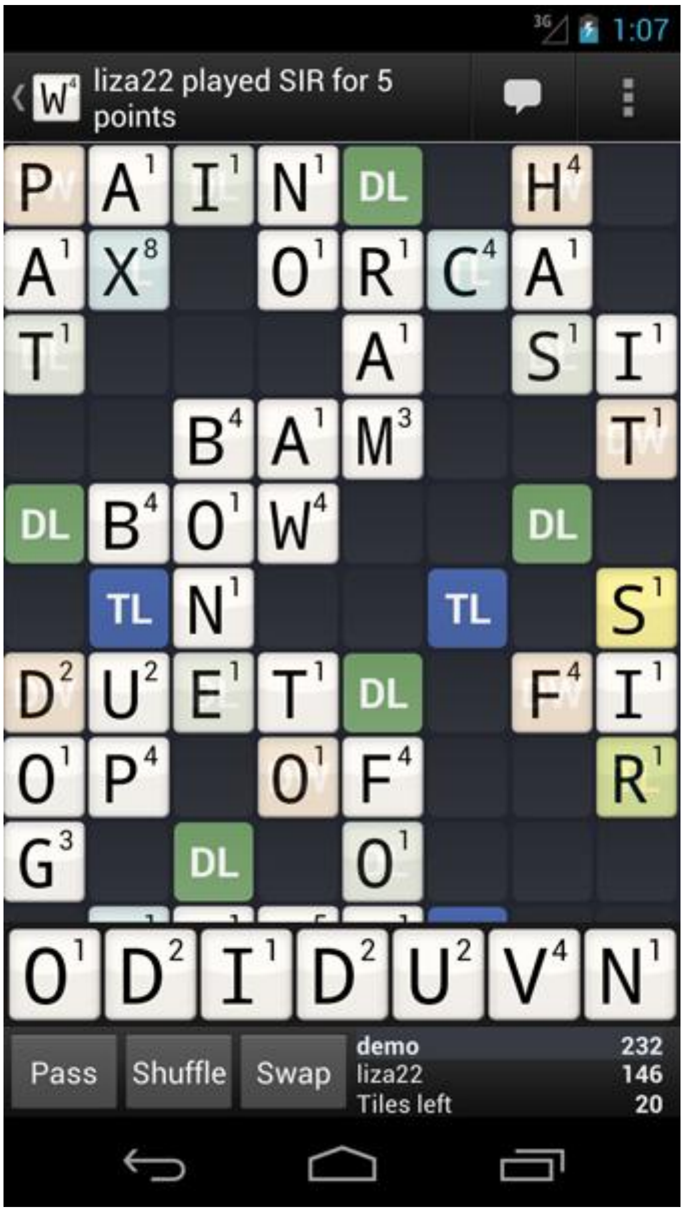


Figure 3. 2: Screenshot of Wordfeud played on a mobile phone.

During a game the players take turns playing words and placing them on the game board. This means that it can take some time between each player's turn and the game loses the kind of flow a synchronous game has. This is a good thing if you just want to play a few minutes now and then, but it punishes the players that want to play for long periods at a time. You could argue that you will be able to play for long periods at a time if you have many games going at once, up to 30, but you will always have to wait for your opponent to finish her turn before you can play.

When developing our application with time-shifting we will need to take elements from both synchronous games like Kahoot! and asynchronous games like

Wordfeud. We will look at how these games work and try to come up with techniques that can be combined to make time-shifted games work.

3.3 Time-shifted Games - Real Racing 3

In 2013 Real Racing 3 was released by EA games after being developed by FireMonkey studios. They coined the term "time shifted multiplayer", TSM, when explaining their multiplayer mode.

Before this, there had been two different types of multiplayer for racing games. One is the traditional multiplayer where two or more players race against each other in real time. The other is "ghosting" where one player will finish the game first, and the other will play the same course later with the "ghost" of the other player visible, but without interaction [15]. A screenshot of the game can be seen in Figure 3. 3: Screenshot of Real Racing 3 action. below.

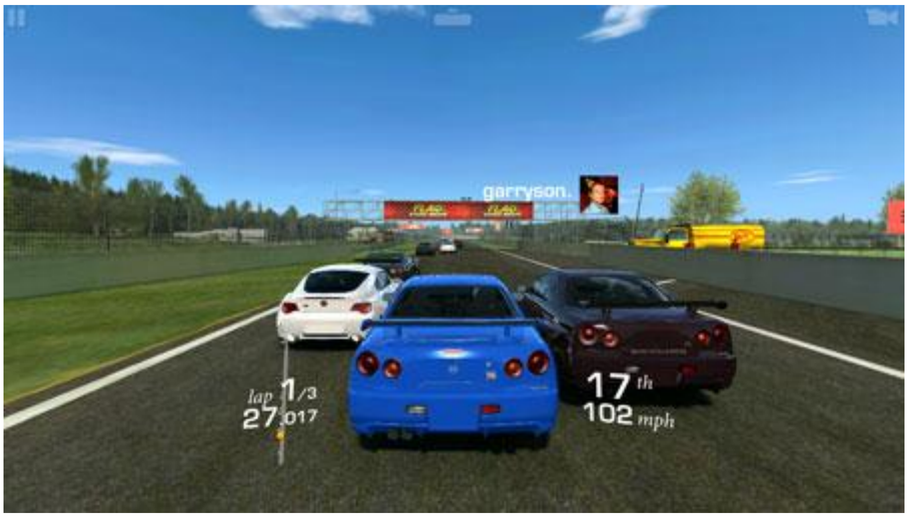


Figure 3. 3: Screenshot of Real Racing 3 action.

TSM is a mix of the traditional multiplayer, and ghosting, where they create a profile based on the first player's performance and driving style. This profile is then used to understand how the player would react to different situations, like getting cut off or bumped into, and use this to create an interactive "ghost". The goal of the TSM is to give players the feeling of playing against each other in real-time even though they are not.

4 Technologies

In this chapter we will give an introduction several technologies for web-based development that we have used in our application. This chapter starts with introducing the client side technologies first and moving downwards into the application until we reach the database technologies.

4.1 HTML

HTML5 is the fifth and latest revision of the HTML (HyperText Markup Language) [35]. HTML is a markup language for web applications and is used to create the structure of an application. HTML5 can be used for cross-platform development which made it an obvious choice for this project, even though we only developed an application for Android. The application was mainly tested on an emulator in the beginning of the development process to make sure all the elements worked properly. Later on we used mobile phones to do the tweaking to make sure everything looked right. We did experience some differences in how the application looked on the emulator and phones so our choice of not using the emulator all the time was the correct one. We did however speed up the development process by using the emulator in the beginning since deploying the application on your phone is a bit time consuming.

4.2 CSS

CSS stands for Cascading Style Sheets and is used to style HTML elements in an application [34]. We used the third version of CSS, which is the newest one, in our application. CSS makes it easy to change color of the application background, fonts, button styles and more. We used CSS actively to make the application look somewhat fresh and intriguing. The progress bar used as a timer for each question is purely styled by CSS and requires quite a few lines to look like it does. The actual movement of the timer is done by a JavaScript.

4.3 JavaScript

JavaScript is a dynamic programming language mostly used for client-side scripting and its role is to add functionality to HTML documents and communicate with the server. It is a multi paradigm programming language, supporting object oriented, functional and imperative programming styles [36]. Despite its name suggesting otherwise, JavaScript is mostly based on C and has no connection to the Java programming language.

On the frontend part of the application we used JavaScript to handle changes between views. We also used JavaScript to make some of the styling look more alive, like the progress bar, and to make the notification text behave as it does.

4.4 jQuery

jQuery is a JavaScript library that makes it easier to manipulate HTML elements and animations across several browsers [31]. jQuery can also be used to modify the styling of an element, handle events like button clicks and add effects like fading and sliding to HTML elements. We were a bit uncertain on how jQuery would work when developing applications for phones, but it seemingly works as it would on a computer and did not create any problems. We did however only use the most basic methods in the library, and there might be some problems if you dig deeper.

4.5 Backbone.js

Backbone is a MVC framework for web development designed to structure web applications [30]. It separates the views, which can be thought of as the screens users see, from the models, and handles events in the application.

While backbone is not directly used in our project, our development platform Parse is heavily based on it.

4.6 Parse

Parse is the development platform we used for this project. It is a cloud-based platform built on top of Backbone.js, which in turn uses the Underscore.js library [28]. Parse is designed to simplify the development and their written goal is to "totally eliminate the need for writing server code or maintaining servers" [29].

Parse contains a database, pre made objects and methods for handling standard functionality like login for users and queries. Several SDKs for different platforms including iOS, Android and JavaScript are available. Parse also has functionality for push notifications that makes device-to-device communication very easy, in addition to channel subscriptions that can be used to group users to make communication easier.

4.7 Android Studio

Android studio is an IDE for developing android applications based on the popular IntelliJ IDE.

It simplifies application creation by having pre made templates for the many different file types used in developing android apps, and abstracting away the low level requirements for making apps run on mobile devices. In addition to this, it presents a simple drag and drop editor for creating the most common elements for the user interface.

Android studio also offers a large variety of emulated devices allowing the user to get a feel for how the app would look and behave on different devices. This seems like a really good feature, but in reality it does not always behave the same way in the emulator as it does on a phone.

5 Time-shifted games - Time-shifted Quiz Game

We have earlier created a proof of concept for a time-shifted quiz game, and we will present our main findings in this chapter. In Section 5.1 we will discuss the two different methods we have found to create time-shifted behavior in games. Section 5.2 will discuss the opportunities and limitations time-shifted games face if it is to become useful.

5.1 Methods of Creating Time-shifted Behavior

In our research, we found two different methods for creating time-shifted behavior, both with their limitations to where they can be applied. It is important to note that time-shifting is a new field, and other methods might exist that we are unaware of.

5.1.1 The Player Based AI Method

In this method you create an artificial intelligence to emulate the behavior of a player that has already finished the game.

The only big commercial attempt of a time-shifted game so far is FireMonkeys Real Racing 3 which was also the game that coined the term "time-shifted multiplayer". The method they used for time shifting was to create an AI based on the player's driving style and lap times. This is a possible solution but raises some issues.

In several genres, creating these player based AIs is very difficult, including for racing games like Real Racing, shown by the criticism the game received [17].

One important aspect of this is that humans and computers are good at different aspects of games. To use quiz games as an example, a well-programmed AI will never get a math question wrong, but humans do all the time. To create an AI for this, we have to analyze what kind of mistakes the players does, at what frequency and so on.

Humans on the other hand, are better at understanding abstract concepts than computers. A question like "What is gray and has a long trunk?" is easy for us, but creating an AI that can understand and answer questions like this has yet to be done.

A player based AI can only be successfully used in situations where computers are better than humans, if it correctly mimics the mistakes the player it is based on would do.

Chess is a good example of this and apps like Play Magnus allows you to challenge an AI that is as good as the current reigning world champion. Using the Elo rating [41], the AI is as good as he was at different ages [18]. But the same problem that Real Racing 3 had exists in this game as well. While creating an AI that is equally good as the player they are based on is relatively easy to do, creating one that mimics the play style by doing similar types of mistakes is often significantly more difficult.

This limitation becomes more apparent in games where humans outperform computers. Like the quiz example given earlier shows, sometimes it is impossible to create an AI based on a player because no known AI methods can mimic a humans actions in the given situation.

Another issue with the player based AI method is that, even if the AI perfectly mimics the behavior of the player, you are still beating or losing to an AI and not the actual player. This knowledge is likely change how the player feels about the experience. Intuitively we would assume that the intermission and enjoyment would be lower than in a real time game, but more research would be required to understand the extent, or if it is even correct.

5.1.2 The Stored Input Method

Instead of creating an AI, we can store the input directly and this is the strategy we used in our quiz application. When the first participant plays through the game, it stores all the relevant input from that player and the timing of it. When the next participant plays, the game will feed the input from the first participant into the game at the given time. For the second participant, this creates the illusion that they are playing the game in real time since the input of the first participant behaves in the exact same way as the player did.

This method creates a game play that is closer to the real time experience since the play done by the stored input is the same as the player did. But this method also have more limitations. Since the time shifting is done by directly storing the input with no AI making decisions, there is no way to predict new situations. Because of this, this method cannot be used in games where these new situations can arise. Practically, this means that the game cannot allow for any interaction between the players where they affect each other's game play. As an example, this method is not possible in a racing or fighting game where you have to react to your opponent's attacks or position.

In a quiz game, these direct interactions do not happen. While it is possible for a player to react to another, e.g. a quick answer by an opponent might influence how much time the player uses, but it does not directly affect the play in the way being crashed into in a racing game would.

5.2 Opportunities and Limitations of Time-shifting

In this section we will look at the opportunities opened up and limitations imposed by time-shifting in games that we found during our work with creating a proof of concept for a time-shifted quiz-game.

5.2.1 Opportunities of Time-shifting

There are several opportunities for Time-shifting in games.

Increased convenience compared to synchronous multiplayer games

In a synchronous multiplayer game every player has to be ready to play at the same time, and continue to play for the duration of the game. This is the main limitation time-shifting in games aim to get around by allowing players to get the positive effects of real time synchronous game play, but to add the convenience of the users playing at a time convenient to them regardless of the other participants in the game. Time-shifting allows players to choose when to play, pause and end a session without affecting the other players of the game.

New types of multiplayer games

With mobile platforms becoming increasingly popular, they bring with them a new style of playing. While console and computer gaming often consists of long, continuous sessions, games for mobile platforms often favor a style of playing where you play whenever you have a dull moment and often for a short period of time. An example would be to play while you are waiting for the bus, or while waiting for a lecture to begin. Because these sessions often happen at different times for different people, playing with your friends become very difficult as your dull moments would have to occur at the same time, and last equally long. Time-shifting can be a solution to this by creating the illusion of real time game play even when playing at different times.

Added realism

The Stored Input Method explained in 5.1.2 The Stored Input Method can create computer controlled players that perfectly mimic the actions of a real player and this is might create more realistic behavior than hard coding an artificial intelligence for the game. This is especially true for games where humans currently outperform computers. As explained in chapter 4.1.1 it is very difficult to create AIs that mimic human behavior and in situations where humans are better than computers it might be impossible. With The Stored Input Method we circumvent the problem by not creating an AI at all, but using an actual humans results and feeding them back into the system.

Educational use

Time-shifting creates the opportunity to further develop serious games in education. Games designed to improve education in a classroom environment already exists and are increasingly used[33]. Time shifting can be used to bring the immersion these games provide outside of the classroom [19].

A recent user test was conducted on the game Kahoot! where computer science students used the game in every single lecture for a whole semester and answered a survey at the end. The overall results were the following: 85% of the students thought it was fun to compete against others, 72% said they concentrated more when playing against other students, 89% said they were engaged and had fun while playing and 83% wished Kahoot! was used in other classes [19]. The result of this user test is really positive and shows that there is a potential for the use of a quiz type of game in education.

Quiz games have also been tested on younger students with great success. The game Lecture Quiz, which is an earlier version of Kahoot!, was tested on 40 7th graders at a Norwegian primary school [19][20]. The game was used in a social science class to test the students' understanding of the subject and give the teachers feedback on how much the students knew. It was also a good way of letting the less chatty students show what they knew without them having to raise their hand. The test clearly showed that bringing technology into education creates much, sometimes a bit too much, excitement for youngsters and that they would love to use it again. The teachers were also happy with the result but they did not think that a quiz could replace a normal test. They did however think it was a good option for making classes more varied [20].

These user tests show that educational games have potential, but more research is required to see if the results given here are transferable to a time shifted educational game.

5.2.2 Limitations of Time-shifting

During our work with the quiz game and research in the field, we discovered several limitations and problems that for the time being are unsolved. Since time shifting is a new and small field, there might be more limitations that have yet to be discovered, and possibly unknown solutions to some of them. Thus it is important to remember that the following list is not necessarily all inclusive, but represent the unsolved limitations the authors are currently aware of.

Time shifting is impossible for the first player

The illusion of simultaneous play will not be possible for the first player. The method used to time-shift is to store relevant input from the players, and then present it at given times to the next players. So player B can play against the stored data of player A, player C against player A and B and so on. But for A there will be no data available.

One possible mitigation of this problem is to use both the time-shifting methods discussed in Section 5.1 Methods of Creating Time-shifted Behavior. Using our quiz game as an example, we use for player B the stored input method with input from A to create time-shifted behavior. This is of course impossible if A has not yet played, but we could use the player based AI method on another quiz finished by A to get an estimate of how A would do in this quiz. Once again, like it is discussed above, this raises the problem that you are not actually playing against the person, which could hurt the user experience.

Winner problem

This limitation is a special case of the one above. Picking the winner of a time-shifted game is difficult because the games are played at different times, so when player A finishes a time-shifted game, her opponent player B has not yet finished the game, and his results are not available yet. This is likely to break the illusion of real-time game play as she will either have to wait for Bs results before knowing whether or not she won, or we might end up in a situation where they both think they win or both lose the same game.

To exemplify this, assume that we are using the Player Based AI Method for the first player(A) and then the Stored Input Method for the second player(B). In this situation the A might beat the AI based on B, and thus think she has won the game. But then the real B beats her score in his play through. In this situation they both think they won the game which is obviously not ideal when we're trying to recreate the real-time multiplayer experience.

If we don't use the Player Based AI Method on A, then she will have to wait for B to finish his game before she knows whether or not she has won, which is obviously different from the real-time experience we are trying to create.

So far we have not found any good way of circumventing this problem, and in our application we avoid it by not having time shifting behavior for the first player. It is however worth to mention that we can store every person who has ever played the quiz, so unless you are the actual first person to play it, you will have time-shifted opponents, but they will not be the person you wanted to challenge.

Players should have little to none influence on each other

Time shifting in games where players compete on score, like in our quiz game, are relatively straight forward. Player A finishes his game first, and the game saves the score, other relevant input and the timing of the input. When player B plays, the game will feed her information about player A at the correct times, creating the illusion that they are playing at the same time.

In games where the players actions affect each other, like racing games, it gets way more complex. As an example: player A finishes his lap with no problems. When B plays, she crashes into A at some point during the lap. This would obviously affect how A behaves, but since he already has finished the game, there is no way of knowing how. Real Racing 3 tried to create a time shifted racing game by profiling the players, and then creating AIs based on it. While this is a possible solution, you are not actually playing against each other anymore since the same game might be different for the involved players (In B's game, there was a crash, but no crash happened in A's variation).

Because of the novelty of the time-shifting field, exactly how much and what kinds of interactions that can be implemented is unknown. In Section 11.5 we discuss an existing game where players influence each other in a more direct way, and time-shifting with the Stored Input Method could still be possible.

Two-way communication

One of the goals of time shifting is to recreate the multiplayer experience, and a part of that is communication between players. With time shifting, one way communication from player A to everyone that plays after him is easily achievable as it is a type of input that we can store together with its timing and feed to all following players. Unfortunately, A will already have finished or moved to a different part of the game by the time B receives the message, so no return message can be sent for that part of the game.

Part III

Own Contribution

6 Architecture

In this chapter we will give an overview of the architecture for the application. The first section covers the requirements for the application which is further divided into the functional and nonfunctional requirements. The second section show a model of the architecture with explanation and the last section shows the architecture for the database.

6.1 Requirements

Our main goal for the project is to create an application suitable for testing time-shifted games on users. This means that it has to be of a high enough quality that users are comfortable using it. If we fail to create a good user experience, this is likely to pollute the results we`re trying to get about the usefulness of time-shifting. The requirements are, as usual, divided into functional and non-functional requirements. Section 6.1.1. will explain our functional requirements for the application and why we have chosen them. Chapter 6.1.2 does the same for the non-functional requirements.

6.1.1 Functional Requirements

A functional requirement is, as the name implies, a functionality the system is required to have. In non-technical terms it is what the system is required to be able to do for it to work correctly.

Some of these functionalities are more critical than others and we divide them into three different priority levels.

High priority means that the application will be useless without it for our testing purposes, and failing to complete any of these will leave us with a useless application.

Medium priority is functionality that, if not present, will become a source for pollution in our test results or prevents us from testing parts of our application. Failing to fulfill a medium requirement does, in contrast to a high requirement, not leave us without a testable application, just one that is partially testable or gives less reliable results.

Low priority is the lowest level of priority in our application. Functionality with this priority are not critical to our application, but we think they add to the experience either by making the time-shifted games feel more like real-time or from a purely usability perspective. Failing to fulfill these requirements are not expected to affect our results in any meaningful way, but it might hurt the overall usability of the application.

Table 6.1 lists all the functional requirements for our application:

ID	**Name**	**Priority**	**Description**
FR1	Register	High	The user must be able to register with an account the first time she starts the application to associate her device with the user account.
FR2	Sign in	High	The user must be able to sign in and out of her user account.
FR3	Start game	High	The user must be able to start a quiz battle.
FR4	Real-time multiplayer	High	The users must be able to play multiplayer games against each other in real-time.
FR5	Time-shifted multiplayer	High	The users must be able to play time-shifted multiplayer games against other players.
FR6	Multiplayer	Medium	The user must always have an opponent, human or AI.

FR7	Timer	Low	The application must present a timer to let players know when the next question is ready.
FR8	Answer notification	Medium	The application must let the user know when an opponent has answered.
FR9	Score	Low	The player must be able to see their own score and the scores of the top players between questions and after the quiz is finished.
FR10	Challenge	Medium	The users must be able to challenge their friends to play a quiz against them.
FR11	Friends	Medium	The users must be able to send, accept and reject friend requests from other users.
FR12	Security	Low	The user accounts should be protected by a password created by the users at time of account creation.

Table 6.1: Functional Requirements.

6.1.2 Non-functional Requirements

While functional requirements deal with what the system should be able to do, non-functional requirements gives us operational criteria that the system must reach.

The application is not supposed to be a finished product ready for release, but good enough that we can test time-shifting on users. Because of this, several of standard non-functional requirements are not prioritized. Among others, we have chosen to not prioritize security or availability since the application will only be used in a controlled environment, and only for a short period of time. The non-functional requirements are listed in Table 6.2.

ID	Name	Priority	Description
NFR1	Realistic time-shifted bots	High	70% of the users should not be able to differentiate between time-shifted and real time players.
NFR2	Capacity	Medium	The application should handle up to 50 simultaneous players.
NFR3	Enjoyment I	High	80% of the users of the application should prefer it over ordinary studying.
NFR4	Enjoyment II	High	70% of the users should feel that time-shifted multiplayer gives the same enjoyment of a game compared to real-time multiplayer alone.

NFR5	Maintainability	Low	The source code should be divided into appropriate classes dependant on their corresponding view.

Table 6.2: Non-functional Requirements.

6.2 Application Architecture

The overall architecture of the application is illustrated in Figure 6. 1:

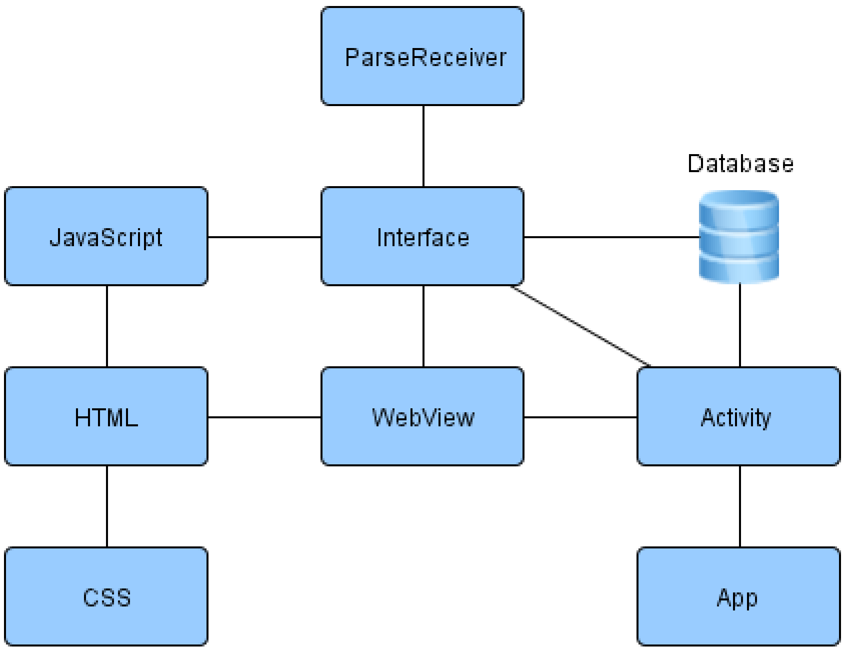


Figure 6. 1: Overall architecture of the application.

The class that initiates the application is 'App'. In 'App' the connection to Parse is initialized and the first activity is launched. Each activity is connected to the database to do queries for users, get questions for quizzes, save user scores and so on. The purpose of an activity is to have a common connection point for the interface, web view and HTML page. When a new activity is created it is put on top of the activity stack. It is possible to go back to an underlying activity while the one on top of the stack is still there, but that is not recommended practice and the top activity should be destroyed if the player is not supposed to return to that activity.

Each interface is connected to the a parsereceiver, the database, a JavaScript file and a web view. The interface handles communication between clients through the parsereceiver, queries the database and calls JavaScript functions. To send a message from one client to another through the parsereceiver we use JSON objects. Whenever a client's parsereceiver receives a JSON object it parses it and calls functions in the interface accordingly, e.g. when a user joins a game lobby a message is sent to all other users in that lobby to notify them that someone joined. To make the user aware of a new player in the lobby the interface then calls a JavaScript function that prints the newcomers name in the lobby list. This function call is done through a JavaScriptInterface which is a part of the Android Webkit.

More specifically the parserecievers works as follow: When it receives a JSON object it checks the value of the type field in the object and calls functions accordingly. To make sure the JSON objects are only sent to the correct clients we use Parse channels. To receive JSON object on a channel the client needs to subscribe to the channel and only then will she receive the object. Our solution for this is to make a client subscribe to a channel based on the lobby they are in so that only the clients connected to that lobby can communicate with each other with JSON objects.

The user interface of the application is realized with webviews which is simply web pages consisting of HTML, CSS for styling and JavaScript. In addition to ordinary JavaScript that you would find in any web page we also use it to call functions in the matching interface. E.g. when a user answers a question by pressing a button the JavaScripts calls a function in the interface to send a JSON object to all the players in that quiz to notify them that the user has answered. The two-way communication between JavaScript and Java in the interface is a very important part of the application and makes it a lot easier to control function calls and communication between clients.

6.3 Database Architecture

The database architecture is illustrated in Figure 6. 2:

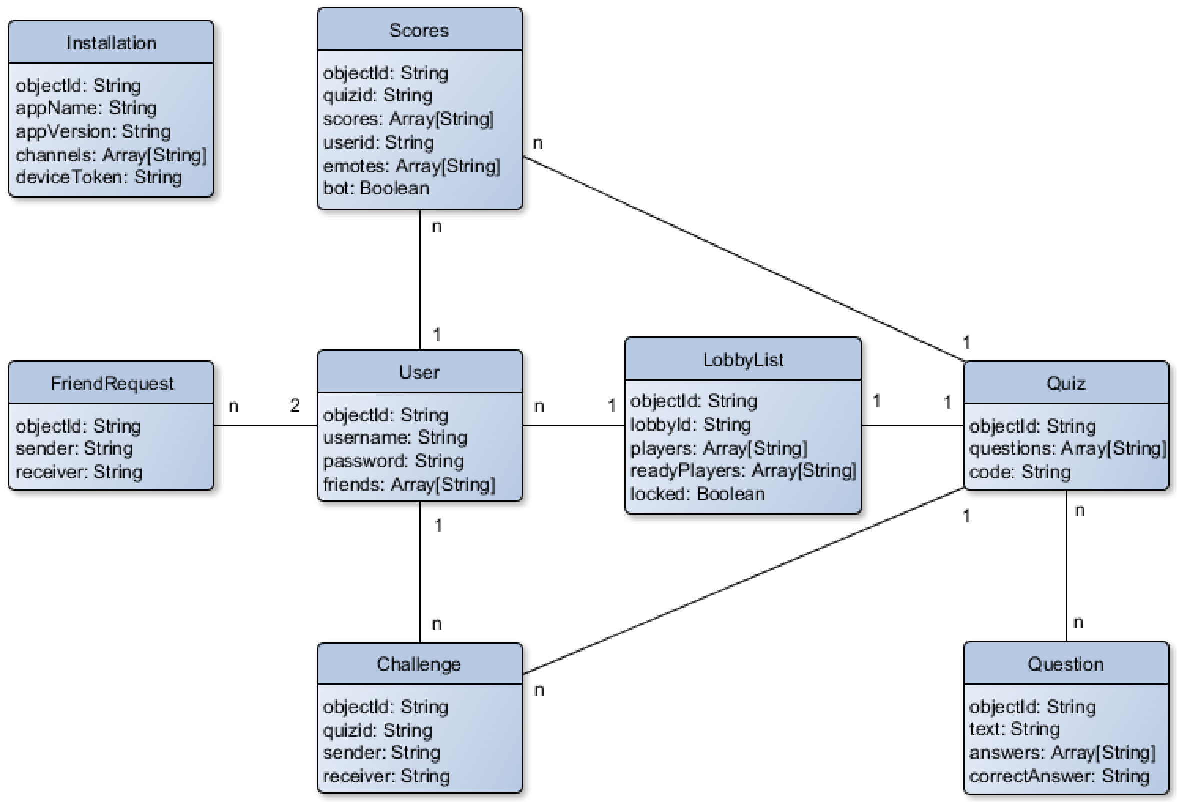


Figure 6. 2: Database architecture.

Installation

The Installation table contains information about the device the application is installed on and what version of the application the device is currently running. The channels array stores information of which channels the device sends and receives messages through. The deviceToken is a unique identifier, a fingerprint, for the device.

User

The User table contains all the information needed for a user of the application. The table is partially predefined by Parse and uses an objectID as the unique identifier. It also contains username, password and a friends list.

LobbyList

The LobbyList table contains information of a game lobby. It uses objectId as a unique identifier, lobbyId to connect it to a quiz, an array of connected players, an array of the connected players that are ready and a Boolean to tell if the lobby is locked or not. The two arrays of players are used to make sure that a quiz does not start until there is enough players and all players are ready. The locked Boolean is used to prevent users to enter a busy lobby. Lobbies are busy when the quiz is ongoing and when cleanup after a failed quiz is performed.

Quiz

The Quiz table contains information about a quiz and is used to hold a list of questions. The table consists of the unique identifier objectId, a questions array and a quiz code. The quiz code is compared to the lobbyId in LobbyList to give players in a lobby the correct quiz. The questions array holds the identifiers of questions that are included in the quiz

Question

The Question table holds data for each question used in the application. It contains a unique identifier in objectId, a question text, an array of answers and the correct answer

Scores

The Scores table holds all the scores for every user and every quiz. It contains a unique identifier, quizId, userId, an array of scores, an emotes array and a bot Boolean. The userId and quizId fields are used to connect a score to the correct user and quiz. The emote array is used to save the emotes a players says after a question. This array is also used to make the time-shifted bots repeat the emotes used by the player when someone is playing versus a time-shifted friend. The bot Boolean is used to check if the scores comes from a computer controlled player or a real player.

Challenge

The Challenge table contains information about the challenges that are sent from player to player. The table contains a unique objectId, the quizId of the quiz a user is challenged in, who the sender of the challenge is and who the receiver of the challenge is.

Friend Request

The Friend Request table contains information about the friend requests that are sent from player to player. The table contains a unique objectId, the name of the sender and the name of the receiver.

In addition the data fields mentioned above every class in the database contains the fields createdAt, updatedAt and ACL (Access Control List).

7 The Application

In this chapter we will show and explain how the application looks and works. In Section 7.1 we will look at it from a users perspective explaining what the users see and can do when they use the app. In Section 7.2 we will look at the implementation of it, how we used the different technologies, our reasoning behind our choices, and issues we encountered.

7.1 User Interface

In this chapter we will present the use of the application from a players perspective. The application is divided into several screens for the different parts of the application.

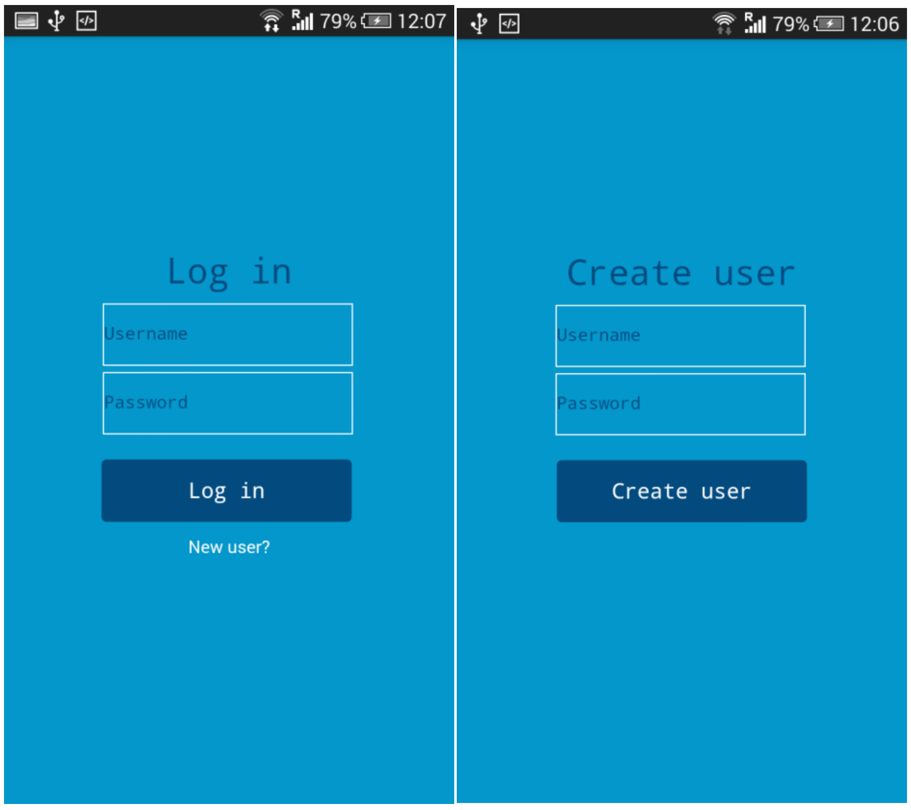


Figure 7. 1: Screenshot of the Login and Signup Screen.

7.1.1 Login and Signup Screen

In the login screen you can log in with your username and password, as shown in Figure 7. 1: Screenshot of the Login and Signup Screen.. This is required to play the game. If you do not have a registered user you can click the "New user?" text to open the signup screen.

In the signup screen you can input the desired username and password. If the username is already taken, a text saying "Username is already taken" will appear and the account will not be created. The app does not have any requirements to password length or symbols, nor does it require the user to activate their account through e-mail, which is usually the norm. These features were excluded because there is simply no need for it. After signing up you will automatically be sent to the login screen.

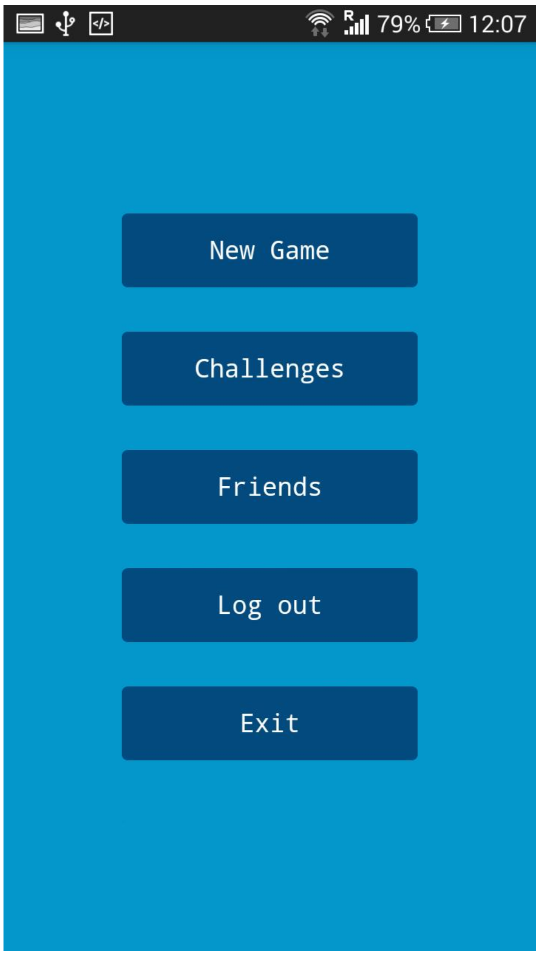


Figure 7. 2: Screenshot of the Menu Screen.

7.1.2 Menu Screen

In the menu screen, shown in Figure 7. 2: Screenshot of the Menu Screen., the user has several options: Start a new game, go to the screen for challenges, check their friend list, log out or exit the application. If the user clicks "Start new game" she is sent to the quiz selector screen. If she clicks "Challenges" or "Friends" she is sent to the challenge screen or the friend screen accordingly. The difference between logging out and exiting the application is that on exit the application will be closed but the user stays logged in to her account.

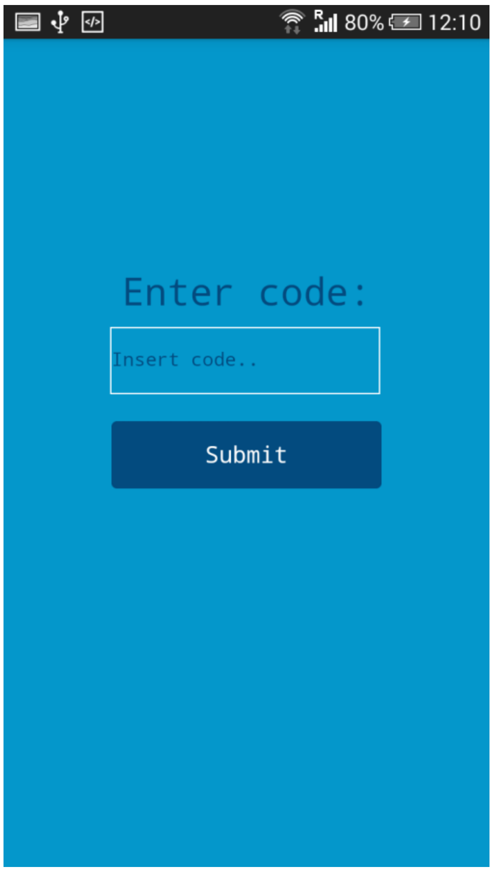


Figure 7. 3: Screenshot of the Quiz Selector Screen.

7.1.3 Quiz Selector Screen

In the quiz selector screen, shown in Figure 7. 3: Screenshot of the Quiz Selector Screen., the user can input a quiz code to create a new lobby for the specified quiz. If a lobby already exists for that quiz code the user is added to that lobby. This is done automatically if the user is joining a quiz through a challenge from a friend.

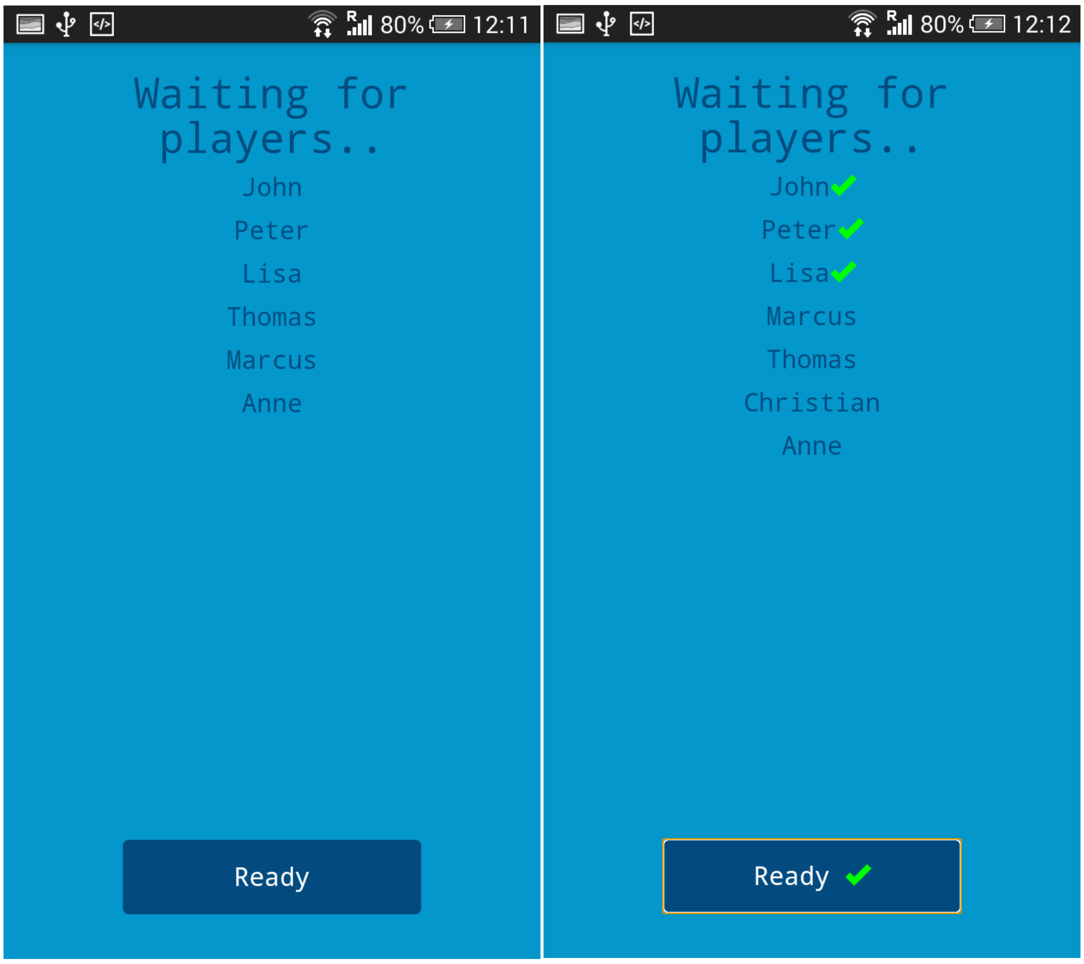


Figure 7. 4: Screenshot of the Quiz Lobby Screen.

7.1.4 Quiz Lobby Screen

The quiz lobby screen has several stages, as shown in Figure 7. 4: Screenshot of the Quiz Lobby Screen.. In the first picture the user has joined a lobby with a few players and is told to wait for more players. When there is enough players in the lobby the ready button becomes visible. This is to ensure that there is a minimum number of players in each lobby so that the experience of playing is better. When the players feel ready they can click the ready button and a green check mark is shown next to their name. When all players are ready a countdown will start and the quiz will begin.

The players shown in the lobby can be human players, predefined bots or bots made from previous players. To hide that a player in the lobby is a bot they are set to join a lobby after a random time and click the ready button after a random time.

This makes it harder for the human players to distinguish between human players and bots.

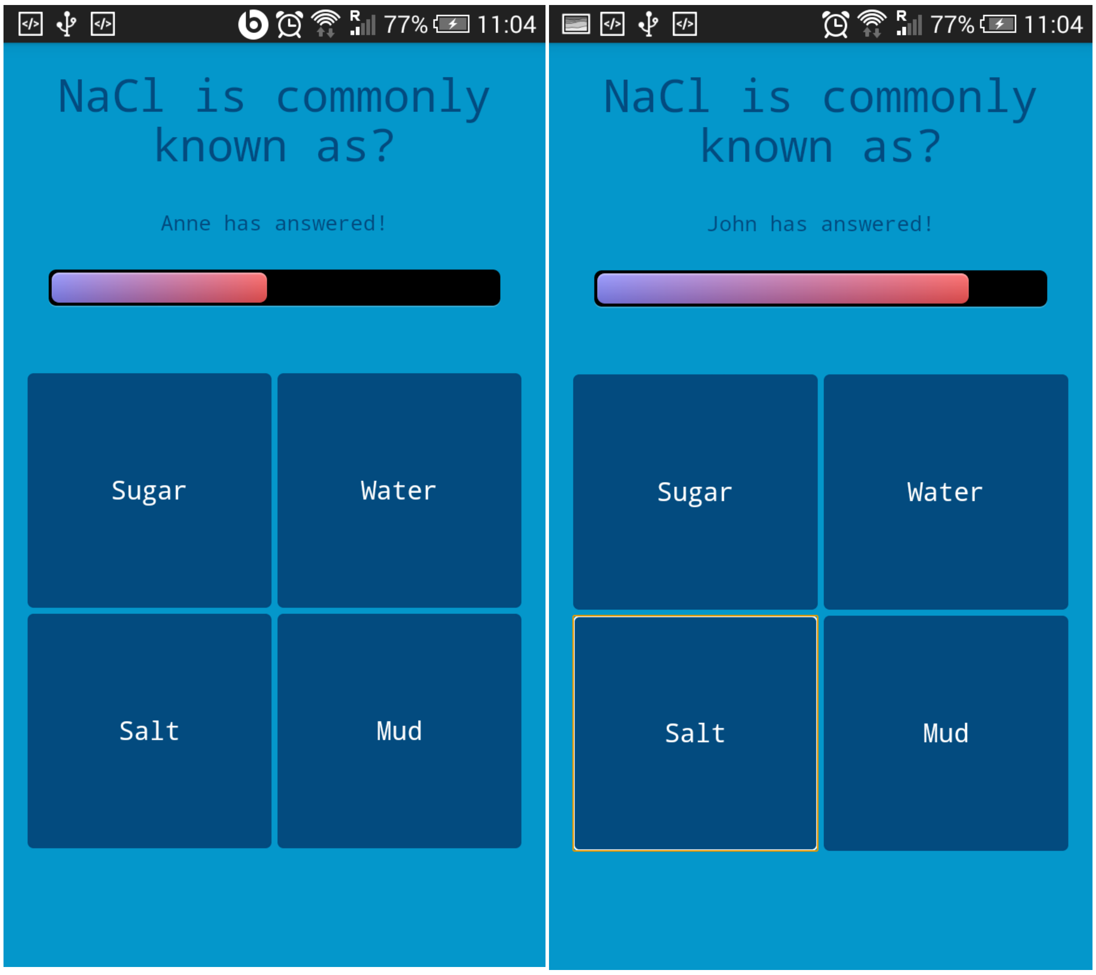
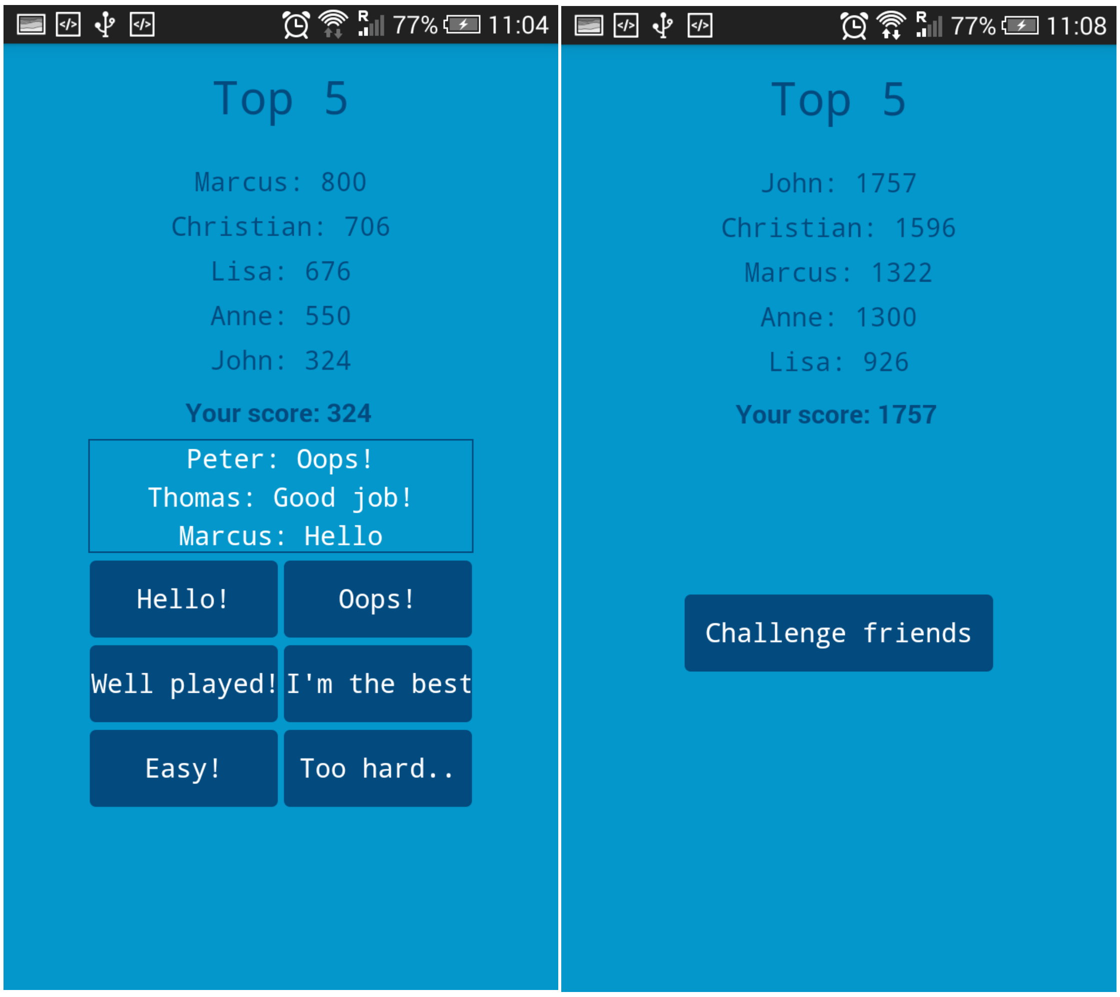


Figure 7. 5: Screenshot of the Question Screen.

7.1.5 Question Screen

In the question screen the user is shown a question text, a notification text, a progress bar and four alternatives. Whenever a player, human or bot, answers a question the notification text changes to "<username> has answered!". This can be seen in Figure 7. 5: Screenshot of the Question Screen.. The progress bar continuously moves towards the right and turns more and more red to indicate that the time is running out. If the player does not answer before the time runs out, or answers incorrectly, she gets zero points from that question. If a correct answer is given, she receives points based on how long she took to answer. When the user answers the question the other players are notified and the selected alternative is highlighted. When the timer runs out the players are sent to a score screen.


77% 11:04

Top 5

Marcus: 800
Christian: 706
Lisa: 676
Anne: 550
John: 324

Your score: 324

Peter: Oops!
Thomas: Good job!
Marcus: Hello

77% 11:08

Top 5

John: 1757
Christian: 1596
Marcus: 1322
Anne: 1300
Lisa: 926

Your score: 1757

Figure 7. 6: Score and Final Score Screens.

7.1.6 Score Screen

In the score screen the players are shown the top five scores for that question in addition to their own score. That way the players know how far ahead or behind they are compared to the top five. Each score screen also has a small chat where players can send messages to each other by using the predefined emotes on the bottom. These emotes are also used by the bots to make it even harder to distinguish them from human players. To avoid people spamming the chat with emotes there is a limit of one emote per player per score screen. The emote a player uses after a quiz is saved so that it can be replayed when the player is used as a bot in another random lobby or when challenging a friend. If there are more questions remaining in the current quiz the players will automatically be sent to the next question after some time. If not, they can keep watching the final score or go back to the main menu, as seen in Figure 7. 6: Score and Final Score Screens..

In the final score screen the players are shown the top five scores for the whole quiz in addition to their own score. There is also a button for going to the create challenge friends screen. The challenge friends feature lets the players challenge someone on their friends list to do the same quiz as they just completed. Whenever a quiz is completed the quiz lobby is cleared and all the players are unsubscribed from the quiz channel. The score and emotes for each player that took the quiz is stored in the database and can later be used to create a bot based on how the player played and which emotes the player sent between questions.

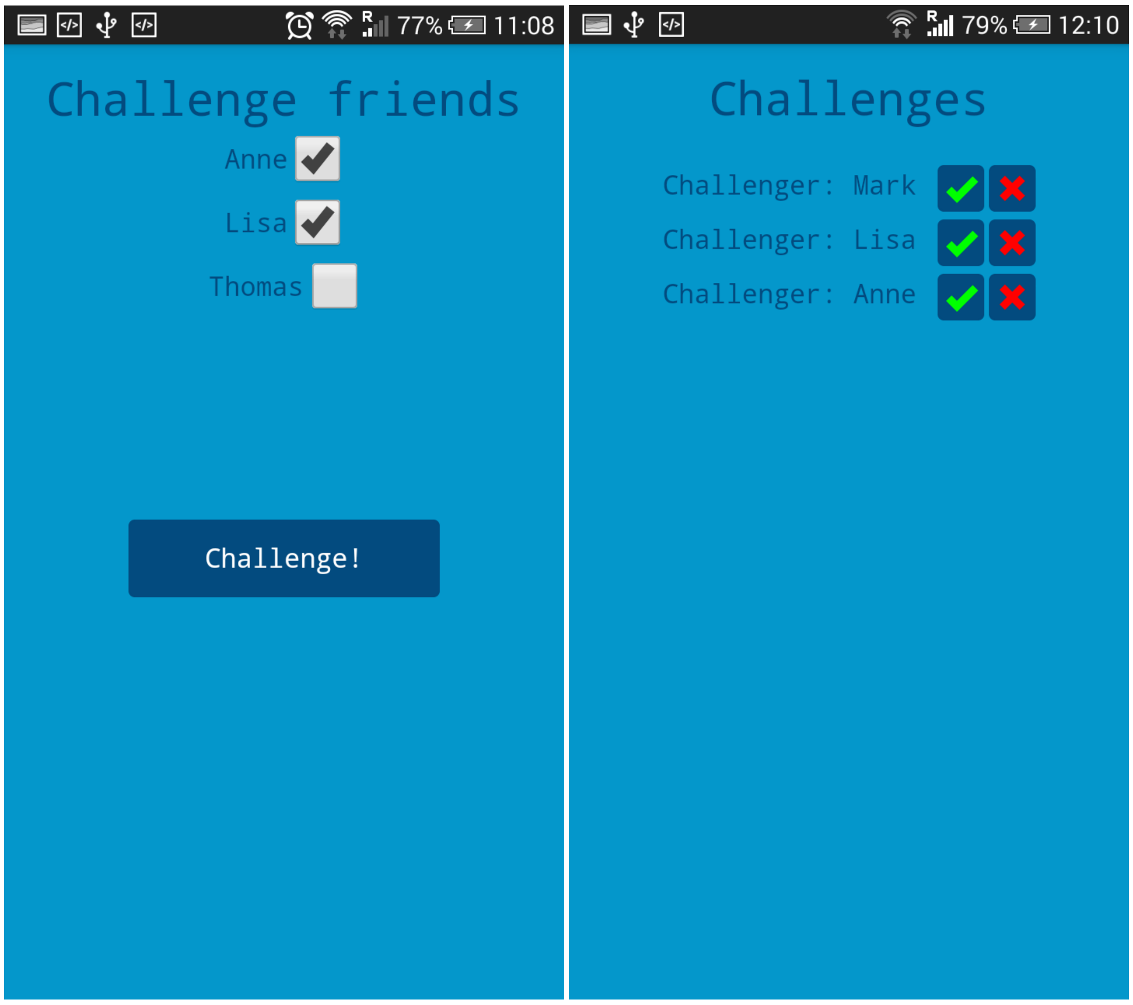


Figure 7. 7: Screenshot of the Challenge Friends and Challenges Screen.

7.1.7 Challenge Friends Screen

In the challenge friends screen the user will get a list of all their friends and a checkbox they can check to select multiple friends. The sending of the challenges is completed by clicking the button in the middle of the screen, as shown in Figure 7. 7: Screenshot of the Challenge Friends and Challenges Screen..

In the challenge screen the user will see all the challenges she has received from her friends. She can accept or decline the challenges by clicking the buttons next to the challenge. If she clicks the accept challenge button she will automatically be sent to a lobby for the quiz she has been challenged in. If she clicks the decline challenge button the challenge will be removed.

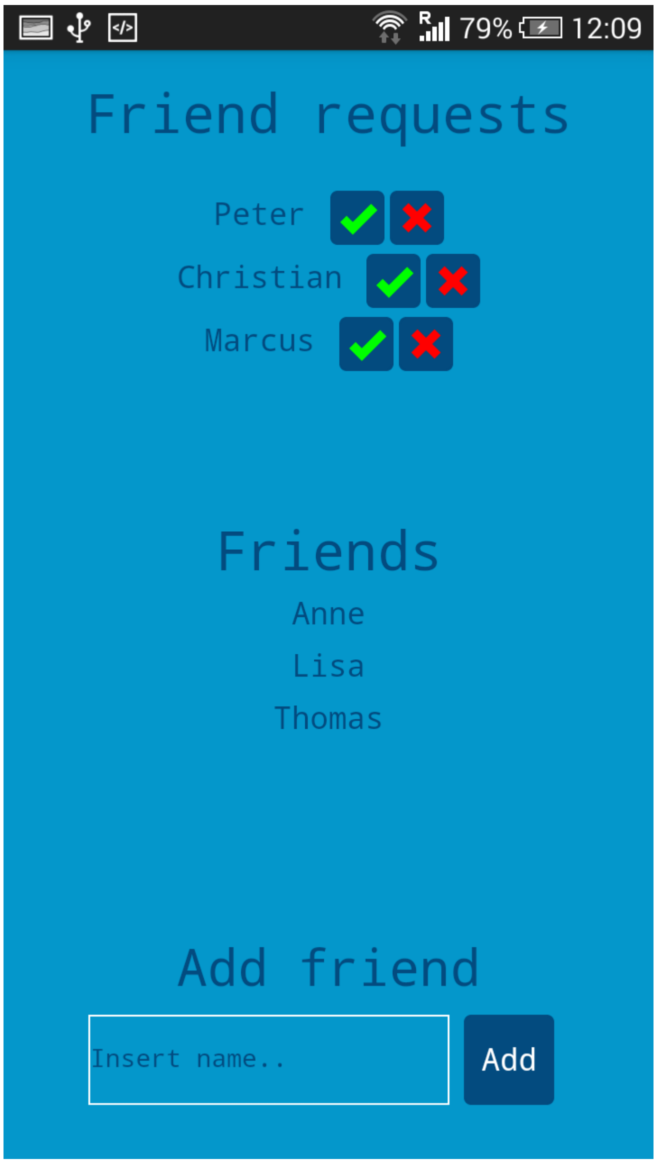


Figure 7. 8: Screenshot from the Friends Screen.

7.1.8 Friends Screen

In the friend screen, Figure 7. 8: Screenshot from the Friends Screen., the user will see her new friend requests, her friends list and have the opportunity to add new friends. Friend requests are accepted or decline by clicking the accept or decline button next to the name of the user that sent the friend request. If she chooses to accept a friend request the friend is automatically added to her friends list. If she chooses to decline a friend request the request is removed. To add a friend the user simply has to input the name of her friend in the input field and click "Add". The recipient will then receive a friend request which will be displayed in the same way as in this screen.

7.2 Implementation

In this chapter we will go in depth into how the application was implemented, reasoning for our choices and issues we encountered.

Much of our inspiration for this application was the Kahoot! quiz game in addition to an earlier project where we developed a prototype of a similar application. In the same way Kahoot! is designed to improve the interaction within the classroom, our goal for the project was to create a similar experience for home or group work in addition to implementing and testing how time-shifting would influence the experience.

7.2.1 Platform Choice

Our choice for a platform was really easy to make since we wanted to develop an application for a phone and we both had access to Android phones, in addition, Android Studio has a good reputation and we wanted to try it out. Since we had experience with the Parse framework from a previous project and because Parse has really good support for Android, it was an easy choice. Parse speeds up the process of creating a backend and our earlier experience of the backend components in Parse was positive.

7.2.2 Login Screen and Signup Screen

The login and signup screens were very quickly implemented. Parse has a special "User" object and contains premade methods for the creation and authentication of them. The two basic operations for login and registration both use Parse`s inbuilt methods. When a user logs in she is directed to the menu screen and when registering she is directed to the login screen. The use of a user registration separates our application from Kahoot! where you select a username for each game, but we consider it unlikely to become a source of confusion since user registration is common in most online applications and very intuitive. It is also important to have a set user account for each user to make friends lists and challenges to work.

7.2.3 Main Screen

The main screen consists of five buttons: New game, Challenges, Friends, Log out and Exit. The main screen does not have any complicated functionality and is mainly used to direct the user to other screens. To direct the user from one screen to another we use URLs. Each screen is a web view which has a corresponding HTML page from where we get the URL. The difference between logging out and exiting the application is that logging out directs the user to the login screen while exit closes the application but keeps the user logged in. That way the user can skip the login next time she opens the application.

7.2.4 Quiz Selector Screen

This screen contains a input field where users can enter the code for the quiz they want to participate in. This request is sent to the server which checks if a lobby for that quiz already exists or not. If the lobby does not exist, the server will create a lobby object containing the ID of the lobby and an array for players. Then it will add the user’s ID to the array and direct her to the lobby screen. If the lobby does exist, the server will add the user to the array of the corresponding lobby ID and direct her to the lobby view.

7.2.5 Lobby Screen

The lobby screen consists of a list of all the players currently in the lobby waiting for the game to start. Whenever the lobby has three or more players the ready button appears. Whenever a player joins a lobby a message is sent to all the other players in the lobby to make the client print that the player is connected. The same type of message is sent when a player clicks the ready button. If a player joins a lobby late and the lobby already has many players that have clicked ready, the client queries the database to get the list and status of the other players. This way the players will always get the correct print on their client, no matter when they join. When every player in the lobby have clicked the ready button a countdown will start followed by the first question. This is completely automated and differ from Kahoot! where you have an administrator that decides when the quiz starts.

Whenever a player joins a lobby she is subscribed to a channel for that lobby where all the communication is sent. The messages between clients are sent as JSON objects by ParsePush. When a client receives a pushed message it is handled by a custom parsereceiver that reads the type attribute of the JSON object, handles the rest of the data and calls functions accordingly. When a quiz is completed the users are unsubscribed from the channel to avoid users getting messages from other lobbies that might use the channel later on.

In addition to players there may be bots in game lobbies. These bots are based on previous players and behave just like a player would do. They join lobbies, click the ready button, answers questions and send emotes to the other players. Bots will be explained in depth in Section 7.2.12.

7.2.6 Quiz Screen

At the start of a quiz the client gets the question and possible alternatives from the database using a Parse query, and represents this to the user by modifying the HTML. During an active question a progress bar shows the players how much time they have left to answer. This is also used to calculate how many points the player gets from answering correctly. Staring at 1000 and moving towards zero when the time runs out and the players are directed to the score screen. Wrong answers gives no points.

When a player selects an answers a JSON object is sent over the channel to the other players which is handled by the parse receiver. The parse receiver then calls a function that handles the printing in HTML for the other players to notify them that someone answered. The client also sends a query to the database to add the score for the current question. The scores are calculated by the client and stored in arrays in the database. At the end of each question the scores are summed up.

In the prototype we created in the proof of concept, the client had to constantly pull data from the database to see if anyone had answered. This resulted in a lot of extra data sent back and forth between the clients and the server. In this version the communication is solved in a better way by the use of Parse push and saves the clients a lot of data transfer and computation.

7.2.7 Score Screen

The score screen shows the top five scores so far in the quiz, in addition to the score of the player herself. These scores are queried from the database and shown in HTML. The reason for only showing the top five players is that it can be discouraging to see, and let others see, your score if you are performing poorly. This view is presented by Tomas. W. Malone (1980) in his paper "What makes things fun to learn" and, according to him, this can demotivate players from participating in more games. There is also a simplified chat in every score screen, except the final score screen. In this chat the players can use predefined emotes to communicate with the other players. The choice between using predefined emotes and a chat where players can type what they want is based on simplicity. We did not want the time between question to be too long and typing messages on a phone in a short amount of time would make it feel hectic. The emotes sent between players in also handled by the parse receiver and added to the chat box of every client. We also chose to make the chat scroll slowly through the emotes player sends so the emotes does not disappear too fast if many players sends emotes at the same time.

7.2.8 Final Score Screen

The final score screen differs a bit from the other score screens. It does not have the chat where players can send emotes, because we do not feel there is any incentive for the players to stay in the final score screen for a long time. It does however have the challenge friend button where players can challenge their friends to play the same quiz as they just did.

An important part of the final score screen is the initiation of the database reset after a quiz is completed. After the scores have been queried from the database for the last score screen the lobbylist and readylist in the database is cleared to make the lobby ready for new players. The scores and behavior of the players that just finished the quiz are stored and marked as bots so that they can be used in new lobbies to play against new players.

7.2.9 Challenge Friends Screen

In the challenge friend screen the players can send quiz challenges to their friends by checking the boxes by their friends' name and clicking the challenge button. When the challenge button is clicked a challenge is stored in the database with the name of the challenger, the player being challenged and the quiz code.

7.2.10 Challenges Screen

In the challenge screen the player will get a list of all the challenges they have received. These challenges are queried from the database based on the username of the current player and shown in a list. Each challenge shows the username of the user that sent the challenge and the current user got the choice of accepting or declining the challenge. If a challenge is accepted the player is sent directly to a lobby for that specific challenge. If there is an existing lobby for that quiz, and it is not full, the player will be added to it automatically. If there is no existing lobby a new one will be created. If the challenge is declined the data is simply removed from the database and hidden from the challenge list. The challenge is also removed after it is accepted so it cannot be accepted more than once.

7.2.11 Friends Screen

The friends screen has two list, friend requests and friends, an input field and a button. The elements of the two lists are queried from the database based on the name of the user. The friends list is simply a list printed in HTML with the names of the users friends while the friend request list has some more functionality. Each friend request has an accept and decline button. When the user clicks the accept button the name of the friend is added to her friends list in the database and added to the list shown to the user. If the user clicks the decline button the request is removed from the list and from the database.

To add a friend and send a friend request the user is required to input the name of the person they want to add and click the add button. Then the request is saved in the database with the name of the sender and receiver.

7.2.12 AI Controlled Players

AI controlled players, or bots, are a key part in the time shifting of our application. We use two types of bots, traditional bots and input based bots.

Traditional bots are similar to bots seen in other games. They are controlled by an AI, and try to mimic the behavior of humans as best it can. In our application this is done by having them answer at random times regardless of question difficulty.

The input based bots are the ones creating the time-shifting in the application. These bots gets their behavior and score from previous users. Whenever a user, called A, is playing a game, the application will store how many points A gets at every question. Once all questions has been answered, the application will create a bot in the database with the scores for every question and the username of A. The emotes a user sends after a question is also stored in the database. When the next player, B, plays the same quiz, and there is not enough human players, the application will load the bot based on A into the game. When accepting a challenge from a friend the player will always play against the bot made from their friend's score and behavior.

Given the points earned by A at every question, the application can calculate how fast A answered and present the "A has answered" message at the appropriate time, creating the illusion for B that they are playing the game simultaneously.

In our application the priority when entering a lobby is: human players, then input based bots and lastly traditional bots. The consequence of this is that you will only play against traditional bots if there are not enough human players and input based bots to fill the lobby. In other words, you get to play against either a real players or a time-shifted players.

7.3 Issues

Like most other development projects, several issues presented themselves during development, and in this chapter we will look at the most influential ones and how we solved or worked around them.

7.3.1 Push Notification

Our application is a quiz application where you compete against other players that are either real time or time-shifted. The players should be notified when another player answered, and to do this we needed communication between the clients. The problem we ran into with this was that Parse, our database/server platform, did not provide the possibility for clients to do push notifications with JavaScript. In other words, there were no way for a client to tell the server or the other clients that they had answered the question.

We were aware of this limitation before we started, as we ran into it when developing our Proof of Concept. Our solution was to use Parses Android version, which did provide client pushes to handle the communication between clients, and use web technologies for everything else. While this efficiently solved the communication problem, it added some increased complexity to the architecture which is discussed in the next chapter.

7.3.2 Complex Architecture

Because of the limitation to the JavaScript version of the Parse framework discussed in 7.3.1, we had to use a more complex architecture when developing the application. Because our application was to be created using web technologies, but the communication had to be done through Android, we had to create a way for our applications web technologies to communicate with the Java based Android code. In Android, this is done by creating a JavaScript interface class which allows JavaScript to call methods in Java and vice versa.

We quickly discovered that our initial idea of using Java only as a way to communicate with other clients and all other logic in JavaScript, was unnecessarily difficult as it required us to pass data through several layers. To combat this, we changed our architecture and moved most of the logic back to the Java layer, and used JavaScript for handling the user interface.

7.3.3 Parse Channels

This issue requires some insight into how we handle communication in our application. The first paragraph here will provide that insight and the second one will explain the actual issue and our solution to it.

The communication between clients in our application is done through channels provided by Parse. The way this works is you can send messages to a channel and everyone in that channel receives this message. When a player enters a lobby, they are automatically subscribed to that lobby`s channel and when the quiz is finished they are unsubscribed from it.

In the Parse database there is an installation object for every device that has installed the application, and this object contains a list of all the channels that device is subscribed to. These objects in the Parse database are mutable, so with access to the project you can enter and change every field of every class including the ones in the installation class.

The issue for us was that whenever the application would crash before the quiz was finished, the device would not be unsubscribed from the channel of the quiz. This causes some problems since you will receive messages from that quiz even if you are participating as long as you are in the channel. During development, our solution to this was to simply enter the installation objects of the crashed devices and manually delete the channels. What we didn't initially know, was that the Parse channels are stored differently than everything else in the database and while deleting the channels would cause them to appear to be deleted when looking at the databases user interface, the device would still be in the channel.

The Parse documentation did not provide any explanation for this behavior and where exactly the channels are saved remains a mystery to us, but we speculate that it is handled in the Android gradle build somewhere. Parse doesn`t provide any functionality for unsubscribing from every channel either, so the solution we created to deal with this issue was to create a method that would attempt to unsubscribe from all the channels we had created one after another.

We created a button simply called "unsubscribe" which would run this unsubscribe functionality and in case of a crash during a quiz every participant in that quiz would have to click that button before they could join another quiz.

8 Testing the Application

With a proof of concept already in place from an earlier project, our main goal was to test how time-shifting affects the user experience, and to what extent players notice or suspect that opponents are time-shifted and not playing at the same time.

Section 8.1 gives an overview of the testing process and the testing environment, Section 8.2 takes a look at the actual tests and in chapter 8.3 we will look at lessons learned and what we could have done differently when testing.

8.1 Test Introduction

In this chapter we will present an overview of the entire test process. Section 8.1.1 will present how the participants were selected, Section 8.1.2 takes a closer look at how we planned the execution of the test and our reasoning for the choices made and in Section 8.1.3 we present the questionnaire.

8.1.1 Participant Selection

Initially, we wanted to use participants from the course TDT4240 Software Architecture. Since some of the questions we want answered are about the possibility of using time-shifting in education, having participants that are currently in education was preferable since it would be easier for them to have an opinion about its viability. Since our supervisor is responsible for the course TDT4240 Software Architecture we would have access to exam relevant questions to use in the quiz. One last consideration for us in selecting participants was that we wanted to have tech-savvy students to test it on since installing applications to android phones from outside the Play-store requires you to turn off some safety features. We figured that students who understood why this is necessary, and are capable of doing it themselves, would simplify the testing process.

Unfortunately, we underestimated how difficult it would be to get participants once the exam period had started. We wrongfully thought that exam relevant questions would be enough to attract some of the courses students, and in the end we only got two participants from the course. Because of this we had to find more participants elsewhere. We still wanted the participants to currently be students or just finished with their studies and ideally quite tech-savvy. We broadened our search to include IT-students outside of the course and through some contacts at

UiO (University in Oslo), we managed to attract some more IT-students that met our requirements.

8.1.2 Test Planning

For practical reasons, we decided to reserve a room at our university and do the testing there.

Our reasoning for this was that some of the participants might require help installing the application or with other problems, and this in addition to explaining the testing procedure would be easier with all the participants in one room.

While discussing this approach, a possibly big problem presented itself in that the participants might use information outside of the test to decide whether or not an opponent was time-shifted or not. Since the participants were all from the same course, there was a good chance that they would know each other's names, and possibly their gaming nicknames as well. Using this information they could compare the participants they saw with their names or nicknames, and use this to decide which of the in game players were time-shifted.

Our solution to this was to create accounts for the participants to use with set names. The in game names of these accounts where those of well known cartoon figures. This was done to make it easier to remember and distinguish between players for the participants than generic names like Player1, Player2 etc. would be.

One last problem was that the players would still be able to see when one of the other participants touched their phones to answer a question or stopped playing, and using this information combined with that they were still in a quiz, they would be able to see how many of the participants were in the same quiz as them. To avoid this, we placed the participants in a circle facing outwards, as shown in Figure 8. 1, so that they could not see each other without making an effort, and told them to try not to look at their fellow players.

After finishing the quiz, the players were given the questionnaire presented in Chapter 8.1.3 which was answered using their mobile devices. The last question of the questionnaire is whether or not they are willing to answer follow up questions.

We did this to have the option of contacting the participants at a later stage if we discovered something new during the test that we wanted to pursue.

Test Structure

1. Here we will give a quick summary of how the test is to be performed.
2. Participants in the course TDT4240 Software Architecture are given the opportunity to participate in the test.
3. Selected participants are told the time and place that the test will take place.
4. The goals and setup of the test is explained.
5. Participants download the application onto their own devices.
6. All participants are placed in a circle facing outwards, as shown in Figure 8. 1: Illustration of how test participants were seated., and given their user name and the name of the quiz to join.
7. Participants finish the quiz, and answers the questionnaire.
8. The quiz is over and participants stating they are willing to answer follow up questions will be contacted when these questions are ready.

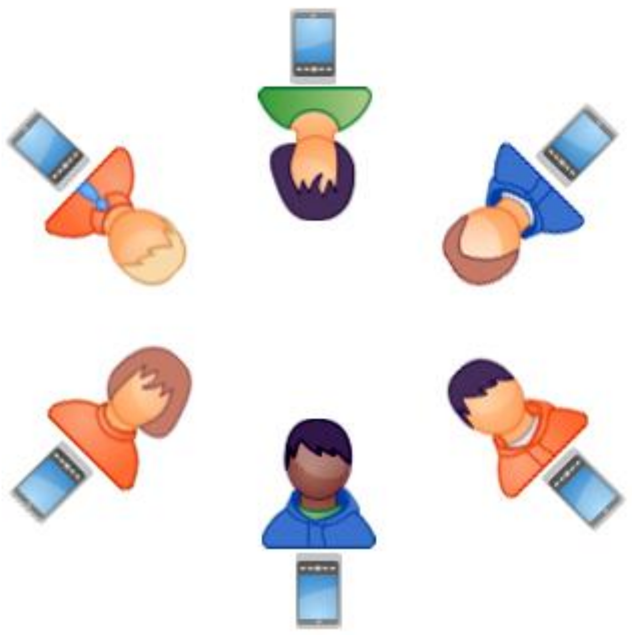


Figure 8. 1: Illustration of how test participants were seated.

8.1.3 Questionnaire

The questionnaire can be found in Appendix C - Questionnaire, here we will look at the structure of it, and the purpose of the questions. We divided it into four parts: basic information, time-shifting, educational and the System Usability Score.

Basic information

The first part of the questionnaire are basic questions about the user and their mobile gaming habits. In addition to this we ask what kind of connectivity they used during the test. Bad connections can hurt the user experience and separating poor reviews due to the application and connectivity is important. This will also give us feedback about how good your connection has to be to get a good experience with the game.

Time-shifting

The second part of the questionnaire contains questions related to time-shifting. The first part lists the usernames of all the players that played the quiz and the tester is asked to separate them into AIs, Humans or Uncertain. Comparing their answers with who the actual AIs where, gives us some insight into how well players notice a difference between players. We continued to ask questions about how easy distinguishing between them were, and how they did it. This part finishes by asking several questions relating to time-shifting. Both how it affects the user experience and how the problems we try to solve with time-shifting affects their gaming experience. The last question of this part is an open question where they can add any ideas they have on how to improve time-shifting.

Educational

We were interested in if time-shifting could be used to improve education and added several questions to the questionnaire regarding the potential of time-shifting as an educational tool.

The first question we ask regarding education is whether or not the participants are experienced with other educational games. We consider this to be an important question as their experience with other games might give them a better understanding of how it would work in practice. It also gives them something to compare our time-shifted solution to the other possibilities out there.

Our next question is whether they would like time-shifted games to be part of their everyday school life. This is here to give us an overview of if time-shifting is something students are interested in, and if they think that it could improve their learning experience. We consider time-shiftings place in education to be that of an motivational tool, so knowing if potential users are at all interested is important .

The next question is whether or not they would like to use time-shifting as part of their studying for exams and tests. When we first started working with time-shifting and education, this was the situation we envisioned it would be useful in. We thought that having the possibility to challenge and play against your friends with questions for an upcoming test would be a good way to make the studying more fun. This question is here to see if the participants share in our view that this could be a good supplement to traditional studying.

The last question about education in the questionnaire asks the participants if they would want time-shifted games to be used for homework. Since the technology used to create the time-shifted behavior already requires us to save the answers from players, the possibilities are definitely there that it can be implemented into automatic homework assignment where the lecturer creates the questions and students need to get a certain amount of questions right for the homework assignment to be approved.

System Usability Score

The System Usability Score is a standard created to give a simple way to measure usability of a system. For us, the SUS is not very important since the application is never supposed to be released and only used to give us feedback about the concept of time-shifting in games and education. The reason we have included the SUS in our questionnaire is to reduce possible noise in our results. If our application is really poorly made, that could influence how the participants answer the, to us, more important questions about time-shifting and education. By introducing the SUS into our work we have a way to tackle possible questions about whether or not the quality of the application influences the participants attitude towards time-shifting as a concept.

8.2 Test Execution

In this chapter we will look at the execution of all the tests we did. Section 8.2.1 explains the pre-test and the reasoning behind conducting it. In Section 8.2.2 we explain the different phases of each test and what they consisted of. For descriptions of the individual test see Appendix B.

8.2.1 Pre-test

Before the actual test was done, we wanted to have a pre-test to resolve possible issues that could present themselves during the test, and to give us some insight into how the testing process would play out. Our main concern was that during development when we switched from emulated devices to using our own phones, one of the devices did not support the method we used for navigating between screens, while the other had no issues with it. While this issue was resolved, we needed to make sure that other similar issues that only appear on certain devices would not happen during our actual test.

Another reason for the pre-test was that we wanted to test the questionnaire and make sure that the questions where interpreted the intended way, and get feedback about possible ambiguous questions.

We used some of our friends as participants in the pre-test. Possible bias was a concern for us when testing on our friends as they might unconsciously give us more favorable answers than participants with no connection to us. This is mainly a concern for the SUS part of the test since it measures a subjective impression of the application. In the questions related to whether or not they can distinguish between time shifted and real-time players, we do not think bias is an issue as their answers are either right or wrong without any personal influence.

We conducted this test as described in Section 8.1.2. During the quiz one of the participants failed to answer a question which caused a crash in the other applications as they would query the database for a score that did not exist. This happened after the second question and we restarted the quiz after noting the issue and telling the participants that they should just guess if they did not know the correct answer. Other than this, the quiz ran smoothly and the questionnaire was answered with no apparent confusion about the questions. Since the participants understood the questionnaire and the application worked on all the devices, we were ready to proceed with the actual tests.

8.2.2 Tests

Our tests were performed at three different locations in addition to one test we performed online were the participants would play from their own home. The three locations were two different group rooms at the NTNU campus and at the University at Oslo.

The reason why we had two different group rooms at NTNU was that during our first testing session we ran into an issue with the communication between the clients caused by an unstable Wi-Fi connection at that location, and to avoid further issues we reserved another room for the next testing session.

Introduction Phase

The testing was conducted as presented in Section 8.1.2 with participants sitting in a circle facing away from each other to prevent them from using visual cues to help determine who the real-time players were. We wanted to make sure that the participants could not determine the number of real-time players by counting the number of persons in the room with them. Our initial plan was that we would run two different quizzes at the same time, and distribute the participants between them. Unfortunately, the number of students signing up for any of our tests were not high enough for this to be a good option. While it would be still be doable, the ratio of time-shifted to real-time players would be very high and we opted to instead just run one quiz with all the participants. To recreate the situation where the players were uncertain of the number of real-time players, we told the participants that either, one or both of us would be participating as well. For the online test, we didn`t reveal any information about how many of their opponents would be real-time and time-shifted.

Before the quiz started we explained time-shifting as a concept, but not our method of creating it, to the participants. While some of them seemed surprised that it was possible, they all quickly understood the concept.

Once everyone was familiar with what time-shifting was and how the test would be conducted, we started with the next phase of the test.

Installation Phase

The application was uploaded to Dropbox and we had prepared a QR-code to it in addition to the possibility of typing the URL. Using Dropbox to distribute the application did in retrospect prove to be a bad idea as it caused us several issues which will be presented in Section 8.3.5. Our backup solution was to transfer the application directly from a computer over cable, and we did this for the participant phones that had issues with Dropbox.

The online test was conducted after the others, and to avoid issues with Dropbox we sent the application over Skype. This proved to be much easier and the online participants had no issues installing.

Quiz Phase

After the participants had installed the application we started the quiz. Being a prototype, the application was not completely stable and would on some occasions crash. The participants were made aware of this before the quiz started and in the case that a crash happened, we only restart if we felt that they had not been given enough time to get a good impression of how time shifting works. If we did restart the participants would have to play though all the questions before the crash again, which would be boring. Since the questions in the quiz had no significance to us, this would not influence the results in any way as long as enough questions were played for them to get an impression of time-shifting in practice.

We experienced two crashes during the tests, all of them related to the transition between the question screen and the score screen. The cause of the crash was a null pointer if a player would reach the score screen before all the other participants had answered. We created a quick fix to the issue as soon as we became aware of it. The crashes happened after the second and 8th question. We restarted the quiz when it crashed after the second question as the participants had only been in the score screen once. In the other crash we felt that 8 out of 10 questions would be enough, and ended the quiz after the crash.

The quick fix created another opportunity for the participants to separate between time-shifted and real-time players, as the time-shifted players who had answered incorrectly to a question would send a notification the split second before the switch from question screen to score screen. One of the participants in the online test noticed this, but were not able to use it to correctly separate the real-time and time-shifted players.

After the quiz was finished the participants were given the questionnaire

Questionnaire phase.

The participants did for the most part answer the questionnaire without any help and the results are presented in Section 9.1. Their only issue was in the SUS part as several of the participants were uncertain of the meaning of the word "cumbersome". This was something we were aware of from the pre-test, but since the SUS questions are standardized, we did not feel at liberty to change the wording of the question.

After the participants had answered the questionnaire they were free to go. The last question in the questionnaire asked them to leave their email if they would be available in case we wanted to ask them some follow up questions. While several of them did leave their emails, we found that our results from the tests were enough to answer our research questions and opted not to use this option.

8.3 Lessons Learned

In this chapter we will look back at the tests conducted and discuss what we did right and what we could have done differently.

8.3.1 Getting Test Participants

The biggest problem for us was to attract participants to test our application on. Our initial plan was to only test on students from the Software Architecture course, but we finished our application a few days after the last lecture of the course. We thought that using questions relevant for the exam would be enough to attract participants, but since the exam period had already started this proved more difficult than expected.

We think that the biggest problem was that it was difficult to reach the participants once the lectures were over, and a big part of it was that we had no way of contacting the participants other than through the university's intranet. Once the exam period starts, there is less reason for the students to check messages delivered there. If we had the application ready to test before all the lectures were over, it would most likely be easier to get in contact with and get them to participate.

8.3.2 Behavioral Assumptions

In the pre-test, we discovered a bug that would crash the game if one of the participants had not answered before the timer on the question ran out. Up to this point we had not noticed the bug as we had always answered in time during testing. After discussing it we opted not to fix it as we didn't think it would happen again. The application gave no penalty for answering incorrectly and we thought that if we made sure to explain this in the other tests, everyone would just pick an answer randomly if they did not know what the correct one was. This turned out to be a false assumption as other participants also opted not to answer if they did not know.

The lesson learned from this is that you should avoid making assumptions about how participants are going to interpret and react to information.

8.3.3 Testing at Site

During the first test, we had some starting issues that we had not run into before when doing a last run-through before the start of the test. Apparently the university's internet connection in that room was not completely stable, and this caused some trouble with the timing with messages between the clients. This was discovered in the game lobby of the application and was fixed by using mobile internet instead of the Wi-Fi provided by the university. This problem came before starting the quiz, but it served as a reminder that you should, if possible, test your application in the exact environment that you want your user testing to take place.

8.3.4 Simple Language

Another important lesson learned during our testing is the importance of using simple and understandable terms in the questionnaire. This is especially important when the questions are not in the participant's native language. While we were aware of this and tried to make our questions as simple as possible without losing precision, the SUS questions caused some confusion among some of the participants. Several of the participants asked what the word "cumbersome" meant, and looking at the results, the question containing the word had a high number of "neutral" answers compared to the rest of the questions. While it is definitely possible that the question was correctly understood, it could indicate that some participants didn`t fully understand the question and went for a "safe" answer.

8.3.5 Distribution

During our tests we had some problems distributing the application to the participants. Because of the diversity of Android devices even popular software like Dropbox have issues. In our case we ran into an issue where a certain type of phone, Sony Experia S3 Compact, were getting a 403-Access denied error [43]. Since you cannot predict what kind of issues you might run into you should have a backup plan for distribution.

Part IV

Evaluation

9 Evaluation of the Results

In this chapter we will look at the results from the tests and evaluate them. In Section 9.1 we will analyze the results from questionnaire. In Section 9.2 we will look at the System Usability Scale for the application. In Section 9.3 we will go through the functional requirements and explain how we fulfilled them. In Section 9.4 we do the same for the non-functional requirements. In the last section, Section 9.5, we will present the GameFlow analysis of the application.

9.1 Questionnaire Evaluation

In this section the answers from the questionnaire are analyzed. In Section 9.1.1 contains general information about the participants. Section 9.1.2 we cover the questions concerned with time-shifting. In Section 9.1.3 we look at the extent which the participants were able to distinguish time-shifted and real-time players. In the last section, Section 9.1.4, we analyze the questions about the use of time-shifting in education.

9.1.1 General Information

At the start of the survey we asked the testers to answer some general questions to give us an overview of who they are, the connectivity they used during the test and what kind of mobile game habits they have. We got the following results:

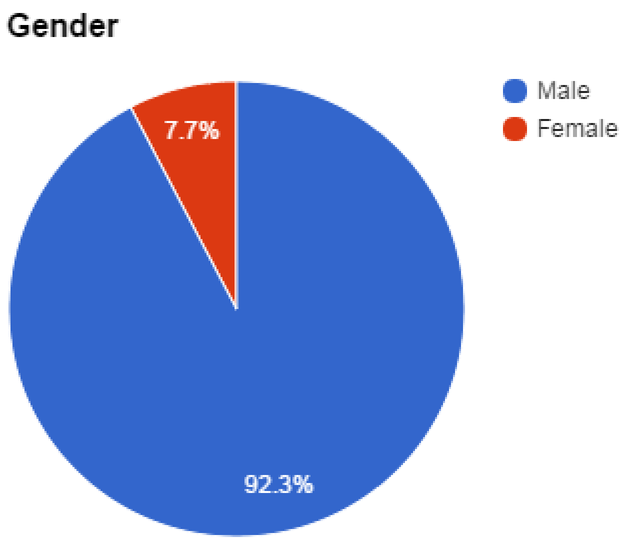


Figure 9. 1: Gender distribution of testers.

The gender distribution is more or less like expected amongst the computer science students as shown in Figure 9. 1. It would however be interesting to test the application on more females to see if there is a difference in how male and female testers perceive the application.

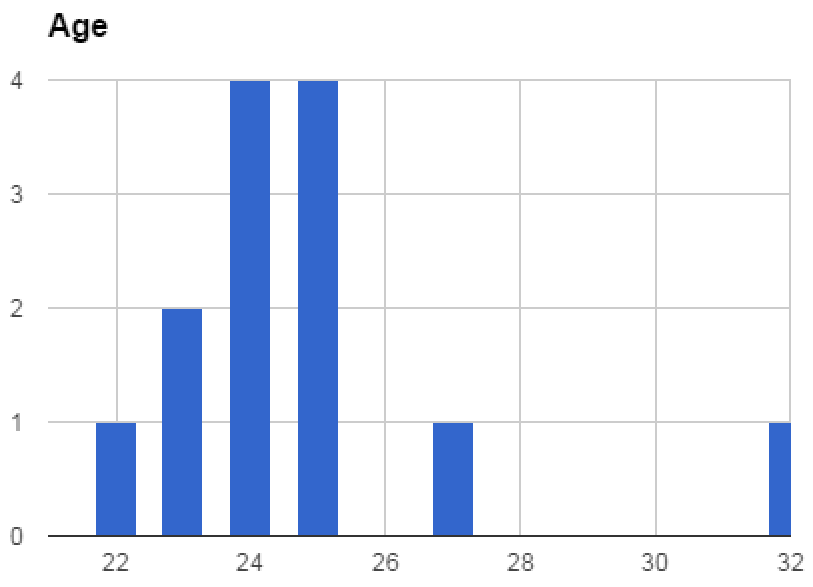


Figure 9. 2: Age distribution of testers.

The age distribution of the testers is as expected amongst university students, with one exception as shown in Figure 9. 2. Since we only tested the application on university students, more research is required to see if these results holds for other demographics.

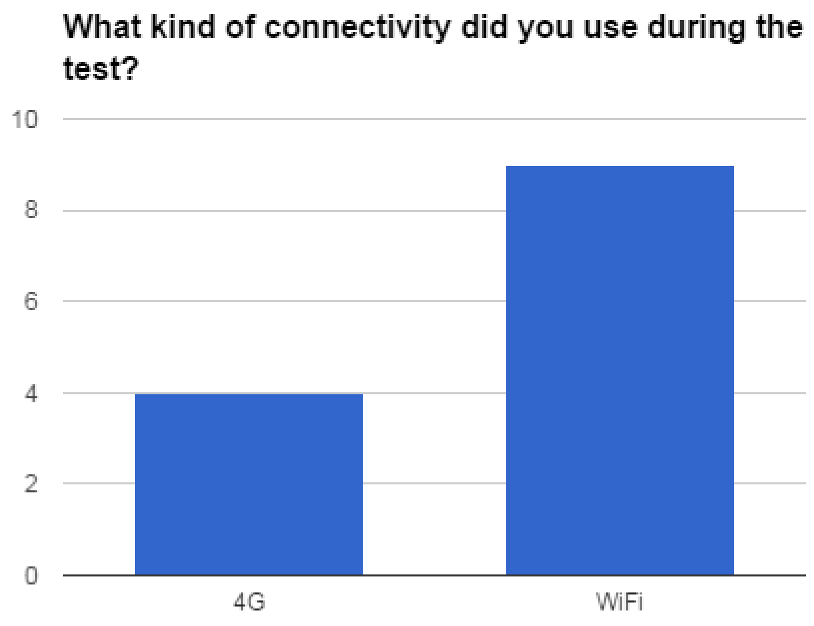


Figure 9. 3: Connectivity during testing.

Before starting testing the application we were a bit worried about how different connections would impact the testing. We were certain that using Wi-Fi would be more than good enough, but it could be troublesome if someone would lose connection during a quiz and fall out of the lobby. We were also worried that using 3G/4G could be unstable and create problems. Luckily no one lost connection and it did not impact the tests. All the participants used fast connections as shown in Figure 9. 3.

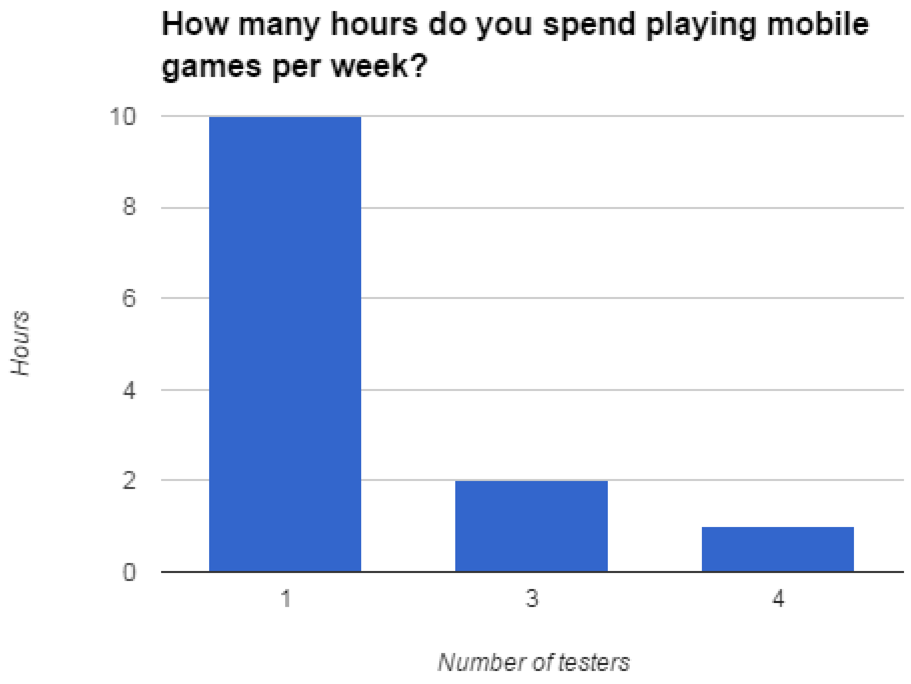


Figure 9. 4: Mobile game habits.

The amount of hours the testers use on mobile games is lower than we hoped for, with the exception of one very active player as shown in Figure 9. 4. This means that they might not have that much experience with mobile games to compare our application to. Testing on more active mobile gamers might give other results as they might be more interested in using mobile games for education.

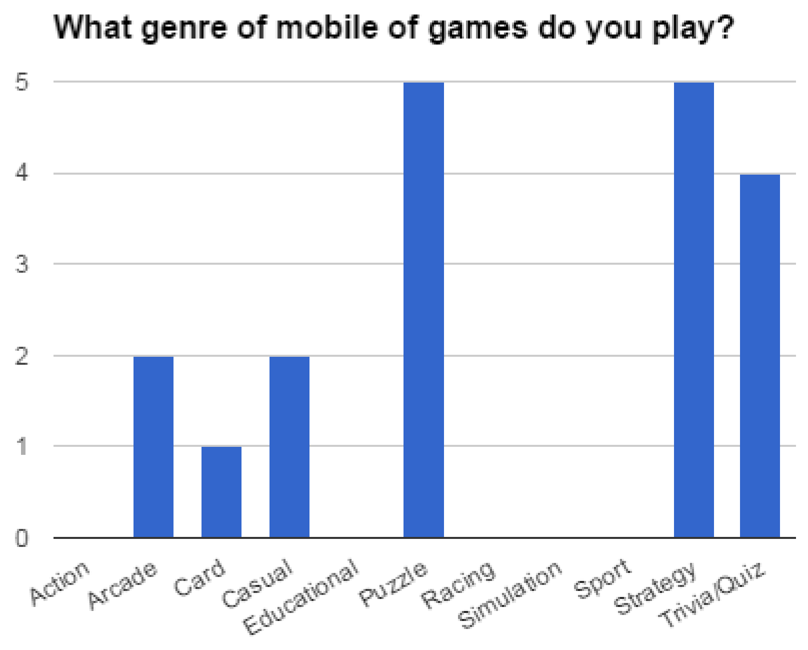


Figure 9. 5: Mobile game genres.

There was a good diversity of genre of mobile games the testers play, but as shown in the previous figure they are not very active players. However, Trivia/Quiz games is one of the most played genres amongst the participants, as shown in Figure 9. 5, which means they have some experience with applications similar to ours.

9.1.2 Time-shifting

The second part of the questionnaire was focused on time-shifting as a concept. In this part we hope to discover how willing the participants are to use time-shifted games and if there are some surprising trends.

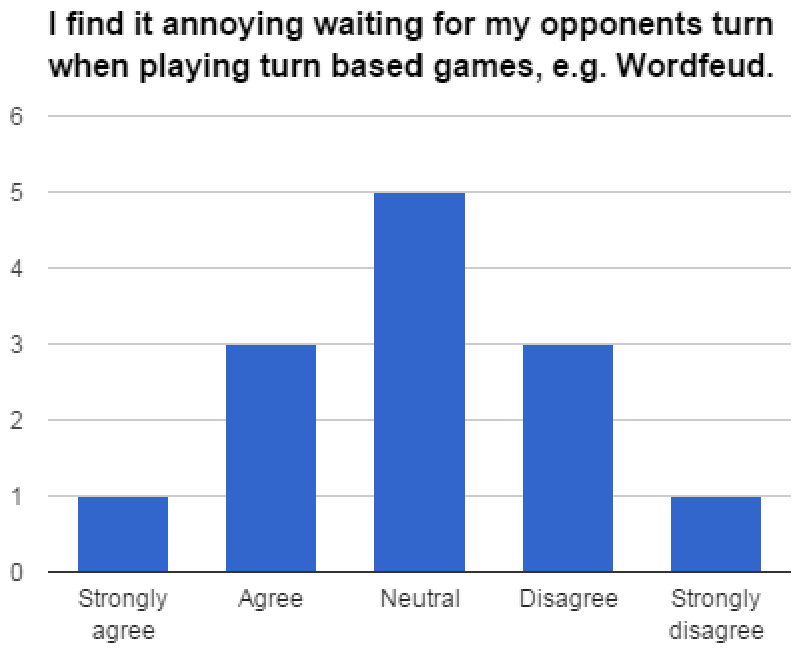


Figure 9. 6: Annoyance of waiting for opponents.

We asked the testers if they find it annoying waiting for opponents turn when playing turn based games. The results were overall neutral with some positive and some negative answers as shown in Figure 9. 6. It is difficult to draw any conclusions from these results since the answers differ, but it is clear that the participants do not agree on the topic. What we can read from the figure is that there are some people that think it is annoying to wait for their opponent and therefore time-shifting might be a good option for them.

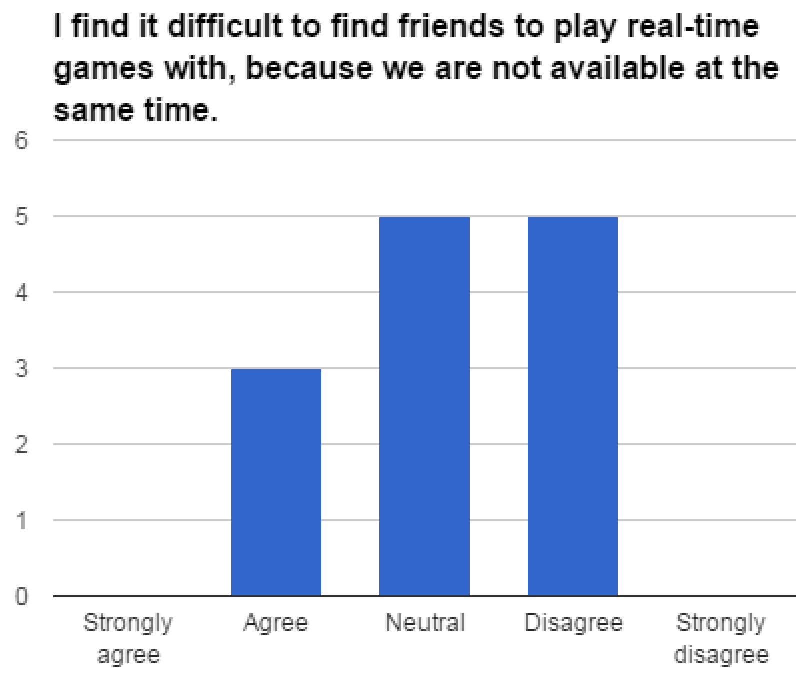


Figure 9. 7. Difficulty of finding opponents for real-time games.

When asked if they found it difficult to find friends to play real-time games with, no strong answers presented themselves, but the participants seemed to be leaning towards disagreeing as shown in Figure 9. 7. However, the trend is not strong and it is difficult to draw any conclusions from this question.

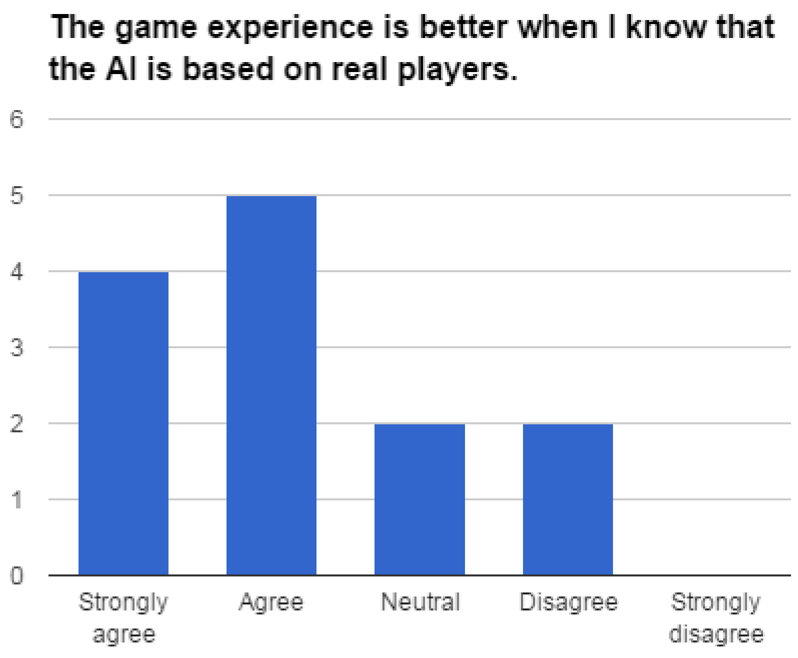


Figure 9. 8: Game experience with AI based on real players.

We wanted to see how the single player experience changed for the testers when they knew the AI was based on real players instead of non-player AI. It turns out that the majority is clearly positive to an AI based on other players and they do feel like the game experience is better when solved this way, as shown in Figure 9. 8. This was an expected result since AIs in games have gotten a reputation of being stupid and predictable. By making the AI based on other players they get a more humanlike behavior. This means that the AI will be less predictable and not do irrational choices like AIs in a lot of games does today. The benefit of this will be even clearer in games that are more complex and requires a more complex AI.

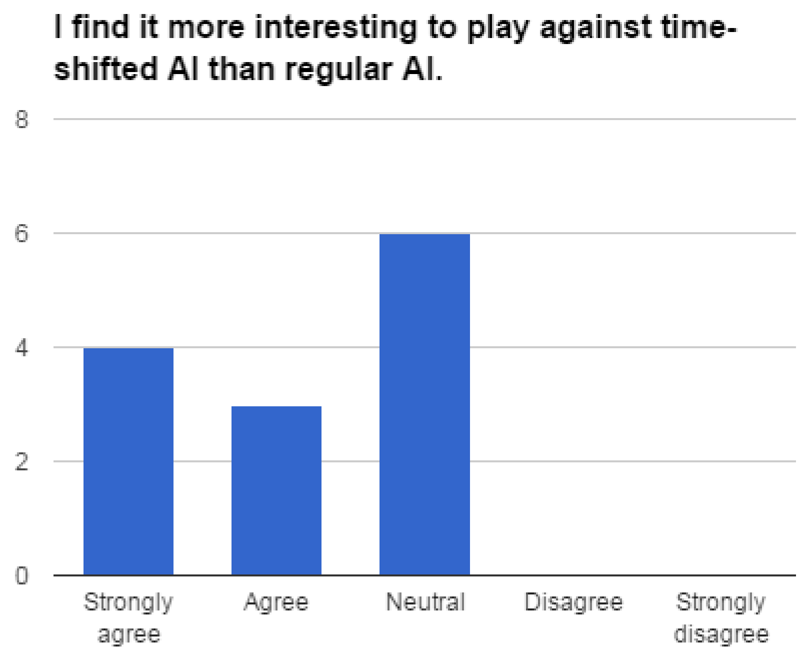


Figure 9. 9: Question regarding time-shifted AI versus regular AI.

We also asked if the players found it more interesting to play against time-shifted AI than regular AI. The results were overly positive here as well which supports the result from Figure 9. 8. There were some testers that answered that they were neutral to the question, as shown in Figure 9. 9: Question regarding time-shifted AI versus regular AI. . The reason for this might be that the game we developed has somewhat of a simple AI which might make the value of a time-shifted AI less than it would have been in a more complex game. The most important part is that people are positive to time-shifted AI in single player games and that it seems like there is a bright future for the technology.

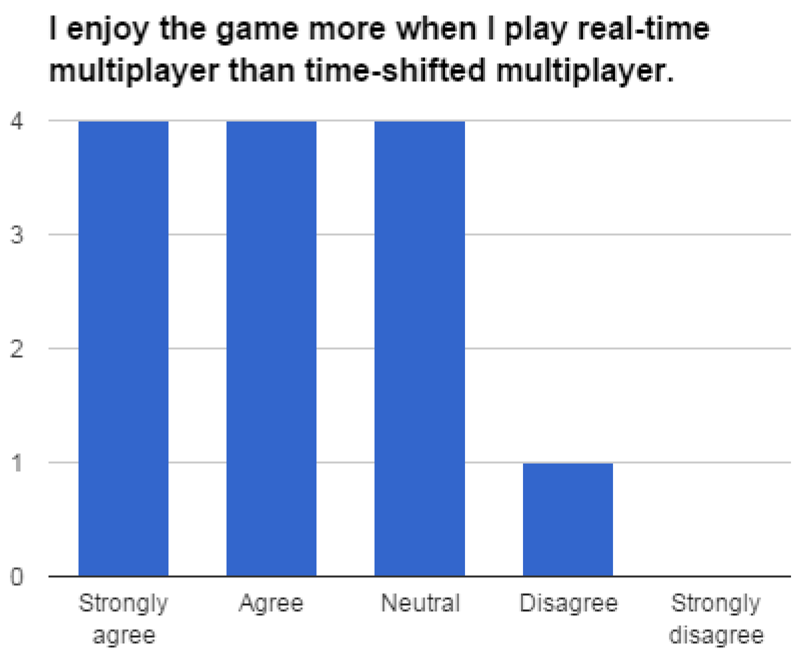


Figure 9. 10: Question regarding enjoyment of multiplayer games.

We asked the participants to compare the enjoyment when playing real-time and time-shifted multiplayer. As we can see in Figure 9. 10, the overall trend is that players prefer real-time to time-shifted with more than half of the participants answering "strongly agree" or "agree". One of the participants answered that he disagreed with the statement, indicating that he might prefer to play against time-shifted opponents. While it is possible that this is the case, this surprising answer might also be caused by the wording of the question. He might have no preferences and thus choosing to disagree with the statement that he enjoys real-time more than time-shifted.

Overall we see that the participants feel that there is a difference. This means that time-shifting should not be considered as a replacement for real-time, but rather as an option when real-time is not possible.

9.1.3 Distinguishing Between Real-time and Time-shifted Opponents

One of the most important parts of making the combination of time-shifted players and real players work is how easy they are to distinguish. Ideally we wanted our time-shifted players to be indistinguishable from real time players and we were really anxious to see how well the testers could separate time-shifted players from real players. We wanted to have an equal number of time-shifted and real time players in each lobby while testing, but due to the lack of motivation for people to attend the tests this did not happen. Therefore, during testing each lobby had a majority of time-shifted players compared to real time players, which in theory would make it easier to pick out the possible time-shifted players from the real time players.

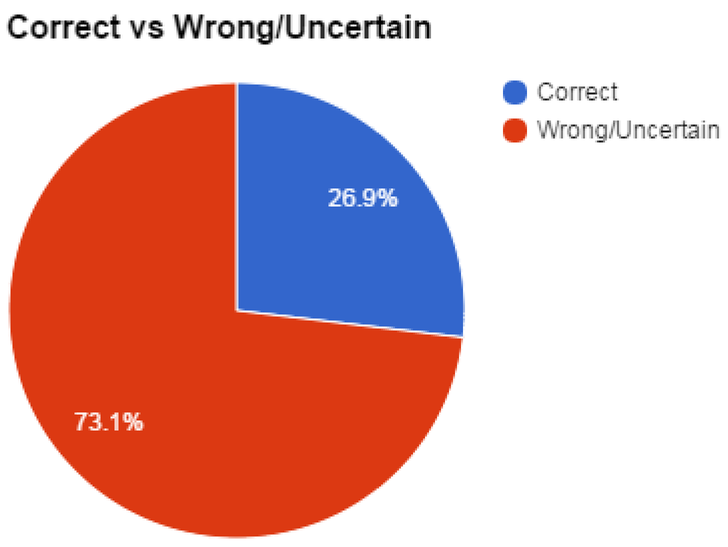


Figure 9. 11: Accuracy of opponent classification.

The results however were a bit surprising. When asked to classify their opponents after a test only 26.9% of the classifications were correct, as shown in Figure 9. 11. On the other hand, 73.1% of the time the testers classified an opponent wrongly or answered that they were uncertain on whether or not an opponent were time-shifted or playing in real time. Even though the testers knew that most of their opponents were time-shifted they still guessed wrong or were uncertain in close to three out of four cases. These results show that it really is hard to differentiate between players playing in real time and those that are time-shifted.

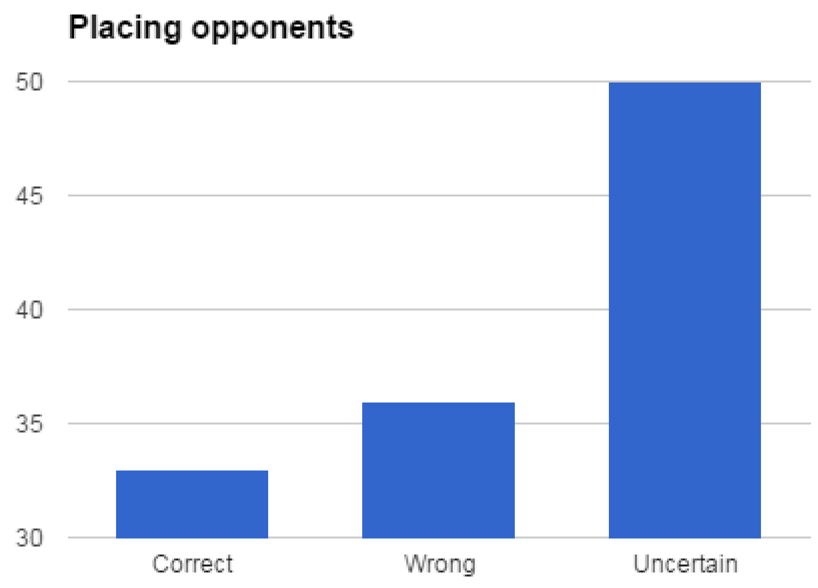


Figure 9. 12: Placing of opponents.

In Figure 9. 12 we have further divided the testers classifications into three parts: correct, wrong and uncertain. In 33 of the cases they guessed correct, in 36 of the cases they guessed wrong and in 50 of the cases they were uncertain. From this chart we can see that there is not that much of a difference in correct and wrong classifications, but there is a majority of uncertain classifications.

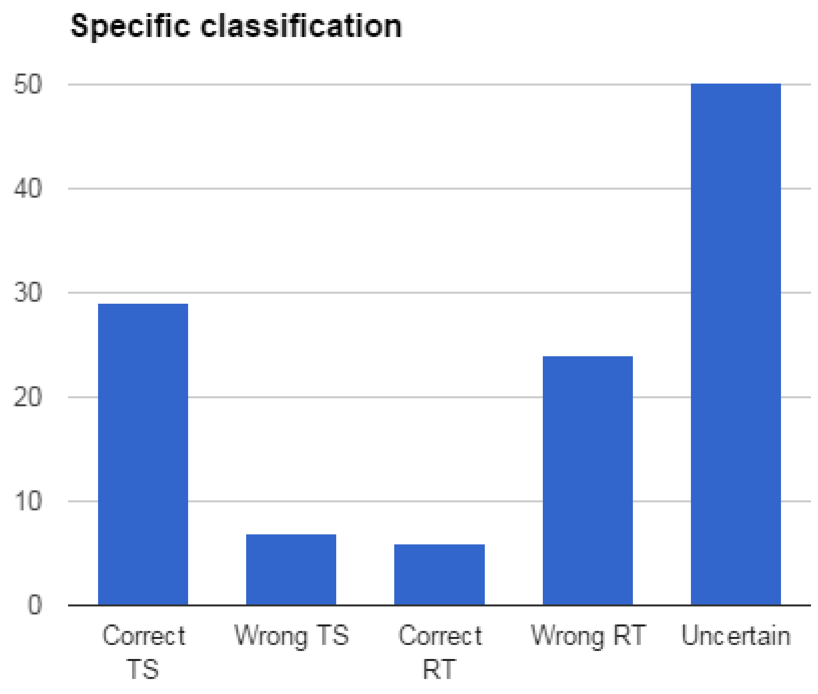


Figure 9. 13: Specific classification of opponents.

To go even further into the depth of the classifications we have separated each possible result in Figure 9. 13. This chart shows that the testers often guess correctly when classifying a time-shifted player (Correct TS), but often wrong when classifying a real time player (Wrong RT). As mentioned earlier we expected it to be easier to classify time-shifted players since there was a majority of them in each test. We did, however, not know how difficult it would be to classify who was playing in real time. It turned out that it was really difficult and that most of the classifications of real time players were wrong. This might suggest that our approach to time-shifting does work and it does make it hard to distinguish between time-shifted players and real time players.

In the previous graphs we saw that players had problems differentiating between time-shifted and real-time players. However, they did correctly classify an opponent as time-shifted most of the time. When asked how easy it was to distinguish between time-shifted players and real-time players we got the results shown in Figure 9. 14.

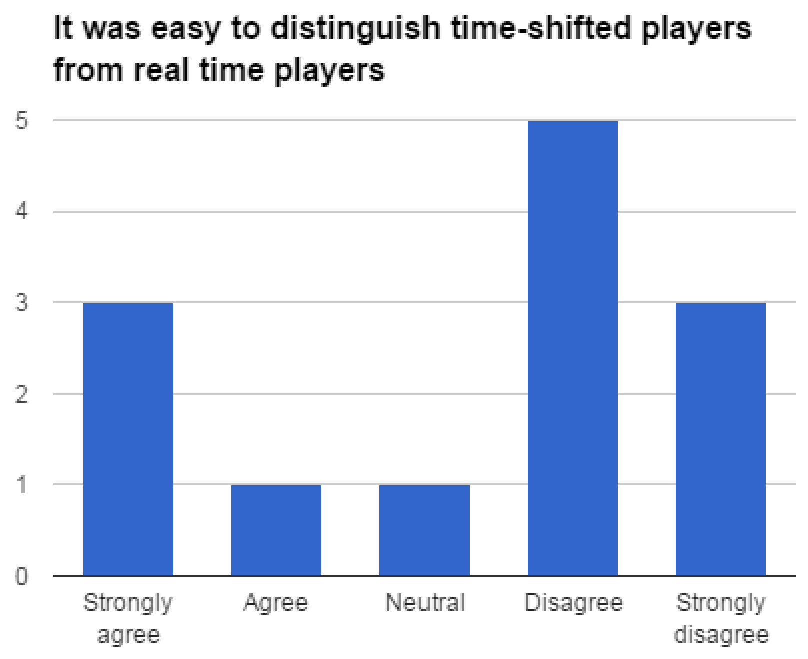


Figure 9. 14: The ease of distinguishing players according to the players themselves.

There was a clear majority of testers that said it was difficult to distinguish between players. This supports the results from the previous charts and explains why the testers were so uncertain when asked to classify their opponents. There was also a few that disagreed with the majority and said they thought it was easy to distinguish between players.

The testers were also asked to explain how they distinguished between players. Some commented that they thought players were time-shifted because they used a lot of emotes and that the emotes felt 'random'. The truth is that the timing of the emotes from time-shifted players are displayed at a random time, but which emote is displayed is not random. The emote displayed is the one the player used when she played and it is 'replayed' when used by a time-shifted player. If an emote is displayed right after the players are sent to a score screen and you need godlike reflexes to send an emote that quickly it will look a bit random. To negate this we added a fixed time to the random time of sending emotes so this would not happen.

Another possibility for the emotes feeling random is because of players themselves sometimes just send a random emote for fun. There is no stopping a player from sending an emotes saying 'I'm the best' when she is in fact outside the top 5. This might make the other players think that she is time-shifted because of the randomness of the emote.

Others commented that they thought the players that clicked ready quickly in the lobby were time-shifted. We also tried to negate this by adding a set timer to the random time of a time-shifted player joining and clicking ready. We feel that the set time added to the sending of emotes and time-shifted players joining and clicking ready is long enough to not give away anything. The results of how often testers classified players correctly also supports this claim by showing that testers most likely are wrong when classifying a player. Some players also commented that it was difficult to classify players and that they just guessed or chose uncertain.

9.1.4 Education

Since we are especially interested in the potential of time-shifting in education, we had several questions related to this in our questionnaire.

Our first question related to education was whether they were experienced with educational games.

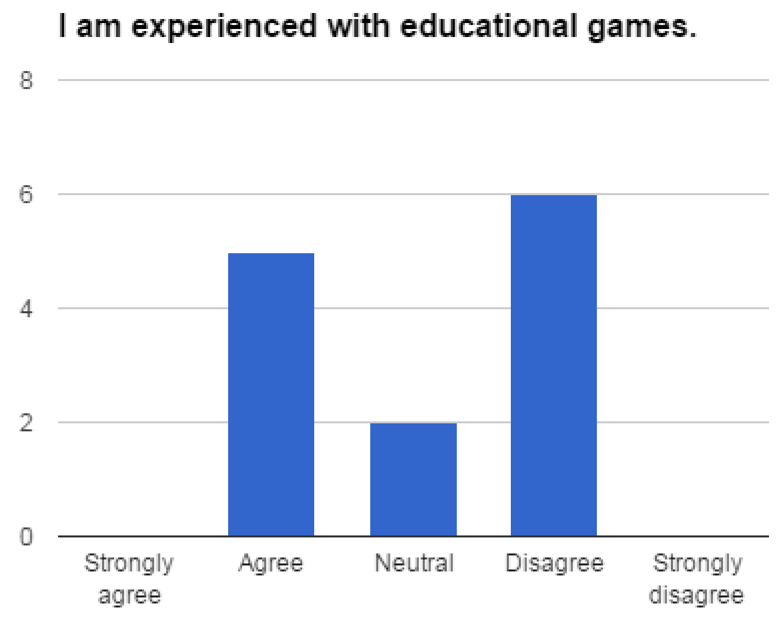


Figure 9. 15: Experience with educational games.

As we can see in Figure 9. 15, all the answers lie around the center with no one strongly agreeing or disagreeing. This implies that everyone has some experience with educational games, but it's also safe to assume that it has not been used extensively through their education.

The importance of this result is that it shows that participants have some understanding of how educational games can be used. This is important since it gives them something to compare our application to and that the concept of games in education is not foreign to them

In our next question we wanted to know what the participants felt about using time-shifting as part of their everyday school life. When we created this question we thought of "everyday school life" as in lectures and other school work, but the term might have been to diffuse and it's difficult to know if the participants interpreted the question the way we intended.

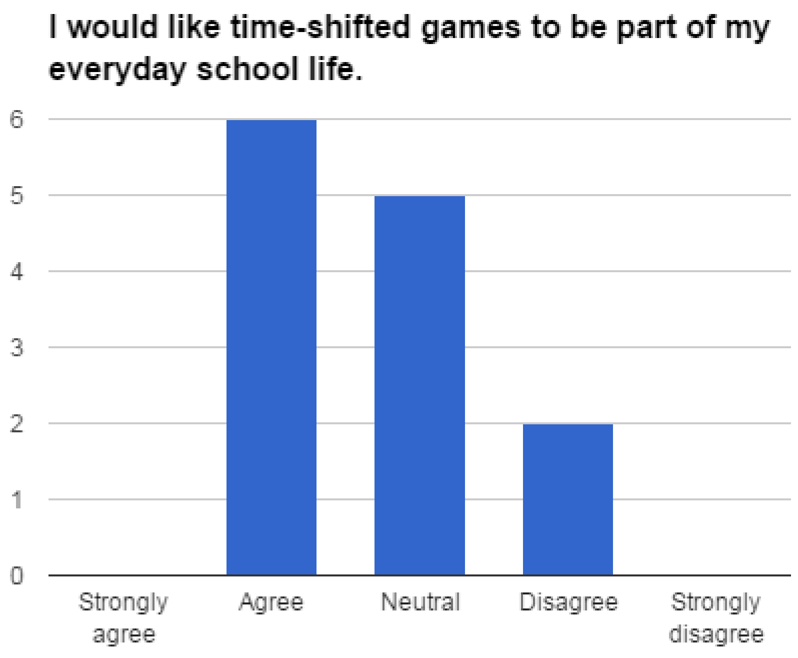


Figure 9. 16: Time-shifted games as part of everyday school life.

From the results in Figure 9. 16 we can see that the participants don't seem to have very strong opinions about it with no one strongly agreeing or disagreeing. However, the trend is slightly positive "agree" being the most commonly selected option. There is still a large amount of "neutral" answers which could imply that the question was not concrete enough and that the participants went for the 'safe' options. Another explanation is that they did not understand what 'everyday school life' meant and that they don't see it as very impactful on their education.

Another education related question we wanted answered was if the participants would want to use time-shifted games to prepare for tests or exams. When we started working with time-shifted quiz games, this was what we considered the most likely use for it.

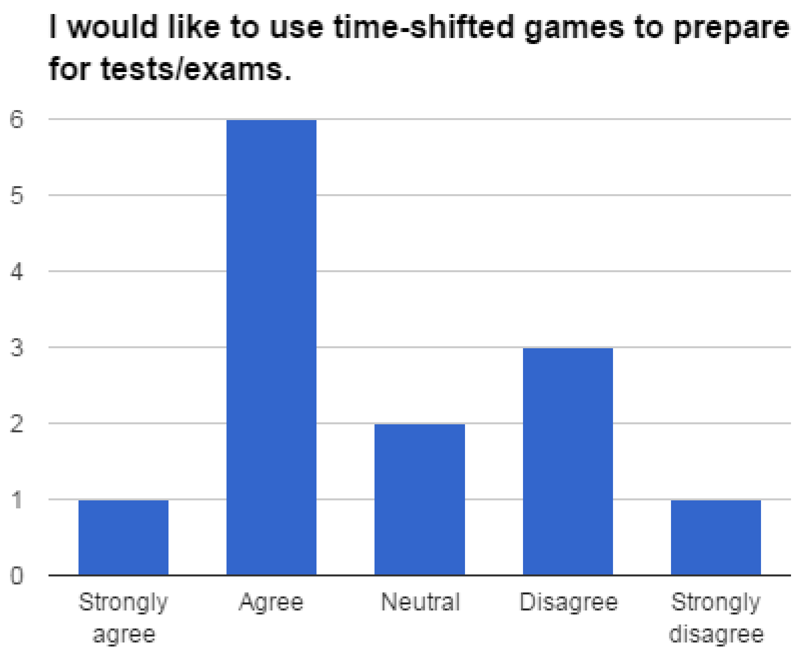


Figure 9. 17: Time-shifted games as preparation for tests/exams.

The results in Figure 9. 17 are quite spread. While the trend is overly positive with 'agree' being the most popular option, 'disagree' followed second. Because of the large spread, it is difficult to make any good judgments over how successful time-shifted game will be to help students prepare.

A small majority seem to think it would be a good option, so it might be worthwhile developing it least as an optional tool. While the question asks about the concept of time-shifting, it is reasonable to believe that the quality of our application would influence the participants perception of time-shifting overall. Our game was created by two students over a relatively short time period, and a more refined version might give more positive feedback.

Our last question related to education was about if the participants would want to use it for homework. Since the answers from previous players have to be stored to create the time-shifted behavior, it should be quite easy to use this as homework for a class since they do not have to do it at the same time.

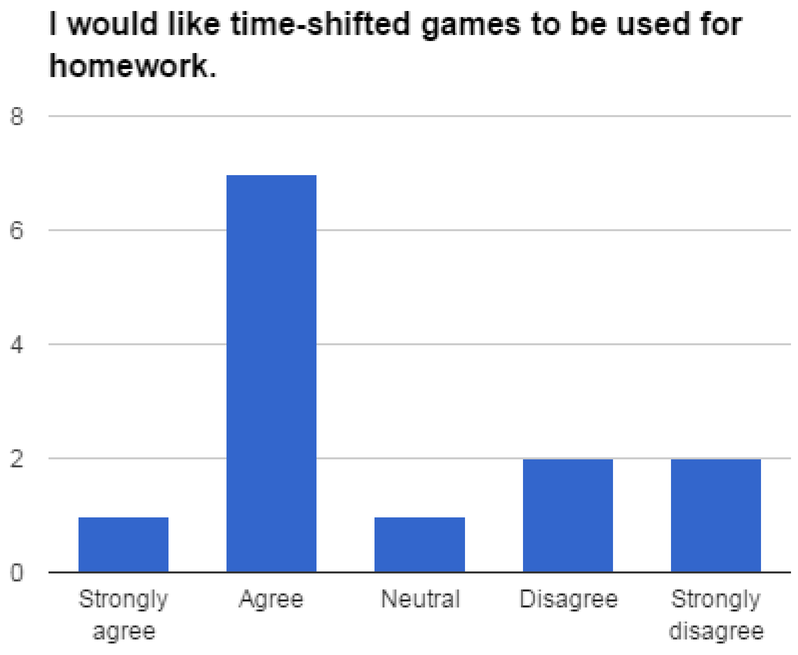


Figure 9. 18: Usage of time-shifted games for homework.

The results in Figure 9. 18 were largely positive. The option 'agree' was by far the most common one with more votes than all the other options put together. This implies that the participants sees real potential in this use of time-shifting. We also see that there were very few 'neutral' answers, which leads us to believe that the participants had a good understanding of what time-shifted homework would entail.

While the trend here was largely positive, there were still quite a few answers negative to the use of time-shifting in homework. An interesting observation is that the players answering that they were experienced with educational games tended to be more positive to its use in homework. Out of the five participants that answered 'agree' on the experience question, there were three positive and only one negative answers. The last one answered 'neutral'. In addition to the one experienced participant giving a negative answer, only participants answering 'disagree' on experience gave negative answers to using it in homework. While we should be careful not to draw hasty conclusions, the correlation here might be a causation. If this is the case, then the participants currently negative could be expected to change their opinions simply by being exposed to more educational games.

9.2 System Usability Scale Evaluation

In this chapter we will present the results from the SUS part of our survey and comment on our findings.

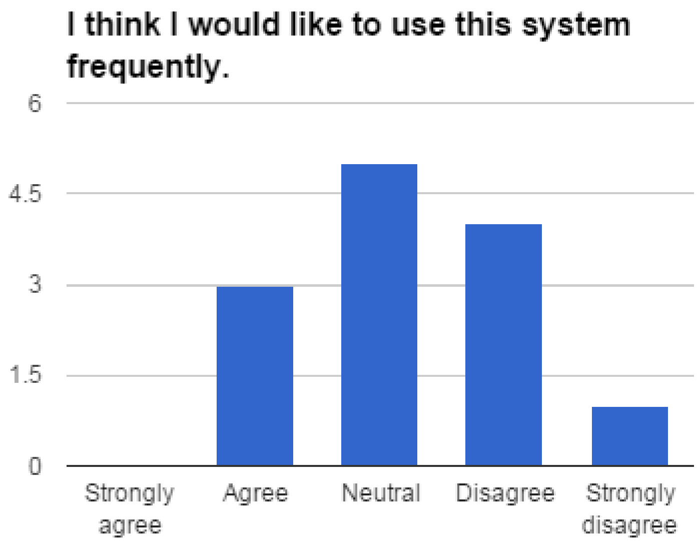


Figure 9. 19: Frequent use of the system.

On the first question the users were asked if they would like to use this system (the game) frequently. Figure 9. 19 shows that the testers mainly are neutral to the system, but there is a slightly negative attitude to the system. The reason for the slightly negative attitude can have several reasons. First of all the users might not like mobile games at all, and therefore gives the system a negative score. Second, they might not like the look and feel of the game and are left with a negative experience. Third, they might not see a problem with the way mobile games are today and do not feel there is a reason for using time-shifting in games.

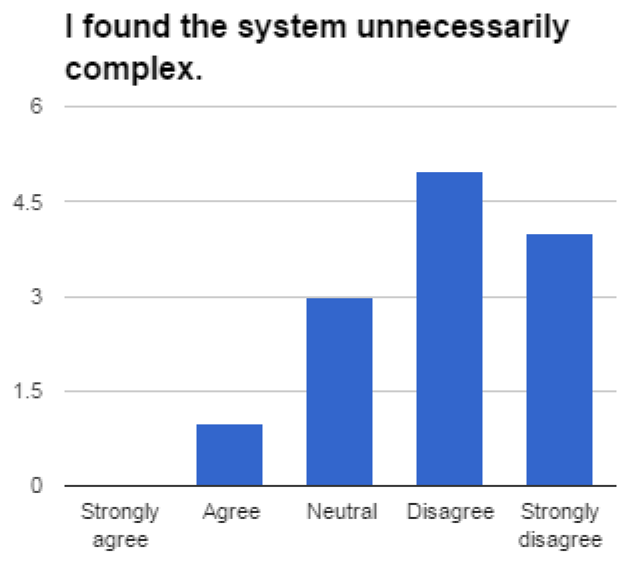


Figure 9. 20: Complexity of the system.

When asked if the users found the system unnecessarily complex there was a clear trend of users disagreeing, as seen in Figure 9. 20. This is a good result and makes it less likely that the feel of the system impacts the previous question. We do, however, not know if the look of the application affects the likelihood of people wanting to use the application.

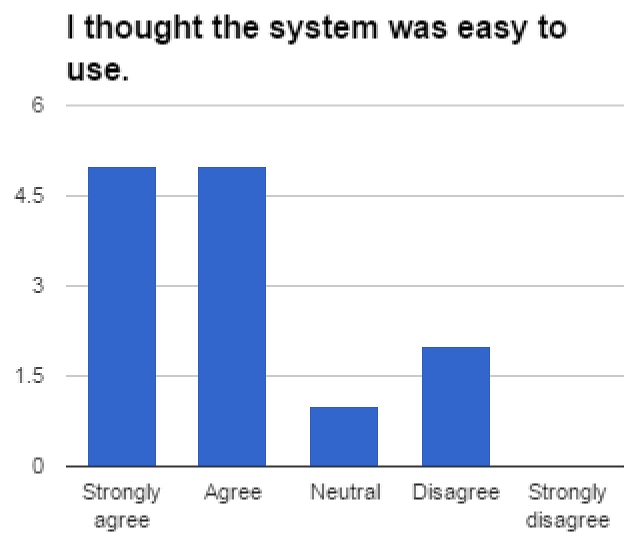


Figure 9. 21: Ease of use.

As Figure 9. 21 shows, there is also a clear positive trend on the question about how easy it was to use the system which underlines the findings from Figure 9. 20.

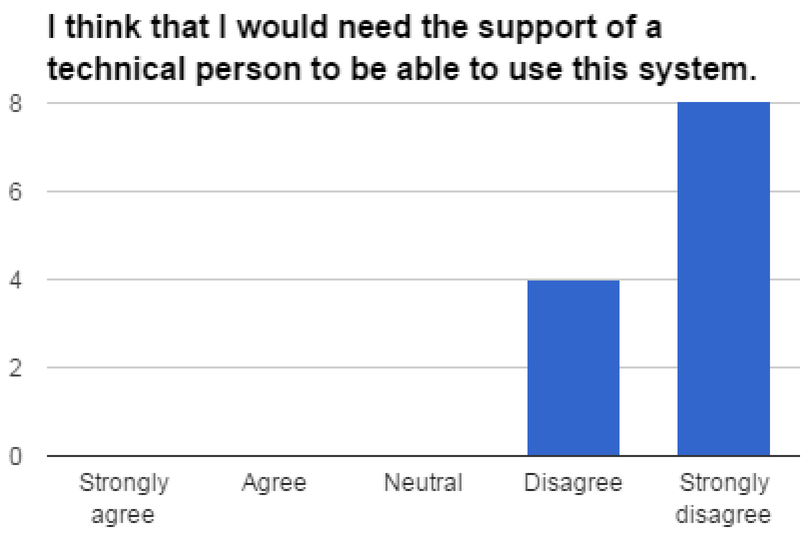


Figure 9. 22: The need for technical support for the system.

The people that participated in the tests agreed that there was no need for a technical person to help them use the system, as we can see in Figure 9. 22. This is no surprise since they all have a technical background and are very competent with technology. This does, however, not say anything about how your average Joe would feel when using the application and whether or not he would need help from a technical person. The reasoning behind only using people with a technical background is simply to make the testing run smoothly with the testers needing next to no help to install and use the application.

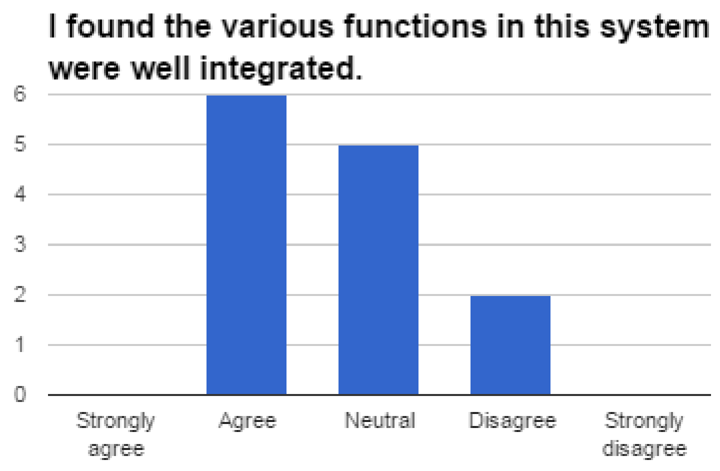


Figure 9. 23: Integration of functions in the system.

When asked if they found the various functions well integrated in the system, as shown in Figure 9. 23, there was a positive trend but there were also some that disagreed. The reason behind this is unclear, but the fact that the application crashed from time to time might have an impact on the results of this question. There is, however, functionality that we implemented that is working flawlessly but it was not tested with the users. The reasoning behind this is that the main purpose of the testing was to see how time-shifting was received and not if the friends list and challenge functionality was working as intended. The friends list and challenge functionality was thoroughly tested during development and is working as intended.

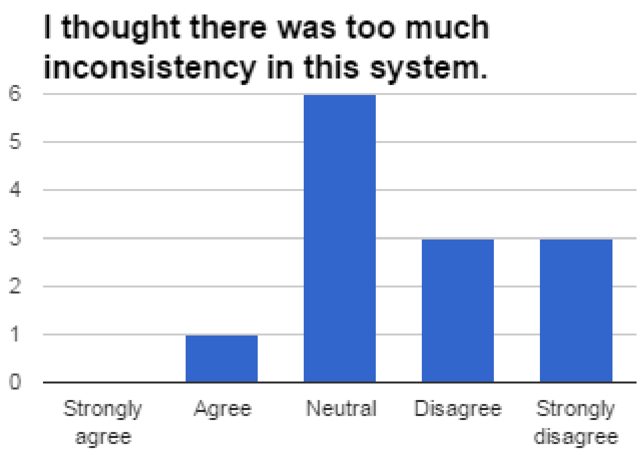


Figure 9. 24: Inconsistency in the system.

On the question about inconsistency in the system, Figure 9. 24 , the responses were mixed. This is most likely due to the fact that the application had some random crashes. Before testing we told the testers that the application could crash and there might be restarts during testing which can be interpreted as inconsistency. The main parts of the application however does work as intended without any inconsistencies.

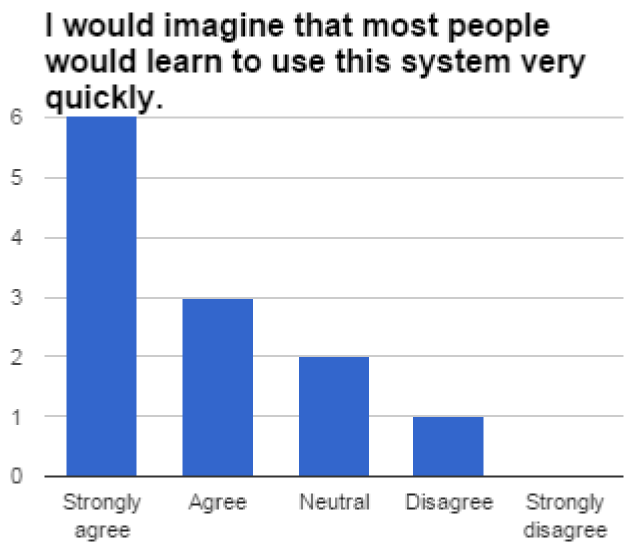


Figure 9. 25: Learning to use the system quickly.

When asked if they think that most people would learn to use the system quickly the answers were more or less positive, as seen in Figure 9. 25. What we can take from this is that the application probably has a low starting threshold for new users, which is really positive.

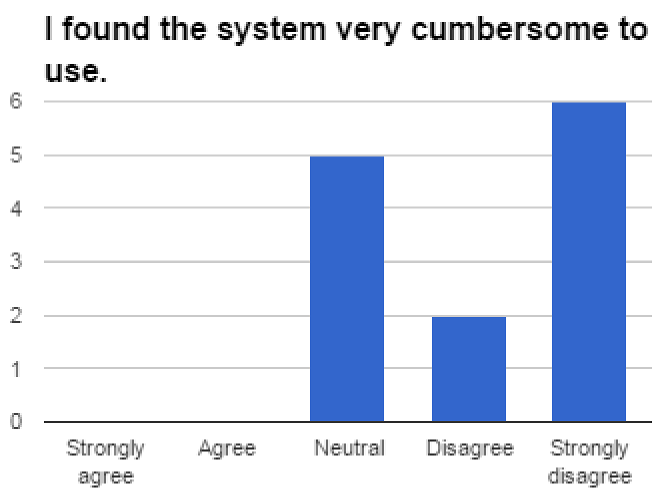


Figure 9. 26: Cumbersome to use the system.

From the results of the previous questions there is a trend that implies that the system is easy to use, but there was some neutral answers to the question about how cumbersome the system was to use as shown in Figure 9. 26. A possible reason behind this is that some of the users did not really know what cumbersome meant, some of them asked and some of them did not, and this might make it more likely for them to answer neutral.

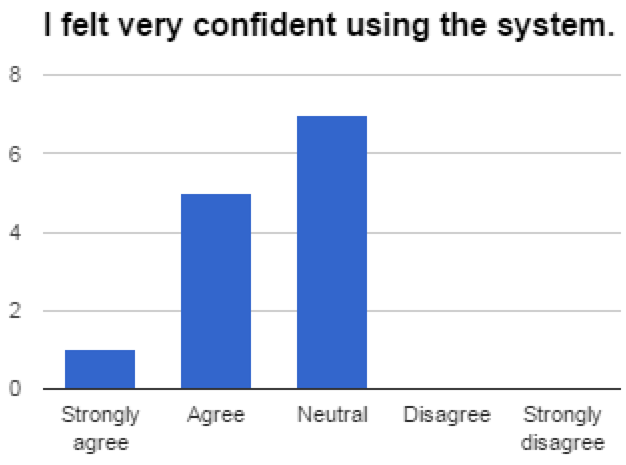


Figure 9. 27: Confidence when using the system.

On the question about how confident the users felt using the system the results from Figure 9. 27 were a bit surprising. Almost half the testers answered that they were neutral while the other half was slightly positive. The reasons behind this can be that people hardly ever feel confident using something new and therefore takes the safest option, which is neutral. Another reason, that builds on the previous one, is that the wording of the question might be a bit 'strong' since it says 'very confident' which makes it more likely for the testers to not give a 'strong' answer.

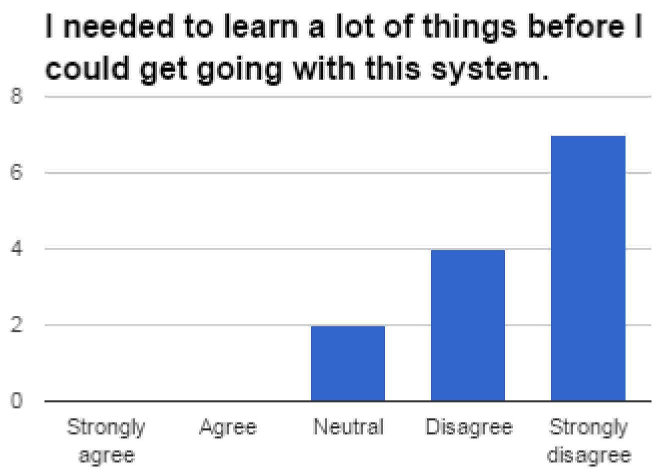


Figure 9. 28: Learning needed to get going with the system.

The last question of the SUS test asked if the testers felt like they needed to learn a lot before using the system. The results were clearly positive, as shown in Figure 9. 28, meaning they did not feel they needed to learn a lot before using the system. This simply underlines all the other results that the application is all over easy to use for new users.

9.2.1 System Usability Scale Total Score

For all the results shown above we calculated the total SUS score. The SUS scores goes from 0-100 where a high score is desirable. The results are shown in Table 9.1:

Participant	SUS score
P1	45
P2	85
P3	65
P4	80
P5	80
P6	67,5
P7	77,5
P8	90
P9	55
P10	87,5
P11	57,5
P12	62,5
P13	75
Total	**71,04**

Table 9.1: System Usability Scale scoring.

The highest SUS score among the participants was 90 while the lowest was 45. The difference here is quite big and luckily the average is closer to the highest score than the lowest score. The average turned out to be 71.04 which is quite good and we are happy with the result. This means that the testers did not have any problems using the application or felt that they needed assistance to learn how to use it.

When using the SUS scale, a score of 68 is considered average [7], which means that our application scores slightly above average. The intention of the SUS scale in our application was to determine if the quality of the application would negatively affect the participants view on the concept of time-shifting. With our above average score, this will most likely not be the case.

9.3 Functional Requirements

This section will look at the functional requirements presented in Section 6.1.1 and to what extent they were implemented.

FR1: Register, High priority, *The user must be able to register with an account the first time she starts the application to associate her device with the user account.*

This requirement was quickly implemented using functionality provided by the Parse framework. Every device is associated with an installation in the database the first time the users installs the application. However, an installation can have several user accounts connected to it.

FR2: Sign in, High priority, *The user must be able to sign in and out of her user account.*

The functionality required for this was, like the Register, implemented using methods from the Parse framework. The main screen contains a "Log out" button which, when pressed, will call the Parse's User Logout functionality. If a user has not logged out, but only exited the application by clicking the "Exit" button, the application will remember the user and redirect her to the main screen instead of the login screen next time she starts the application.

FR3: Start game, High priority, *The user must be able to start a quiz battle.*

To provide this functionality we used the same method as Kahoot! with players entering the quiz ID for the quiz they want to participate in, before they are navigated to the lobby screen together with the other participants. Users can also accept challenges they have received from their friends to be directly sent to a game lobby without having to enter a quiz ID.

FR4: Real-time multiplayer, High priority, *The users must be able to play multiplayer games against each other in real-time.*

This functionality was implemented by creating multiplayer lobbies where channel broadcasting is used for communication between the players. The channel that messages are broadcasted on is the same as the quiz ID to easily connect all the players and make sure every player is updated on what the other players are doing.

FR5: Time-shifted multiplayer, High priority, *The users must be able to play time-shifted multiplayer games against other players.*

Since time-shifted multiplayer requires someone to handle the bots we decided to always have a master for each quiz. The master is always the first player that joins the game lobby and triggers all the bot events like joining lobby, answering questions and so on. The master will experience the game just like all the other players since all the bot handling is working in the background. By doing this there is no way for the first player to know if he only plays against bots, humans or a mix.

FR6: Multiplayer, Medium priority, *The user must always have an opponent, human or AI.*

To make sure that a user never plays alone we made a lower limit of three players in a lobby. If there are no other players joining the lobby it will be filled with AIs. If not, there will be a mix of humans and AIs or just humans.

FR7: Timer, Low priority, *The application must present a timer to let players know when the next question is ready.*

The timer is represented by a progress bar implemented in JavaScript and CSS. The progress bar slowly fills itself from empty to full giving the players a visual representation of how much time they have left to answer.

FR8: Answer notification, Medium priority, *The application must let the user know when an opponent has answered.*

This functionality was implemented by using the channel broadcasting to trigger an event for each client. The trigger calls a function in JavaScript that shows "<username> has answered" written above the progress bar for each player. The broadcast is sent by the client that answered and received by everyone else in the same channel.

FR9: Score, Low priority, *The player must be able to see their own score and the scores of the top players between questions and after the quiz is finished.*

Between every question in a quiz, the players are navigated to the "Score screen" where the top five players and the players own score and username is shown. After all the questions have been answered, the players will be navigated to the "Final score screen" similar to the "Score screen" where the top five and their own score will remain visible until they decide to exit the quiz. These scores are queried from the database after each question to always keep them up to date.

FR10: Challenge, Medium priority, *The users must be able to challenge their friends to play a quiz against them.*

To implement this functionality we utilized the friends list in FR11. After a player finishes a quiz she can click the "Challenge" button to get a list of her friends. Here she can select the friends she wants to challenge. The challenges are stored to the database and is shown to the players being challenged the next time they log in.

FR11: Friends, Medium priority, *The users must be able to send, accept and reject friend requests from other users.*

If the user clicks the "Friends" button in the main menu she will get a list of friend request that she can accept or decline by clicking the corresponding buttons. She will also get a list of her current friends and the option to send friend requests to other users by typing in their name in an input field and clicking the "Add" button.

FR12: Security, Low priority, *The user accounts should be protected by a password created by the users at time of account creation.*

To make sure the user accounts are somewhat secure we used Parse's inbuilt user accounts with hidden passwords. Since this is just a game it is not necessary to have a lot of security, but it is always good to have some.

9.4 Non-functional Requirements

NFR1: Realistic time-shifted bots, High priority, *70% of the users should not be able to differentiate between time-shifted and real time players.*

As shown in Figure 9. 11: Accuracy of opponent classification. only 26.9% of the classification of opponents were correct. That leaves 73.1% that were wrong. This is very close to what we predicted and a good overall result. As mentioned in Section 9.1.3 there was a higher probability to guess a bot correctly because of how the testing was conducted. If there would have been more people attending the tests the results would probably be a bit different, most likely towards a higher percentage of wrong guesses.

NFR2: Capacity, Medium priority, *The application should handle up to 50 simultaneous players.*

The application was never tested with 50 simultaneous players, but there is nothing indicating that it would not work. The amount of data sent between players is so little that there will not be an issue for players. What might be a problem is the large amount of notifications popping up during a question. If a lot of players answers at the same time it will be hard for the players to keep track of who has answered and at what time. The same problem arises in the score screen when players sends emotes. If a lot of players sends emotes at the same time the chat will be flooded and hard to keep track of.

The issues mentioned above might arise when all 50 players are in the same lobby. If they are spread over several lobbies there will not be any problems. The data sent between players in different lobbies are handled separately and will not disturb other players. In theory there can be thousands of simultaneous players as long as the number of players in each lobby is within a reasonable amount.

NFR3: Enjoyment I, High priority, *80% of the users of the application should prefer it over ordinary studying.*

In Section 9.1.2 we looked at three different charts about how positive the testers were to using the application for studying. Those three figures are merged into Figure 9. 29.

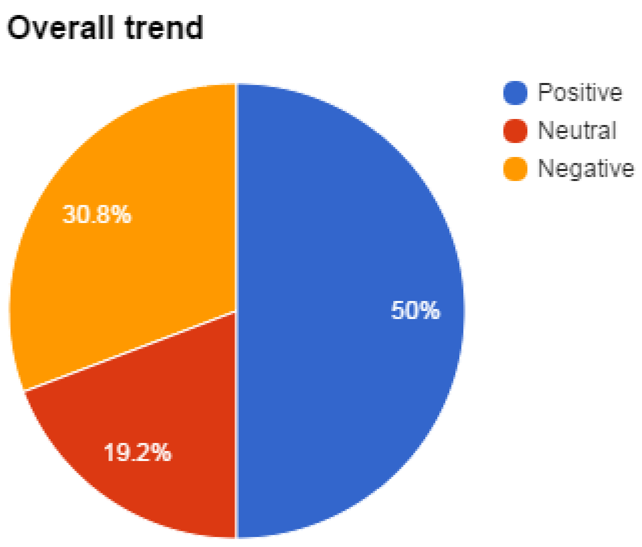


Figure 9. 29: Overall trend.

Most of the testers, 50%, were positive while 19,2% were neutral and 30.8% negative. We can argue that the neutral testers count as positive or will be positive after more exposure to it. As discussed in Section 9.1.4, the was a trend that the players with the most experience using educational games were the most positive. This arguably means that the participants that are neutral now will turn positive once they get more experience with the technology. If we combine the positive and neutral testers we reach 80.3% which is better than the 80% goal. While this might be stretching the positiveness of neutral answers too far, it is worth noting that more exposure appears to increase positiveness. Overall the testers are positive to using the application over normal studying, but not quite as positive as we had hoped.

NFR4: Enjoyment II, High priority, *70% of the users should feel that time-shifted multiplayer gives the same enjoyment of a game compared to real-time multiplayer alone.*

In Research Question 1.2 we looked at the testers enjoyment when playing real-time multiplayer compared to time-shifted multiplayer. We have used the results from Figure 9. 10 and transferred it to a percentage distribution where the testers agreeing counts towards real-time, the testers disagreeing counting towards time-shifted and neutral towards neutral.

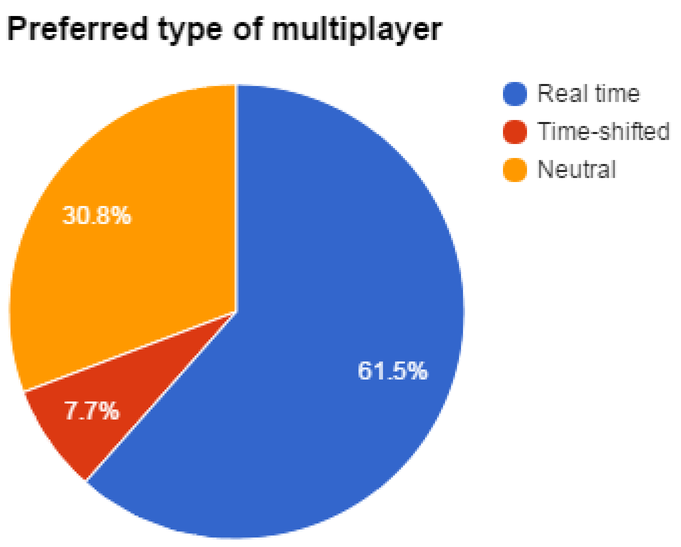


Figure 9. 30: Preferred type of multiplayer.

The results are not quite what we hoped for. As we can see in Figure 9. 30 there is a slight overweight, 61.5%, of testers liking real-time multiplayer the most, next comes neutral at 30.8% and last time-shifted with 7.7%. This shows that real time multiplayer is preferred by most, which makes sense, but there is also some hope for time-shifted multiplayer. With 30.8% showing no preference and 7.7% preferring time-shifted multiplayer. It is important to note that time-shifted multiplayer will not necessarily take over real-time multiplayer, but rather become its own type of multiplayer genre.

NFR5: Maintainability, Low priority, *The source code should be divided into appropriate classes dependant on their corresponding view.*

To keep the code easily maintainable, every view, that is the different pages on the application, is divided into three parts. The java backend that communicates with the database and handles most of the functionality, the front end written in HTML, CSS and JavaScript that handles the user interface, and the Interface class between them handling communication between them.

The classes in the view only effects that view, and, with the exception of the Parse Receiver support class, no other classes than those within the view affects it. This separation allows for easy extendibility and bug fixing as an issue in a view must be within the classes constructing it.

9.5 GameFlow Analysis

In Section 2.4 we presented a framework for evaluating enjoyment in games. The creators of the framework use a score system from one to five for different questions within the eight criteria of the game.

Like mentioned in Section 2.4.2 analyzing the flow ourselves is not ideal because of possible bias, and it is important that readers keep this in mind through this evaluation. Despite its flaws, we still think that the GameFlow framework will give insight into the strengths and weakness of our application giving a clearer view of what improvements can be done to improve it.

Our analysis will be based on the answers from the questionnaire and the tests to be more precise than just using our subjective opinion.

Another thing to keep in mind is that our game is very simplistic, while the framework is designed for all types of computer games, some significantly more complex than ours. Because of this, some of the criteria are not well fitted for analyzing our application. The criteria will be presented written in bold and our analysis in plain text. Every criteria will be given a score between 0 and 5 as discussed in Section 2.4.2, the results for all the criteria in a element are then normalized creating the score for the element.

9.5.1 Concentration

Games should require concentration and the player should be able to concentrate on the game.

Games should provide a lot of stimuli from different sources.

We use several sources of stimuli in our game. In the quiz we use sounds and textual notification when players answer, a graphical bar indicating how much time they have left to answer, and a textual display of the question and answers.

In the score screen we show the top players and a simple chat for the players while they wait for the next question.

However, the game fails to give the players anything to do after answering a question other than waiting for the other players to answer and the timer to run out.

Score: 3/5

Games must provide stimuli that are worth attending to.

All the stimuli in the game are there to give the players information. The sound indicates when players answer, the textual notification lets them know who it was and the timer gives information about how much time they have left to answer.

In the score screen the high score and their own score give them information about how they are doing compared to the top players, and the chat allows them to do simple communication with their opponents.

Score: 5/5

Games should quickly grab the player's attention and maintain their focus throughout the game.

In all the tests, the players appeared to become immersed in the game once the lobby countdown told them that the game was about to start. For the most part, the players remained focused on the game, but would often lower their phones if they answered a question early, and had to wait for the question timer to run out.

This was especially noticeable in the pre-test, where the participants was a group of friends that would talk and laugh up to the point where the quiz started and then remain in quiet concentration on the game for the duration of the test.

Score: 3/5

Games should have a high workload, while still being appropriate for the players perceptual, cognitive, and memory limits.

The workload of the game largely depends on the difficulty of the questions and as such can be fitted to the target audience. Looking at the SUS results we find that the participants found the game easy to use indicating that they did not find the amount of stimuli overwhelming. However, there did appear to be some down time for the players that answered quickly, as they would just sit and wait for the timer to run out

Score: 3/5

Overall score: 14/20 = 3.5/5

Players should not be distracted from the tasks they need to concentrate on

This criteria, while somewhat relevant, would give a false impression because of the simplicity of our application. In the researcher's original presentation of GameFlow, this criteria referred to the game forcing you to do tasks other than the main task you wanted to focus on. An example from real-time strategy games was having to leave the battlefield to create more units. In our game, there isn't enough complexity for these types of extra tasks to exist, which would give us an unfairly high score.

9.5.2 Challenge

Games should be sufficiently challenging and match the player's skill level.

Challenges in games must match the players skill levels.

The challenge in the quiz depends on the difficulty of the quiz. When used for education, it will be up to the quiz creator, most likely the lecturer, to create questions at an appropriate difficulty level.

Score: 3/5

Games should provide different levels of challenge for different players.

Same the answer to the criteria above. The game contains no automatic way of adapting the difficulty to players of different skill level. While this could have been done using methods described in Section 2.3.4 under Dynamic Difficulty, using correctness and answer speed as metrics, it was not necessary for the tests we wanted to perform.

Score: 3/5

The level of challenge should increase as the player progresses through the game, and

increases their skill level.

The game has no automatic way to increase difficulty as every quiz is treated as its own game with no connection to the overall performance of the players.

Score: 2/5

Games should provide new challenges at an appropriate pace.

Every quiz is treated separately, and there is no overall logic to provide appropriate challenges. It used in education, it would be up to the lecturer to create new quizzes and figure out the correct pace of increasing the difficulty.

Score: 3/5
Overall score = 11/20 = 2.75/5

9.5.3 Player Skill

Games must support player skill development and mastery.

Players should be able to start playing the game without reading the manual.

Not relevant due to the simplicity of the game, N/A

Learning the game should include online help so players don't need to exit the game.

Not relevant due to the simplicity of the game, N/A

Players should be taught to play the game through tutorials or initial levels that feel like playing the game.

No tutorial was created for the game to teach basic game elements. Because of the simplicity of the game and the controlled situation in which it would be played, we relied on a simple out of game explanation of the basics like how scores are decided etc.

Score: 3/5

Games should increase the players' skills at an appropriate pace as they progress through the game.

Not relevant as every quiz is treated as its own game, N/A

Players should be rewarded appropriately for their effort and skill development.

The game gives currently has no reward system, but relies on the nature of competition to create intrinsic rewards for the players.

Score: 3/5

Game interfaces and mechanics should be easy to learn and use.

None of the participants had any problems starting or using the application. Looking at the results from the SUS, the users where generally positive to the ease of use of the application, with "strongly agree" and "agree" being the most picked answers when asked if they thought the system was easy to use.

Score: 5/5

Overall score: 11/15 = 3.67/5

9.5.4 Control

Players should feel a sense of control over their actions in the game.

This element is as a whole not very relevant for our application. The criteria for it are there to measure players control over their game units or avatar, and how they affect the game world. For a quiz game, there is no units to control or game world to affect, so the criteria are a poor fit to analyze our game. All the criteria can be seen in the introduction to GameFlow in chapter 2.4.2 The GameFlow Framework.

9.5.5 Clear Goals

Games should provide the player with clear goals at appropriate times.

Overriding goals should be clear and presented early.

Because of the simplicity of the game, this criteria is not a good fit for analyzing. The overriding goal of the game is to get the highest possible score. This is never explicitly presented in the game, but should be quite obvious to anyone playing the game. Looking at the SUS results we see that the participants agree that the application was easy to use, and in none of the test did we get any indication that the participants was uncertain about what they should do.

Score: 5/5

Intermediate goals should be clear and presented at appropriate times.

Intermediate goals in our game are to answer the individual questions given in the quiz. As mentioned above, this should be obvious to anyone and none of the participants appeared to be confused about the goal. While answering correctly was obvious for all participants, not all the participants immediately understood that faster answers were preferable and that speed would influence the score.

Score: 4/5

Overall score: 9/10 = 4.5/5

9.5.6 Feedback

Players must receive appropriate feedback at appropriate times.

Players should receive feedback on progress toward their goals.

In the score screen, between the questions, the players are presented with their own score and that of the top players. This gives them good feedback about how the stack up against the other players and how close they are to achieving the goal of winning the quiz.

The immediate goal of correctly answering question have no progression as you either have answered or you have not yet answered.

Score: 5/5

Players should receive immediate feedback on their actions.

When answering a question the game notifies you with a water drop sound and a textual message that you have answered. While there is some passing of data involved from the click on the answer to the notification comes, it should appear immediate to the humans

Score: 5/5

Players should always know their status or score.

Scores are only visible at the score screen, so while answering questions players do not know their last score and have to wait for the next score screen to see their score after the question.

While in the quiz screen, the two possible statuses a player can have is answered or not yet answered. If the player has answered that status is indicated on the selected option shown in Section 7.1.5 Question Screen. This indication should possibly have been made more visible, but none of the participants commented, or appeared to have problems noticing it.

Score: 3/5

Overall score: 13/15 = 4.33/5

9.5.7 Immersion

Players should experience deep but effortless involvement in the game.

The criteria for this element are presented in Section 2.4.2 The GameFlow Framework and revolve around how immersed players are in the game. They are however very specific in their wording stating that players should become less aware of their surroundings and everyday worries. Measuring this would require us to set up some very specific user tests to answer questions that`s not too relevant for our research.

Without doing such tests however, addressing the criteria would be nothing more than pure speculation which we feel won't add anything of value to our analysis.

Instead we will settle for stating that the participants appeared to be very immersed for the duration of the quiz, and their answers in the questionnaire show that they prefer playing against time-shifted opponents over regular bots, indicating that they would be more immersed in our game than in a regular single-player quiz game.

Total score: Unknown.

9.5.8 Social Interaction

Games should support and create opportunities for social interaction.

Games should support competition and cooperation between players.

Our game supports competition by providing time-shifted opponents in addition to other real-time players. There is no support for cooperation as every person is their own team competing against every other player.

Score: 3/5

Games should support social interaction between players (chat, etc.).

Social interaction with time-shifting is a complicated. In Section 5.2.2 we present some of the problems with two-way communication when time-shifting and since we don't want to give away which of the opponents are time-shifted and not, the same limitations have to exist for communication between real-time players. We do however have a simplified chat that provides communication through pre-defined emotes which allows for basic communication without revealing if the players are time-shifted or not.

Score: 3/5

Games should support social communities inside and outside the game.

The application has a in game friend system where you can add other players to your list of friends, and challenge them with quizzes you have finished. When the challenged accept the quiz, they will face up against the time-shifted challenger in addition to other time-shifted and real-time players entering the quiz at the same time.

The game is missing in-game functionality to challenge your friends to a real-time quiz. And the only way to get to do this, is to join the same lobby at the same time.

There is no community support outside the game, as it is just a prototype created for testing purposes.

N/A
Overall score = 6/10 = 3/5

9.5.9 GameFlow Total Score
Our scores of the different GameFlow criteria is shown in Table 9.2:

Criteria	**Score**
Concentration	3.5/5
Challenge	2.75/5
Player Skill	3.67/5
Control	N/A
Clear Goals	4.5/5
Feedback	4.33/5
Immersion	Unknown
Social Interaction	3/5
Overall Score	**3.625/5**

Table 9.2: GameFlow scoring.

In the GameFlow framework, a score of 3 is average, so a score above that is good considering it is for a prototype. However we can detect a few problem areas in our application. The '*Challenge*' element is our weakest part judging from this analysis. the main problem appears to be that we rely on the quiz creator heavily on the quiz creator to determine the challenge. This means that the first step in improving the enjoyment of the user would be to add more mechanisms to adapt the difficulty. A simple way of doing this would be to have quizzes that are level locked, so once you can finish a quiz with a good score, a more difficult quiz becomes available.

Another method discussed in [23] would be to dynamically change the difficulty. The obvious metric to change for the quiz is how quick the timer is, but other methods like allowing the participants several guesses could also be worth exploring.

'Social Interaction' is another element where our application has scores quite low but this might be more difficult to improve. Like we discuss in Section 5.2.2, communication in time-shifted games can be very tricky, and while it surely is ways that the current solution can be enhanced, it will probably never be as good as communication on real-time games.

In the "concentration" element, most of the weaknesses was related to the wait from players answered to the timer runs out. Very quick players will spend large parts of the game just waiting which obviously is not fun. One improvement to this could be to the score screen once all players have answered even if there is still time left on the timer. Adding a list with all the players names showing how many has answered at any given time might also help as the quick players can watch how the rest of the players are doing.

Analyzing the *'Player Skill'* element, we found that the game might need some method to explain the game mechanics. While the goals are quite obvious, how the scoring is not as nothing indicates to the player that fast answers are preferable to slow ones. While a tutorial might be too much for such a simple game, a rulebook or added clarity inside the game would improve the enjoyment for new players.

The remaining elements are either not relevant or have a good enough score that improvements to them should not be prioritized. It should be noted that *'Immersion'* has not been analyzed because of the difficulty in getting solid evidence one way or the other and we opted not to speculate. Like mention in the introduction, the scoring is based on the results from the user tests, but done by the us. Because of this, this analysis and the scores should be seen more as guidelines to find points of improvement than a measurement of quality.

Part V

Conclusion

10 Conclusion

In this chapter we will present our conclusions for Research Question 1 in Section 10.1, Research Question 2 in 10.2 and present our key findings in Section 10.3.

10.1 RQ1 - How does time-shifting affect the user experience?

One of the main questions we wanted to answer with this project is whether time-shifting affects the user immersion and enjoyment compared to single player games and real time games. To try answer this we wanted to take a closer look on how time-shifting is perceived by people that are not familiar to the time-shifting term in gaming. Overall, we found that the participants were positive to time-shifting both for educational use and as a genre. This is further discussed in Section 9.12.

RQ1.1 *How does time-shifting affect users immersion and enjoyment compared to single player games?*

Time-shifting has a positive effect on the enjoyment felt by the user. As shown in Figure 9. 9, more than half of the participants "agreed" of "strongly agreed" that time-shifted opponents were more interesting than the regular AIs you would meet in a single player game. The rest of the participants answered that they had no preference and there were not a single negative answer to this question.

RQ1.2 *How does time-shifting affect users immersion and enjoyment compared to real time games?*

While preferable to single-player, time-shifting does not fully recreate the enjoyment of real-time multiplayer. As shown in Figure 9. 10, the overall trend is that the participants enjoy real-time play over time-shifted. Because of this, we believe that time-shifting should be seen as an option when real-time multiplayer is impossible and not as a replacement.

10.2 RQ2 - How difficult is it to differentiate between real-time players and time-shifted players?

Ideally we wanted our time-shifted players to be indistinguishable from real-time players, and in our application we were to a large degree capable of doing this. As shown in Figure 9. 11, the participants had very low accuracy in correctly distinguishing between real-time and time-shifted players. As shown in Figure 9. 14, most of the participants answered that they found it difficult to distinguish, which fits with the results. Interestingly, some of the participants thought it was easy to distinguish even when most of their answers were wrong.

RQ2.1 *How did players differentiate between time-shifted and real time players?*

The participants had many different methods for trying to differentiate between time-shifted and real-time players. While none of these methods were effective in our application, being aware of them will be useful when creating other time-shifted games. One of the methods used was to look at the emotes in the chat. They thought that the players who were fastest to emote or sent random messages were time-shifted players.

Another method related to timing was in the lobby, where some of the participants thought that the time-shifted players would set themselves to ready faster than the real-time players. During the questions, some of the participants thought that players that gave very quick answers would be time-shifted. As we can see, all of these methods are related to the timing of actions, and developers should be extra careful when handling this in their applications to not give away who the time-shifted players are. The last method, and the only one that could have been successful, was to notice a bug in our application. This bug is explained in Section 11.2.3 and caused the time-shifted players that answered a question incorrectly to always answer in the last split second before moving to the score screen. This proves that developers will have to be very thorough when creating time-shifted games as even small bugs can let players know the identity of time-shifted players and break the illusion of real-time game play.

RQ2.2 *What methods are available for solving the communication problem in time-shifting?*

As discussed in Section 5.2.2, two-way communication between players in the traditional sense of talking to and answering each other is completely impossible by the very nature of time-shifting. The first player will already have finished the game by the time the second player starts. This is further discussed in the beginning of Section 5.2.2 where we conclude that communication from the second player to the first one will not be possible while the first player is doing her playthrough.

Communication from the first player to the second one is much simpler. Using the method described in 5.1.2 The Stored Input Method, we can store the message sent and the timing of the message and then play it when the second player reaches the point in the game where the message was sent.

But there are still some limitations to how this can be done and while preserving the illusion of real time play. The problem is that messages that make sense for the first player might not in the second players play through. As an example the first player might say 'YES! , I won' after finishing a game with the highest score, but if the second player beats the score, that message is wrong and will break the illusion of real-time play.

There are a few methods that can be used to solve this problem.

Emotes

The method we used in our application is to limit the possible messages that can be sent so that illogical messages cannot be sent. Instead of players writing their own messages, we created several possible messages that players can pick. By doing this, we could make sure that that situations like the one above could never happen as the emotes were ambiguous enough that you could not make statements that`s flat out wrong. This proved to be quite successful as our participants could use the chat to communicate, but were, as shown Figure 9. 13, not capable of correctly separating time-shifted from real players.

Dynamic Messages

Another way to deal with illogical messages is to modify the messages so that they fit the situation. This could be done by modifying key words in the message. In this situation the "YES, I won" would be replaced by the fitting placement so the message becomes "YES, second place". There are two issues with this solution. The first is purely technical. Detecting keywords in free language might simply be too difficult for present day technology.

A solution to this is to combine the dynamic messages with the emote method. Using the emotes to limit the language so that the keywords are known and then fitting them to the current situation.

The other limitation is that when you replace parts of the message, it is no longer the player that said it. a person might want to say "YES, I won" but would use another or no message at all if finishing second. This could create behavior that is quite different from that of the real player and even from humans in general, breaking the spell of real-time multiplayer and reminding the players that they are not actually playing against a real person.

How different this behavior would be and how it affects the players is something that will require more research.

10.3 Key Findings

In this chapter we will sum up our key findings from our research and what conclusions we think can be drawn from them. We will also discuss the weaknesses in our study and how they may affect our results.

10.3.1 Efficiency of Time-shifting

One of the questions we wanted answered in this thesis was to what extent players were capable to notice the difference between real and time-shifted players. What we discovered, and present in Figure 9. 11, is that the participants did not manage to do this with a high degree of accuracy. This was quite surprising to us and is very promising for the future of time-shifting.

Our game was very simple, and possibly the easiest game to create time-shifted behavior for, and more research is necessary to see if the results we found can be recreated in other genres of games.

10.3.2 Time-shifting as a Game Mechanic

Results from the questionnaire show that they strongly preferred time-shifted AIs over regular AIs with not a single participant answering negatively to it. When asked to compare time-shifted multiplayer to real-time multiplayer, the overall trend was that they preferred the real deal.

What this shows is that replacing real-time multiplayer with time-shifted is not a good move to increase the player's enjoyment. Instead it should be used as an improvement to single-player games when real-time play is impossible or inconvenient.

10.3.3 Time-shifting in Education

When asking the participants about their views on time-shifting in education. Our findings, that are presented in Section 9.4 Non-functional Requirements, show that the overall trend is positive, but not overwhelmingly so. However there was a correlation between exposure to educational games and positivity towards using time-shifting in education. If this correlation turns out to be a causation, introducing time-shifting in education will cause students to have a more positive view once they get used to it.

Before testing we thought that the game would be best suited for preparing for tests, and while the overall results show that the participants want to use time-shifting for that purpose, the area they were most positive to was using it in homework. Because of this, attempts on using time-shifting in education should focus on this first. Using it for homework is convenient for lecturers and developers as well. The creation of the time-shifted players already stores and checks all the answers, decreasing the workload of creating the functionality and decreasing time spent by lecturers checking individual answers.

10.3.4 Weaknesses

The main weakness of this study is the low number of participants, and the selection of these. While certain trends present themselves, a larger study with a higher number of participants would help us determining the strength of these trends and increasing our certainty that our findings are correct.

The selection of participants was not random as we wanted tech-savvy students to simplify the installation process. This means that the trends we found won't necessarily hold for the overall population of students as it is possible, and maybe even likely, that these students are overall more positive to more technology in education than everyone else. This does however increase our confidence in that time-shifted AIs can be made indistinguishable from real-time players. Most of the participants in our test were IT-students which means that they are familiar with programming and thus should be better suited to discover programmed behavior, e.g. looking at join timings in the lobby etc.

Another weakness is that the participants might not have been able to abstract the concept of time-shifting away from our application. This means that participants answering positively to the use of time-shifting in education might have been positive to the use of quizzes and not care whether it was time-shifted or not.

11 Further Development

In this chapter we will look at the natural next steps of our application and areas within time-shifting that should be looked into. In Section 11.1 we will look at our application and the improvements that can be done to it. In Section 11.2 we talk about time-shifting and how it can be used in education. In Section 11.3 we discuss the issues revolving player interaction in time-shifted games. Section 11.4 suggests how games should be created to better fit time-shifting. The section, Section 11.5, takes a look at the communication between players in time-shifted games.

11.1 Added Functionality

In this chapter we will look at the improvements that can be done to our application. The chapter is divided into two parts. In Section 11.1.1 we will look at the bug fixes and fixing functionality which currently doesn't meet the expectations we had when we envisioned it, and in Section 11.1.2 we will look at functionality that currently does not exist, but could be added to improve the application.

11.1.1 Possible Fixes to the Application

Since the application was created for testing purposes, we did not prioritize issues that would not end up influencing the user testing. These issues could cause problems when the application is used outside of a controlled environment, and has to be resolved if the application is to be used on a broader audience.

Multiple lobbies

The way we currently handle communication between clients is that every lobby has a channel, and every player that joins the lobby automatically subscribes to that channel. For simplicity, every channel got the name of the lobby it was connected to. This creates the issue that if you join a channel that`s already in a quiz, you would miss out on the questions that had already been finished, and it would create problems in the databases score objects. To prevent this, we lock the lobby and channel once the quiz is ongoing, and players trying to join will be notified that the lobby is busy.

This solution worked well for testing where we would only have one quiz running anyway, but if the application is to be used for a bigger audience, this would lead to quizzes being busy most of the time.

The best solution to this is to change how the lobbies currently work. Instead of having one lobby for each quiz, each quiz could spawn several lobbies with their own channel. This would require some sort of logic to determine when a new lobby is required, and when a used lobby becomes available again.

Visual lobby bug

Whenever someone enters a lobby, their client queries the database for every player currently in the same lobby and the return message populates the lobby with their names as shown in 7.1.4 Quiz Lobby Screen. A message is also sent in the channel to let everyone in the lobby know that a new person has joined so they can add that name to the list.

When two players join at approximately the same time, they will sometimes get the person they joined with name both from the list of players before they joined, and from the message sent by that player. While this is nothing more than a visual bug that does not affect the game in any other way, it should still be fixed as it can happen quite frequently.

Non-Answering bots

This was one of the issues we thought players in the tests might be able to notice, and figure out who of the players were time-shifted and not. For time-shifted players, the scores are already in the database as they have finished the quiz already, and the way we get it to send the answer message at the correct time is to use their score for the current question and send the message when an answer would give you that amount of points. But if the time-shifted player had answered incorrectly to a question, their score for that question is 0 and to the other players it will look like they didn't answer at all.

11.1.2 New Functionality

In this section we will discuss possible new functionality that we think should be added to the application, but that we did not have the time to create do to the time constraints of the project.

Quiz creator

It is highly recommended that the application has functionality to let the players create their own quizzes. This will greatly increase the probability that people actually start using the application and it greatly improves the usefulness. It can be used in lectures, as homework, to study for tests, at parties and so on.

The current method for creating quizzes is to write the questions, answers, code etc directly into the database objects. This is only possible for owners of the Parse project to do, in other words only the two developers are currently able to create new quizzes and questions. The current architecture has questions and quizzes separated, so different quizzes can use the same questions, and a quiz creator should take advantage of this to lower the workload when creating new quizzes.

Non time-shifted bots

This is closely connected to the chapter above. The way we handle bots now is that all of them should be time-shifted. So if you start a new quiz, everyone of your opponents should either be real-time players joining at the same time as you, or players that have finished the quiz before.

But when a new quiz is created there will be no earlier players to use as bots.

There are a few ways to deal with this problem. The first is to generate bots with random scores for the different questions in the quiz, and add them to the database. These bots will then be phased out as more players play through the quiz until they are no longer needed.

An improvement to just randomly generating the scores is to have "profiles" for creating the bots. An example of a profile could be the "fast player" that answers quickly, but often incorrectly.

Another and in our opinion better way to create realistic scores is to base it on scores from other quizzes. If the new quiz shares questions with an older quiz, the bot generator could borrow real scores from those questions. If the question too is new, the generator could just pick a random score from a random quiz. While this probably won`t be a perfect solution, it will most likely give significantly more realistic behavior of the bots than just picking the scores at random.

More platforms

To improve the application even more it would be a good idea to make it playable on other platforms than Android. There are several benefits from doing this. The most obvious one is that it would reach a broader audience if PC and iPhone users could play the game as well. Another advantage is that the quiz creator would most likely be cumbersome to use on a phone if you need to create new questions. The possibility to do this on PCs would probably be a huge advantage.

How much work the transition from Android and onto other platforms would take is difficult to predict, but since we created large parts of the application using web technologies, it should be less than it otherwise would have been. The extension onto other platforms should not be a big issue since HTML5 works on cross platforms and there is a lot of frameworks that makes it easy to make responsive web pages, e.g. Bootstrap [32].

11.2 Improved Education

In our project we have studied how time-shifting affects the enjoyment and immersion when used in education, but not if it improves efficiency of learning. Competition can be a strong driving factor to improve performance in addition to improving enjoyment. Studying if/how using time-shifting can improve the quality of the education in other ways than by improved enjoyment would be interesting. It is plausible that the competitive factor of playing against friends and the convenience added by time-shifting might help students learn faster than traditional teaching methods.

11.3 Player Interaction

Certain limitations exists in what types of games time shifting is useful for. As discussed in Section 5.2.2, games where players directly interact are difficult to time shift, and more research is required to understand the limitations and possibilities time shifting can have in these game genres.

A good candidate for this is the game Rayman Legends for the Wii U console, a single or multiplayer co-op platformer. In this game, there were essentially two different roles the players could control. The main character with the usual platformer behavior like running and jumping, and another flying sidekick character controlled using the special Wii U pad to cut ropes, hold shields to protect the main character etc. In some levels, both characters are required to be present at the same time. For multiplayer games this is solved by having one player controlling the sidekick and one the main, but in the single player variation the AI would have to control one of them. The behavior of the AI would be very different from that of a real player as it would just follow a predefined, straightforward path through the level.

What makes this game interesting in a regards to time shifting is that the sidekick and main characters have no direct influence on each other, but they both manipulate the same levels at the same time. Because of this, it would be very interesting to study the viability of time shifting, instead of using AI, on one of these characters and study how that affects the user's immersion and enjoyment of the game.

11.4 Specialized Games

In our project and Real Racing 3, the one big commercial attempt on a time shifted game, the method has been to take an already existing multiplayer game type and adding time shifting to it. These games were designed to be played synchronously, and while similar, time shifting and synchronous have different features. More research into how we can utilize the unique features of time shifting can lead to new game types better suited for it. This is further discussed in Section 12.3 where we present a simple game concept to demonstrate how this could be done.

11.5 Improved AI Communication

In this project we created a small chat between questions with predefined emotes to emulate some of the communication done between players in real-time games. Saving the emotes used by the first player in a time-shifted game, and printing it for the next player, gives us a kind of communication, it assumes that a person would use the same emote independently of what other players emote and how their scores relate to the others. As an example, a player who had the best score at a question might be expected to use another emote than one that the 3rd best score even if the actual value of the score was the same. Since this is a plausible situation when playing in a time-shifted fashion, improvements to the AI to adapt the emotes used could be beneficial to the experience. This does however raise the question about how and when to do this. Correctly predicting when a player would use different emotes might prove to be a very difficult task, and you also have to consider that when you change someone`s emote, it is no longer the time-shifted players comment, but the comment of an AI based on that player. Possible solutions to this could exist, for example the emotes could be picked to avoid situations where they might be wrong when used in a time-shifted game. An emote like "YES, first place!" can be wrong in one game, and correct in another and thus creating this problem, but "YES, great score!" would not.

Another alternative would be to have dynamic emotes that adapt to the scores " YES, happy with [Placement]". More research into how this can be done and how it affects the user experience is necessary to understand the consequences of these and other possible solutions.

12 Future of Time-shifting

Time-shifting in games is still very new, and few attempts on creating content using it exist. Only one big commercial attempt has been tried when Real Racing 3 tried to change the landscape of multiplayer for racing games with bots based on the profiles of the players. This attempt was widely criticized [17] and was perhaps too ambitious. In this chapter we will discuss what we think could be situations where time-shifting could improve the gaming experience. In Section 12.1 we discuss mobile games in general, in Section 12.2 we how AI can be used in time-shifting and in 12.3 we give an example of how time-shifting can be used in a new type of game genre.

12.1 Mobile Games

We think that the real potential for time-shifted games lies in the mobile platforms where it can create a good multiplayer experience when it would otherwise would be difficult to do so. Mobile games are often played on the go, where you play a game while you have a spare moment between other activities. A typical example of this would be while waiting for the bus. This is often referred to as "dip in, dip out"[1] gaming where the sessions are short and happen at seemingly random times. For real-time multiplayer with friends to work in this situation these "random timings" where they want to play must happen at the same time for all the involved players. If they are indeed random, this is unlikely to happen.

This is in our opinion where time-shifted games have its real potential, by recreating the feeling of real-time multiplayer in "dip in, dip out" games. If we look at the one attempt of using time-shifting in a commercial game, Real Racing 3, they tried to replace the original real-time multiplayer with time-shifting. This was criticized by many fans of the game, demanding that they added in a traditional multiplayer option as well. This was done in the 2.0 update for the game in December [6] where they allowed up to four players to compete against each other in real-time.

We think this is an important sign that time-shifting should not be seen as a replacement for real-time multiplayer. Even if you mimic players perfectly, the experience of being in the same room while playing against your friends can not be replaced. Instead time-shifting should aim to find its place between real-time and single player games. Recreating some of the positive aspects of real-time while retaining the convenience of single player.

12.2 AI Creation

While not its original intention, we discovered while creating our application that time-shifting can be an excellent method for creating realistic AIs for certain game types. Since time-shiftinging a player consists of creating an AI that mimics the players behavior, time-shifting might have some potential in creating AIs that are realistic. In Section 5.1 Methods of Creating Time-shifted Behavior, we discuss two different methods for creating time-shifted AIs and while the intention usually is to allow friends to play against each other at convenient times, these AIs can just as well be used against other players.

We think that this could be particularly useful in games where the Stored Input Method is viable. In these types of games you can create AIs that mimic real players so well that they become indistinguishable from real players

12.3 New Genres of Games

As mentioned in Section 3.3 Time-shifted Games - Real Racing 3, the current attempts of using time-shifting has been to take an already popular genre and using time-shifting on it. In the case of Real Racing 3 the traditional multiplayer was replaced by a time-shifted one. Like we discussed in Section 12.1, this is not a good way to utilize time-shifting. Instead game creators should try to come up with new games and genres that can turn the nature of time-shifting into a positive part of the game.

Because of the novelty of time-shifting in games, it is hard to imagine how, or even if this can be successfully done, but we think that it is an area worth exploring.

To exemplify what we mean by utilizing the nature of time-shifting positively, we have created this simple game concept.

The game is a standard platformer where there are two players, one is a guide and the other a normal player. The guide will be a time-shifted player. The overall game is dark with light only circling the two players. The guide's role is to show the player how to get through a level in one piece. The player's goal will be to follow what the guide shows her, while simultaneously being a guide for the potential new players.

When the game is finished the player can send the game to a new friend. In the new play through the previous player, the sender, will be the guide while the new player, the receiver, will try to follow her guidance. After the new player is done with her play through she can send it to another friend, and so on. While this game is overly simple, we hope it will serve as a eye opener for how time-shifting can be a positive part of a game and not just a substitution for real-time multiplayer.

Part VI

Appendix

Appendix A - User Data

Raw Data

Gender	Age	What kind of connectivity did you use during the test?	How many hours do you spend playing mobile games per week?	What genre of mobile of games do you play?
Male	25	4G	0	
Male	24	WiFi	2	chess
Male	25	WiFi	1	Puzzle, Trivia/Quiz
Male	25	4G	2	Strategy
Male	23	WiFi	1	Casual, Puzzle, Strategy
Female	24	WiFi	1	Puzzle
Male	24	4G	0	Arcade, Trivia/Quiz
Male	24	WiFi	0	Trivia/Quiz
Male	23	4G	2	Puzzle

Male	25	WiFi	1	Arcade, Casual, Puzzle, Trivia/Quiz
Male	32	WiFi	0	Strategy
Male	27	WiFi	10	Card
Male	22	WiFi	0	Strategy

Table A.1: General user data.

Opponent Classification

Ole	**Dole**	**Doffen**	**Mikke**	**Mini**	**Pluto**	**Langbein**	**Donald**	**Dolly**	**Skrue**
AI	AI	AI	Human	AI	Uncertain	Uncertain	AI	Uncertain	Me
Human	Human	Human	Human	AI	Human	Human	Human	Human	Human
Uncert	Human	Uncert	Human	Me	Human	Human	Human	Uncert	Uncert

ain		ain						ain	ain
Human	AI	Me	Human	AI	AI	AI	AI	Human	AI
AI	AI	AI	AI	AI	AI	AI	Me	Human	AI
AI	AI	Uncertain	Uncertain	Uncertain	Me	Human	Human	Uncertain	AI
Uncerta	Me	AI	AI	AI	Uncerta	Uncerta	Uncerta	Uncerta	AI

in					in	in	in	in	
Human	Uncertain	Uncertain	Uncertain	Human	Uncertain	Me	Uncertain	Uncertain	Uncertain
Uncertain	Uncertain	Uncertain	Me	Human	Uncertain	Uncertain	Uncertain	Uncertain	Uncertain
Me	Uncertain	AI	Uncertain	Uncertain	AI	Human	AI	Human	Uncertain

Uncertain	AI	Uncertain	Uncertain	Uncertain	Human	Uncertain	Uncertain	Me	Uncertain
AI	AI	Uncertain	Human	Uncertain	AI	Me	Human	Uncertain	Uncertain
AI	Uncertain	Uncertain	Uncertain	Me	Uncertain	Uncertain	Uncertain	Uncertain	AI

Table A.2: Opponent classification.

It was easy to distinguish humans from AIs.	**How did you distinguish between humans and AIs?**	**Do you have any ideas on how to improve time-shifting in games?**
Agree	They taunted/talked a lot	Don't make the AIs so eager
Neutral	difficault	no
Disagree	Pure instinct	
Strongly agree	I remember	
Disagree	Random chat messages	
Strongly disagree	By guessing	
Strongly agree		
Disagree	First to ready up	
Strongly disagree	Didnt heff a chance!	
Disagree	Virket som ai kanksje var litt delayed	No
Disagree	Jeg gjettet ut i fra scoor. og hvem som satt seg som "klar" først	

Strongly disagree	At the last milliseconds all the bots vote and you can see it, though I hadnt had a paper to write it down and my memory is a mess. Finally I just thought that the bots were the best players on my quiz.	Didn't know it existed until now.
Strongly agree	Instant answers/responses	no

Table A.3: Difficulty distinguishing opponents.

Game experience

I find it annoying waiting for my opponents turn when playing turn based games, e.g. Wordfeud.	**I find difficult to find friends to play real-time games with, because we are not available at the same time.**	**The game experience is better when I know that the AI is based on real players.**	**I find it more interesting to play against time-shifted AI than regular AI.**	**I enjoy the game more when I play real-time multiplayer than time-shifted multiplayer.**
Neutral	Neutral	Strongly agree	Neutral	Neutral
Neutral	Agree	Agree	Agree	Strongly agree
Neutral	Disagree	Neutral	Neutral	Neutral
Strongly disagree	Neutral	Strongly agree	Agree	Agree
Agree	Disagree	Strongly agree	Strongly agree	Agree

Neutral	Neutral	Neutral	Strongly agree	Agree
Disagree	Agree	Agree	Strongly agree	Agree
Agree	Disagree	Agree	Agree	Strongly agree
Disagree	Disagree	Agree	Strongly agree	Neutral
Agree	Neutral	Agree	Neutral	Neutral
Neutral	Agree	Disagree	Neutral	Disagree
Disagree	Neutral	Disagree	Neutral	Strongly agree
Strongly agree	Disagree	Strongly agree	Neutral	Strongly agree

Table A.4: Game experience part I.

I am experienced with educational games.	**I would like time-shifted games to be part of my everyday school life.**	**I would like to use time-shifted games to prepare for tests/exams.**	**I would like time-shifted games to be used for homework.**
Disagree	Agree	Disagree	Strongly disagree
Disagree	Disagree	Disagree	Disagree
Disagree	Neutral	Agree	Agree
Disagree	Agree	Agree	Agree
Agree	Neutral	Agree	Agree
Agree	Neutral	Disagree	Neutral
Agree	Agree	Neutral	Agree
Neutral	Agree	Agree	Agree
Agree	Disagree	Strongly agree	Strongly agree
Disagree	Agree	Agree	Agree
Disagree	Neutral	Neutral	Disagree
Agree	Neutral	Strongly disagree	Strongly disagree
Neutral	Agree	Agree	Agree

Table A.5: Game experience part II.

System Usability Scale

I think I would like to use this system frequently.	I found the system unnecessarily complex.	I thought the system was easy to use.	I think that I would need the support of a technical person to be able to use this system.	I found the various functions in this system were well integrated.
Strongly disagree	Agree	Disagree	Disagree	Disagree
Disagree	Strongly disagree	Strongly agree	Strongly disagree	Agree
Neutral	Disagree	Disagree	Strongly disagree	Neutral
Neutral	Strongly disagree	Strongly agree	Strongly disagree	Neutral
Neutral	Disagree	Strongly agree	Strongly disagree	Agree
Disagree	Neutral	Agree	Strongly disagree	Neutral

Agree	Disagree	Agree	Strongly disagree	Neutral
Agree	Strongly disagree	Agree	Strongly disagree	Agree
Disagree	Disagree	Agree	Disagree	Disagree
Agree	Strongly disagree	Strongly agree	Strongly disagree	Agree
Neutral	Neutral	Neutral	Disagree	Agree
Disagree	Disagree	Agree	Disagree	Agree
Neutral	Neutral	Strongly agree	Strongly disagree	Neutral

Table A.6: System Usability Scale part I.

I thought there was too much inconsistency in this system.	**I would imagine that most people would learn to use this system very quickly.**	**I found the system very cumbersome to use.**	**I felt very confident using the system.**	**I needed to learn a lot of things before I could get going with this system.**
Neutral	Neutral	Neutral	Agree	Disagree
Neutral	Strongly agree	Strongly disagree	Strongly agree	Strongly disagree
Neutral	Agree	Disagree	Agree	Disagree
Strongly disagree	Strongly agree	Neutral	Neutral	Strongly disagree
Disagree	Agree	Strongly disagree	Agree	Disagree
Disagree	Neutral	Strongly disagree	Neutral	Strongly disagree
Neutral	Strongly agree	Disagree	Agree	Strongly disagree
Strongly disagree	Strongly agree	Strongly disagree	Agree	Strongly disagree

Agree	Agree	Neutral	Neutral	Disagree
Disagree	Strongly agree	Strongly disagree	Neutral	Strongly disagree
Neutral	Disagree	Strongly disagree	Neutral	Neutral
Neutral	Strongly agree	Neutral	Neutral	Neutral
Strongly disagree	Strongly agree	Neutral	Neutral	Strongly disagree

Table A.7: System Usability Scale part II.

Other

Final comments on time-shifting, the testing, the application, etc. ?
Easy to use
No
Fox error of timeshift has stoppes randomly. Other wise good
Jeg fant det vanskelig å skille time-shifted AI og realtime-spillere. I lobbyen så var det to-tre som var pålogget da jeg kom inn, men så dukket det plutselig opp to rett etterpå. Hvem av disse var time-shifted? Who knows!?
I don't know if the last ms bot voting was a trick if not, maybe you should emulate a normal one.
Worked fine other then the crash

Table A.8: Final comments.

Appendix B - Test Descriptions

B.1 First Test at NTNU

The first test was conducted on participants from the course TDT4240 Software Architecture. Unfortunately we underestimated how difficult it would be to attract students during the exam period, and only two of the course students signed up for the test. To make matters worse one of the participants didn`t show up for the test and so we ended up testing on only one person this day. It would obviously be very easy to separate time shifted and real-time players if you are the only participant so we, the researchers, stepped in and participated in the test with him. The participant was not told if both, one or neither of us where in the quiz with him to avoid him using this information to decide how many of the players where real time and time shifted.

B.2 Second Test at NTNU

Since it was proved difficult to attract participants from the Software Architecture course, we decided to look elsewhere as well. The second attempt to get participants was to contact other computer science master students. We managed to get two more participants for this testing session.

This session ran smoothly with no issues.

B.3 Test at UiO

The next test was conducted at the University in Oslo on four computer science students from that university. While installing the application we ran into an issue with Dropbox for android phones. Two of the participants had phones that would give them a 403-forbidden error when attempting to download the application through Dropbox. The error would only happen in certain web browsers, and could be resolved by deleting the Dropbox cookies in them, but since this was our first time experiencing this problem, we were unaware of this solution. Instead we solved the issue by transferring the apk file, which is the installation file for android application, directly from one of our computers.

In the pre-test, we discovered an issue where the quiz could crash if one of the participants didn`t answer. After this was discovered, we fixed the issue and uploaded the new version to Dropbox. Unfortunately, the application we had on the computer used in the testing did not have the updated version since we intended the participants to use the version on Dropbox. This lead to the two participants that got the application from the computer had a version where this issue had not been fixed, and at the 7th question one of the participants didn`t answer and the application crashed for these two participants. With only three questions left, we decided not to redo the test as we considered the 7 questions to be enough time to get a good impression about how time shifting would work. After the quiz, the participants were given the questionnaire.

B.4 Online Test

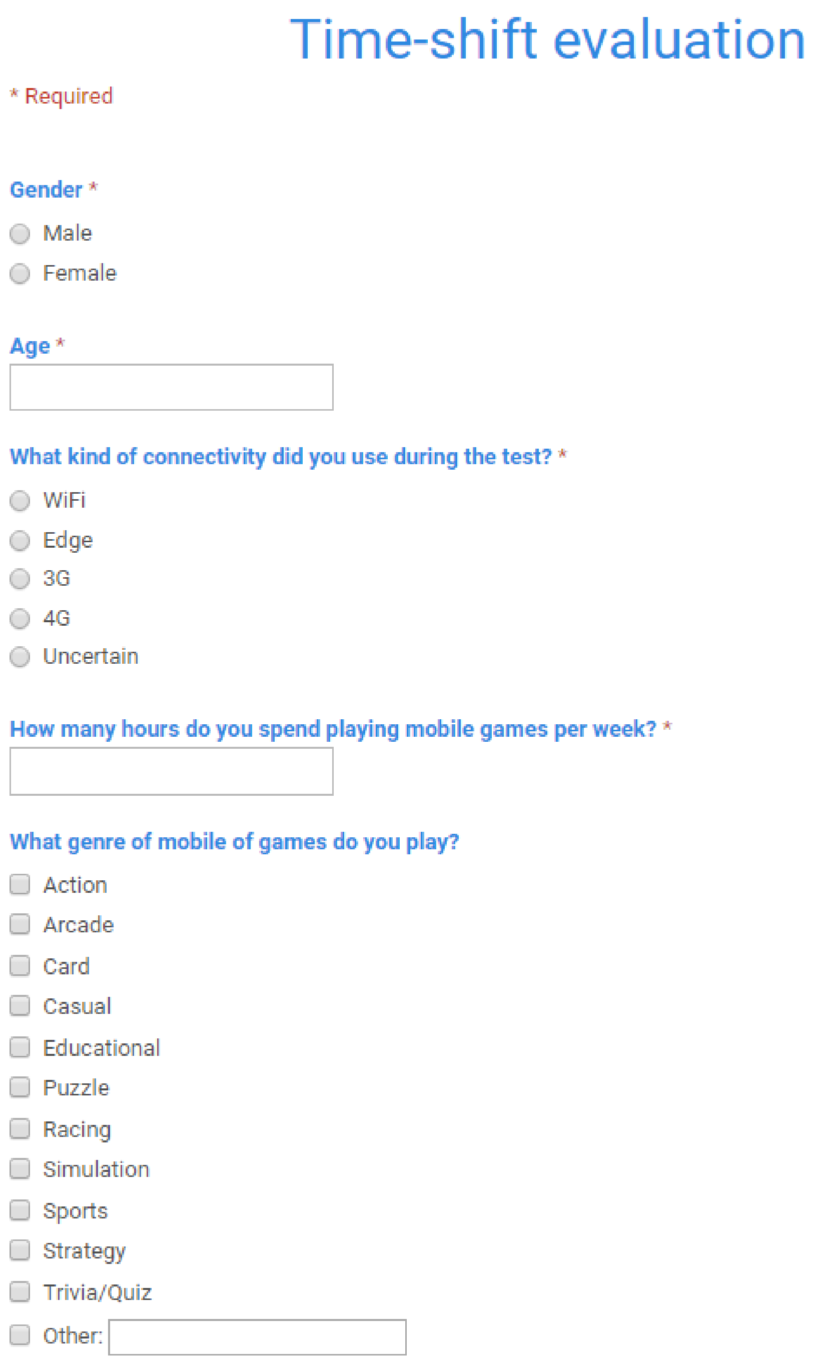
In our testing thus far, none of the participants had been able to efficiently distinguish between time-shifted and real-time players. So for the last test we tried to get participants that we thought would be well suited to discover differences. The two participants in this test was a full-time developer and a student who plays computer games competitively for one of the biggest teams in Norway. Our reason for testing on these participants was that we wanted to see what methods they would use to distinguish between players. Only a few of the participants we had tested so far had a method for trying to do this, and we hoped that the current participants would give us more insight into how exposing a time-shifted player could be done. They would also serve as a good test to see if it is as difficult to distinguish as our results so far has implied.

Unlike the other tests, this one was not performed at one location. The two participants performed the test from their home, with no contact with each other. Because of this, they had no way of knowing how many of their opponents were playing in real-time and how many that were time-shifted. This was also the first test were one of the participants used a tablet instead of a phone.

When starting the quiz, the tablet user encountered a crash and we were forced to restart the quiz. What caused this crash is difficult to know since he was not at the same location as us, but since he could not re-enter the application until the quiz was finished we speculate that it might be connected to the Parse channels we use for communication. Another possibility is that it had something to do with him using a tablet and not a phone. The development and testing was done on phones and while it theoretically shouldn't matter, it is a possibility to consider.

After restarting the quiz, everything ran smoothly until the finish and the participants were given the questionnaire to answer.

Appendix C - Questionnaire



Time-shift evaluation

* Required

Gender *

- ○ Male
- ○ Female

Age *

What kind of connectivity did you use during the test? *

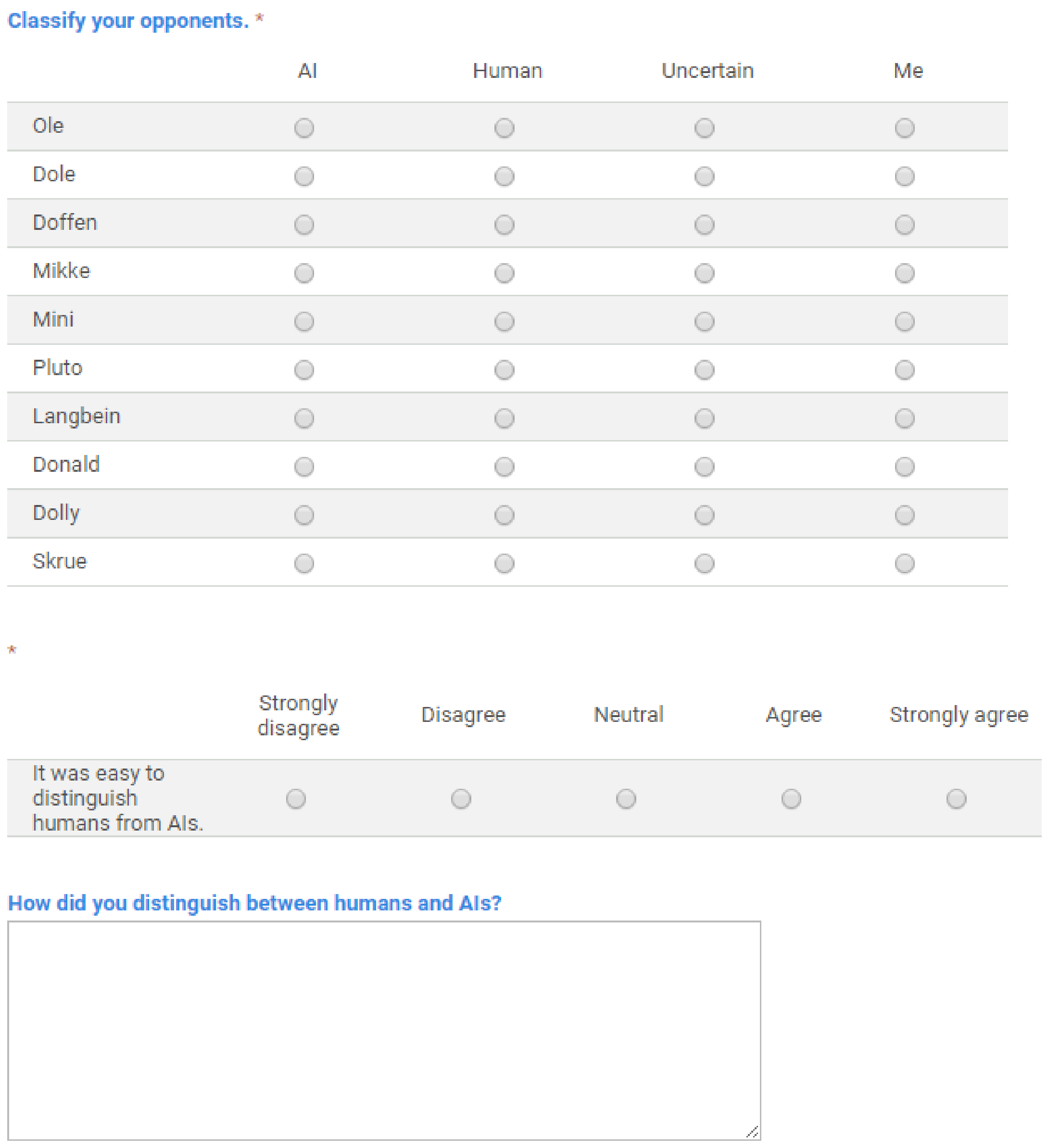
- ○ WiFi
- ○ Edge
- ○ 3G
- ○ 4G
- ○ Uncertain

How many hours do you spend playing mobile games per week? *

What genre of mobile of games do you play?

- ☐ Action
- ☐ Arcade
- ☐ Card
- ☐ Casual
- ☐ Educational
- ☐ Puzzle
- ☐ Racing
- ☐ Simulation
- ☐ Sports
- ☐ Strategy
- ☐ Trivia/Quiz
- ☐ Other:

Figure C. 1: Questionnaire part I.


Classify your opponents. *

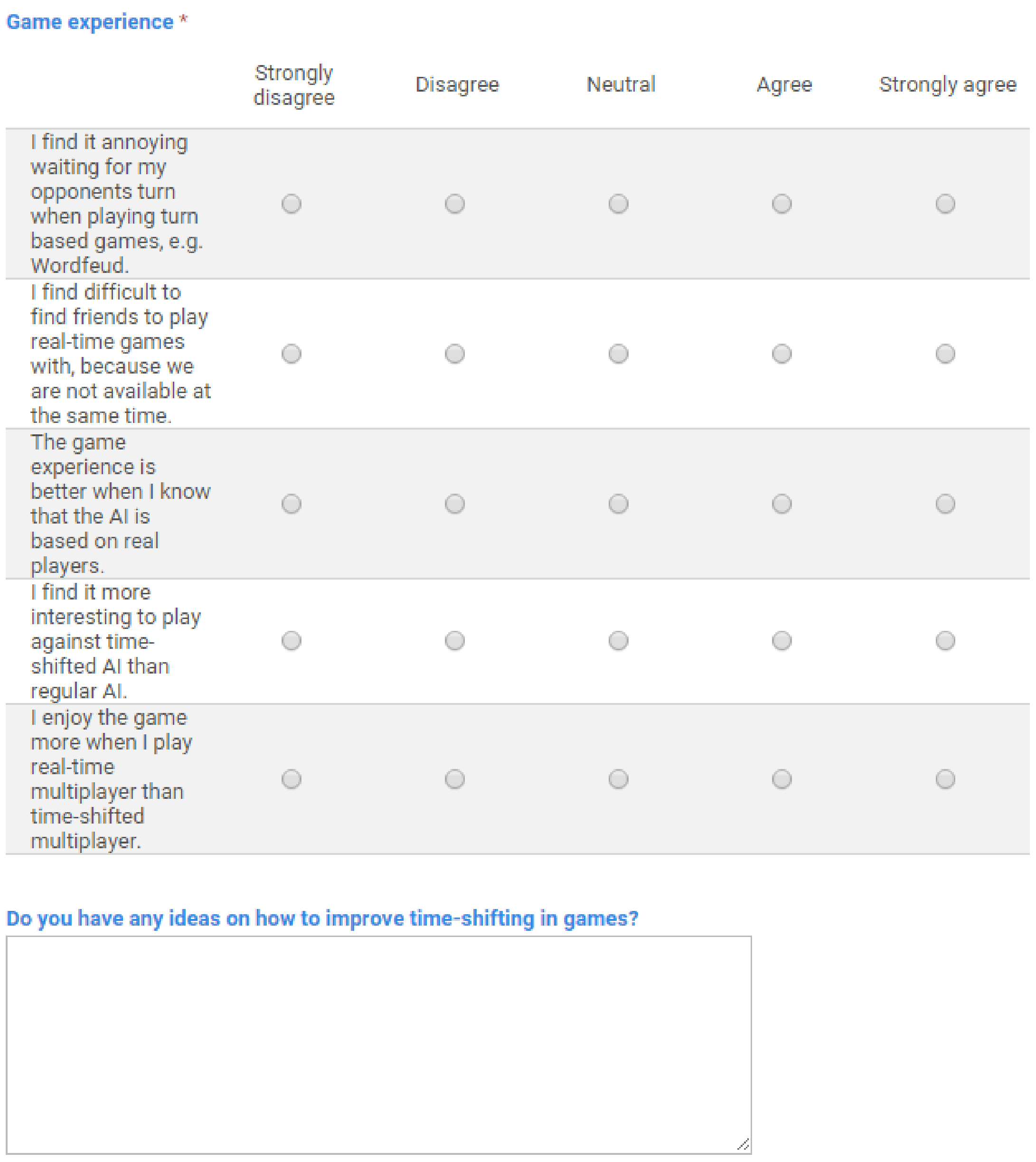
	AI	Human	Uncertain	Me
Ole	○	○	○	○
Dole	○	○	○	○
Doffen	○	○	○	○
Mikke	○	○	○	○
Mini	○	○	○	○
Pluto	○	○	○	○
Langbein	○	○	○	○
Donald	○	○	○	○
Dolly	○	○	○	○
Skrue	○	○	○	○

*

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
It was easy to distinguish humans from AIs.	○	○	○	○	○

How did you distinguish between humans and AIs?

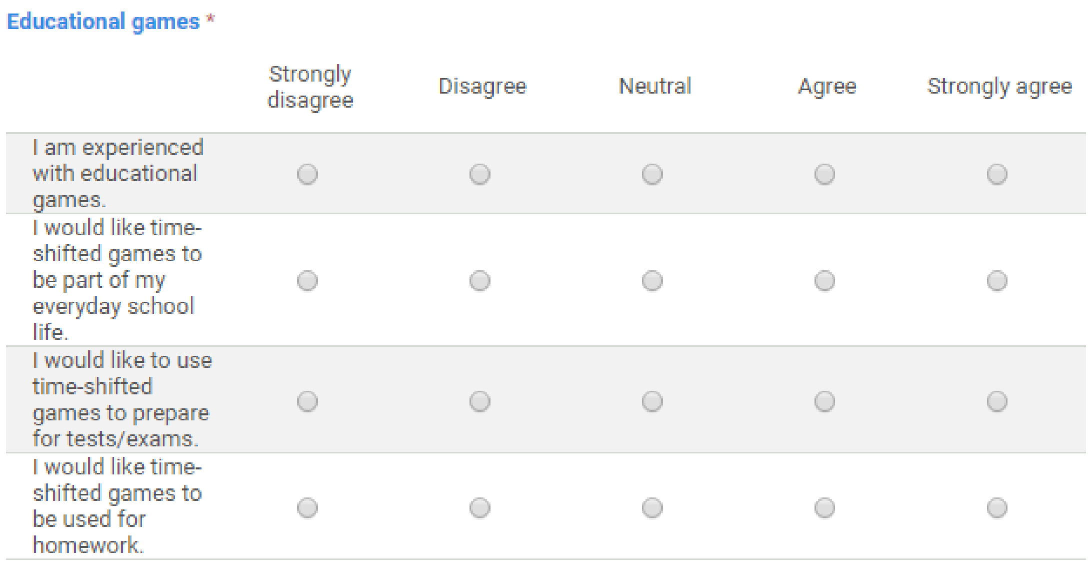
Figure C. 2: Questionnaire part II.


Game experience *

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
I find it annoying waiting for my opponents turn when playing turn based games, e.g. Wordfeud.	○	○	○	○	○
I find difficult to find friends to play real-time games with, because we are not available at the same time.	○	○	○	○	○
The game experience is better when I know that the AI is based on real players.	○	○	○	○	○
I find it more interesting to play against time-shifted AI than regular AI.	○	○	○	○	○
I enjoy the game more when I play real-time multiplayer than time-shifted multiplayer.	○	○	○	○	○

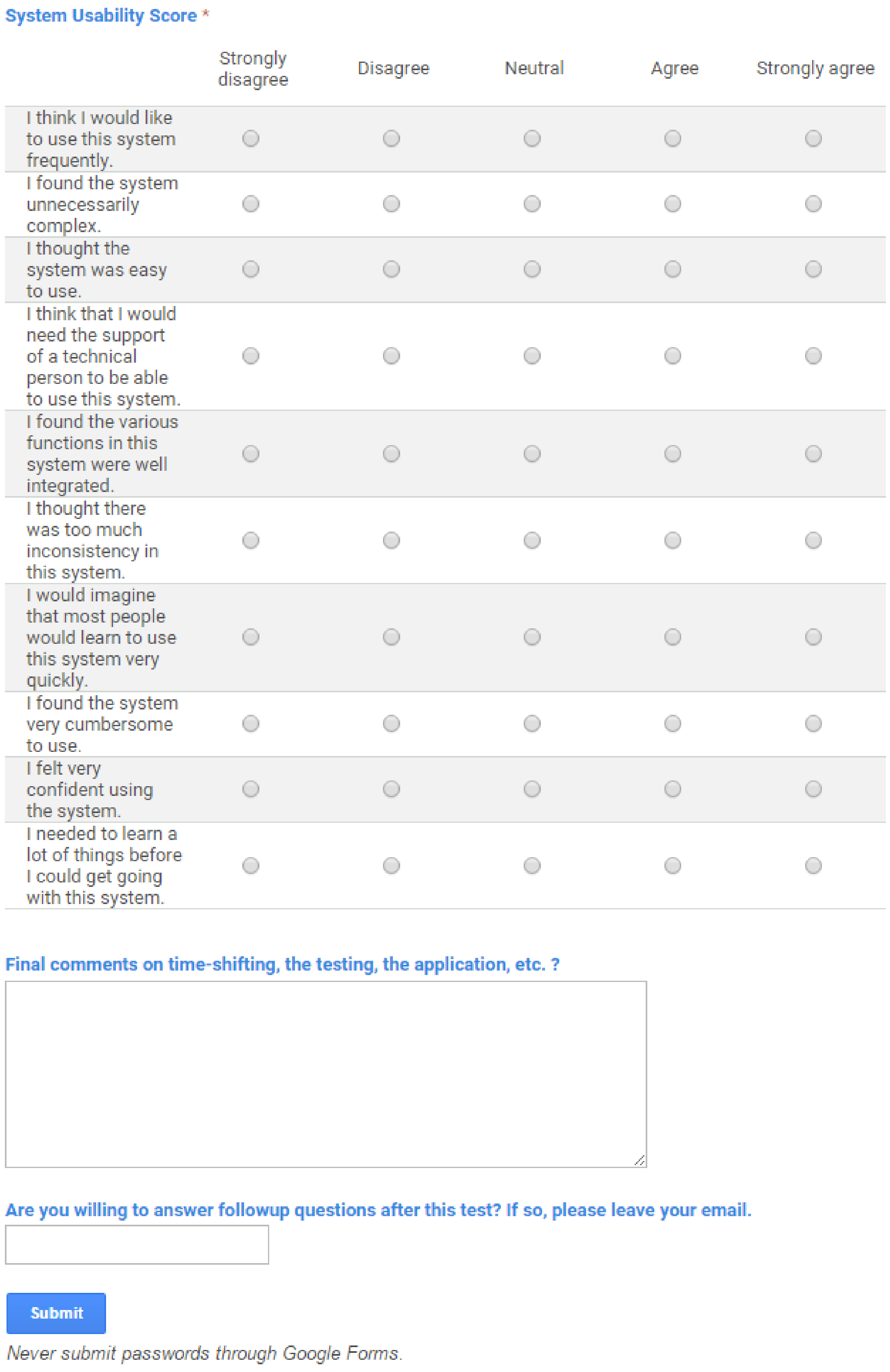
Do you have any ideas on how to improve time-shifting in games?

Figure C. 3: Questionnaire part III.


Educational games *

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
I am experienced with educational games.	○	○	○	○	○
I would like time-shifted games to be part of my everyday school life.	○	○	○	○	○
I would like to use time-shifted games to prepare for tests/exams.	○	○	○	○	○
I would like time-shifted games to be used for homework.	○	○	○	○	○

Figure C. 4: Questionnaire part IV.


System Usability Score *

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
I think I would like to use this system frequently.	○	○	○	○	○
I found the system unnecessarily complex.	○	○	○	○	○
I thought the system was easy to use.	○	○	○	○	○
I think that I would need the support of a technical person to be able to use this system.	○	○	○	○	○
I found the various functions in this system were well integrated.	○	○	○	○	○
I thought there was too much inconsistency in this system.	○	○	○	○	○
I would imagine that most people would learn to use this system very quickly.	○	○	○	○	○
I found the system very cumbersome to use.	○	○	○	○	○
I felt very confident using the system.	○	○	○	○	○
I needed to learn a lot of things before I could get going with this system.	○	○	○	○	○

Final comments on time-shifting, the testing, the application, etc. ?

Are you willing to answer followup questions after this test? If so, please leave your email.

Never submit passwords through Google Forms.

Figure C. 5: Questionnaire part V.

References

[1] K. Corti, "The opportunities in mobile gaming are in asynchronous social multiplayer games", Gamasutra, 2012. [Blog] Retrieved from http://www.gamasutra.com/blogs/KevinCorti/20121123/182151/The_opportunities_in_mobile_gaming_are_in_asynchronous_social_multiplayer_games.php/.

[2] App Meter. Wordfeud. [Statistics] Retrieved from http://www.appmtr.com/facebook/app/160231364042233-wordfeud/.

[3] Think Gaming. Clash of Clans. [Statistics] Retrieved from http://thinkgaming.com/app-sales-data/1/clash-of-clans/.

[4] P. Norvig, S. Russell, "Artificial Intelligence: A Modern Approach", 2009.

[5] M. Zyda, "From Visual Simulation to Virtual Reality to Games", Computer vol. 38, no. 9, pp. 25-32, 2005.

[6] Toucharcade. [Webpage] Retrieved from http://toucharcade.com/2013/12/17/real-racing-3-updated-with-real-time-online-multiplayer-and-two-new-supercars/.

[7] Usability.gov. [Homepage] Retrieved from http://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html/.

[8] M. Overmars, "A Brief History of Computer Games", 2012.

[9] M. Csikzentmihaly, "Flow: The psychology of optimal experience", 1991.

[10] P. Sweetser, P. Wyeth, "GameFlow: A Model for Evaluating Player Enjoyment in Game", Computers in Entertainment - Theoretical and Practical Computer Applications in Entertainment vol. 3, no. 3, Article A3, 2005.

[11] Kahoot!. [Homepage] Retrieved from https://getkahoot.com/.

[12] Kahoot!. [Statistics] Retrieved from https://play.google.com/store/apps/details?id=no.mobitroll.kahoot.android&hl=en/.

[13] Kahoot!, "Kahoot! - LAUNCHedu finalists at SXSWedu". [Blog] Retrieved from http://blog.getkahoot.com/post/47531504634/kahoot-launchedu-finalists-at-sxswedu/.

[14] Wordfeud. [Homepage] Retrieved from http://www.wordfeud.com/.

[15] S. Sarkar, "The tech behind Real Racing 3's time-shifted asynchronous multiplayer", Polygon, 2013. [Blog] Retrieved from http://www.polygon.com/2013/2/15/3989990/real-racing-3-preview-time-shifted-multiplayer-tech/.

[16] Josefine Skolehjelp: Talljakten. [Webpage] Retrieved from http://www.kreagames.no/josefine-matematikk-talljakten/.

[17] R. Rich, "Real Racing 3 Review", 148 Apps, 2013. [Review] Retrieved from http://www.148apps.com/reviews/real-racing-3-review/.

[18] Play Magnus. [Homepage] Retrieved from http://magnuscarlsen.com/playmagnus.

[19] A.I. Wang, T. Øfsdahl, O.K. Mørch-Storstein, "An Evaluation of a Mobile Game Concept for Lectures", Software Engineering Education and Training pp. 197-204, 2008.

[20] U. Eikeseth, "Her jublar elevane over samfunnsfagprøven", NRK, 2012. Retrieved from http://www.nrk.no/viten/dataspel-i-timen-testar-elevane-1.8196102.

[21] FourSquare. [Homepage] Retrieved from http://www.foursquare.com/.

[22] Metacritic. [Homepage] Retrieved from http://www.metacritic.com/.

[23] V. Chapman, R. Hunicke, "AI for Dynamic Difficulty Adjustment In Games", Proceedings of the Challenges in Game AI Workshop, Nineteenth National Conference on Artificial Intelligence, 2004.

[24] J. Nielsen, J. Olseng, "Game Based Learning: The Knowledge Challenge game", Depth project report, NTNU, December 2014.

[25] A. J. Stapleton, "Serious games: Serious opportunities", 2004.

[26] S. Deterding, M. Sicart, L. Nacke, K. O'Hara, D. Dixon, "Gamification. Using game-design elements in non-gaming contexts", CHI'11 Extended Abstracts on Human Factors in Computing Systems, pp. 2425-2428, 2011.

[27] V. R. Basili, "The experimental paradigm in software engineering", In Proceedings of the International Workshop on Experimental Software Engineering Issues: Critical Assessment and Future Directions, pp. 3–12, 1993.

[28] Parse. [Homepage] Retrieved from http://www.parse.com/.

[29] Parse JavaScript Guide. [Documentation] Retrieved from https://parse.com/docs/js_guide/.

[30] Backbone.js. [Homepage] Retrieved from http://www.backbonejs.org/.

[31] jQuery. [Homepage] Retrieved from http://www.jquery.com/.

[32] Bootstrap. [Homepage] Retrieved from http://www.getbootstrap.com/.

[33] G. Anderson,"Norways Kahoot! is an edtech rocket with 900.000 signups last week", Artic Startup, 2014. Retrieved from http://arcticstartup.com/2014/10/27/norways-kahoot-is-an-edtech-rocket-with-900000-signups-last-week/.

[34] CSS. [Homepage] Retrieved from http://www.w3.org/Style/CSS/Overview.en.html/.

[35] HTML5. [Homepage] Retrieved from http://www.w3.org/TR/html5/.

[36] JavaScript. [Documentation] Retrieved from http://www.similartech.com/categories/javascript

[37] Habit RPG. [Homepage] Retrieved from https://habitrpg.com/.

[38] Supercharged. [Homepage] Retrieved from http://education.mit.edu/projects/supercharged/.

[39] vHealthCare. [Homepage] Retrieved from http://www.breakawaygames.com/.

[40] Engadget. [Homepage] Retrieved from http://www.engadget.com/2011/10/20/the-gadget-show-builds-an-fps-simulator-that-shoots-back-video/.

[41] Elo rating. [Article] Retrieved from http://gobase.org/studying/articles/elo/.

[42] Blog about Stack overflow. [Blog] Retrieved from http://michael.richter.name/blogs/why-i-no-longer-contribute-to-stackoverflow/.

[43] Blog about Dropbox error. [Blog] Retrieved from https://trustiko.com/error-403-in-dropbox-when-you-login-how-to-fix-the-problem/.

[44] Blog about Pac-man AI. [Blog] Retrieved from http://gameinternals.com/post/2072558330/understanding-pac-man-ghost-behavior/.

[45] Blog about the history of AI. [Blog] Retrieved from https://sites.google.com/site/myangelcafe/articles/history_ai/.

[46] V. R. Basili, G. Caldiera, H. D. Rombach, "The Goal Question Metric Approach", 1994.