NTNU – Trondheim
Norwegian University of
Science and Technology

Gamifying Schools: Utilising Game Concepts to Enhance Learning

Nicolay H Hvidsten
Severin Sverdvik

Master of Science in Computer Science
Submission date: June 2013
Supervisor: Alf Inge Wang, IDI

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

Problem Description

Schools face unique problems in motivating students and keeping them engaged. The gaming industry is booming, as well as attracting huge audiences among the target group, and the use of e-learning and video games in education been proven numerous times to be superior to old techniques in therms of learning effectiveness. The focus of this project is to synthesize the potential of gamification in education by creating a web platform with game elements for school use.

Abstract

Engaging elementary school students in various school subjects has always been a difficult task for teachers. In recent years, research on using computer games for educational purposes has shown that situated learning can effectively engage middle school students in critical thinking about authentic scenarios. In this thesis we investigate the area of utilising games for educational purposes, and propose a web platform with game elements for school use. Furthermore, we develop an application prototype containing the key features of the proposed application, to act as a proof of concept. We believe the proposed application may contribute to more engaging and enjoyable learning for elementary school students, and as a result, increase the potential for true learning. It could provide several benefits for teachers as well, allowing them to translate their knowledge into tangible game artefacts to customise their curricula tailored to their strengths. In addition, it could also provide teachers with tools to assess the class in a more accurate way, and thus spend their time more wisely, leading to a more efficient use of the teacher as a mentor and guide, and potentially increasing student-teacher interaction.

Sammendrag

Å engasjere grunnskoleelever i ulike fag har alltid vært en vanskelig oppgave for lærere. I de senere årene har forskning på bruk av dataspill for pedagogiske formål vist at bruk av dataspill til læring kan effektivt engasjere ungdomsskoleelever i kritisk tenkning om autentiske situasjoner. I denne oppgaven undersøker vi utnytting av spill for pedagogiske formål, og foreslår en web-plattform med spillelementer for bruk i grunnskolen. Videre utvikler vi en prototype som inneholder de viktigste funksjonene i den foreslåtte webapplikasjonen, for å demonstrere konseptet. Vi mener den foreslåtte applikasjonen kan bidra til mer engasjerende og underholdende læring for grunnskoleelever, og som et resultat, øker potensialet for bra læring. I tillegg kan det gi læreren et godt verktøy å vurdere klassen på en mer nøyaktig måte, som kan føre til en mer effektiv bruk av læreren.

Contents

Part I

Introduction

Chapter 1

Introduction

Disclaimer

To avoid confusion when reading this thesis, we will clarify straight away that we will use various terms to refer to the software application that is the main focus of this paper. Said application will be a gamified web application, utilizing game mechanisms on top of the interactive and ubiquitous web platform - and will be referred to both in the context of a game and a web application, as it, in a way, transcends both these definitions. We will also use the terms video game and game interchangeably.

1.1 Context

The gaming industry has grown into one of the world's leading entertainment industries [34], and video games are widely played by young people of both genders [68]. The use of video games in adolescents is so common that the average American teenager will have played over 10,000 hours of video games by the age of 21 [45], which is, per author Malcolm Gladwell, the amount of time required to achieve expertise in a given subject matter.

Author Jane McGonigal claims that harnessing the potential from this substantial time investment could provide benefits in a wide variety of disciplines including (but not limited to) business management, happiness engineering, medical recovery and education [68] .

As the term *gamification*[1] becomes increasingly widespread, and the process of gamification an important part of everyday life, the potential of utilizing

[1]The discipline of introducing game mechanics to non-game domains [51].

video games in new settings has become a very hotly discussed topic, with special focus on educational benefits.

While it is still a popular belief that video games provide no actual benefit other than pure entertainment, and some studies claim that they are harmful [4], the application of video games in new settings has garnered many positive studies and innovative projects, such as Becta[2], Edutopia[3] and Khan Academy[4] - to name a few.

1.2 Motivation

As the authors of this project have both garnered some experience with video games, playing from a young age as well as designing and developing, we feel that the potential of this medium is not being fully utilized. There are several occasions where a video game has served both as motivation and a teacher to us, but we feel that the available applications do not facilitate the learning of skills and content that is interesting to the area of education (although games such as Dragon Box [24] and applications like Khan Academy [57] are paving the way).

To accommodate this intuition, we set out explore possibility of including a pervasive educational game in elementary school in our pre-thesis project "Pervasive Learning Environment" [52]. After an extensive study of current literature and available products, we found that there is a lot of academic work supporting the use of games for educational purposes, but there is still a great gap between the opportunities of e-learning and the tools that are currently available to schools. These findings encouraged us to continue on that path for this thesis, and attempt to create a gamified learning platform for elementary schools that could illustrate the opportunities of such an approach, and the lack of similar tools in the current educational model.

We learned that the emotional state in humans when playing games and learning well share many commonalities; the learner must be engaged and immersed in the activity, and a meaningful context should be provided. Also, the current educational model suffers from a one-size-fits-all approach where students are bored and learning suffer as a result. We were able to conclude that the primary goal of the previously mentioned approach must be to make

[2]http://becta.org.uk
[3]http://www.edutopia.org/technology-integration
[4]http://khanacademy.org

an eventual application engaging and fun, without sacrificing focus on learning effect. Furthermore, we concluded that an ideal solution for the school system would be a web-based platform with game elements, where teachers could easily structure instruction/problem-sets and tasks, and monitor class performance in order to highlight problem areas.

The conclusion of our pre-thesis project and further exploration of the idea of creating an application utilizing the principles discussed in said study, has convinced us that there is merit to the approach of implementing game mechanics in the educational plane. Even though the concept of gamification is still in its infancy, there are numerous applications of this concept in a variety of disciplines. The area of using video games for educational benefits is currently receiving a lot of attention, which is a highly motivational factor for this thesis.

1.3 Project Goals

The goal of this project is to design and develop an application prototype that utilizes game mechanics and pedagogical principles to facilitate augmented learning. While it is doubtful that we will be able to fully design and develop a functional and engaging application in the limited timespan that is the scope of this project, we intend for our prototype to serve as a *proof of concept*.

In addition to the actual application prototype, we aspire to document our every design choice and provide a thorough literature study to support our claims and said choices.

Finally we wish to contribute to the discussion of gamification within education, both through our prototype and the conclusions we draw from our literature study. In this way we intend to further the research into this exciting area.

1.4 Project Structure

We have divided the thesis into six chapters, ordered chronologically by where they belong in the process, though the different parts of the process do overlap.

1 Introduction An introduction to the context in which we approach the thesis, the motivation for the project and the project goals.

2 Research approach This part presents the planning of the project, the research questions and how we landed on those, as well as our methodology for evaluating our findings and results.

3 Literature study This part describes the current state of the art in the area of learning games and research on the area of learning, in order to pinpoint the elements that are required to facilitate effective learning.

4 Design and development The design and development part describes the process we went through in order to arrive at the concept for our prototype, as well as an overview of the system design of our implementation.

5 Results This part presents the results of development phase and eventual tests conducted.

6 Discussion This part discusses the results from our prototype in the context of our findings in the literature study.

7 Conclusion Finally, we present the conclusions drawn from the results of the prototype, and highlight further work that could be useful to this project.

Part II

Research Approach

Chapter 2

Research questions

In order to structure our approach to this thesis, we have formulated a set of research questions. The research questions are based on what we learned in the pre-thesis project [52]. In said project, we proposed to create a *gamified* digital learning platform for elementary school students.

The term *gamified* is drawn from the concept gamification, which has been defined as "a process of enhancing a service with affordances for gameful experiences in order to support user's overall value creation" by Huotari and Hamari [51]. While this particular definition is written in a marketing context, it applies equally well to learning and what we wish to achieve. In our case, the elements that are introduced in order to *gamify* the application includes features such as rewards, competition and fun graphics. More on gamification in Chapter 5.

RQ1	Games and learning
RQ1.1	Which video game features contribute to learning?
RQ1.2	What are the theoretical advantages of gamified learning platform?
RQ1.3	What is the best approach to implement such a platform?
RQ2	Earning acceptance from schools
RQ2.1	How can teachers be given incentive to utilise such an application?
RQ2.2	How can students be given incentive to utilise such an application?
RQ2.3	How can such an application be used to complement current educational system?

Table 2.1: Research Questions

We learned in our pre-thesis study that games can be used to improve learning, and we want to further investigate into this subject in order to dissect the process and identify which features can be leveraged in schools. We wish utilise the knowledge from that theoretical study and design a complete proposition of a gamified web application for use in school.

Furthermore we wish to convert a subset of said design into a working prototype. The intention is to perform a field test using the prototype in order to confirm or refute the effect of key features of the proposed application.

2.1 Research Methods

There are many different approaches and methodologies one can use when conducting research. In software engineering, three methods stand out according to Basili [8]; the engineering method, the empirical method and the mathematical method.

The engineering method: In the engineering method, researchers observe solutions that already exist and propose a better solution based on what they learn from the existing work. The new solution is then tested and analysed to improve upon it. This process is repeated until no further improvements seem possible. The model assumes that there are existing solutions available for researchers to analyse, be it a product or a process. "A crucial part of this method is careful analysis and measurement" [8].

The empirical model: Researchers propose a new model that is not necessarily based on an existing model. Statistical and/or qualitative methods are developed in order to assess the new model. The empirical method can be used on new solutions to find out if they are better than the existing solutions.

The mathematical model: In the mathematical model, a formal theory or set of axioms is proposed. A theory is developed and results are compared to empirical results. This is an analytical approach that does not require an experimental design, but provides a framework for developing and understanding models.

In this thesis, we will be making use of both the engineering method and the empirical model. The engineering method will be used in the sense that we will look at existing solutions and learn from them, but rather than doing careful analysis and measurements to improve upon those solutions, we will take the main ideas and tailor them to work in with our initial proposition [52] and build our own prototype. The prototype we land on will be subject

to improvements using the engineering method as well.
The empirical model will be used to measure the impact of our prototype; in order for us to be able to measure the usefulness of our application, we will need to conduct tests with the target users and develop quantitative and qualitative methods to assess the results of those tests.

2.2 Evaluation approaches

There are a great number of ways to evaluate a software product such as performance testing, security testing, usability testing, integration testing, field studies and so forth [109]. When evaluating a finished product, all types of tests are important to a varying degree. When evaluating a research prototype, on the other hand, one must focus on the types of evaluation methods that will provide feedback on the points that the researchers set out to investigate. In addition, one must conduct tests to ensure that the system acts as intended and does not interfere with the results of the main evaluation.

Chapter 3

Process

This chapter describes our planned process for this thesis and the reasoning behind our choices. This includes planning, literature study, design, development and evaluation.

3.1 Literature Study

In order for us make sure we are able to attack the problem with emphasis on the right points, using a suitable approach and suitable technology, we need to conduct a literature review of the current state of affairs within the field of games and learning, as well as relevant technology. We will mainly look into recent articles and research on the topic of games and learning, and study documentation of relevant technology in order to be able to select an appropriate platform for our application prototype. We will also look into key aspects of learning so that we are qualified to tailor a well defined evaluation approach.

3.2 Design and Development

To be able to search for answers to the research questions of this thesis, we will need an application with which we can conduct tests. Hence, we will design and develop an application prototype, using appropriate technology according to the findings in our literature study. The design and development phase will be the most demanding phase. Designing a new application from scratch can be very time-consuming [39]. For this reason, it is important to lay a solid foundation in the form of concrete design plans. Although the initial planning phase is important, we will not spend too much time on it, because "design is a complex, iterative process. The initial design solution will likely be wrong and certainly not optimal" [39]. Hence, we must allow time to go back and improve; software development should be an agile iterative process. That said, we do by no means have the time to create an optimal solution, but we will aim to make the application as good as possible with the time at hand. The next obvious pitfall and time-consuming aspect is avoiding errors. "Error removal is the most time-consuming phase of the life cycle [39]. Often times, introducing features and functionality means breaking something else. Therefore, we will select a development process that allows for sufficient testing, while being reasonably fast at the time time.

3.3 Evaluation Approach

Since the research goals of this thesis are not particularly application-specific, we need to select forms of evaluation with care, in order to be able to measure the factors that may aid in providing answers to the research questions. As we identified in the pre-thesis study [52], abstract concepts like fun and immersion are user reactions we seek to induce with the application prototype, and these can only be qualitatively measured through some types of user tests, like those we touched upon in the previous section. In addition, early-phase testing must take place to ensure that the prototype meets a certain standard, and that the time spent on more elaborate testing is not wasted due to faults or insufficiencies in the application prototype. In addition, it is important that results gathered are interpreted in a relevant fashion, and that tests are designed with a focus on the types of questions we want answered.

For these reasons we will write automated tests to test the basic functionality of the prototype. Furthermore we will perform basic usability testing in order to prime the prototype before we declare it ready for more elaborate tests. Ideally, we would take the prototype out into the field and test it on target users, children in primary and elementary school, and their teachers. Should we not be able to conduct field tests with real end users, we will perform a theoretical evaluation based on the knowledge we obtain in the literature study.

Part III

Literature Study

Introduction

There has been much academical interest in utilizing technology and games for educational purposes. Studies have shown time and time again that combining games and learning pose several benefits [58], while those who have the power to put such concepts into use claim that the quality of the games available is too poor [63]. In this chapter, we will investigate the current state of the art in the area of learning games and research on the area of learning, in order to pinpoint the elements that are required to facilitate effective learning. In this, we hope to reveal reasons why current educational games are considered sub par, and identify elements that can help us to propose a platform that could be considered useful by the target user group. In addition, we will look into the different technology options, that could facilitate our ideas, in order to create a prototype that captures our vision.

Chapter 4

Technology

Since we have chosen a very popular field in software development, the creation of web-based applications, there are a plethora of different technologies available. We will discuss which would be most suitable, and then provide our rationale for choosing the technologies we wish to make us of in our project, relating back to our initial requirement specification.

4.1 Web application as a platform

In our pre-thesis project [52], we proposed to deploy the application as a web application. In this section we will highlight the technical advantages of such a decision.

The application requires a means of content distribution and synchronization, and as such a client-server model is preferred. In the case of a computationally inexpensive application like the one proposed, a centralized approach where a server is in charge of backend logic and storing data, and clients request content from that server is an obvious choice. Having one main entity that looks after everything, rather than having many different entities trying to communicate in order to stay synchronized, greatly simplifies development and error recovery. The disadvantages of a centralized approach include that it requires connectivity to the network of choice(usually the Internet), both the client and the server must have sufficient bandwidth, and the computational demands on the server are much higher. The application will not require much resources and the Internet is usually highly available in Norway - especially in school. Hence, we do not consider these issues to create any big concerns for our application.

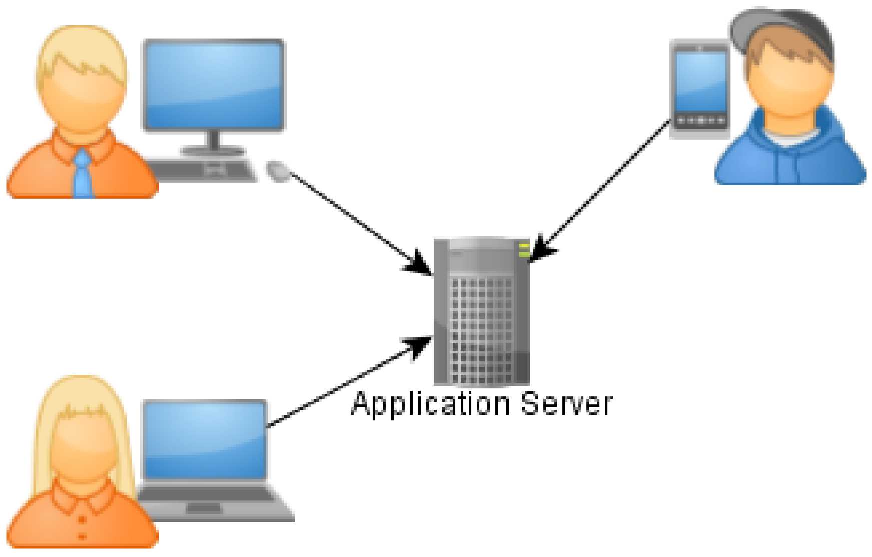


Figure 4.1: Accessibility graph.

A goal is to make the application very easily accessible. Deploying the application as a web application enables users to access it through software they have already installed on their computers, tablets and smartphones - a web browser.

There have recently been a plethora of web applications using gamification to achieve educational results, and the ones which have achieved the necessary user base report extraordinary results [56]. This was another important motivational factor for our architectural choice of using a web application as our platform, these innovative applications functioning as our proof of concept.

In addition to arguably being the ultimate platform in terms of accessibility and satisfying the desire for a centralized approach, web applications come with several strengths:

Excellent front-end capabilities

The web has progressed from being a set of poorly aligned tables to a 2D-heaven. With the latest advances in HTML and JavaScript, one can do virtually anything 2D in a fairly effective manner. While 3D-graphics could certainly make a very strong contribution to this application, it is a whole different can of worms and far beyond the scope of this project.

JavaScript

JavaScript is an interpreted programming language, mainly used to run client-side scripts and alter content in a users' web browser [40]. JavaScript offers opportunities to do virtually anything in the web browser, which mitigates many previous limitations of web applications. The opportunities of JavaScript cater specifically to the project of this thesis, because game elements may be introduced. In addition, there are many libraries available that simplify the process of creating interesting and dynamic user interface components.

Existing frameworks.

The proposed application will require a high degree of easily modifiable and dynamic content, with advanced backend functionality. The only realistic way of going through with such an implementation is to make use of a *Web Application Framework (WAF)* [12]. A WAF is a software framework that is designed to support the development of dynamic websites. Implementing a Web application from scratch is a very-time consuming process. Luckily, there are many WAFs to aid the developer by providing additional abstractions, tools and libraries to handle typical web application challenges [12].

4.2 Choosing a framework

There are countless WAFs to choose from, many of which offer the same features with slight variations or in different programming languages. Since time at the essence we have compiled a short-list of open-source frameworks that are based on programming languages we have some familiarity with, and that are known for supporting a quick development pace.

Apache Struts 2. A Java-EE-based open-source web application framework. It is based on Java Servlets and Java Server Pages (JSP) [12]. Being based on Java-EE it is built on a solid and secure foundation, with extensive support for code testing and maintenance.

Django. A high-level Python web application framework that encourages rapid development and clean, pragmatic design [22]. Django focuses on automating the development process as much as possible, encouraging code-reuse and a modular design. Django is being used by big companies such as Mozilla, Instagram and The Washington Times.

Ruby on Rails (RoR). A high-level web application framework based on Ruby. Like Django, it encourages rapid development and clean, pragmatic design [95]. Ruby on Rails also focuses on automating the development process as much as possible, encouraging code-reuse and a modular design. Ruby on Rails is utilized by GroupOn and Github, among others.

Play! A Java-based web application framework. Play is based on a lightweight, stateless, web-friendly architecture [85]. It is heavily inspired by Ruby on Rails and Django, but also significantly younger than its counterparts. It aims to support rapid development by using strong conventions in order to reduce the amount decisions the developers need to make. Play framework is used by large companies such as LinkedIn and The Guardian.

4.3 Differentiating the frameworks of choice

It is clear that Django, RoR and Play take very similar approaches - that of reducing development time - which is ideal for the purpose of this thesis. Possible downsides of these three include scalability and security challenges, which are not critical in the case of our proposed prototype. Apache Struts 2 appears older and more cumbersome, and while newer is not always better, it is often so in the area Computer Science. The focus is therefore separating the aforementioned three.

All of the aforementioned frameworks have been utilized by large global companies, proving their ability to stand a test of a certain magnitude. Given that Play is significantly younger, there is a little research on it, and also there is substantially less accompanying documentation. RoR and Django are according to research [84], [5] equally good - choosing between them comes down to personal preference. Both also have large developer communities. Since the authors have some previous experience using Ruby, Ruby on Rails is the framework of choice.

Advantages of using Ruby on Rails

Ruby on Rails (RoR) comes with many built-in features and architectural conventions that will enable swift development of a high quality product. In

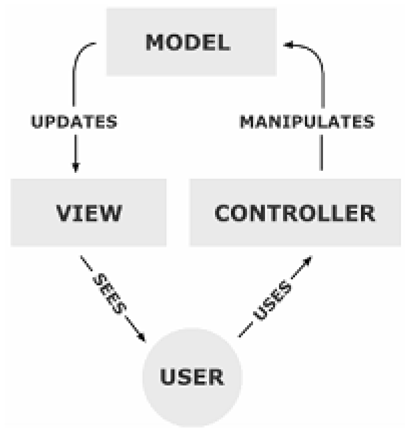


Figure 4.2: MVC process drawing. Source: Wikipedia.

this section we will discuss the features that may be relevant to the application prototype of this thesis.

Model-View-Controller. RoR employs a *Model-View-Controller* (MVC) architecture [12], which is advantageous because it separates the representation of information from the user's interaction with it by having a model component that contains the application data, a controller that mediates input, and a view to display a representation of the application [64]. This makes it easy for developers to implement core functionality as if they were just programming in the language of choice, receiving input through the controllers and displaying the output with dynamic templates. MVC encourages reuse of code and separation of concerns in such a way that it is ideal for a clean and modular design. Figure 4.2 shows an illustration of the flow of MVC.

Security Considering the ethical issues we have mentioned regarding data privacy, it is important to have solid user authentication. Also, like in normal school assignments - users must not be able to cheat in order to pass tests, complete assignments and so forth. RoR comes with built-in user authentication using secure sessions, and support for different roles making it easy to differentiate teachers and students. In addition, there is built in mitigation for CSRF-attacks, XSS-attacks, SQL-injections, password cracking and any other commonly known attack, that could threaten the integrity of the system. While it is highly unlikely that elementary school students have the capabilities to perform such attacks, it is not as unlikely that some have an older brother that might, as the recipes are usually easy to find on the web.

Also, these mitigations are automatically put in place when using RoR, so there is no extra burden on the developer and no design decision has to be made on the matter.

Systematized testing We have previously mentioned the importance of testing for quality assurance and consistency. RoR comes with a framework that trivializes testing facilitating a rapid and consistent development process.

Database controller From experience, database interaction can often be a difficult and time-consuming task for the developer. RoR utilizes a built-in object-relational mapper (ORM) allowing the developer to communicate with the database as if it was just a collection of Ruby objects, even when doing complex lookups. This greatly simplifies database interaction, helping to decrease development time.

Gems A *gem* is an external Ruby library that can be utilized within RoR. Because of the high modularity of RoR's design, there are currently a plethora of different gems available for download which trivializes many advanced, common problems when developing web applications.

Disadvantages of using Ruby on Rails

As with anything, there are technical disadvantages to using RoR. A technical limitation of using Ruby is that - being a dynamically typed scripting language - it is very slow compared languages like Java and C Sharp. For this reason, claims have been made that RoR scales poorly and the hardware demands are high [43]. This is however not a great concern to the application prototype of this thesis, as there is no need for having a high amount of users on the same server. In addition, hardware costs are very low compared to developer time these days, and cloud services such as Amazon S3 have made it relatively inexpensive to scale web applications seamlessly.

Chapter 5

Technology and Learning

5.1 Gamification

Gamification is a widely used term to describe the process of adding game elements to a utility task in order to increase user motivation. However, it is not widely understood, and it is being used very uncritically by many who do not quite understand the concept [74], [55]. Karl M. Kapp defines gamification:

> “ Gamification is using game-based mechanics, aesthetics and game thinking to engage people, motivate action, promote learning and solve problems. ”

When referring to *gamification* throughout this thesis, we are basing it on this definition. When using variations of the therm *gamify*, we are referring to the act of applying gamification to a task or a software application.

The goal of gamification is to transform a utility task into something that people wish to invest time and energy, in the interest of this thesis - a learning task. The completion of said task should produce a quantifiable result to hopefully produce an emotional response in the user. This response is often equally triggered when doing regular school work as well, but then usually as a final release at the end of a long and demotivating task. With gamification, it is possible to stage a series of sub-tasks, triggering said response at regular intervals, keeping the student motivated throughout the process [55]. Important sub-goals include engaging users, motivating action and promoting learning [55]. In the subsequent paragraphs we will discuss some of the key elements of gamification.

Engage. Engaging users is an explicit goal of gamification. It entails gaining the attention of a user and involving her in the task in question. Engaging users is seen as a primary goal of gamification.

Motivate action. "Motivation is a process that energizes and gives direction, purpose or meaning to behaviour and actions" [55]. Another feature of gamification is that it can instil and maintain motivation in users. To achieve this it is important that challenges are well balanced - the user will lose interest if the challenge is too easy or too difficult.

Game mechanics. Common game mechanics that may be useful for *gamifying* utility tasks include "includes levels, earning badges, point systems, scores and time constraints" [55]. Game mechanisms are not in and of themselves considered to be enough to transform a boring task into an engaging experience, but they are important components in the gamification process.

Aesthetics. Aesthetics is another important component of gamification . The reason for this is that a person's interest in the gamified application is deeply affected by her perception of said application. If the application is aesthetically pleasing, it will help invoke user interest [55]. Like with game mechanics, a pretty interface alone is not enough to engage users, but it is an important component.

Promote learning. Ultimately, our interest in using gamification is to promote learning. Games share many commonalities with educational psychology techniques that are already being utilized by teachers [55]. Ideally, one can find the middle-ground and leverage these advantages by transforming otherwise boring tasks into game-like experiences, without sacrificing any of the learning objectives.

5.2 Characteristcs of Learning

In our pre-thesis study [52], we uncovered many important characteristics of learning that highlight how e-learning and gamified learning may enhance education. In the following sub-sections, we present the essence of those discoveries, because we feel they are vital to understanding the context of this thesis.

The topic of learning and education is widely discussed, with many exciting theories challenging the current norm of the established school system. The importance of passing on our collective knowledge to the next generation is undeniable and as such many people have a strong opinion about education. After all it is something we all go through, and many argue that the proficiency with which we pass on knowledge is what separates us from animals. However, when people are talking about their own education (or education in general) there is one word that almost invariably dominates the conversation: *boring*. This word has become so ingrained within our educational model that it naturally occurs whenever anything related to school is discussed - is this simply an unavoidable consequence of trying to educate children, or can something be done to make learning more fun?

In order to answer this question, one must first consider exactly what takes place when learning occurs - what the variables are. Before we can even do that, we need to define exactly what we mean by *learning*. There are of course countless definitions floating about, each one stating essentially the same thing as the next, only phrased differently. To define learning we have sought out the aid of the Oxford Dictionary:

> “ Learning can be defined as knowledge acquired through study, experience, or being taught. ”

What we can draw from this is that learning essentially is a mental process where we acquire new or modify existing knowledge. However, there is a distinction to be made - there is a difference between what we refer to as *learning* and *true learning*.

In his classic book *How Children Fail* [49], author and teacher John Holt harshly criticizes the current educational system for its lack of engagement and points to countless examples where he feels the schools have failed their students; he claims that schools provide fragmented, disjointed facts which impedes true learning. The solution proposed by Holt is to present knowledge as a whole (by introducing a context within which the knowledge can be applied), to make the student see what the knowledge represents rather than just make them repetitively manipulate the same symbols.

So how can schools become more engaging? There are a vast majority of different opinions on the matter, as well as a lot of inconclusive research. We will discuss the current state of affairs, and in which areas the current approach to education is lacking.

Augmented learning

The concept of augmented learning is defined as *on-demand learning:* learning where the environment adapts to the learner [23]. On-demand learning entails that the learner is presented with a context for the information he is intended to understand. This method of learning contrasts with traditional associative learning, which usually involves the absorption of knowledge taking place prior to later application.

In order for a complex topic to be fully understood by the learner, context is essential because it allows the learner to appreciate how the knowledge can be applied [49]. Research suggests that this might a better approach to learning (as opposed to the current one), as it presents the knowledge as a whole, rather than split into disjoint sets of information that seem irrelevant to each other [90].

The concept of augmented learning is already widely used in video games. A constantly evolving environment wherein the student can apply recently acquired knowledge is equivalent to what author and game developer Raph Koster calls 'grokking patterns' [61]. The process he describes is that of recognizing a pattern within a game and then applying this knowledge to complete challenges utilizing this pattern. According to both Koster and Jane McGonigal, this is a very effective way of inducing flow, which very strongly promotes immersion [19].

It is hard to introduce augmented learning in schools based on a traditional educational model, as this new approach requires an ever-changing context to accompany new items of knowledge, in contrast to a one-size-fits-all lecture for all students in a classroom. Software applications however, are excellent at constantly changing and adapting their environment because they do not have to concern themselves with the constraints of the real world. Also, they have the ability to store knowledge about the user and change in accordance to said knowledge. Because of these extremely important attributes in relation to support augmented learning, video games might be an ideal candidate for such an approach [59].

5.3 Games and Learning

In order to discuss games and learning, we must first define what a game is. For a common word like *game*, there are countless definitions. Salen and

Zimmerman [96] define a game as:

> “ ”A system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome” ”

This definition might seem a little vague to some, but that is necessary in order to create a definition that encompasses the full meaning of the word. One of the main aspects that defines a game is the artificial conflict. Artificial means that it is separated from the real world - made up, and conflict embodies a contest of powers. Conflict stimulates the desire in humans to win, which motivates the player to perform, and the conflict can be with the game itself or with other players [96]. Since a game is artificial, there needs to be rules to provide a framework for play to emerge. Finally, games need to have a quantifiable outcome - a conclusion - where players either win, lose or receive some kind of score. This is an important motivational factor.

Using games for learning

Marc Prensky, an internationally acclaimed speaker, writer and consultant in the areas of education and learning, says it best: ”Our students have changed radically. Today’s students are no longer the people our educational system was designed to teach.” He points to the fact that students have not just changed their attire and slang (as is the case for all new generations), a big *discontinuity* has taken place, or as he calls it - a singularity, an event that changes things so fundamentally that there is no going back. This singularity is the arrival and rapid dissemination of digital technology in the last decades of the 20th century [87].

This shift in the average student must be addressed by the schools, and Prensky points to the fact that there is indeed one medium which fundamentally engages this new generation: *video games.* As such, we will delve into the connection between games and learning in this section.

This far we have discussed different kinds of video games, learning, meaning, engaging students and augmented learning. We will go into more detail about exactly what makes video games suitable for an educational setting, and provide some arguments for and against this proposal. The concept of augmented learning will also be studied more deeply, and we will see how video games can actually facilitate scenarios in which augmented learning takes place. We will also provide some examples from the real world that utilize these principles.

After the boom of video games following the new millennium there has been much discussion about video games, and whether or not they can provide value to society other than pure entertainment. The youth's obsession with video games made education seem like a rather obvious choice of application, but there were many concerned opponents of this idea - parents, teachers and psychologists claiming that video games "corrupted the youth" rather than engaged it [3].

Although there is a lot of conflicting research regarding the effect violent video games have on the youth, there is no denying the potential within games - especially since video games tend to engage students in a way no other media can [37]. For instance, a study dating back as early as 1992 revealed that video games have been shown to be particularly effective when used to teach easily quantifiable content such as mathematics and language [88].

It has also been shown that computer based quiz games can give positive long term results in student retention. This was attributed to that the students were more focused, because they found the method more enjoyable [89]. When drawing these kinds of conclusions, one must be careful to consider why the students found the method more enjoyable. If the increased focus can be attributed simply to that it is a new and refreshing change, rather than the computer game itself, we are back where we started.

A different study points to that computer games may have positive effects on classroom dynamics. Underprivileged students in Chile were given educational video games that a portion of the class would play for 30 hours over a 3-month period. The students that were playing the provided game showed significantly better progress than the external control groups. However, the classmates of the students playing the game, who were not playing themselves, showed a similar progress to the students who were playing the game. Teachers also reported an improved motivation to learn [92].

Games as a proposed solution to make schools more engaging, and potentially facilitate true learning, sounds exciting enough, but exactly what is it about video games that make them suitable for application within an educational setting?

Characteristics of games

As previously discussed, there are some key aspects of video games that pertain specifically to learning and absorbing knowledge, namely *just in time* and *on-demand* learning. There are however other characteristics of video games that make them suitable for educational needs as well, which we will discuss in this section.

Immersion is a key component to video games, and also a key component to effective learning [38]. According to studies in cognitive science, people who play video games often feel like their bodies and minds have been stretched into a new space [14], inducing a very motivational state.

Naches is another important characteristic of video games pertaining to their utilization in an educational setting. Naches is originally a Yiddish word for the bursting pride we feel when someone we have taught or mentored succeeds [68]. A study has shown that players felt a kind of vicarious pride from playing over another's shoulder - giving them advice, *especially on games they had already mastered.* The author of the study, cognitive scientist Cristopher Bateman, reported that "Players seem to really enjoy training their friends and family to play games, with a whopping 53.4 percent saying it enhances their enjoyment" [9].

If one could "gamify" education in such a sense that it could induce the feeling of naches, then it becomes obvious that students who have already mastered certain topics would become a valuable asset in the classroom, which again would contribute to the social well-being of the classroom environment [49].

Fiero is another emotion connected to video games that is more prevalent than others, and it is especially important for this discussion. This emotion known as *fiero* is possibly the most primal emotional rush we can experience [68] - in fact it is so intense that the English language does not even have a good word for it [62]. *Fiero* is what we feel after we triumph over adversity, and you know it when you feel it - or see it for that matter, because nearly everyone expresses fiero in the same way: we throw our arms over our heads and yell [68]. The fact that virtually all humans physically express fiero the same way is a good indication that this emotion is related to some of our most primal instincts - our brains and bodies have evolved to experience fiero early in our emotional development. According to researchers at the Center for Interdisciplinary Brain Sciences Research at Stanford, fiero is the emotion that first created a desire to leave the cave and conquer the world - a craving for challenges we can accomplish [47].

In schools, just about everything seems like a challenge, but much of it seems insurmountable. This is why fiero is so important in this context, because if we could find a way to make schoolwork induce fiero, i.e. make the challenges achievable, then students might actively pursue these tasks in order to experience this primal emotional surge. As we initially pointed out, games are excellent at generating fiero, especially since *more challenging* obstacles generate *more fiero*. In addition to this, nearly all games come with a difficulty setting to ramp up the fiero-generation, while most schoolwork comes with just one setting, which is what makes the challenges seem insurmountable in the first place. It seems that gamifying school work might be a good approach to realize this potential.

Flow was introduced by the positive psychologist Mihaly Csikszentmihalyi [19], and he defined it as the mental state of operation in which a person performing an activity is fully immersed in a feeling of energized focus, full involvement, and enjoyment in the process of the activity. What is interesting about flow is that when you experience it, you want to stay immersed within that state - both quitting and winning are equally unsatisfying outcomes [68]. According to Csikszentmihalyi, flow is completely focused motivation. This means that when experiencing flow, one is potentially experiencing the ultimate emotions for performing and learning well. This is where the connection between video games and flow becomes apparent, as video games are excellent generators of flow [68].

As anyone who has ever played educational games will tell you, not every video game is good at generating flow. However, successful video games are good at this, mainly because good learning in games is a capitalist-driven Darwinian process of selection of the fittest [37]. These games excel at providing the players with a gradually increasing difficulty curve, which always keeps the players at the edge of their capabilities - otherwise referred to as ”just-in-time learning” [59]. This, according to Csikszentmihalyi, is a surefire way to induce flow.

Just-in-time learning is the concept of having the student learn a piece of information just when she needs to apply it. This piece of information can for instance be a fact, a method or a concept, and the idea behind is that new material is much more likely to be remember by the student this way [77]. The reason for this is that the human brain stores knowledge by association, context and semantics [49]. If given the opportunity to a apply newly attained knowledge right away, the student is much more likely to find a slot in her memory system that fits this new piece of information. E-learning systems excel at facilitating just-in-time learning, because they allow rapid interactions between student and "teacher", so that one piece of information can be attained and applied at the time.

Feedback and rewards. A common game mechanism is that of progress feedback. The most effective way of providing the end user feedback on the progress they have made is through graphical representations [29]. User agency and engagement can be promoted through relevant user feedback at set intervals, for instance for completing a certain set of challenges - even though they only represent a small fraction of the total challenge. This process of promoting user engagement through relevant feedback can, according to research, be very effective of increasing motivation [29], and employs the same methods as augmented learning [59].

Coupled with good gradual difficulty progression and a well-paced reward system, these mechanics could work in synergy to induce both flow and fiero. We support this claim with the research done on the induction of these two emotions - flow is effectively induced in subjects where they are kept on the edge of their ability, which in itself is often motivation enough to continue [19]. However, this motivation is based only on the intrinsic rewards in the form of endorphins issued by the brain as reward for solving problems that we need to engage our entire mind to solve. According to McGonigal, this kind of motivation can be intensified through *extrinsic* rewards as well [68], which a gradually increasing progress bar would be a good example of. There are four intrinsic rewards associated with video games, identified by author Jane McGonigal [68]: satisfying work, the experience or hope of being successful, social connection and meaning.

It should be noted that extrinsic rewards are of far less value, in terms of user engagement and agency, than intrinsic rewards, but in tandem they are more powerful than intrinsic rewards alone. McGonigal points to the fact that many players find merely gaining a new high score is motivation enough in itself to play a game, which is a purely extrinsic reward [68]. Based on these observations we feel we can make the claim that complementing the intrinsic rewards with extrinsic rewards representing gradual progression would increase student motivation and engagement.

An argument can also be made that the player does not necessarily need to be constantly aware of her progress, but rather given the feedback at relevant moments. This could appeal to the player in the same way as rewards, and could indeed increase engagement further [29].

Accidental learning. An interesting concept that may be invoked by educational video games is "accidental learning". It means that users of the application learn the intended subject matter simply by interacting with the application's features and mechanics without any notion of the motives behind the application itself, which according to the research of Jane McGonigal and James Paul Gee is extremely effective [68], [38]. An illuminating example of this is language learning through video games. The motivation for becoming proficient in a language when your continued enjoyment of a well designed video game, both in sense of immersion and aesthetics, is contingent upon said proficiency is very powerful [68].

Games and Augmented Learning

We previously discussed how video games might be an excellent platform for augmented learning, and in this section we will shortly touch on why that is.

As stated initially, video games excel at generating context for any given scenario, which is a very important condition for augmented learning to take place. However, context is not the only thing that matters in this sense: it is also important to consider how the knowledge is being presented, and if the student can be given the correct learning curve in order to fully acquire this new piece of knowledge.

A common approach in the game industry is to have the lower levels of a video game include a tutorial, which gradually introduces the player to all aspects of game play. These characteristics are introduced *just in time*, which increases the likelihood of the information being not only absorbed, but also understood [38].

The flexibility of video games also allows the possibility of choosing from a variety of ways to present the knowledge: Perhaps the math equation becomes the head of dragon, and you must solve it in time before he eats the princess - or maybe the science experiment in physics class to demonstrate gravitational pull is a boulder rolling down a hill to crush some unsuspecting goblin. It seems like video games can provide much of the needed conditions in order to meet the demands of augmented and *true* learning.

Because of this very suitable connection between video games and augmented learning, the potential for educational gain becomes obvious. If one could somehow harness the augmented learning potential in video games and apply it to the existing educational model one could very well find the necessary tools to motivate otherwise unmotivated students [49].

Peer motivation in the context of games

Another game mechanism providing benefits to user engagement and immersion is that of peer motivation. The approval of our fellow humans has always been an incredibly powerful motivation [67], and this has commonly been employed by video games in order to increase user agency, through high score lists and so forth. A more recent example is the booming growth of social media influence in modern-day society, where the major motivational factor for use is that of receiving approval from your selected peers. An entire industry is even based upon this very concept - that of social media games - and many have been largely successful [11].

5.4 E-learning

E-learning is a broadly inclusive term describing a the use of digital media for education. There are many different views on what e-learning is or should be, the European e-Learning Action Plan defines e-learning as "the use of new multimedia technologies and the Internet to improve the quality of learning by facilitating access to resources and services as well as remote exchanges and collaboration" [48]. This definition is very elaborate and probably confusing to many.

Holmes defines e-learning as "online access to learning resources, anywhere and anytime" [48]. This definition might not be very precise, but it grasps the essence of what e-learning has been seen as for years. Any student in the western world today will be familiar with using a website for downloading and maybe uploading homework, viewing the syllabus and checking deadlines. While this is certainly an improvement from handouts, it does not mean it should stop there. Holmes' definition is perhaps a little uninspiring, and means such as virtual 3D-worlds, games, video and audio telecommunications are accidentally left out. These are elements that could not only enhance education quality, but also make education more fun.

For the sake of this thesis, we will refer to e-learning as

“any digital platform designed to be a learning tool that the learner interacts directly with”

This includes but is not limited to, websites, games, virtual worlds and so forth, but excludes trivial use of technology such as PowerPoint-presentations, videos, etc.

Software as a tutor

It has long been known that a one-to-one teacher-student ratio dramatically increases learning effectiveness. In 1984, Benjamin Bloom conducted a study measuring just how much more effective it really is [50]. Bloom studied children who had been pulled out of class to receive special one-to-one instruction. He was able to conclude that the children given one-on-one instruction performed two standard deviations better than their classmates who did not receive special tutoring. This equals the effect that is required to take an average performer into the top 2 percent. Allegedly, ”Bloom’s findings caused a stir in education, but ultimately they didn’t significantly change the basic structure of the classroom” [105]. The reason for this is that increasing the teacher-student ratio comes with a very obvious price-tag, and it is easy to imagine that a one-to-one ratio would be very expensive. One of the big goals, and certainly a strong-point of e-learning, is an application’s ability to emulate the function of a personal tutor, thus creating a virtual one-to-one student-teacher relationship. This might sound complicated to some, but breaking down the most important functions of a tutor, it is evident that these are pretty simple to implement into a software application:

Presenting hints. Providing hints that point the student towards the solution can be done by pre-generating generic hints that can be shown, one at the time, by the click of a button. Hints cannot easily be tailor-made for the individual student, but it is possible with the use of AI-algorithms and knowledge about previously completed tasks. Either way, generic hints generally go a long way.

Instant feedback on problem-correctness. Providing instant feedback upon completing a task is trivial for a software application. Simply store the possible answers and perform a check.

Dynamic task difficulty. By storing information about previously completed tasks on a given subject by a given student, a software application can tailor task difficulty in order to provide the correct level of difficulty. This is important to maintain student motivation, and is known to induce the *flow* state [71].

Presenting new material. For a long time, presenting new material with a quality that is comparable to a real lecture, has been a difficult task for e-learning applications. However, with the emergence of online video streaming platforms in recent years, presentations are subject to few limitations, the only exception being the possibility for students to get instant answers to eventual questions.

Motivating the student. Motivating the student is a very broad problem, and possibly something a software application cannot do in the same manner as a real human being. However, the recent focus on *gamification* has given many e-learning applications tremendous motivational abilities, which we discuss further in the *gamification* section of this thesis.

While all the aforementioned functions are more or less trivial to implement from a technical point of view, combining them into a good e-learning platform can be difficult. Most of the challenge lies in forming the courses and tasks, and creating user-friendly interfaces, so that user forgets he is on an e-learning platform, and is immersed in the learning material. Another limitation is that the discussed approach only works for subjects that have clearly defined answers. Luckily, the category of subjects with clearly defined answers that are a part of general education is pretty quite big.

Chapter 6

Related work

The following sections present e-learning applications that have mastered the previously discussed aspects to some extent, and are currently on the rise.

6.1 Duolingo

Duolingo is a free language-learning website for anyone to use, where users take lessons and complete tasks [25]. The project was started by CMU professor Luis von Ahn and student Severin Hacker. Initially, it served only English-speakers wanting to learn Spanish, but now also provides service for Spanish, Portuguese and Italian-speakers to learn any of the aforementioned languages, as well as French and German. The application has a gamified approach; participants are rewarded with points for completing tasks. Each time a sub-task is failed, a *life* is lost, and if the user runs out of lives she will have to start the exercise over. When a sufficient amount of points is gathered, the user is rewarded with access to more difficult tasks, and tasks at one level can be repeated until enough points are gathered. This keeps students engaged by always providing them with tasks of appropriate difficulty. The application is also gamified in the sense that it has a simplistic responsive appearance, with encouraging symbols and progress-indicators. In addition, users may become a *master* of certain areas in a language. There are also time-based challenges for advanced users, and the application even records frequency of use for each word in the vocabulary, which allows users to go back and practice words they might be likely to forget.

Duolingo excels at displaying to the user how far he has progressed, and where in the subject-hierarchy the knowledge he has attained belongs. This is shown in Figure 6.1.

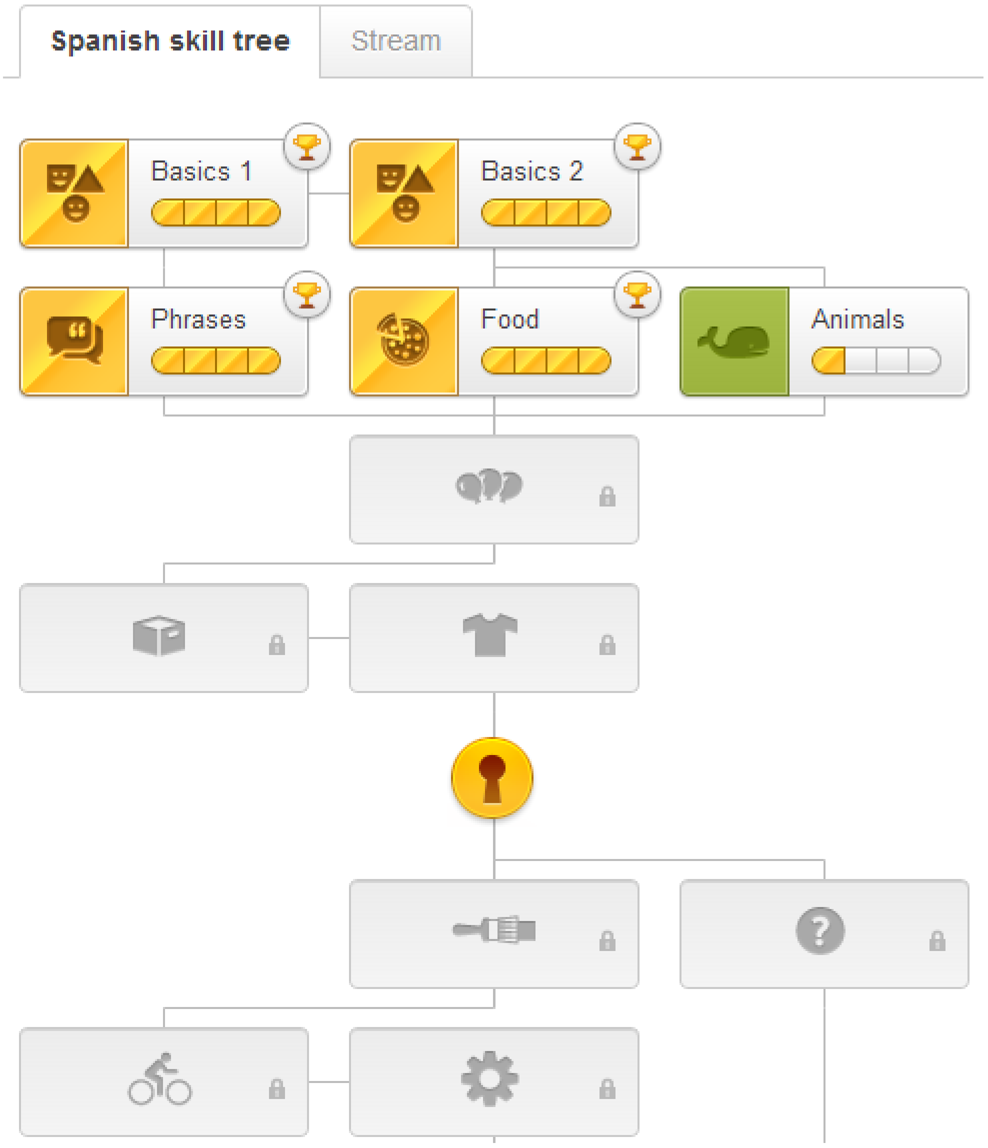


Figure 6.1: Duolingo skill tree.

Duolingo has been very successful in commercial terms, and by the fall of 2012 they managed to raise 15 million dollars from investors [104]. More importantly, it has been declared successful educationally successful by a research team from CUNY and USC [110]. In this study, 88 students taking a novice Spanish class used Duolingo over the course of the semester. The students were then tested with the widely recognized WebCAPE [82] test, and the students that used Duolingo performed significantly better than those who did not. In addition, they reported to enjoy this form of learning.

Duolingo is interestingly similar to the application outline we proposed in our pre-thesis project [52], though we had no knowledge of the existence of Duolingo at the time.

6.2 Memrise

Memrise is another free learning website, created in order to ”make learning your favourite playtime activity” [70]. The creators are experts in neurology, and claim they wish to leverage the science behind humans and learning to facilitate effective and fun learning about anything. It utilizes flashcards[1] and mnemonics[2] Memrise also makes use of some gamified elements, particularly concerning score and progress-indicators, and they even have a global leaderboard in order to encourage users to compete and do well. In addition, topics are learned using a 3-phase garden analogy; plant seeds, harvest and water the garden. In the first phase, material is learned for the first time by being presented and tested repeatedly. When a seed is successfully planted, it has to rest for a minimum of 4 hours, and a maximum of 24 hours. In this timespan, the plant may be harvested by answering a test on the material. If the test is answered correctly, the plant is transferred to the garden of the user. Material located in the garden has to be attended to periodically. This analogy is a beautifully simple way of forcing a learner to attend to the material at intervals that are have been scientifically proven to improve chances of storing newly learned information in long-term memory [70]. The garden analogy is gamified by adding illustrations and animations in order to help motivate users to care for their garden. Memrise has also had commercial success, and has recently managed to raise over 5 million dollars of funding [81].

[1]A flashcard is A card printed with words or numbers and briefly displayed as part of a learning drill [21].

[2]A device, such as a formula, image or rhyme, used as an aid in remembering [21].

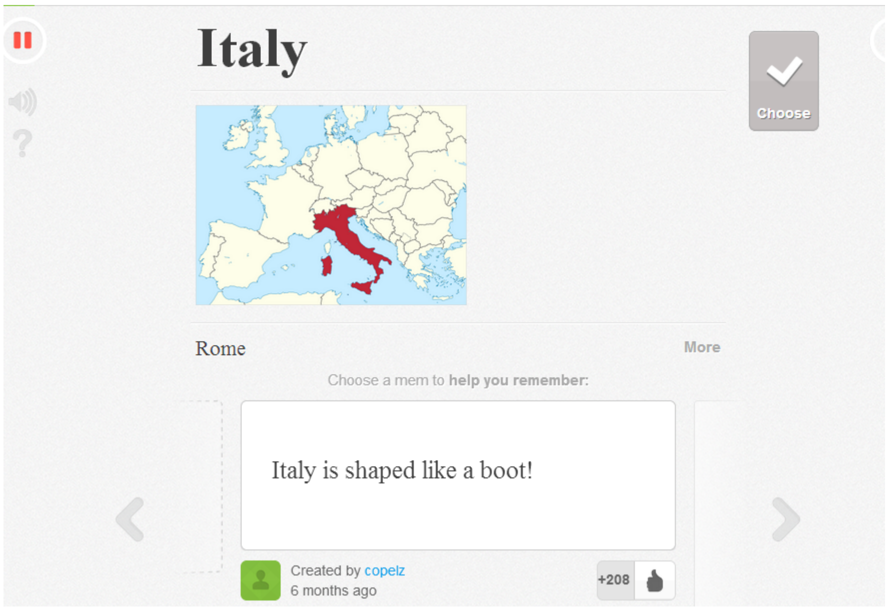

Italy
Choose
Rome
More
Choose a mem to help you remember:
Italy is shaped like a boot!
Created by copelz
6 months ago
+208

Figure 6.2: Memrise presents pieces of information with a learning rule.

6.3 Edutopia

Edutopia is a non-profit foundation started by Star Wars creator George Lucas in 1991 [28]. Lucas criticises the current educational model for being too fixated on testing, and claims that ”abstract concepts and isolation is education’s worst enemy” [30]. He also claims that students learn mathematics because they have to, and that kids want to discover things rather than learn them. The goal of Edutopia is to help change this with extensive use of e-learning[3]. Their main base of operations is their website, which acts like a community or social network that targets teachers and schools in order help them improve their teaching methods. While Edutopia’s approach is not directly related to the approach of this thesis, it is interesting in that their motivations are the same; evolving the educational model of elementary schools to be up to date with the opportunities of new technology.

It is also noteworthy that Edutopia’s website contains a lot of articles and videos promoting the organization, and seemingly has a large amount of users, but it is very hard to come by independent third-party information on the foundation. This is especially strange considering the amount of time the organization has been around. Also, academical interest in the foundation

[3]An Internet-based teaching system [21].

seems to be zero. Their intentions do however seem unquestionable.

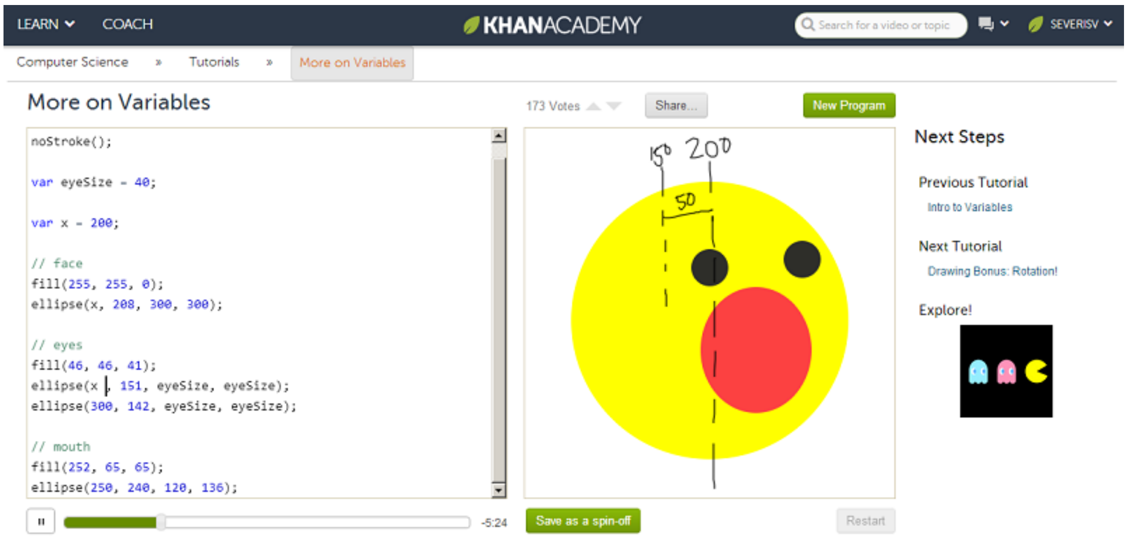
6.4 Coursera

Coursera is an education company that cooperates with top universities to offer free online courses for anyone to take [18]. Their aim is to blur the disparities concerning access to high level education throughout the world, by offering anyone, anywhere, the opportunity to take high level academic courses. The format is based on the classic educational model, except that lessons are taught through videos, and tests are taken online, both hosted on their web-based platform. Coursera has received widespread recognition, and in the start of 2013, The American Council of Education declared five of their courses eligible for real university credit [107].

Coursera has some interesting aspects beyond those of accessibility. When taking regular classes, some students feel that half of their job is simply showing up, and do not pay full attention to the content being taught. With online courses, students have the option to pause a lecture or go back if they did not understand something, and student participation can be induced in a way that is not possible with traditional lectures [35]. This provides student immersion and engagement, which as we know, is an integral part of effective learning [52]. Another advantage is that different concepts in the syllabus can be structured and graphically presented in a way that makes it easier for the learner to grasp the connections and dependencies within the subject of choice. In a traditional course the student has to form this type of structure himself, which can be advantageous, but also less likely to be completed.

6.5 Khan Academy

Khan Academy is one of the pioneers within the field of e-learning, when looking only at platforms with measurable success. It was started in 2006, and can be seen as sort of a mix of Coursera and Duolingo, although Khan Academy is far older and more extensive than its aforementioned siblings. Khan Academy provides thousands of video tutorials covering vastly different topics, mostly focused on physics and mathematics [57]. Each tutorial lasts around 10 minutes, which makes them easily accessible and digestible - no time to get bored, or feel an overload of information. A practice problem directly pertaining to the information discussed in the tutorial accompanies each video, so students can begin working on relevant problems right after



```
noStroke();

var eyeSize = 40;

var x = 200;

// face
fill(255, 255, 0);
ellipse(x, 208, 300, 300);

// eyes
fill(46, 46, 41);
ellipse(x, 151, eyeSize, eyeSize);
ellipse(300, 142, eyeSize, eyeSize);

// mouth
fill(252, 65, 65);
ellipse(250, 240, 120, 136);
```

Figure 6.3: Khan academy instruction.

watching the tutorial. The people at Khan Academy have also introduced another level of abstraction - the classroom: students can be grouped together in classes, and the teacher can view all available data for all students to pinpoint exactly which subjects are hard to understand, and which students have already mastered various subjects - who can in turn become valuable assets to assist co-students in said mastered subjects. In addition to this very adaptable, tailored and statistics-centered system, game mechanics are also introduced - students can earn badges and points by completing practice problems, beating challenges and solving tasks. These badges are ordered in a hierarchy, from easy to obtain to almost impossible, and the people at Khan Academy report an astounding amount of users spending countless hours of their spare time working on getting the most prized badges - just for bragging rights [105].

Although many embrace Khan Academy and what is has accomplished, some are critical [105]. Critics argue that Khan Academy encourages uncreative and repetitive drilling, leaving children locked to their computers rather than being out interacting in the real world. The inventor himself has claimed that he is ”not an educational professional; he’s just a nerd who improvised a cool way to teach people things” [105], but he has also been critical of the current educational model [56].

6.6 Summary

This chapter discussed a number of projects that are interesting to this thesis to varying degree. This section explains why.

Coursera and Edutopia are technology-centred projects that work closely with schools and universities in America. They have been received well, and Coursera even offers courses that have been declared eligible for real university credit. These projects are interesting because they are paving the way for e-learning in official education, receiving praise along the way.
Khan Academy, Memrise and Duolingo are all web applications with a gamified approach that are doing very well at the moment. They all leverage the advantages of using a gamified web platform, such as easy access, instant progress feedback and rewards. These applications share many commonalities with the envisioned prototype for this thesis, and their success is therefore very encouraging.

Part IV

Design and Development

Introduction

From the theoretical study we have concluded that the educational sector could benefit from the use of software applications in order to facilitate true learning [52], engagement and motivation of students. This could be achieved by applications fulfilling certain criteria; motivational game mechanisms (such as rewards, gradual progression inducing *flow*[19], etc.), social aspects of a virtual classroom community and resources necessary for school work, such as practice tasks, video tutorials and mentors.

The conclusion was based upon extensive amounts of research done within the field of education, motivational psychology and video game mechanics. In addition to all the published research we have also taken into account the results of the numerous web applications taking full advantage of the benefits that were proposed by said research, such as Duolingo [25], Memrise [70], Edutopia [28], Coursera [18] and the exceedingly popular Khan Academy [57].

The aim of this chapter is to illuminate how we envision the gamified web application that is the focus of this thesis, and detail all the steps in the development process. We will begin by discussing the various risks and challenges one is faced with when attempting to introduce a non-conventional entity into an educational setting, and then move on to identify the risks and challenges specific to the kind of application we intend to develop.

Next, we will outline the architectural decisions, the rationale for these decisions, and discuss how they relate to the risks and challenges. We will then outline the functional and non-functional requirements for the proposed application. We will distil these requirements into an application specification which will feature clearly defined application components. The specification will be the basis for the application, and ideally we would implement all features described in this specification. Any discrepancies between our application prototype and the specification will be discussed further in Part VI - Discussion.

We will provide a data model with the intention of showing the interaction between the various components of our application. This diagram may naturally vary from the final implementation, and our discussion and inclusion of it will be on a general basis, attempting to illuminate how we intend for the various features of our application to interact with each other and the end user.

After completing the application specification we will discuss how we will ensure that the development process is sound through extensively testing the components we implement.

Chapter 7

Risks and Challenges

In the development of a software application there are numerous risks and challenges involved, related both to technical and business aspects [39]. Examples of such challenges include defining the target audience, as well as designing the application to appeal to said audience and so forth.

This section will first outline the challenges and risks related to the application domain, and then proceed to identify the application-specific risks and challenges. The discussion of domain-specific risks and challenges will make no implementational assumptions about the application, but rather discuss the domain in the context of introducing an application featuring game mechanics. The challenges and risks directly related to the application of this thesis will be discussed in the application-specific section.

The process of identifying risks is exceedingly important when trying to innovate and find new spaces in which to attempt to create a niche for a software application to prosper, in this case the educational plane. Luckily, the use of games and web applications within education has increased steadily in the recent years, and this will mitigate the risks and challenges we are faced with. Applications such as *it's learning*[1], Khan Academy and DragonBox[24] have all paved their own respective ways, allowing us an increased scope of agency, which will be reflected in the risks and challenges presented in this section. Some of the information found in this section is gathered from the study in our pre-thesis project [52].

[1]"it's learning is an international learning platform provider with over two million users in primary and secondary schools, as well as Further and Higher Education" [54]. it's learning is not directly concerned with e-learning, more with managing teaching materials, schedules, exercise delivery etc.

Many of the risks and challenges may be discussed both with respect to the application and the domain, but they will take on a different character when discussed in the concept of the former versus the latter, and will therefore be discussed in both sections.

7.1 Application Domain Challenges

Student engagement

One of the big challenges regarding school work is for it to engage and motivate students [52]. It is paramount that the end users of the application, students and teachers, see the potential benefits of the application through their own experience. Should it fail to appeal to either user group, results would surely be underwhelming and we would not see the desired effects.

The proposed application needs to engage the student, otherwise learning will suffer [68], [90] and most likely it will only appear as another hindrance for the student rather than a supplement to the standard educational model [49].

Appealing to both genders

An important challenge is that of appealing to both genders, as the argument can be made that video games appeal more to males than females [44]. Appealing to different age groups is another challenge directly related to this, which could be grouped together as the common challenge of appealing to different target audiences. One must consider the differences in preference among said target audiences when attempting to mitigate this risk, and keep this in mind when distilling the various risks and challenges into a requirement specification.

Usability

A challenge facing an application of any sort being used within an educational setting is that of usability. The argument could be made that this is a challenge facing any application, but it is specifically so in the case of this thesis because the merits of our proposed project is solely contingent upon the engagement and motivation for use induced by the game mechanics of the application.

Should the application fail to engage students and/or teachers because of poor usability, none of the desired effects outlined in the pre-study will be achieved, and as such satisfactory usability is a major domain specific challenge.

Appealing to different target audiences

The proposed application would ideally be used by all age groups of students within the Norwegian elementary school system, as well as their teachers. This brings up the challenge of difference in preferences, both in terms of usability and engagement. The benefits we attribute to use of our application is entirely contingent upon these two properties, and as such the challenge of appealing to all end users is very important. We intend for our application to achieve high usability for all target audiences, but it is only in the case of student usage we intend to benefit from the inherent game mechanisms. The metrics for usability for teachers is thus highly decoupled from those of the students, and it is important to keep this distinction in mind when discussing functional and non-functional requirements.

Aesthetics

To maintain target audience interest, an aesthetically pleasing interface is increasingly important [53]. While it can be argued that this is more so the case for the students, it is also important that the interface appeal to the teachers in order to motivate them to keep using the application. We must also re-iterate the importance of maintaining a mind-set which covers the preferences of *all* target audiences when creating the interface, as students in first grade will surely have different preferences for an aesthetically pleasing interface than a student in the tenth.

In relation to the importance of aesthetics, examples of educational games where the graphical elements were neglected, de-prioritized in relation to the subject matter that was intended for the students to learn, and as such failed to immerse and engage students [52]. This mind-over-matter approach, whilst seemingly noble, makes it all too equivocally clear to the players that the purpose of the game is not entertainment, but to teach them something, which by experience is a sure-fire way to make most students lose interest.

7.2 Application Domain Risks

Inherent risks pertaining to video games

The most prevalent risk associated with utilizing games for educational purposes is the fear that learning objectives may not be congruent with game objectives [52]. This risk was identified in a paper by authors Alice Mitchell and Carol Savill-Smith in their 2004 paper "The use of computer and video games for learning" and points out that while video games may certainly facilitate learning, the knowledge obtained by playing the games might not coincide with the defined learning objectives [73].

Another risk identified by Mitchell and Savill-Smith is the risk of distracting players from actual learning because the gameplay is too engaging - for instance making players focus more on scoring points than actually attempting to learn. Luckily, research has shown that if one designs the application correctly and induces the desire to win and complete tasks within the game space, then this can be a sufficiently motivating factor in itself for the player to spend more time on the given subject [75]. One has to take both extremes into account when designing the game to find the middle ground, which would ideally both induce flow and allow the player to reflect upon what she is doing, to facilitate true learning.

The risk that seems to be the most common for educational games is that they often fail to be engaging. Research has shown that many educational games are just barely preferred to standard classroom instruction [63]. This piece of information is crucial, however, to solidify the claim that video games can be beneficial inside the classroom - students prefer games, even poorly designed ones, to standard lectures, which would imply that a well designed game with educational objects could potentially create a sense of engagement.

Age span of target audiences

The challenge of appealing to target audiences of different age is also a prevalent risk to consider within the context of employing video games in a classroom setting. Ideally, an application would appeal to all pupils from the moment they first begin their education until they graduate, however this goal seems rather improbable to achieve.

For instance, the youngest students might prefer to be given their tasks and assignments set to a fantasy setting, which might prove very engaging to them. This, however, would most likely not fare well with their older counterparts, who would perhaps prefer an application that is more based on social networking web applications, which is heavily utilized in the older age brackets of students [106].

Subject matter and relative importance

The weighing of the various subjects in relation to one another is an additional factor to take into consideration. The risk of creating an application which skews the attention of the students in a way that is incompatible with their syllabus - the subject utilizing the gamified application could potentially steal focus away from other subjects.

Since the educational setting is extremely broad and diverse, the non-functional requirements could end up becoming too broad, trying to encompass too many requirements, or too much of a compromise.

Challenging the entrenched system

There is still a very heavily indoctrinated routine of utilizing standard norms for education - i.e. using (often nearly obsolete) textbooks and standard classroom lectures, even though extensive research has concluded that this sort of experience might in fact harm student motivation [49].

There is an incredible amount of research being done within education and there are a plethora of different schools of thought, often contradictory [15]. This seems to be the case because education is difficult to quantify and categorize, which leads to extensive theoretical approaches. Because of this, it might be difficult to convince teachers that the research our application is based upon is sound, even though it is heavily supported by recent research [49], [57], [90], [91]. In addition to the fact that some teachers might not see the merit of this research, other might subscribe to different schools of thought, and as such might also be less inclined to attempt to fully integrate our proposed application into their educational routine.

Even though Norwegian schools have the best equipment in all of Europe [2], the effective use of computers for completing exercises such as math problems is not widespread. This might be because of the heavily indoctrinated routine of using textbooks, which makes the transition to computer-based solutions harder to accomplish. When asked about this, a researcher at the Center for IT in education said the following: "[The use of computers in education] is not as widespread as one might think. We have good access to the equipment, but more challenges considering the usage of it [2]". This raises awareness of a very pertinent challenge to consider for our proposed solution - how we can design a web-based application to challenge the old norms of the educational system.

7.3 Application Specific Challenges

When discussing application-specific challenges, we base the challenge identification on the scope of the kind of application we intend to create - a gamified web-application. While many of the application domain challenges have already been covered, we will discuss in more detail the risks specifically pertaining to the proposed application.

Genders revisited

Since the proposed application would not function like a full-fledged game in the traditional sense, some of the previously identified, more general domain-specific, challenges take on a slightly different character. For instance, the challenge of appealing to both genders is somewhat mitigated, because the system would mirror some features found in social websites, which are very popular among female audiences [65].

Flow and Fiero

An application-specific challenge would be to induce *fiero* and *flow* outside of a traditional video game scenario, which is hard to achieve even within a traditional video game [61]. Directly related to this is the challenge of keeping students immersed and engaged without having the environmental aspects of video game graphics to rely on, which according to assistant professor at MIT, Kurt Squire, is an important part of the appeal of video games and their inherent benefits [99].

Inherent web application challenges

Seeing as we intend for the application to function in many ways as a standard web application, ubiquitous access becomes important. This does not only entail maintaining the application servers in order to ensure that the service remains available to end users, but also concerns cross-device and cross-browser support. The main challenge related to ubiquitous access is developing a responsive design [1] of the user interface. In order to establish ubiquitous access to a web service one might need to implement support for a plethora of different devices [31], which is and identified goal in the case of this thesis.

Different ends of the spectrum

Ideally, our application would be used independent of the age of students and yield the same benefits, which an application shaped as a game in the traditional sense would be more suited to accomplish. A game featuring the same objectives and narrative could potentially appeal to any age group by simply replacing the game artefacts based on the subject matter the students are intended to learn.

This challenge takes on a different form in the specific case of this thesis, mainly because we simply want to utilize and harness the benefits of game mechanisms, without creating a game in the traditional sense. We consider this to be the principal application-specific challenge, and this will be reflected in the future part of the development process.

Furthermore, we need to decide how far away from a traditional gaming experience we want to move, and how many features commonly associated with social networking web-applications we want to introduce. Certainly there are merits to both sides of this spectrum: immersion and engagement on one side, usability and a broader appeal on the other. The challenge of finding this middle ground will be very important, and the aim is to find a compromise as we carve out the requirements specification, and further distil these requirements into a final application specification.

7.4 Application Specific Risks

The development of an application to be run on the web platform has many risks inherently, such as (but not limited to) browser security issues, server faults and support for different browsers, but for the sake of brevity we will not discuss these at length in this thesis. We will instead identify the risks specific to our proposed application and discuss them utilizing the inherent risks more as a general backdrop.

Information privacy and storage

A risk to consider is the amount of data we intend to store, as there is potentially a lot of data that could be useful to an application used in a classroom setting. A common problem with such an application is that there is simply too much data, and this is not only confusing to the end user, but also puts a limitation on scalability and performance [54]. There is a risk of over-storing data which might, in the end, not make any positive difference to the end user. In addition, there are ethical issues to consider concerning the privacy of students. The challenge is therefore to carefully consider which data to store and present.

In the same vein as storing redundant and non-essential data is that of information privacy. Since we would store sensitive data in the context of school work [80], this information must necessarily only be available to authorised end users. The risk of having sensitive information accessed by unauthorised users is extremely important, and becomes even more poignant in the setting of a web application, that could potentially have many attack surfaces.

Ubiquitous access

The importance of ubiquitous access in order to promote engagement is potentially the most important risk of all. If users do not have access to the system when they wish to use it, the end user experience will suffer greatly, as well as the confidence in the use of the application. This could also potentially interrupt an important school project assigned through an application such as the one we propose. This risk is especially related to the inherent risks associated with web applications. Considering the impact a DDOS[2] could have on the requirement of ubiquitous access, this a very important risk to keep in mind.

The user inferface

An over-complicated user interface is another important risk to take into consideration as well [78], as the engagement of the application is directly contingent upon the end user experience. This risk is related to the risk of hoarding data, as more data quite possibly requires a more complex user interface to present it.

[2]Distributed Denial of Service attack. A common attack to attempt to make a network resource unavailable to its intended users.

Chapter 8

Architectural Decisions and Rationale

This chapter will provide the rationale for the architectural decisions, and discuss how they mitigate the various risks and challenges outlined in the previous chapter. These decisions are heavily influenced by the conclusions drawn from our literary study and our pre-thesis project [52], and attempt to carve out an application that achieves the benefits outlined in this thesis.

We have suggested that a software application fulfilling certain criteria could be a beneficial addition to an educational setting. These criteria were defined as certain motivational *game mechanics*, *social aspects* and necessary *resources* for school work. In order to achieve these criteria, we concluded that a web-based application would be the best approach, and this is the first major architectural decision we will discuss.

8.1 General decisions

We also considered making an educational game in the traditional sense, much in the same vein as DragonBox [24], but after conferring with our supervisor we decided that a web application would be more suitable for what we desired to accomplish. Had we chosen to develop an educational video game in the traditional sense, such a game would necessarily be focused on a particular subject, and we would lack the necessary pedagogical skills and expertise in said subject in order to ensure the quality of the material presented to the end user and the learning experience.

Said lack of expertise, both within pedagogics and various subject matter, was a spurring factor when we were considering the various options available to us when faced with making a game specifically for the sake of education. We felt that a good solution would be to simply have the experts create the application content related to their respective fields, thus ensuring the quality of said content. Our role would be that of the facilitator - providing the setting in which an expert, such as a teacher, would feel comfortable distilling their expertise into the application.

Seeing as almost every student and teacher in the Norwegian school system have access to a browser, through either a computer or a smartphone [2], Norwegian schools would be well suited to introduce a web application into their curricula. Also, by using the browser as our platform, ease of use is greatly enhanced, as users are not required to install any additional software on their respective devices. In addition, the platform is accessible to the students and teachers from home, as well as from school.

Inherent properties of web applications

The architectural choice of using the browser as a platform does entail certain inherent risks and challenges, these must be taken into consideration when distilling the requirement specification for the application. Also, such a solution relies upon a Client-Server architecture, normally to an extent that a single web server or a cluster of servers handles requests from all clients. An inherent property of web applications is that of the *thin client*: "a computer or a computer program which depends heavily on some other computer (its server) to fulfil its computational roles". A property which would be increasingly useful as the intention is for the end users to be able to access the application using their own devices, which we do not want to place any restrictions on.

The cross-platform comparability provided by web applications is another beneficial inherent property, especially considering the ever growing smartphone market [33]. This essentially entails that the application can be utilized by any device that has a web browser. In this regard it is important to stress the significance of responsive design - web design that adapts to screen size in order to function well on different devices - and cross-browser support.

One could make the argument that responsive design encompasses cross-browser support, seeing as it aims to provide optimal viewing experiences, regardless of browser. However, cross-browser support is so important that we mention it specifically, because the application might fail in accomplishing its goals should it fail to properly respect this challenge inherent to web applications. Also, because there is still some disparity within browser usage [111], this challenge remains an important part of the design process.

The choice of creating a web-based application also provides an important benefit in relation to updating the application. Due to the time constraints of this thesis, we will most likely be unable to implement all the desired features in time for our deadline, and will instead finish a prototype with the key features. However, as the decision has been made to utilize the web platform, rolling out updates should be trivial, as only the server needs to be updated. This was another reason for choosing the web platform, as schools might be more inclined to try out an application that is easily upgradable, rather than a native program featuring cumbersome updating possibilities, if any.

Mitigating challenges based on age difference in target audiences

Another benefit of a web application is that all target audiences, with the exception of the youngest students, have extensive experience using such applications [106]. According to the statistics of social network application usage, one of the most growing segments among users are those in the age bracket of 35-44 and 45-54, which provides support to the claim that even older teachers have experience using similar applications as the one proposed in this thesis. Providing a familiar environment for both students and teachers can potentially have benefits for immersion, and certainly has merit in terms of usability, since the end users will inherently know what to expect from such an application.

It is important to note that the choice of using the browser as a platform does not necessarily imply that we could not create an educational video game in the traditional sense. The browser could still be used as a platform for a traditional video game, which is becoming increasingly popular [60]. The arguments given in the introduction of the chapter still hold, but we wish to illuminate that the benefits of creating a web application is not limited to the type of application we propose to create.

One of the identified challenges within the application domain is that of engagement and motivation of the end user, specifically in the case of the student. Creating a web application rises to this challenge through its ubiquitous access, allowing users to constantly interact with the application. It should be stated, in this regard, that merely by creating a web application one does not guarantee ubiquitous access per its definition [31], but one does go a long way in doing so - in the sense that browsers are extremely common.

One needs to keep in mind that the only real hindrance in this regard is that of device support, specifically support for smartphone devices through responsive design. We make the argument here that if a service is accessible through a browser, but does not function in the intended way based on the device that is being used, then ubiquitous access to that service has not been established.

Combined with the addictiveness of motivational game mechanics and social aspects (both of which appeal to the rewards centre of the brain), the constant interaction provided by ubiquitous access could potentially be one of the cornerstones of facilitating true learning - the most important goal of the proposed application. Had we chosen to create a native application, these combined benefits would have to be relinquished, which we argue is reason enough in itself to opt for a web based approach.

Content by Teachers

By providing teachers with a tool for organizing and structuring their own knowledge in the form of learning material and tasks, the application could accomplish the outlined goals, and successfully facilitate true learning. The results would be highly contingent upon the quality of the content, but as we have emphasised, teachers are the most qualified authors of said content, and we thus maximise the likelihood of success by making use of their expertise. To succeed at this, the aim is to integrate CRUD (Create, Read, Update, Delete) operations related to application content a seamless process.

Since a portion of the teachers in the Norwegian school system are of older age, and thus not as technology-savvy as their younger counterparts, the requirement of a gentle learning curve for the creation of content is important. To simplify the process of the CRUD operations of game objectives, the intention is to provide the option to link the tasks and tutorials together seamlessly - providing a natural progression for the students to follow.

Should the various created game objects be pedagogically well-designed and well organised, this architectural choice could potentially be most appealing element of the application. First and foremost, teachers will have a boiler plate set-up for each of their subjects, with a set of game artefacts representing every aspect of their curricula.

Modular design

By maintaining modular design through the development process, the various application modules should also be de-coupled from each other, and as such the content for each subject can be updated re-used infinitely, regardless of the end users. In this way, well-designed game objects could potentially substantially lower the administrative efforts of the teachers, since they only have to create said objects *once*. Of course the curricula may change, but by making the CRUD operations seamless this becomes a trivial matter. Taking the time to create game artefacts is a one-time investment, and a crucial aspect of the appeal of our application. This does however require the teachers to be convinced that this investment is worth their time.

Naturally, simply creating well-designed game artefacts is not enough in itself to achieve the benefits we have identified in our literature study; these artefacts still need to be coherently coupled with the outlined game mechanics in order to promote student engagement. However, the importance of creating content based on pedagogical expertise is paramount in order to ensure that students experience true learning - the game mechanics function more in the augmentation capacity. To put it simply: The mechanics keep the students motivated to use the application, while well-designed content ensures that they are truly learning.

8.2 Social features

As per our pre-thesis paper conclusion [52], one of the criteria the proposed application should aspire to fulfil is that of the social aspect. There are numerous benefits associated with the implementation of social features in web applications to promote user engagement [6], and, in accordance with the research done by Jane McGonigal, introducing social features to a game environment increases the motivation for time investment [68] in said environment. Thus the choice to implement social features that increase user agency and engagement, potentially in complement to the benefits resulting from the application's inherent game mechanisms.

Tapping into the growth of social media

The usage of social media websites continues to grow rapidly, which fits very well with research claiming the existence of a strong, innate desire within the human psyche to use social networking applications [27]. Tapping into this behaviour could yield even more engagement and motivation within the scope of the proposed application, but it should be noted that this is very treacherous ground - applications trying too hard to emulate features of popular social networking websites often fail to evoke the same response from their user base [97]. Thus, even though the architectural decisions allows for social features to be introduced into the application, care must be taken to avoid the pitfalls associated with such an endeavour.

A challenge identified in the previous chapter was that of appealing to both genders, with research indicating that males are more attracted to video games than their female counterparts. By distancing the project from traditional video games, and rather implement their underlying mechanics, moving towards a social web application, we hope to mitigate the distance between the genders - girls spending more time on social networking sites than boys [17].

We also discussed the importance of usability for the end user, both in terms of students and teachers. While we have previously discussed this challenge within the scope of creating a web application, the argument can also be made that the end users all have extensive experience using social network website [65], [106] and as such a web application using similar features would ease the learning curve and increase usability.

The importance of peer motivation

An example of a well-integrated social feature within the scope of the proposed application would be that of showing the progress of different school classes, and creating a sense of peer motivation to help your class to achieve victory. Seeing your progress in relation to the that of other students might also be a very motivating factor, as well as the entire social community aspect of having a discussion forum or something similar where subject matter can be discussed.

It should be noted that we have made no direct assumptions about what kind of social features we intend to implement in our application, but that rather we have made a conscientious architectural decision to include these kinds of features, due to their inherent benefits.

8.3 Desired Game Mechanics

This section will define the set of mechanisms we intend to utilize in the application. The game mechanics we deem most important would be those associated with facilitating *augmented* and *true* [52] learning, the latter defined as the concept of introducing meaning alongside knowledge.

Gradual progression

There are two main concepts related to game mechanics that are exceedingly important when we are talking about facilitating true learning, the concepts of *flow* and *fiero* [68]. Naches is also an important emotional state to discuss in this context, but it is not as connected to game mechanics as the two former examples.

There are several game mechanics associated with generating these emotions, flow generally resulting from a well-designed difficulty curve and addictive tasks, while fiero is usually the result of overcoming a large obstacle.

In order to induce flow, the application must encourage the student to work on progressively harder tasks and exercises in order to be constantly on the verge of her ability [19]. This game mechanism will base itself on the learning curve of the particular student, and is thus very specific to each end user. This brings on a whole new challenge; different students need to be challenged on different levels. This particular challenge is closely related to the decision to have application content be created by pedagogical and subject matter experts.

While flow certainly can be induced through adapting the difficulty curve and keeping students on the verge of their abilities, there is no formulaic way to achieve this. Also, if students find the particular tasks boring and/or tedious, this could easily break flow as well. Thus, in order to ensure the maximum potential for flow induction, we intend to introduce levels of gradual progression. Research has shown that games properly pacing their progress through gradually granting its users experience points and rewards are highly addictive [100] - popular examples include World of Warcraft by Blizzard Entertainment, and FarmVille by Zynga. The gradual progression might be in the shape of levels, badges and/or other rewards.

An emotional state also related to video games, but not that common in practice is *naches* which, as we discussed in our literary study, is a feeling of bursting pride experienced by a tutor, brought on by the success of his pupil. During the literature study we could not find examples of video games tailored around the concept of naches, which supports the argument that this emotion has received less attention than its more popular siblings: flow and fiero. Within the scope of this thesis, however, naches has a natural place. Naches appeals not only to the student completing game challenges, but also potentially of the *teacher* - or even better, other students. By providing students who have mastered certain subject matter the option of tutoring other students, we could potentially harness the benefits of naches, increasing student engagement even further.

The means of introducing gradual progression in the application will be the tried and tested concept of gaining experience points and levels through completing game objectives. By giving each objective a certain amount of points one could achieve the desired effects of flow by simply having students complete tasks within the application. However, it should be noted that the amount of points given for each game objective would necessarily be the result of some pedagogical deliberation in order to properly reflect the effort put into the objective by the student. As the system can make no guarantees regarding gradual progression on its own, only so in the context of carefully selected tasks. Thus, this feature will be designed with the aim of utilizing the pedagogical expertise of the teachers, and simply provide the necessary tools for these expert users to create a system tailored to the various subject matter.

Rewards

Another feature that will be used to induce flow is that of rewards for completing certain pre-defined game objectives. These rewards will be on the form of badges, titles or trophies. The exact implementation is of less importance than the concept it represents, but the specifics will be detailed in the application specification. Research has shown that a well-designed reward system can function the same way as gradual progression, in fact it is a manifestation of this very concept [68], in generating flow.

This process can then be supplemented by gradually superior rewards, further increasing user agency and engagement (for instance receiving badges, and then the possibility of creating her own unique challenges to present to other students). Because flow is so contingent on the difficulty curve and the authors of this project are no pedagogical or subject matter experts, the aim is for the application to become a framework facilitating these features, and provide the tools in order to allow the teachers to create the necessary and properly weighted game challenges.

Fiero: a more challenging emotion

The goal of inducing fiero might be harder to accomplish, especially considering that it takes a very well-designed game to have this effect on its players [68]. Considering the context of the proposed application, and the fact that many students are already very de-motivated concerning school work [90], any feature designed with the intention of generating fiero would need to be extremely engaging to the student and of exactly the right amount of difficulty according to her ability. We have already emphasised the importance of a gradual and well-weighed difficulty curve in order to induce flow, but the generation of fiero is also highly contingent upon this factor as well. While gradually increasing difficulty is not as important in this regard, creating the sensation of overcoming a major obstacle without said obstacle feeling like too much of a hindrance to the player is a very demanding process. This is again within the domain of the pedagogical, and as such the same arguments pertain here as in the concept of generating flow - our application should aim to facilitate this process, and provide the teachers the necessary tools for accomplishing these features.

An example of a successfully implemented feature in the application scope we have defined inducing flow and fiero might be a set of tasks constantly keeping the students at the edge of their ability, with *one final* task relevant to the subject matter the students have just worked extensively with. This final task would require some extra effort on the students' part, but would ideally not suffer from artificial difficulty [108] (i.e. difficulty stemming from unreasonable demands on the students' knowledge, for instance something not presented to the student in the previous tasks - which is exceedingly common in traditional school work exercises [91]).

Well-designed game challenges

The application content should be the result of the teachers' pedagogical deliberations, and weighed to fit the desired learning curve of the students. A modular approach when designing the application should also yield the possibility for teachers to customise each task to suit individual student needs, which is incredibly important when facilitating true learning [49].

Many of the domain-specific risks identified are also mitigated by the architectural choices of creating a web application implementing select game mechanics. For instance, the risk of game objectives not being congruent with learning is nearly completely mitigated due to this choice, seeing as all content within the scope of the application are related to subject material. This claim can also be made for the risk of distracting players through too engaging gameplay mechanics. Even though some of these mechanics, like a certain reward, might not intrinsically be related to subject matter and/or knowledge, they are always received as a result of completing a game objective, which is clearly related to these concepts.

Progress feedback

While progress feedback can easily be achieved in traditional video games by showing very intuitive graphical elements in the user interface (for instance a progress bar, a character level indicator or even simply increasing the size of the player's character), this needs to be incorporated within the setting of our scope - a web application.

Granularity is also of great importance in the regard of progress feedback, seeing as the scope of our application could potentially encompass all the subjects the students are working on in school - which would include a notable amount of learning-related game artefacts. In order for the student to not become overwhelmed and/or demotivated by the sheer volume of artefacts, the aim is to provide relevant progress feedback at different levels of granularity.

Peer motivation in the context of games

Recall from the previous section that we intend to introduce social features to the application to harness the potential within peer motivation. The aim is to use these social features in union with the inherent benefits to peer motivation within games. Based upon research done in approval motivation, user engagement could benefit greatly from sharing notifications about rewards among peers in the application [29], [67]. An example on how to accomplish this would be to notify all students in the same class whenever one of their own have completed an achievement, or show a leaderboard detailing the progress made by each class in relation to each other. The subject of peer motivation also raises another important point, intrinsic versus extrinsic rewards and motivation.

The game mechanism of providing well-paced rewards (in the form of badges, titles and so forth) and gaining levels are all examples of *extrinsic* rewards, game artefacts that are directly related to the application in an *explicit* manner. Intrinsic rewards are based on fulfilling the end user's intrinsic goals and motivations, such as the goal of social connection and meaning.

For the sake of clarity we need to point out that there are two very different aspects to consider when discussing intrinsic versus extrinsic rewards in the context of games. Emotions, such as flow, fiero and naches, are intrinsic rewards due to well-designed game mechanics, while extrinsic rewards in the scope of video games are virtual game objects, like bades, achievements and so forth.

Peer motivation fits very well into this discussion because it relies heavily on intrinsic reward, more specifically *social connection*. Since motivation based on intrinsic reward is very powerful [68], and as such this could provide a huge boon to user agency and engagement. For this reason, peer motivation will be an important focus for us when we develop the application prototype.

Chapter 9

Functional and Non-Functional Requirements

In order to formally carve out how we envision the system, we will present a requirements specification, consisting of both functional and non-functional (henceforth referred to as **NF**) requirements. This chapter provides the functional and NF requirements for the application we wish to develop, based on the literature study in Part III. These requirements are highly affected by the various risks and challenges identified in Chapter 7, both application and domain specific. They are meant to mitigate the risks and challenges to a satisfactory extent.

Since functional requirements are often concerned with *specific component behaviour* [112], whereas non-functional requirements generally operate within a wider scope, we will first identify the non-functional requirements.

Terms and Relationships in the requirements

Some terms are used frequently in the requirements, and will be key components of the application. The application will contain many *subjects*, these subjects may contain many *topics*, and these topics will contain *tasks* for users to perform. Users may be *students* or *teachers*. An illustration of this can be seen in Figure 9.1. Note that this is a tentative model, and is likely to be reviewed in more detail at a later stage.

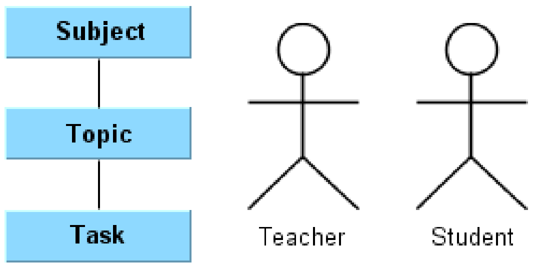


Figure 9.1: Tentative data relationships and user roles.

9.1 Non-Functional Requirements

This section will present the non-functional requirements for the proposed application. The requirements are designed to define an application that fulfils the previously outlined goals, as well as mitigating the mentioned risks and challenges. Many of the NF requirements identified in the following subsections will have sub-requirements that can be defined as functional; this is a direct result of the *data requirement* NF type [26].

Many of the identified goals of the application will be directly related to the NF requirements described in this section. We have previously emphasised the challenge of creating an application properly employing game mechanics that have been found to increase engagement [68] in union with attempting to facilitate true learning among the student end users. The focus on user engagement and motivation will be heavily featured in the NF requirements, as this is an area that pertains to *quality requirement*, which is not easily related to functional requirements, but rather their NF counterparts [26].

The non-functional requirements will first be presented in an application-wide context, and then based on the application's separate modules - such as game mechanics, social features and so forth.

General NF requirements

ID	**Description**
G-NFR1	The application must have an intuitive interface with respect to the different target audiences
G-NFR2	The application should make it a trivial process commence relevant school work from anywhere in the application
G-NFR3	The application should have a modular design so that its visual components can be dynamically changed to reflect the preferences of different target audiences
G-NFR4	The application should be ubiquitously accessible
G-NFR5	The application should provide at least 99 percent uptime
G-NFR6	The application should only store relevant data pertaining to student learning, not other sensitive information
G-NFR7	The application should implement a strict role system to enforce access authorization
G-NFR8	An authenticated user can access all data per the access level defined in the user's associated role(s)
G-NFR9	The application should make use of the latest developments in Javascript and HTML to create a highly interactive interface to promote user agency and engagement

Table 9.1: General non-functional requirements

Component-specific NF requirements

Game mechanics

ID	Description
M-NFR1	The application must make use of *augmented learning* through its inherent game mechanics
M-NFR2	The application's inherent game mechanics should use documented approaches to induce *flow*, *naches* and *fiero*
M-NFR3	The application should keep students up to date on their recent task progress
M-NFR4	When a student works on a task she should be have access to her progress in all levels of granularity
M-NFR5	The tools for creating game artefacts should be very intuitive
M-NFR6	The application should provide teachers with the option of linking together relevant topics
M-NFR7	Teachers should be able to tailor the difficulty curve of each task
M-NFR8	The application should facilitate simple management of the organizational hierarchy of topics
M-NFR9	The application should allow the properties of the game artefacts to be altered depending on the preferences of target audience through its inherent design
M-NFR10	Upon completing a task, the student should be presented a related task immediately
M-NFR11	The application should make use of interactive graphical elements to present tasks to the student in a novel way in order to promote user engagement
M-NFR12	The visual presentation of the application should be the result of target audience preferences
M-NFR13	The application should allow students who meet certain criteria to create their own tasks to promote peer motivation
M-NFR14	The application should have an extensive reward system featuring different kinds of rewards
M-NFR15	The application's reward system should have a modular architecture so that it can be altered based on target audience preferences

Table 9.2: Non-functional requirements related to game mechanics

Social features

ID	Description
S-NFR1	Every student has to all their accomplishments and rewards through their profile
S-NFR2	Students should be able to view the public profile of other students
S-NFR3	Students should be able to choose whether or not their task progress is visible to their peers
S-NFR4	Whenever a student completes a game objective or task, this should be reflected on her profile
S-NFR5	When a student works on a task, she should be able to see which of her fellow students have completed that task
S-NFR6	A student should be able to offer to assist other students with certain game objectives or tasks
S-NFR7	The application should facilitate student communication regarding school work
S-NFR8	The collective progress of a class of students should be easily viewable and contrasted to the progress of other classes in the same grade

Table 9.3: Non-functional requirements related to social features

Learning concepts

ID	Description
L-NFR1	The application should accumulate relevant student learning data and present it to teachers in a meaningful way
L-NFR2	Teachers should have access to all levels of information granularity regarding a student's school work
L-NFR3	Teachers should be able to easily group students together in classes, and assign which subjects each class will be enrolled in
L-NFR4	The application should accumulate relevant class statistics and present them to teachers in a meaningful way
L-NFR5	Students should be able to ask relevant questions directly connected to a given task
L-NFR6	The application should make relevant learning resources available to the student when she is working on a task
L-NFR7	The teachers should be able to group together all their created topics into subjects based on subject matter
L-NFR8	When a student is using a learning resource the system should store information regarding when she started using it and if she finished interacting with it

Table 9.4: Non-functional requirements related to learning concepts

9.2 Functional Requirements

The functional requirement specification begins by outlining requirements that specify system behaviour on a general scale, before moving on to component-specific behaviour.

For the sake of brevity, the most obvious functional requirements will not be included - those that are common to all modern web applications. Instead the focus is on the functional requirements that are specific to the proposed application.

Examples of functional requirements we consider superfluous in this regard would, for instance, be requirements like "When an unauthenticated user clicks the "sign up" link he is shown an HTML form where he can enter his information" or "Should an unauthorized user attempt to access a protected resource he will be shown an error page".

General

ID	**Description**
G-FR1	Once the user logs in he will be shown navigational links based on the access level associated with his role
G-FR2	When a user is created he will be given a defined role which defines his access level within the application
G-FR3	The system shall support two languages: Norwegian and English
G-FR4	The system shall perform client-side validations for all form inputs in accordance with the principles of responsive design
G-FR5	The system will have defined roles that separate the access levels of teachers and students

Table 9.5: General functional requirements

Users

ID	**Description**
U-FR1	When a user creates a new account the system will send him a confirmation email in order to validate his account
U-FR2	When a user enters invalid information the system will provide him relevant validation error messages
U-FR3	When an authenticated user accesses the profile of another user, the application will display the profile information that user has made public
U-FR4	When an authenticated user clicks the ”profile” link, he is taken to the profile page
U-FR5	When a student views her profile, she is presented with information about school work progress
U-FR6	When a student is views her profile, she is presented with information about achieved rewards
U-FR7	The profile page of a student will present her user information, as well as associated roles and classes.
U-FR8	When a student is views her profile she is provided the option to edit her user information
U-FR9	When a teacher views the profile of a student, the system presents all student information, regardless of privacy settings

Table 9.6: User-related functional requirements

Classroom

ID	**Description**
C-FR1	The classroom page will display the names of all enrolled students
C-FR2	The classroom page will display the accumulated progress of the class sorted by subject
C-FR3	The classroom page will display all subjects the class is currently enrolled in
C-FR4	The classroom page will showcase all rewards earned by the class

Table 9.7: Classroom-related functional requirements

Game Artefacts

ID	Description
GA-FR1	When a student clicks the ”practice” link, the system will display a list of subjects she is enrolled in, and ask her to choose which subject to work on
GA-FR2	When a student views the page of a subject she is enrolled in, the system will display her progress in the subject, and topics belonging to the subject
GA-FR3	When a student clicks on the name of any topics belonging to a subject, she is taken to that topic's page
GA-FR4	When a student views the page of a topic she can view the names of all children and parent topics
GA-FR5	When a student starts working on a new topic, the system will immediately store the relationship between the topic and the user
GA-FR6	If a student submits an incorrect answer when trying to complete a task, the system should provide her with an error message and store the erroneous attempt
GA-FR7	The task page should feature a ”show hint” button to allow students to view hints to help them arrive at the correct answer
GA-FR8	When a student clicks the ”show hint” button, the system should persist the number of hints a student has used when working on the corresponding task
GA-FR9	When a student completes a task, the system will provide a response message to congratulate the user, and present the user with the page of a new, related task
GA-FR10	Each task page should contain links to all related tasks and learning resources
GA-FR11	Teachers and students fulfilling certain criteria will be provided links to create, delete or update tasks
GA-FR12	The task page will contain a progress bar that is updated whenever a student makes progress
GA-FR13	When a student completes a task, the system will automatically grant the student points based on how many points the task is worth
GA-FR14	Whenever a student is working on a task the application should provide links to relevant learning resources on the game artefact page

Table 9.8: Topic- and task-related functional requirements

Rewards

ID	Description
R-FR1	When a student completes a task or general objective, the system will present the student with a graphical representation of any associated rewards
R-FR2	When a student receives points for completing a task or objective, the application will check her score against thresholds. If the student's points exceeds a pre-defined threshold, she will gain a level
R-FR3	Whenever a reward is issued to a student a notice will pop up informing her of it

Table 9.9: Reward-related functional requirements

Teaching

ID	Description
T-FR1	When a teacher inspects the profile of a student, he should be given links to inspect the student's progress in her current subjects
T-FR2	By clicking any of the topics related to a student, the teacher should be given detailed information about the relationship between each task and the student, such as failed attempts, hints used and so forth
T-FR3	When a teacher is inspects the progress of a student in a certain topic the system will provide him links to the progress of the student in all topics
T-FR4	When creating a new subject, the application will show topics and classes within the system that the teacher may associate with the subject.
T-FR5	When a user creates a new topic, the application should list all previously created topics
T-FR6	When a teacher associates different topics, the application should store this hierarchical relationship and represent it visually using JavaScript
T-FR7	Students who have completed a topic should be presented with a link on the topic page to allow them to become mentors for other students
T-FR8	When a student is interacting with a tutorial the system shall store the time and whether or not she completed it
T-FR9	The system should support both textual and video tutorials

Table 9.10: Teaching-related functional requirements

9.3 Summary

This chapter outlined all the functional and non-functional requirements which together constitute a complete requirement specification for the proposed application. The requirement specification begins by focusing on the NF requirements because they tend to encompass application-wide behaviour, and are very closely related to effects caused by application *qualities* in contrast to functional behaviour.

After first discussing the various NF requirements, beginning with an application-wide scope, and then moving on to component-specific requirements, we able to formally state the desired qualities the system should achieve, per the NF requirement definition [26].

Once the NF requirements had been outlined, we began identifying the *functional* requirements we wish for the application to fulfil, many of which were a direct result of the previously identified NF requirements. All the functional and NF requirements presented in this chapter were the result of the risk and challenge modelling and architectural rationale.

Chapter 10

Application Specification

Based on the risks and challenges, architectural rationale and the requirement specification, this chapter attempts to synthesise all these design documents into a final *application specification.* The implementational details regarding all the concepts introduced earlier will be presented, in addition to a relational data model showing how these modules constitute the application.

10.1 Application components

To clearly define the various components of the application we will to explicitly state the various objects that constitutes the proposed application. We have purposefully used vague terms to avoid making any implementational assumptions in the design process. We will now present a more explicit application specification. The Data Model in Section 10.2 will explicitly state the relationships between these objects as well as their attributes.

To fulfil the requirements regarding the separation of user access through a role system we have defined the following objects: **users** and **roles**. Whenever a user is created it is given a default role with a certain access level, which would be a good way to separate operations available to teachers versus students.

For the purposes of the application we have defined the following roles: *student*, **teacher**, **administrator**, **mentor** and **advanced student**. We introduced the roles *mentor* and *advanced student* due to the requirements pertaining to peer motivation.

We have defined the following objects to fulfil the requirements specified:

skills, **tasks** and **rewards**. A *skill* can consist of several *sub-skills* (requiring an hierarchical data structure) and/or *tasks*. Each skill and/or task could have an associated *reward*, which would be decided by the user creating the artefact, which would be triggered when a student completes the artefact's objective.

We have emphasised the importance of properly structuring the application's content (skills and tasks) in order to gauge the students' progress in various subject matter. The obvious choice for providing a context for the artefacts would be to order them by subject, and provide an interface for teachers to see student progress. In accordance with the requirement specification we define the following objects in this context: **subjects**, **answers**, *tutorials*, **hints**, **questions**, **skill progressions** and **classes**.

A subject consists of several skills, which is a way of methodically systematising content by subject matter, an important aspect discussed at length in our architectural rationale. This could also allow for the searching of skills based on subject, which might be a feature to introduced at a later staged.

In order to provide teachers with an interface through which they can see the progress of students, we have introduced the **answer** object. Every time a student attempts a task, a new *answer* object is created, storing all pertinent information (task and user ids, the student's answer, whether it was correct or not and so forth). This way the system can allow teachers to fetch information at the highest level of granularity concerning the school work of their students.

We have talked about creating game artefacts that would work as learning resources as well as tasks and exercises. To achieve this, we wish to employ objects called **tutorials** and **hints**. The former would act as a stand-alone artefact students could interact with to help them with their school work - such as a textual or video tutorial. The latter would be tightly coupled to tasks, with each *task* having several *hints*, allowing students to show hints to help facilitate augmented learning.

The *question* object functions much in the same way, allowing students to ask questions regarding certain tasks, skills and/or tutorials. By storing information about what artefact the question pertains to, these objects can be easily searchable as well, and rewards could also be added to give students further incentive to ask questions.

Tutorials work in the same way as skills - by having teachers create them and associate them with subjects depending on their subject matter. We feel this object could strongly contribute to facilitating augmented learning, much in the same way they employ tutorials in Khan Academy [56]. They could also have rewards associated with them, in the same way as other game artefacts.

In order to properly convey the progress each student makes in a particular skill, this data needs to be stored in an object. Creating a distinct data structure for this purpose is in adherence to the requirement specification, hence the *skill progression* object. This structure contains all information regarding when a student started working on a skill, how many of the skill's sub-skills and/or tasks she has completed, if she has achieved its related rewards and so forth.

The teacher has to receive data not only for each particular student, but also for the class as a whole. For this reason we introduce the *class* object, which will consist of many *users* with the role *student* and *one* user with the role *teacher*. This object can store all the necessary aggregated progress information through its associated user objects (students) as per the requirements.

10.2 Data Model

Section 10.1 discussed the various components (objects) needed in the application to adhere to the requirement specification. To explicitly convey how we envision the objects, their inherent attributes and their relationships this section provides a set of figures covering the topic.

Users and roles

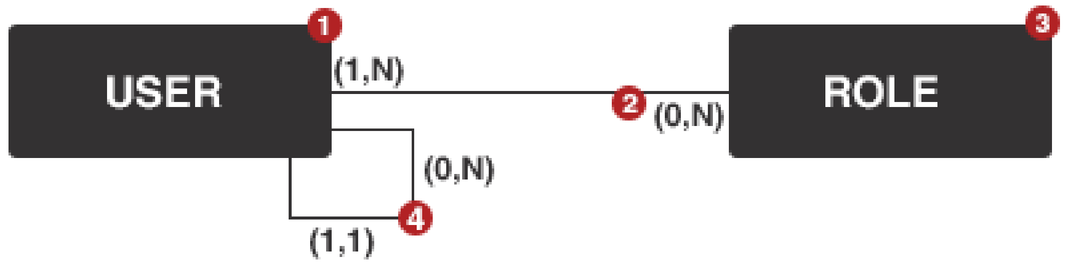


Figure 10.1: Data model for the relation between users and roles

1: The *user* model contains all pertinent user data, such as first name, last name, password, email, points, level and so forth.

2: The relation between *user* and *role* is that of **many-to-many**. One user can have many roles and a role can belong to many users. This way, a user can both be a student and a *mentor*.

3: The *role* model contains the access level of the role and the role's name: *teacher*, *student*, *administrator*, *mentor* or *advanced student*.

4: Should the student achieve a certain level, she can be promoted to *mentor* or *advanced student*. These roles allows her to assist other students, which is shown through this relation; any student with the roles *mentor* or *advanced student* can tutor **0..N** other students, while a single student can have **only one** mentor.

Users and game artefacts

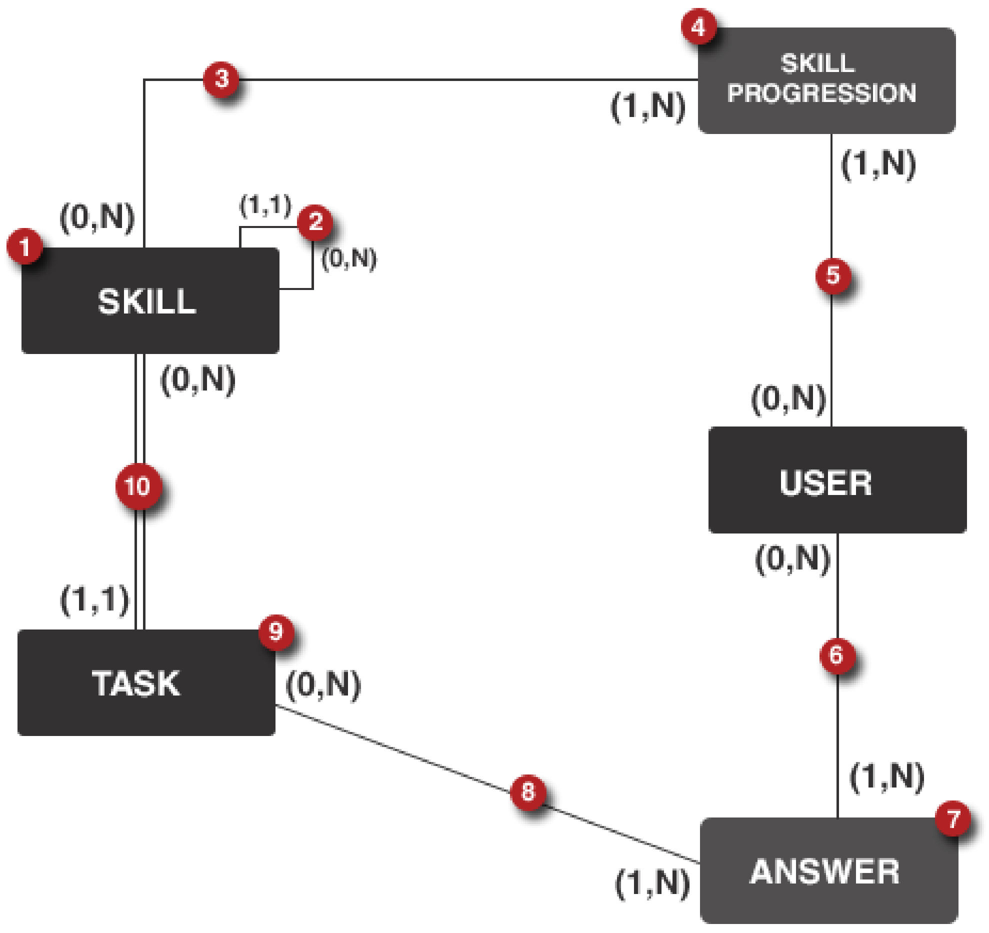


Figure 10.2: Data model for the relation between users and game artefacts

1: The *skill* model contains the following fields: *name*, *description*, *points*.

2: In adherence to the requirement specification, the skill object has a tree structure - allowing one skill to have several sub-skills in addition to any potential tasks. Any skill can at any time have **0..N** sub-skills, but **only one** parent. The skill will be considered completed when all tasks *and* sub-skills are completed.

3: A skill can at any time have **0..N** skill progressions associated with it, while a skill progression needs to have *at least one* skill associated with it. This is the way students are associated with skills.

4: The *skill progression* model contains the following fields: *date started*, *date completed*, *status* and *current progress*. By having access to this detailed information, teachers can see when students began working on certain skills, and how long they have taken to complete them. They can also see students' current progress, which could give an indication which students are having problems with particular skills. When a skill is completed, the status is updated and the student is awarded the designated points of the skill. Skill progressions need both an instance of *user* and *skill* in order to exist, which is the reason for the slightly faded colour.

5: When a student starts working on a skill, a new skill progression object is created to store information about when the work was commenced.

6: When a student starts working on a task, a new answer object is created, storing information about when the work was commenced.

7: The *answer* model contains the following fields: *date started*, *date completed*, *failed attempts*, *status*, *hints used*. In order to adhere to the requirement specification, data regarding student school work at the highest level of granularity needs to be stored. The answer model is the result of that requirement. By storing data not only about when the student started working on a task, but also how many failed attempts she has and how many hints she has used, teachers have access to information that can potentially be very valuable . Answers need both an instance of *user* and *task* in order to exist, which is the reason for the slightly faded colour.

8: A task can at any time have **0..N** answers associated with it, while an answer needs to have *at least one* task associated with it. This is the way students are associated with tasks.

9: The *task* model contains the following fields: *name*, *question*, *solution* and *points*.

10: A task belongs directly to a skill, in a **one-to-many** relationship. This is shown by using two lines to show how each task object belongs to a particular skill object. A skill can have **0..N** tasks, while a task needs to belong to **only one** skill.

Users, game artefacts & learning resources

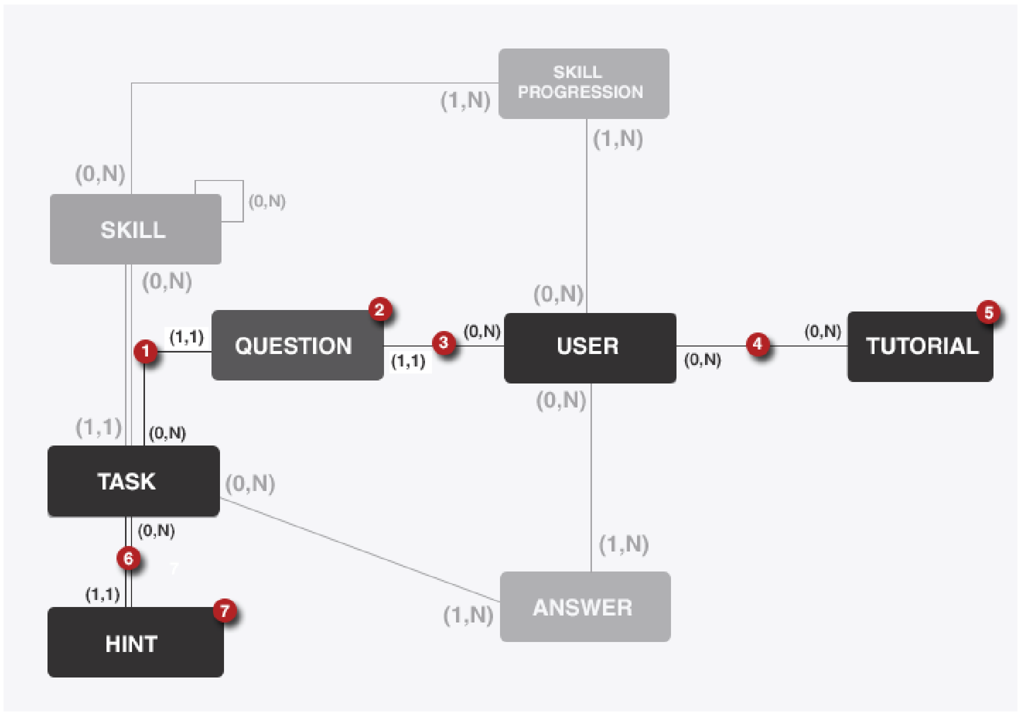


Figure 10.3: Data model for the relation between users, learning resources, skills and tasks.

1: Should a student be stuck on a particular task, she can create a new question object which relates to that task. A task can at any time have **0..N** questions associated with itself, while a question needs to have *only one* task associated with it.

2: The *question* model contains the following fields: *question*, *date created*, *status*, *points*, *date answered*. Questions can be answered by teachers and all students, but points are only awarded to students for answering them. Teachers can assign points to questions in order to give students incentive to answer them. When a student answers a question she is awarded the associated points and/or rewards.

3: Users can at any time have **0..N** questions associated with them, while a question needs to have *only one* user associated with it. This relation also stores the field *status*, which indicates whether the user asked the question or answered it.

4: If a student is stuck on a skill and/or task, or just want to start working on a new topic, she can view a video tutorial or read a textual one. In order to provide teachers with highly granular information about the students' learning process, the date when the student started interacting with the tutorial will be stored, along with information regarding whether or not she finished the tutorial through a *status* field. Saving this information allows for rewards to be associated with tutorials.

5: The *tutorial* model contains the following fields: *text*, *video url*, *description*. We intend to implement both textual and video tutorials in the proposed application.

6: Should a student be stuck on a particular task, she can view hints relevant to the current task she is working on. A hint belongs directly to a task, in a **one-to-many** relationship. This is shown by using two lines to show how each hint object belongs to a particular task object. A task can have **0..N** hints, while a hint needs to belong to **only one** task.

7: The *hint* model contains only one field: *hint*. This stores the textual hint as a string, and whether or not it was used is stored in the *answer* model through its task association.

Game artefacts and learning resources

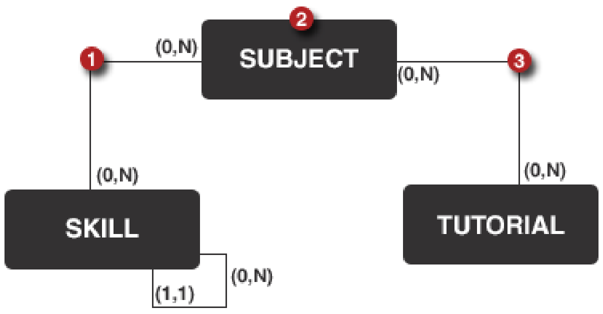


Figure 10.4: Data model representing the relations between tasks, skills and learning resources

1: Each subject consists of a set of skills. If a skill is associated with a subject, all sub-skills of that skill will also be associated with the same subject. An important aspect to notice about this relationship is that it is **many-to-many**, meaning that at any time a particular skill can be associated with **0..N** subjects, and any subject can be associated with **0..N** skills. This allows for skills to be shared between subjects, for instance if they have overlapping curricula, and could also allow for skills to act as cross-subject projects. Also, since tasks directly belong to skills, they become associated with subjects through their parent skills.

2: The *subject* model contains two fields: *name* and *description.* The name of the subject will indicate its subject matter (for example 'basic algebra', 'napoleon', 'organic chemistry' and so forth) and the description may contain important information, such as related learning materials and so forth.

3: Tutorials can, in the same way as skills, be associated with subjects. This allows teachers to organize all topics in a subject in different tutorials, which can be linked to particular skills through the subject's description.

Users, classes & learning resources

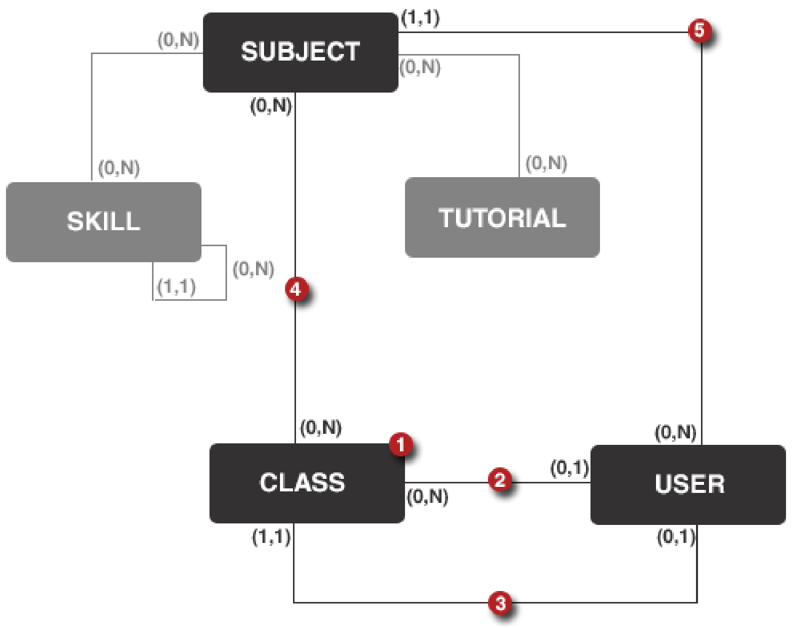


Figure 10.5: Data model representing the relations between users, classes and learning resources

1: The *class* model contains the following fields: *year*, *name*, *description*. The year indicates the school year of students in this class. The name of the class and its description is how classes in the same year are separated.

2: This relation represents the **student** relationship between students and classes. A user with the role *student* may at any time be enrolled in no more than *one* class, which would contain information regarding which *year* the student is in.

3: This relation represents the *class teacher* relationship between teachers and classes. A user with the role *teacher* may at any time administer no more than *one* class (as is customary in the Norwegian elementary school system). A teacher can, however, be associated with any subjects.

4: A class can at any time be enrolled in **0..N** subjects, and a subject can at any time have **0..N** classes associated with it, per this **manyt-to-many** relationship.

5: This relation represents the *subject-teacher* relationship. A user with the role *teacher* may at any time teach **0..N** subjects (i.e. he will be responsible for the skills within the subject, and will issue tasks to students), but any one subject may have **only one** teacher.

Users and rewards

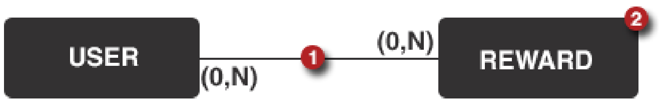


Figure 10.6: Data model representing the relations between users and rewards

1: Users with the *student* role can achieve any number of rewards, and any reward can be associated with **0..N** students, which is indicated by the **many-to-many** relationship.

2: A reward will contain the following fields: *name*, *type*, *points worth*. The intention is to have a plethora of different rewards (titles, badges, achievements and so forth), so this might very well be implemented as an abstract super class with many different sub-classes. We have also considered the option of having students purchase rewards by using their gained points (in contrast to gaining rewards by completing skills and tasks they are directly related to), which is the reason for the *points worth* field.

Game artefacts, learning resources & rewards

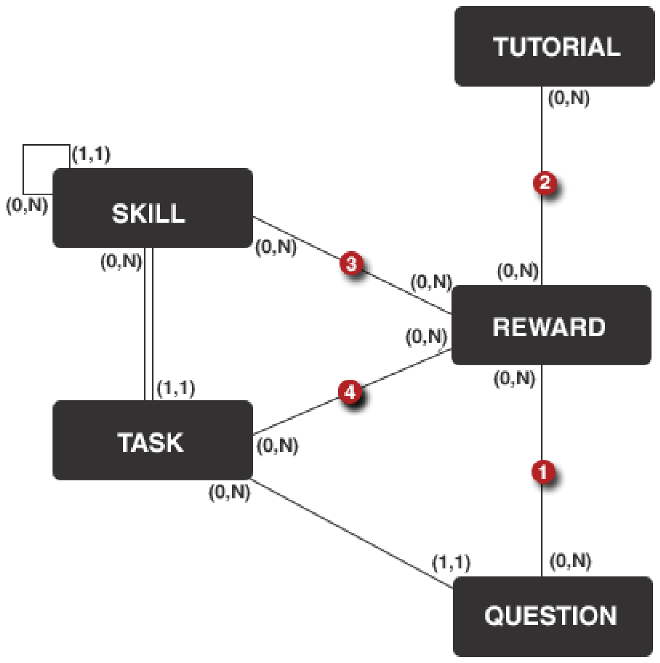


Figure 10.7: Data model representing the relations between game artefacts, learning resources and rewards

1: When a student asks a question pertaining to a particular task, students have the option to spend some of their own points to associate rewards with this question. In addition, the application associates certain rewards with answering questions in general, to give students incentive to answer them and thus increase peer motivation (for instance a reward called 'question master'). As per the **many-to-many** relationship, a question can have **0..N** rewards related to it, and a reward might be related to **0..N** questions.

2: To motivate students to read and watch the various tutorials, the application to provides them with rewards for doing so, as per this relationship. The **many-to-many** relation indicates that a tutorial can have **0..N** rewards and that a reward can be associated with **0..N** tutorials.

3: Providing rewards for completing skills is a very important requirement, and this relation represents that requirement. The properties of the **many-to-many** relationship indicate that a reward can be associated with many skills and vice versa.

4: In the same vein as gaining rewards for completing skills, there may also be specific rewards associated with particular tasks. This **many-to-many** relationship has the same intrinsic properties as the other discussed in the context of this figure.

The entire application

Figure 10.8 provides a data model representation of the application as a whole, containing all entities and their relationships.

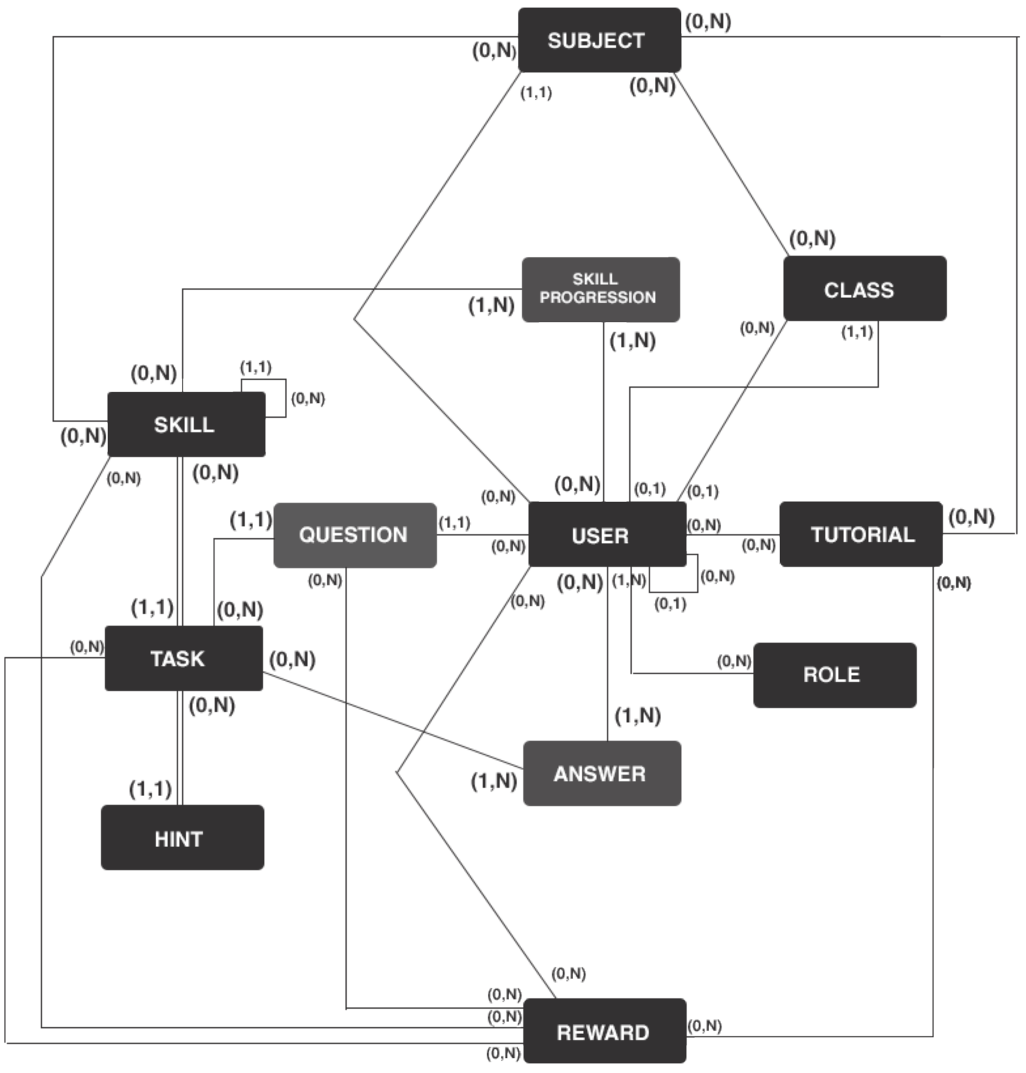


Figure 10.8: Data model representing the entire application

10.3 Software Testing

By software testing in this context, we are referring to automated software tests that check for bugs and inconsistencies. While there will not be time for an extensive software testing scheme, we plan to write some unit tests and functional tests in order to ensure that the basic functionality of the prototype works as intended, and no unexpected surprises cause problems during user testing.

To write and run the automated software tests, we will use the tools provided with the Ruby on Rails framework. These tools makes it easy to write tests by providing skeleton test code. In addition, it allows tests to simulate web browser behaviour, simplifying the otherwise time-consuming process of testing the interface [95].

Part V

Results

Introduction

This part of the thesis presents the application prototype produced as a result of the previous parts. An overview of the prototype, and an implementation report to illuminate how specific challenges were solved.

Ideally, this part would also contain results from field tests, but unfortunately, we were unable to find a school that would allow us to conduct these tests. A high number of local elementary schools were contacted, without success.

During the development process it became apparent, due to the limited time span of the thesis project, that the prototype would not fulfil all the NF and functional requirements specified in Chapter 9. Thus, the prototype development was focused on a subset of the requirement specification in order to create a system that would contain the features deemed most essential in accordance with the findings of the literary study.

This application prototype would ideally contain *all* the game mechanics and social features discussed in the architectural rationale, but the time constraint forced us to prioritise which features would eventually be implemented. It follows from this constraint placed upon the prototype application that much of the risk and challenge mitigation attributed to the requirements specified in Chapter 9 would be limited.

Chapter 11

Application Prototype

This chapter briefly presents the key features of the result of the prototype development. To properly visualise the prototype's essential features, it is provided in the form of screen shots of actual user interaction, along with short explanations. This is intended to supplement the more in-depth coverage of the prototype featured in Chapter 12.

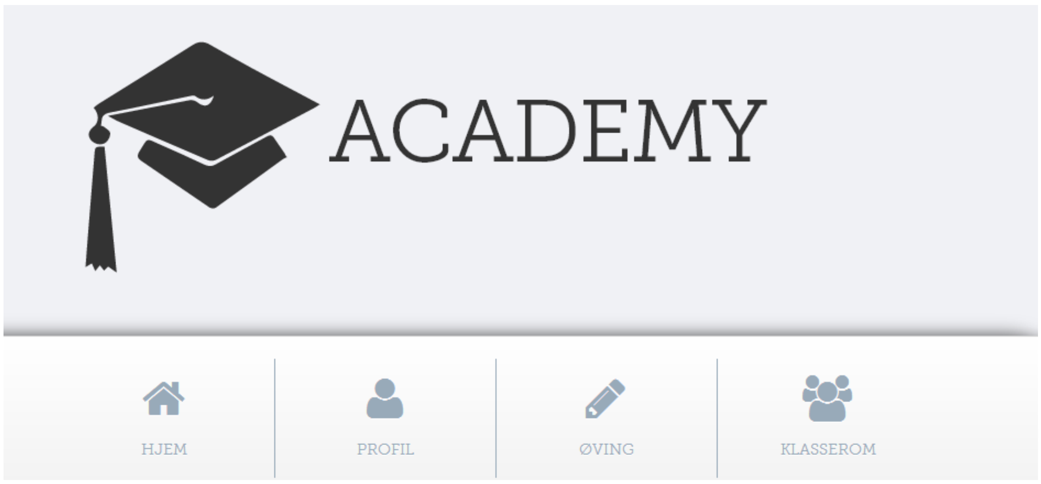


Figure 11.1: Landing page. The application has a simplistic design, with four navigation links at the bottom. These links allow users to quickly move to either their profile, a list of subjects to work on, or their virtual classroom. The simplistic design was a conscientious choice made in order to appeal to students who have had experience using social media websites, as the application attempted to mimic some of the visual aesthetics of such a site.

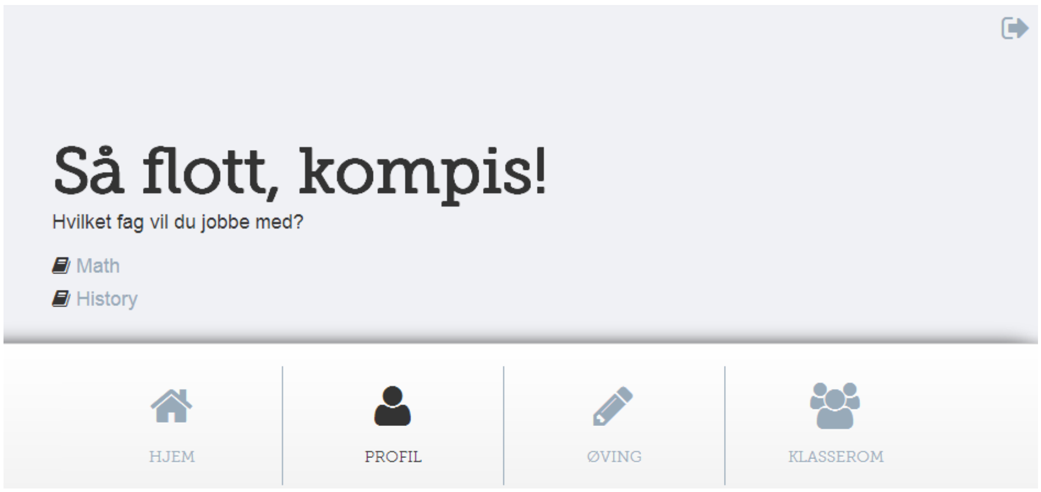


Figure 11.2: The practice page lists the subject the user may work on. Clicking any of these will take her to the index page of that subject.

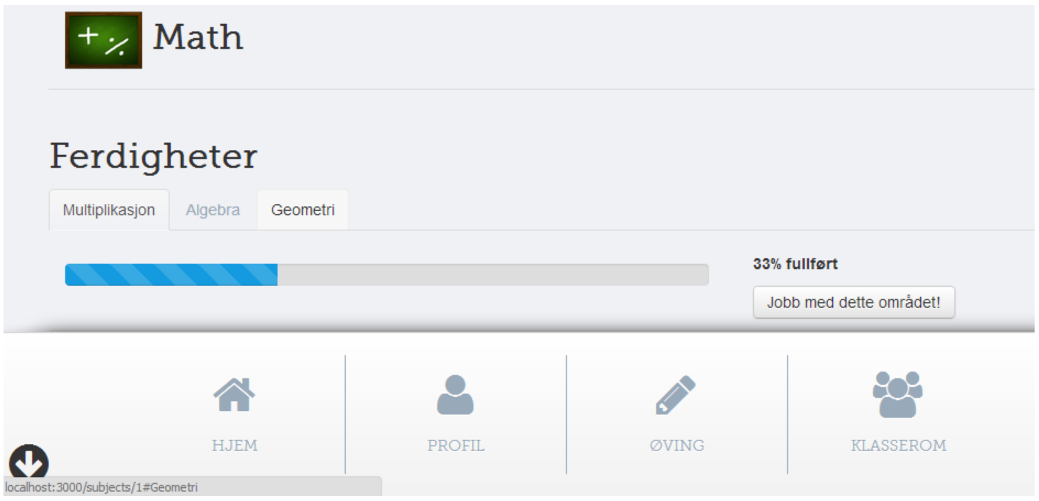


Figure 11.3: The index page of a subject shows the available subject areas (skills) to work on, and the students' progress in those skills.

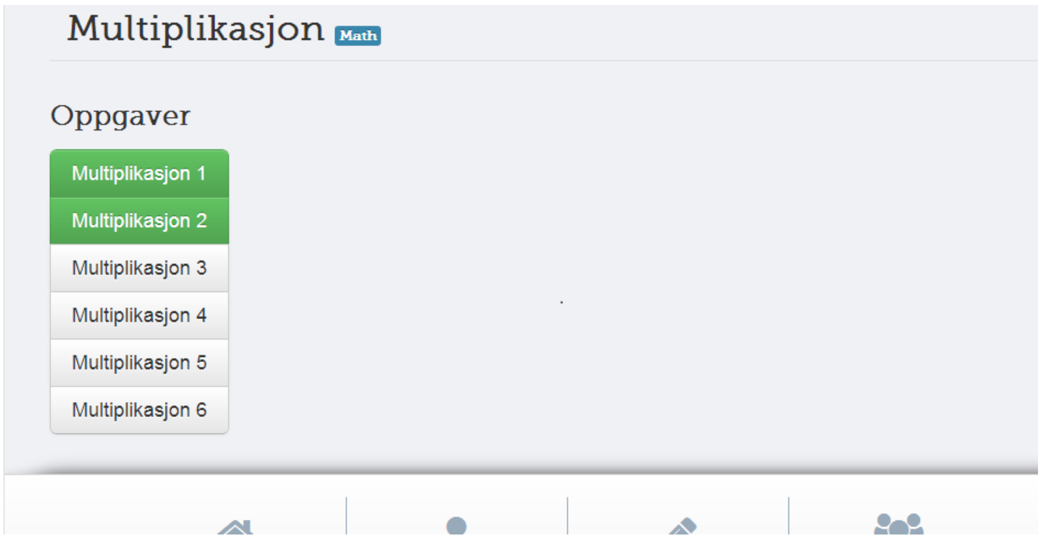
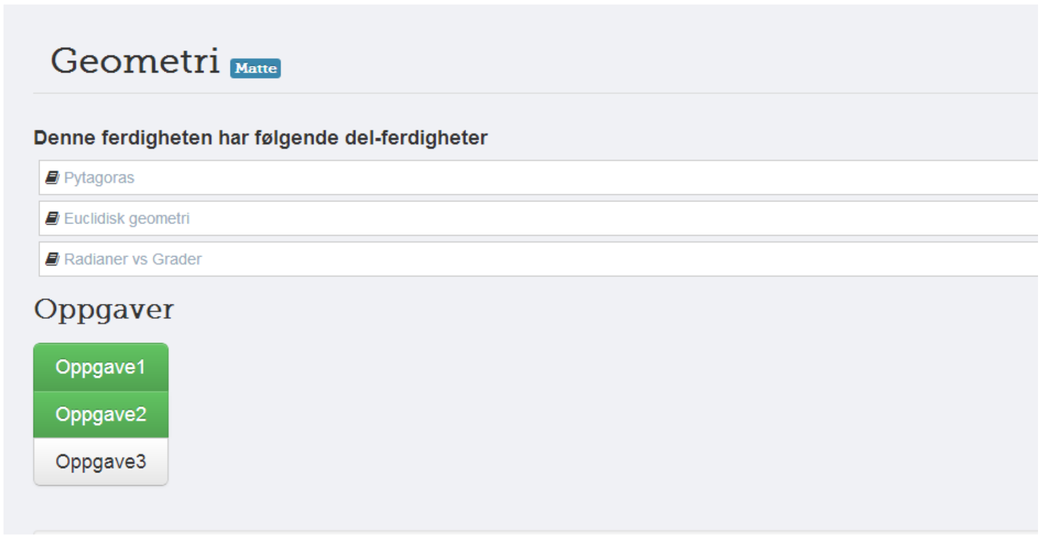


Figure 11.4: The index page of the skill "Multiplikasjon". Tasks in this skill are listed, and completed tasks are marked with green.

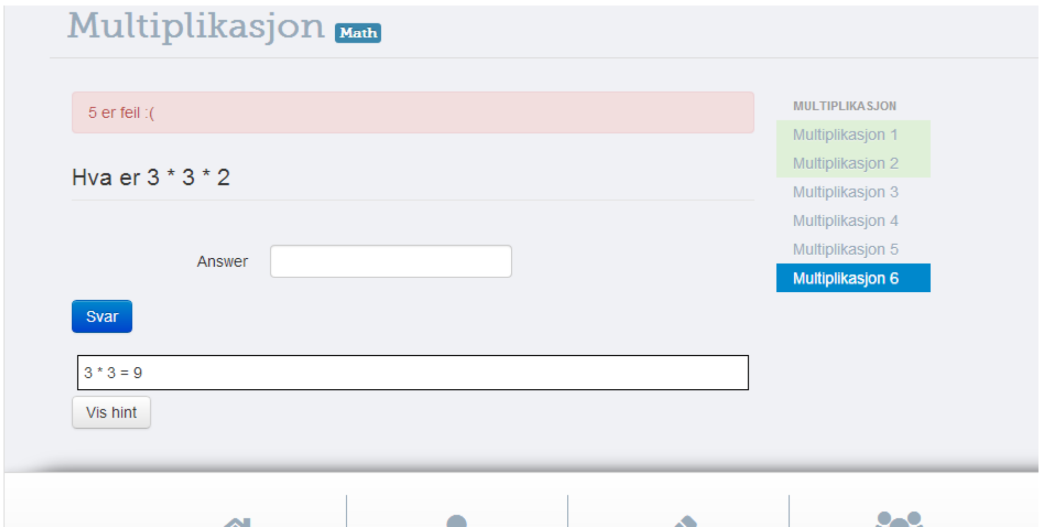

Geometri Matte

Denne ferdigheten har følgende del-ferdigheter

- Pytagoras
- Euclidisk geometri
- Radianer vs Grader

Oppgaver

Figure 11.5: The index page of the skill "Geometri". This particular skills show-cases the how this kind of object can be used to contain several sub-skills in order to group together relevant tasks and topic areas.


Multiplikasjon Math

5 er feil :(

Hva er 3 * 3 * 2

Answer

3 * 3 = 9

Figure 11.6: Task page. Shows current task, an overview of the other tasks in that skill. Feedback is instantly shown at the top, and towards the bottom there is a button to display hints.

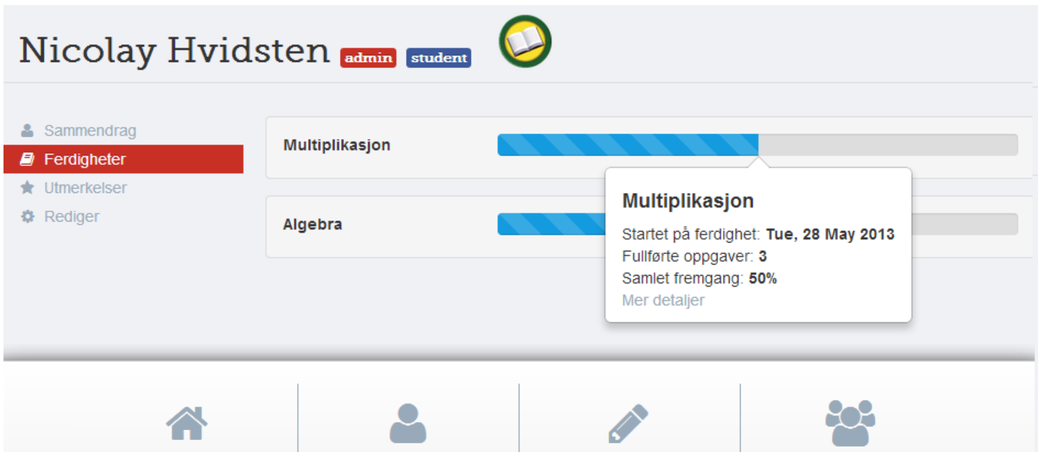


Figure 11.7: Student profile page. Here, the pupil or her teacher can get a complete overview of everything the student has achieved. Clicking the progress-bar of a skill will display detailed information. All information is fetched from the server through AJAX calls to create an interactive, responsive interface.

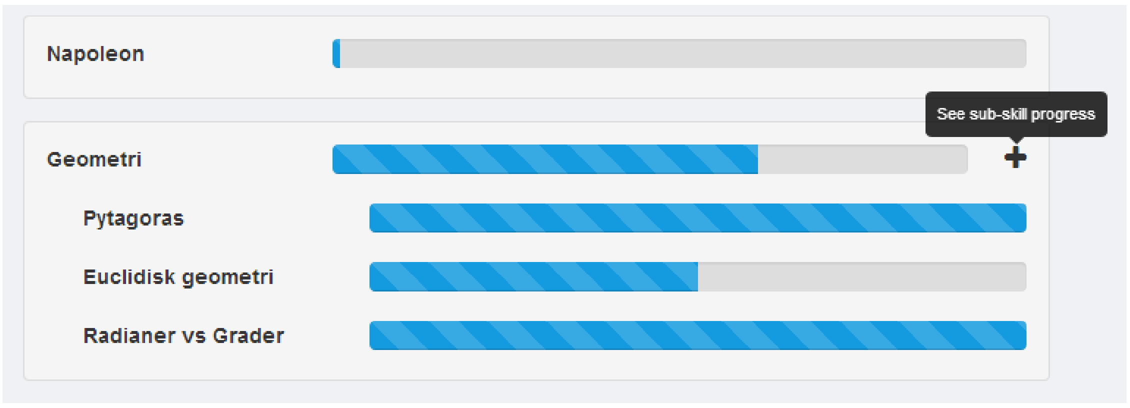
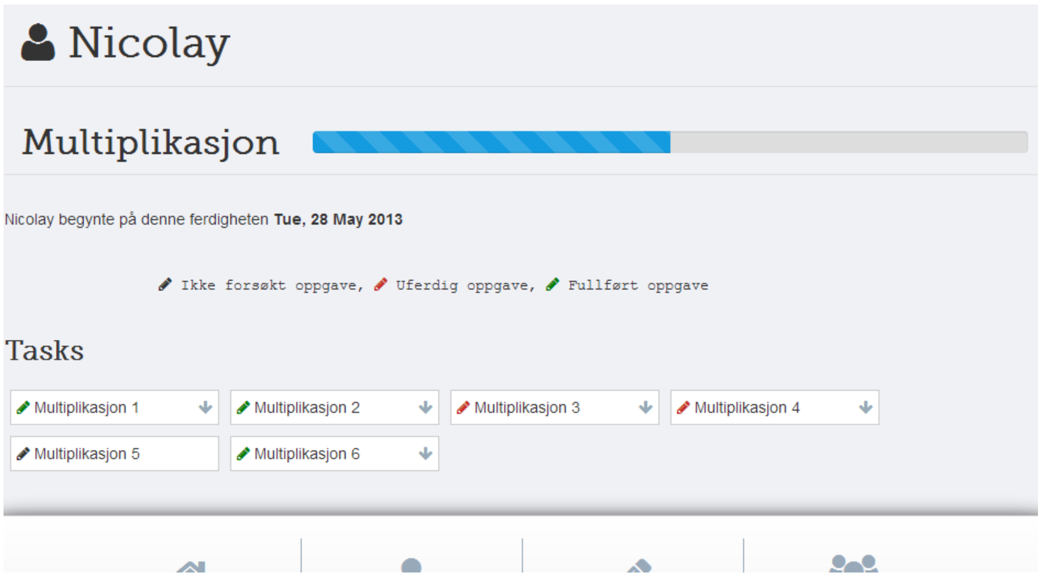


Figure 11.8: The student and the teacher have access to a highly granular set of information regarding school work progress.


Nicolay

Multiplikasjon

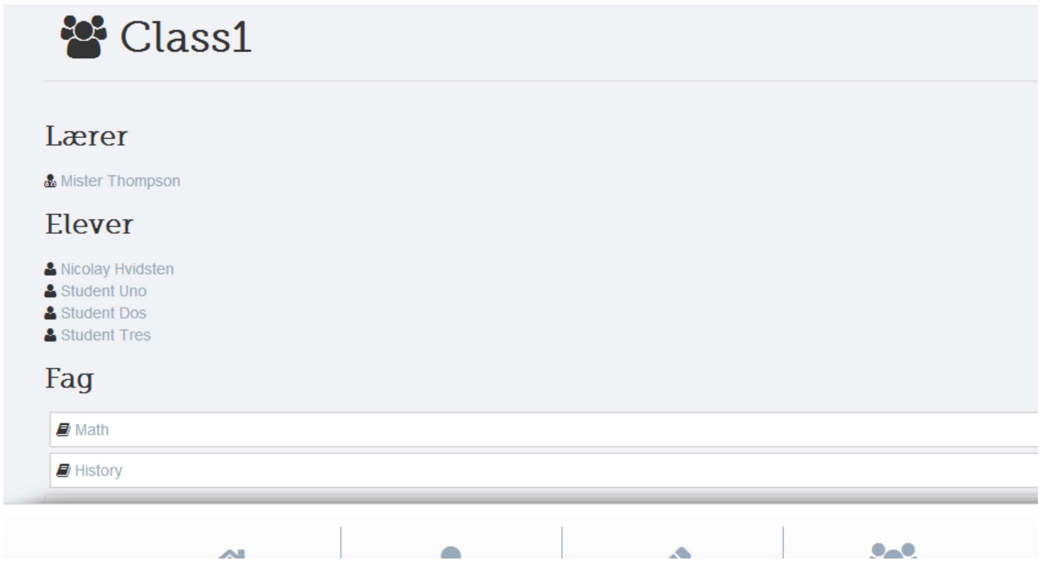
Nicolay begynte på denne ferdigheten **Tue, 28 May 2013**

Ikke forsøkt oppgave, Uferdig oppgave, Fullført oppgave

Tasks

Multiplikasjon 1, Multiplikasjon 2, Multiplikasjon 3, Multiplikasjon 4, Multiplikasjon 5, Multiplikasjon 6

Figure 11.9: Teachers also have access to a detailed view of the task progression of any of their students, for any given skill.


Class1

Lærer

Mister Thompson

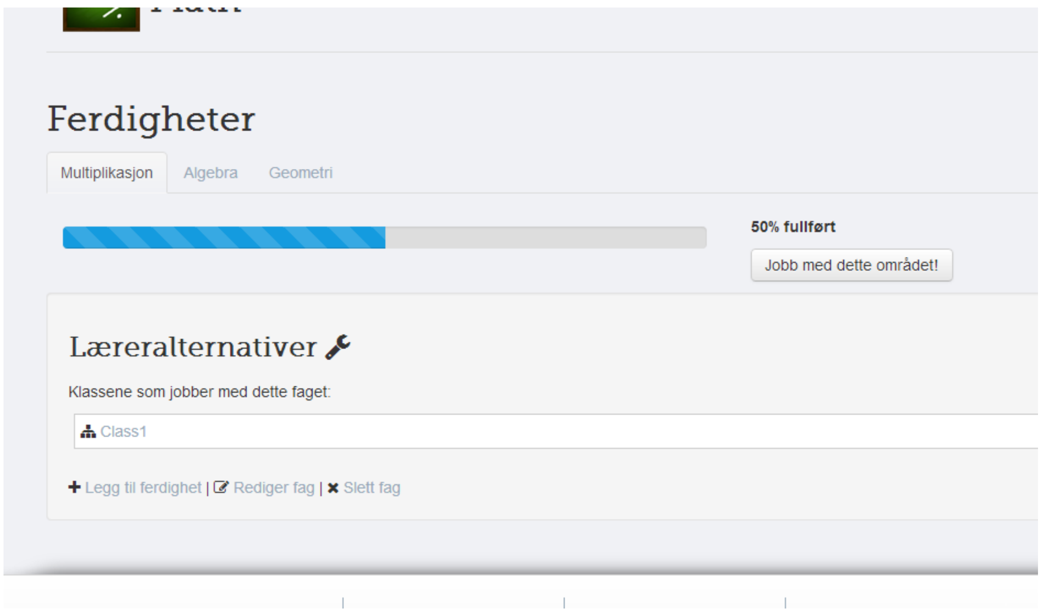
Elever

Nicolay Hvidsten
Student Uno
Student Dos
Student Tres

Fag

Math
History

Figure 11.10: The virtual classroom provides several tools for teachers to view the progress of each individual student, and also see which subjects a particular class is enrolled in.


Ferdigheter

Multiplikasjon Algebra Geometri

50% fullført

Jobb med dette området!

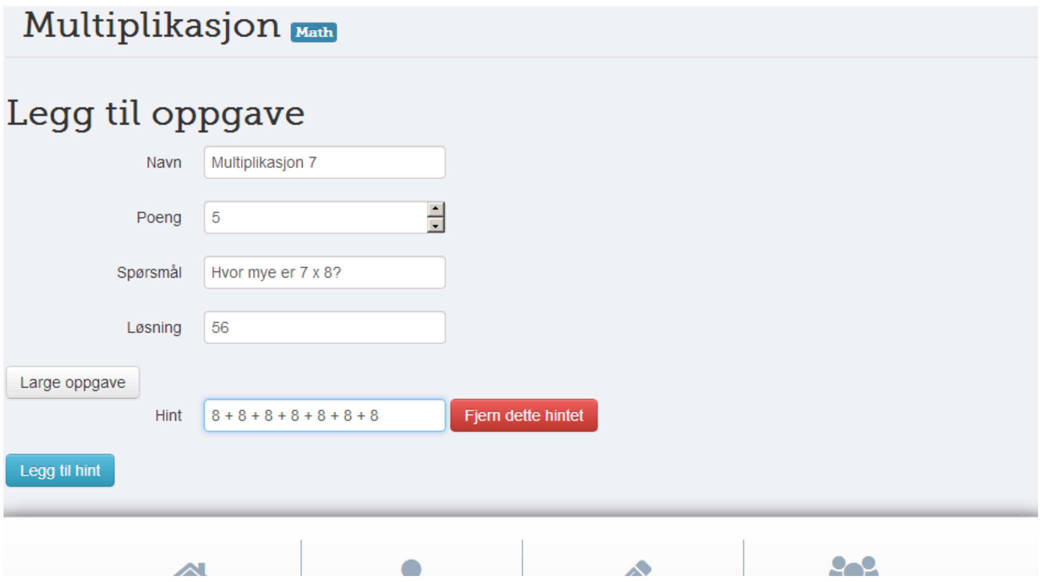
Læreralternativer

Klassene som jobber med dette faget:

Class1

Legg til ferdighet | Rediger fag | Slett fag

Figure 11.11: If the current user is a teacher, additional options to edit various content will be added to the same pages that the students use.


Multiplikasjon Math

Legg til oppgave

Navn	Multiplikasjon 7
Poeng	5
Spørsmål	Hvor mye er 7 x 8?
Løsning	56
Hint	8 + 8 + 8 + 8 + 8 + 8 + 8

Figure 11.12: The interface for adding tasks.

Chapter 12

Implementation Report

The features resulting from the requirement specification had to be subjected to a prioritisation process, and only the ones deemed most essential for the application to have its intended effects on the end user would be selected for implementation.

The importance of generating flow was chiefly considered the most important feature, and the prototype reflects this view. Certain components discussed in Chapter 10 have also been omitted from the implementation, namely *tutorials* and *questions*. The rationale for this omission is that these features were considered of less importance within the setting of generating flow and fiero, which would be the primary goals of the prototype.

The social features based on having students being mentors were also omitted. The main reason for this omission, was the fact that the prototype would not include the tutorial and question objects, which would have been invaluable in this regard. This led to the conclusion that features based on the mentor feature would not be well-designed, and thus should be omitted.

Chapter 4 concluded that the application prototype would be built using *Ruby on Rails* (RoR), which had several consequences. The development process will benefit from aspects inherent to RoR-applications, such as the MVC-architecture, routing mechanisms and testing frameworks provided by the framework. The context of Ruby on Rails is important to keep in mind when viewing the provided code examples.

12.1 Data Model

This section will provide and discuss the associative data model visualising the relationships between the model implementations discussed in Section 12.3.

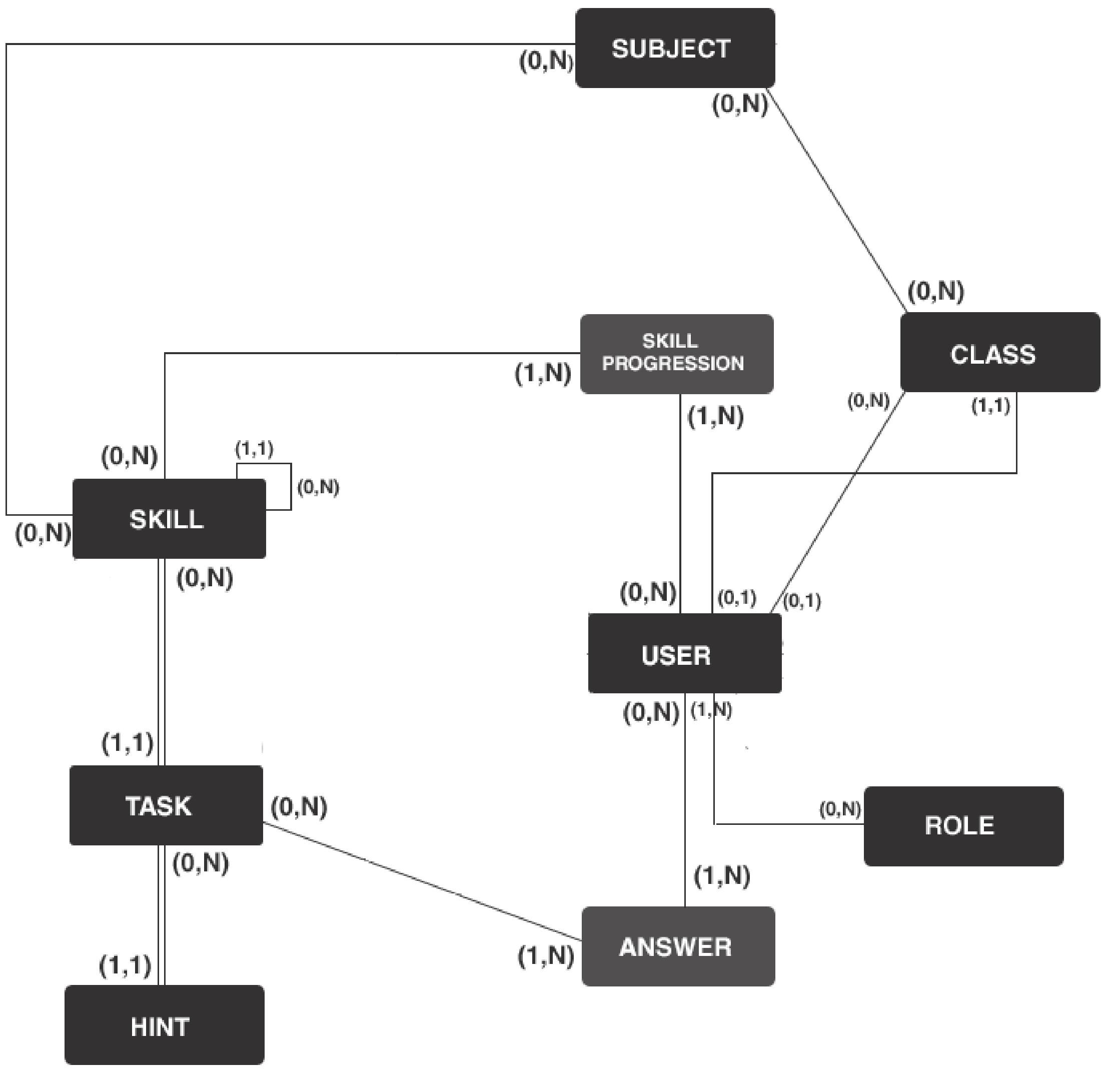


Figure 12.1: Associative data model for the prototype

The discrepancies between the prototype data model and the application data model presented in Section 10.2 due to implementational choices will be discussed.

The first, obvious discrepancy would be the lack of *tutorial* and *question* models, the omission of which was discussed in the introduction of this chapter. The lack of a *reward* model might be more surprising, but this is due to an implementational choice rather than a feature omission, and will be discussed in more detail in Section 12.4.

The feature concerning student mentors was also left out of the prototype implementation, which can be seen by the lack of a self-reference in the user model.

Another relationship lacking from the prototype data model is that between subjects and users, which would represent the *teaching* association (teacher A is *teaching* Subject B). The only reason for this omission is time constraints, as debugging and properly implementing game mechanics were given priority. The fact that the association between teachers and subjects had no actual effect on the properties of the prototype in its current state other in the scope of meta-application knowledge also contributed to the de-prioritisation of this relationship.

12.2 Modules

This section will provide cursory descriptions of the various modules implemented in the application prototype. For detailed discussions regarding these modules, please refer to 12.3.

Module	**Description**
Users and roles	When a new user is created he can be given any number of different roles. New roles and users can be created without having any effect on the rest of the prototype.
Game artefacts	Skills, answers, skill progressions and tasks are contained within their own module, and are linked together. Whenever a teacher creates a skill he is given the option to create sub-skills and/or tasks to complement the newly created game artefact. When a student interacts with a skill or a task this particular module creates the necessary skill progression and answer objects to store interaction information.
Teaching tools	The teacher has access to all granularity levels of information regarding student progress, and this particular module ensures that teachers have access to game artefact objects containing information about student school work.
Classrooms	The social features are contained within the virtual classroom where students can view the profiles of other students to compare rewards and progress. The teacher can also use the classroom to quickly get information about the progress of all students in his class.
Subjects	These sets of objects are in their own module due to the fact that there is no direct interaction between students and subjects. These learning resource objects are utilized to sort and organize game artefacts based on topic, but are entirely de-coupled from the actual game artefacts themselves.
Rewards	Rewards are given to students through completion of game artefacts, and the reward system is de-coupled from the rest of the application: Rewards and be added or modified without it not having any impact on the rest of the system.

Table 12.1: Prototype Modules

12.3 Code

This section will discuss in detail the various components the prototype consists of, and the relationships between them. For the sake of clarity there will also be provided examples of code as illustration when necessary.

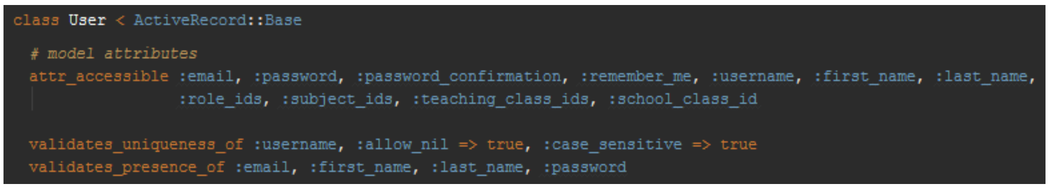
Before discussing prototype functionality at a component-specific level, some features regarding RoR should be pointed out. An important part of any web application in relation to security and consistency is that of *data validation*, and RoR provides an excellent set of functions in this regard.

RoR's Object Relational Mapper (ORM) implements a plethora of validation methods, and all models within the prototype inherits from this class. This entails that all objects within the prototype can use the validation methods implemented in the ORM, ensuring object consistency. The developer can also introduce his own validation methods to perform application-specific custom validation, which we will also provide some examples of.

The ORM also provides an intuitive way of handling associations between objects based on their defined primary and foreign keys in their respective database tables. The most commonly used methods are *has_many* and *belongs_to*, representing **many** and **one** cardinality relationships, respectively.

The ORM also provides several *callback* methods that can be invoked in different parts of an object's life cycle [94]. For instance, a callback method can be invoked when a new object is saved, deleted, added to an association and so forth. There will be ample examples of the prototype employing all ORM methods in the code samples provided in this section.

Users

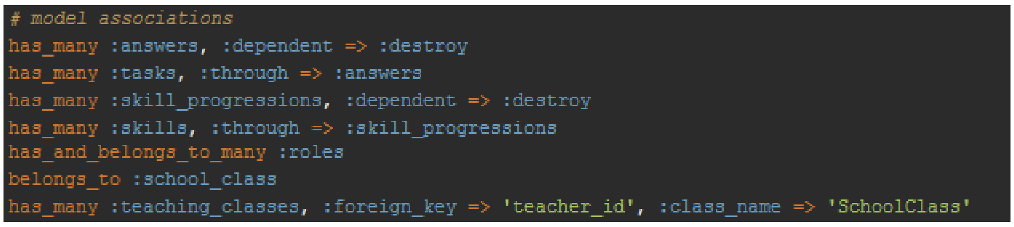


```ruby
class User < ActiveRecord::Base

  # model attributes
  attr_accessible :email, :password, :password_confirmation, :remember_me, :username, :first_name, :last_name,
                  :role_ids, :subject_ids, :teaching_class_ids, :school_class_id

  validates_uniqueness_of :username, :allow_nil => true, :case_sensitive => true
  validates_presence_of :email, :first_name, :last_name, :password
```

Figure 12.2: User model attributes

The attributes of the user model contains relevant user information (first name, last name, email address and so forth) and also some application-specific attributes (password, remember_me, username etc.). The latter are used by the prototype to perform certain tasks, such as allowing users to be remembered by the application, creating custom usernames and so forth.

The user model validates that a specified username is unique within the scope of the prototype, and is given two parameters to indicate that a username can be *nil* and that the uniqueness is based on case sensitivity. It also ensures that none of the attributes in the parameters are nil.



```
# model associations
has_many :answers, :dependent => :destroy
has_many :tasks, :through => :answers
has_many :skill_progressions, :dependent => :destroy
has_many :skills, :through => :skill_progressions
has_and_belongs_to_many :roles
belongs_to :school_class
has_many :teaching_classes, :foreign_key => 'teacher_id', :class_name => 'SchoolClass'
```

Figure 12.3: User model associations

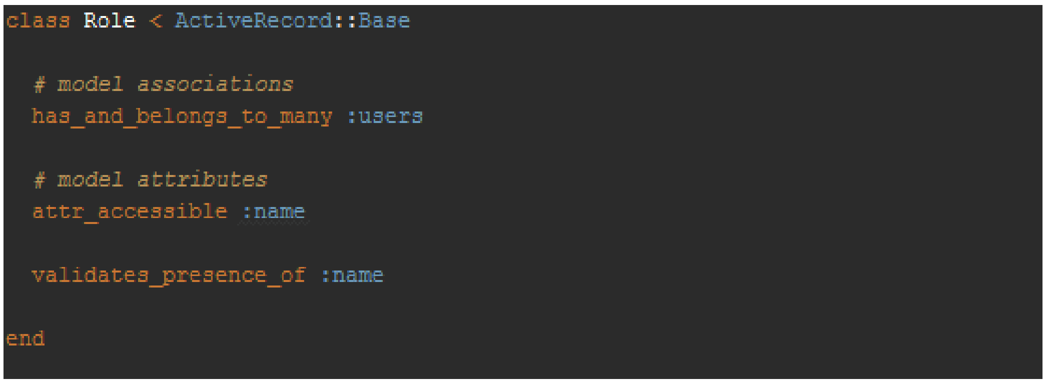
The associations of the user model can be recognised from the data model in Figure 10.8. The methods provided by the RoR ORM facilitate the manipulation of associations between object models, as can be seen in Figure 12.3. The models directly related to the user model through these associations are: *answer*, *skill_progression*, *role* and *school_class*.

As can be seen from the association with the *answer* model, an extra parameter is passed to the ORM method as a dependency which ensures that when the user is deleted, the associated answers are deleted as well. The connection between users and tasks might seem puzzling at first, but this is the way RoR handles *many-to-many* relationships between two models when additional data is stored [94].

The relation between users and skills is of the exact same sort as that between tasks and users, for the same reasons as indicated above, and data relevant to this relation is stored in the *skill_progression* object.

The most basic way of indicating a *many-to-many* relationships in RoR is by using the *has_and_belongs_to_many* ORM method [94]. This simply indicates that there is no additional information to store regarding this relation, and as such there is no need for an additional object to store relation information.

Roles



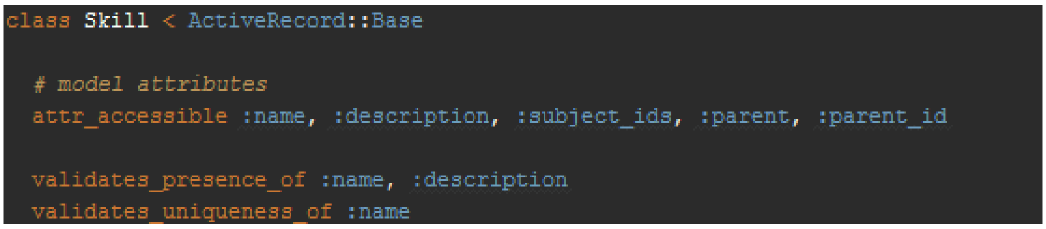
```
class Role < ActiveRecord::Base

  # model associations
  has_and_belongs_to_many :users

  # model attributes
  attr_accessible :name

  validates_presence_of :name

end
```

Figure 12.4: Role model class

The implementation of the *role* model only contains one field: *name*, which specifies the role. The prototype only employs three different role names: *student*, *teacher* and *administrator*.

The only associations necessary for roles is the users associated with the roles. The *has_and_belongs_to_many* ORM method is used.

Skills



```
class Skill < ActiveRecord::Base

  # model attributes
  attr_accessible :name, :description, :subject_ids, :parent, :parent_id

  validates_presence_of :name, :description
  validates_uniqueness_of :name
```

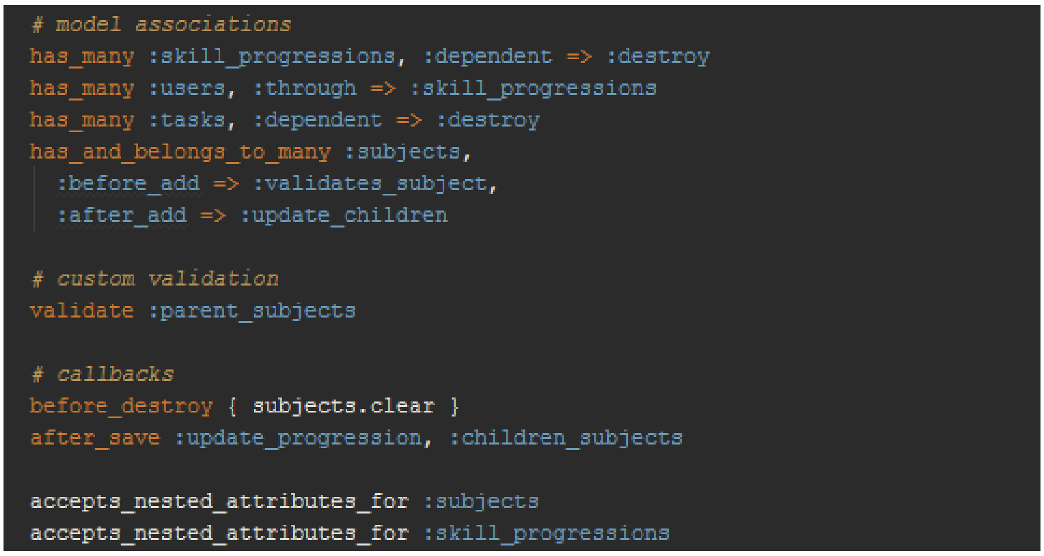
Figure 12.5: Skill model attributes

The *skill* model contains two attributes submitted by the creators of the object: *name* and *description*. The ORM's validation methods are used to ensure that these attributes are present and that the provided name is unique. The (*subject_ids* attribute is used by RoR to allow skills to be added to subjects through the subject interface. The final two attributes, *parent* and *parent_id*), are used by a particular RoR gem called *ancestry*[1].

[1]https://github.com/stefankroes/ancestry

Recall from the requirement specification in Chapter 10 that skills should implement a hierarchical structure which allowed for the organisation of game artefacts per teacher preference. In order to fulfil this particular requirement the prototype implements a *tree structure*, allowing for the creation of skill tress - all using the ancestry gem.

The *parent* attribute references the parent node of a particular skill, while the *parent_id* references the id of said parent node. ancestry allows RoR ActiveRecord models to be stored as a tree structure in a PostGreSQL database, using these additional attributes. In addition to their function in relation to maintaining an hierarchical structure in the database, both of these attributes are made accessible in order for RoR to properly assign parents through the interface of the children skills.



```
# model associations
has_many :skill_progressions, :dependent => :destroy
has_many :users, :through => :skill_progressions
has_many :tasks, :dependent => :destroy
has_and_belongs_to_many :subjects,
  :before_add => :validates_subject,
  :after_add => :update_children

# custom validation
validate :parent_subjects

# callbacks
before_destroy { subjects.clear }
after_save :update_progression, :children_subjects

accepts_nested_attributes_for :subjects
accepts_nested_attributes_for :skill_progressions
```

Figure 12.6: Skill model associations

As was indicated by Figure 10.2, the *skill* object would be associated to users through another object called *skill progressions*. The fact that tasks belong to skills is delineated by the use of the *has_many* ORM method, with the additional parameter specifying that deletes of this model should be cascaded to children.

A specific skill can be associated with any number of subjects through the association specified in this class. Due to the hierarchical tree structure the prototype uses in the implementation of skills, the need for additional *call-*

back methods arises to maintain the associations between skills, sub-skills and subjects.

The skill class uses several callback methods, in different stages of the object's life cycle. In the context of associations, the *before_add* method is called *before* a skill is added to a subject, while the *after_add* is called after. A parameter is passed to each method, *validates_subject* and *update_children* respectively, which are private methods in the skill class.

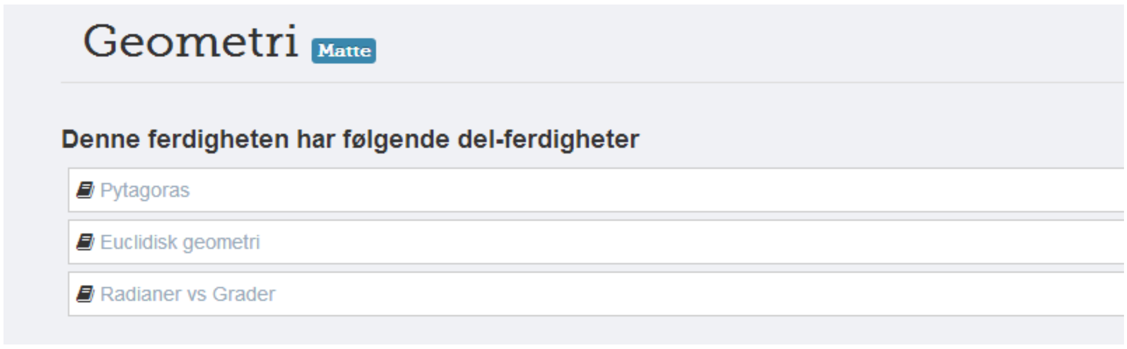
The skill model utilizes the *before_destroy* method, which is invoked before a skill object is deleted. In this case, the skill removes all associated subjects when it is deleted, meaning that the ORM updates the join table of skill and subject and deletes all rows with the current skill's ID. This ensures database consistency. The reason this has to be explicitly stated is that the *has_and_belongs_to_many* ORM method does not support the *dependant* parameter (which is utilized in the task and skill progression relations, respectively).

The last callback method utilized by the skill model is *after_save*, which is invoked whenever an object's state is persisted to the database. Please refer to Table 12.2 for a cursory explanation of the functional behaviour of the method parameters passed to the callback methods.

Method	**Description**
validates_subject	Rolls back the database transaction to store the skill object if the subject passed as a parameter is already associated with this skill. This ensures that the join table between these two models has unique entries.
update_children	Recursive function which ensures that all children skills of the current skill object has the same subjects associated as their parent. Please refer to Figures 12.7 - 12.12 for a visual example of this implementation.
parent_subjects	Custom validation method called before a skill object is persisted. Validates that the skill to be saved does not have any associations to subjects that are not associated with its parent. Should such an association be found the database transaction is rolled back and an error message presented to the user (refer to Figure 12.12 for an example).
update_progression	Called after a skill is saved in order to modify any existing progressions associated with this skill. Consider the impact removing/adding a task and/or sub-skill would have on a the progression of a particular skill to see the necessity of this method call.
children_subjects	Functions much in the same way as *update_children*, but is instead concerned with the *removal* of subject associations. Since there are no parameters specified in the RoR ORM's implementation of the *has_and_belongs_to_many* to deal with the removal of associations, this must be explicitly stated in the model.

Table 12.2: Skill model private method explanations.

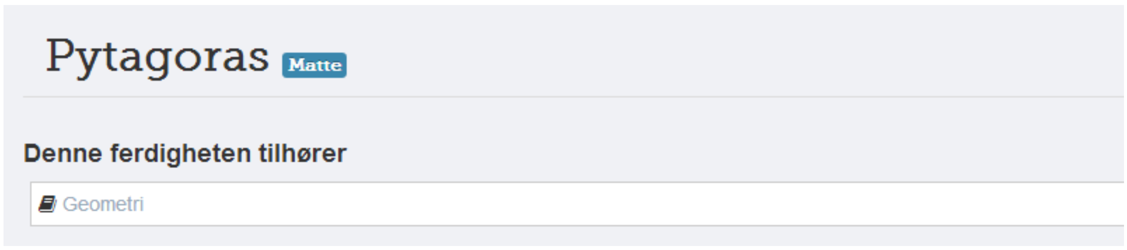
To properly illustrate the relationship between subjects, skills and sub-skills, please refer to the following figures.


Geometri Matte

Denne ferdigheten har følgende del-ferdigheter

- Pytagoras
- Euclidisk geometri
- Radianer vs Grader

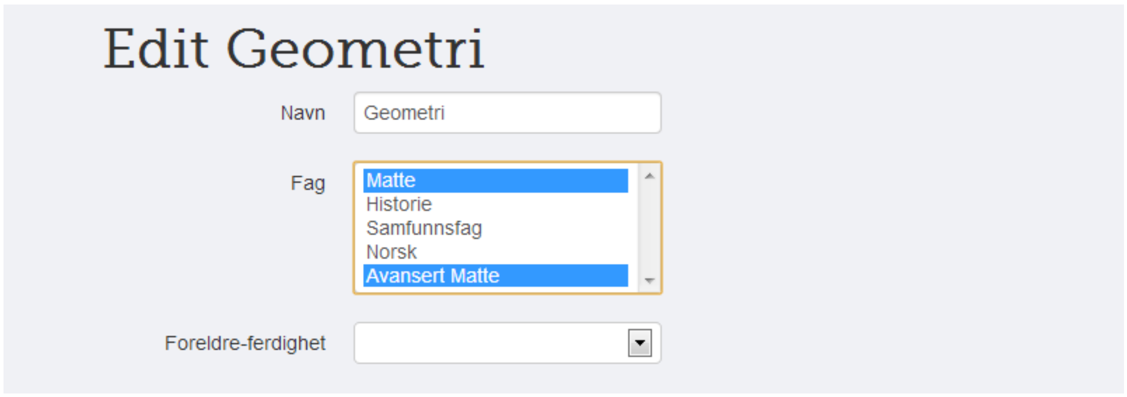
Figure 12.7: The 'Geometri' skill has the subject 'Matte' associated with it, which should also be associated with all its sub-skills.


Pytagoras Matte

Denne ferdigheten tilhører

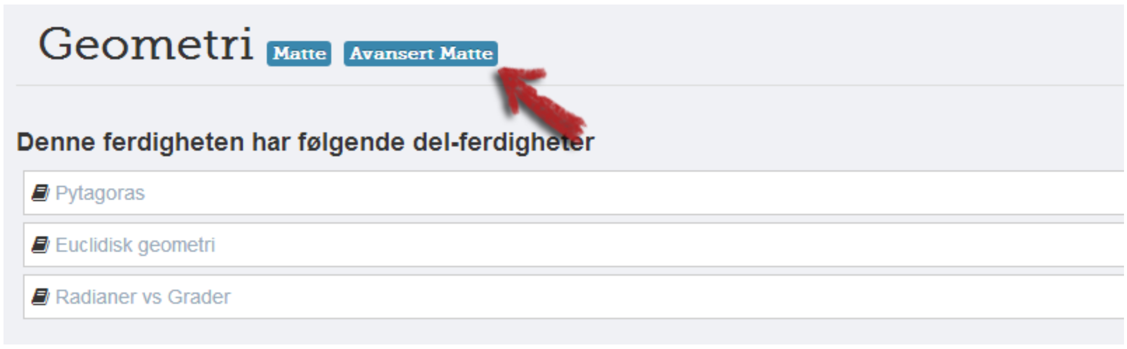
- Geometri

Figure 12.8: The skill 'Pytagoras' is a sub-skill of 'Geometri', and thus also has the subject 'Matte' associated with it.


Edit Geometri

Navn	Geometri
Fag	[✓ Matte], Historie, Samfunnsfag, Norsk, [✓ Avansert Matte]
Foreldre-ferdighet	

Figure 12.9: Editing the skill 'Geometri' to add another subject: 'Avansert Matte'.


Geometri Matte Avansert Matte

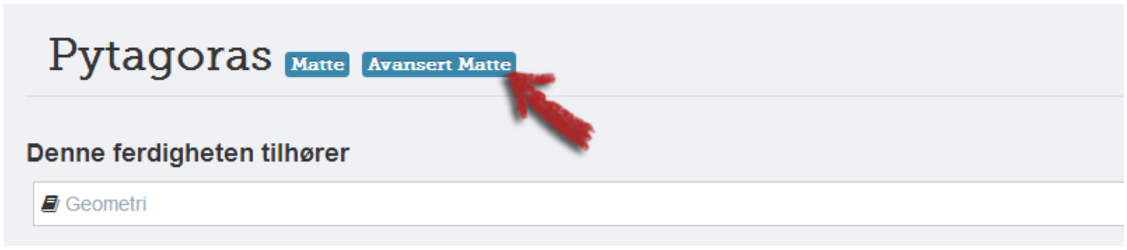
Denne ferdigheten har følgende del-ferdigheter

Pytagoras

Euclidisk geometri

Radianer vs Grader

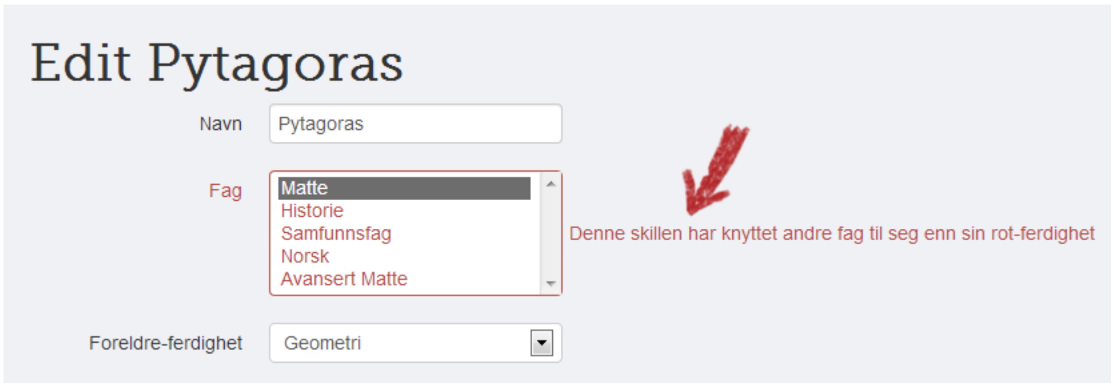
Figure 12.10: The 'Geometri' skill now has two subjects associated with it: 'Matte' and 'Avansert Matte'.


Pytagoras Matte Avansert Matte

Denne ferdigheten tilhører

Geometri

Figure 12.11: The skill 'Pytagoras' has had its subject associations updated through the interface of its parent, with no teacher input needed.


Edit Pytagoras

Navn: Pytagoras

Fag: Matte, Historie, Samfunnsfag, Norsk, Avansert Matte

Denne skillen har knyttet andre fag til seg enn sin rot-ferdighet

Foreldre-ferdighet: Geometri

Figure 12.12: Should a teacher explicitly try to change the subjects of a sub-skill, he must first change the subject associations of the parent.

One final aspect to note regarding the skill model class is the *accepts_nested_attributes_for* ORM method. This method is required in order to have RoR approve updating associations through HTML forms.

Tasks

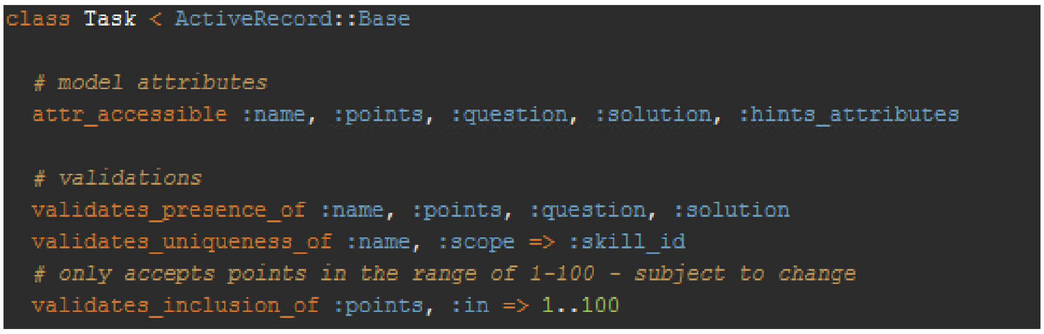

```ruby
class Task < ActiveRecord::Base

  # model attributes
  attr_accessible :name, :points, :question, :solution, :hints_attributes

  # validations
  validates_presence_of :name, :points, :question, :solution
  validates_uniqueness_of :name, :scope => :skill_id
  # only accepts points in the range of 1-100 - subject to change
  validates_inclusion_of :points, :in => 1..100
```

Figure 12.13: Task model attributes

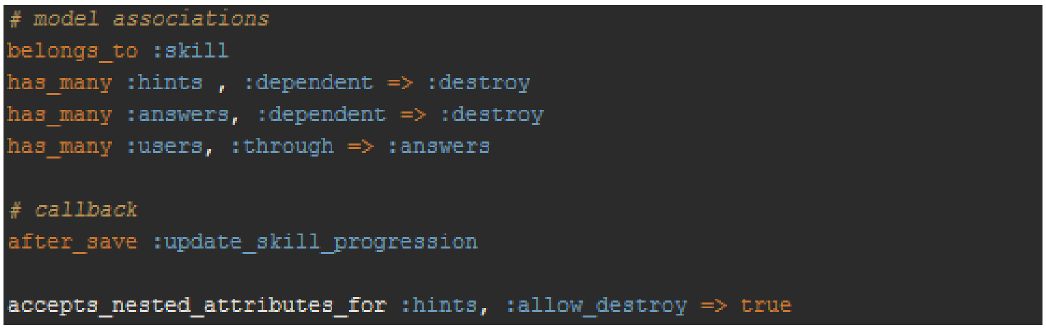
The *task* model contains several user-submitted attributes: *name*, *points*, *question* and *solution*. The *points* attribute contains the point value for completing this task. The *question* and *solution* attributes represent the actual problem presented to the student and the solution to said problem respectively.

The *hints_attributes* attribute is used by a particular *gem* for RoR called *nested form* [2], which allows HTML forms in RoR views based on a certain object to update and/or create objects that implement the *belongs_to* association to the original object. By using the *nested form* gem, adding hints to tasks is simplified. Users can add hints to tasks while creating the task.

The validation for the task model is generally the same as it has been for the other models, with a few alterations for class-specific behaviour. The uniqueness ORM validation method is sent an additional parameter, (*scope*), which indicates that the task's name needs only be unique in the scope of the skill the task belongs to. This means that a teacher could create several tasks called 'Task1' as long as he assigns them all to different skills.

Validates_inclusion_of checks whether the attributes' values are within the given set. In the prototype we defined that points assigned to tasks had to be within the set of 1..100, which would be condition to change upon feedback from user testing (which unfortunately never happened).

[2]https://github.com/ryanb/nested_form_gem



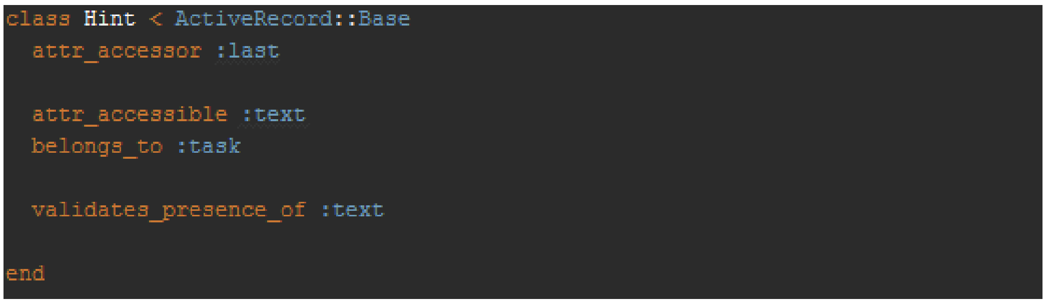
```
# model associations
belongs_to :skill
has_many :hints , :dependent => :destroy
has_many :answers, :dependent => :destroy
has_many :users, :through => :answers

# callback
after_save :update_skill_progression

accepts_nested_attributes_for :hints, :allow_destroy => true
```

Figure 12.14: Task model associations

Since tasks directly belong to a certain skill, they implement the *belongs_to* ORM association method. The task model also has its own relations defining direct ownership the *hint* model. By adding the *destroy* parameter, the ORM ensures that once a task is deleted from the database, all hints referring to this task will also be removed.

In the same way as the *skill* model, the *task* model also employs the *accepts_nested_attributes_for* ORM method in order to allow associations to be updated through the task interface. Because of the way RoR handles resources (objects) these methods are necessary. This ORM method works to complement the *nested form* gem, which allows the manipulation of hint *attributes* (in contrast to the association).

Hints



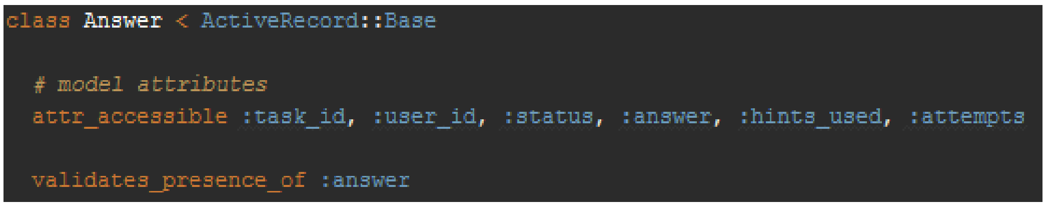
```
class Hint < ActiveRecord::Base
  attr_accessor :last

  attr_accessible :text
  belongs_to :task

  validates_presence_of :text

end
```

Figure 12.15: The hint model

The *hint* model contains very little information - only an attribute storing the textual hint, the presence of which is validated by the RoR ORM. The other end of the relation between tasks and hints can be seen in this class, indicating that hints belong directly to a certain task (in adherence to the application specification).

One thing to note about the hint model class is that it employs a *virtual attribute* called *last*. Virtual attributes are a special kind of attributes that are not persisted, and are only available during runtime.

Answers

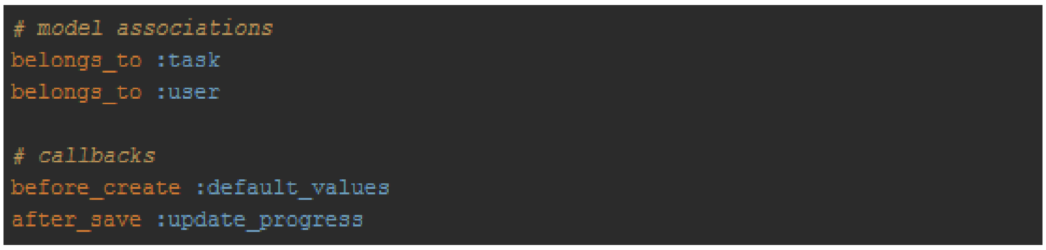


```
class Answer < ActiveRecord::Base

  # model attributes
  attr_accessible :task_id, :user_id, :status, :answer, :hints_used, :attempts

  validates_presence_of :answer
```

Figure 12.16: Answer model attributes

The *answer* model contains several attributes that are utilized by the system (such as *task_id*, *user_id*, *status* and so forth), and only one user-submitted one: *answer*. The attributes containing id's are made available in order to store answer-information through either the *task* or *user* interfaces. The other attributes, (*hints_used*, *attempts* and *status*), are used by the prototype in the context of storing relevant answer information coupled with game mechanics.

The *answer* attribute stores the student's submitted response to the associated task, and the prototype answer handling logic, discussed in Section 12.4, performs the necessary operations in relation to decide whether or not the submitted answer is correct or not.

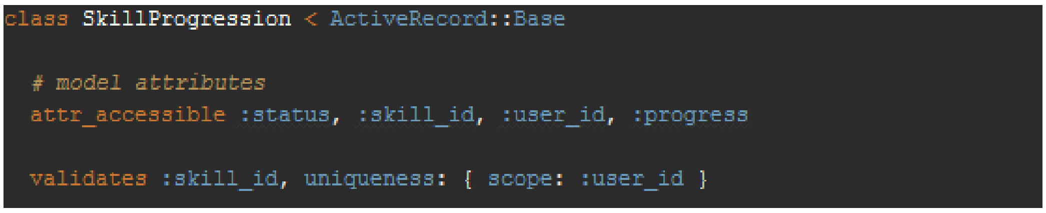


```
# model associations
belongs_to :task
belongs_to :user

# callbacks
before_create :default_values
after_save :update_progress
```

Figure 12.17: Answer model associations

The *belongs_to* ORM associations means that the answer model directly belongs to tasks and users, without an instance of which an answer could not exist. The answer model also employs two callback methods.

Skill Progressions



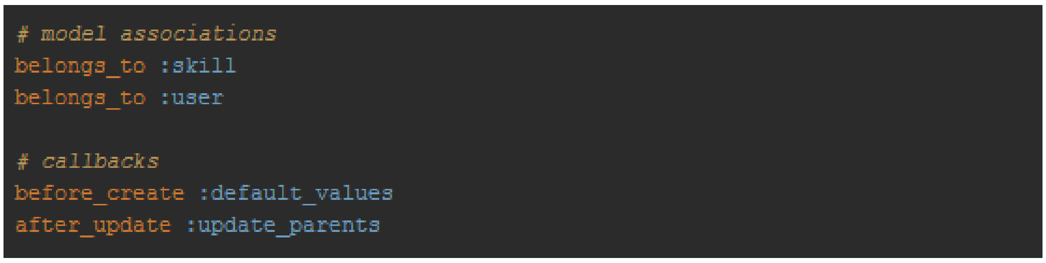
```
class SkillProgression < ActiveRecord::Base

  # model attributes
  attr_accessible :status, :skill_id, :user_id, :progress

  validates :skill_id, uniqueness: { scope: :user_id }
```

Figure 12.18: Skill Progression model attributes

The *skill progression* model stores information regarding the progress of a student within the context of a certain skill. The attributes in question are: *status*, which is utilised by the prototype's game mechanics; *skill_id* and *user_id*, both of which are needed to allow attributes of skill progressions to be stored through the interfaces of skills and users, respectively; *progress*, which is a float number representing the actual progress made by a student in a certain skill (*status* and *progress* are discussed more in detail in Section 12.4).

There should be only *one* skill progression object per user - skill combination, which is achieved by using an ORM validation method enforcing the uniqueness of *skill_id* in the scope of a particular *user_id*.

It should be noted that the chosen solution to this challenge is one of many, and was simply chosen due to ease of testing. As per the RoR documentation of associations, there are a plethora of methods of enforcing certain criteria upon the relations between objects [94], we simply chose what best suited the prototype's needs.

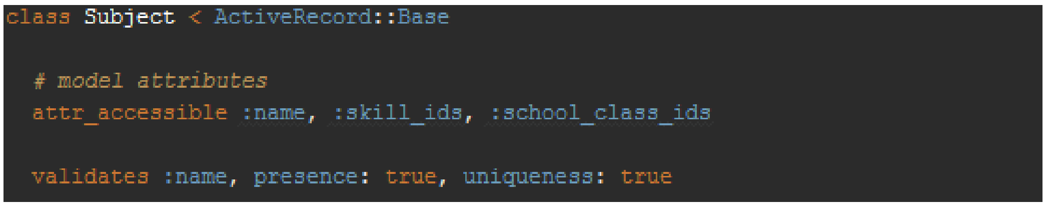


```
# model associations
belongs_to :skill
belongs_to :user

# callbacks
before_create :default_values
after_update :update_parents
```

Figure 12.19: Skill Progression model associations

The skill progression model uses two *belongs_to* ORM association methods to enforce that a skill progression cannot exist without a related skill *and* user.

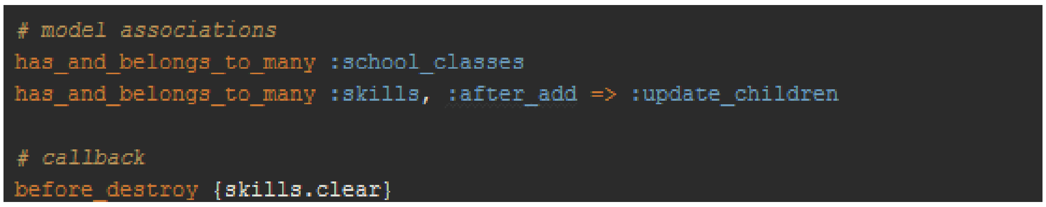
Subjects



```
class Subject < ActiveRecord::Base

  # model attributes
  attr_accessible :name, :skill_ids, :school_class_ids

  validates :name, presence: true, uniqueness: true
```

Figure 12.20: Subject model attributes

The *subject* model also only contains one attribute whose value is decided by user input, the *name* attribute. The other two attributes are used by RoR to allow skills and school class objects be updated through the subject interface, and as such only the user submitted attribute values needs to be validated. In this case the ORM validation methods used enforce the presence and uniqueness of the subject name.

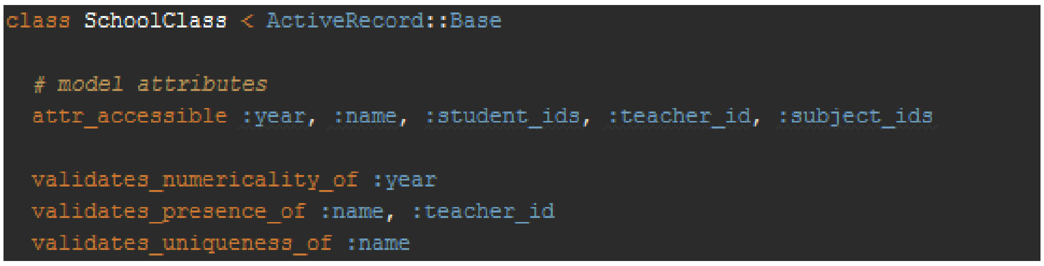


```
# model associations
has_and_belongs_to_many :school_classes
has_and_belongs_to_many :skills, :after_add => :update_children

# callback
before_destroy {skills.clear}
```

Figure 12.21: Subject model associations

The relations that were implemented were associations to school classes and skills. The latter employs a callback ORM method to that the private *update_children* function is called whenever a new skill is added to a subject. The subject model employs the same callback method as the skill model by using the *before_destroy* ORM method to ensure the join table has no pointers to non-existent data.

School Classes

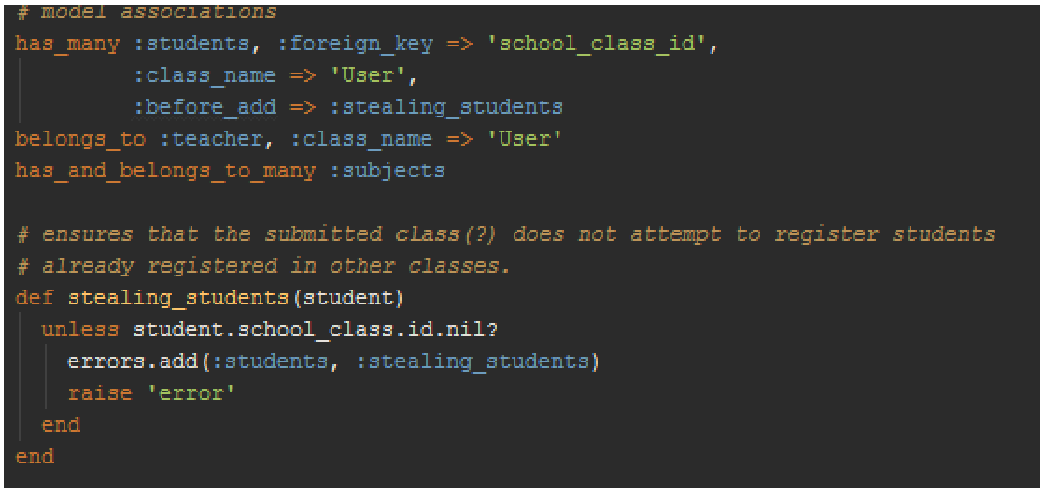


```ruby
class SchoolClass < ActiveRecord::Base

  # model attributes
  attr_accessible :year, :name, :student_ids, :teacher_id, :subject_ids

  validates_numericality_of :year
  validates_presence_of :name, :teacher_id
  validates_uniqueness_of :name
```

Figure 12.22: School class model attributes

The *school class* model contains two attributes that are submitted by end users: *year* (indicating which grade that students enrolled in the particular class are in) and *name* (in order to separate classes). These two attributes are validated through the RoR ORM validation methods, and by validating the presence of *teacher_id* the prototype enforces that a teacher must be assigned to a class before the class can be created.

The remaining attributes (*student_ids*, *teacher_id* and*subject_ids*) are used in the same way as the other *id* attributes in the previously discussed model classes - enabling associated objects to be manipulated through the school class interface.



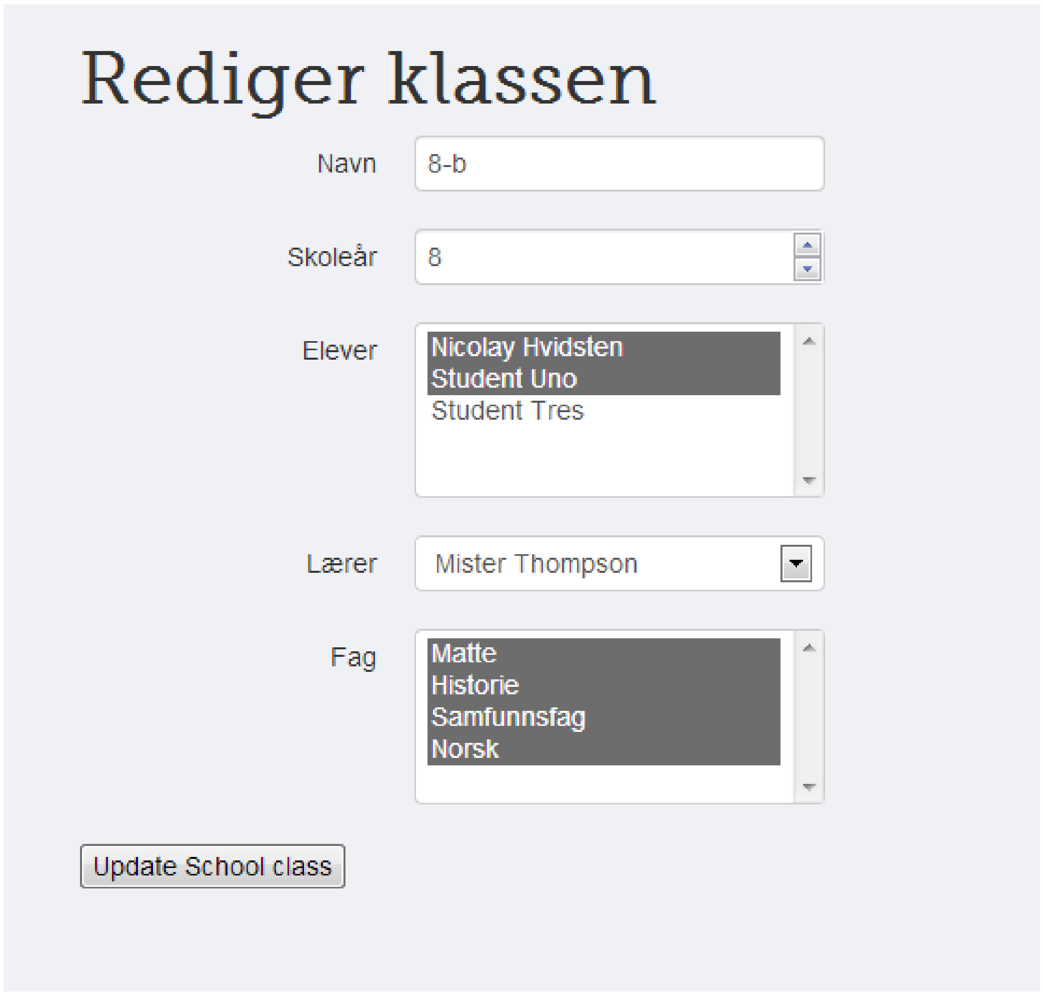
```
# model associations
has_many :students, :foreign_key => 'school_class_id',
         :class_name => 'User',
         :before_add => :stealing_students
belongs_to :teacher, :class_name => 'User'
has_and_belongs_to_many :subjects

# ensures that the submitted class(?) does not attempt to register students
# already registered in other classes.
def stealing_students(student)
  unless student.school_class.id.nil?
    errors.add(:students, :stealing_students)
    raise 'error'
  end
end
```

Figure 12.23: School class model associations

There are *two* relations between the *user* and *school class* models. This was necessary to implement two distinct associations to reflect this due to the different cardinality restrictions of the two relations, which was tricky to accomplish in RoR - mainly because the ORM takes care of so much behind the scenes. The solution to this challenge was to clearly specify the class name of the associated model, and have the model that *owns* the other model specify the foreign key.

Another thing to notice in this regard is the *stealing_students* function passed as a parameter to the *before_add* ORM callback method. This method ensures that when a school class object is stored it does not attempt to register students that are already registered to other school classes. It simply checks whether or not the student in question has a value in the foreign key attribute pertaining to school classes, and if so raises an error.

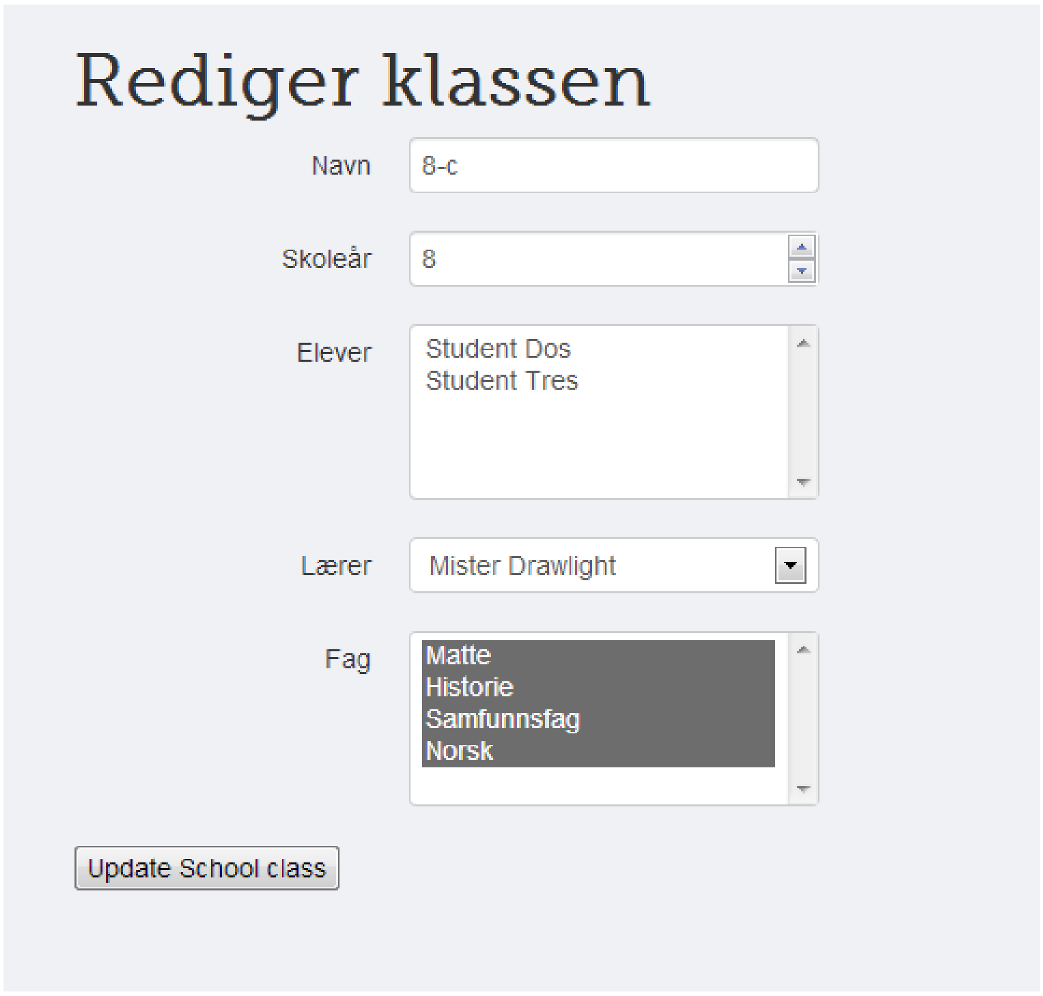
The prototype attempts to avoid this complication through its HTML views as well, as shown in Figures 12.3 and 12.3. However, due to race conditions, should two teachers be editing two different school class objects at the same time they, would see the same list of available students, which is why the additional callback method is needed. When the *stealing_students* function raises an error, the result is shown in Figure 12.3.



Rediger klassen

Navn	8-b
Skoleår	8
Elever	Nicolay Hvidsten Student Uno Student Tres
Lærer	Mister Thompson
Fag	Matte Historie Samfunnsfag Norsk

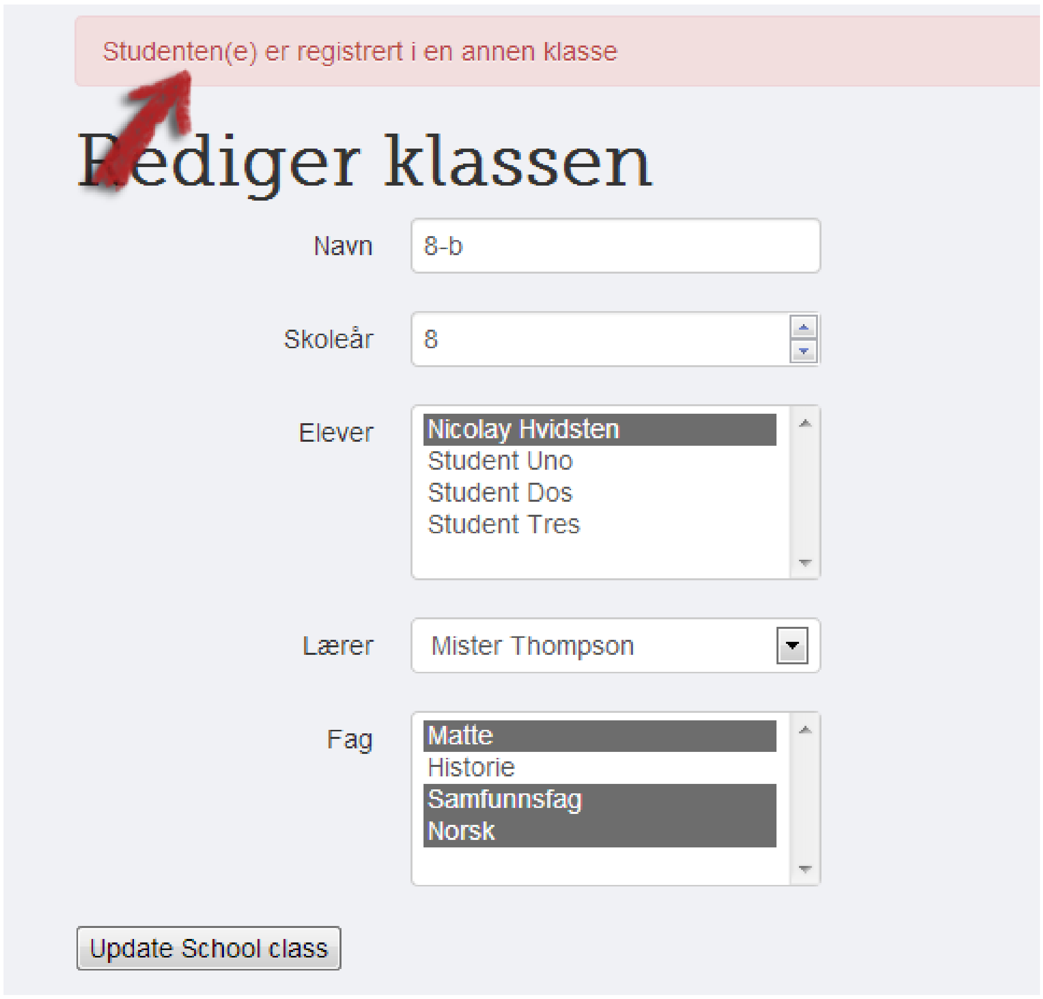
Figure 12.24: The HTML view contains only students registered in the current school class, and those not enrolled in any particular school class yet.



Rediger klassen

Navn	8-c
Skoleår	8
Elever	Student Dos Student Tres
Lærer	Mister Drawlight
Fag	Matte Historie Samfunnsfag Norsk

Figure 12.25: As can be seen in this figure, the students enrolled in school class 8-b cannot be found in the list of available students for school class 8-c.


Studenten(e) er registrert i en annen klasse

Rediger klassen

Navn	8-b
Skoleår	8
Elever	Nicolay Hvidsten Student Uno Student Dos Student Tres
Lærer	Mister Thompson
Fag	Matte Historie Samfunnsfag Norsk

Figure 12.26: The teacher is shown an error message when trying to register students already registered to another class.

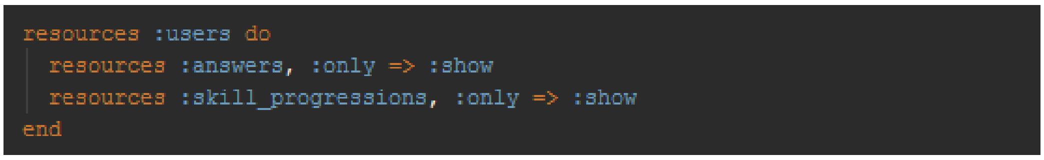
12.4 Addressing Specific Challenges

This particular section will attempt to highlight how the prototype solves specific implementational challenges. Before the discussion is scoped to the context of these challenges, some mention needs to be made regarding the way RoR handles routing and objects in a web application setting.

Resources and routing in Ruby on Rails

RoR employs a virtual *router* to recognise URLs and dispatch them to the actions of specific controllers [94]. This router is specified in a file called *routes.rb*, which defines all *resources* within an application and automatically declares common routes for the resources' controllers, such as URLs for creating, editing or deleting. Each defined resource is given these routes unless otherwise explicitly stated in *routes.rb*, and this innate benefit of using RoR is employed by the prototype to enforce its hierarchical structure in adherence to Figure 12.1.

Many of the features that will be discussed in this section rely upon the functionality provided by the RoR resources routing system, so a cursory understanding is necessary in order to provide the backdrop for said discussion. In order to illustrate the inner workings of the routing system, consider the following example which demonstrates how the prototype enforces the requirement that answers and skill progressions should be accessed through an associated student (i.e. user).



```
resources :users do
  resources :answers, :only => :show
  resources :skill_progressions, :only => :show
end
```

Figure 12.27: User, answer and skill_progression resource routes

Figure 12.4 uses the RoR resource routing system to enforce that in order to access information contained in either answer or skill progression objects, an end user needs to use an URL adhering to the following format:

```
/users/:user_id/answers/:answer_id
/users/:user_id/skill_progressions/:skill_progression_id
```

Thus, the prototype can enforce a rigid hierarchical structure in adherence to the application specification. Also, note the extra parameter provided to the *resource* definitions of answer and skill_progression. This parameter indicates that only the *show* action should be associated with this resource mapping, meaning that any URLs trying to call any other action in the answer or skill_progression controllers would be rejected.

This restriction makes sense when one considers that these objects are instantiated through application data logic as a consequence of user input, rather than by the actual input itself.

It should also be noted that the first line of the code example ensures that users are a top-level resource within the prototype, and provides URL routes for all CRUD-operations regarding users adhering to the following scheme:

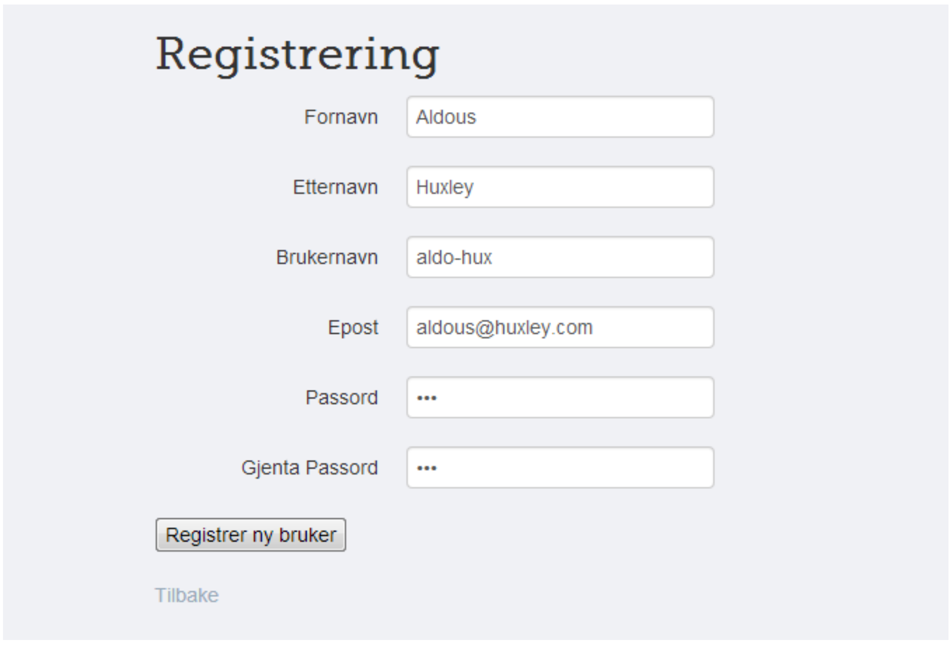
```
/users/new
/users/:user_id
/users/:user_id/edit
/users/:user_id/delete
```

Authorisation and roles

An important requirement outlined in Chapter 9 was that of enforcing access levels to sensitive information through a role system. One of the advantages of using RoR in this regard was that there are currently an abundance of different third-party modules concerning this particular area. Since the prototype needed a form of authorisation to identify users, the decision was made that the role system should work in conjunction with the authorisation system in order to keep the code concerned with data access localised to as few areas as possible.

The most common third-party module (called *gem* in the RoR-world) utilised for employing an application-wide authorisation system in RoR applications at the moment is the *devise* gem [3] which generates all necessary controllers and views, and even supports a very basic plug-and-play role system.

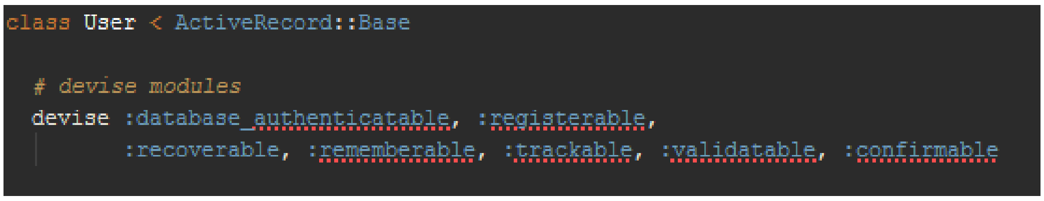
[3]Devise gem: https://github.com/plataformatec/devise


Registrering

Fornavn	Aldous
Etternavn	Huxley
Brukernavn	aldo-hux
Epost	aldous@huxley.com
Passord	•••
Gjenta Passord	•••

Figure 12.28: Example of a standard view generated by devise to register new users.

Devise is built in a modular fashion, allowing the application developers to choose from a set of functionalities that might be in their interest. By including devise, the prototype achieved an authorisation solution which also enforced email confirmation, resetting and hashing passwords, and allowed for users to store tokens in order to be remembered by the system in the future.



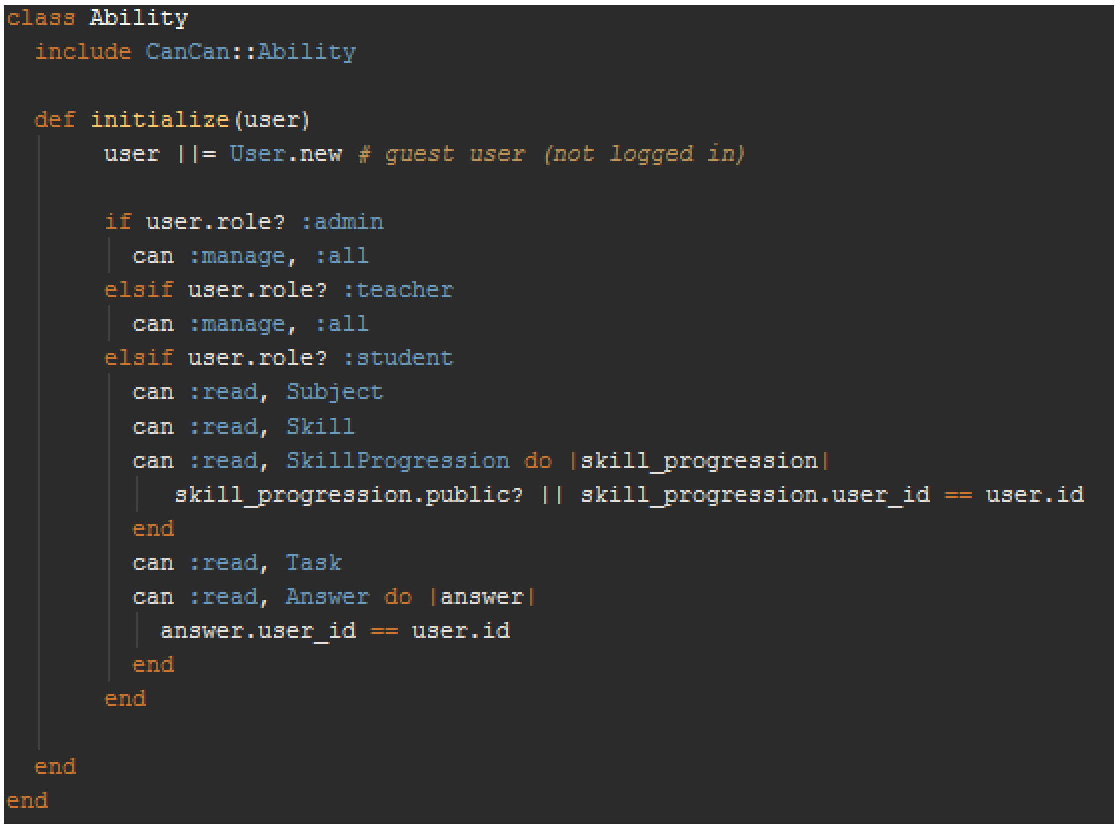
```
class User < ActiveRecord::Base

  # devise modules
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :trackable, :validatable, :confirmable
```

Figure 12.29: The devise modules are defined in the user model

Figure 12.4 shows all the devise modules employed by the prototype, which saved a lot of time concerning the authorisation solution. By taking advantage of the features in the devise gem, the prototype quickly had a working authorisation solution, but a role system was needed to fulfil the requirement specification, and the built-in system in devise turned out to be rather in-flexible.

To solve this particular problem, a gem called *Cancan*[4] was added to the prototype. Cancan is an authorisation library which restricts which resources a given user has access to, and is widely used in conjunction with devise to create a role-based authorisation solution. In order to avoid trying to re-invent the wheel, the choice was made to follow this boiler-plate solution, which proved to be very successful.

Cancan uses a single class to determine which resources a given user has access to called *ability.rb*. This allows for a very modular approach to role-based access levels. Figure 12.4 shows how the Cancan gem was used in the prototype.



```ruby
class Ability
  include CanCan::Ability

  def initialize(user)
      user ||= User.new # guest user (not logged in)

      if user.role? :admin
        can :manage, :all
      elsif user.role? :teacher
        can :manage, :all
      elsif user.role? :student
        can :read, Subject
        can :read, Skill
        can :read, SkillProgression do |skill_progression|
          skill_progression.public? || skill_progression.user_id == user.id
        end
        can :read, Task
        can :read, Answer do |answer|
          answer.user_id == user.id
        end
      end

  end
end
```

Figure 12.30: The ability.rb class

[4]Cancan gem: https://github.com/ryanb/cancan

In order to enforce the important requirement of not allowing students to access sensitive information regarding the progress of other students, an additional code block is necessary in the authorisation definitions for skill progressions and answers.

One of the requirements specified in the context of social features and peer motivation was that of allowing students to make certain parts of their progression public, and display it in their profile. In this regard it is very important that *only* the public information is made available to other students, and an additional parameter consisting of a code block is added to the *can* function for this very purpose. If the added code block evaluates to *false* the current user is denied access, which can be used to fulfil the requirements for information privacy.

The additional code blocks call a method named *public?* on their respective game artefact objects, which indicates whether or not the associated student has flagged this particular object as public. If this method call returns *false*, then a check is made to see if the student attempting to read the object state is the student associated with the object, in which case access is granted. Should both these tests fail, however, then the student is denied access.

It is important to note that while these two objects could potentially contain sensitive information, the rendering of the RoR views can be customised based on the role of the logged in user. This means that even though a student can read the state of, for instance, an answer object that has been made public by another student, the application can still ensure that only a sub-set of the object's attributes are presented in the view. Refer to figures 12.4 and 12.4 for an example of how the rendering of RoR views can be customised based on the logged in user.

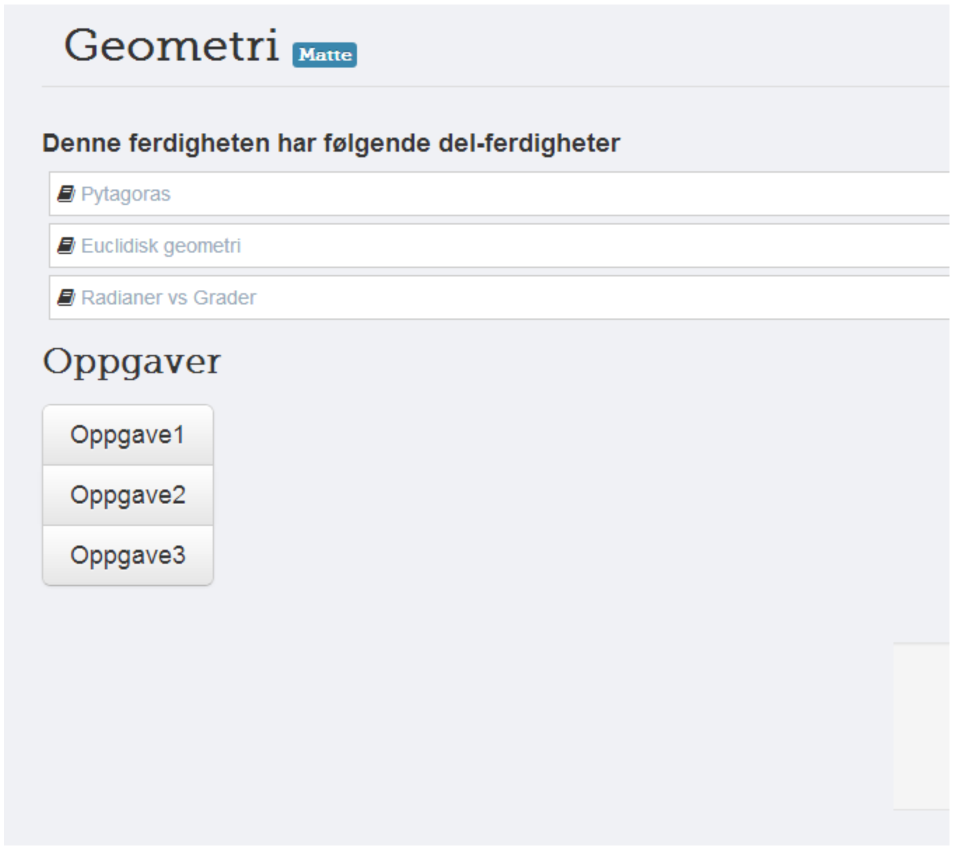
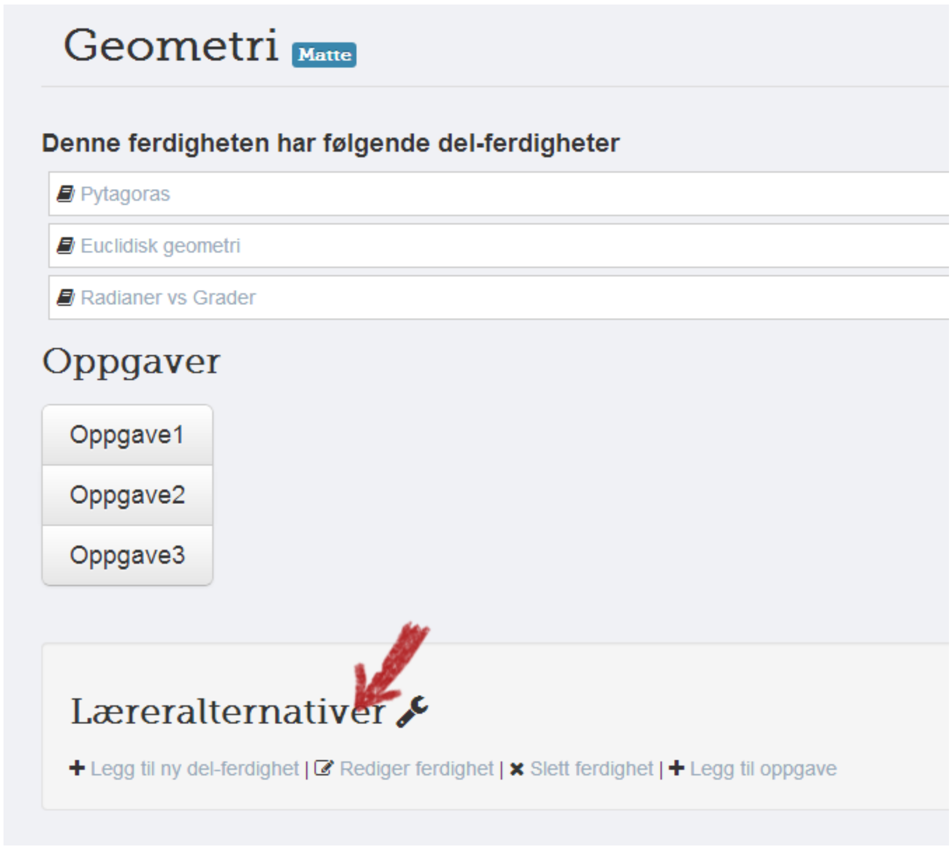


Figure 12.31: Skill view rendered for a student


Geometri Matte

Denne ferdigheten har følgende del-ferdigheter

Pytagoras

Euclidisk geometri

Radianer vs Grader

Oppgaver

Oppgave1

Oppgave2

Oppgave3

Læreralternativer

+ Legg til ny del-ferdighet | Rediger ferdighet | ✖ Slett ferdighet | + Legg til oppgave

Figure 12.32: Skill view rendered for a teacher

The view rendering customisation illustrated in figures 12.4 and 12.4 were achieved by the code in Figure 12.33.

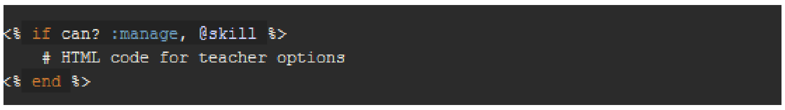

```
<% if can? :manage, @skill %>
   # HTML code for teacher options
<% end %>
```

Figure 12.33: Cancan code for view customisation

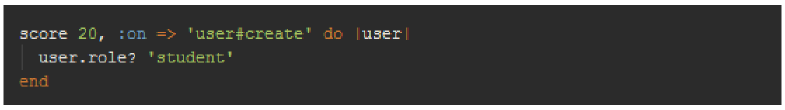
Rewards

The importance of properly implementing gamification features to the prototype led to a vigorous research regarding best-practice approaches and existing gems, which ultimately led to the installation of the *merit* gem [5]. This particular gem implements reputation-based rewards like points, rankings

[5]merit gem: https://github.com/tute/merit

and badges, all of which are examples of tried-and-tested extrinsic rewards.

The merit gem is designed in a highly modular fashion, providing three classes in which the application can specify rules for obtaining points, badges and rankings - called *point_rules.rb*, *badge_rules.rb* and *ranking_rules.rb*, respectively. The rules specified in these classes utilise the RoR resource routing system by applying an application-wide filter which matches RoR controller actions against defined rule. Refer to Figure 12.34 for an illustrative example.

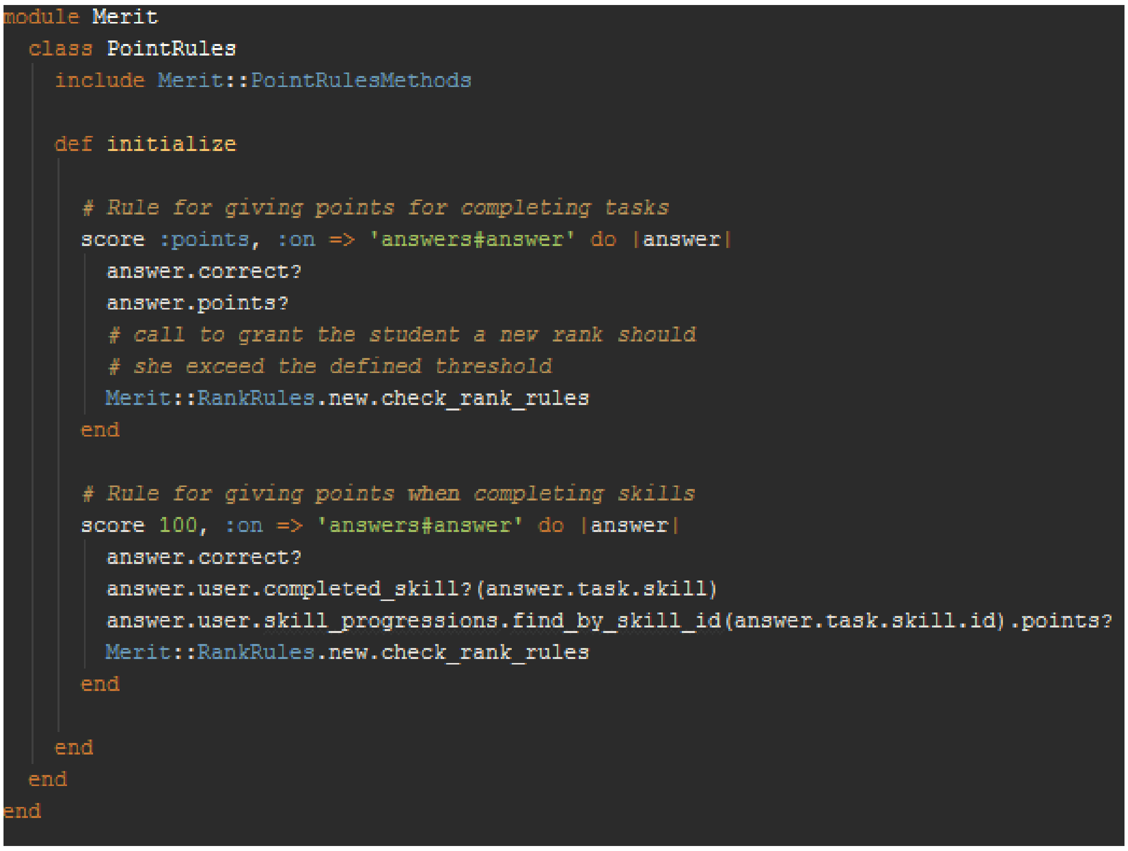


```
score 20, :on => 'user#create' do |user|
  user.role? 'student'
end
```

Figure 12.34: Code example of a merit rule from point_rules.rb evaluated after the *create* action of the UserController has been invoked

The code in Figure 12.34 runs after an administrator or teacher adds a new user to the application, through the application-wide merit filter. After RoR has initiated the proper controller response to the URL for creating new users, the merit filter recognises this controller as being registered to a particular rule (in this case, a *points* rule.) The rule stipulates that the currently logged in user should receive 20 points for adding a new student, and passes an additional code block to the rule generator which evaluates to *true* if the newly created user has the *student* role.

This is naturally only meant as an illustrative example of how the merit gem works, which will function as a backdrop for future examples when more advanced rules are in place. The example should showcase the modularity of the merit gem, allowing a nearly infinite amount of rules to be defined in a class de-coupled from the prototype's inner workings.



```
module Merit
  class PointRules
    include Merit::PointRulesMethods

    def initialize

      # Rule for giving points for completing tasks
      score :points, :on => 'answers#answer' do |answer|
        answer.correct?
        answer.points?
        # call to grant the student a new rank should
        # she exceed the defined threshold
        Merit::RankRules.new.check_rank_rules
      end

      # Rule for giving points when completing skills
      score 100, :on => 'answers#answer' do |answer|
        answer.correct?
        answer.user.completed_skill?(answer.task.skill)
        answer.user.skill_progressions.find_by_skill_id(answer.task.skill.id).points?
        Merit::RankRules.new.check_rank_rules
      end

    end
  end
end
```

Figure 12.35: The point_rules.rb class

According to the requirement specification a student should receive a certain amount of points when completing game objective, in relation to the difficulty of said objective. While a well-designed points system with associated levels and so forth would arguably take some trial and error to properly implement, the purpose of including these features to the prototype was that of a proof of concept.

The code supplied in Figure 12.35 shows two rules associated with granting points to students based on the same controller action: the *answer* action in the AnswerController class. Both rules grant the students points for completing a task or skill.

As can be seen from Figure 12.35, the rule definitions use two different approaches when deciding the number of points to award the student - the former uses an attribute while the latter uses a fixed number. The reason for this disparity is simply that dynamic point allocation based on skill difficulty was de-prioritised in the prototype.

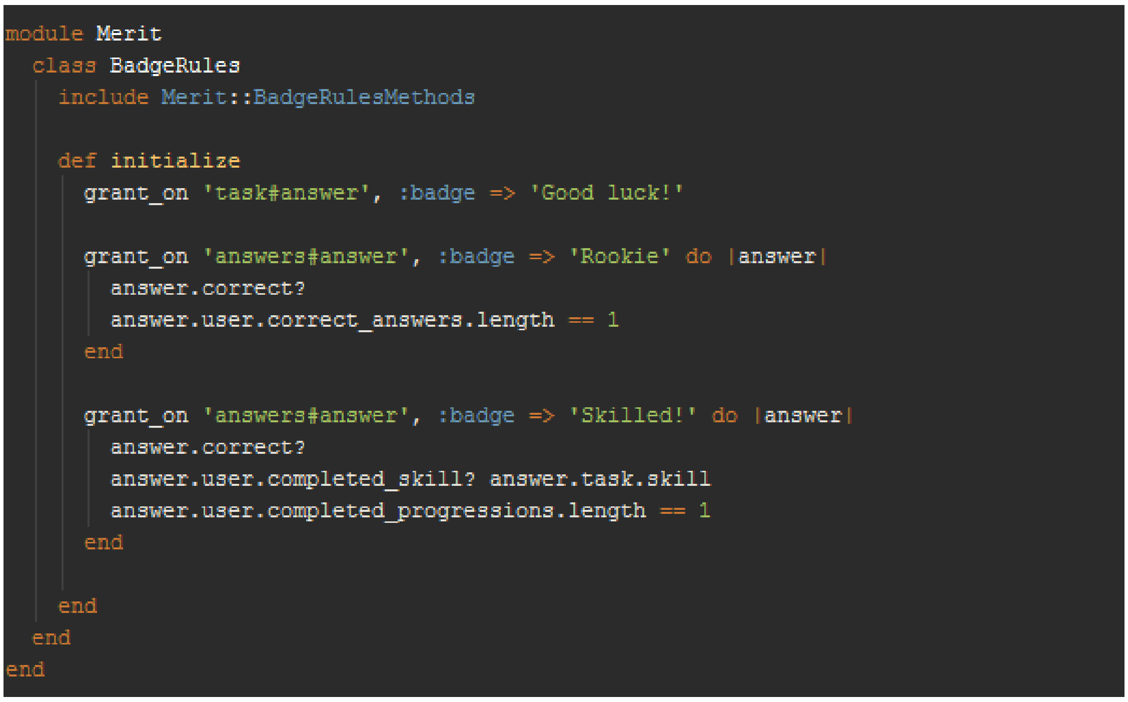
Both rule definitions provide an additional code block which must evaluate to *true* in order for the student to be granted the associated points.

Method	**Description**
answer.correct?	Returns *true* if the current answer object passes a check against the solution of its associated task, *false* otherwise.
answer.points?	Returns *true* if the student has not yet received points for completing the task associated with the current answer, *false* otherwise.
user.completed_skill?	Returns *true* if the student has completed the skill passed as a parameter to the call, *false* otherwise.
skill_progression.points?	Returns *true* if the student has not yet received points for completing the current skill_progression, *false* otherwise.

Table 12.3: Merit method explanations.

The code block passed to the first rule evaluates to *true* when the both *answer.correct?* and *answer.points?* return *true*, which, per Table 12.3, is the case when the current student has submitted the correct answer and has yet to receive points for completing the task associated with this answer.

Should either of these methods return *false* then the data flow in the merit gem terminates, effectively returning control flow to the prototype. When both methods return *true*, an additional method internal to the merit gem, called *check_rank_rules*, is invoked. This final method call does not evaluate to *true* or *false*, but instead checks the defined rank rules for the prototype. If the student exceeds a threshold for gaining a new level by being awarded the points associated with the current answer, then she is granted a new level as well in real-time.

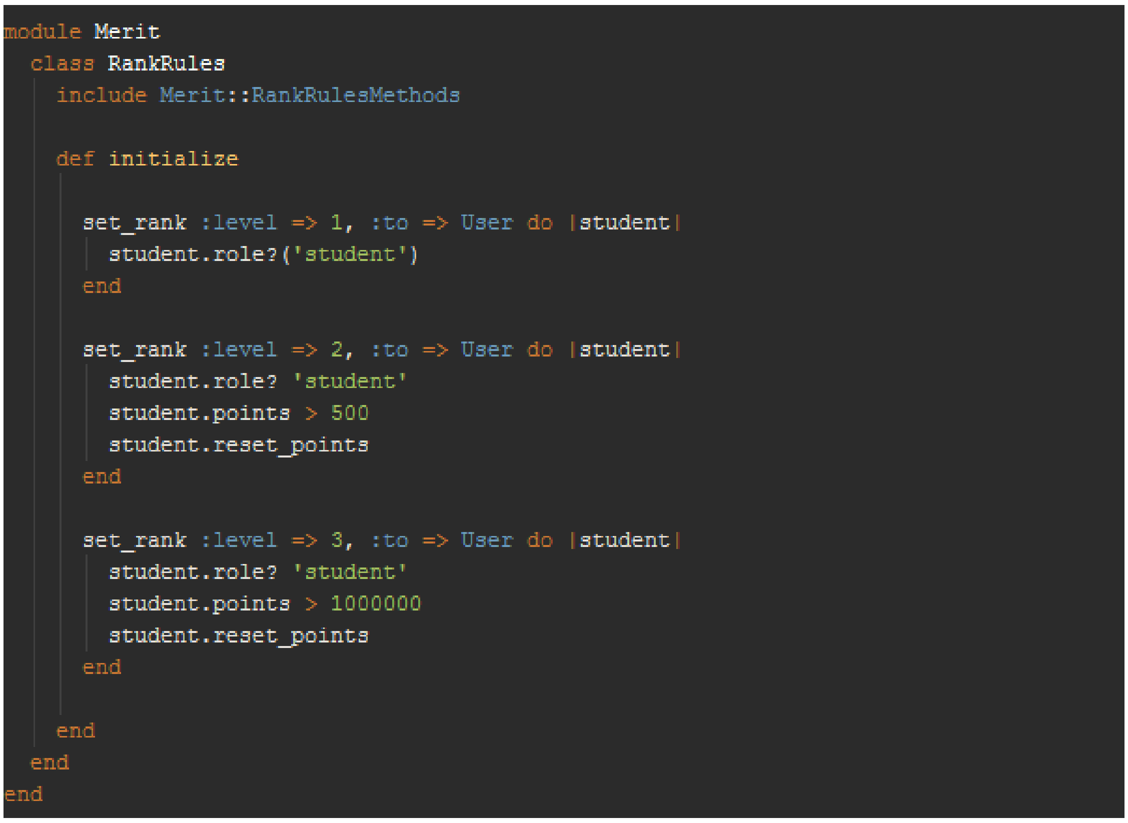


```ruby
module Merit
  class BadgeRules
    include Merit::BadgeRulesMethods

    def initialize
      grant_on 'task#answer', :badge => 'Good luck!'

      grant_on 'answers#answer', :badge => 'Rookie' do |answer|
        answer.correct?
        answer.user.correct_answers.length == 1
      end

      grant_on 'answers#answer', :badge => 'Skilled!' do |answer|
        answer.correct?
        answer.user.completed_skill? answer.task.skill
        answer.user.completed_progressions.length == 1
      end

    end
  end
end
```

Figure 12.36: The badge_rules.rb class

merit badge rules function in much the same way as the rules for points, which are checked through the application wide-merit filter, matching RoR actions against strings in the rule definitions. Badges are defined in an external file, containing all attributes (such as badge name, id, image and so forth). This file is loaded into the RoR application when it initialises, which keeps badges on an abstraction level higher than the database.

The particular badge to grant a student is defined by the `:badge` parameter, which uses the name attribute to search through the external file storing badge information. Just as in the case of point rules, an additional code block can be passed along as an optional parameter, specifying the criteria which must be met in order for the badge to be granted.



```
module Merit
  class RankRules
    include Merit::RankRulesMethods

    def initialize

      set_rank :level => 1, :to => User do |student|
        student.role?('student')
      end

      set_rank :level => 2, :to => User do |student|
        student.role? 'student'
        student.points > 500
        student.reset_points
      end

      set_rank :level => 3, :to => User do |student|
        student.role? 'student'
        student.points > 1000000
        student.reset_points
      end

    end
  end
end
```

Figure 12.37: The ranking_rules.rb class

merit ranking rules operate in a slightly different way than the badge and point rules in the sense that they do not employ the application-wide merit filter to test rules in response to particular RoR controller actions. Instead the ranking rules are never checked unless explicitly stated in the prototype code or formulated as a cron job.

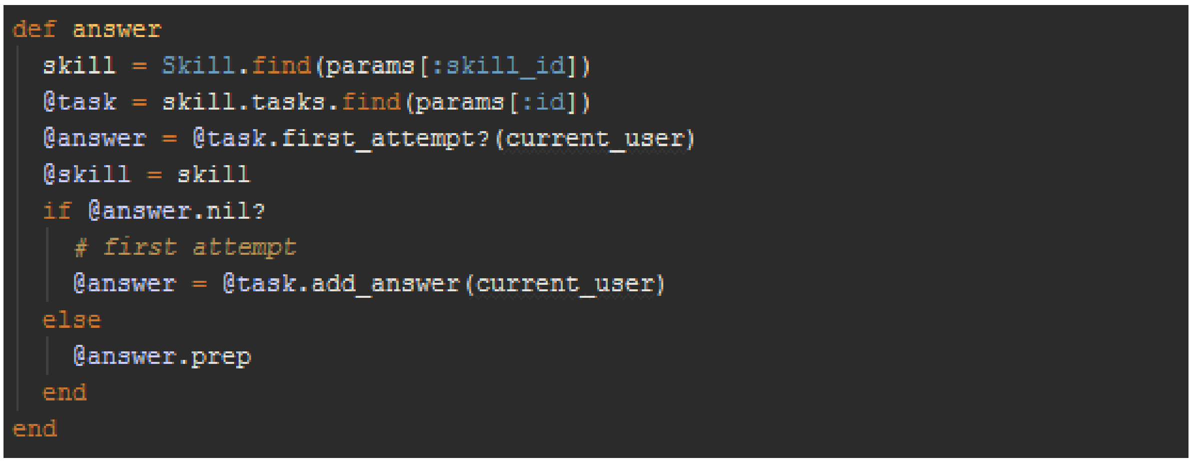
Recall from the discussion of the point rules that the code block passed to the score function invoked a method called `check_rank_rules` which effectively runs all rules in the ranking_rules class. In this case each rule evaluates a certain threshold for the *points* attribute in the student model (recall from the discussion of the user model that there was no defined *points* attribute - this is added by the merit gem), and awards a student a new rank based on the defined threshold.

Users and game artefacts

It was clearly stated in the application specification in Chapter 10 that all relevant information regarding the interaction between students and game artefacts should be stored and made available to teachers. While the interface options to present the various information has been demonstrated in this chapter, the actual prototype component of storing and updating the data set has yet to be addressed in detail.

Recall from Figure 12.1 that students are associated with skills and tasks through two specific objects: skill progressions and answers, respectively. While the connection between students and artefacts could be accomplished through a simple join table, it is in adherence to RoR best practices to define explicit object representations of these join tables if additional attributes are stored pertaining to the join [94].

As mentioned in Section 12.3 the answer model contains all relevant information regarding a student's interaction with a particular task (such as *attempts*, *hints_used* and so forth), which is made available to the student and her teacher through the student's profile. In order to first establish a connection between a student and a particular task, the prototype creates a new *answer* object whenever a student starts working on a new task.



```
def answer
  skill = Skill.find(params[:skill_id])
  @task = skill.tasks.find(params[:id])
  @answer = @task.first_attempt?(current_user)
  @skill = skill
  if @answer.nil?
    # first attempt
    @answer = @task.add_answer(current_user)
  else
    @answer.prep
  end
end
```

Figure 12.38: The *answer* action of the TasksController class.

The code in Figure 12.38 is executed when an URL adhering to the following scheme is received by the Prototype's router:

```
\skills\skill_id\tasks\task_id\answer
```

Note that the controller checks whether or not it is the first time the current user is attempts the task through the `first_attempt?` method. If the current user has previously interacted with this task (regardless of whether or not she has completed it), an answer row referencing this particular task and user should already be in the database. An object representing this row is returned from the method call if it exists, otherwise *nil* is returned.

If *nil* is returned from the initial check, then the prototype knows that it is the first time the current user is interacting with this particular task. Thus, a new row must be added to the answers table in the database, referencing this task and the current user, which is achieved through the `add_answer` method call.

In the case that the initial check returns an actual answer object, the controller calls the object's `prep` method, which removes the previously stored *answer* attribute (the textual answer to be evaluated against the task's solution) in order to provide the user with a clean answer object. Note that this attribute removal is not persisted in any way, and is only employed to avoid showing the user her previously submitted answer to the task she is now trying to complete (regardless of whether her previously submitted answer was correct or not).

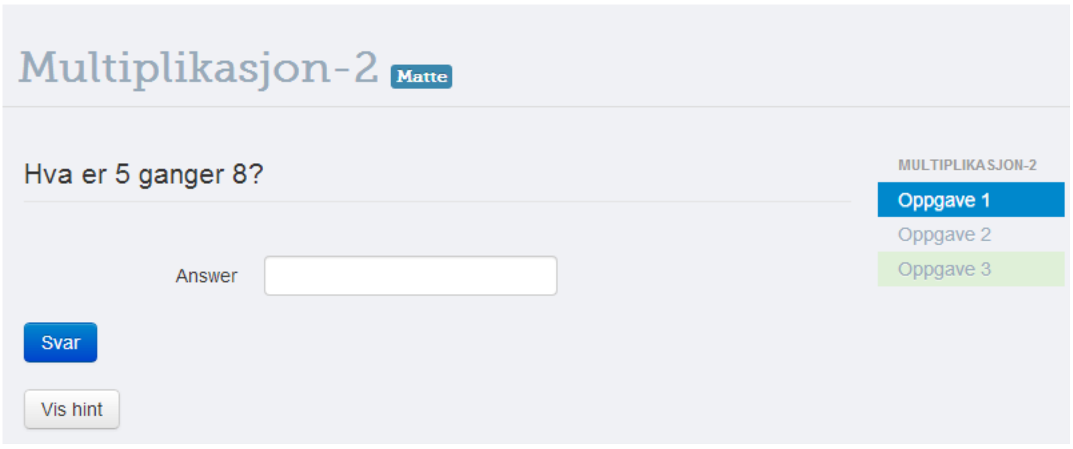
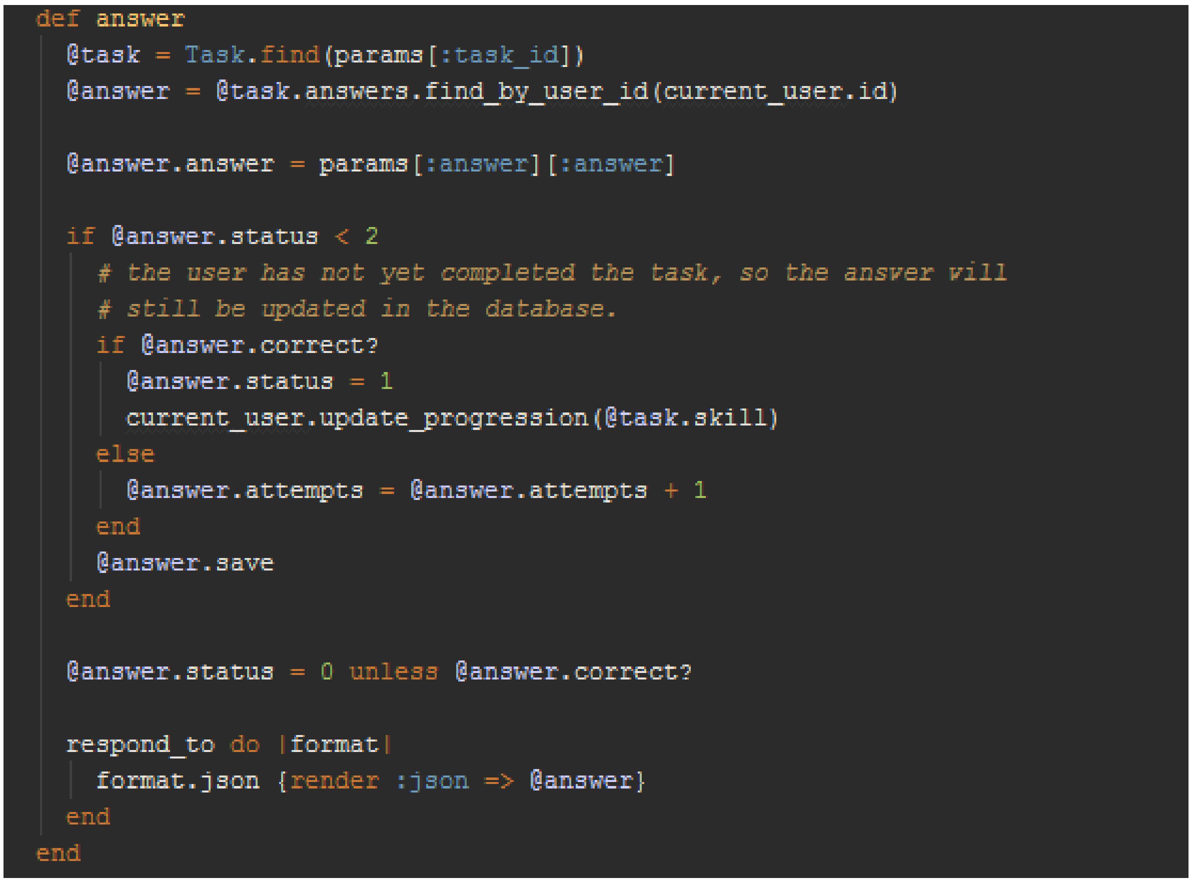


Figure 12.39: The view for answering tasks.

Figure 12.39 shows the answering view rendered by the *answer* action in the TasksController class. Since this is the first time the current user attempts this particular task, a new answer object has been stored in the database. When the student presses the 'Svar' button, a call is made to the *answer* action of the AnswersController class.



```
def answer
  @task = Task.find(params[:task_id])
  @answer = @task.answers.find_by_user_id(current_user.id)

  @answer.answer = params[:answer][:answer]

  if @answer.status < 2
    # the user has not yet completed the task, so the answer will
    # still be updated in the database.
    if @answer.correct?
      @answer.status = 1
      current_user.update_progression(@task.skill)
    else
      @answer.attempts = @answer.attempts + 1
    end
    @answer.save
  end

  @answer.status = 0 unless @answer.correct?

  respond_to do |format|
    format.json {render :json => @answer}
  end
end
```

Figure 12.40: The *answer* action of the AnswersController class.

The first thing to note about the code presented in Figure 12.40, is that there is no need to see if an answer object has been created for the current task and user, since the *answer* action of the TasksController class took care of this. The answer object is fetched from the list of answers associated with the task by providing the id of the current user, and the textual answer submitted through the HTML form seen in Figure 12.39 is added to this object by accessing the URL parameters.

The following conditional statement uses the *status* attribute (which was briefly mentioned in Section 12.3) to determine whether or not the current user has already completed this task. This particular attribute can be either **0**, **1** or **2**: the task is incomplete; the task is completed, but the user has not yet received the associated points; the task is fully completed - the user has received the task's associated points.

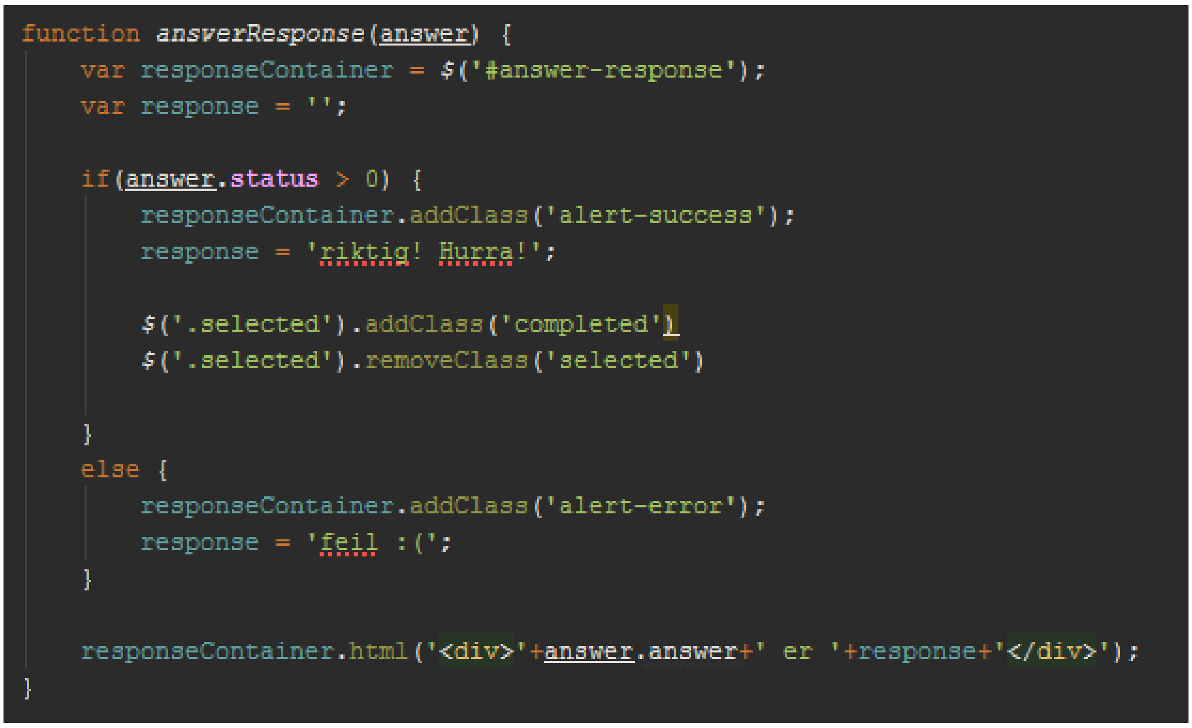
Should the status be less than 2 (i.e. the task is still not fully completed) the answer still needs to be updated in the database and as such the `correct?` method is called on the answer object, to see if the submitted answer from the HTML form is correct. If it is correct then the answer's status is updated, and (seeing as the task is now completed) the current user's skill progression is updated as well to reflect that one of the tasks belonging to a particular skill has now been completed.

This additional update (to the user's skill progression) allows for skills to be completed by completing tasks associated with the skill, or even associated with the skill's distant children. This allows for a dynamic and responsive feedback to the user, due to the fact that all points, ranks and badge rules are updated *after* the code in the *answer* action has been executed.

However, if the submitted answer is incorrect then the prototype increments the *attempts* attribute, indicating the number of failed attempts the student has associated with the task.

Note that the answer object is persisted to the database regardless of whether the submitted answer was correct or not in order to store information regarding failed attempts and/or update the status.

The conditional *if* statement is the only part of the *answer* action that actually concerns itself with updating and storing information to the database regarding users and game artefacts. The actual, visual response to the user regarding her submitted answer is accomplished in the one line of code following the conditional statement, which sets the answer's status to 0 if the answer is wrong, and leaves it otherwise. Note that this last change is not persisted in any way, and is used by JavaScript in the answer view to provide feedback to the user.



```
function answerResponse(answer) {
    var responseContainer = $('#answer-response');
    var response = '';

    if(answer.status > 0) {
        responseContainer.addClass('alert-success');
        response = 'riktig! Hurra!';

        $('.selected').addClass('completed')
        $('.selected').removeClass('selected')

    }
    else {
        responseContainer.addClass('alert-error');
        response = 'feil :(';
    }

    responseContainer.html('<div>'+answer.answer+' er '+response+'</div>');
}
```

Figure 12.41: The JavaScript response code uses the status attribute set in the controller.

By checking the *status* attribute decided by the final line of code in the AnswersController class, the JavaScript decides what feedback to show the user. For an example of the different responses generated by the JavaScript shown in Figure 12.41, please refer to figures 12.42 and 12.43.

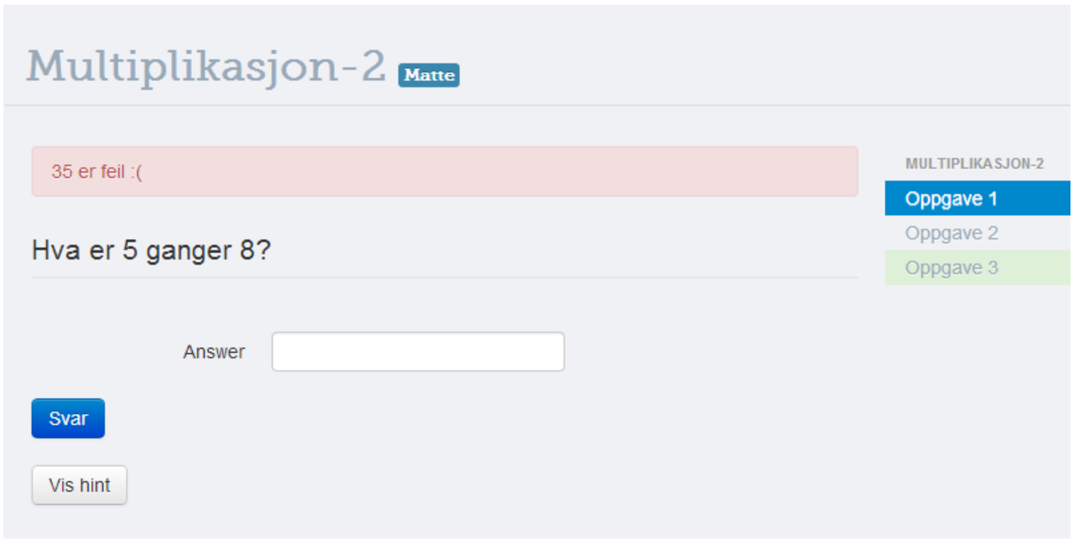


Figure 12.42: The JavaScript-based response to an incorrect answer.

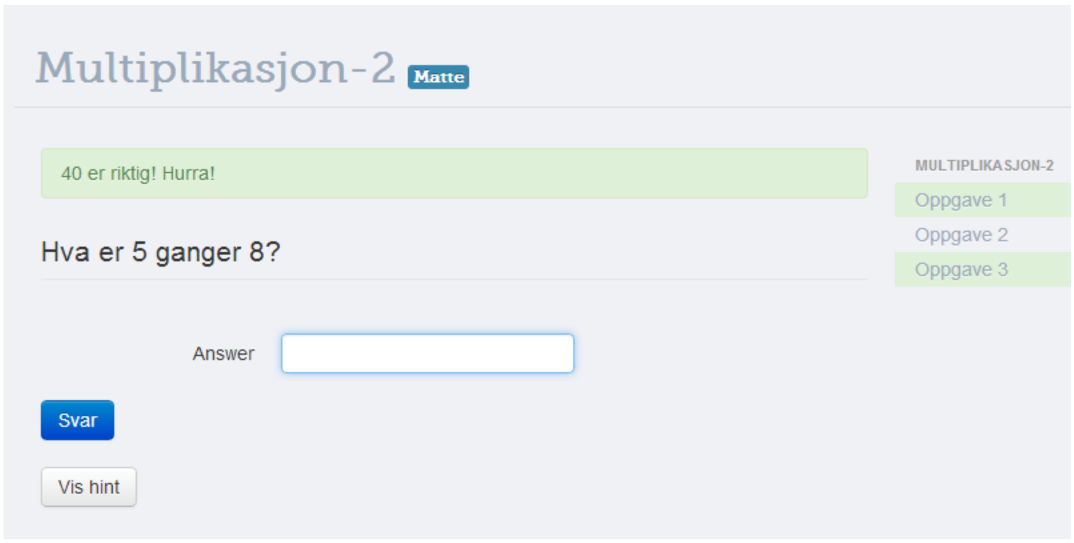
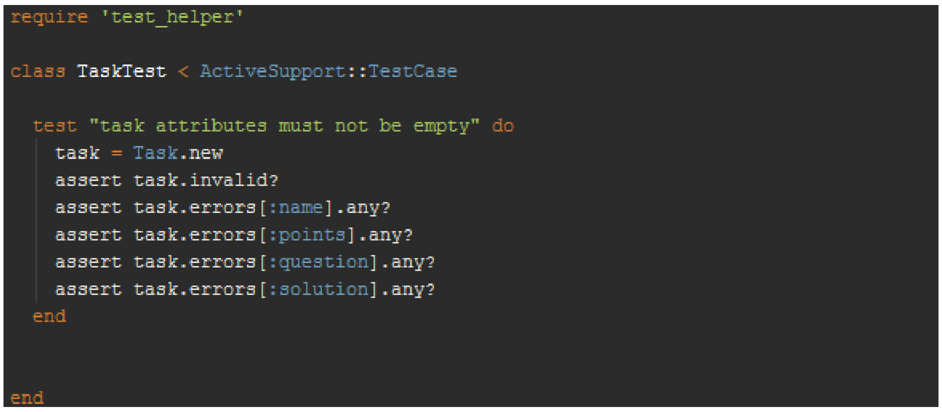


Figure 12.43: The JavaScript-based response to a correct answer.

12.5 Software Testing

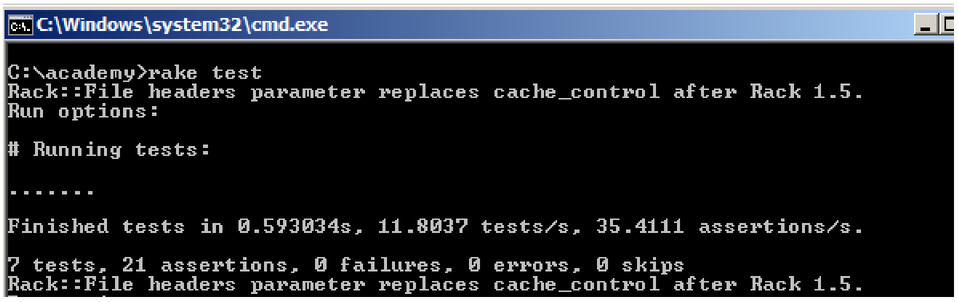
The plan was to write unit tests for the data models, and functional tests to test operations. Unit tests were successfully written and run according to plan, an example can be seen in Figure 12.44. We did however run into problems with the functional tests because we used a third-party module

(Devise) to handle access restrictions. This prevented the built-in functional test tool to access the necessary parts of the application, and we did not find the time to resolve this issue.



```
require 'test_helper'

class TaskTest < ActiveSupport::TestCase

  test "task attributes must not be empty" do
    task = Task.new
    assert task.invalid?
    assert task.errors[:name].any?
    assert task.errors[:points].any?
    assert task.errors[:question].any?
    assert task.errors[:solution].any?
  end


end
```

Figure 12.44: Code for unit tests that check data validation of tasks.



```
C:\academy>rake test
Rack::File headers parameter replaces cache_control after Rack 1.5.
Run options:

# Running tests:

.......

Finished tests in 0.593034s, 11.8037 tests/s, 35.4111 assertions/s.

7 tests, 21 assertions, 0 failures, 0 errors, 0 skips
Rack::File headers parameter replaces cache_control after Rack 1.5.
```

Figure 12.45: The result of tests run through RoR's own testing tool.

Part VI

Discussion

Introduction

The overall aim of the thesis was to design and develop a prototype of a gamified learning platform for elementary school education, in order to reveal opportunities surrounding this type of application. In addition, we wanted to find out whether this type of learning is fun and engaging to students, and whether teachers appreciate and see the use for such an approach.

Part VI - Discussion will discuss to what extent the suggested application is likely to achieve these goals, to what extent the application prototype reached these goals and limitations of the suggested concept. It will also discuss the main limitations of this thesis.

Chapter 13

Obstacles

13.1 Unable to find school for field tests

A big obstacle in terms of testing the impact of the prototype application, was that we were unable to find a school that would let us do field tests. A reason for this may be that the difficulty of this task was highly underrated. As we started contacting local schools towards the end of the school year, teachers were seemingly very busy and thus unwilling to take on the extra workload our field tests would place on them.

13.2 Time Constraints and RoR Complexity

One of the key reasons Ruby on Rails was chosen as the web application framework for this project was that it is built to support rapid development. While we found this to be true, we also found that it is rather complex and takes time to learn. For this reason, significant portions of the development phase were consumed trying to master the framework, and we were only able to implement a subset of the desired features. For this reason, we had to make decisions on which features to include, which will be discussed briefly in this section.

The application prototype necessarily needed to include the game mechanics and social features discussed in the architectural rationale, but only as a proof of concept, meaning that it would not be a full realisation of these concepts - rather a demonstration of their feasibility. As previously emphasised, the game mechanics of choice need to be well-designed in order to have the intended effect.

The application specification defines several objects in order to more clearly define the properties our application. The lessened scope of the prototype application does not encompass all these objects, but instead a subset based on our prioritisation of the prototype's features. In this regard, we defined the following objects in the prototype: **users**, **roles**, **subjects**, **skills**, **tasks**, **rewards**, **answers**, **hints**, **skill progressions** and **classes**. These were considered to be the most essential objects for the prototype application to function as intended.

The prototype consists of several user roles, both students and teachers, with disparate access levels. In order to enforce access levels and authorize user, we included the role system and assigned each created user to a particular role.

As the intention is for teachers to translate their pedagogical and subject matter knowledge into learning materials such as **tutorials**, **skills** and **tasks**. These three distinct objects represent what the user interacts with in the context of the prototype application's game mechanics. When prioritising which requirements to fulfil in the prototype, we concluded that the generation of flow should be the main focus, as this is the concept we perceive to be the most important for the application to succeed.

Chapter 14

Prototype Evaluation

This chapter presents a theoretical evaluation of the prototype. The literature study identified key features that are required to a varying degree in order for the application to achieve its goals. This was formalised in the requirements specification, and this chapter will discuss to what extent these features where implemented in the application prototype, with basis in the requirements specification.

14.1 Implemented Features

This section discusses the key features that were successfully implemented to some extent in the prototype.

Intuitive User Interface

The first major domain-specific challenge we identified was that of student *engagement* and *motivation*. The first non-functional requirement directly pertains to said challenge by stating that the application must have an intuitive interface *regardless of target audience*. While coming to terms with this requirement certainly does not eliminate the challenge of student engagement on its own, it emphasises how a poorly designed user interface may well completely destroy user interest in the application. In addition, we identified challenges related to the teacher as an end user. In order for an e-learning platform to be successful, it has to be understood an embraced by the teachers as well as students.

The application prototype targets this by making use of a simplistic interface with few alternatives. It stays true to common website conventions, and makes use of icons that are already familiar to most Internet-users in order to make the interface easy to understand. While it might seem dangerous to assume that teachers are already Internet-users, it will true at least in the case of Norway, as teachers are already required to use web application tools in their work. The level of proficiency that teachers have with web platforms will only increase as years go by, and older teachers retire.

Instant Access to Tasks

The second general functional requirement, G-NFR2, states that *the application should make it a trivial process to commence relevant school work from anywhere in the application.* It is important to eliminate accessibility as reason for delaying school work, and students should be able to jump right into a task of their choice once they have attained new knowledge they wish to practice.

The application prototype addresses this by always displaying a sizeable "practice" button at the bottom of the user interface. Clicking the mentioned button takes the user to a page displaying subjects to work on. This is a somewhat limited solution however, and does not scale well as the amount of learning material increases. In a finished application it would be desirable with a feature that allows the student to resume recently worked on skills with the click of a button.

Data Handling

We previously discussed the risk of storing and presenting excess amounts of data, which in turn could lead to a cluttered and unintuitive user interface. In this context the importance of information privacy was also emphasised. Teachers need this information in order to properly gauge the learning and progress of their students, but other students need not be privy to detailed information about their classmates. In addition, task management and solutions must be hidden from students, but available to teachers.

This is taken care of by requiring each user to authenticate with a username and password, and by the separation of roles. Student accounts may only see the tasks they have earned access to, and the publicly available information of fellow students. Teachers are allowed to see everything. The technical integrity of this solution is taken care of by Ruby on Rails' built-in support for authorization, which also protects from common security attacks.

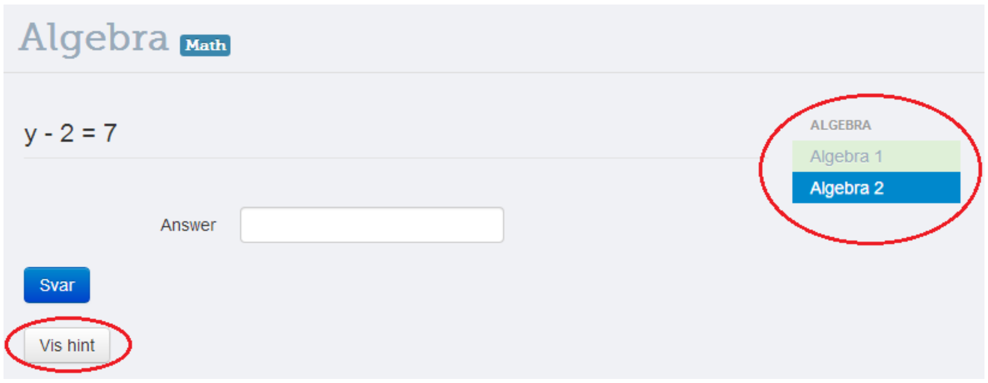
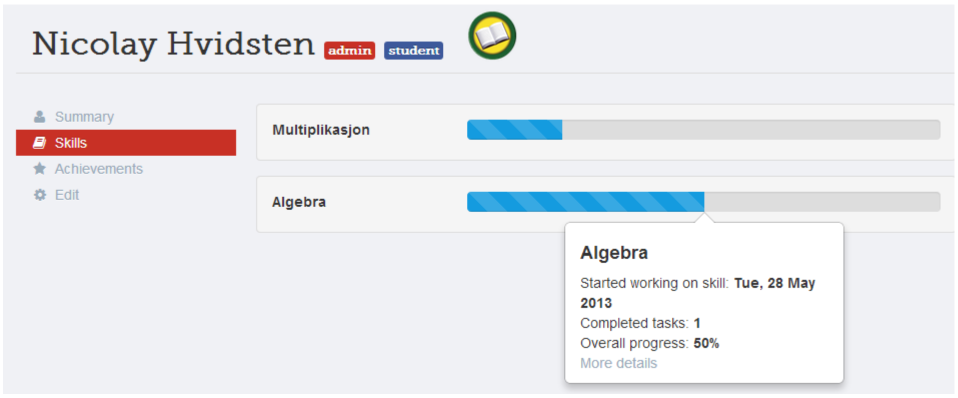


Figure 14.1: Users have the option to show hints when solving a task. To the right is task-progression of current skill.

Generating Flow

Chapter 5.3 discussed the concept of *flow* at length. The importance of this property for the promotion of true learning and student engagement was formalised as the requirement - *The application's inherent game mechanics should use documented approaches to induce flow, naches and fiero.*
Recall from the literature study that flow is *completely focused motivation*, and that when a person experiences this sensation she does not want it to end, but to remain immersed - both winning and quitting are equally unsatisfying outcomes. The process of inducing this sensation is rather complex however, but there is a recurring result discovered in studies: that of providing the users with tasks of gradually increasing difficulty, constantly keeping them on the edge of their ability. Other ways of promoting the generation of flow through game mechanics include well-paced rewards and interactive interfaces to keep the student engaged.

The goal of providing users with gradually increasing difficulty is not directly the responsibility of the application design, but the responsibility of its content, which per the intention of this thesis is to be added by didactical experts - teachers. The application must however facilitate this. The application prototype solves this by allowing teachers to add optional hints (see Figure 14.1) to each task, reducing the difficulty if needed. This does however reduce the points earned in order to motivate students to refrain from using hints when they are not required to. In addition, the application prototype promotes flow by displaying current progress and rewarding users with badges for completing tasks. This can be seen in Figure 14.2.



Algebra
Started working on skill: **Tue, 28 May 2013**
Completed tasks: **1**
Overall progress: **50%**
More details

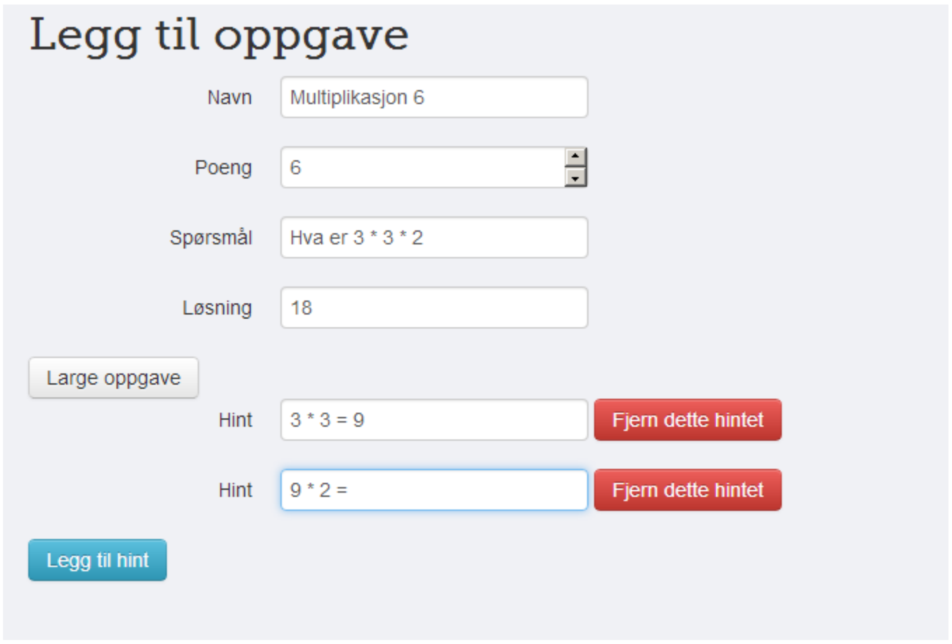
Figure 14.2: Users get a view of their current progression. Badges earned are shown at the top, next to the users' name.

Presenting Task Progress

One of the defined requirements is that *the application should keep students up to date on their recent task progress*. This requirement represents another means of facilitating augmented learning in the sense that the environment adapts to the learning progression of the student. This requirement also aids in mitigating the challenges related to motivating and engaging students in the sense that stipulates that a student will receive relevant feedback regarding a game objective, and resuming work on game artefacts related to this objective should be a trivial process. The ease of access to game mechanics promotes engagement, and also mitigates risks concerning a cumbersome interface, since relevant school work is certainly always an important resource the students should have ubiquitous access to whenever interacting with the application.

This is achieved in the application prototype by displaying the task progress of all the tasks in the current skill, as well as displaying a progress overview on the user profile. This can be seen in Figure 14.1. The idea is to provide the student with information showing her progress in a skill while she is working on a specific task within that skill. That way she can see her total progress in the specific *skill*, which will often be more motivating than seeing her overall progress in the entire *subject*, because perceived progress is faster when zooming in at a subset of tasks.

Ideally, in a finished application, this would be supported by additional features such as skill-tree structuring and displaying different tasks in a hierarchy. In addition, we would have like to provide more dynamic and interesting



Legg til oppgave

Navn	Multiplikasjon 6
Poeng	6
Spørsmål	Hva er 3 * 3 * 2
Løsning	18

Large oppgave

Hint	3 * 3 = 9	Fjern dette hintet
Hint	9 * 2 =	Fjern dette hintet

Legg til hint

Figure 14.3: The page for adding tasks.

graphics to go with the indication of progress, for example a progress bar as this has been shown to be the most effective way of visualising feedback to the end user.

Task Management

NF requirement M-NFR5 addresses that of task creation. It was previously identified that teachers many teachers less familiar with the use of technology, and as such the application an intuitive means of task creation.

The prototype application attempts to achieve this by taking a minimalist approach to the user interface, as seen in Figure 14.3. In addition to a simple and clean design, there are no explicit teacher pages. Teachers simply navigate to the skill that they wish to supplement through the same interface that the students use, as seen in Figure 14.4. This serves a double purpose by making it very clear where in the skill-hierarchy the task is added, while at the same time as providing the teacher with an understanding of how the application looks from a student point of view.

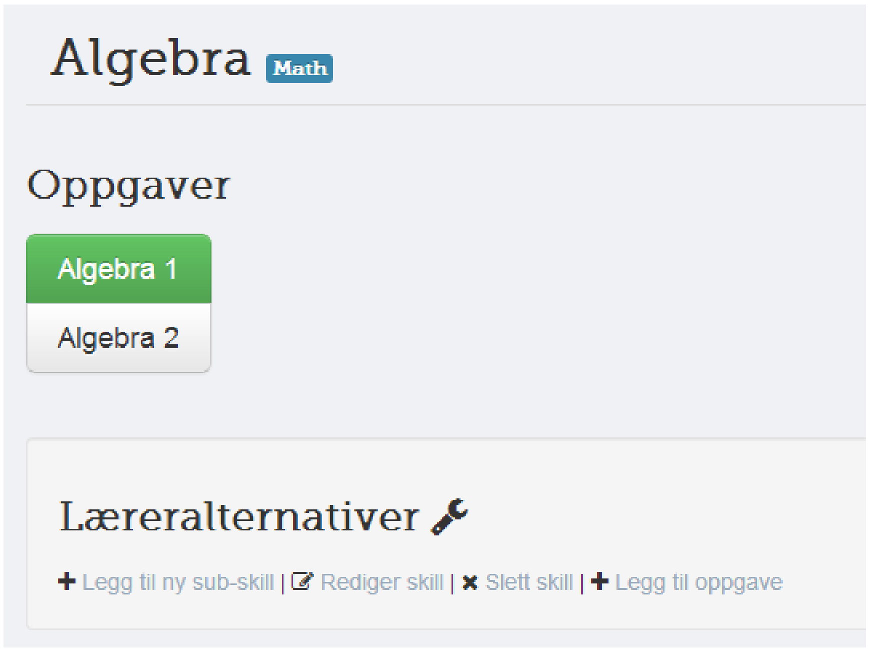

Algebra
Math
Oppgaver
Algebra 1
Algebra 2
Læreralternativer
Legg til ny sub-skill | Rediger skill | Slett skill | Legg til oppgave

Figure 14.4: Teachers are able to edit a skill from the skill page. Editing options are not visible to students.

Rewards

An important requirement states that the application should have a reward system featuring different kinds of rewards. This is handled to some extent in the implementation by rewarding players with points, level-ups and badges for completing tasks.

A more elaborate rewards-system that includes rewards such as access to new avatars and instant rewards in the form of graphic animations would be desireable in a finished application. Also, the option for teachers to add tailor-made badges is a preferred feature that we did not get the time to implement.

14.2 Desired Features

This section discusses the desired features of the proposed application that we did not find the time to implement in the prototype.

Augmented Learning

The first non-functional requirement we have specified in the scope of game mechanics relates to the concept of *augmented learning*, namely that of constantly providing the student with additional information regarding relevant subject matter *just in time.*

A good example of fulfilling would be that of providing the student with resources and examples pertinent to the game objective she is currently working on *while she is working on it.* This is only vaguely reflected in the application prototype by the functionality providing hints to each task. Preferably, a finished product would include access to learning resources such as animated representation of problems and opportunities surrounding the task theme. Video tutorials and instructions would also make good contribution in this sense.

The mentioned features would not only be useful in that of enhancing augmented learning, but when we consider the application-specific challenge of student engagement and motivation, said features may aid the work-flow of the student, promoting immersion in the application.

Gradual Progression

In Section 14.1, we discussed the use of gradual progression in task difficulty as a means of generating flow. While the application prototype facilitates this to some extent, it does not offer the option to have the system restrict access to skills or tasks, based on previously completed attained skills and completed tasks. This is a big limitation of the application prototype.

A finished application should allow for teachers to design hierarchies of learning material and associated tasks in order to force students to attain new knowledge in the correct order, using the attained knowledge as building blocks to a more general subject understanding.

The visual representation of such a hierarchy is of paramount importance, not only for teachers to make use of it as intended, but as a tool for showing students how different parts of a subject are tied together. An excellent example of how this can be achieved Duolingo's skill tree as seen in Figure 6.1.

Interesting Graphics

A number of the non-functional requirement target the generation of immersion and flow, and it was identified that the use of graphic animations should be a key factor in promoting these. Keeping the student immersed was identified as on of the main application-specific challenges, with the disadvantage of not being able to rely upon immersing graphics in contrast to traditional video games. While a web application cannot quite compete with a stand alone video game, web applications see few limitations in the realm of 2D graphics and animation.

For these reasons, finished application should make use of the latest developments in HTML5 and JavaScript in order to create an appealing and game-like graphical environment. It is important that the application does not move too far away from a traditional video game in order to experience the benefits associated with game mechanics, and a novel user interface might be a good middle ground. Similar applications like Duolingo, Khan Academy and Memrise have all had success using interactive JavaScript technologies to make their task answering process a more interesting experience.

Interchangeable Appearance

The non-functional requirement M-NFR12 states that *the visual presentation of the application should be a result of target audience preferences.* The idea behind this requirement is that a if a finished product were to be used in elementary school, there would be a large gap between the kind of appearance a first-grader would enjoy, versus an appearance a ninth-grader will find interesting.

Ideally, the application would counter this by facilitating the use of different graphical themes. The theme for a given class could be set by the teacher. This would also allow the teacher to provide their own touch to their virtual classroom.

Student Created Tasks

A previously discussed emotion related to game mechanics encompassed by *naches.* Recall from our literary study that naches is the feeling of vivacious pride felt by a tutor when their student accomplishes something of magnitude.

A finished implementation could facilitate this by allowing students who fulfil certain criteria to assume the role of mentor for other students, allowing them to monitor their pupils' progression and provide helpful comments. Mentors could also be allow to create tasks for other students to complete, as attachments to the curriculum structured by the teacher.

Peer Motivation Through Competition

A feature identified in the application design process was the facilitation of competition between classes. Such a feature takes advantage of humans' disposition for being engaged by competition, and as such motivates students to do well in order to beat other classes. The intention was to facilitate this through leader boards and heads-up challenges between classes. This was however de-prioritised to focus on the core features of the application.

Ubiquitous Access

An intrinsic property of web applications is that of near ubiquitous access - as long as users are connected to the Internet, they can access the application through their web browser. As previously explained, this allows users to connect to the service from a variety of platforms, such as tables, smartphones or computers. This does however require the application be properly designed to accommodate different devices. In this case that means to provide a responsive design that adapts to the screen size of the users' device. While the application prototype does not address this, it would be highly encouraged in a finished product.

Availability

We have previously mentioned that we feel ubiquitous access covers the aspect of availability, but we still feel the need to formally state a requirement that addresses this challenge. Availability is defined as "the ability of the user community to access the system" [83]. We attempt to cover this requirement in G-NFR5, which deems that the system should be *highly available*.

As the application prototype is not formally deployed, it does not address availability. The challenge of availability is however easily countered by using a cloud service to host the application. There are many providers available that offer scalable hosting services with an incredibly high guaranteed uptime at a reasonable price, and there are even free providers with reasonable guarantees [72].

Chapter 15

Limitations

This chapter discusses the limitations regarding the process and results of the thesis.

15.1 Lack of Field Tests

The lack of field tests is the major elephant in the room. As was discussed in Section 3.3 - Evaluation Approach, the goal was to perform a field test in order to evaluate the features of the prototype on real end users. The lack of this type of test is clearly the biggest limitation of this thesis as we would have received invaluable feedback from both students and teachers. This would provide a very good indication about whether we had chosen the correct features in the context of reaping the potential benefits we identified in our literary study.

The result of this discrepancy is that the prototype evaluation was forced to take place on a purely theoretical basis. Furthermore, it is by definition impossible for the authors to perform an unbiased evaluation, as the evaluators are also the advocates and creators of the proposed solution.

15.2 Limitations of the Proposed Application

Teacher Created Content

Section 8.1 stresses the importance of the teacher as an end user, and how the application would ideally function as a framework within which teachers could apply their pedagogical and academical expertise, translating their knowledge into well-organised task structures.

We did however fail to consider that the games that have already been proven successful as learning tools, were the results of pedagogical deliberation by experts far more specialized than the average teacher, which was one of the main reasons for their success. Since studies have shown that adaptive education is a very advanced topic [101] with many parameters to consider, we assume that it would be an equally difficult task to create game artefacts to adhere to this concept.

In addition, it turns out that creating complete structures of carefully selected tasks is a very time-consuming process. This would inflict an extra burden on teachers, as they currently do not create most of the tasks themselves, but assign them from a book. This extra burden on the teachers places a big limiting risk on the proposed application of this thesis.

This risk could possibly be mitigated by appealing to the teachers' sense of involvement in their own curricula, allowing them to tailor their subjects in a more personalised way than by assigning tasks from text books. However, it is unlikely that this sense of involvement would appeal to all teachers, especially the older generation whom could possibly have taught the same curricula for years and thus might be disinclined to change their approach, and thus the limitation still needs to be taken into account.

The argument can be made that ideally each subject in elementary school would be the result of teachers' pedagogical deliberations regarding the subject matter presented in Kunnskapsløftet [1] rather than a set of (possibly outdated) pre-defined task from a text book. Unfortunately the world is not ideal, and as such the tools presented by the proposed application might go unused, regardless of theoretical merit.

Human interaction

A major aspect of promoting learning and creating an interactive, social classroom environment is that of human interaction between students based on exploration of new topics [49]. The features and properties of the proposed application in this thesis are not overly concerned with this aspect, which would lead to a focus on human-computer interaction rather than human to human.

[1]http://www.udir.no/Lareplaner/Veiledninger-til-LK06/

While the possibility of having students function as mentors to other students was mentioned, the human interaction in this capacity was not emphasised. Khan Academy recognises interactive-based (in the human sense), tangible exploratory learning as one of its four key components [57], and emphasises the importance of employing the virtual resources to empower the physical classroom.

The features of the application concerned with student engagement, immersion and peer motivation still have their intended effect, but this is learning exceedingly based on virtual resources. Ideally the application would be tailored to promote student interaction through group projects and tangible exploratory tasks. An example of which could be to utilise pervasive technologies (such as GPS) to facilitate projects that would require students to work together outside of the classroom.

Empowering the physical classroom through virtual resources would be the ideal goal for the application, but the proposed solution in this thesis might be overly focused on the virtual aspects.

Discrepancies between subject matter and virtual resources

Not all subjects are suited to employ virtual resources, and in this regard the application proposal of this thesis comes up short. In order for the application to successfully challenge the entrenched norms and systems in place in the Norwegian elementary school, it needs to be applied to all subjects in order to become seamlessly integrated, which in its current state it cannot.

A way to mitigate this problem would be to cater to subjects not suitable for virtual resources (for instance physical education (PE) or home economics) by providing additional artefacts based on interactions outside of the virtual scope. An example of which might be to give each student in a PE class the option of tailoring a work-out schedule using the application (perhaps as a sort of skill object) and then complete sessions (theoretically implemented as an alteration of tasks objects) adhering to this schedule.

The argument could be made that these features could be included in the application in future iterations, thus potentially mitigating much of this limitation, but the fact of the matter remains that this limitation was left out of the risk and challenge modelling process in Chapter 7 and thus remains a limitation to an application adhering to the requirements presented in this thesis.

Offline capabilities

Because the application's virtual resources require a constant internet connection, they become unavailable should a school not have internet access. Deploying and maintaining an advanced web application demands expertise in computer science. Schools are very unlikely to possess this expertise, preventing them from making use of this type of application locally.

This is a limitation directly related to the architectural choice of using the web as the platform for the application, and thus cannot be truly mitigated in any other way than by creating an offline version of the system. However, the possibility of such an approach was not covered in this thesis, and will thus be considered a limitation of the proposed application.

Part VII

Conclusion

Chapter 16

Conclusion

16.1 Summary

This thesis set out to investigate the area of using video games in education, and develop an application prototype that utilises some of the benefits of video games in an educational setting, in order to provide a platform that could enhance classroom education. To ensure solid grounds upon which to base the proposition, a literature study was conducted to investigate the area of education and games in order to discover the underlying issues with the current educational model, effective learning and the use of games in this context.

This literature study brought the authors deep into the research area of combining games and education, and identified several explicit game mechanics associated with learning - such as the generation of *flow* through well-paced difficulty and gradual progression, *fiero* through triumph over well-designed, difficult obstacles and *naches* through achievements of tutored players. All these explicitly identified mechanics are a result of attempting to answer RQ1.1.

The extensive literature study resulted in a thorough application specification proposing a web platform with learning tasks and resources - utilising game elements to engage and immerse the student in the application, in order to make her more receptive to learning new material.

The idea behind this concept is that the emotional state of humans when playings games and learning effectively share many commonalities, which was in literature found to be widely supported by research; these theoretical advantages provide a solid answer to RQ1.2. the recent vast growth of web-based gamified applications, and the impressive results they report, leads the authors to believe this could be a solid approach.

The application specification proposes a platform where students may complete sets of instruction/problem-challenges on school subjects in order to gain points and rewards. Gaining points will reward the student with access to more difficult tasks, ensuring a gradual progression through subject matter. The proposition also includes features such as badges, competition and interesting graphics with the aim of being fun and engaging to students by appealing to both intrinsic and extrinsic reward mechanisms. This important combination of intrinsic and extrinsic rewards could work as an extremely powerful incentive for students to use the application, which was the major concern of RQ2.2.

Based on the literary study performed in both the thesis and pre-thesis projects the authors concluded that the best approach to implement such an application to fulfil the necessary criteria would be through a gamified web application. The aspects of ubiquitous access and social features innate in such applications were among the most important reasons for this conclusions, which answers RQ1.3.

RQ2.1 concerns itself with teacher incentives, and the proposed application includes an interface for teachers to add tasks and learning resources, so that they may tailor its use to their curriculum, and advanced monitoring tools for teachers to monitor their students' progress. Providing teachers with these invaluable statistics, and allowing them to tailor game artefacts adhering to their pedagogical and subject matter expertise, could potentially function as very powerful incentives for teachers to utilise such an application.

The application specification was distilled and developed into a prototype containing a portion of the features described in the application specification. Features were prioritised based on perceived importance, and the final application prototype acts as a proof of concept.

The authors believe the proposed application may contribute to more engaging and enjoyable learning for elementary school students, and as a result, increasing the potential for true learning. In addition, it could provide tools for the teacher to assess the class in a more accurate way, and thus spend his time more wisely, leading to a more efficient use of the teacher and potentially increasing student-teacher interaction; which pertains to RQ2.3.

By changing the way subjects are defined, through a set of game artefacts (such as skills and tasks presented in this thesis) rather than a pre-defined set of chapters from a potentially outdated textbook, student engagement can be increased. Also, though contingent upon the quality of the game artefacts, true and augmented learning can be facilitated through this approach. It is the opinion of the authors of this thesis that a teacher who is truly passionate about the subjects he teaches would relish the opportunity to translate his knowledge into tangible artefacts to provide a more interactive curricula for his students, *if* the provided tools and incentive for doing so is good enough.

The proposed application may face some challenges. If one does not pay special attention to quantifying learning effects and effects on classroom dynamics, it may be hard to document actual benefits of the system. Without well documented effects, the application might be difficult to introduce to a school system of long tradition.

The importance of *complementing*, rather than replacing, the physical classroom must also be emphasised in this regard, per RQ2.3. By creating interactive and creative game artefacts within the application which promote human interaction between students outside of the application-setting, one could potentially achieve this complementation.

The lack of user tests and empirical data places a big limitation on the thesis. For similar future studies, it would be preferable to conduct tests on young students and teachers to gather views and ideas different to those of the various research communities.

16.2 Further Work

The most natural step forward upon the completion of this thesis would be a full implementation of the application specification, with a following field study testing the resulting product on real end users. It is vital that such a field test is conducted with high quality learning materials and tasks, as the application itself is only a facilitator and cannot fulfil its purpose without quality content.

An identified limitation of the proposed application is that adding content may be too time consuming for many teachers. In addition, it was identified that setting up an application such as the one purposed is far beyond the expertise of any regular school employee. For these reasons, we propose that further work on this type of concept steers towards a more centralized approach, where schools authorities develop well-formed tasks and learning materials congruent with the curriculum of Norwegian schools. Teachers can then select materials from a central database to assign to their class.

Another identified limitation of the proposed application is that it only focuses on student interaction within the application. Future work on similar concepts should focus on facilitating situations where students are required to cooperate face-to-face, as well as in the virtual sphere.

Bibliography

[1] A list apart. *Responsive web-design.* http://alistapart.com/article/responsive-web-design, [Accessed 10.05.2013]

[2] Aftenposten. *Norske skoler på PC-toppen i Europa.* http://www.aftenposten.no/nyheter/iriks/Norske-skoler-pa-PC-toppen-i-Europa-7178641.html, [Accessed 07.05.2013]

[3] Anderson, C. , Dill, K. *Video Games and Aggressive Thoughts, Feelings, and Behaviour in the Laboratory and in Life.* Journal of Personality and Social Psychology , 78, 772–790, 2000.

[4] Anderson, C , Bushman, B. *Effect of Violent Video Games on Aggressive Behaviour, Aggressive Cognition, Aggressive Affect, Physiological Arousal, and Prosical Behaviour: A Meta-analytical Review of the Scientific Literature.* Psychological Science vol. 12 no. 5 353-359, 2001.

[5] Askins, B. , Gree, A. *A Rails/Django Comparison.* The Open Source Developers' Conference Papers, 2006.

[6] Baird, D. E. , Fisher, M. *Neomillenial User Experience Design Strategies: Utilizing Social Networking MEDIA TO SUPPORT "ALWAYS ON" LEARNING STYLES.* Journal of Educational Technology Systems, vol. 34, 2006.

[7] Barnum, C. M. *Usability Testing Essentials.* Morgan Kaufman, 2011.

[8] Basili, V.R. *The Experimental Paradigm in Software Engineering.* Springer, LNCS 706, 1993.

[9] Bateman, C. *Only a Game: Top Ten Video Game Emotions.* http://onlyagame.typepad.com/only_a_game/2008/04/top-ten-videoga.html, 2008.

[10] Bernsen, N. O. , Dybkjær, L. *Multimodal Usability.* Springer-Verlag, 2009.

[11] Business Wire. *Zynga's FarmVille Becomes Largest and Fastest Growing Social Game Ever.* http://www.sys-con.com/node/1084929. [Accessed 09.03.2013]

[12] Casteleyn, S. , Floarian, D. , Dolog, P. , Matera, M. *Engineering Web Applications.* Springer, 2009.

[13] Charlier, N. , De Fraine, B. *Games-Based Learning in Teacher Education: A strategy to Integrate Digital Games into Secondary School.* 2nd European Conference on Games Based Learning, 2008.

[14] Clark, A. *Natural-Born Cyborgs: Minds, Technologies, and the Future of Human Intelligence* Oxford University Press, 2003

[15] Coe, Robert. *Understanding comparability of examination standards.* Research Papers in Education, Volume 25, Issue 3, 2010.

[16] Constantinidou, F. , Baker, S. *Stimulus modality and verbal learning performance in normal aging.* Brain and Language, vol. 82, 2002.

[17] Common Sense Media Research. *Social Media, Social Life: How Teens View Their Digital Lives.* http://www.commonsensemedia.org/sites/default/files/research/socialmediasociallife-final-061812.pdf. [Accessed 14.05.2013]

[18] *Coursera website.* https://www.coursera.org/, [Accessed 14.03.2013]

[19] Csikszentmihalyi, M. *Beyond Boredom and Anxiety: Experiencing Flow in Work And Play.* Jossey-Bass, 2000.

[20] Dawes, L. , Dumbleton, T. *Computer Games in Education project: Report.* http://becta.org.uk, 2006.

[21] *Dictionary.com.* http://dictionary.reference.com/.

[22] *Django Project website.* https://www.djangoproject.com/ [Accessed 20.02.2013]

[23] Doswell, J. *Augmented Learning: Context-Aware Mobile Augmented Reality Architecture for Learning.* Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06), 2006.

[24] *DragonBox website.* http://dragonboxapp.com/ [Accessed 05.03.2013]

[25] *Duolingo website.* http://www.duolingo.com, [Accessed 13.03.2013]

[26] Eide, P. L. H. *Quantification and Traceability of Requirements.* http://www.idi.ntnu.no/grupper/su/fordypningsprosjekt-2005/eide-fordyp05.pdf, [Accessed 14.03.2013]

[27] Edudemic. *5 Reasons We Use Social Media.* http://edudemic.com/2013/03/5-reasons-we-use-social-media/. [Accessed 14.05.2013]

[28] *Edutopia website.* http://www.edutopia.org/ [Accessed 13.03.2013]

[29] eLearn Magazine. *Gamification: Using Game Mechanics to Enhance eLearning.* http://elearnmag.acm.org/featured.cfm?aid=2031772, [Accessed 13.02.2013]

[30] eLearningworld.eu *From Starwars to Edutopia – George Lucas on Education.* http://www.elearningworld.eu/news/from-starwars-to-edutopia-george-lucas-on-education/ [Accessed 13.03.2013]

[31] Erl, T. , Zaigham, M. , Puttini, R. *Cloud Computing: Concepts, Technology & Architecture*, Prentice Hall, 2013.

[32] Expanded Ramblings. *How Many People Use the Top Social Media, Apps and Services?* http://expandedramblings.com/index.php/resource-how-many-people-use-the-top-social-media/. [Accessed 14.05.2013]

[33] Forbes. *Samsung rides smartphone growth all the way to the bank.* http://www.forbes.com/sites/greatspeculations/2013/04/30/samsung-rides-smartphone-growth-all-the-way-to-the-bank/, [Accessed 10.05.2013]

[34] Forbes. *New Reports Forecast Global Video Game Industry Will Reach $82 Billion By 2017.* http://www.forbes.com/sites/johngaudiosi/2012/07/18/new-reports-forecasts-global-video-game-industry-will-reach-82-billion-by-2017/ [Accessed 17.04.2013]

[35] Fredricksen, E. , Pickett, A. , Shea, P. *Studen Satisfaction and Perceived Learning with On-line Courses: Principles and Examples from the SUNY Learning Network.* Proceedings of the 1999 Sloan Summer Workshop on Asynchronous Learning Networks, 1999.

[36] Fu, F. , Su, R. , Yu, S. *EGameFlow: A scale to measure learners' enjoyment of e-learning games* . Computers and Education Volume 52, Issue 1, 2009.

[37] Gee, J. P. *Good Video Games and Good Learning.* Peter Lang Publishing, 2007.

[38] Gee, J. P. *What Video Games Have to Teach Us About Learning and Literacy.* Palgrave Macmillan, 2003

[39] Glass, R. L. *Facts and Fallacies of Software Engineering.* Addison-Wesley Professional, 2002.

[40] Gosselin, D. *Javascript. Fith Edition.* Cengage Learning, 2010.

[41] Goward, C. *Yes you should test that.* Sybex, 2013.

[42] Guemide, B. , Benachaiba, C. *Exploiting ICT and E-learning in teacher's professional development in Algeria.* Turkish Online Journal of Distance Education-TOJDE, vol. 13, 2012.

[43] Hartl, M. *Ruby on Rails tutorial.* Pearson, 2013.

[44] Hartmann, T. , Klimmt, C. *Gender and Computer Games: Exploring Females' Dislikes.* Journal of Computer-Mediated Communication, 11(4), 2006.

[45] Healthy Community Networks. *10,000 hours of video games - to save the world?* http://hcdhome.wordpress.com/2010/08/23/10000-hours-of-video-games-to-save-the-world/. [Accessed 17.04.2013]

[46] Hilgard, Ernest R., *Rote memorization, understanding, and transfer: an extension of Katona's card-trick experiments*, Journal of Experimental Psychology 46 (4): 288–292, 1953.

[47] Hoeft, F. *Gender differences in the mesocorticolimbic system during computer game-play.* Journal of Psychiatric Research, March 2008.

[48] Holmes, B. , Gardner , J. *E-learning. Concepts and practice.* Sage Publications, 2006.

[49] Holt, J. *How Children Fail.* Pitman Publishing Company, 1964.

[50] Hopkins, D. *School Improvement for Real.* RoutledgeFalmer, 2001.

[51] Huotari, K. , Hamari, J. *Defining gamification: a service marketing perspective.* Proceeding of the 16th International Academic MindTrek Conference, 2012.

[52] Hvidsten, N. , Sverdvik, S. *NTNU Computer Science Specialization Project: Pervasive Learning Environment* http://folk.ntnu.no/severisv/pervasive_learning_environment.pdf. [Accessed 16.02.2013]

[53] Ilmberger, W. , Schrepp, M. , Held, T. *Cognitive Processes Causing the Relationship between Aesthetics and Usability.* HCI and Usability for Education and Work. 4th Symposium of the Workgroup, 2008.

[54] *it's learning website.* http://www.itslearning.eu/its-learning-adopts-microsoft-platform [Accessed 19.04.13]

[55] Kapp, K. M. *The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education.* Wiley, 2012.

[56] Khan, S. *Let's use video to reinvent education.* TED Talk, http://www.ted.com/talks/salman_khan_let_s_use_video_to_reinvent_education.html, [Accessed 17.03.2013]

[57] *Khan Academy website.* http://khanacademy.org, [Accessed 17.03.2013]

[58] Kirriemuir, J. , McFarlane, A. *Literature Review in Games and Learning.* http://telearn.archives-ouvertes.fr/docs/00/19/04/53/PDF/kirriemuir-j-2004-r8.pdf, 2004.

[59] Klopfer, E. *Augmented Learning: Research and Design of Mobile Educational Games.* The MIT Press, 2008.

[60] Kontaku. *The 6 Best Games on Web Browsers* http://kotaku.com/5878907/the-6-best-games-on-web-browsers [Accessed 10.05.2013]

[61] Koster, R. *A Theory of Fun For game Design.* Paraglyph Press, 2004.

[62] Lazzaro, N. *Why We Play Games: Four Keys to More Emotion Without Story.* http://www.xeodesign.com/xeodesign_whyweplaygames.pdf, 2004.

[63] Leddo, J. *An intelligent tutoring game to teach scientific reasoning.* Journal of Scientific Systems, 10, 1996.

[64] Leff, A. , Rayfield, J.T. *Web-Application Development Using the Model/View/Controller Design Pattern.* IEEE Enterprise Distributed Object Computing Conference. pp. 118-127. 2001.

[65] Lenhart, A. , Purcell, K. , Smith, A. , Zickuhr, K. *Social Media & Mobile Internet Use Among Teens and Young Adults.* http://web.pewinternet.org/ /media/Files/Reports/2010/PIP_Social_Media_and_Young_Adults_Report_Final_with_toplines.pdf, 2010.

[66] Lindgaard, G. *Usability Testing and System Evaluation: A Guide for Designing Useful Computing Systems.* Nelson Thornes, 1994.

[67] Martin, H. *Revised Measure of Approval Motivation and Its Relationship to Social Desirability* Journal of Personality Assessment, vol. 48, no. 5, 1984.

[68] McGonigal, J. *Reality Is Broken: Why Games Make Us Better and How They Can Change the World.* The Penguin Press, 2011.

[69] *Measuring Usability With The System Usability Scale (SUS).* http://measuringusability.com/sus.php [Accessed 09.05.2013]

[70] *Memrise website.* http://www.memrise.com, [Accessed 02.04.13]

[71] Merrick, K. E. , Maher, M. L. *Motivated Reinforced Learning.* Springer, 2009.

[72] Metz, Rosalyn. *Cloud Computing Explained.* http://www.educause.edu/ero/article/cloud-computing-explained, [Accessed 14.03.2013]

[73] Mitchell, A. , Savill-Smith, C. *The use of computer and video games for learning.* dera.ioe.ac.uk/5270/1/041529.pdf, 2004.

[74] Nacke, L. , Khaled, R. , Dixon, D. , Deterding, S. *From game design elements to gamefulness: defining "gamification".* Proceedings of the 15th International Academic MindTrek Conference, 2011.

[75] Natvig, L. , Line, S. , Djupdal, A. *"Age of Computers:" An Innovative Combination of History and Computer Game Elements for Teaching Computer Fundamentals.* Frontiers in Education, 34th Annual, 2004.

[76] Nieh, J. , Novik, N. , Yang, S. *A Comparison of Thin-Client Computing Architectures.* Technical Report CUCS-022-00, 2005.

[77] Novak, G. M. , Middendorf, J. *Just-in-time teaching.* American Journal of Physics, vol. 67, 1999.

[78] Obendorf, H. *Minimalism. Designing Simplicity.* Springer, 2009.

[79] Opplæringsloven § 1-2, 1998.

[80] Oslo elev- og læringsombud. http://elevombud.no/. [Accessed 02.05.2013]

[81] Pando Daily. *Memrise adds 5.1 million in new funds.* http://pandodaily.com/2012/12/07/memrise-adds-5-1-million-in-new-funds/ [Accessed 26.03.13]

[82] *Perpetual Technology Group website.* http://www.perpetualworks.com/webcape/details. [Accessed 05.03.13]

[83] Piedad, F. , Hawkins, .M *High Availability: Design, Techniques and Processes.* Prentice Hall, 2001.

[84] Plekhanova, J. *Evaluating web development frameworks: Django, Ruby on Rails and CakePHP* http://ibit.temple.edu/wp-content/uploads/2011/03/IBITWebframeworks.pdf [Accessed 20.02.13]

[85] *Play Framework website.* http://www.playframework.com/. [Accessed 20.02.13]

[86] Prensky, M. *Don't Bother Me Mom, I'm Learning!* Paragon House, 2006.

[87] Prensky, M. *Digital Natives, Digital Immigrants.*
On the Horizon, vol. 9, 2001.

[88] Randel, J.M. , Morris, B.A. , Wetzel, C.D. , Whitehill, D.V. *The effectiveness of games for educational purposes: a review of recent research.* Simulation and Gaming, 23, 1992.

[89] Ricci, K.E. *The use of computer-based videogames in knowledge acquisition and retention.* Journal of Interactive Instruction Development, 7. 1994.

[90] Robinson, K. *Sir Ken Robinson: Why Schools Kill Creativity.* TED Talk, http://www.ted.com/talks/ken_robinson_says_schools_kill_creativity.html, June 2006.

[91] Robinson, K. *Sir Ken Robinson: Bring On The Learning.* TED Talk, http://www.ted.com/talks/sir_ken_robinson_bring_on_the_revolution.html, May 2010.

[92] Rosas, R. , Nussbaum, M. , Cumsile, P. , Marianov, V. , Correa, M. , Flores, P. , Grau, V. , Lagos, F. , Lopez, X. , Lopez, V. , Rodriguez, P. , Salinas, M. *Beyond Nintendo: design and assessment of educational video games for first and second grade students.* Computers and Education 40, 2003.

[93] Rubin, J. , Chisnell, D. *Handbook Of Usability Testing.* Wiley, 2 edition, 2008.

[94] *Ruby on Rails guides.* http://guides.rubyonrails.org/. [Accessed 22.05.2013]

[95] Ruby on Rails website. http://rubyonrails.org/ [Accessed 20.02.2013]

[96] Salen, K. , Zimmerman, E. *Rules of Play: Game Design Fundamentals.* MIT Press, 2003.

[97] Search Engine Journal. *R.I.P. - Top 10 failed social media sites.* http://www.searchenginejournal.com/r-i-p-top-10-failed-social-media-sites/57554/. [Accessed 14.05.2013]

[98] Somervell, J. , Wahid, S. , McCrickard, D. S. *Usability Herustics for Large Screen Information Exhibits.* Human-Computer Interaction. IOS Press, 2003.

[99] Squire, K. D. *Video games in education.* Int. J. Intell. Games and Simulation 2 (1), 49-62, 2003.

[100] Stetina, B. U. , Kothgassner, O. D. , Lehenbauer, M. , Kryspin-Exner, I. *Beyond the fascination of online-games: Probing addictive behaviour and depression in the world of online-gaming.* Computers in Human Behaviour, 2010.

[101] Strandkleiv, O. , Lindbäck, S. *Hva er tilpasset opplæring?* http://www.elevsiden.no/tilpassetopplaering/1104529521. [Accessed 15.03.2013]

[102] Sugar, W. , Crawley, F. , Fine, B. *Examining teachers' decision to adopt new technology.* Educational Technology and Society, vol. 7, 2004.

[103] Søgnen, A. *I første rekke: Forsterket kvalitet i en grunnopplæring for alle.* http://www.regjeringen.no/Rpub/NOU/2003 2003/016/PDFS/NOU200320030016000DDDPDFS.pdf. [Accessed 25.05.2013]

[104] Techcrunch.com. *Duolingo Raises 15M Series B Round Led By NEA.* http://techcrunch.com/2012/09/17/duolingo-raises-15m-series-b-round-lead-by-nea-will-expand-to-more-languages-and-to-mobile-soon/. [Accessed 22.03.13]

[105] Thompson, C. *How Khan Academy Is Changing the Rules of Education.* Wired Magazine, 2011.

[106] The 60 second marketer. *Facebook Users by Age.* http://60secondmarketer.com/blog/2011/08/02/facebook-users-by-age-infographic/. [Accessed 29.04.2013]

[107] The Wall Street Journal. *Coursera Makes Case for MOOCs.* http://online.wsj.com/article/SB10001424127887324715704578483570761525766.html. [Accessed 15.05.13]

[108] Tv tropes. *Fake difficulty.* http://tvtropes.org/pmwiki/pmwiki.php/ Main/FakeDifficulty, [Accessed 14.02.2013]

[109] van Veenendaal, E. *Standard glossary of terms used in Software Testing.* http://www.dstb.dk/xdoc/wss/39/docs/95/ISTQB-Glossary-of-Testing-v2.1-2010.pdf. [Accessed 01.05.2013]

[110] Vesselinov, R. , Grego, J. *Duolingo Effectiveness Study.* http://static.duolingo.com/s3/DuolingoReport_Final.pdf. [Accessed 17.03.2013]

[111] W3 Schools. *Browser Stats.* http://www.w3schools.com/browsers/brow sers_stats.asp. [Accessed 10.04.2013]

[112] Wiegers, Karl E. *Software Engineering*, Redmond Microsoft Press, 2003.