Abstract

**Web-Based Interactive Math Learning Environment for Secondary Education**

*Malin Lindvall and Stavros Mavrikis*

Present-day students did not have the same upbringing as students of previous generations; today's learners are "digital natives", those who grew up using the computer [64]. A generation gap exists between current students and teachers, the "digital immigrants" [64]. It would be interesting to identify interface components that could make a digital learning environment more similar to the pen-and-paper method, in an attempt to bridge the generation gap. In this thesis, we describe how we designed and developed an application, called Make it Equal+, in an effort to improve computer-enhanced learning methods by combining aspects of the pen-and-paper approach and educational games. We tried to connect those two components and extract the best of both worlds - such as providing the freedom of the traditional pen-and-paper method, while at the same time giving immediate feedback that is commonly found in educational games. Usability evaluations showed positive results and helped us identify the application's interface elements that reminded the evaluators of solving problems using the pen-and-paper method.

Keywords: educational games, pen-and-paper, computer-enhanced learning
Acknowledgements

We would first like to thank Interactive Solutions for the provision of our work space and their support throughout the project. They offered us advice and technical expertise whenever needed. We would also like to thank Roger Lasu, our supervisor and the person whose idea we implemented, for his valuable user-centered feedback during our frequent progress meetings.

Additionally, we would like to express our gratitude to our reviewer, professor Lars Oestreicher at Uppsala University, Department of Information Technology; his support and advice on the user interface elements of the project was invaluable. We would also like to thank the math professor and researcher, who preferred to remain anonymous, that gave us his expert opinion on the math learning features of the application developed for this thesis. Finally, we would like to thank the individuals who took the time to help us evaluate the usability of our application.
# Table of contents

List of figures ........................................ xi
List of tables ........................................... xiii
Nomenclature ............................................ xv

## 1 Introduction
1.1 Background ................................................................................. 1
1.2 Motivation .................................................................................. 2
1.3 Previous research ....................................................................... 2
  1.3.1 Monkey’s Revenge ................................................................. 2
  1.3.2 Aplusix ............................................................................... 3
  1.3.3 ALEKS .............................................................................. 4
1.4 Problem formulation ................................................................. 5
1.5 Research Question ....................................................................... 5
1.6 Scope/Limitation ......................................................................... 6
1.7 Target group .............................................................................. 6

## 2 Method
2.1 Method Description .................................................................... 7
2.2 Artifact creation .......................................................................... 7
2.3 Evaluation .................................................................................. 8
  2.3.1 Expert Opinion ................................................................. 8
  2.3.2 Usability Evaluation ........................................................ 8

## 3 Requirements
3.1 Goals ...................................................................................... 11
3.2 High-Level Solution ................................................................... 13
  3.2.1 User-friendly interface for children .................................... 13
3.2.2 Fluid experience ........................................ 13
3.2.3 Personalized experience for each user ................. 14
3.2.4 Storage of content ...................................... 14
3.2.5 Create a space where users can freely organize solution steps 14
3.2.6 Provide an aide to guide the user’s solution solving process 15

4 Implementation ................................................. 19
4.1 Software Tools .............................................. 19
4.1.1 Visual Studio .............................................. 19
4.1.2 ASP.NET .................................................... 19
4.1.3 AngularJS ................................................... 21
4.1.4 HTML ......................................................... 21
4.1.5 CSS .......................................................... 22
4.1.6 JavaScript .................................................... 22
4.1.7 C# .......................................................... 22
4.1.8 Bootstrap .................................................... 23
4.2 Design Choices ................................................ 23
4.2.1 User interface features .................................... 23
4.2.2 Revisions along the way .................................. 25
4.3 Architecture and Implementation ........................... 31
4.3.1 Back End ....................................................... 31
4.3.2 Front End ....................................................... 34

5 Results .............................................................. 47
5.1 Expert Opinion .................................................. 47
5.2 Usability Evaluation ........................................... 48
5.2.1 8-year-old girl - Group 1 .............................. 49
5.2.2 12-year-old boy #1 - Group 1 ......................... 50
5.2.3 12-year-old boy #2 - Group 1 ......................... 50
5.2.4 23-year-old woman - Group 2 ....................... 51
5.2.5 25-year-old woman - Group 2 ....................... 52
5.2.6 26-year-old man - Group 2 ......................... 53
5.2.7 28-year-old man #1 - Group 2 ....................... 54
5.2.8 28-year-old man #2 - Group 2 ....................... 55
5.2.9 29-year-old man - Group 1 ......................... 56
5.3 Requirements .................................................. 57
### Table of contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Analysis</td>
<td>59</td>
</tr>
<tr>
<td>6.1</td>
<td>Analysis of the expert’s opinion</td>
<td>59</td>
</tr>
<tr>
<td>6.2</td>
<td>Analysis of the usability evaluation</td>
<td>59</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Issues with the current interface</td>
<td>60</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Validation of design choices</td>
<td>64</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Answering the research question</td>
<td>65</td>
</tr>
<tr>
<td>7</td>
<td>Discussion</td>
<td>67</td>
</tr>
<tr>
<td>7.1</td>
<td>Discussion of results</td>
<td>67</td>
</tr>
<tr>
<td>7.2</td>
<td>Relation to previous work</td>
<td>68</td>
</tr>
<tr>
<td>8</td>
<td>Conclusion</td>
<td>69</td>
</tr>
<tr>
<td>8.1</td>
<td>Future Research</td>
<td>69</td>
</tr>
<tr>
<td>References</td>
<td></td>
<td>71</td>
</tr>
</tbody>
</table>
List of figures

1.1 Monkey’s Revenge ........................................... 3
1.2 Aplusix .......................................................... 4
1.3 ALEKS ............................................................ 5

3.1 Pen-and-paper problem solving method ..................... 11
3.2 Use cases .......................................................... 16

4.1 IDE: Visual Studio ................................................. 20
4.2 Description of ASP.NET 5 ........................................ 20
4.3 AngularJS Components .......................................... 21
4.4 Proportional display of fractions ................................ 24
4.5 Original Design of Interface ..................................... 28
4.6 First revision of interface design ................................. 29
4.7 Second revision of interface design .............................. 29
4.8 The welcoming view ............................................. 35
4.9 The sign in view ..................................................... 36
4.10 The register view .................................................. 36
4.11 Choosing a subject ............................................... 37
4.12 Main game view ................................................... 37
4.13 Cell expansion .................................................... 40
4.14 Operator cell expansion ......................................... 41
4.15 General hint ...................................................... 42
4.16 Specific hint ...................................................... 43
4.17 Workspace with borders ........................................ 45
List of tables

4.1 The users table ................................................... 32
4.2 The roles table ................................................... 32
4.3 The user roles table .............................................. 32
4.4 The subjects table ............................................... 33
4.5 The chapters table ................................................ 33
4.6 The sections table ................................................ 33
4.7 The problems table .............................................. 34
Nomenclature

**Acronyms / Abbreviations**

API  Application Programming Interface  
CSS  Cascading Style Sheets  
GBL  Game-Based Learning  
HTML  Hyper Text Markup Language  
HTTP  Hypertext Transfer Protocol  
IDE  Integrated Development Environment  
MiE  Make it Equal+  
MVC  Model–View–Controller  
MVVM  Model–View–ViewModel  
PBKDF2  Password-Based Key Derivation Function 2  
RDBMS  Relational Database Management System  
REST  Representational State Transfer  
SEG  Serious Educational Game  
SPA  Single-Page Application  
SVG  Scalable Vector Graphics  
URI  Uniform Resource Identifier  
VS  Visual Studio
Chapter 1

Introduction

The topic of Serious Educational Games (SEGs) has seen a surge in popularity in recent years [53] and Game-Based Learning (GBL) is widely used in a variety of environments nowadays [54]. Moreover, research reveals that students have “become disengaged with traditional instruction” [59]. Since SEGs have a promising part in education [61], they could contribute to closing the generational gap between teachers and students, the "digital immigrants" and "digital natives", respectively [64].

1.1 Background

Researchers have been studying the potential of games in education and how they can motivate and pique the players’ interest since the 1980s [67]. By the end of 1990s, the research expanded to electronic games and large-scale online games.

SEGs are computer games that are designed for educating users in a specific field [53]. They have the potential to improve learning outcomes and might even be better at teaching science than normal lectures [61]. The problem however, specifically for math and algebra, is that the majority of software tools designed and developed for this purpose have not accomplished what they set out to do in the classroom [60].

Many SEGs that teach algebra, cannot offer the same level of freedom and ease of use when solving exercises as the simple pen-and-paper method. The majority of the available products offer multiple-choice exercises only, or simply provide input fields that the student can use to type in the final solution to an equation. Even the more sophisticated games restrict the users in some way, such as limiting the number of steps they can take in the process of solving an equation, or providing a narrow range of tools that they can use to solve it (e.g. not providing exponents or logarithms), at their disposal.
1.2 Motivation

Many students struggle with mathematics during their whole education. Designing and developing an application which aids and guides a struggling student could change their perception of mathematics and encourage them to improve their problem solving skills until they have a firm grasp of the content.

Providing the students with a fun and interesting environment that also gives them the freedom of forming expressions as they would on paper, might pique their interest and motivate them to solve more exercises. Building such an application could also persuade teachers to integrate more computer-enhanced learning methods into the classroom, since it could appeal more to the new generation of learners.

1.3 Previous research

There are a large number of math games available online, such as Math Playground [43], Weebly Interactive Sites [39], Cool Math Games [21] or Educational Math Games [28]. Most of those are are simple flash games though that, while providing aesthetically pleasing visuals and graphics for children, cannot be considered as SEGs and would not be used in a classroom.

Some of the more advanced and sophisticated educational environments include "Monkey’s Revenge" [45], "Aplusix" [8], and "ALEKS" [4].

1.3.1 Monkey’s Revenge

Educational games carry the risk of reduced learning efficiency in the classroom because their gaming aspects might absorb the users’ intellectual resources. The educational games can overwhelm the users with details and complexity meant to maximize fun in the gaming environment, and overload their working memory [66]. Monkey’s Revenge (Figure 1.1) aims to avoid that risk by creating a learning environment and only including some game-like elements, with the ultimate goal being to preserve the learning effectiveness of a traditional system, while at the same time making it more engaging and fun.

Rai and Beck conducted a study in which four different versions of the system were developed, with increasingly more game-like elements added to each of them, such as narrative and visual feedback [66]. 252 middle school students, aged 12 to 14 years old, participated in evaluating the different versions of the learning environment.

The results showed that the students preferred the version with the most game-like elements and significant learning growth was observed among them. The results also
1.3 Previous research

suggested that "adding more tutorial features in a game-like environment leads to higher learning" [66].

Fig. 1.1 Monkey’s Revenge[65]: Some of the game-like elements include two characters and a progressing story between them, as well as basic graphics.

1.3.2 Aplusix

Aplusix (Figure 1.2) is an interactive environment for learning algebra. Aplusix gives users the freedom to form algebraic expressions as they see fit, similar to what they can do on paper [60], which stand apart from other similar tools. Aplusix also offers suggestions and hints to the user when they get stuck in an exercise.

Various tests and experiments have been conducted with Aplusix in different countries over the years, and the conclusion was that it can be beneficially used in schools to help a student to learn algebra. The results of the previously cited paper [60] also show that Aplusix can have a positive influence on a student’s learning process if certain conditions are met, such as the student already has a strong arithmetic base and a fundamental understanding of the equal sign.
1.3.3 ALEKS

Assessment and Learning in Knowledge Spaces (ALEKS), uses artificial intelligence in order to determine the types of problems a student should learn and practice. In contrast to the previous examples, although it is a web-based system, ALEKS does not have game-like elements; instead, information is presented to the student in the same fashion as a textbook (Figure 1.3). ALEKS can evaluate the student’s skill level and modify the content based on the results. Accurate feedback is presented when the student submits erroneous solutions or stops making progress. ALEKS also provides the student with new content when the system detects that the student is prepared to learn new information [56].

Canfield and Ward conducted a small study in order to determine students’ attitudes toward the learning environment. In the study, 30 entry and exit surveys were completed by students. Of the 30 students surveyed, 24 expressed that “they learned as much or more mathematics with ALEKS, they would take another course which used ALEKS, and they would recommend the use of ALEKS to another student” [56]. The results indicate that learning environments have the potential to be accepted by the students with a positive attitude, meaning that they may actually enjoy solving math problems using the tool.
1.4 Problem formulation

Math-learning environments used by children aim to maintain the user’s interest and have a high level of usability in order to encourage the student to continue practicing math. An environment consisting of easily understood features could enable the user to solve problems. One feature that is missing from current math-learning environments is a workspace where the user can solve problems on the screen, which causes the student to resort to solving problems on paper. For example, mathematics-learning technologies such as ALEKS can provide certain features to the learner. However, ALEKS does not provide any type of working space for the student since it has the visual design of a traditional textbook with an input box, where the user can type the solution. Other environments have similar problems.

By requiring the student to use a separate method to solve the problem, the math-learning environment simply becomes another textbook-like tool and may not seem intriguing to the user. In this case, the environment only outputs information to the user and the user must only input the final solution. Integrating the problem solving method with the general features of existing environments could improve the overall experience of practicing math.

1.5 Research Question

Learning environments seem to be a key component to education’s future since technology can potentially improve the methods used to teach mathematics [55] and most students have access to a computer and the internet [63]. We thought it would be interesting to identify
the interface components of a mathematics learning environment which would simulate the pen-and-paper learning method and include game-like elements in order to appeal to a generation that has "become disengaged with traditional instruction" [59].

We decided that an important aspect to making a math-learning environment more interesting to the user would be including a workspace where the user can solve problems on the screen, which led us to the following research question.

**Research Question:** "Which interface components contribute to simulating the pen-and-paper method of solving mathematical problems in a computer-enhanced learning environment?"

### 1.6 Scope/Limitation

The scope of the project was limited to 8- to 15-year-old students; higher level mathematics taught to older students would require more tools to be provided to them for solving increasingly complex problems. The learning material and formatting of the workspace was also limited to algebra, since adjusting it to different subjects (e.g. trigonometry) would necessitate more developing time. Additionally, a math-learning environment typically includes a tutorial to teach the student more math topics; however, a tutorial was not within the project’s scope due to time constraints.

### 1.7 Target group

This thesis is intended for software developers interested in designing and implementing an educational environment or a Serious Educational Game for teaching math in secondary education. The authors hope that it will also be useful to developers researching about educational games in general.
Chapter 2

Method

In this chapter, design science, the scientific approach used to answer the research question of this thesis, is described.

2.1 Method Description

Design science is an empirical and inductive research method [27]. It is an outcome-based research methodology, where the focus lies on the creation of purposeful artifacts for a specific problem domain, with the goal being to solve that particular problem. Design science is useful when there are no current solutions to a specific problem and the proposed artifact is new and innovative for that particular realm. The artifact is then evaluated to ensure its utility and usefulness.

We chose design science for this thesis because it allows for the creation of a new artifact [26], since we could not find any other available application that would allow the user to solve a problem on the screen with enough flexibility. A new design would be needed in order to combine the pen-and-paper methodology with a computer-enhanced learning environment, an attempt to close the gap between teachers’ and students’ generations.

2.2 Artifact creation

In order to answer the research question, we designed and developed a web-based software application, Make it Equal+ (MiE). The application’s design was influenced by previous research and other freely available software tools, but we implemented the application from scratch and enhanced it with a variety of new ideas, features, and elements.
The application’s interface was influenced by previous works, exploring what children want in a learning environment and the aspects of a game or learning environment which capture a user’s attention and encourage them to continue using the environment. The method was supplemented with an iterative quality improvement process, called Plan Do Check Act, or PDCA, which allows for many changes throughout the design and development phases [3].

The resulting prototype of the application then underwent an evaluation process, starting with it being presented to an expert in mathematics education and receiving their feedback and opinions, and followed by a usability evaluation conducted on children and adults.

2.3 Evaluation

2.3.1 Expert Opinion

We showed the math-learning application to an expert in mathematics education, who shared his opinions regarding how the application could allow students to develop ideas. His opinion was important because it revealed the perspective of researchers who are trying to improve the math-learning and teaching experiences. The expert was contacted in order to identify ways in which the application could be improved and to verify whether or not the concept could be beneficial to students in the future. Based on his feedback, within-scope comments initiated changes on the application and other comments were added to the list of future improvements.

2.3.2 Usability Evaluation

A usability evaluation was also conducted in order to identify the strengths and weaknesses of the interface, to detect which areas did or did not allow the user to perform certain tasks. The goal of the evaluation was to extract improvements to be made in the future and a verification of how well the interface was designed from a usability perspective.

We welcomed any insights the users had regarding the interface, because it would be considered part of the answer to the research question. The suggestions from the users were taken into consideration to improve upon the interface in the future and allow for a design which is even more natural and similar to the pen-and-paper method of solving exercises.

The usability evaluations resulted in qualitative feedback from the users. The feedback was analyzed and categorized into positive / neutral and negative responses, depending on the comments made by the evaluators. The responses considered to be negative were added to the list of possible future improvements. Neutral and positive responses were studied and
2.3 Evaluation

added to a list of factors which are possibly important for making the application similar to the pen-and-paper method. The experiences of the evaluators were considered highly important when answering the research question.
Chapter 3

Requirements

3.1 Goals

The goal of this thesis was to design, develop, and evaluate a prototype of an interactive math-learning application. Following are some of the considerations that were made regarding the design of the application:

1. When solving a math problem, there are a series of steps to be followed called PEMDAS or BODMAS. The two acronyms serve the same function of guiding a student through the correct order of solving a problem; however, different words are used to describe the same process. Due to the identical meaning, only PEMDAS will be described. P stands for *parentheses*, indicating that the student should first solve...
expressions found within parentheses. E stands for *exponents*, since exponents should be solved before any other operation. M and D stand for *multiplication* and *division*, respectively, and the student should solve any multiplication and division expressions in the order they appear from left to right. Finally, A and S stand for *addition* and *subtraction*, respectively. The student should solve any addition and subtraction expressions in the order they appear from left to right. When solving the problem in 3.1 the PEMDAS order was followed, listing the solution as a series of steps.

2. Students using a digital learning application could feel more motivated to continue practicing math if the application had a user-friendly interface and resembled a game. The inclusion of a workspace could make the application more usable. In order to encourage students to use the program, it could allow the student to manipulate the workspace in a similar fashion as the modification of an equation on paper.

3. Students may experience anxiety if the problem to be solved is too difficult, or boredom if the problem is too simple [57]. Instead, if the problem matches the student’s skill level, they may achieve the feeling of flow and continue using the application without thinking about time or their surroundings. According to Van Eck, a problem cannot be solved too easily in order for the application to be engaging; instead, it should “constantly require input from the learner and provide feedback” [59].

4. Use of the program could provide the student with an aide, or intelligent tutor, even when a human tutor is not available. The type of information provided by the aide could vary in order to help the student during different phases of the problem solving process.

5. Creating a tool which enables students to develop their ideas comfortably and aids them throughout solving the problem could encourage the students, especially the ones who are struggling, to continue working on challenging problems. Usage of the tool could enhance the learning experience of solving a problem.

6. Storing the problems to be solved as exercises in a permanent manner could make the application more extensible and future-proof; teachers could even be able to add their own exercises, if provided with an interface that gave them that possibility.

Based on the aforementioned considerations, previous research, and ideas of the authors, the high level requirements of the application were summarized as:

3.2 High-Level Solution

3.2.1 User-friendly interface for children

One goal was to make the interface appealing to children of our target age group, 8-15 years old. That meant including large, clear text with direct and meaningful descriptions. The buttons were also designed to be big and clearly visible, with informative text or images which plainly describe their function.

The colors chosen for the background were vibrant and at the same time pleasing to the eyes, hoping to draw the attention of the user and to make the application interesting and engaging. Most of the buttons were also colored similarly to blend in with their surroundings, with the exception of some buttons that required special attention from the users, indicating a warning; those buttons were colored differently to provide contrast.

All the pages of the application, besides the workspace, were decorated with background images to give it a more lively and enjoyable appearance. With that goal in mind, some of the elements and buttons of the toolbars were also designed to include images inside them instead of explanatory text.

Another idea that we had was to include a mascot in the application - a character that the user could associate themselves with. The character would talk to the user, teaching them new concepts or helping them through any hardships they might be experiencing with an exercise.

3.2.2 Fluid experience

We wanted the application to feel like a game, for the user to quickly and easily navigate through the different parts and levels, without having to depend on the speed of their internet connection or on its continuous availability. The way to do that was with either a native, desktop application, or a single-page web application.

- Provide the users with a fluid experience.
- Give a personalized experience to each user.
- Permanently store the available problems to be solved.
- Provide a space where the users can freely formulate expressions and shape them however they want.
- Present an aide to guide the users’ solution solving process.
The problem with native applications is that they depend on the end user’s system; an application built for Windows will not work on Linux for example, nor will it work on a tablet or a mobile phone. Since it was our desire for the resulting application to be able to run on any kind of platform, we chose to build it as a single-page web application.

3.2.3 Personalized experience for each user

It was our opinion that each user should have their own personalized experience with the application. Their progress through the available content should be tracked and any interface options they have changed should be remembered for future sessions.

Since it was decided that the prototype would be built as a single-page web application, providing that kind of personalized experience would be possible via an authentication / authorization mechanism, a log in system. Keeping track of who is currently logged in could allow the application to be catered to their specific needs. We decided to build such a system and used cookies to save the users’ information throughout their session and to authenticate them when communicating with the server.

3.2.4 Storage of content

When building a web application, the most common and widely used way of permanently storing content is in a database. We categorized the data that needed to be saved into different tables and used a relational database management system (RDBMS) to store them. RDBMS was chosen over other alternatives, like NoSQL for example, because of the nature of the data, which had a simple tabular structure and did not require multiple levels of nesting.

The tables were named Users, Roles, User Roles, Subjects, Chapters, Sections and Problems. They are self-explanatory and contain all the math data (exercises and their solutions for example) of the application, as well as the users’ information, such as their credentials and roles. Every table can easily be modified or filled with additional data.

3.2.5 Create a space where users can freely organize solution steps

Space for users’ solving of problems

A scratch-pad resembling a paper was added to the environment in order for the user to solve a problem. The scratch-pad area was designed to have guidelines or a grid, since math problems can be solved on blank, lined, or grid paper. Within the scratch-pad or workspace area, we wanted to enable the user to be able to easily add and manipulate values, variables, and operators in a way that is similar to problem solving on paper. Additionally, the ability
to make the workspace look like the user’s writing on paper with regards to sizing and formatting, could appeal to the user.

**Restore to previous state**

As a user makes progress solving a problem within the workspace, they could make mistakes. For example, an element could accidentally be removed from the workspace. A typical feature of text editors is the ability to "undo" a move and the same feature should be available to the user within the workspace. This feature could save the user a lot of time and prevent frustration, especially when learning to use the workspace.

The same logic was applied to a "redo" button. If the undo button is pressed too many times, the user may need to press the redo button, which could save time while solving a problem.

Additionally, the user may feel that they need to reset the workspace at some point. The ability to start over may help the user to solve the problem in a new, better way. Hence, a feature to reset the workspace could be very helpful.

**Display solution in steps**

Oftentimes, teachers consider the method used to reach a solution to be more important than the solution itself. Although the user’s goal in solving a problem would be to find the solution, it is important that they can show the steps completed during the problem solving process. We designed the "Solution Steps" feature to help the users organize and document their thoughts, as they typically do on paper.

**3.2.6 Provide an aide to guide the user’s solution solving process**

**Checking of answers**

We decided to verify each step of the problem solving process in order to keep the user on the right path to finding the correct solution. We considered this to be especially important if the user is working on the problem without the guidance from a tutor.

An algorithm to verify that the user’s work is correct was used throughout the application. In order to keep the user on the correct track, the algorithm was used to determine whether the user’s current step matched the previous step. If the current step was not equal to the previous step, it would be considered incorrect, and vice versa.

Additionally, when a user submits a solution, an algorithm was used to determine whether the solution was simplified. The term "simplify" has two definitions: (1) "to reduce
the numerator and denominator in a fraction to the smallest numbers possible" , and (2) "to remove brackets, unnecessary terms and numbers" [58]. The algorithm was used to determine whether the solution was simplified in both senses of the term.

Since different users can submit different answers – especially when the solution is in terms of another variable – it was important that the solution was not compared to a hard-coded answer.

For example, both of the following submissions would technically be correct:

\[ x = 2y - 1 \]
\[ x = -1 + 2y \]

Although they do not have the same exact format, the algorithm would determine both of the solutions to be simplified and correct.

**Feedback for the user**

We wanted the user to be provided information and feedback to help them solve the problems that they are encountering. Each user should be encouraged to think critically, not simply given the correct answer upon submission of an incorrect solution. To address this requirement, we decided that if a user writes an incorrect step, an algorithm would determine what the error was. The user would receive a hint, which would either give detailed information about the error or general information which could be applied to any problem in that subject. Additionally, the feedback could include other methods to solving the problem or using the explicit form if applicable.

Fig. 3.2 Use cases
3.2 High-Level Solution

The use cases of the application are displayed in Figure 3.2. In order to use the system, the user must have a valid account and log in to the application. The process of logging in consists of clicking the "Sign In" button and submitting proper credentials. Once logged in, the user can select subject, then select chapter, followed by select section, and finally select problem in order to choose a specific problem to solve. Once a problem has been selected, the user is able to modify the workspace with a variety of features. The user can place an element in the workspace in order to start solving the problem. This can be accomplished by using drag-and-drop elements along with typing the elements directly into the workspace. If the user makes an error, the user can undo a move by clicking the "Undo" button. This will revert the most recent change. After the user has undone a move, they can also redo a move by clicking the "Redo" button to bring back a change that was undone. Finally, the user can clear the workspace by clicking the "Reset Workspace" button, allowing the user to start working on a problem using a blank workspace.

The user can also go to the next step by clicking the "Next Step" button. To alter the list of solution steps, the user can click another "Undo" button to undo a step which removes the most recent solution step. Additionally, the user can click the solution steps’ "Redo" button to redo a step. By clicking the "Next Step" button in the workspace, the user is also able to add the current step to the list of solution steps. The step will only be added if it is equal to the previous step, which will be verified by an algorithm. When the user is satisfied with the solution, the user can submit the solution by clicking the "Submit Solution" button. If the submission is correct, the user can then move to the next problem by clicking the "Next Problem" button which pops up on the screen. Finally, when the user is finished using the application and wants to log out, the user can click the "Sign Out" button.
Chapter 4

Implementation

In this chapter the system that was implemented, MiE, and all the software tools used during the process are described.

4.1 Software Tools

4.1.1 Visual Studio

Visual Studio [51] (VS) is an Integrated Development Environment (IDE) from Microsoft [52]. An IDE is a software application that supplies developers with extensive tools and facilities for program development [1]. Those tools usually include a code editor, build automation support, a debugger and some kind of intelligent code completion component.

VS supports multiple languages, such as C#, Visual Basic, F#, C++, HTML, JavaScript and it can be extended to support virtually any programming language. Some of its powerful features include code refactoring [20], cross-language debugging, universal Windows platform development, cross-platform mobile development, rich web tools support, Git [33] integration, access to thousands of extensions and, last but not least, IntelliSense [38].

Visual Studio Community 2015 is the latest version of VS, as of writing this thesis, and available for free [31]; this is the version used by the authors. VS was chosen partly because of its rich features mentioned above, but also due to the authors’ desire to build the application using ASP.NET, which is described in the next section.

4.1.2 ASP.NET

ASP.NET [10] is a free server-side web application framework for web development using HTML, CSS and JavaScript [12]. There is support for building standard websites, mobile
applications and sites, and also Web APIs and services. ASP.NET 5 is the latest version of ASP.NET and the one used for this project. It is open source and combines ASP.NET MVC [11], Web API and SignalR [49] into one package and is cross platform [13].

ASP.NET was used to implement the server component, or the back-end, of the system. In software engineering, the front end is the presentation layer of a system (the interface that the user interacts with) and the back end is the data access layer [32]. A Web API was developed to provide the application with the data required for its operation, and authentication / authorization for the various users.

**Fig. 4.1** Visual Studio Community 2015: the main IDE used for the development of MiE.

**Fig. 4.2** Description of ASP.NET 5 [15]
4.1.3 AngularJS

AngularJS [5] is an open source, client-side web application framework for Model – View – Controller (MVC) and Model – View – ViewModel (MVVM) architectures [6]. Its main focus is to provide support for developing single-page applications (SPA). An SPA is a web application or web site, in which all essential code for its display and operation is sent from the server only once, in a single page load [50].

From the very beginning of development, MiE was designed by following SPA architecture principles, only contacting the server to dynamically transfer data to and from the database after the initial page load; this provides a much smoother experience for the user, who does not have to wait at all when navigating between different pages of the application.

![Fig. 4.3 The various components of AngularJS[7]](image)

4.1.4 HTML

Hyper Text Markup Language (HTML) is the core markup language for creating web pages and applications [34]. Web browsers, like Google Chrome [19], Mozilla Firefox [30] and
Internet Explorer [37], can decode HTML files and translate them into visual and aural information [36]; they interpret HTML tags, such as `<p>` and `<img>`, as the web page’s contents.

HTML5 is the latest version and it offers new multimedia and graphical elements, namely `<video>`, `<audio>` and `<canvas>` [35], along with various other improvements. It also adds support for Scalable Vector Graphics (SVG) and MathML [42] for mathematical expressions.

### 4.1.5 CSS

Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation and style (e.g. layout, fonts, colors) of a web page [23] - it outlines how elements should be rendered on screen, on paper or on other media. CSS is based on the design philosophy of *separation of presentation and content*, whose main notion is to distinguish between the actual content of a document and the way it is introduced to the users [48].

CSS3 is the newest adaptation of the language and its many new features include rounded corners, gradients, shadows and animations, as well as new layout tools such as flexible box, multi-columns and grid layouts [24].

### 4.1.6 JavaScript

JavaScript is a programming language that is a core component of building web applications and pages, along with HTML and CSS [40]. While HTML and CSS deal with the visual aspect of the application and the user interface however, JavaScript helps with its functionality. It is used both for client-side and server-side applications (e.g. Node.js [47]). Despite the similar name, it is very different to the Java programming language, and the two should not be confused with each other.

### 4.1.7 C#

C# is an object-oriented programming language with a syntax similar to C, C++, and Java [22]. Although the syntax may be similar to other C-style programming languages, there are a quite a few differences such as portability, typing, memory access, and meta-programming [18]. C# is dependent on the Common Language Infrastructure and was developed by Microsoft within the .NET Framework initiative [22].
4.2 Design Choices

4.1.8 Bootstrap

Bootstrap [16] is an open-source front-end framework, whose main goal is to help with the development of web sites and applications by providing many HTML- and CSS-based design templates and tools, such as buttons, forms and other interface elements [17]. One of its greatest features is the ability to scale a website depending on the size of the screen, from mobile phones to desktops.

4.2 Design Choices

MiE was designed to be a web application according to SPA architecture principles. Our intention was for the application to provide a fluid user experience similar to a desktop application; the children should perceive it as an interactive computer game instead of a website. The users only connect to the server once during the initial page load. Any subsequent calls to the server are only for log-in, log-out services and for loading the game data; the application can be used offline, if needed, after the game loads.

4.2.1 User interface features

According to a study from the University of Jyväskylä [62], in which children were able to design interfaces of learning environments, there are a few components which children especially consider to be important when it comes to interfaces. The components were categorized into the following groups: (1) navigation and appearance of the user interface and (2) theme and functionality of the page’s content. Key concepts children emphasized in the study, with regards to appearance, were “content rich layouts, little empty space” and “realistic appearance” [62]. With regards to functionality, the following topics were emphasized: “much freedom of choice in functionality to allow exploration”, the “possibility to create something”, and “a main character” [62].

The focus of this thesis was to create a space for the user to perform pen-and-paper tasks on the computer, so the workspace page became of great importance. The appearance of the user interface and the functionality of the content needed to be designed carefully in order to fulfill the needs of the users. Since the motivation behind developing this learning application was to encourage children to continue practicing math, the findings from the University of Jyväsklä study [62] served as guidelines when we designed the application’s user interface.
Content rich layouts, little empty space

We designed the interface to have features located throughout the screen. At the top, breadcrumbs are used to help the user navigate through the system. Also, the name of the user and the sign-out button are visible, allowing the account-holder to easily be determined and the site to be exited, respectively. The center is blank because it is the workspace, where the user can solve problems. Other than that, the user is able to drag-and-drop from the border of the workspace area to the workspace. Additionally, the Solution Steps are found to the right. This section is to be filled by the user as progress is made. The page is filled with features that can help the user to successfully solve problems.

Realistic appearance

We designed the tool so that the values input into the workspace can be manipulated to look like they would on the user’s paper. The realistic component, in this situation, would be the version which is written on paper. The user can make values in the workspace appear larger, or proportional to their counterparts, as can be seen in Figure 4.4.

![Fig. 4.4 The user can display fractions as they would on paper - proportional to their counterparts.](image)

Additionally, the workspace uses the font called Kalam, which is similar to a typical handwriting font. The appearance of the font could make the workspace seem more personalized, as if the user had written the problem on paper. This appearance could appeal to the user and make the user more comfortable with what has been dragged into or written in the workspace. After the solution has been submitted by the user, the solution is displayed using MathJax [44], a library used to display expressions and equations in a mathematical format in a browser.

Freedom of choice in functionality to allow exploration

The user has the freedom to make the equation appear as realistic as possible, due to the freedom to manipulate the workspace as they desire. There are many operators that the user
4.2 Design Choices

can use to represent their answer, but if they want to make it look like something they would write on paper, then they are able to. However, the program does not force them to represent the problem in a certain way. The user should have the means to be creative when solving these problems, which is definitely the case when it comes to grid-cell manipulation.

**Possibility to create something**

The user is given the flexibility of drag-and-drop and keyboard input, allowing them to format the problem based on the way they conceptualize it. In the workspace, the user creates equations, which are either marked correct or incorrect throughout the solution solving process and in the end have created their personal solution to the problem. Dragging-and-dropping the values and operators into the workspace, or typing them from the keyboard, could give the user the feeling of taking control of the problem and creating a solution as desired.

**A game character**

As the user progresses through solving a problem, there is a possibility that they have an issue with reaching the next correct step. In the case that the user cannot solve the next step, the hint (light-bulb) button can be clicked to get inspiration, and a “teacher-owl” will appear and present the user with an appropriate type of hint. Although the owl does not currently have a large role in the workspace, the character could make the learning application feel more personal.

### 4.2.2 Revisions along the way

The math-learning application required many features in order to allow the user to develop their thoughts and solve problems using the workspace. Many features were considered and designed during the first phase of the project; however, the original and final designs differed very much. Since the Plan Do Check Act improvement model is iterative, it allowed for changes to be made when problems were detected.

**Workspace grid**

The first design revision was the workspace grid (Figure 4.5). The workspace area is the space where the user can drag-and-drop elements into or type values and operators in order to display their solutions and steps. The appearance of the workspace was designed to be similar to a sheet of paper, where the user has the freedom to write and express equations in
a personalized manner. Since the workspace should allow the user to freely make decisions, it was important to consider that although the workspace should not hinder the user from making progress, it should help the user to solve problems in an organized way. To satisfy these conditions, there needed to be a balance between flexibility and organization.

During the initial design, elements could be added to any part of the workspace which allowed the user to have a great amount of flexibility when forming their equations. An unlimited amount of flexibility could be seen as a positive: it would allow the user to form their equations in any way they seemed fit; however, it could also be seen as a negative: potentially leaving the user with a set of numbers, variables, and operators which have little organization. The workspace grids were then set in place, one on either side of the equal sign.

The grids allow the user to add numbers and variables to grid cells. With the addition of a grid, another problem arose: placement of operators. The discussion became whether the user should be able to interact with an operator in the same manner as they interact with values (numbers and variables), or if the operators should be placed on the borders of grid cells. The positive aspect of "snapping" operators to the borders of cells would be to clearly demonstrate the relationship between two values. One negative aspect to snapping would be that it reduces the flexibility of the user. For example, if the user preferred to write: $6 + -3$, they would not be able to elegantly write it in the workspace. The user must then write $6 - 3$ in order to write the solution in an elegant way. The final design allows the user to add the operators to the cells, giving them the freedom to place operators wherever they please. An exception to this decision was the fraction bar, which was handled in a very different manner, to be discussed later in the report.

**Complex fractions**

Complex fractions were one of the greatest challenges faced during the design and development of the math-learning application. A fraction is composed of a numerator and a denominator; a complex fraction is a fraction which has a fraction in either the numerator, denominator, or both. Fractions are a key concept within algebra, so it was considered to be a very important component of the application, and representing complex fractions in an appropriate fashion was the challenge. A series of revisions were made in order to represent complex fractions within the application.

The original design of displaying fractions within the workspace included only non-complex, or “simple” fractions. This was the basis for the design and development of representing fractions since there were many questions to be answered regarding how fractions should be handled. Displaying the simple fractions in the workspace was not an issue,
the user would drag a fraction bar from the toolbox and place it at the bottom border of the numerator. However, it was noticed in the early stages of the project that having a single fraction bar would not be a good solution for the complex fractions to be handled in the near future.

When complex fractions are written on paper, a wide fraction bar is used to represent the complex fraction and a narrow fraction bar is used to represent a simple fraction within the complex fraction. However, when using the original design of the workspace, the user was to drag-and-drop a single fraction bar operator in order to represent the wide and narrow fraction bars. A revision was urgently needed because problems started to arise when parsing different fractions. For example, $\frac{5}{3}$ and $\frac{5}{4}$ would both result in $\frac{3}{4}$ after parsing, due to the order of operations. The revision included using two different operators, a wide and a narrow fraction bar, to distinguish between the complex and simple fractions, respectively.

After distinguishing between the two different fraction operators, it was important to consider the different types of problems that would be solved by the user. The math-learning application was designed with complex problems in mind, including fractions with large numerators and denominators. Considering the large numerators and denominators, it was decided that fractions should be able to span multiple cells, not only vertically, but horizontally as well. This problem revealed other issues because it caused a complete analysis of how fractions would be represented by the user and how the fractions should be parsed based on the user’s representation. The first revision was that of expanding the fraction bar, based only on the width of the numerator. However, this would not allow the user to use multiple cells to represent a numerator or a denominator. Instead, the user would be required to merge cells, which is explained later, in order to create a very wide fraction. Additionally, if only the denominator was very large, the numerator would still be required to be merged with neighboring cells. It was then decided that “width-expansion of the fraction bar by expanding the numerator” was not user-friendly; a relatively simple fraction could then be very complicated to represent within the workspace. This revision was soon replaced.

Following the discovery of the problem regarding horizontal spanning of fractions, a separate grid which only accepted fraction operators was designed. The revision was more user-friendly since the “fraction grid” could be manipulated without affecting the “value grid”. The fraction grid was a better approach since it did not restrict the user to specific guidelines when the user needed to make a fraction span several cells horizontally. The same grid could be used for wide (complex) and narrow (simple) fraction bars, each type increasing in size when the fraction grid cells were expanded. The problem of parsing the grid was also easier to solve because the location and size of each operator were used to determine which cells included numerators and denominators.
Toolbar organization

An important feature of the math-learning application is the set of toolbars which surround the workspace. The initial design of the interface had three main sections: Solution steps, Workspace, and Toolbar- each having one vertical section (Figure 4.5).

![Figure 4.5 Original Design of Interface: one vertical section for each - Solution Steps, Workspace, and Toolbar](image)

Having three distinct sections would allow the user to easily identify the purposes of each. The design was quickly revised due to the realization that the user would be dragging the operators from the toolbox and dropping them in the workspace, meaning that the user would always need to drag each element over the right portion of the workspace in order to drop it in the left portion. The design was not user-friendly since the user would need to drag the element a much longer distance than necessary and the user would have a higher chance of dropping the element in a location where it was not meant to be dropped. The toolbar was then redesigned.

After discovering that it would not be ideal to require the user to drag the operators from the right-hand side of the screen, the toolbar was instead placed in the center, where it would be equidistant to the left and right grids of the workspace (Figure 4.6). Additionally, the values and variables were placed above the operators, since they are also draggable.

Finally, the toolbar was revised a third time, when the operator toolbar was placed below the workspace grids (Figure 4.7). The operator toolbar was placed in this particular position in order for the function of each draggable element to be easily understood. The position of the element will aid the user in determining which direction they should point the cursor. The movement "up-then-down" will be reserved for values (numbers and variables) and the movement "down-then-up" will be reserved for operators. The position could play an important role in how confidently the user may drag and drop elements within the workspace. Additionally, the Solution Steps were shifted to the right side of the screen, since the user
4.2 Design Choices

Fig. 4.6 First Revision of Interface Design: Toolbar placed above workspace, values and variables added to toolbar

may associate the right side of the screen with making progress, or moving on to the next step.

Fig. 4.7 Second Revision of Interface Design: Solution Steps moved to the right and operator toolbar moved below the workspace

Fraction operator images

Another component that was considered for revision during the design and development of the math-learning application was distinguishing between the simple and complex fractions. As mentioned previously, a second grid was introduced to allow for specific placements of fraction bars. Also, two types of fraction bars were available to the user: complex (wide) and simple (narrow).

The specific challenge at hand was how to distinguish between these two types of fraction operators. Since the concept of the workspace is drag-and-drop, the different types of fraction operators would clearly have to be represented on the draggable’s icon. Initially, the
draggable fraction operators had images of division bars; however, the division bars did not actually represent meaningful information to the user. Although the images were distinct from each other, the images did not portray the difference between the two operators. Due to this issue, the operators were not user-friendly and a revision was needed.

After many discussions about the representations of wide and narrow division bars without leaving the user confused or causing the user to select a fraction operator instead of the subtraction operator, a decision was made. The revision included tool tips and new images for each of the fraction bar operators. If the user hovers the cursor over the tool tip, an image representing the function of the operator appears. In the image, the type of fraction that the operator could be used for was highlighted in red.

Additionally, the draggable images were changed to represent a fraction bar with designated areas for the numerator and denominator. The two draggable images differ in the width of the fraction bars - the complex fraction bar is wider than the simple fraction bar. These revisions were made in order to help the user understand the functions and purposes of the operators in an easy way.

**Grid cell expansion**

A final component considered for revision was the expansion of grid cells. In order to appropriately display and represent their solution, the user would need the workspace grids to be flexible. Fractions are usually written using the size of the values to indicate the existing relationships. For example, in \( \frac{1}{3} \), the denominator is not a fraction itself, so the 3 is larger than the numerator, which is a fraction.

Since the sizes of the values can be very important to those solving problems on paper, it was also a key component that needed to be included in the workspace. After considering the problem at hand, the grid was modified to be more dynamic than originally planned: grid cells should expand horizontally and vertically. By expanding a cell in the horizontal or vertical direction, the user can increase the size and center the value horizontally or vertically in the cells, respectively.

The next component to consider regarding the expansion of grid cells was how the user would expand cells. In order to provide a user-friendly, interactive application for the user, it was important to consider which actions would feel natural to the user to expand a cell. As a preliminary design, it was decided that the user could easily expand the cell by right-clicking and selecting the “Expand Cell” option from a menu. Additionally, the user could right-click on an expanded cell and select the “Collapse Cell” option from the menu in order to reduce the size of the cell. The functionality was very successful and the user could easily expand
and decrease the sizes of the cells; however, right-clicking to reveal a menu followed by even more clicking to select an option caused the user to click many times. Having to click on the workspace too many times could be inconvenient for the user, especially since they would never have had to “click” while solving a problem using the traditional pen-and-paper method. Since the goal was to simulate the solving of a problem with pen-and-paper, the action which initiated an expansion or collapse of a cell needed a revision.

A proposition of how a user could expand the cells in a user-friendly manner was by hovering an element over a cell’s edge. When a user hovers an element over either of the two types of grids (fraction and normal) and drags the element over a cell border, the cell would expand. The hovering motion can be used to expand cells in both horizontal and vertical directions; however, the fraction cells can only be expanded in the horizontal direction since there are no vertically neighboring cells. Following an expansion, reverting the cells to their original size would be done automatically by the application when the cells became empty again. Additionally, the fraction grid cells would automatically expand or collapse when the normal grid cells directly above and below expand or collapse, respectively, which would provide a more user-friendly experience for the user. Hovering would reduce the user’s number of clicks, which could result in a more user-friendly application.

### 4.3 Architecture and Implementation

The application can be separated into two main components: the back-end, or server-side, and the front-end, or client-side. Both elements are independent from each other and either can be refactored or re-implemented from scratch using different programming languages / frameworks without compromising the structure and functionality of the application, as long as the interfaces used to communicate between them remain consistent.

#### 4.3.1 Back End

ASP.NET Web API[14] was used as the framework for implementing the server component of the MiE application; it uses C# as its programming language. Since it needs the .NET framework to run, the server requires installation on a computer running Windows as its operating system.

The server acts as an intermediary between the client and the database and all communication between them is accomplished through it. It provides HTTP RESTful services, including registering a new user, signing in and signing out. All the game data is sent to the client when the user successfully signs in. Every supported service can be accessed by a
client via a unique Uniform Resource Identifier (URI) and an HTTP verb, like GET or POST. For example, when the user clicks on the "Register" button, the client will issue an HTTP POST request to the server using a URI such as "api/auth/register".

**Database**

A database stores all the application’s necessary data, such as the users’ log-in credentials and roles. The database also stores the game data of MiE, such as the algebraic problem statements. A widely used and open-source database, MySQL [46], was chosen for the project.

The data are stored in tables, which are mirrored in the form of model classes by the server; each table has its own corresponding class, whose variables emulate the table columns’ names. Entity Framework [29] then uses these classes to map the relational data into domain-specific objects. The following tables were used for implementing the application: Users (table 4.1), Roles (table 4.2), User Roles (table 4.3), Subjects (table 4.4), Chapters (table 4.5), Sections (table 4.6) and Problems (table 4.7).

- The Users table holds all the necessary information regarding the users of the application. Their first and last names are stored for identification purposes, mainly for the teacher to be able to recognize each user. Their emails and passwords, on the other hand, are used to log into the system; passwords are not stored in plain text format, but rather in bytes after being processed by a key derivation function, which protects them from malicious attacks, which is described in the Authorization section.

<table>
<thead>
<tr>
<th>userID</th>
<th>firstName</th>
<th>lastName</th>
<th>email</th>
<th>passwordKey</th>
<th>passwordSalt</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>test</td>
<td>test</td>
<td><a href="mailto:test@test.com">test@test.com</a></td>
<td>BLOB</td>
<td>BLOB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>roleID</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Admin</td>
</tr>
<tr>
<td>2</td>
<td>User</td>
</tr>
</tbody>
</table>
• The Roles table contains the names of the roles available to be assigned to each new user. The current version of MiE has two types of roles, "Admin" and "User".

• The User Roles table holds the assignments of each user to a specific role.

• The Subjects table contains the titles of the math subjects available.

• Similarly, the Chapters table holds the titles of the chapters available. Additionally, it contains some general hints for that particular chapter that are provided to the user via the hint button. There is also a foreign key pointing to corresponding subject.

• The Sections table contains titles of the available sections and a foreign key to a corresponding chapter.

• Finally, the Problems table includes the problem descriptions and solutions, as well as the variables that are part of each problem.

Authentication

Authentication of a user is accomplished by a sign-in system. The user must first create an account by providing an email and a password. Then, they can use those credentials to access the application. When registering for a new account, the credentials are sent via a POST request to the server; after checking to see if the email already exists, a cryptographic hash function, PBKDF2, is applied to the password before storing it in the database.

Password-Based Key Derivation Function 2 (PBKDF2) employs a pseudo-random function, along with a salt value, to the user’s password and repeats the procedure many times to

Table 4.4 The subjects table

<table>
<thead>
<tr>
<th>subjectID</th>
<th>title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Algebra</td>
</tr>
</tbody>
</table>

Table 4.5 The chapters table

<table>
<thead>
<tr>
<th>chapterID</th>
<th>title</th>
<th>subjectID</th>
<th>hints</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Equations</td>
<td>1</td>
<td>{&quot;hints&quot;: [&quot;...&quot;]}</td>
</tr>
</tbody>
</table>

Table 4.6 The sections table

<table>
<thead>
<tr>
<th>sectionID</th>
<th>title</th>
<th>chapterID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>One variable equations</td>
<td>1</td>
</tr>
</tbody>
</table>
Table 4.7 The problems table

<table>
<thead>
<tr>
<th>problemID</th>
<th>description</th>
<th>solution</th>
<th>sectionID</th>
<th>variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Solve for x: ‘4*4x + 2x = 72’</td>
<td>{&quot;x&quot;: &quot;4&quot;}</td>
<td>1</td>
<td>{&quot;var&quot;: [&quot;x&quot;]}</td>
</tr>
<tr>
<td>2</td>
<td>Solve for p: ‘...’</td>
<td>{&quot;p&quot;: &quot;10&quot;}</td>
<td>1</td>
<td>{&quot;var&quot;: [&quot;p&quot;]}</td>
</tr>
</tbody>
</table>

generate a derived key [2]. This key, along with the salt used, are stored in the database and are used to authenticate a potential log-in attempt.

When the user attempts to log into the system, the password provided undergoes the same procedure as during the registration process and the new derived key is checked against the existing key for that specific email address. By following this process, the user’s actual password not need to be stored and is protected against malicious attacks. Cookies are used to keep the user logged in as long as their browser is open or until they click on the log-out button. By using cookies, the user must not resend their credentials for every communication call to the server.

Authorization

Each user is assigned a role when registering for a new account and that information is saved on their browser for the duration of a session. Two roles were created for the application, "User" and "Admin"; however, these roles do not hold much meaning for the current version but were included as features to be added in the future. An idea is to provide an API interface to teachers, where they can add their own mathematical problems, tutorials and hints.

4.3.2 Front End

The main framework used for the client-side was AngularJS. The application shares some common folders, such as images and external libraries and a global CSS file, which is responsible for styling all the views.

Structure

The modules and folders are organized according to their functionality and purpose. Each folder contains a main module file, along with accompanying controllers, factories, and views. The folders are Authentication, Game, Home, and Navigation.

- The Home module is responsible for the entry page of the application. Figure 4.8 shows the welcome screen; the user is given the option of either signing in or registering for a new account.
• The Authentication module takes care of authenticating (Figure 4.9) and registering (Figure 4.10) the users. It communicates with the server, sending the users’ credentials and receiving the responses. If the server’s response to the request is successful, the Authentication module saves the user’s information and game data in cookies and the root scope. Otherwise, if the request is denied, the Authentication module displays error messages. It also ensures that all the appropriate form fields are filled correctly before sending a request to the server.
• As the name suggests, the Navigation module’s purpose is to navigate the user throughout the pages of the application. After logging in, the user must select a desired subject (Figure 4.11), chapter, section, and finally problem, to be taken to the game screen and start solving the exercise. The user’s choices are saved in the root scope, as well as in the session’s cookie; that way, even if the page is reloaded, the selections will be remembered.
Finally, the Game module is responsible for the main functionality of MiE. Since it is the biggest and most important part of the application, it is extensively described in the following section. The main game view can be seen in Figure 4.12.

Fig. 4.11 Choosing a subject

Fig. 4.12 The main game view: The workspace consists of two rectangle tables on either side of an equal sign and the toolbars can be found on the top and bottom of them. The user can drag elements from the toolbars and drop them on either of the two tables, formulating expressions. The two tables represent the two sides of an equation. Clicking on the Next Step button adds the current contents of the workspace as a new step in the Solution Steps area.
The game module

The game module is split into a controller and various factories. The controller regulates all the functionality and calls the different factories when needed. The workspace, as seen in Figure 4.17, consists of two main rectangle tables on either side of an equal sign. The two rectangle tables represent the two sides of an equation. At the top there is the problem description text area and the tutorials button to the left of it. The toolbar elements can be found on the top and bottom of the workspace area, along with other buttons, each providing a different functionality such as undo / redo, helpful hints, showing / hiding the cell borders, resetting the workspace, and moving on to the next step.

Initialization All the variables and game data are initialized and loaded at the beginning of the game’s execution, including icon tooltips, toolbar elements’ options, problem description, tutorials, and more. The dimensions of the workspace and the game screen are adjusted to the user’s resolution. The two tables that are part of the workspace use a set number of rows and columns, but an option to change these values can easily be added in the future for the user to customize their workspace.

Using the workspace There are two ways of operating on the provided workspace: dragging and dropping elements into its cells or typing directly into them. Every component in the toolbar can be dragged and dropped over the tables and their contents will get appended to the appropriate cell. When an element is hovering over a cell, that cell is given a different background color to be distinguishable from the rest - it is highlighted. After an element has been dropped, it can be dragged someplace else again. The second way is to just click on a cell, it will get focused and a cursor will appear inside of it. The cursor indicates that the cell can be typed into directly and the user is able to type in the whole solution without using any drag-and-drop elements.

Fractions We needed to place a limit on the number of levels of fractions over fractions the user could work with, since the workspace should be visible as a whole at all times on the screen. It was decided that one level, a fraction over another fraction, was good enough for high school mathematics. There are two operators for that purpose in MiE, the complex fraction bar and the normal fraction bar. Either can be used when creating a normal fraction, but when the user wants to write a complex one, e.g. $\frac{1}{\frac{2}{3}}$, they must use the different operators to distinguish between the inner and the outer fractions. There were two reasons for this particular design. Firstly, it allowed for a straightforward understanding of what the user meant and parse it correctly. For example, if the user wrote 1/2/3, they could have meant $\frac{1}{\frac{2}{3}}$. 

or $\frac{1}{2}$. Secondly, the use of two types of fraction bars was a way to emulate the pen-and-paper method since, when solving exercises on paper and writing complex fractions, it is common to use a bigger and a smaller division bar as well.

**Square roots** Square roots can be used by dragging the appropriate icon from the toolbar inside the workspace. It uses the HTML tag "\$\sqrt{\text{<span style='text-decoration:overline'>}</span>" to display the symbol and place an overhead line over the text underneath it. As of the time of the thesis, this was the only way to display square roots in HTML without using a third party library; MathJax, which was used extensively throughout the application, could not be used for the workspace because of its method of typesetting expressions.

**Exponents** For exponents, the "\$<sup></sup>" tag is used, which places any text inside it as a superscript to its neighboring text. Adding more of the same tags inside each other offers multiple levels of exponents. The user can type the desired exponent into the suitable field in the toolbar and then drag and drop it over the workspace. It will be appended as superscript on the existing contents of the cell; if the cell is empty, on the other hand, a base of 1 is used.

**Logarithms** Logarithms work similarly to the exponents, using the "\$<sub></sub>" tag to place the base of the logarithm in subscript. Including more tags provides the ability to have multiple levels of logarithms. The user can edit the appropriate input field with the desired base and after dropping it into a cell, they can next drop, or type, the argument.

**Cell expansion** An important feature of the application is the ability to expand the workspace cells freely, both vertically and horizontally. This way, the user can change and adjust the workspace to their preference. It can be seen in action in Figure 4.13.
We added listeners to the dragging function of the toolbar elements: the position of a dragged element is constantly calculated and compared to the position of the workspace cells. If an element hovers over the edge between two cells for more than one second, the left cell adds the right cell’s width to its own and the right cell is hidden from view. Similarly, for the vertical expansion, the top cell adds the bottom cell’s height to its own and the bottom cell is hidden. The text inside the cells also merges together. The operator cells used for the division bars can be expanded and merged together in the same manner; however, only in the horizontal direction (Figure 4.14).
4.3 Architecture and Implementation

Fig. 4.14 Operator cell expansion: Two cells can be merged together in order to form fraction bars which span multiple cells. The user may find the expansion to be useful for fractions with large numerators or denominators.

The expansion is not limited to just two cells merging, the user is able to merge the whole workspace table into one large cell, if desired. If at any point an expanded cell has no elements in it, it will revert back to its default state automatically, freeing the user from the responsibility of doing it themselves. The reset workspace button also reverts all cells to their default state.

**Parsing the workspace** Every time the user clicks on the Next Step button, the contents of the workspace tables get parsed into a string, which then gets checked against the original problem equation to see if the step is correct. Parsing is done by taking the text of each cell from both tables and translating it into plain text math and AsciiMath [9] format. Combining the result of each cell produces one string for each table and then those strings get concatenated together, with an equal sign in the middle, to generate the final output.

The problem stems from the fact that it’s not possible to simply read the table rows cell by cell, because fractions have to be taken into account as well. For that reason, the cells above and below a fraction operator get grouped together with parentheses to correctly parse them the way the user meant, before moving on to the next cell or group of cells.

**Hints** When the user is puzzled by a problem, hints are provided by the application. Clicking on the light bulb button will show the hint, along with a mascot, a "teacher - owl" chosen for this purpose. There are two kinds of hints available, generalized and specific;
when the user clicks on the button, the application cycles through the generalized hints first and then offers the specific hints if the user is stuck on a step.

1. Generalized hints are specialized for an entire domain or chapter, e.g. "Equations". They are static and saved in the database for each chapter. Some examples include "Whatever you do to an equation, do the same thing to both sides of that equation!", "Don’t forget to isolate the variable on one side of the equation!" and "Don’t forget to follow the order of operations!" (Figure 4.15).

![Fig. 4.15 A generalized hint being displayed, these hints are pre-defined.](image)

2. Specific hints are, as the name suggests, explicit to the step or problem the user is stuck on. They are dynamic, meaning that the application will try and suggest that the user focuses their attention on a part of the equation, based on their previous step. To accomplish that, the application uses regular expressions to parse each solution step and try and deduce whether a section of it can be simplified further. An example can be found at Figure 4.16.
Checking steps and final submissions Checking to see whether solution steps and final submissions are correct is achieved using a third party library, KAS[41]. The parsed workspace equation-string is given as input and converted into an expression that KAS recognizes, and then compared to the initial problem description. The result is a boolean value and, depending if it is true or false, the step gets appended to the solution steps area or an error message is shown. In the case of a final submission getting checked, it undergoes an extra analysis process, to determine whether it is as simplified as it can be.

Providing specific hints When the user clicks the Next Step button, an algorithm detects whether the submission is correct. If the submission is incorrect, another algorithm provides the user with specific hints to help solving the problem. The algorithm uses regular expressions on the most recent correct step and follows the order of operations (PEMDAS) to determine the first operation to be performed next in the order of operations; this operation is returned to the user as a specific hint.

Resizing the views Every view of the application is adjusted according to the user’s screen size and resolution. The parent container to all elements of the game uses the CSS units \( \text{vh} \) and \( \text{vw} \) to define its height and width accordingly. These units are percentages of the user’s view-port, so the container, along with all its children elements, is automatically resized to fit the current window.
Elements with fixed dimensions, such as buttons and text font sizes, are also resized depending on the resolution; `@media` rules [25] are used in the main CSS file to classify various screen sizes and resolutions into five categories: extra small (mobile phones), small (tablets), medium (laptops), large (desktop computers) and extra large (higher resolutions). For each category, icons, buttons, and font sizes are assigned different values in order for all components to fit together elegantly.

**Undo and Redo** A necessary feature for almost every kind of user interface is the ability to undo and redo actions. In MiE, this is managed by implementing an undo stack and pushing every action the user makes onto it. If user clicks the undo button, the latest action that has been made is popped from the stack and the elements affected are restored to their previous state. When that happens, the action in question is also pushed onto the redo stack; that way, the user can redo an undone activity in the same way.

**Other functionality** There are various other features present in the application:

- The font size of the text in a cell is adjusted to that particular cell’s size. When the user types into a grid cell, as more content is added, the font size will shrink allowing for additional content to be added to the cell. If the user then expands the cell, the text will get bigger to adjust for the larger cell size.

- The division bars also get adjusted based on their parent cell’s width, while at the same time maintaining the ratio of the complex fraction bar as double the size of the simple fraction bar.

- When an element is dropped into a cell of the workspace, it is snapped to the middle of the cell.

- There is a "borders" button on the top right of the workspace that shows and hides the cell borders (Figure 4.17).

- The borders also become visible while an element is being dragged around.
Fig. 4.17 Clicking the borders option in the top right corner of the workspace reveals the cells’ borders.
Chapter 5

Results

5.1 Expert Opinion

We consulted a professor with significant experience in researching new ways of teaching mathematics about the math-learning application that we designed and developed.

The expert noticed that we had a "hint" button and he began to describe the didactic contract, the guidance method that currently exists in classrooms. The didactic contract consists of a couple of simple steps: first, the student raises a hand to indicate to the teacher that some help is needed with a math problem. Next, the teacher gives them a hint in order to solve the problem. These two steps are repeated until the student has solved the problem. Although this process leads the student to the correct solution, the teacher could have provided so many hints that the student did not actually do any of the critical thinking required to solve the problem. Getting assistance whenever the student is hesitant results in the student being heavily dependent on the teacher’s guidance; if the teacher is not available to guide the student through the next step, the student will start to guess. The expert suggested that a math-learning application which has hints that give too much help could reinforce the didactic contract and have a negative effect on the learning process.

The expert suggested that our application could be improved by expanding the "hint" concept; the expansion could make our application differ from the didactic contract by developing different types of hints which could encourage the student to think in new ways. The expert suggested the following hints to present to the user.

- Tell the user to "articulate what you’re thinking, perhaps explain to a neighbor”.

- Tell the user to "try an easier, similar problem to develop your problem solving skills regarding this topic".
• Instruct the user to "draw a figure or diagram representing the problem".

• Display a graph or diagram to help the student visualize the problem.

• Challenge the user: "check your submission to verify that it is correct!".

• Ask the user to reason and justify the answer, i.e. before submission, ask: "are you sure it is correct?".

• Provide or redirect the user to information about the current section or chapter’s content.

The expert made the suggestions for different perspectives of hints because they could potentially affect the learning process. He expressed the importance of the student using the application as a tool to solve problems, but actually discovering and finding the logic behind problem solving by themselves. Additionally, the expert described that the emphasis should not be on how to solve the problems, instead understanding the problems should be emphasized, including: the importance of learning the relationship between variables, understanding the meaning of the equal sign, and realizing the relationship between the two sides of the equation.

5.2 Usability Evaluation

The usability evaluation was performed after the implementation of the application was finished. The goal of the evaluation was to test how well the interface was designed from a usability perspective and extract improvements to be made in the future, as well as identify components that made the application similar to the pen-and-paper method of solving problems.

Although the target age group for the evaluators would ideally be between 13-15 (so they would already have a good grasp of algebra), due to lack of resources, the users chosen were of ages between 8-12 and 23-29 years old. However, since the goal was to answer the research question and to assess the usability of the interface, the results of these two age groups are still very useful. The difference in age between the children chosen and the ideal group is very small, and they could offer valuable feedback on whether they found the interface engaging and enjoyable to use. Additionally, the chosen children’s ages fall in the range of the thesis’ scope, which is 8-15 years old. The adults’ input and ideas on improvements and indicating which elements make the application’s process akin to pen-and-paper could also be very beneficial to us.
The older users were Uppsala University students, who were able to solve equations with confidence. They were told to solve three different types of algebra problems to urge them to use most, if not all, of the features and functionalities of the application. The younger users were in grades 2 and 5, meaning that they had not yet studied algebra; instead, they were presented with three basic arithmetic problems.

Three children and six adults were part of the evaluation. Of the children, one was an 8-year-old girl and two were 12-year-old boys. Of the adults, there were four men and two women, aged 23 to 29 years old. The users were divided into two groups; users in group 1 were given a small tutorial on using the workspace, while users in group 2 were only informed of two basic functionalities: cell expansion and the ability to type into the cells. We wanted to identify which elements and functionalities were intuitive to use without any prior explanation.

The evaluators were asked to verbalize their thoughts throughout the sessions and to comment on features of the applications that they liked or disliked, as well as if there was something that did not function the way they were expecting it to. At the end of each session, the evaluators were asked whether they noticed the elements or buttons that they had not used and whether they could identify their functions. Each evaluator was also asked general questions regarding their overall experience with the interface and if they could identify components that made it look similar to the pen-and-paper method that they are used to. The following observations, grouped into positive / neutral and negative categories, resulted from the evaluations:

### 5.2.1 8-year-old girl - Group 1

- **Positive / Neutral:**
  - She needed guidance throughout the problem solving process due to the nature of the learning application - she had not learned algebra yet, so simple problems were provided to her.
  - The user enjoyed dragging and dropping elements on the page.
  - She did not type during the evaluation.
  - She seemed very curious about the application.
  - The user expressed that using the interface was “fun.”

- **Negative:**
  - She did not notice the undo or redo buttons.
– She had a few problems with expansion of cells. For example, she accidentally expanded a cell and did not know how to make it revert to its original size.

5.2.2 12-year-old boy #1 - Group 1

• Positive / Neutral:

– The user had not learned algebra yet so he solved three arithmetic problems.

– He was very eager to solve the problems and he wanted to solve the whole problem at once.

– The user took advantage of the formatting by moving elements in cells in order to improve the appearance.

– The user took advantage of the flexibility of the workspace by moving and adding elements to any part of the workspace.

– He liked the drag-and-drop elements, he did not type anything directly into the cells.

• Negative:

– He struggled to solve some steps in the beginning because he did not understand how to add the operators. He quickly learned how to use the operators and values.

– He did not notice the hint button.

5.2.3 12-year-old boy #2 - Group 1

• Positive / Neutral:

– The user had no experience with algebra, so he was given simple math problems instead. The problems did not involve the fraction bars, so the two types of fraction bars were not discussed.

– He was very eager to solve the problems and did it in a series of steps.

– The user liked the drag-and-drop elements, he did not type anything directly into the cells.

– He incorrectly added some values when moving to the next step; after some advice, he realized that he should copy the previous line in order to ensure that all values were being included in the current step.
5.2 Usability Evaluation

– He used the borders option, he thought it was useful to see where he could drop the elements.

– Although there seemed to be some issues in the beginning, the last two problems were solved with ease.

– He thought using the application was “fun”.

• Negative:

  – He suggested that the hint button would be more noticeable if it had text associated with it, not just an image.

5.2.4 23-year-old woman - Group 2

• Positive / Neutral:

  – The user liked typing the operands (numbers and variables).
  – She liked dragging and dropping the operators.
  – She really liked the display of the solution steps and suggested that the most recent step could be displayed in the center of the screen so that she could easily focus on it.
  – Sometimes she took advantage of the formatting options that the workspace grid had to offer.
  – Sometimes she placed the elements in an unorganized way, taking advantage of the flexibility of the grid.
  – She noticed the undo button and clicked it to recover a term that was accidentally removed.
  – She was able to navigate throughout the application with ease, the buttons with text were easy to understand.

• Negative:

  – She started solving problem too close to the equal sign and ended up having very little space between the values and the equation sign.
  – The user dragged a minus sign instead of a fraction bar. The three operators are very similar and she expressed her confusion regarding the distinction between them.
Results

– She did not look at the operator tooltips.
– The user did not notice the fraction bars and instead attempted to use an inline division instead of a fraction bar.
– The user did not distinguish between the two types of fraction bars.
– She did not like having to expand the grid cells manually, instead she would like the cells to merge automatically.

• She identified the following components as elements that make the application similar to pen-and-paper:
  1. The solution steps area reminded her of the format used to organize her solution on paper.

5.2.5 25-year-old woman - Group 2

• Positive / Neutral:
  – With the exception of operators which required dragging, such as the fraction bars, she mainly typed during the evaluation.
  – She used a scratch sheet of paper to solve simple mathematical problems.
  – The user did not distinguish between the two fraction bars; however, the problems presented to the user did not require distinguishing between the two.
  – She noticed the undo and redo buttons but she did not need to use them to solve the problems.

• Negative:
  – She thought there was too much clicking, dragging, and dropping.
  – She did not notice the hint button while solving the problem.
  – She did not notice the borders button.
  – She did not like having to look to the right of the screen to see the previous step, she would have preferred to see it near the center of the workspace.
  – She stated that she preferred writing over dragging, dropping, clicking, and typing.
  – The user tried expanding a cell once, and expressed that it took too long to do it, she thought the function could be faulty.
5.2 Usability Evaluation

– She started adding elements too close to the equal sign and needed to move the elements to the left of the workspace in order to have space the whole problem.
– She did not notice the tooltips.
– The user accidentally pressed the "backspace" key on the keyboard, resulting in the application navigating to the previous page and losing the progress of her solution solving process.
– The user expressed that she would prefer to have the equation elements within the workspace remain after moving on to the next step in the solution.

• She identified the following components as elements that make the application similar to pen-and-paper:
  1. Using the two grids of the workspace reminded her of solving the problem on paper, on the two sides of the equal sign.
  2. The solution steps area reminded her of the format used to organize her solution on paper.
  3. She thought the font that was used in the workspace looked similar to hand-writing.

5.2.6 26-year-old man - Group 2

• Positive / Neutral:
  – He loved that the application could parse "*", "•" and two consecutive elements (e.g. 2x) as multiplication operators.
  – He noticed the hint button, correctly deduced what its functionality was, but did not use it throughout the session.
  – The user realized what the undo and redo buttons were on his own, and used the undo button once when needed.
  – He like the solution steps area.
  – Overall, he liked the application and thought it was a very nice concept.

• Negative:
  – The user pressed the "back" button on the mouse accidentally, resulting in the application navigating to the previous page and him losing his progress through a problem. He expressed his displeasure having to redo all the steps.
– The user could not deduce how to delete an element from the workspace. He answered "Oh, that makes sense" upon being informed of how to do it.

– He did not notice the borders button at all.

– He had some trouble expanding a cell initially, the functionality would not perform as expected.

– He preferred having smaller text inside the cells than having to merge them to provide more space.

– He had trouble understanding the difference between the two division bar icons, even after looking at the tooltips.

• He identified the following components as elements that make the application similar to pen-and-paper:


5.2.7 28-year-old man #1 - Group 2

• Positive / Neutral:

  – He liked the solution steps area and being able to see his solution history like he would on paper.
  
  – The user preferred typing into the cells instead of dragging and dropping elements.
  
  – In general, he was impressed with the application and liked it.

• Negative:

  – The user tried to write all steps into one instance of the workspace, he did not realize that the workspace represents one step of the solving process each time.
  
  – He disliked the fact that the equal sign was always in the middle of the workspace.
  
  – He did not realize what the exponent and logarithm fields until he looked at the tooltips.
  
  – After being informed of the logarithm field’s functionality, he expected to be able to type the logarithm’s base into the field and drag and drop it over an existing element on the workspace.
  
  – The user did not understand what the fraction elements did, he mistook their tooltips for examples.
5.2 Usability Evaluation

- The user did not understand that it was possible to add more than one element into a cell.
- He did not like that the operators are below the workspace, far from the numbers.
- He expected pressing Enter on his keyboard to take him to the next step.
- He did not notice the undo / redo buttons, neither on the workspace area nor on the solution steps area. He did realize their functionality, however, when he was asked if he could guess what it was.

- He identified the following components as elements that make the application similar to pen-and-paper:

  1. The solution steps area is exactly like solving an equation on paper: step by step.
  2. The font type of the elements dropped in the workspace makes it look like handwriting.
  3. The overall "feel" of solving a problem in the application is very similar to the way he does it on paper. He said "it’s exactly like I would do it on paper, only instead of writing I used my keyboard and mouse."

5.2.8 28-year-old man #2 - Group 2

- Positive / Neutral:
  - He used the inline division operator (/) instead of a fraction, he said that it was faster.
  - He preferred typing into the cells instead of dragging and dropping elements.
  - He experimented with the various buttons and elements, and identified most of them, including the hint button, the borders button, the fraction bars and the logarithm and exponent elements.
  - He expressed his fondness for the application.

- Negative:
  - The user did not realize that he could add more than one element into a cell.
  - He expected the left parenthesis element to be prepended to the existing element when dropped on top of it in the workspace.
– The user did not notice the undo / redo buttons, neither on the workspace area nor on the solution steps area. He correctly guessed their functionality when asked about it.

– The user expected to be able to type the logarithm’s base into the field and drag and drop it over an existing element on the workspace.

• He identified the following components as elements that make the application similar to pen-and-paper:

1. The solution steps area reminded him of how he solves an equation on paper step by step.
2. The font type of the elements dropped in the workspace makes it look like handwriting.
3. The big size of the workspace looks similar to the open space and flexibility of an A4 paper.

5.2.9 29-year-old man - Group 1

• Positive / Neutral:

– The user used the cell expansion functionality extensively, both horizontally and vertically.

– He spent time making the equations look nice on the workspace, e.g. aligning numbers and fractions on the same horizontal level, by expanding the appropriate cells.

– The user toggled the borders option on, he liked seeing the cells at all times.

– He frequently used the fraction elements.

– He liked being able to shape the workspace however he wanted and spent a large amount of time doing so.

– The user liked the solution steps area.

– The user experimented with and used all the elements of the workspace.

– He suggested an optional hint functionality where the user is informed whether the step he submitted is the optimal one.

– Overall, he loved the application.

• Negative:
– He had some trouble with expanding the cells sometimes.

• He identified the following components as elements that make the application similar to pen-and-paper:

1. The solution steps area reminded him of how he solves an equation on paper step by step.
2. The fact that you form each part of an equation separately, on either side of the equal sign.
3. The problem description area being at the top, which is similar to how you write the description on paper and then start solving the problem below.

5.3 Requirements

The requirements were fulfilled by various components present in the application.

• User-friendly interface for children: Evaluators expressed their fondness for the application; the evaluators were also able to identify the functions of icons and buttons that were present in the application, indicating that the were easy to understand.

• Provide the users with a fluid experience: We were able to fulfill this requirement by creating an SPA, which acts more like a desktop application rather than a website.

• Give a personalized experience to each user: Each user can register and have an individual account. This requirement has only been fulfilled in part, since the personalized experience was intended to include problems which are designated for users who are studying certain subjects.

• Permanently store the available problems to be solved: We fulfilled this requirement by using a MySQL database to store the application’s content.

• Provide a space where the users can freely formulate expressions and shape them however they want: We attempted to fulfill this requirement by creating the workspace; however, some evaluators perceived the workspace to be more rigid than they would have liked. Perhaps this requirement was not completely fulfilled since improvements could be made.

• Present an aide to guide the users’ solution solving process: This requirement was fulfilled via the hint button. Following the expert’s advice, the hint button presents
general and specific hints to help the user when they cannot solve certain steps of problems.
Chapter 6

Analysis

6.1 Analysis of the expert’s opinion

The expert mainly focused on the information presented upon clicking the hint button, which is the gateway to guidance during the problem solving process. The expert suggested that different types of hints could be developed in order to stimulate the user’s thinking process. The suggestion regarding hints had an important role in the concept behind the hint button; prior to meeting the expert, the hints presented to the student were very detailed and indicated what the user could submit as the next step.

The expert’s suggestion allowed us to make changes regarding the hint button, which now includes specific and general hints. The specific hints were altered after meeting with the expert; the current specific hints guide the user to a portion of the expression which could be solved next in the problem solving process without explicitly stating the next step. The general hints currently being displayed in the application are quite limited; however, this feature could be improved in the future, where more advanced and stimulating questions could be presented to the user in times of need.

6.2 Analysis of the usability evaluation

Analyzing the observations from the evaluations led to identifying issues with the current version of the interface, validating some design choices, as well as fulfilling the main goal of this thesis: answering our research question.
6.2.1 Issues with the current interface

The issues presented here were mostly observed from the evaluations of the users in group 2, who were not given any explanation on how to use the workspace before starting the evaluation session.

Hint button

The hint button was not noticed by some of the evaluators, indicating that it might not be visible or clear enough to the user. One evaluator suggested some text to be associated with it, to make it more noticeable.

Borders button

The borders button was also not noticed by some evaluators. It is not a necessary function of the application, only an option, but it should still be redesigned to make it more apparent. Another solution might be to add an options menu to the application and include the borders option there.

Rigid workspace

Two evaluators started solving a problem too close to the equal sign and then did not have enough space between the elements and the equal sign. They both had to manually drag the elements away from the center to provide more space. A possible solution could be to provide a button that left-justifies the contents of the workspace, or even do it automatically.

Another evaluator disliked the fact that the equal sign was statically placed in the middle of the workspace. This was actually an idea that came up during the design phase of the application: to automatically shift the equal sign on either horizontal direction depending on the number of elements present on each side. However, it was never implemented due to lack of time. Another idea would be to provide the users with the option of manually moving the equal sign themselves.

Fraction icons

There were various issues observed regarding the fraction elements. Even though they are grouped together, a bit further away from the other operators, one evaluator accidentally selected the minus operator instead of a fraction element, suggesting that they have similar icons. Additionally, even after looking at the tooltips, some evaluators could not distinguish
the difference between the two fraction elements and what they represent. Even worse, one evaluator did not even notice the elements.

Fortunately, after explaining the functions of the fraction bar icons and their differences to the evaluators, there were no problems with the way they were used afterwards. This suggests that adding a tutorial on how to use the workspace might be enough to solve this issue. Also, a redesign of the fraction bar icons and tooltips could be very helpful.

Operator tooltips

Many evaluators did not notice the tooltips of some operators. This is troublesome because the tooltips are supposed to explain the function of some of the more complex operators. A solution could be to make them bigger, change their color, or position to make them more noticeable.

Position of previous step

One evaluator expressed their displeasure at having to look to the right of the screen, to see the previous step in the solution steps area. When solving an equation on paper, it is common to have the previous steps right above the current one. Placing the latest step above the workspace might resolve this issue.

Cell expansion

Some of the evaluators had trouble expanding the cells correctly. This was because they moved the element from one cell to the other and back in an attempt to place it right on top of the dividing border between them; the timeout listener that checks whether an element is hovering over the edge for 1 second is reset every time the element switches cells. The cell expansion functionality was implemented in this manner; for the issue to be fixed, the functionality would need to be redesigned and implemented in a different way.

Not saving current progress

Two evaluators accidentally pressed either the back button on the mouse or the backspace key on the keyboard and lost their current progress through a problem because the application navigated to the previous page. The current version of the application does not save the user’s progress after every step but the feature would be useful in case the user refreshes the page. The issue can be fixed using cookies on the user’s browser, which is the same way we temporarily store the user’s log-in information.
Deleting an element

Some evaluators could not deduce on their own how to delete an element from the workspace and resorted to clicking on the cell and pressing the delete button on their keyboard. The current design, dragging the element and dropping it outside of the workspace, is evidently not intuitive enough. An easy fix would be to add a tutorial on how to use the workspace.

Multiple lines in the workspace

One of the evaluators initially tried to write all steps of solving the equation into one instance of the workspace, in one step. He thought that the multiple lines available to write into were for that purpose. This is another issue that can be fixed by including a tutorial on using the workspace.

Exponents and logarithms

The way the exponent and logarithm elements work was not apparent to some evaluators, even after they looked at the elements’ tooltips. This is an indication that their appearance and tooltips need to be redesigned.

Adding an element before an existing one

Two of the evaluators expected the logarithm element to work differently. Currently, it works by typing the required base into the logarithm field and dropping it in the workspace. Afterwards, the argument is dropped in the cell that the logarithm is in. The aforementioned evaluators, however, tried to first drop the argument into a cell and drag and drop the base afterwards. The same expectation came from another evaluator about the left parenthesis element; he thought that by dropping a left parenthesis on an existing element, it would place it before that element.

This was a design choice that we made, to always append the new element to the end of the existing elements. It was designed that way because otherwise the user would not be able to add a logarithm in the same cell after other elements for example, or add a left parenthesis after a number.

Adding multiple elements in a cell

It was not intuitive to some evaluators that it is possible to add more than one element in a cell. This can be fixed by adding a tutorial on how to use the workspace.
Position of the bottom toolbar

One evaluator disliked the position of the bottom toolbar (containing the operators); they thought it was too far away from the top toolbar, which contained the numbers. This was a design choice aiming to distinguish the two toolbars, in an effort to not confuse the users.

Pressing Enter on the keyboard

Another evaluator was almost exclusively typing their solutions instead of dragging and dropping, and tried to go to the next step by pressing the Enter button on their keyboard instead of clicking on the Next Step button. This suggestion will be taken into consideration for future improvements.

Undo / Redo

Most of the evaluators did not notice the undo / redo buttons. All of them, however, correctly identified their functionality when the buttons were pointed out to them. The buttons’ icons might need to be redesigned, otherwise a tutorial on using the workspace should suffice to solve this issue.

Clearing the workspace after each step

One evaluator disliked that the workspace automatically cleared after each step. The workspace is cleared after each step in order to emulate the pen-and-paper method: the user starts with a blank line on the paper after each step. However, it would be simple to provide the user the option of clearing the workspace or keeping the elements of the workspace after each step.

Automatically merging cells

One of the evaluators did not like having to expand the cells manually and would prefer if it was done automatically when the text in a cell gets too small. The automatic expansion of cells was considered during the design phase, but in the end we decided against it. An issue associated with the feature was that it would be difficult to determine the contents of the next cell and whether or not the user would like them to automatically be merged with the current cell.
6.2.2 Validation of design choices

An enjoyable experience

Many evaluators described that they enjoyed using the application. Most evaluators thought that the experience was fun and the application was impressive.

Drag-and-drop functionality

Many evaluators described that they enjoyed the drag-and-drop functionality of the elements.

Typing into workspace

The adult evaluators had the tendency to type into the workspace more often than they used the drag-and-drop functionality. This could be due to their prior experience with computers, most of it being typing on a keyboard. Some evaluators typed since it was faster than dragging and dropping elements.

Element formatting

A few evaluators took advantage of the formatting options available within the workspace. The same evaluators also placed elements randomly throughout the grid at times since they understood the manner in which the elements would be parsed by the application.

Solution steps

Many evaluators expressed that they really enjoyed the way the solution steps were displayed. These evaluators explained that they were fond of this style because it reminded them of the pen-and-paper method of solving problems.

Borders option

One evaluator used the borders option to display the borders at all times. The evaluator expressed that it was easier for him to place elements within the grid when he knew exactly where the elements could be placed.

Buttons with text

Buttons to reset workspace, move to the next step, and submit solution were very clear to evaluators, indicating that the text chosen for these buttons conveyed the appropriate information needed to perform tasks.
Choosing operators

One evaluator expressed that it was very useful that the workspace was flexible in regards to using different operators. For example, the user can drag "•" or type "*" into a grid cell and both would indicate multiplication. Additionally, one evaluator used inline division to represent a fraction since he thought it was faster to type the division sign than to drag a fraction bar.

Icon representation

Observations from the evaluations reveal that most icons within the application were easily interpreted by the evaluators, who could identify each icon’s functionality. This indicates that the designs of most icons are satisfactory.

6.2.3 Answering the research question

In order to answer our research question after building MiE, we specifically asked the adult evaluators if they could identify components of the interface that make it similar to their typical, pen-and-paper method of solving exercises. Following are the main components that were identified.

Solution steps area

All of the evaluators mentioned that the solution steps area reminded them of the way they solve an exercise, step by step, on paper. They said that the structure was the same too: each consecutive step underneath the previous one.

Handwriting font

Most of the evaluators noticed that the font used in the workspace area is akin to handwriting on paper.

Two main grids of the workspace

Two evaluators said that having two separate areas on the workspace, one for each side of the equal sign, is very similar to how an equation is formed on paper - two sides of the equation joined by an equal sign.
Large workspace area

One evaluator added that the big open space of the workspace area reminds him of the size and flexibility of an A4 paper; he could write and form his expressions wherever he wanted.

Problem description on top

One of the evaluators also said that the position of the problem description area reminded him of how, on paper, you write the description on top and continue solving the problem below it.
Chapter 7

Discussion

In this chapter we discuss our results and findings and to what extent our research question has been satisfactorily addressed.

7.1 Discussion of results

Research Question: "Which interface components contribute to simulating the pen-and-paper method of solving mathematical problems in a computer-enhanced learning environment?"

After the evaluations, the users identified interface components which reminded them of using the pen-and-paper method of solving problems. The resulting components were: the solution steps area, the handwriting font, the two main grids representing the two sides of an equation, the large workspace area, and the problem description being right above the workspace. The responses indicate that the identified interface components present within the interface contribute to simulating the pen-and-paper method on the screen. We cannot state that the aforementioned list of components encompasses all interface components that play an important role in making an application which simulates the pen-and-paper method; it can be stated, however, that the results of the evaluations partly answer the research question.

Our findings show that, despite the interface issues of the current version of the application, it was well received among the evaluators. All of them expressed their fondness for it, especially the children, indicating that it might pique the curiosity and interest of other children as well and motivate them to practice mathematics more. Further, more extensive studies can be done with a future version of the application to determine its effect on the students’ learning growth and whether it can be an effective learning tool inside the classroom or for homework.
7.2 Relation to previous work

Although there is various other research done on the subject of educational games and environments, examples being the aforementioned "Monkey’s Revenge" [45], "Aplusix" [8], and "ALEKS" [4], their goals and subsequent results were different from this thesis. The other educational environments mostly focused on the affected learning growth of the students after using the environments, whereas our focus was on identifying interface components that make the environment similar to the pen-and-paper method. Each aforementioned environment had interface components that were similar and different to the interface of MiE.

Monkey’s Revenge (Figure 1.1) does not provide a workspace for the children to solve the exercises. Instead it presents multiple choice problems and the students are expected to solve them in their head or using a different environment, like their notebooks. Similar to MiE, Monkey’s Revenge provides a hint functionality; however, the hint functionality differs from MiE, since Monkey’s Revenge provides very specific hints to the students. Make it Equal+, on the other hand, tries to show generalized hints or specific hints that do not give away the answer to the student, so they still have to solve the issue by themselves.

Aplusix (Figure 1.2) has a workspace area where the users type the solution to equation problems in a step-by-step manner, like they would on paper. This is similar to our workspace, except that Aplusix keeps all the steps in the workspace itself. The functionality of Aplusix is very similar to our application, but the interface and layout have many differences in terms of colors, buttons and complexity.

The interface of ALEKS (Figure 1.3) contains a problem description in the center. Below the description is the user’s solution area, where the student can input the final solution to the problem by typing or selecting functionalities from a menu. This differs very much from Make it Equal+, since MiE allows the student to submit not only solutions but also the intermediate solution steps. Similarly to MiE, ALEKS’s interface includes Clear, Undo, and Help buttons to the right of the solution area.
Chapter 8

Conclusion

We designed and implemented Make it Equal+, an interactive math-learning environment for secondary education. Through usability evaluations we identified interface components that contribute to simulating the pen-and-paper method of solving mathematical problems in a computer-enhanced learning environment, answering our research question. The evaluations show that the application can be well received by users and might help them practice mathematics.

8.1 Future Research

As a result of the evaluations, many issues regarding the application’s interface and functionality were specified. The feedback obtained from those evaluations can be used to improve the application. Additionally, interesting topics that were outside of the thesis’s scope can be developed in the future. In order to teach the user new concepts, a tutorial could be available. The tutorial could include videos, lectures, notes, and interactive activities. Also, the current application was designed for users aged 8 to 15 years old who aim to solve algebraic problems; in the future, more features could be added to the workspace in order for users in a wider age group to solve problems in more subjects, such as geometry, trigonometry, and calculus.

After improving the interface and implementing a more complete version of the application, it could be used to research whether simulating the pen-and-paper method in a computer-enhanced learning environment positively affects the students’ learning growth. An extensive study would have to be conducted in various classrooms to produce definitive results.
References


References


References


