A System That Assists Educators On Creating Personalised Assessments

Andrei Loghin

A dissertation submitted in partial fulfilment of the requirements for the degree of
Bachelor of Science
of the
University of Aberdeen.

Department of Computing Science

2021
Declaration

No portion of the work contained in this document has been submitted in support of an application for a degree or qualification of this or any other university or other institution of learning. All verbatim extracts have been distinguished by quotation marks, and all sources of information have been specifically acknowledged.

Signed:

Date: 2021
Abstract

Given the effect of the COVID-19 pandemic the use of online assessments has grown to an unprecedented level. Many of the available assessment platform lack functionality or have it implemented in way that makes it hard to use.

This project aims to develop an application that aids educators in the creation of online assessments and offers functionality that suits their needs. This will be achieved by understanding what exactly needs improvement regarding educators’ current experience by conducting an initial survey, developing an application that addresses the key shortcomings of current online assessment platforms, evaluating the prototype by introducing educators to the new platform and updating the software, as necessary, based of their feedback. The tool proven successful but limited in use being only an online assessment hosting service.

The project results in the development of a system that helps build online assessments that can be used as a base for a more complex learning management system or as a standalone application.
Acknowledgements

I would like to thank my project supervisor, Dr. Bruno Yun (University of Aberdeen) for his continuous support, advice and enthusiasm throughout the project, as well as for testing the application and helping gather educators for the empirical evaluation.

I would also like to thank all the educators who were part of our preliminary survey and empirical evaluation for offering their valuable feedback and dedicating us some of the little free time they have.
Contents

1 Introduction 8
  1.1 Motivation .......................................................... 8
  1.2 Objectives .......................................................... 9
    1.2.1 Personalized assessments .................................... 9
    1.2.2 Predictable interface ........................................ 9
    1.2.3 Individual feedback ......................................... 9
  1.3 Structure of the Report ........................................... 9

2 Application Functionality 11
  2.1 System Requirements ............................................. 11
    2.1.1 Preliminary research ....................................... 11
    2.1.2 User-driven requirements .................................. 12
    2.1.3 Existing systems ............................................. 13
    2.1.4 Core requirements .......................................... 18
    2.1.5 Functional requirements ................................... 18
    2.1.6 Non-functional requirements ................................. 22
  2.2 Technologies Involved ............................................. 22
    2.2.1 Ruby on Rails ................................................ 23
    2.2.2 PostgreSQL .................................................. 23
    2.2.3 Bootstrap .................................................... 24
    2.2.4 Heroku ....................................................... 24
    2.2.5 GitHub ....................................................... 24
  2.3 Design ............................................................. 25
    2.3.1 System architecture ......................................... 25
    2.3.2 Database design ............................................. 26
    2.3.3 User-interface design ...................................... 27

3 Novelties of the System 34
  3.1 Addressing Random Variables Within Questions .................. 34
    3.1.1 Existing solutions .......................................... 34
    3.1.2 Our solution ................................................ 36
  3.2 Intuitive Integration of Regular Expression Questions .......... 37
    3.2.1 Existing solutions .......................................... 37
    3.2.2 Our solution ................................................ 39
## 4 Development Process

4.1 Accounts ........................................... 42
   4.1.1 Implementation overview ..................... 42
   4.1.2 Problems encountered & solutions ............. 43

4.2 Questions .......................................... 43
   4.2.1 Implementation overview ..................... 43
   4.2.2 Problems encountered & solutions ............. 44

4.3 Assessments ....................................... 45
   4.3.1 Implementation overview ..................... 45
   4.3.2 Problems encountered & solutions ............. 46

4.4 Submissions ....................................... 47
   4.4.1 Implementation overview ..................... 47
   4.4.2 Problems encountered & solutions ............. 47

## 5 Evaluation ................................. 49

5.1 Application Testing .............................. 49
   5.1.1 Unit testing ................................ 49
   5.1.2 Functional testing .......................... 50

5.2 Empirical Evaluation ............................ 52
   5.2.1 Evaluation design ............................ 52
   5.2.2 Results & updates ............................ 53

## 6 Future Work & Conclusion .................. 58

6.1 Future work ..................................... 58

6.2 Conclusion ...................................... 59

## A User Manual .................................. 61

A.1 Managing an Account ............................... 61
   A.1.1 Create an account ............................ 61
   A.1.2 Access an account ............................ 61
   A.1.3 Log out from an account ..................... 62
   A.1.4 Review account details ...................... 62
   A.1.5 Edit account details ......................... 63
   A.1.6 Remove an account ........................... 64

A.2 Managing an Assessment .......................... 64
   A.2.1 Create an assessment ......................... 65
   A.2.2 Review assessment details .................... 65
   A.2.3 Edit assessment details ....................... 66
   A.2.4 Remove an assessment ......................... 67
   A.2.5 Create a question ............................. 68
   A.2.6 Review question details ...................... 72
   A.2.7 Edit a question ............................... 72
   A.2.8 Remove a question ............................. 73
CONTENTS

A.2.9 Share an assessment .................................................. 74
A.2.10 Review a submission ................................................ 74
A.2.11 Remove a submission .............................................. 75
A.2.12 Give further feedback for a submission ...................... 75
A.3 Submit an Assessment Attempt ..................................... 76

B Maintenance Manual .......................................................... 77
B.1 Building and Running the System .................................. 77
B.1.1 Local installation ...................................................... 77
B.1.2 Heroku deployment .................................................. 78
B.2 Software Dependencies .................................................. 78
B.3 List of System Files ...................................................... 78
B.4 Future Adaptations and Changes ................................. 82
Chapter 1

Introduction

1.1 Motivation

Given the particularities of the current COVID-19 pandemic, now more than ever, the education system needs alternate means of assessing student knowledge as it became harder if not impossible to assess students on campus. There is a noticeable switch to online assessment methods and while there is a vast option range, there seems to be only a few that are widely used (e.g. Blackboard across UK or Moodle across France) (Mobo, 2020), (Raza et al., 2021).

In order to better understand the needs and likings of educators we carried a preliminary questionnaire. The results of which showed that while some platforms are already available to create and conduct assessments, there are still plenty of educators who have not yet used or are not familiar with online assessment methods. Of course, there are also students who are in a similar situation. The same results have also shown that most of those who have used online assessments methods before were interested in switching to a new system if it suited their needs better. The three main areas of improvements determined from the survey were: (1) the option to offer assessments that are personalized to each student, (2) an easier to use interface, and (3) the option to give better feedback to students. This was reflected by the results of our questionnaire, i.e. 80% (respectively 50% and 40%) of the respondents considered the feature to be something that could improve upon their current experience.

Following our initial investigation and testing with three of the most famous assessment management systems, namely Blackboaord\(^1\), Moodle\(^2\) and VirtualX\(^3\), we have confirmed the need to improve those aforementioned features. First, while Blackboard offers a considerable amount of assessment flexibility for their users, the interface has proven to be un-intuitive and even confusing at times. It was the case that even experienced users did not know how to use a particular functionality within the platform or did not know it even existed. Second, Moodle has a lighter and easier to use interface but lacks in terms of assessment customisability when compared to Blackboard. Third, VirtualX is in some ways the worst of both worlds, offering little flexibility and a somewhat difficult to use interface.

The analysis of our results showed that there exists a gap in the market for an assessment management system tailored for educators who are willing to trade some of the most complicated functionalities for an easier to use interface, the ability to personalize assessments, and feedback.

\(^1\)https://www.blackboard.com/teaching-learning/learning-management/blackboard-learn
\(^2\)https://moodle.org/
\(^3\)http://virtualx.sourceforge.net/
The goal of our project is to develop a system that delivers such an experience.

1.2 Objectives

The objective of our project is to develop a new assessment system, called Test Creator, to satisfy the educators’ needs. To successfully develop our system, we are planning to focus on and overcome the three main drawbacks of the current systems as highlighted from the initial user research, namely:

1. Personalized assessments for students (see Section 1.2.1)

2. Predictable interface (see Section 1.2.2)

3. Individual feedback to students (see Section 1.2.3)

1.2.1 Personalized assessments

In order to address these requirements in our system, we will provide a new type of question that will include “personalised” variables which will take different values for each student. This enables the educator to set formulas, variable intervals and constraints in order to create the base from which individual questions that assess the same piece of knowledge are generated. Each student will have to use the variable values they have been allocated to reach the correct result for their submission. The intended impact of this feature is to impede cheating by reducing collusion (i.e. students sharing the correct answer for a question).

1.2.2 Predictable interface

The main problem a development team faces when a system offers a lot of functionality is that they have to integrate it within the user interface which can prove to be difficult. If not done well, users can get overwhelmed by large numbers of buttons, text fields within a form or even a menubar, submenu and a large form on the screen at the same time. The aim of our system is to reduce the elements shown on the screen to a minimum by only showing the strictly necessary elements for each situation and by making sure that element labels and error messages are descriptive enough to translate the information to the user but not too complicated, as to avoid confusion.

1.2.3 Individual feedback

It is often the case that students would benefit from a direct feedback from their instructor. From a student point of view, the automatic feedback offered by online tests sometimes creates more confusion than clarification, often requiring getting in contact with the educator for proper clarification. The aim of our system is, on top of the classic automatic feedback, to enable educators to give more constructive feedback to students.

1.3 Structure of the Report

This report relates the process of creating the application described in the above subsections. The body is split between four main chapters, Application Functionality, Novelties of the System, Development and Evaluation.

In Chapter 2, we go into more detail on the structure of the system, what technologies are involved, the system requirements and design. Chapter 3 offers an in-depth look at how the two
main challenges of the system, i.e. addressing random variables and integrating regular expressions within questions have been managed. In Chapter 4, we take an in depth look at the development process, including design changes, difficulties encountered and ingenious solutions. Lastly, in Chapter 5 we will discuss how the system evaluation was developed and carried.
Chapter 2

Application Functionality

2.1 System Requirements

This section tackles the conception stage of the system. We are going to go through the preliminary user research, competitor system experimentation, the requirements derived from this research and a full list of functional and non-functional requirements. The definition of the requirements at the initial stage is crucial as it dictates the structure and functionality of the system. The functional requirements refer to how the system works, the expected features and even to help identify the main modules of the system. Besides these, there are criteria of the system that are not related to the system functionality but to the overall user experience, these are called non-functional requirements.

2.1.1 Preliminary research

In order to better understand what the market is lacking, we organised a preliminary questionnaire using Google Forms. In line with our goal for this questionnaire, we set 12 questions related to the respondent’s background and current online assessment experience. The read-only version of this survey can be found at: https://forms.gle/E5wVLj3T4WwR1k9u8

Question 1 to question 3 are focused on the background of the respondent, for example whether they are a student or educator and the level of education they attend or teach. Question 4 to question 12 are entirely related to the user experience of the respondent. We were interested to know how often they interacted with assessment platforms, and which platforms they are familiar with and prefer. We also wanted to know what makes them prefer a platform in favour of others and how satisfied they were with their current experience. In relation with their current satisfaction, we wanted to know what they found lacking from their experience and most importantly if they were willing to switch platforms if there was a better option out there. We felt that it was important to also offer the option for the user to add their own answers to most questions as there might have been factors that we overlooked.

The participants were recruited from the teams acquaintances or workmates who were known to have a background in education. The responses to the questionnaire were disappointing number-wise with only 11 responses but what surprised us was the range of answers offered by our participants. We had a mix of both educators and students in a ration of 4:1 with most of them part of higher education.

The results have shown a general liking of Blackboard with 56% of respondents while others had some experience with other platforms, including Google Forms, which confirmed our fear that
there are users who use Google Forms outside its intended purpose, as can be seen in Figure 2.1. A surprising detail was that 20% of our respondents have never used an online assessment method.

Most of our respondents rated their satisfaction level as a 3 on a Likert scale from 1 to 5 corresponding to very good to use to very bad, with no response at either extreme. When asked about what can be improved about their experience, 80% of them showed interest in assessments that are more tailored to students, followed with 50% of respondents wishing for an easier to use user interface and 40% wanting more options to offer further feedback to students.

Another important piece of information came from what users consider to be the best quality of such a system. The results have shown predictable system behaviour as the best quality, closely followed by students being registered by default, see Figure 2.2.

Lastly, our respondents have shown a general will to upgrade if another platform suited them better as shown in Figure 2.3.

### 2.1.2 User-driven requirements

As a result of the initial user research we outlined a number of system requirements. We decided that two of these deserve more attention, the addition of random variable and regular expression
2.1. System Requirements

Figure 2.3: Will to upgrade platform

questions, and the ability to offer students direct further feedback.

The formula and regular expression type questions appeared as part of the wish of educators to offer more personalized assessments to students. Formula questions work based on a mathematical formula containing variables that are generated to be different for each student. Regular expression questions are a type of fill in the blank question that is defined as a pattern recogniser that can validate multiple correct answers with an easier setup than the traditional version. Both these types of question are not new as they exist in other platforms, but in our internal tests, the way they are implemented has proven to be confusing even for experienced users. In tandem with the desire for an easier to use system coming from the users, we aim to provide a simpler and clearer way to create formula and regular expression type questions and even offer a bit more customizability over competitor systems in the process. More detail about how these types of questions are implemented can be found in Chapter 3.

Moreover, from the student perspective, We have found that there are cases when the general feedback offered for a question after an assessment ends is not enough and even creates more questions than it answers. For instance, if a student is asked to solve a physics problem (many having more than one solution) and the feedback offers only the expected solution, it will never explain what the student has done wrong or if they were even on the right track to begin with. The result of the initial user research points to the same directions, with 40% of respondents wanting the possibility to offer more specific feedback after the submission was recorded, thus we aim to provide educators with the possibility to do so. Given that the system is only meant to serve as an assessment platform, students are not required to have accounts, but the results and all feedback will be transmitted to them via email.

2.1.3 Existing systems

In order to get a grasp of how the current systems behave we carried experiments with 3 similar platforms, Blackboard1, Moodle2 and VirtualX3 with the goal to compare them in terms of features and user experience. The first two being popular choices for education institutions across

1https://www.blackboard.com/teaching-learning/learning-management/blackboard-learn
2https://moodle.org/
3http://virtualx.sourceforge.net/
Blackboard is by far providing the most flexibility in terms of creating questions, offering many question types and a considerable amount of customizability. As drawbacks we found that there is no real randomness in the allocation of variables for formula type questions as they are selected from a preset set of values (see Figure 2.6) derived from the selected intervals (see Figure 2.5), without the possibility to set relations between them (e.g. $x<y$). While the Regular Expression (RE) implementation is powerful in nature, creating questions proved to be a very un-intuitive process, with nothing to hint the user to the right path. Similarly, the inclusion of variables within the text of the question was not explained to the user in any way, besides the documentation (see Figure 2.4). The documentation was easy to reach and navigate but lacked detail in some crucial places, for example how blank spaces are placed within the question text for Fill the Blank questions or how the negative marking works for Multiple Answer questions. The creation process for a True / False question can be seen in Figure 2.7.

When compared to Blackboard, Moodle is offering less features, most notably having partial integration of REs, only accepting part of its syntax as noted in their User manual under the “Wildcard usage” section\(^4\) and can be seen in Figure 2.9. The interface is considerably less cluttered and easier on the eye (see Figures 2.8 to 2.11), compared to the busyness of the Blackboard interface (see Figures 2.4 ot 2.7). The user manual\(^5\) requires minimal effort to reach and offers plenty of details.

Lastly, VirtualX has proven to be far from the other two both in terms of features and user

\(^4\)https://docs.moodle.org/310/en/Short-Answer\_question\_type
\(^5\)https://docs.moodle.org/
2.1. System Requirements

Figure 2.5: Create formula with variable question in Blackboard (part II)

Figure 2.6: Create formula with variable question in Blackboard (part III)
Figure 2.7: Create True / False question in Blackboard

Figure 2.8: Create Fill the blank question in Moodle (part I)
2.1. System Requirements

Figure 2.9: Create Fill the blank question in Moodle (part II)

Figure 2.10: Create True / False question in Moodle (part I)
experience. It offers only a few question types, which include only the basic Multiple Choice Questions, True / False, etc. (see Figures 2.12 to 2.14). Without the presence of REs, formula type questions or any kind of variable inclusion. The interface was not intuitive and the user manual has proven hard to reach and lacking in detail. Despite all this, it presented an interesting design choice on which we will expand in Section 2.3.3.

2.1.4 Core requirements

Throughout the initial research stage we have identified three core requirements for our system, Creating and managing an account, Creating and managing an online test and taking an online test. These are displayed using UML use case diagrams in Figures 2.15 to 2.17:

- Figure 2.15 relates the use case for creating and managing accounts. It includes the main actions needed to maintain an account, creating, editing and accessing it.

- Figure 2.16 describes the use case for crating and managing an online test. We identified 5 main required actions, launching and ending a test, creating a test, editing a test, giving further feedback to students, launching a test and ending a test.

- Figure 2.17 illustrates the use case for attempting an online assessment. This includes only the main action of attempting the assessment with its respective sub-actions.

2.1.5 Functional requirements

- User should be able to manage an **online test**
  - A user should be able to create a test
    - They should be able to add basic questions (i.e. Multiple Choice, Multiple Answer, True / False, etc.)
    - They should be able to add formula with random variable questions
2.1. System Requirements

Figure 2.12: Create Multi Choice question in VirtualX (part I)

Figure 2.13: Create Multi Choice question in VirtualX (part II)
2.1. System Requirements

Figure 2.14: Create True / False question in VirtualX

Figure 2.15: Create/Access/Manage an account
2.1. SYSTEM REQUIREMENTS

Figure 2.16: Create/Manage an online test

Figure 2.17: Take an online test
2.2. Technologies Involved

- They should be able to set the formula, variable intervals, relationships between variables and set the accepted answer error for formula type questions
- They should be able to remove questions
- They should be able to add correct answer(s)
- They should be able to add RE questions
- They should be able to set test availability
- They should be able to generate a link for the test
  ○ A user should be able to save a test within its own account
  ○ A user should be able to edit a test from its own account
  ○ A user should be able to launch and/or end a test
  ○ A user should be able to give further feedback to students (by email)

• User should be able to attempt an online test
  ○ A user should be able to enter identification details
  ○ A user should be able to select answer(s) for the questions
  ○ A user should be able to submit a test
  ○ A user should be able to see its score for a test
  ○ A user should be able to receive further feedback from their educator (by email)

• User should be able to create and manage its own account
  ○ A user should be able to register, create an account and log into the platform
    - They should be able to enter their personal details (name, email, etc.)
    - They should be able to create a password with certain strength requirements (length, upper and lower case, numbers etc.)
  ○ A user should be able to edit their information
    - Edit password
    - Edit personal details

2.1.6 Non-functional requirements

• Low response time and overall “snappiness” of the system (may be influenced by the internet connection)

• Adaptability to different screen sizes

• Predictable and appealing interface

2.2 Technologies Involved

Once the requirements were set, we explored different technologies to fulfil them. In the following items, we describe the main technologies involved in our system and why we decided to use them:

• Ruby on Rails as the framework of the system (for more details, see Section 2.2.1).
• PostgreSQL as the database solution (for more details, see Section 2.2.2).

• Bootstrap for a responsive and modern user interface (for more details, see Section 2.2.3).

• Heroku as a deployment platform (for more details, see Section 2.2.4).

• Github for version control (for more details, see Section 2.2.5).

2.2.1 Ruby on Rails
Ruby is an object-oriented programming language created by Yukihiro Matsumoto in 1993, and publicly released in 1995 (Ruby, 2021). It has proven, over the years, to be an intuitive and clean programming language, taking advantage of its interpreted language nature (Smyth, 2010).

What really set Ruby off was the introduction of Ruby on Rails⁶, a framework built for easier development, deployment and maintenance of web applications. Ever since its conception in 2004 and the initial release in 2005 (David, 2005), it kept improving with every update, being open source and in continuous development based on feedback and contributions of its users.

It appeared as a result of the frustration of developers with the technologies they were using to create web applications. It is based on the Model-View-Controller (MVC) architecture which was not new at the time but Rails improves upon it, increasing modularity and maintainability of the system by assigning a standard place for each piece of code (Ruby et al., 2013).

Moreover, Rails offer integrated testing features, which are crucial for proper web development. As new functionality is introduced to the system Rails automatically creates test stubs for it. As a result, more testing takes place as it is easier to carry.

One of the main foundation pieces of Rails is the “Don’t Repeat Yourself” (DRY) principle described by Hunt and Thomas (2019) which supports the idea that every piece of knowledge in a system should be expressed in one place. This makes the code short and readable, increasing maintainability.

Our previous experience with Ruby on Rails was successful, offering us all the tools necessary to develop a robust web application. Some famous examples of applications that are based on Ruby on rails are: GitHub, Airbnb, Twitch and SoundCloud (Ruby on Rails, 2021). Django⁷, Sinatra⁸ and Laravel⁹ are some alternative frameworks that could have been also used.

2.2.2 PostgreSQL
PostgreSQL is an open source relational database management system with a long history, dating back to 1985, with the release of version 1.0 in 1995 (PostgreSQL, 2021). It offers enterprise-class features such as the ability to create aggregate functions (e.g. GROUP_BY(), COUNT(), etc.) and also utilize them in window constructs, common table and recursive common table expressions, SQL windowing functions, and streaming replication, which are rarely found in open source databases (Obe and Hsu, 2012).

The main advantage of PostgreSQL is the ease of extending the database, usually without compiling any code. On top of offering these advanced features, it also performs them quickly as

---

⁶https://rubyonrails.org/
⁷https://www.djangoproject.com/
⁸http://sinatrarb.com/
⁹https://laravel.com/
stated by Obe and Hsu (2012). This is why most Ruby on rails development teams prefer to use it in production instead of the SQLite database management system included by default with the framework.

### 2.2.3 Bootstrap

Bootstrap is an open source product from Mark Otto and Jacob Thornton that appeared as a need to standardize the toolsets of frontend engineers at their employing company at the time, Twitter (Spurlock, 2013).

Ever since the initial launch in 2011, it has been adopted by many developers and has evolved from being an entirely CSS-driven project to include a host of JavaScript plugins[^10] and icons[^11] that go hand in hand with forms and buttons. At its base it provides a responsive and robust 12-column grid structure for the user interface (Spurlock, 2013). Moreover, it provides styling for most common web applications elements such as froms, menus, buttons and tables[^12]. With the option to build upon or customise its elements to suit particular needs the use of Bootstrap ensures a robust base for the UI of a modern web application.

### 2.2.4 Heroku

“Heroku is a cloud platform that lets companies build, deliver, monitor and scale apps.” (Heroku, 2021) Heroku is still a relatively recent service, appearing in 2007 as an idea to enhance productivity by James Lindenbaum, Adam Wiggins, and Orion Henry (Middleton and Schneeman, 2013). Despite its current function as a deployment platform, it started as an online code editor for Ruby on Rails applications. In 2009, after observing the interest of their users they relaunched Heroku as the deployment platform we know today (Middleton and Schneeman, 2013).

Using Heroku drastically reduces the time, software and hardware necessary to deploy a web application for testing purposes in production or even for long term hosting. The development team only has to build their code, deploy it and the application should be up and running. One of the nicest features of Heroku is that it can directly link with a Github repository which makes it even easier to deploy the application.

### 2.2.5 GitHub

Before talking about GitHub it is important to introduce what Git is. Git is a distributed version control system, meaning that each member of a team working on a project in Git has a copy of not only the current state of the files, but the full history of the project (Bell and Beer, 2017).

GitHub is a platform where you can upload a copy of a Git repository. Besides storing the Git repository, GitHub also makes it easier to collaborate with other people on your projects by providing a centralized location to share the repository. Actions like forking, Pull Requests, Issues and a web-based interface to view it from (Bell and Beer, 2017).

Even if at a first glance it appears to be a tool aimed towards teams, GitHub offers a number of advantages to individual developers such as the ability to undo changes, a complete history of all the changes, documentation of why changes were made, the ability to change anything without repercussions, and multiple streams of history (Bell and Beer, 2017).

[^10]: https://getbootstrap.com/docs/3.4/javascript/
[^11]: https://primer.style/octicons/
[^12]: https://getbootstrap.com/docs/4.0/getting-started/introduction/
2.3 Design

2.3.1 System architecture

The chosen architectural pattern for our system is Model-View-Controller (MVC) which is similar to the 3-tier architectural pattern. The MVC architecture is specific for the Ruby on Rails development framework that was selected as the base of this system.

As mentioned by Ruby et al. (2013), back in 1979, Trygve Reenskaug introduced the MVC approach for developing interactive applications. He came up with the idea to split the application into three main types of components: Models, Views and Controllers.

The Model component is responsible for processing, associations and validations, as well as enforcing rules that are required to manipulate the application data. This ensures that invalid data are stopped before reaching the database. In parallel, it is also responsible for the state of the application, being the only component in contact with the database, acting both as a data store and a gatekeeper. In the case of our application, the models are responsible for structuring data for user accounts, assessments, questions and submissions.

The View component is the layer of interaction between the user and the application, providing the user interface based on data provided by the model. This layer of the system is solely responsible for the transfer of data between the user and the model, but not for handling incoming data. It is often the case that many views are in contact with the same model but for different purposes. For instance, there can be one view for displaying an assessment information and another to create a new assessment, both sending and receiving data from the same model.

The last component of this architecture is the controller. It acts as a negotiator between the views and the models. Controllers are responsible for data validity, allocating data fetching and processing to the model layer, delegating resources and selecting the right representational information to pass to specific views. In other words, they orchestrate the application.
To better illustrate the workings of the MVC architectural pattern, Figure 2.19 presents the sequence of action that take place within the system. First, the browser sends a request to the controller. Then, the controller interacts with the model to fetch the appropriate data, which in turn queries the database. Once the data is fetched, the controller selects the right information to display and sends it to the view. Lastly, the view presents the requested data to the user via the UI.

This type of architecture helps implement the modularity principles\(^\text{13}\) such as encapsulation, minimal coupling and maximal cohesion. Having the system organised in such way offers greater maintainability and ensures a higher up-time of the system as even in the event of a failure in one of the modules, the system can still run with limited functionality until the issue is resolved. Moreover, changes can be made to one part of the application without affecting others. For instance, if the customer desires a change in the user interface only the views are affected, making the process of adding and removing features easier and less prone to general system failures. This also opens the possibility for parallel development of the view, model and controller components by different development team members prior to creating the controllers to tie them together. This enabled us to constantly switch between the components, avoiding a monotonous development process.

### 2.3.2 Database design

Considering the expected relatively high volume of data entries of our system, an efficient and reliable storage solution is necessary. Our initial initial design for the database is depicted using the Entity Relationship Diagram (ERD) shown in Figure 2.20. It consisted of five entities, User, Assessment, Submission, Question and Answer. For each of these entities we store entity specific data such as type, title and text for a question, as well as an id that denotes the relationships between entities, in the case of a question we store the id of the assessment they are part of. The relationships between the entities are as follows: A user can have one or many assessments. An assessment can have one or many questions as well as one or many submissions. Lastly, a question can have one or many answers. After much consideration and various ideas on improving storage efficiency, we concluded that some changes had to be made.

The most notable difference was the way we store answers. We steered away from having them as a separate entity and included them as part of the question using special formatted strings. This decision was made as answers for different question types required different kinds of data.

\(^{13}\)http://www.cs.sjsu.edu/faculty/pearce/modules/lectures/ood/principles/Modularity.htm
stored which would have led to having a large amount of fields for Answer entities which would have never been entirely filled on any answer entry. For example, for a fill the blank type question, the only thing to store would be the correct answer while for a formula with random variable question we have to store the formula, variables and their ranges and relations between variables, if any. We have also added an options field to question entities that stores options such as negative marking, case sensitive and partial marking, again, using special formatted strings. Small changes were also made to the data stored for submissions and assessments. All the changes can be seen in Figure 2.21.

### 2.3.3 User-interface design

With the initial user research results showing that 50% of respondents wished for an easier to use interface we have given much importance to how the user interacts with our system, taking full advantage of what Bootstrap has to offer.

We created a lightweight interface, avoiding clutter and unnecessary data displayed on screen similar to the experience provided by Moodle which has proven to be the easiest to use out of the 3 tested platforms presented in Section 2.1.3 (see in Figure 2.8). Our UI has all the additional required functionality arranged in such a way to make it easy to access for users. One interesting design choice that we found on VirtualX was the use of a template for creating different types of questions (see Figures 2.15 to 2.17). We consider this to be a great approach in order to limit confusion created by using different page arrangements for different question types. We plan to implement the question creation form in a similar manner, using dynamic fields on a single template.

Some of the mock-ups from the early design stage ranging from low to medium detail can be seen in Figures 2.22 to 2.25 and screenshots from the development process can be seen seen in

---

**Figure 2.20:** Initial Entity Relationship Diagram
2.3. Design

Figures 2.26 to 2.30.

There were no substantial changes between the mock-ups and the actual user interface, many of them consisting of small adjustments meant to increase usability, such as the order of the form fields and indentations or emphasis on important pieces of text. A major decision we have taken was to move the list of questions that are part of an assessment from the create assessment screen to the view assessment screen. Resulting in the user first creating the assessment and then adding the desired questions to it, without the risk of having to start everything from the beginning in case of an error. This was a necessary change as poor internet connection or other issues encountered during the assessment creation process would have led to the entire work being lost, as with the new approach if such a problem occurs only the question the educator is currently working on would be potentially lost.
2.3. Design

Figure 2.22: Low detail mock-up of the interface to create a question

Figure 2.23: Low detail mock-up of the interface to create an assessment
Figure 2.24: More detailed mock-up of the interface to create a formula type question

Figure 2.25: More detailed mock-up of the interface to create a RE type question
Figure 2.26: Development stage view assessment page

Figure 2.27: Development stage create question screen for MCQ type questions (part I)
Figure 2.28: Development stage create question screen for MCQ type questions (part II)

Figure 2.29: Development stage create question screen for RE type questions (part I)
Figure 2.30: Development stage create question screen for RE type questions (part II)
Chapter 3

Novelties of the System

This chapter describes the novelties of our system. This includes why they are necessary, competitor system approaches and our solution.

3.1 Addressing Random Variables Within Questions

This section focuses on the response to the interest in assessments that are tailored to students highlighted by the results of the user research (see Section 2.1.1). The intended impact of using random variables is to impede cheating by reducing collusion (i.e. student sharing the correct answer for a question). Using personalised variables will also encourage students to put in practice what they have learned in a completely new scenario.

3.1.1 Existing solutions

From the competitor system testing we carried in the initial stages of our project (see Section 2.1.3), we found that two out of the three evaluated systems offer the possibility to create formula questions (with variables). Those two platforms were Moodle and Blackboard.

Moodle

Moodle’s approach to integrating variables into questions comes in two forms “Calculated” and “Calculated multichoice” questions. They work in a similar manner, with the latter offering a choice of results using different formulas instead of having the user to fill out a text box with the correct answer.

We found their solution to be minimalist and easy to use but lacked configuration options (see Figures 3.1 and 3.2). On Moodle there is no possibility to set an interval for each variable, or define relations between them or other variable constraints, which is appropriate when dealing with simple formulas. A nice touch we noticed was the inclusion of an answer tolerance, enabling educators to accept numerical answers that are within an accepted margin of error. For instance, 0.5 as a margin of error accepts anything between 1.5 and 2.5 for the defined answer 2. When more complex formulas are required, it is essential to offer the possibility to fully define the set of values a variable can take. For instance, when calculating the area of a rectangle using, \( A = w \times l \), it is crucial that the values generated for \( w \) and \( l \) are strictly positive.

We are not certain if Moodle developers have implemented some functionality “behind the scenes” in order to cope with more complex formulas or if they have introduced these types of question with the intention of delivering an approachable way to integrate simple formulas into questions.
3.1. Addressing Random Variables Within Questions

Figure 3.1: Create Calculated question in Moodle (part I)

Figure 3.2: Create Calculated question in Moodle (part II)
3.1. ADDRESSING RANDOM VARIABLES WITHIN QUESTIONS

Blackboard

In contrast with the offering from Moodle, Blackboard has taken into consideration the use of more complex formulas in their solution to integrate variables into questions. They offer the “Calculated formula” question which acts like a Fill in the blank question only that the answer is calculated using the predefined formula and generated variable values.

The Blackboard solution features a really nice formula builder (see Figure 2.4) which offers the option to select the interval for each variable (see Figure 2.5) and lastly, generates a set of variable values which the user can review and alter to their liking (see Figure 2.5).

We found this approach to be the greatest from the ones we tested, both in terms of usability and functionality. However, it is still missing the use of relations between variables or other variable constraints. Leaving the user with the responsibility to spot any problem with the generated variable values and enforce the necessary constraints manually. This can prove to be a problem when a formula with many constraints is used as it becomes prone to errors and very time consuming for users to manually test every set of variable values.

A drawback of the Blackboard approach is the lack of an answer tolerance option. This is a must taking into account the rounding errors that can occur when a series of formulas are applied by hand versus using a calculator. For example, when calculating the area of a right triangle knowing the length of it’s hypotenuse \(c\) and one of the other sides \(a\). One could apply Pythagoras theorem to find the length of the third side \(b\) then apply \(A = \frac{ab}{2}\). For \(a = 5\) and \(c = 8\) when calculating by hand the result would be 15.6 while using a calculator the result is 15.61. This not even considering that a student might solve the problem by finding the length of the height perpendicular to the hypotenuse and use that to calculate the area of the triangle.

3.1.2 Our solution

As emphasised previously, we considered the lack of full variable constraints a drawback of the tested systems, While Moodle offered complete randomness regarding variable values it lacked support for more complex formulas and while Blackboard showed more attention to such formulas it lacks the true variation, having a small set of variable values to choose from. We wanted to offer the random variable generation merged with the attention to variable definition as a full package.

In order to achieve this we have turned our attention to Mathematical Optimization methods. From these we focused on Integer Linear Programming (ILP).

In the words of Snyman and Wilke (2018) “Mathematical Optimization may be described as the science of determining the best solutions to mathematically defined problems, which may be models of physical reality or of manufacturing and management systems”. It plays a crucial role in life as we know it, extending its applications from minimizing energy configurations of general structures, such as bridges, to making economic decisions in society and industry, for instance maximizing profit or minimizing cost. It has a relatively short history, introduced when the simplex method for solving the special class of linear programming problems was developed by Dantzig et al. (1955).

Integer linear programming (ILP) investigates linear programming problems in which the variables are restricted to integers with the general goal to maximize a given equation. It can be considered as the search for lattice points in polyhedra, or as solving systems of linear equations (Schrijver, 1999).
We have found the workings of ILP fascinating and saw the potential of using it as a method to offer different variable values to different students whilst satisfying both variable intervals and constraints. We have noticed that changing the maximisation equations yields different variable values. Taking into consideration that every submission must contain a unique identifier (Student ID) we decided to generate a unique equation to maximize for each submission based on the identifier (see Section 4.4.2 for more detail) and use it in combination with the predefined variable intervals and constraints defined for each formula type question. This will ensure variation in variable values between submissions and also the ability to reliably generate the same variable values again for a given student. The latter is important so as to ensure the assessment remains the same once it has been started for a given student and outside factors such as page refreshes will not change the variable values. We found that we could use the ILP mechanism outside its classic use case and generate both maximization and minimization equations and also, extend the variable values to the negative spectrum of integers.

However, the use of ILP for this task restricts us to use only integers as variable values, while rational numbers would be ideal. Moreover, through testing, we have determined that the optimisation equations have to be quite different between submissions in order to generate the whole spectrum of possible values for the variables. The way we process the unique identifier to obtain the optimization equation addresses this weakness but cannot ensure great variation of values between submissions for all possible cases.

To further clarify the way this works we are going to walk through an example. Let's say an educator sets a question to use the formula \( x + y \), the value of \( x \) to be between 0 and 10, the value of \( y \) to be between 0 and 20 and the constraint \( x > y \). Once a student starts a submission the optimization relation is created based on their id. On top some additional constraints are created for each variable based on this equation, this is done in order to achieve better value diversity between students. For student \( A \) with id 1234 it will be \( 49 \times x - 11 \times y \), the optimizer having to minimise it and for student \( B \) with id 12345 it will be \( 41 \times x + 49 \times y \), the optimizer having to maximise it. The additional constraints for student \( A \) are \( x \leq 4, y \leq 2 \) and for student \( A \) are \( x \geq 4, y \geq 2 \). After the optimizer runs student \( A \) receives \( x = 3, y = 2 \) and student \( B \) receives \( x = 10, y = 2 \), each of them having to calculate the answer for the question using these variables.

### 3.2 Intuitive Integration of Regular Expression Questions

This section adds to the initiative to offer assessments that are tailored to students highlighted by the results of the user research (see Section 2.1.1). We considered that Regular Expression questions would be a welcome addition along with the introduction of formula with variable question. Regular expression questions represent an evolved state of the classic fill in the blank questions, offering the answer a greater level of flexibility. The expected impact is to reduce the cases when an answer is marked incorrect when it is actually correct. For instance, there are questions that have multiple different right answers (e.g. water being the same with \( \text{H}_2\text{O} \)), or cases when the capitalization of a word does not make a difference in the correctness of an answer.

#### 3.2.1 Existing solutions

The experiments carried on competitor systems in the initial phase of the project (see Section 2.1.3) have shown the embrace of REs in two of the three systems tested. Those two platforms...
were Moodle and Blackboard.

**Moodle**

Moodle only makes clear use of Regular Expressions within their assessment features in the form of “Wildcards”\(^1\). These can be used as part of their “Short Answer” questions and offer limited functionality.

Wildcards are denoted by the “*” character and are used a placeholder for anything. For example, “m*ing” will accept any string that starts with “m” and ends in “-ing”. The Wildcard can be cleverly used to create questions that accept quite complex and varied answers, such as “*one*two*” which accepts any string that contains the words “one” and “two”. Educators having the possibility to add multiple such correct answers, even with different marks value in order to further expand their question potential (see Figure 2.9). We found this concept intriguing and considerate towards inexperienced users but insufficient for creating more complex questions. For instance, if an educator wants a question to accept a string containing any amount of 0s followed by a 1 it is impossible to make use of the wildcard, but easily achievable with REs by using “/(0*1)/” as a validation expression.

We suspect that other features related to text answer interpretation, such as “Case Sensitive”, make use of REs but we cannot be certain without access to the source code. We are not certain why they made the choice to include this little of the power of REs, but we tend to believe that this related to the usability of their platform, which has proven to be great overall. This trade of functionality for usability still yielded a platform appropriate for most common use cases.

**Blackboard**

Once again, Blackboard offers a more complex approach compared to Moodle. Blackboard incorporates Regular Expressions as part of their “Fill the blank” question unsing the option called “Pattern Match” (see Figure 3.3).

We found that their approach used the full potential of Regular Expressions but this was uncertain at first, as it is not clear what they refer to as "pattern". Until we consulted the documentation we had no way of knowing what we were supposed to input into this field. The nice thing about their implementation is the inclusion of the pattern checker (see Figure 3.4). This enables users to verify if their expression works the way they intended to. As apparent from Figure 3.4 we tried to validate a wide variation of forms for the answer “Romeo and Juliet”. We included options for capitalized or all lowercase names and replacing the “and” with “&”, “,” or just a plain space. The expression checker worked as expected and tested right for all our inputs. However, the user can only test one answer at a time, making the checker slow to use and less useful for spotting and correcting errors within the expression. For example, for the “[oO]ne[[TT]wo” expression, it would be harder to spot the capital “t” typo as it works fine for “One” and “one” so the user assumes that it would work as well for the “two”-“Two” pair as they used the same method of validating both versions. The use of multiple test cases at once enables the user to run the test for all the desired accepted and unaccepted answers and see exactly where it fails and also see how each change in the expression affects the overall accepted and unaccepted cases.

We suspect that Regular expressions have been used by the development team behind user

\(^1\)https://docs.moodle.org/310/en/Short-Answer\_question\_type
features related to text interpretation, such as their “exact match” option, but this has to remain an assumption as we do not have access to the source code. The lack of clarity on how to use pattern matching without using the documentation left us disappointed. We think that it is a shame that such a powerful feature can go unobserved by most users just because it is not intuitive and clear to use.

3.2.2 Our solution

Taking into consideration the approaches of Moodle and Blackboard on integrating Regular Expressions in their systems, we wanted to deliver an alternative that aids educators in the creation of their assessments. While Moodle offers an enjoyable user experience with its minimalist approach it lacks full function and while Blackboard makes use of the Regular Expression potential it proved to be easy to leave unnoticed and confusing to use. In order to reach our goal we created an interface that is both intuitive and powerful.

We created a type of question that is dedicated to Regular Expressions that is separate from the basic “Fill in the blank” question. This decision was made for rising awareness of the existence of this option. The user only has to enter their expression in the clearly marked field (see Figure 2.29), without having to refer to the documentation for additional support.

We liked the idea of the user not having to go to another service to test their expression so we included an expression checker that is accessed by clicking the dedicated button next to the expression field. The checker appears in the form of a modal (see Figure 2.30). Here the user can check and update their expression. Due to the frustrating experience from Blackboard we decided to enable users to test up to ten answers at once, users can add and remove answers as desired. Once the check button is clicked the accepted answers turn green and the declined ones turn red.
To further aid the creation and testing of expressions, we have included a “Cheat-Sheet” containing the most common regular expression operators within this modal. It is important to note that any change to the expression within the expression checker is automatically saved to the original expression (more detail in Section 4.2.1).

Last but not least, we also intended to use Regular Expressions to implement the functionality related to checking answers using the three options provided for “Fill in the blank” questions (see Figure 3.5). This proved to be too complicated of a solution and ended up using string altering functions instead.

Figure 3.4: Create Fill the blank with RE in Blackboard (part II)
Figure 3.5: Create Fill the blank question in Test Creator
Chapter 4

Development Process

The development of our system followed a part Test Driven Development (TDD), part Rapid Application Development (RAD) approach. We made use of Unit tests for part of the system functionality and and functional tests for the rest, we discuss this in more detail in Chapter 5.

As explained by Beck (2003), TDD is a development technique that has the goal to produce clean code that works. It accomplishes this by driving development using automated tests, it implies an order to the tasks of programming, writing tests that fail, develop code that passes the tests and refactor the code to eliminate duplication.

James Martin (1991) defines RAD as “a development lifecycle designed to give much faster development and higher-quality results than those achieved with the traditional lifecycle”. He describes its key objectives as: high quality systems, fast development and delivery and low costs. It relies on four phases: requirements planning, user design, construction, and cutover. Requirements planning revolves around team members discussing and agreeing on the project scope, system requirements and constraints. User design relies on the interaction of the development team and users in order to construct prototypes for all system processes. Construction implies the actual implementation and cutover refers to the final tasks of the development process such as testing and data conversion. The key element being the repetition of the user design and construction phases until it is confirmed that the product meets all requirements.

4.1 Accounts

4.1.1 Implementation overview

The chosen design for accounts is inspired from the accounts we use on many platforms daily. A simple email and password combination and an account that hold basic information about the user, email address, first and last name.

The implementation is based on the on the guide given by Hartl (2015) in Chapters 7 to 10 of his book. Registering users and accessing accounts uses reCAPTCHA\(^1\) to avoid unwanted traffic on our website. The implementation includes a session helper that offers users the possibility that, once logged into their account they are remembered until they choose to log out, even if they close the tab or browser.

User accounts have been used throughout the application to restrict the pages an actions a user can access and do. This has been accomplished by using checker functions called “before_action” in all the controllers. The user registration has been designed to meet adequate security standards

\(^1\)https://www.google.com/recaptcha/about/
by requiring the use of a secure password that contains digits and both lower and upper case characters. Furthermore, users are required to activate their accounts by following a link provided automatically via email before they can access their accounts. In case of a user forgetting their password there is a password reset function that sends an email to the account email address with a reset link that the user can use to reset their password and regain access to their account.

4.1.2 Problems encountered & solutions
The implementation of accounts raised only one major issue, this was related to the account activation. The activation key was not generated correctly when sent to the user, resulting in the impossibility of activating an account. After much troubleshooting, we found out that the the token used in the email was regenerated instead of being pulled from the database, resulting in having two different tokens, one stored in the database and another sent to the user. The issue was resolved by removing the second generation of the token.

We faced some problems with the access control to pages and actions that were easily sorted out by correcting the before_action checks. We also faced some minor problems with the password reset as the account was, initially, not updated with the new password, leaving the record untouched and users without the possibility to regain access to their account, the issue being caused by the order of commands within the controller method that was corrected.

4.2 Questions
4.2.1 Implementation overview
The implementation of questions was by far the most time consuming of the entire project but the most rewarding at the same time. The creation of questions is available only for registered users. There are 6 question types supported by the application, Multiple Choice (MCQ), Multiple Answer (MA), Fill in the blank (FTB), True/False (TF), Regular Expression (RE) and Formula with Random Variables (FRM).

As mentioned in Chapter 2, we wanted to have a single form that had dynamically updated fields for each type of question. This was accomplished by using JavaScript to detect any change made in the Type drop-down menu and having a multitude of hidden fields that are dynamically updated by it. These fields include options and all the fields required to define the answer for that specific type of question. To follow this dynamic nature of the form we have implemented the fields necessary for full definition of a FRM question to appear as the formula is written, detecting the variables within the formula and showing the fields used to set the interval for each variable. A simpler approach was used for showing the fields that set the mark value of each answer when “allow partial marking” is selected in a MA question.

The fields for the question description and feedback use the TinyMCE\textsuperscript{2} editor. This is a “What You See Is What You Get” type editor that allows educators to format the text in exactly the way they want it to be displayed to the students. This includes support for special symbols and both mathematical and chemistry formulas besides the usual text editor functions we are all used to, such as, paragraphs, fonts, text sizes, lists, etc.

An extra function used to aid educators in the creation of RE questions is the integrated expression builder/checker that provides educators with a “cheat-sheet” and functionality to test

\textsuperscript{2}https://www.tiny.cloud/
whether their expression behaves as expected or not. The builder offers the option to test up to 10 different answers at once, making the validation and troubleshooting process easier for the user as they can see exactly what part of the expression is not working as expected, fix the issue and test it all again to make sure it words properly, similarly to the process of using unit tests during the development of a system.

For users that do not want to experiment with regular expressions we have included 3 options for FTB questions that help educators account for some variation in student answer, i.e. case sensitive, allowing multiple spaces and allowing answers that contain the correct answer. FRM questions offer a special option that we considered absolutely necessary when dealing with formulas that use variables. This option is called “answer error” and defines the error allowed in answers, for instance if this is set to 0.5 and the right answer is 2 any answer between 1.5 and 2.5 will be considered correct. All the questions offer the option for negative marking, when set this describes the marks that will be subtracted from a student’s score only if their answer for that question is worth 0 marks. This encourages students to give an answer to a question only if they are confident in their solution, introducing the element of risk in answering questions out of luck.

The view question screen contains all the information regarding the question and a preview of how the question will be rendered for students.

4.2.2 Problems encountered & solutions

The amount of work put into implementing questions attracted after itself the most problems we had during the whole development process. Having 6 types of questions that have to work together and be stored as equivalent records in the questions table from the database meant that a universal way of storing the different types of answers and options was necessary.

Our solution was to use special formatted strings to store the two. Options use a sequence of labels and the boolean value attached to it, for instance if a questions has negative marking enabled the string will include “NEG1” else “NEG0”. For options that also require a numerical value the label boolean pair is succeeded by the “P” label and the numerical value, for example, “NEG1P-5” means that the question has a negative marking of -5.

A similar approach has been used for storing answers, only this time special bracket characters have been used to denote the start and the end of an answer text followed by the percentage of the total marks the answer is worth, for instance, “⟨Bear⟩0% ⟨Banana⟩100% ⟨Dog⟩0% ⟨Cat⟩0%” is the formatted answer string for a MCQ question that includes the four answers and marks the second as being the correct solution. The use of these special brackets is not ideal as their very rare use within the question text will create an error within the system. All the answer types have different variations of this special formatted string that encodes all the answer data as a precise piece of information. Within the view question page these special strings are interpreted and displayed in plain English sentences that are easily understood by the user.

As mentioned in Section 3.1.2, our solution for addressing random variables uses Integer Linear Programming (ILP). Our system uses an ILP library called Ruby-Cbc\(^3\). The implementation of this went without many problems after we found a reliable way to build problem using the data provided by the user. However, this library is dependent on the coinor-libcbc-dev\(^4\) Ubuntu

\(^3\)https://github.com/gverger/ruby-cbc
\(^4\)https://packages.ubuntu.com/bionic/coinor-libcbc-dev
4.3 Assessments

4.3.1 Implementation overview

The implementation of assessments was the simplest of the four stages but of high importance towards the function of the system as it represents the common element between questions and submissions. The creation of assessments is available only for registered users. The data retained for the assessment alone is minimal, only containing the assessment title, description, duration and availability window.

We found it of high importance to offer users and easy way to set the availability window of the assessment. To achieve this, we used Tempus Dominus. Tempus Dominus is a DateTime
picker plugin developed to work in Bootstrap based applications. It offered us an easy and pleasant to use DateTime picker for the assessment creation process.

We considered it would be useful to use the same time zone across the application in order to avoid confusion created by users being in different time zones. Assessments use the UTC time in all time related interactions, e.g. setting an assessment availability window and later showing it in the submission screen to students. The same time standard was used for account and submission details.

Similar to creating questions, assessments make use of the TinyMCE\textsuperscript{11} text editor on their creation page. This is done to enable users to create the description they desire for their assessments, including formulas, special characters or even special formatting.

### 4.3.2 Problems encountered & solutions

The main problem we faced when implementing assessments was conveying all the information about the assessment (i.e. assessments details, questions and submissions) to the educator. We wanted to offer the right amount of information on the view assessment screen, low enough as not to overpopulate the screen but high enough so that the educator would not have to navigate back and forth to see the basic data related to the assessment. The bulk of the information is related to the questions and submissions. We aimed to include the main pieces of information about each of the two. For questions, we decided to include the title, type and marks, while for submissions we decided on showing the student id, email address, score, submission time and duration (see Figure 4.1).

We faced some minor problems with the validation of the dates and parsing the correct format that were easily resolved. The Tempus Dominus plugin used for the date picker required some trial and error to achieve the configuration we desired. Lastly, we had an error in our code that would show all the existing questions and submissions on the platform instead of just the ones.

\textsuperscript{11}https://www.tiny.cloud/
corresponding to the assessment, this was related to assigning the right assessment id to questions and submissions when they are created.

4.4 Submissions

4.4.1 Implementation overview

As the last major stage of our implementation, the submissions make all the work up until now worth the effort. We have chosen to allow any visitor to the website who possesses the submission link for a specific assessment to submit an attempt, avoiding the need of creating accounts for all the students using the system. This choice was made as, without control over who can create a new account and who can attempt an assessment students with malicious intent could create fictive accounts and see the questions and correct answers for an assessment. The submission process is mainly split into two stages, identifying the user and recording the attempt.

Student identification is made based on their Student ID and email address and at this stage the student is displayed the main information about the assessment, i.e. the assessment name and creator, its description, availability window and duration, as well as the way the submission process works.

The second stage involves the assessment questions, once the timer starts the student is free to complete their assessment. To avoid any malicious intent, we made sure that the timer is not reset and the potential formula variables are not changed on page reloads. To allow students to take the full advantage of their time, we have implemented an auto-submit feature that is triggered once the time has run out. Upon submission, the assessment results and automated feedback are sent to the student via email.

We enabled educators to review the submissions of their students and offer further feedback if they find it necessary, via the form found on the same screen. We decided to allow educators to remove submissions from unknown individuals or offer students a second chance if they decide this is deserved by allowing them to delete any submission for one of their assessments.

4.4.2 Problems encountered & solutions

During the implementation of submissions, we faced an unexpected problem, storing both the student details and the submission attempt as one entity in the submission table. This could not be accomplished by using a single controller method that is responsible of a two part form as this would let students restart the submission after they have seen the questions. The issue is that there would not be any way of knowing that the student started the submission as the only time a record would be made in the database would be the at very end when the answers are submitted.

We came up with an ingenious way of using the standard new and edit functions of a Rails controller. Once the students identifies themselves, a new submission is created and stored containing their details, blank answer and score as well as the submission start time. When the student is ready to submit their assessment the previously created submission record is updated with their answers, scores for each question and the duration of their submission. This approach ensures that a student cannot restart a submission after they have seen the questions and the timer could not be reset by a simple page reload.

We had a minor issue with the auto-submit feature as the JavaScript timer does not return full seconds and thus the timer rarely hits a true zero as we initially expected. We solved this by
submitting the form once the first timestamp after the true zero is recorded, giving the form timer an error of less than a second, which we found acceptable. Other small issues were encountered and addressed during the submission implementation.
Chapter 5

Evaluation

5.1 Application Testing

This section covers the initial testing done on our application. Throughout the development process we have used two types of testing, unit and functional testing. We are going to present the testing methodology, test results, problems that we encountered throughout the process and design changes that took place as a result of testing for each of the two testing methods.

5.1.1 Unit testing

Generally, unit testing is defined as the “testing of individual hardware or software units or groups of related units” (IEEE, 1990). In the case of software development it refers to the testing of individual building blocks within a system, such as methods. This kind of testing improves productivity as it becomes easier to spot unexpected behaviour directly at the source, especially when the Test Driven Development (TDD) process is practiced. TDD implies creating tests that describe the system functionality before the system is fully developed. Usually in the development team, there is a group dedicated to writing tests which is different from the group responsible for implementation. This ensures that code works as intended from the beginning and saving considerable debugging time later on.

Testing methodology

Creating unit tests is not a complicated task when looking at only a few tests as their syntax is quite simple and manageable. Unfortunately, this is not often the case. In our case the users controller contains 17 test cases that cover just the permissions to perform different actions within the controller. Thus, it becomes apparent that writing tests is not a trivial activity as it is very tedious and should be taken seriously if great results are expected.

Tests are created in order to confirm that a controller behaves as expected and in line with the system requirements. To follow the example of the user controller, tests must address each of the controller methods in all three possible authentication states, not logged in, logged in as user, logged in as admin. Sometimes the expected behaviour is the same between all states, as is the case of creating a new user account, this should be available for anyone. However, most of the time the expected behaviour is different, as in the case of removing a user account, not logged in users should not be able to remove any account, logged in users should be able to delete their own account and admins should be able to remove any account.

Once all the required tests are written they are run and then, methods within each controller are implemented to pass the tests and thus respect the specification of the system.
Test results & updates

The result of each test is simple to interpret, it either passes or fails. Interpretation of the test package is what helps improve the overall function of the code, individual tests only pointing to the source of the problem.

The use of unit tests helps early debugging as tests are run all at once and if a bug fix affected a previously working piece of code it becomes quickly apparent. We used our tests to verify how the different authentication states affect the possible actions within the controller, in the case of the user controller these are Create, Edit, Remove and Show. In our case the initial test passed in 13 out of the 17 test cases, pointing us to the source of the problem.

The four failed tests were for the Show and Remove actions where all the users could show and remove the accounts of any user of the system. The fix for the Show action was quick as we forgot to include the necessary checks before the page was loaded. However this was not the case for the Remove action, here the checks were in place and the order they were performed in looked correct but after much frustration it turned out to be a typo.

The tests for the other controllers were performed in a similar manner, showing us the mistakes we have made throughout the implementation process and helped us keep track of, and address bugs ready to happen. During the testing process we have concluded that a design change was necessary. Initially we intended to forbid logged users from creating a submission for an assessment. This however, is not correct, because logged users can be both educators and students as they can teach a course and attend another at the same time, situation that we initially overlooked and later corrected.

5.1.2 Functional testing

Functional testing refers to “testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions” (IEEE, 1990). In the case of our system this implies testing each item on the functional requirements list for compliance and also verify the function of other features related to them.

Testing methodology

Similarly to unit tests, functional tests have taken place throughout the implementation. They have been carried each time a screen and the functionality related to it has been implemented.

To better illustrate the workings of such tests we are going to describe the testing process for the question creation page. Before the beginning of the actual test it is important to define the checks that have to be made in order to ensure proper function of the page. Table 5.1 represents the list of checks we followed for testing the create question screen. These are high level checks as we were familiar with the intended functionality.

After the test is done any checks that failed are addressed and the test is carried again. This sequence repeats until all the checks pass and the page respects specification.

Test results & updates

The results of functional tests show us what functionality is still not implemented up to specification. Individual tests showing us where the gaps are and where to focus our attention.

Running functional tests during implementation helps towards making the final user experience match the expectations. These are used as a preliminary test before the system is released to
5.1. Application Testing

<table>
<thead>
<tr>
<th>Check</th>
<th>Yes</th>
<th>No</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does the page load?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Does changing the question type update the form fields?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do all the form fields accept appropriate inputs?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Does checking the partial marking update form fields?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Does writing the formula update the form fields?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do all the required fields request inputs?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do all the other errors trigger as intended?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1: List of checks for the create question screen

the test users for the final testing stage. Functional tests were used to validate each of the screens within the application. For example, the test for the account creation screen (see Table 5.1) had to be carried a total of 3 times for all the checks to pass.

<table>
<thead>
<tr>
<th>Check</th>
<th>Yes</th>
<th>No</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does the page load?</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Does changing the question type update the form fields?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do all the form fields accept appropriate inputs?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Does checking the partial marking update form fields?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Does writing the formula update the form fields?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do all the required fields request inputs?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do all the other errors trigger as intended?</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: Results after the first functional test for the create question screen

The first run of the test showed us issues related to updating form fields, accepted input types, variable recognition and error triggers (see Table 5.2). After changes were made to address this issues the second run of checks begun and confirmed our fixes for accepted input types and triggering errors but highlighted new problems for variable recognition and updating form fields (see Table 5.3). The following debugging session proved successful with the next run of the test passing all the checks.

All the other screens were tested in a similar manner, continuously evolving until matching the specifications. One major design change resulted from functional testing was introducing explanation notes for fields that were not clear enough by themselves. This helped educators in the creation of questions by avoiding to consult the user manual for just small details.
5.2 Empirical Evaluation

In the words of Chin (2001) “Empirical evaluation refers to the appraisal of a theory by observation in experiments. The key to good empirical evaluation is the proper design and execution of the experiments so that the particular factors to be tested can be easily separated from other confounding factors.” Due to the goal of our system to improve the experience of educators using online assessment methods we deemed it crucial that the final product is dictated by their feedback. We focused our empirical evaluation on the usability of the system in relation with the educator needs. Unfortunately, by the time we realised this evaluation required ethical approval the estimated time to obtain it would have placed the evaluation period outside our time-frame. However, we made sure that no personal data of any kind would be collected as part of our evaluation and all answers would be completely anonymous.

5.2.1 Evaluation design

We knew that, in order to get useful results out of the evaluation we had to design the survey in such a way as to highlight the general user experience within the platform. We created a questionnaire consisting of three main parts, tester background, task completion and general impression. The read-only version of this survey can be found at: https://forms.gle/yqCyjp8VefGc9qeG9

The first part (Section 1) focuses on the tester background, we knew that our testers come from the education system but we needed a little bit more detail to better understand how the system is perceived based on their previous experience. We were interested in their status, student or educator, what education level they come from, what online assessment platforms they are familiar with and how often they use them, similar to the first section of our initial questionnaire.

The second part was split into 5 sections (Section 2-6), each corresponding to one task to be completed. We selected the tasks as to cover the activity of an average user using the platform. The tasks are: creating or accessing an account, creating an assessment, creating two questions, one of them having to be one of the two special kind of questions we offer (Regular Expression or Formula with Random Variables) and submitting an attempt for the newly created assessment.

For each of the 5 tasks we wanted to know whether they managed to complete the task, how hard was it to complete (on a Likert scale from 1 to 5), and if they had any feedback related to the

<table>
<thead>
<tr>
<th>Check</th>
<th>Yes</th>
<th>No</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does the page load?</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Does changing the question type update the form fields?</td>
<td>X</td>
<td></td>
<td>When multiple choice is selected, fields from fill the blank are also loading.</td>
</tr>
<tr>
<td>Do all the form fields accept appropriate inputs?</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Does checking the partial marking update form fields?</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Does writing the formula update the form fields?</td>
<td>X</td>
<td></td>
<td>No variables are recognised.</td>
</tr>
<tr>
<td>Do all the required fields request inputs?</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do all the other errors trigger as intended?</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3: Results after the second functional test for the create question screen
5.2. Empirical Evaluation

5.2.1 Design

Task. The description of each task offered a detailed list of the steps to follow in order to complete the tasks.

The last part of the questionnaire (Section 7) contains questions related to the general user experience with our system. We asked our testers to rate the overall user experience (on a Likert scale from 1 to 5), if they encountered any major issues, what they feel could be improved and whether they would switch to our system as a solution for online assessments. We considered that it is beneficial to allow users to share any other information they felt important.

We selected our testers in a similar manner to the initial questionnaire, from the teams acquaintances or workmates who were known to have a background in education. We made sure that all the answers were completely anonymous and no personal data of any kind was collected. Also at the end of the testing period the new accounts and all the assessments and questions were removed from our database.

5.2.2 Results & updates

At the end of our testing period we have accumulated 21 answers, this is an improvement compared to the number of responses we had for our preliminary research survey but still lower than the number of testers who confirmed us they would participate. Overall, the results are promising with some valuable feedback provided by a few testers.

Our testers were in proportion of 81% educators, with the rest sharing their views on the platform as students. 95% of testers were part of Higher education, with the rest representing Further Education. An interesting and fortunate result is the average amount of interaction testers had with online assessment platforms. Our testers were split almost perfectly between, multiple times a week, once a week, a few times every term and once or twice a year, offering us insight from users with a varied range of experience with online assessments (see Figure 5.1). The platform our testers were familiar with the most was Blackboard, followed by Moodle and Aula with 52%, 33% and 19% of respondents, respectively. We have not seen any users of platforms dedicated only to assessments, besides the 3 responses from testers who have used only the proprietary University platform for online assessments.

57% of our testers have chosen to create a new account and 79% of them have rated the
5.2. Empirical Evaluation

Figure 5.2: Tester difficulty rating for creating a new assessment

process as being very easy to complete. With additional feedback from users that this is a very similar experience with creating accounts on other websites, which shows us that this part of our system is very intuitive to use accounting for past experience. It is important to note that none of the testers have rated their experience as difficult or very difficult.

All but one of our testers managed to create a new assessment. The user who did not manage to complete the assessment creation process encountered an error upon clicking on the create assessment button. This issue was caused by a routing error in the code that was corrected immediately after the problem was reported. 86% of our testers have found the difficulty level below the average for such a task (see Figure 5.2). This result showed us that the lightweight interface is working as intended, not overwhelming users.

All our testers managed to successfully create the first question. Almost half of our testers (43%) have chosen to create a Fill in the blank question. The popularity of this question surprised us as we were under the impression that multiple choice questions were by far the most popular question type for an online assessment (see Figure 5.3). 15% of testers chose to create either a regular expression or formula with random variables question which shows us that there is interest for this special type of question even if they are not transferred directly from physical assessments and only available in the digital world. 95% of our testers have found the difficulty level below the average for such a task. This surprised us as the creation of questions is by far the most complex task present in our application, but nevertheless, an exceptional result.

Once again, all our testers managed to successfully complete the given task, creating a second question. This time the users were restricted to choose one of the two special types of questions, regular expression or formula with random variables. There was a slight preference of regular expression over formula type questions, 53% of respondents choosing to create this type of question as their second question. This result is not surprising taking into account that fill in the blank has proven to be the most popular type of question and regular expression questions are an evolved version of them. The rated difficulty level is more concentrated towards the middle of the spectrum, with only 15% of respondents finding the task very easy to complete, but yet no tester has
5.2. Empirical Evaluation

found the process to be difficult (see Figure 5.4). This result shows us that the increased complexity of creating this special type of questions is affecting the ease of use of the system but not enough for users to find it difficult to use.

Submitting an attempt was the last task we had planned for our testers. All but one of them managed to complete the task without any issues. The one instance when the submission was not complete was caused by an error we had within our controller that caused the marking algorithm to malfunction and thus giving an error upon submission. Luckily, this error was reported early on and we managed to fix the problem before any other tester encountered it. The difficulty rating offered by our users was low for 95% of them with no tester finding it difficult to submit an attempt (see Figure 5.5), further highlighting the effects of a lightweight user interface.

The final section offered us an insight on how the users feel about the application overall. 57% of testers found the application to be easy to use while 33% of them found it very easy to use and 10% considered it was neither easy or difficult to use. No testers have rated their experience as difficult. We are glad to see this results as they prove that all the measures we have taken
As part of our user testing, we gathered some valuable feedback from our testers. Part of this served as confirmation of the usefulness of our design choices while the rest helped us improve some areas of the application.

Some users reported an account creation experience that was similar to most other websites, which makes their current knowledge applicable for our system too, users not having to understand a new process for achieving the same goal. Other users appreciated the inclusion of a date picker
as it makes choosing the availability dates and times for assessments a lot more intuitive.

Regarding questions some users liked the extra options available for the questions, such as case sensitiveness for fill in the blank questions or the partial marking for multiple answer questions. For the latter type of question one tester mentioned the usefulness of the automated split of value between all the correct answers. When prompted to create a question of a special type testers have appreciated the possibility to have true random variables used in formula type questions, some have particularly liked the possibility to define constraints and relations between variables as this makes the use cases of formula type questions much broader. Some of the testers who have created a regular expression question valued the inclusion of the expression checker and found the cheat sheet included in it useful.

For the submission of attempts users have appreciated the format the results are sent in, these are sent as a pre-filled read-only form that matches the form shown when submitting the attempt including the score for each question and the automated feedback, making it easy to follow compared to receiving the results in a tabular form or even only the final score.

As mentioned before, there was some feedback from our testers that helped us improve our implementation. The most requested change was the removal of the UTC timezone across the whole application and showing the time in the user’s timezone instead. This led us to making the change to the user’s timezone. This is achieved by using JavaScript to detect the user timezone and convert the UTC time and date stored in the database to suit. The same mechanism is used when dates and times are set, these are input in the user’s time zone and converted to UTC before saving. This change removes the confusion created by each user having to convert their time to UTC in order to make sense of the displayed time. Even though in the design phase we considered the use of a universal time zone to be beneficial, we have been proven that this was not a great idea and have updated our design to match user expectation.

Another suggestion we got from a considerable amount of testers was to offer some kind of example questions as these would make the question creation process clearer. We decided to include an example assessment with one of each type of question and 2 example submissions to each new account. This way, when a new user joins the platform they are greeted by this on their assessment page so that they can form a quick idea on how assessments as a whole are set up. Also, as part of the question creation process we received some feedback noting that some of the option fields might benefit from some more clarification in the way they work. This was addressed by including more explanation, in the form of subtext, for each of the available question options.

A piece of general feedback for the platform coming from users who have used platforms such as Blackboard or Moodle stated that they would be more willing to adopt this new platform if it was to include more functionality related to the whole teaching process, similar to their current platform. We will discuss this in more detail in Section 6.1. With a general consensus that the application represents a nice concept of a more approachable assessment platform that could only grow by the inclusion of course support.

The questionnaire results are available at https://drive.google.com/file/d/19aS4aprCmF-82L4LwEsefRWMVR-ooAAc/view?usp=sharing.
Chapter 6

Future Work & Conclusion

6.1 Future work

Even from the inception stage of the project, we had many ideas of features to include on the platform in order to make educators’ experience better and lighten the load of their work. Unfortunately, due to having a rather small development period for the project we had to set our goal to offer the most function we could in this time-frame while still building a high quality application. Thus, we have limited our application to the management of online assessments.

During the inception stage, we believed that the addition of offline assessments would be beneficial for our system. Starting from the classic approach where a student is given a task in the form of a PDF file and then they have a deadline to upload their solution and extend it to the point where an educator could generate a PDF file of an assessment they have created on our platform and use it as the task to complete. Until half way into the development process, we thought we could include this in our software but the difficulties we faced developing the core of our system forced us to drop this idea for now. In a similar situation was the idea enabling educators to export their assessment in an XML format so that they can share it with other educators who could import it to their account and end up with the same assessment. We still believe that both these ideas have great potential and would benefit educators if they will be implemented in the future.

If we were to further develop the application we would consider adding more types of questions, such as matching or sequence questions. We would like to also refine the way the current questions work, allowing educators to choose how many answer options they want to have for a multiple choice or multiple answer question, allow multiple blank spaces to be part of one fill in the blank question, and most importantly, researching a better way to generate the random variables as our current approach is not generating values varied enough for our linking, even though it works reasonably well. In our tests, only 6-7 values being used for a variable with 10 valid value options. We also think that an update to the expression builder would make more educators include this type of question in their assessment. This update would include a drag and drop function for regular expression elements that would take advantage of the modular nature of regular expressions and transform the expression builder into an experience similar to building with LEGO.

Towards the end of the project we have accumulated even more ideas of how the system could be improved. We thought that creating accounts for students would benefit their experience as well. This would enable them to have a log of all their attempts where they could review
their results. Students having accounts could benefit educators too as they could create groups of students, sign them up for an assessment and students would be notified automatically as soon as it becomes available. Growing this idea further, as mentioned in the feedback of a few testers, the system could extend as a course hosting service, where educators could also upload lectures or practicals and make them available to students. This transformation would take the system to the next level and could become a competitor for systems such as Blackboard or Moodle.

Even with the current limitations of our system we truly believe that if has great potential and hope that it could be used as a base to build upon or just inspiration for future initiatives of helping educators form the next generations.

The application is its current state is available at https://la-test-creator.herokuapp.com/. A test account has been created and is accessible with the following credentials: Email: fp.test.creator@gmail.com, Password: Pass1234. The GitHub repository of this application is available at https://github.com/Loghin85/TestCreator.

6.2 Conclusion

In conclusion, the project has proven to be a success, achieving its goal of providing a better environment for educators to create assessments developed with their needs in mind. Test creator fulfills all the requirements specified in the inception stage of the project. The current state of the application serves as a base for a more complex learning management system. The application has also been improved from the design phase, now offering new features such as providing users with a regular expression checker/builder, some based on the feedback provided during the empirical evaluation of our system.

As we aimed to create an online assessment platform that offers a more pleasant experience for educators their input was mandatory. The preliminary questionnaire carried during the design stage of our application helped us understand what educators look for in an application meant for assessment management. This, together with the testing of other similar platforms helped us build the list of requirements for this system.

We have identified three areas that need major improvement, assessment personalizing for students, user interface and the lack of individual feedback to students. The first issue was addressed by introducing true random variable questions. These make use of Integer Linear Programming to interpret the question and generate individual variable values to each student. To overcome the second issue we have made a careful selection of the displayed information and positioned the display elements in places where one would expect to find them, resulting in a lightweight interface that is easy to follow and navigate. The last issue was solved by the inclusion of a further feedback feature in the form of a text area placed on the view screen of a submission that once filled is sent to the student by email, enabling educators to offer students all the feedback they feel necessary besides the default automatically generated feedback.

The development process posed many challenges ranging from time management to library incompatibility. Most problems were resolved with minimal effort but there were issues that required up to a few weeks to resolve. The lesson to be learnt from this is to always allocate more time than one thinks is necessary for the completion of a task. It is also important to consider that there might be issues that are out of ones capability to solve, such as the general framework error
that halted all development with Ruby on Rails for a few days that happened sometime around the mid point of the project.

Evaluation represented a main area of our project, unit and functional testing being a crucial part of the development process but most importantly the feedback from the empirical evaluation contributed towards creating a better experience for educators. The test group was assembled from educators with different backgrounds and levels of expertise with online assessments. This diversity within the group offered us feedback from many different points of view, helping us develop a platform that suits the needs of educators ranging from inexperienced to proficient.

If we were to start the project again, we would like to include educators more into the design process of the interface and main functions of the application as their input was valuable but given our scheduling of the empirical evaluation the time available to update our application using their feedback was limited, only offering us enough time to analyse and implement part of the suggestions.

This project gave us an insight of what it takes to design and create an assessment, all educators having to create them on a regular basis. The issues they faced were in some cases surprising but most of the time expected. The completion of this project imposed us a greater appreciation towards the efforts educators make in order to form the new generations to the best of their abilities.
Appendix A

User Manual

This document contains detailed instructions on how to complete all the major tasks within Test Creator. These include, managing an account, managing an assessment, and submitting an attempt, each with their respective sub-tasks.

A.1 Managing an Account

A.1.1 Create an account

To create an account press on the “Sign up” button from the top right corner of the screen. If you are logged in into an account you will have to log out first. Fill out the form with your details. On form submission an activation email will be sent to your email. Follow the link in the activation email you have received to activate your account. You will not be able to access your account until it has been activated.

A.1.2 Access an account

To access an account press on the “Log in” button from the top right corner of the screen. If you are logged in into an account you will have to log out first. If you have not done so already follow
the link in the activation email you have received to activate your account. You will not be able to access your account until it has been activated.

**A.1.3 Log out from an account**

To log out from an account press on the downward arrow next to your name from the top right corner of the screen. In the sub-menu press on “Log out”

**A.1.4 Review account details**

To review your account details press on the downward arrow next to your name from the top right corner of the screen. In the sub-menu press on “My profile”.

---

**Figure A.2:** Access an account

**Figure A.3:** Log out from an account
A.1.5 Edit account details

To edit your account details press on the downward arrow next to your name from the top right corner of the screen. In the sub-menu press on “My profile” (see Figure A.5). Next, press on the “Edit” button, fill out the form with the updated details and click “Submit”(see Figure A.6).
A.1.6 Remove an account

To remove your account press on the downward arrow next to your name from the top right corner of the screen. In the sub-menu press on “My profile”. Next, press on the “Remove” button and confirm your action in the prompt.

Please note, removing your account will erase all the assessments, questions and submissions linked to your assessments.

A.2 Managing an Assessment

Managing assessments is available for registered users only, before following any of the instructions below please log in.
A.2.1 Create an assessment

To create an assessment click on the “Assessments” button in the top menu. Then, press the “Add assessment” button located at the bottom of the table (see Figure A.8). Fill out the assessment title and description. Choose an assessment duration and set the start and end dates and times using the date-time picker. Once this is done, press on “Create Assessment” (see Figure A.9).

Figure A.8: Create an assessment (part I)

Figure A.9: Create an assessment (part II)

A.2.2 Review assessment details

To review the assessment details click on the “Assessments” button in the top menu. Then, click on the eye icon corresponding to the assessment you want to review.
A.2.3 Edit assessment details

To edit assessment details click on the “Assessments” button in the top menu. Then, click on the pencil icon corresponding to the assessment you want to edit. Alternatively, navigate to the review page of assessment you want to edit (see Section A.2.2). Then, press on the “Edit Assessment details” button at the bottom of the screen.
A.2.4 Remove an assessment

To remove an assessment click on the “Assessments” button in the top menu. Then, click on the bin icon corresponding to the assessment you want to remove and confirm your action in the prompt. Alternatively, navigate to the review page of assessment you want to remove (see Section A.2.2). Then, press on the “Delete Assessment” button at the bottom of the screen and confirm your action in the prompt.
A.2. MANAGING AN ASSESSMENT

A.2.5 Create a question

To create a question navigate to the review page of assessment that should contain the new question (see Section A.2.2). Next, press the “Add question” button located at the bottom of the questions table. Fill out the question title type and text. Once a type is chosen the options and answer fields will update according to your choice, refer to the subsections below for more details. Once the options and answer are set, set the marks awarded for this question and fill out the feedback. To finish, click on the “Create Question” button located at the bottom of the screen.

Multiple choice question

Multiple choice questions offer the option of negative marking. They require four choices to be defined with only one of them chosen as correct.
Multiple answer question

Multiple answer questions offer the options of negative marking and partial marking. They require four choices to be defined with any number of them chosen as correct.

Fill in the blank question

Fill in the blank questions offer the options of negative marking, case sensitiveness, allow multiple spaces and allow answers containing the correct answer. They require an answer that is considered correct. The position of the blank space is denoted by “[blank]” within the question text. Only one blank space is accepted.
A.2. MANAGING AN ASSESSMENT

True / false question

True / false questions offer the option of negative marking. The correct answer must be chosen between true and false.

Formula with random variable(s) question

Formula with random variable(s) questions offer the option of negative marking and answer error. They require a formula and variable value intervals. Further constraints and relations between variables can be defined. Variables are denoted by strings of capital letters contained between square parenthesis (e.g. [X]). As the formula is written the variables are recognised and variable interval fields will appear below. The position of the variables is denoted in the same way as for formulas (e.g. What is the sum of [X] and [Y]?). Defined relations must respect the following rules:

- The relations defined should only contain references to the variables defined in the formula.
Only "\leq", "\geq" and "\neq" are accepted as comparison operators and to their right side only one number is accepted i.e. $2\times[X]\leq4$ is accepted while $2\times[X]\leq2\times[Y]$ is not.

All operations must have an operator present i.e. $2[X]$ is not valid and should be written as $2\times[X]$.

There must be no space within a relation and no parentheses are allowed.

Relations are separated by comma ",".

**Figure A.20:** Formula with random variable(s) question form

**Regular expression question**

Regular expression questions offer the option of negative marking. They require a regular expression that accepts the correct answers to be defined. The builder/checker can be used for easier configuration. The position of the blank space is denoted by “[blank]” within the question text. Only one blank space is accepted.

**Figure A.21:** Regular expression question form
A.2.6 Review question details

To review question details navigate to the review page of assessment that contains the question (see Section A.2.2). Next, click on the eye icon corresponding to the question you want to review.

![Figure A.22: Review question details](image)

A.2.7 Edit a question

To edit a question navigate to the review page of assessment that contains the question (see Section A.2.2). Next, click on the pencil icon corresponding to the question you want to edit. Alternatively, click on the eye icon corresponding to the question you want to edit. Then, press on the “Edit” button at the bottom of the screen.

![Figure A.23: Edit a question (method I)](image)
A.2.8 Remove a question

To remove a question click on the “Assessments” button in the top menu. Then, click on the eye icon corresponding to the assessment that contains the question. Next, click on the bin icon corresponding to the question you want to remove and confirm your action in the prompt. Alternatively, click on the eye icon corresponding to the question you want to remove. Then, press on the “Delete” button at the bottom of the screen and confirm your action in the prompt.
A.2.9 Share an assessment

To share an assessment navigate to the review page of assessment (see Section A.2.2). Copy the submission link found in the details section of the page. Share this link with your students.

A.2.10 Review a submission

To review a submission navigate to the review page of assessment that should contain the submission (see Section A.2.2). Next, click on the eye icon corresponding to the submission you want to review.
A.2.11 Remove a submission

To remove a submission navigate to the review page of assessment that should contain the submission (see Section A.2.2). Next, click on the bin icon corresponding to the submission you want to remove and confirm your action in the prompt.
A.3 Submit an Assessment Attempt

To submit an attempt follow the URL provided by your educator. If the assessment is not yet available please return after the availability windows as begun. Fill out your Student ID and email address and press “Start”. To submit your attempt press “Submit”.

Please note, at the end the allocated time the test will be automatically submitted. You have only one attempt, once started you will not be able to start again or finish the test later. Please make sure that that your attempt has been submitted before closing the tab, else a 0 mark attempt will be submitted.
Appendix B

Maintenance Manual

This document includes instructions on how to run the system, software dependencies, a list of system files and future adaptations and changes.

B.1 Building and Running the System

The application can be run locally from the source code (see Section B.1.1) or added to a GitHub repository and deployed to Heroku (see Section B.1.2).

B.1.1 Local installation

Running the application locally requires prerequisites to be installed, these include:

- Ruby 1
- PostgreSQL 2
- Node.js 3
- Yarn 4
- The Ubuntu coinor-libcbc-dev package 5

Please refer to section B.2 for the exact versions used in this project.

Once these are installed on your machine Rails, Bundler and the required gems can be installed using the command sequence:

$ gem install rails
$ gem install bundler
$ bundle install

Then the database can be started and set up using the following sequence of commands:

$ sudo service postgresql start
$ bundle exec rails db:setup

Lastly the application can be started using:

1https://www.ruby-lang.org/en/documentation/installation/
2https://www.postgresql.org/download/linux/ubuntu/
4https://classic.yarnpkg.com/en/docs/install/#windows-stable
5https://packages.ubuntu.com/bionic/coinor-libcbc-dev
B.2. Software Dependencies

$ bundle exec rails s

Now the application should be available at http://localhost:3000/.

B.1.2 Heroku deployment

For this approach Heroku and GitHub accounts are required. Once these are set up, for the original project repository available at https://github.com/Loghin85/TestCreator. Create a new Heroku project, and link the new GitHub repository to it, set two new environment variables, called GMAIL_USERNAME and GMAIL_PASSWORD to the login details of an Gmail account you want to use for sending out emails. Add a new buildpack pointing the following address: https://github.com/Loghin85/heroku-buildpack-apt and make sure this is placed before the heroku/ruby buildpack. Once the database has been migrated the application is ready to deploy.

B.2 Software Dependencies

The application has been developed and tested using Ruby v3.0, Rails v6.1.3.2, PostgreSQL v12.6, Node.js v12.21, Yarn v1.22.5 and coinor-libcbc-dev v2.10.3+repack1-1build1. Different versions of these software may work with the application but in some case some adjustment might be needed.

B.3 List of System Files

Table B.1 includes a list of all the files that were modified from the default project files generated by Rails.

<table>
<thead>
<tr>
<th>Package</th>
<th>Filename</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>Gemfile</td>
<td>Contains a list of the gems used for this application</td>
</tr>
<tr>
<td>config</td>
<td>routes.rb</td>
<td>Contains the valid paths within the application.</td>
</tr>
<tr>
<td>config.locales</td>
<td>en.yml</td>
<td>Contains the error messages to be displayed</td>
</tr>
<tr>
<td>db</td>
<td>schema.rb</td>
<td>Contains the structure of the database</td>
</tr>
<tr>
<td>test</td>
<td>fixtures</td>
<td>Contains the fixtures used for unit testing</td>
</tr>
<tr>
<td>test</td>
<td>controller</td>
<td>Contains the unit tests to be run for controllers</td>
</tr>
<tr>
<td>app.controllers</td>
<td>account_activation_controller</td>
<td>Contains the controller methods for implementing account activation</td>
</tr>
<tr>
<td>app.controllers</td>
<td>application_controller</td>
<td>Contains the main controller methods</td>
</tr>
<tr>
<td>app.controllers</td>
<td>assessments_controller</td>
<td>Contains the controller methods for implementing assessments</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>app.controllers</td>
<td>password_resets_controller</td>
<td>Contains the controller methods for implementing password resets</td>
</tr>
<tr>
<td>app.controllers</td>
<td>questions_controller</td>
<td>Contains the controller methods for implementing questions</td>
</tr>
<tr>
<td>app.controllers</td>
<td>sessions_controller</td>
<td>Contains the controller methods for implementing sessions</td>
</tr>
<tr>
<td>app.controllers</td>
<td>submissions_controller</td>
<td>Contains the controller methods for implementing submissions</td>
</tr>
<tr>
<td>app.controllers</td>
<td>test_controller</td>
<td>Contains the controller methods for the home page</td>
</tr>
<tr>
<td>app.controllers</td>
<td>users_controller</td>
<td>Contains the controller methods for implementing user accounts</td>
</tr>
<tr>
<td>app.helpers</td>
<td>application_helper</td>
<td>Contains the general helper methods</td>
</tr>
<tr>
<td>app.helpers</td>
<td>assessments_helper</td>
<td>Contains the helper methods used for implementing assessments</td>
</tr>
<tr>
<td>app.helpers</td>
<td>questions_helper</td>
<td>Contains the helper methods used for implementing questions</td>
</tr>
<tr>
<td>app.helpers</td>
<td>sessions_helper</td>
<td>Contains the helper methods used for implementing sessions</td>
</tr>
<tr>
<td>app.helpers</td>
<td>submissions_helper</td>
<td>Contains the helper methods used for implementing submissions</td>
</tr>
<tr>
<td>app.mailers</td>
<td>application_mailer</td>
<td>Contains the general methods responsible for sending emails</td>
</tr>
<tr>
<td>app.mailers</td>
<td>submission_mailer</td>
<td>Contains the methods responsible for sending emails related to submissions</td>
</tr>
<tr>
<td>Path</td>
<td>File/Class</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>app.mailers</td>
<td>user_mailer</td>
<td>Contains the methods responsible for sending emails related to user accounts</td>
</tr>
<tr>
<td>app.models</td>
<td>application_record</td>
<td>Contains general rules and methods responsible for processing and validating the data</td>
</tr>
<tr>
<td>app.models</td>
<td>assessment</td>
<td>Contains rules and methods responsible for processing and validating the data used within assessments</td>
</tr>
<tr>
<td>app.models</td>
<td>question</td>
<td>Contains rules and methods responsible for processing and validating the data used within questions</td>
</tr>
<tr>
<td>app.models</td>
<td>submission</td>
<td>Contains rules and methods responsible for processing and validating the data used within submissions</td>
</tr>
<tr>
<td>app.models</td>
<td>user</td>
<td>Contains rules and methods responsible for processing and validating the data used within user accounts</td>
</tr>
<tr>
<td>app.views.assessments</td>
<td>_form.html.erb</td>
<td>Contains the structure of the UI used for the edit and create forms of the assessments</td>
</tr>
<tr>
<td>app.views.assessments</td>
<td>edit.html.erb</td>
<td>Contains the structure of the UI used for editing assessments</td>
</tr>
<tr>
<td>app.views.assessments</td>
<td>index.html.erb</td>
<td>Contains the structure of the UI used for displaying all assessments</td>
</tr>
<tr>
<td>app.views.assessments</td>
<td>new.html.erb</td>
<td>Contains the structure of the UI used for creating new assessments</td>
</tr>
<tr>
<td>app.views.assessments</td>
<td>show.html.erb</td>
<td>Contains the structure of the UI used for displaying an assessment</td>
</tr>
<tr>
<td>app.views.layouts</td>
<td>_menu.html.erb</td>
<td>Contains the structure of the UI used for the application menu</td>
</tr>
<tr>
<td>Directory</td>
<td>File Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>app.views.layouts</td>
<td>application.html.erb</td>
<td>Contains the layout of the UI used for the application</td>
</tr>
<tr>
<td>app.views.layouts</td>
<td>mailer.html.erb</td>
<td>Contains the structure of the UI used for emails</td>
</tr>
<tr>
<td>app.views.password_resets</td>
<td>edit.html.erb</td>
<td>Contains the structure of the UI used for updating a user account password</td>
</tr>
<tr>
<td>app.views.password_resets</td>
<td>new.html.erb</td>
<td>Contains the structure of the UI used for requesting to update a user account password</td>
</tr>
<tr>
<td>app.views.questions</td>
<td>_form.html.erb</td>
<td>Contains the structure of the UI used for the edit and create forms of the questions</td>
</tr>
<tr>
<td>app.views.questions</td>
<td>edit.html.erb</td>
<td>Contains the structure of the UI used for updating a question</td>
</tr>
<tr>
<td>app.views.questions</td>
<td>new.html.erb</td>
<td>Contains the structure of the UI used for creating a new question</td>
</tr>
<tr>
<td>app.views.questions</td>
<td>show.html.erb</td>
<td>Contains the structure of the UI used for displaying a question</td>
</tr>
<tr>
<td>app.views.submission_mailer</td>
<td>assessment_feedback.html.erb</td>
<td>Contains the structure of the UI used for creating a feedback email</td>
</tr>
<tr>
<td>app.views.submission_mailer</td>
<td>assessment_results.html.erb</td>
<td>Contains the structure of the UI used for creating a results email</td>
</tr>
<tr>
<td>app.views.submissions</td>
<td>edit.html.erb</td>
<td>Contains the structure of the UI used for submitting the answers for a submission</td>
</tr>
<tr>
<td>app.views.submissions</td>
<td>new.html.erb</td>
<td>Contains the structure of the UI used for identifying students for a submission</td>
</tr>
<tr>
<td>app.views.submissions</td>
<td>show.html.erb</td>
<td>Contains the structure of the UI used for displaying a submission</td>
</tr>
<tr>
<td>app.views.submissions</td>
<td>duplicate.html.erb</td>
<td>Contains the structure of the UI used for informing users that they have attempted the assessment already</td>
</tr>
</tbody>
</table>
Table B.1: List of modified system files

<table>
<thead>
<tr>
<th>app.views.submissions</th>
<th>received.html.erb</th>
<th>Contains the structure of the UI used for informing users that the submission has been recorded</th>
</tr>
</thead>
<tbody>
<tr>
<td>app.views.test</td>
<td>contact.html.erb</td>
<td>Contains the structure of the UI used for the contact page</td>
</tr>
<tr>
<td>app.views.test</td>
<td>faq.html.erb</td>
<td>Contains the structure of the UI used for the frequently asked question page</td>
</tr>
<tr>
<td>app.views.test</td>
<td>index.html.erb</td>
<td>Contains the structure of the UI used for the home page</td>
</tr>
<tr>
<td>app.views.user_mailer</td>
<td>account_activation.html.erb</td>
<td>Contains the structure of the UI used for creating an account activation email</td>
</tr>
<tr>
<td>app.views.user_mailer</td>
<td>password_reset.html.erb</td>
<td>Contains the structure of the UI used for creating a password reset email</td>
</tr>
<tr>
<td>app.views.users</td>
<td>_form.html.erb</td>
<td>Contains the structure of the UI used for the edit and create forms of the user accounts</td>
</tr>
<tr>
<td>app.views.users</td>
<td>edit.html.erb</td>
<td>Contains the structure of the UI used for updating a user account</td>
</tr>
<tr>
<td>app.views.users</td>
<td>new.html.erb</td>
<td>Contains the structure of the UI used for creating a new user account</td>
</tr>
<tr>
<td>app.views.users</td>
<td>show.html.erb</td>
<td>Contains the structure of the UI used for displaying a user account</td>
</tr>
</tbody>
</table>

B.4 Future Adaptations and Changes

The application in its current form could be used as is or as a module in a more complex Learning Management System.

We plan to improve the application by refining the application functionality, especially, the number of answer options for multiple choice and multiple answer questions, the variable value generation algorithm for formula questions and improving the regular expression builder by using a drag and drop like feature for regular expression elements.
Bibliography


discrete mathematics and optimization.

