

SECURE WEB-BASED EXAM PAPER GENERATION SYSTEM

By
Ashley Holmes

Supervisor: Mr. Stephen Sheridan

SUBMITTED IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
BSC (HONS) IN COMPUTING
AT
INSTITUTE OF TECHNOLOGY BLANCHARDSTOWN
DUBLIN, IRELAND
23 OCTOBER 2016

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of **BSc (Hons) in Computing** in the Institute of Technology Blanchardstown, is entirely my own work except where otherwise stated, and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfilment of the requirements of that stated above.

Dated: 23 October 2016

Author:

Ashley Holmes

Abstract

The main goal of this project will be to develop a secure web-based exam paper generation system. The system should support multiple users, subjects and courses. The end product should allow for the generation of pools of questions within a topic area and should also be able to generate exam papers based on a random selection of questions within a given pool or subject area.

The system should store historical data about which questions were used in what year and should avoid using the same question year on year. The system should also allow the user to have the ultimate say in what questions are chosen by allowing randomly selected questions to be replaced with other questions from with the same pool.

The system should also provide the ability to link marking schemes with questions and should ensure marking schemes are correct and add up to the correct totals. The system should be entirely secure from the login process to the storage of data and data in transit.

Keywords: Web-frameworks, Security, Encryption, Databases, HTML, CSS, Java

Motivation: With the growth of mature students entering third level educational institutions around Ireland. Mainly due to the economic downturn which has resulted in many working class people finding themselves unemployed.

It is for this reason why programmes were put into place by the Irish Government and European Union to establish a structure by investing in these peoples education and skills and thus offering them support when it comes to seeking employment once again. Either in fields which they are familiar with or something completely different.

It is initiatives like these which is supported and funded by authorities and governing bodies such as SUSI and the ESF Ireland which have lead to bringing the unemployment rate of persons aged 25 - 74 years down in the last two years by 2.7% CSO (2016) ESF (2016).

The difference between the total number of people now employed since September 2014 and September 2016 is 50, 200 persons. That is a great deal of people in terms of a two year period. These people attending the institutions could have not seen a classroom for more than twenty to thirty or up to forty years or more let alone used a computer and thus require a greater deal of help or attention. As some of them might be exploring an IT field.

It is this one on one attention that they desperately need and freeing up a lecturers time for this would be of great benefit. Since being exposed to college life and the reality of being a mature student have witnessed this firsthand. Most lecturers teach numerous subjects. And for each subject they need to put together numerous exam papers to account for semesters and repeats. This could total many hours of composition for their many subjects.

If a system was in place to automatically generate exam papers this will take away from the time that it takes to compose them. They would be able to give this time back to their students. In the form of a meeting for Q & A or for revision work. This is the impact that an exam paper generation system could have on the lives of mature students if successful.

Problem statement: Compiling an exam paper which will test the students knowledge in a particular subject is challenging in its own way. Firstly, there are the time constraints. Examiners a being faced with more and more work each year within the same space of time. If you add up the amount of time per semester over a given year which is set aside to put together an exam paper it will add up to many hours.

There needs to be a better approach to minimise these hours. It will take a great deal of time to produce a good quality paper. The questions need to be taken from the curriculum which was or is being delivered to the students over the semester. This brings upon the need to develop a paper from as many of the important areas of the module as possible. As this will be the process of determining the students

performance with regard to the questions which are asked and the complexity of them. The result of good exam question will determine the sort of student the college will produce.

Approach: The approach towards this project will be to do as much research as possible around the work of others. Reading the research papers which are available on the internet and in journals.

Which technologies and methodologies were used and incorporated into their work. If they had any shortcomings. See which improvements can be made. One can streamline a complicated version if one is available. And combining the methods of others to form one which is more successful. Furthermore, how long ago their work took place as perhaps a technology has caught up to what they were trying to achieve and would now be in a position to overcome any weaknesses.

For the purpose of this project the SDLC of choice will be the Agile Model. Which takes measures such as planning, analysis, designing building and testing. This will be done in small increments. Each time building on what is currently available and adding to it. Until all features are in place for a full system release.

Results: A conclusion has yet to be established. The results should represent a system which is better than the one in place offering a defined gain to the current method for examination compilation. This will be determined once the system is in place and can be tested in a scientific manner by comparing the two methods.

Conclusions: To follow.

Acknowledgements

I would like to thank my wife Jennifer Holmes and Mr. Stephen Sheridan...

Table of Contents

Abstract	ii
Acknowledgements	v
Table of Contents	vi
List of Tables	vii
List of Figures	viii
Abbreviations	ix
1 Introduction and Background	1
1.1 Title	1
1.2 Background	1
1.3 Main Research Question(s)	2
1.4 Justification / Benefits	2
1.5 Feasibility	3
1.6 Proposed Methodologies	3
1.7 Expected Results	3

1.8	Conclusion	5
1.9	Project plan	5
2	Literature Review	7
2.1	Abstract	7
2.2	Literature Review	7
3	Methodology Chapter	12
3.1	Introduction	12
3.1.1	What is Agile?	12
3.1.2	How does it work?	14
3.1.3	How is it different?	15
3.1.4	Agile vs Waterfall	16
3.2	Data Collection Methods	16
3.2.1	Internet Search	16
3.2.2	Supervisor Input	16
3.2.3	Journals in Library	16
3.3	Method of Analysis	16
3.3.1	Formulation of Where to Start	16
3.3.2	Early System Implementation	17
3.3.3	Review of Literature	17
3.4	Summary	17
4	Implementation - "Building the solution"	18
4.1	Introduction	18

4.2	Terminal	18
4.2.1	Command Line Instructions	18
4.3	Browser	23
4.3.1	Deploying the Application	23
4.4	IDE	28
4.4.1	PhpStorm IDE for PHP	28
4.5	Database	32
4.5.1	Creation of User Entity and CRUD	32
4.6	Twig	36
4.6.1	Templates are fine tuned	36
4.7	Forms	39
4.7.1	Understanding FormTypes	39
4.8	Validation	41
4.8.1	Validation of the Forms	41
4.9	Theming	44
4.9.1	Form Theming	44
4.10	Fixtures	45
4.10.1	Fixtures with Faker	45
4.11	Passwords	50
4.11.1	Password Fixtures	50
4.12	Passwords	54
4.12.1	Password Services	54

5	Implementation of the System	56
5.1	Implementation Principles	56
5.1.1	Object-Oriented Approach	56
5.1.2	Design Patterns	56
5.1.3	Choice of Language	56
5.2	Stages of Admin Implementation	56
5.2.1	Login	56
5.2.2	Administration	57
5.2.3	Subsection header 3	57
5.3	Stages of User Implementation	57
5.3.1	Subsection header 1	57
5.3.2	Subsection header 2	57
5.3.3	Subsection header 3	57
5.4	Design	57
5.4.1	Subsection header 1	57
5.4.2	Subsection header 2	57
5.4.3	Subsection header 3	58
6	Testing and Evaluation	59
6.1	Introduction	59
6.2	Tests Conducted	59
6.3	Algorithms	59
6.4	Summary	59

7 Conclusion and Future work	60
7.1 Contributions	60
7.2 Limitations	60
7.3 Future Work	60
7.4 Data Collection	60
Bibliography	61
Appendices	63
A Web Application Login Screen	64
B Web Application Question Entry	65
C Generate a Question	66
D Show questions	67
E Show	68

List of Tables

1.1	Table to represent the Work Breakdown Structure	6
-----	---	---

List of Figures

1.1	Graphical illustration of the Agile Model (tutorialspoint, 2016)	4
1.2	Gantt Chart	5
2.1	Schematic diagram of the system function module (tutorialspoint, 2016) . .	9
2.2	Technology road-map of the system (tutorialspoint, 2016)	9
2.3	Flow chart of the automatic paper generation method (tutorialspoint, 2016)	10
3.1	Graphical illustration of the Agile Model (tutorialspoint, 2016)	13
3.2	Increments of the Agile Model (agile in a nutshell, 2016)	14
3.3	Iterations of the Agile Model (agile in a nutshell, 2016)	14
3.4	Making a list (agile in a nutshell, 2016)	15
4.1	Symfony Documentation	19
4.2	Symfony Installation Setup	19
4.3	Symfony Application Setup	20
4.4	Terminal Window Top	21
4.5	Terminal Window Bottom	21
4.6	Php Server Run	22
4.7	Php Server Start	22
4.8	Symfony Browser	23

4.9 Configuration Checker	24
4.10 Web Debug Toolbar and Profiler Extended	25
4.11 Web Debug Toolbar and Profiler Extended	26
4.12 Web Debug Toolbar and Profiler Extended	27
4.13 Web Debug Toolbar and Profiler	28
4.14 DefaultController	29
4.15 Twig Template in IDE	29
4.16 Adding Bootstrap	30
4.17 Adding Bootstrap	31
4.18 Home Page	31
4.19 Parameters Yaml	32
4.20 Mamp Ports	33
4.21 Entities	34
4.22 User Table	34
4.23 Twig Templates	35
4.24 Form Type	35
4.25 index.html.twig	37
4.26 new.html.twig	38
4.27 show.html.twig	38
4.28 edit.html.twig	39
4.29 UserType.php	40
4.30 Form Label	41
4.31 Form Validation	42
4.32 Entity Validation	42
4.33 Constraints	43

4.34	Form Errors	44
4.35	Form Theming	45
4.36	Button Theming	46
4.37	Form Theming Result	46
4.38	Fixtures	47
4.39	Bundles	48
4.40	Faker	48
4.41	Object Manager Flush	49
4.42	Purge Database	50
4.43	Security.yml	51
4.44	Hash Password	52
4.45	getSalt Method	53
4.46	Password	53
4.47	UserInterface	55
1	Graphical illustration of the Login Screen	64
2	Graphical illustration of the Question Entry	65
3	Graphical illustration of the Generate a Question Paper	66
4	Graphical illustration of the Menu List to view the Questions	67
5	Graphical illustration of the Show list	68

Abbreviations

SUSI	Student Universal Support Ireland
CSO	Central Statistics Office
ESF	European Social Fund
IT	Information Technology
SDLC	Systems Development Life Cycle
REQ	Requirement Analysis
UML	Unified Modeling Language
ERD	Entity Relationship Diagram
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
IDE	Intergrated Development Environment
JSP	JavaServer Pages
BLOB	Binary Large Object
SQL	Structured Query Language
PHP	Hypertext Preprocessor
JAVA	Just Another Vulnerability Announcement
CDN	Content Delivery Network
ORM	Object Relational Mapping

Chapter 1

Introduction and Background

1.1 Title

SECURE WEB-BASED EXAM PAPER GENERATION SYSTEM

1.2 Background

With every new college year and semester, lecturers are faced with the prospect of composing examination papers for the next coming months. Since this can prove to be a very tedious and physically demanding task. More over, it can also be very challenging due to the time consumption and nature of the process for the examiner. The traditional method of composing papers can be automated. Therefore, there exists an opportunity to provide a service to simplify the process.

The use of a Web-based examination paper generation system which makes use of a relational database and database tables to cross-reference the newly created table of randomised questions with the tables from the previous years or semester. Resulting to a non-repeating question sheet.

1.3 Main Research Question(s)

- 1. What will the end user experience be like, or will they prefer the old fashion method to what they are used to?**

Not everyone can adapt to change as well as others. This is why getting used to a new system could take some time. Some folk may even get frustrated to the point where they find the new interface impossible to use. This is not the intention. The project is meant to make the process more streamlined and user friendly. This is why a great deal will be taken in the design of the user interface to make the experience a pleasurable one.

- 2. Could the presence of an automated generation of questions system improve the accuracy of questions over a manual generation?**

There are many factors which can affect a human being's output when given a task. These factors could range from fatigue. Being distracted by a colleague. Or not having the focus needed to complete the task at hand. This is where machines have the advantage over us. Humans suffer from what is called, "Human error." Whereas a machine can produce the same output with precision and repetition. This is why an automated system would work in college environment.

1.4 Justification / Benefits

When it comes to that time of year where lecturers need to set aside the time to create their examination papers for the modules which they deliver. This is where this project will come into its own with the aim of taking the stress out of the procedure and to provide examiners an easy to use means of examination paper compilation. This usability will come from a combination of a clean and simple user interface along with useful tools to create examination papers.

1.5 Feasibility

Since there are numerous examples of this implementation on the internet. This comes from reading research papers from other students in colleges and technical institutions all around the world. Furthermore the prerequisites obtained from this projects supervisor ensures that the project is technically feasible. However, some research needs to be undertaken regarding the security and encryption aspects. This will be the main technical difficulty and therefore there needs to be a sufficient technical understanding of the technologies involved in order to complete the project.

1.6 Proposed Methodologies

To articulate the methods and techniques used in this plan. Below is the outcome after reviewing various SDLC methodologies with reference to tutorialspoint (2016):

- Adoption of the Agile Model.
- Suits the requirements for this project.
- Widely accepted within companies within the IT industry.
- Valuable learning curve in gaining experience with this model.
- Model has the ability to adapt and tailor itself within each increment as the project moves forward.
- Advantageous to the project.

1.7 Expected Results

As noted in the Feasibility section the project should be feasible from a technical standpoint. It is therefore expected that the project will result in a fully-functioning web site that makes

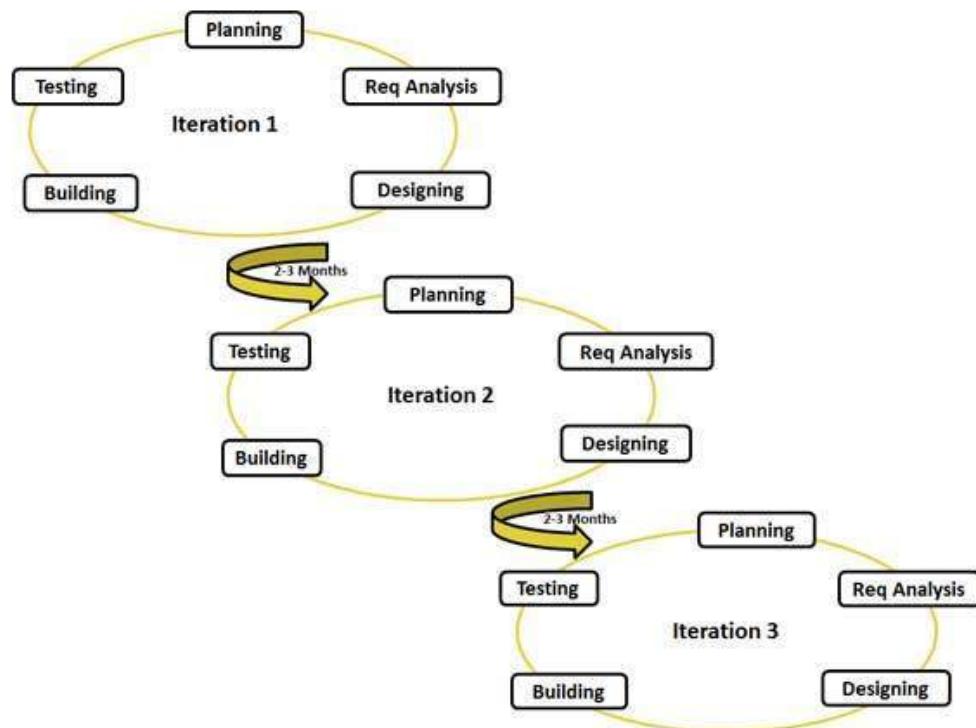


Figure 1.1: Graphical illustration of the Agile Model (tutorialspoint, 2016)

use of the technologies provided.

1.8 Conclusion

This project aims to provide a simple and easy to use service through the use of various Internet technologies combined with automatic generation of question papers and functions. It is hoped that such a service can reduce both the time and difficulties experienced by examiners during a busy time of the year.

1.9 Project plan

Table 1.1 Which shows the Work Breakdown Structure.

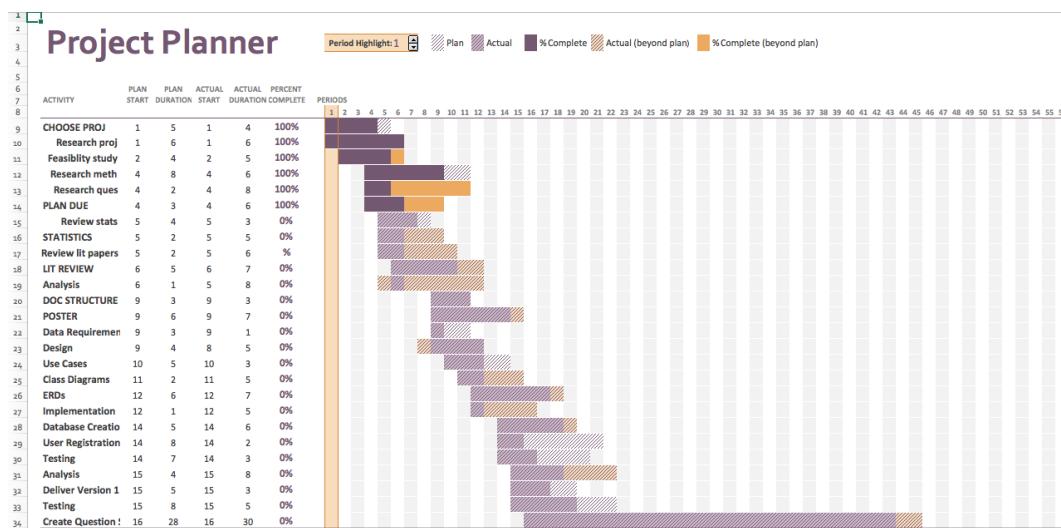


Figure 1.2: Gantt Chart

Task Name	Start	Finish	Duration
Planning	12/09/2016	17/10/2016	42d
Project Plan	12/09/2016	26/09/2016	14d
Research Project Ideas	12/09/2016	26/09/2016	5d
Project Proposal	27/09/2016	12/10/2016	5d
Feasibility Study	12/10/2016	14/10/2016	13d
Research Methodologies	14/09/2016	16/10/2016	7d
Create Project Proposal	16/10/2016	26/10/2016	5d
Submit Project Proposal	26/10/2016	26/10/2016	0d
Literature Review	26/10/2016	31/10/2016	2d
Submit Literature Review	31/10/2016	07/11/2016	0d
Development	07/11/2016	12/12/2016	108d
Version 1	12/12/2016	12/09/2016	28d
Analysis	14/09/2016	30/09/2016	7d
User Registration	14/09/2016	30/09/2016	7d
Question Entry	28/10/2016	12/09/2016	7d
Create Question Section	28/10/2016	12/09/2016	7d
Design	28/10/2016	31/10/2016	7d
Use Cases	28/10/2016	31/10/2016	2d
Class Diagrams	31/10/2016	07/11/2016	2d
ERDs	31/10/2016	07/11/2016	2d
Wireframes	31/10/2016	07/11/2016	1d
Implementation	31/10/2016	12/09/2016	14d
Database Creation	31/10/2016	07/11/2016	2d
Home Page	31/10/2016	07/11/2016	3d
User Registration	31/10/2016	07/11/2016	4d
Question Entry Page	31/10/2016	07/11/2016	4d
Deliver Version 1	07/11/2016	12/12/2016	0d
Testing	07/11/2016	12/12/2016	11d
Review Version 1	07/11/2016	12/12/2016	7d
Analysis	07/11/2016	12/12/2016	7d
View Listings	07/11/2016	12/12/2016	7d
Show Questions	12/12/2016	12/12/2016	7d
Generate Questions	12/12/2016	15/12/2016	7d
Design	12/12/2016	15/12/2016	7d
Use Cases	12/12/2016	15/12/2016	3d
Wireframes	12/12/2016	15/12/2016	2d
ERDs	12/12/2016	15/12/2016	3d
Implementation	09/01/2017	15/12/2016	14d
Database Changes	09/01/2017	15/12/2016	2d
View Questions Page	09/01/2017	09/01/2017	4d
Generate Question Page	09/01/2017	09/01/2017	4d
Testing	09/01/2017	09/01/2017	12
Deliver Version 2	09/01/2017	09/01/2017	7d

Table 1.1: Table to represent the Work Breakdown Structure

Chapter 2

Literature Review

2.1 Abstract

This chapter looks at existing research and development samples undertaken by other students from many countries around the world. These undertakings which have been published were sourced from publishings in academic papers, journal articles and books and gathered together from the major works to form part of the research of this narrow topic as they are in the same field of various implementations of random and automatic examination paper generation.

2.2 Literature Review

(Guang Cen, 2010) presented a method to eliminate (Mumbai, 2016) the tradition of the manual composition of examination papers which would usually rely on the writers own experience and style of question and knowledge. Although great care would be taken to achieve the best possible outcome of questions with traditional methods there was still the problem of a limited scope of topics and a time consideration. This would bring upon the separation between teaching and creating test papers by means of an automated computer system (Yang Yu, 2008). Comprised of JEE the test system includes modules such as user, subject, question, paper and classification management. Included in this is a question entry

and generation module. These modules can be seen in Figure 1 Schematic diagram of the system function module The question entry and generation makes use of browser and server architecture with a connection to a database of questions. Between this layer is a test server and a WWW server making up the middle layer (Chen, 2008). Figure 2 Technology road-map of the system shows a flow chart of the system architecture and the use of the MVC pattern with a JSP view, Java Beans, Servlet Controller and a MySQL database (Liu, 2008). The page layout uses divs and CSS technology. In addition to this is support from JavaScript (R. Johnson, 2004). It is the browser which allows the user to choose the subject which they intend on examining. A question type such as student input and a difficulty level. With all these combined parameters, a paper is generated using the generation algorithm (Wang, 2008). This will then be stored in the test database which can be recalled at any stage through system functions for query, or to update the database and for maintenance. It runs in separated modes for user and administrator use. In the end the final document is processed into a Microsoft Word .doc file for distribution in an exam environment. From Figure 3 Flow chart of the automatic paper generation method it shows how the document is generated.

There are 3 categories which this system falls under. They are, random algorithm based systems, backtracking systems and artificial intelligence and information processing. The first two do not satisfy the specifications (Guiying Deng, 1998). It is the latter which has been improved to avoid the disadvantages of the first and second algorithms. Giving it the ability of searching for questions based on experience and knowledge which guarantees a high standard and quality of examination papers (Hou, 2003). Through using a system with artificial intelligence and information processing the algorithm works quickly and effectively by not selecting a repeated question in a random manner. Questions and answers are separated. It also allowed the user a choice of topics, degree of difficulty, proportion or mark allocation and number of questions per section.

(Yajuan Zhang, 2011) proposed that although the traditional algorithms in a test paper generation system satisfy the requirements of shuffling the questions. Under certain constraints they do not perform as well as others which have been newly adopted. Here follows a

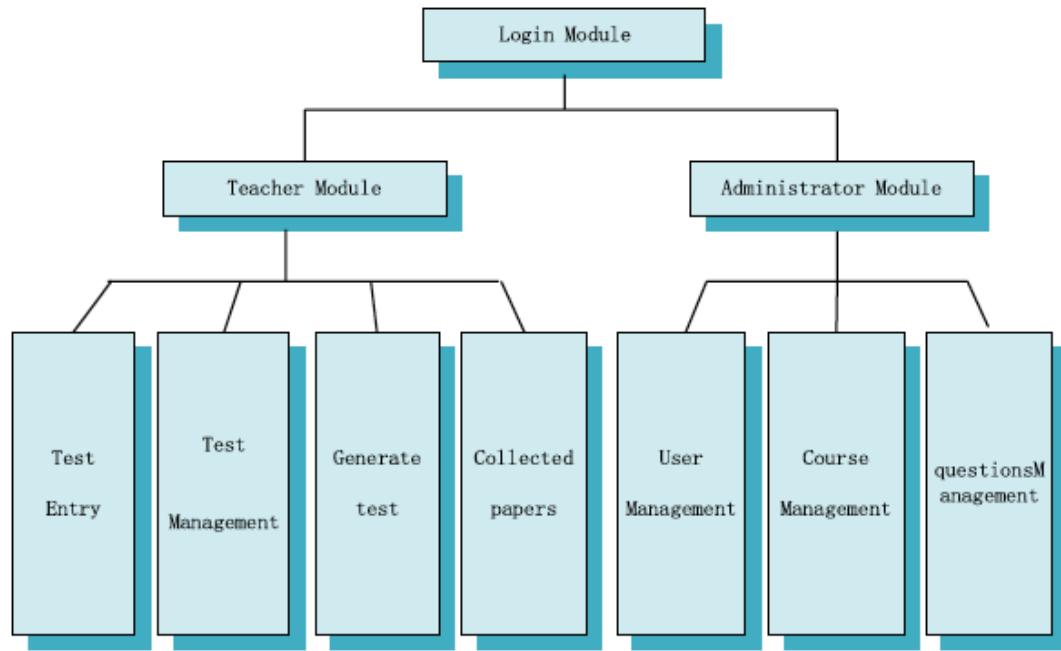


Figure 2.1: Schematic diagram of the system function module (tutorialspoint, 2016)

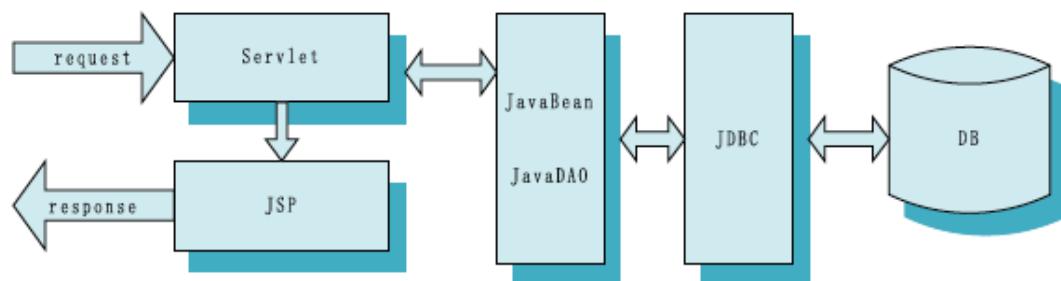


Figure 2.2: Technology road-map of the system (tutorialspoint, 2016)

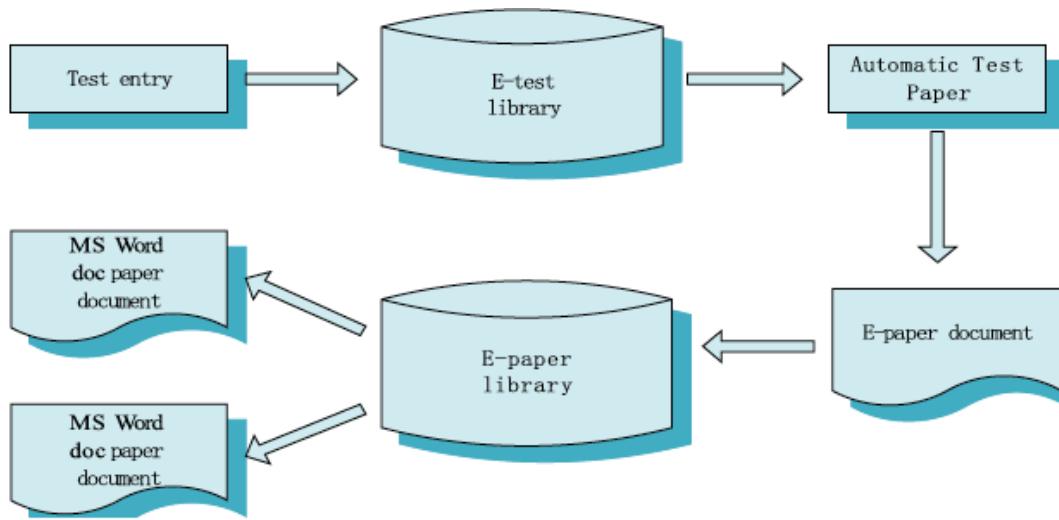


Figure 2.3: Flow chart of the automatic paper generation method (tutorialspoint, 2016)

discussion of analysing five intelligent algorithms and how these existing global optimisation algorithms can be integrated into improved shared global optimisation algorithm and dynamic multi branches tree algorithm. These included, improved genetic algorithm, differential evolution algorithm and ant colony optimisation. The particle swarm optimisation algorithm and simulated annealing algorithm. These are divided into two categories which are population evolutionary and others are individual evolutionary and each uses different searching and selection mechanisms. There have been many different studies on trying to improve the algorithms used in terms of speed optimisation however, the improvements were only minor ones. And with the expansion of the system new classifiers need to be constructed for the added samples. The characteristics of the different global optimisation algorithms such as the improved genetic algorithm. Based on the genetic algorithm with modifications such as improvements to integer coding which displays higher search speeds performing well and is very practical. It also avoids premature which occurs in the genetic algorithm. The genetic algorithm performs a randomised search simulating natural selection and genetic variation to problem solve. With the disadvantage of having a low search efficiency with premature convergence. The differential evolution algorithm is simple and effective in that the population size remains unchanged throughout the operation process. These operations

include variation, crossover and selection with advances such as simple principle, control parameters, robustness a high convergence rate and straightforward realisation. The ant colony algorithm simulates an ant colony and their routing behaviours in nature. Finding a solution through information exchange and cooperation among the ant colony. However, the mechanism for feedback has a slow convergence speed. The particle swarm optimisation algorithm has good performance however, needs to be used indirectly in getting the optimal solution of multiple object optimisation problems. As during a search its own position needs to be updated through a follow up of individual extreme value and global extreme value. Simulated annealing algorithm finds the probability sense using a random search. Which is a global optimisation method. (Dan Liu, 2013) derived a method for test paper generation through using the ant colony algorithm. A comparison is also made between using other algorithms such as a random variable algorithm a backtracking algorithm and an artificial intelligence algorithm. Describing the random variable algorithm is that it extracts questions and if they meet certain conditions it then forms a test paper based on these conditions. However, it can fail to meet these requirements. Which in turn offers a poor success rate. The backtracking algorithm works well on small scale generation. Once the scale is largely increased so the time taken to process the generation increases. A new approach would be to compose test papers using the ant colony algorithm as it can search at a far greater speed with intelligent search.

Chapter 3

Methodology Chapter

3.1 Introduction

Agile is not a methodology but more of an alternative to the existing SDLC Models. To articulate the methods and techniques used in this plan. In figure 3.4 is the outcome after reviewing various SDLC methodologies with reference to tutorialspoint (2016):

- Adoption of the Agile Model.
- Suits the requirements for this project.
- Widely accepted within companies within the IT industry.
- Valuable learning curve in gaining experience with this model.
- Model has the ability to adapt and tailor itself within each increment as the project moves forward.
- Advantageous to the project.

3.1.1 What is Agile?

Agile is an iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end. It works by

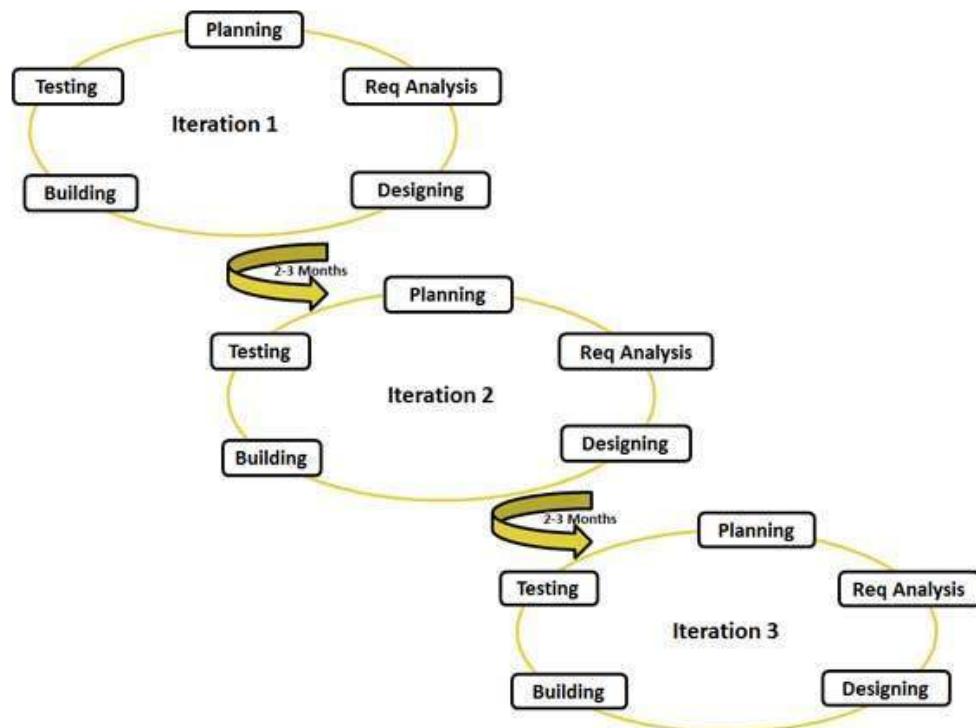


Figure 3.1: Graphical illustration of the Agile Model (tutorialspoint, 2016)

breaking projects down into little bits of user functionality called user stories, prioritizing them, and then continuously delivering them in short two week cycles called iterations.

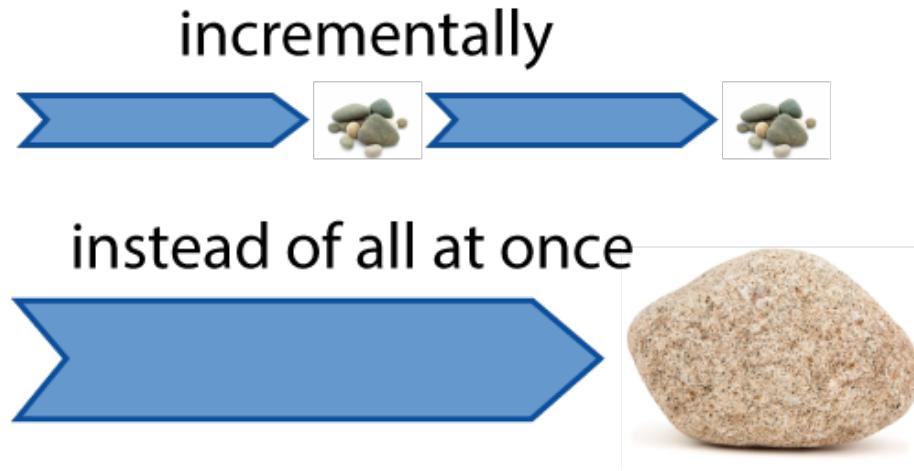


Figure 3.2: Increments of the Agile Model (agile in a nutshell, 2016)

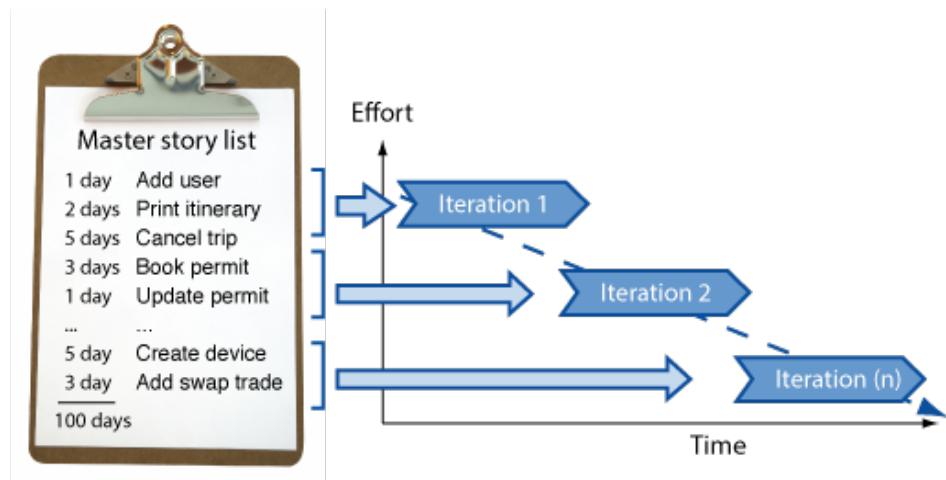


Figure 3.3: Iterations of the Agile Model (agile in a nutshell, 2016)

3.1.2 How does it work?

At its core, Agile does the same thing you and I do when faced with too much to do and not enough time. Then, using Agile estimation techniques, you size your stories relatively

to each other, coming up with a guess as to how long you think each user story will take. Like most lists, there always seems to be more to do than time allows. So you ask your customer to prioritize their list so you get the most important stuff done first, and save the least important for last. Then you start delivering some value. You start at the top. Work your way to the bottom. Building, iterating, and getting feedback from your customer as you go. Then, as you and your customer start delivering, one of two things is going to happen. You'll discover:

You're going fast enough. All is good. Or, You have too much to do and not enough time.

At this point you have two choices. You can either a) do less and cut scope (recommended). Or you can b) push out the date and ask for more money.

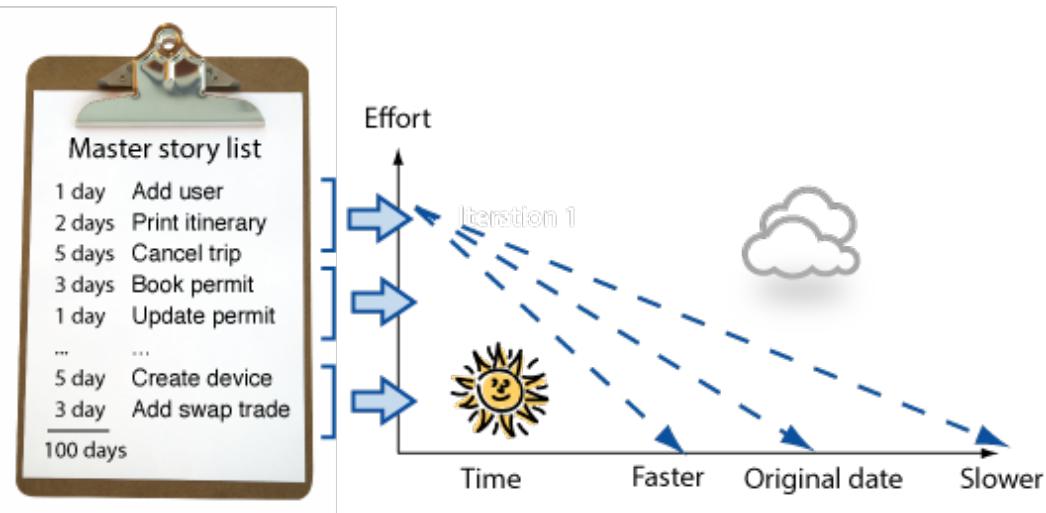


Figure 3.4: Making a list (agile in a nutshell, 2016)

3.1.3 How is it different?

Analysis, design, coding, and testing are continuous activities

You are never done analysis, design, coding and testing on an Agile project. So long as

there are features to build, and the means to deliver them, these activities continue for the duration of the project.

3.1.4 Agile vs Waterfall

Traditional Waterfall treats analysis, design, coding, and testing as discrete phases in a software project. This worked OK when the cost of change was high. But now that it's low it hurts us in a couple of ways. First off, when the project starts to run out of time and money, testing is the only phase left. This means good projects are forced to cut testing short and quality suffers. Secondly, because working software isn't produced until the end of the project, you never really know where you are on a Waterfall project. That last 20% of the project always seems to take 80% of the time.

3.2 Data Collection Methods

3.2.1 Internet Search

Discusses where I obtained my information

3.2.2 Supervisor Input

Discussion on how to store the encrypted data in tables

3.2.3 Journals in Library

Journals which were read and relevant to this project

3.3 Method of Analysis

3.3.1 Formulation of Where to Start

Researching the title of the project

3.3.2 Early System Implementation

This project was linked to an assignment which helped in my progression

3.3.3 Review of Literature

Undertaking a literature review aided with making comparisons of other systems produced

3.4 Summary

Summary with all terms discussed within the chapter

Chapter 4

Implementation - "Building the solution"

4.1 Introduction

This chapter is a walkthrough of the steps which describe the construction of the application.

4.2 Terminal

4.2.1 Command Line Instructions

As this application was centred towards security and being secure as the data it would hold would need to be kept safe. Therefore it made sense to focus on a good login with authentication and authorisation. To start working with Symfony, one needs to setup the Symfony environment through an installation process before Symfony applications can be created. Instructions for this can be found on the SensioLabs Symfony website. Navigating to the Documentation page. In there can be found Chapter 1 which has the Setup instructions under the Get Started dropdown menu. They explain the different ways for installing and setting up the Symfony framework for both Mac OS and Windows. Along with some troubleshooting ideas if there are any problems with the installation. This application was built on a Mac OS so the instructions would vary slightly due to the command line instructions used. Using the installer made it easy to create this application with Symfony and only needed to be done once as it was installed globally on the machine.

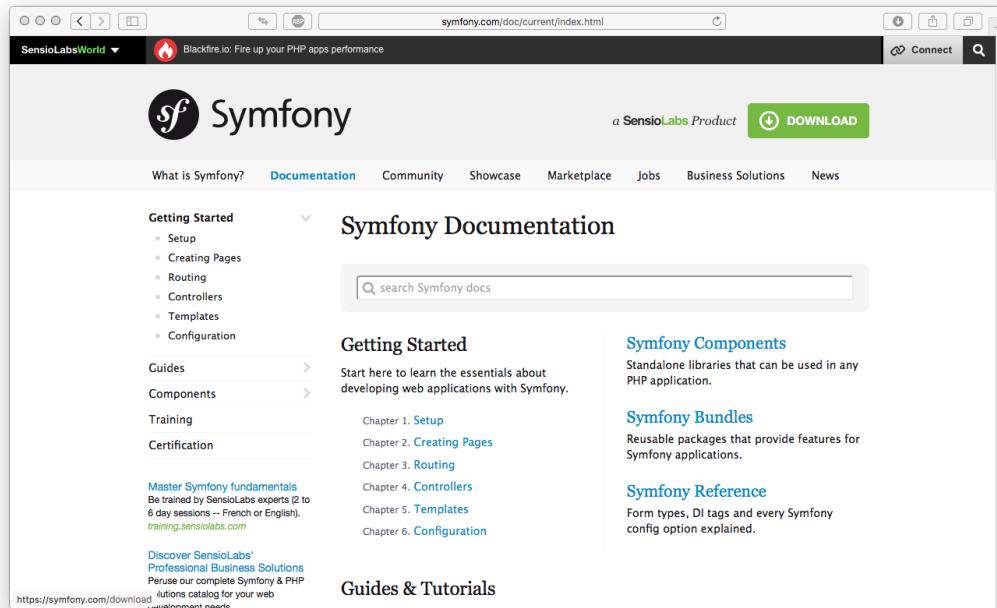


Figure 4.1: Symfony Documentation

```
# Linux and macOS systems
$ sudo mkdir -p /usr/local/bin
$ sudo curl -LsS https://symfony.com/installer -o /usr/local/bin/symfony
$ sudo chmod a+x /usr/local/bin/symfony

# Windows systems
c:\> php -r "readfile('https://symfony.com/installer');" > symfony
```

Figure 4.2: Symfony Installation Setup

With this completed moving into the directory or environment that the application will live which was the desktop. A final command in the terminal window was issued. This time starting with symfony new and by giving a project a name of choice thereafter. In this case the name COMPH4021-Project was used. The project was based on the current version of Symfony which is version 3.2.8. However, other versions can be specified after the project name in the terminal window. Once this part was completed. All required components were downloaded into a project folder with the name of which was given. The components are a set of files and directories which form the web application which use the Symfony libraries. The installer also carries out a check to make sure all requirements are met. If requirements are not all met a list is generated which provides the changes that are needed. In this case no changes needed to be made.

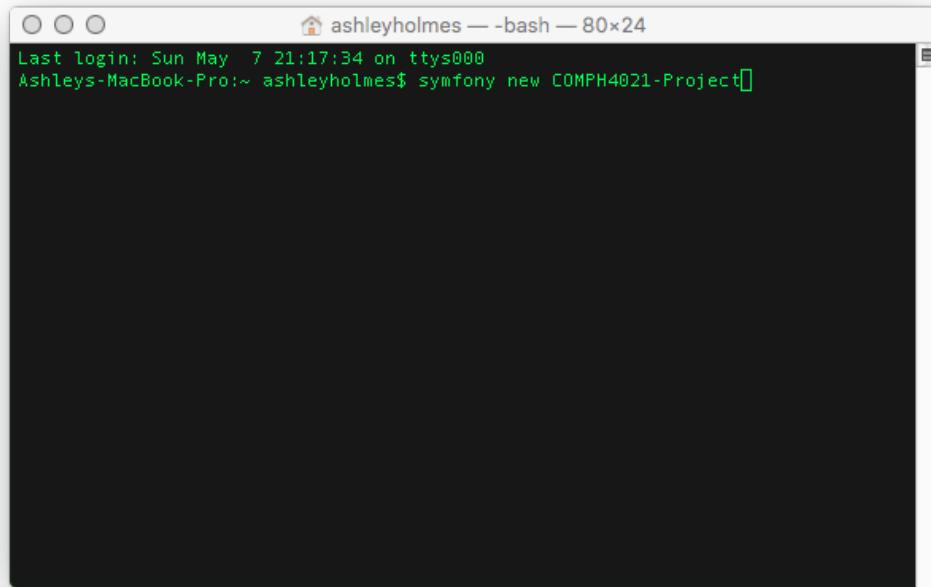


Figure 4.3: Symfony Application Setup

The below figure 4.4 displays the command issued to download the project folder and following that in figure 4.5 one can see that the project is being prepared and where it will be stored. In this case it was stored in /Users/ashleyholmes/Desktop/COMPH4021-Project.

```

Last login: Mon May  8 17:44:01 on ttys006
[ashley@MacBook-Pro: ~ashleyholes]$ cd Desktop
[ashley@MacBook-Pro: Desktop ashleyholes]$ symfony new COMPH4021-Project
  Downloading Symfony...
  0 B/S, 5.5 MiB [=====] 15 KiB/S, 5.5 MiB [=====] 31 KiB/S, 5.5 MiB
  113 KiB/S, 5.5 MiB [=====] 79 KiB/S, 5.5 MiB [=====] 47 KiB/S, 5.5 MiB [=====] 95 KiB/S, 5.5 MiB [=====] 63 KiB/S, 5.5 MiB [=====]
  132 KiB/S, 5.5 MiB [=====] 239 KiB/S, 5.5 MiB [=====] 207 KiB/S, 5.5 MiB [=====] 188 KiB/S, 5.5 MiB [=====] 148 KiB/S, 5.5 MiB [=====] 127 KiB/S, 5.5 MiB [=====] 198 KiB/S, 5.5 MiB [=====] 164 KiB/S, 5.5 MiB [=====]
  271 KiB/S, 5.5 MiB [=====] 305 KiB/S, 5.5 MiB [=====] 303 KiB/S, 5.5 MiB [=====] 273 KiB/S, 5.5 MiB [=====] 254 KiB/S, 5.5 MiB [=====] 223 KiB/S, 5.5 MiB [=====] 199 KiB/S, 5.5 MiB [=====] 164 KiB/S, 5.5 MiB [=====]
  309 KiB/S, 5.5 MiB [=====] 369 KiB/S, 5.5 MiB [=====] 367 KiB/S, 5.5 MiB [=====] 308 KiB/S, 5.5 MiB [=====] 303 KiB/S, 5.5 MiB [=====] 353 KiB/S, 5.5 MiB [=====] 319 KiB/S, 5.5 MiB [=====] 289 KiB/S, 5.5 MiB [=====]
  399 KiB/S, 5.5 MiB [=====] 463 KiB/S, 5.5 MiB [=====] 460 KiB/S, 5.5 MiB [=====] 407 KiB/S, 5.5 MiB [=====] 407 KiB/S, 5.5 MiB [=====] 479 KiB/S, 5.5 MiB [=====] 479 KiB/S, 5.5 MiB [=====] 424 KiB/S, 5.5 MiB [=====]
  511 KiB/S, 5.5 MiB [=====] 581 KiB/S, 5.5 MiB [=====] 581 KiB/S, 5.5 MiB [=====] 527 KiB/S, 5.5 MiB [=====] 527 KiB/S, 5.5 MiB [=====] 506 KiB/S, 5.5 MiB [=====] 506 KiB/S, 5.5 MiB [=====] 474 KiB/S, 5.5 MiB [=====]
  545 KiB/S, 5.5 MiB [=====] 625 KiB/S, 5.5 MiB [=====] 591 KiB/S, 5.5 MiB [=====] 639 KiB/S, 5.5 MiB [=====] 639 KiB/S, 5.5 MiB [=====] 607 KiB/S, 5.5 MiB [=====] 607 KiB/S, 5.5 MiB [=====] 574 KiB/S, 5.5 MiB [=====]
  655 KiB/S, 5.5 MiB [=====] 703 KiB/S, 5.5 MiB [=====] 703 KiB/S, 5.5 MiB [=====] 751 KiB/S, 5.5 MiB [=====] 751 KiB/S, 5.5 MiB [=====] 719 KiB/S, 5.5 MiB [=====] 719 KiB/S, 5.5 MiB [=====]
  799 KiB/S, 5.5 MiB [=====] 846 KiB/S, 5.5 MiB [=====] 846 KiB/S, 5.5 MiB [=====] 815 KiB/S, 5.5 MiB [=====] 815 KiB/S, 5.5 MiB [=====] 882 KiB/S, 5.5 MiB [=====] 882 KiB/S, 5.5 MiB [=====]
  911 KiB/S, 5.5 MiB [=====] 946 KiB/S, 5.5 MiB [=====] 946 KiB/S, 5.5 MiB [=====] 910 KiB/S, 5.5 MiB [=====] 910 KiB/S, 5.5 MiB [=====] 994 KiB/S, 5.5 MiB [=====] 994 KiB/S, 5.5 MiB [=====]
  975 KiB/S, 5.5 MiB [=====] 1018 KiB/S, 5.5 MiB [=====] 1018 KiB/S, 5.5 MiB [=====] 976 KiB/S, 5.5 MiB [=====] 976 KiB/S, 5.5 MiB [=====] 1029 KiB/S, 5.5 MiB [=====] 1029 KiB/S, 5.5 MiB [=====]
  1.0 MiB/S, 5.5 MiB [=====] 1.0 MiB/S, 5.5 MiB [=====] 1.0 MiB/S, 5.5 MiB [=====] 1.0 MiB/S, 5.5 MiB [=====] 1.0 MiB/S, 5.5 MiB [=====] 1.0 MiB/S, 5.5 MiB [=====] 1.0 MiB/S, 5.5 MiB [=====]
  1.1 MiB/S, 5.5 MiB [=====] 1.1 MiB/S, 5.5 MiB [=====] 1.1 MiB/S, 5.5 MiB [=====] 1.1 MiB/S, 5.5 MiB [=====] 1.1 MiB/S, 5.5 MiB [=====] 1.1 MiB/S, 5.5 MiB [=====] 1.1 MiB/S, 5.5 MiB [=====]
  1.2 MiB/S, 5.5 MiB [=====] 1.2 MiB/S, 5.5 MiB [=====] 1.2 MiB/S, 5.5 MiB [=====] 1.2 MiB/S, 5.5 MiB [=====] 1.2 MiB/S, 5.5 MiB [=====] 1.2 MiB/S, 5.5 MiB [=====] 1.2 MiB/S, 5.5 MiB [=====]
  1.3 MiB/S, 5.5 MiB [=====] 1.3 MiB/S, 5.5 MiB [=====] 1.3 MiB/S, 5.5 MiB [=====] 1.3 MiB/S, 5.5 MiB [=====] 1.3 MiB/S, 5.5 MiB [=====] 1.3 MiB/S, 5.5 MiB [=====] 1.3 MiB/S, 5.5 MiB [=====]
  1.4 MiB/S, 5.5 MiB [=====] 1.4 MiB/S, 5.5 MiB [=====] 1.4 MiB/S, 5.5 MiB [=====] 1.4 MiB/S, 5.5 MiB [=====] 1.4 MiB/S, 5.5 MiB [=====] 1.4 MiB/S, 5.5 MiB [=====] 1.4 MiB/S, 5.5 MiB [=====]
  1.5 MiB/S, 5.5 MiB [=====] 1.5 MiB/S, 5.5 MiB [=====] 1.5 MiB/S, 5.5 MiB [=====] 1.5 MiB/S, 5.5 MiB [=====] 1.5 MiB/S, 5.5 MiB [=====] 1.5 MiB/S, 5.5 MiB [=====] 1.5 MiB/S, 5.5 MiB [=====]
  1.6 MiB/S, 5.5 MiB [=====] 1.6 MiB/S, 5.5 MiB [=====] 1.6 MiB/S, 5.5 MiB [=====] 1.6 MiB/S, 5.5 MiB [=====] 1.6 MiB/S, 5.5 MiB [=====] 1.6 MiB/S, 5.5 MiB [=====] 1.6 MiB/S, 5.5 MiB [=====]

```

Figure 4.4: Terminal Window Top

```

Preparing project...
✓ Symfony 3.2.0 was successfully installed. Now you can:
* Change your current directory to /Users/ashleyholmes/Desktop/COMPH4021-Project
* Configure your application in app/config/parameters.yml file.
* Run your application:
  1. Execute the php bin/console server:start command.
  2. Browse to the http://localhost:8000 URL.
* Read the documentation at http://symfony.com/doc
[ashley@MacBook-Pro: Desktop ashleyholes]$ 

```

Figure 4.5: Terminal Window Bottom

Figure 4.6: Php Server Run

Figure 4.7: Php Server Start

The next step would be to change directory into the COMPH4021-Project directory as this is where the built in Php server needs to be run from. NGINX and Apache may be used as alternatives however, since the Php server is built in. It makes development much easier. Executing the following command `php bin/console server:run` starts the server. However, once issuing this command the open terminal window would now need to remain out of use while the server is running. One can use a separate terminal window or open a new tab to issue any addition commands which are needed or make use of the PhpStorm terminal window. To run other processes in the background, issuing a `php bin/console server:start` will make it possible to execute commands in the same window which was done here. The difference can be seen in figure 4.6 and figure 4.7.

4.3 Browser

4.3.1 Deploying the Application

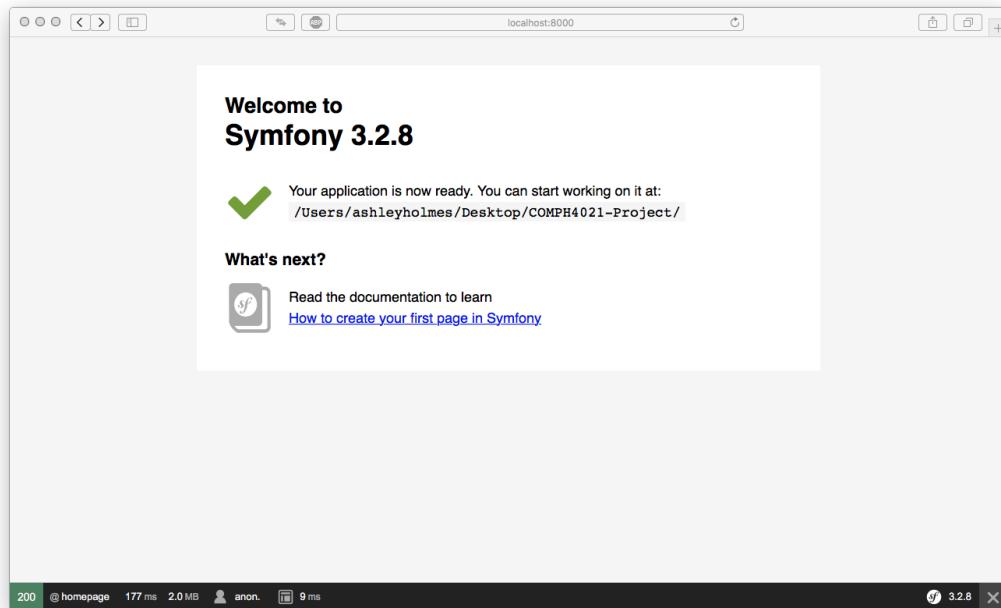


Figure 4.8: Symfony Browser

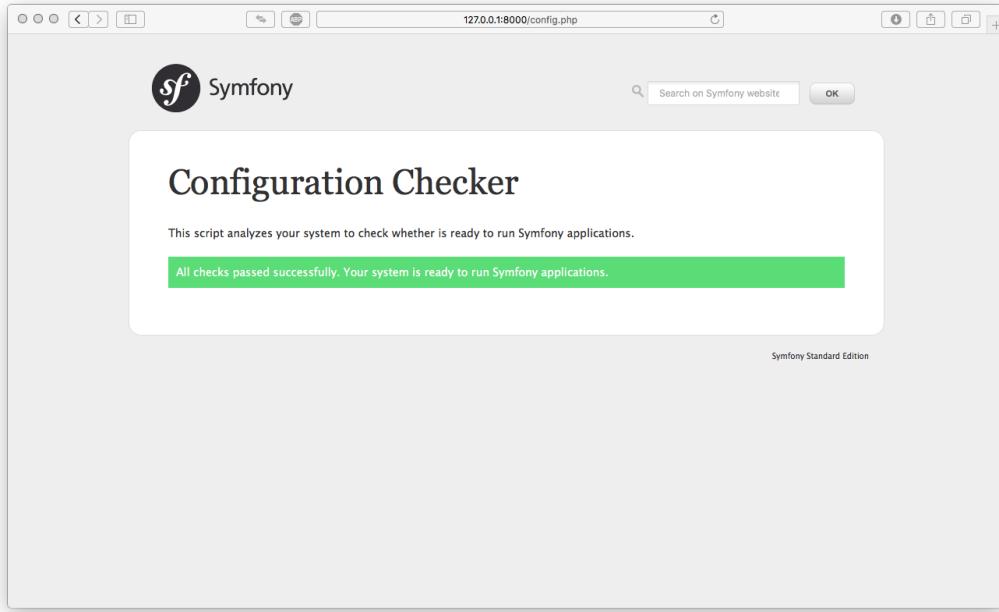


Figure 4.9: Configuration Checker

Now that the configuration phase has been completed one now navigates to the browser and using the URL `http://localhost:8000` as shown in the terminal window in figure 4.5 under the Run your application heading. This brings up the following page in figure 4.8. It is being executed by the Symfony framework from the files inside the project folder. The code is depicted in figure 4.13. In the bottom of the window is the web debug toolbar which is in a maximised position and can be minimised by clicking on the X in the right hand corner. This often offers better visibility when developing. If the mouse is used to hover over the toolbar it will display information such as routing, controllers which were executed, time it took to load the page, which way the user is authenticated on the page and more debugging information and a link to the resources and documentation as shown in figure 4.12. Clicking on the icons will give much more information. The URL `http://127.0.0.1:8000/config.php` would show the user the instructions needed for further configuration. This is reference to figure 4.9.

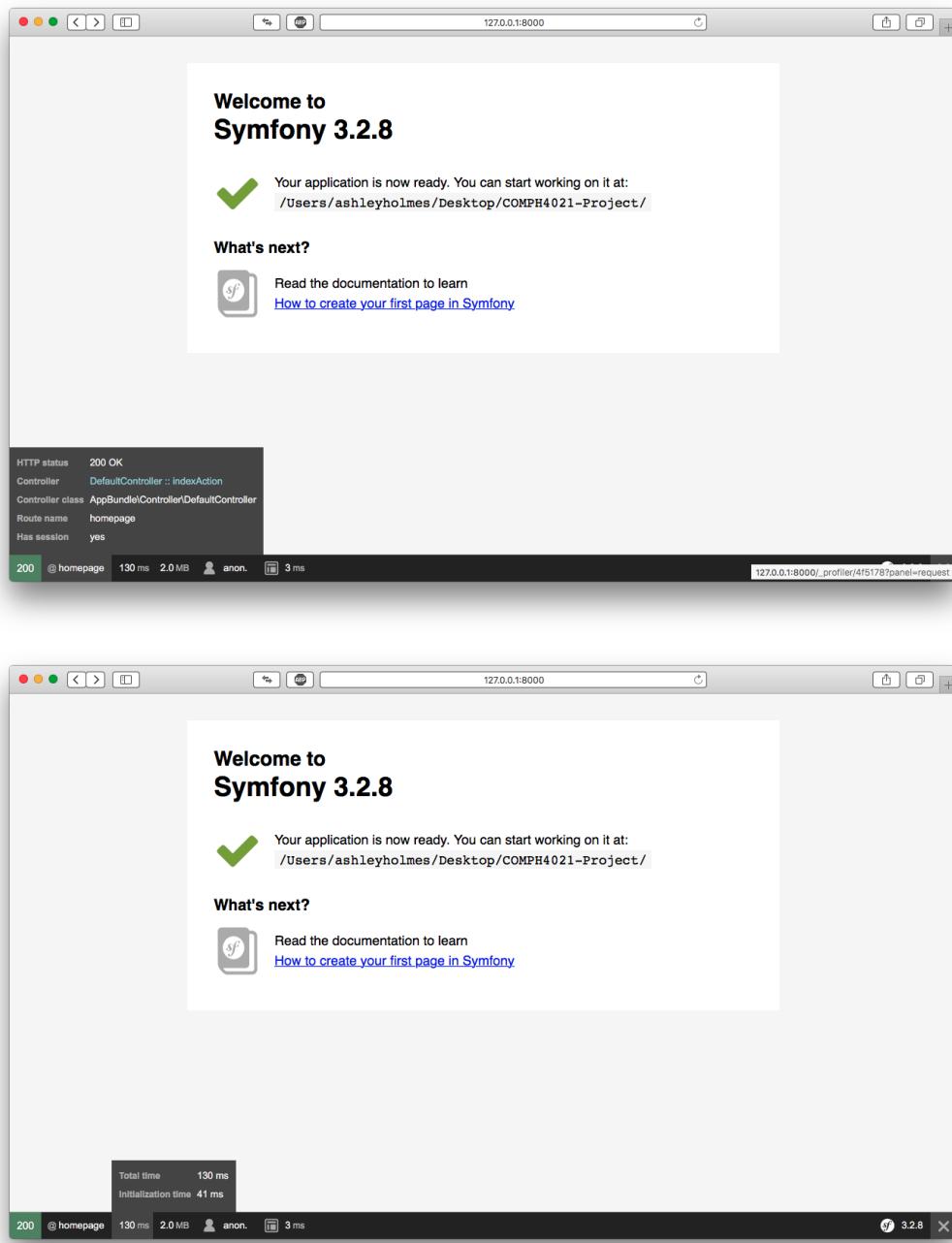


Figure 4.10: Web Debug Toolbar and Profiler Extended

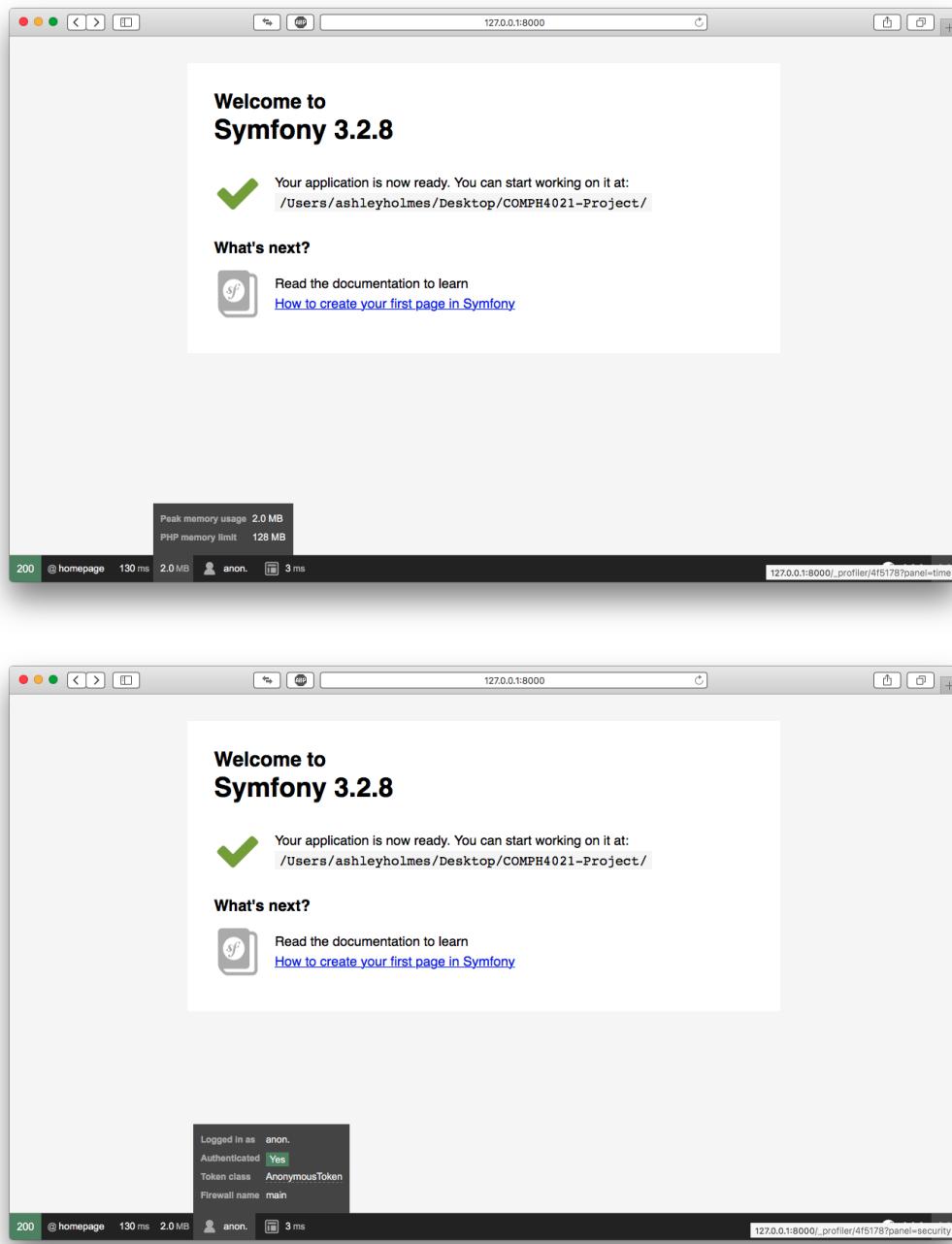


Figure 4.11: Web Debug Toolbar and Profiler Extended

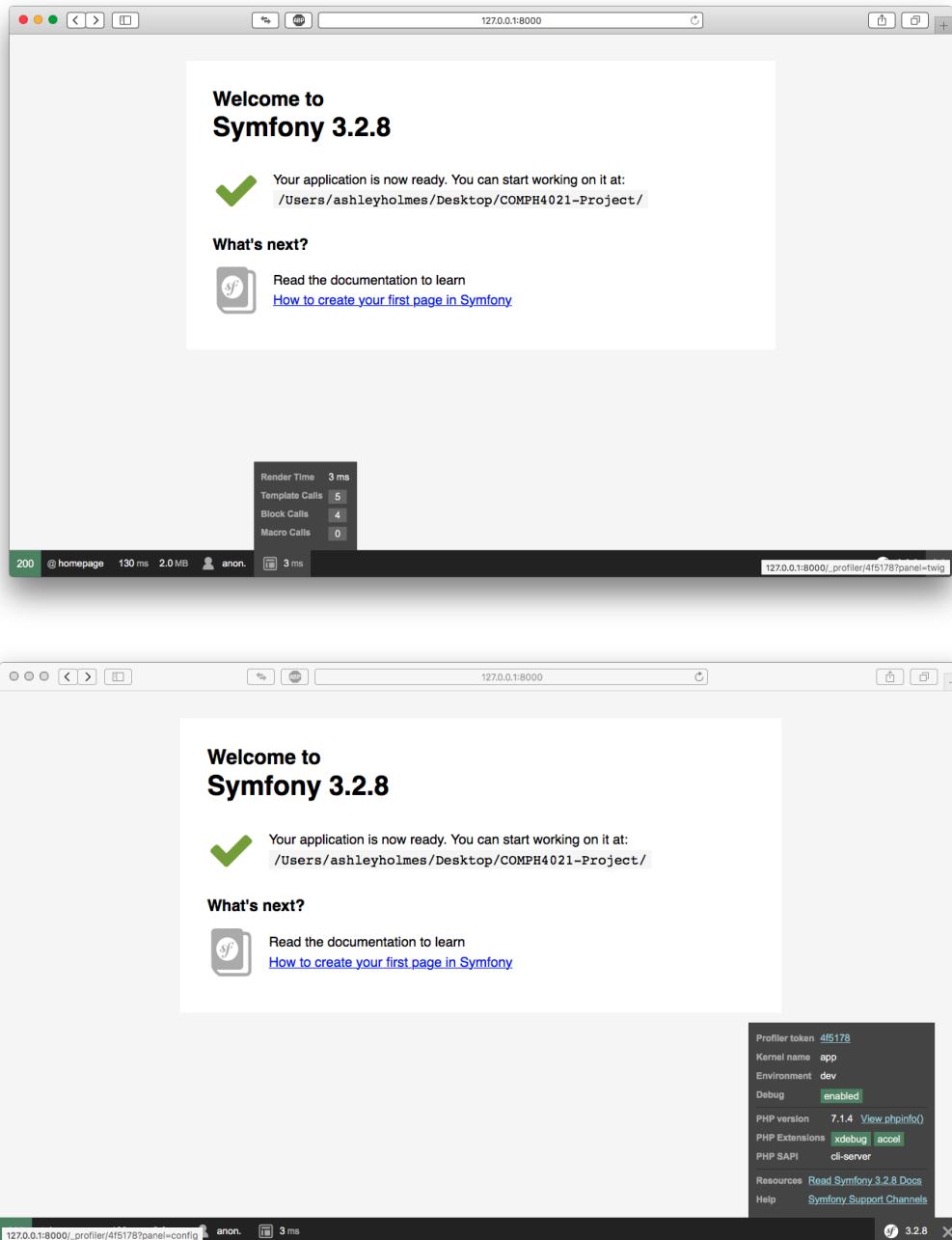


Figure 4.12: Web Debug Toolbar and Profiler Extended

```
if (in_array($this->getEnvironment(), ['dev', 'test'], true)) {
    $bundles[] = new Symfony\Bundle\DebugBundle\DebugBundle();
    $bundles[] = new Symfony\Bundle\WebProfilerBundle\WebProfilerBundle();
    $bundles[] = new Sensio\Bundle\DistributionBundle\SensioDistributionBundle();
    $bundles[] = new Sensio\Bundle\GeneratorBundle\SensioGeneratorBundle();
    $bundles[] = new Doctrine\Bundle\FixturesBundle\DoctrineFixturesBundle();
}

_wdt:
    resource: "@WebProfilerBundle/Resources/config/routing/wdt.xml"
    prefix:  /_wdt

_profiler:
    resource: "@WebProfilerBundle/Resources/config/routing/profiler.xml"
    prefix:  /_profiler

_errors:
    resource: "@TwigBundle/Resources/config/routing/errors.xml"
    prefix:  /_error

_main:
    resource: routing.yml

web_profiler:
    toolbar: true
    intercept_redirects: false
```

Figure 4.13: Web Debug Toolbar and Profiler

4.4 IDE

4.4.1 PhpStorm IDE for PHP

Most of the files live in src/AppBundle. Looking at the controller called DefaultController in the below figure 4.14. This controller class defines what is seen in figure 4.8. It renders the Symfony Welcome Page. Take note of the @Route("/", name="homepage") annotation on line 12. It matches the route in figure 4.12. The next process was to remove the extraneous code which was not needed in the Twig Template and the DefaultController. In the template itself, it was using some variables which was passed in line 18 of figure 4.14 in the DefaultController.php class. With this removed, Bootstrap was enabled as a CDN and added to the templates to provide consistency across all pages. In order to use Bootstrap, it needed to be downloaded from the Bootstrap website. From the Getting started section there are examples which can be chosen as a starting template to use as a foundation for all templates. This was added to the base.html.twig and then modified to make it more specific to the project. Twig uses inheritance and all templates extend from the base.html.twig. In addition to this a small amount of custom CSS was added to achieve the result in figure 4.18

```

<?php
namespace AppBundle\Controller;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Request;

class DefaultController extends Controller
{
    /**
     * @Route("/", name="homepage")
     */
    public function indexAction(Request $request)
    {
        // replace this example code with whatever you need
        return $this->render('default/index.html.twig', [
            'base_dir' => realpath($this->getParameter('kernel.root_dir').DIRECTORY_SEPARATOR),
        ]);
    }
}

```

Figure 4.14: DefaultController

```

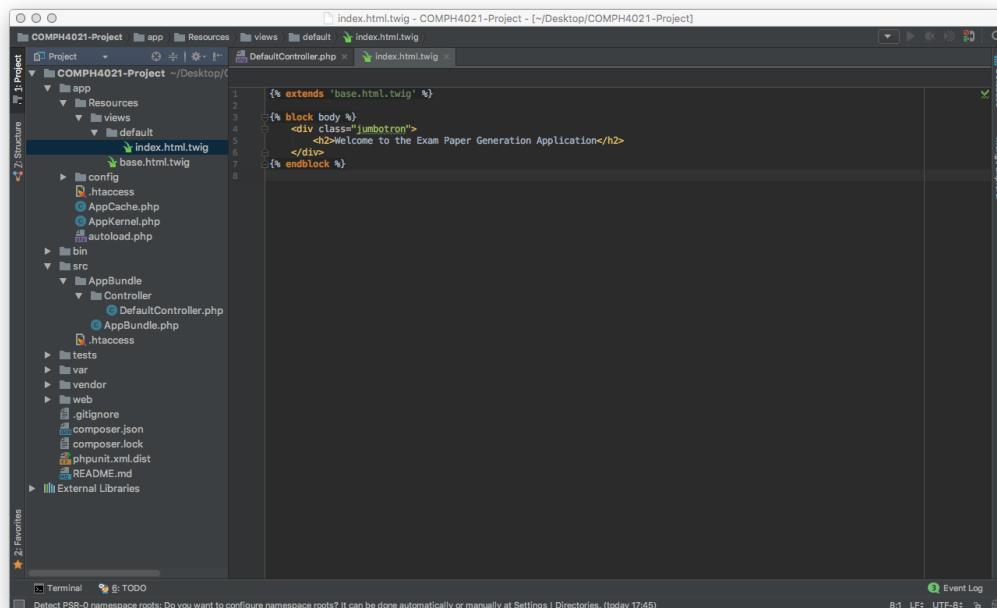
{% extends 'base.html.twig' %}

{% block body %}
    <div id="wrapper">
        <div id="container">
            <div id="welcome">
                <h1>Welcome to</h1>
            </div>
            <div id="status">
                <div>
                    <img id="icon-status" width="1792" height="1792" viewBox="0 0 1792 1792" xmlns="http://www.w3.org/2000/svg">
                        Your application is now ready. You can start working on it at:
                        <code>{{ base_dir }}</code>
                    </div>
                </div>
            <div id="next">
                <h2>What's next?</h2>
                <div>
                    <img id="con-book" version="1.1" xmlns="http://www.w3.org/2000/svg" x="0px" y="0px" viewBox="-12.5 9 64 64" en-
                        <path fill="#AAA" d="M6.0,40.8c2.4,0.0,4.5-0.7,4.9-2.5c2.1-2.4c3.2-1.3c-3.2l-0.8-0.8c-0.4-0.5-0.6-1.3-0.2-
                            c0.4-0.5-0.9-0.8,0.1-0.5c1.3-0.4,1.9-1.3,2.9-2.2c-0.4,1.4-0.7,2.0-0.9,4.2l-0.2,1c-0.7,4.1-3.6,2-2.7,7.5
                            c-0.6,0-1.4,0.6-1.4,1.7c1.9,2.4,1.8c0.8,0.2-0.5-0.3,0.2-0.3,0.3-0.4c0.2-0.1,0.5-0.3,0.4-0.8c0.2-0.7-0.1
                            c2.4,0.1,3.7-1.3,3.7-2.3c0-0.6-0.5-1.2c0-0.3-2c-0.4,0-0.6,0.1c0.1,0.6c-0.1,1.0,0.6,0.1,1.0,1.5c-0.5,0.3-1
                            c0.1,0.6-0.1,0.6c0.1,0.6-0.1,0.6c0.1,0.6-0.1,0.6c0.1,0.6-0.1,0.6c0.1,0.6-0.1,0.6c0.1,0.6-0.1,0.6c0.1,0.6-0.1
                            c0.9,0.1,1.2-1,1.2-1c-1.2-0.2-1.6-1.8c-1.5,0.1-2.8,0.9-3,7.2,1c-1.1,1.3-1.8,2,9-2.3,4.5c-0.9-0.8-1
                            c-1.1,0.7-2,3-0.5-3,4.0,3c-0.5,0.4-0.8,1.1,1.6c-0.4,1.5,0.4,2.4,0.9,0.8,3.4l0.9,1c0.2,0,2,0,6,0,8,0,4,1.5c-1
                            c-0.4-0.2-2-0.5-0.9-0.9c0.1-0.2,0,2-0.3,0.3-0.5c0.1-0.3c0.2-0.6-0.1-1.4-0.1-1.4-0.1-1.6c-0.6-0.2-1.2,0-0
                            C4.3,38.4,4,7,40,6,8,40,8z M46.1,20.9c-4.7-7.5-7.1-5.9-3.8c34.8,10,8,32,7,9,30,2,9,2.3,9,1c2-2.8,1
                            L-7,58.6c0,4,8,0,1.1,13,9,11,6,14,11,134,-7,0,1c3,9,0,7-3,4c-7,0,16,1,20,9z H-6,3,36,40-6,6,6,5-15,6,14,5
                            c0,8,14,5,7,14,5,15,652.1,52,14,2,5c0.1,52-0.3,45-0.3,36,42,42,1,65,1x0,1,8-1,5,3,1-1,5,3,1H4.6c-6.7
                            c2.8,0-2.4,-5-5,4v37.9h3.7c1.6,0,2,9,1,4,2,9,3,1v65.1l42,1,65,1z"/>
                </div>
            </div>
        </div>
    </div>

```

Read the documentation to learn
[http://symfony.com/doc/{{ constant\('Symfony\Component\HttpKernel\Kernel::VERSION'\) }}/page_create](http://symfony.com/doc/{{ constant('Symfony\Component\HttpKernel\Kernel::VERSION') }}/page_create)
 How to create your first page in Symfony

Figure 4.15: Twig Template in IDE

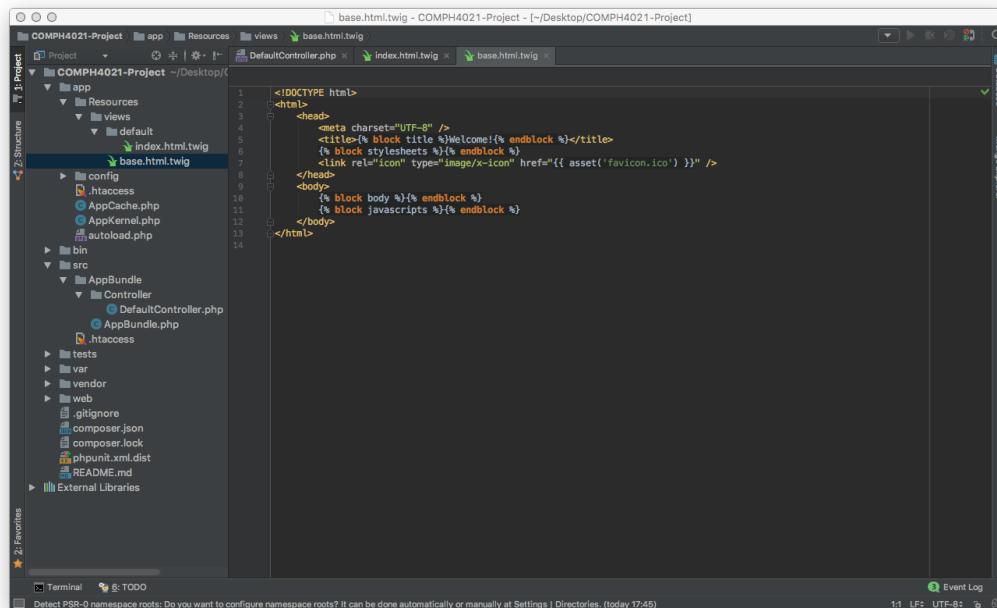


The screenshot shows the 'index.html.twig' file in an IDE. The code is as follows:

```
{% extends 'base.html.twig' %}

{% block body %}
    <div class="jumbotron">
        <h2>Welcome to the Exam Paper Generation Application</h2>
    </div>
{% endblock %}
```

The IDE interface includes a Project Explorer on the left, a code editor with syntax highlighting, and various toolbars and status bars at the bottom.



The screenshot shows the 'base.html.twig' file in an IDE. The code is as follows:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>{% block title %}Welcome!{% endblock %}</title>
        <link rel="icon" type="image/x-icon" href="{{ asset('favicon.ico') }}"/>
    </head>
    <body>
        {% block body %}{% endblock %}
        {% block javascripts %}{% endblock %}
    </body>
</html>
```

The IDE interface includes a Project Explorer on the left, a code editor with syntax highlighting, and various toolbars and status bars at the bottom.

Figure 4.16: Adding Bootstrap

The screenshot shows the PyCharm IDE interface with the following details:

- Project Bar:** Shows the project name "COMPH4021-Project" and the current file "base.html.twig".
- Toolbars:** Includes "File", "Edit", "View", "Run", "Tools", "Help", and "PyCharm Help".
- Sidemenu:** Shows "File", "Edit", "View", "Run", "Tools", "Help", and "PyCharm Help".
- Left Panel (Structure):** Displays the project structure with the following hierarchy:
 - Project: COMPH4021-Project (~/Desktop/COMPH4021-Project)
 - app
 - Resources
 - views
 - default
 - base.html.twig
 - bin
 - src
 - AppBundle
 - Controller
 - DefaultController.php
 - tests
 - var
 - vendor
 - web
 - .gitignore
 - composer.json
 - composer.lock
 - phpunit.xml.dist
 - README.md
 - External Libraries
- Right Panel (Preview):** Shows a preview of the "base.html.twig" file content.
- Bottom Status Bar:** Shows "Event Log" and "Detected PSR-0 namespace roots: Do you want to configure namespace roots? It can be done automatically or manually at Settings | Directories. (today 17:45)"

Figure 4.17: Adding Bootstrap

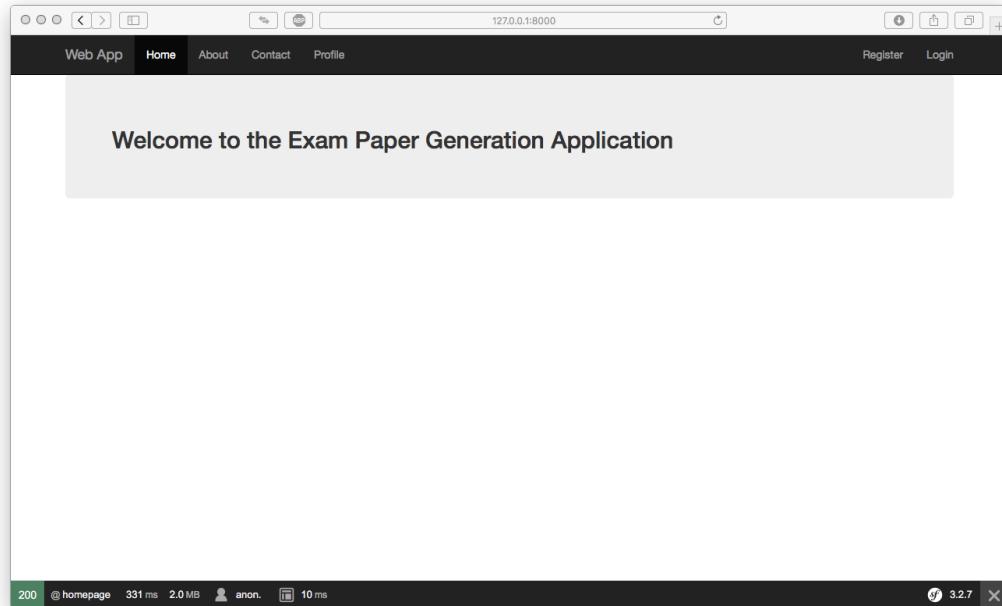
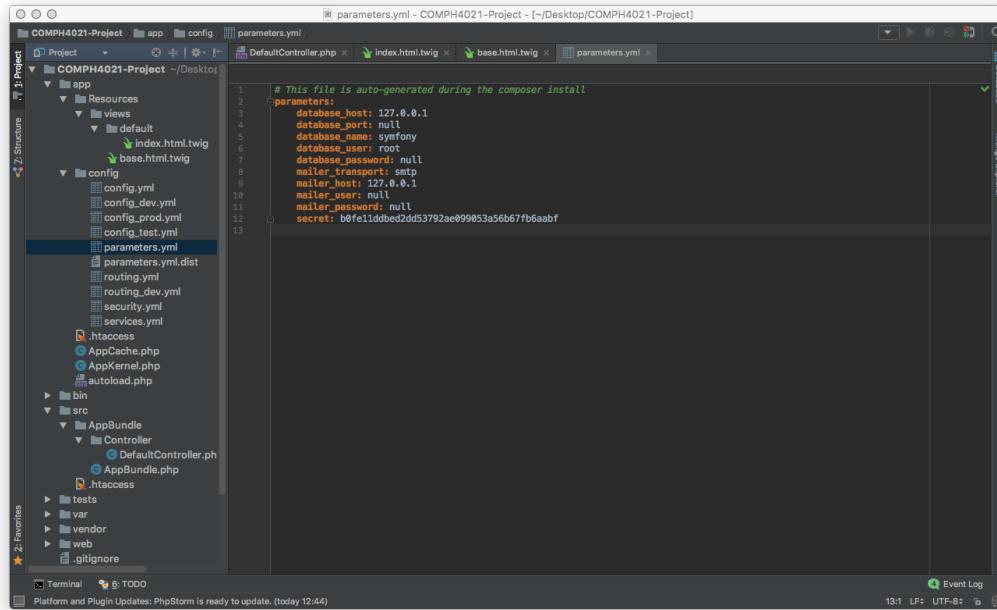


Figure 4.18: Home Page

4.5 Database

4.5.1 Creation of User Entity and CRUD



```

parameters:
    database_host: 127.0.0.1
    database_port: null
    database_name: symfony
    database_user: root
    database_password: null
    mailer_transport: smtp
    mailer_host: 127.0.0.1
    mailer_user: null
    mailer_password: null
    secret: b0fe11ddbed2dd53792ae099053a56b67fb6aabf

```

Figure 4.19: Parameters Yaml

This section covers Object Relational Mapping or ORM. With using an ORM, the concept of working with a database becomes easier as it is an easy switch from one to another such as Mongo to SQL or Postgres. Not having to worry about the actual database. Symfony takes care of the abstraction. In addition the ORM generates the CRUD off the entity which is made and makes everything work. The first thing that needs to be done is make a change to the parameters.yml which can be seen in figure 4.19. The change was to the database name from symfony in line 5 to a name of choice. In this case the name project was used. The database port changed to 8889 and the database password changed to root to correspond with the database login details. With having done that an external MySQL server was needed to make a communication to the database itself. The port number is the port on the local MySQL server. In this case MAMP was used in figure 4.20.

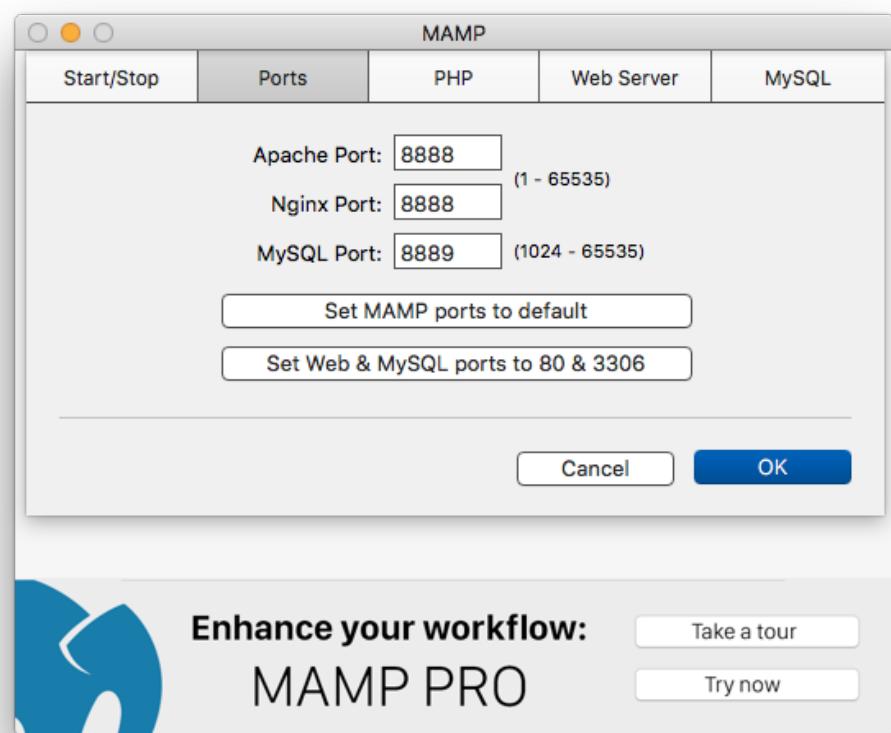


Figure 4.20: Mamp Ports

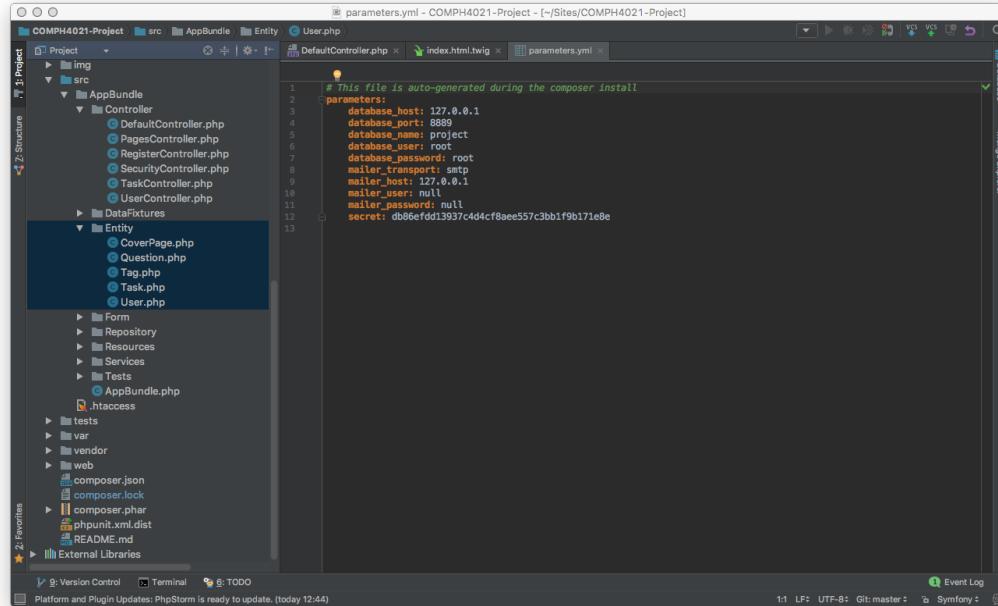


Figure 4.21: Entities

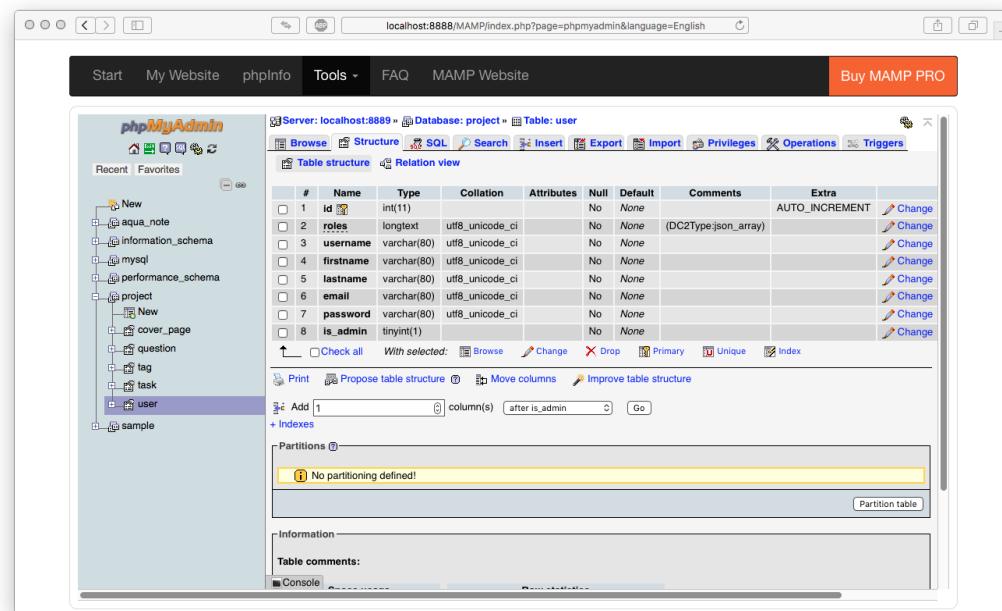
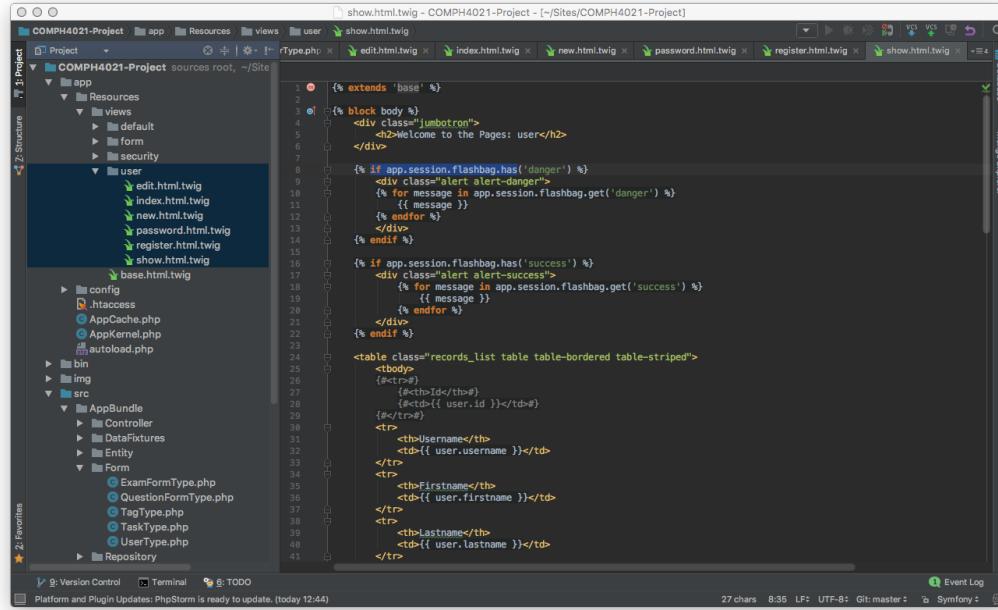


Figure 4.22: User Table



```

show.html.twig - COMPH4021-Project - ~/Sites/COMPH4021-Project

show.html.twig
show.html.twig

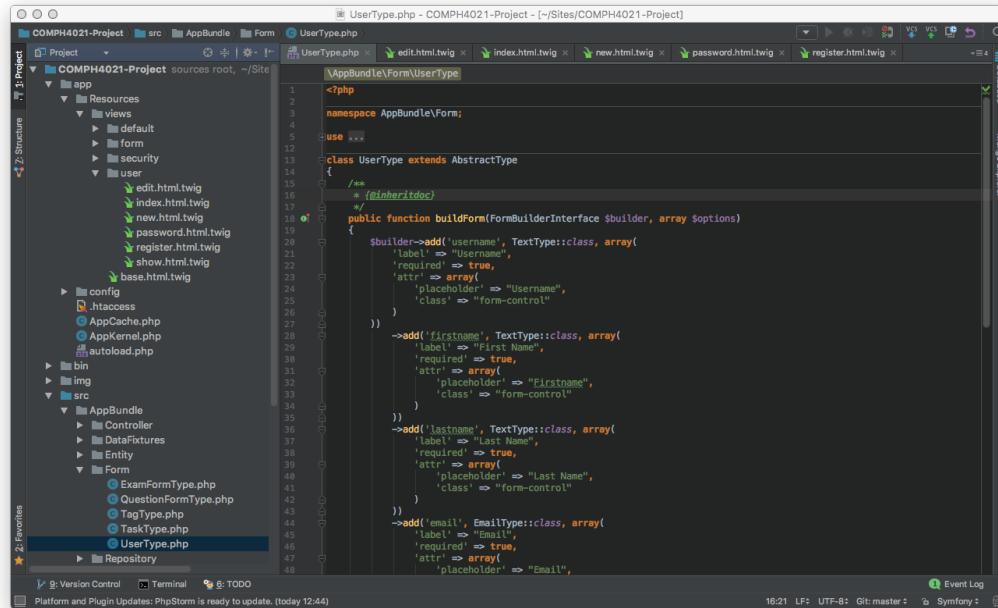
1  {% extends 'base' %}

2  {% block body %}
3      <div class="jumbotron">
4          <h2>Welcome to the Pages: user</h2>
5      </div>

6  {% if app.session.flashbag.has('danger') %}
7      <div class="alert alert-danger">
8          {% for message in app.session.flashbag.get('danger') %}
9              {{ message }}
10         {% endfor %}
11     {% endif %}
12 
13     {% if app.session.flashbag.has('success') %}
14         <div class="alert alert-success">
15             {% for message in app.session.flashbag.get('success') %}
16                 {{ message }}
17             {% endfor %}
18         </div>
19     {% endif %}

20     <table class="records_list table table-bordered table-striped">
21         <tbody>
22             <tr>
23                 <th>Id</th>
24                 <th>Username</th>
25                 <th>Firstname</th>
26                 <th>Lastname</th>
27             </tr>
28             <tr>
29                 <td>{{ user.id }}</td>
30                 <td>{{ user.username }}</td>
31                 <td>{{ user.firstname }}</td>
32                 <td>{{ user.lastname }}</td>
33             </tr>
34         </tbody>
35     </table>
36 
```

Figure 4.23: Twig Templates



```

UserType.php - COMPH4021-Project - ~/Sites/COMPH4021-Project

UserType.php
UserType.php

1 <?php
2
3 namespace AppBundle\Form;
4
5 use ...
6
7 class UserType extends AbstractType
8 {
9     /**
10      * @inheritDoc
11     */
12     public function buildForm(FormBuilderInterface $builder, array $options)
13     {
14         $builder->add('username', TextType::class, array(
15             'label' => "Username",
16             'required' => true,
17             'attr' => array(
18                 'placeholder' => "Username",
19                 'class' => "form-control"
20             )
21         ))
22             ->add('firstname', TextType::class, array(
23                 'label' => "First Name",
24                 'required' => true,
25                 'attr' => array(
26                     'placeholder' => "Firstname",
27                     'class' => "form-control"
28                 )
29             ))
30             ->add('lastname', TextType::class, array(
31                 'label' => "Last Name",
32                 'required' => true,
33                 'attr' => array(
34                     'placeholder' => "Last Name",
35                     'class' => "form-control"
36                 )
37             ))
38             ->add('email', EmailType::class, array(
39                 'label' => "Email",
40                 'required' => true,
41                 'attr' => array(
42                     'placeholder' => "Email",
43                     'class' => "form-control"
44                 )
45             ))
46     }
47 }
48 
```

Figure 4.24: Form Type

Issuing a command in the terminal. `php bin/console doctrine:database:create` will create a database with the name project which was added to the `parameters.yml` file. Creating the entities is done in the same manor. `php bin/console doctrine:generate:entity` brings up a wizard which asks where to put the entity. Giving the command `AppBundle:User`. The entities were all placed in a directory called Entity which resides in `src/AppBundle`. All the work was done in `AppBundle`. The fields in the database are then added along with a repository class which will be discussed later. There is now a username, firstname, lastname, email and password with type String and field length of 80 characters. An example of this is in figure 4.22. Symfony created two different files. The first is the user entity which is the model which is used to base database manipulation, object validation or form validation. The second thing which was done was to create the schema which is done in the terminal with the command of `php bin/console doctrine:schema:create`. This creates a table structure ready for use off of the user object. To create the CRUD in the terminal it was `php bin/console generate:doctrine:crud`. The wizard asks where the entity would be found and based off. As discussed it is in `AppBundle:User` and giving write actions which allows for the update and delete. All the templates were created and added to the Resources directory which are used to create the users in figure 4.23 also a class called `UserType.php` which is a `FormType` and is what all the forms are based off in figure 4.24. With this section completed a bit of functional testing was done to make sure that users could be added to the database, edited and removed.

4.6 Twig

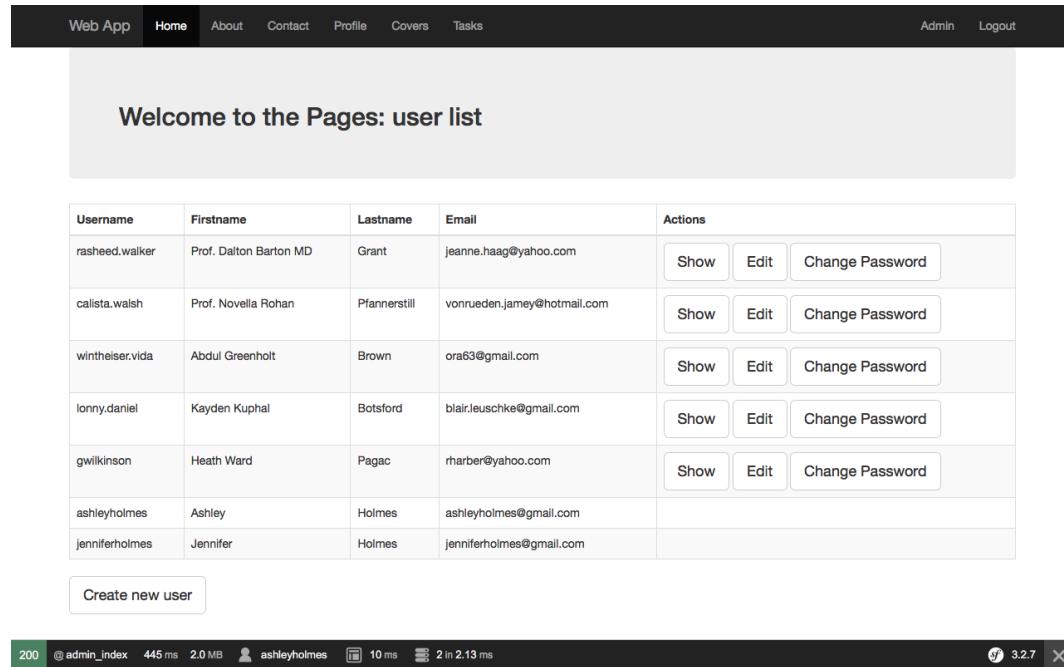
4.6.1 Templates are fine tuned

The templates which are created by Symfony was raw html with no styling. Each form had a basic layout. They were ugly. This is why form theming which is discussed later in the chapter and styling needed to be added to make them look as they do in the below figures. Styling included:

- Addition of placeholders.

- Adding a striped table.
- Button styling.
- Removal of the unordered lists.
- Adding column spanning.
- Changing links to be buttons.
- The bootstrap classes were added.
- Button styling.

Some of the data was adjusted which would have been displayed such as the id and passwords which were removed as they should not be visible from the client side.



The screenshot shows a web application interface. At the top, there is a dark navigation bar with white text containing links: 'Web App', 'Home' (which is the active tab), 'About', 'Contact', 'Profile', 'Covers', and 'Tasks'. On the right side of the navigation bar are 'Admin' and 'Logout' links. Below the navigation bar, the main content area has a light gray header with the text 'Welcome to the Pages: user list'. The main content is a table with the following data:

Username	Firstname	Lastname	Email	Actions		
rasheed.walker	Prof. Dalton Barton MD	Grant	jeanne.haag@yahoo.com	Show	Edit	Change Password
callista.walsh	Prof. Novella Rohan	Pfannerstill	vonrueden.jamey@hotmail.com	Show	Edit	Change Password
wintheiser.vida	Abdul Greenholt	Brown	ora63@gmail.com	Show	Edit	Change Password
lonny.daniel	Kayden Kuphal	Botsford	blair.leuschke@gmail.com	Show	Edit	Change Password
gwilkinson	Heath Ward	Pagac	rharber@yahoo.com	Show	Edit	Change Password
ashleyholmes	Ashley	Holmes	ashleyholmes@gmail.com			
jenniferholmes	Jennifer	Holmes	jenniferholmes@gmail.com			

At the bottom left of the table area is a button labeled 'Create new user'. At the very bottom of the page, there is a footer bar with the following information: '200 @ admin_index 445 ms 2.0 MB ashleyholmes 10 ms 2 in 2.13 ms' and a small logo for '3.2.7'.

Figure 4.25: index.html.twig

Welcome to the Pages: user creation

Username
Username

First Name
Firstname

Last Name
Last Name

Email
Email

Password
Password

[Create](#) [Back to list](#)

Make Admin

Figure 4.26: new.html.twig

Username	rasheed.walker
Firstname	Prof. Dalton Barton MD
Lastname	Grant
Email	jeanne.haag@yahoo.com

[Back to list](#) [Edit](#) [Change Password](#) [Delete](#)

Figure 4.27: show.html.twig

Username
rasheed.walker

First Name
Prof. Dalton Barton MD

Last Name
Grant

Email
jeanne.haag@yahoo.com

Back to list Save Changes Change Password Delete

Make Admin

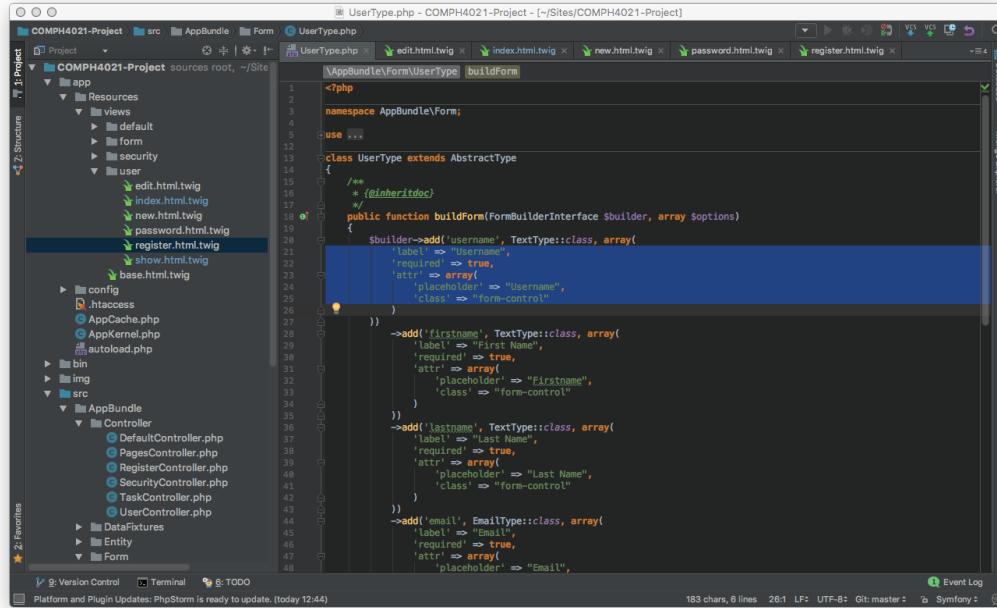
Figure 4.28: edit.html.twig

4.7 Forms

4.7.1 Understanding FormTypes

The FormType works hand in hand with the user object which is created and thus is called the UserType which can be found in `src/AppBundle/Form/UserType.php`. This is what Symfony created when going through the process of creating CRUD. The FormType is used to take control of the input tags and labels. As mentioned in the previous Twig section with regards to adding the placeholders. This is actually done in the `UserType.php` class including the CSS classes. When the user clicks on the Create new user button. Symfony extracts information from that FormType and it builds the form. In figure 4.29 one can see that the `add` method takes up to three arguments. If only one argument is added, Symfony will make the best guess about what is being done. If the second argument is passed in, the second argument is the type of input that is being provided. It could take an Entity or several other arguments. However it does help Symfony make a better decision about what is being done. The third

argument is where one can explicitly say what the options are such as input tags a class as mentioned before. When an array of options are provided Symfony stops guessing what is being done and it takes explicit direction. This is why an array of options are given to explicitly say everything. From line 20 to 25 there is an array of options such as:



```

<?php
namespace AppBundle\Form;

use ...
class UserType extends AbstractType
{
    /**
     * @inheritDoc
     */
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder->add('username', TextType::class, array(
            'label' => "Username",
            'required' => true,
            'attr' => array(
                'placeholder' => "Username",
                'class' => "form-control"
            )
        ))
        ->add('firstname', TextType::class, array(
            'label' => "First Name",
            'required' => true,
            'attr' => array(
                'placeholder' => "First Name",
                'class' => "form-control"
            )
        ))
        ->add('lastname', TextType::class, array(
            'label' => "Last Name",
            'required' => true,
            'attr' => array(
                'placeholder' => "Last Name",
                'class' => "form-control"
            )
        ))
        ->add('email', EmailType::class, array(
            'label' => "Email",
            'required' => true,
            'attr' => array(
                'placeholder' => "Email"
            )
        ))
    }
}

```

Figure 4.29: UserType.php

- The input label - is expressly put in with the form rendering where in the twig template figure 4.30 line 12, 17, 22, 27, and 32 is the form label.
- Attribute required is true forces the user to complete this section of the form.
- The array of other attributes are which go inside the input tag.
- Placeholder and class are added here.
- Class is a Bootstrap class of form control.

```

1  {% extends 'base' %}           // Line 1
2  {% form_theme form 'form/formthemecoverwrite' %} // Line 2
3
4  {# block body #}             // Line 4
5      <div class="jumbotron"> // Line 5
6          <h2>Welcome to the Pages: register</h2> // Line 6
7      </div> // Line 7
8
9      <div class="form"> // Line 9
10         {{ form_start(form, {attr: {novalidate:"novalidate"}}) }} // Line 10
11         <div class="form-group"> // Line 11
12             {{ form_label(form.username) }} // Line 12
13             {{ form_widget(form.username) }} // Line 13
14             {{ form_errors(form.username) }} // Line 14
15         </div> // Line 15
16         <div class="form-group"> // Line 16
17             {{ form_label(form.firstname) }} // Line 17
18             {{ form_widget(form.firstname) }} // Line 18
19             {{ form_errors(form.firstname) }} // Line 19
20         </div> // Line 20
21         <div class="form-group"> // Line 21
22             {{ form_label(form.lastname) }} // Line 22
23             {{ form_widget(form.lastname) }} // Line 23
24             {{ form_errors(form.lastname) }} // Line 24
25         </div> // Line 25
26         <div class="form-group"> // Line 26
27             {{ form_label(form.email) }} // Line 27
28             {{ form_widget(form.email) }} // Line 28
29             {{ form_errors(form.email) }} // Line 29
30         </div> // Line 30
31         <div class="form-group"> // Line 31
32             {{ form_label(form.password) }} // Line 32
33             {{ form_widget(form.password) }} // Line 33
34             {{ form_errors(form.password) }} // Line 34
35         </div> // Line 35
36         {{ form_end(form) }} // Line 36
37     </div> // Line 37
38     {% endblock %} // Line 38
39

```

Figure 4.30: Form Label

4.8 Validation

4.8.1 Validation of the Forms

At this stage without populating the form and by using the create button to create a new user in the form there was html5 validation. A notification message would pop up with an instruction such as, "Please fill out this field." In order to validate the form or objects on the server side. The form validation needs to be turned off. Figure 4.31 shows how this was done.

Line 10 is being passed an instance of the FormType in the form start method. Adding arguments to that such as attr: novalidate:"novalidate". With this attribute added there was no server validation. This was performed on the object which is the entity User.php class by adding a use statement such as in line 7 of figure 4.32.

A use statement in Symfony is a class. With the use statement added it was possible to assert constraints on top of the properties for each of the form fields with reference to figure

```

<?php
namespace AppBundle\Entity;

use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;
use Symfony\Component\Security\Core\User\UserInterface;
use Symfony\Component\Validator\Constraints as Assert;
use Doctrine\ORM\Mapping as ORM;

/**
 * User
 *
 * @ORM\Table(name="user")
 * @ORM\Entity(repositoryClass="AppBundle\Repository\UserRepository")
 */
class User implements UserInterface
{
    /**
     * @var string
     * @Assert\Unique
     * @ORM\Column(name="username", type="string", length=80)
     */
    private $username;

    /**
     * @var integer
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @var string
     * @ORM\Column(name="password", type="string", length=255)
     */
    private $password;
}

```

Figure 4.31: Form Validation

```

<?php
namespace AppBundle\Entity;

use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;
use Symfony\Component\Security\Core\User\UserInterface;
use Symfony\Component\Validator\Constraints as Assert;
use Doctrine\ORM\Mapping as ORM;

/**
 * User
 *
 * @ORM\Table(name="user")
 * @ORM\Entity(repositoryClass="AppBundle\Repository\UserRepository")
 */
class User implements UserInterface
{
    /**
     * @var string
     * @Assert\Unique
     * @ORM\Column(name="username", type="string", length=80)
     */
    private $username;

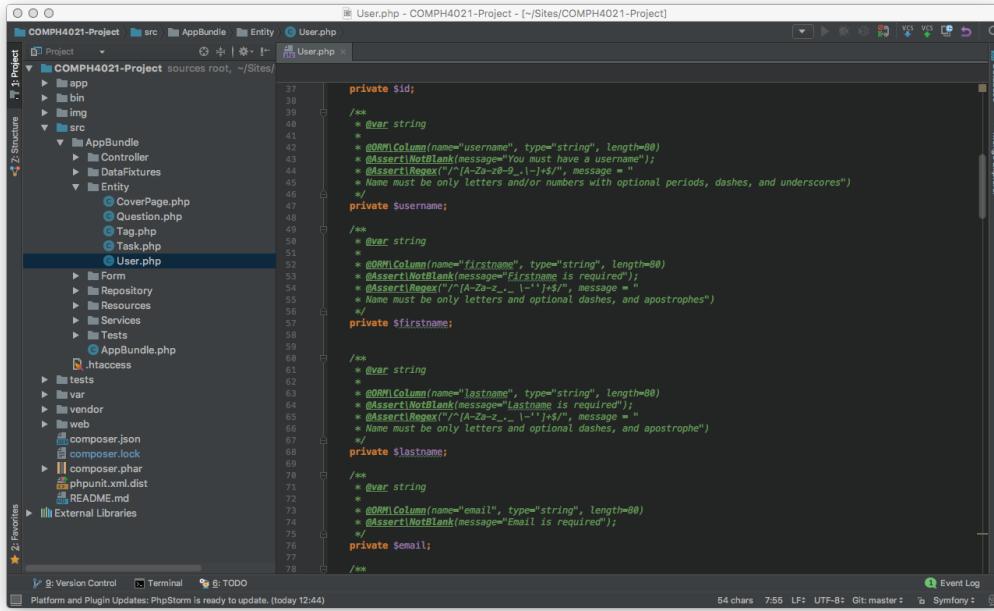
    /**
     * @var integer
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @var string
     * @ORM\Column(name="password", type="string", length=255)
     */
    private $password;
}

```

Figure 4.32: Entity Validation

4.33.



The screenshot shows the PhpStorm IDE interface with the User.php file open. The code defines a User entity with properties \$id, \$username, \$firstname, and \$lastname, each annotated with @ORM\Column and @Assert constraints. The annotations include length restrictions (e.g., length=0 for \$id), string type (e.g., type="string" for \$username), and regular expression patterns (e.g., @Assert\Regex("/[A-Za-z0-9_-]+\$/") for \$username). There are also @Assert\NotBlank annotations with messages indicating required fields and uniqueness requirements. The PhpStorm interface includes a Project tool window, a Structure tool window, and various status indicators at the bottom.

```

private $id;
/** @var string
 * @ORM\Column(name="username", type="string", length=0)
 * @Assert\NotBlank(message="You must have a username")
 * @Assert\Regex("/[A-Za-z0-9_-]+$/")
 * @Assert\Length(min="5", max="25", message="Name must be only letters and/or numbers with optional periods, dashes, and underscores")
 */
private $username;

/** @var string
 * @ORM\Column(name="firstname", type="string", length=0)
 * @Assert\NotBlank(message="First name is required")
 * @Assert\Regex("/[A-Za-z'-]+$/")
 * @Assert\Length(min="2", max="25", message="Name must be one or more letters and optional dashes, and apostrophes")
 */
private $firstname;

/** @var string
 * @ORM\Column(name="lastname", type="string", length=0)
 * @Assert\NotBlank(message="Last name is required")
 * @Assert\Regex("/[A-Za-z'-]+$/")
 * @Assert\Length(min="2", max="25", message="Name must be only letters and optional dashes, and apostrophe")
 */
private $lastname;

/** @var string
 * @ORM\Column(name="email", type="string", length=0)
 * @Assert\NotBlank(message="Email is required")
 */
private $email;

```

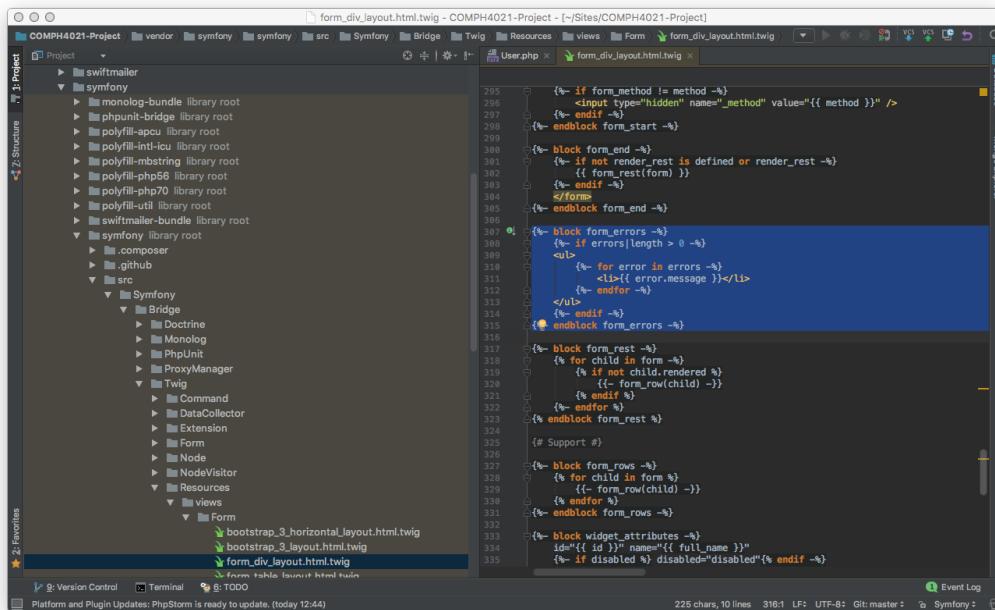
Figure 4.33: Constraints

Functional testing was done at this point in order to make sure everything worked. Another feature which was added was having a unique username and a unique email. This was achieved with yet another use statement which can be found in line 5 of figure 4.32. In the class declaration there are two arguments passed for each property in line 18 and line 19. The property name that needs to be checked against uniqueness. And a message was also passed in, incase the test failed. The same was created for username. It is possible to set both the username and the email as double arguments for one unique entity call however, this will allow the uniqueness of the email to get through as long as the uniqueness of the username is not there. This is why they were passed in separately. Functional testing was performed in order to confirm the changes which were made worked appropriately.

4.9 Theming

4.9.1 Form Theming

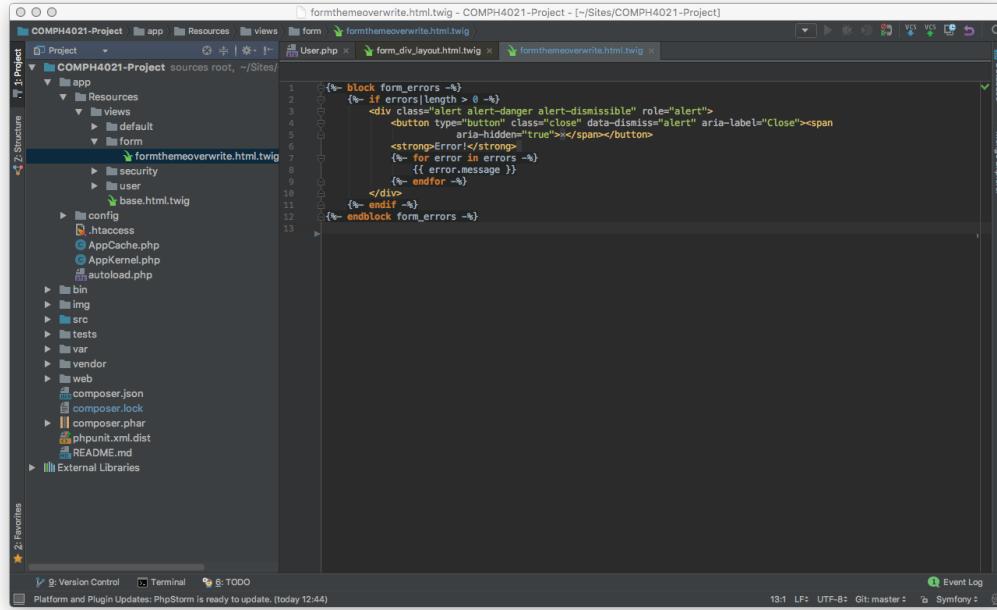
Form theming is a concept which made all the forms look appealing such as the error messages and buttons. Things to remember are, never to adjust the core of Symfony. Instead it is overridden by doing things like telling Symfony where things are that need to be overridden. To proceed with this the error code can be found in the core of Symfony. Figure 4.34 shows the error message which was overridden.



```

295     {%- if form_method != method -%}
296         <input type="hidden" name="_method" value="{{ method }}" />
297     {%- endif -%}
298     {%- endblock form_start -%}
299
300     {%- block form_end -%}
301     {%- if not render_rest is defined or render_rest -%}
302         {{ form_rest(form) }}
303     {%- endif -%}
304     {%- endblock form_end -%}
305
306     {%- block form_errors -%}
307     {%- if errors|length > 0 -%}
308         <ul>
309             {%- for error in errors -%}
310                 <li>{{ error.message }}</li>
311             {%- endfor -%}
312         </ul>
313     {%- endif -%}
314     {%- endblock form_errors -%}
315
316     {%- block form_rest -%}
317         {%- for child in form -%}
318             {%- if not child.rendered -%}
319                 {{ form_row(child) -}}
320             {%- endif -%}
321         {%- endfor %}
322         {%- endblock form_rest %}
323
324     {# Support #}
325
326     {%- block form_rows -%}
327         {%- for child in form %}
328             {%- if child is rowable -%}
329                 {{ form_row(child) -}}
330             {%- endif -%}
331         {%- endblock form_rows -%}
332
333     {%- block widget_attributes -%}
334         id="{{ id }} name="{{ full_name }}"
335         {%- if disabled %} disabled="disabled"{% endif -%}
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
607
608
609
609
610
611
612
613
614
615
615
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1
```

- register.html.twig



```

1  {{% block form_errors -%}
2  {%- if errors|length > 0 -%}
3  <div class="alert alert-danger alert-dismissible" role="alert">
4      <button type="button" class="close" data-dismiss="alert" aria-label="Close"><span>
5          <strong>Error!</strong>
6          {%- for error in errors -%}
7              {{ error.message }}
8          {%- endfor -%}
9      </span></button>
10     </div>
11 {%- endif -%}
12 {{% endblock form_errors -%}}
13

```

Figure 4.35: Form Theming

The buttons were given their own theming and implemented as follow in figure 4.36.

4.10 Fixtures

4.10.1 Fixtures with Faker

Fixtures are used to add data into a database in a controlled manner for the purpose of testing or for the initial data which is required for the application to run. The four actions which were necessary for this was as follows:

- Use Composer to download the appropriate dependencies.
- Adjust the AppKernel.php to use the dependencies.
- Write the fixture.

4.10. FIXTURES

```

UserController.php - COMPH4021-Project - [/Sites/COMPH4021-Project]
Project Controller UserController.php
  UserController.php
    /**
     * Creates a change password form
     * @param User $user
     * @return Symfony\Component\Form\Form
     */
    public function createPasswordForm(User $user)
    {
        $passwordForm = $this->createForm(UserType::class, $user, array(
            'action' => $this->generateUrl('admin/{id}/password_update', array('id' => $user->getId())),
            'method' => 'PUT',
        ));

        // Remove the fields we don't want from the FormType
        $passwordForm->remove('email');
        $passwordForm->remove('username');
        $passwordForm->remove('firstName');
        $passwordForm->remove('lastName');

        // Add submit button
        $passwordForm->add('submit', SubmitType::class, array(
            'label' => 'Save New Password',
            'attr' => array('class' => 'btn btn-default btn-lg'),
        ));
        return $passwordForm;
    }

    /**
     * Handle the form. Then redirect
     * @param Request $request
     * @param $id
     * @return Symfony\Component\HttpFoundation\RedirectResponse|Symfony\Component\HttpFoundation\Response
     */
    public function updatePasswordAction(Request $request, $id)
    {
        // Get the entity manager
    }
}

```

Figure 4.36: Button Theming

Web App Home About Contact Profile Covers Tasks Admin Logout

Username

Username

Error! You must have a username

First Name

Firtname

Error! Firtname is required

Last Name

Last Name

Error! Lastname is required

Email

Email

Error! Email is required

Password

Password

Error! Password is required

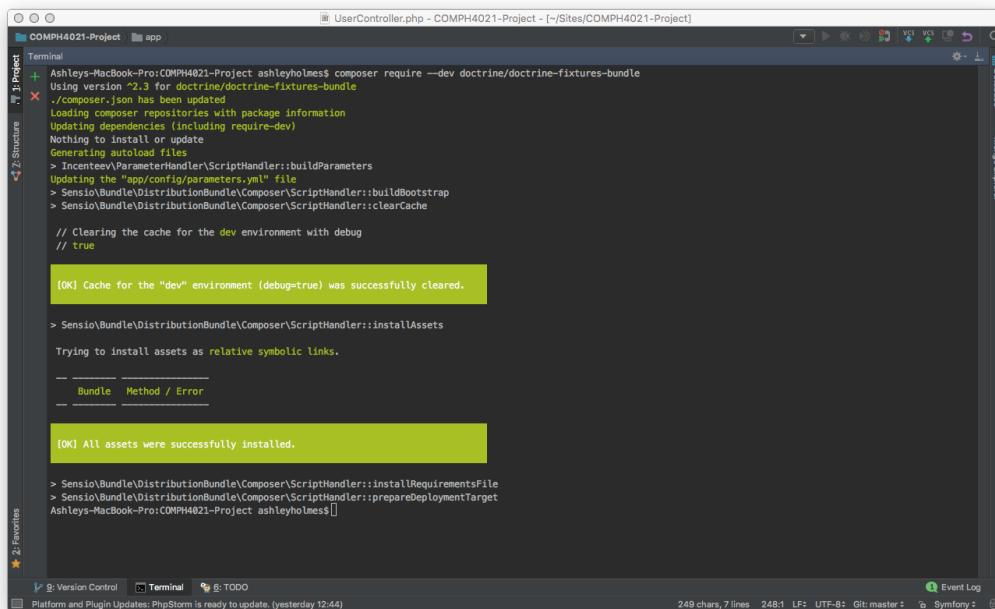
Create Back to list

200 POST @ admin_new 326 ms 2.0 MB ashleyholmes 16 ms 1 in 1.21 ms 3.2.7

Figure 4.37: Form Theming Result

- Use the fixture in the console.

Every good application needs some way to test the data which is being worked on. This is why fixtures is a good place to start. Before this can be done Composer needs to be installed on the computer and it needs to be installed globally. Instructions for this can be found on the Symfony website. Composer is a dependency manager and it is used to require a bundle from Packagist. Packagist is the main Composer repository. It links packages with Composer and shows Composer where to get the code from. With that completed a terminal command may be issued: composer require –dev doctrine/doctrine-fixtures-bundle with reference to figure 4.38.



```
Ashleys-MacBook-Pro:COMPH4021-Project ashleyholmes$ composer require --dev doctrine/doctrine-fixtures-bundle
Using version "2.3" for doctrine/doctrine-fixtures-bundle
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Nothing to install or update
Generating autoload files
> Incenteev\ParameterHandler\ScriptHandler::buildParameters
Updating the "app/config/parameters.yml" file
> Sensio\Bundle\DistributionBundle\Composer\ScriptHandler::buildBootstrap
> Sensio\Bundle\DistributionBundle\Composer\ScriptHandler::clearCache

// Clearing the cache for the dev environment with debug
// true

[OK] Cache for the "dev" environment (debug=true) was successfully cleared.

> Sensio\Bundle\DistributionBundle\Composer\ScriptHandler::installAssets
Trying to install assets as relative symbolic links.

[OK] All assets were successfully installed.

> Sensio\Bundle\DistributionBundle\Composer\ScriptHandler::installRequirementsFile
> Sensio\Bundle\DistributionBundle\Composer\ScriptHandler::prepareDeploymentTarget
Ashleys-MacBook-Pro:COMPH4021-Project ashleyholmes$
```

Figure 4.38: Fixtures

The following line of code figure 4.39 was added to AppKernel in the bundles array of which activates the bundle. It was now possible to build a load in fixtures class. In AppBundle a directory is created with the name DataFixtures and a class called PopulateUserTable.php. In figure 4.40

4.10. FIXTURES

48

```

1 <?php
2 use ...
3
4 class AppKernel extends Kernel
5 {
6     public function registerBundles()
7     {
8         $bundles = [
9             new Symfony\Bundle\FrameworkBundle\FrameworkBundle(),
10            new Symfony\Bundle\SecurityBundle\SecurityBundle(),
11            new Symfony\Bundle\TwigBundle\TwigBundle(),
12            new Symfony\Bundle\MonologBundle\MonologBundle(),
13            new Symfony\Bundle\SwiftmailerBundle\SwiftmailerBundle(),
14            new Doctrine\Bundle\DoctrineBundle\DoctrineBundle(),
15            new Sensio\Bundle\FrameworkExtraBundle\SensioFrameworkExtraBundle(),
16            new Knp\Bundle\SnappyBundle\KnpSnappyBundle(),
17            new AppBundle\AppBundle(),
18        ];
19
20        if (in_array($this->getEnvironment(), ['dev', 'test'], true)) {
21            $bundles[] = new Symfony\Bundle\DebugBundle\DebugBundle();
22            $bundles[] = new Symfony\Bundle\WebProfilerBundle\WebProfilerBundle();
23            $bundles[] = new Sensio\Bundle\DistributionBundle\SensioDistributionBundle();
24            $bundles[] = new Sensio\Bundle\GeneratorBundle\SensioGeneratorBundle();
25            $bundles[] = new Doctrine\Bundle\FixturesBundle\DoctrineFixturesBundle();
26        }
27
28        return $bundles;
29    }
30
31    public function getRootDir()
32    {
33        return __DIR__;
34    }
35
36    public function getCacheDir()
37    {
38        return dirname(__DIR__).'/var/cache/'.$this->getEnvironment();
39    }
40
41    public function getLogDir()
42    {
43    }

```

Figure 4.39: Bundles

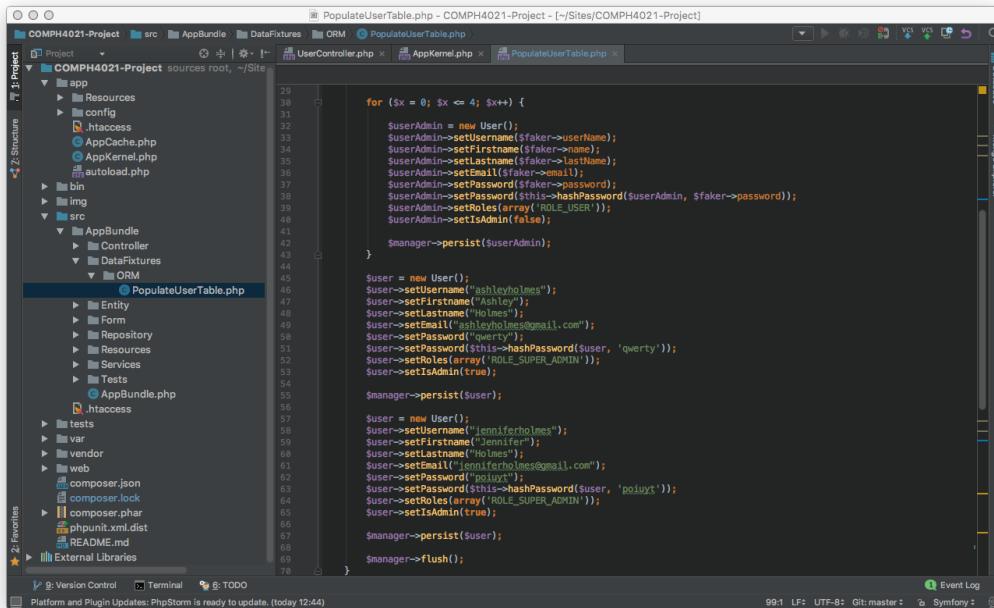
```

1 <?php
2 // * Created by PhpStorm. ...
3
4 namespace AppBundle\DataFixtures\ORM;
5
6 use Faker;
7 use Doctrine\Common\DataFixtures\FixtureInterface;
8 use Doctrine\Common\Persistence\ObjectManager;
9 use Symfony\Component\DependencyInjection\ContainerAwareInterface;
10 use Symfony\Component\DependencyInjection\ContainerInterface;
11 use AppBundle\Entity\User;
12
13 class PopulateUserTable implements FixtureInterface, ContainerAwareInterface
14 {
15     /**
16      * @var ContainerInterface
17      */
18     private $container;
19
20     public function load(ObjectManager $manager)
21     {
22         // TODO: Implement load() method.
23         $faker = Faker\Factory::create();
24
25         for ($x = 0; $x <= 4; $x++) {
26
27             $userAdmin = new User();
28             $userAdmin->setUsername($faker->username);
29             $userAdmin->setFirstname($faker->name);
30             $userAdmin->setLastname($faker->lastName);
31             $userAdmin->setEmail($faker->email);
32             $userAdmin->setPassword($faker->password);
33             $userAdmin->setPassword($this->hashPassword($userAdmin, $faker->password));
34             $userAdmin->setRoles(array('ROLE_USER'));
35             $userAdmin->setIsAdmin(true);
36
37             $manager->persist($userAdmin);
38
39         }
40
41         $user = new User();
42         $user->setUsername("ashleyholmes");
43         $user->setFirstname("Ashley");
44
45     }
46
47     /**
48      * Hashes the given password.
49      *
50      * @param string $password
51      * @return string
52      */
53     protected function hashPassword($user, $password)
54     {
55         $encoder = $this->container->get('security.encoder_factory');
56         $encoder = $encoder->getEncoder($user);
57         return $encoder->encodePassword($password, $user->getSalt());
58     }
59
60 }

```

Figure 4.40: Faker

on line 9 there is a namespace for the file which is also a bundle much like a directory. It becomes a bundle when a bundle class is added to it. Adding a namespace to a class is organising files from one directory, into a sub directories. The PopulateUserTable class lives in a directory called ORM which lives in a directory called DataFixtures which lives in AppBundle. Which is essentially a folder hierarchy. Each one will be unique as each class name is different. Use statements are included from line 11 to line 16. The one on line 15 has a method which is required as it implements the FixtureInterface. Line 13 is the EntityManager which enables the manipulation of the EntityManager in order to persist the object to the database which is the use statement in line 12. Line 25 has a public function called load and is what is required by the FixtureInterface and will bring in the object to the function which is also called dependancy injection. Line 11 is the Faker use statement. This is a PHP Library which generates fake data and can be seen in the admin index page. Line 42 shows how the EntityManager persists it to the User object and line 69 in figure 4.41 which writes the data to the database.



```

PopulateUserTable.php - COMPH4021-Project - [~/Sites/COMPH4021-Project]
Project: COMPH4021-Project sources root, ~/Sites/COMPH4021-Project
  app
    Resources
    config
      .htaccess
    AppCache.php
    AppKernel.php
    autoload.php
  bin
  img
  src
    AppBundle
      Controller
      DataFixtures
        ORM
          PopulateUserTable.php
    AppBundle.php
    htaccess
    tests
    var
    vendor
  web
    composer.json
    composer.lock
    composer.phar
    phpunit.xml.dist
    README.md
    External Libraries

2: Favorites

29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69

for ($x = 0; $x <= 4; $x++) {
    $userAdmin = new User();
    $userAdmin->setUsername($faker->username);
    $userAdmin->setFirstname($faker->name);
    $userAdmin->setLastname($faker->lastName);
    $userAdmin->setEmail($faker->email);
    $userAdmin->setPassword($faker->password);
    $userAdmin->setPassword($this->hashPassword($userAdmin, $faker->password));
    $userAdmin->setRoles(array('ROLE_USER'));
    $userAdmin->setIsAdmin(false);

    $manager->persist($userAdmin);
}

$user = new User();
$user->setUsername("ashleyholmes");
$user->setFirstname("Ashley");
$user->setLastname("Holmes");
$user->setEmail("ashleyholmes@gmail.com");
$user->setPassword("qerty");
$user->setPassword($this->hashPassword($user, 'qerty'));
$user->setRoles(array('ROLE_SUPER_ADMIN'));
$user->setIsAdmin(true);

$manager->persist($user);

$user = new User();
$user->setUsername("jenniferholmes");
$user->setFirstname("Jennifer");
$user->setLastname("Holmes");
$user->setEmail("jenniferholmes@gmail.com");
$user->setPassword("poluyt");
$user->setPassword($this->hashPassword($user, 'poluyt'));
$user->setRoles(array('ROLE_SUPER_ADMIN'));

$manager->persist($user);
$manager->flush();
}

```

Figure 4.41: Object Manager Flush

The line in the console window is what is used to purge what is currently in the database

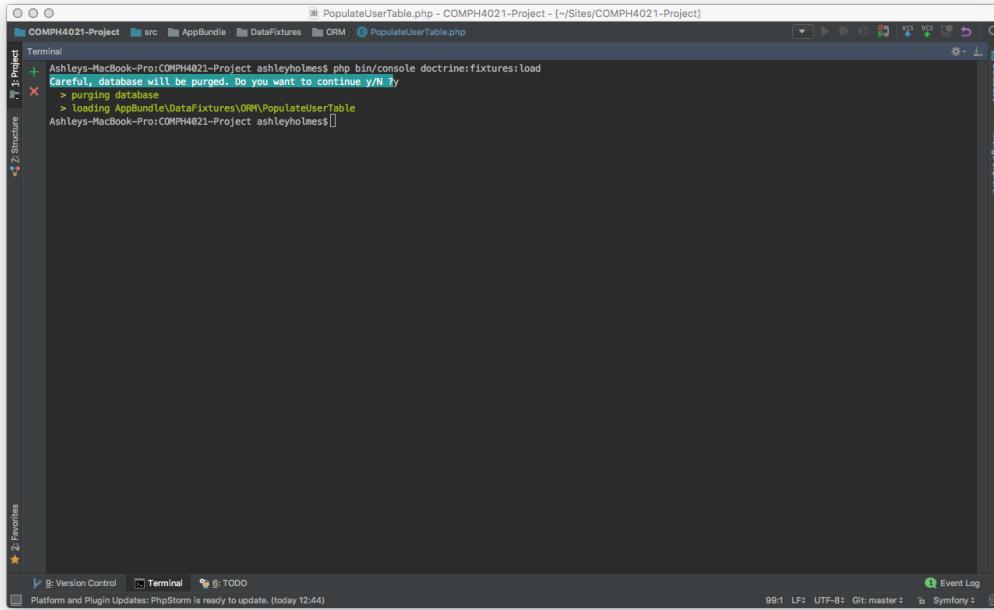
A screenshot of the PhpStorm IDE interface. The central window is a terminal session titled 'PopulateUserTable.php - COMPH4021-Project - [~/Sites/COMPH4021-Project]'. The terminal output shows the command 'doctrine:fixtures:load' being run, with a warning message: 'Careful, database will be purged. Do you want to continue y/N ?'. The user has responded with 'y'. The terminal also shows the path 'AppBundle\DataFixtures\ORM\PopulateUserTable'. The bottom status bar indicates 'Platform and Plugin Updates: PhpStorm is ready to update. (today 12:44)'. On the right side, there are toolbars for 'Database' and 'Mongo Explorer'. The left sidebar shows project structure and favorites.

Figure 4.42: Purge Database

and add new data to it. Each user is persisted separately.

4.11 Passwords

4.11.1 Password Fixtures

This section covers passwords and all that was done to encode them. There is a place where to make passwords and there is also a place in the fixtures where users are inherently uploaded. The first thing which was done was to work on the fixtures. There were a few things to do to make that work.

- Make the fixture ContainerAware by means of the ContainerAwareInterface.
- Adjust the security.yml file with the bcrypt encoder.
- Write a method which uses the bcrypt to hash the password.

- Encode the passwords in the fixture.

When the ContainerAwareInterface was implemented in the class declaration all of the services are then made available for use. This is needed to encode the password in the fixtures file. The file needs to be made ContainerAware. The encoder was added to the security.yml which is figure 4.43 lines 4 and 5.

```

# To get started with security, check out the documentation:
# http://symfony.com/doc/current/security.html

security:
    encoders:
        AppBundle\Entity\User: bcrypt

    role_hierarchy:
        ROLE_USER: [ROLE_PROFILE_PAGE, ROLE_PROFILE_IMAGES]
        ROLE_ADMIN: [ROLE_USER, ROLE_APPROVE_PURCHASE]
        ROLE_SUPER_ADMIN: [ROLE_ADMIN, ROLE_MAKE_ADMINS]

    # http://symfony.com/doc/current/security.html#configuring-how-users-are-loaded
    providers:
        database_users:
            entity:
                class: AppBundle\User
            # entity: { class: AppBundle:User, property: email }
        access_denied_url: /login

    firewalls:
        # disables authentication for assets and the profiler, adapt it according to your needs
        dev:
            pattern: /(_profiler|wdt)|css|images|js/
            security: false

        secured_area:
            pattern: /
            form_login:
                login_path: /login
                check_path: /login_check
                username_parameter: email
                csrf_token_generator: security.csrf.token_manager
            logout:
                path: /logout
                target: /login
                anonymous: ~

    access_control:
        - { path: ^/profile/view, roles: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/profile/edit, roles: ROLE_USER }
        - { path: ^/orders, roles: ROLE_ADMIN }
        - { path: ^/admin, roles: ROLE_SUPER_ADMIN }

```

Figure 4.43: Security.yml

With that being done bcrypt was available for use. Use statements were also needed to make it work which needed to be added to the PopulateUserTable.php class. The use statements can be found on lines 14 and 16 of figure 4.40 and a private property called container was added in line 23. There is also a public function called setContainer which takes an argument of ContainerInterface and initially sets the container variable to null. Followed by this is a function which does the password encryption in figure 4.44.

The entity gets passed into the hashPassword method and the password which needed to be encrypted. With the fixture being containerAware it can be used to get the password encoder from the inbuilt security service. The security service was enabled in the security.yml

```

    /**
 * @param EntityManager $manager
 * @param User $user
 * @param string $password
 */
public function hashPassword(User $user, $password)
{
    $encoder = $this->container->get('security.encoder_factory')
        ->getEncoder($user);

    /**
     * bcrypt comes with its own salt. PHP has a built-in mechanism that it uses to create salt and it puts salt
     * in the String that it returns so we do not need salt but Symfony requires salt, we have to go ahead and use
     * all the methods that are required
     */
    return $encoder->encodePassword($password, $user->getSalt());
}

```

Figure 4.44: Hash Password

file and specified which encryption method which was used. Line 88 is where the encoder variable name is to store that object or service. The service is then captured by calling the containers get method and passing in the service which is to be used. In this case, it is the security encoder factory. That service has a method called getEncoder and the entity is passed in once the method is invoked. The object which has been specified encoder has a method called encodePassword and if passed a password and salt to encode it and it will return an encrypted password. The salt to use also needs to be defined however, bcrypt comes with its own salt, PHP has a built-in mechanism which it uses to create salt and it puts the salt in the string which it returns. So salt is not needed however, it is required by Symfony as all the methods need to be used which are required. This is why a getSalt function had to be created in the User entity figure 4.45 and returns null as bcrypt does not require anything.

Encoding the password is executed in line 38 of figure 4.41 by capturing the hashPassword method which was created and passing in the user object and the String of the password. The encrypted passwords are in figure 4.46

```

User.php - COMPH4021-Project - (~)/Sites/COMPH4021-Project

User.php
UserController.php
PopulateUserTable.php
AppKernel.php
User.php

Project
src
  AppBundle
    Controller
    DataFixtures
    Entity
    Form
    Repository
    Resources
    Services
    Tests
  AppBundle.php
  htaccess
  tests
  var
  vendor
  web
  composer.json
  composer.lock
  composer.phar
  phpunit.xml.dist
  README.md
External Libraries

114 /**
 * @param $isAdmin
 * @return $this
 */
public function setIsAdmin($isAdmin)
{
    $this->isAdmin = $isAdmin;
    return $this;
}

115 /**
 * @return null
 */
public function getSalt()
{
    return null;
}

116 /**
 * Get id
 * @return int
 */
public function getId()
{
    return $this->id;
}

117 /**
 * Set username
 * @param string $username
 * @return User
 */
public function setUsername($username)
{
    $this->username = $username;
    return $this;
}

```

Figure 4.45: getSalt Method

localhost:8888/MAMP/index.php?page=phpmyadmin&language=English

name	firstname	lastname	email	password	is_admin
earline	Ame	Paucek	mckenna14@krajcik.com	\$2y\$13\$UThwX5RCOSMSSXpz.r3CunN5HF4ahHsBdC1R/7h3Yv...	0
rick	Ms.	O'Connell	ibrahim.hane@hotmail.com	\$2y\$13\$GzJpZChwFlgOkoGz3jt5gWUbspLiSgV...	0
onnelly	Orville	Dickens DVM	white.dejon@oconnell.com	\$2y\$13\$UTwX5RCOSMSSXpz.r3CunN5HF4ahHsBdC1R/7h3Yv...	0
	John	Ratke	tschmitt@hotmail.com	\$2y\$13\$SwfM.y6rBLs1R3sSnFcJuQPHoaGflLkvLgNWF96aG...	0
	Veronica	Tromp	gboyle@hettinger.com	\$2y\$13\$WWAxtphoQ4MH25pF66.ausT2zxI48BkkHoSEObAx3C...	0
olmes	Ashley	Holmes	ashleyholmes@gmail.com	\$2y\$13\$TnCIDQZrmhs.IWZur.4AehHGoxcZSUekLG9ueMa7b...	1
olmes	Jennifer	Holmes	jenniferholmes@gmail.com	\$2y\$13\$bqGObszsji0i8ZMM24F.EggNkjg/29h0ynOqVuR...	1

Figure 4.46: Password

4.12 Passwords

4.12.1 Password Services

The last section covered encoding passwords from a fixture. This section is oriented around encoding passwords from a controller and writing the service. To do this, the following steps are needed:

- Implement the UserInterface.
- Create the service for encoding the passwords.
- In the services.yml file implement the service.
- Use the service container to call the encoding method.

Symfony is a set of bundle which does what it is asked to do. It is possible to create a security system from scratch if that is what was intended. However, Symfony's in-built system is so robust. Using it saved much coding time and gave much more features than it would be possible to code from the bottom up. To use Symfony's security system the UserInterface needed to be implemented. From the documentation. The UserInterface has 5 different methods to implement figure 4.47 shows this.

The screenshot shows a web browser displaying the Symfony API documentation. The URL is api.symfony.com/3.2/Symfony/Component/Security/Core/User/UserInterface.html. The page title is "UserInterface".
The left sidebar contains a navigation tree for the UserInterface interface, including: AdvancedUserInterface, ChainUserProvider, EquatableInterface, InMemoryUserProvider, User, UserChecker, UserCheckerInterface, UserInterface (selected), and UserProviderInterface. Sub-categories under UserInterface include Util, Validator, AuthenticationEvents, Security, SecurityContext, SecurityContextInterface, Csrf, Http, Serializer, Stopwatch, Templating, Translation, Validator, and VarDumper.
The main content area starts with a section titled "interface UserInterface". It includes a brief description: "Represents the interface that all user classes must implement. This interface is useful because the authentication layer can deal with the object through its lifecycle, using the object to get the encoded password (for checking against a submitted password), assigning roles and so on. Regardless of how your user are loaded or where they come from (a database, configuration, web service, etc), you will have a class that implements this interface. Objects that implement this interface are created and loaded by different objects that implement UserProviderInterface".
A "Methods" section lists the following methods:

Method Signature	Description
(Role string)[] <code>getRoles()</code>	Returns the roles granted to the user.
string <code>getPassword()</code>	Returns the password used to authenticate the user.
string null <code>getSalt()</code>	Returns the salt that was originally used to encode the password.
string <code>getUsername()</code>	Returns the username used to authenticate the user.
void <code>eraseCredentials()</code>	Removes sensitive data from the user.

Figure 4.47: UserInterface

Chapter 5

Implementation of the System

5.1 Implementation Principles

5.1.1 Object-Oriented Approach

How OO aided the system

5.1.2 Design Patterns

MVC withing the Netbeans IDE

5.1.3 Choice of Language

Java over PHP

5.2 Stages of Admin Implementation

5.2.1 Login

Discuss the login procedure

5.2.2 Administration

Discuss the Administation side

5.2.3 Subsection header 3

fdsdfsdfs

5.3 Stages of User Implementation

5.3.1 Subsection header 1

fdsdfsdfs

5.3.2 Subsection header 2

fdsdfsdfs

5.3.3 Subsection header 3

fdsdfsdfs

5.4 Design

5.4.1 Subsection header 1

fdsdfsdfs

5.4.2 Subsection header 2

fdsdfsdfs

5.4.3 Subsection header 3

fdsdfsdfs

Chapter 6

Testing and Evaluation

6.1 Introduction

Introduction into the testing

6.2 Tests Conducted

This will include do the tables work, checking encryption etc. Storing the data.

6.3 Algorithms

Algorithms used to randomise the tables. And colony, traditional. The FisherYates shuffle.

The Knuth FisherYates shuffle

6.4 Summary

Summary of findings

Chapter 7

Conclusion and Future work

7.1 Contributions

Contributions of this project towards Faculty and the affect on the student

7.2 Limitations

Limitations of project

7.3 Future Work

Integrate into an undertaking currently being deployed at DCU called GURU

7.4 Data Collection

Data analysis performed and exploration. As to who has submitted their examination papers

Bibliography

Student Universal Support Ireland. online, <https://susi.ie/>, (last accessed: October 2016), 2016

Research Questions. online, <http://www.twp.duke.edu>, (last accessed: October 2016), 2016

Agile - Methodology. online, <http://www.agilemethodology.org>, (last accessed: October 2016), 2016

tutorialspoint - SDLC - Agile Model. online, <http://www.tutorialspoint.com>, (last accessed: October 2016), 2016

Central Statistics Office - Unemployment Rate. online, <http://www.cso.ie>, (last accessed: October 2016), 2016

European Social Fund - Investment Funds Programme. online, <http://www.esf.ie>, (last accessed: October 2016), 2016

Citizens Information - Third level places for unemployed people. online, <http://www.citizensinformation.ie>, (last accessed: October 2016), 2016

Guang Cen, Yuxiao Dong, Wanlin Gao, Lina Yu, Simon See, Qing Wang, Ying Yang, Hongbiao Jiang A implementation of an automatic examination paper generation system. online, <http://www.sciencedirect.com>, (last accessed: October 2016), 2016.

Ramandeep Kaur, Shilpy Bansal A Review on various Techniques for Automatic Question Generation. online, <http://www.ijettcs.org/>, (last accessed: October 2016), 2016.

Rohan Bhirangi, Smita Bhoir Automated Question Paper Generation System. online, <http://www.ermt.net/>, (last accessed: October 2016), 2016.

Yvonne SKALBAN, Le An HA, Lucia SPECIA, Ruslan MITKOV Automatic question generation in multimedia-based learning. online, <http://www.aclweb.org/>, (last accessed: October 2016), 2016.

Kapil Naik, Shreyas Sule, Shruti Jadhav, Surya Pandey Automatic Question Paper Generation System using Randomization Algorithm. online, www.erpublication.org, (last accessed: October 2016), 2016.

Sandeep Singh Yadav, Mandeep Singh Yadav Development of System for Automated & Secure Generation of Content. online, <http://www.mecs-press.org/>, (last accessed: October 2016), 2016.

Surbhi Choudhary, Abdul Rais Abdul Waheed, ShrutiKA Gawandi, Kavita Joshi Question Paper Generator System. online, <http://www.ijcstjournal.org>, (last accessed: October 2016), 2016.

agile in a nutshell - What is Agile. online, <http://www.agilenutshell.com/>, (last accessed: December 2016), 2016

Appendices

Appendix A

Web Application Login Screen

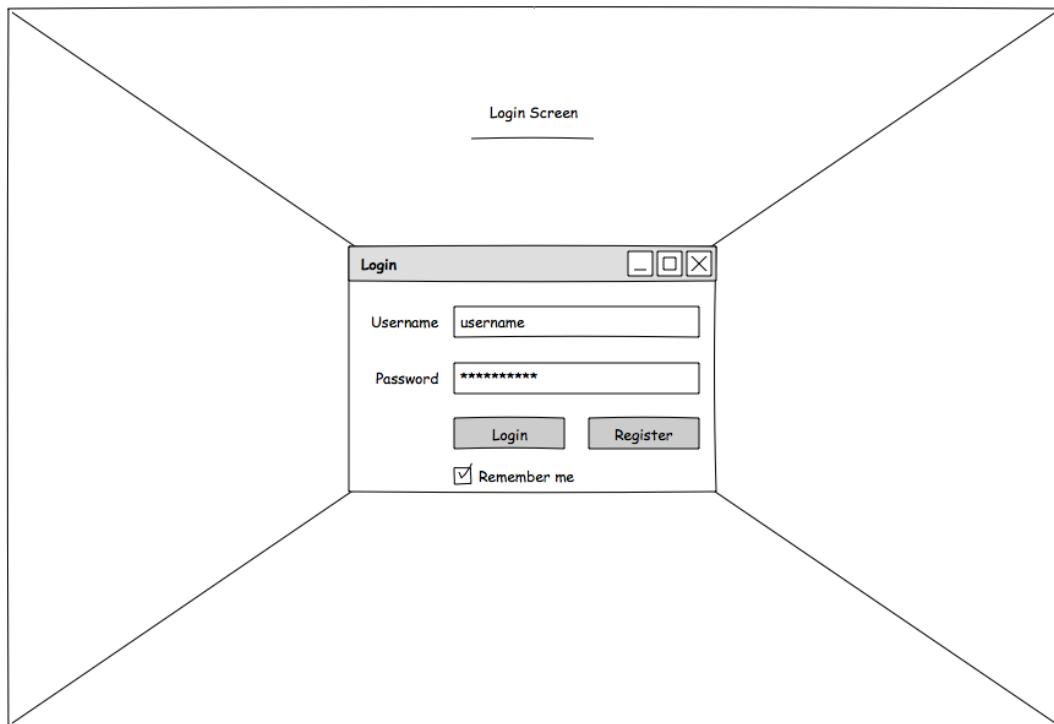


Figure 1: Graphical illustration of the Login Screen

Appendix B

Web Application Question Entry

Question Entry

Id: Question Id generated

Input Question: Paste or type question here...

Menu:

- Add Question
- Create Question
- Show Questions

Logout Submit

865 x 592

Question: Question part 1 - 5

CAO code: BN000

Programme Title: Computing

Module Code: Comp H0000

Module Title: Module

Figure 2: Graphical illustration of the Question Entry

Appendix C

Generate a Question

Generate Question Paper

Menu

[Add Question](#)
[Create Question](#)
[Show Questions](#)

Logout Submit

865 x 592

CAO code	BN000
Semester	1 - 2
Full / Part Time	Full / Part
Programme Title	Computing
Module Code	Comp H0000
Module Title	Module
No. of questions	text goes here

Figure 3: Graphical illustration of the Generate a Question Paper

Appendix D

Show questions

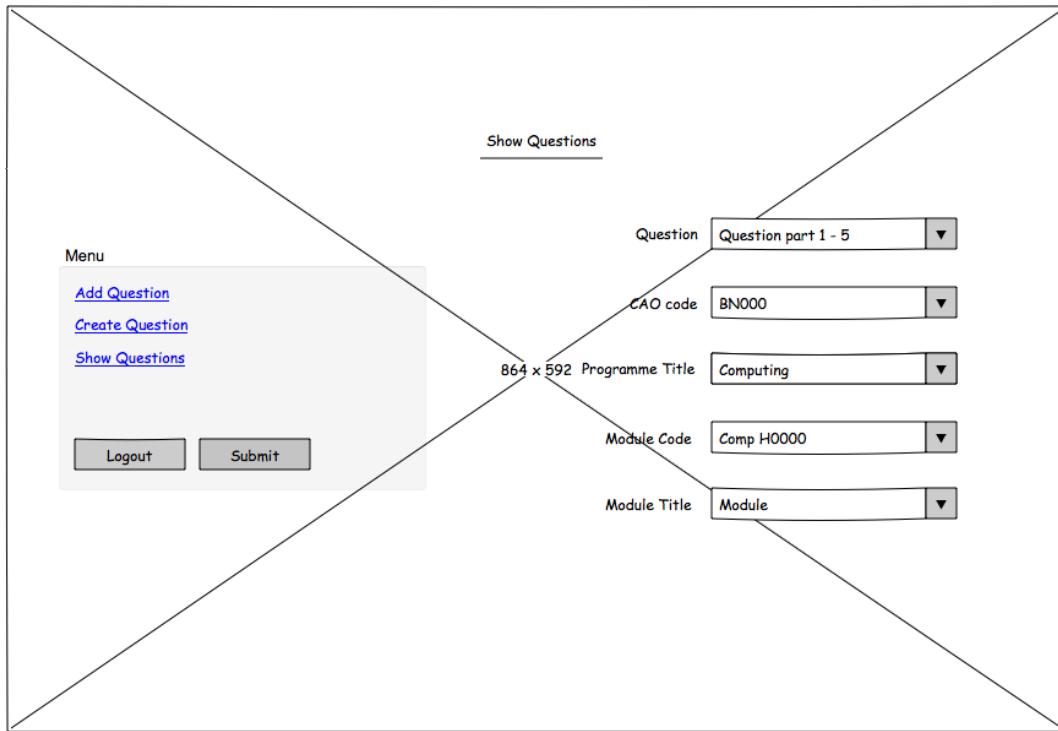


Figure 4: Graphical illustration of the Menu List to view the Questions

Appendix E

Show

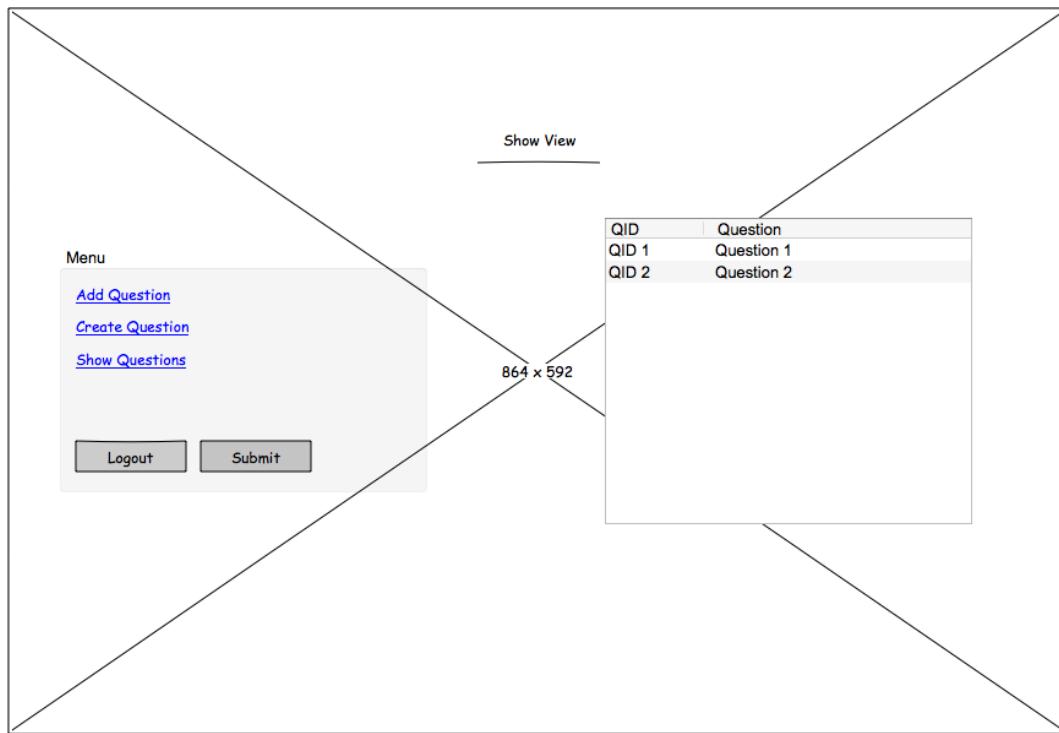


Figure 5: Graphical illustration of the Show list