Project

Dissertation

School of Computing & Information Engineering

|  |  |
| --- | --- |
| Name: | Dylan Keys |
| Project Title: | Rocket Learn (Learning Management System) |
|  |  |
| Supervisor: | Cathryn Peoples |
| Second Marker: | Debbie Rankin |
|  |  |
| Submission Date: | 4th May 2018 |



Plagiarism Statement

I declare that this is my own work and that any material I have referred to has been accurately and consistently referenced. I have read the University’s policy on plagiarism and understand the definition of plagiarism as given in the Project Handbook. If it is shown that material has been plagiarised, or I have otherwise attempted to obtain an unfair advantage for myself or others, I understand that I may face sanctions in accordance with the policies and procedures of the University. A mark of zero may be awarded and the reason for that mark will be recorded on my file.



Acknowledgements

I would like to thank my friends and family for their support throughout this project, and those colleagues and fellow students that took time to participate in the analysis and testing sections of my project.

I would furthermore like to thank my supervisor and extended staff within the school for their support throughout this project work.

Finally, I would like to thank Ulster University for the use of their facilities throughout the academic year, where the majority of planning and development take place.

It has been a pleasure.

Table of Contents

Abstract 1

Chapter 1: Introduction 2

1.1 The Problem 2

1.2 The Solution 5

1.3 Dissertation Structure 7

Chapter 2: Analysis 9

2.1 Existing Systems 9

2.2 Document Analysis 10

2.3 Stakeholders 10

2.4 Survey 11

2.5 Industry Observations 13

2.6 Software Development Methodology 14

2.7 Professional Issues 14

2.8 Defining Scope 16

2.9 Ethical Considerations 20

2.10 Project Risk 20

Chapter 3: Design 21

3.1 System Requirements 21

3.2 User Interface Design 25

3.3 Architectural Design 32

Chapter 4: Implementation, Testing and Evaluation 36

4.1 Implementation 36

4.2 Testing 50

4.3 Evaluation 53

Chapter 5: Conclusions 56

5.1 Key Developments 57

5.2 Future Developments 57

5.3 Project Conclusion 58

References 60

Appendices 62

Appendix 1 - Key Totara features 62

Appendix 2 - Key Moodle features 62

Appendix 3 – User Interface Wireframes 63

Appendix 4 – Admin UI Navigation Flow Diagram 65

Appendix 5 – Google Forms Survey 65

Appendix 6 – Deliverables and their activities 66

Appendix 7 - Glossary of terms 69

Appendix 8 – Assignments 69

Appendix 9 – Testing Fail, Corrective Action 70

Appendix 10 – Administrative settings 71

Appendix 11 – Pingdom Uptime Check 72

Appendix 12 – Server Uptime / Load Average 72

Appendix 13 – Rocket Learn functions (lib.php) 73

Table of Figures

[Figure 1 – Totara Learn settings 3](#_Toc513208842)

[Figure 2 – Moodle settings 4](#_Toc513208843)

[Figure 3 - Depiction of the problem 5](#_Toc513208844)

[Figure 4 - Depiction of the solution 7](#_Toc513208845)

[Figure 5 - Project kanban 14](#_Toc513208846)

[Figure 6 - In Frame / Out of Frame diagram 18](#_Toc513208847)

[Figure 7 - Triple constraints model (Project Management Institute, 2013) 20](#_Toc513208848)

[Figure 8 - Rocket Learn colour scheme 26](#_Toc513208849)

[Figure 9 - Layout flow 26](#_Toc513208850)

[Figure 10 - Layout example, Rocket Learn login page 27](#_Toc513208851)

[Figure 11 - Mobile layout wireframe (course page) 28](#_Toc513208852)

[Figure 12 - Course page wireframe 29](#_Toc513208853)

[Figure 13 - User dashboard wireframe 30](#_Toc513208854)

[Figure 14 - User management admin page wireframe 31](#_Toc513208855)

[Figure 15 - Rocket Learn use case diagram 32](#_Toc513208856)

[Figure 16 - Database Design 33](#_Toc513208857)

[Figure 17 - Courses page, a list of all created courses 38](#_Toc513208858)

[Figure 18 - Enrolment screen for a course 38](#_Toc513208859)

[Figure 19 - Course page (Student view) 39](#_Toc513208860)

[Figure 20 - Course page (Teacher/Admin view) 40](#_Toc513208861)

[Figure 21 - addToElement and addToTopic functions 41](#_Toc513208862)

[Figure 22 - view.php - course page PHP code 45](#_Toc513208863)

[Figure 23 - checkProgress() function 46](#_Toc513208864)

[Figure 24 - enrol.php - enrolment script 46](#_Toc513208865)

[Figure 25 - cron.php script 47](#_Toc513208866)

[Figure 26 - crontab entry 47](#_Toc513208867)

[Figure 27 - Assignment view.php (Check if user has submitted) 48](#_Toc513208868)

[Figure 28 - Assignment view.php (Student section of view.php) 49](#_Toc513208869)

Table of Tables

[Table 1 - LMS Cloud Hosting Costs 15](#_Toc513208831)

[Table 2 - Functional requirements 21](#_Toc513208832)

[Table 3 - Non-functional requirements 23](#_Toc513208833)

[Table 4 - System actors 23](#_Toc513208834)

[Table 5 - User stories 24](#_Toc513208835)

[Table 6 – Core components database tables 34](#_Toc513208836)

[Table 7 - Course element database tables 35](#_Toc513208837)

[Table 8 - Course topics database tables 36](#_Toc513208838)

[Table 9 - Navigation bar breakdown 37](#_Toc513208839)

[Table 10 - Unit test cases 51](#_Toc513208840)

[Table 11 - Acceptance test cases 52](#_Toc513208841)

# Abstract

Implementing a learning management system is a project that entails considerable lead time before any significant implementation. A lot of this time and money is spent on training the administration team, who in turn have to train their end-users (i.e. students of the system). This need for professional training can be linked back to the complexity of the systems on the market, especially when it comes to administration options.

The solution is Rocket Learn, a learning management system that focuses on functional simplicity; in order to reduce the need for extensive training time and cost. Functional simplicity can be defined as a system that does not lack in functionality but has little complexity. The SaaS model is the favoured route to market, as this is where the trend currently lies.

Rocket Learn is a system that embodies the main functionalities (course creation, course completion, activities and progress training) that make up a learning management system and have been implemented with the purpose of reducing administration options and therefore overall complexity.

# Chapter 1: Introduction

The e-learning market was estimated to be worth over 165 billion US dollars in 2016 and is expected to exceed 240 billion US dollars by 2023 (Docebo, 2016); this highlights the enormity of e-learning in today’s world. Learning management systems were estimated to make up 5.22 billion US dollars of this market in 2016, with that market value expected to increase at a CAGR (compound annual growth rate) of 24.7% (marketsandmarkets.com, 2016).

Learning management systems are the modern method of delivering learning material to users. They are predominantly delivered in the form of a web application, and they are used to track, report on and deliver courses to users. Teachers will set up courses, and learners complete them. They are used in educational institutions and organisations all over the world. An LMS can be used to support classroom (in-person) teaching, as well as facilitating material for wholly-online learners; who are expected to make up 25% of all students by 2021 (Learning House, 2015).

## 1.1 The Problem

Complexity, time and training costs are three massive issues that face the deployment of learning management systems. All three of these issues link closely together. The complexity of learning management systems requires users to be trained, before use; this includes the administration team, the teachers/content-creators and the students. Training costs time and money; a standard day of on-site training can be quoted by an industry leader at £750 excluding the trainer’s travel expenses, which are added to the cost (Synergy Learning, 2018).

Learning management systems are not often products that can be immediately picked up and used by all, due to the complexities of their configuration and continued maintenance. The administration sections (settings pages) on learning management systems are difficult to grasp for those new to learning management systems due to the number of options available. Figures 1 and 2 show the administrations options on two of the leading learning management systems (Totara Learn and Moodle respectively). In figure 1 you can see that the administration section has over twenty top-level options, most of these are then expandable into their own sections, with some selections taking the user four layers deep into the menu. Figure 2 shows the latest version of Moodle’s administration options, they have made an attempt at tackling this problem; by sorting the top level options into their own tabs, with specific settings listed under these tabs; this definitely makes the process much more unbearable, but still shows the user an almost incomprehensible amount of options on each page.

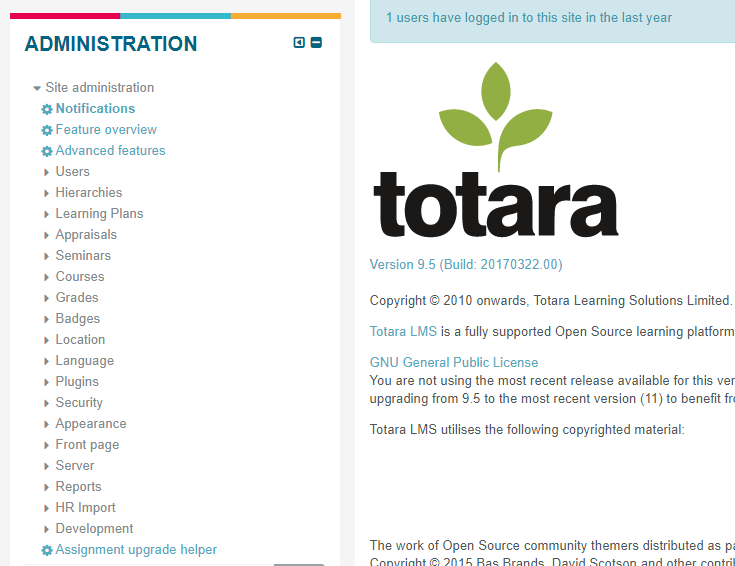


Figure 1 – Totara Learn settings

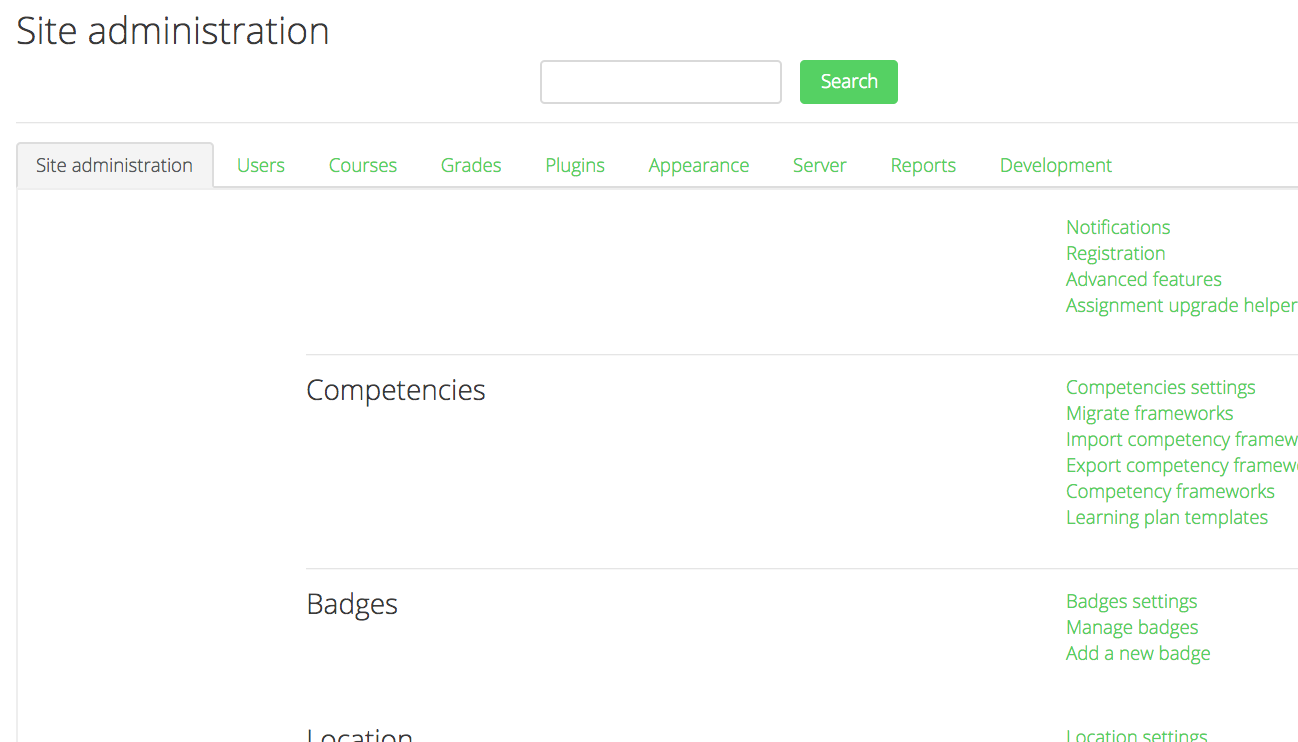


Figure 2 – Moodle settings

The complexity of learning management systems adds significant training costs and time to the already high amount of time and expenses required to roll out a system. According to an industry survey carried out by Capterra, people spend 59% more than they expect to in rolling out their LMS and often end up dissatisfied (26% of users) leading to 31% changing their LMS yearly (Capterra, 2015).

Based on observations within the industry (elaborated upon in section 2.8) the majority of organisations that look to implement an LMS have a learning and development department that oversees planning through to implementation of the system. These are the people in touch with the LMS provider in an almost daily schedule through the process. When the system is web-accessible, the implementation stage begins for the organisation. This implementation more often than not does not start until after the team have been given formal (scheduled training days with a trainer going out to the organisation’s offices to deliver the training first-hand) or informal (over the phone via the LMS provider’s support or project management teams). This not only takes up a considerable amount of training time and costs but also usually leaves the LMS sitting dormant until after training is complete. This is wasting time in the setting up of courses and learning material all contributing to further delaying any possible ‘go-live’ date within the organisation (when the intended end-user is using the system). This process is depicted in figure 3. The complexity of the systems in the most part causes the strenuous nature of this cycle.

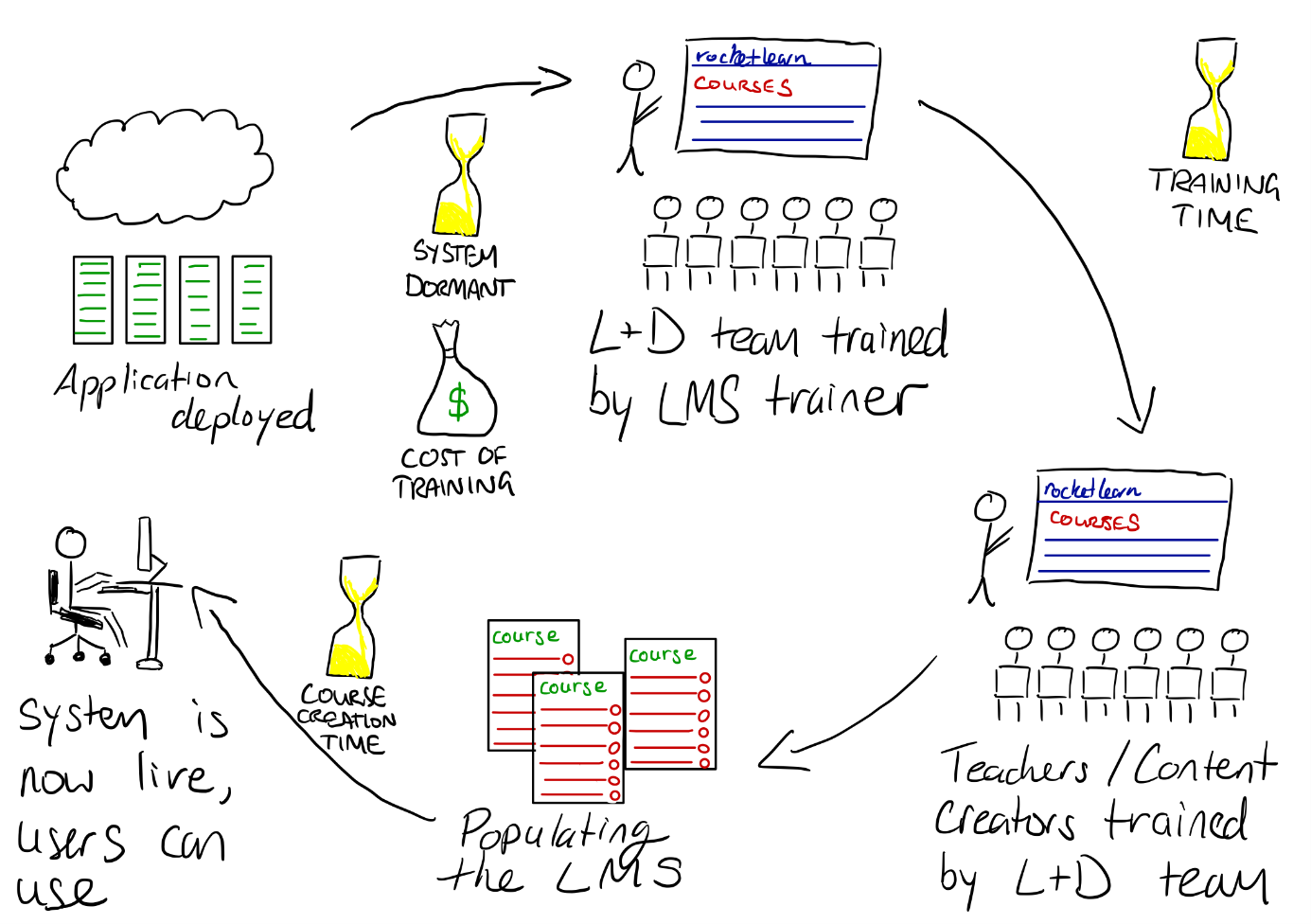


Figure 3 - Depiction of the problem

## 1.2 The Solution

Rocket Learn is a course-centric (a system centred on the main functionality of courses) learning management system (LMS). It allows teachers to create learning content for learners. Learners’ progress is tracked and displayed as part of their learning record and also to allow teachers to track results from their courses.

Courses can have varying levels of visibility. The default visibility setting is for a course to open to all users, they will then be required to enrol themselves on the course before being able to access the content. Courses can also be hidden from all users, where only the teacher and administrator can access, this will be used for courses that are still in development. The third visibility option is ‘restricted’; this is when a course is only visible to those that have been enrolled on the course (as well as the teacher and administrators), meaning the teacher or administrator will have to enrol users on these courses manually. An example use case for a ‘restricted’ course is for closed classes (where a fee may have to be paid before access is granted, for example).

Rocket Learn is a database-driven PHP web application that is built upon the Bootstrap framework. The application has built-in layout responsiveness which alters how on-screen elements are displayed within the application that makes the application usable across a multitude of devices (including mobile and tablets). The system has been designed with simplicity in mind; every aspect has been fine-tuned to decrease complexity where possible. The system does not lose functionality due to the ‘simplicity’ approach; it instead reduces the strain of setting up and maintaining elements of the system (such as courses and quizzes).

The solution is captured in Figure 4. The time and resource constraints have been eliminated; this is because the need for training has been alleviated due to the simplicity of administrating or using the system. Time is also significantly impacted when considering a cloud application deployment versus the standard self-hosting approach. One or more administrators manage the system, while teachers create and maintain courses that will facilitate user learning. User accounts can be created and managed by the administrator(s) as well as via self-registration (if the administrator enables this feature). Once they have an account, users can log on and access open courses and any course they have been enrolled on.

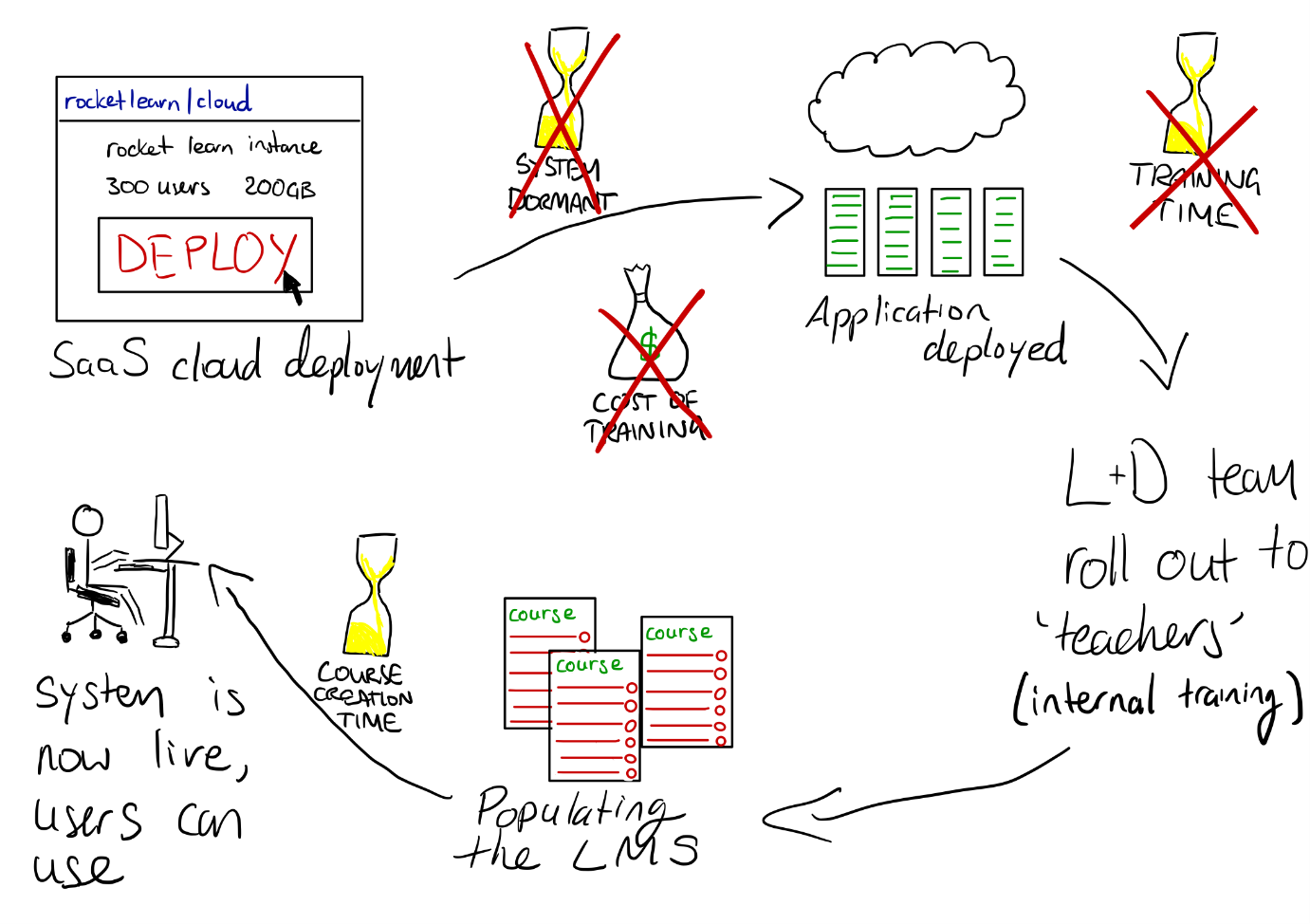


Figure 4 - Depiction of the solution

Following the investigations that were carried out (outlined in the Analysis section), a refined scope was formed for Rocket Learn.

Rocket Learn bridges the gap between complexity and simplicity in the learning management system market; with an aim to reduce the need for professional training, therefore significantly reducing costs.

## 1.3 Dissertation Structure

**Chapter 2: Analysis**

This chapter shows the information that was gathered and analysed in order to produce meaningful requirements.

**Chapter 3: Design**

This chapter outlines the design of the front and back-end of the system, why these decisions were made, as well as describing the tools and technologies used.

**Chapter 4: Implementation, Testing and Evaluation**

This chapter outlines the core system development, explaining functionalities and what role they play within the Rocket Learn system. The system testing is also described in this chapter along with the overall evaluation.

**Chapter 5: Conclusion**

This chapter outlines the project’s conclusion, reflection on the process undertaken throughout the project, where issues occurred, and the suggested future developments of the system.

**References**

This section expands on the references used throughout the report.

**Appendices**

This section contains supporting information that is referenced throughout the report.

# Chapter 2: Analysis

## 2.1 Existing Systems

Learning management systems are not new on the e-learning scene. The first ever LMS was launched in 1991 (EKKO) with a significant injection of systems launching the early 2000s. Some of the current market leaders have been analysed in the sections that follow.

2.1.1 Moodle

Moodle was launched in 2001 and was one of the first open-source learning management systems. It remains today and is one of the leading learning management systems with over 120 million users worldwide (Moodle, 2017). Moodle’s unique selling point is their highly customisable and modular approach to an LMS. The modular aspect of Moodle is used to build and extend on top of Moodle core functionality by using plugins; as of April 2018, there are over 1400 third-party plugins (not developed by Moodle themselves) available to download from Moodle’s plugin directory (MoodlePlugins, 2018). A list of the key Moodle functionalities has been provided in Appendix 1.

2.1.2 Totara Learn

Totara is an LMS that focuses on the corporate market; it is a fork of Moodle code that has many more advanced features on top, all tailored to its target market. Organisational structuring, as well as the ability to review and manage user performance and the ability to customise system reports to the organisation’s needs. A list of key Totara functionalities has been provided in Appendix 2.

2.1.3 Blackboard Learn

Blackboard Learn’s product range spans across all areas of education, business and government, with the majority of their business taking place in the educational sector. Their bespoke ranges are designed to meet the specific industry’s needs. Blackboard Learn remains one of the world’s most popular learning management systems with total user numbers of over 24 million users (Ortner, 2017). Due to their bespoke nature, Blackboard do not list the cost of their products on their marketing website, however it is safe to say it can be one of the more expensive learning management system options on the market; with one report showing Cardiff University disclosing in a freedom of information request in 2012 that their total annual costs of their Blackboard Learn system (including infrastructure and support) was approximately £415,000 (Bradley, 2012).

## 2.2 Document Analysis

Document analysis was carried out on two of the leading LMS products; Moodle, which is marketed more towards educational institutions and has over 120 million users worldwide (Moodle, 2017) and Totara, which is marketed more towards organisations.

Through this research and investigation, it has been found that the Totara Learn LMS has significantly more complex features than the more widely-used Moodle. While Moodle sticks to the essential functionalities for creating, using and maintaining courses, Totara has a vast amount of features on top of this which can be used for analysis, reporting and varying modes of participation on courses. A breakdown of the key features (shown in Appendices 1 and 2) indicates the variance in complexity between the two systems. Complexity is a noticeable issue in most learning management systems.

## 2.3 Stakeholders

2.3.1 Software Developer/Project Manager

The developer and project manager have managed this project to completion and will also carry out all tasks from identifying the requirements to final implementation.

2.3.2 Project Supervisor

The project supervisor has been in an advisory role in aiding the delivery of the project.

2.3.3 School/Organisation Head (Principal, CEO etc.)

Heads of educational intuitions will play an important role in being a customer and providing feedback on the product following the implementation.

2.3.4 Teachers

Teachers will be the course and system administrators of the system when rolled out to an educational institution. Therefore their feedback will be vital.

2.3.5 Students

Students will make up the bulk of users on the system. Learning material will be added by teachers and administrators for their benefit, to complete. The system must be usable by this stakeholder.

2.3.6 Learning and Development Team

Learning and Development teams tend to be the department tasked with rolling out a Learning Management System to an organisation. They will be the most frequent users and administrators of a system when brought into the organisation. Their feedback and ongoing collaboration will be vital for product development.

## 2.4 Survey

A survey was conducted to analyse end-user and professional opinions on what is important about an LMS and what could be improved. The two primary audiences which the survey was released to were a group of final year students and staff members from one of the world’s leading LMS providers (who are based in Northern Ireland), Synergy Learning. This meant that the responses gained from the survey were a mix of the everyday LMS user and working professionals, including LMS experts. The survey consisted of mainly Likert scale survey questions which respondents were asked whether they strongly agreed, agreed, neither agreed or disagreed, disagreed or strongly disagreed. This design approach was used to allow the respondent to express their opinion better, not forcing a binary ‘yes or no’ answer and to uncover a varying degree of views to understand the feedback better. Google Forms was used to build the survey. The survey gained 18 responses which was within the target of 15-20 responses; the data retrieved can be seen in Appendix 5 and analysed in section 2.4.1.

2.4.1 Survey Analysis

**Are you a user or do you work with Learning Management Systems?**

The majority of those that answered the survey were professionals, working with learning management systems on an almost-daily basis.

**Administrators should be able to organise users and courses into segregated groups.**

All respondents agreed with this statement, with two-thirds strongly agreeing. This has highlighted grouping was highlighted as an important feature.

**These segregated group areas can co-exist on a system with 'public' courses. (A public course being one that is open to all users on the system)**

This question refers to courses only being made visible to members of a particular group co-existing in a system with courses open to all users. Over two-thirds agreed, with 11.1% strongly agreeing. This indicates an acceptance that courses, both courses with segregated visibility and those globally accessible can exist on a system.

**Reporting will be integral for teachers and administrators.**

All respondents agree that reporting is an integral tool within learning management systems, with over three-quarters strongly agreeing. This has highlighted reporting as an essential feature of the system.

**Social network interactions are an important system feature to implement.**

Over two-thirds agree (or strongly agree) with this statement. This has highlighted social network integration as a feature that would be useful in an LMS.

**Bulk user actions (such as uploading users to the system) is a key functionality for administrators.**

Almost all of the respondents agreed (or strongly agreed) with this statement, with 5.6% neither agreeing nor disagreeing. This has highlighted bulk-user actions as a critical function of a learning management system.

**Which of these elearning systems do you currently use?**

This question is the precursor to the next two, to gather which systems the respondents use, and what they like/dislike about these systems.

Most of the respondents use Moodle and Totara, closely followed by Mahara (an ePortfolio system, commonly integrated with Moodle to support learning material) and Blackboard.

**List some of the main features you like / dislike on these systems.**

Six out of the eighteen respondents gave a feature that they liked on the systems outlined in the previous question. These varied from individual activities (such as quizzes and glossaries) to overall functionality (such as easy to use and informative).

Disliked features focused on a lot of the complex functionalities on Totara, such as appraisals, HR sync (the ability to sync users from an HR system into the LMS), 360 feedback and goals. Others were generic to the majority of learning management systems, such as long navigation (as highlighted with the high amount of options available in navigation blocks), the complicated language used in learning management systems, and the management of roles.

The complex functionalities as outlined from the responses are those that will be excluded from Rocket Learn to increase ease-of-use in both functionality and administration of the system.

The complicated language was highlighted as an issue within learning management systems, as in LMS jargon overused on applications causing confusion as to what a functionality does (such as organisations, teams and positions within Totara Learn); this is something that has been considered when naming parts of the Rocket Learn system, to help users better understand the functionalities from the offset (such as Record of Learning, being restructured to Learning Record).

**Please feel free to add any additional feedback, possibly a suggested feature or overall improvement.**

“Don’t make it too complicated” was one of the responses given in the ‘suggest a feature of overall improvement’ section. This lines up perfectly with the mantra of Rocket Learn.

## 2.5 Industry Observations

After spending over a year at a learning technology company who specialise in providing, supporting and hosting learning management systems, the following observations were able to be drawn.

2.5.1 Training

The majority of LMS providers (including Synergy Learning) offer professional training services to their customers, where a trainer will spend *x* amount of days on-site with a customer showing them how to use their newly acquired LMS.

At Synergy Learning, it was observed that the average training session is 2-3 days, with some taking the contracted trainers all around the world to deliver the training in-person.

## 2.6 Software Development Methodology

The Rocket Learn project has been developed with the Agile (Kanban) methodology. Due to the project’s large number of facets, the visual approach of kanban works very well. The ability to visualise what stage of development every task is in is a massive advantage through task planning to completion over other methodologies. The Rocket Learn kanban board can be seen in figure 5 below. Each of the tasks are represented in card format with accompanying critical information (such as title, status and priority).

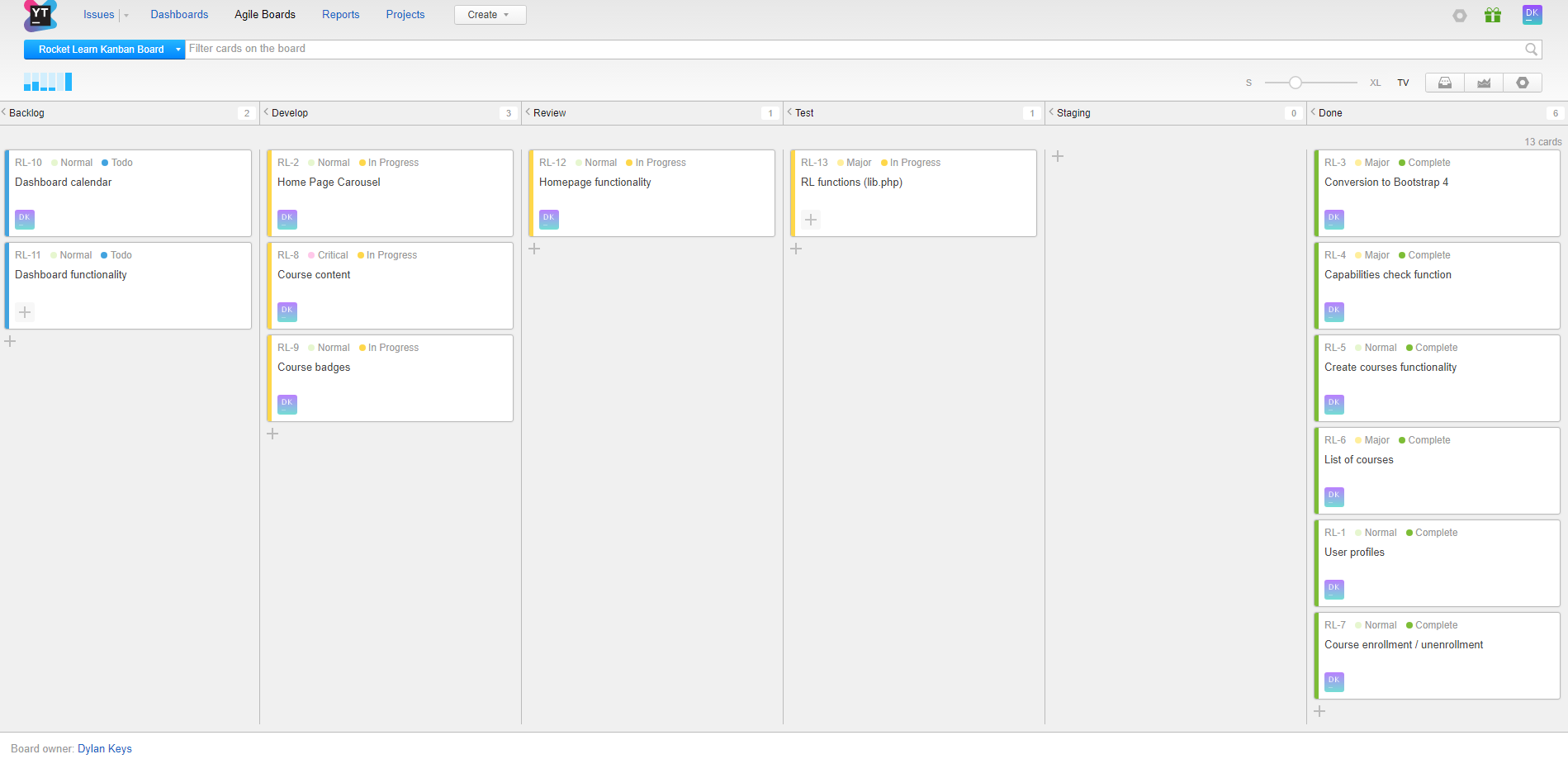


Figure 5 - Project kanban

## 2.7 Professional Issues

### 2.7.1 Business Case

As identified in the project investigation, while analysing two of the leading learning management systems, one focuses on educational institutions, and the other is a more complicated system with a vast amount of organisational tools. The gap in the market is functional simplicity; a need that is not currently being met.

**Self-hosted**

The classic approach to deploying a learning management system is by setting up the hosting environment on the organisation’s/institution’s own server and placing the application code (whether licenced or open-source) on there to be run. This is far from a click-and-deploy service, and requires significant setup time and know-how; this all on top of the sizable hosting costs, which could be as high as £18,000 (Weiss, 2013) and never-ending security/maintenance requirements.

**Cloud hosting (SaaS)**

Due to the time, expertise and cost of a self-hosted setup, the leading learning management systems are now leaning towards cloud hosted solutions, where the user can quickly deploy their LMS at the click of a button. Pricing for these solutions are also usually very competitive when compared with the time and costs of self-hosted solutions; as shown in table 1, fees for cloud hosting for 1000 users from previously analysed competitors. When compared with the value reported for self-hosting, even the more complex of learning management systems work out much more attractively priced.

On top of the ease of getting an LMS instance deployed security will be offered by the cloud provider, taking the majority of accountability off the organisation. This factor is especially highlighted with the introduction of General Data Protection Regulation (GDPR) within the EU (EU, 2016); which requires any organisations that collect or process data from EU residents to meet the guidelines and comes with significantly heavier penalties for organisations that do not meet them. The provider would cover the server elements of GDPR, and give the administrator more time to focus on the system rather than the environment where it is hosted.

Table 1 - LMS Cloud Hosting Costs

|  |  |
| --- | --- |
| LMS | Cost |
| Moodle | £550 per year |
| Totara | £7,300 per year |

2.7.2 Technical Benefits

* Every aspect of the system has been developed with the task of decreasing complexity in mind. This allows for combining features that are usually only found on complex systems (such as course badges and user learning records) into Rocket Learn, along with the core-LMS functionalities (such as courses and activities). The complex features are essential to increase the usability of Rocket Learn by making its use feasible in more categories (such as education and corporate).
* Ease of bulk-user uploads, useful for larger institutions
* The application can be used on almost any internet-enabled devices and remain easily usable, due to the responsive design (Mobile CSS and Bootstrap), allowing the frontend to be resized to the varying device screen sizes.

2.7.3 Costs

* Developer/Project Manager – £30 per hour – estimated 263.5 hours - £7,905
* Yearly application hosting costs – 1&1 Web Hosting £4.99 per month - £59.88
* Annual domain fee – rocket-learn.com - £9.99

## 2.8 Defining Scope

2.8.1 In-Frame / Out of Frame

In Frame / Out of frame analysis was carried out on the project in order to reduce the scope and avoid ‘scope creep’; the diagram can be seen in figure 6. *In-Frame* deliverables are those tasks that are part of the project scope and therefore implemented as part of the system. *Out of Frame* tasks are those that have fallen outside the boundaries of the project scope (due to resource constraints, such as time) and are not be implemented in the system. Tasks identified as *On the Frame* are those that are not essential to the functionality of the application and will only be implemented if the scope allows for it after the *In Frame* components have been completed.

**In Frame functionalities**

Core

* **Courses** – made up of topics and their elements, created by teachers and completed by students.
* **Dashboard** – used to display a user’s progress on courses, and to access their learning record (a PDF of their completed courses) if they have completed courses.
* **User Profile** – containing user information, with the ability to edit their stored information, including password.
* **Responsive design** – allowing the application to be used on mobiles as easily as it is on a desktop.
* **User permissions** – restricting certain areas/pages to those with a role that has specific permission.

Course elements

* **Assignments** – completable activity on a course, allowing a teacher to set a deadline for a file submission for students that is subsequently graded.
* **Site pages** – a page resource on the site that can be used to convey information.
* **Quiz** – a multi-choice quiz that can be set up by teachers for students to complete, a pass grade is set at the creation of the activity allowing grades to be automatically calculated when a user attempts the quiz. A start and end time is also set on creation, with the quiz only being available during these date/times.
* **Video** – an embedded video resource from YouTube.
* **File** – a file resource that is attached to a course for download by students.

Administration

* **Application config** – the main application settings, including the site name, navigation colour and the ability to enable/disable user self-registration.
* **User management** – a list of all users on the system that can be deleted, along with a create user account option.
* **Reports** – a page of brief system statistics for administrators.
* **Slideshow** – customisation options for the homepage slideshow, slides can be edited, deleted and created.

Miscellaneous

* **Learning record** – a downloadable PDF that is populated with a user’s completed courses.
* **Course enrolments** – the ability for a user to self-enrol, or be manually enrolled by an administrator/teacher. Teachers are also enrolled on courses via the role select option on the course’s manual enrolment page. Any enrolled user can unenrol by selecting the unenrol option on the course page.
* **Course completion** – the ability for a course to be marked as complete when a student has completed all of the elements within the course completion criteria. The completion criteria can be set by a teacher or administrator. The completion criteria are selected for a list of all the activities that are on a course. A cron job has been set up that runs every minute checking for new course completions. A user’s progress on a course is indicated on their dashboard (within the completed courses table) and on the course progress bar.
* **Element completion** – this is calculating if a user’s given grade is equal to, or higher than the set pass grade on the activity.



Course enrolments

Learning record

Slideshow

Figure 6 - In Frame / Out of Frame diagram

2.8.2 Triple Constraints

The project triple constraints are three of the most significant restrictions on the project. They are the scope, schedule and resources. The triple constraints are linked, and therefore one cannot be changed without one or both of the other constraints also being affected.

Being aware of the project triple constraints in planning significantly increases the project’s chance of success, as mitigating factors against will be considered in designing to avoid failure in meeting the defined deliverables.

**Scope**

When planning the scope of the project, it would be impossible not to consider the available resources and have the project finish successfully. The schedule and resources are relatively well defined in this project and were applied to every deliverable in the plan.

**Schedule**

The schedule has been known from day one (deadline day, 4th May 2018). This again could not be discounted from any planning, or else the project would no doubt not be finished in time for the deadline.

**Resources**

The available resources are relatively simple in this project, in that it is known who will be managing the project and developing the system and the fact that this will not change throughout the project timeline.

The available facilities were also easily defined regarding the machines the project will be developed and planned on (university computer labs and personal devices).

Server space and the constraints that go along with this (database and server storage) have been taken into consideration with every planned deliverable. As if any deliverable exceeded these constraints it would not be possible to implement and cause issues.



Figure 7 - Triple constraints model (Project Management Institute, 2013)

## 2.9 Ethical Considerations

The ethics of this project was approved and deemed to be under Category Z - This project does not involve invasive interaction with people and does not require a full Ethical review. This includes the use of anonymous questionnaires for testing software etc. I confirm that this project meets the definition of research in the category above.

* All risks and ethical, procedural implications have been considered
* The project will be conducted at all times in compliance with the research description/protocol and by the University’s requirements for recording and reporting
* This application has not been submitted to and rejected by another committee.

2.10 Project Risks

Project risks were analysed as part of the requirements gathering phase, the risk register that was generated can be seen in Appendix 14.

Chapter 3: Design

## 3.1 System Requirements

3.1.1 Functional Requirements

The functional requirements of Rocket Learn have been defined in table 2.

Table 2 - Functional requirements

|  |  |
| --- | --- |
| Requirement | Description |
| Account creation | **Self-registration** The user must create an account to be able to enrol in courses. However, self-registration can be enabled/disabled by an administrator.  **Manual creation**  Administrators can manually create accounts for users. This is the only method of account creation available if the self-registration has been disabled. |
| Access courses (Enrolment) | Users must be manually enrolled or self-enrolled on courses to access and complete the course |
| Administrative settings | Administrators and those with the relevant will be able to access system and course settings to manage the system. Examples include site name changes and navigation colour changes. |
| Levels of authorisation | Role management and assignment will dictate which permissions the users will have. For example, system administrators will have system-wide permissions, whereas teachers will only have control of their courses. |
| Reporting | Following the investigations on the project, it was very evident that reporting would be a critical piece of functionality for the users – whether this is system reports or user-based reports such as course completions. |
| Course creation | Administrators can create courses and assigned a teacher who will have full control of everything that goes on inside the course. A safety precaution in place is that teachers cannot delete courses, this should reduce any chance of accidental deletions, as fewer users will have this privilege. |
| Course visibility | Courses can be setup with three different visibility options:   * **Open** – open to all users for self-enrolment. * **Restricted** – manual enrolments only, students can only view/access course if enrolled. * **Closed** – Hidden from all student users. |
| Site page creation | Site page creation for administrators. These can help delivery items such as news and policies. All site pages will be accessible from the homepage. |
| Quiz creation | Multi-choice quizzes can be created by teachers and administrators. The passing grade can be configured so that when a user completes a quiz their grade can be deemed as a pass or fail automatically. The quiz is created with start and end date/times that mean the quiz is only available between these times. Questions can be added and removed from the quiz with varying amounts of answers. The answers also display in a random order to the user, so there is no set sequence for correct answers. |
| Assignment creation | Assignments can be created that require a file to be uploaded by a student before the set deadline. Once the submission is made then the teacher can download the submission, give feedback and grade the effort. |
| Embed a video | A youtube video can be embedded on courses by providing a youtube link. This converts the link to embed code and displays on the course topic it was created on. |
| Embed a file for download | Files can be added to courses for download by students. |
| Add a web link to a course | Web links can be added to courses for students to access. |
| Add, edit and delete course elements | Teachers and administrators can add, edit and delete course elements and configure them to be part of the course completion criteria. |
| Complete courses | When a user meets/completes the set completion criteria for a course, they should be marked as complete. Once a course is completed, the user will receive a badge in said course. |
| Timestamps appended to uploaded files | When a user uploads a file, whether it be for an assignment submission, or as a resource on a course, a timestamp is appended to the file name to ensure every file has a unique name. |
| Progress bar | The progress bar indicates how much of the course completion criteria a student has completed. |
| Learning record PDF | A PDF is generated at the user’s request that contains a list of their completed courses on the system. This can be downloaded at the user’s discretion. |
| Course search | The ability to search through the available course catalogue, the search field will allow the user to search the database for the title of the course. |
| Site news | The ability for administrators to add news/posts to the homepage. |
| Change site theme (navigation bar colour) | Administrators can change the colour of the site navigation bar from the application configuration settings. |
| User dashboard | The user’s dashboard is used to help them keep track of their activity on the system; that being the courses they are currently in progress on and the ones they have completed. |

3.1.2 Non-functional Requirements

The non-functional requirements for Rocket Learn have been defined in table 3.

Table 3 - Non-functional requirements

|  |  |
| --- | --- |
| Requirement | Description |
| Stability | The Rocket Learn application is hosted externally with 1&1 who offer a guaranteed uptime of 99.9%, as well as providing 24/7 support for those rare occasions that an issue may occur. |
| Performance | 1&1 have some of the best full page load times at an average of 2.21s (WebhostChecker, 2017). |
| Usability | The ease of use in the application. This has been well refined regarding jargon and user interface. New users should require next to no training to use. The LMS is used to facilitate learning and therefore won’t require the learner to be on the system for long periods of time. The most extended expected stint on the application would be to complete quizzes which can be used for hour-long tests. |
| Scalability | The system scale can be increased and decreased based on the number of users and courses. These can be created and deleted by those with the relevant system permissions. In scaling up extra server resources may be required to facilitate this, this could require a reconfiguration of self-hosted infrastructure. For cloud (SaaS) customers the resource scalability would happen automatically when the customer purchases a higher tariff. |
| Security | The system can be browsed by guests, but the main functionalities are only available to authenticated users (those who have logged in). User self-registration is only an option if the site administrator enables it, otherwise user accounts must be created manually by administrators. |

3.1.3 Actors

The system actors have been listed and described in table 4.

Table 4 - System actors

|  |  |
| --- | --- |
| Actor | Description |
| Guest | A guest is a non-authenticated user that has not logged in. Essentially every user is a guest until they log in. Guests may access some pages, but with substantial system restrictions. |
| Standard | A standard user is one that is logged in (authenticated user) to the system but has no administrative rights. |
| Students | Students will be authenticated users that have enrolled on and attempting to complete courses. |
| Teacher | Teachers are essentially course administrators; they manage their courses and the users participating. |
| System Administrator | System administrators are at the top of the chain and full administrative control of the application. |

3.1.4 User Stories

The user stories for the system actors are shown in table 5.

Table 5 - User stories

|  |  |  |  |
| --- | --- | --- | --- |
| Actor | Wants to… | Because… | Completion Criteria |
| Guest | Explore the available courses. | They are interested in signing up for the application. | User searches courses from the homepage or browses them on the courses page. |
| Standard | Explore the system with the intention to enrol on a course | They are signed up to the application to learn but have not yet enrolled | Successfully browses the system as a standard user. |
| Student | Enrol on a course. | They would like to become proficient in the area of knowledge. | Student logs into the system and finds a course via search or from the courses page and clicks on the link. |
| Student | Complete a course | They would like to improve their learning record and knowledge of the course’s area. | The user completes all of the activities within a course’s completion criteria. |
| Teacher | Set up a course. | It must be set up ready for learners to enrol. | Course settings edited to the teacher's specification, the relevant course elements added and course completion is configured. |
| System Administrator | Assign a teacher to a course. | The teacher needs to set up course for learners to enrol. | Enrol teacher in course creation or afterwards in the course enrolments. The teacher will now be assigned to course. |

## 3.2 User Interface Design

3.2.1 Purpose and Communication

The site message is courses. Courses are the most integral part of Rocket Learn; it is the purpose that users will be interacting with the system. The pages outside of the courses, including the homepage, dashboard, contact and profile, play a support-like role. Every page is associated in one way or another with a course; for example, the homepage allows users to search for courses, and the user’s dashboard holds data on the users enrolled courses, such as completed and actively working on courses.

The use of headings and sub-heading on pages that serve more than one function are essential in communicating page functionality and delivering the intended information to the end-user. Course pages and their elements are sectioned and headed with the use of topics (boxes that differ in colour to the page background).

3.2.2 Typefaces

The san-serif font-style has been selected for Rocket Learn based on its increased online readability compared to the unnecessarily fancy styles within serif fonts (Neumark, 2018). The san-serifs fonts used will be inherited from the Bootstrap CSS ("Helvetica Neue", Helvetica, Arial, sans-serif).

3.2.3 Responsive Design

Responsive design is achieved by making use of the Bootstrap grid layout. Along with the use of the *@media* rule within CSS3 to create specific mobile styling to enhance the grid layout in areas that are visually untidy.

3.2.4 HCI Considerations

Colours

The aesthetics of Rocket Learn will be altered by using the Bootstrap framework which will aid the delivery of a neat and tidy, professional system. The Rocket Learn colour scheme has been selected to set a light tone to the system, this is shown in figure 8. The system will be used for users to learn and be assessed, and therefore must be ‘easy on the eye’. The colours do not conflict and therefore meet HCI principles of luminance and colour diffusion. This was selected following an investigation into light aesthetically pleasing colours and analysing other top sites with the similar colour scheme (such as Facebook, Dell, American Express, GE). This can, however, be changed by a system administrator in the settings to match an organisation/school branding.

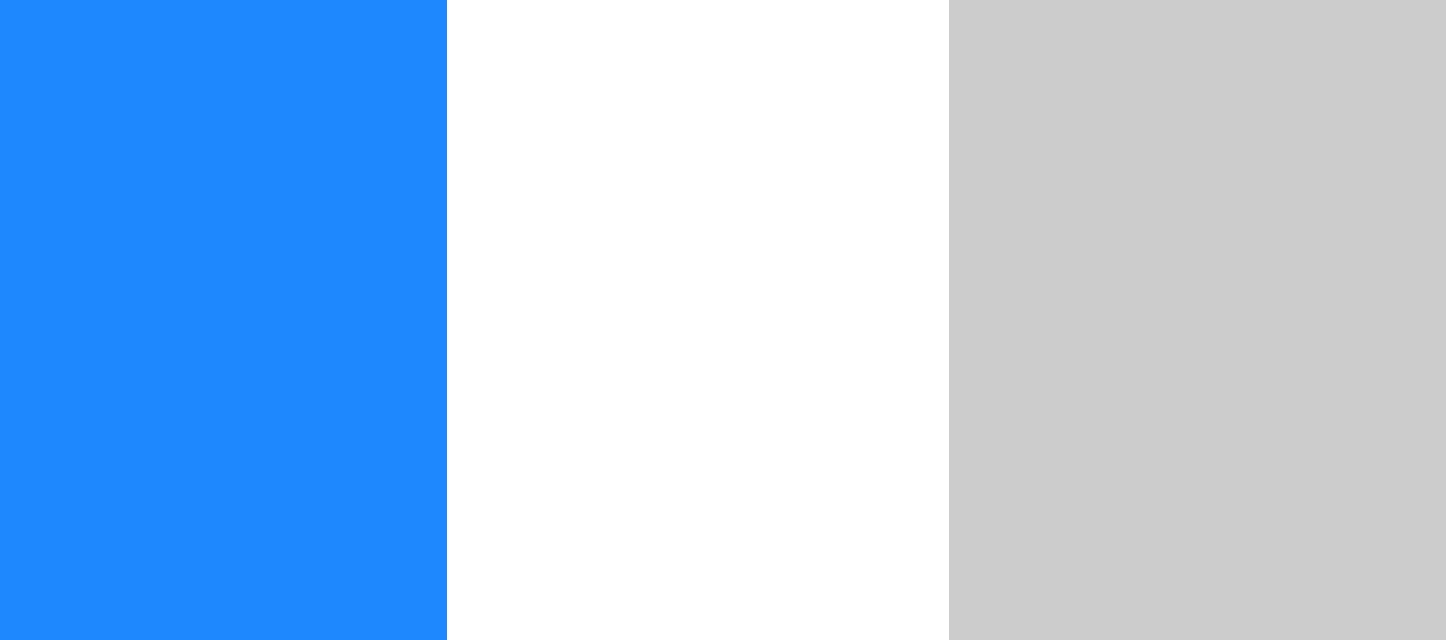


Figure - Rocket Learn colour scheme

Consistency

The navigation bar will remain consistent throughout all site pages, enabling the user to navigate to any of the core system pages (home, course catalogue, contact and profile, also the administration page for administrators).

### 3.2.5 Overall Layout

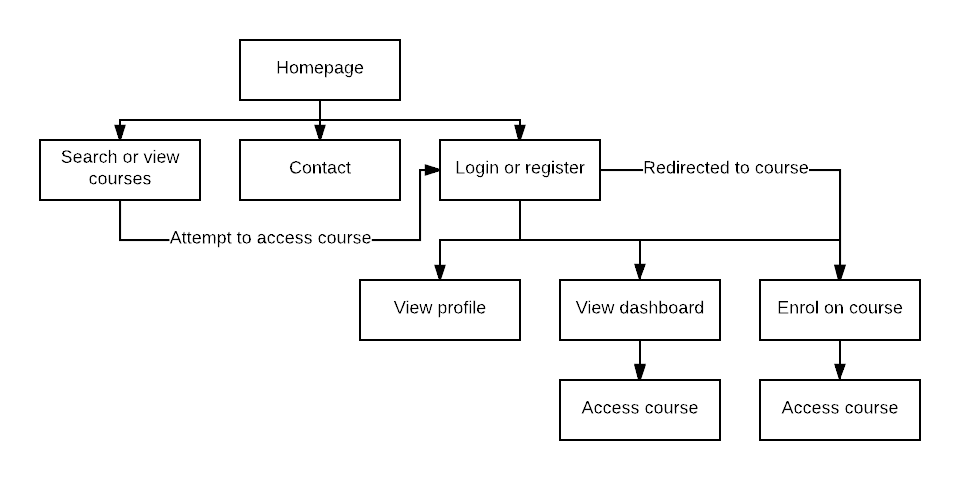


Figure 9 - Layout flow

Figure 9 shows the user interface navigation of a learner user (who is using the system to complete courses). Before any user logs into the system, they have access to three main functions: the available courses list, the contact page, the login/register page. All other pages will redirect the user to the login page for them to log in or register; if this is a course the user will be returned to that course they attempted to access after they have logged in or registered. Once logged in the user will have access to their profile, dashboard and the available courses.

Appendix 4 shows the system administrator user interface navigation. Until they log in, they are treated as a guest as they have yet to be authenticated as a system administrator. The system administration options are shown when the administrator access the systems settings; these can be seen in the appendix.

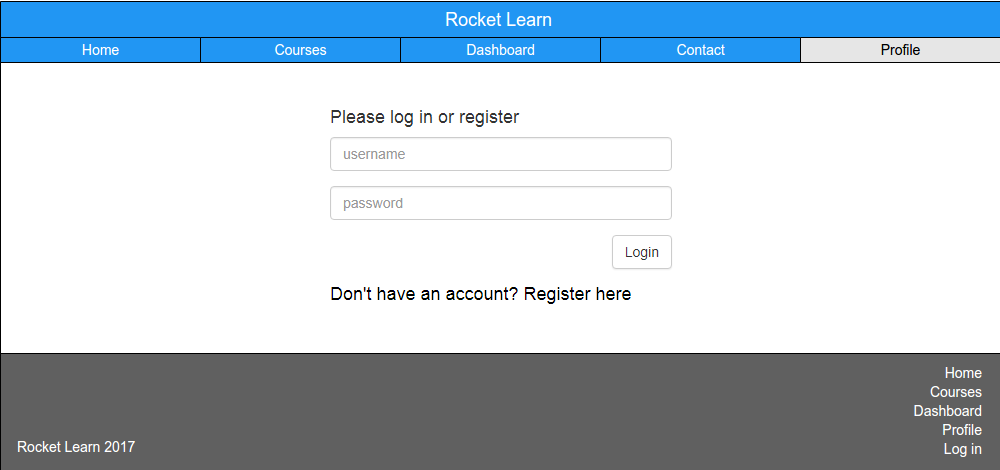


Figure 10 - Layout example, Rocket Learn login page

The application’s navigation bar and footer will predominantly remain consistent throughout the application. The only exceptions to this will be active navigation buttons depending on what page the user is on, whether they are logged in or not and the role they have on the system (for example an administrator will have access to the administration settings). Simplicity was the key element for the navigation, as complicated navigation areas are something that was picked up during the competitive analysis of learning management systems; for a standard user, there will only be five option to choose from in the navigation, all of which contain no convoluted LMS jargon. The footer follows the same principles as the navigation bar, simplistic. The footer features a login link, which changes to “Logged in as <user full name>” when a user is logged in; alongside the top layer page links.

The application layout changes depending on the screen size of the device that the user has logged in with. As you can see in figure 11, the navigation bar will become condensed and the buttons transferred into a drop-down menu. The remaining wireframes for the main site pages can be viewed in Appendix 3.

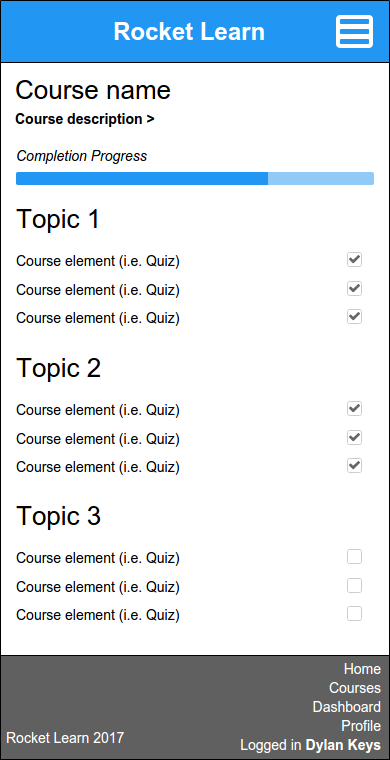


Figure 11 - Mobile layout wireframe (course page)

3.2.6 Course pages

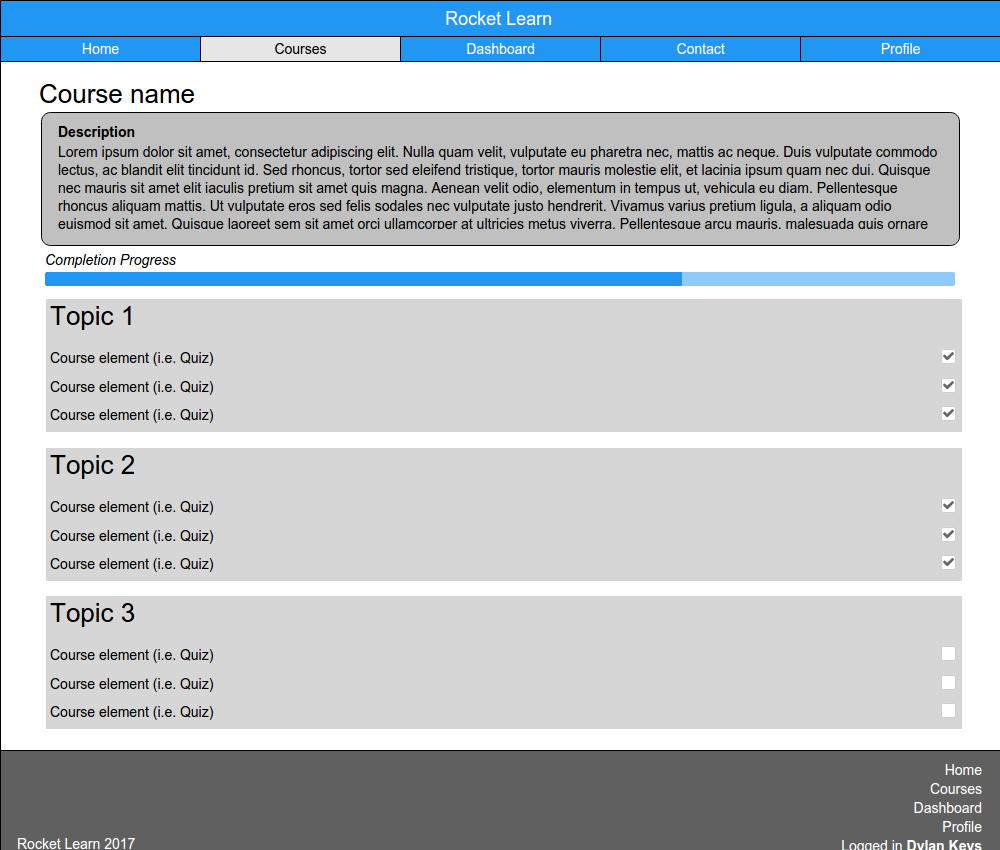


Figure 12 - Course page wireframe

The course page is the one that deserves the most design attention due to its importance within the system and the fact that users will spend the majority of their time on the system on these pages.

The course page is headed with the name of the course; this is important information that indicates what the course the user is on. It is the first thing the user should see as the page loads, which is essential (to know that the page you are on is the intended one).

The next element is the course description that will be set by the teacher/admin to communicate what the course entails to the user; this will also be displayed on the enrolment page to aid a learner’s decision on whether to enrol onto the course or not.

The course progress bar is located under the description, this takes into account the course completion criteria (as set by the teacher/admin) and works out how many of those elements the current user has completed; this progress will then be depicted on the progress bar. It helps the user understand how much of a course they have completed/still to complete.

The course topics then take up the remainder of the page. These are the sections where the course elements are placed in by the teacher/admin. The topics are made up of the header, summary (to describe the section) and the course elements. The course elements will be accompanied by:

* An icon that matches the element type
* A link for the user to access the element
* A checkbox that indicates whether the user has completed the element; this makes it simple for the user to see which elements they have completed and yet to complete, which only adds to the completion visibility as provided by the progress bar.

Figure 12 shows a student’s view of the course page. When administrators or teachers (on a course they are assigned to) access a course page they will be presented with course administrative options that allow them to edit the course settings, add/remove/edit a topic, add/remove/edit a course element on a topic.

3.2.7 Dashboard

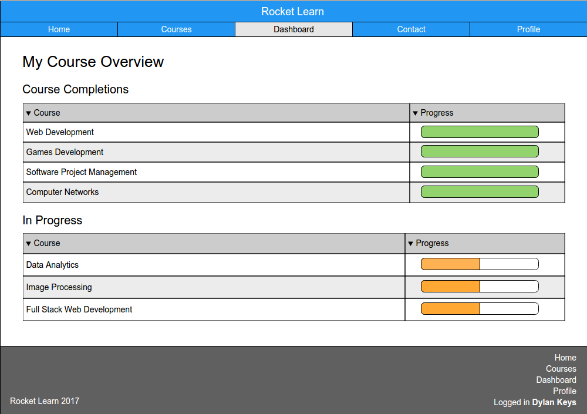


Figure 13 - User dashboard wireframe

The user’s dashboard is their system activity overview. Users can access this page to view their progress on any courses they have been enrolled on. Two tables will be used to show the user their completed courses and any that are currently in progress. A progress bar will also be displayed to indicate whether the course is in progress or complete. Courses in progress will be marked with an orange progress bar, whereas complete courses will have a full green progress bar. Users will also be able to access their learning record from their dashboard, which will contain their completed courses, this can be exported from the system for use in the user’s portfolio or CV.

3.2.8 Settings pages

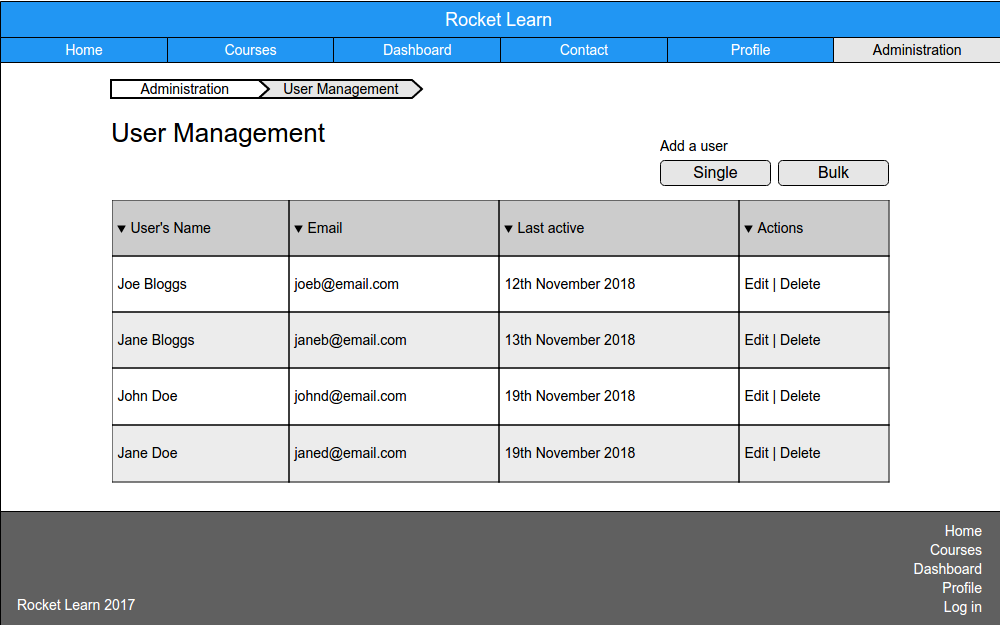


Figure 14 - User management admin page wireframe

Settings pages are accessible to site administrators from the *administration* page. Each of the settings pages can go several pages deep and could leave the user feeling lost had page breadcrumb not been added; this shows the pages the user has traversed from the top level administration page, making it easier to navigate back to a previous page via the breadcrumb links.

Each of the settings pages have different elements that correspond to the functionality in question. In figure 14 the user management administrative page wireframe can be seen, which includes a table with administrative actions and the breadcrumb links as mentioned previously.

3.2.9 Actor Use Cases

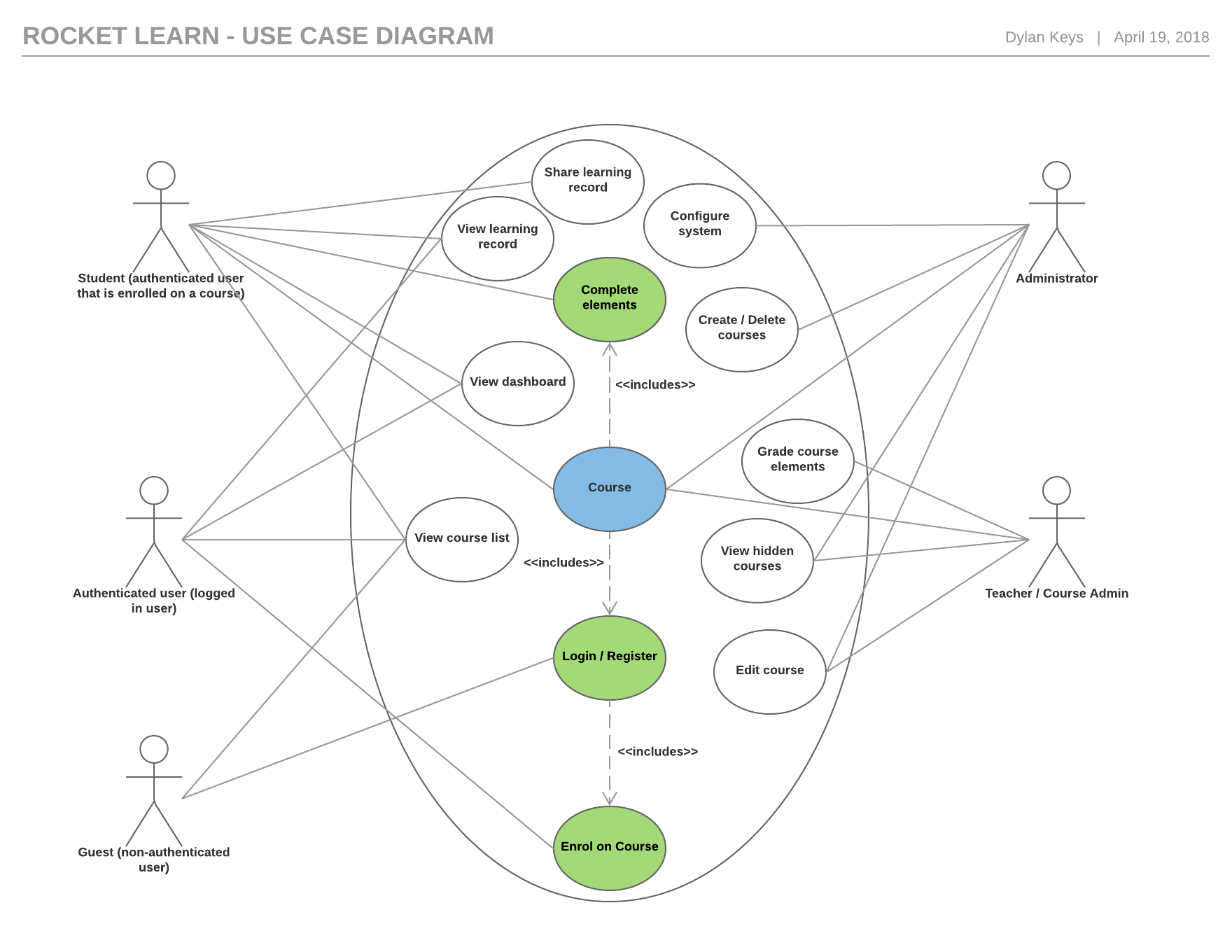


Figure 15 - Rocket Learn use case diagram

The use case diagram (shown in figure 15) indicates which actors have access to specific core functionalities, revolving around courses. Administrative roles and functionalities are segregated to the right of the diagram, with standard functionalities on the left; it should be noted that the administrative actors (administrator and teacher) can access all of the functionalities listed on the left, the connecting lines were left out for tidiness.

## 3.3 Architectural Design

The application is hosted on an Ubuntu 16.04 Linux server running PHP 7.1 which has significantly better performance statistics that PHP 5.6. The application’s database runs on MySQL 5.5. The front-end will be developed in HTML, CSS, JavaScript and PHP (with Bootstrap 4). The backend will be developed in PHP and SQL, with use of the PHPMyAdmin panel. The web server making the application publically available on the web is Apache.

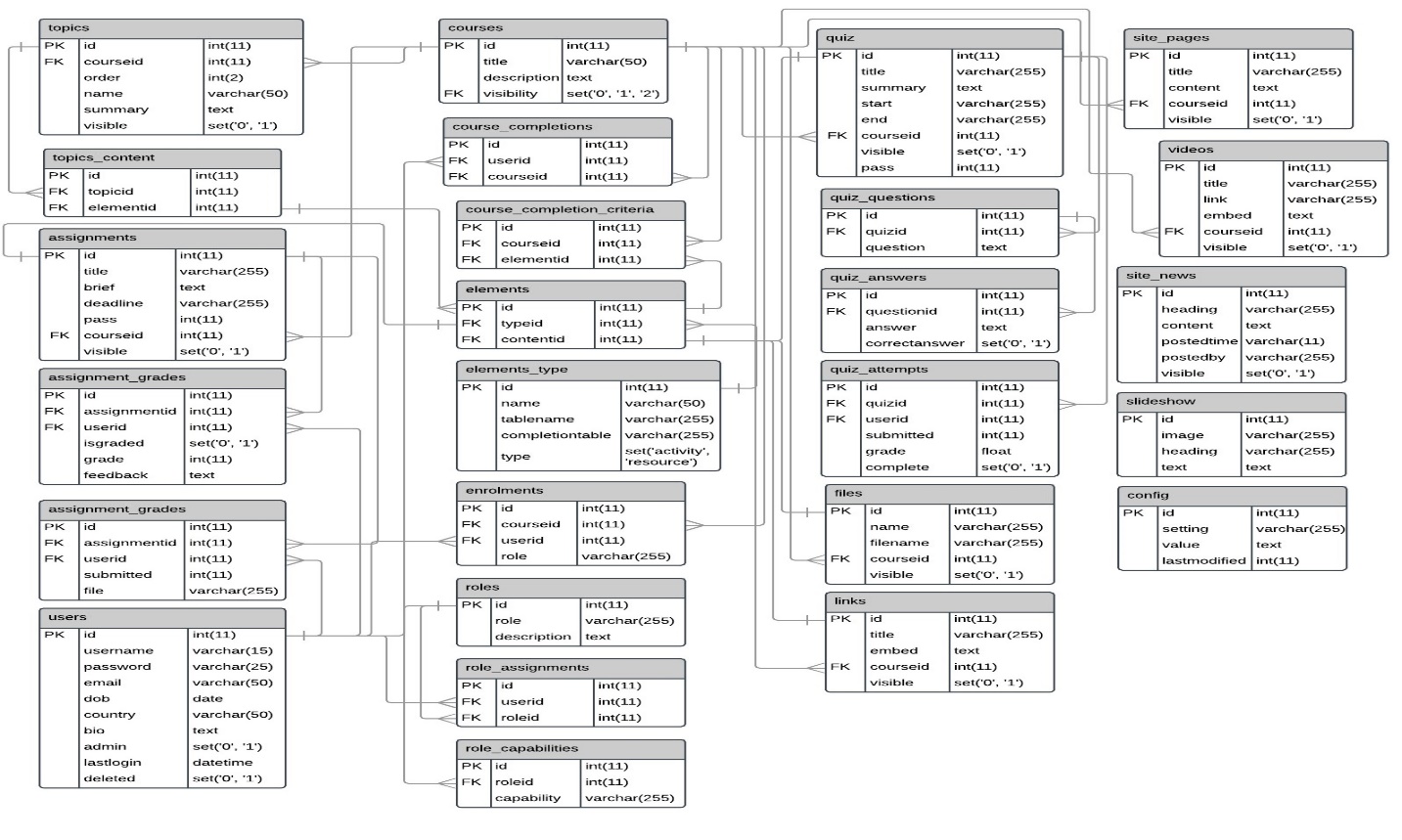
3.3.1 Database

Figure 16 - Database Design

Figure 16 shows the applications database design. The keys have been noted as PK and FK. PK indicates a unique field, the primary key. FK indicates a foreign key, in reference to the primary key of another table. The relationships are shown via the one-to-one and one-to-many lines.

3.3.2 Database Tables (Core Components)

Table 6 – Core components database tables

|  |  |
| --- | --- |
| Table | Description |
| user | This table will store user’s information, along with unique fields such as *lastlogin* which will hold the last time the user logged in, this is stamped on every user login. |
| groups | This table stores data on all the system groups and any exclusive settings the group may have (such as different theme colours/logos). |
| roles | The roles table stores the data on each role including the context it can be assigned (i.e. is it a system or course role). |
| role\_assignments | This table is a log of all the role assignments. The foreign keys of *userid* and *roleid* refer to the user and role tables to reference the user and the role they were assigned. |
| role\_capabilities | This table holds the specific system capabilities that are required for users to access specific functionalities. Each capability is assigned to a *roleid* and therefore may appear in the table more than once (if it is assigned to more than one role). |
| contexts | The contexts table is a collection of all the contexts within the system. Contexts are predominantly used to reference where something has been assigned (is it system-wide or specific to an area such as a course). |
| elements\_type | This table is a collection of all the course elements that can be added to courses for users to complete. The table holds each element’s specific database table; this is used to populate certain system queries dynamically, depending on the element selected within the application. |
| courses | The courses table is a collection of all the courses created on the system and their individual settings. |
| enrolments | The enrolments table contains a log of all course enrolments. This will be any user than is enrolled on a course. |
| course\_completions | This table is a log of all the course completions on the system. When a user meets a course’s completion criteria a new entry will be added to this table. This will be used in user’s Learning Record. |
| course\_completion\_criteria | This table keeps a log of the completion criteria that have been added to courses. |
| config | The config table holds all of the system settings and their values. This table will be used heavily throughout the system settings. |

3.3.2 Database Tables (Course Elements)

The tables for course elements are made up of a group of core fields (on top of their specific fields):

* **Title/Heading -** This is used to identify the particular instance of the element on the GUI.
* **Summary, Content or Brief -** This is the text used to describe, instruct, or inform the user when the element has been accessed.
* **Course ID -** The course ID is used to check if the user attempting to access the element is enrolled on the course on which the element was created.
* **Visibility -** This indicated whether standard system users (non-admins) can access the element. If set to ‘1’ the element is considered visible otherwise it will be set to ‘0’ indicating that the element should be hidden.
* **Specific fields -** Most elements will have their own specific fields to meet their unique functionalities; these are outlined along with a description of the table in table 7 below.

Table 7 - Course element database tables

|  |  |
| --- | --- |
| Table | Description |
| assignments | This table stores any assignments created, along with their settings. The assignments table has a unique deadline field that indicates when the assignment will no longer be available to students for submission. |
| files | This table holds the data relating to files that have been added to courses. The files table has a filename field that is specific to its functionality, to reference the file that has been stored on the server. |
| quiz | This table holds the data for quizzes, on top of the standard course element fields the quiz table has multiple unique fields; the start and end fields mark when the quiz opens and closes for student attempt, and the pass field indicates what score is required for the user to complete the quiz. |
| site\_pages | This is a basic table that holds the data related to site pages. These can be created to stand alone, or on a course, therefore the *courseid* is set to null by default and only populated when a site page is created on a course. |
| videos | The videos table holds the data in relation to youtube videos added to courses, stores the original link, and the video embed code, in addition to common fields. |
| links | The links table works in a similar manner to the videos table, with the embed column being used to hold the HTML links that will be embedded on to the course page. |

3.3.3 Database Tables (Course Topics)

Four tables (including the element type’s table, such as assignments) are used to connect a course element to a specific course topic.

Firstly, when an element is added to a course, the element type table (e.g. assignments) is populated with the settings input by the creator.

Table 8 - Course topics database tables

|  |  |
| --- | --- |
| Table | Description |
| elements | The id from the newly inserted row in the element type table is then inputted into the *elements* table (*contentid*) where a unique element ID will be automatically assigned (*id*) along with the ID associated with the element type (*typeid*), this allows an element’s type to be identified from this table. |
| topics | This table holds the data associated with a topic on a course. The *order* field allows for the topics to be reordered on the course page, as the SQL query includes an *order by* on this field. The visible field also allows for a topic to be hidden from standard users. |
| topics\_content | The *id* field from the elements table is then inserted into the *topics\_content* table along with the unique topic ID (a foreign key associated with the primary key in the topics table). This binds the relationship between an element and a course topic. |

3.3.2 Database Normalisation

The normalisation process was carried out on the database to reduce data redundancy and improve the data integrity. Data that applies to multiple records, such as the element types have been placed in their own specific table and relationships created where this data applies via a foreign key.

# Chapter 4: Implementation, Testing and Evaluation

## 4.1 Implementation

The Rocket Learn implementation has been broken down into deliverables and tasks; these can be seen in Appendix 6. These tasks will be handled using an Agile (kanban) methodology, as outlined in section 2.6. A project management tool has been set up to use throughout this project using this methodology (YouTrack, 2018). The system code and database is hosted on a 1&1 shared web space.

The initial version of the application was developed in Bootstrap 3, as at the time of starting development this was the latest major version. During system development, Bootstrap released version 4 of their framework. A decision was made halfway through development (December 2017) to make the switch to Bootstrap 4 and take advantage of the new and improved framework. This resulted in some major cosmetic changes to the application, as a lot of the previously used components had changed or become depreciated.

4.1.1 Tools used

The project code (HTML, JavaScript, PHP and CSS) was developed between two source code editors, Sublime and Notepad++. These editors were selected due to their superior PHP syntax highlighting, making code easily readable during development. The code was version controlled by using Git (via GitHub), changes were pushed back to the repository at important intervals, ensuring the code was backed-up and version controlled (Keys, 2018). The MySQL database was hosted on 1&1 and managed through PHPMyAdmin.

4.1.2 Navigation bar

The navigation bar sits atop every page on Rocket Learn. It has two main parts; the navigation section with links to core pages within the system (aligned to the left) and the user section (on the far right). The breakdown of the elements is shown in table 9.

Table 9 - Navigation bar breakdown

|  |  |
| --- | --- |
| Image | Purpose |
|  | To enable navigation to core pages within the system from any page. |
|  | To indicate the current logged in user and provide quick links to the user’s profile and to log out. |
|  | To enable quick sign in/up from any page, when the user is not logged in. |

4.1.3 Courses

The course catalogue shows the courses page with the list of all the created and active courses pulled from the courses table in the database. The user is able to search this catalogue.

Figure 18 shows that a standard user has attempted to access a course but is not enrolled. Therefore the system will display the user the enrolment form. The user must now enrol on this course before gaining access.

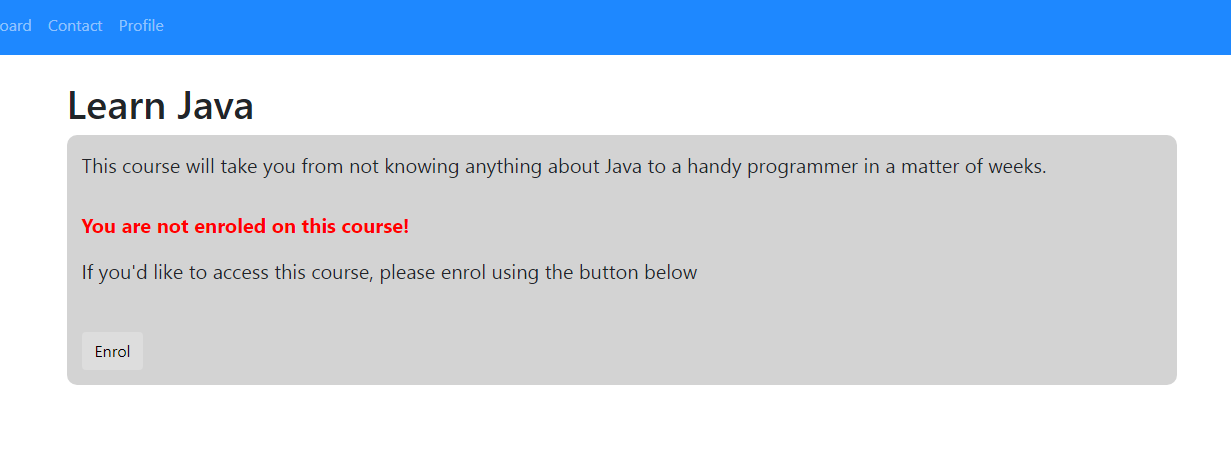


Figure 18 - Enrolment screen for a course

Figure 19 shows the aftermath of a standard user clicking the enrol button. They have been directed to the course that they just enrolled on and now have access as a student. They will now be able to access and complete the elements throughout the course’s topics. The user has access to one ‘course button’ (located below the description), this enables them to unenrol from the course, thus removing their access.

The course page shows the user their progress towards the course’s completion criteria, as set by the teacher/administrator.

The user will leave the course page when they click on the available course elements that have been added by the teacher/administrator. In the example shown in figure 19, the course has three visible topics; Introduction, Assignment 1, and Exam Revision. Each of these topics contain a course element.

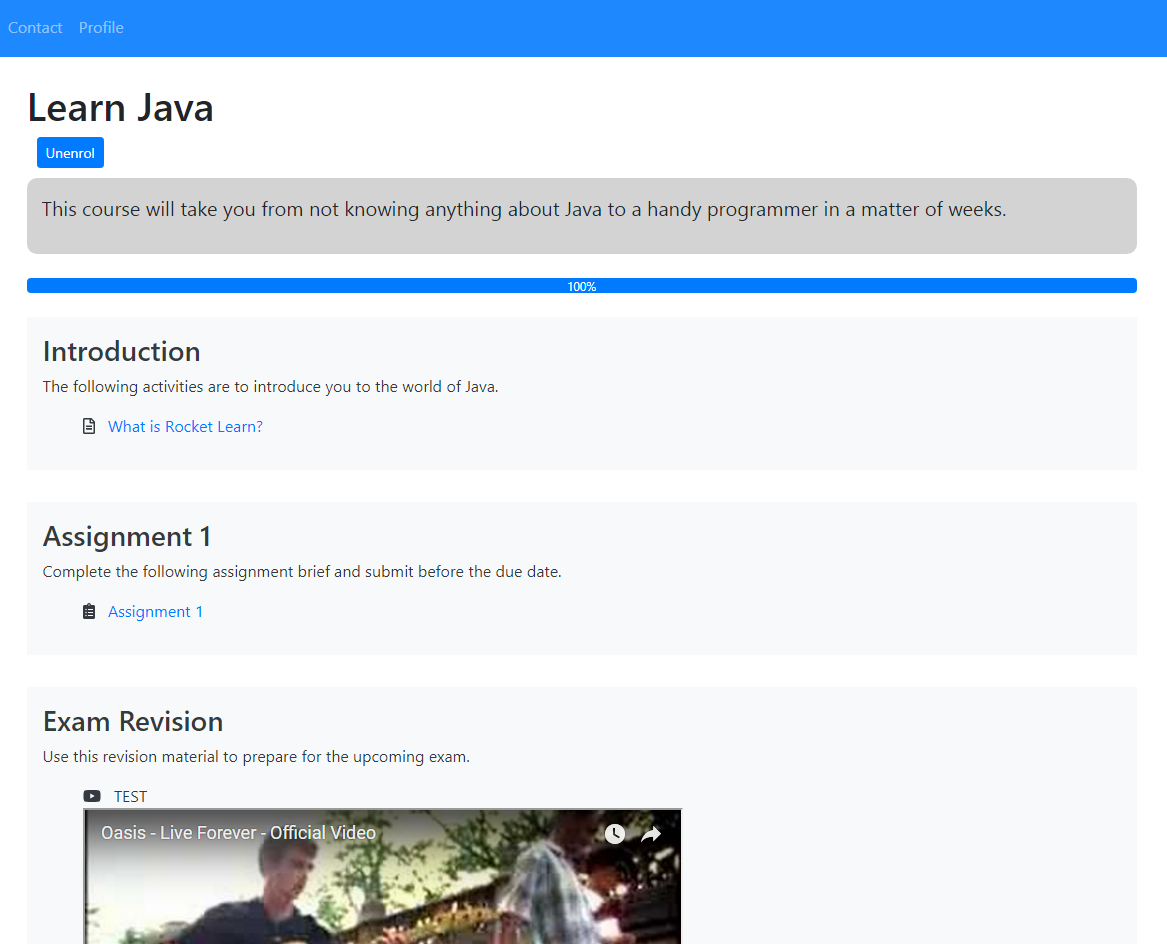


Figure 19 - Course page (Student view)

The course page shown in figure 19 is taken from a student user’s view, the teacher/administrator has a different view that provides them with the options to manage the course (shown in figure 20). They have access to the course settings, enrolments and completion criteria pages. This allows them to configure the course. Topics can be created via the ‘Add a topic’ link at the bottom of the course page and edited or deleted via the edit link in the bottom left of any topic. Additionally, a course element (activity or resource) can be added to a topic via the ‘Add an activity or resource to this topic’ link.

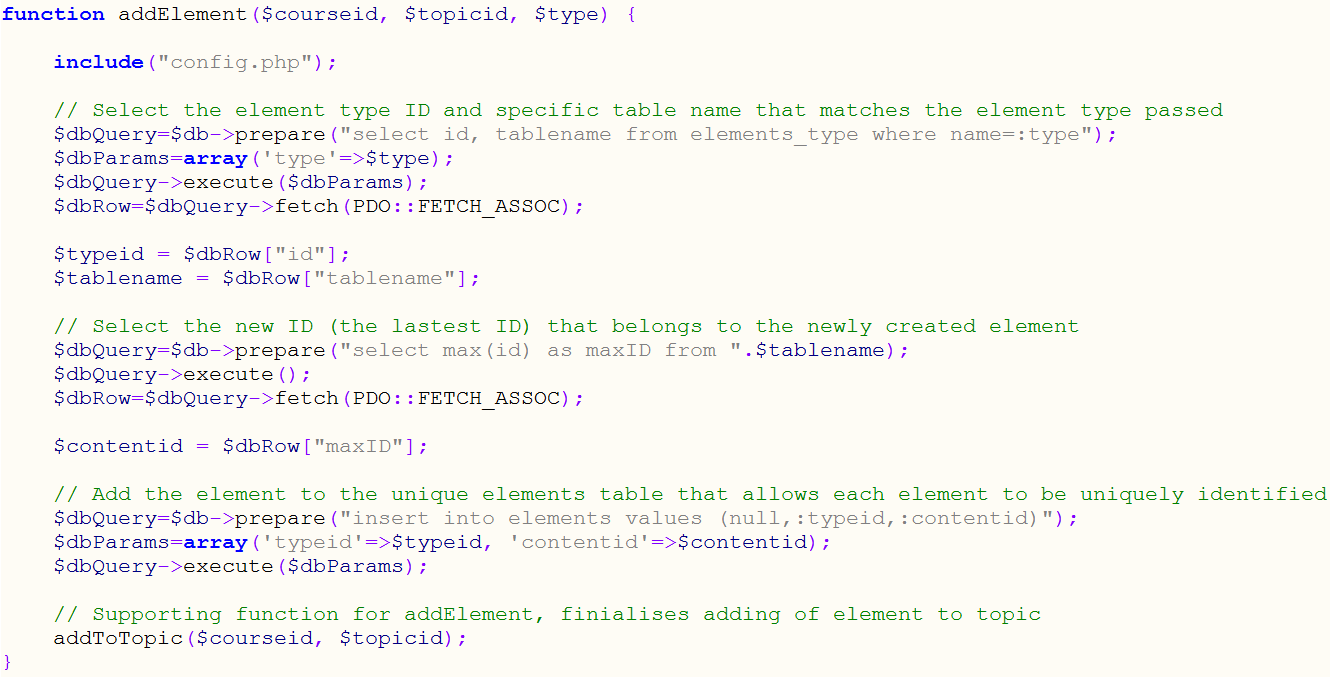


Figure 20 - Course page (Teacher/Admin view)

Course elements are split into two categories: activities and resources. Activities are completable aspects of a course; these will be displayed as options when selecting the course completion criteria. Resources are supporting elements that are not completable, but will likely aid the completion of activities by providing the student with additional information (via a web link, site page, video, or file).

Course elements can be added to a topic by teachers and administrators. Each of the elements has their own specific create forms, but are all added to the course by the *addElement()* and *addToTopic()* functions (shown in figure 21). The *addElement()* function is initially called following the create form submit, by this point, the element will have already been created (inserted into its specific table). Three parameters are passed into the *addElement($courseid, $topicid, $type)* function; they are the course ID and topic ID where the element is being added to, as well as the element type (such as assignment). The rest of the process is as follows:

* Select the specific table name and ID from the *elements\_type* table in the database that matches the element type parameter.
* Select the new ID (of the element that has just been created) from the specific element table.
* Add the new specific element ID and type ID (obtained from the *elements\_type* table in the first part of the process) to the *elements* table, giving the new element a unique ID.
* Call the *addToTopic($courseid, $topicid)* function and pass in the two parameters, to begin the “add element to course topic” part of the process.
* Select the new unique ID from the elements table and insert it into the *topics\_content* table along with the unique topic ID to link the element to the course topic that it was created on.



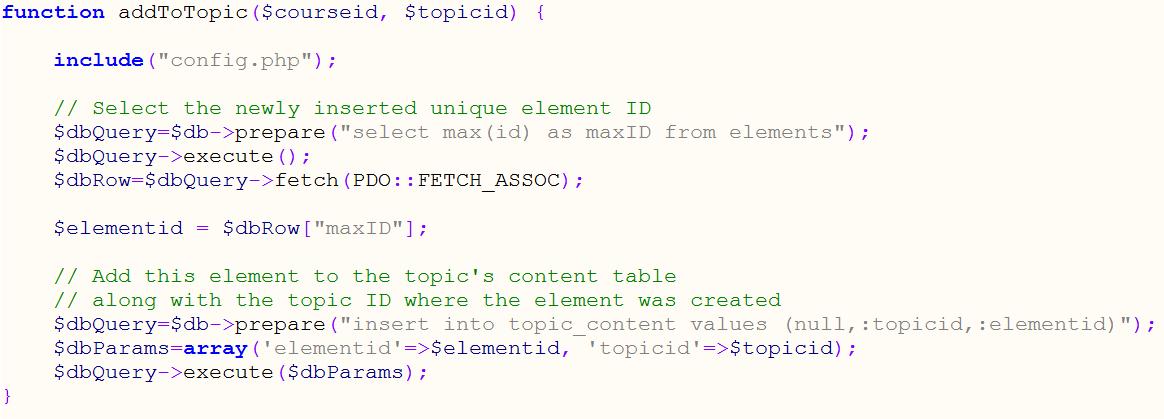


Figure 21 - addToElement and addToTopic functions

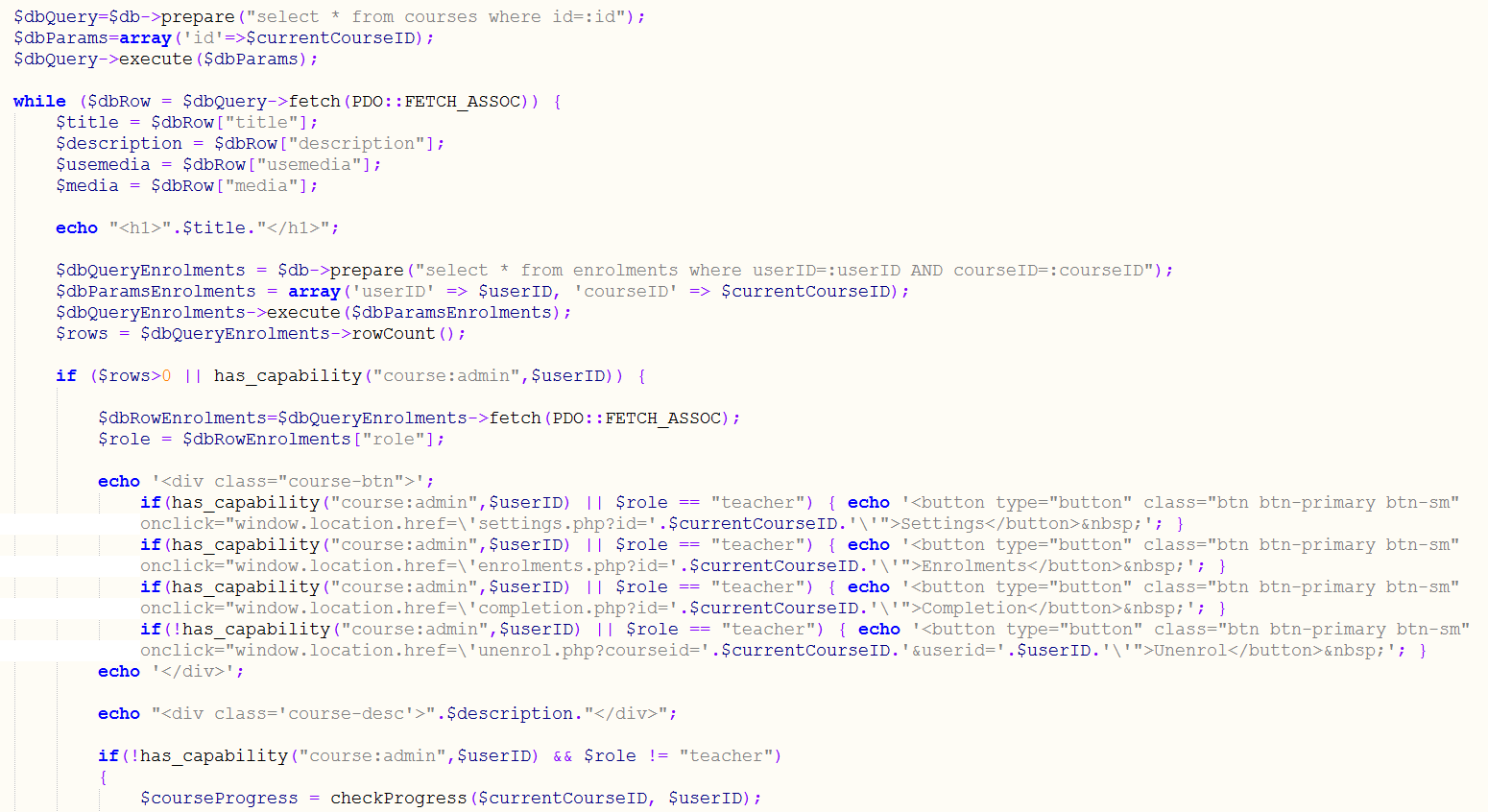
Figure 22 shows the PHP code that is used on the course page to display the course. Note that the configuration file (that includes the database connection) is included at the top of this page along with the code that checks if the user is logged in (both out of screenshot). The code included in the screenshot shows a query firstly to select the relevant course using an ID which has been passed to this page using the *GET* function and stored in variable *$id*, all of the course data has then been pulled in via the *$dbRow* variables. Another query is then sent to the database to check if the user is enrolled on the course, this is done via the enrolments table where the *userid* and *courseid* match the current course and user. If the query does not return any results, the enrolment form is shown in place of the course content. If the query does return a result (meaning the user is enrolled) or if the user has the *course:admin* permission then the course content is shown.

The course title and description are firstly pulled from the *courses* table in the database via a query. This is followed by the inclusion of the ‘course buttons’, which are the course settings (which only displays for users with the *course:admin* permission) and unenrol (which only displays for non-admin users) buttons.

The course progress bar is then displayed for students on the course. The user’s progress is calculated and returned by the *checkProgress($courseid, $userid)* function. The function pulls in the course completion criteria for the course and checks how many of these the user has completed, thus providing a percentage (this can be seen in figure 23).

The course topics and their elements are then pulled from the database. A query is firstly used to loop through each of the topics that have been added on the course, and retrieve the elements that are associated with the given topics. This is achieved by connecting four tables: topics\_content, elements, elements\_type and the specific element table (for example, the assignments table). The process is as follows:

* The unique element ID for each element is selected from the topics\_content table (this refers to the unique primary key in the elements table)
* The element’s type ID (referring to the unique primary key within the elements\_type table) and content ID (referring to the unique ID within its specific table) are selected from the elements table.
* The element type’s database table name (where its data is stored) and name (such as assignment, quiz) are then selected.
* The element’s specific database table is then queried to select the element’s ID and title, to be used in displaying and linking the element for the user to access.
* The element type’s name is then used to differentiate between how different elements show be displayed, and also to set the different icons for the elements (this is achieved via a switch statement).













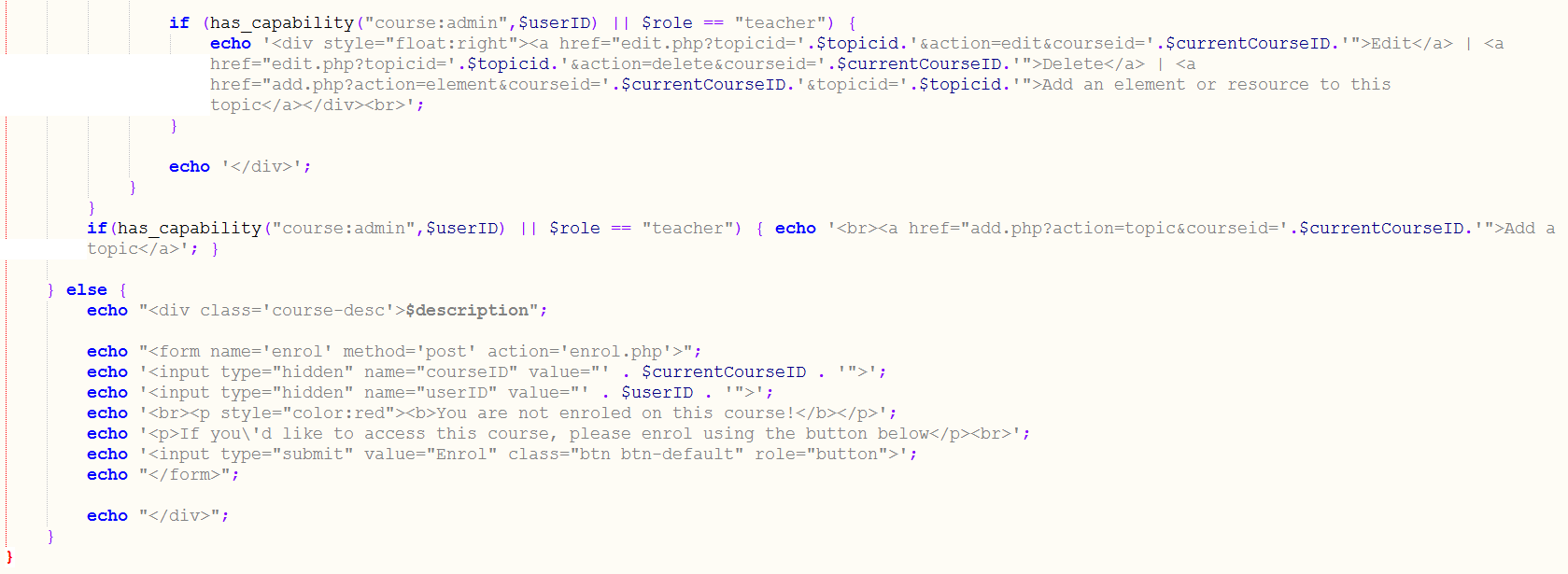


Figure 22 - view.php - course page PHP code





Figure 23 - checkProgress() function

When the user clicks the enrol button on a course they are not enrolled on, their user ID and course ID are passed to the enrolment script, which will result in them being enrolled on the course. This script adds an entry to the enrolments table and then redirects the user back to the course, where they will be enrolled on, so the course content will appear instead of the enrolment screen (shown in figure 24).

Testing has and will be completed as a separate task after each deliverable task is complete. This will ensure each function is tested before moving on to the next.

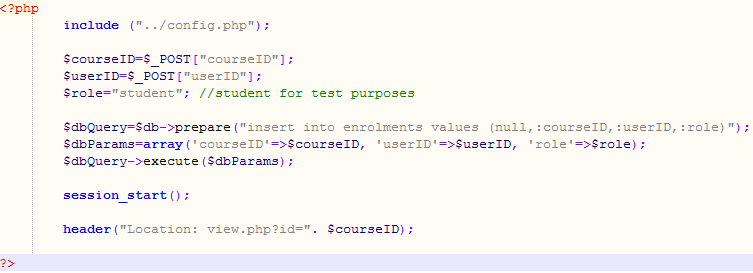


Figure 24 - enrol.php - enrolment script

When a user completes a course (by completing all of the activities that make up the course completion criteria) they can see this progress immediately from the progress bar, which as previously stated is calculated by the checkProgress() function on course page load. However, the overall course completion will not be recognised until the cron.php script (which can be seen in figure 25) runs and checks for any new course completions. The cron.php script is run via a cronjob which is scheduled to run every minute on the crontab; the entry on the crontab can be seen in figure 26. The process of the cron.php script is as follows:

* Select the unique ID from the courses table, and for each one:
* Loop through all users.
* If the user does not have a valid completion for this course ID, then call the checkProgress() function and check if the user has completed this course; if so:
* Insert the course completion record into the course\_completions table.
* Repeat, this for all course IDs.



Figure 25 - cron.php script



Figure 26 - crontab entry

4.1.4 Assignments

Assignments are activities that allow students to submit work (in the form of a file/attachment) and have teachers give a grade and feedback. Figure 27 shows the top of the assignment view.php page, where a query is run that checks the *assignment\_submissions* table for a submission for the currently logged in user. If a result is returned from the query (i.e. greater than zero rows) then the *$submitted* variable is set to true. Otherwise, it is set to false.

Figure 28 shows the main body of code for the student section of the assignment’s view.php. If the student has submitted an assignment a query is run that joins the *assignment\_grades* and *assignment\_submissions* tables to select details of the student’s submission, such as the time and file, and also retrieve the grade data, such as teacher feedback and grade. The *isgraded* column in the *assignment\_grades* table indicates whether the teacher has graded a student’s assignment, if so details of the submission and grade are shown, with the feedback being displayed in the feedback portion of the Bootstrap accordion; otherwise a message is shown to indicate that the assignment is still to be graded.

If the student has not submitted an assignment and the current time is less than the deadline set by the teacher, then the file upload form is shown. This form posts to the upload.php file, this is shown in Appendix 8, the code was originally sourced from a tutorial on tutorialspoint.com (tutorialspoint, 2018) with additional code added from a Stack Overflow answer to rename the file on upload (Stack Overflow, 2016), but has been changed for use on Rocket Learn; the file name change code was altered to change the filename a timestamp.

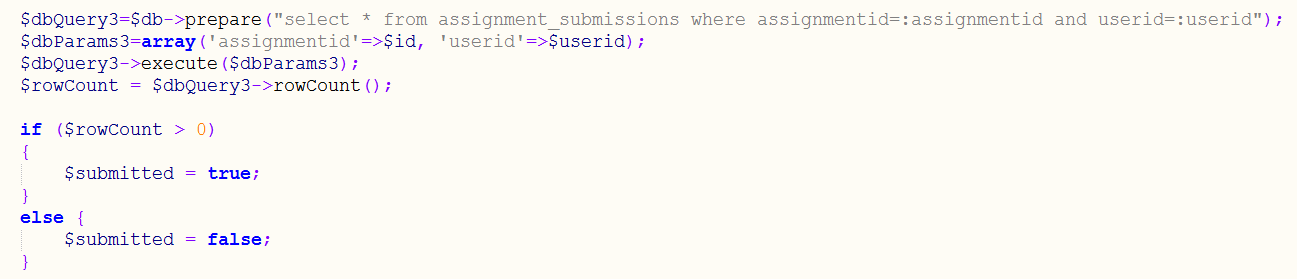


Figure 27 - Assignment view.php (Check if user has submitted)





Figure 28 - Assignment view.php (Student section of view.php)

4.1.5 Quizzes

The quiz activity enables the teacher to create a multiple choice quiz. The teacher first must create the quiz instance that entails the quiz name, summary, start date/time, close date/time, visiblility and passing grade. Once submitted the teacher will be directed to the question creation portion of the quiz creation. Here the teacher can add multiple choice questions with up to five answers; although the correct answer is always input first when creating questions, the answer’s order is randomised when shown to the students, this is achieved by appending ‘order by RAND()’ to the SQL query when retrieving the answers. The quiz display code can be seen in Appendix 15.

4.1.6 System Administration

Sticking to the true nature of a learning management system, system administrative functionalities feature heavy throughout the system. These make up the main CRUD (create, read, update and delete) elements throughout the system. Course elements for example are read by the user, and created, updated and deleted by system administrators; the same can be said for courses and even user accounts.

The *has\_capability()* function (as shown in Appendix 13) is used in an *if* statement to check if a user should be able to view a particular page, or elements on a page.

## 4.2 Testing

4.2.1 Unit Testing

Unit testing was carried out throughout the development of the system when new functionalities were developed, of these tests some failures occurred and were met with corrective actions, these have been listed below.

Scenario: User login

* **Expected outcome:** User would be logged in and redirected to their user profile following valid credentials being provided
* **Actual outcome:** User redirected back to login following valid credentials being provided
* **Corrective actions:** Session IDs were echoed out on both the login and profile pages, which revealed that the pages were on different sessions. This was corrected by placing the *session\_start()* declaration before any output on all files on the system.

Scenario: Adding assignments (or any element) to Courses

* **Expected outcome:** Following creation, teacher redirected to course where the element appears on topic where the teacher selected to add
* **Actual outcome:** Error for GET variables not being set within the URL appeared, rather than redirection to the course page, however the element was added to the correct topic.
* **Corrective actions:** Restructured assignment create page, specifically changing the location of where the *addElement()* function was being called. This helped find the issue, that the checks for the URL GET variables being set were happening before the redirection back to the course, therefore after the call to *addElement()* the GET variables were not being set and caused the error. The corrective action was to place the *addElement()* above the GET variable checks. This can be seen in Appendix 9, with the call to *addElement()* being enclosed in the first if, then the GET variable checks in the secondary else if.

Test cases (Unit Testing)

Table 10 - Unit test cases

|  |  |  |  |
| --- | --- | --- | --- |
| Scenario | Expected outcome | Actual outcome | Pass/Fail |
| User account creation | User account successfully registered and showing in backend when self-registering and manually created by an administrator. User should now be able to log in | User account inserted to database successfully when both self-registered and manual created by the administrator. User successfully logged in both times. | Pass |
| Student enrolments on course | User will be able to access the course as a student when self-enrolled and manually enrolled by the teacher. | User can access course when both manually enrolled by the teacher and when self-enrolling on open courses. | Pass |
| Teacher enrolment and ability to use administrative functions on enrolled course | When the teacher is manually enrolled on course, they can access and administrate (e.g. control settings, enrolment, completion criteria and add topics/elements) | Teacher is able to access the administrative functionalities of the course they were manually enrolled on. These administrative functionalities are not available on other courses or elements from other courses. | Pass |
| Complete quiz | Quiz is graded as completed with 100% of the questions correct. | Quiz was graded as completed and 100% of the questions correct | Pass |
| Complete course | Course becomes marked as complete on the user’s dashboard and also the course progress bar. | Course is marked as complete on the dashboard and 100% is shown in the course progress bar | Pass |
| Search for courses | The courses are filtered with only the “Learn Java” course shown following the search. | Search results showed the “Learn Java” course following the search | Pass |
| Create a course | “Course created” message is shown following submitting the create form and course is available from the catalogue. | Course was created successfully, with success message shown and available from the course catalogue. | Pass |
| Attempt to access restricted admin area as a standard user | User is redirected to error page, stating that they do not have permission to access the page | Redirected to error page, which stated that the currently logged in user did not have permission to access the page | Pass |

4.2.2 Acceptance testing

Acceptance testing was carried out to ensure that the final system meets the requirements, as set out in the design phase of the project.

Table 11 - Acceptance test cases

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Expected outcome | Actual outcome | Pass/Fail |
| Stability | Server uptime of 99.9% | 1&1 (the web host) guarantees 99.9% uptime | Pass |
| Performance | Page load time of at least 2.21s. | An uptime check, run by Pingdom shows a page load time of 498ms. Making the site faster than 97% of Pingdom tested sites. Shown in Appendix 10. | Pass |
| Scalability | The system scale can be increased and decreased based on the number of users and courses. | These can be created and deleted by those with the relevant system permissions. In scaling up extra server resources may be required to facilitate this, this could require a reconfiguration of self-hosted infrastructure. For cloud (SaaS) customers the resource scalability would happen automatically when the customer purchases a higher tariff. | Pass |
| Security | The system can be browsed by guests, but the main functionalities are only available to authenticated users (those who have logged in). User self-registration is only an option if the site administrator enables it, otherwise user accounts must be created manually by administrators. | User can access courses when enrolled, but not before. Users with invalid permissions receive errors on administrative pages. | Pass |

## 4.3 Evaluation

4.3.1 System Objectives

The goals set out by this system were to provide adequate resources and functionality for teachers to create learning material and facilitate student’s completion of these learning materials; as well as providing effective administrative capabilities to manage the system, all implemented with an emphasis on simplicity.

Courses are the containers for teacher’s learning material, and are used to facilitate the student’s learning. A teacher can setup a course while in ‘closed’ visibility and then make the course available (through the ‘open’ or ‘restricted’ visibilities) following the setup of the course and its elements. Courses are organised into topics, allowing the teacher to provide clear and tidy structure within the course material. Once the elements have been added to the course topics the teacher can configure the course completion criteria, setting out what activities a student must complete on the course to complete the course. Students can be manually enrolled on courses that are ‘restricted’ to created closed groups/classrooms, or can self-enrol onto ‘open’ courses (which any logged in user can access following enrolment). When completing a full course setup a teacher will access (at the most) five settings pages (including the element create pages, which are not part of the actual course), this is where simplicity shows, learning management systems (such as Moodle and Totara) can require configuration on over ten settings pages. This meets the objective of teachers being able to create learning material and facilitate student’s completion of the material.

Administration plays a large role in all learning management systems, this does not differ in Rocket Learn. System administrators within the system are given the capabilities to access all areas, enabling them to configure and manage the system at the highest level. Functionalities including the ability to disable user self-registration, change the site name and change the site theme allow the administrator to configure the application to their specific needs, allowing Rocket Learn to remain relevant in different sectors and use cases. Other main administrative functionalities are the user management tools, these allow the administrator to add, delete and edit users, giving them full control of who uses the system, and the details stored. The reporting section gives administrators a top level view of select system statistics, these may be used for external reports on the system’s effectiveness within an organisation or institution. This section of the system, although effective has significantly decreased the amount of pages you would expect to see on a leading LMS such as Moodle or Totara. The administration section on Rocket Learn is split over seven sections, compared with upwards of sixteen top level administration options on the latest versions of Totara and Moodle, with significantly more below options below these top tier options.

4.3.2 System Critique

Although the main objectives set out in the solution to the problem has been tackled and met, there are parts of the system that with more time, could have been improved or added.

One of the areas that improvements could have been is notifications, both system and email. This falls under the category of time constrained, the focus had to be placed on producing a functional system that is not complicated in functionality or user experience (UX). With more time notifications would be used to alert users of pending deadlines, and new elements becoming available on their courses (for example, a quiz becoming open).

Issues were encountered with sending email due to SPF records that made sending emails from the server difficult, and eventually priorities had to be set on achieving the system objectives. This meant that the reset password functionality could not be completed, which time had already been spent on designing the backend.

Courses currently only have the one level of sorting, which is by title via the search functionality. This could be expanded further by implementing course categories, allowing courses to be sorted into separate categories, and adding this to the search functionality. Teacher names could also be added to the course catalogue, with them being listed under each course; this could then provide another search filter.

# Chapter 5: Conclusions

The idea behind Rocket Learn was to alleviate some of the long lead times that organisations and institutions go through when implementing a learning management system. Through analysis it was found that there was considerable time and costs associated with LMS training, and that training was required in most cases due to system complexity. At the beginning of requirements analysis grouping was highlighted as a major functionality for Rocket Learn, however the reality of this was that grouping would add significant development time and complexity to a majority of aspects of the system (including users, courses, permissions and system visibility levels), going against the purpose of the system. Rocket Learn significantly decreases the amount of administrative options associated with standard system functionalities, such as setting up a course, adding users, changing site appearance and adding activities and resources to courses (such as quizzes).

Rocket Learn is developed primarily in PHP, with support from HTML, CSS and JavaScript. The application is built upon the Bootstrap framework; development initially begun on Bootstrap 3, which was converted to the latest version (4) back in December, half way through the project. This involved significant design changes due to the changes within the framework. Ultimately this decision was the right one to make as the benefits of the newer components have helped heighten the user experience within the system.

The implementation plan started with prioritising the setup of a course and administration functionalities (such as user management). Site pages were targeted next as this would have added to the system’s functionality in a relatively short development time. Course topics was one of the more difficult areas to implement in terms of the logic alone. This took several weeks to find the right database logic, in terms of the relationships and how it would be displayed on the course pages. After the careful planning of the backend, the frontend was connected within several days. With the course topics implemented the natural next step was to begin working on the course elements and the code to connect these to a given topic on a particular course. This was accomplished by reversing the topics display code on the course page, and condensed into two functions. Time constraints emerged throughout the development of the course elements and the administrative functionalities surrounding these; each course element can be created, read, updated and deleted to the administrator’s requirements.

The remaining functionalities were around system refinements and making the user experience as smooth and simple as possible. The small amount of configuration required to setup courses and their elements is a significant pro to this system and embodies the purpose of this project.

## 5.1 Key Developments

5.1.1 Quiz activity

The quiz activity was logically complex, which required significant planning before any implementation was done, in the end this functionality demonstrates not only complex use of code, but also effective UX, as it is simple feature to configure and use.

5.1.2 Adding course elements to topics

Adding course elements to topics was the most meticulously planned part of the system due to the logically intense nature of the implementation, as with many of the complex elements within Rocket Learn the development was done in reverse; meaning the backend structure was built and then a dummy data set was manually input, with the goal to get the dummy data displaying within the system. Once this displays then the frontend can start to be connected with the backend, in the form of inserts and selects.

## 5.2 Future Developments

The time constraints involved in this project has meant functionalities that were part of the initial analysis and that would go a long way to improving the overall system and user experience have been left out. These are listed below:

5.2.1 User groups

This would have enabled teachers and administrators to create groupings of users and enrol whole groups on to courses, which could be used to create classes for educational institutions. This would have required new development in most areas of the system, including course enrolments, course visibilities, and course element visibilities.

5.2.2 Notifications

This would’ve enabled users to stay notified of deadlines and announcements (via the site news functionality).

5.2.3 Messaging

Messaging would have enabled users to communicate with each other and added elements of collaboration to courses.

5.2.4 Increased contextual role assignments

Currently teacher enrolments can be considered a role assignment at a course context, however this has not been implemented in an expandable manner. Roles being assigned at various contexts would enabled the ability to have (for example) administrators on courses and groups rather than just at system level.

5.2.5 GDPR functionalities

The new GDPR (General Data Protection Regulation) law within the EU (EU, 2016) will require organisations and institutions handling EU citizen data to comply with the regulations otherwise strong penalties can be incurred. Complying with these regulations would not be easily achieved without system functionalities to aid, these are listed below:

* **Site policies** – Setup a site policy that user’s must agree to before gaining access to the system, and track the exact policies that each user has agreed to.
* **Data access and portability** – The ability for an administrator to export all of the data associated to a given user upon request. Ready for export to another system if required.
* **Data deletion** – The ability for a user to request a full or partial deletion of their data within the database.

5.2.6 Accessibility options

The ability and desire to learn stretches all people big and small, regardless of abilities and ailments. The ability to cater for the different needs of people would be an important future development of this system. Enabling those who are visually, orally or aurally impaired to use the system with ease.

## 5.3 Project Conclusion

The overall goal of Rocket Learn was to achieve simplicity where it does not currently exist, in order to reduce the amount of time and money thrown at training services within the learning technology sphere. This project displays significant groundwork towards a marketable product that solves this problem.

*“That's been one of my mantras - focus and simplicity. Simple can be harder than complex: You have to work hard to get your thinking clean to make it simple. But it's worth it in the end because once you get there, you can move mountains.”* – Steve Jobs

# References

Bradley, D. (2012) *The cost of e-learning and blackboard - a Freedom of Information request to Cardiff University - WhatDoTheyKnow*, *What Do They Know*. Available at: https://www.whatdotheyknow.com/request/the\_cost\_of\_e\_learning\_and\_black (Accessed: 2 May 2018).

Capterra (2015) *LMS Industry User Research*, *Capterra*. Available at: https://www.capterra.com/learning-management-system-software/user-research (Accessed: 19 April 2018).

Docebo (2016) *eLearning Market Trends and Forecast 2017-2021*, *Docebo*. Available at: https://www.docebo.com/resource/elearning-market-trends-and-forecast-2017-2021/?utm\_campaign=2016 11 eLearning Market Trends Report&utm\_medium=Download WP&utm\_source=ELI Blog Post (Accessed: 20 April 2018).

EU (2016) *GDPR EU Regulation - 2016/679*, *European Union*. Available at: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A32016R0679 (Accessed: 3 May 2018).

Keys, D. (2018) *Rocket Learn*, *GitHub*.

Learning House (2015) *Online College Students 2015*, *Learning House*. Available at: https://www.learninghouse.com/knowledge-center/research-reports/ocs2015-report/ (Accessed: 5 April 2018).

marketsandmarkets.com (2016) *Learning Management System Market by Application &amp; Delivery Mode - 2021 | MarketsandMarkets*, *marketsandmarkets*. Available at: http://www.marketsandmarkets.com/Market-Reports/learning-management-systems-market-1266.html.

Moodle (2017) *Moodle.org: Moodle Statistics*, *Moodle*. Available at: https://moodle.net/stats/ (Accessed: 22 November 2017).

MoodlePlugins (2018) *Moodle plugins directory*, *Moodle*. Available at: https://moodle.org/plugins/ (Accessed: 12 April 2018).

Neumark, V. (2018) *Web Typography (Fonts) Best Practices*, *rootid*. Available at: https://rootid.com/think/web-typography-fonts-best-practices.

Ortner, M. (2017) *Top LMS Software Solutions Infographic - Capterra Blog*, *Capterra*. Available at: https://blog.capterra.com/top-lms-software-solutions-infographic/ (Accessed: 2 May 2018).

Project Management Institute (2013) *A guide to the project management body of knowledge (PMBOK ® guide)*, *Project Management Institute*. doi: 10.1002/pmj.20125.

Stack Overflow (2016) *php - Rename and upload file - Stack Overflow*, *Stack Overflow*. Available at: https://stackoverflow.com/questions/40690967/rename-and-upload-file?utm\_medium=organic&utm\_source=google\_rich\_qa&utm\_campaign=google\_rich\_qa (Accessed: 2 May 2018).

Synergy Learning (2018) *Synergy Learning*, *Synergy Learning*. Available at: www.synergy-learning.com.

tutorialspoint (2018) *PHP File Uploading*, *tutorialspoint.com*. Available at: https://www.tutorialspoint.com/php/php\_file\_uploading.htm (Accessed: 2 May 2018).

WebhostChecker (2017) *Web Hosting Services Rankings &amp; Reviews - WebhostChecker*, *webhostchecker.com*. Available at: http://www.webhostchecker.com/#full\_page\_load/UnitedKingdom/UnitedKingdom/Linux/Mar2017 (Accessed: 23 November 2017).

Weiss, C. (2013) *LMS Setup Fees: For some vendors it’s a cash cow – By Craig Weiss*, *The Craig Weiss Group*. Available at: https://elearninfo247.com/2013/06/19/cashcow/?utm\_campaign=elearningindustry.com&utm\_source=%2Ftrue-cost-of-a-learning-management-system&utm\_medium=link (Accessed: 16 April 2018).

YouTrack (2018) *JetBrains YouTrack*, *YouTrack*. Available at: https://dylankeys.myjetbrains.com (Accessed: 24 November 2017).

# Appendices

## Appendix 1 - Key Totara features

|  |  |
| --- | --- |
| Feature | Description |
| Learning plans | Assign relevant learning targets to users. |
| Certifications | A set of courses of which the completion is only valid for a set amount of time. Must be recompleted after expiry to be marked as certified. |
| Dynamic audiences | A group that has a rule set for membership, once any user meets the rule set they will be dynamically added to the audience. |
| Report builder | Build custom system reports from a wide selection of system sources. |
| Job assignments | Add a user’s role/position within their organisation to their user account. |
| HR Import | Import updated user information from an organisation’s HR system. |

## Appendix 2 - Key Moodle features

|  |  |
| --- | --- |
| Feature | Description |
| Plugin management | Moodle is expandable by plugins and shown within an easy to manage interface. |
| Calendar | Allows for events and course deadlines to be tracked. |
| Simple text editor | Simple and effective text editor provided on any areas with editable text. |
| Track progress | Minimal reports are provided for admins and teachers. |
| Notifications | Notification will be sent to system users to make them aware of system events. |
| Bulk course and user creation | Users and course can be created in bulk. |

## Appendix 3 – User Interface Wireframes

|  |  |
| --- | --- |
|  | Contact |
|  | Courses directory |
|  | Contact |
|  | Profile |

## Appendix 4 – Admin UI Navigation Flow Diagram



## Appendix 5 – Google Forms Survey

Collation of answers gained from the survey are listed below, these have been analysed in the Analysis section of the report. Click the link below to view the survey:

<https://goo.gl/forms/ukYNpPWVM7k0Okfk1>

|  |  |
| --- | --- |
| Question | Responses |
| Are you a user or do you work with Learning Management Systems? | Work with - 55.6%  User – 11.1%  Both – 27.8%  Neither – 5.6% |
| Administrators should be able to organise users and courses into segregated groups. | Strongly agree – 66.7%  Agree – 33.3% |
| These segregated group areas can co-exist on a system with 'public' courses. (A public course being one that is open to all users on the system) | Strongly agree – 33.3%  Agree - 55.6%  Neither agree/disagree – 11.1% |
| Reporting will be integral for teachers and administrators. | Strongly agree – 77.8%  Agree – 22.2% |
| Social network interactions are an important system feature to implement. | Strongly agree - 11.1%  Agree – 55.6%  Neither agree/disagree – 22.2%  Disagree – 11.1% |
| Bulk user actions (such as uploading users to the system) is a key functionality for administrators. | Strongly agree – 72.2%  Agree – 22.2%  Neither agree/disagree – 5.6% |
| Which of these elearning systems do you currently use? | Moodle – 14 users  Totara – 12 users  Mahara – 9 users  Blackboard – 7 users  Totara Social – 1 user  None – 1 user |
| List some of the main features you like on these systems. | Easy to use  Audience management, cohort, learning plans, hierarchies, report builder  Admin can log in as a certain student for a course and custom email notifications  Extensibility, modular design, community  Informative  Quizzes, Glossaries, Database |
| List some of the main features you dislike on these systems. | Appraisals in Totara LMS  HR sync, appraisals, 360 feedback, Goals  Long navigation, logout function once logged in as another user  Roles administration is not well designed / managed.  Outdated User Interface  Complicate language. E.g. Audiences/cohorts.  SCORM!!! |
| Please feel free to add any additional feedback, possibly a suggested feature or overall improvement. | Linking Learning plans and appraisals much more closely together  Don't make it too complicated |

## Appendix 6 – Deliverables and their activities

|  |  |
| --- | --- |
| Deliverable Name | Deliverable Activities |
| **Gather User Requirements** |  |
|  | Create a survey consisting of open end and closed end questions |
|  | Send out the survey |
|  | Analyse the survey responses, identifying trends or anomalies |
| **Analysis and Design Report** |  |
|  | Discuss the project problem including background research carried out around the problem |
|  | Complete the 'W' boundary questions to identify the scope of the project |
|  | Carry out background research on what applications are available on the market and identify their strengths and weaknesses |
|  | Discuss the solution which has been proposed to solve the problem |
|  | Explain the advantages of using MySQL as a database |
|  | Draw rich pictures to demonstrate the problem and solution |
|  | Complete a health and safety from |
|  | Complete the ethical approval form |
|  | Discuss progress of current implementation |
|  | Consider the business case, project risks and the ethical consideration |
|  | Identify the user requirements and separate them into functional and non-functional |
|  | Create user stories |
|  | Create user journey flow diagram |
|  | Design the application user interface of all screens using Moqups |
|  | Design the architecture of the application |
|  | Discuss the architecture chosen within the report |
|  | Discuss how the implementation will be carried out |
|  | Provide a summary of the Analysis and Design report |
| **Setup database** |  |
|  | Create database design in the form of an ERD |
|  | Use design to create tables and relationships in PHPMyAdmin |
|  | Connect front-end to the database |
|  | Test database by using elements in the front-end that will delete, create and update records in the database |
| **Develop front-end template** |  |
|  | Setup Bootstrap 4 environment |
|  | Develop navigation bar |
|  | Develop footer |
|  | Refine responsiveness |
|  | Test layout and responsiveness |
| **Course functionality** |  |
|  | Add create course functionality |
|  | Add view course functionality |
|  | Add course enrolment/un-enrolment functionality |
|  | Test course functionality and behaviour |
|  | Add course enrolment methods |
| **Course elements** |  |
|  | Design the elements that will be available |
|  | Develop these elements, so that they can be added to courses |
|  | Test elements and course integration |
| **Demo** |  |
|  | Select application to record screen and audio for demo |
|  | Record application demo, with voice-over audio |
|  | Upload to YouTube |
| **Login system** |  |
|  | Front-end for login and register pages |
|  | Link up backend on login and register pages |
|  | Add validation on login and register forms |
|  | Test validation |
|  | Test login and register functionality |
| **Role management** |  |
|  | Add relevant database structure |
|  | Develop role management system within all pages |
|  | Test role functionality |
| **System settings** |  |
|  | Add settings to config table within database |
|  | Add frontend for settings pages |
|  | Link up backend on settings pages |
|  | Test system settings functionality |
| **Completion system** |  |
|  | Add course completion criteria |
|  | Add element completion settings to course elements |
|  | Front-end additions to display completion progress |
|  | Link up backend with the data for completions |
|  | Add certificates |
|  | Test completion system |
| **Reporting** |  |
|  | Add course reports |
|  | Add user reports |
|  | Add system reports |
|  | Test all reports |
| **User management** |  |
|  | User creation page |
|  | Bulk upload users functionality |
|  | Test user management functionality |
| **Groups** |  |
|  | Add groups functionality |
|  | Segregate courses/pages based on group |
|  | Test groups functionality |
| **Contact functionality** |  |
|  | Send message to set roles on contact page |
|  | Develop front-end |
|  | Link to back-end |
|  | Set up map using Google Maps API |
|  | Test contact functionality |
| **Learning record** |  |
|  | Build page that pulls in a user's course progress and completion from database |
|  | Allow for the data to be downloaded in different formats (PDF, CSV) |
|  | Test learning record functionality |
| **Dissertation** |  |
|  | Final report |
|  | VIVA demonstration |

## Appendix 7 - Glossary of terms

Terms that are associated with Rocket Learn components and learning management systems in general.

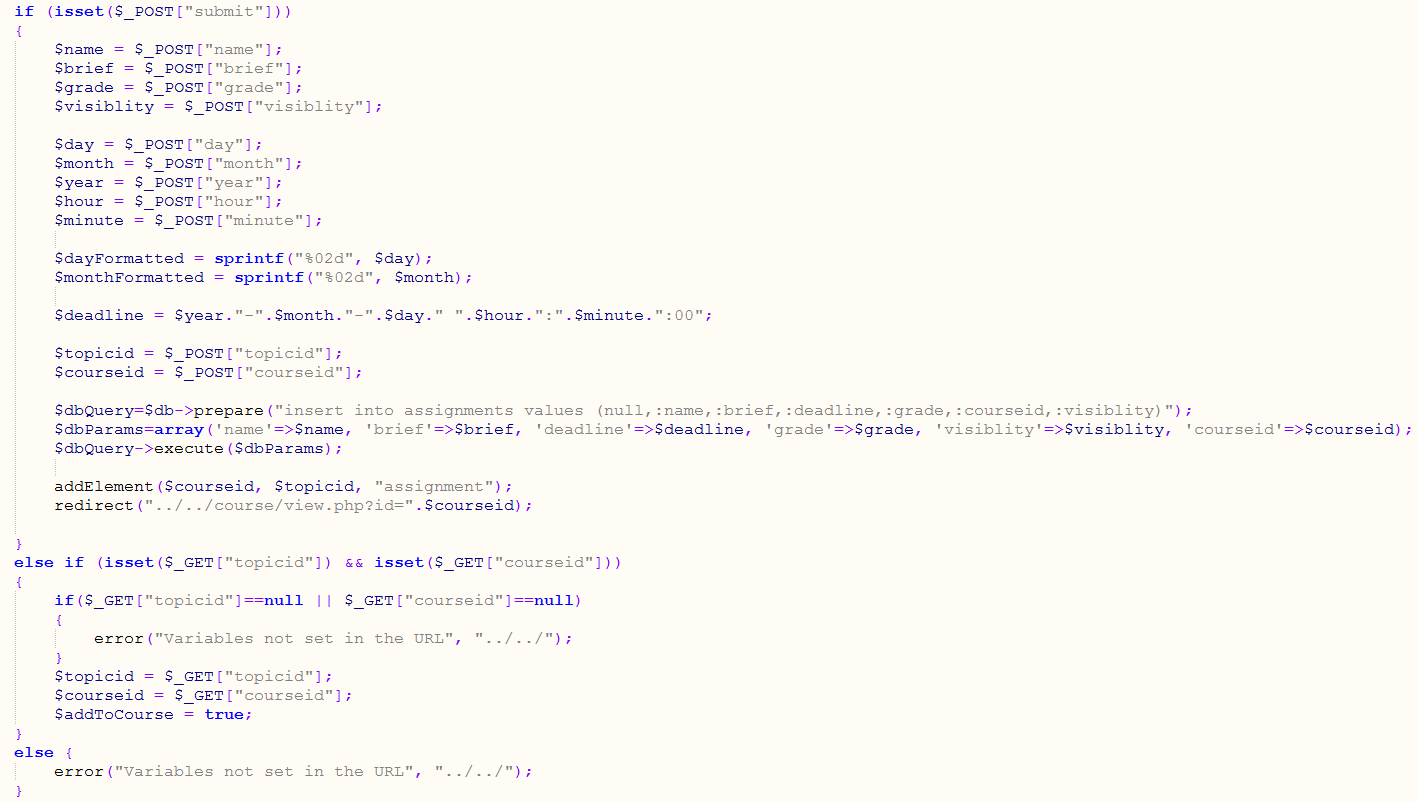
|  |  |
| --- | --- |
| Term | Meaning |
| Course element | An activity or resource that has been added to a course by the teacher or administrator for the students to view or complete (for example, an assignment is a course element). |
| Activity | A course element that can completed by a student, and added to a course’s completion criteria. |
| Resource | A course element that can be viewed by students, mainly used to support activities. |

## 

## Appendix 8 – Assignments

|  |  |
| --- | --- |
| Description | Screenshot |
| Assignment view.php |  |
| Assignment upload.php |  |

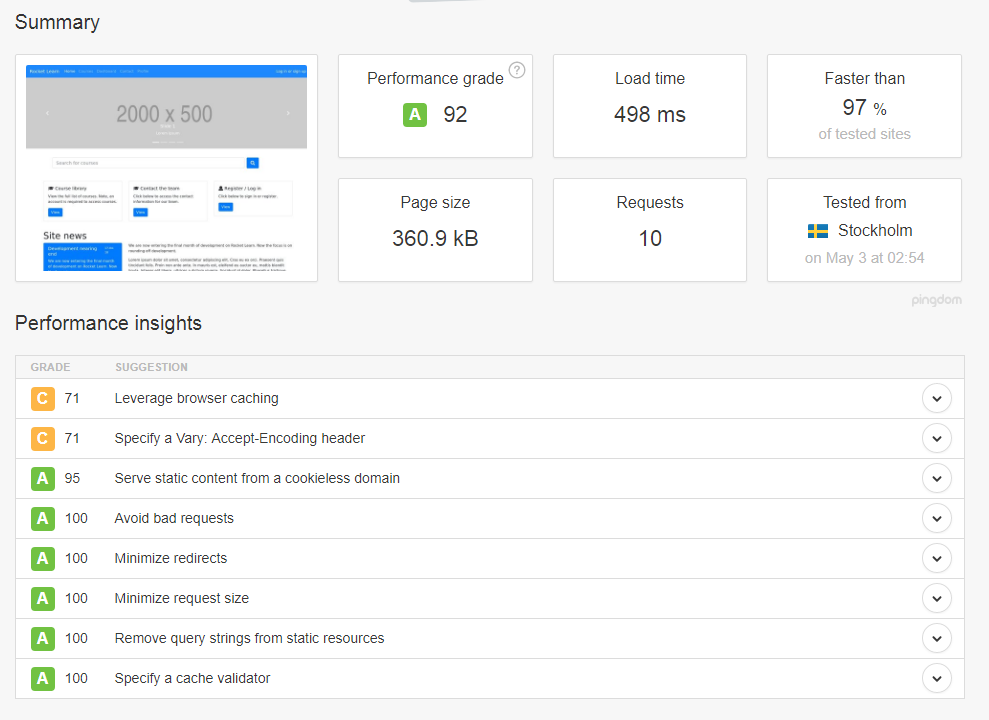
## Appendix 9 – Testing Fail, Corrective Action



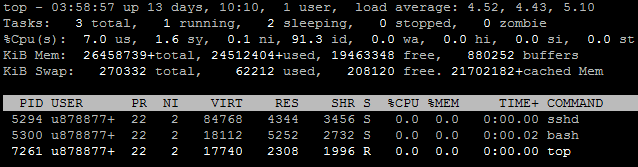
## Appendix 10 – Administrative settings

|  |  |
| --- | --- |
| Description | Screenshot |
| Settings home |  |
| Reporting, powered by Chart.js |  |

## Appendix 11 – Pingdom Uptime Check

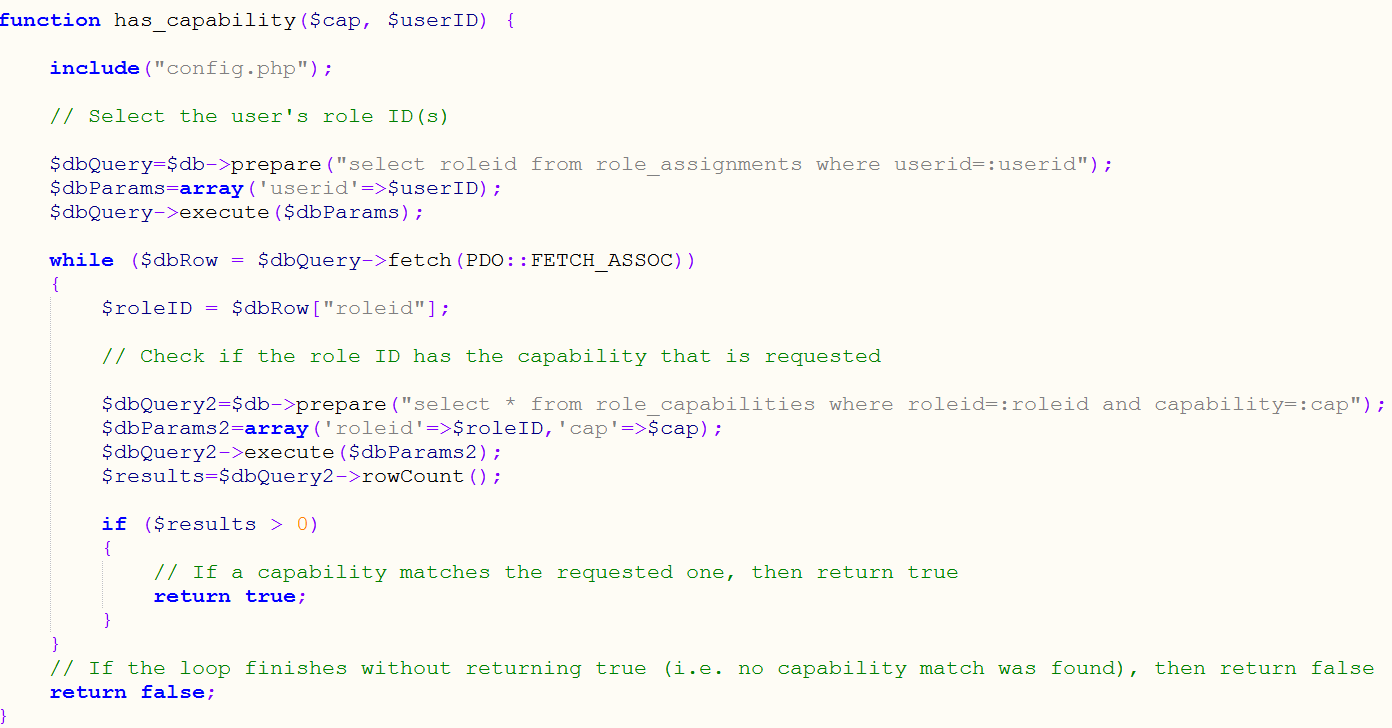


## Appendix 12 – Server Uptime / Load Average



## Appendix 13 – Rocket Learn functions (lib.php)

has\_capability()



Appendix 14 – Risk Register

