Web-Based Rostering System for Volunteer Worked Organisations

Final Report for CHM9360 MSc Project

Author: Oliver Roy Thomas Earl (ole4@aber.ac.uk)

Supervisor: Mr. Richard Shipman (rcs@aber.ac.uk)

26th September 2019

Version: 1.4 (Release)

This thesis was submitted in partial fulfilment of a MSc degree in Computer Science (Software Engineering) (G493Z)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, United Kingdom

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name: Mr. Oliver R.T. Earl

Date: 21/07/2019

Consent to share this work

By including my name below, I hereby agree to this thesis being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name: Mr. Oliver R.T. Earl

Date: 21/07/2019

Acknowledgements

I dedicate this work to my late aunt, Deborah Jones, who tragically passed away in June 2019 during her courageous and tenacious fight against cancer. On top of being a wonderful aunt and akin to a second mother, you were inexplicably supportive and reassuring in every moment of my academic career. I miss you greatly.

I am grateful for the endless support and empathy provided by my colleagues in the Department of Computer Science at Aberystwyth University throughout my most challenging year in academia thus far. In particular to my supervisor Richard Shipman who enthusiastically accepted me onto the project without hesitation and ensuring that I remained on-target and focused despite some initial setbacks, and to my personal tutor Edel Sherratt, who was always available to offer much-needed motivation and a listening ear at a moment's notice.

Furthermore, I wish to extend a heartfelt thanks to my family and friends throughout the world for their support, donations, and words of encouragement. I could not have done it without you.

Finally, I would like to express my further gratitude to those I have had the privilege to work alongside at Aberystwyth University, Aberystwyth University Students' Union, Aberystwyth Nightline, the Nightline Association, and the National Union of Students. You enabled me to have a positive and profound impact in their lives and wellbeing of my fellow students throughout the year, and for that I am extremely thankful.

It has been an unrivalled privilege to have spent the majority of the last five years studying in idyllic West Wales, enrolled at a university brimming with history, expertise, and compassion. Aberystwyth will undoubtedly be a place I shall always call home.

Thank you very much.

Diolch yn fawr iawn.

Abstract

Corris Railway, a historic narrow-gauge railway in Machynlleth, Wales, is seeking a web-based application to replace its manual spreadsheet rostering system. A great deal of requirements analysis is required to deal with a wide array of volunteer role types and levels of competency, locations, vehicles, rules, and complex domain knowledge that must be accounted for when producing a timetable and roster of operations. In addition, the software needs to be fully responsive and work equally well on mobile devices, whilst still being fully functional and easy to use by its volunteers.

Railrota is the end result – a full-stack monolithic application developed using the artisanal Laravel framework built for the PHP programming language.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Background

Corris Railway is a historic narrow-gauge railway located in Mid Wales, near the Welsh town of Machynlleth, originally having opened back in April 1859 and was then known as the Corris Machynlleth and River Dovey Tramroad. [1] After a hiatus of around seventy-two years, the railway resumed regular passenger services in June 2002, and continues to operate to this day. [2]

The need for a rota system that can deal with the demands of rostering the various volunteer staff for this railway is the defining purpose of the project, and there is a need to be able to effectively work with the various roles, constraints, and requirements that pertain to the day-to-day operation of the railway.

1.2 Project Aims and Objectives

1.2.1 Functional Requirements

The aim of this project is to construct a rostering system, that is able to deal with the various demands of rostering volunteer staff for the Corris Railway.

A range of staff must be supported by the system, including locomotive crew, controllers, guards, and museum staff. Additionally, staff are able to supervise less experienced trainees, so this must also be accounted for. Staff must also be able to volunteer on days or weekends without explicitly specifying what - so the system must also be able to perform some optimisation.

Each day has different set requirements and therefore requires different staff. A standard preset exists, but special events will have their own needs. Some days might have differing needs at different parts of the day.

The project objectives that must be fulfilled for successful completion can be broken down atomically as the following functional requirements:

- All users / volunteers
  - All users must be able to navigate the roster, and view current, upcoming, and/or historic shifts
  - Users must be able to fill in and edit their personal information
  - Users must be able to assign themselves to shifts of which they are able to do
  - Users must **not** be able to assign themselves to shifts that they are unable to do
  - Users must **not** be able to assign themselves to simultaneous roles on the same shifts
  - Users must be able to indicate periods of availability
  - Users must be able to indicate their willingness or desire to do an upcoming shift (i.e. an unspecified upcoming weekend)
  - Users must be able to operate the web application easily from a mobile device such as a smartphone
- Administrative users
  - Administrators must be able to modify user information
  - Administrators must be able to assign roles and privileges that reflect a volunteer's ability and status
  - Administrators must be able to add, edit, and delete shifts
  - Administrators must be able to mark a shift as closed
  - Administrators must be able to define a shift's staffing requirements
  - Administrators must be able to email users regarding upcoming shifts
- The application must run on a predetermined operating environment

1.2.2 Desirable Functionality

1.2.2.1 Automated Rostering

The ability to automatically assign volunteers to shifts is considered beyond the scope of this project. Despite this, such functionality remains desirable due to the enormous time it can save administrative staff in ensuring that shifts remain open and filled. This is a complex computational problem however, and is explored in substantial detail in the literature review of this document.

1.2.2.2 Localisation Support

Localisation, often referred to as i18n, or internationalisation, entails allowing the software solution to support multiple languages, or locales. The primary motivation for this is derived from Corris Railway being situated in an area of rural Wales, where there are a substantial quantity of Welsh speakers, as many as 67% having at least some working knowledge according to the 2011 Census. [3]

While the The Welsh Language Act of 1993 only applies to public sector firms within Wales - ensuring that Welsh and English languages are treated as equals, there is mounting pressure for this legislation to extend to the private and third sectors respectively. Therefore, while it is not obligatory or necessarily required to be present within the first release of software, it is something to be kept in mind during development and potential further releases. [4] [5]

1.2.3 Operating Environment

The environment in which the software must operate is provided in detail in Appendix 5. Any proposed solution must be executable within these constraints.

1.3 Motivation

There are several important motivational factors that ultimately determined the selection of this project topic.

The most important motivating factor for the selection of this project is a genuine, unabated interest in the project specification, the work that would be involved, and the area of computing in which it inhabits. As described by O'Keefe et al, performance in completing tasks and obtaining the most optimal results are characterised when interest is high. [6] This was also further elaborated in a later online publication that interest allows "people to persist when persisting would otherwise cause them to burn out." [7]

With a substantial history of working with web applications and Internet technologies throughout academia, within industry, and in personal open source projects, a task to design, implement, test, and deploy a substantial full-stack program with a well-defined set of challenging requirements and technical and organisational hurdles that must be overcome is an exciting prospect, which contributes substantially to its potential success. In addition, as a long-time volunteer with experience donating both time and technical know-how to a range of charities and non-profit initiatives, the experience and domain expertise acquired from operating within those environments, as well as having extensive understanding of the difficulties and unique challenges presented by volunteer management and rostering respectively will prove highly transferable and useful throughout

the overall system design phase.

Furthermore, this project is intended for real-world implementation and application for the aforementioned steam railway and by its various volunteer workforce; placing it outside the safe confines of a classroom assignment and placing even greater importance on code quality and application ease of use.

Chapter 2

Literature Review

2.1 Introduction

A roster, sometimes referred to as a rota or schedule, is a list with dates (or shifts) with staff, such as employees or volunteers assigned to them, indicating what work will be done and when it will be done. It might also indicate when staff are unavailable, such as them being on leave. [8]

Automatically assigning staff to a roster is a well-known problem in computing. As described by Ernst et al, it is highly challenging to 'determine optimal solutions that minimise costs, meet employee preferences, distribute shifts equitably among employees and satisfy all the workplace constraints.' [9]. As described, 'hard' constraints must be conformed to in order for the roster to be deemed valid. Such constraints might include: [10]

- Availability of staff - staff cannot be away or on leave
- Staff cannot carry out more than a certain number of consecutive shifts
- Staff cannot exceed maximum numbers of shifts, or fall under a given minimum
- Staff must be of the right sub-type, or meet criteria such as skill level or qualification

The efficiency of the outcome is determined not only by its validity, but also by how well it meets 'soft' constraints, such as regularity of shifts.

In general, software for tackling this problem exist to reduce the intrinsic difficulties and time overheads presented by manually rostering workforce, which was a role traditionally carried out continuously by secretarial staff, who would ensure that constraints where not violated as aforementioned, and would optimise the rota as much as possible for the sake of soft constraints to be best met. Today, most of this work is done by software, although occasionally conflicts must be resolved by a human. [11]

Based on the project's aims and objectives, it is already established that an automated or semi-automated solution to automatically roster staff is unnecessary and beyond the scope of the application, as there are dedicated personnel responsible for ensuring that necessary gaps are filled, and volunteers add themselves to roles which they are qualified to fulfill at their own discretion. Nevertheless, having an understanding for this computational problem remains useful for the design and implementation of the project and is subsequently worth researching.

2.2 The Nurse Scheduling Problem (NSP)

The aforementioned problem of automatically finding the most optimal way to assign workers to shifts, with both hard constraints that must be met in order to be valid, and soft constraints that determine the optimality of the solution, is commonly referred to as the Nurse Scheduling Problem (henceforth NSP) and is considered by most sources to be of NP-hard (non-deterministic polynomial time) complexity. [12]

The reason behind nursing being the focus for this particular scheduling problem is that it is particularly challenging to roster for shifts that exist around the clock (healthcare institutions such as care-homes and hospitals work during the day and night), that require staff with varying required skill and competency levels for different types of work, whilst also ensuring that other soft constraints such as regularity and routine of schedules to benefit nurse wellbeing are met. [13]

This has a strong parallel with the current challenges presented by the steam railway, where although there exists only one shift per day, vastly simplifying the problem, there is also a need for various volunteers with different skill levels and related constraints to be met.

The two most well-researched and documented solutions to this problem involve the usage of genetic algorithms, and simulated annealing (including derivative methods), although other implementations have been explored in detail.

2.2.1 Genetic Algorithms (GA)

Genetic algorithms (GA) are described by McCall as a heuristic search and optimisation technique originating in the 1960s that is modelled on Darwinian natural evolution, whereby a population of chromosomes - each a solution to a problem and a measurement of its effectiveness, called fitness. Those with the highest fitness are combined to produce new child chromosomes, and this process continues until criteria pertinent to the solution are met. [14]

Furthermore, random mutations have a possibility of occurring on each new generation, further modelling the real-world phenomenon of evolution and to diversify the population of chromosomes, as well as to increase the possibility of discovering the most optimal solution. [15]

2.2.2 Simulated Annealing

Baird describes simulated annealing as an optimisation algorithm that is used to find global optima in the presence of local optima. He explains that moves are randomly selected and always accepted if the current solution is improved as a result of that move. Should this not be the case, the move has a chance of being made anyway, with the probability determined based on the 'badness' of the move, which in turn is determined by the quantity in which the solution is made worse. [16]

As written by Press et al, the name of this algorithm is derived from an analogy with its namesake in thermodynamics, in the way that 'liquids freeze and crystallize, or metals cool and anneal' and how when high temperature liquids are cooled slowly, 'atoms are often able to line themselves up and form a pure crystal that is completely ordered'. [17]

Ko et al explains that simulated annealing is an acceptable means of providing a good solution rather than the most optimal one, and that more accurate results can be obtained within shorter periods of time by using Cost Matrix-based simulated annealing (CSMA), a derivative method using a cost matrix for the transition rule. [18]

2.3 Prominent Rostering Solutions

As scheduling software began to come into prominence in the 1980s as manual work tracking systems and spreadsheet-based solutions, digital software solutions specifically intended for the rostering of volunteer staff started to arise as a more efficient replacement. While countless solutions exist, and software specifically designed for paid members of staff could also be used for rotaing volunteers, three well-known solutions have been explored. The first of which, is a bespoke solution specifically designed for historic railways. The following two are generic solutions that are popular solutions for volunteer-based organisations.

2.3.1 Heritage Operation Processing (HOPS)

Heritage Operation Processing (HOPS) is a timetabling, rostering, and management solution designed specifically for railway operators by railway operators, designed to speed up and make more accurate time-consuming processes, including timetabling and rostering, and it began development in Gloucestershire Warwickshire Railway in August 2009. [19]

The wealth of experience and domain knowledge displayed by the developers of HOPS in their product is clear, as not only does the product go above and beyond the requirements for this project, but it also offered a wealth of functionality, included fully automated rostering functions, but also discussion forums allowing for communication of users, invoicing and financial systems, and even the ability to order, track, and pay for commonly required equipment from HOPS themselves. This

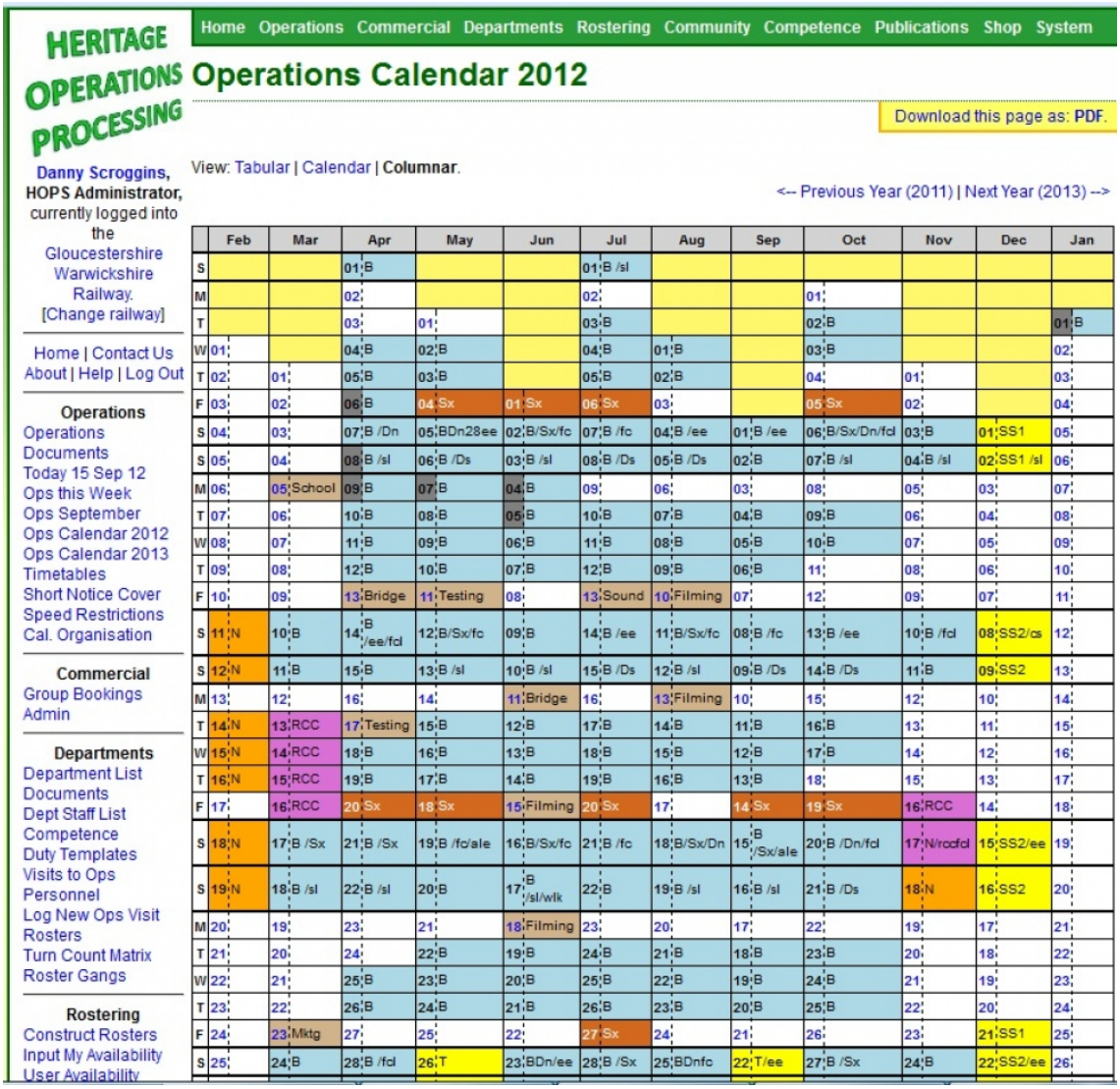


Figure 2.1: The Operations UI as displayed in HOPS.

[20]

functionality appears to be fully customisable, and clients can opt out of functionality that they do not want. The vast majority of HOPS' functionality can be used free-of-charge. [19] [21]

While HOPS does fulfill the customers requirements almost in full, the interface is visibly complex and difficult to use. Oral conversations with the client revealed that they are well aware of HOPS, but the difficulty in using the interface is a put off for the majority of volunteers, and as a result they resort to using manual spreadsheet solutions as shown in Appendix C. In addition, HOPS is not responsive, and does not resize itself properly in order to operate in smaller viewports, such as that of a smartphone or tablet. This would require some redesign of the web application front-end and the inclusion of media queries in order to rectify. [22]

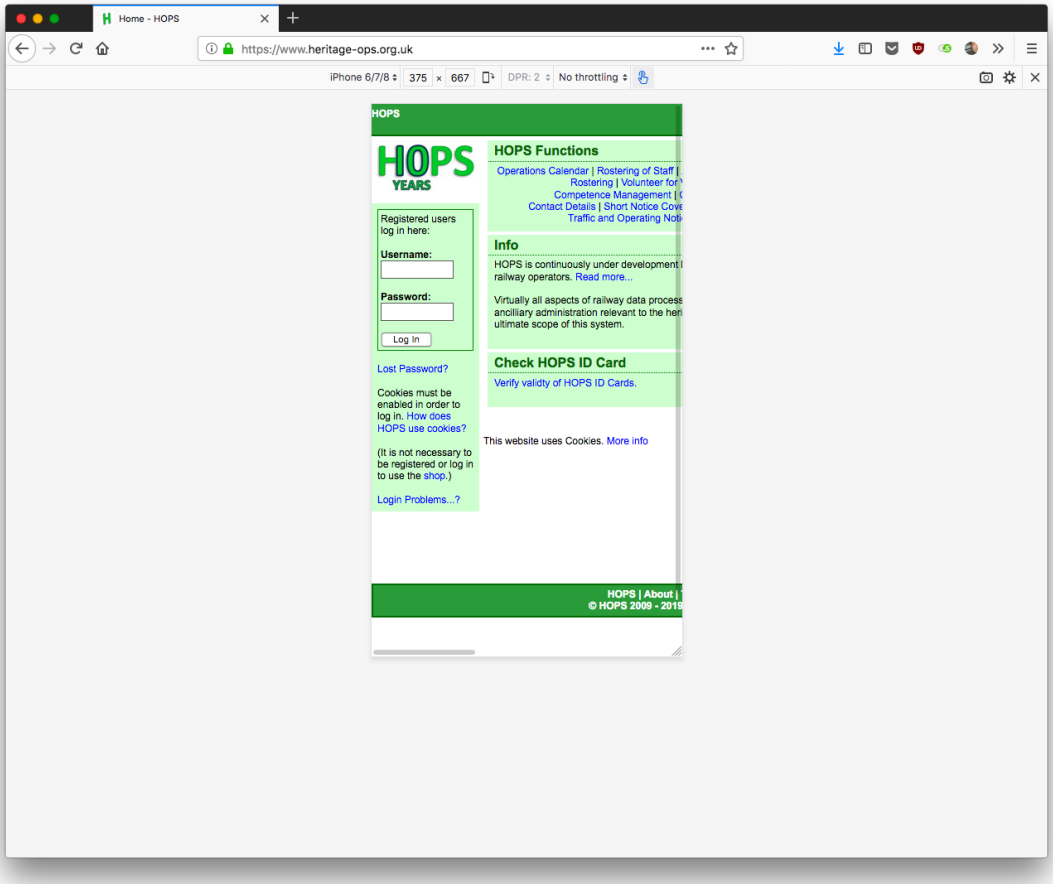


Figure 2.2: HOPS interface is not responsive, as shown in Mozilla Firefox Responsive Design Mode.

2.3.2 Three Rings

Three Rings is an online volunteer management system that is entirely run by volunteers, and is at this time of writing used by prominent charities such as The Samaritans, Macmillan Cancer Support, and Citizens Advice. [23] It was founded by Dan Q whilst volunteering for Aberystwyth Nightline in 2002, a student-run listening service that catered for the needs of students at Aberystwyth University. [24] [25] In addition, Three Rings has a flexible pricing system, whereby smaller charities and helplines are charged very little, as low as £40 a year, expanding to £600 for nationwide charities - the fee expected is derived from an organisation's annual turnover and volunteer count. [26]

Like HOPS, Three Rings is centred around rota functionality that allows for different kinds of shifts to be defined, and users can view and modify their own information. Resultingly, as the application is also designed and subsequently advertised as a mobile-friendly application, it can

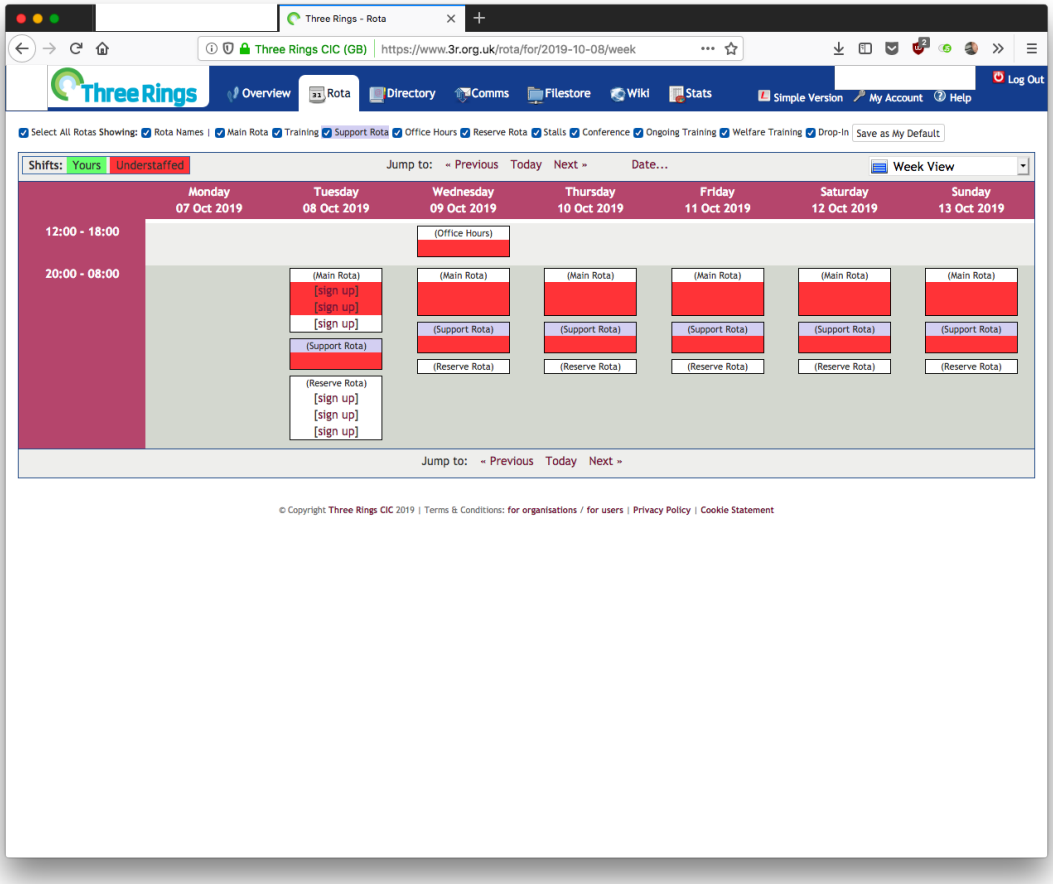


Figure 2.3: Three Rings interface as seen on a desktop web browser, with confidential data redacted.

be easily utilised on a handheld device, and meets all the customer's requirements. Furthermore, the software includes complex stats reporting and an area for uploading and sharing files, as well as support for algorithmically populating vacant rosters with eligible volunteers. [27] [28]

Finally, Three Rings was recommended, alongside more rudimentary freeware solutions such as Google Calendar, by Voluntary Action Sheffield in a review of various volunteer shift and rota solutions, described as a 'low cost option with a large range of features.' [29]

2.3.3 ABC Roster

ABC Roster is a proprietary freeware rostering application that is now used by a wide range of businesses and charitable organisations that prides itself on being easy to use, 'with a convenient and intuitive way of creating schedules quickly.' It appears to include all required functionality, such as availability management, the ability to email staff, as well as desirable bonus functionality

such as the automated production of schedules and exportability. [30]

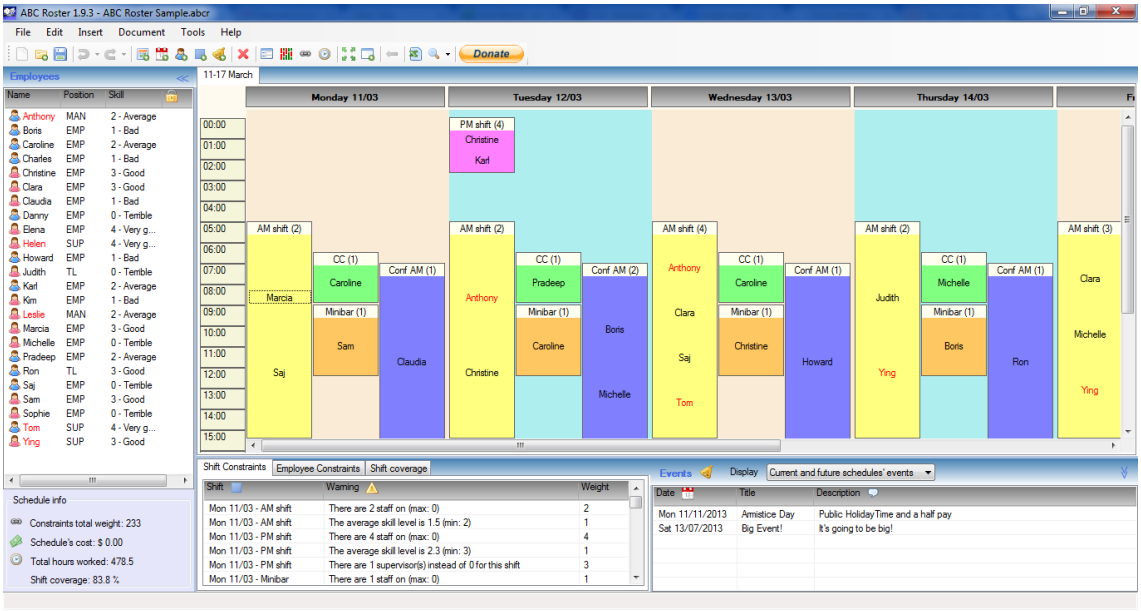


Figure 2.4: ABC Roster interface, shown running on Microsoft Windows. [30]

While a strong contender, with a wealth of functionality and sporting an interface that is straightforward to use, this is a desktop application for Windows, and is both not available for other platforms, such as macOS or GNU/Linux, and is not usable on mobile devices, as it is neither a mobile or web-based application. As the application needs to be able to run on the system requirements outlined on Appendix E, this is clearly an issue, beyond violating the requirements of a web-based application that can also be accessed on mobile.

In spite of this, there are things that can be taken away from ABC Roster and noted in the construction of a new solution. It's capability to define different roles with differing competencies is noteworthy, as well as the capacity to export rotas into printable form. [30] The software's documentation also provides a detailed insight into the workings of the algorithm responsible for automatically filling in rosters with staff - identifying constraints, weights, and penalties, with an algorithm in likeness to simulated annealing as explored previous:

$$(3x - 2y) * 4w$$

Let x as the required number of employees; y as the actual number of employees, and w as the constraint weight. The penalty for satisfied constraints is zero. [31]

2.4 Research Conclusion

While as previously discussed, resolving the NST and implementing an automated rostering solution is beyond the scope of this project, it provides an important insight into the maturity of the computational problem, and importantly the solutions that exist to address said problem, including the software packages in which they are contained. This research is nevertheless a valuable pursuit, as with all agile software projects, there is nothing prohibiting this functionality to become desirable in future revisions, or even as a desired inclusion into an early deliverable. As stated by Ambler, 'project stakeholders have the right to define new requirements, change their minds about existing requirements, and even re-prioritise requirements as they see fit.' [32]

The investigation of existing software solutions has provided a wealth of insight into shared functionality, what solutions do well, and some of their shortcomings that renders them inappropriate. It can be safely concluded from the research that a solution like Three Rings is very ideal for the task at hand, and subsequently this software project should look to emulate its offerings in some form, whilst incorporating domain specialism demonstrated by a product such as HOPS. In a nutshell, this research has provided valuable insight into what the ideal software solution looks like: responsive and easy to use, with a UI that is similar to the existing calendar / spreadsheet system, and specifically catered for the Corris Railway.

Should this research be continued or expanded upon, it would be ideal to explore even more examples of rostering software, including those used by major corporations and organisations that are not explicitly within the third sector. Furthermore, additional research into implementations of NST solutions is a good direction, so that should this be desired in a future revision of the software, it can be readily and promptly installed.

Chapter 3

Methodology

3.1 Agile Methodology

Agile methodologies, a concept solidified in 2001 with the creation of the Agile Manifesto, is the notion of prioritising the delivery of working software and a focus on individuals, including the customer, the developers, and everyone in-between over comprehensive tools, documentation, and rigid plans that are unable to adapt to incoming change. [33]

As described by Highsmith et al, agile methodologies embrace inevitable change, as opposed to traditional approaches that seek to 'anticipate the complete set of requirements early and reduce cost by eliminating change'. [34]

3.1.1 Adaptation of Scrum

With prior experience of using agile methodologies in both industry, and in previous academic work, it was quickly decided upon that Scrum would be a suitable candidate framework to adapt and incorporate aspects of to better appropriate it for solo development, and to use it in conjunction with Feature-Driven Development, which would be the primary methodology used.

Scrum is characterised by three primary roles: a Product Owner, a Scrum Master, and a Developer, all of whom carry out unique roles and carry distinct responsibilities within the organisation or development body. In addition, events, such as but not limited to sprints - a fixed unit of time to denote work nominated from a product 'backlog' is to be completed, with regular face-to-face meetings known as daily scrums (or stand-ups), of which the framework derives its name, used to relay progress information and potential setbacks to other members of the team. [35] In turn, the terminology and analogies used by Scrum, including the name itself, are borrowed from the game of rugby. As stated by Takeuchi et al, 'as in rugby, the ball gets passed within the team as it moves as a unit up the field.' [36]

The main adaptations made to the framework and incorporated into the custom agile methodology used were the assimilation of all roles into a singular role - incorporating the actual development work expected of a software engineer, the responsibilities of a Scrum Master ensuring that the methodology is followed as pragmatically as possible, and the various aspects and duties carried out by a Product Owner, such as communicating and discussing requirements or changes with the customer. [37]

Secondly and most notably, changes were made to the traditional system of sprints as measurements of time. They were instead simplified into a straightforward time-frame Gantt chart, with date estimates and guidelines that could be modified where necessary, but nevertheless provided an indication of progress and momentum as the ultimate project deadline draws closer. This Gantt chart is available as Appendix D.

3.1.2 Adaptation of Feature-Driven Development

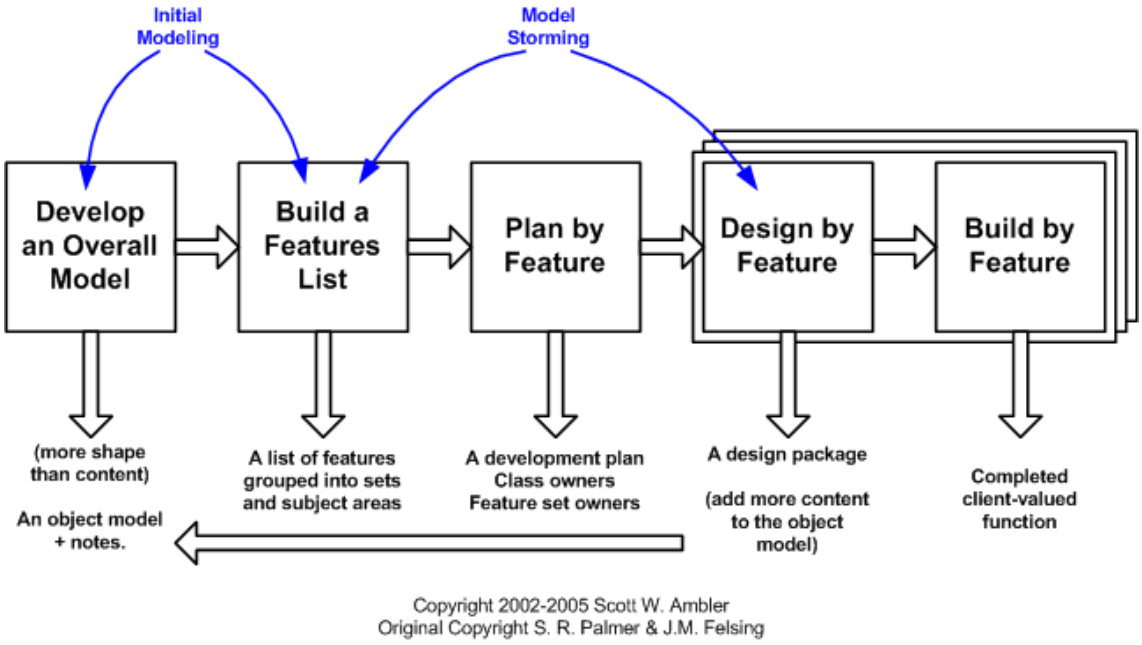


Figure 3.1: The Feature-Driven Development Lifecycle

[38]

With a well-laid out set of initial requirements defined at the start of the project, Feature-Driven Development was determined to be the most appropriate methodology for use with the project alongside the aforementioned adaptations adopted from Scrum. The concept, consisting of the five processes shown in figure 3.1 was formulated in 1997 by Jeff Luca in order to aid the software development needs of a bank in Singapore. [39] These processes can be broken down as follows:

1. Develop an Overall Model
    - The first step is to produce what is, as described by Goyal, a 'high-level walkthrough of the scope of the system and its context.' [40]
2. Build a Features List
    - The second process is to produce a hierarchical list of features, typically organised by category. [40]
3. Plan By Feature
    - The third process determines the order in which features are developed, and typically would also involve the distribution of tasks to team members. [40]
4. Design by Feature
    - According to Karam, the lead programmer selects what features will be developed next, and a domain expert is expected to begin analysing and 'designing a solution to each feature.' [41]
5. Build by Feature
    - Development work on the nominated features begins, and Karam goes on to further explain, that unit tests should also be produced to ensure that the code produced meets the expectations of the lead programmer. [41]

As a result of the usage of this methodology, with the only notable modification being the consolidation of roles into a single person, and a simplified requirements list as discussed in Chapter 1.

3.2 Version Control

Version control in software engineering began as the adoption of best practices and techniques that were used in neighbouring fields of industrial manufacturing and design, predominantly for the similarly complex drawings and blueprints of advanced components. [42] Today, the vast majority of software projects depend on Git - an open source, distributed version control system (DVCS) that was developed by Linus Torvalds originally for use in Linux kernel development, further popularised by online repository hosts such as GitHub. [43]

This project makes use of a Git repository in order to track changes and a historical record that can be reverted back to should code regressions or decision changes be made, as well as keeping an online copy uploaded to the GitHub online repository hosting service so that it is additionally

protected against accidental deletion or corruption. The use of Git nevertheless complements agile development even as a solo developer, as Robinson elaborates that 'simply being able to see changes and go back to an earlier version can be a huge help.' [44]

3.3 Supporting Technologies

This section explores some of the computer applications that were used in order to aid the implementation of the explored hybrid methodology.

3.3.1 Discord

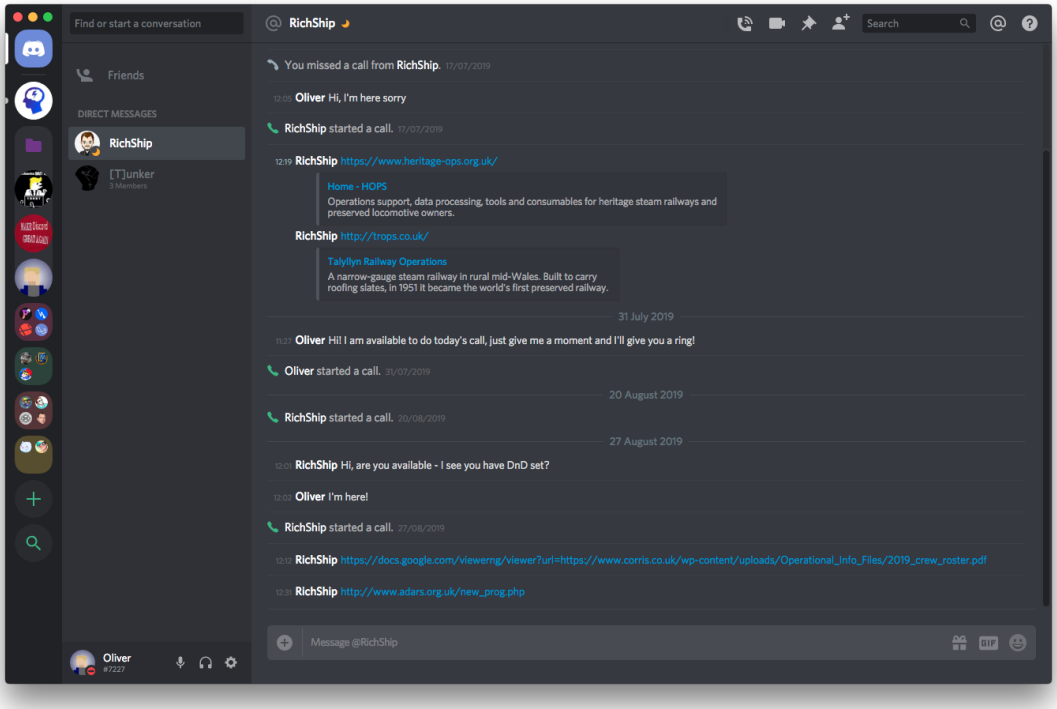


Figure 3.2: A screenshot showing communication history between client and developer, in Discord.

Communication, particularly face-to-face communication is an important component of agile methodologies, as described by Inzaurraga as 'making the feedback cycle faster' and 'ensuring development meets the client's expectations'. [45] The further importance of face-to-face communication is emphasised by Cao et al, who state that such communication 'obviates the need for time-consuming documentation and approval processes' and that it allows 'customers to steer the project in unanticipated directions.' [46]

Due to geographical constraints making face-to-face communication difficult, an ideal alternative is audio and video conferencing, which Discord was quickly mutually decided upon to be the facilitatory technology. Discord is a proprietary Voice over IP (VoIP) application available for desktop and for mobile platforms, or usable in a web browser, with functionality including voice and video calling, and importantly, the ability to share a user's desktop view with call participants, a feature known as screen sharing. [47] Despite the program originally being designed for use by those interested in video games, the application has quickly grown in appeal to other audiences, and competes openly with popular rival messaging and VoIP software, such as Microsoft's Skype. [48]

Weekly voice and video meetings with shared screens were conducted with the client using Discord, as well as facilitating immediate, ad-hoc instant messaging and link sharing whenever necessary, as shown in figure 3.2.

3.3.2 Microsoft Outlook

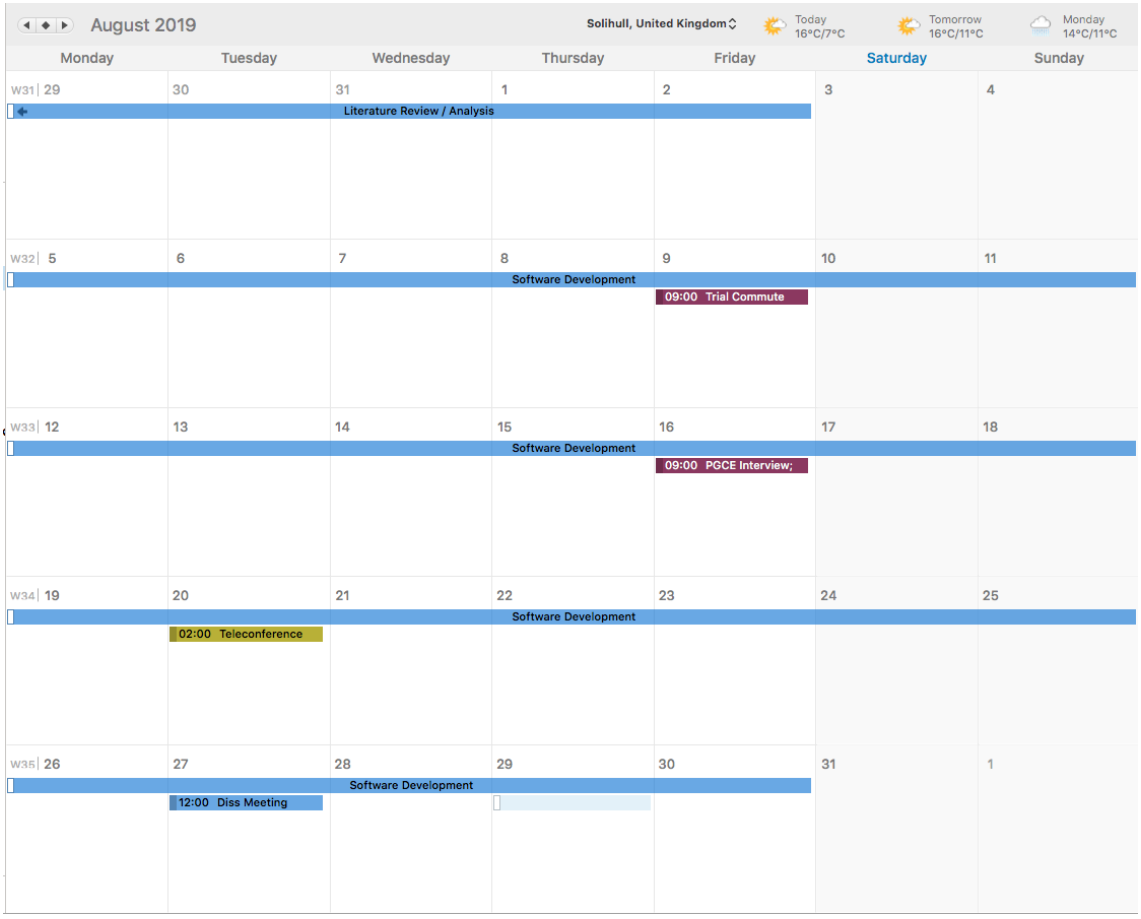


Figure 3.3: Calendar view in Microsoft Outlook, displaying development time periods.

In order to ensure that development remains disciplined and that the time-frames illustrated in Appendix D are maintained as closely as possible, the date durations were added to an appointments calendar within Microsoft Outlook, a popular email, calendar, and newsgroup client developed by Microsoft for Windows and macOS, [49] which provided reminders as deadlines approached and as demonstrated by figure 3.3 were easily viewable at a glance.

While more fitting alternatives for agile software development exist, such as facilities provided by GitHub itself, and also solutions such as Atlassian JIRA, this solution works fine and does not violate the popular KISS (Keep it simple, stupid) principle - a notion that originated in aeronautics, but has since become popular among many branches of engineering, software included. [50]

3.3.3 Evernote

Evernote is a notepad application with a wealth of rich-text formatting features, pre-built templates, and cloud synchronisation support. As it includes ready-to-use templates for to-do lists, complete with clickable checkboxes, it a useful tool for quickly keeping track of functionality that was deemed complete and fully tested. [51] This proved satisfactory for solo development, although for cooperative undertakings, more specialised tools such as the aforementioned Atlassian JIRA, or the use of a Kanban board would be more optimal.

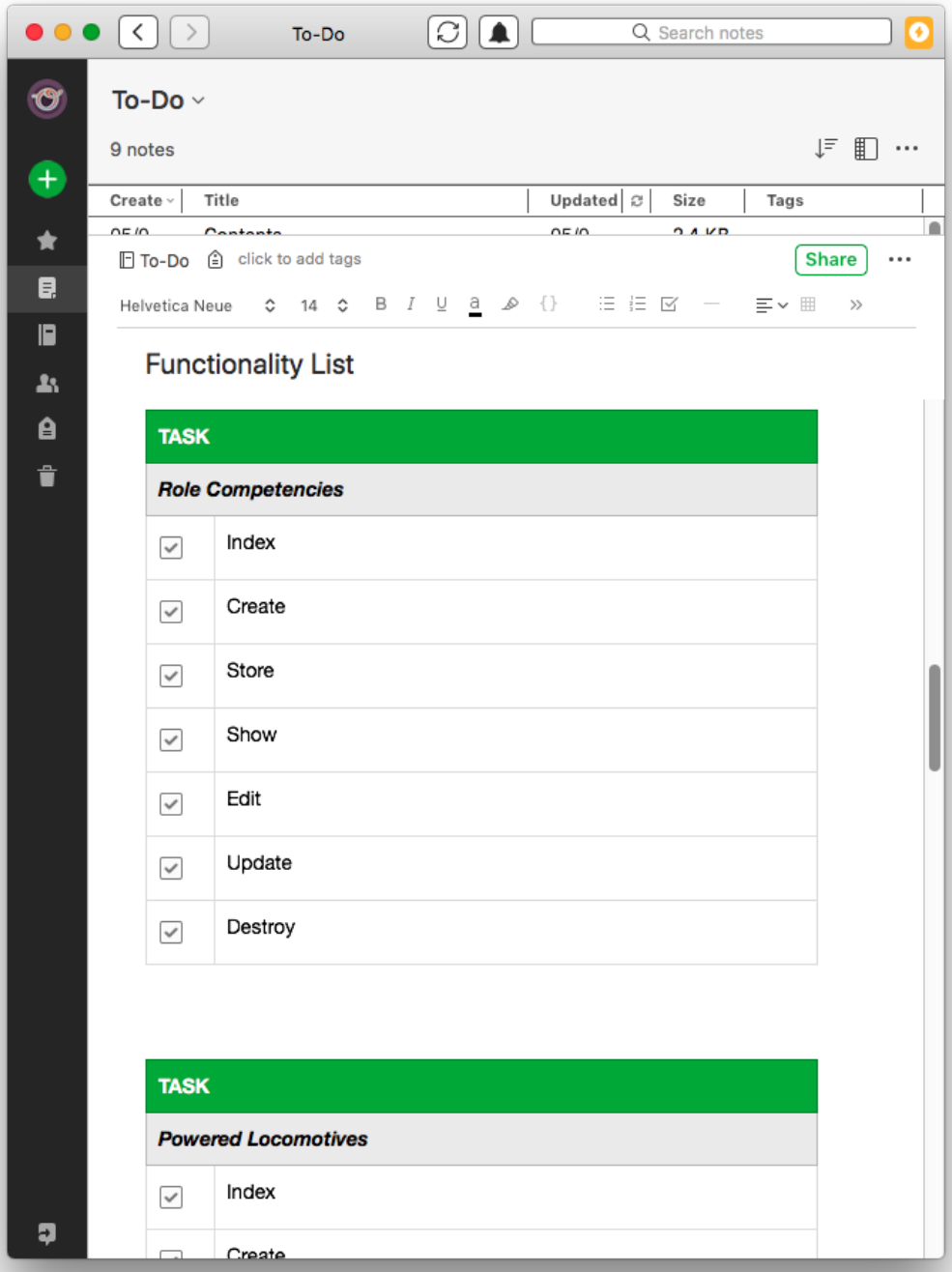


Figure 3.4: An example of feature checklists displayed in Evernote.

Chapter 4

Design

4.1 Analysis of Relevant Technologies

Determining what programming language(s) and framework(s) to use in a project implementation is an important decision process. A chosen technology must be runnable within the constraints defined in Appendix E, and ideally should be a technology that the development team, in this case a solo developer, is familiar with as to minimise excessive amounts of time learning.

On the other hand, there is an important balance that must be struck between choosing a tool that is familiar, and that which is most effective, as argued by Vinegar who states that 'writing quality, sustainable software comes second to fiddling with what's sexy'. [52]

Ultimately, the languages were refined down to PHP, Node.js, and Python, as they are all comfortably understood by the developer, appropriate for the task of constructing a full-stack web application, and should run on the designated GNU/Linux system environment without issue.

4.1.1 PHP

PHP, otherwise known by the recursive acronym PHP Hypertext Preprocessor, is one of the most famous and important programming languages on the Internet, originally written in 1994 by Rasmus Lerdorf. It can be used in a procedural or object-oriented style, and is allegedly used on more than 20 million websites today. [53]

While PHP can be used on its own to great success, there are numerous frameworks available that build on top of PHP to make it better suited for producing full-stack web monoliths that will likely require the use of the popular Model View Controller (MVC) design pattern. This design pattern is described in greater detail in Design Patterns.

The most popular framework of this nature is Laravel, originally conceived by Arkansas de-

veloper Taylor Otwell, who in around 2011 was unhappy with the most popular framework at the time, CodeIgniter. [54] Today, it features a range of features such as built-in authorisation and authentication, MVC clearly built-in as the core design pattern in mind, security mitigations, and most notably, abstracts a lot of the complex database work that would be required in this project behind an Object-Relational Mapper (ORM) called Eloquent, heavily based on ActiveRecord that is found in Ruby on Rails. This functionality would drastically simply development efforts and reduce the quantity of boilerplate code that would need to be written, when the framework can simply be leveraged instead. [55] [56]

4.1.2 Node.js (JavaScript)

JavaScript, sometimes referred to as ECMAScript due to standards specifications it conforms to, [57] was famously written in 10 days by then Netscape employee Brendan Eich for use on the Internet as a client-side, browser interpreted language. [58]

Node.js is an implementation of JavaScript on the server-side and outside of the DOM (Document Object Model) found in webpages that was created in 2009, where over the last decade it has been adopted rapidly by organisations big and small as it unifies the choice of language to JavaScript across the entire web stack. [59]

Like PHP, while it can be used on its own (sometimes referred to as 'vanilla'), there exists a range of frameworks that simplify and better purpose Node.js for use in building complex full-stack web applications and RESTful (Representational State Transfer) APIs. (Application Programming Interface) The most popular example of which, is Express, a minimalist and unopinionated (flexible and easily restructured) released in 2010. [60]

While Express eases the workload in developing working routing and other components that make up the MVC design pattern, it is as previously mentioned, minimalist. This means that a lot of the complex work such as working with databases, providing security layers, and programming authentication would need to still be carried out, or delegated to additional libraries, further complicating development.

4.1.3 Python

Python is a scripting language that was created in 1990 by Dutch programmer Guido van Rossum. [61] Today, it is one of the most popular programming languages in the world, supporting multiple paradigms and is used not only as an introductory language intended for beginners, but also in fields such as machine learning, embedded systems, and of course, web development. [62]

The two most popular Python web frameworks are Flask and Django - with Flask taking an unopinionated, minimalistic, plugin approach like Express does, whereas Django more closely

resembles Laravel and Ruby on Rails for its rich features, such as built-in authentication, and the use of an ORM.

4.1.4 Conclusion

After a substantial amount of consideration, it was decided upon to use Laravel and PHP for use of the project. Whilst JavaScript and Node.js continue to increase in popularity thanks in part to a unification of languages across the entire web stack, allowing for the reuse of libraries and skill sets for development, testing, etc. it would take significantly more time to develop much of the underlying functionality and architecture that would be necessary for powering an MVC web application, due to Express' minimalism and specific design for lightweight APIs and microservices, as opposed to monolithic, full-stack apps. [63]

Django and Python would be a perfect fit for the application - providing a near identical set of functionality and boilerplating necessary to develop such an application quickly. Ultimately, personal preference and experience with the PHP ecosystem and best practices, including years of prior usage in substantial academic work and personal projects made PHP a more comfortable choice, and as there are no known downsides or impracticalities from opting for said language, it was an easy decision to make, as opposed to unnecessarily familiarising oneself with new technologies when there are looming deadlines for completion of work, as well as needing to demo work and progress to the client on a weekly basis. As written in the Agile Manifesto, working code is the most ideal indicator of progress. [64] [65]

4.1.5 Front-End Considerations

With PHP and Laravel determined as the back-end language and framework of choice, attention must be paid to any necessary design considerations, libraries, or technologies that will be needed on the front-end. Laravel includes Laravel Mix, an implementation of popular JavaScript workflow tool Webpack, which is used to minify any JavaScript or stylesheet code written, as well as to compile superset languages such as Sass into their basic languages (in this case, CSS) so that it can be interpreted by a web browser. By default, the stylesheet markup language Sass, and the JavaScript framework Vue.js are included with Laravel, but they can be readily interchanged with other technologies such as React or Less. [66]

One of the main requirements put forth by the customer in regards to front-end design is the need for responsive design, that is, for the application to work and present itself seamlessly regardless of viewport size or device specification: i.e. on a computer, tablet, or smartphone. By including Bootstrap 4, a popular CSS and JavaScript framework developed by Twitter which is additionally included in Laravel by default, the application can be made with responsiveness in

mind. While it is not necessary and responsiveness can be implemented in plain CSS or Sass, such as by leveraging the new CSS Grid standard, this is arguably not as quick to scaffold and present to a customer, at the cost of the overhead of requiring to download additional styling and code to be executed within the browser. [67]

4.2 Design Patterns

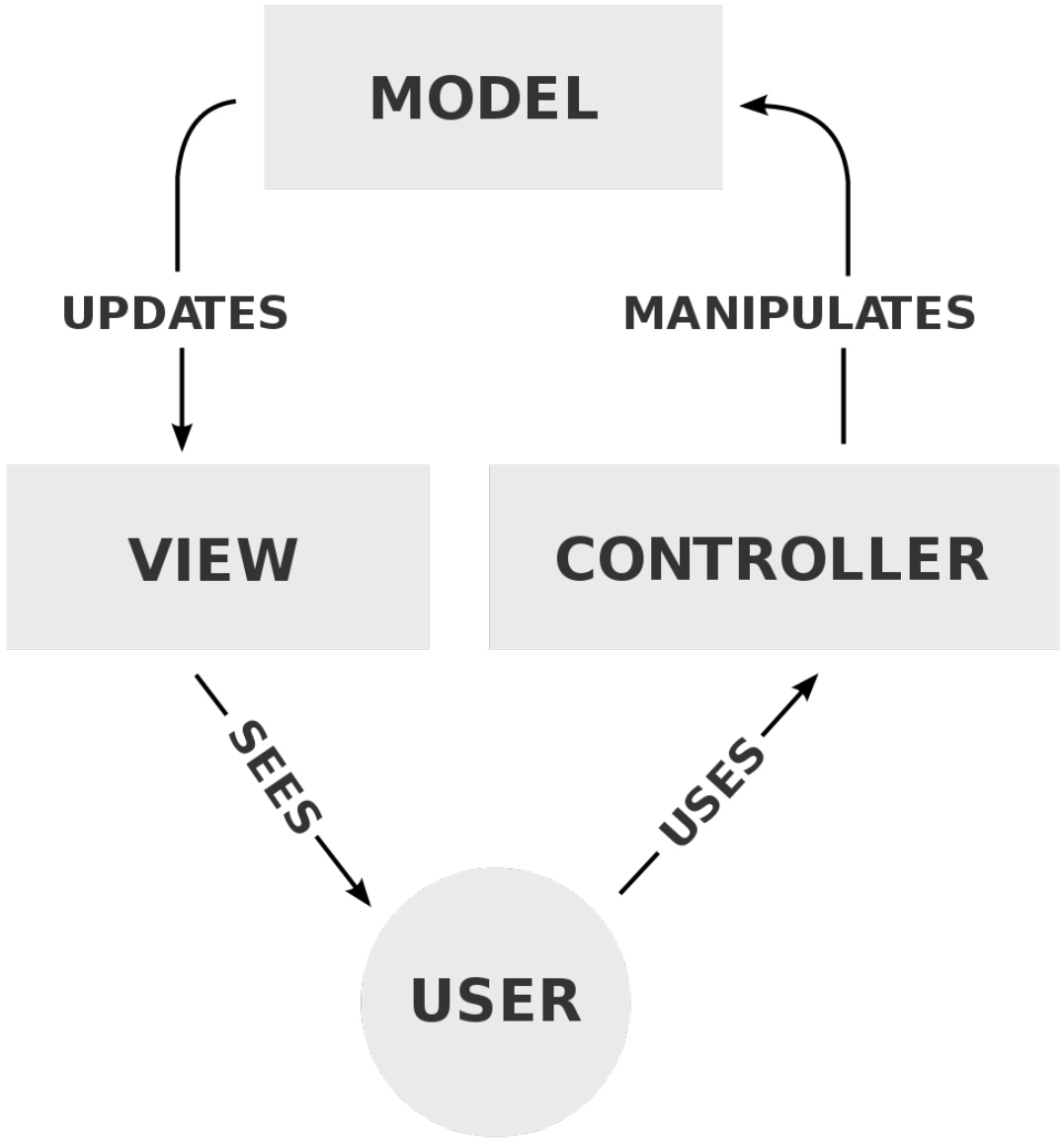


Figure 4.1: Model View Controller (MVC) Design Pattern

[68]

Laravel, like many application frameworks utilises the Model View Controller architectural pattern in order to help separate concerns and to help compartmentalise code into easily reusable sections. [69] These concerns can be broken down like thus:

- The model relates directly due to the data domain, reflecting data held in the database, and almost always is responsible for all actions that involve database access.
- The view is what the user sees and interacts with, which can pass instructions back to controllers.
- Controllers facilitate communication between the Model and the View, and encapsulate other business logic also. [70]

It is important to note however, that although Laravel utilises MVC, like many other frameworks, it is does not necessarily consider itself to be such and continues to be a disputed topic in the web development community. [71] Nevertheless, the application will be developed with this pattern in mind.

4.3 Data Persistence

PHP supports a wealth of relational databases out-of-the-box, and this functionality is extensible with the use of drivers. For development, the use of a SQLite or local MySQL database is more than sufficient, and both Laravel and PHP itself will work with different databases without the need for different code or parameters. The PDO (PHP Data Object) is the means in which PHP applications can interface with databases, which fully abstracts away any underlying database mechanisms and allows developers to write fully database agnostic code, without the need of knowing what dialect of SQL, or even writing database queries at all. Resultingly, the customer is free to use whatever database technology they wish, provided that they have the necessary drivers installed on their PHP installation or within their container, such as MySQL, PostgresSQL, or SQLite. [72]

4.4 Security Considerations

An application working with user information, regardless of sensitivity must address certain security considerations in order to conform with the Data Protection Act of 2018, which includes the ability for users to view, edit, and delete aspects of their personal information. This was updated that year to include the British implementation of the European General Data Protection Regulation (GDPR) which has come into effect as the United Kingdom withdraws from the European Union. Charities are not exempt from this legislation. [73] [74]

It is a well-established concept in software engineering that data from a client (i.e. data from a user) cannot be trusted and must be properly sanitised and validated in order to protect underlying data systems from invalid input, especially malicious input designed to bring harm to data or the systems containing it, such as by means of a SQL Injection attempt - manipulating data inputs to run arbitrary SQL code. [75] [76]

A wealth of functionality exists within Laravel and will be leveraged in order to mitigate security risks as aforementioned, including full validation and authorisation checks, automatic escaping of data retrieved from database, and CSRF (Cross Site Request Forgery) protection. Passwords are also automatically encrypted when stored in the database and remain so when accessing the user model, providing additional protection. [77] [78]

4.5 Development Environment

The application in a macOS (UNIX) environment, on a desktop computer with the following specifications:

- Mac Pro (Early 2008)
- macOS 10.3.6 High Sierra Operating System
- 2 x 2.8GHz Quad Intel Xeon E5462 Processors
- 32GB 667MHz Random-Access Memory (RAM)
- NVIDIA GeForce GTX1050 Ti Graphics Card
- Solid-State Drive

Software development is carried out using JetBrains PhpStorm IDE (Integrated Development Environment) thanks to its powerful Laravel and PHP code analysis and refactoring tools that are provided out-of-the-box, making it the most specialised program compared to standard code editors. [79]

4.6 Use Case Design

Based on the full requirements list in Chapter 1, there are two primary use cases identified for the application - both for everyday volunteers (users), and administrators. The above diagram shows that the a great deal of functionality is accessible by both user groups, and that administrators have a superset of what could be considered user privileges, such as the ability to add, delete, and

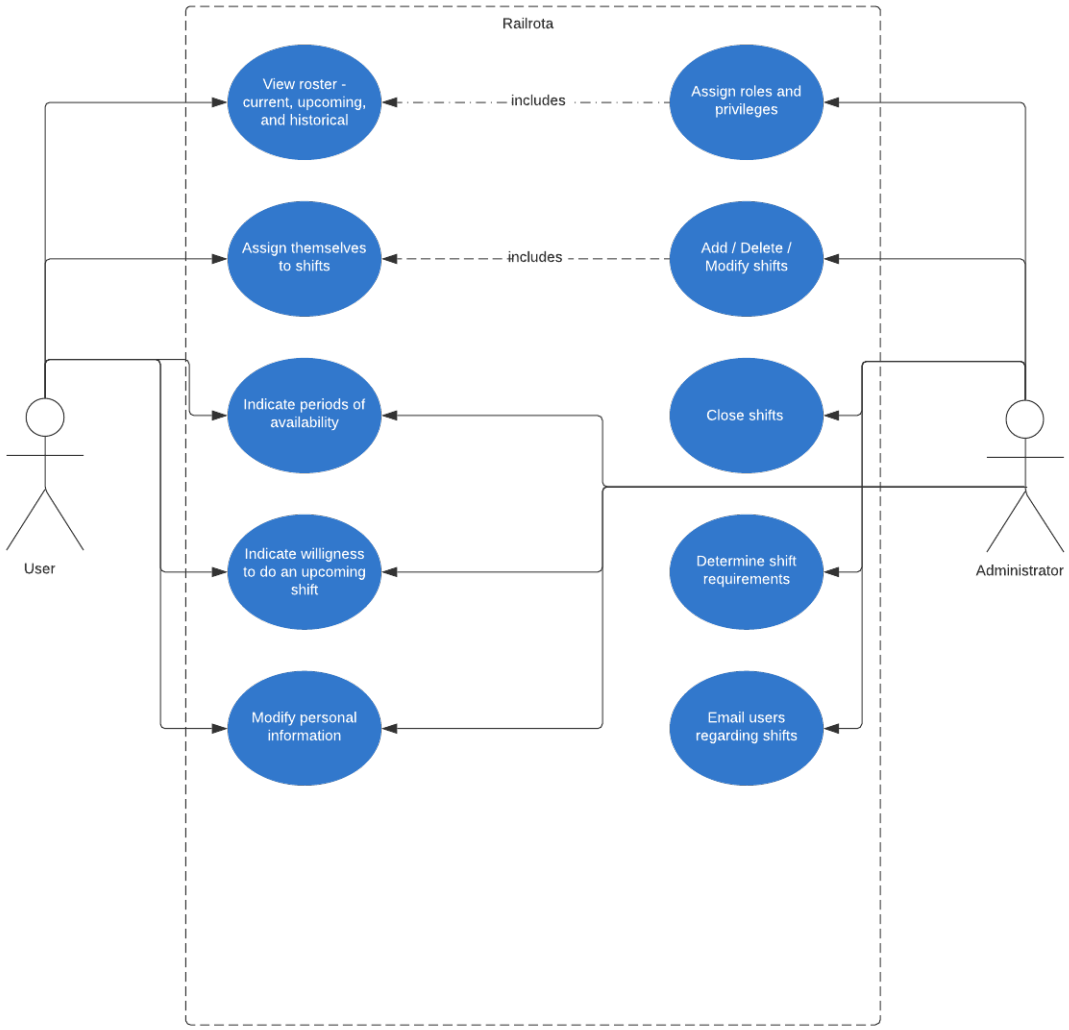


Figure 4.2: Use Case Diagram.

modify shifts, in addition to being able to register for a shift. This is because administrators could readily be volunteers themselves and may wish to use the application as such without the need for an additional account.

Users on the other hand, do not have use cases that involve the manipulation of data outside of their own personal information that can and should be updated whenever necessary.

4.7 User Interface Design

The UI design went through two phases - initial sketches done on paper based on initial ideas for how the layout for the application will appear, followed by digital artwork that can be considered the final design before implementation.

In traditional development methodologies, design is something that is done extensively at the start of application development with the final outputs expected to closely resemble the end product. However, agile methodologies, including this Scrum adaptation, recognise that design is a continuous process throughout development, and that the end product may vary vastly from any drawings or early documentation as the result of changing requirements or additional variables such as customer feedback - thus making design a far more cooperative, agile process. [80]

4.7.1 Early Concept Art

Early concept art can be found in Appendix F due to their large resolutions.

4.7.2 Final Iteration

These rudimentary designs were expanded upon in much more detail that demonstrates a far more mature iteration of the design process.

4.7.2.1 Homepage

The homepage design as visible from desktop viewports provides insight into the UI and how it should appear, most notably showing off the navigation bar.

4.7.2.2 Responsive Design

With the aforementioned emphasis and importance of responsive design and an importance of the design to work and view correctly on mobile devices such as smartphones and tablets, a reduced viewport design was considered important. In particular, this is to demonstrate the use of a collapsible 'hamburger menu', a commonly used drop-down menu optimised for touchscreens, and readily implementable in Bootstrap 4. [81]

4.7.2.3 Operations Design

This design expands upon the drawing, showing in more detail how each operation might be displayed on the application, with respective information on available shifts, the volunteers either

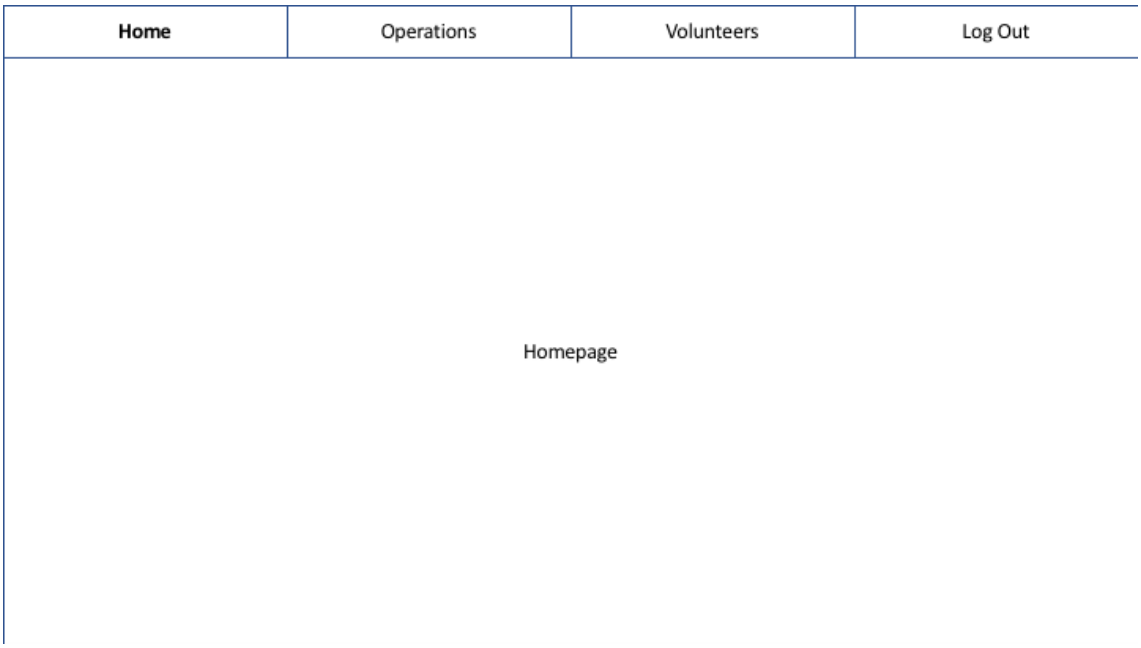


Figure 4.3: Default design of the homepage as viewed on a desktop web browser.

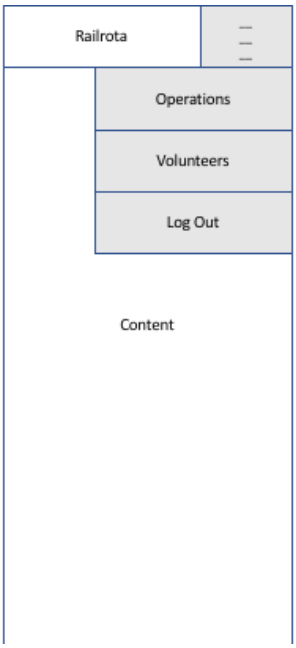


Figure 4.4: Design of how the web application responsively appears in smaller viewports.

staffing them or whether there are vacancies, and additional buttons for actions on that particular operation.

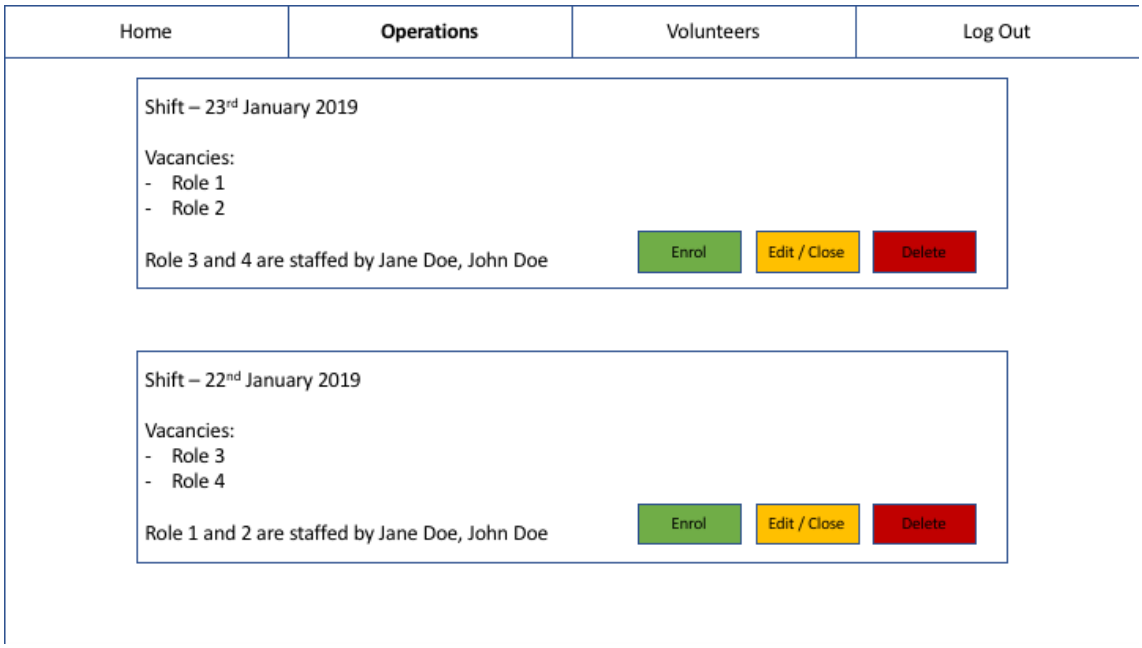


Figure 4.5: Design of the operations page that displays operations and shifts.

Figure 4.6: Design of the operations page that displays operations and shifts.

4.7.2.4 Volunteers Design

An additional design was drawn up to demonstrate how a tabular view might look, in this instance, displaying a list of volunteers with their personal details, so that a volunteer or an administrator can quickly look up a fellow volunteer and view their public information, such as their availability

or contact information.

4.7.2.5 Glance / Calendar Design

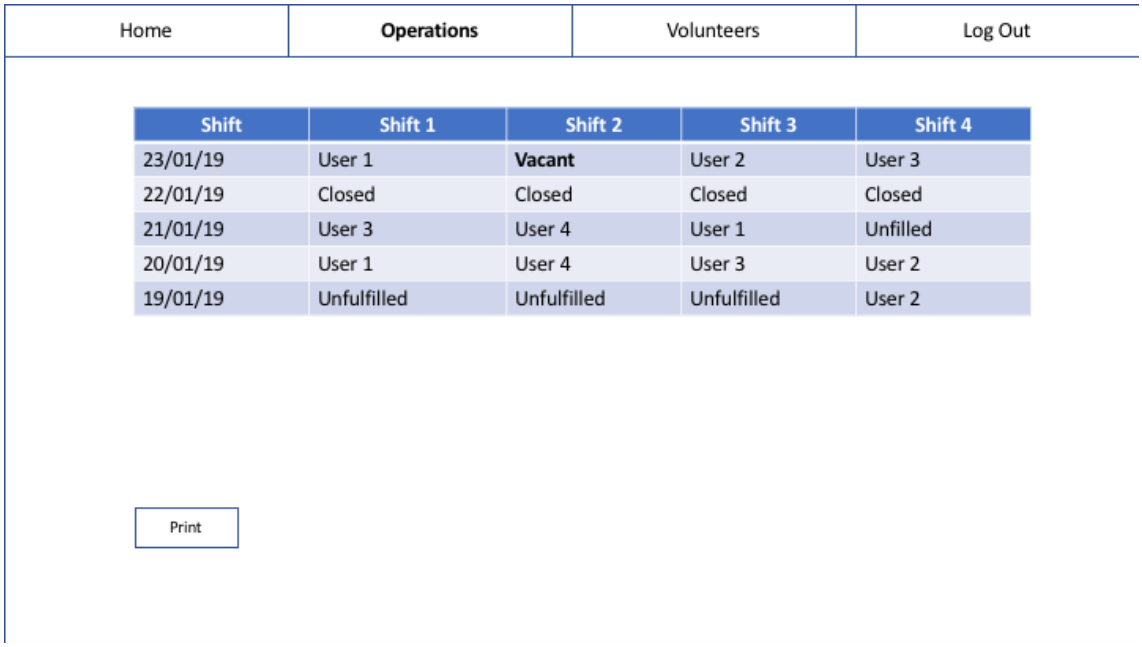


Figure 4.7: Design of the more compact operations view, more akin to a calendar.

Following a conversation with the customer, they expressed a desire to have a view, potentially downloadable and/or printable, that could show 'at a glance' the available shifts and the current staffing at that moment in time. This should somewhat resemble the existing system that is shown in Appendix C. This is desirable functionality that came into being after development actually began, with the intention in including it into a release of the software, time permitting.

Chapter 5

Implementation

5.1 Application Naming

While no formal name was proposed by the customer for the application, it is referred to informally as *Railrota*, both in the source code and within supporting documents such as a mock logo used in the front-end, Git repository, and various readmes. This can be changed at any time in the `.env` environment file in the program root directory. Therefore, from this point, the application may be occasionally referred to as Railrota.

5.2 Directory Structure

The directory structure (for the application only) is defined as follows:

- **app** contains various Laravel application components, such as models, controllers, and policies
- **bootstrap** Framework code used by Laravel to bootstrap the application
- **config** Framework code that define configurations for various Laravel utilities and functionalities
- **database** Database factories, migration definitions, and seeding tools are in this directory.
- **docs** Documentation generated by Sami is stored here. This is explored in more detail in Chapter 7.
- **node_modules** This contains Node.js dependencies installed by Laravel Mix and are used in development. This is discussed in greater detail in Chapter 7.

- **public** is the front-facing directory containing the application entry-point, as well as compiled assets and images used for social media.
- **routes** HTTP Application routes are defined in the files contain within this directory.
- **resources** Views written in Laravel Blade are kept in this folder, along with stylesheets and scripts that comprise the front-end.
- **storage** Laravel stores log files here, and generated PDF files for download are kept here. A cache for quick loading is also stored here.
- **tests** PHPUnit tests are kept here, and are covered in Chapter 6.
- **vendor** Similar to *node_modules*, this folder stores PHP dependencies used by Laravel, as well as additional third-party libraries.

[82]

5.3 Method of Action

The Laravel developers acknowledge that having a basic, high-level understanding of how a framework works is important for not only having a basic grasp of how the application works, but also to make 'everything feel less magical'. [83] This is important as frameworks are characterised by carrying out substantial amounts of work for you, in order to drastically reduce development time and to produce better quality applications, and are generally recommended by the greater web development community. [84]

When a HTTP (Hypertext Transfer Protocol) request is received by the web server by navigating to the **index.php** file in the *public* directory, the application is quickly bootstrapped, loading all necessary libraries and components by means of the PSR-4 autoload file, which is a widely accepted PHP coding standard and was generated by Composer, the PHP package manager when the application was first installed. [86] From here, the application can determine how to 'route' the request based on route definitions and the URL (Uniform Resource Locator) path, delegating actions to the appropriate controllers and their models defined in those definitions, before passing data to a view, and returning it to the user. [83] According to Stauffer, Laravel's approach 'brings your ideas to reality with no wasted code' and 'modern programming standards.' [87]

5.4 HTTP Routes

The controllers for each resource (i.e. users, operations) as well as individual webpages uses a RESTful (Representational State Transfer) architecture, and have methods that correspond to the

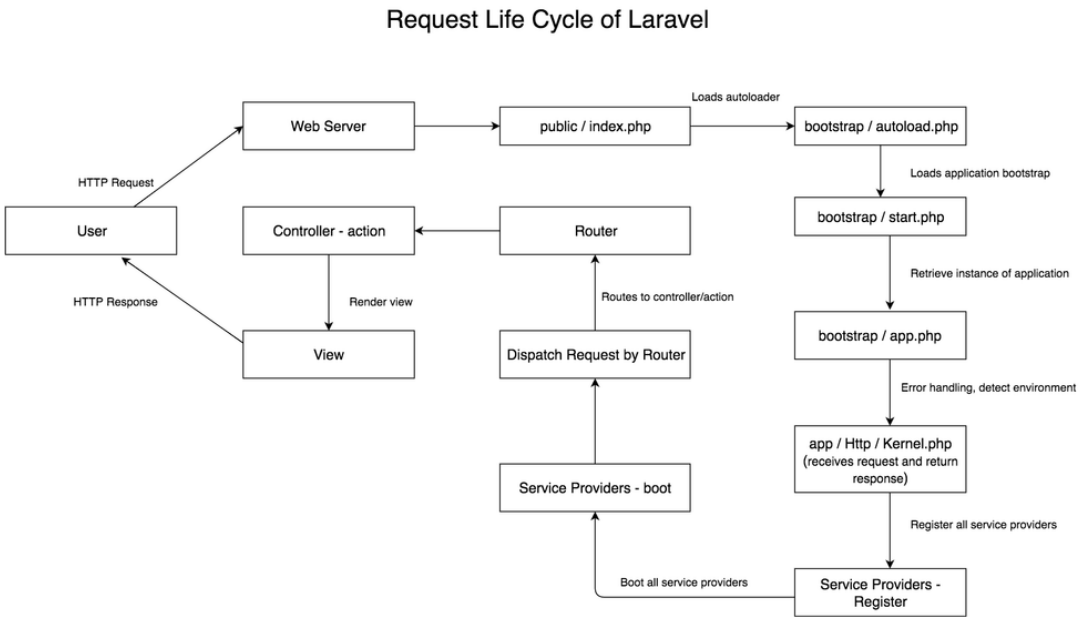
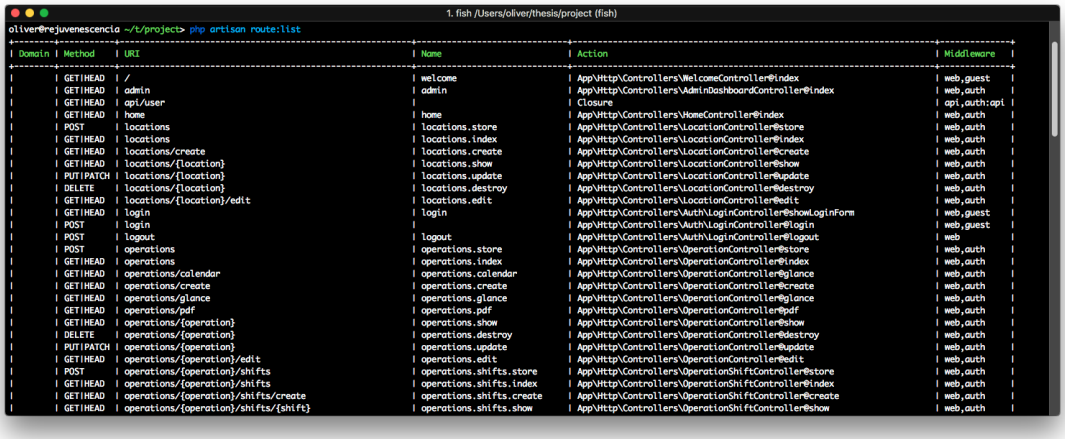


Figure 5.1: Diagram that shows how the Request Lifecycle for the Laravel framework operates. [85]



```
1. fish /Users/oliver/thesis/project (fish)
oliver@rejuvenescencia ~/t/project> php artisan route:list
+--------+-----------+-----------------------------------------------------------+-----------------------------+-----------------------------------------------------------------------------+-------------+
| Domain | Method    | URI                                                       | Name                        | Action                                                                      | Middleware  |
+--------+-----------+-----------------------------------------------------------+-----------------------------+-----------------------------------------------------------------------------+-------------+
|        | GET|HEAD  | /                                                         | welcome                     | App\Http\Controllers\WelcomeController@index                                | web,guest   |
|        | GET|HEAD  | admin                                                     | admin                       | App\Http\Controllers\AdminDashboardController@index                         | web,auth    |
|        | GET|HEAD  | api/user                                                  |                             | Closure                                                                     | api,auth:api |
|        | GET|HEAD  | home                                                      | home                        | App\Http\Controllers\HomeController@index                                   | web,auth    |
|        | POST      | locations                                                 | locations.store             | App\Http\Controllers\LocationController@store                               | web,auth    |
|        | GET|HEAD  | locations                                                 | locations.index             | App\Http\Controllers\LocationController@index                               | web,auth    |
|        | GET|HEAD  | locations/create                                          | locations.create            | App\Http\Controllers\LocationController@create                              | web,auth    |
|        | GET|HEAD  | locations/{location}                                      | locations.show              | App\Http\Controllers\LocationController@show                                | web,auth    |
|        | PUT|PATCH | locations/{location}                                      | locations.update            | App\Http\Controllers\LocationController@update                              | web,auth    |
|        | DELETE    | locations/{location}                                      | locations.destroy           | App\Http\Controllers\LocationController@destroy                             | web,auth    |
|        | GET|HEAD  | locations/{location}/edit                                 | locations.edit              | App\Http\Controllers\LocationController@edit                                | web,auth    |
|        | GET|HEAD  | login                                                     | login                       | App\Http\Controllers\Auth\LoginController@showLoginForm                     | web,guest   |
|        | POST      | login                                                     |                             | App\Http\Controllers\Auth\LoginController@login                             | web,guest   |
|        | POST      | logout                                                    | logout                      | App\Http\Controllers\Auth\LoginController@logout                            | web         |
|        | POST      | operations                                                | operations.store            | App\Http\Controllers\OperationController@store                              | web,auth    |
|        | GET|HEAD  | operations                                                | operations.index            | App\Http\Controllers\OperationController@index                              | web,auth    |
|        | GET|HEAD  | operations/calendar                                       | operations.calendar         | App\Http\Controllers\OperationController@glance                             | web,auth    |
|        | GET|HEAD  | operations/create                                         | operations.create           | App\Http\Controllers\OperationController@create                             | web,auth    |
|        | GET|HEAD  | operations/glance                                         | operations.glance           | App\Http\Controllers\OperationController@glance                             | web,auth    |
|        | GET|HEAD  | operations/pdf                                            | operations.pdf              | App\Http\Controllers\OperationController@pdf                                | web,auth    |
|        | GET|HEAD  | operations/{operation}                                    | operations.show             | App\Http\Controllers\OperationController@show                               | web,auth    |
|        | DELETE    | operations/{operation}                                    | operations.destroy          | App\Http\Controllers\OperationController@destroy                            | web,auth    |
|        | PUT|PATCH | operations/{operation}                                    | operations.update           | App\Http\Controllers\OperationController@update                             | web,auth    |
|        | GET|HEAD  | operations/{operation}/edit                               | operations.edit             | App\Http\Controllers\OperationController@edit                               | web,auth    |
|        | POST      | operations/{operation}/shifts                             | operations.shifts.store     | App\Http\Controllers\OperationShiftController@store                         | web,auth    |
|        | GET|HEAD  | operations/{operation}/shifts                             | operations.shifts.index     | App\Http\Controllers\OperationShiftController@index                         | web,auth    |
|        | GET|HEAD  | operations/{operation}/shifts/create                      | operations.shifts.create    | App\Http\Controllers\OperationShiftController@create                        | web,auth    |
|        | GET|HEAD  | operations/{operation}/shifts/{shift}                     | operations.shifts.show      | App\Http\Controllers\OperationShiftController@show                          | web,auth    |
```

Figure 5.2: A print out of HTTP routes defined within the application, their corresponding URLs, associated controllers and methods, and HTTP actions.

CRUD verbs - create, read, update, and delete. There are also additional methods that help facilitate the CRUD operations, such as 'show' for viewing individual entries, and 'edit' to display a form in order to modify an existing record. Different actions may expect different HTTP requests, such as GET, POST, PATCH, and DELETE respectively. Each action, whether viewing one's own profile, editing a user's profile, registering for a shift, or deleting an operation, are all represented by their own route within the application, including custom defined routes that do not correspond to CRUD, such as those used for (de)registering to an operation. [88] [89]

5.5 Models and Relationships

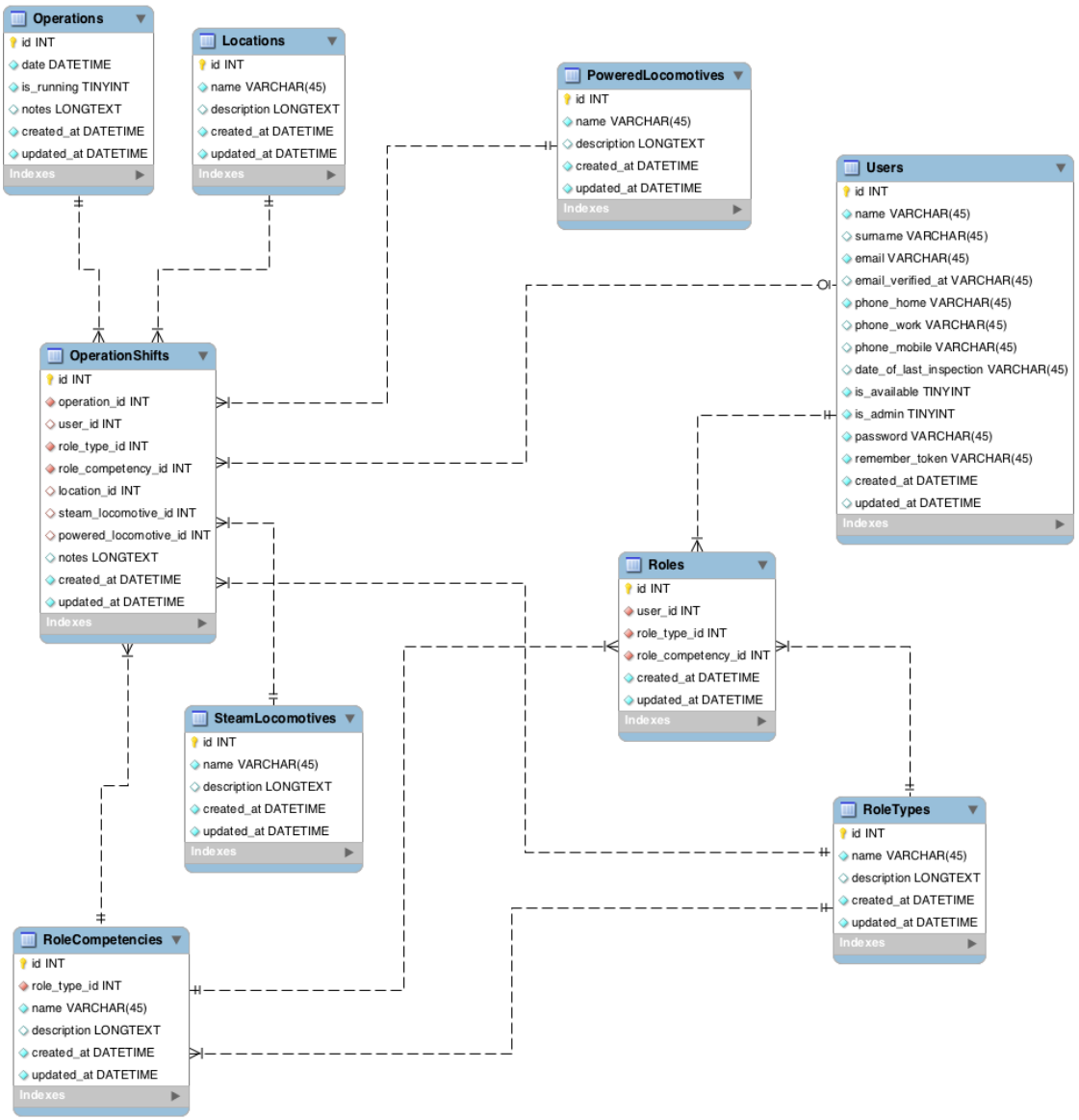


Figure 5.3: UML diagram showing the relationships between database tables and their structure. The relationships are defined by their Eloquent relationships.

Database models define their column names and data types within migration files, alongside foreign keys, their references, and any constraints. As shown by Figure 5.3, many of the tables belong to shifts, as each shift can have corresponding vehicles associated with it, a designated location, a required role and competency level, as well as the volunteer that is assigned to that volunteer, which is of course null when the shift is vacant.

5.6 Views and Laravel Blade

Controller methods, as part of their response to the user's request will return a view that can be populated with data processed or retrieved, such as returning the current resource object for display when viewing an individual item, like a role type. Laravel uses a language called Blade for its templating engine, which allows for reusable front-end code, and more legible syntax for inserting data into views or extending existing template files than using raw PHP alone, though vanilla PHP script is nevertheless supported and will be interpreted all the same. [90] [91]

An example of a data being passed to a view can be seen in the following code, that is taken from the controller for operations, where all operations retrieved from the database (by Eloquent) are injected into the view as an argument, to be parsed and displayed there:

```
public function index()
{
    $operations = Operation::orderBy('date', 'desc')->
        paginate(3);
    return view('operation.index', compact('operations'));
}
```

The name of the view to be rendered by Laravel is 'operation.index', which will now have access to the contents of `$operations`. In the following snippet of Blade code, you can see the contents of this variable being iterated through and printed in tabular form, as well as the usage of vanilla PHP script where it has been deemed necessary:

```
<tbody>
@foreach ($operations as $operation)
    @php
        $operationDate = \Carbon\Carbon::parse($operation->
            date);
    @endphp
    <tr>
    ...
    </tr>
    ...
@endforeach
</tbody>
```

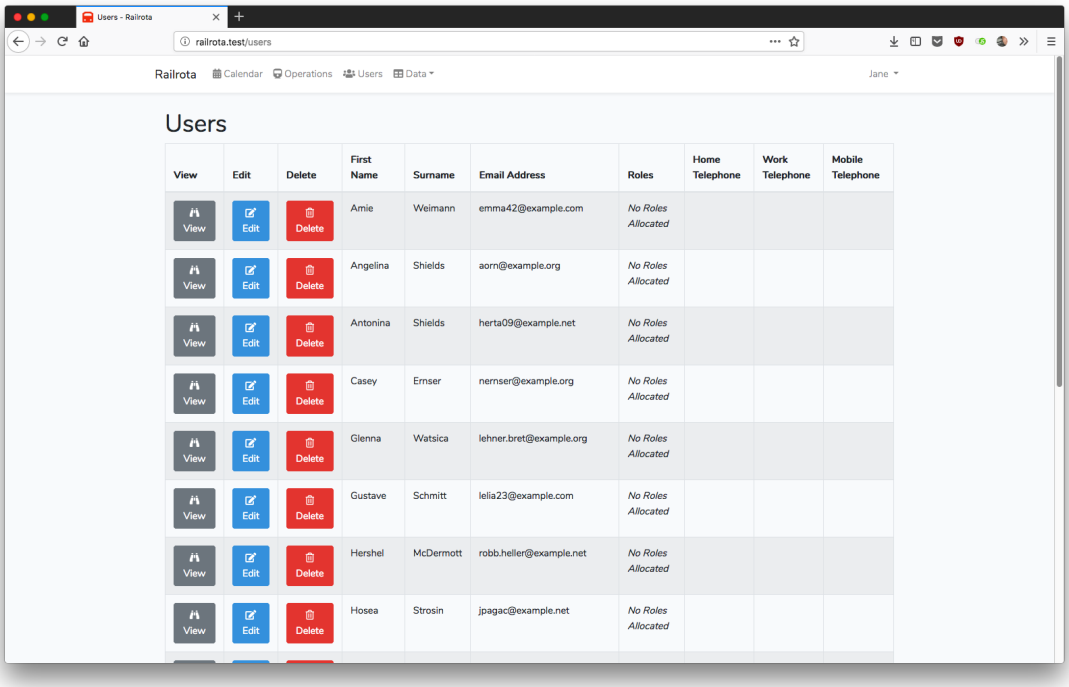


Figure 5.4: Screenshot of the Users interface within the application.

5.7 Users

All volunteers are users within the application, with users that have access to administrative privileges being denoted by a Boolean true value in **is_admin**. Users can modify a subset of their own information, but are unable to assign themselves roles or privileges, or modify a date value indicating the last time they were inspected.

The following destroy CRUD method in the User controller, demonstrates how after checking the policies that the current user is authorised to delete a user (so is an administrator), it checks first whether they are attempting to delete themselves, which is understandably undesirable behaviour, and redirects back with an error message if this is the case. Otherwise, the **delete** Eloquent method is called on the user object, and it is deleted from the database; redirecting back with a success notification. The User object is resolved from the unique ID in the URL. [88]

```
public function destroy(User $user)
{
    $this->authorize('delete', $user);

    if ($user->id === Auth::id()) {
        flash()->error('You can\'t delete yourself!')->
            important();
```

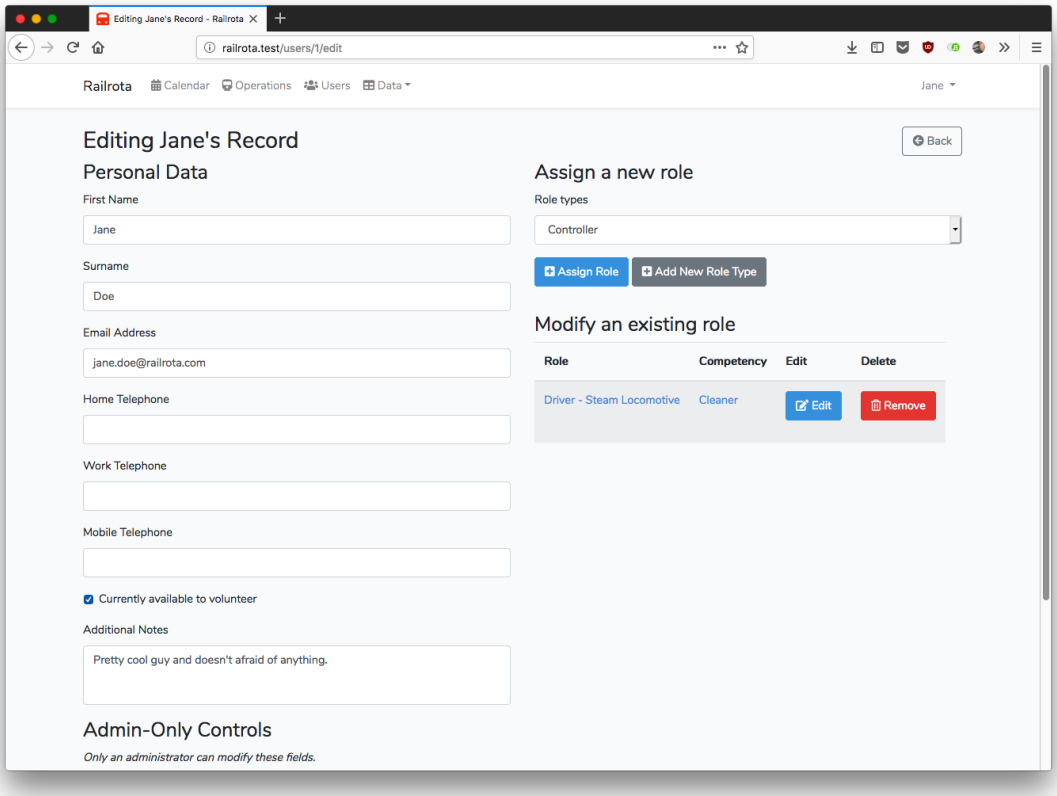


Figure 5.5: Screenshot of the Edit User interface, as seen by an administrator.

```
            return redirect()->back();
        }

        $user->delete();
        flash()->success("{$user->name} has been deleted
            successfully!")->important();
        return redirect()->route('users.index');
    }
```

5.7.1 Authentication

Authentication functionality can be scaffolded seamlessly by Laravel by issuing the `php artisan make:auth` command at on the command line, and provides all the necessary registration, login, and password reset functionality - and automatically ties this into the User model.

By declaring routes within the authentication middleware group as the following code demonstrates, Laravel will ensure that a user is authenticated in order to access any of the routes contained

therein, otherwise redirecting back to the login page. [92]

```
Route::group(['middleware' => ['auth']], function() {
    Route::resource('users', 'UserController');
}
```

5.7.2 Authorisation and Policies

Authorisation - determining who can and who cannot access resources or carry out certain actions is determined by classes known as policies within Laravel - where authorisation logic is stored. [93] In this application, a policy is necessary wherever there are actions that users and administrators have different levels of access to, such as being able to create roles, operations, or modify user data.

A corresponding method is typically called in a controller's respective policy (i.e. UserController will have a matching policy called UserPolicy) and within the controller action itself, the appropriate method for that action is passed as an argument, alongside the currently authenticated user object, like thus:

```
$this->authorize('create', $user);
```

This calls the corresponding create method in its policy file, which will return either true for successful authorisation and allowing the remainder of the controller method to execute. Otherwise, the action is prevented. A call to this method absolutely must exist for any authorisation to occur, regardless of whether there is a corresponding policy class to the aforementioned controller and model. The corresponding 'create' policy method in the UserPolicy file is as follows:

```
public function create(User $user)
{
    return $user->isAdmin();
}
```

This simple but effective method calls a custom method on the model itself that returns whether or not the attribute **is_admin** on the currently authenticated user is true or not. If the value isn't truthful, then execution halts as authorisation has subsequently failed.

5.8 Role Types

In order to volunteer for a shift, a user must have the relevant role type that is also shared by the shift. For example, a shift might have a *Blockman* vacancy available, but in order to volunteer for

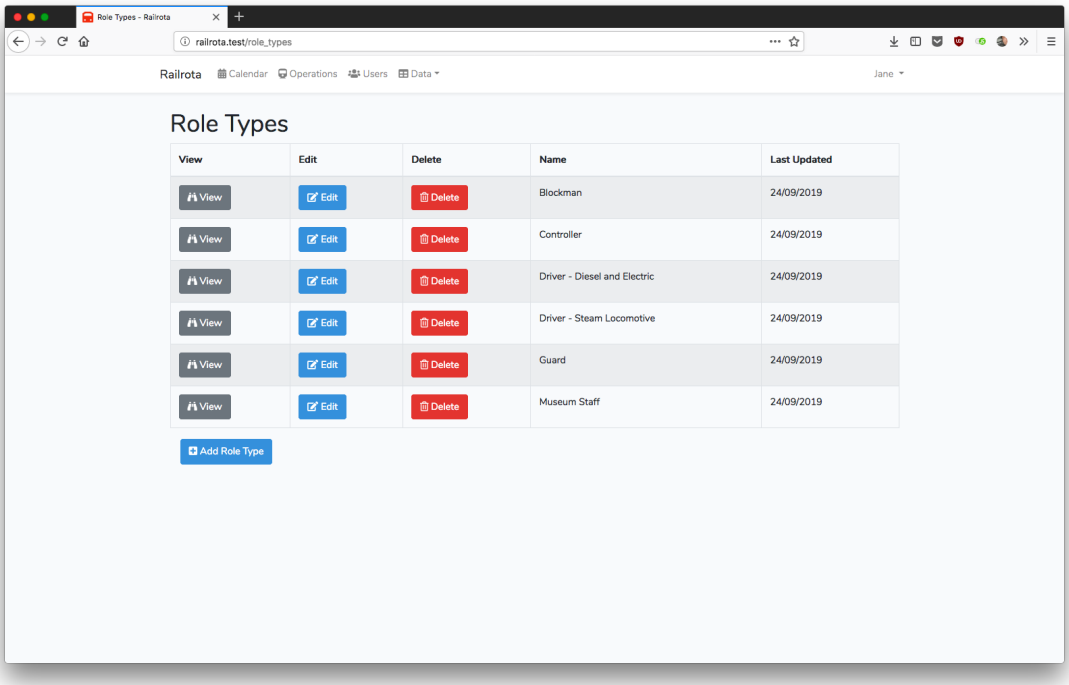


Figure 5.6: Screenshot of the Role Types interface within the application.

the role, they need to have that role themselves. Role types are treated as resources with CRUD routes, and can only be modified in any form by administrators.

Role types are not hard-coded into the system, and are their own database table, although a list of defaults is defined in the source code that can be used during initial setup, and consist of the following roles:

- Controller
- Guard
- Blockman
- Driver (Diesel/Electric)
- Driver (Steam)
- Museum Staff

Additional role types can be added to the system to accommodate any future needs of the railway, or existing ones may be modified or deleted as necessary.

5.9 Role Competencies

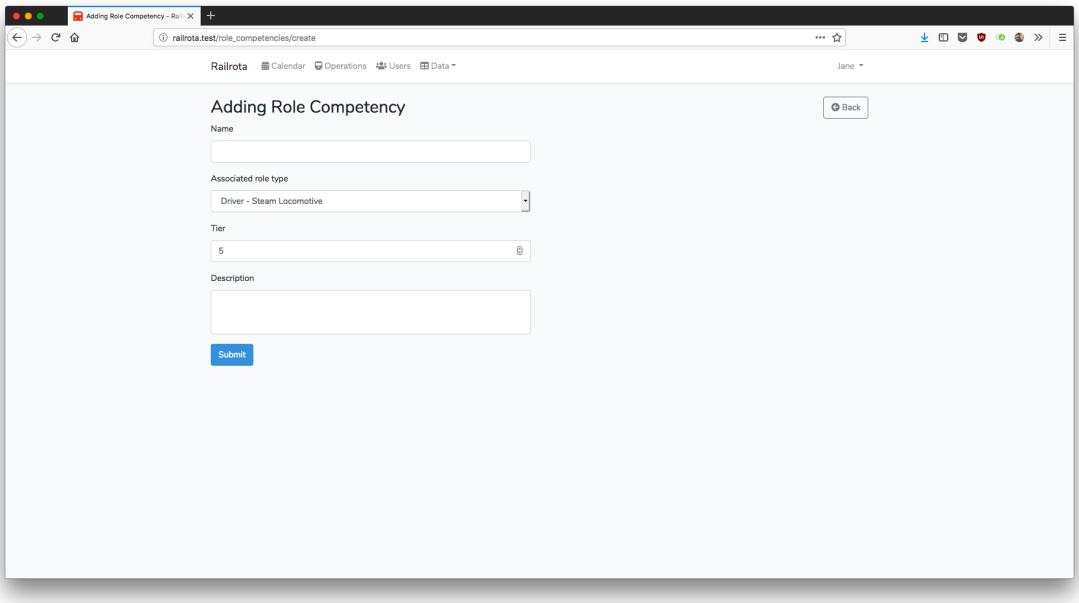


Figure 5.7: Screenshot of an administrator adding a new competency level to the a driver role.

Each role can have multiple competency levels, each with their own tier between 1 and 10. Those with a competency in a specific role can volunteer for shifts equal or lesser to their numerical tier.

For example, if there is a Blockman Shift requiring at least a Trainee, which have a numerical tier of 1, then those with Passed (2) or Examiner (3) can also sign up for the same work. If the shift required someone with Passed (2) level, then a Trainee (1) would not be able to register for it.

Each of the default role types have predefined default competencies also, but additional competencies can be added and those existing may also be modified. Furthermore, the numerical tiers are not unique, so multiple competencies for a role may share the same tier. Shifts can also not have tier requirements set, and can therefore be met by any level, including if a user does not have a competency level defined for whatever reason.

This algorithm that ensures that competency levels are met is explored in more detail in the Operations and Shift subsection.

5.10 Roles

A role type and role competency is linked to a user by means of a role - which can be viewed as properties of a user, as users can have multiple roles each with their own role type and role competency.

As they are used akin to a pivot table, tying information on the aforementioned tables together, there are no forms to directly create them, and the necessary controls to do so are only visible to an administrator on the respective user's edit UI, as shown by figure 5.5.

5.11 Locations and Locomotives

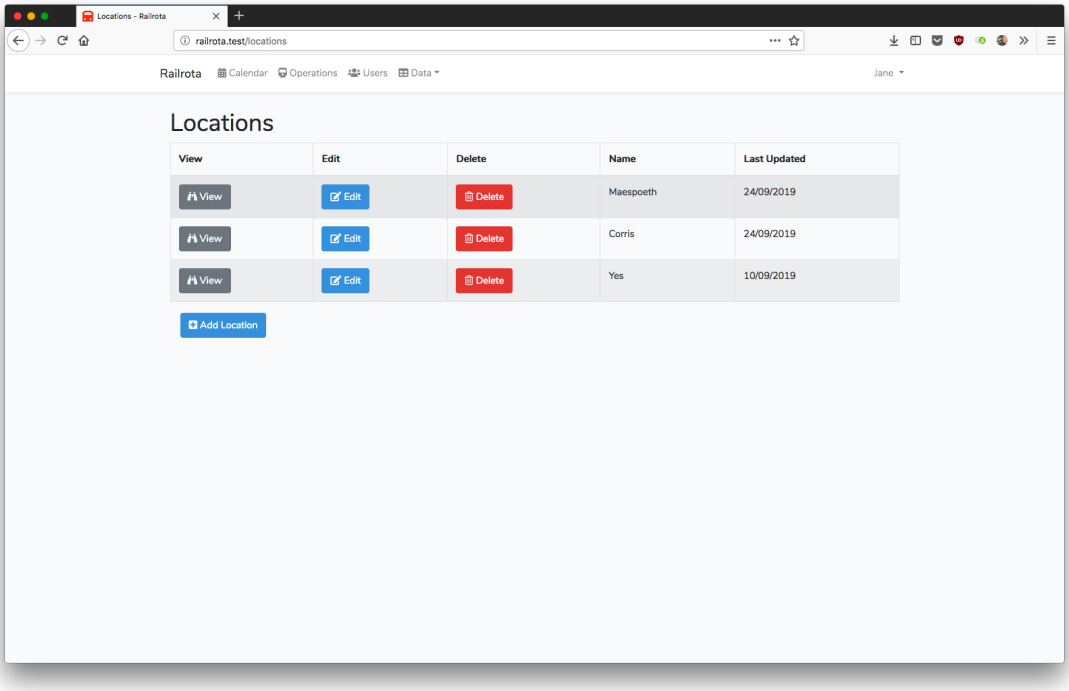


Figure 5.8: Screenshot of locations being listed within the application, which is viewable by all users.

Locations and two types of Locomotives, both Steam and Diesel/Electric are all individual tables in the database but have been grouped for the sake of brevity, as they all function in an identical way, with the only difference being that they have different default values that can be used during initial set up. The information is minimal - the name, an optional description, and timestamps.

Shifts can have optional location information, providing information to users where a shift is

located (i.e. what station or railway, or some other geographical location) and if it involves a vehicle, such as a steam locomotive, which one. These resources can only be created or manipulated by administrators, but users can view information about them in the same way they may view other resource data.

When deleting any of these nullable resources, shifts that point to affected table entries will also be nulled.

5.12 Operations and Shifts

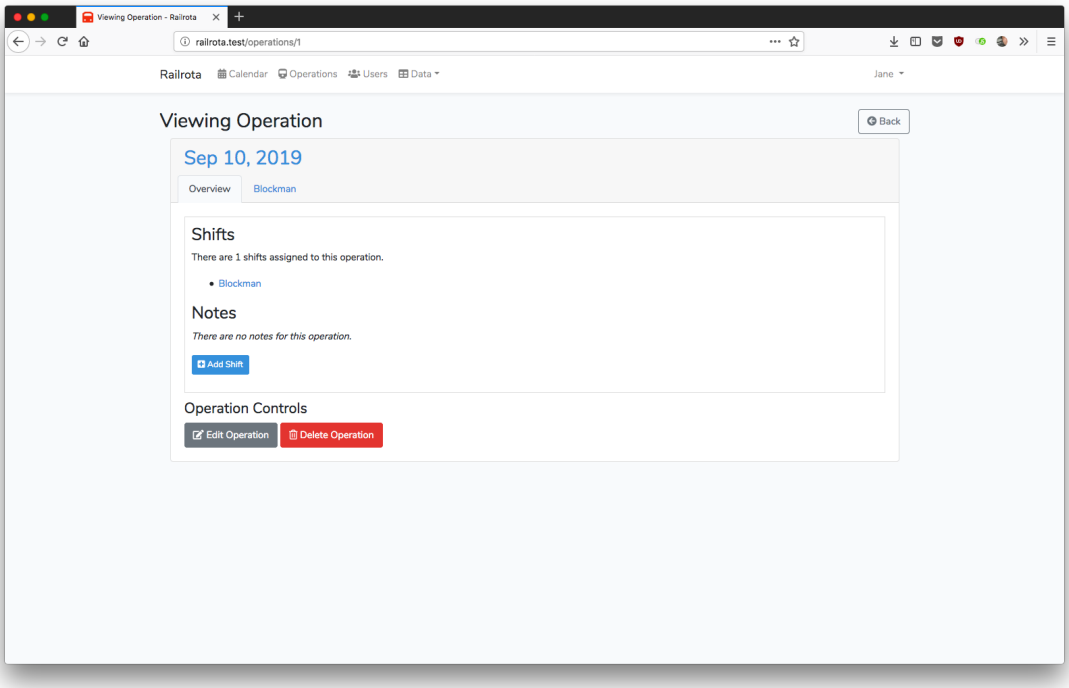


Figure 5.9: Screenshot of the operations interface, containing one shift and administrative controls.

Operations are each individual day where the railway will be operating, and each operation can have an unlimited amount of shifts. A shift is a specific volunteering vacancy, with its only requirement being the role type, i.e. what kind of work is required for that shift. Additional information such as location, locomotives, and competencies can also be specified, but are optional. Users can be manually assigned to shifts on creation, such as if an administrator knows in advance that a volunteer is to staff a certain shift, which can, with warning, override any typically enforced role requirements.

In a nutshell, an operation can be thought of as a workday, with each vacancy therein being an individual shift. This system is inspired by the existing system showed in Appendix C and addi-

tional domain knowledge examined from research into the HOPS platform. [21] If the operation is declared closed for whatever reason, such as under-staffing of critical volunteers, it is impossible for users to sign up for shifts.

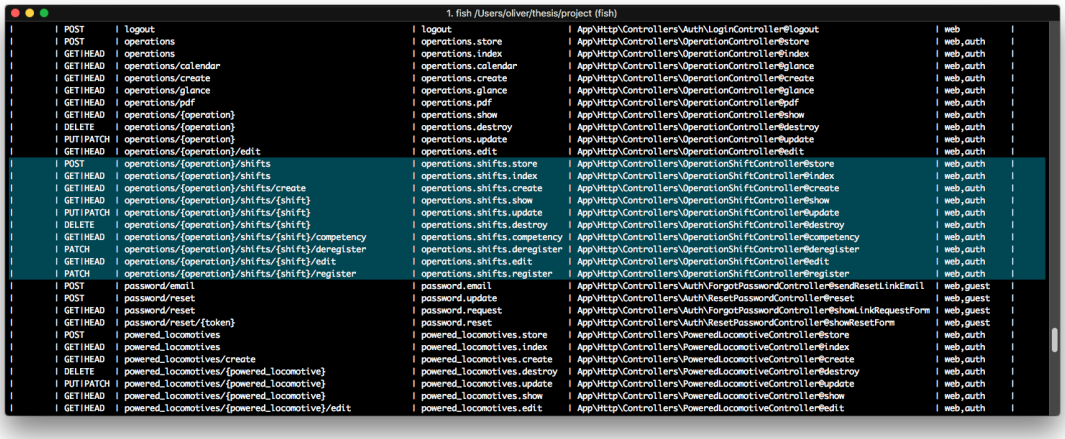


Figure 5.10: Screenshot showing how nested components (operations and shifts) appear within Laravel.

All operations can be viewed in a reverse-chronological, paginated view, or individual operations and their shifts can be viewed independently. This is one of three primary ways of viewing shift availabilities and their information, designed to be a more modern, bespoke solution for navigating data in line with early concept artwork. It is important to note that shifts are a nested resource of operation - shifts cannot be viewed individually as they are intrinsically dependent on their operation. How this is represented in Laravel is shown in figure 5.10. If an operation is deleted, shifts associated with it are also deleted to prevent the occurrence of orphaned shifts within the database.

Whilst administrators have complete control over operations and shifts and as aforementioned, can manually assign users to shifts, regular users are expected to sign up for shifts that they are capable (in terms of role and role competency) to do. This is referred to internally as 'registering', with pulling out of a shift and leaving it vacant as 'de-registering'.

Whilst de-registration is relatively straightforward and generally risk-free, registering for a shift is complex and carries out substantial amounts of validation checks to ensure that a volunteer is eligible to do so. First, the currently authenticated user is checked for administrative privileges. If they have them, they are able to bypass all further checks, as they are expected to ensure that rules are enforced themselves and not be confined by the rules of the system.

```
public function register(Operation $operation, $id)
{
    ...
```

```
            $roleCheck = Role::where([
                ['user_id', '=', Auth::id()],
                ['role_type_id', '=', $operationShift->
                    role_type_id]
            ])->first();

            if (is_null($roleCheck)) {
                flash()->error('You do not have the required
                    role type.')->important();
                return redirect()->back();
            } else {
                if (!is_null($operationShift->role_competency_id
                    )) {
                    if ($roleCheck->role_competency->tier <
                        $operationShift->role_competency->tier) {
                        flash()->error('You do not have a
                            sufficient competency / grade tier.')
                            ->important();
                        return redirect()->back();
                    }
                }
                if (!is_null($operationShift->user_id) &&
                    $operationShift->user_id !== Auth::id()) {
                    flash()->error('This shift is not vacant.')
                        ->important();
                    return redirect()->back();
                }
            }
        }
        ...
    }
```

Otherwise, the user is first retrieved from the database and checked to see if the user has a role associated with them with the required role type. If they do not, then an error is displayed and the user is redirected back. The next check ensures that, if the shift has a competency requirement specified, that the user meets that requirement by having a tier level that is not less than required, and failure is met with the same fate.

Lastly, the system checks that the shift has not already been filled and the user is erroneously

attempting to overwrite another volunteer's place in the system. While the interface would normally not allow for this, the check has been put into place to protect against malicious or accidental data submission, such as from an old or manipulated form.

With these checks complete, the remainder of the method continues executing, and data is saved to the database using Eloquent, before redirecting the user to a view and alerting them of their success.

5.13 Calendar View

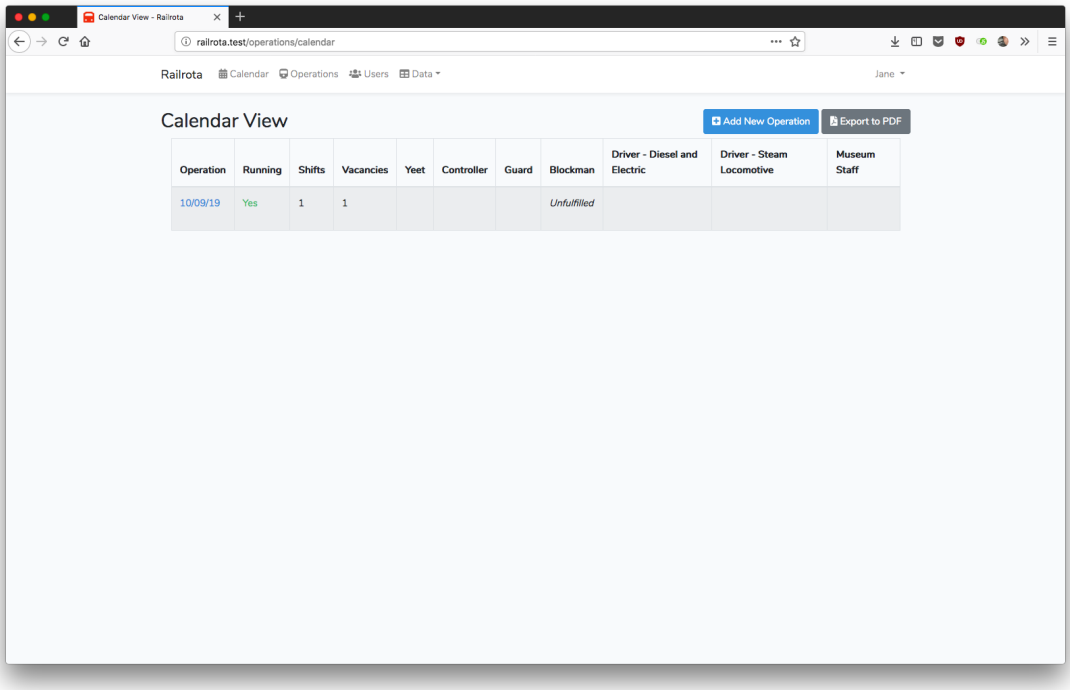


Figure 5.11: Screenshot showing the calendar view, with one operation containing a vacant shift.

Originally coined 'at a glance' view, or just 'glance' view, this functionality was intended to mimic the existing calendar used in Appendix C to make the transition to the new application easier, and to provide a straightforward means of being able to view vacancies and volunteer staffing en masse. This went through numerous iterations, following multiple rounds of feedback with the customer, before reaching its current phase, reflected in figure 5.11.

Information is spread horizontally, where each operation will contain the names, or a registration link, for vacancies of any shifts running of each type. If there are multiple shifts with the same role type, they will co-exist responsively within the same table field. Historic vacancies - vacancies that went unfilled the day following the operation are labelled as 'unfulfilled'.

5.14 PDF Export Functionality

Operations and Shifts

Operation	Running	Shifts	Vacancies	Controller	Guard	Blockman	Driver - Diesel and Electric	Driver - Steam Locomotive	Museum Staff
29/08/19	Yes	6	2	Eusebio Metz	Oliver Earl	Armand Turcotte	Vacant	Vacant	Jessie Kozey
28/08/19	No	0	0						
27/08/19	Yes	2	1	Eusebio Metz		*Unfulfilled*			

Figure 5.12: Screenshot of an exported PDF containing operation and shift data.

As the previous system was purely spreadsheet based, it was a simple task to print and distribute copies of the current timetable to volunteers for their convenience. While this is not mandatory functionality requested by the user, it was deemed important enough for inclusion in order to better bridge the gap in workflow between systems.

The system works by using a PHP library known as DOMPDF, which renders HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) of webpages into PDF documents. Getting this set up to work with the Laravel framework would have been particularly challenging, if not for a pre-existing open source Laravel wrapper called laravel-dompdf developed primarily by Dutch developer Barry vd. Heuvel. This made installation incredibly straight forward, and the wrapper provided the necessary Laravel facades and configuration modules to quickly access the library's features and integrate it into the framework. [94] [95]

A custom route was defined to generate PDF documents, which calls a pdf method in the operations controller:

```
public function pdf()
{
    $operations = Operation::orderBy('date', 'desc')->get();
    $roleTypes = RoleType::all();
    $filename = Carbon::now()->format('ymd') . '_operations.
        pdf';
```

```
        $pdf = \App::make('dompdf.wrapper');
        $pdf->loadView('pdf.operations', compact('operations', '
            roleTypes'));
        $pdf->save(storage_path() . '/' . $filename);

        return $pdf->download($filename);
    }
```

The workings of this method are rudimentary to understand, as most of the heavy lifting is done behind the scenes by the library. First, all operations are retrieved in descending order by leveraging Eloquent and stored within `$operations`, and all role types are similarly retrieved and stored within another variable. A filename string is built by using the Carbon date-time library, which is included with Laravel, and concatenating '_operations' to the end of it. This produces a filename in the format of '20190127_operations.pdf'.

With this necessary information obtained, a new instance of laravel-dompdf is instantiated, and is requested to load a particular view that utilises basic, print-appropriate layout and styling, and injecting the previously retrieved data to be parsed by Blade. Once done rendering, a copy of the generated PDF file is saved in the Laravel storage directory. Lastly, the file is presented to the user for download.

5.15 Additional Webpages

There are three additional webpages, in addition to those used by application resources, and those used for authentication (i.e. login, register).

The homepage view, which behaves as the application's actual homepage when the user has logged in, shows some convenient information on that volunteer's upcoming shifts, and their total number of shifts so far. This is different to the much more simple 'welcome' view, that is displayed to an unauthenticated user when they first navigate to the application's web address, and prompts the user to login.

Administrators also have access to an administrative panel - though this view does not provide any additional functionalities by itself per se, it does provide quick links to resources pages that an administrator might need to access when adding or changing elements of the application. It also provides a convenient link to the diagnostic tool Laravel Telescope, explored in further detail in Chapter 6.

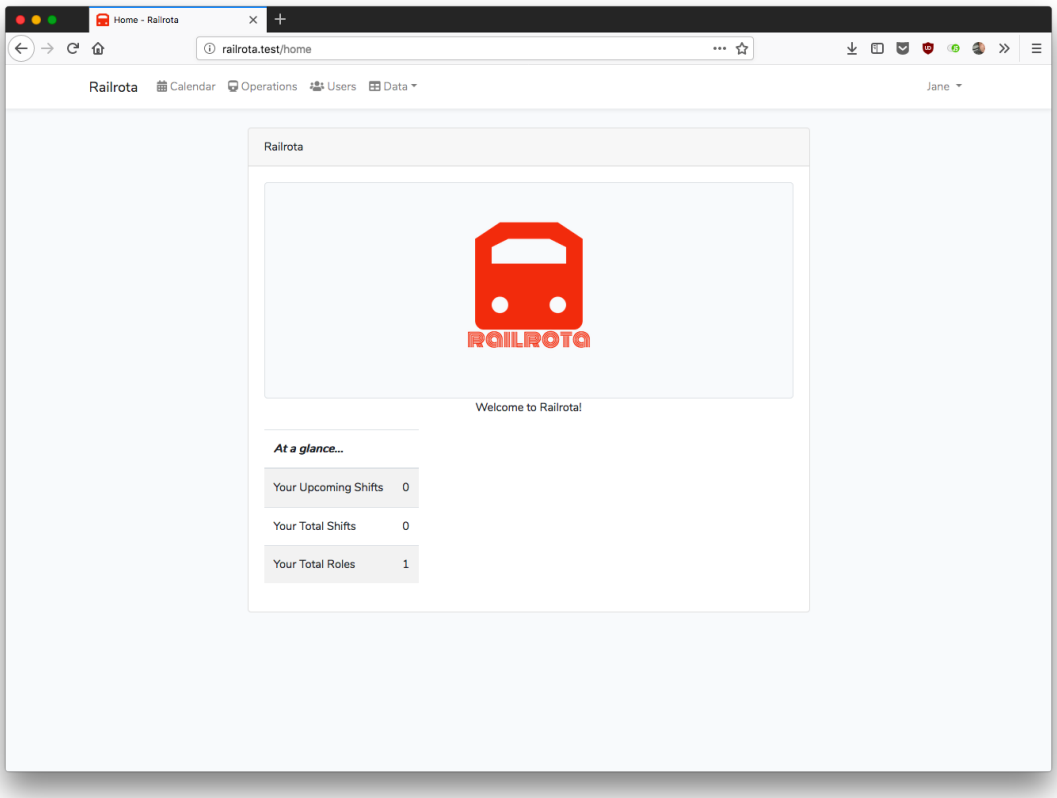


Figure 5.13: Screenshot of the homepage displayed after authentication.

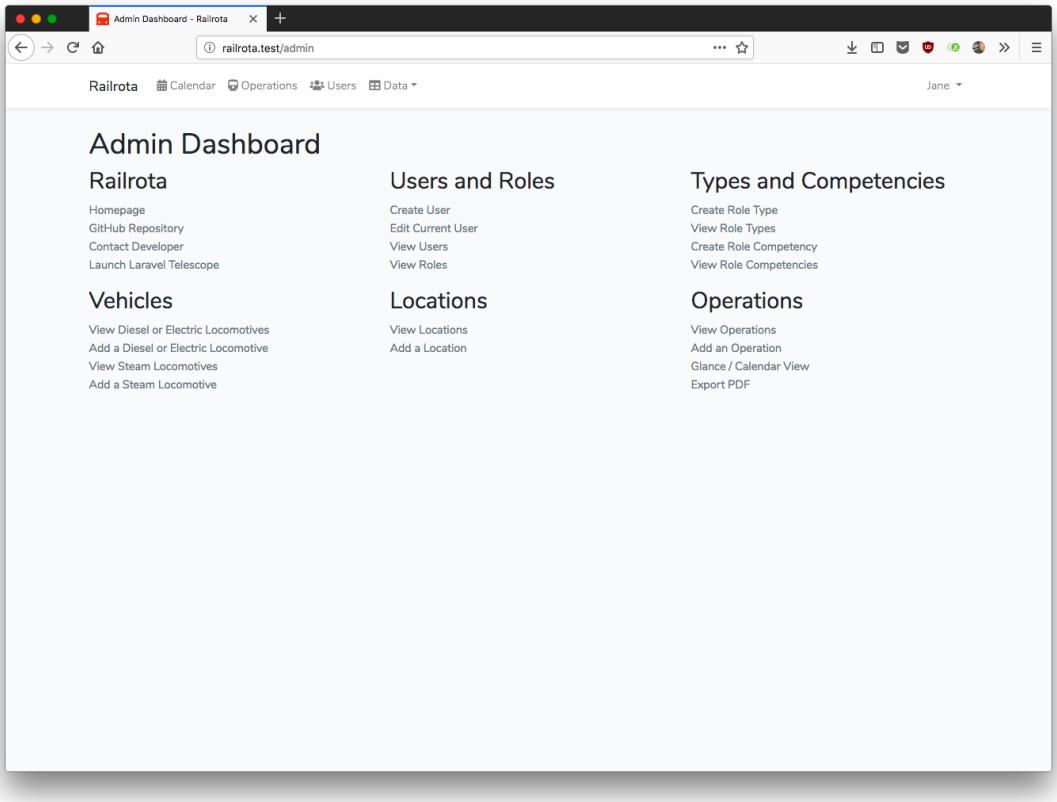


Figure 5.14: Screenshot of the administration panel, only accessible by users with administrative rights.

Chapter 6

Testing

6.1 Introduction

Software testing is an integral part of agile software development - with the use of Feature-Driven Development, it is important to know when a feature has reached the state of 'done', and this is typically once automated tests have confirmed that the functionality works as intended. [41] Radigan explains that incorporating agile testing practices, such as those used in FDD or Test-Driven Development (TDD) are a great way of minimising technical debt, and allowing for QA (Quality Assurance) staff to 'keep up with development'. [96]

The Laravel framework is reportedly built for 'testing in mind', and PHPUnit, a xUnit (akin to JUnit) testing framework for PHP developed by Sebastian Bergmann is included by default. There are also browser-facing testing tools included under Laravel Dusk, but due to time-constraints were omitted. The automated tests used by this application were therefore all written in PHPUnit. [97] [98]

6.2 Automated Testing

Two subdirectories are defined within the tests directory, one for unit tests, and the other for feature testing. All tests from both subdirectories are executed when PHPUnit is directed to begin testing. Two base classes also exist within this parent directory for subsequent tests to extend from; these files are provided by Laravel. [97]

A characteristic shared by both types of test is their use of a `setUp` method, which can set up certain parameters for testing, such as the use of a predefined volunteer user or administrator user for testing, so that they do not need to be set up with each test. `DatabaseMigrations` is also imported into the class to instruct PHPUnit to ensure that database migrations are ran before testing

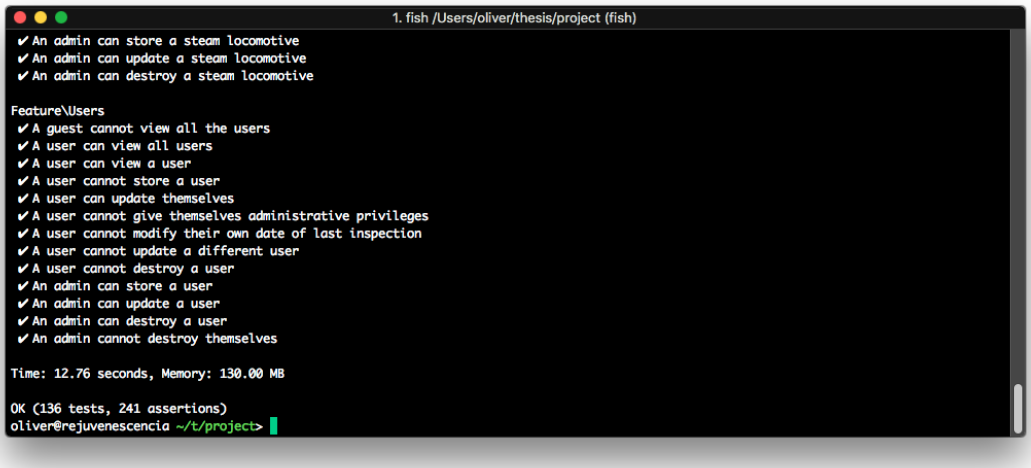


Figure 6.1: Screenshot of PHPUnit running, and successfully running all available tests.

commences, so that tests do not attempt to run on a blank dummy database, and subsequently fail as tables they depend upon, do not exist. [97]

Finally, all tests are written with method names resembling sentences, with a prefix of 'test'. This is so that when tests are run, as shown in figure 6.1, the purpose of each test is parsed into a readable sentence.

The full results of unit and feature testing can be found in Appendix G.

6.2.1 Unit Testing

Unit tests are designed to test very specific parts of the application, such as specific methods within classes. According to Hamill, a unit test should 'test a particular behaviour within the production code. Its success or failure validates a single unit of code'. [99]

An example unit test is this code taken from one of the program's unit test classes:

```
public function
    test_an_operation_shift_can_derive_its_parent_operation()
{
     $this->assertNotNull($this->getShift()->create()->
         operation);
}
```

The purpose of this test can very quickly be derived from its method name - that it checks whether a shift can derive its parent operation. As nested resources, all shifts must be attached to

some resource, and if the associated operation cannot be derived using an Eloquent method (i.e. `$shift->operation`) then that shift is likely to be orphaned, or a regression has happened somewhere that has broken the relationship defined between the classes. Unit tests exist to ensure that these regressions are picked up on as immediately as they occur, a concept known as regression testing. [100]

6.2.2 Feature Testing

In contrast to unit tests, feature tests test complex behaviour that can span across different methods. The approach here is that for each resource, every route is tested for the appropriate response in accordance with expected behaviour, combined with the use of negative testing - to ensure that behaviour that a user should not be able to is accounted for as best as possible. [101]

```
public function test_an_admin_can_store_an_operation()
{
    $this->actingAs($this->admin);
    $operation = factory('App\Operation')->make();

    $response = $this->post(route('operations.store'),
        $operation->toArray());

    $response->assertRedirect();
    $this->assertDatabaseHas('operations', [
        'is_running' => $operation->is_running,
        'notes' => $operation->notes,
    ]);
}
```

In this test, the entire *operation.store* route is tested to ensure that those with administrative privileges can create new operations. An administrator has already been built by the testing suite, so the `actingAs` method tells PHPUnit that user will be used to carry out actions and will be authenticated as such. An operation is then generated, and the test attempts to POST this to the route, as would normally happen by a HTML form. The response to this is stored within `$response` for assertions to be made against its contents.

PHPUnit is expecting the response to have redirected the user, and then proceeds to check that the database contains the operation that was posted. If this is the case, then we know that this was successful.

The reverse test equivalent to this checks that users cannot attempt to store operations of their own, like thus:

```
public function test_a_user_cannot_store_an_operation()
{
    $this->actingAs($this->user);
    $operation = factory('App\Operation')->make();

    $response = $this->post(route('operations.store'),
        $operation->toArray());

    $response->assertForbidden();
    $this->assertDatabaseMissing('operations', [
        'is_running' => $operation->is_running,
        'description' => $operation->description,
    ]);
}
```

The test works in a near identical way, with the same preliminary actions being taken, except this time a pre-generated user without admin privileges has been used. The assertions however this time, are that instead of being redirected, that the application has returned a HTTP 403 Forbidden error code, and that the entry created has not been saved to the database.

6.3 Manual Testing

Some manual testing was done to complement the automated test coverage, primarily for the authentication routes - being able to log in, log out, register, and reset a password. Though automated testing is used for the vast majority of the application behaviour, manual testing can be complementary for behaviour that is user-focused and not 'black-and-white' - Nordeen goes on to explain the importance of both in parallel: 'testers must dispel old black-and-white notions of manual testing versus automation testing. Both have their place in modern software development, and it's more a matter of knowing when and where to use each.' [102]

Manual testing tables are available on Appendix H.

6.4 Laravel Telescope

Telescope is a debug assistant, that allows authorised users to inspect recent database queries, recent exceptions, log entries, and other useful information for diagnosing problems with the application. This is incredibly useful in the development process, but can be helpful for quickly uncovering problems in production also. [103] [104]

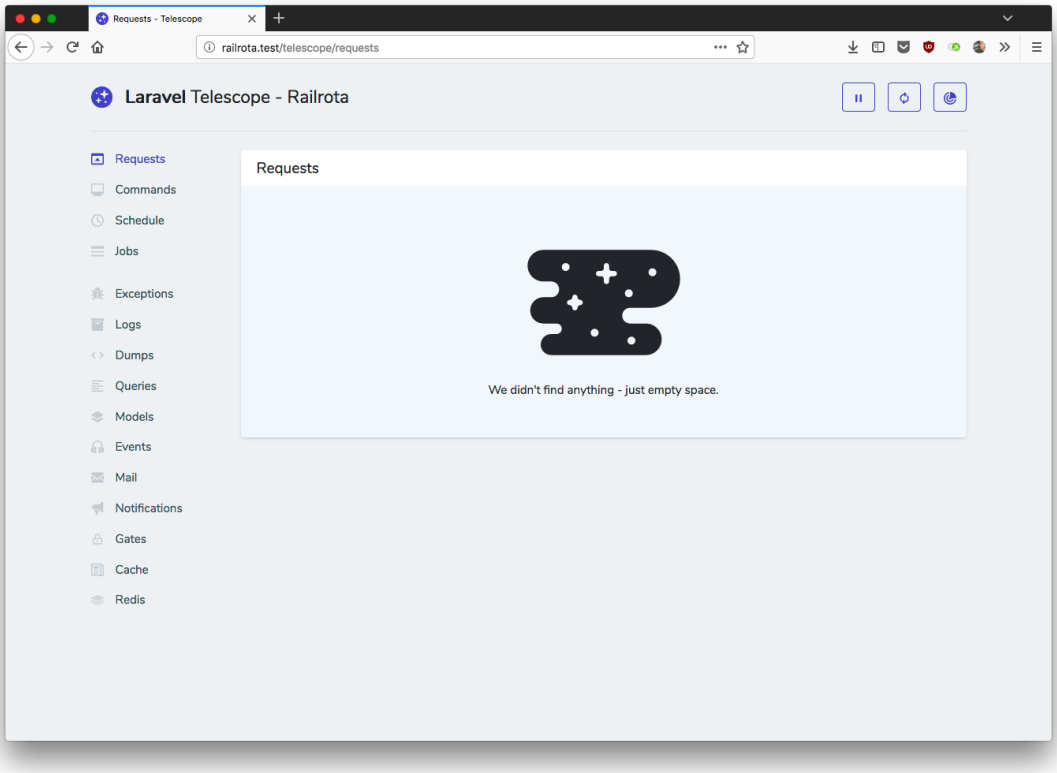


Figure 6.2: Screenshot of the Laravel Telescope interface.

It is not automatically included or installed with Laravel and must be installed manually. Admin users that are authorised to access Telescope whilst it is running in production mode can be defined by their IDs within the `.env` environment file.

This functionality was left included as it was immensely useful during the development process to simplify the often tedious and complex process of debugging PHP, and as it is protected by authentication and authorisation, there are no immediate downsides leaving it in the codebase.

Chapter 7

Deployment

7.1 Introduction

This section looks to explore some of the intricacies of the deployment process and certain elements of significance throughout, such as the use of the Composer PHP package manager, Bash scripting, and how to generate docblock documentation from PHPDoc in the Railrota source code.

7.2 Composer

Composer is a PHP package and dependency manager that is responsible for installing the required libraries and components that are specified within a `composer.json` file, and is also responsible for generating PSR-4 auto-loaders on behalf of Laravel. [105] [86]

Composer is necessary for installing Railrota, or any Laravel application, and must be installed as part of the deployment process, either globally or used locally within the directory.

7.3 Laravel Mix

Laravel Mix, as mentioned in Chapter 5, is an implementation of Webpack for compiling front-end assets, and requires Node.js to run. It is not necessarily required on the server environment, as assets can be compiled in a development environment that does feature Node.js, before being uploaded to a server. [66]

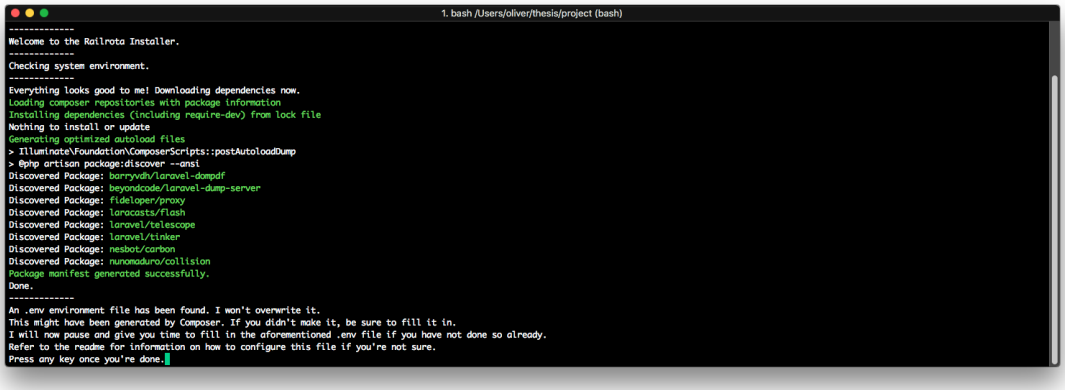


Figure 7.1: Railrota installer currently running in a Bash terminal; invoking Composer and Laravel where necessary.

7.4 Deployment Process

7.4.1 Deployment Script

A script written in Bash, `install.sh`, has been written to help automate the installation process, that will take care of ensuring the necessary programming language prerequisites are met, that Composer is installed (or will retrieve a copy of it if not available), and for invoking the appropriate tools for installing Railrota in full. It will also attempt to invoke Laravel Mix to compile front-end assets, but will bypass this if Node.js and / or a suitable JavaScript package manager, either NPM or Yarn, are not installed.

Full information on this script is available in the project `readme.md` Markdown file, or in Appendix I.

7.4.2 Manual Deployment

Full instructions on how to manually deploy Railrota can be found in the project `readme.md` Markdown file or in Appendix I, though it can be safely assumed that any prior experience in installing Laravel 5.x applications is more than sufficient, as the process is does not significantly deviate as such.

7.4.3 Generating Documentation

PHPDoc is a code commenting convention commonly used in PHP applications, and is comparable to implementations such as JavaDoc in the Java language. It has been proposed as a working

standard, but as of this time of writing remains unfinalised. [106] [107]

Full HTML-based documentation can be generated from these code comments using Sami, a discontinued (but nevertheless functional) API documentation generator written itself in PHP. Information on how to run Sami and produce a *docs* directory can be founded in the `readme.md` Markdown file or in Appendix I. The standalone tool is included in the repository and is not a library or framework. [108]

Chapter 8

Critical Evaluation

8.1 Introduction

The aims of this chapter are to provide a critical analysis and evaluation of various aspects of the development process, including providing insight into what could have been done better and the various shortcomings that could have been approached better throughout the life-cycle of the project.

8.2 Implementation of Requirements

8.2.1 Functional Requirements

The requirements that were successfully implemented for this release of the project are as follows:

- All users must be able to navigate the roster, and view current, upcoming, and/or historic shifts
- Users must be able to fill in and edit their personal information
- Users must be able to assign themselves to shifts of which they are able to do
- Users must **not** be able to assign themselves to shifts that they are unable to do
- Users must be able to indicate periods of availability
- Users must be able to operate the web application easily from a mobile device such as a smartphone
- Administrators must be able to modify user information

- Administrators must be able to assign roles and privileges that reflect a volunteer's ability and status
- Administrators must be able to add, edit, and delete shifts
- Administrators must be able to mark a shift as closed
- Administrators must be able to define a shift's staffing requirements
- The application must run on a predetermined operating environment

The behaviours that were not successfully implemented are:

- Users must **not** be able to assign themselves to simultaneous roles on the same shifts
- Users must be able to indicate their willingness or desire to do an upcoming shift (i.e. an unspecified upcoming weekend)
- Administrators must be able to email users regarding upcoming shifts

The first of these two requirements was intentionally left out after a conversation with the client, where it was expressed that it was certainly possible for a particular volunteer to be carrying out more than one role, although unlikely, during an operation. This might be the case for example where an experienced member of staff is monitoring a trainee, whilst also carrying out some miscellaneous function. Nevertheless, should the customer change their mind on this and wish to have this requirement reinstated, it could be reverted in a straightforward manner.

The next point, being able to indicate willingness or desire to do an upcoming shift was actually overlooked and, due to human error, not implemented before development was frozen. While embarrassing, an important lesson can be learned from this experience on the importance on continuously ensuring and checking that requirements are being met at all stages of the project, and that any ongoing work should be directly attributable to a requirement, such as by using user stories.

Lastly, the functionality to email users with upcoming shift information was simply not able to be implemented in time for the initial software release, purely a result of time constraints. This was communicated to the customer as a matter of urgency; reassuring them that it would be included in a future revision of software, and was met with approval.

Despite having missed three functional requirements within the first release, this success rate of 80% is nevertheless a positive one, especially given that changes and feedback were continuously exchanged throughout the development process. This should be considered a successful delivery of functionality, which ultimately is the key indicator of progress and success in an agile software project. [65]

8.2.2 Desirable Functionality

Automated rostering was considered beyond the scope of this release due to it being a complicated computational problem, and following discussion with the customer, it was deemed not to be an issue, but perhaps functionality that could be considered in a distant future release as the number of railway volunteers increases. They also stressed that they did not want optional automated rostering from potentially complicating the UI of the application, which was what initially prompted its inception in the first place, as existing solution HOPS was considered too complex for the average user to navigate.

The ability to implement localisation into the program was researched and was considered for the first release. Ultimately however, time constraints made it necessary to triage this functionality for a future release. This was discussed with the customer, as due to the high number of Welsh speakers in Machynlleth [3] this might be a greater priority than an initial analysis might reveal. The customer nevertheless provided reassurance that it was not necessary for a first release of the software, but would gladly welcome it in a future revision.

8.3 Adherence to Agile Methodology

Overall adherence to the hybrid FDD methodology was mostly successful, with the Build by Feature iteration taking up the vast majority of development time, and fitting comfortably within the predetermined time-frame. However, one of the major violations of this methodology that happened periodically was not producing unit or feature tests as components were developed, and simply moving onto the next feature or route - without a concrete indication of behaviour being done. In order to rectify this, substantial time had to be allocated to going back and ensuring adequate test coverage was in place to prevent any accidental regressions in behaviour.

Furthermore, it was difficult to remain within the confines of the time-frames used in place of Scrum sprints. With all the best planning and foresight in the world, certain real life events can occur that can be problematic obstacles for solo developers who do not have other software engineers whom they can delegate work to when problems arise. In spite of these difficulties, 80% of planned required functionality has made it into the initial release, showing a great deal of adaptability and resourcefulness on the developer's behalf, which are crucial elements in agile software development in a world where, for better or for worse, programmers do not have psychic abilities.

8.4 Difficulties Encountered with Domain Knowledge

By far the greatest difficulty with this project was wrestling with an onslaught of specific domain knowledge and terminology unique to railways, steam railways, and the work carried out thereon. Whilst the customer made every effort possible to ensure a wealth of documentation and terminology guidelines were available, it remained nevertheless difficult to fully assimilate without slowing down development efforts. Some problems did arise from confusion regarding specific terminology at multiple points in development, but thankfully due to regular meetings with the client these problems were diagnosed early and could be quickly rectified with further clarification and insight from the customer.

This truly demonstrates the importance of face-to-face meetings, or in this instance, oral conversation, where information can far more readily be transferred from customer to developer without the need of lengthy or confusing documentation. This theory is backed up by Chow et al who identified customer involvement as a strong indicator for success in agile projects. [109]

In a nutshell, when faced with a project that requires a substantial deal of domain knowledge or expertise, it is best to communicate with those with that expertise, i.e. the customer, as much as possible so that any issues can be quickly rectified and that large portions of the codebase need not be re-engineered.

8.5 Suitability of Development Choices

The use of a framework like Laravel drastically reduced the amount of boilerplate code that would normally need to be written for an application. Not a single line of SQL (Structured Query Language) was even needed to be written for the program due to the amount of heavy lifting that the Eloquent ORM is capable of doing. For these reasons especially, the choice of Laravel for a development framework (and subsequently PHP as a language) was a fitting and appropriate choice. After speaking informally with a colleague who was embarking on a similar but unrelated endeavour using vanilla PHP, it became apparent just how much database code that they had to write, which could be replicated with a simple Eloquent method under Laravel.

It is believed however that the use of Python and Django together would have been equally as sufficient as the offerings of said framework are similar, but this would have involved significantly more learning, and with some of the real life obstacles that slowed down development, this could have caused further unnecessary delays. It can be concluded that choosing a fully featured framework was the correct choice, instead of potentially needing to 'reinvent the wheel' by either using a more cut down framework like Express, or using no framework at all. This is really evident when one evaluates how much code and checks must be put into place for a comprehensive and

secure authentication and authorisation system to work, but with Laravel, one only needs to put into place policies and call the appropriate method. [92]

8.6 Improvements to Testing Methodology

Whilst the automated and manual test coverage is great and covers almost all Railrota functionality, one crucially missing element from the testing is ensuring that the application is easy to use for the customer - as the customer, or any other prospective volunteer, has not had the opportunity to test-run the software, besides being shown video demos during meetings.

A major improvement to the testing methodology that should be rectified in the future, would be to host regular meetings with the customer, and preferably prospective users, and allow them to operate the software freely and without prompt - recording their behaviour and feedback to make improvements to the user experience. (UX)

8.7 Shortcomings and Recommendations for Further Releases

Whilst the software is generally good quality expected of a Master's level software engineer, it is not without flaws indicative of someone learning a web framework for the first time.

One of the most glaring code smells that would be come immediately obvious to any experienced developer is the re-usage of code, both within controllers and within Laravel Blade templates. Although effort has been undertaken to ensure that code is compartmentalised as much as possible so that it is not reused, particularly in controllers, there is a lot of code that simply gets repeated. Moving some of this shared code into shared methods, or simply redesigning the algorithms into more concise functions would help clean up a lot of this underlying programming.

Furthermore, while much of the web interface has been abstracted into partial templates for re-usage, there are large portions of Blade logic that are repeated, and more effort could be made to reduce their recurrence so that if something needs to be changed, it can be changed once, rather than in multiple positions. Due to time constraints during development, the decision of 'function over form' had to be undertaken, which is reportedly common in many agile projects. However, for future revisions, developers should look to undo these code smells and make the code more maintainable for future work.

Chapter 9

Conclusion

It is without a fraction of a doubt that the project can be considered a tremendous success. Initial feedback from the customer has been immensely positive, in particular thanks in part to being able to take criticism and requests from a meeting and reproducing them in code for the very next week. Combining a unique approach in the operations view, along with a more traditional calendar and PDF exportation capability was absolutely the correct approach and allows for an immense quantity of flexibility in how a user interacts with the software.

If it were possible to start the project from scratch, more time would be dedicated to the initial research into the domain area and to more thoroughly research and discuss keywords and terminology where there are any doubts or areas of interpretation, to ensure that the customer and developer remain on the same page at all times. In addition, taking the additional effort to adhere to a strict agile methodology, such as Extreme Programming (XP) would likely have proved a better solution than making adaptations and producing a hybrid solution, as having a concrete methodology to fall back on when issues arise would have helped better ensure that timelines are adhered to, and that development remains disciplined at all times - ensuring that work is marked as 'done' and tests are produced to prohibit regressions without having to go back and produce tests at a later stage.

Furthermore, despite it being a strength of the project, more communication with the customer would have been a fantastic addition to the workflow, in particular setting up live demos or sending demo versions of the software for users to play with and provide feedback on - to better find what works, and what does not.

If it were possible to continue the project for another month, as previously stated, improving code quality and maintainability would be the first priority.

If the developer were in charge of their own grading, it would be considered at least high Merit level because:

- The software works tremendously well and fulfills 80% of required functionality, is easy to use, and fulfills the criteria to be a replacement for the customer's current spreadsheet based system
- Whilst unique and hybridised, a great deal of effort was put into researching appropriate agile methodologies, and adhering to it.
- A wealth of research was carried out into alternatives existing in this already niche field, whilst touching on an important and historic computational problem in the Nurse Scheduling Problem. (NST)
- This project is ready for the real-world and can be used from day #1 by Corris Railway, which was one of the motivations for undertaking the project in the first place.
- An immense amount of learning has been facilitated from this project undertaking - having learned a great deal about agile software development, the Laravel framework, PHP, and indeed, steam railways and how they operate.

In conclusion, this project has been an immense pleasure to implement and has as aforementioned, provided an unparalleled learning experience that has reflected much that has been learned throughout the year. The end result is a CRUD-compliant, full-stack web application with a full testing suite that uses modern web technologies and conforms to PSR PHP web standards, ready for a life time of success and continued development in the hands of the Corris Railway, located in beautiful Machynlleth.

Appendices

Appendix A

Third-Party Code and Libraries

Please note: There are libraries and frameworks included with Laravel, such as Vue.js and Axios, but they are unused. Only libraries and frameworks listed here have been used in Railrota. They are however, compatible with Laravel's MIT licensing.

1.1 Laravel Framework

Laravel Framework - Laravel is the main PHP web framework that empowers the application. This includes features commonly used by the Railrota application, such as the Eloquent ORM for working with the database, the Carbon date-time library, the Laravel Blade templating engine, the Artisan command line tool, and Laravel Mix for compiling front-end assets. The framework is released using the MIT license. This framework was used without modification.

https://github.com/laravel/laravel

1.2 Laravel Telescope

Laravel Telescope is an optional library for Laravel to aid with debugging. The library is released using the MIT license. This library was used without modification.

https://github.com/laravel/telescope

1.3 dompdf-laravel

DOMPDF Wrapper for Laravel - HTML/CSS rendering engine used for converting webpages into PDF pages. Laravel wrapper for the DOMPDF library. The library is released using the MIT

license. This library was used without modification.

https://github.com/barryvdh/laravel-dompdf

1.4 Sami

Sami API Documentation Generator - A standalone tool used for compiling PHPDoc / docblock comments into full documentation in HTML form. The tool is released using the MIT license. This tool was used without modification.

https://github.com/FriendsOfPHP/Sami

1.5 PHPUnit

PHPUnit - unit testing suite included with Laravel, but as it is used in development and not in production, included as a separate entry here. The framework is released under the BSD-3 license. This framework was used without modification.

https://github.com/sebastianbergmann/phpunit

1.6 Bootstrap 4

Bootstrap - HTML/CSS and JavaScript framework for responsive web design. This framework was released under the MIT license. This framework was used without modification.

https://github.com/twbs/bootstrap

1.7 jQuery

jQuery - JavaScript library for DOM manipulation and used by Bootstrap. This library was released under the MIT license. This framework was used without modification.

https://github.com/jquery/jquery

1.8 Font Awesome

Font Awesome - CSS framework used for inserting icons into front-end UIs. This framework's code is licensed under the MIT license, with the icons being released under the CC BY 4.0 Li-

cense. This framework is used without modification. More information can be found under the framework's LICENSE file.

https://github.com/FortAwesome/Font-Awesome

Appendix B

Ethics Submission

This appendix includes a copy of the ethics submission for the project, including Ethics application number.

30/07/2019

For your information, please find below a copy of your recently completed online ethics assessment

Next steps

Please refer to the email accompanying this attachment for details on the correct ethical approval route for this project. You should also review the content below for any ethical issues which have been flagged for your attention

Staff research - if you have completed this assessment for a grant application, you are not required to obtain approval until you have received confirmation that the grant has been awarded.

Please remember that collection must not commence until approval has been confirmed.

In case of any further queries, please visit www.aber.ac.uk/ethics or contact ethics@aber.ac.uk quoting reference number **13424**.

Assesment Details

AU Status
Undergraduate or PG Taught

Your aber.ac.uk email address
ole4@aber.ac.uk

Full Name
Oliver Roy Thomas Earl

Please enter the name of the person responsible for reviewing your assessment.
Richard Shipman

Please enter the aber.ac.uk email address of the person responsible for reviewing your assessment
rcs@aber.ac.uk

Supervisor or Institute Director of Research Department

cs

Module code (Only enter if you have been asked to do so)
CHM9360

Proposed Study Title
Web-Based Rostering System for Volunteer Worked Organisations

Proposed Start Date
01/07/2019

Proposed Completion Date
27/09/2019

Are you conducting a quantitative or qualitative research project?
Mixed Methods

Does your research require external ethical approval under the Health Research Authority?
No

Does your research involve animals?
No

Are you completing this form for your own research?
Yes

Does your research involve human participants?
No

Institute
IMPACS

Please provide a brief summary of your project (150 word max)
This project is designed to facilitate the rostering of volunteer staff for the Corris Railway in Machynlleth. It is designed to be web-based, and straightforward to use by all expertise levels. The main problem encountered and the need for research is overcoming the NP-hard Nurse Scheduling Problem (NSP).

Where appropriate, do you have consent for the publication, reproduction or use of any unpublished material?
Not applicable

Will appropriate measures be put in place for the secure and confidential storage of data?
Yes

Does the research pose more than minimal and predictable risk to the researcher?
No

Will you be travelling, as a foreign national, in to any areas that the UK Foreign and Commonwealth Office advise against travel to?
No

Please include any further relevant information for this section here:
Not applicable.

If you are to be working alone with vulnerable people or children, you may need a DBS (CRB) check. Tick to confirm that you will ensure you comply with this

requirement should you identify that you require one.
Yes

Declaration: Please tick to confirm that you have completed this form to the best of your knowledge and that you will inform your department should the proposal significantly change.
Yes

Please include any further relevant information for this section here:
Not applicable.

Appendix C

Example Timetable

The following is an example rota that was produced manually prior to the development of this rostering software.

2019 Museum & Crew Roster

Last Update 17/06/19 20:26	Comments	Museum	Maespoeth Controller & Assistant	No.7 Driver	No.7 Fireman	Guard
March	*Appendix C*					*Example Timetable*
Sun 17th Mar	***David Williams Photo Charter 8:30am to 2:30pm (NO PUBLIC SERVICE)***	**N/A**	David Coleman	**Patrick Jolley**	Dave Ingram	Steve Parr
April						
Sat 6th Apr	***Operational Training Weekend***	**NO PUBLIC SERVICES**				
Sun 7th Apr						
Mon 8th Apr	***Ffestiniog Travel - 9:15am (13 in party) (NO PUBLIC SERVICE)***		Graeme Jolley	Bob Winchurch	David Coleman	Dick Kidner
Wed 10th Apr	**RAILWAY FOR THE DAY (Andrew Hobden) 9:30am start (Steam) *(NO PUBLIC SERVICE)***	**N/A**	Richard Shipman	David Coleman		**N/A**
Fri 19th Apr	**(Good Friday) Cardiff & Avonside Railway Society. 14 in party. 1:00pm service. Pay on the day. Normal fares.**	David & Carol Roberts	David Coleman	Richard Foyn	Sam Knappett	Tony Pritchett
Sat 20th Apr		David & Carol Roberts	Richard Shipman/ (Richard Foyn)	Bob Winchurch	Ben Brotherwood	Andrew Pugh
Sun 21st Apr	**(Easter Sunday)**	Kate & Jane Arnold	Bill McDonald/ (Andrew Pugh)	Trefor Davies	Steve Poynter	John Arnold
Mon 22nd Apr	**(Easter Monday)**	Janice & Charles	Graeme Jolley/ (Richard Foyn)	Trefor Davies	Steve Poynter	John Arnold
May						
Sat 4th May	**Mach Comedy Festival at 9:45 (up to 40 in party)**	David & Carol Roberts	Helen Starksfield	Trefor Davies/ No. 6 (Rod Starksfield)	Jack Evans	Sam Knappett
Sun 5th May		Richard Greenhough	Helen Starksfield	Richard Foyn	Jack Evans	Rod Starksfield
Mon 6th May		Sue & Janice	Helen Starksfield (AM)/ Graeme Jolley (PM)	Bob Winchurch	Jack Evans	Rod Starksfield
Sun 12th May		Sue & Janice	Bill McDonald	Andrew Rawlins	Dave Ingram	Rod Starksfield
Sun 19th May		David & Carol Roberts	Bill McDonald	Trefor Davies	Ben Brotherwood	Sam Knappett
Mon 20th May	**RAILWAY FOR THE DAY (Robert Watson) 9:30am start (Steam) *(NO PUBLIC SERVICE)***	**N/A**	Richard Shipman	Bob Winchurch	Patrick Jolley	**N/A**
Sat 25th May		Graeme & Alf	Helen Starksfield/ (Andrew Pugh)	Bob Winchurch	Ben Brotherwood	Tom Gupta
Sun 26th May	**SPECIAL GALA DAY**	Janice & Amanda	Richard Shipman	Andrew Rawlins/ Trefor Davies/ Helen Starksfield (6)/ Richard Foyn	Sam Knappett	Tom Gupta/ Andrew Pugh/ Steve Parr/ Rod Starksfield
Mon 27th May		Graeme/ John & Jane Arnold	Helen Starksfield/ (Richard Foyn)	Andrew Rawlins	Ben Brotherwood	Tom Gupta/ Graeme Jolley
June						
Sat 1st Jun			David Coleman	Trefor Davies	Dave Ingram	Patrick Jolley
Sun 2nd Jun	**HF Holidays - 4:00pm (24 in party) (£4.80pp - Pay on the day)**	Charles & Janice	Richard Shipman	David Coleman	Steve Poynter	Tony Pritchett
Thu 6th Jun	**Corris Mine Tours 4:00pm (38 in party Grp1) (NO PUBLIC SERVICE)**		Richard Shipman	David Coleman **(No. 6)**	N/A	Dick Kidner
Sun 9th Jun		David & Carol Roberts	Bill McDonald	Mike Green	Jack Evans	Richard Shipman
Tue 11th Jun	**Corris Mine Tours 4:00pm (38 in party Grp2) (NO PUBLIC SERVICE)**		Graeme Jolley	Patrick Jolley **(No. 11)**	N/A	Dick Kidner
Fri 14th Jun	**RAILWAY FOR THE DAY (Nick Moody) 9:30am start (Steam) *(NO PUBLIC SERVICE)***	**N/A**	Graeme Jolley	Bob Winchurch	David Coleman	**N/A**
Sun 16th Jun	**Father's Day**	Sue & Janice	David Coleman	Andrew Rawlins	Trefor Davies	Sam Knappett
Sun 23rd Jun		Richard Greenhough	David Coleman	Richard Foyn	Sam Knappett	Tony Pritchett
Fri 28th Jun	**RACE THE TRAIN First race @ 4:00pm**	David Roberts (?)	Graeme Jolley	David Coleman	Jack Evans (?)	
Sun 30th Jun		David & Carol Roberts		Trefor Davies		

Cont'd

2019 Museum & Crew Roster (Cont'd)

Last Update 17/06/19 20:26	Comments	Museum	Maespoeth Controller & Assistant	No.7 Driver	No.7 Fireman	Guard
July						
Sun 7th Jul			David Coleman	Patrick Jolley	Trefor Davies	Tom Gupta
Sat 13th Jul			David Coleman	Andrew Rawlins	Dave Ingram	
Sun 14th Jul				Andrew Rawlins	Dave Ingram	
Sat 20th Jul		Graeme	Bill McDonald	Patrick Jolley	Sam Knappett	Andrew Pugh
Sun 21st Jul			Andrew Pugh			
Sat 27th Jul		Graeme			Ben Brotherwood	
Sun 28th Jul			Richard Shipman	David Coleman	Trefor Davies	
August						
Sat 3rd Aug	**WIZARDS & DRAGONS WEEKEND**	Sue & Janice	Helen Starksfield	Andrew Rawlins	Steve Poynter	Tom Gupta/ Rod Starksfield
Sun 4th Aug		Sue & Janice	Helen Starksfield	Andrew Rawlins	Steve Poynter	Tom Gupta/ Rod Starksfield
Mon 5th Aug				/ No. 6 (Rod Starksfield)		Helen Starksfield
Tue 6th Aug						
Sat 10th Aug			Bill McDonald	Richard Foyn	Dave Ingram	
Sun 11th Aug			Bill McDonald	Patrick Jolley	Dave Ingram	Amanda Jolley
Mon 12th Aug				Richard Foyn		
Tue 13th Aug						
Sat 17th Aug				Andrew Rawlins	Ben Brotherwood	Sam Knappett
Sun 18th Aug				Andrew Rawlins		
Mon 19th Aug				Andrew Rawlins		
Tue 20th Aug						
Sat 24th Aug	**MODEL RAILWAY EXHIBITIONMODEL RAILWAY EXHIBITION**			Richard Foyn	Trefor Davies	Tom Gupta/ Andrew Pugh
Sun 25th Aug			Andrew Pugh	Trefor Davies	Steve Poynter	Tom Gupta
Mon 26th Aug					Sam Knappett	Tom Gupta
Tue 27th Aug					Sam Knappett	Andrew Pugh
September						
Sun 1st Sep						
Sun 8th Sep			Bill McDonald	Andrew Rawlins		
Sat 14th Sep	**RAILWAY FOR THE DAY (Alan Winwood) 9:30am start (Steam) *(NO PUBLIC SERVICE)***	**N/A**				**N/A**
Sun 15th Sep						
Sun 22nd Sep						
Sat 28th Sep	**RAILWAY FOR THE DAY (Michael Huckfield) 9:30am start (Steam) *(NO PUBLIC SERVICE)***	**N/A**			Ben Brotherwood	**N/A**
Sun 29th Sep						
October						
Sun 6th Oct			Bill McDonald	Andrew Rawlins	Trefor Davies	
Sun 13th Oct						
Sat 19th Oct	**AGM/Diesel Only (No 4:00pm service)**	Sue				
Sun 20th Oct				Trefor Davies		
Sat 26th Oct	**HALLOWEEN SPECIAL**	Sue & Janice		Andrew Rawlins	Jack Evans	Rod Starksfield
Sun 27th Oct				Andrew Rawlins	Jack Evans	Rod Starksfield
December						
Sat 7th Dec	**SANTA SPECIALS WEEKEND (Rod Starksfield is Santa)**	Sue & Charles	Helen Starksfield	Trefor Davies	Andrew Rawlins	Tom Gupta
Sun 8th Dec		Sue & Charles	Helen Starksfield	Richard Foyn	Sam Knappett	Tom Gupta

Timetable: Operational Timetable No.7 (or No.7F).
NOTES: Control is Maespoeth Blockman, in the Signal Box on internal phone number 223.
Staff in (brackets) are trainees.

Appendix D

Gantt Chart

The Gantt chart that was used throughout the project to define the various stages of development, specify start and end dates for each phase, and to help provide context on the time-frame of the undertaking.

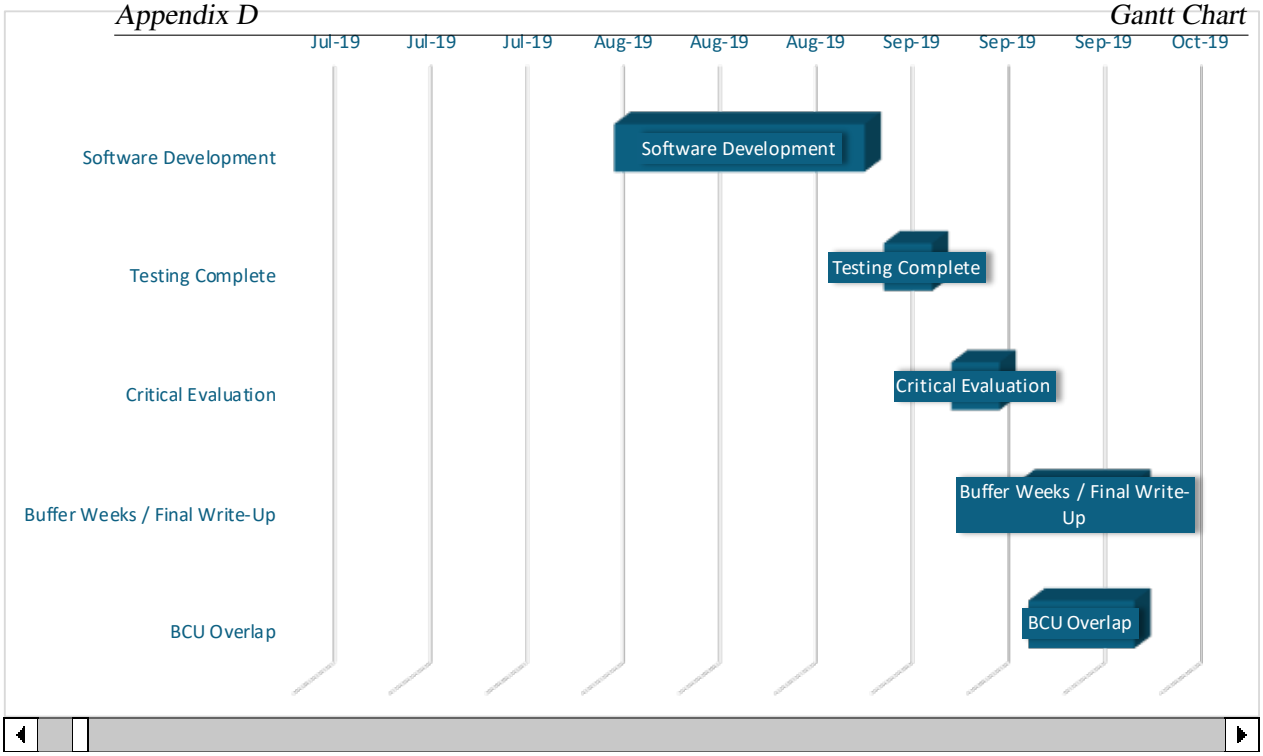
Gantt Chart

Start date:	01/07/2019
End date:	27/09/2019

Enter a sequential set of numbers in the column below.	*Enter the start date for the milestone or activity in the column below.*	*Enter the end date for the milestone or activity in the column below*	*Enter the milestone and/or activity description in the column below. This description will appear in the Project Chart.*
Position	**Start Date**	**End Date**	**Milestone/Activity**
1	01/07/2019	12/07/2019	Project Scaffolding / Outline
2	15/07/2019	19/07/2019	Background, Aims, Objectives
3	22/07/2019	02/08/2019	Literature Review / Analysis
4	05/08/2019	30/08/2019	Software Development
5	02/09/2019	06/09/2019	Testing Complete
6	09/09/2019	13/09/2019	Critical Evaluation
7	16/09/2019	27/09/2019	Buffer Weeks / Final Write-Up
8	17/09/2019	27/09/2019	BCU Overlap

To add more Milestones/Activities, insert new rows above this line

Appendix D *Gantt Chart*



Appendix E

Operating Environment

The following information was obtained in relation to the operating environment - a server located in the city of Amsterdam, Netherlands.

```
rcs@silo:~$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 77
model name      : Intel(R) Atom(TM) CPU  C2350  @ 1.74GHz
stepping        : 8
microcode       : 0x127
cpu MHz         : 1745.626
cache size      : 1024 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 2
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 11
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
    pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss
    ht tm pbe syscall nx rdtscp lm constant_tsc arch_perfmon pebs
```

```
     bts rep_good nopl xtopology nonstop_tsc aperfmperf pni
    pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr
    pdcm sse4_1 sse4_2 movbe popcnt tsc_deadline_timer aes rdrand
     lahf_lm 3dnowprefetch ida arat epb dtherm tpr_shadow vnmi
    flexpriority ept vpid tsc_adjust smep erms
bogomips    : 3491.25
clflush size    : 64
cache_alignment    : 64
address sizes    : 36 bits physical, 48 bits virtual
power management:

processor    : 1
vendor_id    : GenuineIntel
cpu family    : 6
model        : 77
model name    : Intel(R) Atom(TM) CPU  C2350  @ 1.74GHz
stepping    : 8
microcode    : 0x127
cpu MHz        : 1745.626
cache size    : 1024 KB
physical id    : 0
siblings    : 2
core id        : 1
cpu cores    : 2
apicid        : 2
initial apicid    : 2
fpu        : yes
fpu_exception    : yes
cpuid level    : 11
wp        : yes
flags        : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
    pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss
    ht tm pbe syscall nx rdtscp lm constant_tsc arch_perfmon pebs
     bts rep_good nopl xtopology nonstop_tsc aperfmperf pni
    pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr
    pdcm sse4_1 sse4_2 movbe popcnt tsc_deadline_timer aes rdrand
     lahf_lm 3dnowprefetch ida arat epb dtherm tpr_shadow vnmi
    flexpriority ept vpid tsc_adjust smep erms
```

```
bogomips        : 3491.25
clflush size    : 64
cache_alignment     : 64
address sizes   : 36 bits physical, 48 bits virtual
power management:

rcs@silo:~$ cat /proc/meminfo
MemTotal:        4022480 kB
MemFree:          146956 kB
MemAvailable:    2274316 kB
Buffers:          298592 kB
Cached:          1919940 kB
SwapCached:        19680 kB
Active:          2073608 kB
Inactive:        1386788 kB
Active(anon):    1027160 kB
Inactive(anon):   356332 kB
Active(file):    1046448 kB
Inactive(file):  1030456 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:       1074172 kB
SwapFree:         743280 kB
Dirty:                 0 kB
Writeback:             0 kB
AnonPages:       1225512 kB
Mapped:           111816 kB
Shmem:            141628 kB
Slab:             361412 kB
SReclaimable:     303860 kB
SUnreclaim:        57552 kB
KernelStack:        4368 kB
PageTables:        18012 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     3085412 kB
Committed_AS:    3204708 kB
```

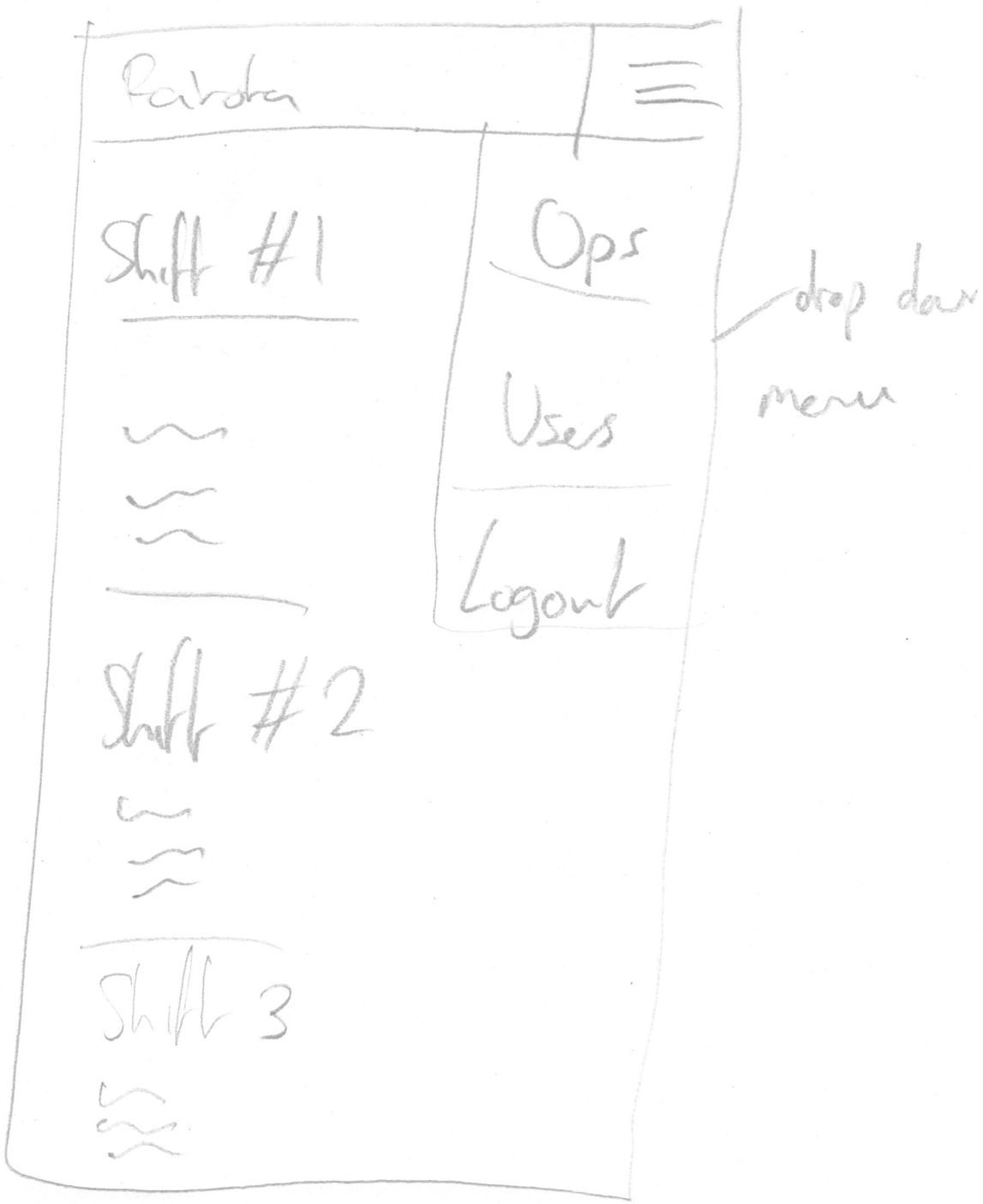
```
VmallocTotal:   34359738367 kB
VmallocUsed:      279104 kB
VmallocChunk:   34359401400 kB
HardwareCorrupted:     0 kB
AnonHugePages:         0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
DirectMap4k:       91136 kB
DirectMap2M:     4065280 kB
```

Appendix F

Concept Art

The following are hand-drawn early artwork concepts before being expanded upon digitally, as part of the initial design and conceptualisation process.

Operations
User
Oper
Users
Operation #1 Date:
Shift #1 User: A
Shift #2 User: B
Multiple on one page


Railrota Mobile Layout Concept
Railrota
Shift #1
Ops
Uses
Logout
drop down menu
Shift #2
Shift 3

Appendix G

Automated Test Results

The following pages are HTML pages generated by PHPUnit.

Unit\Locations

✓ An array of default locations can be returned
✓ A location can retrieve its associated operation shifts

Unit\OperationShifts

✓ An operation shift can derive its parent operation
✓ An operation shift can derive its role type
✓ An operation shift can derive its assigned user when it is occupied
✓ An operation shift cannot derive its assigned user when it is vacant
✓ An operation shift can derive its role competency
✓ An operation shift cannot derive its role competency when it is unspecified
✓ An operation shift can derive its location
✓ An operation shift cannot derive its location when it is unspecified
✓ An operation shift can derive its steam locomotive

✓ An operation shift can not derive its steam locomotive when it is unspecified
✓ An operation shift can derive its powered locomotive
✓ An operation shift can not derive its powered locomotive when it is unspecified
✓ An operation shift can get all data pertaining to its children

Unit\Operations

✓ An operation can retrieve its associated operation shifts
✓ An operation creates a pdf file in laravel storage when the calendar view is exported

Unit\PoweredLocomotives

✓ An array of default powered locomotives can be returned
✓ A powered locomotive can retrieve its associated operation shifts

Unit\RoleCompetencies

✓ A role type can be derived from a role competency
✓ A role competency has a corresponding tier between one and ten
✓ An array of default role competencies for a controller can be returned
✓ An array of default role competencies for a guard can be returned
✓ An array of default role competencies for a blockman can be returned
✓ An array of default role competencies for a powered locomotive can be returned
✓ An array of default role competencies for a steam locomotive can be returned
✓ A role competency can retrieve its associated operation shifts

Unit\RoleTypes

✓ An array of default role types can be returned
✓ A user can retrieve its associated operation shifts

Unit\Roles

✓ A role type can be derived from a role

✓ A user can be derived from a role
✓ A role competency can be derived from a role
✓ A role can also have a null value when no competency is set

Unit\SteamLocomotives

✓ An array of default steam locomotives can be returned
✓ A steam locomotive can retrieve its associated operation shifts

Unit\Users

✓ A user instance can be identified as myself
✓ A user instance can be identified as a non administrator
✓ A user instance can be identified as an administrator
✓ A user can retrieve its associated operation shifts

Feature\Locations

✓ A guest cannot view locations

✓ A user can view all locations
✓ A user can view a location
✓ A user cannot store a location
✓ A user cannot update a location
✓ A user cannot destroy a location
✓ An admin can store a location
✓ An admin can update a location
✓ An admin can destroy a location

Feature\OperationShifts

✓ The operation shifts index redirects to the operations index
✓ An operation shift redirects to its parent operation
✓ A user with a competency can register for an operation shift without a competency with the right role type
✓ A user with a competency can register for an operation shift with adequate competency with the right role type
✓ A user without a competency can register for an operation shift without a competency with the right role type
✓ A user without a competency cannot register for an operation shift with the right role type

✓ A user cannot register for an operation shift without the right role type
✓ A user cannot register for an occupied operation shift
✓ A user can pull out of a shift that they have registered for
✓ A user cannot pull out of a shift that is occupied by someone else
✓ A user cannot pull out of a vacant shift
✓ A user cannot store an operation shift
✓ A user cannot update an operation shift
✓ A user cannot destroy an operation shift
✓ An admin can store an operation shift
✓ An admin can update an operation shift
✓ An admin can destroy an operation shift
✓ An admin can register for an operation shift irrespective of competencies and role type

Feature\Operations

✓ A guest cannot view operations
✓ A user can view operations
✓ A user can view an operation
✓ A user can view operations in the calendar
✓ A user can export operations to pdf download
✓ A user cannot store an operation

✓ A user cannot update an operation
✓ A user cannot destroy an operation
✓ An admin can store an operation
✓ An admin can update an operation
✓ An admin can destroy an operation

Feature\PoweredLocomotives

✓ A guest cannot view powered locomotives
✓ A user can view all powered locomotives
✓ A user can view a powered locomotive
✓ A user cannot store a powered locomotive
✓ A user cannot update a powered locomotive
✓ A user cannot destroy a powered locomotive
✓ An admin can store a powered locomotive
✓ An admin can update a powered locomotive
✓ An admin can destroy a powered locomotive

Feature\RoleCompetencies

✓ A guest cannot view role competencies
✓ A user can view role competencies
✓ A user can view a role competency
✓ A user cannot store a role competency
✓ A user cannot update a role competency

✓ A user cannot destroy a role competency
✓ An admin can store a role competency
✓ An admin can update a role competency
✓ An admin can destroy a role competency

Feature\RoleTypes

✓ A guest cannot view role types
✓ A user can view all role types
✓ A user can view a role type
✓ A user cannot store a role type
✓ A user cannot update a role type
✓ A user cannot destroy a role type
✓ An admin can store a role type
✓ An admin can update a role type
✓ An admin can delete a role type

Feature\Roles

✓ A guest cannot view roles
✓ A user can view roles
✓ A user can view a role
✓ A user cannot store a role
✓ A user cannot update a role
✓ A user cannot destroy a role

✓ An admin can store a role
✓ An admin can update a role
✓ An admin can destroy a role
✓ An admin cannot create roles outside a user edit view

Feature\SteamLocomotives

✓ A guest cannot view steam locomotives
✓ A user can view all steam locomotives
✓ A user can view a steam locomotive
✓ A user cannot store a steam locomotive
✓ A user cannot update a steam locomotive
✓ A user cannot destroy a steam locomotive
✓ An admin can store a steam locomotive
✓ An admin can update a steam locomotive
✓ An admin can destroy a steam locomotive

Feature\Users

✓ A guest cannot view all the users
✓ A user can view all users
✓ A user can view a user
✓ A user cannot store a user
✓ A user can update themselves

✓ A user cannot give themselves administrative privileges
✓ A user cannot modify their own date of last inspection
✓ A user cannot update a different user
✓ A user cannot destroy a user
✓ An admin can store a user
✓ An admin can update a user
✓ An admin can destroy a user
✓ An admin cannot destroy themselves

Appendix H

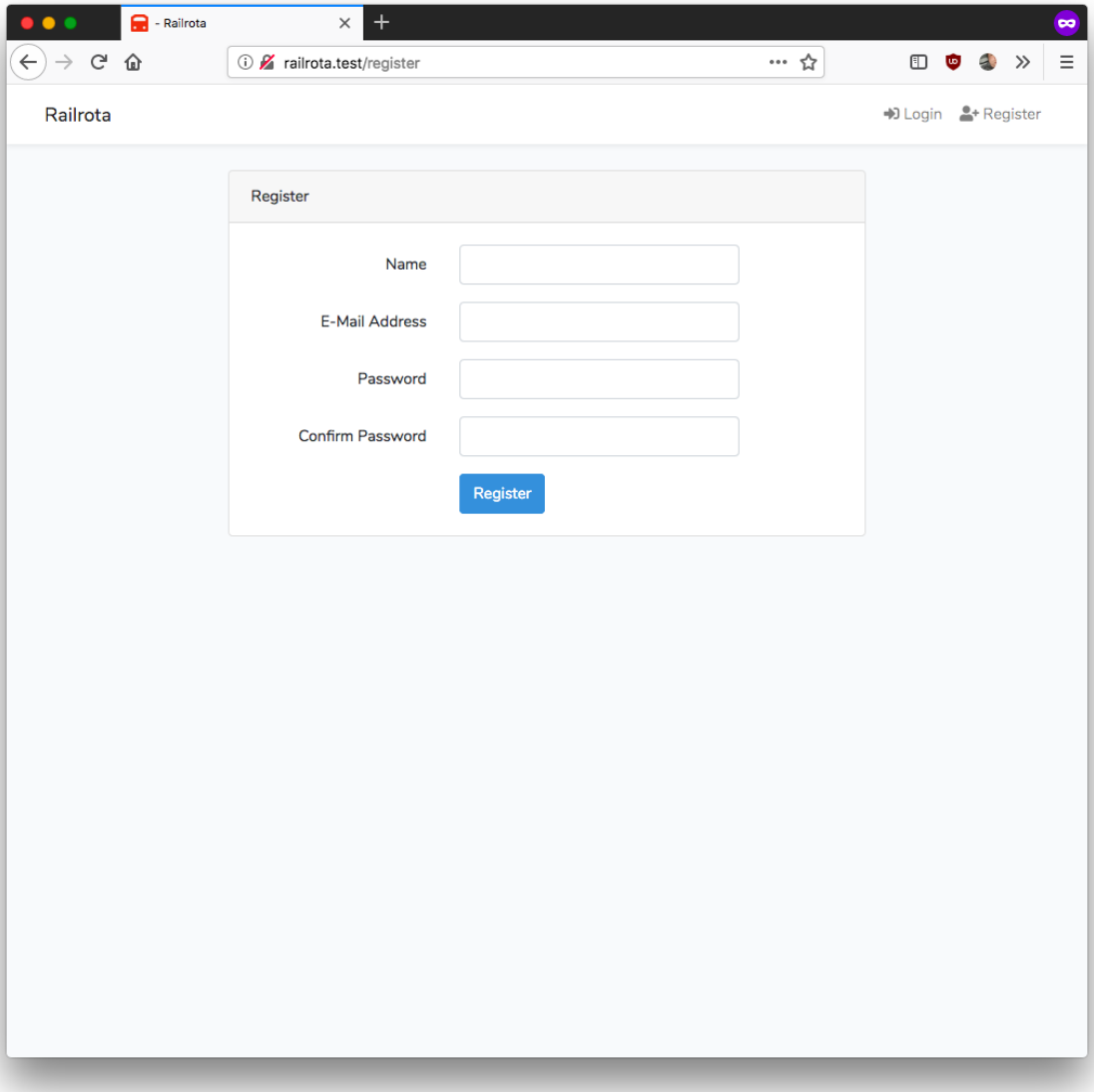
Manual Test Results

Registration:

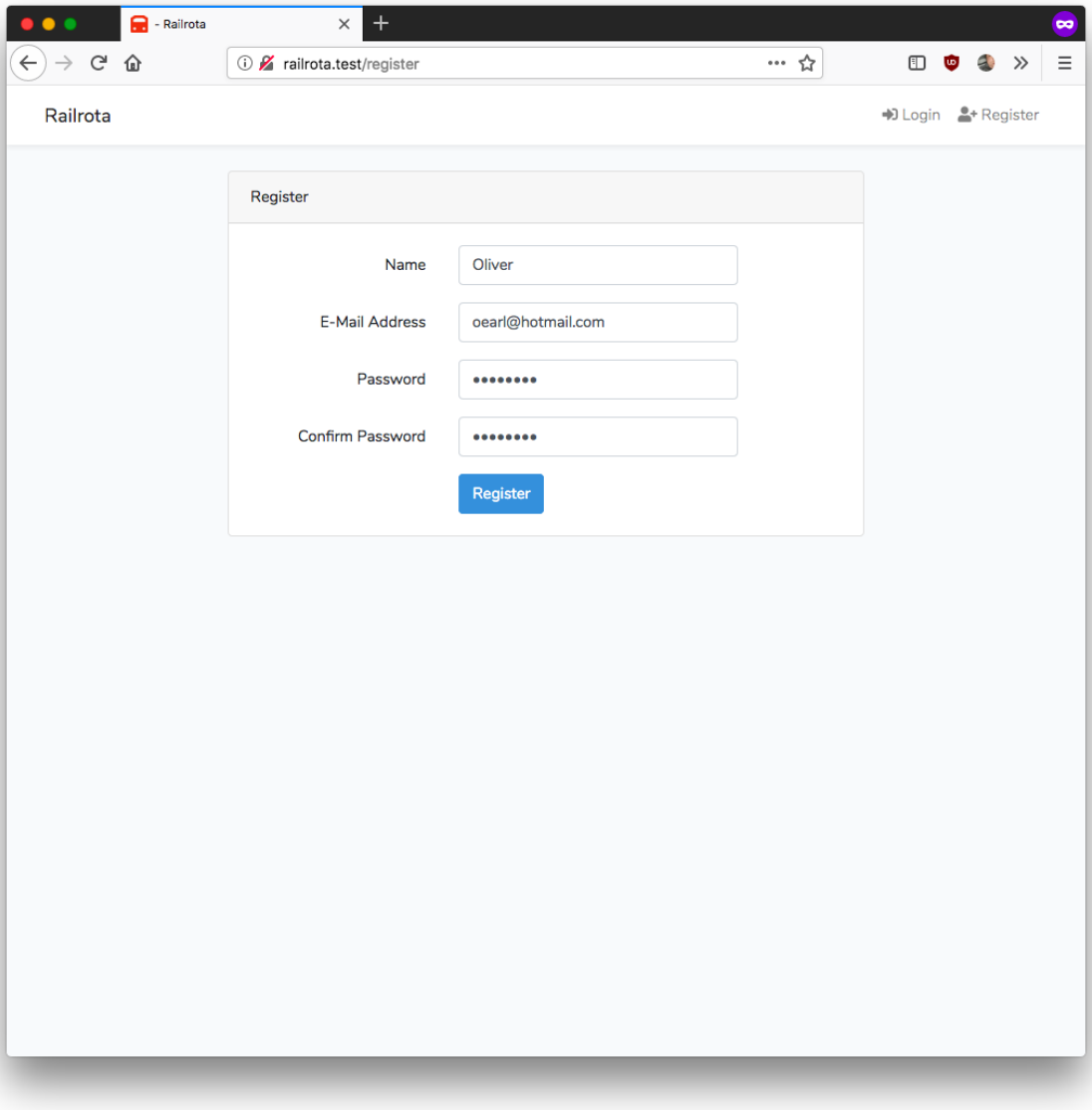
I should be able to register a new user, logging me in automatically as a user without administration privileges.

Test Date	Status
25/09/18	Pass

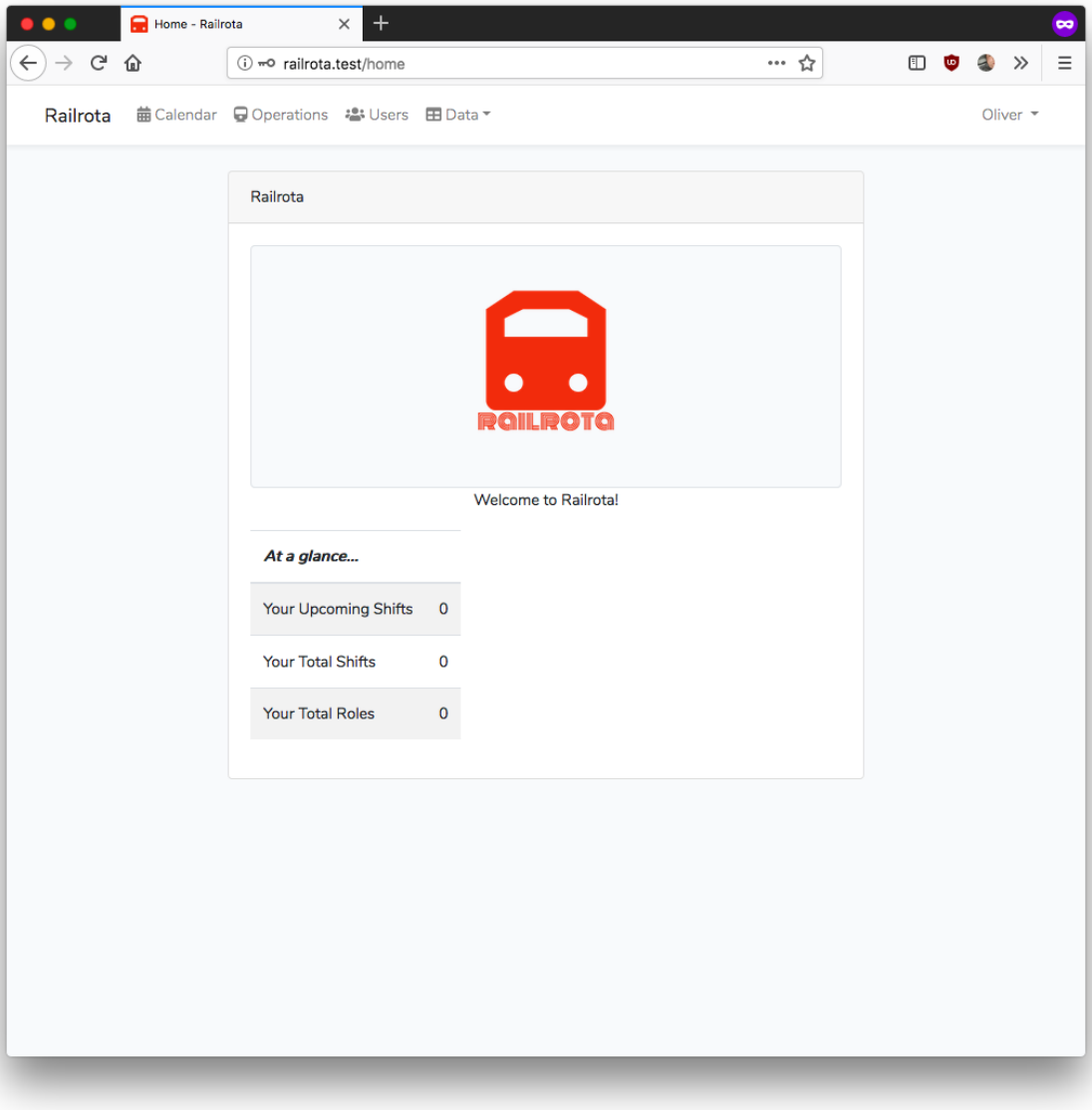
Screenshots:



Railrota Registration View



Railrota Registration View with details filled in



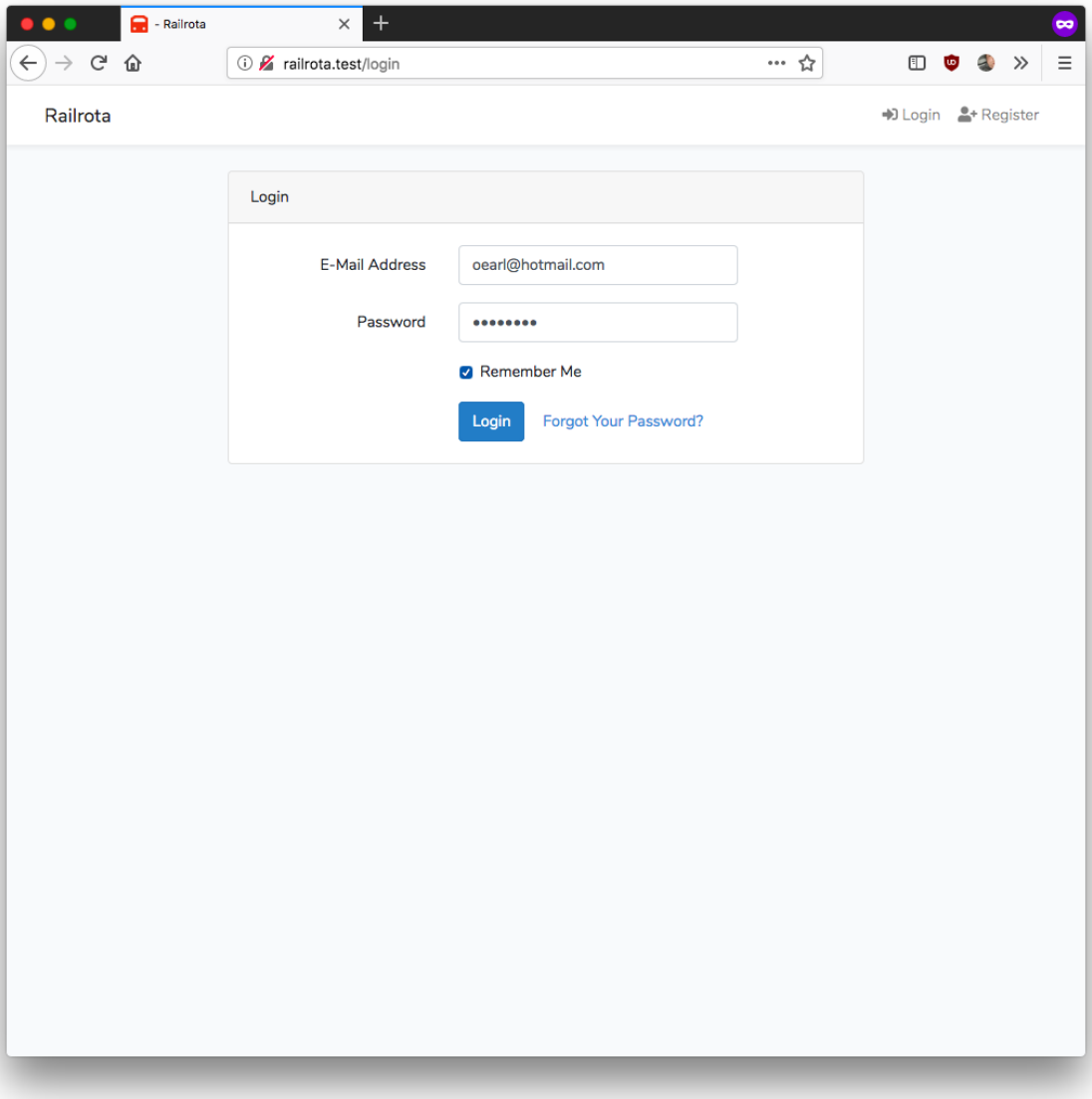
Successful redirection after registration

Login:

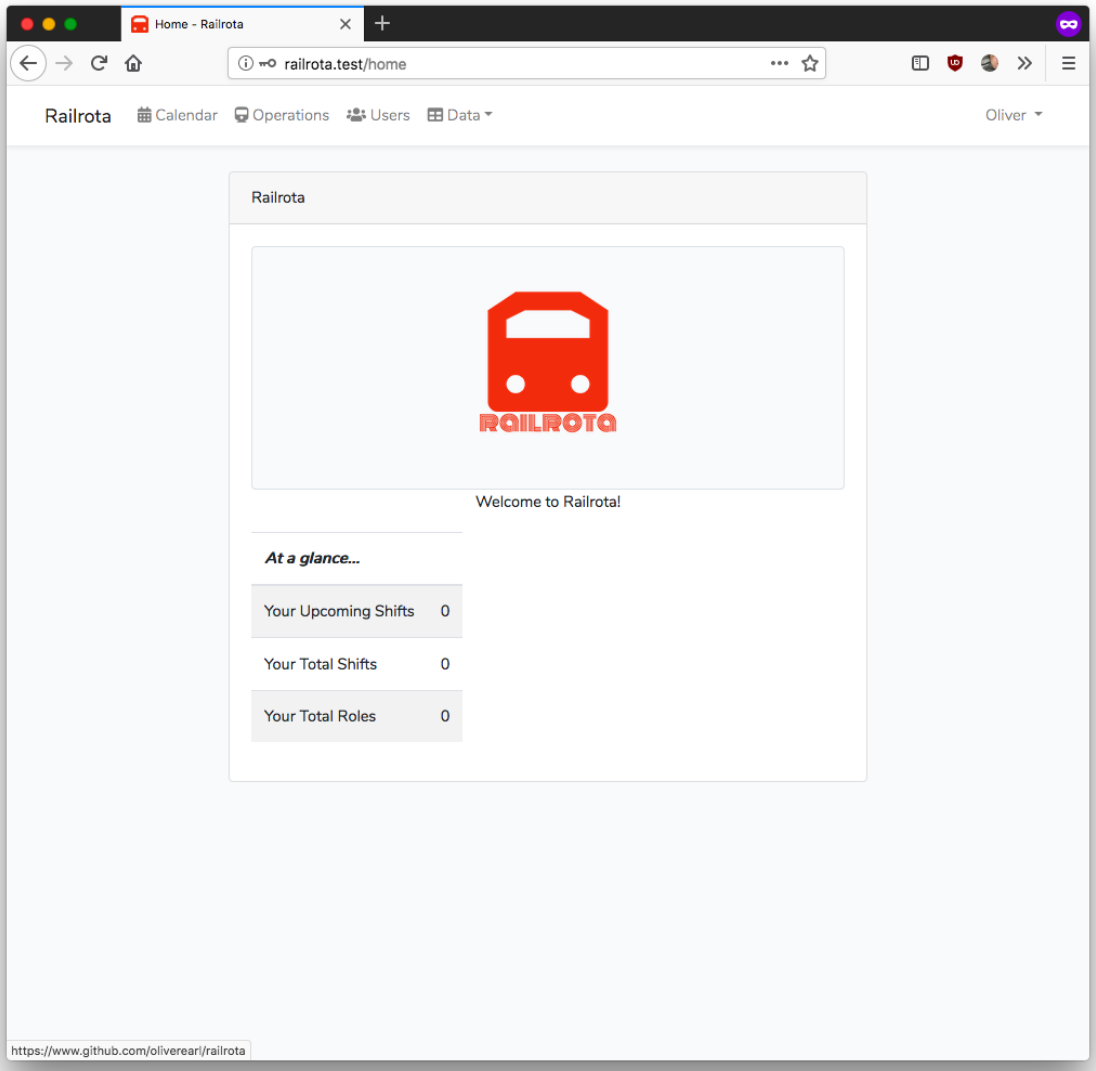
I should be able to login as a registered user after passing the correct credentials.

Test Date	Status
25/09/18	Pass

Screenshots:



Login Route with filled in credentials



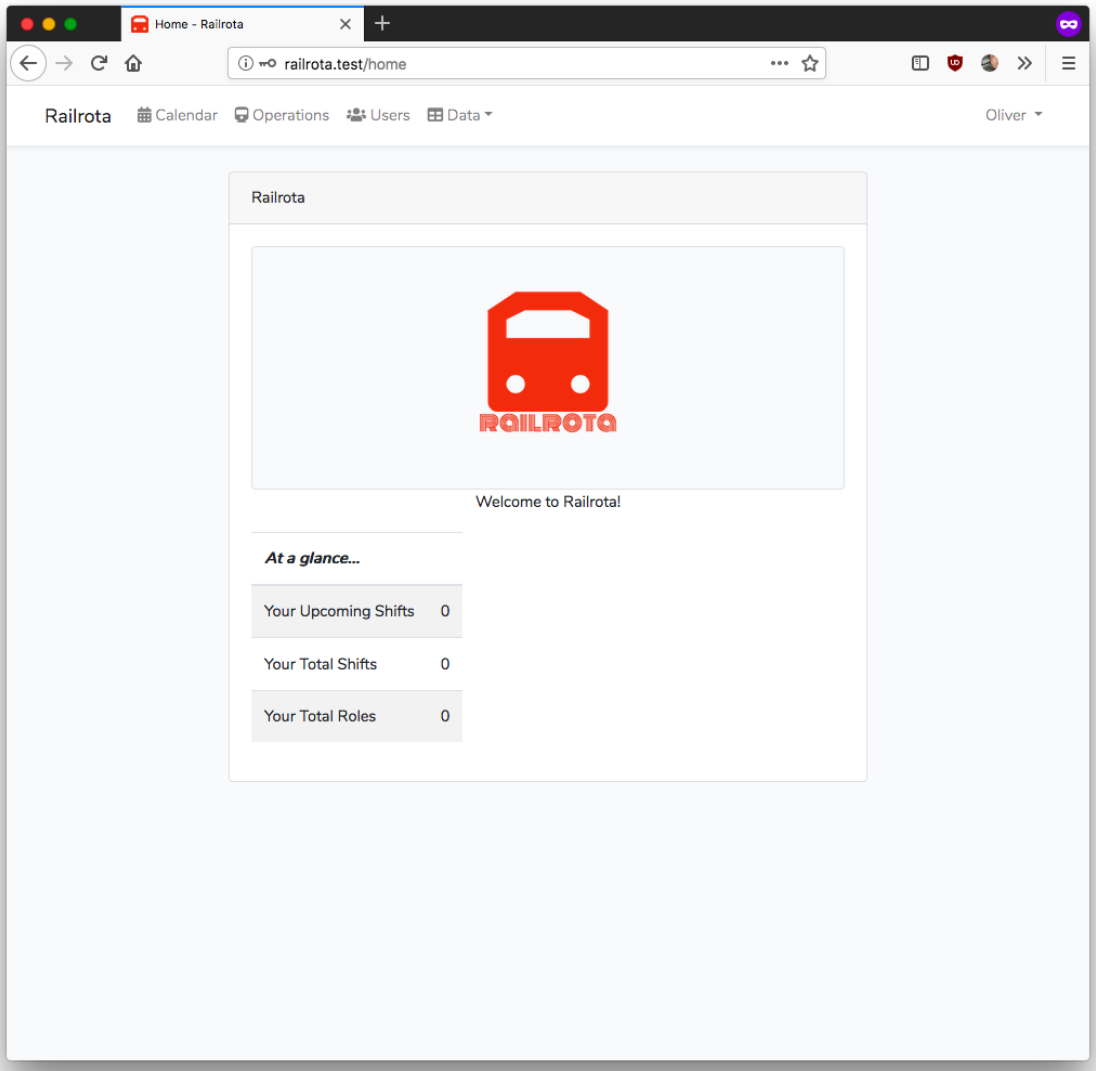
Successful redirection after login

Logout:

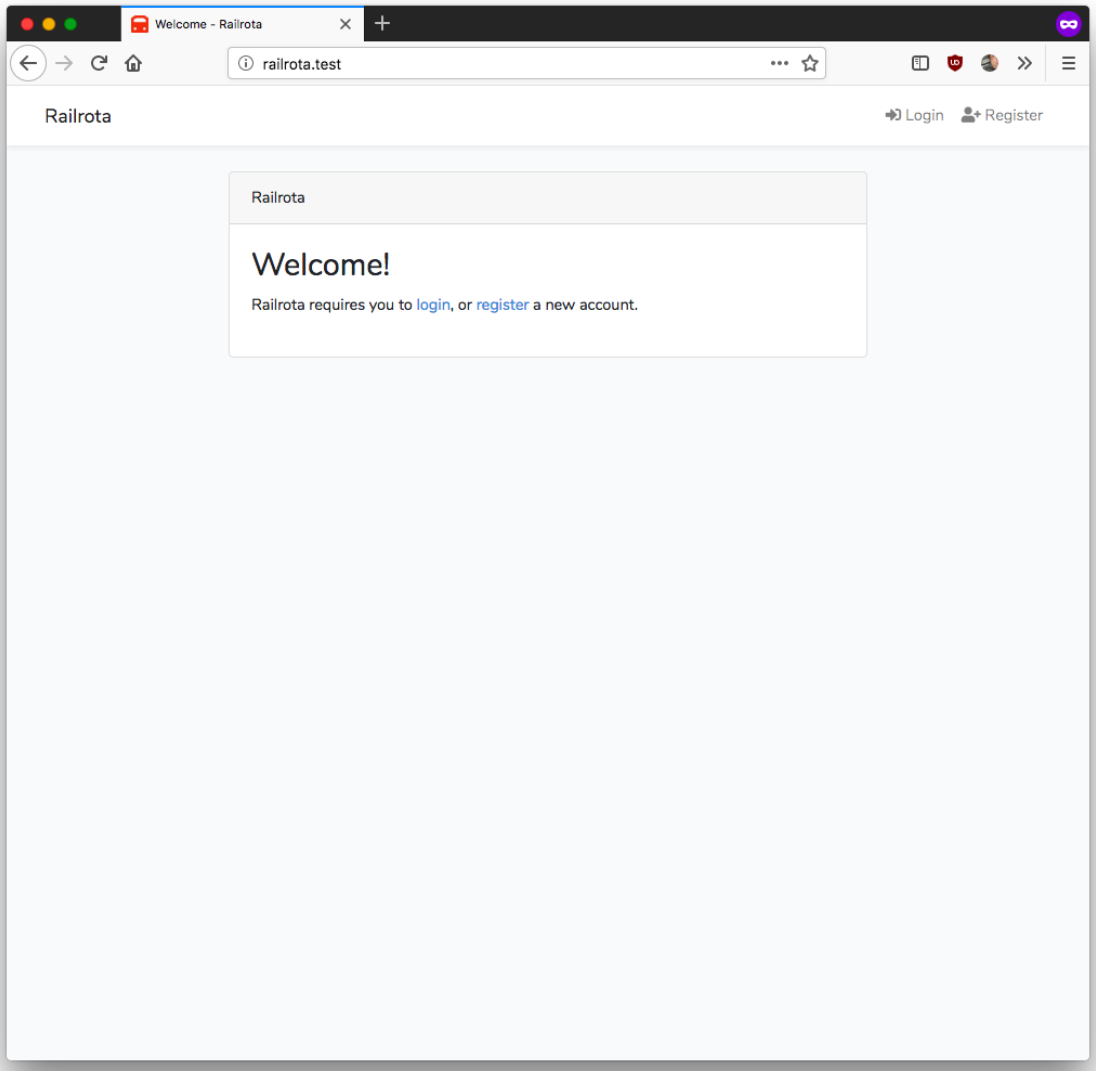
I should be able to log out after being logged in to the application.

Test Date	Status
25/09/18	Pass

Screenshots:



Authenticated and logged into the application



After successful log out

Password Reset:

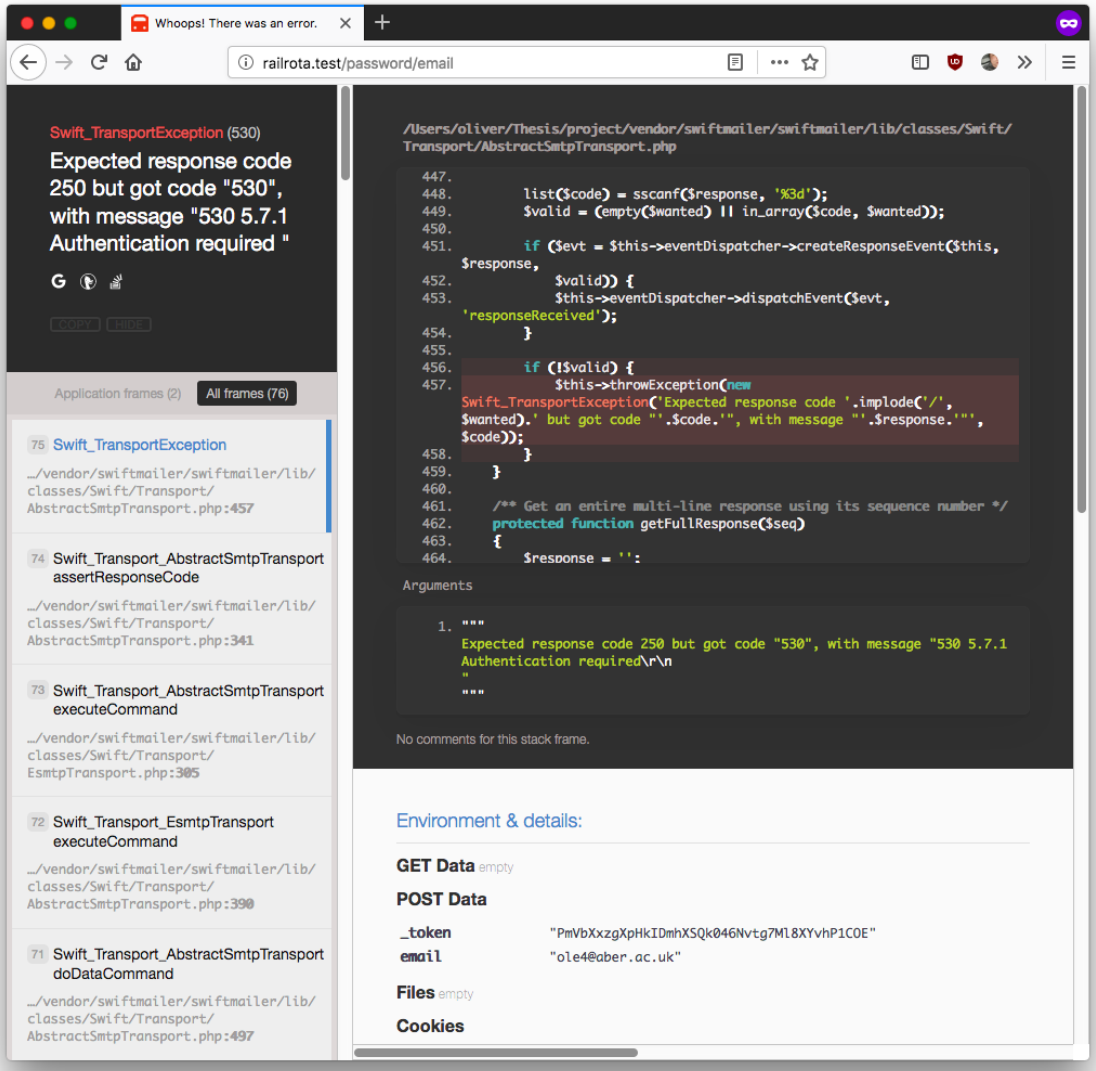
I should be able to reset my password if I forget it by providing my email address.

Test Date	Status
25/09/18	Fail

Failure Analysis:

This failure arose as a result of not having a configured email server within my test environment or having configured Laravel to use a dummy log file for the action. This *should* work in production provided an .env file has been configured with the relevant email server information.

Screenshots:



Exception that occurred when trying to submit the form

Appendix I

Project Readme

While this Markdown file can be found in the project directory itself, it has been included as an appendix to this document for sake of comprehensiveness.

Railrota is a full-stack volunteer rostering web application written in PHP usingLaravel. It is specifically intended for the Corris Railway volunteer base as a means of streamlining railway operations, staffing thereof, and volunteer skills and grades management.

It is the substantial programming component to Oliver Earl's CHM9360 MSc Project. It accompanies the written thesis component in this Git repository, at this time of writing.

Notable functionality includes:

- Responsive, easy-to-navigate application that functions on all screens and devices.
- Supports for custom job roles (role types), and grades (role competencies) attached to them.
- Ensures that staff can only volunteer for work that they are capable of fulfilling.
- Out of the box support for both steam and powered (diesel/electric) locomotives.
- Calendar view for easy viewing of upcoming and historic shifts, and of any vacancies they may fill.
- Support for exporting shift calendar as a downloadable PDF file.

Planned future functionality (outside the scope of the MSc deliverable, but for future FOSS revisions):

- Vue.js integration for SPA-like interface.
- API service allowing access to native mobile or desktop applications.
- Exporting shifts as Outlook, etc. appointment files.
- Email notifications on upcoming shifts and vacancies.
- Default and customisable operation templates.
- Automated rostering using simulated annealing.
- Improved code quality and reduced code duplication.

Live Instance

A live instance of the application is available for testing. It can be found at:
https://www.oliverearl.co.uk/railrota

Deployment

You must meet the system environment requirements to run Laravel 5.8. These requirements can be found in the official Laravel documentation.

This project is confirmed to work under Laravel Valet and Laravel Homestead (Vagrant) for development. It should work fine under XAMPP/WAMP/MAMP or in a Docker container, but these are untested.

Academic Hand-in Specific

If you have received Railrota as an academic hand-in, you should **not need to run any installation processes** unless you run into problems.

Automated Install

An automated installer is available for Linux and macOS. Simply run `bash install.sh` after downloading or cloning the repository. It is currently untested with Windows Subsystem for Linux or other Bash implementations under Windows, so proceed with caution. For help setting up your `.env` file, please see the below section.

Manual Install

- Download or clone the repository.
- You will need Composer. Installing this globally is recommended, but it is sufficient to install the .phar file to the Railrota directory.
- Run `composer install` to fetch and install dependencies. Substitute `composer` for `php composer.phar` on this and all subsequent instructions if you are running Composer locally.
- Copy the `.env.example` file to `.env`. You can customise settings however you wish, but you must specify your database information. You can use whichever database driver that your PHP installation can support - the default is **MySQL**.
- If you wish to run the program in development mode, change the `APP_ENV` value to **local**.
- Run `php artisan key:generate` to populate your `.env` file with a new key.
- Run `php artisan config:cache` to optionally cache your configuration. This is for performance.
- Run `php artisan telescope:install` to finish installing Laravel Telescope. View the Telescope documentation for more information.
- Migrate information into the database by running `php artisan migrate` (or `php artisan migrate:fresh` if you've run it previously) to provide a fresh database structure to your specified database.
- Create an admin user that you can log into using `php artisan db:seed --class=AdminSeeder`. By default, this uses **jane.doe@railrota.com** for its email address, and **password** for its password.
- Additional dummy data can be inserted into the database by using `php artisan db:seed` but this is generally unnecessary.

- You will need Node.js to use Laravel Mix and compile front-end assets, such as Sass and JavaScript.

Install dependencies using **Yarn** or **NPM** and use either `run dev` to produce uncompiled assets, or `run production` for minified, transpiled assets. If you have received this project in an academic hand-in, the files have been precompiled for you.

- Redump the autoloader using `composer dump-autoload`.
- Have fun!

Directories

A quick run-down of the directories is as follows:

- **app** contains Railrota code, such as policies, providers, middleware, controllers, and models.
- **bootstrap** contains Laravel code used for bootstrapping the application.
- **config** contains Laravel and library configuration files.
- **database** contains model factories, database migrations, and seeding classes.
- **docs** contains PHPDoc generated documentation by Sami.
- **node_modules** contains Node.js dependencies relied upon by Laravel Mix.
- **public** is the web-facing directory, containing the application entry-point, compiled assets, and social media images.
- **routes** contains lists of all the application routes.
- **resources** contains Laravel Blade views that make up the presentation of the website, as well as styling and front-end JavaScript.
- **storage** contains log files generated by the application, and generated PDF files downloaded by users.
- **tests** contains feature and unit tests.
- **vendor** contains PHP dependencies relied upon by Laravel, as well as libraries.

Testing

The application includes a full suite of unit and feature tests written using PHPUnit.

You can run PHPUnit tests by running `composer run test` to print out results to stdout. Alternatively, you can run `composer run test-export` to export test results to `test.html` in the root directory.

Documentation

Railrota uses Sami to generate documentation from source code docblocks.

You can run Sami by running `composer run doc` to generate a new **docs** subdirectory.

Troubleshooting

If you are having any problems, it would be advisable to check the Laravel documentation first to see if you are running into compatibility issues.

Alternatively, email me on ole4@aber.ac.uk for prompt help.

If you require assistance after the 27th of September 2019, when I will cease being a student at Aberystwyth University, please email me instead at oliver.earl@mail.bcu.ac.uk as I will be a full-time student at Birmingham City University from that point onwards.

Contributing

Contributing information will become available once the project has been made public.

License

MIT

Annotated Bibliography

[1] N. Devereaux, "Narrow gauge history: Not to be - the end of the corris railway," *Heritage Railway*, vol. 220, September 2016.

An article published in a popular steam railway magazine detailing the history and temporary closure of the Corris Railway.

[2] Corris Railway Society. (2018, August) Renaissance – 1966 on. [Online]. Available: https://www.corris.co.uk/renaissance-1966-on/

An article detailing the later years of the Corris Railway's history and the formation of the Corris Railway Society in 1966 to preserve what was left and to research its history.

[3] Office for National Statistics, "2011 census," March 2011. [Online]. Available: https://www.ons.gov.uk/census/2011census

The 2011 England and Wales census, recorded in 2011.

[4] R. Iorwerth and National Assembly for Wales, "The welsh language and legislation - the latest position," October 2013. [Online]. Available: https://seneddresearch.blog/2013/10/21/the-welsh-language-and-legislation-the-latest-position/

Outdated, but still relevant document from the Senedd outlying the current position of the Welsh language in law.

[5] BBC Corporation, "'laborious' welsh language rules on hold for private firms," June 2018. [Online]. Available: https://www.bbc.co.uk/news/uk-wales-44377188

Article discussing that while certain companies, such as those in energy and transport, are still not required to provide services in Welsh, pressure continues to mount for such regulations to be applied to the rest of the private sector in Wales.

[6] P. A. O'Keefe and L. Linnenbrink-Garcia, "The role of interest in optimizing performance and self-regulation," *Journal of Experimental Social Psychology*, vol. 53, pp. 70 – 78, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002210311400016X

Psychology paper exploring the importance of interest in task performance.

[7] P. A. O'Keefe, "Why interest is crucial to your success," https://today.duke.edu/2014/04/interest, April 2014.

Follow-up online publication further elaborating on the findings of the psychological study and how interest prevents burnout whilst optimising task performance.

[8] HarperCollins Publishers COBUILD Advanced English Dictionary. (2019, July) Definition of 'roster'.

Dictionary definition of the word 'roster', with background information on its meaning, etymology, synonyms, and examples.

[9] A. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier, "Staff scheduling and rostering: A review of applications, methods and models," *European Journal of Operational Research*, vol. 153, no. 1, pp. 3 – 27, 2004, timetabling and Rostering. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S037722170300095X

Paper that explores the various segments of the rostering problem, the various areas in which the problem is encountered and may differ, and solutions / methods used to combat it, including the use of artificial intelligence and mathematical algorithms.

[10] Y. Chen, A. Liu, E. Sciannella, and A. Zhang, "A comparison of approaches to the nurse scheduling problem," December 2016.

Paper comparing different algorithms used to solve the NSP problem, equations, and code examples.

[11] P. Maes, "Agents that reduce work and information overload," *Commun. ACM*, vol. 37, pp. 30–40, 1994.

An old paper from 1994 that discusses where software has reduced or made obsolete tasks and work that would typically involve ardeous labour and working with large amounts of information. Included in this, is the usage of employee scheduling software.

[12] I. Tassopoulos and G. Beligiannis, "A generic two-phase stochastic variable neighborhood approach for effectively solving the nurse rostering problem," *Algorithms*, vol. 6, 05 2013.

A paper detailing a two-phase stochastic variable neighbourhood approach to NSP, which was found to be significantly more efficient than six other approaches that were also investigated.

[13] E. K. Burke, P. De Causmaecker, G. V. Berghe, and H. Van Landeghem, "The state of the art of nurse rostering," *Journal of Scheduling*, vol. 7, no. 6, pp. 441–499, Nov 2004. [Online]. Available: https://doi.org/10.1023/B:JOSH.0000046076.75950.0b

Literature review that explores the state of nurse rostering solutions circa 2004 and provides some background insight into the problem in general.

[14] J. McCall, "Genetic algorithms for modelling and optimisation," *Journal of Computational and Applied Mathematics*, vol. 184, no. 1, pp. 205 – 222, 2005, special Issue on Mathematics Applied to Immunology. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377042705000774

Paper explaining genetic algorithms, primarily as relevant to the field of immunology, but nevertheless useful.

[15] L. Augustine, M. Faer, A. Kavountzis, and R. Patel, "A brief study of the nurse scheduling problem (nsp)," December 2009, https://docplayer.net/10464669-A-brief-study-of-the-nurse-scheduling-problem-nsp.html.

An essay shared by students of Carnegie Mellon University providing background on the Nurse Scheduling Problem, and their own Java-based genetic algorithm (GA) implementation to resolve the problem, including statistical analysis.

[16] L. Baird. (1998, May) Simulated annealing. [Online]. Available: https://www.cs.cmu.edu/afs/cs.cmu.edu/project/learn-43/lib/photoz/.g/web/glossary/anneal.html

Glossary entry from learning materials provided by Carnegie Mellon University.

[17] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C (2Nd Ed.): The Art of Scientific Computing*. New York, NY, USA: Cambridge University Press, 1992, 0-521-43108-5.

A fascinating book on various algorithms and topics, ranging from numerical analysis, signal processing, machine learning, and so on, with examples written in C.

[18] Y.-W. Ko, D. Kim, M. Jeong, W. Jeon, S. Uhmn, and J. Kim, "An efficient method for nurse scheduling problem using simulated annealing," in *The 5th International Conference on Advanced Science and Technology, AST*, vol. 20, 2013, pp. 82–85.

Paper detailing the use of simulated annealing, and cost matrix-based simulated annealing to solve NSP.

[19] HOPS, "What is hops?" March 2011. [Online]. Available: https://www.heritage-ops.org.uk/about.php

History and information behind HOPS.

[20] ——, "Operations calendar overview," 2012 September. [Online]. Available: https://www.heritage-ops.org.uk/support_item.php?i=17

Screenshots and information on the operations calendar interface on HOPS.

[21] ——, "System components," March 2011. [Online]. Available: https://www.heritage-ops.org.uk/about_components.php

A partial detailing of important functionality that is available within HOPS.

[22] Wei Jiang, Meng Zhang, Bin Zhou, Yujian Jiang, and Yingwei Zhang, "Responsive web design mode and application," in *2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA)*, Sep. 2014, pp. 1303–1306.

A paper that covers the benefits and purpose of responsive design, and provides insight into its implementation, such as using CSS media queries.

[23] Three Rings CIC, "Our values," 2017. [Online]. Available: https://www.threerings.org.uk/#values

Features and values of the Three Rings application and of their organisation.

[24] D. Q and Three Rings CIC, "Dan: Why i volunteer with three rings," December 2016. [Online]. Available: https://www.threerings.org.uk/our-team/volunteer/dans-story/

A history behind Three Rings from the Head of Development and founder, Dan, and how he came up with the idea whilst volunteering for a Nightline branch in Aberystwyth.

[25] Aberystwyth Nightline, "Aberystwyth nightline homepage," May 2017. [Online]. Available: http://nightline.aber.ac.uk/en/index.php

Homepage of the Aberystwyth Nightline service, where 3R developer Dan Q volunteered back in 2002.

[26] Three Rings CIC, "Three rings prices," 2017.

Full price estimations for the usage of Three Rings service.

[27] ——, "Three rings features," 2017.

List of features, including screenshots, of Three Rings functionality.

[28] ——, "Autopopulate and suggest somebody," 2019. [Online]. Available: https://www.3r.org.uk/help/admin/autopopulate-and-suggest-somebody

An extract from the Three Rings documentation that explains how to enable the autopopulate functionality and the available rulesets that govern how the algorithm operates. May require login to access.

[29] Voluntary Action Sheffield (VAS), "Software for managing volunteer shifts or online rota's[sic]," November 2014. [Online]. Available: http://www.sheffieldvolunteercentre.org.uk/software-for-managing-volunteer-shifts-or-online-rotas

A review of various shift or rota solutions specifically for volunteering.

[30] ABC Roster, "Abc roster homepage," November 2017.

Homepage for the ABC Roster freeware application, with download and donation links, changelog, screenshots, and feature information.

[31] ——, "Abc roster 2.3 manual," January 2017.

Online manual for the ABC Roster software, version 2.7.

[32] S. Ambler. (2005, October) Agile requirements change management. [Online]. Available: http://agilemodeling.com/essays/changeManagement.htm

An article on agile change management, primarily from both a Disciplined Agile Delivery (DAD) and Extreme Programming (XP) perspective.

[33] Agile Alliance, "What is agile?" 2015 June. [Online]. Available: https://www.agilealliance.org/agile101/

An introduction to agile methodologies and agile development, with a glossary of important terms.

[34] J. Highsmith and A. Cockburn, "Agile software development: the business of innovation," *Computer*, vol. 34, no. 9, pp. 120–127, Sep. 2001.

[35] Scrum.org, "What is scrum? what is scrum?" August 2012. [Online]. Available: https://www.scrum.org/resources/what-is-scrum

Lengthy webpage that provides an introduction and detailed information, as well as infographics, into the Scrum framework, its components, terminology, and further learning resources.

[36] I. Takeuchi, Hirotaka; Nonaka, "The new new product development game," *Harvard Business Review*, January 1986.

One of the earliest known write-ups on the Scrum framework / methodology by Japanese professors Takeuchi and Nonaka. Article describes the framework and its effectiveness in the current climate.

[37] C. Drumond, "What is scrum?" November 2013. [Online]. Available: https://www.atlassian.com/agile/scrum

A series of articles from Atlassian that explain Scrum in detail, and most notably the roles that compromise the methodology.

[38] S. Ambler, "Fdd project lifecycle," 2005. [Online]. Available: http://agilemodeling.com/essays/fdd.htm

Feature-Driven Development image on Agilemodeling.com

[39] Arkk Group, "Feature driven development: A guide," April 2015. [Online]. Available: https://www.arrkgroup.com/thought-leadership/feature-driven-development-a-guide/

A guide and history of Feature-Driven Development.

[40] S. Goyal, "Major seminar on feature driven development in agile techniques for project management and software engineering," 2008. [Online]. Available: http://csis.pace.edu/~marchese/CS616/Agile/FDD/fdd.pdf

[41] L. Karam, "Feature driven development (fdd) processes and comparison to other agile methodologies," July 2017. [Online]. Available: https://www.upwork.com/hiring/for-clients/agile-methodologies/

Article that describes Feature-Driven Development, but also provides a comparison of it to other popular methodologies, such as Scrum and eXtreme Programming (XP)

[42] W. Carstensen, "A brief history of version control," November 2016. [Online]. Available: https://www.red-gate.com/blog/database-devops/history-of-version-control

Detailed article on the history of version control, its iterations, and its origins in engineering.

[43] The Linux Foundation, "10 years of git: An interview with git creator linus torvalds," April 2015. [Online]. Available: https://www.linuxfoundation.org/blog/2015/04/10-years-of-git-an-interview-with-git-creator-linus-torvalds/

An interview with Linus Torvalds, the creator of both Git and the Linux kernel, about why he created Git and his opinions on its stance and position today within software engineering.

[44] G. Robinson, "Git in two minutes (for a solo developer)," October 2014. [Online]. Available: https://www.garyrobinson.net/2014/10/git-in-two-minutes-for-a-solo-developer.html

An article that explains the benefits of using VCS such as Git for even solo developers, and explains how to get started with implementing such a workflow.

[45] J. E. Inzaurraga, "Communication in agile software development… does it matter?" November 2016. [Online]. Available: https://www.hexacta.com/2016/11/10/communication-agile-software-development/

A detailed article exploring the importance of communication within teams and with customers in agile development in relation to the points outlined in the Agile Manifesto.

[46] L. Cao and B. Ramesh, "Agile requirements engineering practices: An empirical study," *IEEE Software*, vol. 25, no. 1, pp. 60–67, Jan 2008.

An in-depth look at the benefits of various agile methodologies and a study based on the results of adoption by various businesses.

[47] A. Melcon, "Discord: Everything you need to know," March 2018. [Online]. Available: https://www.tomsguide.com/us/what-is-discord,review-5203.html

An article explaining the Discord platform, a brief history, and its functionality.

[48] L. Giret, "Discord explained: what it is and why you should care, even if you're not a gamer," October 2018. [Online]. Available: https://www.onmsft.com/feature/discord-explained-what-it-is-and-why-you-should-care-even-if-youre-not-a-gamer

An article looking into the rise of the Discord platform and how it competes with other popular messaging and VoIP platforms, such as Skype, and its appeal outside of the gaming market.

[49] Microsoft Corporation, "Add an email account to outlook," October 2018. [Online]. Available: https://support.office.com/en-us/article/add-an-email-account-to-outlook-e9da47c4-9b89-4b49-b945-a204aeea6726

Quick start guide for Microsoft Outlook that outlies its functionality.

[50] R. Branson, "Keep it simple, stupid," January 2018. [Online]. Available: https://www.virgin.com/richard-branson/keep-it-simple-stupid

An article by Sir Richard Branson, discussing the use of the KISS principle by Virgin in both its aeronautical and astronautical engineering.

[51] H. McCracken, "Inside evernote's brain," August 2018. [Online]. Available: https://www.fastcompany.com/90216018/inside-evernotes-brain

A long and detailed article into the history of the Evernote application, and provides insight as the company looks to reinvent itself going into 2019.

[52] B. Vinegar, "The best tool for the job, isn't always," July 2014. [Online]. Available: https://medium.com/@bentlegen/the-best-tool-for-the-job-isnt-always-6ed364f3f775

Former Lead Front-End Engineer for comment firm Disqus argues that using slightly antiquated technology, in this case Backbone.js, is sometimes a better decision than using something more modern and effective, such as Angular.

[53] J. Wolfe, "The history of php," April 2018. [Online]. Available: https://medium.com/@johnwolfe820/the-history-of-php-ffb920ba4555

A detailed history into the PHP programming language.

[54] J. O'Brien, "A brief history of laravel," July 2016.

A brief history into the Laravel framework and some information about the lead developer, Taylor Otwell.

[55] J. Shah, "Why laravel is consider[sic] as one of the best php framework in 2018," January 2019. [Online]. Available: https://medium.com/techcompose/why-laravel-is-consider-as-one-of-the-best-php-framework-in-2018-4f40def9c69c

An article that details some of the key functionalities found in Laravel, and why they are important.

[56] Laravel LLC, "Eloquent: Getting started," March 2019. [Online]. Available: http://carbondate.cs.odu.edu/#https://laravel.com/docs/5.8/eloquent

Official Laravel documentation on the Eloquent ORM.

[57] Mozilla Foundation, "Javascript language resources," July 2013. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language_Resources

A table listing links to current ECMAScript standards as well as obsolete ones.

[58] D. Buytaert, "A history of javascript across the stack," February 2016. [Online]. Available: https://dri.es/a-history-of-javascript-across-the-stack

A detailed history of JavaScript on both client- and server-side applications, and how the Google V8 engine changed the landscape of the web.

[59] F. Copes. (2018, August) A brief history of node.js. [Online]. Available: https://flaviocopes.com/node-history/

A brief timeline of Node.js history and its major events and milestones across the last decade.

[60] Mozilla Foundation, "Express/node introduction," February 2017. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

A tutorial made by Mozilla which teaches the basics of Node.js and the Express framework, and provides guidance on how to build a basic website and API using them.

[61] Python Software Foundation, "History of the software," June 2001. [Online]. Available: https://docs.python.org/2.0/ref/node92.html

An extract from an old version of the Python Reference Manual that has some history of the Python language.

[62] N. Heath, "Fastest-growing programming language? python's popularity is still climbing," August 2018. [Online]. Available: https://www.techrepublic.com/article/fastest-growing-programming-language-pythons-popularity-is-still-climbing/

An article that discusses Python's climb into popularity and how it continues to grow, as well as its various use cases and history.

[63] A. Shaleynikov, "Node.js vs. django: Is javascript better than python?" November 2018. [Online]. Available: https://dzone.com/articles/nodejs-vs-djangois-javascript-better-than-python

An article that provides a surprisingly in-depth but concise pros and cons analysis of both JavaScript / Node.js and Python / Django.

[64] K. Sierra, "When the "best tool for the job"... isn't." August 2006. [Online]. Available: https://headrush.typepad.com/creating_passionate_users/2006/08/when_the_best_t.html

An article that draws a comparison between the two schools of thought of 'those who use the tool they love' and 'those who use the best tool for job' and argues for the former.

[65] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "Manifesto for agile software development," 2001. [Online]. Available: http://www.agilemanifesto.org/

The Agile Manifesto, the initial and most important document in the history of agile methodologies and one of the most substantial pieces of literature in software engineering.

[66] Laravel LLC, "Compiling assets (mix)," March 2019. [Online]. Available: https://laravel.com/docs/5.8/mix

Official Laravel documentation on Laravel Mix - its implementation of Webpack.

[67] ——, "Javascript and css scaffolding," March 2019. [Online]. Available: https://laravel.com/docs/5.8/frontend

Official Laravel documentation on what it includes out-of-the-box for CSS and JavaScript front-end elements, including the Bootstrap and Vue.js frameworks.

[68] R. Frey, "Mvc design pattern image," May 2010. [Online]. Available: https://en.wikipedia.org/wiki/User:RegisFrey

Creative Commons MVC Design Pattern image from Wikimedia artist Regis Frey.

[69] N. Ighodaro, "How laravel implements mvc and how to use it effectively," May 2018. [Online]. Available: https://blog.pusher.com/laravel-mvc-use/

In-depth article that explores the Laravel framework and how it implements models, controllers, and views.

[70] M. Jailia, A. Kumar, M. Agarwal, and I. Sinha, "Behavior of mvc (model view controller) based web application developed in php and .net framework," in *2016 International Conference on ICT in Business Industry Government (ICTBIG)*, Nov 2016, pp. 1–5.

Paper that evaluates MVC performance when implemented in PHP and in .NET.

[71] T. Otwell, "Taylor otwell, twitter," August 2015. [Online]. Available: https://twitter.com/taylorotwell/status/634384068990267392

Tweet from Laravel development lead and founder Taylor Otwell on his opinion whether Laravel is an MVC framework.

[72] The PHP Group, "Pdo - introduction," April 2008. [Online]. Available: https://www.php.net/manual/en/intro.pdo.php

Official PHP documentation on the PHP Data Object (PDO)

[73] HM Government, "Data protection," August 2017. [Online]. Available: https://www.gov.uk/data-protection

Official UK government information on GDPR / Data Protection Act 2018.

[74] H. The Charity Commission, "Make sure your charity is ready for gdpr," May 2018. [Online]. Available: https://www.gov.uk/government/news/make-sure-your-charity-is-ready-for-gdpr

Official UK government guidance on GDPR and what charities must do to prepare for its implementation.

[75] Z. Atkinson, Leon; Suraski, *Core PHP Programming*, third edition ed. Prentice Hall, 2003.

An outdated, but solid PHP 5 textbook that is cheaply available, covers aspects of object-oriented programming, and holds up reasonably well today.

[76] Microsoft Corporation, "Sql injection," April 2012. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms161953(v=sql.105)?redirectedfrom=MSDN

Documentation from SQL Server 2008 covering SQL injection and mitigation.

[77] Laravel LLC, "Validation," March 2019. [Online]. Available: https://laravel.com/docs/5.8/validation

Laravel official documentation on validation functions and functionality.

[78] ——, "Csrf protection," March 2019. [Online]. Available: https://laravel.com/docs/5.8/csrf

Official Laravel documentation on CSRF protection. (Cross-Site Request Forgery)

[79] JetBrains s.r.o, "Phpstorm features," September 2019. [Online]. Available: https://www.jetbrains.com/phpstorm/features/

Product page for JetBrains PhpStorm.

[80] Á. Kiss, "How to integrate design into your agile process?" October 2016. [Online]. Available: https://uxstudioteam.com/ux-blog/agile-design-process/

An interesting write-up discussing the differences between design in traditional SDLC processes and how it can be moved over to an agile methodology.

[81] J. Marah, B. Jakobus, M. Lambert, and S. Moreto, *Bootstrap 4 Responsive Web Design*. Packt Publishing, July 2017.

Textbook on Bootstrap 4 with instructions, tutorials, and in-depth information on various Bootstrap functionality.

[82] w3schools.in, 2019. [Online]. Available: https://www.w3schools.in/laravel-tutorial/application-directory-structure/

A detailed explanation of the Laravel directory structure.

[83] Laravel LLC. (2019, March) Request lifecycle. [Online]. Available: https://laravel.com/docs/master/lifecycle

Official Laravel documentation covering its request lifecycle and how it works at the high level.

[84] Mozilla Foundation, "Server-side web frameworks," September 2019. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks

An article by Mozilla that provides a huge wealth of information on server-side web frameworks, their benefits, what they can offer you, and examples by language.

[85] Mallow Technologies, "Request life cycle of laravel," June 2016. [Online]. Available: http://blog.mallow-tech.com/2016/06/request-life-cycle-of-laravel/

Diagram that accurately depicts the request lifecycle of Laravel.

[86] PHP Framework Interop Group, "Psr-4 autoloader," January 2015. [Online]. Available: https://www.php-fig.org/psr/psr-4/

The fourth PSR PHP coding standard - PSR-4 Autoloader.

[87] M. Stauffer, *Laravel: Up and Running*. O'Reilly Media, Inc., November 2016.

A really great, though slightly outdated book on Laravel and how to get started. A personal recommendation for getting started with the framework and complements the official documentation nicely.

[88] Laravel LLC, "Http controllers," March 2019. [Online]. Available: https://laravel.com/docs/5.0/controllers#restful-resource-controllers

Official Laravel documentation on HTTP controllers and RESTful methods.

[89] C. Hsieh, H. Hsieh, and Y. C. Cheng, "A method for web application data migration based on restful api: A case study of ezscrum," in *2016 International Conference on Applied System Innovation (ICASI)*, May 2016, pp. 1–4.

[90] Laravel LLC, "Blade templates," March 2019. [Online]. Available: https://laravel.com/docs/5.8/blade

Official Laravel documentation for Blade templates.

[91] P. Underwood, "The ups and downs of php template engines," September 2013. [Online]. Available: https://dzone.com/articles/ups-and-downs-php-template

An article demonstrating the benefits of using a PHP template engine, such as Blade or Twig, over traditional vanilla PHP alone.

[92] Laravel LLC, "Authentication," March 2018. [Online]. Available: https://laravel.com/docs/5.8/authentication

Official Laravel documentation on authentication and its scaffolding.

[93] Larashout, "Laravel policies - controlling authorization in laravel," February 2019.

Article that demonstrates how to use policies to manage authorisation actions in a Laravel project.

[94] DOMPDF Contributors, "Dompdf," May 2019. [Online]. Available: https://github.com/dompdf/dompdf

GitHub page for dompdf PHP library.

[95] B. Heuvel and laravel-dompdf Contributors, "Dompdf wrapper for laravel," October 2016. [Online]. Available: https://github.com/barryvdh/laravel-dompdf

GitHub page for laravel-dompdf wrapper library.

[96] D. Radigan, "Engineering higher quality through agile testing practices," January 2018. [Online]. Available: https://www.atlassian.com/agile/software-development/testing

An article hosted by Atlassian that explores the benefits of agile testing practices in comparison to traditional Waterfall style testing that involves merely producing components and 'throwing them over the wall'.

[97] Laravel LLC, "Testing: Getting started," March 2019. [Online]. Available: https://laravel.com/docs/5.8/testing

Official Laravel documentation on testing and its use of PHPUnit.

[98] S. Bergmann and PHPUnit Contributors, "Phpunit," August 2019. [Online]. Available: https://github.com/sebastianbergmann/phpunit

GitHub page for PHPUnit testing framework.

[99] P. Hamill, *Unit Test Frameworks: Tools for High-Quality Software Development*. O'Reilly Media, November 2004.

An O'Reilly book on unit testing frameworks. Despite being outdated, most of the theory on unit testing and TDD remains the same, and is a helpful book on any developer's bookshelf.

[100] C. Hopping, B. Hellard, and R. Millman, "What is regression testing?" April 2019. [Online]. Available: https://www.itpro.co.uk/desktop-software/28859/what-is-regression-testing

Very detailed article on the different types of regression testing, strategies for regression testing, why it is needed, and when to do it.

[101] B. Fornal, "Negative testing," September 2019. [Online]. Available: https://dev.to/rfornal/negative-testing-4k55

An article discussing negative testing as part of a series on automated tests, why it is important to use negative tests, and provides some solid code examples.

[102] K. Nordeen, "Manual testing: A complement or prerequisite to automated testing," January 2016. [Online]. Available: https://dzone.com/articles/manual-testing-a-complement-or-prerequisite-to-aut

An article that discussed agile testing processes and how manual testing can play into that.

[103] Laravel LLC, "Laravel telescope," March 2019. [Online]. Available: https://laravel.com/docs/5.8/telescope

Official Laravel documentation for Telescope.

[104] A. Abati, "Getting started with laravel telescope: What can it do for you?" 2018 December. [Online]. Available: https://blog.logrocket.com/getting-started-with-laravel-telescope-what-can-it-do-for-you-719aaef07941/

An interesting insight into the individual features of Telescope and a great complementary read to the official documentation.

[105] Composer Contributors, "Composer - introduction," March 2019. [Online]. Available: https://getcomposer.org/doc/00-intro.md

Official Composer documentation and introduction to what it is and how to use it.

[106] PHP Framework Interop Group, "Psr-5 phpdoc," January 2019. [Online]. Available: https://github.com/php-fig/fig-standards/blob/master/proposed/phpdoc.md

Proposed PSR-5 standard for PHPDoc.

[107] Open Source Matters, Inc, "Docblocks - joomla! programming standards," August 2018. [Online]. Available: https://developer.joomla.org/coding-standards/docblocks.html

Official documentation from the Joomla! PHP Content Management System (CMS) that provides information on PHPDoc docblocks.

[108] FriendsOfPHP, "Sami documentation," September 2018. [Online]. Available: https://github.com/FriendsOfPHP/Sami

GitHub repository page for the Sami API documentor.

[109] T. Chow and D.-B. Cao, "A survey study of critical success factors in agile software projects," *Journal of Systems and Software*, vol. 81, no. 6, pp. 961 – 971, 2008, agile Product Line Engineering. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121207002208

A survey study of critical success factors in agile software projects, including customer involvement and face-to-face communication.

[110] M. Herman, "Django vs flask in 2019: Which framework to choose," July 2019. [Online]. Available: https://testdriven.io/blog/django-vs-flask/

A comparison of Flask and Django frameworks, their features, and various pros and cons.