College of Business, Technology & Engineering

DEPARTMENT OF COMPUTING

55-604707 Project - 2020

Project Title: Development of a prototype web-based timetabling and scheduling application tailored for a small business to improve the control employees and managers have over their working week.	
Author: Bryn Bowler	**Student ID: B7013157**
Supervisor: Elizabeth Uruchurtu	**Date of Submission: 30th March 2020**

Undergraduate Computing Programme - BSc/BSc Honours Degree in (please tick) :	
Computer Studies	☐
Computing	☒
Computing (Web Development)	☐
Other degree (please specify):	

Type of Project (please tick)	**Application** ☒	**Investigation** ☐

Confidential (please tick)
YES ☐ **Evidence for confidentiality included? YES -** ☐
NO ☒

Contents

Acknowledgements 5

Abstract 6

Chapter 1. Introduction 7

1.1. Aims and Objectives 7

1.2. Dissertation Outline 9

Chapter 2. Literature Review 10

2.1. Similar Work in this Field 10

2.1.1. Newfoundland University 10

2.1.2. Mobile and Web applications for Small and Medium Size Enterprises 11

2.1.3. PTIME 12

2.1.4. Conclusion 12

2.2. Similar/Related Applications 13

2.2.1. Acuity Scheduling 13

2.2.2. Bookafy 14

2.2.1. Google Calendar 15

2.2.5. Conclusion 15

2.3. Research into Software Development Tools, Programming Languages and Database management Tools 17

2.3.1. HTML 17

2.3.2. HAML 18

2.3.2. Javascript 18

2.3.3. DART 19

2.3.3. PHP 19

2.3.4. AJAX 20

2.3.4. MySQL 20

2.3.5. Content Management System 20

2.3.6 WAMP Server 21

2.3.7. Visual Studio Code 21

2.4 Chapter Conclusion 22

Chapter 3. Project Methodologies 22

3.1. Agile Methodology 23

3.2. Waterfall Methodology 24

3.3. The MoSCoW Method 25

3.4. Conclusion and Chosen Method 26
Chapter 4. Design and Development 26
4.1. Design 26
4.1.1. User Requirements 26
4.1.2. UML Use Case Diagram 28
4.1.3. User Flow Diagrams 30
4.1.4. Good Design Practices for Front End Design 31
4.1.5. Database Design 34
4.1.6. Planned Structure 35
4.3. Development 35
4.4. Conclusion 42
Chapter 5. Testing 42
5.1. Unit Testing 42
5.2. Integration Testing 43
5.3. System Testing 45
5.4. Acceptance Testing 46
5.5. Feedback 48
5.5.1. Engineers 48
5.5.2. Receptionist 49
5.5.3. Manger/Client 49
5.6. Conclusion 51
6. Evaluation 51
6.1. Evaluation of the Application 51
6.1.1. Login Page 51
6.1.2. Personal Timetable Page 51
6.1.3. Job Dashboard Page 52
6.1.4. Job Creation Page 52
6.1.5. Job Swap Page 53
6.1.6. Completed Jobs 53
6.1.7. Timetable Overview 53
6.1.8 Overall Application 53
Chapter 7. Conclusion and Future Work 54
7.1. Limitations of the Project 54
7.2. Functionality and Design Changes 55

7.3. The Future 56
7.4. Aim and Objectives 56
Appendix 57
References 87
Bibliography 90

Acknowledgements

I would like to thank my project supervisor Elizabeth Uruchurtu for her help in bringing this dissertation to fruition and her continued expertise and guidance despite the rockiness of the academic year.

I would also like to thank Gary for his generous acceptance and help as a client alongside all the employees of AvTechnical for their help in testing and evaluating this application.

Finally, I would like to thank my family, especially my Mum and Dad for their continued love and support in anything and everything I do.

Abstract

This dissertation explores the process of developing a prototype timetabling and scheduling web application tailored for a small business. The aim of this is to improve the control employees of this business have over their working week whilst assuring that information can be tracked and manged in a more efficient way to improve productivity. The business associated with this project has an issue. Often there is confusion about which employees will be conducting certain jobs and when they will be doing them. This prototype application aims to solve this by providing a clear timetable to each engineer and supplying tools to other employees to improve the process of timetabling and scheduling said jobs. This project will gather user requirements by discussing with the client what this application could provide to the business, design and develop a prototype application based of these user requirements and test and evaluate its usefulness with the client and employees. Finally, this project will look at the feedback received, discuss what could be improved and deliberate what could be done with the application in the future.

Chapter 1. Introduction

AV Technical Services is a company based in the midlands that install entertainment systems for Pubs and Clubs across the country. This includes Bingo, Karaoke and other games for patrons to enjoy at their local. The company is small, consisting of only a handful of office workers and 3 main engineers. The company deals with the management of different instillations, service calls and uninstallations of their equipment which has them travelling hundreds of miles across the UK a week.

The company currently has an issue. Engineers are sometimes told late on that they will be travelling across the country to perform various tasks at these pubs and clubs they refer to as sites. This can sometimes be frustrating for the engineers as there is confusion about what service calls need dealing with, when they will be dealt with and who will be dealing with them.

This dissertation will aim to solve this problem by creating a timetabling and scheduling application that can be used by all members of the company, giving each engineer their own timetable for the week, helping to make it clear what will be happening for the employees.

Currently the company uses a WhatsApp group and Outlook calendar to try and keep track of what will be happening, but this method sometimes leads to things getting forgotten about, mixed up or misunderstood. With a single application used by all members of the company, allowing engineers to check their own weekly timetable, a receptionist to assign other people jobs on their timetables and the manager being able to have an overview of all upcoming and completed jobs this business could reduce its losses and mistakes.

To help assess what the client is after from this project data will have to be gathered. This data gathering will help to understand and align the requirements of the project with the client’s business needs and to understand what the client wants out of this project. To achieve this an interview will be held with the client that will discuss the basic outline of the project and gather user requirements.

Toward the end of the project the client will become involved in the testing phase. This will require the help of members of all areas of the business to run a thorough testing of the application to gather feedback on what works and what doesn’t, the usability of the application and whether they believe this application could be useful to the business.

Before this, to help pitch this idea to the client a clear aim and a list of objectives and tasks had to be constructed.

1.1. Aims and Objectives

In order to create what is necessary for this project a clear aim, objectives and tasks to complete those objectives must be created. These aim and objectives will help to keep this project on tack and to assure that the necessary steps of this applications development are completed.

Overall, the aim of my project is to develop a prototype web-based timetabling and scheduling application tailored for this client to improve the control engineers and employees have over their working week.

To achieve this aim, the following objectives and tasks in Table 1 have been identified:

Objectives	Tasks
1. Explore similar existing applications and investigate their functionality.	1.1. Investigate the functionality of different applications. 1.2. Investigate what kind of languages and software these applications use.
2. Investigate different methodologies and select the appropriate model for the project.	2.1. Research methodologies and study how they compare. 2.2. Write a report on these methodologies and how they could relate to the project 2.3. Finalise a decision on a selected methodology and discuss why it has been chosen.
3. Research appropriate software tools, programming languages and database management tools.	3.1. Discuss the different applications and what tools and languages are used in their development.
4. Meet with client to review user requirements	4.1. Gather requirements from client to tailor application.
5. Create basic structure and front-end designs for the application.	5.1. Design an application structure that demonstrates how my application would operate. 5.2. Design a front end for my application that demonstrates the layout and the look and feel.
6. Design a database schema and structure.	6.1. Create a database diagram detailing the structure that will be necessary for the success of this application.
7. Begin development of tailored application and discuss deliverable with client.	7.1. Discuss deliverable with client 7.2. Begin coding application. 7.3. Begin creation of database.
8. Test application with employees and gather feedback.	8.1. Test application with employees in the business. 8.2. Gather feedback. 8.3. Document testing and how it was performed
9. Finalise the development of the application.	9.1 Work on final bug fixes and finishing touches of application. 9.2 Detailed review of the application

	in comparison to my design and user requirements.
10. Write final report and evaluation that contains details of how the project was managed and developed.	10.1 Discuss how the project was managed and developed. 10.2 Discuss how the application could be built on in the future.

Table 1. Objectives and Tasks

With an aim and a list of objectives and tasks a road map of this application begins to form. Following these objectives and tasks will help this project to tick off all the necessary sections required for this project to be successful.

1.2. Dissertation Outline

This dissertation will be split up into a handful of chapters. Firstly, after the introduction will be the literature review.

Chapter two will contain the literature review. This will begin by exploring what research has already been conducted into timetabling and scheduling. This research will help to give this project some initial ideas on what has been created, what could be created and to help develop this project idea further.

Next, the literature review will discuss similar/related applications. This section will investigate other similar timetabling and scheduling applications and what functionality they have and how they have solved different problems. This section will also help to expand and develop the initial ideas proposed for this project.

After this there will be a section discussing the different tools and languages that will be necessary to complete this project. Their advantages and disadvantages, and what role they will play in the development of this application.

Chapter three will discuss what methodologies that will be used to assure this project has a clear development cycle and to assure that all the different parts of that development cycle are explored.

Chapter four will look at how the application is designed, and the research involved into how to create a more user-friendly application. From the flashy front-end to the planned design of the more complex back end and functionality.

This design section will then lead onto the development explanation. This section will explain step-by-step how the application was developed, how the tools and languages discussed in chapter three were used and any issues faced during the development of the application.

Chapter five will show the testing phase of the application. It will show the different levels of testing that were implemented. From testing simple lines of code and viewing the results to testing the full system with the client and getting the final feedback.

Chapter six will begin to round off this dissertation with an evaluation. This chapter will explore and evaluate the different areas of my project and what went right and what went wrong. This chapter will also look at the feedback received from the client and users of the application.

Chapter seven will finish this dissertation with a conclusion. This chapter will look at the possible future of this project. Where this project could go and what work could be done if it was to be taken and developed further. This chapter will also discuss the limitations known at the beginning of the project and any limitations discovered during its creation.

Finally, the chapter will look back at the original aim discussed in the introduction and evaluate whether it has been achieved or not.

Chapter 2. Literature Review

This chapter will discuss the research and work currently being done for scheduling applications as well as the features and functionality of similar applications already in use. This chapter will also compare any tools and languages that may be useful to this project. Research into these areas will help to make a more informed decision on how this project will be developed.

2.1. Similar Work in this Field

Timetabling and Scheduling is necessary to running an appointment-based operation and moving away from paper-based timetabling and scheduling could improve a business's productivity and reduce accidental losses from a mismanaged schedule or timetable.

An unautomated timetabling system could be effective in reducing a business's losses. The client associated with this project could greatly benefit from a system to help with job bookings allowing them to keep track of what jobs they have in the coming week and what jobs have been completed.

Timetabling and scheduling applications are used in many areas of business and daily life. From managing people and their working weeks, similar to this project's goal, to algorithmic based scheduling for things like large logistic operations that many big businesses are involved in. This following review of literature will investigate what problems and solutions exist in this field and what research has already been made into different aspects that relate to this project.

2.1.1. Newfoundland University

There are various types of timetabling and scheduling applications. Some of these applications involve automation. A case study from 2017 looked at the impact of an automated timetabling system on a university in Newfoundland (G. C. W. Sabin, 2017). Since 1978 students from the Memorial University of Newfoundland were registered onto a fully computerised system with the intention of optimizing the universities classroom and timetable resources. The result of this switch to a computerised system resulted in an application that performed accurately and efficiently to decrease expenditures across the university. The paper concludes claiming there should be no reason that the results achieved from the paper couldn't be duplicated at other universities.

To improve the set of results this paper produced, it could attempt to backup this closing statement to prove the validity of this timetabling and scheduling system by testing it at other universities. This paper demonstrates the usefulness of a timetabling and scheduling system for a university, it would be interesting to see how this system could benefit a business of a similar scale.

2.1.2. Mobile and Web applications for Small and Medium Size Enterprises

A paper from 2013 conducted a use case analysis on mobile applications for SMEs (Small and Medium-Sized Enterprises) and how valuable they can be (Kolici, Xhafa, Pinedo, Segui, & Barolli, 2013). The paper also investigated the various challenges that may arise from the implementation of these applications. They used applications that could provide a number of different services and functions and evaluated the results of their usefulness.

One specific part of the paper relates closely to a part of this project. The paper claims that: “Mobile applications for supporting collaborative teamwork are also emerging as ways to support field work and geographically distributed teams such as journalists, researchers and disaster management teams.”. The employees of the company associated with this dissertation’s project may not fit any of the mentioned jobs, but they are often geographically distributed as engineers travel the country to perform service calls. The paper goes on to discuss the importance of awareness when working as a team. The paper mentions how: “members of a team need to be informed on the action carried out by other team members and re-act accordingly…”. This is another key aspect of this project. As mentioned in the introduction miscommunication and a loss of some job tickets and information is an issue that could be solved with the development of this application. Keeping track of information and assuring that all employees are aware of any jobs occurring that day or week could lead to an improvement of the company’s productivity.

The paper discusses the possible issues faced by mobile applications and solutions to negate them. They begin by suggesting the implementation of an event that would automatically trigger in response to a failure or internal error. In relation to this dissertation’s project this could means that if a database error were to occur such as a table being deleted the system would have a contingency plan such restoring to a previous database backup.

The paper also explores the issues of synchronization and keeping data consistent. This means avoiding conflicts such as two users modifying data at the same time. To solve this problem, they suggested notifying the two users who changed the same data and having them decide which of their entries to use. In terms of this dissertations project this could happen if two engineers try to accept a job at the same time. Functionality like this could be implemented where the engineers are notified and decide who the job should be delegated to.

The paper mentions the usefulness of notifications claiming that having notifications that could either warn users of any possible issues that could arise with the system or a notification that requests a user to perform an action are also valuable in assuring the system isn’t either broken or backed up with actions that the user needs to perform. This could be very beneficial to this project as notifying employees of any jobs that have not been accepted or completed will assure that no job ticket is forgotten about.

The paper concludes after analysing several real-life application scenarios and identifying the possible issues that could arise in their implementation. They claim that mobile computing is becoming increasingly important to businesses and enterprises, especially SMEs. They mention how these applications could improve operational activity, efficiency and reduce costs.

To improve the results of this paper, more real-life scenarios could have been tested with different applications. They could have also discussed any other issues the users of the applications dealt with and evaluated what they thought about the effectiveness of these applications in their respective workplaces.

2.1.3. PTIME

In this paper the authors describe using a learning cognitive assistant agent named PTIME. PTIME (Personalized Time Management) is a timetable and scheduling solution proposed in 2005 from the American University of Beirut in Lebanon (Pauline M. Berry M. G.-S., 2011). The paper describes arranging meetings in an office environment as "tedious" and explains how organising a meeting is often a case of several emails proposing, rejecting, counter proposing and eventually agreeing on a time. It goes on to suggest a system that allows users to specify desired timeframes and preferences such as duration and location and then presents users with options that could be satisfactory for the meeting. The authors developed PTIME through a study of different user's calendar habits and the criteria that influenced those decisions. They interviewed a series of workers at different positions in the company (managers, administrative staff and researchers) to see how different roles affected how those workers timetabled meetings. This research influenced an algorithm they developed that would be the engine of PTIME. After developing the application, the authors asked users to rank the different options that PTIME suggested to timetable their meeting for. The result concluded that on average PTIME was able to successfully suggest a time for a meeting that was satisfactory. After conducting personal interviews, users expressed a likeness and a highly significant opinion that PTIME would make scheduling a meeting easier.

Similarly to the Newfoundland University paper, this paper could be even more beneficial if the system was tested at different businesses of varying sizes. This paper demonstrates once again the benefit of timetabling and scheduling for a business.

2.1.4. Conclusion

Timetabling and Scheduling solutions will continue to be developed and improved on with more complex algorithms in years to come. The study from the University of Newfoundland expresses the benefits that can come from timetabling and scheduling by discussing the reduction in expenditures across the university. The paper discussing the possible issues of implementing mobile solutions into SMBs was useful in providing some solutions to issues that had not been considered by this project thus far. Reflecting on the PTIME study may influence how this project approaches fair timetabling and how jobs should be distributed amongst engineers. It also opens up ideas around the future of this project and how distribution of jobs could be automated with an algorithm.

Investigation into these papers and the research that has already been conducted around timetabling, scheduling and applications for SMEs will influence the initial user requirements that will be outlined in Appendix A. This table incorporates some of the initial ideas this project had, and ideas picked up from investigation of similar work in this field. This table will update and change as the project progresses.

2.2. Similar/Related Applications

This section of the paper will discuss similar applications to what will be created in this project. Investigating and reviewing these applications will help to generate some ideas and functionality that could be incorporated. It will also help to see what other applications do wrong and what to avoid. This will then influence the user requirements for this project. It is important to note that these are investigations of these applications at the time of this dissertation being written and are subject to future updates.

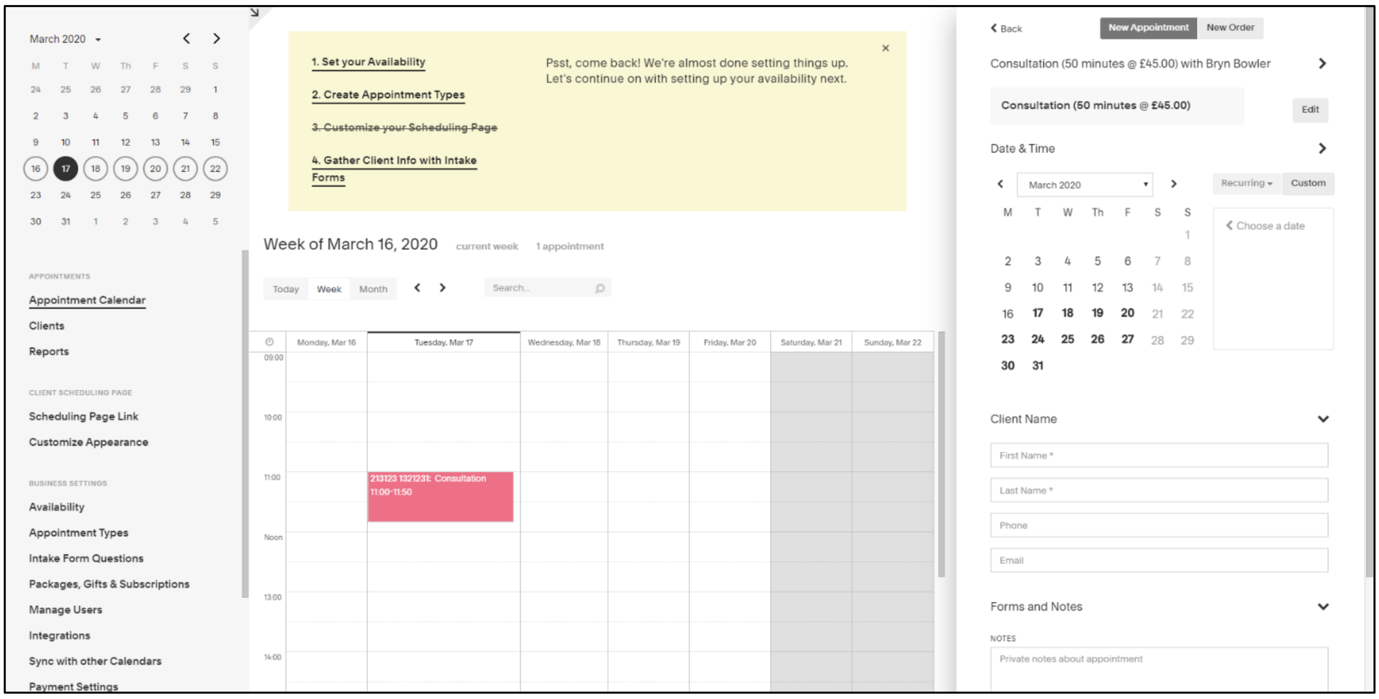
2.2.1. Acuity Scheduling

Acuity Scheduling is a timetabling and scheduling application by Acuity Scheduling Inc created in 2006 (Acuity Scheduling, 2020). Acuity Scheduling strives in several ways. The application begins by suggesting a connection to Google Calendar, iCloud or Outlook allowing users to make the switch to this application easily without losing any appointments they may have made on these other applications. Overall Acuity is fast and responsive when creating and managing appointments. The application is responsive to users, allowing them to drag and drop appointments around the timetable quickly. Acuity also allows for users to share a timetable with other people by simply inviting them through e-mail.

The application offers many features to the user, it also has some interesting smaller functionality too. First off Acuity offers the option to send an email to a client when cancelling an appointment. This could be a useful feature for this project's application as it keeps clients in the know and is quicker than a phone call to cancel and reschedule. Another useful feature is the ability to create a client list. A client list saves the information of any clients that have been previously given an appointment. This feature would become very useful when there are a lot of clients and many appointments to be made as client information can be dragged and dropped into an appointment quickly. The application also allows users to set time off and any hours they will not be available.

As fluid and clean Acuity may seem, it does have some flaws. The biggest one being the layout and overwhelming number of things on screen. Figure 1 shows an example of the Acuity home screen and the abundance of information there is to process. The application can be confusing and frustrating to navigate. The route of this problem could be the lack of colour. The application does a have a nice clean look but it all blends together. Trying to find certain information or conduct certain tasks can lead users in circles. Some of the swiftness of the functionality can be halted by basic and important features that are unnecessarily hidden behind multiple menus. For example, when creating an appointment, you cannot simply set a length of time for the appointment. Initially it seems you can only have appointments for 30 minutes. There is no way on the appointment creation tab to change this. Only after digging through menus will a user realise that you must create 'Appointment Types'. A new appointment type must be created for each different length of time you want an appointment to be, another tedious feature.

Overall Acuity has a lot of the functionality that this project could benefit from and has offered some ideas that could improve the functionality of this application for the client involved. On the other hand, it serves as a harsh lesson in the importance of usability in an application. Acuity is in dire need of a less complex menu design and look. Usability is something this project aims to succeed at alongside solid functionality rather than being left to the wayside.



Psst, come back! We're almost done setting things up. Let's continue on with setting up your availability next.

Figure 1. Acuity Home Screen

2.2.2. Bookafy

Bookafy is another timetabling and scheduling application (Bookafy, 2020). This application has similar basic functionality to Acuity and other timetabling and scheduling applications but has some of its own special features as well. These include the ability to create one-off appointments which are quick and easy to add to the timetable. This application also allows users to create client lists. The application takes a similar approach to appointment creation with their own 'Appointment Types' yet in this application they are explained more clearly. Bookafy also provides a separate screen to view upcoming appointments that just lists every appointment and the information associated with it ordered by the appointment times.

Bookafy also succeeds in providing a clear user layout. Information is shared onto separate pages appropriately and gives the user room to breathe when navigating the application. Even providing the separate appointment view screen previously discussed that display relevant information clearly to a user.

Some downfalls of this application include the inability to share timetables and see timetables other than your own. Another downfall is the cost. Bookafy requires a monthly payment for each user of the application. Bookafy also doesn't allow for much customization of their application. This means that it cannot be tailored to the client that is involved in this application.

Overall Bookafy is a good timetabling and scheduling application. It has solid functionality and a clear layout, but the issue is that it cannot be tailored to a client. It is a basic system that is meant to be used by multiple people for one business. As a result, users are unable to have personalised timetables unless a business wants to pay more.

2.2.1. Google Calendar

One of the most popular timetable and scheduling applications is Google Calendar (Google, 2020). Google Calendar is known for its reliability and wide use. Google Calendar has a lot of the basic features people would want for a calendar application done well. Events can be created and scheduled and extra information that can be added to an event such as a description, images and locations. One feature Google Calendar specialises in is accumulating a range of calendars all in one place. This means that a person's personal and professional calendar can be combined. This is incredibly helpful as it assures that a user's personal and professional calendar don't have events occurring at the same time. Another incredibly useful feature is the "Find a Time" feature. This feature allows users to add people to a group for an event and then, using this tool, find an empty slot on everyone's calendar to schedule it for. Another feature that distinguishes Google Calendar is the ability to sync it with Gmail. This allows for users to easily set up calendar events and reminders from information sent in emails. The main draw of Google Calendar is the fact that it is a solid timetabling application that puts all a user's calendars in one place.

Some disadvantages of google calendar in relation to the client's current issues includes the inability to assign calendar items to other users. Understandably, users are not currently able to add calendar events to other users' calendars without their permission. An aspect of this dissertation's project is that engineers' calendars will be updating with new calendar events often as they are assigned. Another disadvantage of google calendar is that it does not allow users to trade calendar events without setting them up all over again another initial idea of this project.

Overall Google Calendar is a great application for timetabling and scheduling. It provides many useful features to users and has some interesting unique features. Similarly, to the other applications reviewed it lacks the customizability to tailor it to specific business' needs. Which is understandable as it is supposed to be an application that broadly appeals to anyone, but the aim of this project is to create a tailored application for the client involved.

2.2.5. Conclusion

The client involved in this project currently uses Outlook Calendar to manage and plan their service visits. The business gets by using this system, but it could be improved. This application will have similarities to the system currently used by the client but will be more refined to the client's business needs. Creating and scheduling jobs will be very similar but from research into these other applications a feature they lack, that this project will provide, is the ability to set jobs that are assigned to other users calendars and the ability to trade jobs with others users who's calendars will update accordingly.

This section discusses many interesting features of similar applications that may be good to adapt for use in this application. This would include features such as a simple and intuitive design. User experience is an incredibly important aspect of developing any application, but for a timetabling and scheduling application where there can sometimes be an abundance of information, clarity is key. From viewing the timetable to the creation of timetable events understanding the information presented is very important. With the possibility of the client involved in this project hiring new staff it is essential that a user can pick up this application and understand how to use it in a short amount of time.

Another key feature that this project must get right is the basic functionality of creating events and adding information to them. The application must make it easy for users to add information to jobs. This is where this project will steer away from the design of Acuity to approach this in a more simplistic way.

Some features that could be beneficial to incorporate into this project include the responsiveness that is apparent in all three applications. The ability to drag and drop appointments will increase this applications usability. Another feature that this project could benefit from is a client list. The client involved in this project has many clients of their own that they often revisit to provide maintenance and to install more kit. Having a client list would make creating appointments even easier. There may even be the possibility of implementing Google API in the project to incorporate google calendar dates that have already been created. Google API could also be used to embed a google maps feature which will use the addresses added to job forms to easily map out a route for users to take to get to job location. Finally, this project's application will be given to the client for free at the end of the project meaning they won't be paying monthly fees or one-off payments to get a timetabling and scheduling application.

Three key features that differentiate this project from applications that are already out there includes the focus on an application for the specific client, the ability to trade jobs/events with other users and being able to create jobs that are assigned to certain users. Firstly, this project aims to create and application that fulfils the specific needs of this client. This means having the client explain what specific features they want to include in this application, explaining how they will be used and optimising them for the business. Compare this to google calendar which has many great features, but a lot of these features will not be used by the specific client this project is working with. The second point that differentiates this projects application is the ability to trade jobs with other users. For this project an initial idea is the ability for engineers to trade jobs with each-other and manage their own working weeks. In the applications mentioned above the two events/jobs would have to be deleted and re-created in the separate calendars. Finally, what differentiates this project is the ability to create events/jobs for other people. Although you can add guests to some of the applications discussed previously you cannot create events for other people. The initial idea for this this project will involve a receptionist creating the jobs which will be assigned to the engineers.

Discussed previously at the start of chapter two were suggestions of algorithm-based scheduling that involved mathematics to create a much more streamlined timetable. In terms of this project this could be implemented by creating an algorithm that schedules jobs that are in similar locations to each-other or have similar timeframes. Despite this being an interesting idea, the timeframe of this project would limit the ability to do something of that kind of scale considering the mathematics and logic that would be involved. There may be a possibility to implement something on a smaller scale such as jobs that are assigned to specific engineers based on skillsets that they exceed in. For example, if an engineer prefers to do installation jobs, more instillation jobs could be assigned to them compared to repair/servicing jobs.

An updated set of functional and non-functional requirements can be seen in Appendix B.

2.3. Research into Software Development Tools, Programming Languages and Database management Tools

Before starting the development of this application some tools and languages were investigated and compared against each other and by to conclude if they will be useful in achieving the aim and objectives of this application. Choosing the best tools and languages for this project will make its development easier.

This investigation will showcase tools and languages that are necessary for this application and tools and languages that could improve this application beyond its basic requirements. One of the languages that will be necessary this project is a mark-up language. This language will be used to construct the skeleton of this application which functionality will then be added to. This project will also need a programming language that can handle logic. This language will help in the development of certain functionality and any calculations that might be necessary. A database and database management tool will also be essential for this application as it will store data about upcoming and completed jobs as well as other things such as login information for the different users. A tool or management system that can help with the setup and hosting of this web application will also be very useful. Finally, an IDE (Integrated Development Environment) will be chosen to write code and create a logical file structure.

2.3.1. HTML

HTML (Hyper Text Mark-up Language) was created in 1990 by Tim Burners-Lee. The language wraps text into tags. This then tells a web browser how to display text, images and other forms of multimedia on a webpage (Rouse, 2019).

The advantages of HTML include its wide use and reliability. With it being used on roughly 88.8% of websites. (W3Techs, 2020) It could be considered as the essential first building blocks of most web sites or web applications. HTML is also known to be consistent over multiple different web browsers and devices with its core features being useable on all common web browsers. HTML is also considered to be one of the easiest languages to learn as it is a mark-up language and uses very little logic (Ten Ton, 2019). This means that HTML is effective at building a skeleton of a web application which can then have functionality added to it through other languages.

HTML's reliability and ease of use does come with negatives though. These mainly being how limited it is. As previously mentioned, HTML, especially for a web application with lots of different functionality, can only provide a skeleton for further code to be added to. HTML is a mark-up language meaning that it does not include any logic. This means no calculations or if statements which are essential for creating functionality.

2.3.2. HAML

HAML (HTML Abstraction Markup Language) is another mark-up language that works similarly to HTML but tries to simplify the code that is written. For example instead of having two H1 tags with text in the middle written like: “<h1>Hello!</h1>” the code instead is written as: “h1 = Hello!”.

An immediate advantage is the reduction in the amount of code that needs to be written for this application. Without the need for opening and closing tags, this language could make code less tedious to write and read.

A negative of using HAML is that it is more suited for applications that use ruby-on-rails and not as commonly used as HTML. Using HAML could be useful to keep mark-up pages looking more simple and readable but if another person were to pick up this application and use it in the future they may have to learn HAML as a new language if they wish to make some changes to the mark-up.

2.3.2. Javascript

Javascript is a coding language that incorporates logic. Logic allows a coding language to do calculations and statements which are necessary to add functionality to web pages and web applications. Furthermore, Javascript is a language that allows web pages to be more interactive by accessing and modifying content and mark-up used in a web page while it is being viewed in a browser. Javascript specialises in 4 key areas. One of these areas is access to content. This means that Javascript can co-operate with languages like HTML to select elements on a web page easily. As a second point it can then modify this content to add or change attributes of it. For example, changing the size or font of a header or paragraph text. Third and probably most important element of Javascript is its ability to add interactivity and more complicated functionality to a web page. For example, simple things like a menu animation to more complicated functionality like a mortgage calculator that could take information from a from and calculate necessary re-payments for a user. Finally, similar to the last point, Javascript can react to events on a webpage. Button presses can trigger events that could hide information on a page and display something else without the page needing to refresh (Duckett, 2014).

Some advantages of Javascript include its speed which is ranked highly compared to other languages because most functionality is unhindered by calls to back end servers unlike languages such as PHP. Another advantage is Javascript’s interoperability. Javascript exceeds at its ability to cooperate with other languages with ease such as HTML and PHP. Interoperability with these different languages extends its range of functionality much further. It’s simple insertion into web pages means that it can be used to add functionality to the HTML skeleton previously discussed. Javascript is also very widely used. An article from medium.com claimed that “There are over 1.6 billion web sites in the world, and JavaScript is used on 95% of them (1.52 billion web sites with JavaScript).” (Elliot, 2019). Javascript also benefits from the Node.js framework which allows the language to be used for server-side and backend programming (Mosh Hamadani, 2018).

Some disadvantages of Javascript include its tentative consistency over different browsers. Sometimes different browsers approach parts of Javascript code in different ways. This means that the application should be tested in the major browsers in the testing phase to assure all code is consistent.

2.3.3. DART

DART is a programming language developed by Google created to build applications. It was launched in 2011 but was only really used by google associates until 2017.

Some positives of DART are that it is a new-generation language which is considered to be one of the fastest languages developed. The language is backed by google and has many features used by modern languages. These include static typing which is a method of programming which allows for variables to be used without being declared (Bolton, 2017). DART also has some well-constructed frameworks. This includes a testing framework that can conduct three of the four levels of testing which will be discussed in the testing section of this project (DART, 2020).

Despite DART having some solid frameworks and packages to improve it the language has much less than others such as Javascript which has so many due to the length of time it has been around. DART is also unable to code server-side programming. Finally, since DART has only really been gaining attention since 2017 there is only a small amount of documentation and online forums with problems and solutions. DART is also considered interoperable as it can establish connections to databases and has a library that can incorporate HTML (DART, 2020).

2.3.3. PHP

PHP is a free server-side scripting language that can be embedded into HTML. Despite its wide range of uses, for this project it would serve as a way of managing the web application on the server-side. This includes communicating with a database to display information on a webpage and taking information from a front-end and putting it into a database. Some positives of PHP include its large usage and support meaning if any issues arise during this project involving PHP there is a high likelihood of a solution being found through consulting internet forums such as Stack Overflow.

Another advantage of PHP includes its ability to communicate with servers and pull data from databases quickly and easily. Another advantage is its interoperability which allows for it to work easily with other languages that have already been mentioned. Another positive of PHP is that it will work on all browsers as it is a server-side language and code is executed on the server. In relation to this project this will help to have jobs and the data associated with them be stored in databases which can be pulled to the front end and organised into the schedule.

A disadvantage of PHP is that it is open source meaning that the code is openly available for people to see. This could allow for anyone to identify bugs or weak areas of the application and exploit it. This leads into another negative of PHP which is its large amount of vulnerabilities (phpsecurity, 2017). In this project security is a factor to be considered as it is important that the application is available for use 24/7 and some of the data stored may be sensitive.

2.3.4. AJAX

AJAX (Asynchronous JavaScript and XML) is a technique that uses Javascript performing similar functionality to PHP. This sub-section of Javascript provides a connection for data to be passed between the front-end of an application and the database. Compared to PHP, AJAX allows for a more fluid flow on an application. The asynchronous aspect of AJAX means that data can be displayed and changed without having to refresh a page. A negative of AJAX is that it may not work on certain web browsers that don't support Javascript. It is important that this application works on as many devices and platforms possible so anyone can access and use it.

The advantages of AJAX include that it can update data from a database server live meaning that a page doesn't have to be reloaded to update any data displayed on it.

The disadvantages of AJAX include that it cannot be used without Javascript. AJAX also struggles with some issues related to browser incompatibility (tutorialspoint, 2020) (Panchal, 2015).

2.3.4. MySQL

MySQL is an open source database management system that uses SQL. SQL is a language used for managing data in a database. MySQL could be used in this application to store information such as job information.

An advantage of MySQL is that it makes managing a database simple and intuitive. MySQL is also considered a secure database management system as security features come included in the installation (Microsoft, 2017). These include user & privilege settings which allow an admin to manage what areas of the database different users can access. It also includes connection encryption which ensures all information passed to and from MySQL is encrypted. Another security feature is database logging which keeps track of what changes are made to the database by which users at specific times. This helps to identify possible issues or malicious behaviour (Dobrzanski, 2013). MySQL is also considered robust and quick at processing requests and queries claiming to be able to handle over 50,000 simple queries per second on a standard issue PC (Vandim Tkachenko, 2008).

Some disadvantages of MySQL include the fact that it was designed for speed. This means that it is limited in some SQL capacities and functions. These include restrictions on stored functions, views, subqueries and more nuanced areas of SQL (MySQL, 2020). After researching these, with the requirements collected from the client, it is unlikely that these restrictions will affect this application as MySQL would only be used for basic storing and calling of information.

2.3.5. Content Management System

A content management system is a software application that allows for the creation, editing, organizing and publishing of content. Content management systems simplify many of the difficult steps that are required for creating a web site or web application opening up opportunities for millions of people to create and manage their own websites. Some examples of content management systems include WordPress, Drupal and Magento.

In relation to this project, some advantages of a content management system include the fact that it would reduce the initial workload of setting up certain aspects of the application as they come as standard with a content management system. These include the basic layout, user accounts with log in access and basic security. Another advantage of a content management system is the wide variety of extensions and plugins that can be downloaded and added to a website. These extensions and plugins can be edited with code to introduce new features to accommodate for what is needed by the user requirements. Content management systems are also able to connect and interact with database management systems such as MySQL meaning a backend could be setup. Finally, a huge advantage for using a content management system for this project is that the client already uses WordPress as a content management system to store information and guides on bug fixes and solutions to common issues with their jukebox software. This means that not only can this project build off the client's current website, but it also means that the employees already have a basic understanding of how a content management system works.

The main disadvantage of using a content management system such as WordPress compared to developing and application from scratch is its limitations. Developing more complicated functionality on a content management system such as WordPress can involve a lot of hurdles and working around certain blockades that the site poses. For this project the implementation of a feature that would allow for one user to post calendar events on another user's calendar many be difficult considering how the site manages different user accounts. Another negative is that WordPress requires a subscription to allow more flexibility in its system such as allowing the setup of a backend.

2.3.6 WAMP Server

WAMP Server is a windows specific software stack which combines Apache Web Server, MySQL Database and PHP. The apache component of WAMP works as a web server allowing users to access the website or application. MySQL works as the database which the site can add information to or pull information from using the dynamic programming language PHP.

A huge advantage to WAMP server is how easy it is to configure. Most of the work setting up the database and apache web server is handled for the user by WAMP saving time that could be spent on improving the functionality of the application itself.

Some disadvantages of WAMP server includes the exclusivity of it only being available on windows. This means that the web server can only be launched and worked on via a windows machine. This however does not mean the website is only accessible through windows devices.

2.3.7. Visual Studio Code

To write the code needed for the application to work an IDE (Integrated Development Environment) was chosen. Visual Studio Code is a free IDE developed by Microsoft to aid in software and web development.

Visual Studio Code benefits from simple folder and file management. It also benefits from having a large selection of different extensions that can be added to improve the user experience such as error handlers, tag closing extensions and more. Visual Studio Code is quick and supports many different languages.

A drawback of Visual Studio Code is that it doesn't come with debugging tools for all languages and therefor users must browse extensions or plugins to find them.

2.4 Chapter Conclusion

Shown in Appendix C are tables comparing different aspects of the tools and languages reviewed.

After reviewing several different development tools and languages this project could use to develop the application, the best approach may be to create the application using WAMP Server and a combination of HTML, Javascript, AJAX, PHP and MySQL.

HTML was chosen over HAML due to its wide use, reliability and ease of implementation. Choosing HTML would also make it easier for this project to be picked up by other developers in the future and worked on without them having to learn a whole new language.

Javascript was also selected over DART due to it's wide use and amount of support offered online. The amount of forum and documentation related to Javascript greatly outweighs that of DART. This may improve the speed of development for this project.

WAMP Server was chosen as it allows for a simple, configurable web server and database to be setup for the application. HTML was chosen due to it being the most commonly used mark-up language and its simplicity. PHP will serve as a data connection between the front-end and the database. AJAX could be incorporated as it is easily implementable into and would make the application more dynamic and asynchronous.

Finally, Visual Studio Code was chosen for its simplicity and it's easy to use folder structuring.

With the tools and languages investigated, the user requirements were updated to show what had been changed as a result and what can now be achieved. This can be seen in Appendix D.

The upcoming chapter of this project will discuss and compare different project methodologies and how they could improve the development of this project. It will discuss further how user requirements will be gathered and how these different functionalities that will be coded with the tools and languages discussed above will be prioritised.

Chapter 3. Project Methodologies

Researching project methodologies is important for the creation of this application as it will promote good development practices such as planning, structuring and managing the different aspects of the application's development. A Standish group study in 2009 claimed that only a mere 32% of projects are delivered on time and on budget. A paper on software development methodologies went on to say that "In many cases, the failure is the result of either not using a methodology or using the wrong methodology" (Whong, 2013). A good methodology for this project will keep a focus on the deliverable and its purpose to the client by prioritising user requirements and how they are implemented. There are two different methodologies and a method that will be investigated and compared in this project to determine the best way to manage and develop a quality deliverable.

3.1. Agile Methodology

Agile methodology works around the idea of sprints. Sprints are essentially small development cycles that involve the planning, designing, building, testing and reviewing of certain user requirements and functionality of a project. Some principles of this methodology include the idea of regular co-operation with a client to provide feedback on features and functionality of an application, the implementation of user requirements even late into an applications development and regular client interaction and testing of new developments. Figure 2 shows a diagram of the agile methodology.

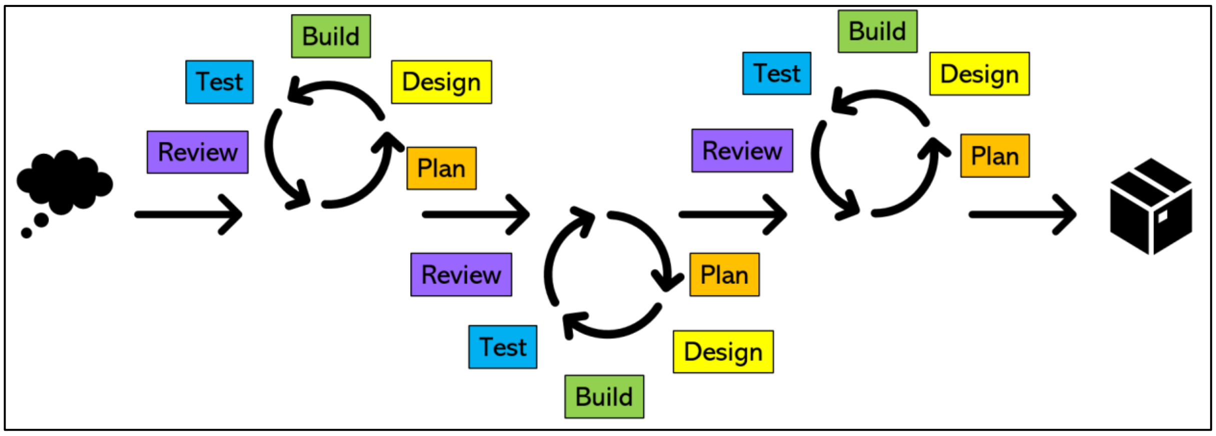


Figure 2. An Example of the Agile Methodology

An article from ObjectStyle discusses the success stories of the agile methodology. They explain the success achieved by Cisco claiming that there was a 40% decrease in critical or major defects of their deliverables as well as products being delivered on time (Krush, 2018). Compared to traditional development agile methodology may reduce the risk of an unsatisfactory deliverable in a couple of ways. Firstly, the client provides regular feedback on the development of the application and suggests new user requirements accordingly. This is useful as if the project were to be developed over the space of 6 months with one set of user requirements and no other contact with the client the final deliverable may not be what the client wanted or envisioned. Another positive of agile is that development is split into small time frames allowing the project to be moulded as it is in production. This means that issues that may arise in development that clash with the user requirements can be discussed with the client to either find and alternative approach or prioritise a different user requirement completely. This avoids development of the application being halted if there is an issue that is especially difficult to resolve. Compare this to the waterfall methodology where if a user requirement is requested and time is spent on the design and how it would be incorporated into the application it is expected that the feature will be implemented despite the possibly time-consuming issues that may arise in said incorporation.

Agile methodology also directs towards a higher quality deliverable. It does this in a couple of ways. Firstly, it encourages the regular testing of new features and functionality not only with developers but with the client themselves. Actual user testing can be valuable to the development of an application as it can demonstrate the quality of UX (User Experience) which can be harder to test for with the developers of the application. Regular testing can also mean that bugs or issues can be identified and fixed early meaning that these issues don't reside in the application till later down the line when they could have a bigger impact and be harder to fix.

Some negatives of the agile methodology may include a projects constant need of attention. Firstly, weekly meetings and reviews with a client could result in a very high-pressure environment where a significant development for the client to test and give feedback on must be produced every week. This may result in certain user-requirements being rushed and developers feeling fatigued, the short development cycles are named sprints for a reason. Comparing this to the waterfall methodology which dedicates much more time to the design and how the core features and functionality of the application could be implemented. This rush may also result in too much focus on big features and functionality leaving important, broader aspects of the project such as UX design, bug fixing and front-end design without much attention.

3.2. Waterfall Methodology

The waterfall methodology splits the development of the application into 5 different steps. As seen in figure 5. These different steps take place over a longer period. First, requirements form the client are gathered and given to the developers. Then the developers go on to design the application's front-end and back-end. 3rd, the developers begin to implement and code the features and functionality. They then move onto verification which involves the testing of the application. Finally, the developers release the application and focus their attention on supporting and maintaining the application to improve its longevity.

Compared to traditional development the waterfall methodology pushes to ensure that the original user requirements are the focus of the deliverable. This means that the project does not drift away from its original intention. The application, through a waterfall methodology, will prioritise the original problem it was created to solve. This is an advantage over other methodologies such as agile as it is possible that extra features and design changes that are common with an agile methodology could impact the quality of the applications main function.

Another positive the waterfall methodology has is its time spent on design and lack of surprises. With all the user requirements for the application laid out from the start the waterfall methodology allows developers to spend more time thinking about the intricacies of design and how different parts of the application will interact. With the waterfall method there is also a lack of surprises. This means that, unlike the agile methodology, sudden design changes or new features that need implementing are not thrust into the development of the project which may lead developers to make poor, rushed design decisions to implement them. With the waterfall methodology everything is planned out from the start. A diagram showing the different stages of the waterfall methodology is shown in figure 3.

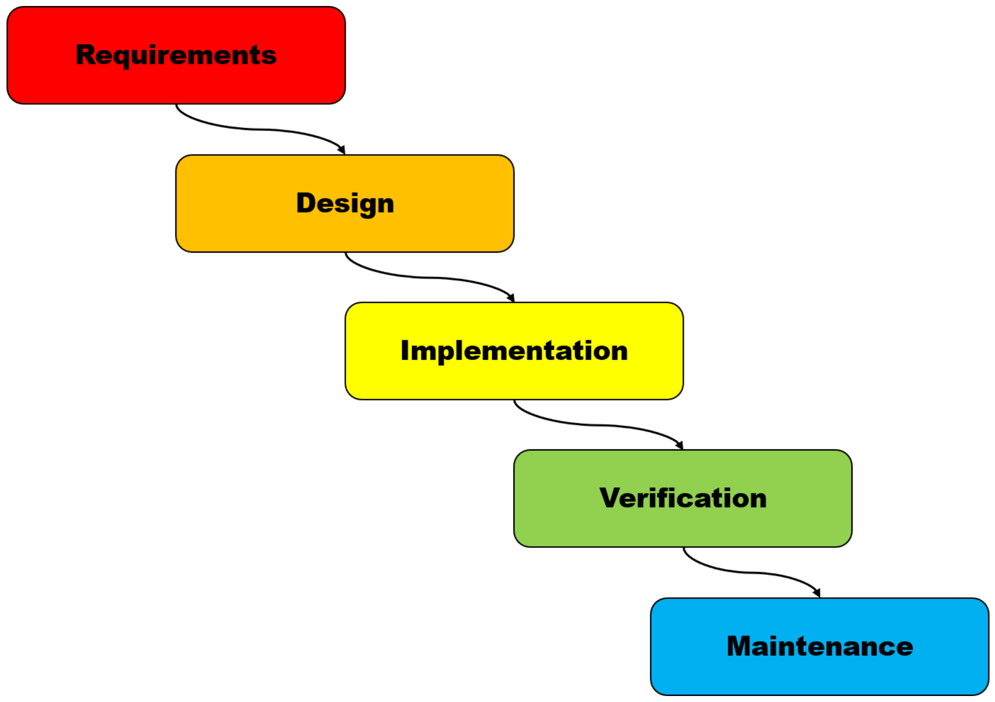


Figure 3. Example of the stages of the Waterfall Model

When discussing the issues of the waterfall methodology one issue is how late the testing stage is into the development of the deliverable. Leaving testing and vital feedback till the end of the project development cycle means that identified faults are harder to fix (Petersen, 2009). Compare this to agile development where testing is pretty much ongoing throughout the whole development providing constant feedback and catching bugs early into development making them easier to fix. The huge disadvantage that burdens the waterfall methodology could be its lack of flexibility and possibly vague or underestimated user requirements. If a development teams begins work on an application with vague user requirements this could result in a deliverable that the client did not envision. In the same vein, underestimating requirements or taking on too many requirements in the initial planning could lead to developers being overloaded with requirements that could not be achieved in the given timeframe or budget (Petersen, 2009). This could then result in the project being unsatisfactory or of little worth to the client. Finally, the lack of flexibility involved in the waterfall model means that if a client has a feature they desperately want in their application that is late into development it would be difficult for developers to implement it because it's interaction with other parts of the application hasn't been accounted for in the initial design.

3.3. The MoSCoW Method

The MoSCoW method is used to prioritise user requirements by splitting them into 4 categories of Must, Should, Could and Wont. These categories help to keep the project focus on the main functionality of the deliverable. Requirements that are a necessity to the final deliverable and therefor the highest priority are placed in the "Must" section. User requirement that are important extras to the main functionality are placed in the "Should" section. Requirements that are desirable or bonus features are placed in the "Could" section.

Finally, requirements that are simply impossible to implement under the budget or timeframe are placed in the “Wont” section (Khadija Sania Ahmad, 2017).

3.4. Conclusion and Chosen Method

Despite the many positives of the agile methodology the time allocated for this project will make it difficult to collect user requirements, design the features discussed, build them, adequately test them and finally sit down for review with the client multiple times. The time this project would spend on assuring the agile methodology is correctly executed would likely hinder what this project could produce for the client compared to the waterfall model. With the waterfall model one solid development cycle can be achieved, more time can be allocated to the design, development and testing phase of this project and the timeline of the development of this project will be more linear and understandable. In collaboration with the waterfall method the MoSCoW method will also be used to help prioritise the user requirements that will be gathered from the client.

The next chapter of this project will look at how the user requirements for this project were gathered using the waterfall methodology discussed and how the user requirements were categorised into the MoSCoW method.

Chapter 4. Design and Development

4.1. Design

4.1.1. User Requirements

Gathering user requirements was an important aspect of not only the waterfall methodology but also an important insight into what necessary features and functionality this application should provide from the client’s point of view. To gather these user requirements an interview was set up with the client. With some basic knowledge of the application and what it intended to achieve the interview questions expanded on the suggested functionality and allowed the client to be more specific on how the application could be tailored to their business. Appendix E shows proof of an email exchange to confirm the client’s involvement with the project. Appendix F shows the interview questions and Appendix G shows the transcribed interview. From this interview and the research already conducted, functional and non-functional requirements can be established. These can be seen in Table 2 below.

These outlined functional and non-functional requirements could then be filtered into the MoSCoW method. As mentioned in the methodology chapter this will help to prioritise what is important to the deliverable of this project. The user requirements sorted into the MoSCoW method can be seen in Table 3.

	Functional Requirements	Non-Functional Requirements
General	- Login Functionality - Different Users - Ability to add more to timetables than just jobs (Small Tasks) - Inn-App Notifications - Job Calculating Algorithm - Google Calendar API - Client List - Database/Application Error Triggers	- Free - Easy to Use - Easy to Navigate - Mobile Accessible - Well-presented information
Jobs	- Ability to create Jobs - Jobs accepted through to the Job Board - Google Maps API for Job Creation - Email to notify sites of cancelled jobs - Engineers can swap jobs	- Easy Job creation
Timetables	- Ability to view personal Timetables - Ability for a Manager to see all Timetables - Responsive Design - Separate List View for Timetabled Appointments - Asynchronous Design	- Easy to read and understand information
Database	- Can store ticket information - Can pass information between different tables	
Restrictions	- No restrictions on Job Acceptance	
Job Boards	- Ability to accept Jobs - Accepted Jobs are automatically assigned to Engineer Timetables	
Statistics Page	- Track Statistics from Completed Jobs (Miles Travelled, Time on Site) - End of the Month Review	

Table 2. Outlined Functional and Non-Functional Requirements

When categorising the user requirements into this MoSCoW table it was important to keep in mind the original goal of this application and the problem it was created to resolve. This being to help timetable and structure a more coherent working week for the Engineers of AVTechincal Services. Therefore, the most vital functionality to solving that issue was placed in the ‘Must’ section. In the ‘Should’ section is ways in which the main function of the system could be improved on once the main functionality has been achieved. The ‘Could’ section focuses more on added extras which will improve the application but stray away from that initial goal. Finally, the ‘Wont’ section contains elements of the application that were either rejected by the client or were considered not feasible due to time and the more significant core features needing to be prioritised.

Must	Should
▪ Different User Accounts ▪ Ability to View Personal Timetables ▪ Jobs accepted through Job Dashboard ▪ Ability to accept Jobs ▪ Free ▪ Easy Job creation ▪ Accepted Jobs are automatically assigned to Engineer Timetables ▪ No restrictions on Job Acceptance ▪ Easy to Use ▪ Easy to Navigate ▪ Timetables must be easy to read ▪ Ability for a Manager to see all Timetables ▪ Well Presented Information ▪ Database can store Ticket Information ▪ Database can pass information between tables	▪ Engineers can swap Jobs ▪ Login Functionality ▪ Inn-App Notifications ▪ Mobile Accessible ▪ Responsive Design ▪ Separate List View for Timetabled Appointments ▪ Asynchronous Design
Could	**Wont**
▪ Security Features ▪ Statistics Page ▪ Google Maps API for Job Creation ▪ Ability to add more to timetables than just jobs (Small Tasks) ▪ Track Statistics from Completed Jobs (Miles Travelled, Time on Site) ▪ End of the Month Review ▪ Google Calendar API ▪ Client List ▪ Database/Application Error Triggers ▪ Email to notify sites of cancelled Jobs	▪ Email Updates ▪ Job Calculating Algorithm

Table 3. User Requirements arranged into the MoSCoW Method

Now the user requirements have been clearly listed and prioritised the next step is to look at how the user will interact with these functionalities. This can be shown in a use case diagram.

4.1.2. UML Use Case Diagram

Use case diagrams are diagrams created to show how a user interacts with a system by showing different interactions, represented by use cases and the different actors, entities outside of the application, that are involved and how the system reacts to these interactions. Use case diagrams represent the application in the centre with a box and the primary actors, users that use the application on the left side. The right side is for secondary actors which are used to help the application complete its goal.

The diagram in figure 4 shows all the actors and steps involved in completing different tasks on the system. The lines connected to the users shows the actors show what part of the process they are involved in. The dotted lines show what other actions are included as part of the initial action. This diagram shows how a system works and the steps involved with its different interactions much simpler and in a much less convoluted way than simply explaining with words.

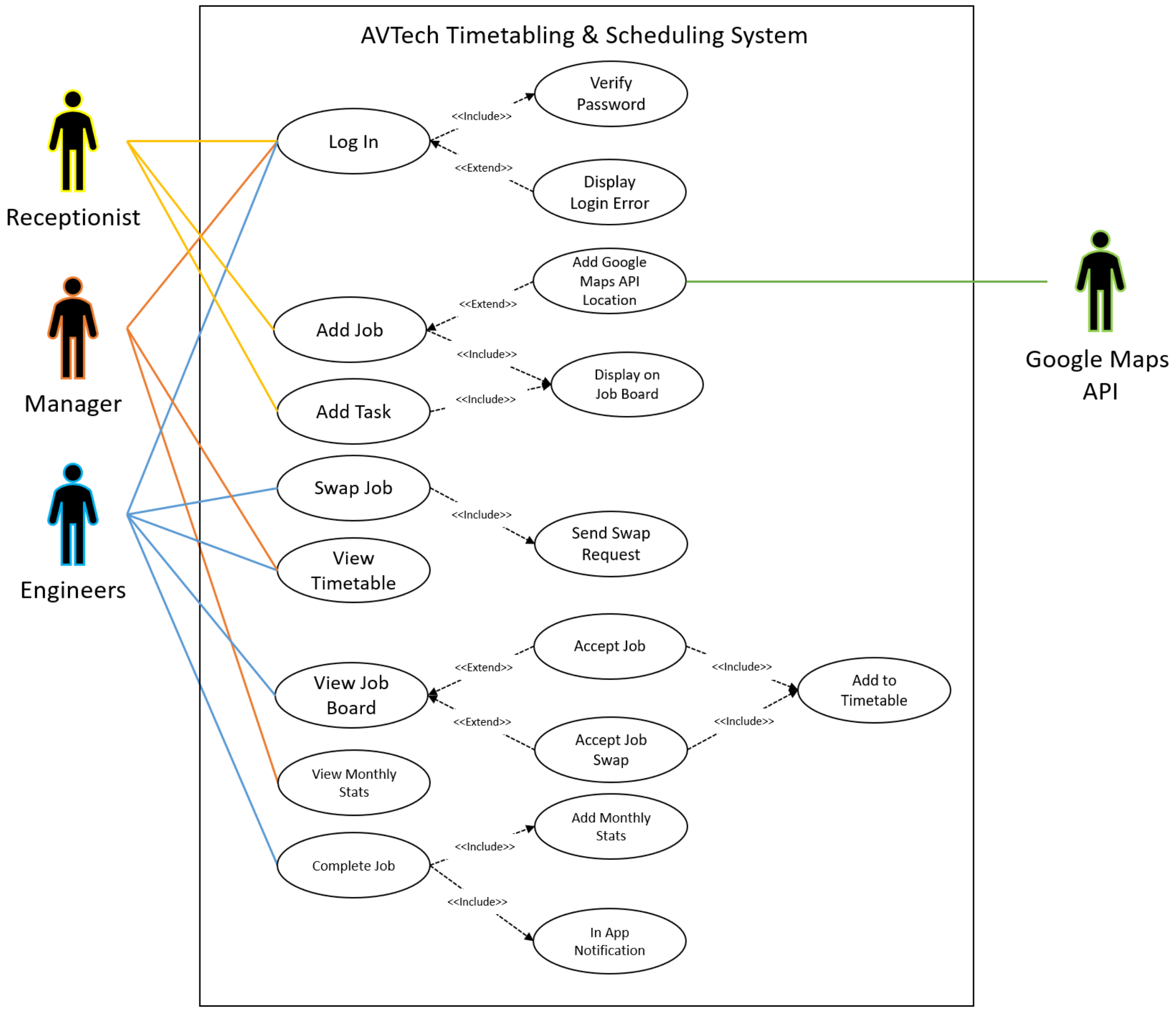


Figure 4. UML Case Diagram showing the AVTech Timetabling & Scheduling System

4.1.3. User Flow Diagrams

User flow diagrams are another way to help get a clearer idea of how an application will be structured. A user flow diagram shows how a user moves through a system, what pages and

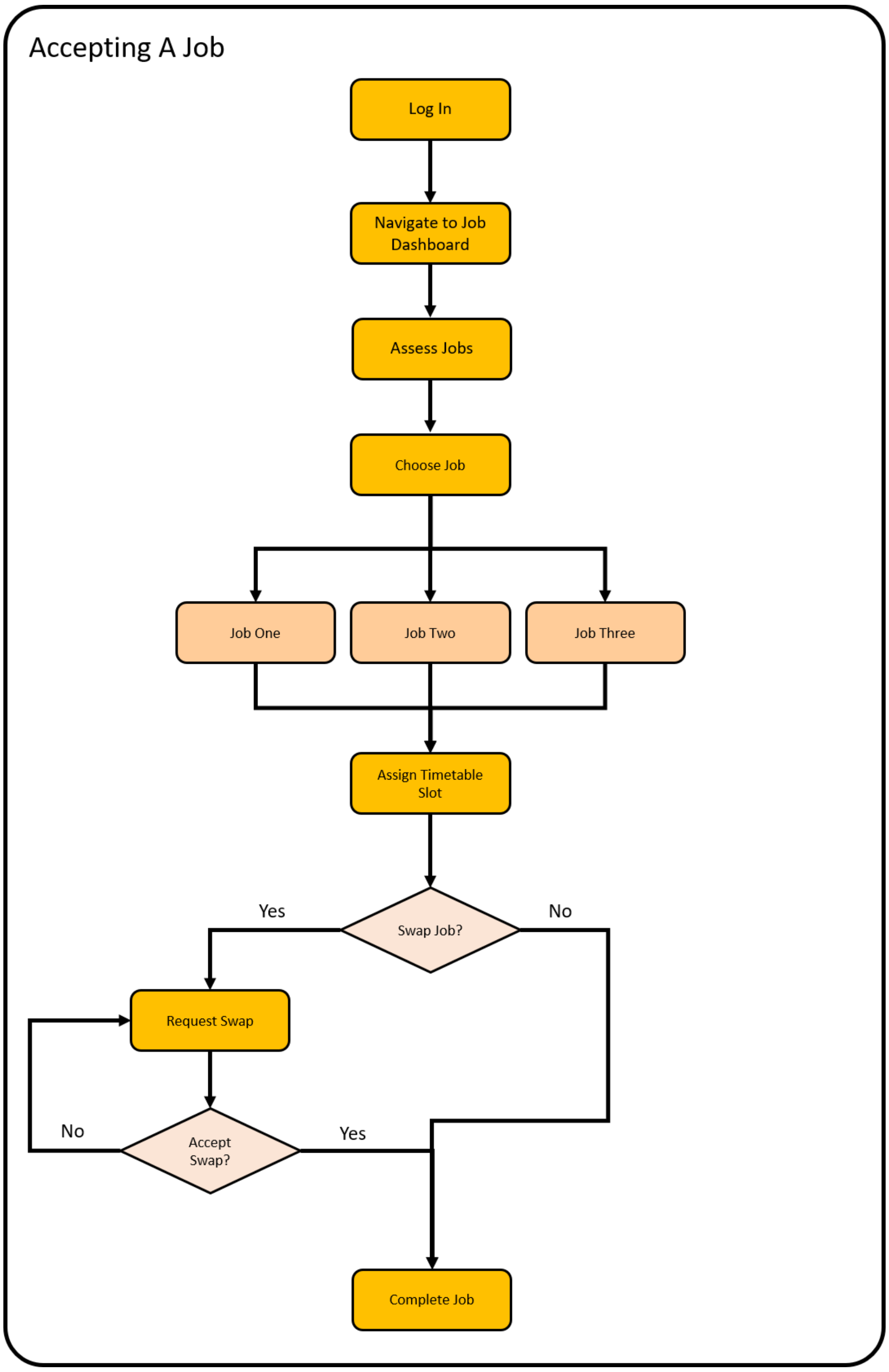


Figure 5. User Flow Diagram depicting how an engineer would accept a Job.

functionality will lead them where and can ask questions about how a user conducts different tasks in the system.

This diagram in figure 5 shows the steps involved with an engineer accepting a job through this application. This information will help map the functionality required for the engineer front-end of the application.

4.1.4. Good Design Practices for Front End Design

Before creating some mock-up front-end designs for this application research was done into web design principles. The goal of researching this information was to improve the look and usability of the application's front end.

In a section of the book Future-Proof Web Design (Dawson, 2012) the author discusses responsive web design. He mentions how because of the introduction of many new devices that can access web pages and web applications the past mentality of a "one size fits all" model is no longer viable.

In relation to this project this could be applied to making the application usable on mobile devices as the users will be on the move. Making the website available on mobile devices would require the system to have responsive web design. This means that elements of the webpage would have to change their size to fit the screen they are displayed on.

In his book, Dawson talks further and mentions three aspects of responsive design.

- The physical space users have, relative to design
- Flexibility regarding portrait and landscape orientation
- Fundamental layout options based on viewport sizes

To accomplish this different CSS techniques were suggested. These included percentages used for width and height. An element with a 50% width tag would only take up 50% of the screen size no matter how big or small. Tags such as VW and HW could also be used, these were measurements of different elements that would change depending on the size of the screen. With these techniques combined the web application should be usable on many devices.

Initially, when the idea was pitched to the client some designs were mocked-up to give a basic idea of how the application would look and work. These can be seen in Appendix H.

Once the user requirements had been collected and this design research had been completed a more detailed front-end redesign of the application was created. This design included all the pages discussed in the user requirements that were not present in the initial design such as the job dashboard board and statistics page. These designs also took into consideration the responsive design techniques discussed in this chapter. These designs can be seen below in Figures 9, 10, 11, 12, and 13.

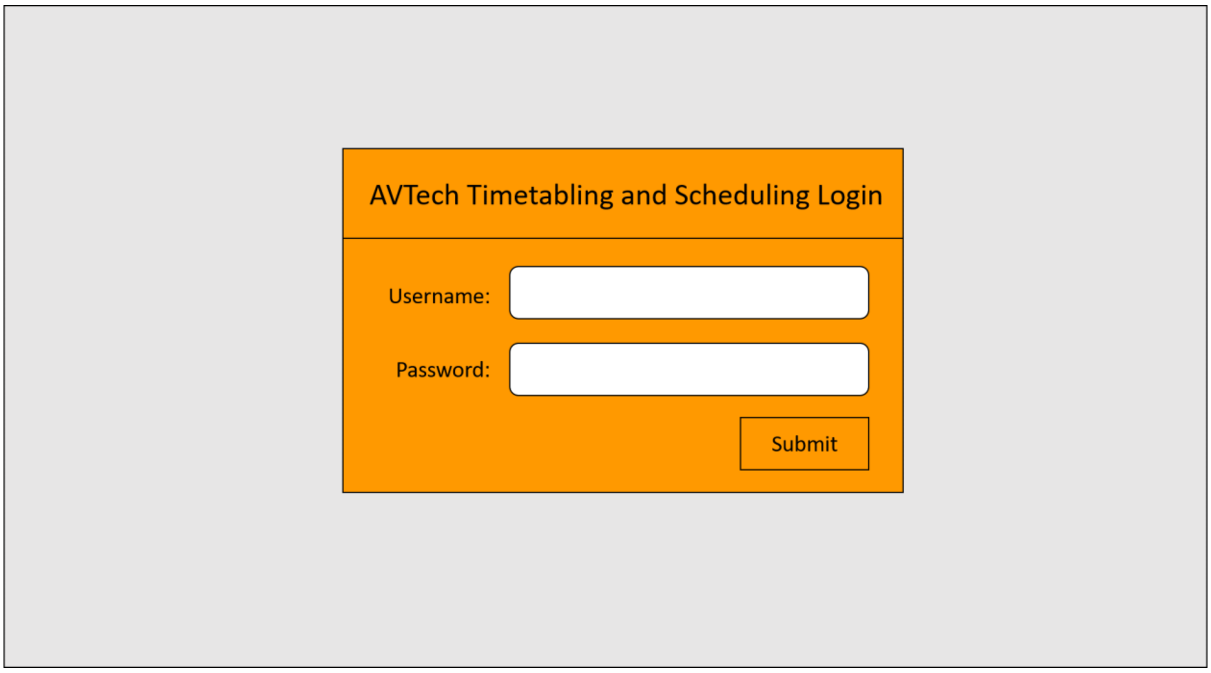


Figure 9. Login Page Design

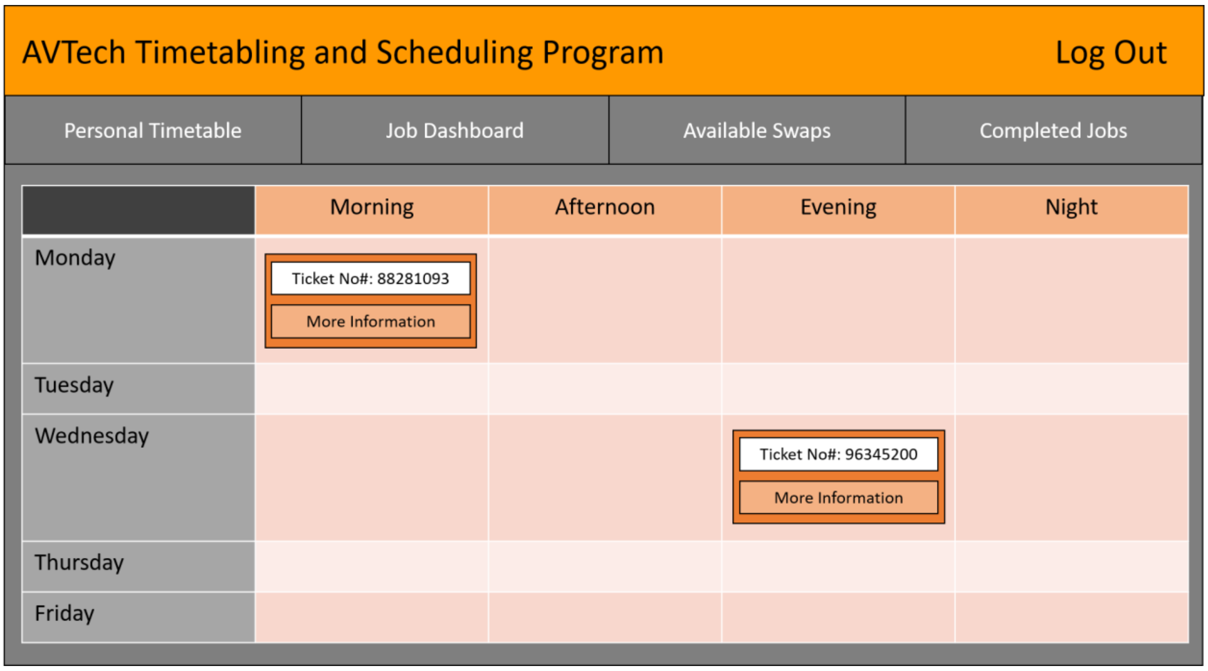
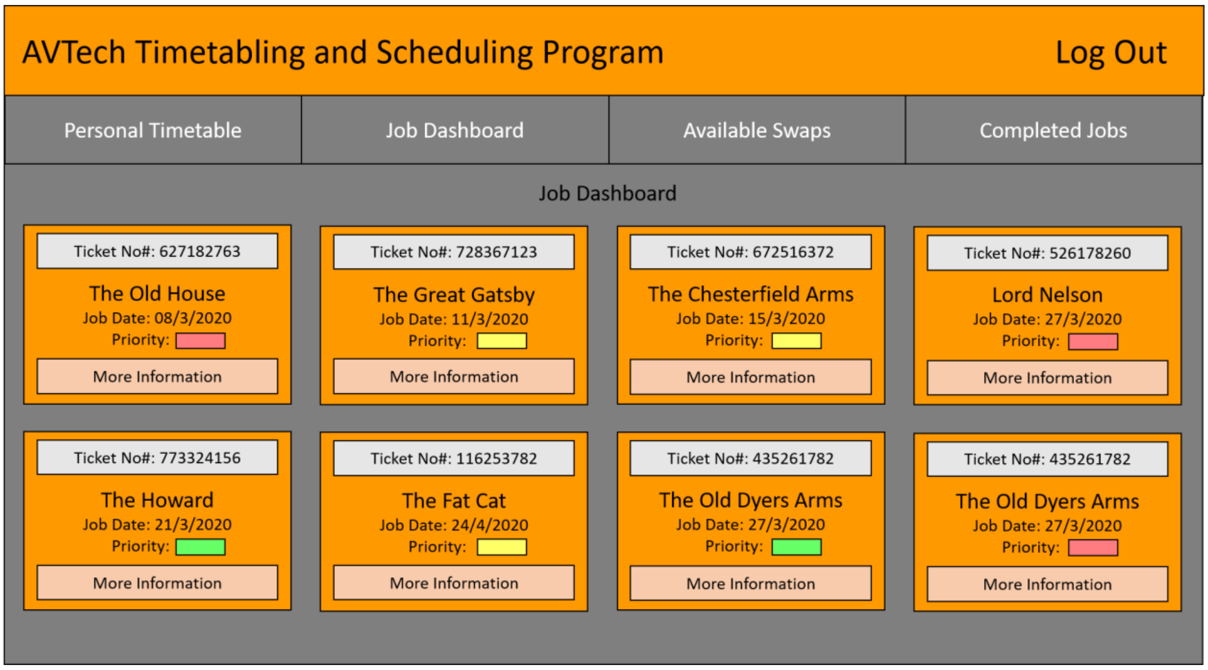


Figure 10. Personal Timetable Page Design

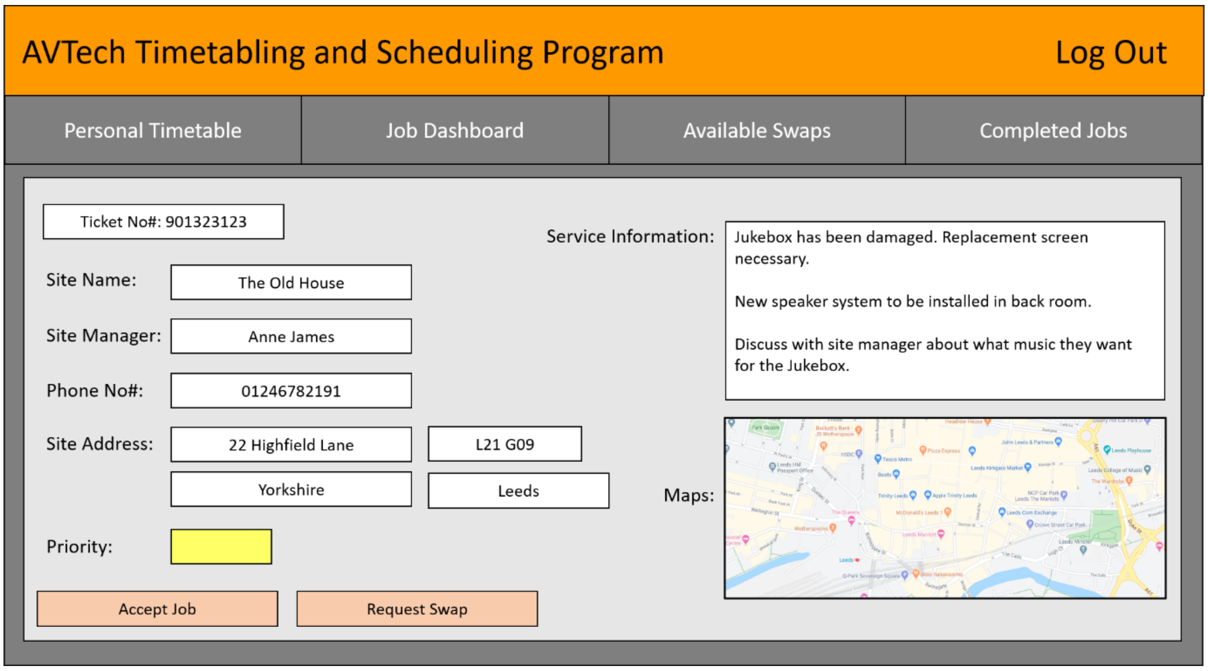

AVTech Timetabling and Scheduling Program — Log Out

Personal Timetable | Job Dashboard | Available Swaps | Completed Jobs

Job Dashboard

Ticket No#: 627182763	Ticket No#: 728367123	Ticket No#: 672516372	Ticket No#: 526178260
The Old House	The Great Gatsby	The Chesterfield Arms	Lord Nelson
Job Date: 08/3/2020	Job Date: 11/3/2020	Job Date: 15/3/2020	Job Date: 27/3/2020
Priority: 🟥	Priority: 🟨	Priority: 🟨	Priority: 🟥
More Information	More Information	More Information	More Information
Ticket No#: 773324156	Ticket No#: 116253782	Ticket No#: 435261782	Ticket No#: 435261782
The Howard	The Fat Cat	The Old Dyers Arms	The Old Dyers Arms
Job Date: 21/3/2020	Job Date: 24/4/2020	Job Date: 27/3/2020	Job Date: 27/3/2020
Priority: 🟩	Priority: 🟨	Priority: 🟩	Priority: 🟥
More Information	More Information	More Information	More Information

Figure 11. Job Dashboard Page Design


AVTech Timetabling and Scheduling Program — Log Out

Personal Timetable | Job Dashboard | Available Swaps | Completed Jobs

Ticket No#: 901323123

Site Name:	The Old House
Site Manager:	Anne James
Phone No#:	01246782191
Site Address:	22 Highfield Lane, L21 G09, Yorkshire, Leeds
Priority:	🟨

Service Information:

Jukebox has been damaged. Replacement screen necessary.

New speaker system to be installed in back room.

Discuss with site manager about what music they want for the Jukebox.

Maps:

Accept Job | Request Swap

Figure 12. Job Information Page Design

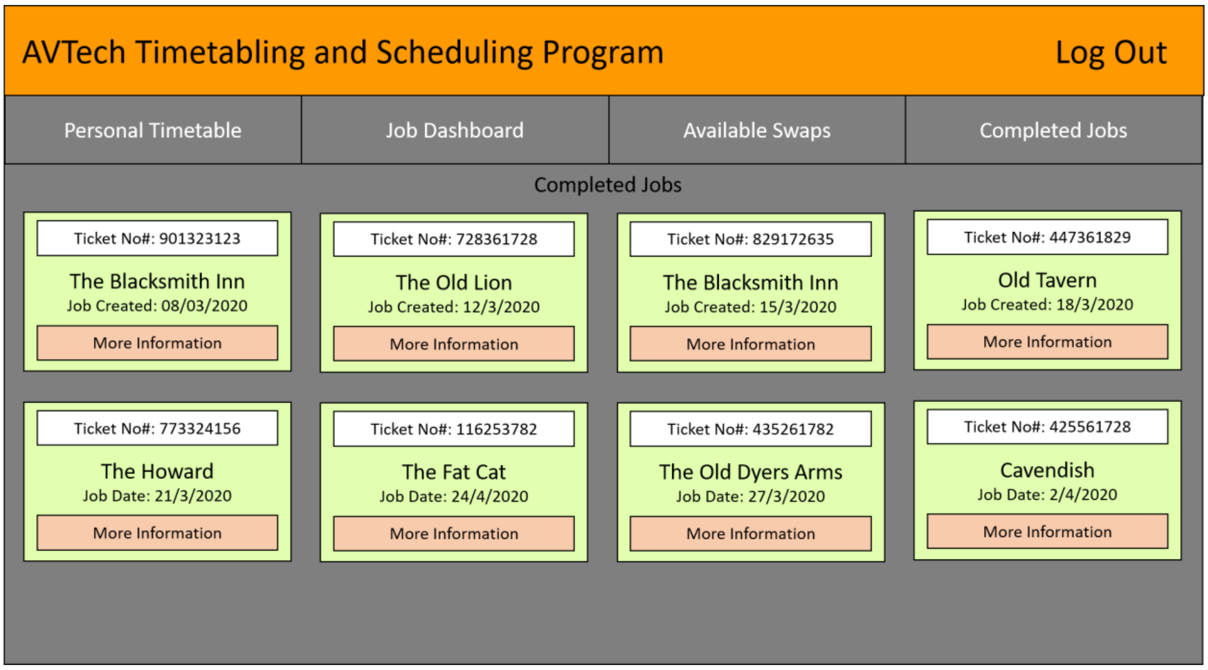


Figure 13. Completed Jobs Page Design

4.1.5. Database Design

To understand what information was necessary to include in a job ticket and therefor a database table to store these job tickets an employee of the company associated with this project explained what kind of information is usually on a job ticket. They mentioned various things like job numbers, site names, contact information, address and service information going on to demonstrate what a job ticket looked like at that point in time.

This information was then used to design a table that would house all the job tickets. This table was referred to as the "jobDashboard" table. This table would house all the job tickets and their relevant information. In the design two other types of tables were envisioned that had the same structure and would house active job tickets and completed job tickets. The active job tickets would be split into tables by which engineer had accepted them and any completed tickets would have their own table. The engineer's tables were named "e1timetable", "e2timetable" and "e3timetable" finally the completed jobs would be put in a table named "completedJobs". The idea with these tables was to have data be inserted from one table from another then deleted from that original table. For this reason, the tables didn't really have any relationships. A diagram of how data moved between these tables can be seen in Appendix I.

4.1.6. Planned Structure

A planned structure demonstrates what the different areas of the system do and how they interact with each other.

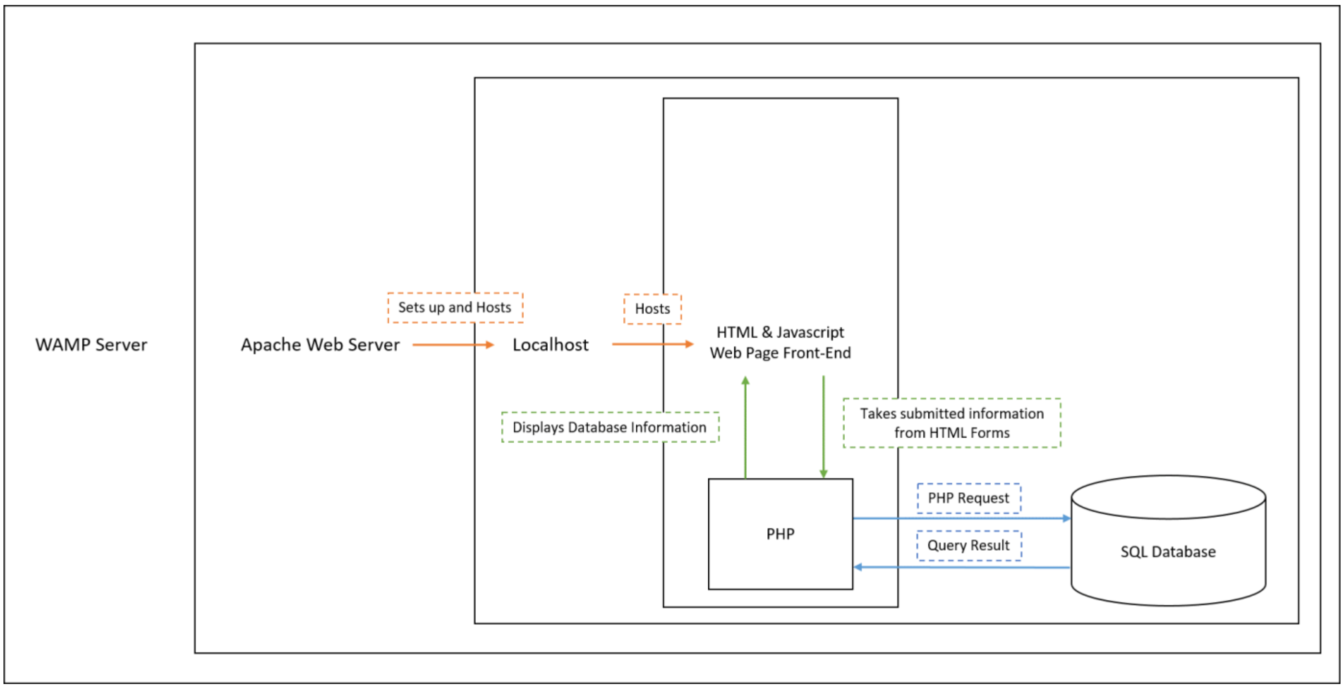

WAMP Server
Apache Web Server
Sets up and Hosts
Localhost
Hosts
HTML & Javascript Web Page Front-End
Displays Database Information
Takes submitted information from HTML Forms
PHP
PHP Request
Query Result
SQL Database

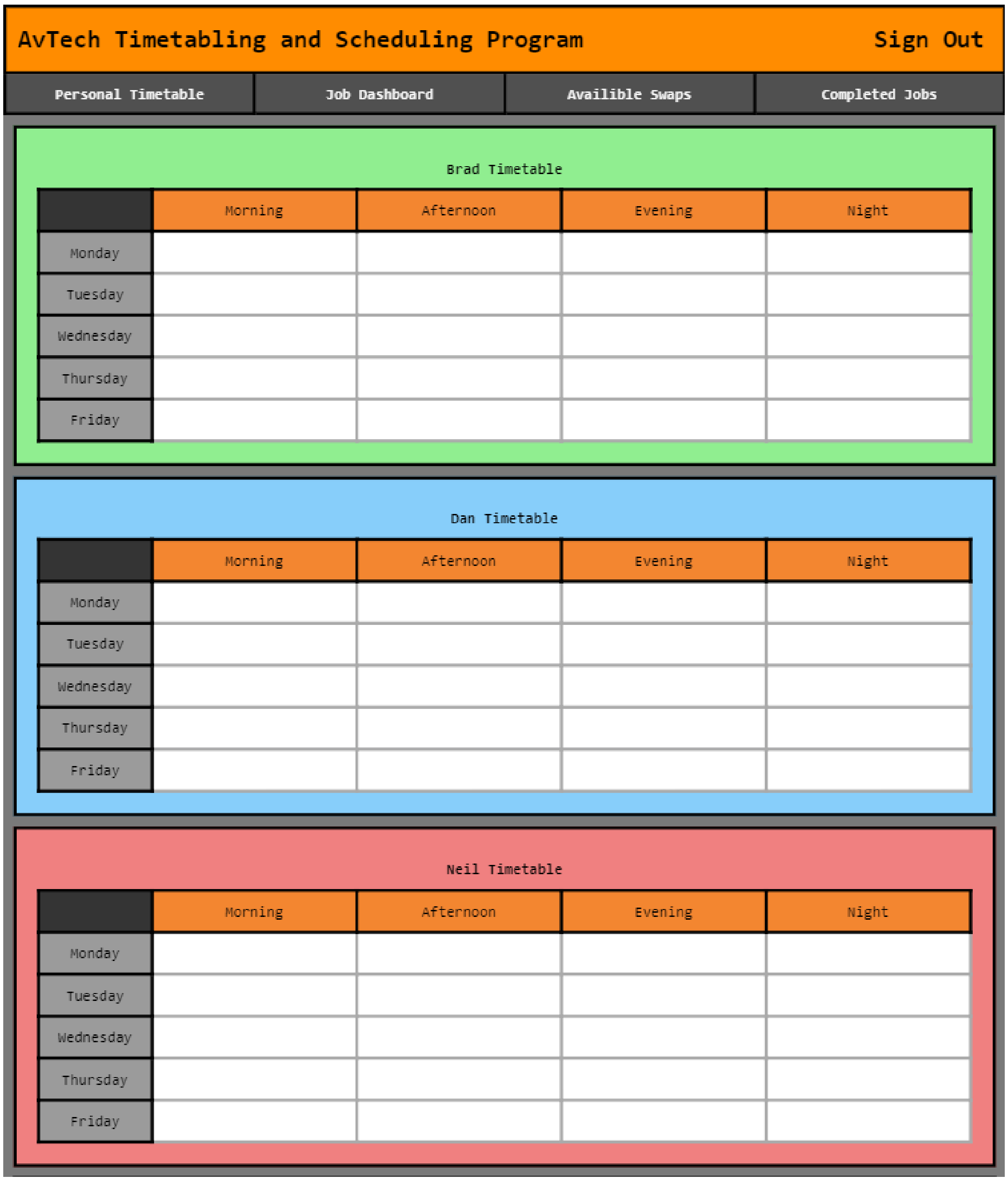
Figure 14. Planned Structure

Figure 14 shows the planned structure for this application. To begin, WAMP server encompasses the whole application allowing for the setup of Apache Web Server which will then setup the application on Localhost. Inside Localhost is the front-end containing the HTML, Javascript and PHP. The localhost also hosts the backend database.

The diagram also shows in a simple sense how data can be inserted from the front-end into the database and then and retrieved from the database to be displayed on the front-end. This loop can be seen in the diagram beginning with the initial HTML form data. This HTML form data is sent to the database by PHP. Once this data is inserted into a table on the database another PHP request can pull back that initial HTML form data and then display it on the HTML front end. In context of the application being used in the work environment this would be the receptionist creating a job on through a HTML form, that job information being inserted into the database and then the job information being pulled back to display on the job dashboard.

4.3. Development

Beginning development required a basic HTML skeleton based off the designs created in the design phase. This included a basic menu at the top of the application, some headers and a table that would be used by the engineers for the timetabling the different jobs. A different HTML page was created for each different user. Receptionist, Engineers one, two and three and the manager. This was then stylised using CSS. The skeleton of these pages and an example of what a job ticket would look like can be seen in figures 15, 16, and 17.



Brad Timetable

	Morning	Afternoon	Evening	Night
Monday				
Tuesday				
Wednesday				
Thursday				
Friday				

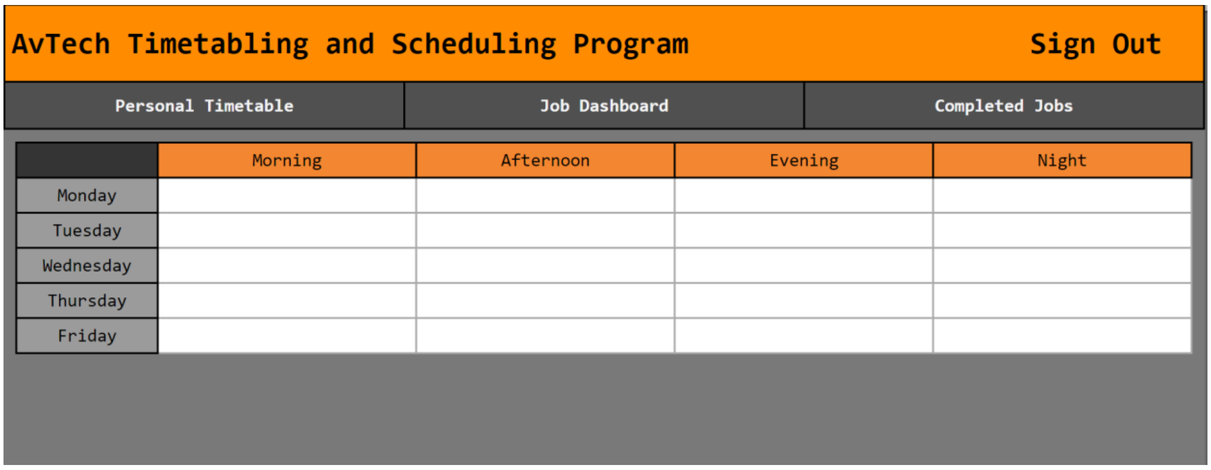
Dan Timetable

	Morning	Afternoon	Evening	Night
Monday				
Tuesday				
Wednesday				
Thursday				
Friday				

Neil Timetable

	Morning	Afternoon	Evening	Night
Monday				
Tuesday				
Wednesday				
Thursday				
Friday				

Figure 15. Manager's Timetable Overview Page

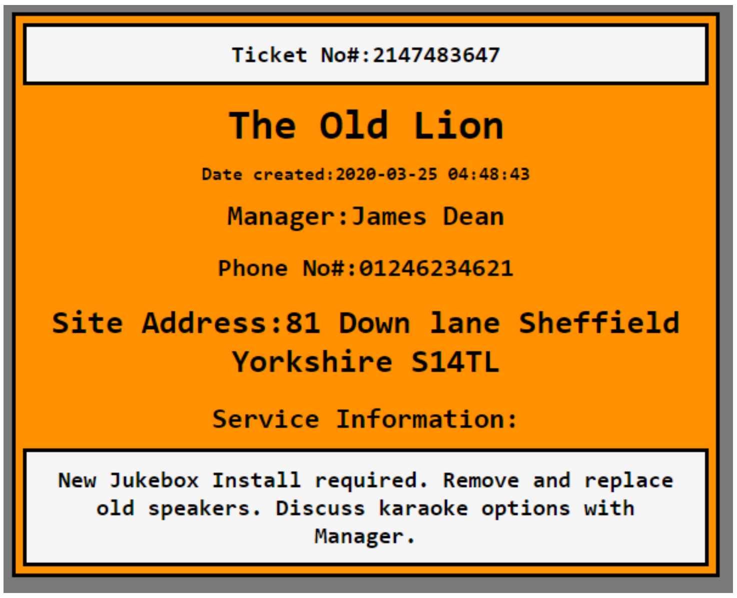

AvTech Timetabling and Scheduling Program

Sign Out

Personal Timetable | Job Dashboard | Completed Jobs

	Morning	Afternoon	Evening	Night
Monday				
Tuesday				
Wednesday				
Thursday				
Friday				

Figure 16. Engineer's Timetable Skeleton Page


Ticket No#:2147483647

The Old Lion

Date created:2020-03-25 04:48:43

Manager:James Dean

Phone No#:01246234621

Site Address:81 Down lane Sheffield Yorkshire S14TL

Service Information:

New Jukebox Install required. Remove and replace old speakers. Discuss karaoke options with Manager.

Figure 17. Example of a Job Ticket

Next a database was created that could house all the different tables that would store information that would be necessary for this project. This was created in MySQL which was linked through WAMP.

Now that the database and basic application layout and navigation had been created the receptionist job creation functionality was developed. To achieve this a form was created that allowed for the input of all the necessary information that was required to create a job. This outline of information was provided by the client and was mentioned in the user requirements. This included information such as the site name, site manager contact information, the priority of the job and service information among other things. The full form can be seen in figure 18.

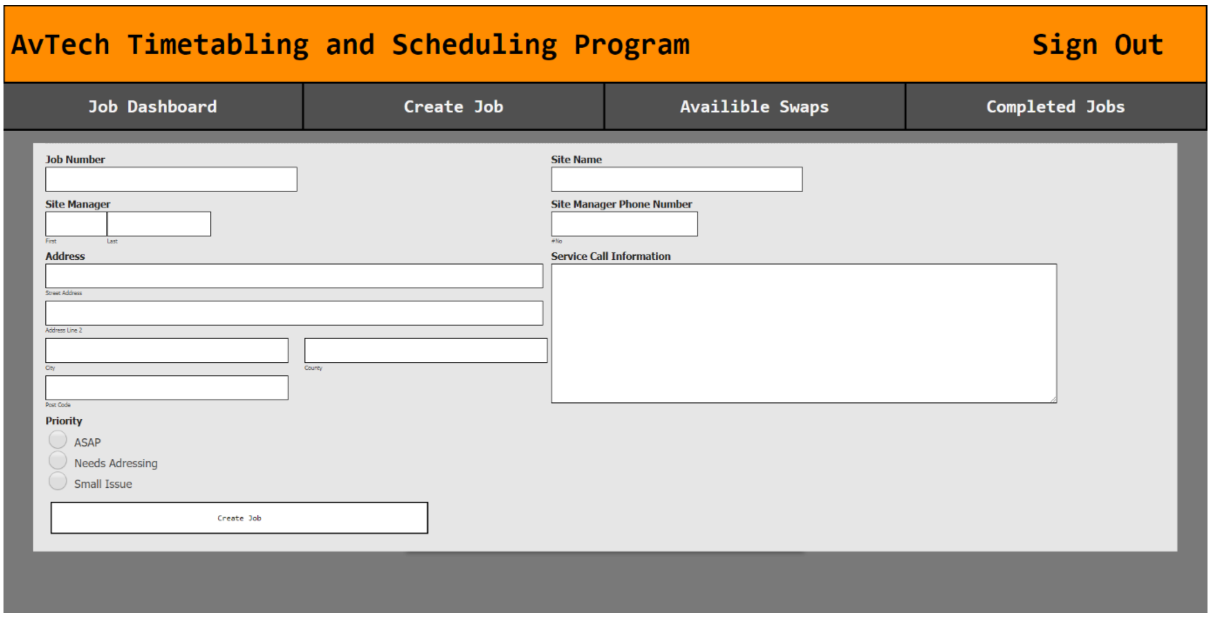
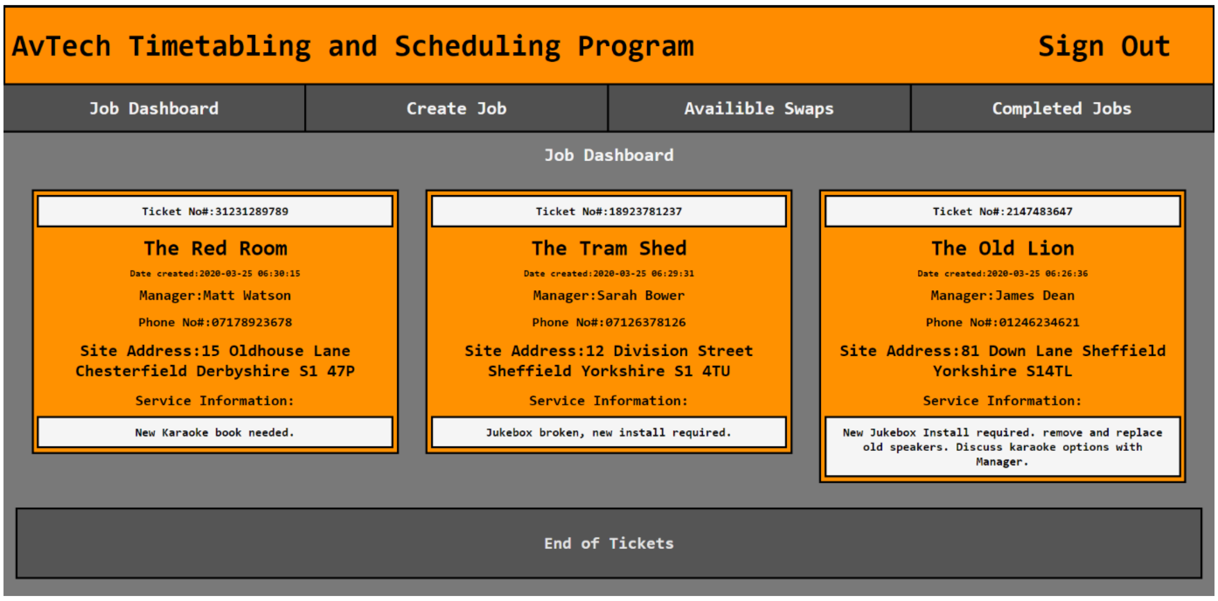


Figure 18. Job Creation Page for the Receptionist with Job Creation Form

Once the form was created a database table was necessary to insert and store information from the job creation form. Back in MySQL a table was created that mirrored the information on the job creation form. This table will be referred to as the job dashboard table mentioned in the design phase. Its purpose is to insert data into from the job creation form. The application would then pull rows of data from this table to display in job tickets on the job dashboard page. To avoid unnecessary errors in the application important data integrity measures were put in place. This meant that all the columns had to have the correct data types and constraints. To assure that the very basic information was inputted to create a job the 'jobNumber' and 'siteName' columns were created with the constraint 'NOT NULL'. This meant that a job could not be created if either of these fields were left blank on the job creation form. The 'jobNumber' column was also given the 'UNIQUE' constraint. This assured that there could be no duplicate job numbers which could lead to errors or jobs being mixed up. Close attention was also applied to the column datatypes. Columns such as 'jobNumber' and 'phoneNumber' were given the INT datatype. This meant that only numbers could be inputted into these fields. Whereas the VARCHAR datatype was applied to other columns allowing for a range of letters, numbers and characters. Finally, a column was created called 'createdDate'. This column automatically inserts the time and date of when a ticket was created. The table structure for the job dashboard table can be found in Appendix J.

Now that the job dashboard table and the job creation form were finished, they had to be linked up. This was done through PHP. A script was created where upon clicking the submit button on the job creation form all the values will be inserted into the database table. The code used to insert the form data can be found in Appendix K.

Next up was pulling the data back from this newly created table. This required some PHP code that created a different ticket for each row in the database. A while loop was coded that creates a ticket for each row in the job dashboard table. A SQL query was created that selected all the data in the table and ordered the tickets by the 'createdDate' column so that the newest tickets show first. The code for the SQL query can be seen in Appendix L. This code establishes a connection to the database then selects all the information in the job dashboard table. The code for this while loop can be seen in Appendix M. This code is used to construct a ticket. At the beginning of the statement the previously mentioned SQL statement is called. The code then displays a simple header for each part of the ticket such as "Site Name:", the php pulls the site name related to that job number from the job dashboard table. This while loop then repeats and constructs the next ticket on the next row of the job dashboard table until there are no more left. Once this is all complete the job tickets will display on the job dashboard page. Some sample data showing a couple of



Ticket No#:31231289789
The Red Room
Date created:2020-03-25 06:30:15
Manager:Matt Watson
Phone No#:07178923678
Site Address:15 Oldhouse Lane Chesterfield Derbyshire S1 47P
Service Information:
New Karaoke book needed.

Ticket No#:18923781237
The Tram Shed
Date created:2020-03-25 06:29:31
Manager:Sarah Bower
Phone No#:07126378126
Site Address:12 Division Street Sheffield Yorkshire S1 4TU
Service Information:
Jukebox broken, new install required.

Ticket No#:2147483647
The Old Lion
Date created:2020-03-25 06:26:36
Manager:James Dean
Phone No#:01246234621
Site Address:81 Down Lane Sheffield Yorkshire S14TL
Service Information:
New Jukebox Install required. remove and replace old speakers. Discuss karaoke options with Manager.

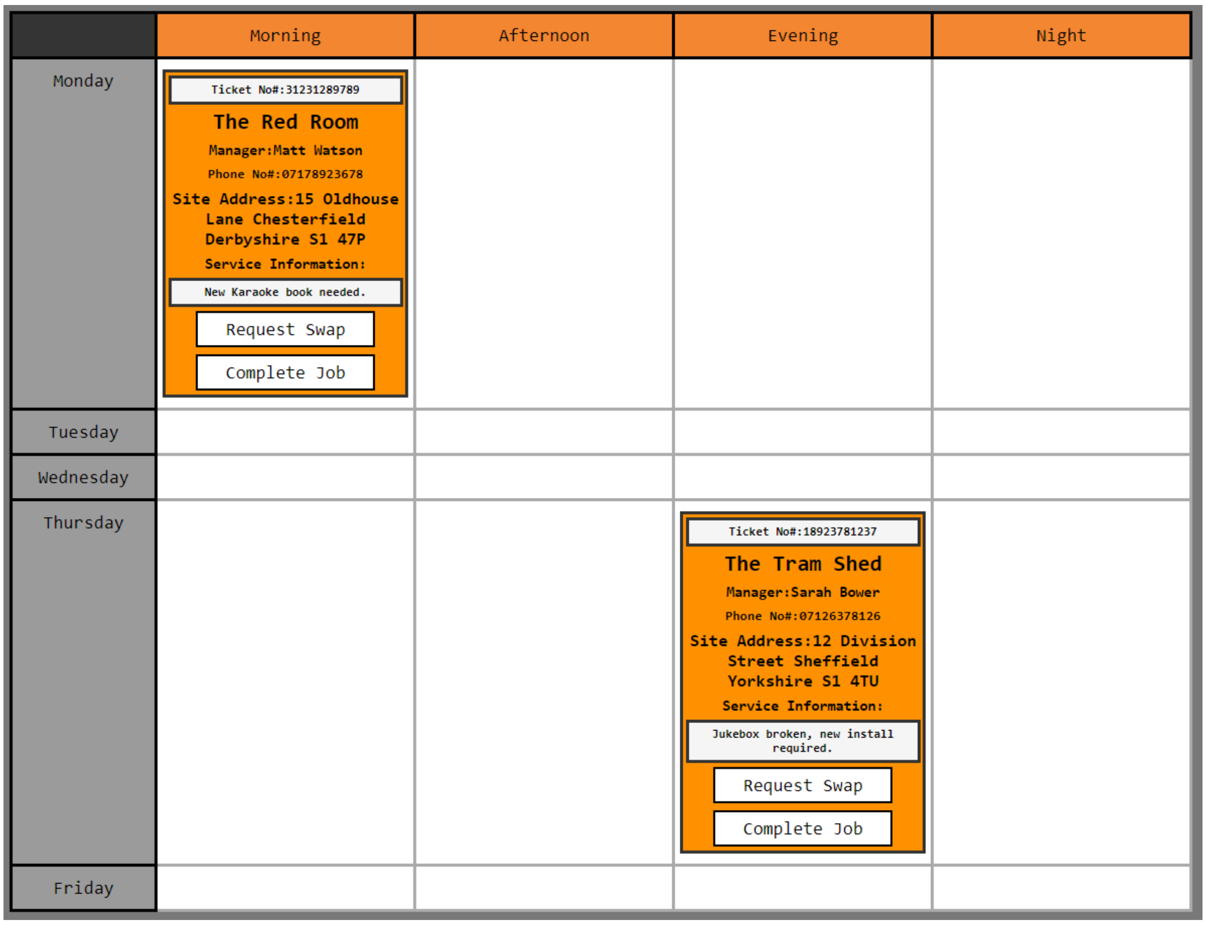
Figure 19. An Example of Tickets on the Job Dashboard Page

different job tickets can be seen in Figure 19.

With job tickets able to be created and then displayed on the application's job dashboard the next step was allowing the different engineers to accept jobs and have them allocated to different slots on their personal timetables. To achieve this some more SQL database tables had to be created. These tables were the engineer timetables mentioned in the design phase. The structure was similar to the job dashboard table as they will contain the same ticket information. The only difference is that the engineer timetable table will have a column named 'timetableSlot'. This column will contain information that will direct the ticket to where it should be in the engineer's timetable on the application. 3 engineer timetables were created, one for each engineer. These tables' structure can be seen in Appendix N.

For the engineers to accept a ticket some extra HTML and PHP code was implemented. The HTML code created a dropdown box and submit button for each ticket on the job dashboard. The dropdown box had an option for each slot on the timetable. After this another PHP script was created. This PHP script was executed when the "Accept Job" button was pressed on a ticket. The code uses hidden text boxes to store the data that is displayed then, when "Accept Job" is pressed, adds the timetable slot they have selected to a form which is then submitted and inserted into the engineer's timetable by calling an SQL insert statement similar to creating a job. Then the ticket is deleted from the job dashboard table. This meant that the job dashboard no longer showed any tickets that had been accepted. This code and the updated look of the job tickets can be seen in Appendix O.

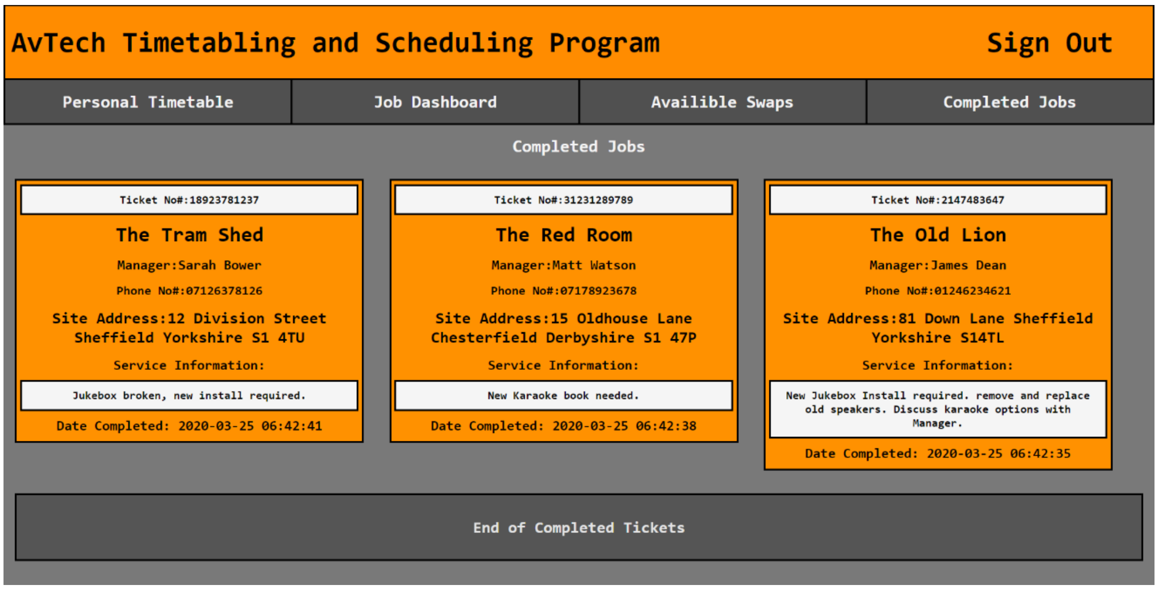
An engineer could now choose a time slot and accept a ticket from the job dashboard, but nothing was displayed on their personal timetable. To display the ticket on the engineer's personal timetable more PHP scripts were made. A PHP script was created for each timetable slot. These scripts were similar to each other and simply selected all the rows in the engineers table where the 'timetableSlot' data matched. So, for example, the Tuesday evening slot would look at the engineers table and retrieve all the rows that had the data 'tuesdayEvening' in the 'timetableSlot' column. It would then display these tickets in the timetable slot named 'tuesdayEvening'. Some sample data with a couple of tickets displayed in an engineer timetable can be seen in Figure 20.



	Morning	Afternoon	Evening	Night
Monday	Ticket No#:31231289789 The Red Room Manager:Matt Watson Phone No#:07178923678 Site Address:15 Oldhouse Lane Chesterfield Derbyshire S1 47P Service Information: New Karaoke book needed. Request Swap Complete Job			
Tuesday				
Wednesday				
Thursday			Ticket No#:18923781237 The Tram Shed Manager:Sarah Bower Phone No#:07126378126 Site Address:12 Division Street Sheffield Yorkshire S1 4TU Service Information: Jukebox broken, new install required. Request Swap Complete Job	
Friday				

Figure 20. Example of Job Tickets displayed in an Engineers Timetable

Engineers could now accept tickets from the job dashboard and allocate them to a slot on their timetable. The only thing missing to finish the job was a complete button. This button would allow engineers to remove the job ticket from their timetable once it was complete. The data would be inserted into a completed jobs table in the database and the ticket would then go to another tab labelled 'Completed Ticket'. This tab would contain all the tickets that had been completed and the information associated with them. To achieve this another PHP statement was created that worked similarly to the engineer's timetable insert statement. This new statement would insert the ticket information into a 'completedJobs' table in the database and then delete the job from the engineer's table in the database. The code for this functionality can be found in Appendix P. An example of some completed tickets can be



AvTech Timetabling and Scheduling Program — Sign Out

Personal Timetable | Job Dashboard | Availible Swaps | Completed Jobs

Completed Jobs

Ticket No#:18923781237
The Tram Shed
Manager:Sarah Bower
Phone No#:07126378126
Site Address:12 Division Street Sheffield Yorkshire S1 4TU
Service Information:
Jukebox broken, new install required.
Date Completed: 2020-03-25 06:42:41

Ticket No#:31231289789
The Red Room
Manager:Matt Watson
Phone No#:07178923678
Site Address:15 Oldhouse Lane Chesterfield Derbyshire S1 47P
Service Information:
New Karaoke book needed.
Date Completed: 2020-03-25 06:42:38

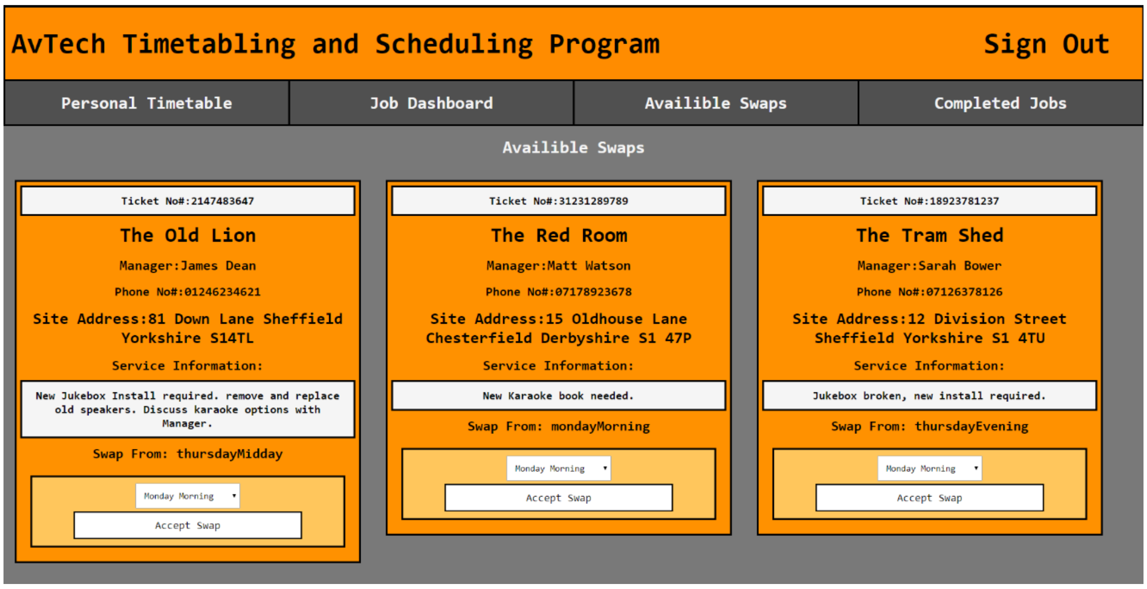
Ticket No#:2147483647
The Old Lion
Manager:James Dean
Phone No#:01246234621
Site Address:81 Down Lane Sheffield Yorkshire S14TL
Service Information:
New Jukebox Install required. remove and replace old speakers. Discuss karaoke options with Manager.
Date Completed: 2020-03-25 06:42:35

End of Completed Tickets

Figure 21. Example of Tickets displayed on the Completed Tickets Page

seen in Figure 21.

With the full cycle of creating, accepting, assigning and completing jobs created the next task was to allow users to swap jobs. To implement this functionality a new table was created in the database. This table would hold any available swaps engineers request until another engineer accepts them or they reclaim the job. This would like the job dashboard page where tickets could be displayed and accepted. On the front end, another tab was created for engineers. In this tab an engineer could see any available swaps. They could then assign any jobs to a timetable slot and accept them. An example of tickets displayed in the available swap section can be seen in Figure 22.



AvTech Timetabling and Scheduling Program — Sign Out

Personal Timetable | Job Dashboard | Availible Swaps | Completed Jobs

Availible Swaps

Ticket No#:2147483647
The Old Lion
Manager:James Dean
Phone No#:01246234621
Site Address:81 Down Lane Sheffield Yorkshire S14TL
Service Information:
New Jukebox Install required. remove and replace old speakers. Discuss karaoke options with Manager.
Swap From: thursdayMidday
Monday Morning
Accept Swap

Ticket No#:31231289789
The Red Room
Manager:Matt Watson
Phone No#:07178923678
Site Address:15 Oldhouse Lane Chesterfield Derbyshire S1 47P
Service Information:
New Karaoke book needed.
Swap From: mondayMorning
Monday Morning
Accept Swap

Ticket No#:18923781237
The Tram Shed
Manager:Sarah Bower
Phone No#:07126378126
Site Address:12 Division Street Sheffield Yorkshire S1 4TU
Service Information:
Jukebox broken, new install required.
Swap From: thursdayEvening
Monday Morning
Accept Swap

Figure 22. Example of Available Swap Tickets

4.4. Conclusion

Research into good design practices allowed for this project to produce a solid pre-development design that could be used as a template for the HTML skeleton.

Pre-development explorations into how the application would function and developing a plan as to how the backend database would work ensured that there were no major areas of development left without a plan. Constructing a planned structure also helps to visualise how the application is hosted to a localhost server for development and testing.

The development phase of this project brought many challenges and also required using the MoSCoW table to prioritise what functionality had to be completed and what extra functionality could be added after. This meant some functionality could not be completed, either due to time or difficulty of programming. These missed features and functionality, and why they were not completed will be discussed further in the evaluation chapter and functionality and design changes subchapter.

Chapter 5. Testing

One of the most trusted and widely used methods of software testing is the 4 levels of software testing. The 4 levels of software testing look at the application from different levels. The importance of this is that even if the application seems working as a whole looking at the code from different levels helps identify any bugs or errors that may go unnoticed.

5.1. Unit Testing

The International Software Testing Qualifications Board defines Unit Testing as “A test level that focuses on individual hardware or software components.” (International Software Testing Qualifications Board, 2018). Unit testing focuses on the most minute parts of code and functionality called units. This level of testing is the most focused part of the 4 testing levels.

To perform unit testing in this application simple lines of code, methods and basic functions were picked out from different areas of the application. They were placed in a table and tested this can be seen in Table 4.

Functionality	Expected Result	Level of Test	Result
Receptionist can view the job dashboard.	Receptionist can navigate to and view the job dashboard.	Unit Test	Success
Receptionist should not be able to accept Jobs.	There is no button to accept jobs on the receptionist page.	Unit Test	Success
An engineer should be able to view the job dashboard.	Engineer can navigate to and view the job dashboard.	Unit Test	Success
Engineers can access the job swap tab.	Engineers can navigate to the job swap tab and view it.	Unit Test	Success

Table 4. Unit Testing Table

5.2. Integration Testing

Once unit testing is complete the next level is integration testing. This is the test of how different parts of the application interact with not only themselves but with different resources involved such as a web server or a database.

To perform Integration testing individual functionalities and larger methods were tested and observed on their successes and failures. An example of testing this could be accepting a job ticket, firstly seeing if the correct data has been inserted into the database and then checking if the ticket is appearing on the front end. If this occurs without any errors or missing information the test is a success. The integration testing table can be seen in Table 5.

Functionality	Expected Result	Level of Test	Result
Can Login to the system with the appropriate login information	Able to access the website with the correct login information.	Integration Test	Success
Users will be re-directed to the appropriate page for their login.	Logging in as an engineer should redirect the user to the engineer page. Logging in as the receptionist should re-direct the user to the receptionist page.	Integration Test	Success
Receptionists can create jobs.	Receptionist should be able to input some relevant information	Integration Test	Success

	and create a job ticket.		
A job created by the receptionist should insert into the job dashboard table in the database.	Viewing the back-end database should show a row in the job dashboard table with the information inputted by the receptionist.	Integration Test	Success
Jobs should be colour-coded by their priority.	Users should see a clear indication of what jobs are the most important by their colour.	Integration Test	Fail
An engineer should be able to choose a timetable spot and accept a job.	An engineer can choose a timetable slot and accept a job.	Integration Test	Success
When an engineer chooses a time and accepts a job the job should be inserted into the relevant engineer's timetable table and deleted from the job dashboard table.	Looking in the database before an after this functionality is tested should show the data inserted into the relevant engineers table and deleted from the job dashboard table.	Integration Test	Success
A job that has been accepted should appear in the relevant timetable slot for that engineer.	Checking the engineer's personal timetable should show the accepted job in the relevant slot on their timetable.	Integration Test	Success
An engineer can request a job swap.	An engineer can request a job swap which will be sent to the job request table.	Integration Test	Success
Engineers can accept job swaps.	Engineers can choose a timetable slot and send a job swap request back.	Integration Test	Fail
The jobs will be swapped from one timetable to another.	The database will insert the jobs into the other engineer's timetable and delete from the original table.	Integration Test	Fail
Jobs can be accepted from the swap tab and allocated accordingly.	Engineers can accept jobs for a certain timetable slot from the	Integration Test	Success

	swap tab. The job will be allocated accordingly.		
Once a job is completed an engineer should be able to complete the job and remove it from their timetable.	The engineer should press the 'Complete Job' button and the job should be removed from their timetable.	Integration Test	Success
Completed jobs are removed from the engineer's timetable table in the database and inserted into the completed jobs table.	Looking in the database before and after a job is completed shows the data is removed from the engineers table and inserted into the completed jobs table.	Integration Test	Success
Completed jobs should be displayed in the 'Completed Jobs' section of the application.	Completed job tickets are displayed in the 'Completed Jobs' tab.	Integration Test	Success
Users can log out of the system.	Users should be logged out of the system upon clicking the log-out button.	Integration Test	Success
Login cannot be bypassed simply.	Login cannot be bypassed by typing the engineer page link into a browser.	Integration Test	Success
A user cannot access a different user's page once logged in.	An engineer cannot access the receptionist's page once logged in.	Integration Test	Fail

Table 5. Integration Testing Table

5.3. System Testing

System testing will begin once integration testing is complete. This stage of software testing tests the application as a whole. Preferably in an environment similar to the one it will finally be deployed in.

To perform system testing the application will be taken into the client's office and setup on their network. Once setup, tests will be done on the application to assure its overall functionality works on the company's own infrastructure.

W3Counter's Browser Share stats for February 2019 shows the most popular browsers for internet users. The graph shows that Google Chrome is the most popular with a 58.1% share of internet users. Second is Safari with 13% of users. Third is Internet Explorer/Edge with 12.9%. Fourth is Firefox with 5.4% of users. Finally, is Opera with 2.7% (W3Counter, 2020).

Therefor further system testing will be done on the main web browsers to assure the system is still functional if a user decides to use a different web browser. Evidence of the application working in the different browsers can be seen in Appendix Q. The system testing table can be seen in Table 6.

Functionality	**Expected Result**	**Level of Test**	**Result**
The application works on the client's localhost successfully.	The application should be accessible on the clients WIFI.	System Test	Success
A user can connect to the application from anywhere.	A user can access the application when not connected to the office WIFI.	System Test	Fail
The application works successfully on the Chrome browser.	The application should be accessible and usable on Chrome.	System Test	Success
The application works successfully on the Safari browser.	The application should be accessible and usable on Safari.	System Test	Success
The application works successfully on the Internet Explorer browser.	The application should be accessible and usable on Internet Explorer.	System Test	Success
The application works successfully on the Firefox browser.	The application should be accessible and usable on Firefox.	System Test	Success
The application works successfully on the Opera browser.	The application should be accessible and usable on Opera.	System Test	Success

Table 6. Integration Testing Table

5.4. Acceptance Testing

Acceptance testing is the final stage of the 4 levels of software testing. This is where the application is tested against the initial user requirements gathered at the beginning of the project. The application is also given to users to observe usability and gather feedback.

The acceptance testing being performed for this project will involve looking at the functionality and usability of the application from the user's point of view. To test the application users were given a document with several different tests to perform. For example: "Create a Job and add it to the Job Dashboard". These tests were created to get users to interact with various functionalities of the application that were developed from the user requirements gathered at the start of the project. Once a test had been performed users were asked to write down feedback on what they thought about the application. The users were also observed as they ran through the various test and notes were taken about their operation of the application. The acceptance test table shows the testing document with the list of tasks for the users. Appendix R shows the users feedback of the different areas of the application and the application in general. The acceptance testing table can be seen in Table 7.

Functionality	Expected Result	Level of Test	Result
A user can Log In	User can log in with the appropriate login information.	Acceptance Test	Success
An engineer can view the job dashboard and what jobs are available and the information associated with them.	The engineer will navigate to the job dashboard page and recall the site name of the oldest Job.	Acceptance Test	Success
An engineer can accept a job and assign it to their timetable.	An engineer will navigate to the job page and assign a job to Thursday afternoon and accept it.	Acceptance Test	Success
An engineer will request a job swap.	The engineer will request a swap for a job in their timetable.	Acceptance Test	Success
An engineer will accept a swap.	An engineer will accept a swap from the available swaps page and assign it to Tuesday morning.	Acceptance Test	Success
An engineer can complete a job.	Engineer can complete a job.	Acceptance Test	Success
An engineer can view completed jobs.	An engineer will navigate to the 'Completed Jobs' page and view the completed jobs.	Acceptance Test	Success
A receptionist can create a job.	The receptionist will navigate to the 'Create Job' page and fill in the relevant information to create a job.	Acceptance Test	Success
A receptionist can view jobs on the job dashboard.	The receptionist can view jobs including the one they have created on the job dashboard.	Acceptance Test	Success
A receptionist can view completed jobs.	The receptionist can navigate to the 'Completed Jobs'	Acceptance Test	Success

	page and view the completed jobs.		
A manager can view the job dashboard.	The manager will navigate to the 'Job Dashboard' page.	Acceptance Test	Success
A manager can view all engineer timetables.	The manager will navigate to the 'Timetables Page' and recall some information from a job and when it is planned to be completed.	Acceptance Test	Success
A manager can view completed jobs.	The manager can navigate to the 'Completed Jobs' page and view the completed jobs.	Acceptance Test	Success
A user can log out.	Users will navigate to the log out button and log out.	Acceptance Test	Success

Table 7. Acceptance Testing Table

5.5. Feedback

Testing the application with the client and employees was a crucial step which helped to get some valuable feedback about how the application looked and worked. Testing the application with other people helped to reveal any issues or improvements that could be made to this application that might not have been picked up on when developing or testing alone.

Something that was in this projects interest when gathering this feedback from the client and the employees was what would be necessary to take this application from a prototype to something that could be viable for their workplace.

This section will look at what feedback written by the client and employees in Appendix Q and expand on it with what was discussed with them as they tested the application.

5.5.1. Engineers

Looking at the feedback some improvements that stand out include the need for more information to be added to a ticket. The engineers mentioned that the more information there is on a job ticket such as notes about the site itself, what has been installed there and serial numbers of any stock that is going to be added or removed all culminate in a job ticket that has a sufficient amount of information to complete a job. Engineers also requested a feature that allows users to add notes and upload photos once a job has been completed. This allows them to document what has been changed by who and to upload pictures of the layout of a job site if they haven't visited there before. Uploading pictures of the job site will make it easier for them to do installations or fixes if they ever revisit a site.

The engineers liked the swap feature but mentioned it needed to be clearer who was requesting a job and to make the feature more streamlined. This meaning that a swap request would involve seeing another user's timetable and selecting which job they wanted to swap for. One engineer said that it should be necessary to provide a note or a reason why a job is being swapped.

The personal timetable page also received some feedback from the engineers. They recommended that the timetable could be expanded into more of a calendar view rather than just displaying a single week. This means that a user can book tickets into their timetable for future weeks. Another engineer also recommended making the timetable into hourly slots rather than morning, midday, evening and night. One engineer mentioned functionality that would let them show what time they had booked off on their timetables meaning no jobs would be booked for that time. Not being able to complete jobs before the job date set on the timetable was also suggested as then a job couldn't be accidentally completed or forgotten about.

Engineers seemed to like the idea of the job dashboard page. They claimed it laid out the information clearly in one place unlike looking through WhatsApp or Outlook Emails. They suggested that the date format on the tickets be changed from American to English.

To improve the application overall the engineers suggested adding a timesheet page. This page would let engineers check-in and check-out when they are on site and at the office. This would then make it easier for a receptionist to pay the engineers accurately at the end of the month. Some smaller suggestions included personalized welcome messages, the ability to customise backgrounds and add pictures to profiles.

The engineers also noticed some small bugs such as some site names not being displayed fully when inserted into the timetable.

Overall, the engineers were intrigued by the application. They mostly liked how it looked and the simple layout. One claimed that he would be up for using the application there and then as it helps with keeping track of information. Another suggested that it could be viable for the future of the business.

5.5.2. Receptionist

The receptionist liked the create job section as it made sure that she didn't forget any information when creating a job. She suggested that there should be somewhere to add serial numbers for stock that would be taken on site.

She liked the look of the job dashboard page and claimed it was easy to understand but needed the serial numbers for the tickets to be complete. The same was said for the job completion page adding that it was helpful for keeping track any jobs that had been completed.

Overall, she liked the application. She said it made creating jobs for engineers much easier than before. She also though it was super helpful to keep track of information.

5.5.3. Manger/Client

The client began by looking at all the different areas of the application. Being guided through how a receptionist makes a job to how a job is finally completed by an engineer. He was then shown his own manager front-end displaying all the engineer's timetables, job dashboard and completed jobs.

Beginning with the timetable overview page the client mentioned that all users should have a personal timetable not just the engineers. He also mentioned that all timetables could do with a feature that allows for small tasks to be created and added to timetables rather than just job tickets. Like the engineers he also mentioned that a calendar view would be more useful rather than just displaying one week at a time.

When reviewing the job dashboard page, he suggested that users should be able to edit tickets before and after completion to add more information.

Moving onto the completed jobs page he suggested a filter that could sort the tickets by site name, who had completed them and when they were completed. This would make it easier for the business to search for what work had been done on the sites in the past, when work was previously done and more smaller details about these jobs. He added that a search feature would also be very useful. He mentioned that the ticket sizes were too big and too many of them would be an overwhelming amount of information. To combat this, he suggested that tickets be made smaller and when clicked on an expanded view would open showing all the information. This suggestion was in one of the original design plans but had to be changed due to issues getting the information to display in an expanded view.

Moving on to the job swap feature he liked the idea but said it would be more viable if he had more than three engineers. With the current amount of engineers jobs aren't really being tossed back and forth too often but if the company expanded then this may be more likely.

As for the create jobs tab he was impressed claiming that this feature made it easy for a receptionist to create jobs and assures that all necessary information was included. Much like the other employees he said a notes feature would be needed to complete the job creation.

As for the overall application he had many suggestions for how it could be expanded upon and what extra features would be required for him to be confident using the application at his business. He began with saying that the statistics page would be very useful to him. He also repeated what the engineers said and asked for a time sheets feature. He also wanted the timesheets to be sent to the receptionist at the end of the month. To make the application more practical he suggested the ability to add more than one engineer to a job, as usually they go in pairs, would also be a good improvement. This would mean that a job would appear on more than one timetable and multiple engineers would be able to accept a job. He also suggested that jobs could be sorted by type. Some service issues can be solved over the phone or in the office through a computer. The client requested that these different type types of job should be specified. Finally, he suggested that, since the business is only small and close-knit everyone should be able to see each-others timetables and what jobs have been accepted by who.

Overall, the client liked the look and idea of the application and claimed that in its current state it works well. He went on to say that if the application could incorporate a few more features that would benefit the business and how it runs he would consider implementing the application at his business.

5.6. Conclusion

This part of the testing phase was very valuable as it showed what issues there were with this application and what the users would like to see added to it. Going out and testing the application in the business environment it was created for did put the application to the test and apart from some minor bugs it fared well. The users did have many suggestions for what could be added to the application or bits that were missing but had no real complaints about the functionality that had been added already.

The users also seemed content with the usability of the application. They found it easy to navigate and to perform the main functionality. Some suggestions about changing job ticket sizes were valid and were included in the original, mock-up designs but couldn't quite be achieved in development.

The main takeaway from testing with the users is a want for more functionality. For this application to be viable the users want more business specific features to streamline how that business runs.

6. Evaluation

6.1. Evaluation of the Application

In this section an evaluation of all the areas of the application will be conducted. The evaluation will split the project by the different pages, similar to how the feedback forms are in Appendix Q.

6.1.1. Login Page

The login page has a simple design which clearly shows what it is supposed to be. The functionality of this page works well. A user enters their login information and is redirected accordingly. The application does have some security. If a user tries to access a page by typing it into the URL (E.G. localhost:7234/avtechreceptionist) they will be redirected back to the login page.

This page could do with some further security. Passwords could be encrypted and not stored in plain text inside the database. Another flaw is that users can navigate between pages using the URL once logged in. These issues would be more of a concern if the application was going to be used by a lot of people or the application was going to be hosted on a real website. If this was the case security would be a much higher priority, but since the client is only hosting it on their own WIFI security breaches are unlikely.

6.1.2. Personal Timetable Page

The personal timetable page was interesting to create. Beginning with a simple table that displayed different slots for each part of the week and day it developed into a dynamic timetable that could be filled with job tickets. This part of the application does its job pretty well, it displays the information nicely and ticket information is shown in a simple manner. The job swap and complete job buttons work as expected, moving data into the appropriate tables in the database when clicked and removing the data from that user's personal timetable.

Probably the biggest issue with this area of the application is in the code. Due to being unable to find a better way of coding it there is over a thousand lines of code for this timetable alone. This is because each timeslot is assigned a name (E.G. Tuesday Evening is TE) and the code, like that of Appendix M, which is necessary to display any tickets needs to be duplicated for each timetable slot. Possibly there could have been a way to achieve this with a while loop, similar to how tickets are displayed on the job dashboard page that would have made the code much cleaner and concise. Despite this abundance of code, the users on the front-end won't be affected by this issue. Another issue this area of the application suffers from is that if a user assigns themselves more than one job in a single time slot the timetable begins to look messy as tickets stack on top of each other to squeeze into the timetable slot. To solve this issue the application could either not allow users to assign more than one job at a time to a single timetable slot or change the front-end user design to make the tickets smaller with an expanded view that displays all the information when clicked on. This design idea was showcased in Figure 12 of the initial design but failed to be implemented, this will be discussed further in the functionality and design changes subchapter.

6.1.3. Job Dashboard Page

The job dashboard page is probably the most complex of all the pages in terms of how it was coded. This page was where the design of the job tickets was first created in the HTML skeleton. The way information is displayed in these tickets is something I think this project succeeded at. The information inserted into a job creation form is inserted into as a row in the job dashboard table and then pulled from the database and displayed in a simple job ticket. These tickets also allow for engineers to choose a timetable slot and accept tickets. This feature also works as originally intended, a drop-down box allows users to select the timetable slot they want and accept the job. This page uses a while loop to continuously display all the tickets in the job dashboard table in a grid that is ordered by when they were created. This means old tickets always appear first, so they are not forgotten about.

Some improvements that could be made to this page include having the drop-down box that lets a user edit a timetable slot appear after a user accepts the job. This would improve the usability as a user could accept the job and forget to choose a timetable slot meaning all the tickets will be accidentally put into the Monday morning slot as that is the default choice. Looking at the testing feedback it is clear that some extra information needed to be added to these tickets, including serial numbers for stock and a notes section for additional information about the site and job.

6.1.4. Job Creation Page

The job creation page allows the receptionist at the business create job tickets. Looking at the testing feedback it was mentioned that this part of the application made adding information to create a job much easier than before. With all the boxes displayed it made sure that no information was forgotten about. The page's functionality takes information, stores it in the database and constructs a job ticket upon clicking the create job button.

One improvement that could be made to this area of the application include the missing serial numbers for stock that was mentioned in the feedback. Another could be implementing data validation on the form. This would make sure that data is entered correctly and if not, an error appears to clarify why. For example, typing letters into the phone number box and trying to create the job will just do nothing. To improve this an error message could appear saying “Only numbers can be entered here.”. This would make it more clear why data is sometimes not inserted to the database.

6.1.5. Job Swap Page

The job swapping feature was a unique feature of this project. From research conducted other applications didn’t have this feature. The feature works to an extent. Users can swap jobs with other users, but it can be tedious and could be improved. Users can swap jobs with this functionality but not directly. It requires users to send their job tickets to a job swap dashboard and then accept another ticket from the choices presented to them. It was intended to work in a way that users could directly request jobs from other user’s timetables and choose one of their own to swap. This feature had to be made simpler due to time constraints and the difficulty of the programming required to achieve it.

6.1.6. Completed Jobs

The completed jobs page was simple to create as it used the same layout as the job dashboard page but showed job tickets that had been completed instead. Similarly, to the job dashboard page the tickets are clear to read, and the relevant information is displayed.

Adding a filter would improve this page as if this application were to be used in a real setting this page would continue to grow with more and more jobs being completed. Adding a filter would allow users to sort all the tickets by date, who completed them, site name and more. This would allow the employees to look up any notes or information of what work was done on previous jobs. Another improvement would be displaying serial numbers and further notes as previously discussed as well as adding who completed the job.

6.1.7. Timetable Overview

Finally, the timetable overview page made for the manager. This page gathered all the engineer’s timetables and put them onto one page for the manager to see. The tables are separated by colour to make it clear which engineer table belongs to who and the tickets they have accepted. This page is helpful to the manger and allows them to keep a track of all the jobs that should be completed that week.

To improve this page a better designed front-end could have been included. Maybe a sub menu of the engineer’s timetables that would show/hide them when clicked. Currently having all the tables on one page could be overwhelming.

6.1.8 Overall Application

As a whole the application works well as a prototype. It demonstrates a basic starting point for a tailored timetabling and scheduling application. It has the ability to store information like upcoming and completed job tickets which the company didn’t have before. Engineers can use the application to document what jobs they will be working on; they can swap jobs with other users and can complete jobs that will be held in a database. A receptionist can use the application as a preferred way of creating tickets that was easier than her previous method and a manger can use the application to have an overview of the working week and what task his employees will be undertaking.

Since the application is a prototype there are many improvements and additions that could be made to improve it. Its basic functionality is solid and a good groundwork to build more functionality off as this is what the application lacks. Looking at the feedback in chapter 5 the most common feedback received was a request for more additions to the application. More features and additions to already existing functionality would be one stepping-stone to this application being viable for a real work environment. Another would be having the application hosting on a website that could be accessed from anywhere. Currently the application is only accessible on a local internet connection. For engineers to be able to see this application when travelling around the country and at different sites will make the application more accessible and useful. Another addition previously mentioned was increased security if the application was to go online. This would entail securing the database to ensure there can be no SQL injections and securing the login further so that it cannot be easily circumvented.

Chapter 7. Conclusion and Future Work

7.1. Limitations of the Project

Several limitations faced this project. Some were known from the beginning such as time. The time given to work on this application during a year of university was enough for this project to develop a prototype application but not enough to complete all the tasks outlined in the user requirements. This is why the MoSCoW method was used. It split the tasks up into what was achievable, what could be done and what was going to be too much work to come close to achieving in the given time. As with all things more time would have allowed for this project to work on the more difficult sections of the MoSCoW grid.

Another limitation known at the beginning of this project was the beginner level knowledge of most of the languages and tools used. The research into these tools and languages made it clearer what they could do and how they would work with each other to develop this application. Despite this when developing the application, a lot of troubleshooting and fixing coding errors took place which would take up hours at a time. The knowledge of these tools and languages and how they work has improved as a result of this project but overall, the development of this project would have been easier if there was a more advanced prior knowledge of how to use these tools and languages.

A limitation realised after collecting the user requirements that reduced the usability and accessibility of this project was cost. The application currently runs from localhost, which means only users the same WIFI connection can use it. To allow this application to be accessible outside of the workplace would have required paying a hosting site a monthly subscription or purchasing a server to host the site personally. Not only was there no budget for this project but when meeting with the client to discuss user requirements he specifically requested for the application to be free.

Towards the end of the project limitations of the small testing group became apparent. With the client involved only having a small business with very few employees a bigger sample size of people to test the application that worked in that business environment was hard to come by. To improve the testing results and get more feedback a bigger testing group would have been useful. More people testing the application would have allowed for further feedback and even more clarity as to what the flaws were and what needed to be added.

7.2. Functionality and Design Changes

As the project began development, issues in terms of what could be done with the knowledge of the tools and languages, the tools and languages themselves and time given were realised. Certain methods, functionality and designs didn't quite work as they were expected to.

One example from the design was having the job tickets be smaller on the page and then clicked on to reveal more information on another page. Due to the way all the tickets were displayed using a while loop it became complicated to get the correct ticket data to carry over to a separate information page. Rather than spend time trying to get this working the alternative approach was to have all the information inside a single ticket. This originally looked okay and was a successful work around. The only issue was that tickets now took up a lot more space on the personal timetable for engineers. This was brought up by the manager in the testing phase, mentioning that a reduced size with information on a different page would have worked better.

The biggest functionality that had to be changed due to the mentioned limitations was the job swap functionality. Originally it would have allowed for users to select a ticket they wanted to swap from another user's timetable and a ticket they wished to swap from their own. Again, due to knowledge and time this had to be changed and instead users just post tickets to a page where they can see other tickets that want to be swapped. This method still works but isn't as elegant as originally planned.

Some features that were shown in the should and could sections of the MoSCoW grid were also not at a stage to be incorporated into the final design. These ideas and functionalities would have been great to include but it was important that the functionality described in the must category had to be working successfully first.

One functionality that wasn't included was the Google Maps API. The main issue with incorporating Google Maps API was that it couldn't automatically identify the site name when creating the form. So, when typing in the pub name or the address the maps wouldn't be able to pick up on these words to search for them. To achieve this more time and knowledge of APIs and how they work could have been helpful.

A statistics page was also difficult to construct as there was only basic information to showcase there such as how many jobs had been completed and how many jobs were available. If the google maps API had worked, more statistics could have been added such as miles travelled, petrol used and more. Currently if a statistics page was added most of the data would either be very simple or would have to be entered manually which isn't useful for the business to spend time doing that.

Small tasks were also missed from the application as it was lower on the list of priorities. For this functionality to be viable every user would have to have their own timetable and database behind it. There would also need to be a new small tasks creation form developed as well as a new ticket design for small tasks. Out of all the missing features mentioned this was probably the next functionality to achieve as all the pieces had already been created.

Due to a time crunch and more of a focus on making the basic functionality work well this was left too late to be incorporated.

7.3. The Future

There is a lot of directions this application could be taken in currently. As it is a prototype application the functionality so far lays a groundwork for further development to be done. The project could continue to be developed specifically for the client involved, introducing the features missing from the application that had been requested by the client and employees. From the feedback the client mentions that if the missing features were incorporated then he would consider using the application in his business. This application could be taken and tailored specifically for that business, working closely with them to develop an application that suits all the user's needs.

From another perspective the application could be taken and tailored to a different business. A few changes would have to be made. These include a redesign of the front-end. This would mean changing menu items, headers, and job ticket information. The application would also require changes to the database, so it holds data relevant to this different business. The functionality developed could stay the same and be used by another business. If the application began incorporating more functionality tailored for the original client adapting it would be more work.

7.4. Aim and Objectives

Recalling the introductory chapter of this dissertation we can look at the aim and objectives and tasks table and discuss whether what this project has achieved what it originally set out to do.

Looking back, the aim of this project was to develop a prototype web-based timetabling and scheduling application tailored for this small business to improve the control employees and managers have over their working week. This prototype application was tailored specifically for this business and some aspects of this project do improve the control employees have over their working week. The feedback received from the employees tells us that this application provides an improvement in job creation, management and tracking of information. Most employees agree that the basic features of the application work well, but the application is in need of more features that would benefit the company for it to be viable to them. As previously mentioned, this prototype application has a solid groundwork for becoming a useful productivity tool for this company. The hope is that more work is done in the future to make this a reality.

Appendix

Appendix A – Initial User Requirements Table

	Functional Requirements	Non-Functional Requirements
General	- Different Users - Job Calculating Algorithm - Notifications - Database/Application Error Triggers	- Mobile Accessible - Assuring Application Synchronization
Jobs	- Ability to create Jobs - Engineers can swap jobs	- Easy Job creation
Timetables	- Ability to view personal Timetables - Ability for a Manager to see all Timetables	- Easy to read and understand information

Appendix B – User Requirements Table post Similar Applications Investigation

	Functional Requirements	Non-Functional Requirements
General	- Different Users - Job Calculating Algorithm - Client List - Google Calendar API - Notifications - Database/Application Error Triggers	- Free - Easy to Use - Easy to Navigate - Mobile Accessible - Well-presented information - Assuring Application Synchronization
Jobs	- Ability to create Jobs - Jobs assigned by a Receptionist - Google Maps API for Job Creation - Email to notify sites of cancelled jobs - Engineers can swap jobs	- Easy Job creation
Timetables	- Ability to view personal Timetables - Ability for a Manager to see all Timetables - Responsive Design - Separate List View for Timetabled Appointments	- Easy to read and understand information

Appendix C – Tools and Languages Comparison Tables

	Reliable	Easy to Implement	Simplistic	Interoperable	Wide Use
HTML	✓	✓	~	✓	✓
HAML	~	✗	✓	✓	✗

Table Showing Mark-up Languages Comparison

	Fast	Easy to Implement	Secure	Interoperable	Wide Use
Javascript	✓	✗	~	✓	✓
DART	✓	✗	~	✓	✗

Table Showing Programming Languages Comparison

	Asynchronous	Easy to Implement	Secure	Interoperable	Wide Use
PHP	✗	✓	~	✓	✓
AJAX	✓	✗	~	✗	✗

Table Showing Server-Side Scripting Languages Comparison

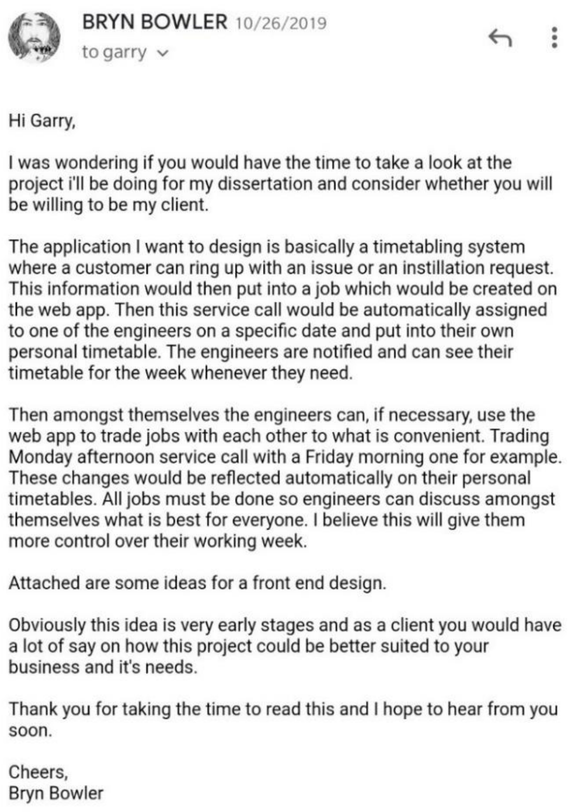
	Reliable	Easy to Setup	Secure	Flexible	Available on all OS
Content Management System	✓	✓	✓	✗	✓
WAMP Server	✓	✓	✓	✓	✗

Table Showing Web Hosting Tools Comparison

Appendix D – User Requirements Table post Tools and Languages Investigation

	Functional Requirements	Non-Functional Requirements
General	- Different Users - Job Calculating Algorithm - Client List - Google Calendar API - Notifications - Database/Application Error Triggers	- Free - Easy to Use - Easy to Navigate - Mobile Accessible - Well-presented information
Jobs	- Ability to create Jobs - Jobs assigned by a Receptionist - Google Maps API for Job Creation - Email to notify sites of cancelled jobs - Engineers can swap jobs	- Easy Job creation
Timetables	- Ability to view personal Timetables - Ability for a Manager to see all Timetables - Responsive Design - Separate List View for Timetabled Appointments - Asynchronous Design	- Easy to read and understand information
Database	- Can store ticket information - Can pass information between different tables	

Appendix E – Proof of Client Confirmation



BRYN BOWLER 10/26/2019

to garry

Hi Garry,

I was wondering if you would have the time to take a look at the project i'll be doing for my dissertation and consider whether you will be willing to be my client.

The application I want to design is basically a timetabling system where a customer can ring up with an issue or an instillation request. This information would then put into a job which would be created on the web app. Then this service call would be automatically assigned to one of the engineers on a specific date and put into their own personal timetable. The engineers are notified and can see their timetable for the week whenever they need.

Then amongst themselves the engineers can, if necessary, use the web app to trade jobs with each other to what is convenient. Trading Monday afternoon service call with a Friday morning one for example. These changes would be reflected automatically on their personal timetables. All jobs must be done so engineers can discuss amongst themselves what is best for everyone. I believe this will give them more control over their working week.

Attached are some ideas for a front end design.

Obviously this idea is very early stages and as a client you would have a lot of say on how this project could be better suited to your business and it's needs.

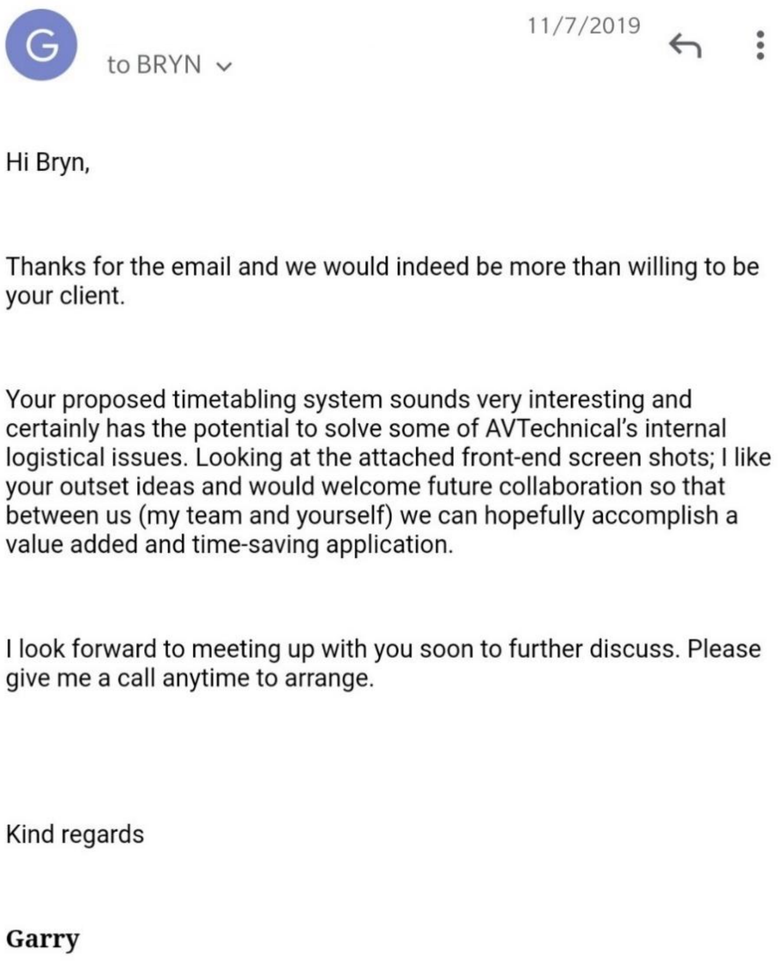
Thank you for taking the time to read this and I hope to hear from you soon.

Cheers,
Bryn Bowler

frontend2.PNG

frontend3.PNG

frontend1.PNG



G to BRYN 11/7/2019

Hi Bryn,

Thanks for the email and we would indeed be more than willing to be your client.

Your proposed timetabling system sounds very interesting and certainly has the potential to solve some of AVTechnical's internal logistical issues. Looking at the attached front-end screen shots; I like your outset ideas and would welcome future collaboration so that between us (my team and yourself) we can hopefully accomplish a value added and time-saving application.

I look forward to meeting up with you soon to further discuss. Please give me a call anytime to arrange.

Kind regards

Garry

Appendix F - User Requirements Interview Questions

User Requirements	Further Explanation
Creating a Job Sheet	What information is necessary for a job sheet?
Assigning a Job	Should a Job be randomly assigned? If not how would you like jobs to be assigned? Should there be a job board where users can assign themselves jobs? Should there be any limits on the amount of jobs one person can be assigned?
Job Swapping	Should job swapping be implemented? Should there be restrictions in place on job swapping? (Amount / Time before a Job)
Information Reviews	How would you want to be informed on information such as job swaps? Should an end of the month information sheet of jobs be created? If so how would you want to be informed of this? (E.g. Email or Through Application)
Further Functionality	Is there any other functionality you can think of that you would like to include?
Hosting this application	How could this application be hosted?

Appendix G – User Requirements Interview 27/12/19

Gary Whitaker: **G**

Bryn Bowler: **B**

Dan Brad and Neil: Engineers currently working at AVTECHINCAL

B: Okay so in the email we discussed the application itself being like a timetabling and scheduling application for the engineers to use. The idea being that a job is created, and it is displayed on their own personal weekly timetables. So, a job would come in and be assigned on a Friday and they can check through the week what jobs will be coming up. Makes things easier for them. There's just a couple of questions about the requirements. What you'd think would be good and how it would suit the business. First question would be:

<u>What kind of information is necessary on the job sheets?</u>

G: Yep, so that would be basics in terms of the date and time the issue came in. The full contact details of the client and then full details of the issue. I'm not entirely sure how the jobs will be allocated between engineers.

B: Yeah that's what I was going to ask you next. I was also going to mention that in the job sheets we could link it to google maps when you put in a location.

G: That's a great idea yeah, we use google maps on jobs already because it's a fantastic app.

B: Okay, moving on to how the jobs are actually assigned. Next question is:

<u>Should a job be randomly assigned?</u>

G: Umm, no. Somebody would have to assign that because there because there are so many tasks, we are doing on a daily basis that we'd have a conversation. So, it's more about a task being there and then after a conversation Dan's doing it, Brad's doing it or I'm doing it.

B: So possibly like a job board? Where jobs are up on a board and you can look and discuss where you can discuss with engineers or engineers can choose jobs.

G: Yeah yeah, like on a calendar type thing

B: <u>Should users be allowed to assign themselves jobs?</u>

G: Yeah, I think that's alright, because we work as a close team. It's a small company. I wouldn't want to think that I was somewhere else and one of my engineers would have to ask me if he could do a job. Obviously, the criteria are getting the job done as quickly as possible. I wouldn't want engineers assigning jobs to other people hahaha.

B: Haha okay okay.

G: No no that's fine because we all work as a team.

B: <u>Should there be any kind of restrictions or limits on the amount of jobs users can take on?</u>

G: By the way is this just limited to jobs like in terms of travelling to a job rather than tasks in house that just pop up.

B: Well if you want to include that functionality that could involve more specific things than just going off site…

G: Sorry sorry, what was the question again?

B: Should there be any kind of restrictions or limits on the amount of jobs users can take on?

G: Yeah um. I mean, probably but it's really hard to work that one out. Let's say, and this is not a real-life scenario, you were in an area with several sites that all needed little tasks applying at each site. That could be the same time consuming as doing just one job.

B: I was discussing this with my project supervisor. If I was to take this project further developing some kind of algorithm that works out distance between jobs and that amount of time a job could take.

G: Well if you wanna go down that road.

B: Well that is if I was to have more time with the project.

G: Well that's fair enough. That would be a very clever thing to do. We tend to know what is possible in a day and very often Neil might say can we add something else as well and I've had to say no because the lads are already working 12 hours a day. If they get on that site there that could be even longer.

B: Fair enough. So, in terms of job swapping which was one of the points of this application which makes it different to how you currently use outlook calendar. Say Dan had a job on Monday that had been assigned to him and Brad had one on Friday that had been assigned to him and they wanted to swap those jobs around.

<u>Should Job swapping be implemented and if so, what restrictions may need to be placed on it?</u>

For example, the amount of jobs that could be traded and maybe the amount of time before a job that a job can be traded so jobs aren't being traded on the morning they have to be completed.

G: Yeah but you can handle it however you want. I personally would trust them. In a real-life scenario, I would genuinely allow them to swap without restriction really and without any sort of time… As long as the job gets done and they are happy within their hours. They will have done it for a reason. It has to be via mutual agreement obviously. So, I don't see that being detrimental.

B: <u>How would you want to be informed and would you want to be informed on job swaps?</u>

To be honest, honestly. I get a lot of emails on a daily basis. I'm copied into this and copied into that and it's already a massive strain and I'm worried about missing important things. So certainly not an email. Maybe an alert as the apps opened.

B: <u>Would you be interested in an end of the month review?</u>

So how many jobs have been completed…

G: Yes!

B: Where they have been completed…

G: And miles travelled, that would be great. That's good analysis for us.

B: So, would you like to be informed of that through email or on the application itself?

G: Uhhh… I think maybe a tab on the application that gives you the stats. That would be great. Keep away from my emails. Unless you want to get an email thing in for your course and gets you more points.

B: No this is more about what's good for you as the client.

G: Ah, what's good for me is like the app opening up and a prompt.

B: Yeah.

G: Not every two minutes though hahaha.

B: Haha.

Okay last couple of questions.

<u>Is there any functionality you could think of that you would like to include?</u>

I know that's a bit of an open-ended question maybe something to think about for next time.

G: I can think about that if that's alright. I'll probably come up with some little tweaks to make it really difficult for you.

B: Hahahaha.

G: You know little features that are unique for us maybe?

B: Yeah, I'll probably come back to you for another interview in about 2 months when we have the basis.

G: Yeah, I don't think they'll be anything too major. Just maybe little things. Say that engineers were on site, because the is an online cloud thing. Potentially something along the lines of time spent on site.

B: Yeah.

G: And obviously some kind of review on what the estimated task was and some notes.

B: Like we already have on the job sheets?

G: Yeah yeah, information is king for us.

B: And a final question.

<u>How will the application be hosted?</u>

I was thinking possibly a WordPress site? Either its own WordPress site or through the WordPress site we've already got?

G: Yeah, we have a knowledge base. Would everyone have different logins for that?

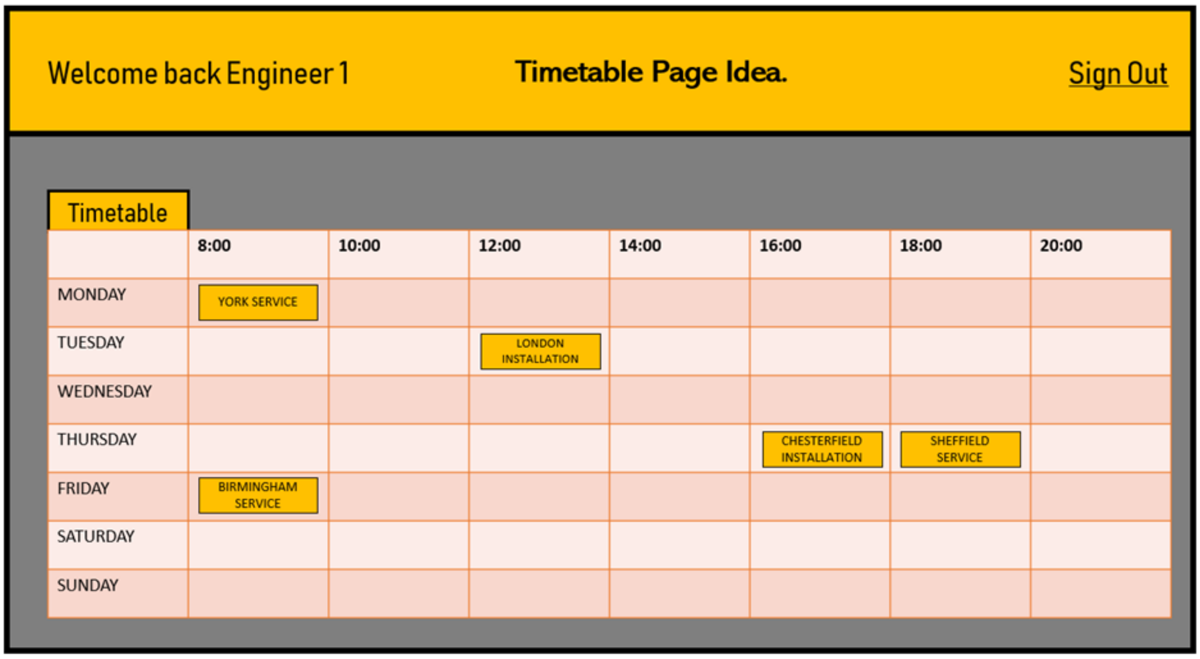
B: Yeah, there would be different logins for every and it could be accessed outside of AV and we get the free hosting.

G: I like the free bit.

B: Hahaha, I think that’s about everything.

G: Is that it? All good then.

Appendix H – Initial Proposed Designs to Demonstrate the Application Idea to the Client

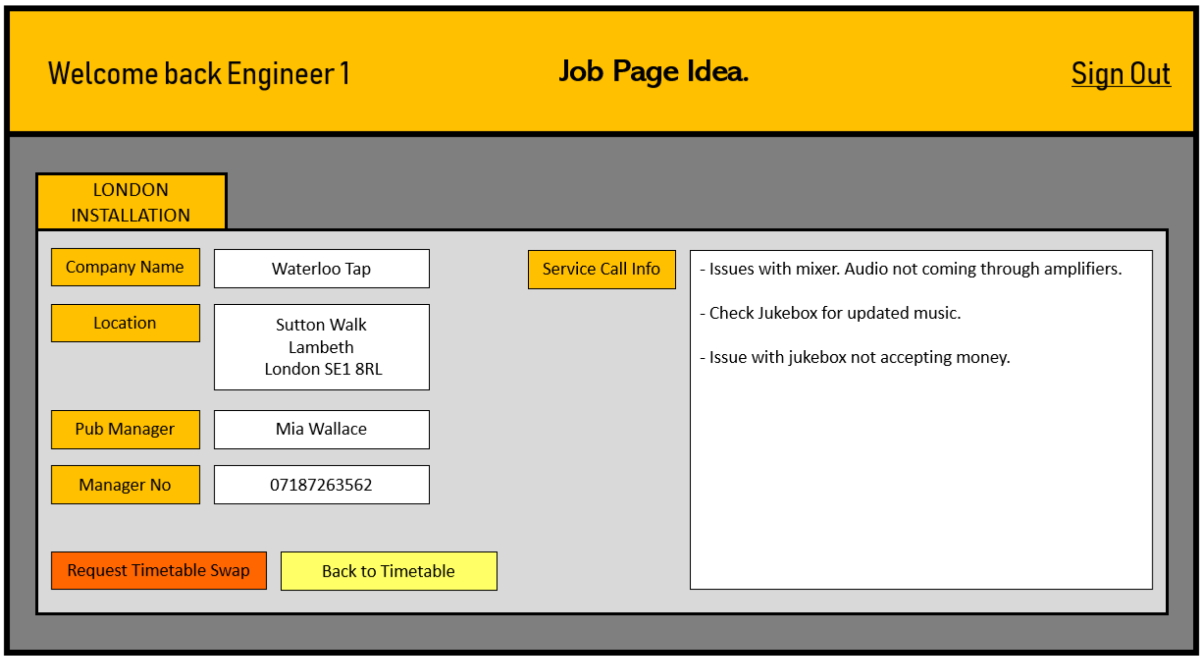


Welcome back Engineer 1 | Timetable Page Idea. | Sign Out

Timetable

	8:00	10:00	12:00	14:00	16:00	18:00	20:00
MONDAY	YORK SERVICE						
TUESDAY			LONDON INSTALLATION				
WEDNESDAY							
THURSDAY					CHESTERFIELD INSTALLATION	SHEFFIELD SERVICE	
FRIDAY	BIRMINGHAM SERVICE						
SATURDAY							
SUNDAY							

Appendix H1. Initial Mock-Up Desktop Timetable Page for Engineers



Welcome back Engineer 1 | Job Page Idea. | Sign Out

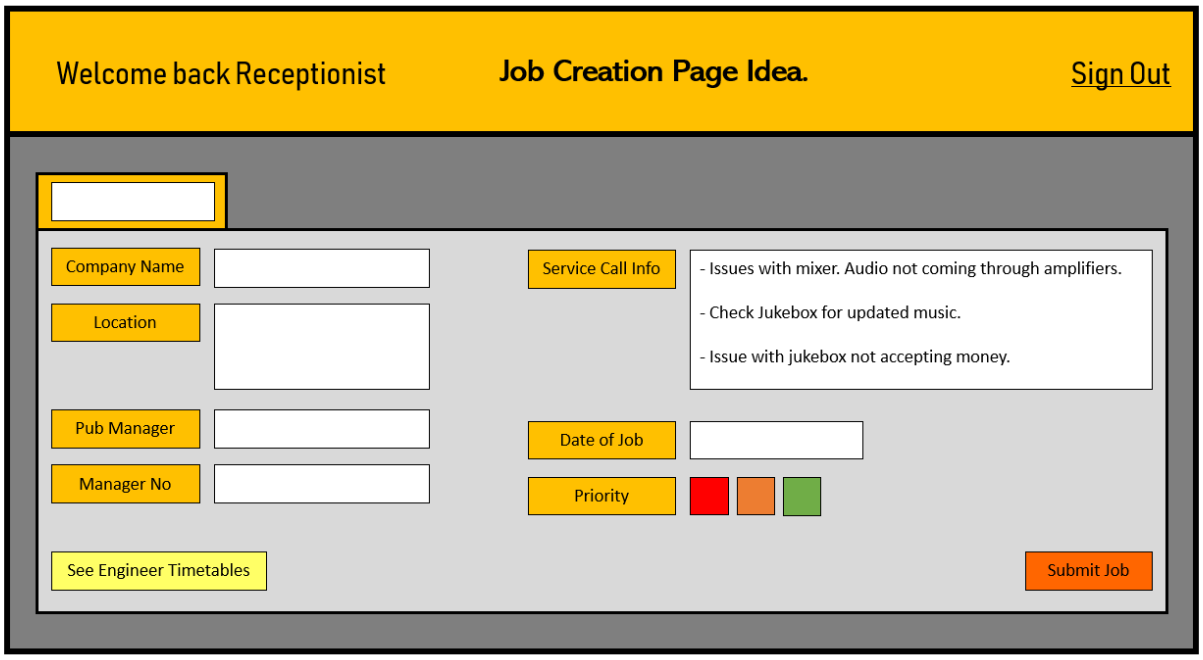
LONDON INSTALLATION

Company Name	Waterloo Tap
Location	Sutton Walk Lambeth London SE1 8RL
Pub Manager	Mia Wallace
Manager No	07187263562

Service Call Info

- Issues with mixer. Audio not coming through amplifiers.
- Check Jukebox for updated music.
- Issue with jukebox not accepting money.

Appendix H2. Job Dashboard Page Design



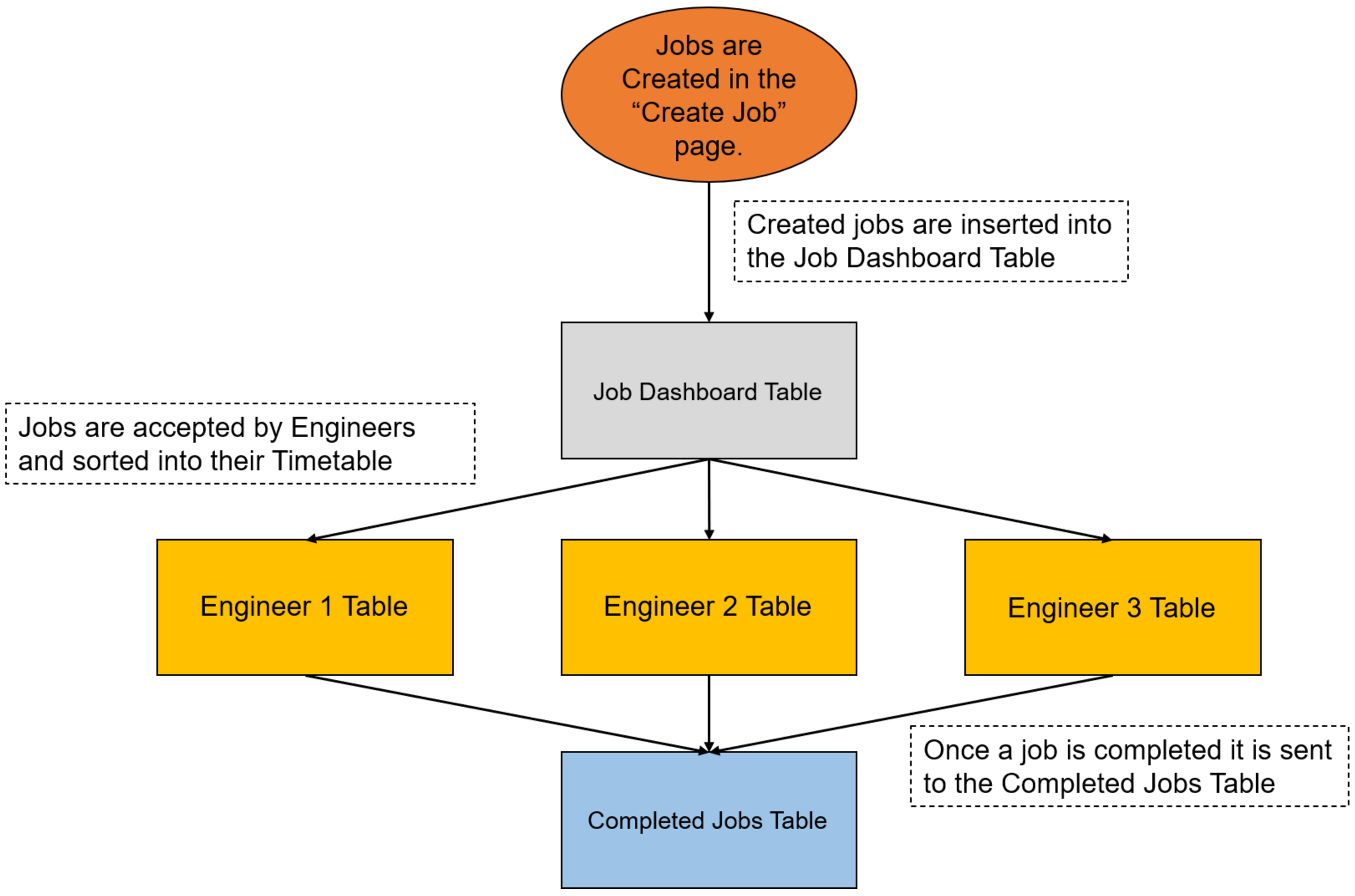
- Issues with mixer. Audio not coming through amplifiers.

- Check Jukebox for updated music.

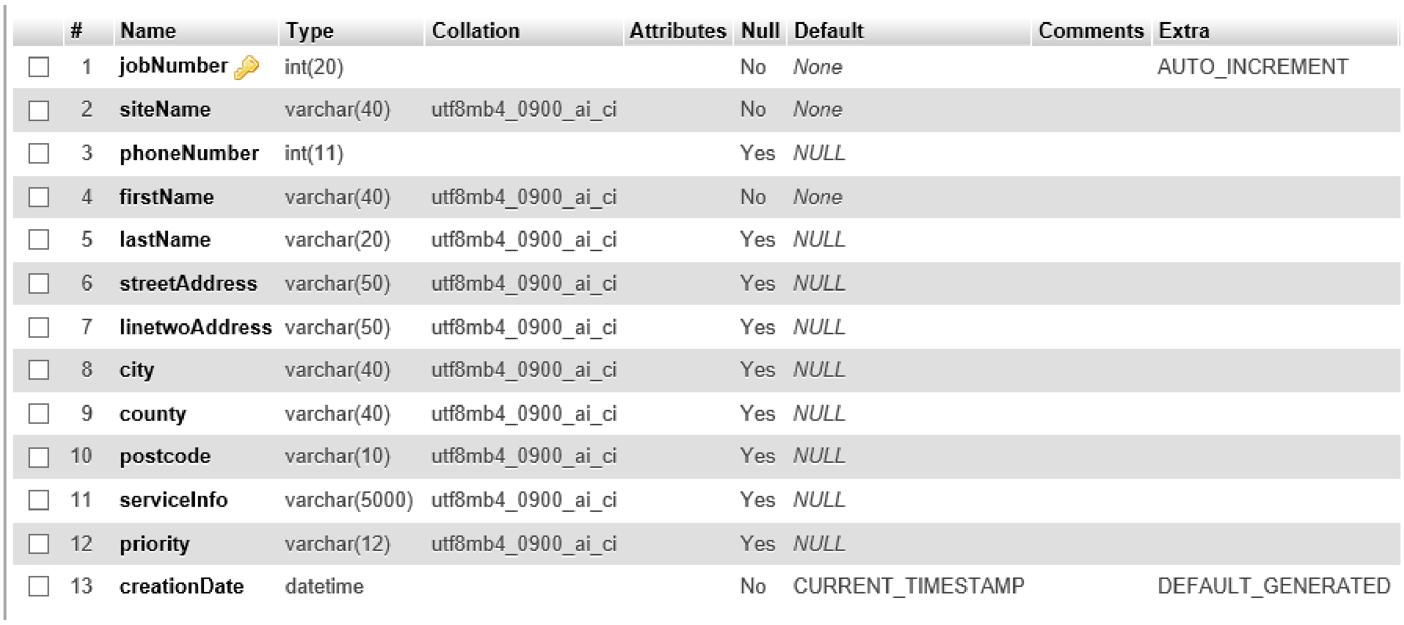
- Issue with jukebox not accepting money.

Appendix H3. Initial Mock-Up Desktop Job Creation Page

Appendix I – Diagram Showing how Data moves between Tables in the Database


Jobs are Created in the “Create Job” page.
Created jobs are inserted into the Job Dashboard Table
Job Dashboard Table
Jobs are accepted by Engineers and sorted into their Timetable
Engineer 1 Table
Engineer 2 Table
Engineer 3 Table
Once a job is completed it is sent to the Completed Jobs Table
Completed Jobs Table

Appendix J – Job Dashboard Table Structure



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	**jobNumber**	int(20)			No	*None*		AUTO_INCREMENT
☐	2	**siteName**	varchar(40)	utf8mb4_0900_ai_ci		No	*None*		
☐	3	**phoneNumber**	int(11)			Yes	*NULL*		
☐	4	**firstName**	varchar(40)	utf8mb4_0900_ai_ci		No	*None*		
☐	5	**lastName**	varchar(20)	utf8mb4_0900_ai_ci		Yes	*NULL*		
☐	6	**streetAddress**	varchar(50)	utf8mb4_0900_ai_ci		Yes	*NULL*		
☐	7	**linetwoAddress**	varchar(50)	utf8mb4_0900_ai_ci		Yes	*NULL*		
☐	8	**city**	varchar(40)	utf8mb4_0900_ai_ci		Yes	*NULL*		
☐	9	**county**	varchar(40)	utf8mb4_0900_ai_ci		Yes	*NULL*		
☐	10	**postcode**	varchar(10)	utf8mb4_0900_ai_ci		Yes	*NULL*		
☐	11	**serviceInfo**	varchar(5000)	utf8mb4_0900_ai_ci		Yes	*NULL*		
☐	12	**priority**	varchar(12)	utf8mb4_0900_ai_ci		Yes	*NULL*		
☐	13	**creationDate**	datetime			No	CURRENT_TIMESTAMP		DEFAULT_GENERATED

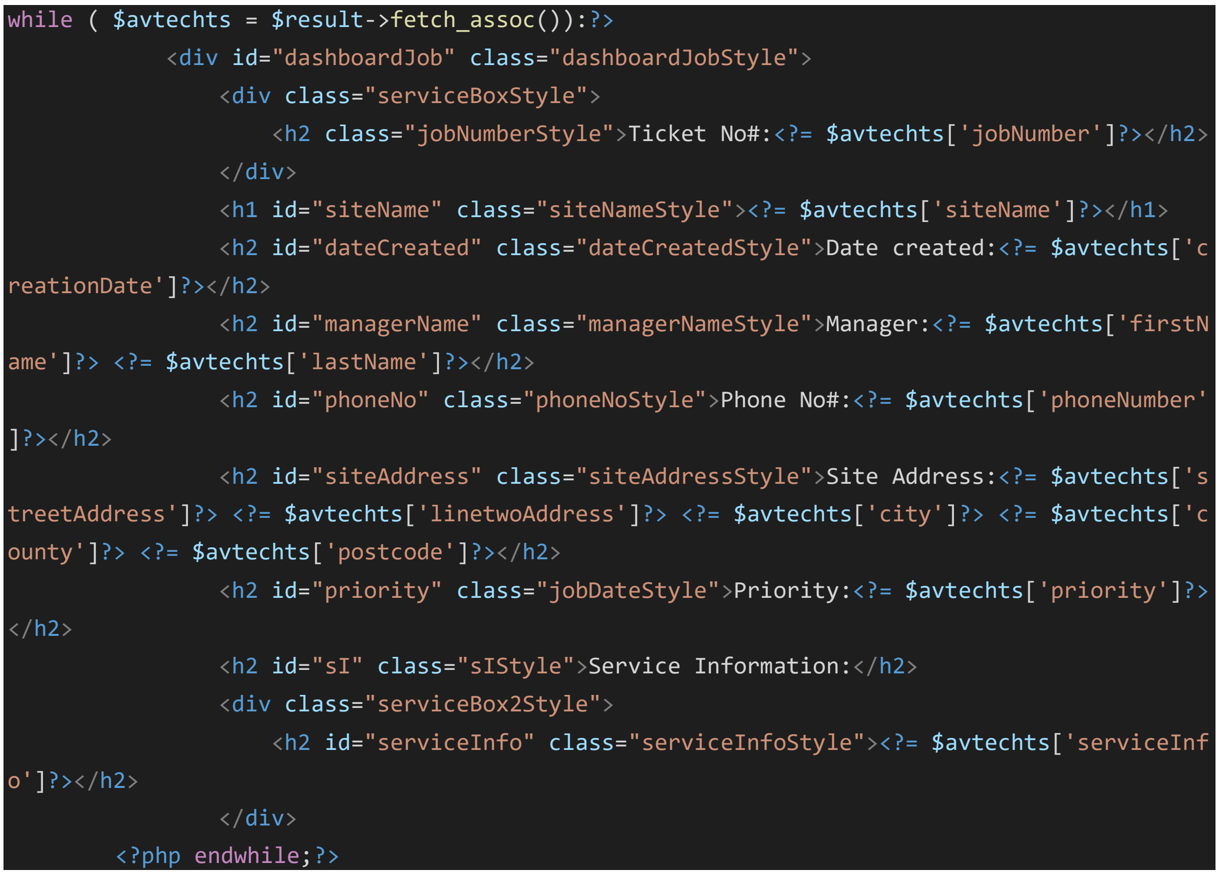
Appendix K – Job Creation Insert Code

```php
<?php

    $con =  mysqli_connect('localhost:3308','root','');

    if(!$con)
    {
        echo 'Not Connected To Server';
    }

    if(!mysqli_select_db($con, 'avtechts'))
    {
        echo 'Database Not Selected';
    }

    $jobNumber = $_POST['jobNumber'];
    $siteName = $_POST['siteName'];
    $phoneNumber = $_POST['phoneNumber'];
    $firstName = $_POST['firstName'];
    $lastName = $_POST['lastName'];
    $streetAddress = $_POST['streetAddress'];
    $linetwoAddress = $_POST['linetwoAddress'];
    $city = $_POST['city'];
    $county = $_POST['county'];
    $postcode = $_POST['postcode'];
    $serviceInfo = $_POST['serviceInfo'];
    $jobDate = $_POST['jobDate'];
    $priority_value = $_POST['priority'];

    $sql = "INSERT INTO avtechts (jobNumber,siteName,phoneNumber,firstName,lastNam
e,streetAddress,linetwoAddress,city,county,postcode,serviceInfo,jobDate,priority)
            VALUES ('$jobNumber','$siteName','$phoneNumber','$firstName','$lastNam
e','$streetAddress','$linetwoAddress','$city','$county','$postcode','$serviceInfo'
,'$jobDate','$priority_value')";

    if(!mysqli_query($con,$sql))
    {
        echo 'Not Inserted';
    }
    else
    {
        echo 'Inserted';
    }
    header("refresh:2 url=avtechreceptionist.php");
?>
```

Appendix L – SQL Query

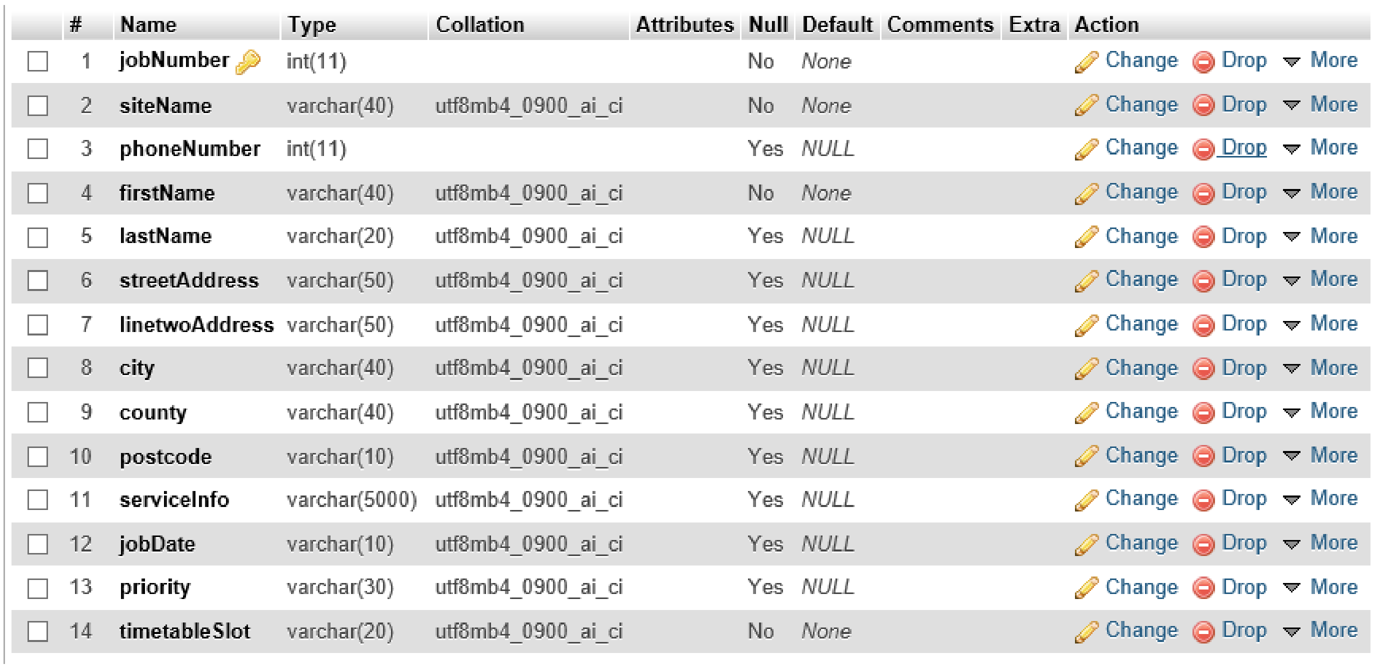
```
$host = 'localhost:3308';
        $user = 'root';
        $pass = '';
        $db = 'avtechts';

        $mysqli = new mysqli($host,$user,$pass,$db);

        $result = $mysqli->query
            ("SELECT * FROM avtechts ORDER BY creationDate DESC")
        or die($mysqli->error);
```

Appendix M – While Loop to display Job Tickets Code



```
while ( $avtechts = $result->fetch_assoc()):?>
            <div id="dashboardJob" class="dashboardJobStyle">
                <div class="serviceBoxStyle">
                    <h2 class="jobNumberStyle">Ticket No#:<?= $avtechts['jobNumber']?></h2>
                </div>
                <h1 id="siteName" class="siteNameStyle"><?= $avtechts['siteName']?></h1>
                <h2 id="dateCreated" class="dateCreatedStyle">Date created:<?= $avtechts['c
reationDate']?></h2>
                <h2 id="managerName" class="managerNameStyle">Manager:<?= $avtechts['firstN
ame']?> <?= $avtechts['lastName']?></h2>
                <h2 id="phoneNo" class="phoneNoStyle">Phone No#:<?= $avtechts['phoneNumber'
]?></h2>
                <h2 id="siteAddress" class="siteAddressStyle">Site Address:<?= $avtechts['s
treetAddress']?> <?= $avtechts['linetwoAddress']?> <?= $avtechts['city']?> <?= $avtechts['c
ounty']?> <?= $avtechts['postcode']?></h2>
                <h2 id="priority" class="jobDateStyle">Priority:<?= $avtechts['priority']?>
</h2>
                <h2 id="sI" class="sIStyle">Service Information:</h2>
                <div class="serviceBox2Style">
                    <h2 id="serviceInfo" class="serviceInfoStyle"><?= $avtechts['serviceInf
o']?></h2>
                </div>
        <?php endwhile;?>
```

Appendix N – Engineer Table Structure

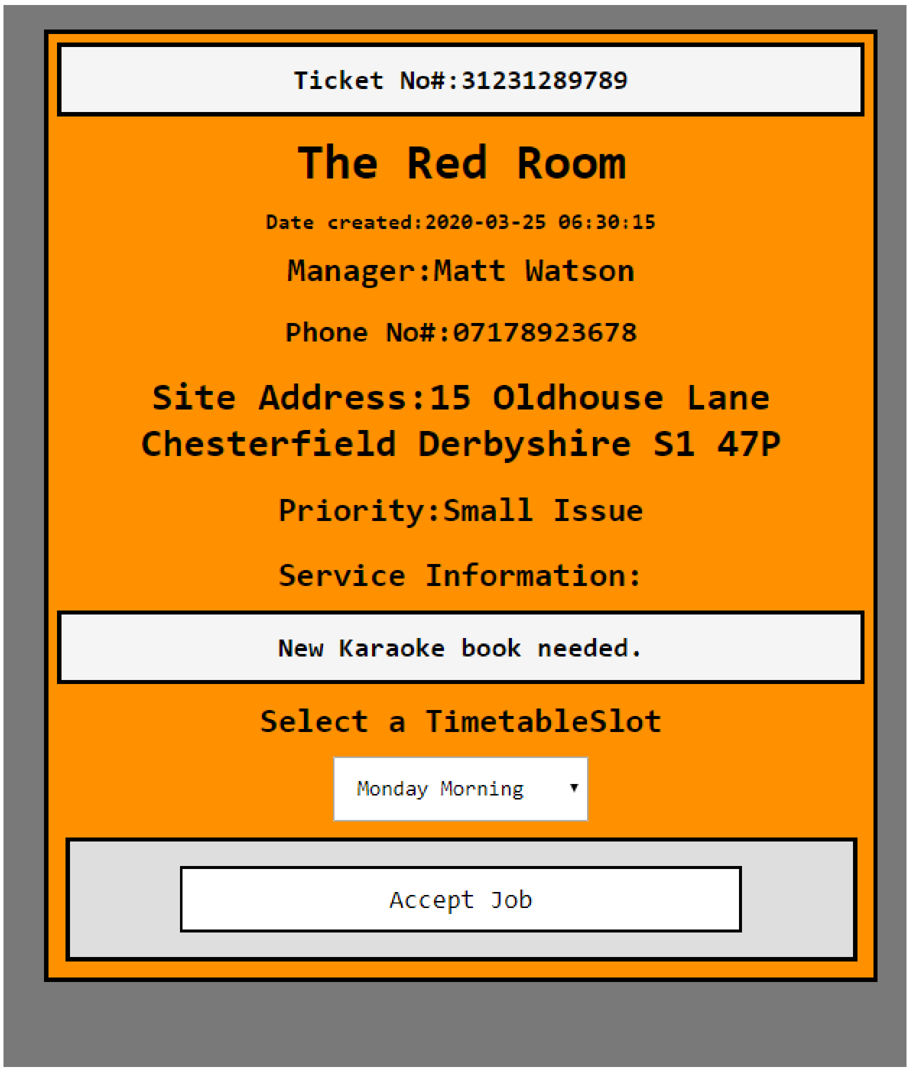


	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	jobNumber	int(11)			No	*None*			Change Drop More
☐	2	siteName	varchar(40)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More
☐	3	phoneNumber	int(11)			Yes	*NULL*			Change Drop More
☐	4	firstName	varchar(40)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More
☐	5	lastName	varchar(20)	utf8mb4_0900_ai_ci		Yes	*NULL*			Change Drop More
☐	6	streetAddress	varchar(50)	utf8mb4_0900_ai_ci		Yes	*NULL*			Change Drop More
☐	7	linetwoAddress	varchar(50)	utf8mb4_0900_ai_ci		Yes	*NULL*			Change Drop More
☐	8	city	varchar(40)	utf8mb4_0900_ai_ci		Yes	*NULL*			Change Drop More
☐	9	county	varchar(40)	utf8mb4_0900_ai_ci		Yes	*NULL*			Change Drop More
☐	10	postcode	varchar(10)	utf8mb4_0900_ai_ci		Yes	*NULL*			Change Drop More
☐	11	serviceInfo	varchar(5000)	utf8mb4_0900_ai_ci		Yes	*NULL*			Change Drop More
☐	12	jobDate	varchar(10)	utf8mb4_0900_ai_ci		Yes	*NULL*			Change Drop More
☐	13	priority	varchar(30)	utf8mb4_0900_ai_ci		Yes	*NULL*			Change Drop More
☐	14	timetableSlot	varchar(20)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More

Appendix O – Accept Job Code and Updated look of Tickets

```
<form id="acceptJobInsert" display="hidden" action="acceptJob1.php" method="post">
                    <select id="assignDateInsert" name="assignDateInsert"">
                        <option value="mondayMorning">Monday Morning</option>
                        <option value="mondayMidday">Monday Midday</option>
                        <option value="mondayEvening">Monday Evening</option>
                        <option value="mondayNight">Monday Night</option>
                        <option value="tuesdayMorning">Tuesday Morning</option>
                        <option value="tuesdayMidday">Tuesday Midday</option>
                        <option value="tuesdayEvening">Tuesday Evening</option>
                        <option value="tuesdayNight">Tuesday Night</option>
                        <option value="wednesdayMorning">Wednesday Morning</option
>
                        <option value="wednesdayMidday">Wednesday Midday</option>
                        <option value="wednesdayEvening">Wednesday Evening</option
>
                        <option value="wednesdayNight">Wednesday Night</option>
                        <option value="thursdayMorning">Thursday Morning</option>
                        <option value="thursdayMidday">Thursday Midday</option>
                        <option value="thursdayEvening">Thursday Evening</option>
                        <option value="thursdayNight">Thursday Night</option>
                        <option value="fridayMorning">Friday Morning</option>
                        <option value="fridayMidday">Friday Midday</option>
                        <option value="fridayEvening">Friday Evening</option>
                        <option value="fridayNight">Friday Night</option>
                    </select>
                <div class="buttonDivStyle">
                        <input id="jobNumberInsert" name="jobNumberInsert" type="h
idden" value="<?= $avtechts['jobNumber']?>">
                        <input id="siteNameInsert" name="siteNameInsert" type="hid
den" value="<?= $avtechts['siteName']?>">
                        <input id="firstNameInsert" name="firstNameInsert" type="h
idden" value="<?= $avtechts['firstName']?>">
                        <input id="lastNameInsert" name="lastNameInsert" type="hid
den" value="<?= $avtechts['lastName']?>">
                        <input id="phoneNoInsert" name="phoneNoInsert" type="hidde
n" value="<?= $avtechts['phoneNumber']?>">
                        <input id="streetAddressInsert" name="streetAddressInsert"
 type="hidden" value="<?= $avtechts['streetAddress']?>">
                        <input id="linetwoAddressInsert" name="linetwoAddressInser
t" type="hidden" value="<?= $avtechts['linetwoAddress']?>">
                        <input id="cityInsert" name="cityInsert" type="hidden" val
ue="<?= $avtechts['city']?>">
                        <input id="countyInsert" name="countyInsert" type="hidden"
 value="<?= $avtechts['county']?>">
```

```
                        <input id="postcodeInsert" name="postcodeInsert" type="hid
den" value="<?= $avtechts['postcode']?>">
                        <input id="serviceInfoInsert" name="serviceInfoInsert" typ
e="hidden" value="<?= $avtechts['serviceInfo']?>">
                        <input id="priorityInsert" name="priorityInsert" type="hid
den" value="<?= $avtechts['priority']?>">

                        <input id="sendInsert" class="button_text" type="submit" n
ame="submit" value="Accept Job">
                    </form>
```



Ticket No#:31231289789

The Red Room

Date created:2020-03-25 06:30:15

Manager:Matt Watson

Phone No#:07178923678

Site Address:15 Oldhouse Lane Chesterfield Derbyshire S1 47P

Priority:Small Issue

Service Information:

New Karaoke book needed.

Select a TimetableSlot

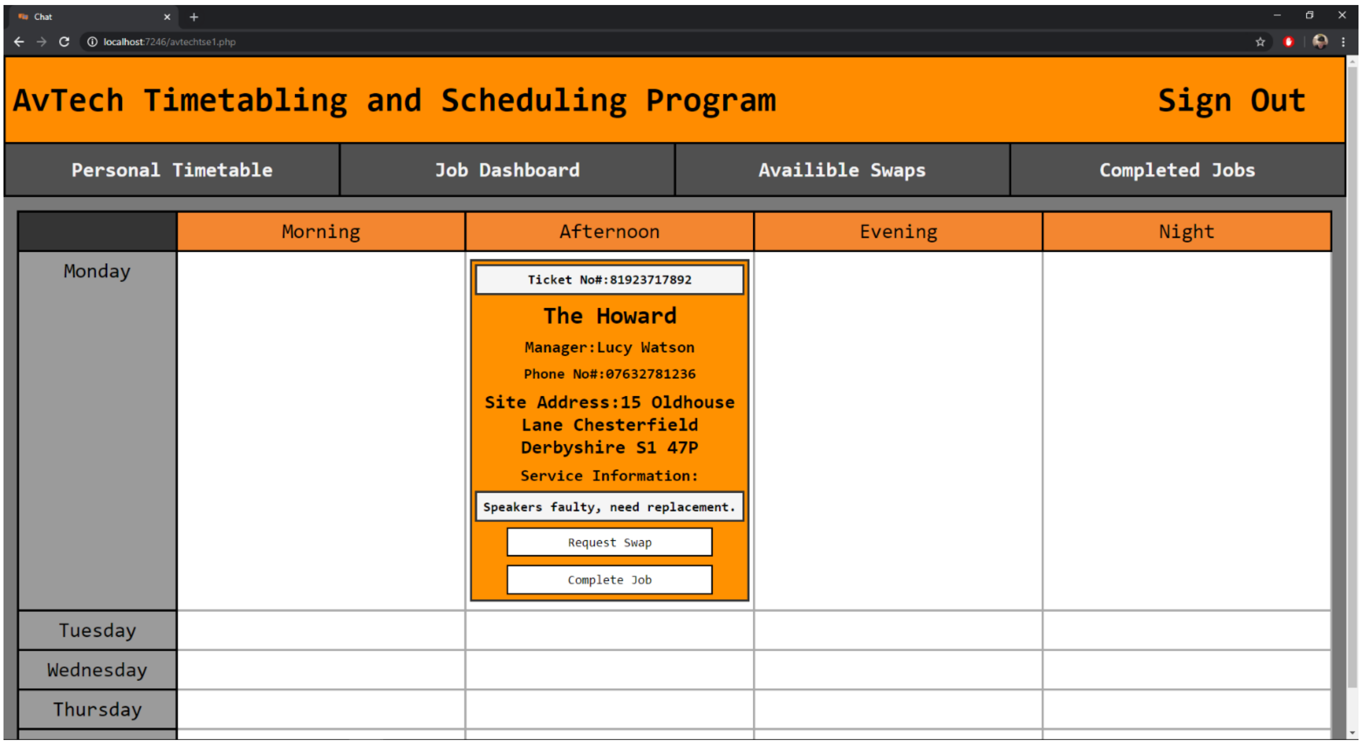
Monday Morning

Accept Job

Appendix P – Code for Completeing Jobs

```php
<?php
    $con =  mysqli_connect('localhost:3308','root','');
    if(!$con)
    {
        echo 'Not Connected To Server';
    }

    if(!mysqli_select_db($con, 'avtechts'))
    {
        echo 'Database Not Selected';
    }

    $jobNumber = $_POST['jobNumberInsert'];
    $siteName = $_POST['siteNameInsert'];
    $phoneNumber = $_POST['phoneNoInsert'];
    $firstName = $_POST['firstNameInsert'];
    $lastName = $_POST['lastNameInsert'];
    $streetAddress = $_POST['streetAddressInsert'];
    $linetwoAddress = $_POST['linetwoAddressInsert'];
    $city = $_POST['cityInsert'];
    $county = $_POST['countyInsert'];
    $postcode = $_POST['postcodeInsert'];
    $serviceInfo = $_POST['serviceInfoInsert'];
    $priority_value = $_POST['priorityInsert'];

    $sql = "INSERT INTO completedJobs (jobNumber,siteName,phoneNumber,firstNam
e,lastName,streetAddress,linetwoAddress,city,county,postcode,serviceInfo,prior
ity)
            VALUES ('$jobNumber','$siteName','$phoneNumber','$firstName','$las
tName','$streetAddress','$linetwoAddress','$city','$county','$postcode','$serv
iceInfo','$priority_value');";

    $sql .= "DELETE FROM e1timetable WHERE jobNumber= $jobNumber";

    if(!mysqli_multi_query($con,$sql))
    {
        echo 'Not Inserted or Deleted';
    }
    else
    {
        echo 'Inserted and Deleted';
    }
    header("refresh:2 url=avtechtse1.php");
?>
```

Appendix Q – Proof of Testing on Different Web Browsers

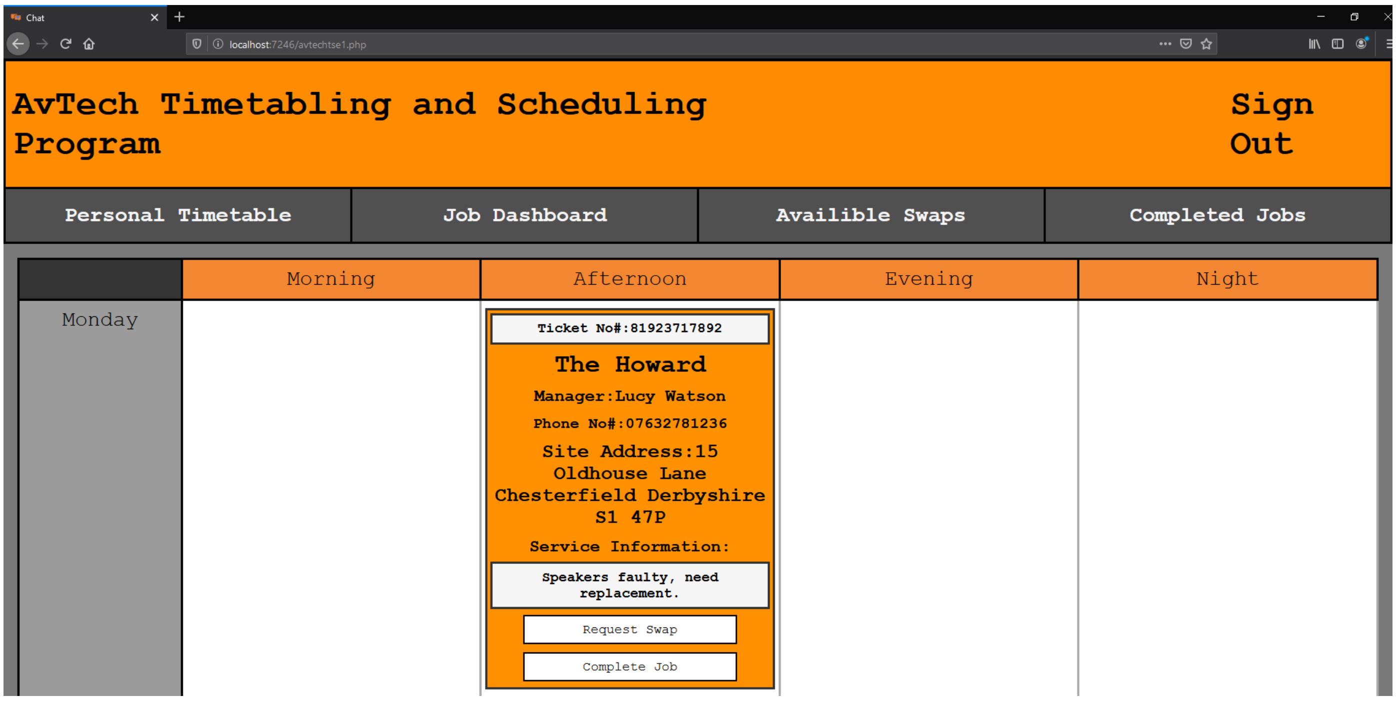


AvTech Timetabling and Scheduling Program | Sign Out

Personal Timetable | Job Dashboard | Availible Swaps | Completed Jobs

	Morning	Afternoon	Evening	Night
Monday		Ticket No#:81923717892 The Howard Manager:Lucy Watson Phone No#:07632781236 Site Address:15 Oldhouse Lane Chesterfield Derbyshire S1 47P Service Information: Speakers faulty, need replacement. Request Swap Complete Job		
Tuesday				
Wednesday				
Thursday				

Appendix Q1. Example of Application working in Chrome

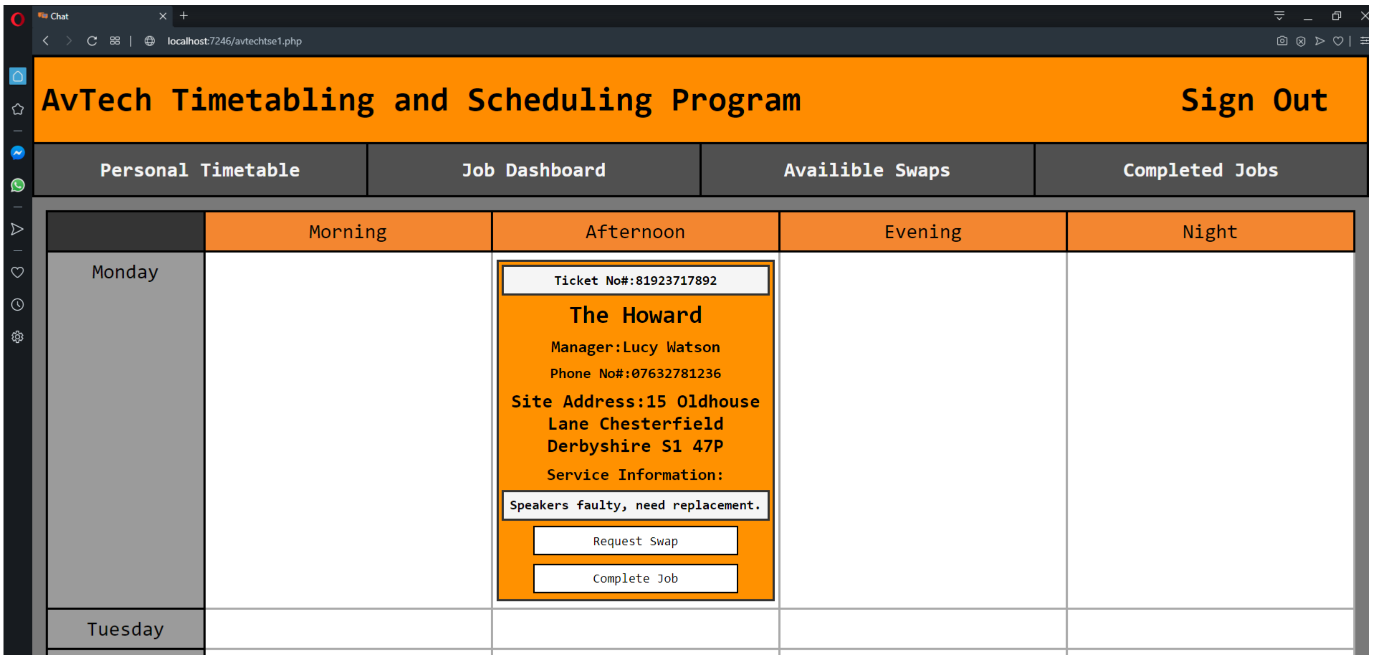


AvTech Timetabling and Scheduling Program | Sign Out

Personal Timetable | Job Dashboard | Availible Swaps | Completed Jobs

	Morning	Afternoon	Evening	Night
Monday		Ticket No#:81923717892 The Howard Manager:Lucy Watson Phone No#:07632781236 Site Address:15 Oldhouse Lane Chesterfield Derbyshire S1 47P Service Information: Speakers faulty, need replacement. Request Swap Complete Job		

Appendix Q2. Example of application working in Firefox



Chat

localhost:7246/avtechtse1.php

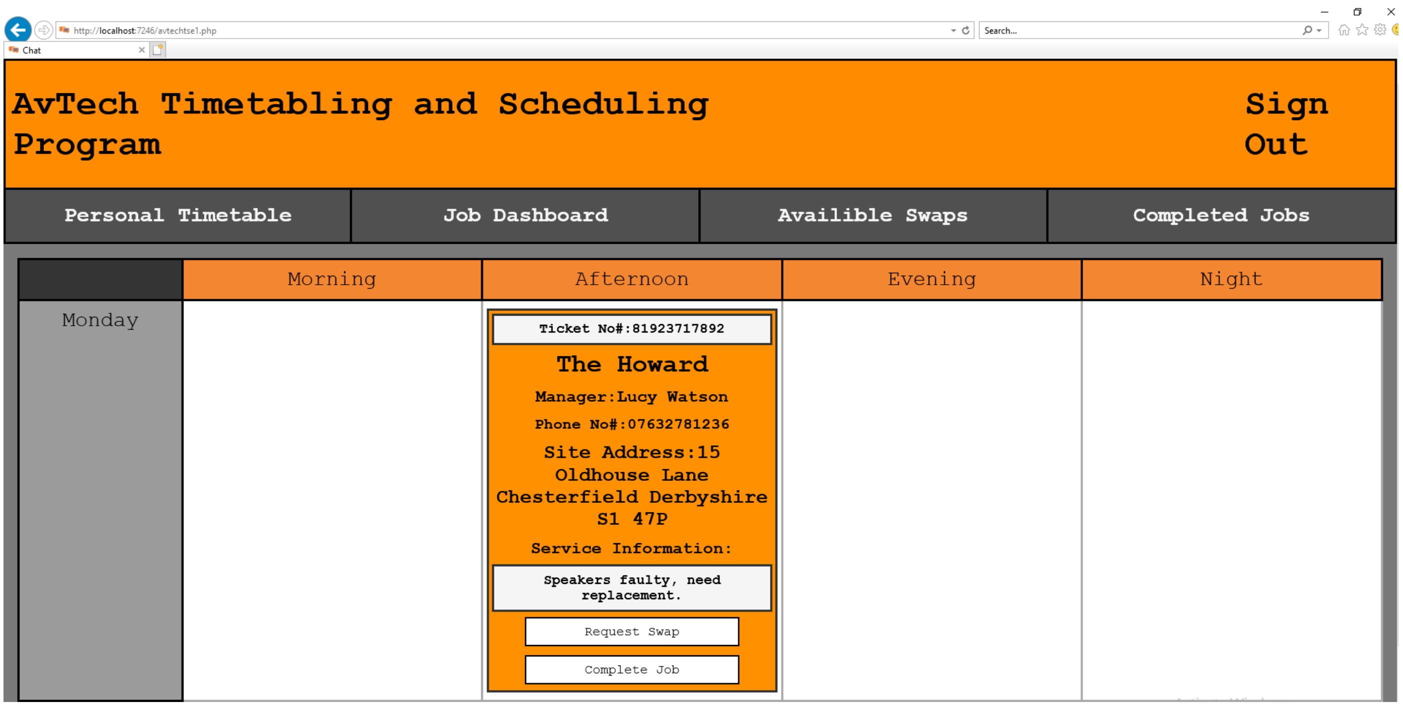
AvTech Timetabling and Scheduling Program

Sign Out

Personal Timetable | Job Dashboard | Availible Swaps | Completed Jobs

	Morning	Afternoon	Evening	Night
Monday		Ticket No#:81923717892 The Howard Manager:Lucy Watson Phone No#:07632781236 Site Address:15 Oldhouse Lane Chesterfield Derbyshire S1 47P Service Information: Speakers faulty, need replacement. Request Swap Complete Job		
Tuesday				

Appendix Q3. Example of application working in Opera



http://localhost:7246/avtechtse1.php

Search...

Chat

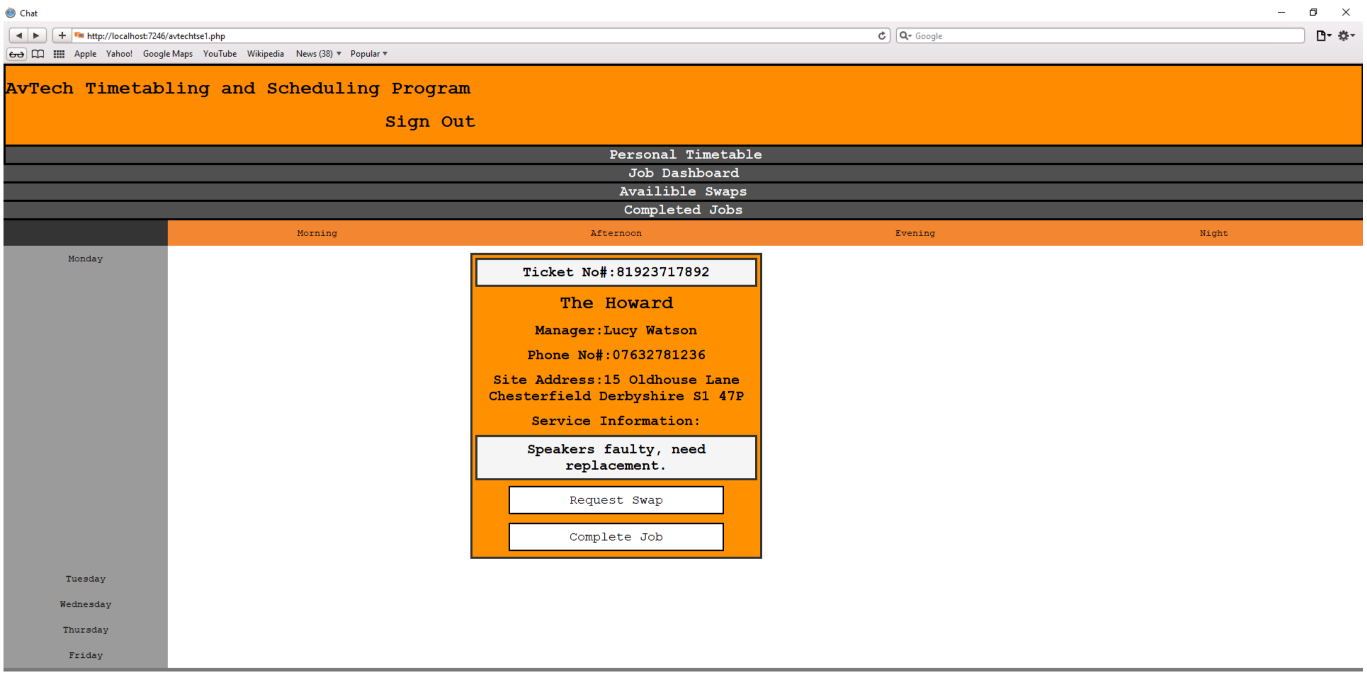
AvTech Timetabling and Scheduling Program

Sign Out

Personal Timetable | Job Dashboard | Availible Swaps | Completed Jobs

	Morning	Afternoon	Evening	Night
Monday		Ticket No#:81923717892 The Howard Manager:Lucy Watson Phone No#:07632781236 Site Address:15 Oldhouse Lane Chesterfield Derbyshire S1 47P Service Information: Speakers faulty, need replacement. Request Swap Complete Job		

Appendix Q4. Example of application working in Internet Explorer


Chat
http://localhost:7246/avtechtse1.php
Google
Apple Yahoo! Google Maps YouTube Wikipedia News (38) Popular

AvTech Timetabling and Scheduling Program

Sign Out

Personal Timetable
Job Dashboard
Availible Swaps
Completed Jobs

	Morning	Afternoon	Evening	Night
Monday		Ticket No#:81923717892 The Howard Manager:Lucy Watson Phone No#:07632781236 Site Address:15 Oldhouse Lane Chesterfield Derbyshire S1 47P Service Information: Speakers faulty, need replacement. Request Swap Complete Job		
Tuesday				
Wednesday				
Thursday				
Friday				

Appendix Q5. Example of application working in Safari

Appendix R – Testing Feedback Forms

Engineer Feedback Form

Name: Brad

Application Areas	Usability Rating /5	Feedback
Login	5	The ability to change a password
Personal Timetable Page	4	Instead of morning afternoon evening and night, change to hourly slots
Job Dashboard Page	4	Have the date the correct way around, English not American. Like the job tickets, displays useful information in a nice way.
Request Swap Page	4	I like this feature. Could do with a feature that lets you ask to swap jobs from other people's timetables to make it simpler.
Completed Jobs Page	3	Should be able to add the serial number of stocks, change the colour of completed tickets to green, the ability to add notes to completed jobs.
Overall Application	4	Would 100% use this now, helps keep track of information. The additions of timesheets that get sent through to a receptionist. Message engineers. Personalized messages. Like the overall look, simple to find things.

Engineer Feedback Form

Name: Dan

Application Areas	**Usability Rating /5**	**Feedback**
Login	5	Simple as it should be.
Personal Timetable Page	4	Looks good. Ability to choose slots that cannot be worked.
Job Dashboard Page	5	Change the date.
Request Swap Page	3	A note showing who has requested the swap. Note section that allows you to provide a notice or a reason. Name of the user who wants to swap.
Completed Jobs Page	4	Error completing a certain job Before you can complete a job add some notes to it. Be able to upload pictures to job tickets.
Overall Application	5	Could be viable to use in the future of the business. Customised background.

Engineer Feedback Form

Name: Neil

Application Areas	**Usability Rating /5**	**Feedback**
Login	5	Just a login page.
Personal Timetable Page	3	Be able to flick through multiple weeks.
Job Dashboard Page	4	Able to move jobs without having to send a swap request. Not be able to complete jobs before the job date.
Completed Jobs Page	2	Be able to edit completed jobs with notes about what has been done at the site. Be able to see who the job was completed by. Be able to filter completed jobs by who completed them, site name, date and more.
Request Swap Page	4	Should get to choose who you could request a swap with. Should be able to deny swaps Error displaying some information
Overall Application	4	Add a photos tab to see users and job site information. Isn’t overcomplicated.

Receptionist Feedback Form

Name: Cheryl

Application Areas	**Usability Rating /5**	**Feedback**
Login	5	Login is easy and simple to use.
Job Creation Page	3	Missing some information such as serial numbers. Would have liked to have had the google maps added to see the route to a pub. The form to add information is very helpful. It makes sure I don't forget to include all the information.
Job Dashboard Page	4	Easy to read but missing information.
Completed Jobs Page	4	Easy to read good for keeping track of what has been completed.
Overall Application	4	Easy to use and makes creating jobs for the engineers easier than before. Keeps track of information which is super useful.

Manager Feedback Form

Name: Garry

Application Areas	Usability Rating /5	Feedback
Login	5	Works as intended.
Timetable Overview Page	4	Personal timetable for all users. Be able to see upcoming weeks and a month view. Another tab to create smaller tasks.
Job Dashboard Page	4	Edit tickets to add more information.
Completed Jobs Page	4	A filter. Completed jobs are filtered by weeks and months. A smaller ticket size with a page that expands to show more information. A search feature that can search clients and engineers.
Available Swaps	3	Like the idea but having more engineers would make this feature more viable.
Job Creation Page	4	The form makes it clear what information is needed for a job ticket. Could do with a notes section. Easy for a receptionist to create jobs.
Overall Application	3	A statistics page would still be useful. A time sheets page that can show what has been done that can record when an engineer arrives and leaves a site.

		Time sheets sent to receptionist at the end of the month. Practicality would come from being able to assign more than one engineer to a job. Separate different service issues by type. Install, fix, in house fix, phone service call. Logging information of the workday. Everyone can see what jobs have been accepted. Everyone can see timetables. Like the look of the application and the basic features work well, adding some more stuff that the application can do to benefit the business would make me genuinely consider this application for real use.

References

Acuity Scheduling. (2020, 03 10). *Acuity Scheduling*. Retrieved from acuityscheduling.com: https://acuityscheduling.com/

Assila, A., Mar, t., Oliveira, a. d., & Ezzedine, H. (2016). Standardized Usability Questionnaires: Features and Quality Focus. *Electronic Journal of Computer Science and Information Technology*, 15-18.

Bolton, D. (2017, 09 14). *Exploring the Dart Programming Language*. Retrieved from dice.com: https://insights.dice.com/2017/09/14/exploring-dart-programming-language/

Bookafy. (2020, 03 05). *Bookafy.com*. Retrieved from www.bookafy.com: https://www.bookafy.com/

DART. (2020). *Dart testing*. Retrieved from dart.dev: https://dart.dev/guides/testing

DART. (2020). *dart:html library*. Retrieved from api.dart.dev: https://api.dart.dev/stable/2.7.2/dart-html/dart-html-library.html

Dawson, A. (2012). *Future-Proof Web Design.* Cheshire: John Wiley & Sons.

Dobrzanski, M. (2013, Feburary 6). *MySQL Security: Overview of MySQL Security Features*. Retrieved from psce.com: https://www.psce.com/en/blog/2013/02/06/mysql-security-overview-of-mysql-security-features/

Duckett, J. (2014). *Javascript & Jquery - Interactive front-end web development.* Wiley.

Elliot, E. (2019, 5 11). *How Popular is JavaScript in 2019?* Retrieved from medium.com: https://medium.com/javascript-scene/how-popular-is-javascript-in-2019-823712f7c4b1

G. C. W. Sabin, G. K. (2017). The Impact of Automated Timetabling on Universities-A Case Study. *The Journal of the Operational Research Society*, 689-693.

Google. (2020, 02 07). *Google Calendar*. Retrieved from calendar.google.com: https://calendar.google.com/calendar/r

International Software Testing Qualifications Board. (2018). *ISTQB Glossary.* Retrieved from glossary.istqb.org: https://glossary.istqb.org/en/search/Unit%20testing

Khadija Sania Ahmad, N. A. (2017). Fuzzy_MoSCoW: A Fuzzy based MoSCoW Method. *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, 1.

Kolici, V., Xhafa, F., Pinedo, E. E., Segui, V., & Barolli, L. (2013). Analysis of Mobile and Web Applications in Small and Medium Size Enterprises. *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing* (pp. 324-330). IEEE.

Krush, A. (2018, January 13). *5 Success Stories That Will Make You Believe in Scaled Agile*. Retrieved from objectstyle.com: https://www.objectstyle.com/agile/scaled-agile-success-story-lessons

M. Duran ToksarJ, D. O. (2014). New Developments in Scheduling Applications. 1-2.

Michael W. Carter, G. L. (1996). Examination Timetabling: Algorithmic Strategies and Applications. *The Journal of the Operational Research Society*, 1-2.

Microsoft. (2017, 03 30). *Overview of SQL Server Security*. Retrieved from docs.microsoft.com: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql/overview-of-sql-server-security

Mosh Hamadani. (2018, January 23). *What is Node.js? | Mosh*. Retrieved from Youtube.com: https://www.youtube.com/watch?v=uVwtVBpw7RQ

MySQL. (2020). *MySQL Restrictions and Limitations.* Retrieved from dev.mysql.com: https://dev.mysql.com/doc/mysql-reslimits-excerpt/5.5/en/

Orn, A. (2017, July 13). *The Pros and Cons of the System Usability Scale (SUS)*. Retrieved from research-collective.com: https://research-collective.com/blog/sus/

Panchal, J. (2015, 12 29). *Pros and Cons of AJAX*. Retrieved from dzone.com: https://dzone.com/articles/pros-and-cons-of-ajax

Pauline M. Berry, C. A.-S. (2006). Conflict negotiation among personal calendar agents. *AAMAS '06 Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 1467-1468.

Pauline M. Berry, M. G.-S. (2011). PTIME: Personalized assistance for calendaring. *ACM Transactions on Intelligent Systems and Technology*, 40-60.

Petersen, K. (2009). The Waterfall Model in Large-Scale Development. *Product-Focused Software Process Improvement*, 11.

phpsecurity. (2017). *PHP Security: Default Vulnerabilities, Security Omissions and Framing Programmers?* Retrieved from phpsecurity.readthedocs.io: https://phpsecurity.readthedocs.io/en/latest/_articles/PHP-Security-Default-Vulnerabilities-Security-Omissions-And-Framing-Programmers.html

Rouse, M. (2019, August). *HTML (Hypertext Markup Language)*. Retrieved from theserverside.com: https://www.theserverside.com/definition/HTML-Hypertext-Markup-Language

Rudy Oude Vrielink, E. J. (2017). Towards improving tenders for Higher Education timetabling software: Uncovering the selection criteria of HEIs when choosing timetabling applications, using ERP as a reference. *Industrial Engineering & Business Information Systems*.

Strickland, J. (2008, November 6th). *How Google Calendar Works*. Retrieved from howstuffworks.com: https://computer.howstuffworks.com/internet/basics/google-calendar3.htm

Ten Ton. (2019, April 26). *Is HTML Easy To Learn? Skirt The Mumbo & Unravel The Truth!* Retrieved from tentononline.com/: https://www.tentononline.com/is-html-easy-to-learn/

tutorialspoint. (2020). *AJAX - Browser Support*. Retrieved from tutorialspoint.com: https://www.tutorialspoint.com/ajax/ajax_browser_support.htm

Vandim Tkachenko, A. L. (2008). Chapter 4. Query Performance Optimization. In A. L. Vandim Tkachenko, *High Performance MySQL, 2nd Edition.* O'Reilly Media, Inc.

W3Counter. (2020, 2). *Browser & Platform Market Share*. Retrieved from https://www.w3counter.com: https://www.w3counter.com/globalstats.php

W3Techs. (2020, 1 6). *Usage statistics of HTML5 for Websites*. Retrieved from w3techs.com: https://w3techs.com/technologies/details/ml-html5

Wassim Sellil Atoui, W. A. (2018). Offline and Online Scheduling Algorithms for Energy Harvesting RSUs in VANETs. *IEEE Transactions on Vehicular Technology*.

Whong, W. (2013). The Importance of a Software Development Methodology in IT Project Management: An Innovative Six Sigma Approach. *Department of Mechanical, Materials & Manufacturing Engineering*.

Bibliography

Acuity Scheduling. (2020, 03 10). *Acuity Scheduling*. Retrieved from acuityscheduling.com: https://acuityscheduling.com/

Assila, A., Mar, t., Oliveira, a. d., & Ezzedine, H. (2016). Standardized Usability Questionnaires: Features and Quality Focus. *Electronic Journal of Computer Science and Information Technology*, 15-18.

Bolton, D. (2017, 09 14). *Exploring the Dart Programming Language*. Retrieved from dice.com: https://insights.dice.com/2017/09/14/exploring-dart-programming-language/

Bookafy. (2020, 03 05). *Bookafy.com*. Retrieved from www.bookafy.com: https://www.bookafy.com/

DART. (2020). *Dart testing*. Retrieved from dart.dev: https://dart.dev/guides/testing

DART. (2020). *dart:html library*. Retrieved from api.dart.dev: https://api.dart.dev/stable/2.7.2/dart-html/dart-html-library.html

Dawson, A. (2012). *Future-Proof Web Design.* Cheshire: John Wiley & Sons.

Dillet, R. (2018, July 2). *Browser maker Opera has filed to go public*. Retrieved from Techcrunch.com: https://techcrunch.com/2018/07/02/browser-maker-opera-has-filed-to-go-public/

Dobrzanski, M. (2013, Feburary 6). *MySQL Security: Overview of MySQL Security Features*. Retrieved from psce.com: https://www.psce.com/en/blog/2013/02/06/mysql-security-overview-of-mysql-security-features/

Duckett, J. (2014). *Javascript & Jquery - Interactive front-end web development.* Wiley.

Elliot, E. (2019, 5 11). *How Popular is JavaScript in 2019?* Retrieved from medium.com: https://medium.com/javascript-scene/how-popular-is-javascript-in-2019-823712f7c4b1

G. C. W. Sabin, G. K. (2017). The Impact of Automated Timetabling on Universities-A Case Study. *The Journal of the Operational Research Society*, 689-693.

Google. (2020, 02 07). *Google Calendar*. Retrieved from calendar.google.com: https://calendar.google.com/calendar/r

International Software Testing Qualifications Board. (2018). *ISTQB Glossary.* Retrieved from glossary.istqb.org: https://glossary.istqb.org/en/search/Unit%20testing

Khadija Sania Ahmad, N. A. (2017). Fuzzy_MoSCoW: A Fuzzy based MoSCoW Method. *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, 1.

Kolici, V., Xhafa, F., Pinedo, E. E., Segui, V., & Barolli, L. (2013). Analysis of Mobile and Web Applications in Small and Medium Size Enterprises. *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing* (pp. 324-330). IEEE.

Krush, A. (2018, January 13). *5 Success Stories That Will Make You Believe in Scaled Agile*. Retrieved from objectstyle.com: https://www.objectstyle.com/agile/scaled-agile-success-story-lessons

M. Duran ToksarJ, D. O. (2014). New Developments in Scheduling Applications. 1-2.

Michael W. Carter, G. L. (1996). Examination Timetabling: Algorithmic Strategies and Applications. *The Journal of the Operational Research Society*, 1-2.

Microsoft. (2017, 03 30). *Overview of SQL Server Security*. Retrieved from docs.microsoft.com: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql/overview-of-sql-server-security

Mosh Hamadani. (2018, January 23). *What is Node.js? | Mosh*. Retrieved from Youtube.com: https://www.youtube.com/watch?v=uVwtVBpw7RQ

MySQL. (2020). *MySQL Restrictions and Limitations.* Retrieved from dev.mysql.com: https://dev.mysql.com/doc/mysql-reslimits-excerpt/5.5/en/

Orn, A. (2017, July 13). *The Pros and Cons of the System Usability Scale (SUS)*. Retrieved from research-collective.com: https://research-collective.com/blog/sus/

Panchal, J. (2015, 12 29). *Pros and Cons of AJAX*. Retrieved from dzone.com: https://dzone.com/articles/pros-and-cons-of-ajax

Pauline M. Berry, C. A.-S. (2006). Conflict negotiation among personal calendar agents. *AAMAS '06 Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 1467-1468.

Pauline M. Berry, M. G.-S. (2011). PTIME: Personalized assistance for calendaring. *ACM Transactions on Intelligent Systems and Technology*, 40-60.

Petersen, K. (2009). The Waterfall Model in Large-Scale Development. *Product-Focused Software Process Improvement*, 11.

phpsecurity. (2017). *PHP Security: Default Vulnerabilities, Security Omissions and Framing Programmers?* Retrieved from phpsecurity.readthedocs.io: https://phpsecurity.readthedocs.io/en/latest/_articles/PHP-Security-Default-Vulnerabilities-Security-Omissions-And-Framing-Programmers.html

Rouse, M. (2019, August). *HTML (Hypertext Markup Language)*. Retrieved from theserverside.com: https://www.theserverside.com/definition/HTML-Hypertext-Markup-Language

Rudy Oude Vrielink, E. J. (2017). Towards improving tenders for Higher Education timetabling software: Uncovering the selection criteria of HEIs when choosing timetabling applications, using ERP as a reference. *Industrial Engineering & Business Information Systems*.

Strickland, J. (2008, November 6th). *How Google Calendar Works*. Retrieved from howstuffworks.com: https://computer.howstuffworks.com/internet/basics/google-calendar3.htm

Ten Ton. (2019, April 26). *Is HTML Easy To Learn? Skirt The Mumbo & Unravel The Truth!* Retrieved from tentononline.com/: https://www.tentononline.com/is-html-easy-to-learn/

tutorialspoint. (2020). *AJAX - Browser Support*. Retrieved from tutorialspoint.com: https://www.tutorialspoint.com/ajax/ajax_browser_support.htm

Vandim Tkachenko, A. L. (2008). Chapter 4. Query Performance Optimization. In A. L. Vandim Tkachenko, *High Performance MySQL, 2nd Edition.* O'Reilly Media, Inc.

W3Counter. (2020, 2). *Browser & Platform Market Share*. Retrieved from https://www.w3counter.com: https://www.w3counter.com/globalstats.php

W3Techs. (2020, 1 6). *Usage statistics of HTML5 for Websites*. Retrieved from w3techs.com: https://w3techs.com/technologies/details/ml-html5

Wassim Sellil Atoui, W. A. (2018). Offline and Online Scheduling Algorithms for Energy Harvesting RSUs in VANETs. *IEEE Transactions on Vehicular Technology*.

Whong, W. (2013). The Importance of a Software Development Methodology in IT Project Management: An Innovative Six Sigma Approach. *Department of Mechanical, Materials & Manufacturing Engineering*.

Yan Wang, E. v. (2005). Designing mobile solutions for mobile workers: lessons learned from a case study. *ICEC '05 Proceedings of the 7th international conference on Electronic commerce*, 582-589.