# Reservation System for a Restaurant

### KV6003: INDIVIDUAL COMPUTING PROJECT

### SCOTT MAINS

### STUDENT ID: W18003567

### COURSE: COMPUTER SCIENCE WITH WEB DEVELOPMENT

### PROJECT SUPERVISOR: JOHN ROOKSBY

### SECOND MARKER: SHELAGH KEOGH

Abstract

This project aims to create a web application for a pizza restaurant that functions as a table reservation system; including an admin dashboard to deal with customers and bookings. The reason to create this project is that there has been an increase in customers using online booking systems since the COVID pandemic, and it is a great way to collect customer data for Customer Relationship Management. The system was built using HTML, CSS, PHP, and JavaScript (React Framework). Different react packages implemented using NPM were also included to make the web application more dynamic and fluid.

Introduction

# Aims of Project

The aims of the project are:

* To investigate how a table reservation system is created and how the data can be used in Customer Relationship Management.
* To build a web application for a pizza restaurant that integrates a table reservation system and a CRM for the restaurant owner.

# Project Objectives

* Create a literature and technology review:
  + Look at existing booking/reservation systems and identify their good and bad features. Take ideas that could be implemented in my design.
  + Examine existing plugins used for calendars that can be integrated into the reservation system.
  + Find existing CRM systems used in restaurants and find ways I can use user data created from the reservation system.
  + Find academic articles on CRM systems and table reservation systems.
  + UCD design for restaurant website and booking systems.
  + Look at privacy and data protection issues regarding storing user data.
* Establish and prioritize the requirements of the product:
  + Find out the client’s requirements
  + Create a storyboard and initial vision of the application
  + Find target audience(s)
  + Persona(s) and Scenarios
  + User requirements document
* Create designs for the web application:
  + Initial design storyboard and wireframes.
  + Low & High-fidelity prototypes
  + Use case diagrams
  + Class diagrams
  + Entity-Relationship diagrams
* Create the product based on the design specification.
* Test the product:
  + Functional testing
  + Usability testing (Users and admin)
* Make changes to the application based on the testing process.
* Evaluate the product.
* Evaluate the processes and my performance

# 1.3 Project Overview

This project is a thorough investigation into how a table reservation system for a restaurant is created and how it can be utilised for a Customer Relationship Management system. A functioning reservation system as a web application has been created for a pizza restaurant to support this. The customer will be presented with a functioning dynamic web application created with React which will then redirect them to the reservation system which allows the user to book a table slot at the pizza restaurant. The customer will be able to book the table as either a guest or create a user account, which will allow them to collect loyalty points and receive special promotions. The admin of the website will have access to a dashboard that gives them an overview of all bookings made on the website and will give them the power to cancel and create bookings. There will be lots of other extra functionality that will act as a Customer Relationship Management (CRM) system, such as being able to send out marketing emails to all registered users and applying “comments” to customers which will allow them to personalise the experience for each guest. The application will be as secure as possible and will have authentication integrated for both the admin and the users using a JWT token.

# Project Context and Justification

Since the COVID-19 pandemic, it has become a necessity to reserve tables at restaurants to avoid queues and prevent congestion from customer walk-ins due to the new government health guidelines. It is also useful for restaurant owners to minimise queues from walk-ins so that the customers don’t have to wait long to be seated and in turn be disappointed. A table reservation system can help tackle these problems by allowing the user to reserve a table for a specific time slot. The table will then become available again for another user once the allotted time has occurred. With the data collected from the table bookings, the restaurant owner can also see when his restaurant is most popular on a seasonal, weekly, and daily basis. On top of this, the user will have the ability to create an account and opt-in for special offers from the restaurant. This data can be very important in a customer relationship context, as they can promote their brand on popular trends with special offers etc. The restaurant needs to try and build a customer relationship, as it is a sure-fire way to boost sales.

A client has reached out to me to design a system that allows the user to book a table at their restaurant based on such parameters as date, time, and table size. They have requested that the reservation system be integrated into a website that promotes their restaurant, with a booking page that takes the user directly to the table booking form. Being able to view the table layout visually would also be a bonus so that they can choose exactly where they want to sit inside the restaurant. The client has requested an interface for him as the admin, so they can view all the bookings that are submitted on a daily and monthly basis. The admin page can also display analytics for the data collected from the reservation system; so that they can see the extent of when most bookings occur and on what days. The user can also create an account when reserving a table at the restaurant. With the account, the user can build up loyalty points with every successful booking they have at the restaurant. This is great for both the restaurant owner and the user, as the user gets a discount, and the owner gets invaluable data and a returning customer.

So overall, the project idea was chosen because of the need of the client and the increase in demand for online alternatives due to the COVID pandemic.

# Summary of Tools and Techniques

The following technology was used in the project to build the web application:

* React – This was used to create the client-side of my project. Everything that the user sees was created using React, and of course standard HTML and CSS as well. Various plugins and packages were also integrated from Node Package Manager such as the calendar component.
* PHP – This was used for the back-end of the web application. PHP can interact with the database and update it with new bookings, update existing bookings or delete entries.
* MySQL – This was used to create a database for the system. The MySQL database stores the user data and booking data created from the reservation system.
* TailwindCSS – I decided to use TailwindCSS to create the design for my project rather than basic CSS. Tailwind is great as you can integrate ready-made components that look sleek and modern. This allowed me to focus more time on the actual functionality of the system without having to worry about the design looking subpar.
* SCRUM – This is an AGILE framework that was employed throughout my project. It split up my workload into different parts and I tackled each bit through “Sprints”. It also made me have close contact with the stakeholder throughout the design process and also when we evaluated it at the end for feedback.

Analysis Chapters

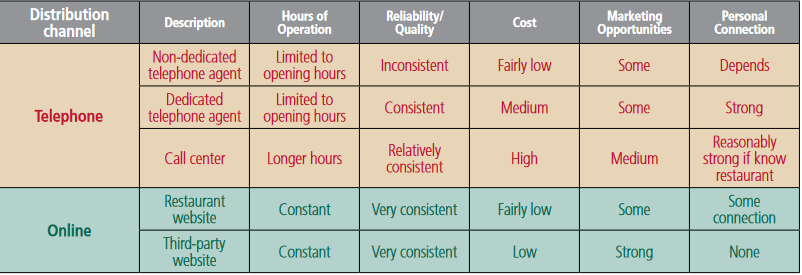
# 2. Literature Review

## 2.1 Introduction

This section will discuss literature that is relevant to booking and reservation systems, and how the user data can be used to create a Customer Relationship System to boost customer relations and increase repeat customers. This will help understand the challenges that surround restaurant reservation systems and the best way to tackle integrating one. I will also discuss how useful booking systems are overall, especially due to the increase in demand for them due to the COVID pandemic.

## 2.2 Reservation Systems

Reservation Systems have become the norm in modern times when scheduling table slots within restaurants. It has been noted that technologically savvy Generation X and Millennials expect easy and smooth access to restaurant services using their mobile devices [1]. In this report, it stated that the best practices for Netwaiter (a notorious restaurant marketing platform) to attract local customers is to offer online reservations and have a mobile-friendly website [1]. Alongside this, it also states that you should have a static web page for the menu within your web application. I will be integrating all these practices within my design.



*Figure 1 Comparison of Reservation Methods*

In a Hospitality report from Cornell University, a study was conducted to garner customers’ views on online reservations compared to phone call reservations [2]. There was evidence that both methods had their perks and weaknesses. A good reason why online reservation systems are beneficial is that hours of operation are constant. This means that the customer can reserve a table outside operating hours, unlike telephone reservations. Alongside this, it is also noted that it is much more reliable and keeps costs low as nobody needs to man the landline to answer calls. In terms of negative aspects, it is important to note that with online reservations there is the chance of reducing personal connection with the customer – especially using third party web applications. Because of this, both lines of communication must be open to the user. It also makes it more personalised if the customer can book directly from the restaurant’s website rather than a third party such as OpenTable, which is exactly how I will be implementing it.

## 2.3 The effects of COVID-19

The need for such a system has increased dramatically due to the COVID pandemic. This is so the restaurant can control store capacity to reduce the spread and enable no congested waiting times for customers. According to Quidini, 32% of customers believe that scheduling a time slot would make them feel safer against COVID-19 [1].

COVID has also exponentially increased the amount of online traffic toward e-businesses and online sales channels. Despite a restaurant not being able to completely remove the offline aspect, it can try and shift certain parts of the business online so that it can shift with this growing trend of online channels.

## 2.4 CRM Systems

CRM stands for customer relationship management, and it is essentially a system that helps business owners nurture their relationships with their clientele [3]. The emergence of things such as the Internet of Things has meant that businesses are now trying to identify business strategies to personalise their relationship with their customers.

There are 4 main components of the relationship between a business and its customers: customer acquisition, customer retention, relationship expansion and defection [8]. Within my system, I am primarily going to focus on how I can improve customer acquisition and retention.

## 2.5 UCD Approach

UCD means User-Centred Design, and the UCD approach essentially means that throughout the iterative design process we always have the user at the centre of the design. Throughout the development process, I am going to make sure I consider the user every step of the way - their needs, objectives, and feedback [10]. There are several key design principles when following UCD. One of these is involving users at the very beginning of the design process. I am going to do this using the questionnaires and the interview with the client. Including these at the start will make sure that the user’s requirements are thoroughly met, and that I do not go down the “wrong path” and create something I can’t reverse.

In a survey that asked questions about UCD methods and their effectiveness [11], it is shown that

## 2.6 RESTful API

Diagram

Description automatically generated

*Figure 1 RESTful API architecture*

Representational state transfer (REST) or RESTful web services is an architectural style for an application program interface. This type of interface deals specifically with HTTP requests to access and use data. The information can be delivered in several different formats: JSON (Javascript Object Notation), HTML, XLT, Python, PHP, or plain text. I will be primarily sending and receiving JSON in my web app as it is language agnostic and readable by both humans and computers [15].

The diagram shown in Figure 2 shows the architecture of a RESTful Web application.

#### 2.6.1 JSON Web Token

I am going to implement the JSON Web Token (JWT) authentication process for this application, which is a token-based authentication and new industry-standard practice for RESTful web applications such as the one I am creating. Within this section, I am going to discuss the architecture for token-based authentication and the best standard practices for storing JWTs.

## 2.7 Security Risks

There are plenty of security risks that could occur during the development of the system. The web application is going to handle mildly sensitive data such as phone numbers and emails – and of course passwords. I also need to be able to restrict access to the admin dashboard so that unauthorised users cannot access this. I am going to discuss how I plan to secure my web application and the variety of security risks that could occur and how I could combat them.

#### 2.6.2 XSS Attacks

An XSS attack, also known as Cross-Site Scripting, is a type of injection in which malicious scripts are injected into your website [12]. This type of attack occurs when an attacker sends code in the form of a browser side script to an unsuspecting user. The browser doesn’t have any way to know if the script is trustworthy. The flaws that allow these attacks to succeed are common and can be found whenever a web application accepts user input in its output without verifying or encoding it.

#### 2.6.3 SQL Injection

A SQL injection attack involves inserting or "injecting" a SQL query into the application via the client's input data [13]. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), perform database administration operations (such as shutting down the DBMS), recover the content of a given file on the DBMS file system, and, in some cases, issue commands to the operating system. SQL injection attacks are injection attacks in which SQL commands are inserted into data-plane input to influence the execution of predetermined SQL commands.

#### 2.6.4 URL Manipulation Attacks

The URL of a website (Uniform Resource Locator) is an address to a unique resource on the web. If a hacker manipulated certain parts of a URL, they could then deliver pages that they aren’t supposed to have access to [13]. On dynamic websites, parameters are usually passed via the URL. An example of this would be the end of the URL ending in “/api/authors?id=59657”. In this example, the URL is accessing an API and using the authors and its ID as parameters to access specific information from the API. If the creator of the website doesn’t secure these parameters, then they could run the risk of exposing web pages and information that they are not supposed to access.

# 3. Requirements Capture

## 3.1 Introduction

A requirement capture plan is a useful exercise to undertake early in a project life cycle to establish the scope of the project. The main reason is to understand the system from a user’s perspective and find their common needs and expectations. Within this section, I am going to discuss the research I gathered during this exercise from the various artefacts I created, and how useful they will be going forward in the overall design of the booking system.

## 3.2 Capture Process

The requirement process is the foundation of every successful software project [4]. It’s been reported by the IDC that 25% of IT projects experience outright failure, and 50% require reworking [5]. This is in part due to the lack of requirements captured in the initial planning stage.

The way I am going to capture my requirements for my system is by creating several different artefacts. Through these artefacts, I will develop the needs of the user in both a customer context and a staff/admin context. I am initially going to conduct a competitor review to gather important functional requirements that already exist in similar systems. Once I have gathered important features from the competitor analysis, I will then conduct a client interview with the stakeholder. Through this interview, I will propose features that I have gathered from the competitor analysis and gather information on what they believe are important requirements for the system. Both artefacts should be enough to gauge the functional requirements from a staff/admin perspective.

I will then send out a questionnaire to prospective users of the system. This questionnaire will gather intelligence on what users look for when trying to reserve a table using an online web-based system.

## 3.3 Competitor Review

The competitor review will consist of the currently existing booking and reservation systems on the market. I have looked specifically at the features that these systems incorporate and how they have utilised them to make it a seamless experience for the user [Appendix A]. I also did a separate analysis looking at the different systems from an admin perspective [Appendix B]. I am going to summarise key points discovered in the competitor analysis below and my most important findings.

### 3.3.1 OpenTable

OpenTable is a restaurant reservation service that was founded in California but now expands internationally [2]. OpenTable is a hub for restaurants in the user’s local area and acts as a discovery platform for restaurants as well as a booking system. The user simply chooses the restaurant they want to book, and they are taken to the restaurant’s page which allows them to see available times and dates. They can then reserve a table according to their needs.

From a client perspective, there are lots of different features available to make it a seamless experience for the user to reserve a table at their chosen restaurant. The first thing worth mentioning is how intuitive the whole process is for finding available tables and reserving a slot. The first thing the user needs to choose is their party size, followed by the date on which they want to book the table. This then shows the time availability according to both parameters. If there are no times available for that specific slot, then the system proposes times within 2.5 hours of the initial time slot. I think this is a great process compared to other systems as it allows the user to immediately know whether there is any availability for their circumstances.

Once the user has chosen a slot, they are taken to a page on which they can sign in with an account or proceed with the booking as a guest. Creating an account allows the user to collect loyalty points that can be exchanged for discounts in future bookings. This creates value for the user so that they will be more interested in creating an account. The overall layout is easily readable and simple. It consists of a very basic colour scheme of red, white, and black so all the text stands out. Any important information can then be put in a different colour to make it stand out from the rest of the page. An example of this is how they show the safety precaution methods in a darker grey colour.

There are many different features available from an admin perspective. OpenTable manages customer relationship management very well and offers many

### 3.3.2 TheFork

TheFork is another restaurant booking and discovery platform now joined with the huge review platform Trip Advisor.

### 3.3.3 SimplyBook.Me

This system is most similar to what I am trying to accomplish with my design. The website is designed first, and the reservation system is built directly into the website. The other competitors usually have been added to a hub, like that of uber eats, rather than have a website specifically designed for the brand and the booking system directly integrated.

### 3.3.4 BookingNinja

### 2.3.5 Most important findings

## 3.2 Requirements for Customers

To establish the requirements for potential customers, I created a Requirement Capture Plan that contains: Personas and Scenarios [Appendix B], a questionnaire for target users to establish further requirements [Appendix C], competitor analysis [Appendix A] and User Stories [Appendix E].

## 3.3 Requirements for Staff

For the requirements for staff, I mainly used the client interview [Appendix D], competitor analysis [Appendix B] and User Stories [Appendix E] to gauge what features need to be added to the system to make their lives easier.

# 4. Tools and Techniques

## 4.1 Introduction

The following section is going to go through all the different languages and technologies I used throughout the project development. I am then going to discuss the project framework I utilised to make everything go smoothly throughout the lifecycle of the project.

## 4.2 Adobe XD

Adobe XD is design software used for web and mobile applications. It is a great tool to create prototypes and mapping out the design and layout of my application. I will be using Adobe XD to create the low fidelity wireframe of my reservation system

## 4.2 Database

I have opted to use MySQL to store the data for the reservation system. MySQL is a relational database management system and is the most widely used database technology used across many huge companies.

## 4.2 Programming Languages

As my project is full-stack, I will be splitting it up into server and client-side tasks that will require various languages. For the server-side, I will be using the PHP language in conjunction with React for the client-side.

## 4.2.1 PHP

PHP is an open-sourced scripting language that is primarily used in web development. I have decided to opt for PHP as it has great versatility.

As my application is going to be an API - in specific a RESTful API - I wanted to measure up all the appropriate languages that could be used.

## 4.2.2 ReactJS

React is a JavaScript library developed and maintained by Meta (formerly Facebook).

## 4.3 Project Framework

The project framework I will be implementing throughout the project life cycle is AGILE. Despite Agile being used primarily within a team context (mainly SCRUM), many aspects can be applied to the solo developer to create a successful project. I am going to discuss some of the things I have implemented from the SCRUM framework below.

The agile methodology is an iterative process that lends itself to rapid application development [5] –fitting perfectly into my project. Everything I do and create throughout the life cycle will be test-driven and done in small sprints. A small sprint is essentially a time-boxed period to accomplish a chosen user story that was set in the research stage. I am going to create a Trello board and turn my user stories into a product backlog; this is a list of all the items I intend to complete at some stage for the product.

The main method I’m going to borrow from AGILE is keeping close contact with the client. It is important that the client is closely engaged in the development and can change the requirements or accept any suggestions proposed.

Design, Implementation, and Testing

# 5. Design Specification

The design specification is then created from the user requirements established in the analysis section. I first created use case descriptions for each main requirement for the system, followed by diagrams to help organise my thoughts and pinpoint every feature I want to include in my wireframe. I then created a low-fidelity wireframe to map out everything, with emphasis on looking at the interface and making it as accessible as possible for the user. I decided against using a High-Fidelity wireframe as I decided to opt to use TailwindCSS which will handle a lot of the design aspects.

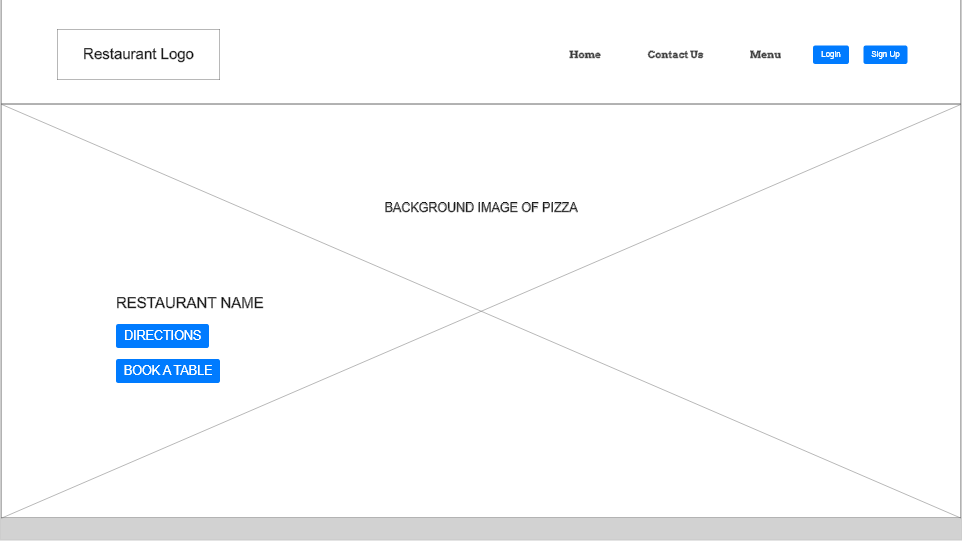
## 5.1 Use case Descriptions

Use case descriptions are a logical flow of how a user would perform tasks on the website. In the requirement capture plan, I generated user stories from both the customer and admin perspectives. In this section, I am going to break down these user story activities into a use case and how the user would achieve it from their perspective. All of the use case descriptions are present in [Appendix \*], but I will break down important things I learned from these descriptions.

## 5.2 Use case Diagrams

## 5.3 Wireframe

The initial wireframe design was basic, and its entire purpose was to map out the features that I proposed in the use case descriptions. The web application is more than just the reservation system, but I am going to focus strictly on the wireframe design of the booking system and the admin dashboard rather than the other static pages present on the website (Menu, contact, home page etc.). The designs shown in this section are the desktop version, but the responsive mobile design is present in [Appendix \*]



*Figure \* Homepage as unauthenticated*

Figure \* is a snippet of the home page and what the user will first see when they open the web application. I added the “Book a table” button and the options to “Log in” and “Sign up” in the Navigation bar. The login and sign-up options are only present if the user isn’t authenticated through the login process.

Graphical user interface

Description automatically generated

*Figure 2 Modal when Sign Up button is pressed*

Figure \* shows the modal window that appears once the user presses the “Sign Up” button in the navigation bar. A modal is a web page element that displays in front of and deactivates all other page content [14]. I opted to use a modal window for the design so that the sign-up form can be accessed anywhere from the website without having to switch pages.

Graphical user interface, application

Description automatically generated

*Figure 3 Modal when Login is pressed*

When the “Login” button is pressed, a similar modal appears for the user to log in to the web application using the email and password they created in the sign-up process. I also added in the option to remember their login details for convenience and a link to reset their password in the circumstances of them forgetting it.

A picture containing diagram

Description automatically generated

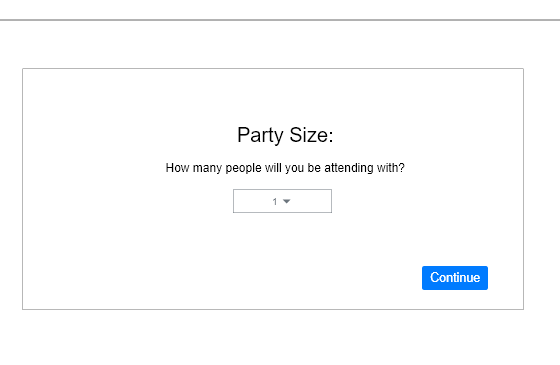
*Figure 4 Navigation bar when authenticated*

When the user logs in they enter an “authenticated” state which changes the Navigation bar. The navigation bar then removes the login and sign-up button and shows a user icon, which can be pressed and displays a dropdown menu with links to the user dashboard and sign out.

Graphical user interface, text, application, email

Description automatically generated

When the user clicks the “Book a table” on the home page, they will be redirected to the booking form page. This will display differently depending on whether they are authenticated or not. If the user isn’t authenticated, they can continue to the guest booking form.



Once the user has either logged in or decided to continue as a guest, they will be navigated to the booking form. The first stage is choosing the party size. When they have chosen the number of people from the select dropdown, the value will be stored as a variable and taken to the next stage of the booking process.

A picture containing table

Description automatically generated

The next stage is for the user to choose the date they wish to attend. The calendar will be a react component imported directly from Node Package Manager. It will allow the user to select a date and store it as a variable to be used in the final confirmation stage. When the user clicks a date, a select dropdown menu will generate all the booking slot times available on that date. A back button is also present if they wish to go back to the previous stage and choose a different party size. If they are happy with their choices, they can click continue and proceed to the confirmation stage.

Graphical user interface, application

Description automatically generated

The final stage of the booking system is for the user to confirm their details and submit them. If the user used the guest booking form, they will be presented with text fields to type in their credentials. If they logged in, then these will be prepopulated with their account details.

Graphical user interface, text, application, email

Description automatically generated

If the booking goes through successfully, a success message will then be displayed confirming their reservation at the restaurant. They will also be e-mailed a confirmation email which will allow them to cancel the booking. The details posted from the booking form will be sent to the MySQL database via PHP and will be assigned its bookingID and a guest entry will be created as a foreign key to the bookingID. If the user was logged in, then the userID will be used rather than creating a guest entry.

## 5.4 Entity Relationship

My initial entity-relationship diagram shown in figure \*

## 5.5 Flowcharts

# 6. Project Structure

The project structure was split up into two section folders. The first folder held all the client-side coding (React) which was labelled “front-end”, and the other folder was all the server-side coding (PHP) labelled “back-end”.

The front end was split into 4 main folders. The first folder is the “asset” folder, which contains all the images and fonts used within the project. The next folder is the “component” folder. This folder contains the bulk of the logic and coding for the booking form. I like to keep all the reusable functions in the components folder as it logically makes the most sense and keeps the structure clean. The next folder is the “containers'' folder, which is essentially most of the HTML and client-side components that the user visibly sees on the web pages. The containers can be reused across the website. Finally, we have the pages folder. The pages folder contains each page on the website, and it initialises all the components and containers created in the other folders. Overall, I feel like my client structure is very intuitive and clean.A picture containing calendar

Description automatically generated

I maintained a similar structure for my backend files. I took an object-oriented approach for my server code and used classes throughout. In the config folder, I created an autoloader magic function that uses an “include\_once” function for all class names in my directory. I then created an exception handler and error handler. The exception handler would output errors depending on whether the request was HTML or JSON. I then create the actual config file which initialises the other functions and defines important information such as basepath and database. Text

Description automatically generated with low confidence

# 7. Testing and Results

## 7.1 Functional Testing

I decided to do my functional testing using the SCRUM methodology. This meant that my software was consistently tested throughout the development cycle, paying specific attention to the complexity, quality, performance, and usability. I did this by testing the code by myself and through Unit Testing using a unit testing framework called PHPUnit.

### 7.1.2 Manual Testing

I conducted functional testing myself across the full project cycle. I would test my code for issues and log them using a test case. You can find the full log of manual testing in [Appendix \*]. Table 1 below shows the format of how I would test my code.

### 7.1.3 Unit Testing

## 7.2 Usability Testing

### Customer Testing

\*INSERT USER TESTING PLAN\*

## Admin Testing

\*INSERT ADMIN TESTING PLAN \*

## 7.1.2 Vulnerability Testing

XSS Attack

SQL Injection Attack

URL Attack

## 7.1.3 Cross-Browser Testing

Firefox

Chrome

Edge

Evaluation and Conclusion

# 8. Evaluation

## 8.1 System Evaluation

Throughout this section, I will evaluate the system I have created through testing and feedback from users/the client. I will also discuss the processes I used within the project development and how they could be improved if I had to redo the whole thing again.

### 8.1.2 Build Quality

The build quality of my system is going to be based on the required criteria that I set out in the initial phase of the project and how well it meets it. I am also going to discuss points in my code which I consider especially high or low quality.

### 8.1.3 Testing

The manual testing process was very insightful and allowed me to identify a few issues which I then rectified. I’ve never conducted testing in previous projects that I’ve worked on, so it was overall a great way to get further experience in making my code more robust. In hindsight, I would have spent more time learning ways to test my code and making it even more solid and secure. This was an issue due to my lack of experience, so will be improved upon next time I set myself on a big task.

### 8.1.4 Requirements

### 8.1.5 Client & User Feedback

### 8.1.6 Literature Evaluation

The literature that I conducted at the start of the project was very useful and I am going to discuss below how the different topics of literature played a role in the final prototype.

### 8.1.7 Tools and Techniques

In this section I am going to discuss all the tools and techniques I proposed at the start of the project and evaluate how I used them and if they were ideal. I will also mention any other tools I used that I may not have thought about using at the start of development.

#### 8.1.7.1 MYSQL Database

MySQL was a good database technology to use within my project. It had functionality that fit well into my project which allowed me to achieve my goals. One of these was how you could apply “events” to the data that is stored within the database. I applied a unique key to the ‘userid’ of each booking that is created, which meant that a user cannot create more than one booking. Using the events functionality of MySQL, I could set an event that is triggered once the booking date is over which deletes the entry in the database. This is a great way to make sure that a user can’t spam booking slots into the database so that they can gain more “loyalty points” for rewards.

If I started the application from scratch, I believe MongoDB would have been a better database to use as it represents the data in JSON objects and is much easier to build applications with its schema-less design.

#### 8.1.7.2 PHP

PHP was used to create the server architecture for the booking system. PHP is a great language to use as I find its syntax very straightforward, and its command functions are simple and easy to implement. Throughout the project cycle, I ensured that I created class components for all the elements of code that I created. This meant that all the code I used was reusable and could be reimplemented across the project without having to rewrite the code.

If I redid the booking system, I would switch the backend architecture from PHP to Node.js. I believe the MERN stack would have allowed me to create the system at a much faster pace as it is more intuitive using Node with React as they are both built upon JavaScript. I also really like how node.js has its own package manager, which essentially means you can install pre-existing packages created by other people straight from the command line. This would have made sections of my backend much easier to implement, and I would have been able to add extra functionality with ease.

#### 8.1.7.3 React

React was a great software to use for the client-side of the project. As previously mentioned, you can install pre-existing packages and components directly into react which makes life much easier and allows you to integrate pre-tested solutions created by other people. An example of this is how I used a “React Calendar” component, which added calendar functionality to my application with ease. It provided me with functions that allowed me to easily grab the date that the user chooses and place it into a variable for submission into the database.

The main issue with react was the decision to initially use Class-based components. Halfway through the development cycle, I realised that using functional components with hooks is much more fluid and modern. It is concluded that developers have entirely replaced class components with functions and hooks [6]. I decided to convert all my previous components which were class-based to functional components instead. This ended up costing me a lot of time and ate into the time I set aside for other functionalities of the application.

#### 8.1.7.4 Visual Studio Code

Visual Studio Code is a code editor that allows the user to be very efficient when writing code. It has a vast extension library that allows the user to add quality of life features to their arsenal when coding. An example of this is the ES7+ Snippets extension. This allowed me to type shortcuts such as “rafce”, which in turn would generate a full template of a functional component.

Visual Studio also has great integration with GitHub.

#### 8.1.7.5 TailwindCSS

I decided to completely replace conventional CSS with a framework called TailwindCSS. Tailwind is a utility-first CSS framework that emphasises responsive design and was used to apply a modern, sleek look to the application.

#### 8.1.7.6 Github

#### 8.1.7.6 Heroku

I used Heroku to deploy the server-side of my application. Heroku is a cloud-based web service that can deploy and operate web applications directly in the cloud. This was a great solution for testing my code in a development environment, as it integrates extremely well with GitHub.

#### 8.1.7.7 Netlify

#### 8.1.7.8 NuMySpace

## 8.2 Process Evaluation

### 8.2.1 Time Plan

My time was split up at the start of the project using a GANTT chart. A GANTT chart is a type of bar chart which helps illustrate a project schedule and map out tasks that need to be completed at certain intervals.

### 8.2.2 Skills

I developed lots of technical and soft skills during the project development.

### .8.2.3 Project Life Cycle

### 8.2.4 Learning Process

# 9. Conclusion

## 9.1 Deliverables

## 9.2 Objectives

## 9.3 Changes and Further Work

I believe I met the criteria for my project to a good standard, but there are several things that I would have changed and improved upon in the final product, and I would have changed a few things within the development cycle to make it a much quicker and smoother process.

I would have liked to spend a lot more time developing the CRM side of the system. I spent most of the project lifecycle developing the functionality of the booking system and creating a dashboard for the admin to change existing bookings. CRM systems have developed a lot in the past years and there is a lot more I could have done with the user information to create a more robust customer relationship to entice users to return to the restaurant.

REFERENCES

[1] Flynn and Buchan, ‘A REFEREED PUBLICATION OF THE AMERICAN SOCIETY OF BUSINESS AND BEHAVIORAL SCIENCES’.

[2] Kimes, ‘How Restaurant Customers View Online Reservations’.

<https://dspace.mit.edu/bitstream/handle/1721.1/86490/46888364-MIT.pdf?sequence=2>

[3] ‘What Is CRM? The Beginner’s Guide - Keap’. Accessed 1 February 2022. <https://keap.com/product/what-is-crm>.

[2] OpenTable. ‘About Us’. Accessed 7 February 2022. <http://www.opentable.com/about/>.

[4] Hussain, Azham, and Emmanuel O. C. Mkpojiogu. ‘Requirements: Towards an Understanding on Why Software Projects Fail’. *AIP Conference Proceedings* 1761, no. 1 (12 August 2016): 020046. <https://doi.org/10.1063/1.4960886>.

[5] shelleydoll. ‘Agile Programming Works for the Solo Developer’. TechRepublic, 5 August 2002. <https://www.techrepublic.com/article/agile-programming-works-for-the-solo-developer/>.

[6] Luojus, ‘Usability and Adaptation of React Hooks’.

[7] Haekal and Eliyani, ‘Token-Based Authentication Using JSON Web Token on SIKASIR RESTful Web Service’.

[8] Hyun and Perdue, ‘Understanding the Dimensions of Customer Relationships in the Hotel and Restaurant Industries’.

[9] Daud and Aziz, ‘RESTAURANT RESERVATION SYSTEM USING ELECTRONIC CUSTOMER RELATIONSHIP MANAGEMENT’.

[10] Reveall.co. 2022. *A Guide to User-Centered Design | Reveall*. [online] Available at: <<https://www.reveall.co/guides/user-centered-design>>

[11] Vredenburg, K., Mao, J.-Y., Smith, P.W. and Carey, T. (2002). A survey of user-centered design practice. *Proceedings of the SIGCHI conference on Human factors in computing systems Changing our world, changing ourselves - CHI ’02*.

‌[12] ‘Cross Site Scripting (XSS) Software Attack | OWASP Foundation’.

[13] Keracheva, ‘URL Manipulation Attacks’.

[14] Juviler, ‘What Is a Modal and When Should I Use One?’

[15] www.redhat.com. (n.d.). *What is a REST API?* [online] Available at: https://www.redhat.com/en/topics/api/what-is-a-rest-api#rest.

‌