National College of Ireland

BSc in Computing

2019/20

Gary Lynch

X17140382

X17140382@student.ncirl.ie

Wait App 

Technical Report

National College of Ireland

Table of Contents

1 Executive Summary

The food service industry is one of the largest industries in the world. It spans across every country and many different sectors such as hospitality and retail either rely on or have their own food service industry. From bar service to restaurants it can be seen everywhere and most people go out on a regular basis to enjoy the service. With this in mind the industry has been slow to update to the modern age even with technical advances in the recent years. We have begun to see the sector finally evolve in places such as America but the rollout is very slow and even more-so than this, it is very expensive. Modern applications also require modern systems to run with them which is a risky business expense to the owners that they may not be willing to make.

My idea is to create a system that will allow waiting staff in restaurants to create and fulfil orders using just a tablet or smartphone. the orders will be sent directly to the kitchen. The goal is to not have this application replace the waiting staff and their jobs but to make their jobs easier. As it stands they have to take multiple journeys to and from the kitchen to send orders in and come back to take more orders, if I can cut out this journey it will be beneficial to everyone as it will allow the overall workflow to be more efficient. With the system being more efficient than just using the paper docket system currently in place it will make the day to day operation of restaurants more manageable and will make rushes in the industry less stressful as they can go from table to table creating orders rather than having to go in and out of the kitchen repeatedly with each individual order.

The application is made using Visual Studio Code and will have a minimalistic design so as not to distract from the features of the application. The goal is to be as simple as possible on the surface level so as not to dissuade any of the people from the idea of the application. The interface is going to hide all of the complicated parts of the application from the user and give them a simple experience when using it. The database will have a login and signup to be more secure and require users to sign in before they can use the application. The application will use a MySQLi database to store all of the information used for all of the transactions.

2 Introduction

2.1 Background

The initial idea of the project came from having my mother in the food service industry as a waitress. She would often come home and complain about how hectic her job was and how difficult it was and eventually I got into the industry myself and completely understood the problem. I realised there was a lot of times where I needed accepting multiple orders at once but already had an order at hand and it became difficult to manage as I had to run in and out of the kitchen to deliver the orders so that the orders could be prepared for the customers. This is where the initial idea came from and a lot of applications try to remove the need for wait staff taking orders which I find is a bad way to approach it. At the point you remove the wait staff it's almost as if you are creating a job for the customer and at the same time you remove the interaction between the staff and the customer which can be what makes the job worth doing at times.

The main gap in the market I am hoping to tackle is using the current systems that the restaurants have in place. From my research other applications require setup fees and additional systems to work where I want my application to be simple to use and work from any smartphone or tablet device. With my Wait App it will work to make the lives of waiting staff easier which in turn will make the experience of the customer better overall. This will also reduce the strain on the kitchen as they will be able to manage the orders on their own and have them automatically sent to them when an order is created.

2.2 Project Scope

The scope of the project is to develop an application that can work on mobile devices like a tablet or phone. The goal for my project is to create a web application that is simple to understand and use for the end user. I plan to have the application tested thoroughly as to avoid unwanted errors that could otherwise ruin the users experience. The application will automatically generate menus based on an SQL database that I will fill with a sample database for a template. It will allow the user to select from a multitude of items and customize them before adding them to their order. When an order is complete the user will be able accept payment for the order through the application. It will also keep track of all sales that are made throughout the use of the application and automatically update it with all confirmed orders. It will be secured behind a login and signup that will restrict access to the rest of the application unless the user is logged in. The goal is also to be accessible to users of all ages as this is the

demographic that would be using the application as people of all ages work in the food service industry.

I originally planned to do the application on Android Studio using a firebase database and a facebook API for the login and an XML API created online for the menu. This proved to be difficult though as the application became clunky and difficult to run on lower end devices, which I was using myself, which made testing difficult. The plan then evolved to use phpMyAdmin for the database as I had swapped to creating a web application as it is more lightweight. This gave me more freedom when it came to the database too as I was not restricted by the firebase tools required by the android application. The SQL database will be the key for the application and will hold most of the complexity as it will establish relationships between tables. The application will have to work on multiple formats too as screens appear in various sizes.

The overall scope is to make an application that constantly updates the database in various ways without hindering the users experience, if this is done in a way where the user will not have to deal with any of the complexity of the database then it will be a much better experience.

2.3 Technical Approach

This project is developed using PhP, SQL and a MySQLi database. At the beginning of development, I had set out a clear plan to follow throughout the development of my application and even with starting the project over twice it made the overall workflow much smoother than if I didn't have a plan to follow. With the pandemic occurring this year it hindered my work somewhat and I had to prioritize some features over others in order to get a functional application that still had the core features I set out with.

At the start of the development cycle I created a list of features that I wanted in the application. This made it easier to create a priority list of features that I could then follow and stick to a schedule and move onto the next feature when one was done enough rather then get tied up too much with a single feature. This also made testing easier as I could develop parts of the application separately and then combine them once I knew they worked.

I put a lot of work into testing the errors of the application and making sure to avoid unwanted bugs so that the users experience could be smooth. The key to the applications success is consistency and that's why I tested extensively each feature to try and prevent unwanted bugs and errors and to test different ways of using and abusing the application to find any ways the user could possibly break the experience.

The goal with doing a lot of preparation was to make future work easier overall. This made it easier to have a plan before I started adding in huge chunks of code that could ultimately not work well with the rest of my project. Visual Studio proved to be very helpful in this regard as it allowed the creation of branches and then I could merge the branches once I had tested the new feature extensively and if I had problems I could rollback to the previous version of the application I had without problems. Even with these guards set up I still had many problems throughout development but these made it easier to overcome. The overall structure of my approach in order can be viewed as the following.

I set up a requirement plan that I could follow, allocating time to each feature in order of priority which I could then allocate more time to important features if needed and spend less time on less important features also.

Before focusing entirely on the front end of the application I focused entirely on the backend as this is where the complexity of the application comes from. Creating working SQL statements and databases was the key to the entire application. This is what preps the application for commercial use in the future because getting the database working correctly means that anybody could swap in menus into the application, meaning that it can be used in almost any restaurant in the future. It also will allow them to track their sales on a day to day basis and gives the application the possibility of adding future features.

Finally, I approached making the design as simplistic as possible, only creating buttons that would handle all of the work to make the application not only easy, but very fast to use. Having the user write in very little, only for the sign up and login, means that we can minimize the errors occurred in the application.

2.4 Technologies

Visual Studio Code – Visual Studio Code is an IDE that can be used for almost all programming languages. It supports a massive number of plugins and overall it makes coding in multiple languages easier. The application also supports github and allows for branching which can make testing for developers relatively easy in comparison to other IDEs.

MySQLi – MySQLi is the improved version of MySQL. It can support way more statements and allowed me to create tables on startup of the application if they did not already exist which was a huge plus as I wanted the application to be usable right from the start for the user, all they had to do was plug in their own menu into the database and add the pictures they wanted for the menu. It also allowed me to automatically create my own sample menu on startup.

phpMyAdmin – phpMyAdmin was the tool used to view the updates and changes to the database throughout, SQL Workbench could also be used for the same thing but due to the fact I was using PhP this seemed like the better option for me. It allowed me to add stuff directly into the database if I wanted to try out features before fulling implementing them. It also allowed me to easily delete entries if there were bugs in the code that could cause problems when testing other features.

2.4.1 – Backend

The backend of the application was created using PhP and JavaScript for the main features.

2.4.2 – Frontend

The frontend of the application was created using Bootstrap as a framework to create a template which I then edited using HTML, CSS and JavaScript.

2.4.3 – Database

The database was created using SQL statements and PhP to run the statements.

2.5 Technical Resources

2.5.1 Tools

- Visual Studio Vode
- Notepad++
- MySQLi Database
- PhPMyAdmin
- XAMPP

2.5.2 Hardware

- Computer to run the server, code and application

2.5.2 Languages

- PhP
- SQL
- HTML
- CSS
- JavaScript
- SCSS

3 System

3.1 Requirements

The user is waiting staff or kitchen staff that want to create orders or view the current orders that have been sent to the kitchen to be prepared.

The user wants a platform that can allow them to create custom orders for the customers that they receive on a daily basis. The owners of the restaurants will want the application to be expandable and allow them to create their own menus or update current menus with new items. They also will want to be able to track their sales by having all sales made through the application stored in the database.

The users want to have a secure sign up and login that will prohibit the features from anybody who does not already have an account. They want their details to be secure in the database and to have their password hidden from all users so that in the extremely unlikely event that the data is breached their information is still secure in the database.

The user wants the application to be simplistic. They user does not want to be overwhelmed by the features of the application have a linear walkthrough of the application that is easy to understand where they are at all times throughout the use of their application. They want the key information to be easily understandable and not obscure. The application will have a fun design and be coloured accordingly to make it aesthetically pleasing.

3.1.1 Functional Requirements

Here is a list of the requirements that the application will offer to the user.

1. Users are required to sign up to the application to use any of the features. The application will save the users id when they sign in so that if they return to the application they will be saved to the session and be allowed to navigate all pages without the need to sign in again.
2. Users will be required to log into the application when they log out as the session will be terminated or if the session has been terminated for a long time eventually it will require another login for extra security.
3. The user will be able to log out of the system at any time.
4. The user will be able to select a table that they wish to serve.
5. The user will be able to select a main from the menu.

6. The user will be able to select sides and drinks on the menu.
7. The user will be able to customise the main order when they have selected it.
8. The user will be able to review the order before adding it to the cart.
9. The user will be able to remove items from their cart.
10. The user can pay for the items in the cart through paypal to confirm an order.
11. The user can view all current orders from all tables from a separate page.

3.1.1.1 Requirement 1 <Sign Up>

3.1.1.1.1 Description and Priority

It is essential for users to sign up before they can use the application. The sign up creates an account for the user with their first and last name, username, email and a password. The password is hashed when created and saved into the database.

3.1.1.1.2 Use Case

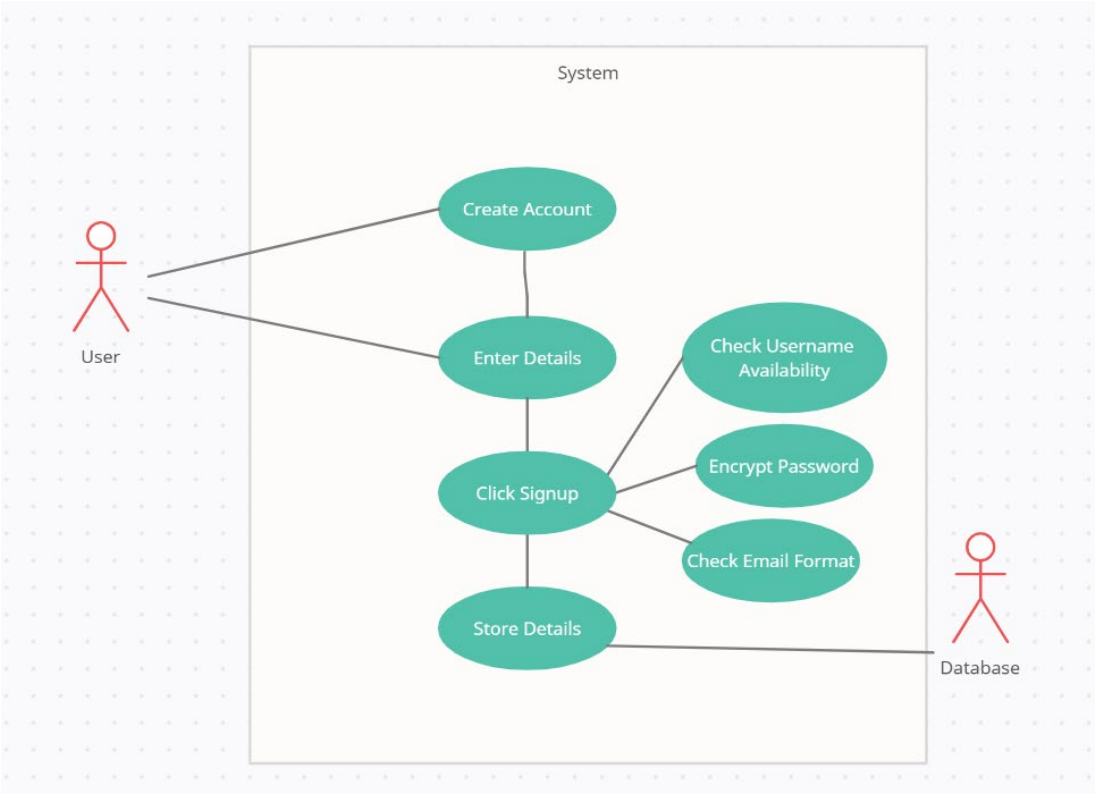
Scope

The scope of this use case is to show the interaction between the user and the database when they sign in.

Description

This use case describes the registration process in detail. It shows each of the steps required to create an account and add it to the database.

Use Case Diagram



Flow Description

Precondition

The system is and the user is on the sign-up screen

Activation

This use case starts when the user chooses the sign-up page.

Main flow

1. The user clicks the sign-up button to navigate to the signup page.
2. The user is brought to the sign-up screen.
3. The user enters in the details to register their account, in this case it is their first name, last name, username, email and password.
4. The user clicks the sign-up button.
5. The system checks all of the inputted data to make sure it matches the requirements set, email format is correct, username not taken.
6. If the information passes this it will create the user.
7. The password is encrypted and the information is added to the databases.
8. The user is then brought to the login screen.

Alternate flow

A1: <Not an email>

1. The system checks to see that the format entered for the email is correct
2. If it is not correct then the user will be prompted to enter in a valid email address

A2: <User already has account>

1. The system checks that the username entered is not already being used.
2. If the username is taken the system notifies the user that the username requested is already taken by another user.

Termination

This use case terminates when the user has successfully created an account and the information is stored in the database.

Post condition

Success:

1. The user has an account created and stored in the database.
2. The user is then redirected to the login screen.

Failure:

1. The user has entered in an incorrect email format and is prompted to enter in a valid email.
2. The username is already taken and they are informed that their username is taken.

3.1.1.2 Requirement 2 <Login>

3.1.1.2.1 Description and Priority

This use case is essential priority as the user needs to be logged into the system to use the features.

3.1.1.2.2 Use Case

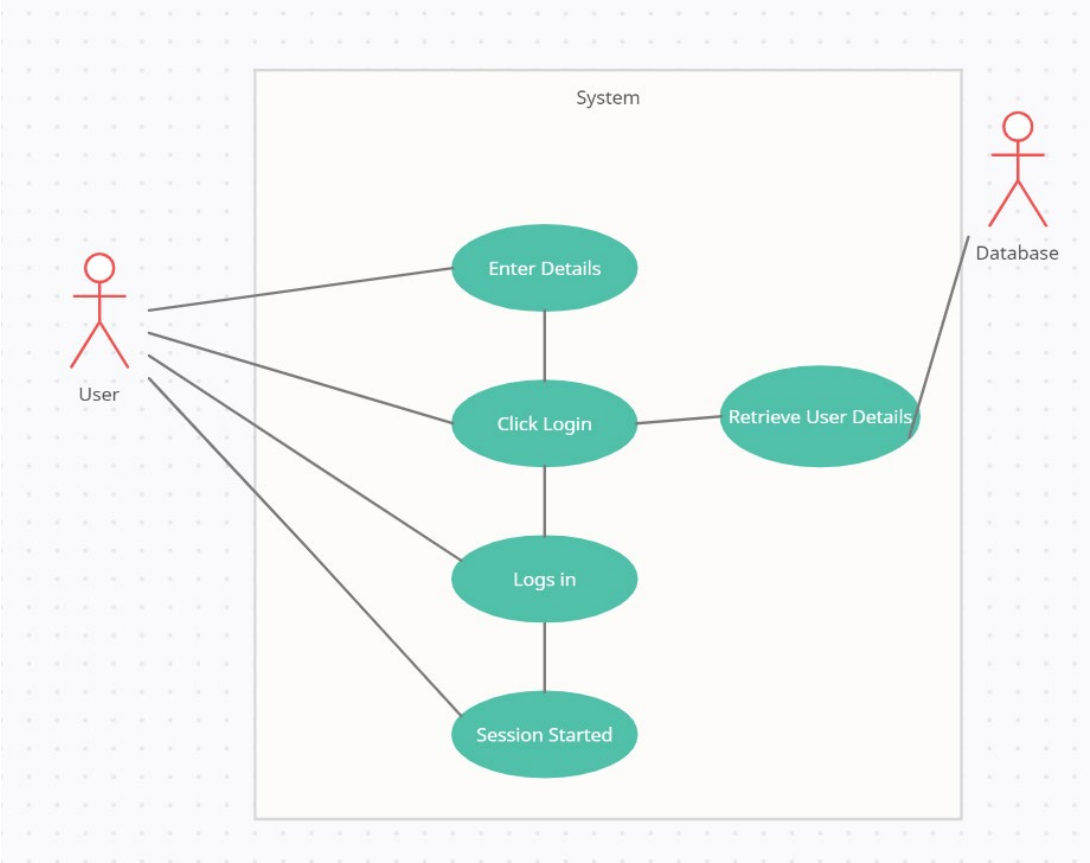
Scope

The scope of this use case is to show the logging in process.

Description

This use case shows the process of the user logging into the system and the information being retrieved from the database.

Use Case Diagram



Flow Description

Precondition

The user has previously registered an account through the sign up and has navigated to the login screen.

Activation

This use case starts when the user enters in their login details

Main flow

1. The user enters in their username or email into the required field.
2. The user enters in their password that they have registered with the email or username.
3. The user clicks the login button.
4. The database checks the entered in details against the databases entries.
5. If the information is correct the user is logged in and navigated to the home page.

Alternate flow

A1: <Username is incorrect>

1. The user enters in a username that does not exist in the database, either through an incorrect spelling or they have not previously registered the account
2. The user is told that the account does not exist
3. The user is prompted to reenter valid account details

A2: <email does not match>

1. User enters in email that does not correspond with an email in the database
2. User is prompted that email does not exist
3. Re-enter

A3: <Password is incorrect>

1. The user enters in a password that does not correspond with the email or username that has been entered in.
2. The user is told that the password is incorrect
3. The user is prompted to reenter in their details

Termination

If the user enters in a valid email/username and password the user is logged in, the session starts and they are navigated to the home screen.

Post condition

Success:

1. The user enters a valid email or username into the required field.
2. The user enters in the correct password that matches the username or email.
3. The user is logged in, the session starts and they are navigated to the home screen.

Failure:

1. The user entered in a detail incorrect or they did not have an account
2. The user is redirected to the login screen and asked to enter in correct details.

3.1.1.3 Requirement 3 <Log Out>

3.1.1.3.1 Description & Priority

This use case describes the user selecting the log out option which terminates the session and denies them access to the rest of the application until they log in again. It is medium priority but not mandatory.

3.1.1.3.2 Use Case

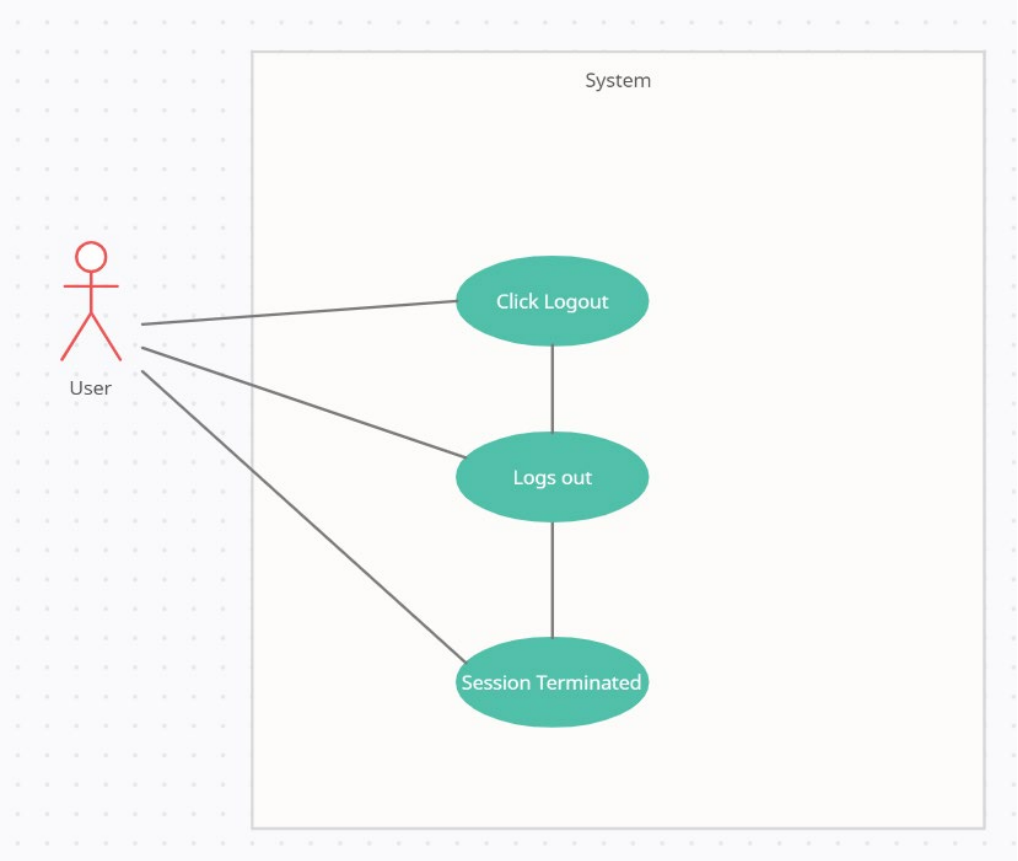
Scope

The scope of this use case is to show the user logging out of the system.

Description

This use case describes the action of the user selecting the log out button and the session being terminated.

Use Case Diagram



Flow Description

Precondition

The user is already logged in prior and they are on a page where the logout button is available.

Activation

This use case starts when the user clicks the logout button.

Main flow

1. The user clicks the logout button
2. The session is terminated and the user is redirected to the login page.

Exceptional flow

E1: <System cannot terminate >

1. The user clicks the button to log out of the system.
2. The database cannot terminate.
3. The user is prompted with a termination error.

Termination

The use case terminates when the user clicks the log out button and the session is terminated, the user is returned to the login screen.

Post condition

Success:

1. User has selected the logout button
2. The session is terminated and the user is logged out.

Failure:

1. The user clicks the logout button
2. An error occurs and the user is not logged out and is prompted with an error

3.1.1.4 Requirement 4 <Select Table>

3.1.1.4.1 Description & Priority

This use case describes the process of selecting a table that the user wants to create an order for. It is high priority.

3.1.1.4.2 Use Case

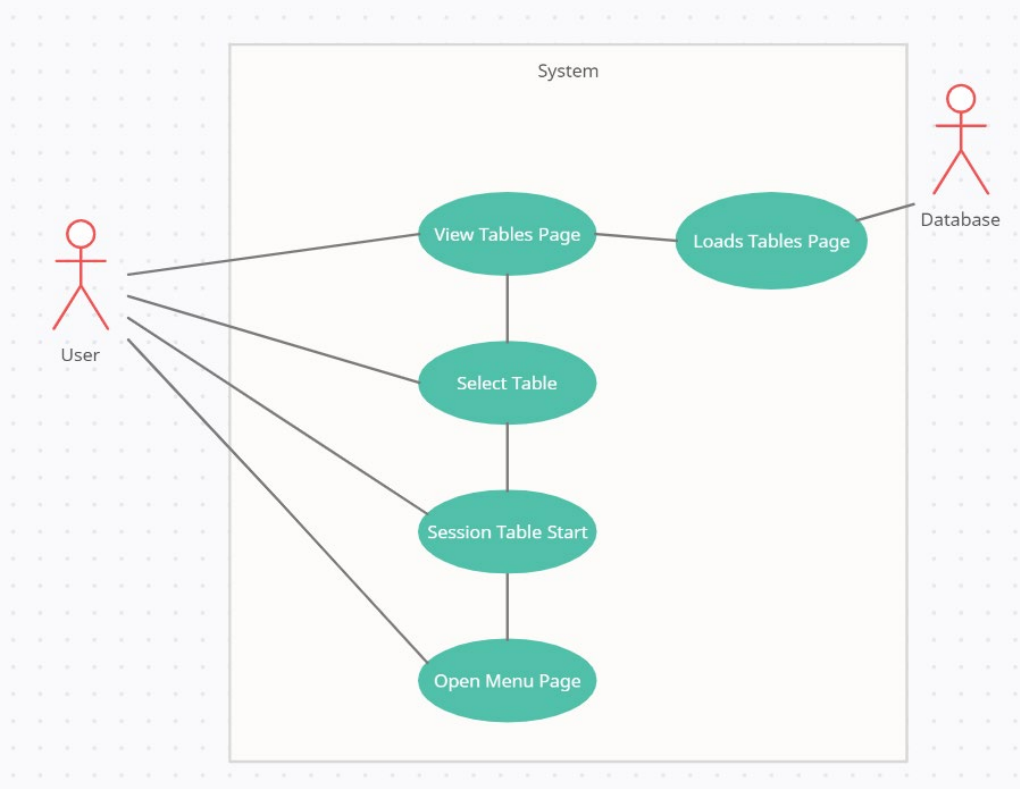
Scope

The scope of this use case is to show how the user will interact with each of the table numbers.

Description

This use case describes the interactions between the tables and the users.

Use Case Diagram



Flow Description

Precondition

The user has logged into the system and is viewing the tables page.

Activation

This use case starts when the user navigates to the table page and selects a table that they want to create an order for.

Main Flow

1. The user navigates to the table page
2. The user looks at the menu of tables that the database creates.
3. The user selects a table that they want to create an order for
4. The ordernumber session is created and the ordernumber of the user is updated
5. The user is navigated to the menu page.

Alternate Flow

A1: <Table page does not load from the database>

1. The user navigates to the tables page
2. The menu cannot be pulled due to a database error and the menu does not load.

Termination

When the user selects a table and the session for the order number is started and the number is updated for the user then the use case is terminated

Post Condition

Success

1. The user navigates to the tables page.
2. The user selects a table to serve.
3. The order number is updated in the database and the session is started
4. The system redirects to the main menu page

Failure:

1. The user does not select a table and therefore does not complete the use case.

3.1.1.5 Requirement 5 <Select Main Order>

3.1.1.5.1 Description & Priority

This use case describes the process of selecting a main order from the main menu page. It is high priority as this is the core idea for the system.

3.1.1.5.2 Use Case

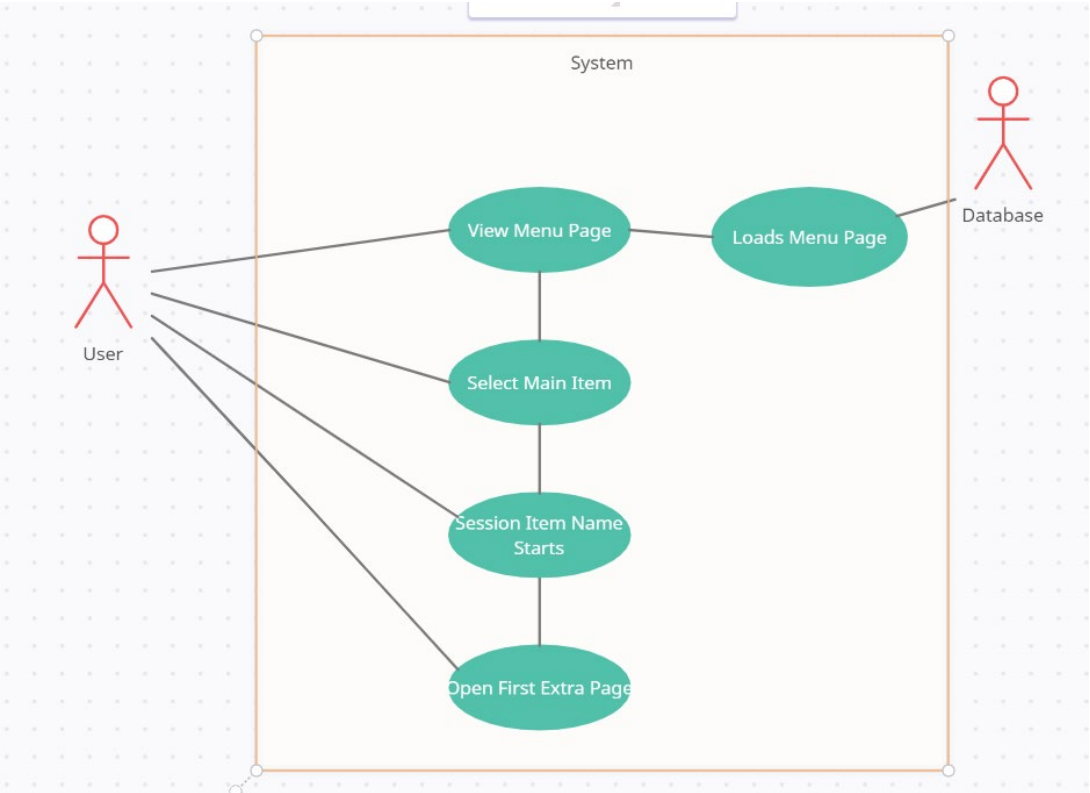
Scope

The scope of this use case is to show the user begin the process of creating an order by selecting a main item to add to their cart.

Description

This use case describes the user selecting an item and beginning their order.

Use Case Diagram



Flow Description

Precondition

The user is logged in, has selected a table and is now on the main menu page.

Activation

The use case starts when the user is on the main menu page and has selected an item to order.

Main Flow

1. The user is on the main menu page looking at the menu loaded in by the database.
2. The user selects a main item to add to their order.
3. The user is brought to the first customization page.

Alternate Flow

A1: <The user has made it to the main menu page without selecting a table>

1. The user has somehow made it to the main menu page without a table number assigned
2. The system will provide an error as they cannot send the order number correctly, this will not be added to the database under a table number

Termination

The use case terminates when the user has selected a main menu item and is navigated to the customization page.

Post Condition

Success

1. The user has selected a main item from the main menu.
2. The user is then navigated to the first customization page.

3.1.1.6 Requirement 6 <Side Order>

3.1.1.6.1 Description & Priority

This use case describes the process of selecting a side order from the main menu page. It is high priority as this is the core idea for the system.

3.1.1.6.2 Use Case

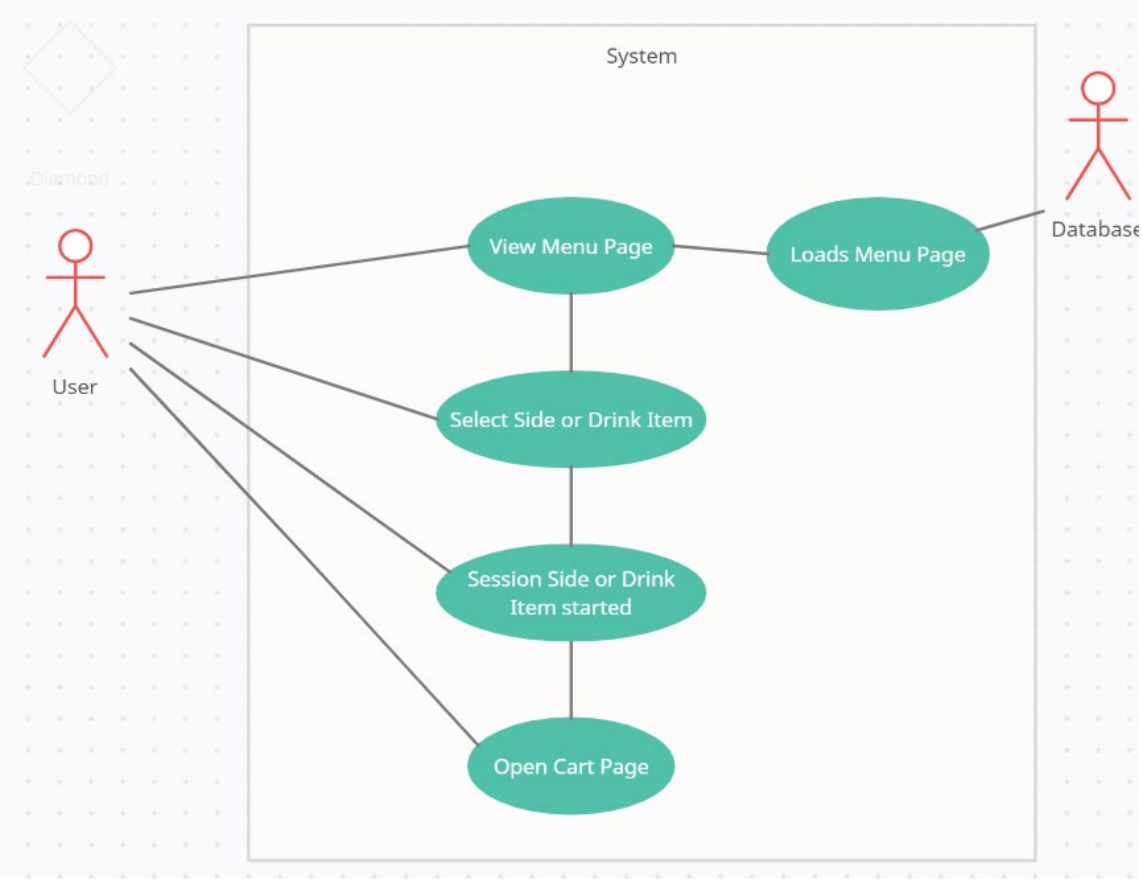
Scope

The scope of this use case is to show the user selecting a side order from the main menu page.

Description

This use case will describe how a user can select a side order and skip the customization phase that the main menu will entail.

Use Case Diagram



Flow Description

Precondition

The user is logged in, has selected a table and is on the main menu page.

Activation

The use case starts when the user has selected a side order item to order

Main Flow

1. The user selects a side order item.
2. The user clicks the item that they want.
3. The session for that item is started and they are navigated to the cart page.

Alternate Flow

A1: <The user has made it to the main menu page without selecting a table>

1. The user has somehow made it to the main menu page without a table number assigned
2. The system will provide an error as they cannot send the order number correctly, this will not be added to the database under a table number

Termination

The process terminates when the user has selected a side order item.

Post Condition

Success

1. The user is navigated to the cart page where they can begin the use case to review the order

3.1.1.7 Requirement 7 <Customize Order>

3.1.1.7.1 Description & Priority

This use case describes the process of customizing your side order in detail. It is medium priority as it is not mandatory for the main functions of the application.

3.1.1.7.2 Use Case

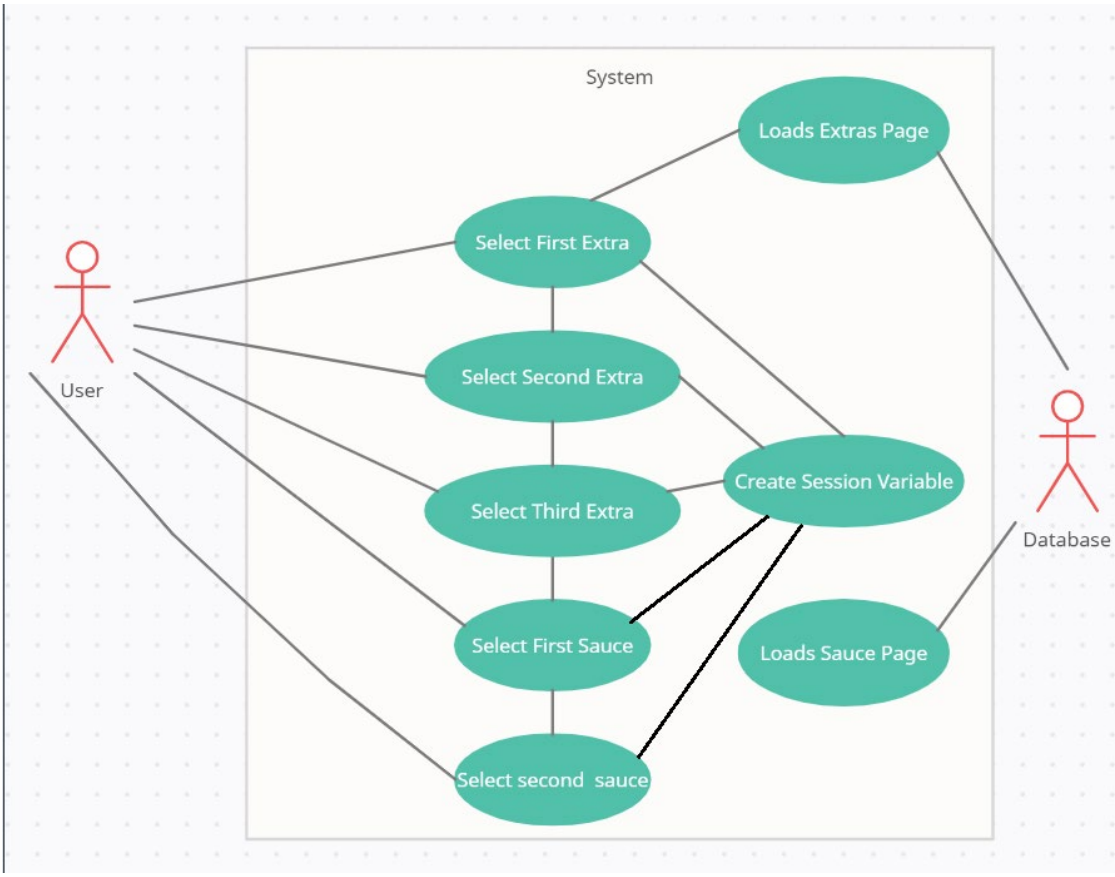
Scope

The scope of this use case is to show the user customizing the main order that they create.

Description

This use case will describe how a user can customize their order after selecting it and before they add it to the database, making each order unique to the user.

Use Case Diagram



Flow Description

Precondition

The user is logged in, has selected a table and main item and is on the customization page.

Activation

The use case starts when the user has selected a main item from the main menu.

Main Flow

1. The user has selected a main menu item to order.
2. The user selects the first extra they want for the item.
3. The user selects the second extra they want for the item.
4. The user selects the third extra they want for the item.
5. The user selects the first sauce they want with their item.
6. The user selects the second sauce they want with their item.
7. The user has the option to select no extras for the item.
8. Each item chosen starts a session variable that can be reviewed before adding it to the database.
9. The user is navigated to the cart page to review the order and begin the next use case.

Alternate Flow

A1: <The user not selected any of the options>

1. The user bypasses selecting an option
2. This creates an empty field and will default the value to null

Termination

The process terminates when the user has selected all customization options.

Post Condition

Success

1. The user is now on the cart page where they can begin the next use case.

3.1.1.8 Requirement 8 <Review Order>

3.1.1.8.1 Description & Priority

This use case describes the process of reviewing the order before adding the item to the database. It is medium priority as it is not mandatory for the main functions of the application.

3.1.1.8.2 Use Case

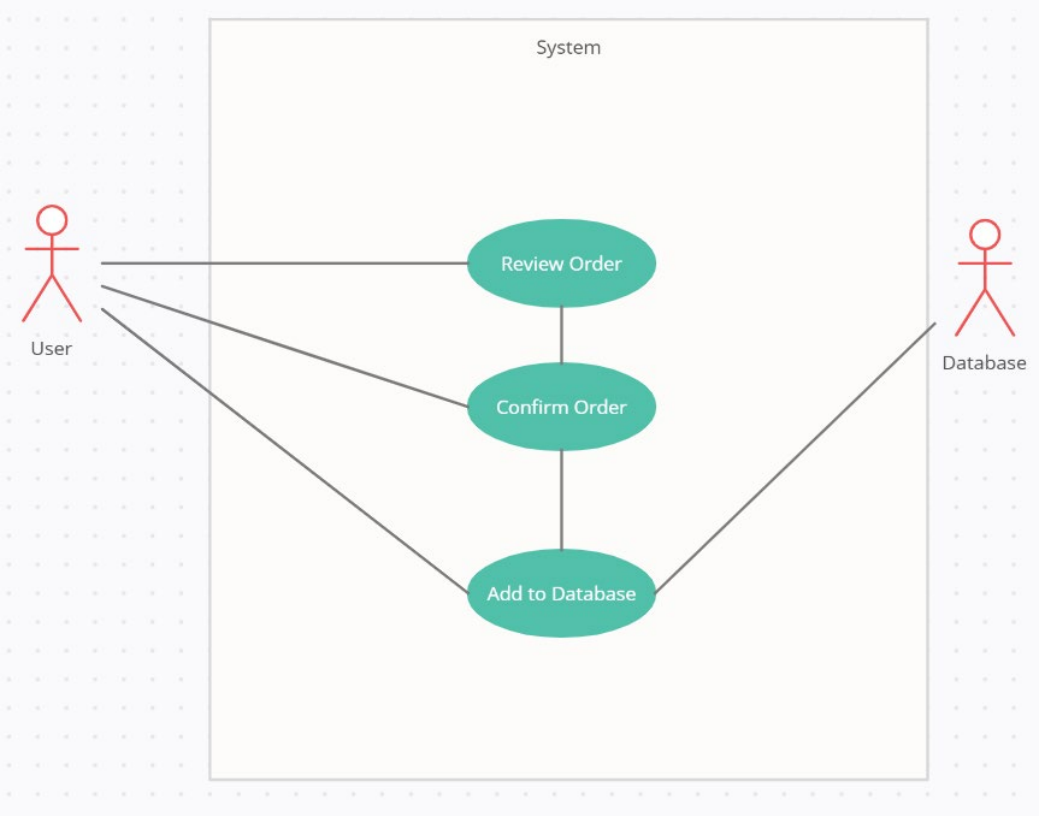
Scope

The scope of this use case is to show the user reviewing orders before adding them to the database.

Description

This use case will describe how a user can review their order before adding it to the database, therefore making sure that it is correct before it is added to the database and order.

Use Case Diagram



Flow Description

Precondition

The user is logged in, has selected a table, has added an item to their cart and is on the cart page.

Activation

The use case starts when the user has selected an item from the main menu and has been navigated to the cart page.

Main Flow

1. The user is on the cart page with an order in the review section.
2. The user selects to add the item to their cart.
3. The user's item is then added to the database.

Alternate Flow

A1: <The user does not want the item>

1. The user added an item to the order that they do not want
2. The user can then return to the menu to redo the item or navigate back to a previous page amend the problem.

Termination

The process terminates when the user has selected to add the item to their cart.

Post Condition

Success

1. The user has added the item to their cart.
2. The page refreshes
3. The item is added to the total order.
4. The item is added to the database.

3.1.1.9 Requirement 9 <Delete from Order>

3.1.1.9.1 Description & Priority

This use case describes the process of deleting from the order. It is medium priority as it is not mandatory for the main functions of the application.

3.1.1.9.2 Use Case

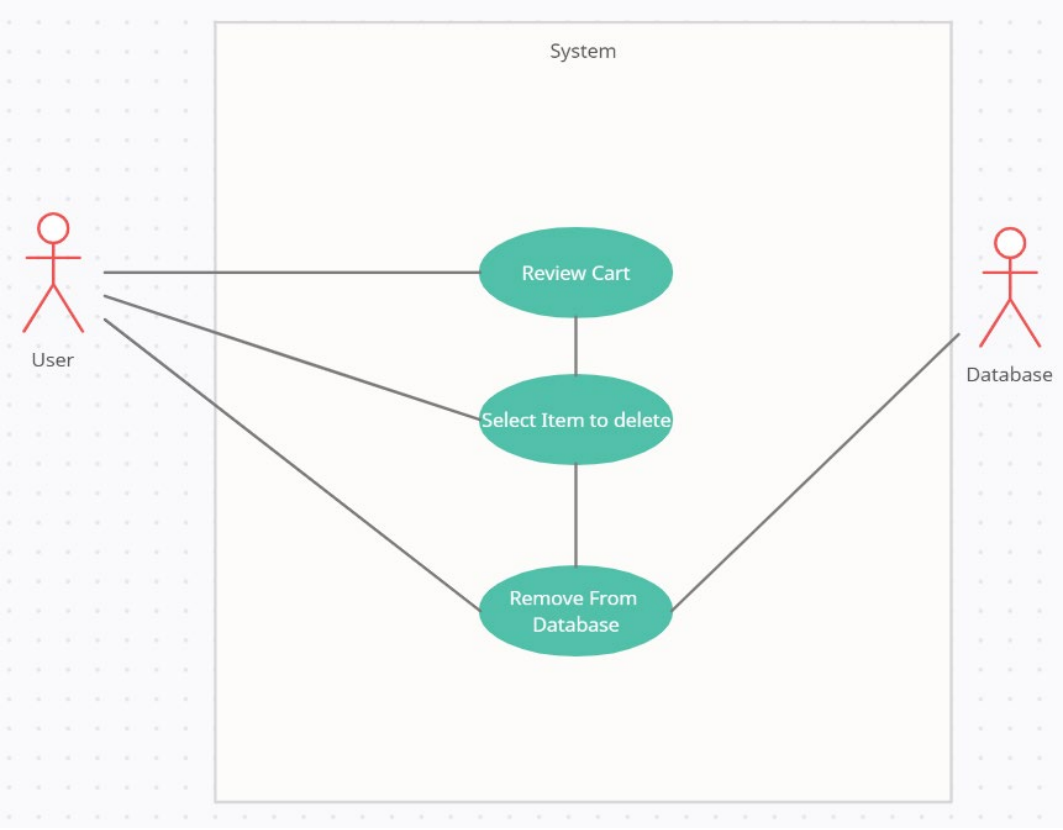
Scope

The scope of this use case is to show the user deleting an item from their cart after they have already added it.

Description

This use case will describe how a user can remove an item from their cart after they have already added it and reviewed it.

Use Case Diagram



Flow Description

Precondition

The user is logged in, has selected a table, added an item to the cart and is on the cart page.

Activation

The use case starts when the user has selected an item to delete.

Main Flow

1. The user has selected an item to delete.
2. The item is then removed from the database.
3. The page refreshes and shows the item now removed from the database.

Alternate Flow

A1: <The item has already been removed by another user>

1. The user selects to remove the item that has already been removed.
2. The page will refresh as normal as the item has already been removed.

Termination

The process terminates when the user has selected an item to remove and it is removed from the database.

Post Condition

Success

1. The item is removed from the database and the page refreshes
2. The user is now on the cart page where they can begin the next use case.

3.1.1.10 Requirement 10 <Delete from Order>

3.1.1.10.1 Description & Priority

This use case describes the process of paying for an order after it is complete. This is medium priority as it is not crucial for the use of the system.

3.1.1.10.2 Use Case

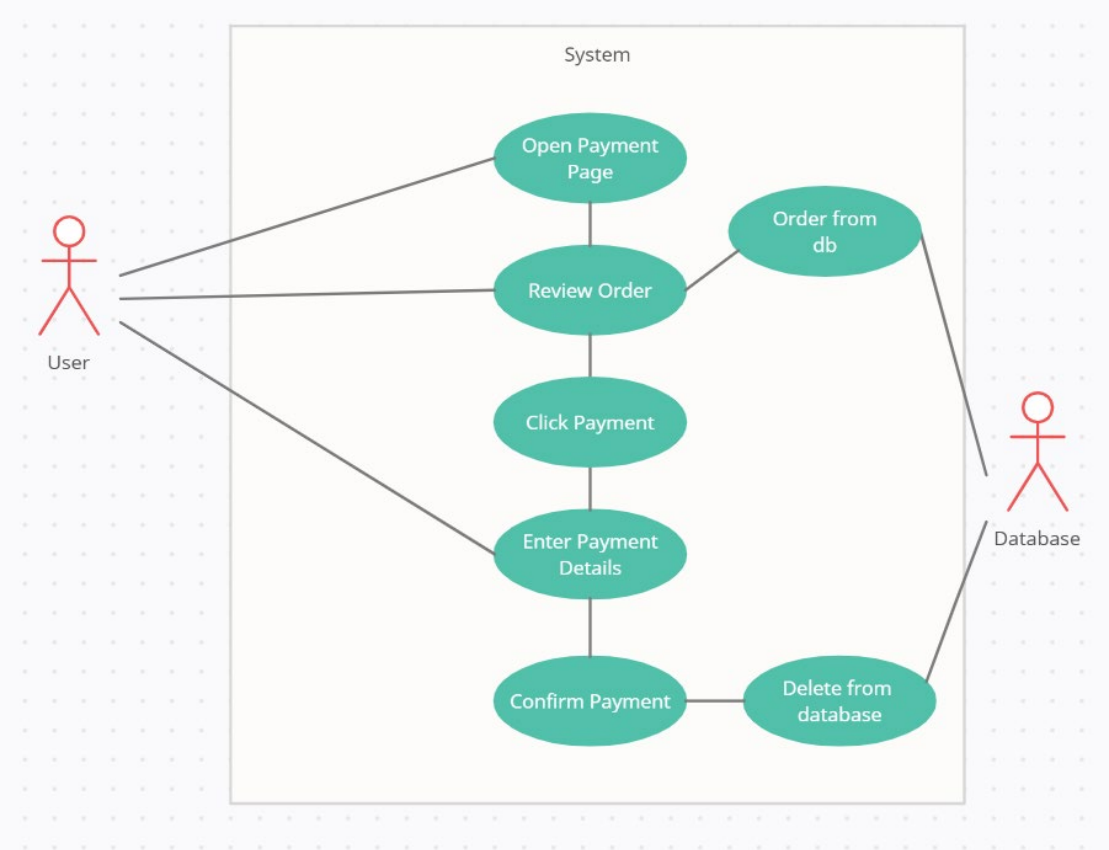
Scope

The scope of this use case is to show the users process to pay for the items in their cart after they have added them all

Description

This use case will describe how a user can pay for items in their cart and clear the database of the order to get it ready for the next one.

Use Case Diagram



Flow Description

Precondition

The user is logged in, has selected a table, has items in their cart and is on the payment page.

Activation

The use case starts when the user has selected to pay for an item.

Main Flow

1. The user has clicked the pay for item option.
2. The user is brought to the review page where they can review the order before payment
3. The user selects the paypal option to pay for the order
4. The user enters in their paypal details
5. The payment is accepted and they are brought to the success page where they have confirmation of their payment and can redirect back to the home page.
6. The order is removed from the database and is then ready for the next order

Alternate Flow

A1: <Paypal information incorrect>

1. The user enters in their paypal details.
2. Their details are incorrect or do not exist
3. The page refreshes as the process did not go through
4. The order is not confirmed and they have to attempt to enter their details again.

Termination

The process terminates when the user has entered in their paypal details correctly and have reached the success page that shows their order was complete.

Post Condition

Success

1. The payment is successful.
2. The items in the cart are deleted and a new order can be made.

Failure

1. The details are incorrect.
2. The user has to enter in their details again.

3.1.1.11 Requirement 9 <Kitchen View>

3.1.1.11.1 Description & Priority

This use case describes the process of viewing all incoming orders for the kitchen. It is medium priority as the user does not need this to use all of the other functions and it offers only convenience to the user.

3.1.1.11.2 Use Case

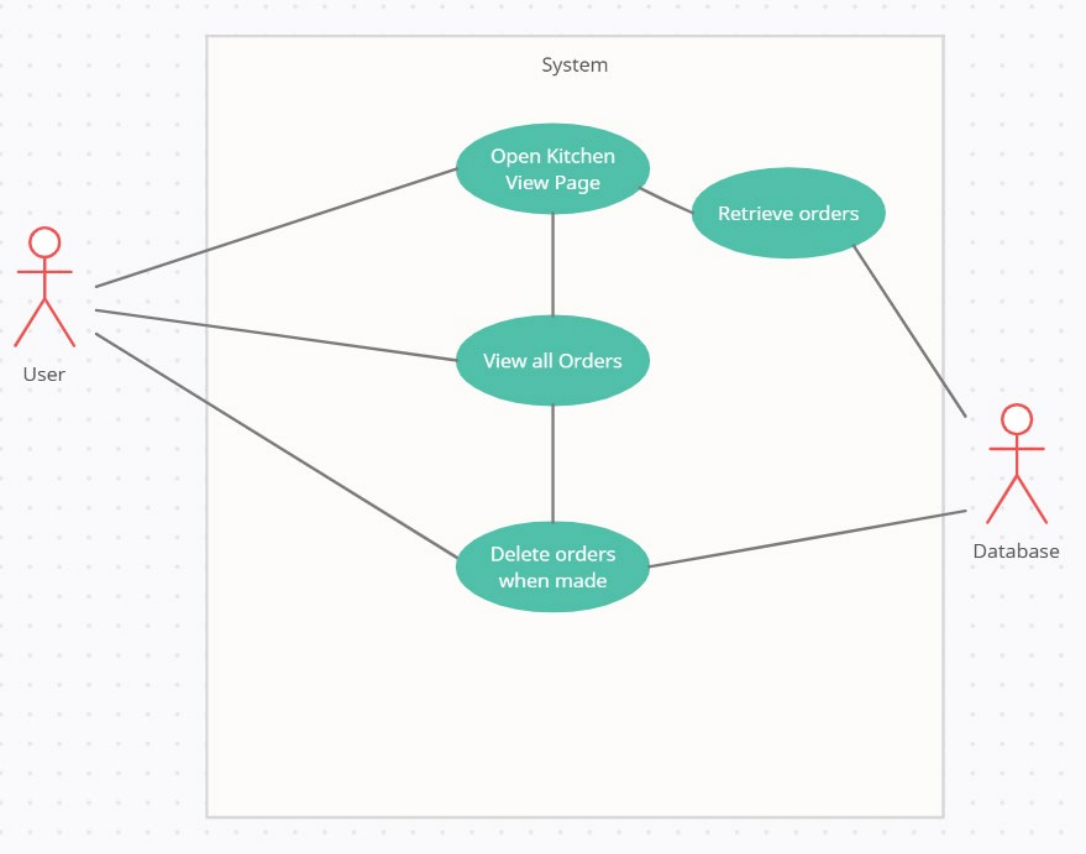
Scope

The scope of this use case is to show the viewing of the kitchen display page.

Description

This use case will describe how a user can view all incoming orders into the database. This will be a separate database that the kitchen use to allow them to remove orders as they make them.

Use Case Diagram



Flow Description

Precondition

The user is logged in, and has selected the kitchen view.

Activation

The use case starts when the user has selected to view the incoming kitchen orders tab.

Main Flow

1. The user has selects to view the kitchen orders.
2. They are navigated to the kitchen orders page.
3. They can view all incoming orders
4. They can delete orders from the database which updates and refreshes the page
5. They can view every order coming into the database regardless of table number

Alternate Flow

A1: <No orders>

1. There are no orders in the page.
2. The page will not display anything as there are no orders to view.

Termination

The process terminates when the user has selected and viewed the kitchen display page.

Post Condition

Success

1. The user has successfully viewed the kitchen orders page.
2. The user is able to freely remove items at the click of a button.

3.1.2 Non-Functional Requirements

3.1.2.1 Performance/Response Time Requirement

The performance of the application will be very important. This application has to be just as or more effective than writing orders down with a pen and paper. The application will require an internet connection as I will be using a database to store orders and user information. The application has some ways to make it faster, such as using as minimal graphics as possible to reduce the load on the processor. For some pages like login/register this will not be necessary as the user will be signing in prior to service but after this the system needs to be quick. This will mean that images will be need to be low resolution to avoid load when the application is loading.

The aim is to also have no file downloads or uploads throughout the application and have everything handled through the database. The main limiting factor will be the connection speed of the internet as this will slow down the interactions between the database and the application

3.1.2.2 Availability Requirement

The applications availability will be required throughout all of service as this will be used throughout the duration. The database will keep the information stored so the system does not need to be available at all times. The database should be launched with the application to recover all the stored data. Once the application has been accessed that's when the application needs to be available. Constant uptime of the application should not be a main priority, as long as the application is accessible during the work hours of the restaurant that will suffice. This will require an internet connection to connect to the database.

3.1.2.3 Recover Requirement

The recovery of data is vital as we need to recover previously used accounts to be sure that there are no duplicates. The database will load all previously stored data. In the case of a password being forgotten they will need to contact an admin to update the account. The application should not have data deleted on downtime from the database and instead rebooting the system should fix a majority of the problems.

3.1.2.4 Robustness Requirement

The application will not be clogged with features that could hinder the running of the application. This means that the applications robustness will not cause problems throughout the lifetime of the application.

3.1.2.5 Security Requirement

The user's password is encrypted so that when someone is viewing the database the password is not viewable by the person that is viewing the database. The password is hashed. Only accounts that have been successfully registered should be able to access the app. All other parts of the application should not be accessible without being logged in. The application will create a session when the user logs in to keep the user logged while they are actively using the application. The user must sign in every time they access the app unless they have a session stored.

3.1.2.6 Reliability Requirement

The application must be functional and reliable as it will be part of the ordering system. The application was tested throughout development for any errors and the reliability of page loading seemed to be consistent throughout. There were very little crashes and therefore the reliability seemed good for it. The main priority is to keep the system the same throughout, to have a linear navigation and to have a set format so that it is not jumping around consistently. The goal was to have a bug free and crash free application. From testing it seems to be free of any major bugs and does not crash.

3.1.2.7 Maintainability Requirement

Maintainability is a reliable feature of the application as it is created with the idea of other people reading the code, it contains comments and sections for each part of the code so that if someone else wants to edit the code they need to just go to the section of the page where the feature is that they want to change. I also have created folders for the files to make it easier to navigate through the code

3.1.2.8 Portability Requirement

The application will be a web app so it should run on all portable devices such as phones and tablets. This will be used primarily by people on the go so portability is hugely important for the success of the application.

3.1.2.9 Extendibility Requirement

Extendibility is definitely an aspect I focused on during the development of the application. Throughout the creation of the application I focused on updating databases that could, in the future, be used to manage stock, count sales and many other features. The app is built to be expanded upon.

3.1.2.10 Reusability Requirement

Multiple pages have a very similar layout so code reusability is vital. This works well as the design of the application should be similar throughout, having code reusability makes it easier to focus on other functions too as the code will only have to be tested one time then it can be ported to other pages. This means that if errors occur in other pages it will be easier to narrow it down as I have already tested a lot of the existing code so the error will lie with the code added to the new pages.

3.1.3 User Requirements

This section explains what the users will need to run this application.

- **PC/Laptop –** The user needs to have some form of a PC or Laptop if they wish to have the server hosted locally, otherwise they can deploy the server online. They can also use the pc or laptop to view the orders sent to the kitchen.
- **Mobile Phone/Tablet –** The user will need to have a mobile phone or tablet to create the orders on the go, this is required for the user who wants to be mobile while using the application, which by our target demographic, should be the majority of users.
- **Internet Access –** The user will need a stable internet connection to update the database on a real time basis, it will also be required to use the application as it is a web application.

3.1.4 Environmental Requirements

This section explains what was needed and will be needed to update the application in the future.

- **PC –** A computer was needed to create the code needed to run the website
- **Visual Studio code –** Visual Studio Code was the IDE of choice in this case as it was the best option for multiple languages at once.
- **Internet Access –** A stable internet connection was needed to work with the databases and to view all of the web pages created.

3.1.5 Usability Requirements

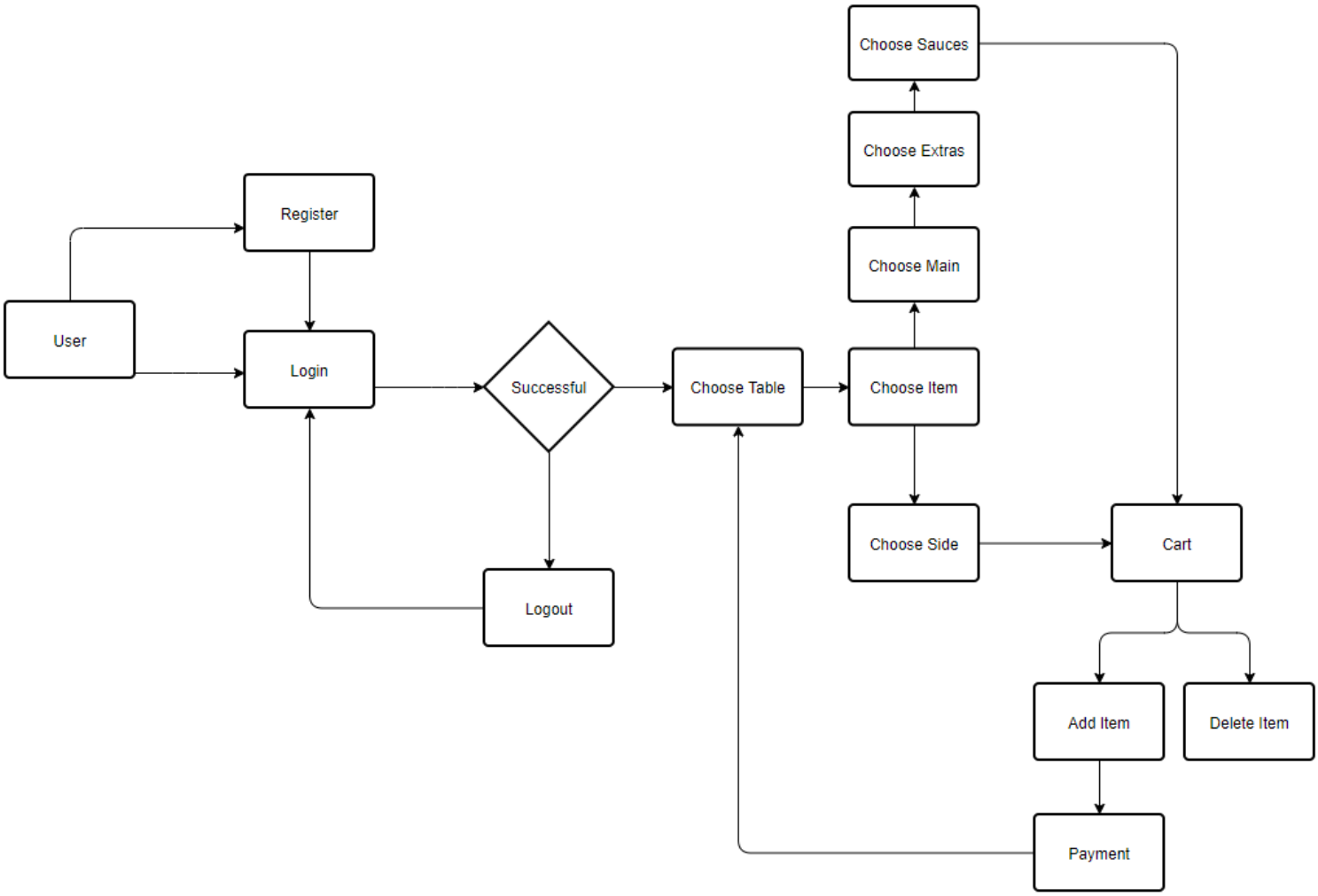
The user will need to have a mobile device to use the application correctly. This is because the application is marketed towards waiting staff who should be on the go and we do not want them returning to a static point throughout the day. The user will be required to create an account prior to using the application as they will need to be signed in to avail of all of the features that the application has.

3.2 Design and Architecture

The application was created in Visual Studio Code using PhP as the main language throughout the development lifecycle. The front end of the application was created using HTML, CSS, SCSS and JavaScript. It was designed to be very user friendly by having a bright and simplistic design and being primarily focused on buttons rather than entering into text fields as this would slow down the workflow of the user. Upon entering the application, the user will be required to log into the system to work through the application, the rest of the application will be locked out unless the user is signed in. If the user is signed in they will have the option to navigate to the tables page or view all current orders coming in. If they user selects to view current orders a page will be displayed showing all of the orders and will give them the option to remove orders when they are made. If the user selects the tables page they will be given a list of tables that is loaded from the database. When they select a table their order number will be updated to correspond with the table. The user can then select from mains or sides/drinks. If the user selects a main they are then brought to the customization page. If the user selects a side they are then brought to the cart page. After the user has gone through the customization of a main item they are also brought to the cart page. They are asked to review the item before they add

it to the database. Once added to the database they can then choose to add more to the database or to check out and pay. If they choose to add more they are brought back to the menu page. If they choose to pay they are brought to the payment page and asked to review the order before they pay. Once reviewed they enter in their payment details. They are brought to a success page which can then bring them back to the tables page to create the next order. The user can log out at any time

3.2.1 Design Walkthrough



Starts with user.

4 Interface Requirements

4.1 Web Application GUI

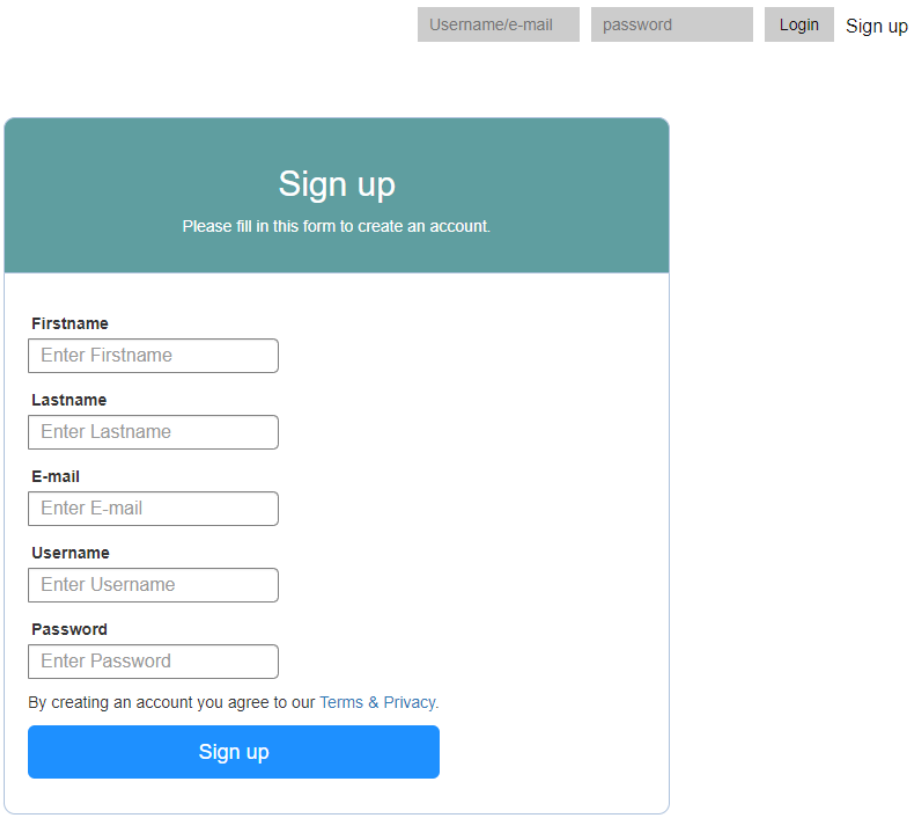


Fig 1. Sign up

The application opens and requests the user to sign up for an account. The signup and login page are very basic as they are not core features of the website and just act as the hub to get into the rest of the application. The application requires users to be logged in to use the features. The signup requests a first and last name, email, username and password. If the user already has an account they can login using the top right bar, otherwise they will have to create an account before they can continue. The password will be hashed once entered in to be extra secure when stored in the database

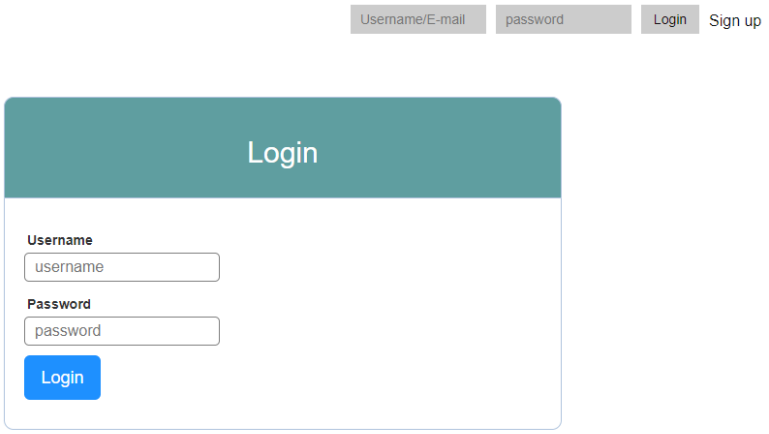


Fig 2. Login screen

Similarly, to the sign-up screen, the login is very basic as to not take away from the core features of the application. It requires an account to be made prior to logging in. It only requests two of the details entered, username/email and password. If the user does not have an account already they can just navigate to the sign-up page using the sign-up button in the top right of the screen. These pages have little design compared to the rest of the application.

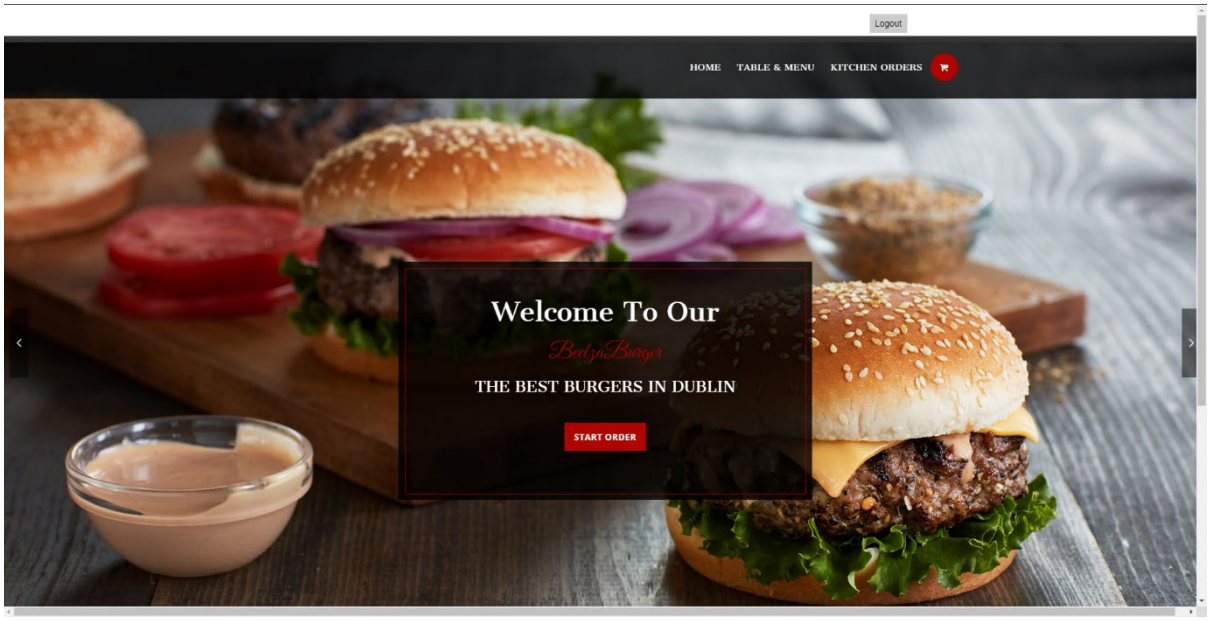


Fig 3. Login screen

The home page acts as the general hub for the rest of the application. It does not have many features other than to act as a nice welcome for the users and to allow the user navigate to the other pages within the application.

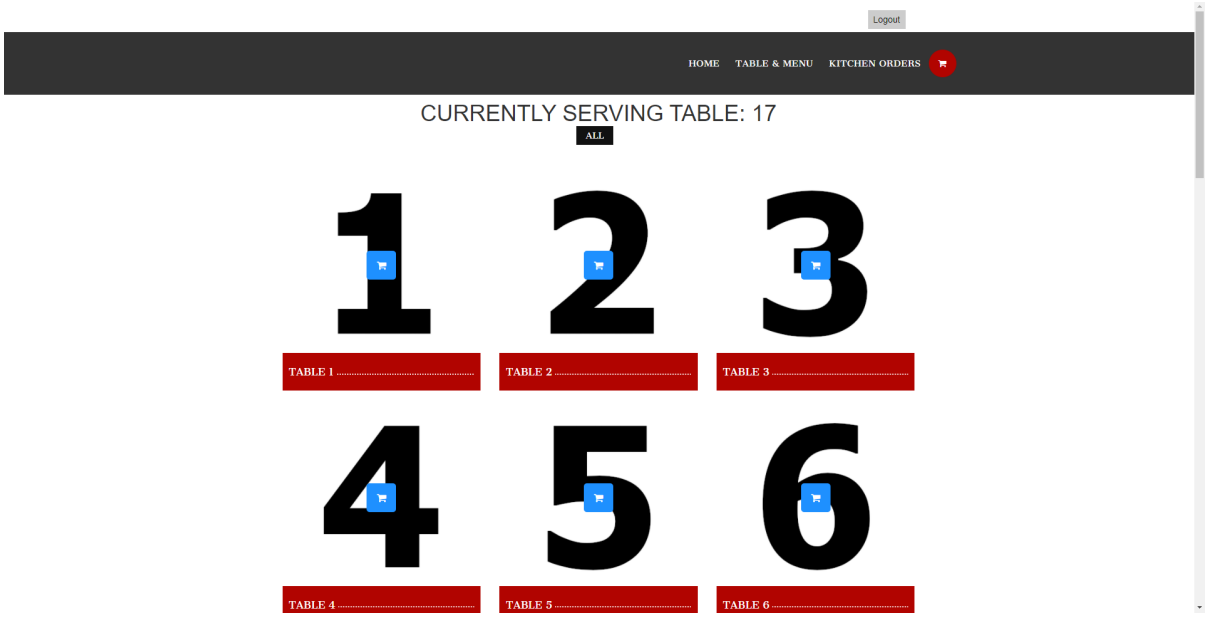


Fig 4. Table page

This page is the table page, it allows the user to select the table that they wish to serve. The page is created by pulling the information from the database, namely the name, the image and some hidden details that will be added to the database later, in this case the order number in the form of the table number. It will display the current table being served and this will be updated upon selecting a table.

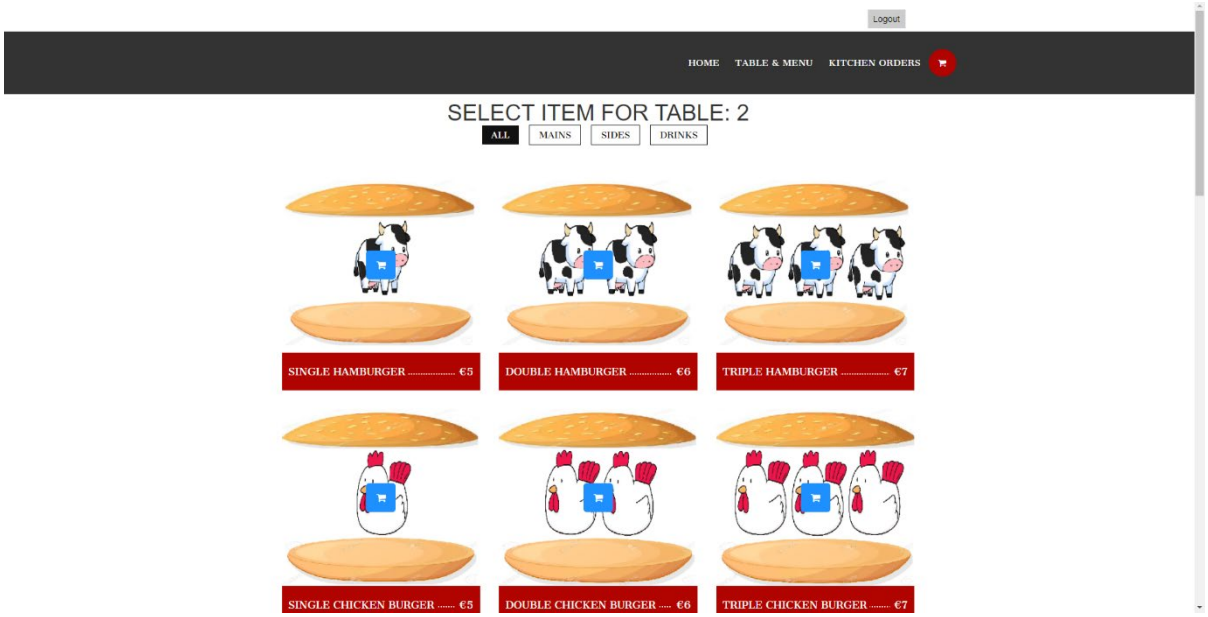


Fig 5. Menu page

This is the main menu page, it will allow the user to select from a list of mains, sides and drinks that are pulled from the database. It is generated the same as the tables page, by pulling the values from the database. If the user selects a main item it will navigate to the customization page whereas if they select a drink or side they will be navigated directly to the cart.

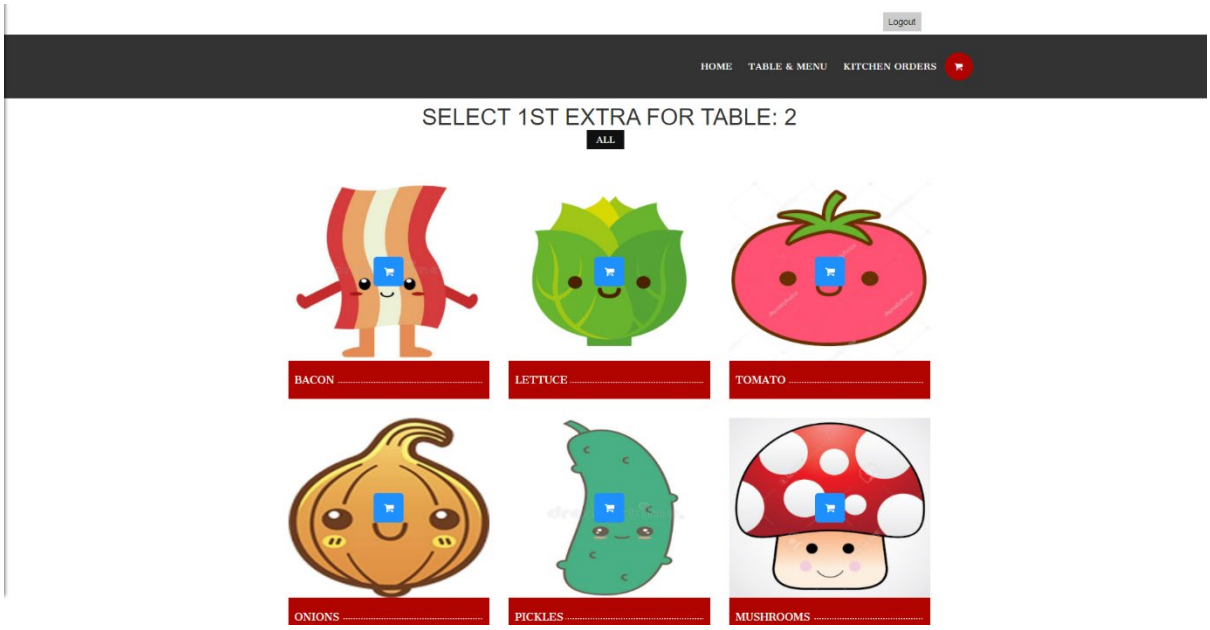


Fig 6. Extras page

This page allows you to customize your main. It will allow you to select three extras to go along with your main order, like the previous page this is also created by pulling the values from the database. After the items have been selected it will navigate to the next page which is sauces.

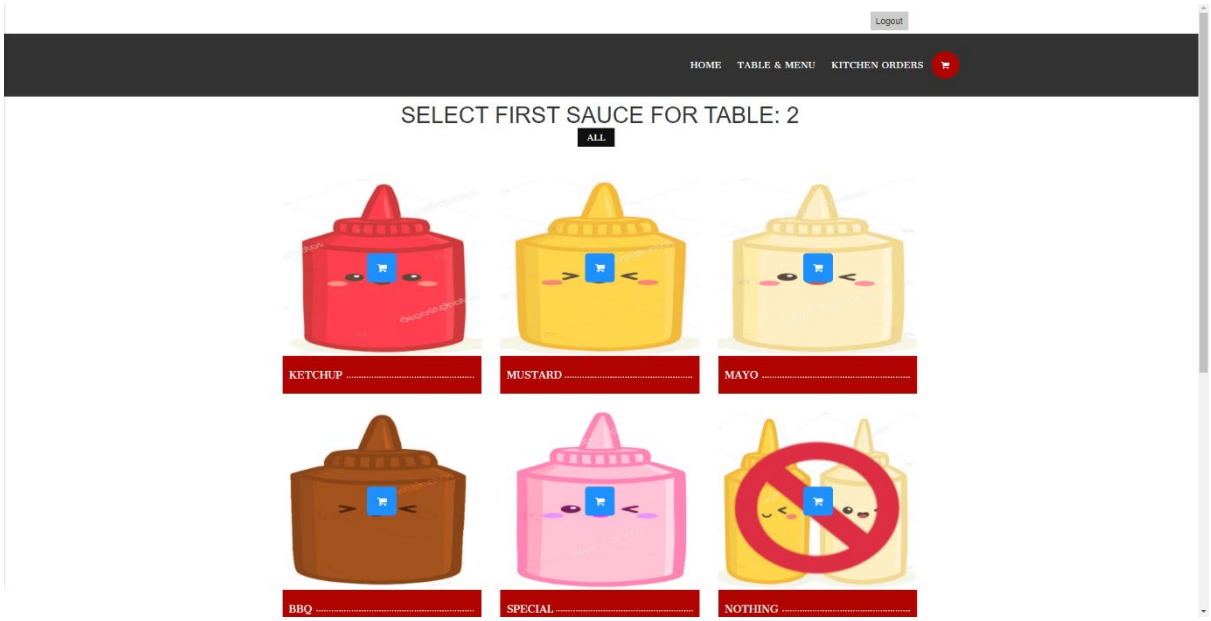


Fig 7. Sauce page

This page allows the user to select what sauces they want with their main. This was also pulled from a database. The idea with all of the menus is to keep them linear with a common design so that the user can see that they are the same throughout.

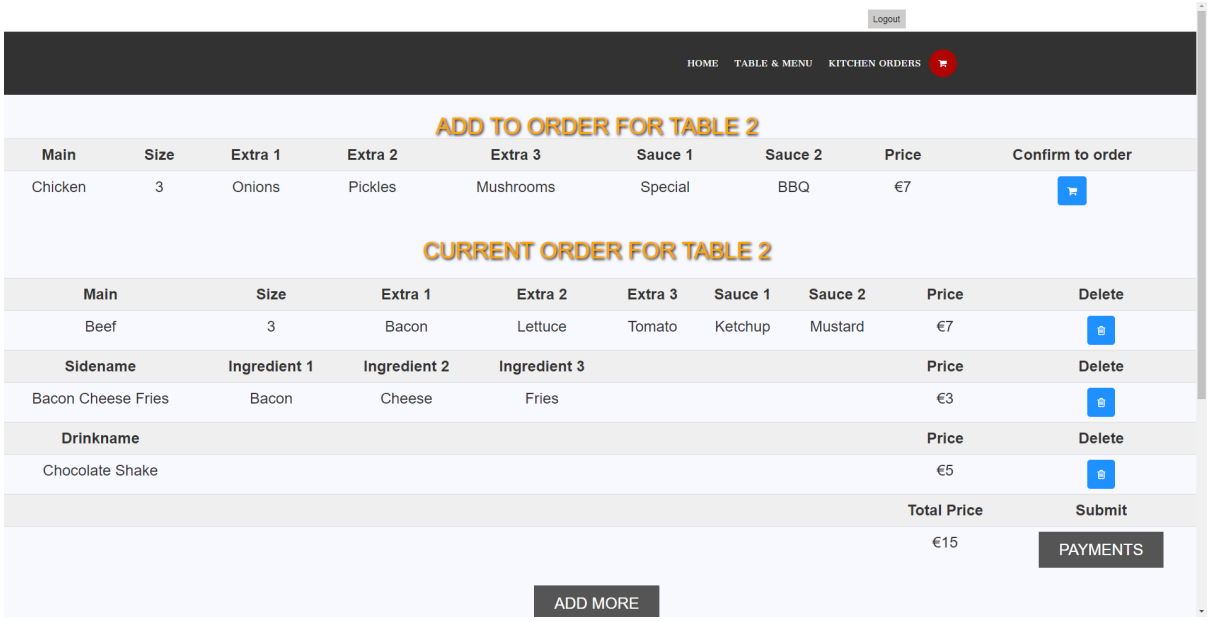


Fig 8. Cart page

This page is where all of the data is updated. The user is asked to confirm the item that they have selected. If they have created an item that they wish to add to the cart then they can add it to the database. If they create an item that they don't want then they can return to one of the previous steps to amend this. If they want to delete something from the current cart the option is there for that in the form of the trash button under the "delete" tab. When an order is added it is added to the database. The database then pulls all of the items added for that current table.

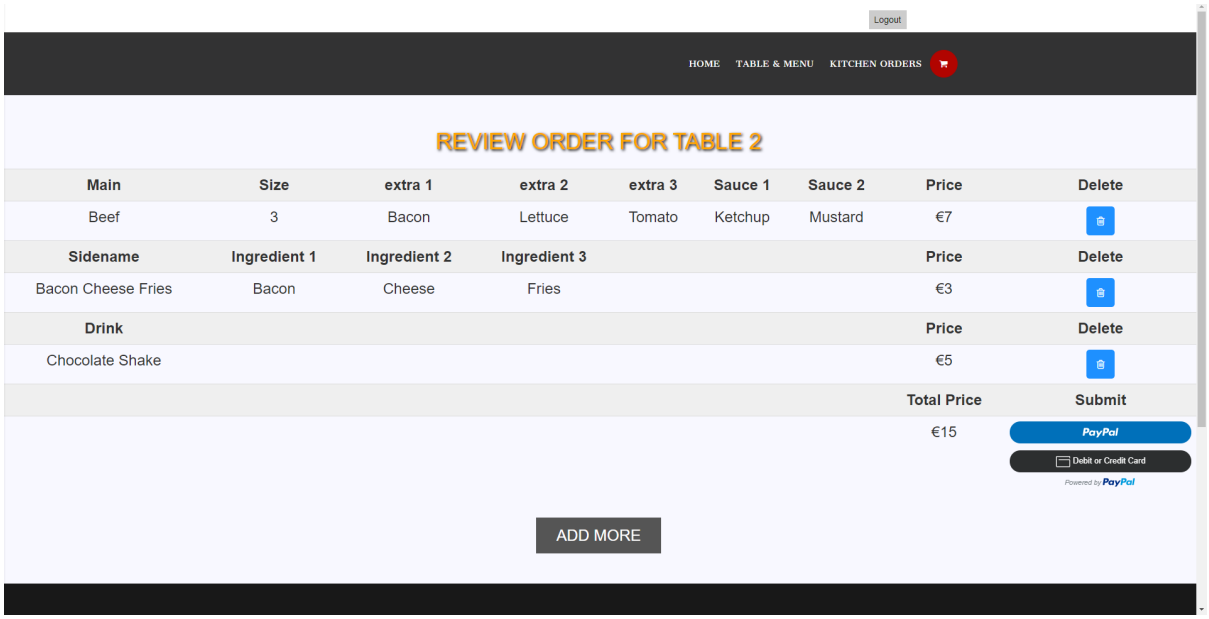


Fig 9. Payment page

The payment page is extremely similar to the previous page except it is made solely for reviewing the page before clicking the payment button. If the user wants to pay for their order then they can select the paypal button.

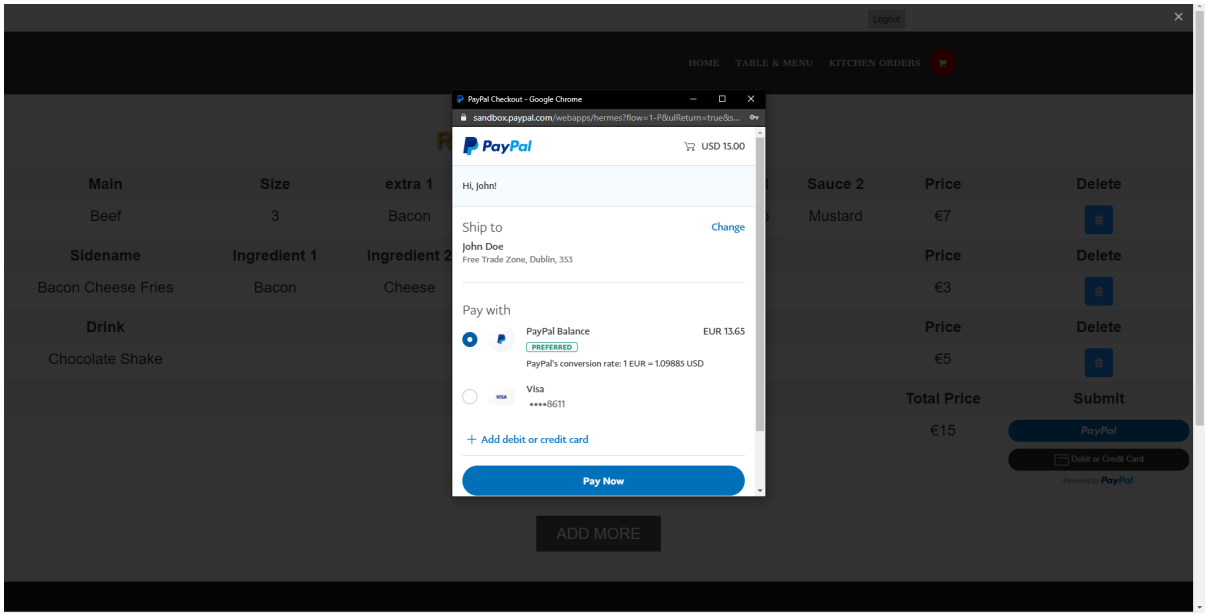


Fig 10. Payment screen

This page allows the user to pay for the order. If the payment goes through the order is removed from the cart and they are rerouted to the success page which confirms that the order has been paid for. If it has then they can go back and create a new order. If the payment fails or they cancel the page will refresh.

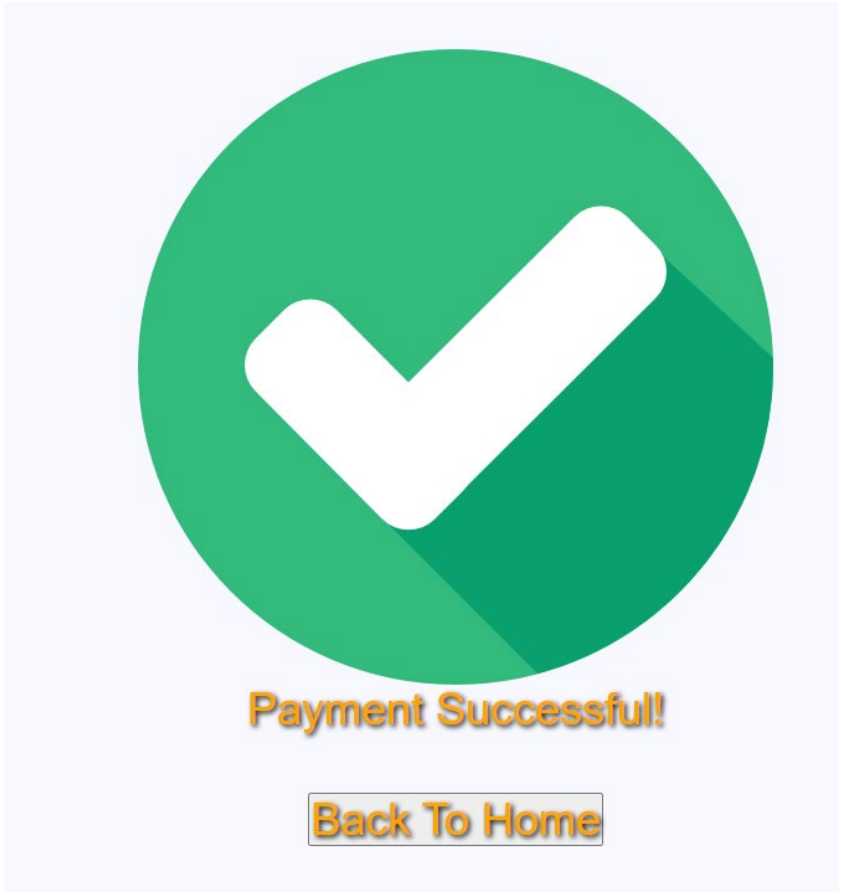


Fig 11. Success screen

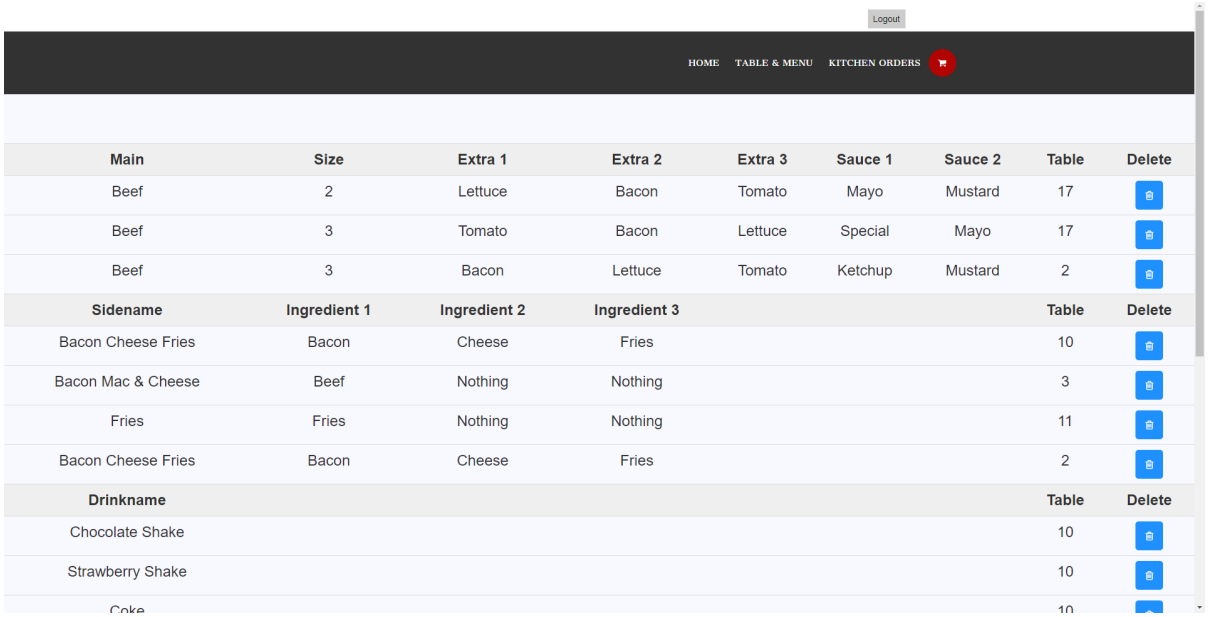


Fig 12. Kitchen order page

This page allows the users to see all pending orders regardless of table. They will be listed in the order that they are added to the database in a first come first serve basis. This is to be used by the kitchen staff and they can delete orders once they are made.

4.2 phpMyAdmin and SQLi Storage

+ Options

	id	firstname	lastname	username	email	password	ordernum
Edit Copy Delete	1	Gary	Lynch	beelzfaster	beelzfaster@hotmail.com	$2y$10$pQqHA7iXvbj6MvWYGBN4LuburKgf2yrHAJYyJ2SY4pC...	2

Fig 13.User database

The SQLi database is where all of the complexity of the application comes from. It stores all of the users, all of the orders and menus. It is where all of the information around the entire project is stored. Tables are automatically created on startup and when the items are added to it they are refreshed in real time. This allows for quick access to all of the information in the database which is secured behind the code that requires a credible login to use.

phpMyAdmin allows admin users to change records if needed and to delete unwanted entries in the case of an error. It is very useful for novice admins to take control of the database as long as they are connected to the database.

	id	firstitem	seconditem	thirditem	size	firstextra	secondextra	thirdextra	firstsauce	secondsauce	ordernum
Edit Copy Delete	11	Beef	Beef	Nothing	2	Lettuce	Bacon	Tomato	Mayo	Mustard	17
Edit Copy Delete	12	Beef	Beef	Beef	3	Tomato	Bacon	Lettuce	Special	Mayo	17
Edit Copy Delete	13	Beef	Beef	Beef	3	Bacon	Lettuce	Tomato	Ketchup	Mustard	2

Fig 14. Kitchen Orders database

	id	itemname	itemtype	firstitem	seconditem	thirditem	size	price	picture
Edit Copy Delete	1	Single Hamburger	Main	Beef	Nothing	Nothing	1	5	beef1.png
Edit Copy Delete	2	Double Hamburger	Main	Beef	Beef	Nothing	2	6	beef2.png
Edit Copy Delete	3	Triple Hamburger	Main	Beef	Beef	Beef	3	7	beef3.png
Edit Copy Delete	4	Single Chicken Burger	Main	Chicken	Nothing	Nothing	1	5	chicken1.png
Edit Copy Delete	5	Double Chicken Burger	Main	Chicken	Chicken	Nothing	2	6	chicken2.png
Edit Copy Delete	6	Triple Chicken Burger	Main	Chicken	Chicken	Chicken	3	7	chicken3.png
Edit Copy Delete	7	Single Veggie Burger	Main	Veggie	Nothing	Nothing	1	5	veggie1.png
Edit Copy Delete	8	Double Veggie Burger	Main	Veggie	Veggie	Nothing	2	6	veggie2.png
Edit Copy Delete	9	Triple Veggie Burger	Main	Veggie	Veggie	Veggie	3	7	veggie3.png
Edit Copy Delete	10	Fries	Side	Fries	Nothing	Nothing	1	3	Fries.png
Edit Copy Delete	11	Bacon Cheese Fries	Side	Bacon	Cheese	Fries	1	3	BaconCheeseFries.jpg
Edit Copy Delete	12	Bacon Mac & Cheese	Side	Beef	Nothing	Nothing	1	3	Mac.png
Edit Copy Delete	13	Vanilla Shake	Drink	Beef	Nothing	Nothing	1	5	Vanilla.png
Edit Copy Delete	14	Chocolate Shake	Drink	Beef	Nothing	Nothing	1	5	Chocolate.png
Edit Copy Delete	15	Strawberry Shake	Drink	Beef	Nothing	Nothing	1	5	Strawberry.png
Edit Copy Delete	16	Coke	Drink	Nothing	Nothing	Nothing	1	2	cokeLogo.jpg
Edit Copy Delete	17	7 up	Drink	Nothing	Nothing	Nothing	1	2	7upLogo.jpg
Edit Copy Delete	18	Fanta	Drink	Nothing	Nothing	Nothing	1	2	fantaOrangeLogo.jpg

Fig 15. Main Menu Database

5 Testing

Throughout the lifespan of the application it was tested extensively. Test classes were made to test features before they were implemented into the project. Before adding any major features, they were created on a branch that could be scrapped if they were unsuccessful. The two types of testing that were used throughout the lifecycle of the application were unit tests integration testing. After unit testing I usually followed with integration testing. The tests were performed using Visual Studio Code, PhPUnit and Composer, although PhPUnit and Composer are not supported anymore so these tests are possibly outdated. But this comes with the downside that from extensive research they seem to be the only tools available for unit testing.

5.1 Usability Testing

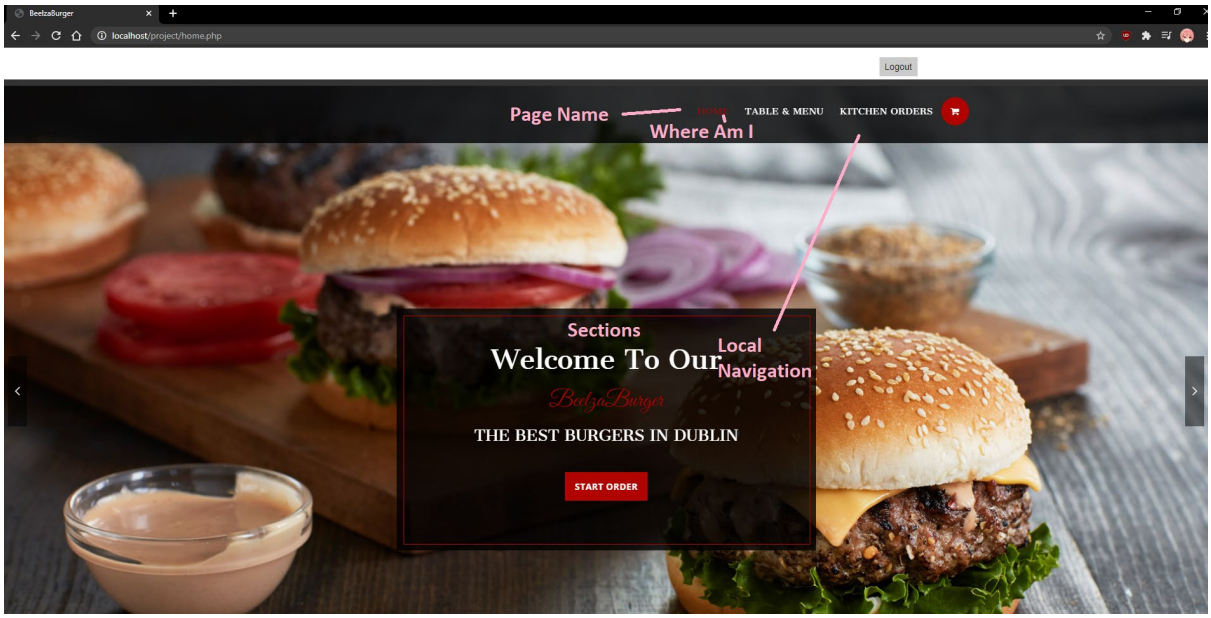
5.1.1 Trunk Test

This test was used to determine the key elements present for the products navigation. To do this I needed to have outside testers look at the five headings of trunk testing. The headings are:

- Site ID – Is there any indication of what website the user is on?
- Page Name – Is there any indication of what page the user is on?
- Sections – Is it clear what the main feature of the page is?
- Local Navigation – Is it easy to navigate through the sections within a given page?
- Where Am I – Is there a breadcrumb trail of where the user is at any given time?

Site ID	The application has the site name in the title of the page. Other than this there is not a strong indication of the website. This is intentional however as the website is made to be a template for other restaurants to slot into
Page Name	The page name is evident at the top of each page. It lets the user know where they are at any section.
Section	The sections are clearly defined and segregated into their own page to avoid confusion
Local Navigation	The navigation bar is visible in all pages except the login page. This makes it accessible to jump from one page to the next at any time

Where Am I	The page they are on is at the title of each page. This is how the pages are labelled, with a header at the top of the page.



5.1.2 5 Second Test

For this test I showed a group of 5 people images of the application. After 5 seconds I asked them multiple questions about the application. The questions were things such as what features were available, what kind of application do you think this is. The results were not up to the standard I hoped, it turns out the overall design is a little too minimalistic as they could not tell what the website was. But this is somewhat intentional as I have created the application in hopes that the main features can be replaced by any restaurant that wishes to add their own menu in. And in terms of identifying the website this is made specifically for the restaurant industry so I believe overall the 5 second test at least showed me that I need to improve the breadcrumb trail of showing users where they are and where they came from.

5.1 Unit Testing

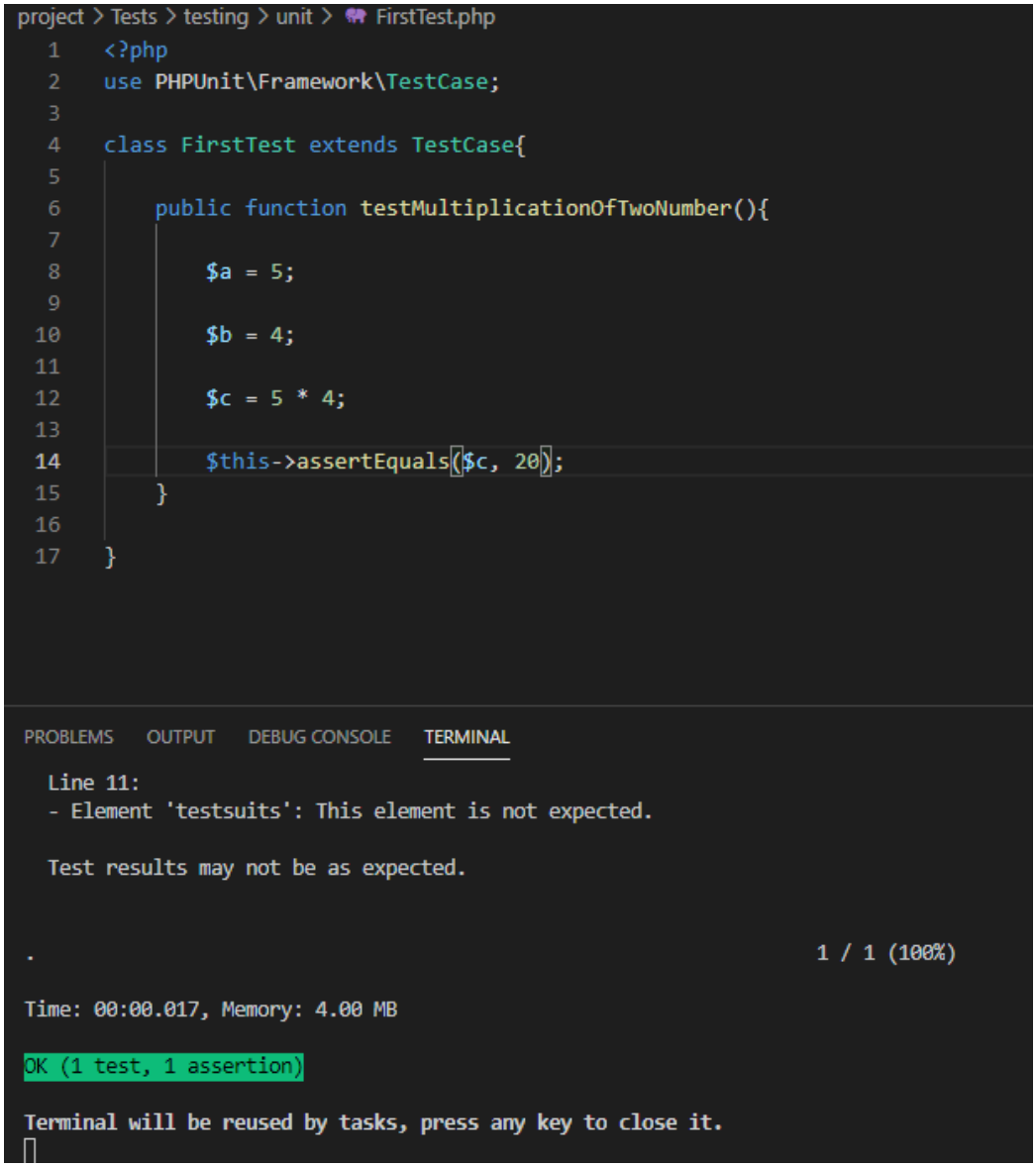
Unit tests were done using PhPUnit, a popular phpunit testing package. It is good for beginners who have little knowledge of what unit tests are. The idea of unit testing is to test small units of code, such as individual classes or methods. This makes it easier to go on and use these units as you know the units that have been tested work correctly and will not cause problems when implemented. The tests are run through the command line by navigating to the test folder in the project. In the application I used the test to test simple functions such as counting the total price before I implemented it or very basically getting the names variables which could be used for the database. These were created as separate classes from the rest of the application so I could test the features prior to adding them into the rest of the application. To run these tests yourself you will need to do the following:

Download Composer, the version of PhPUnit will depend on your version of PhP. If you have above 7 you will need the newest version, if you have less than 7 you will need the older version. Once you have this you will need to create a bin folder in the C directory of your computer for composer to work correctly as this is where phpunit will be situated.

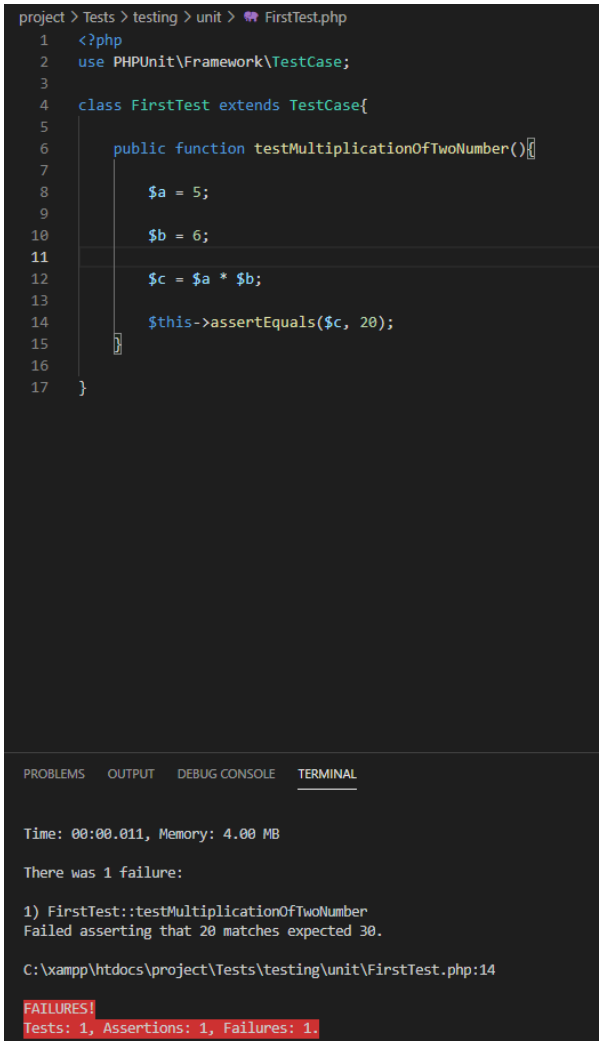
Alternatively, navigate to the folder where the project is stored in the CMD and type the following:

composer require phpunit/phpunit –dev

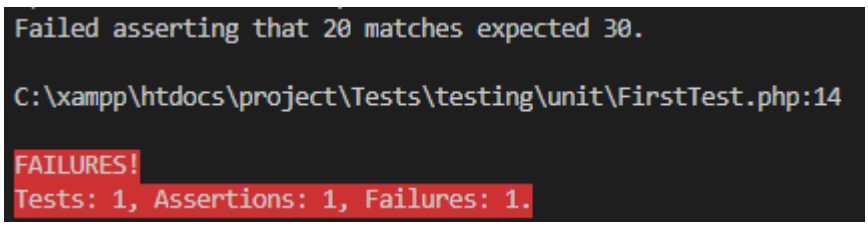
This will create the folder needed to test applications. After this I was able to test my applications features using unit testing. This can be done after composer is downloaded. This is the only way to test unit functions.



This is an example of a unit test I performed to test a basic calculator that I eventually ended up using for the total price. In the terminal the green box means that it was a successful test as it is working as intended, by running the test class I was able to perform a successful unit test on the basic features. It is very simple in concept but it allows me to create a small function that I can test now and implement later. In this case the calculator is checking that the sum of 4 x 5 = 20, simple right? Now let's change the number 4 into the number 6 and see what the output becomes.



The test failed because 6 x 5 does not equal 20. This lets me test small features before I implement them. This is very simple in a small scale but the larger the application becomes the more important these tests become. When it is just a few classes it would be ok to avoid testing smaller features but as an application grows unit testing becomes more important as it lets you avoid breaking the entire codebase due to a failed method. It also tells you exactly why a method failed, in the description I can see why the unit test failed.

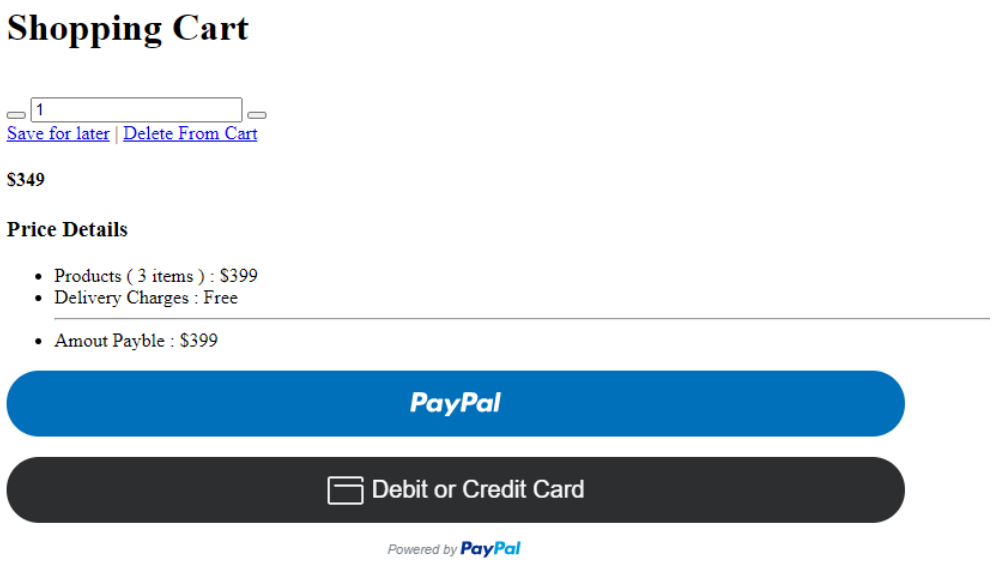


It failed asserting that 20 was the answer but instead it got 30.

5.1.1 Testing the Application

Even with the unit testing playing a fairly large part of the testing, the easiest way to test the application was in real time. As this was a smaller scale project unit testing, especially since PhPUnit is now unsupported, became time consuming. Making tests for small features became difficult when I could branch off using github and test the code that way before merging the branches. In this case I just created multiple test classes which I removed from the final application but still have saved in "testfolder". This allowed me to test the features of the application or test ideas I had in my mind prior to submitting. These classes were used to test SQL, design of the web page and test basic functions before I tied them into the rest of the application.

This is an example of a test I made for the basic design when I was first experimenting with tables. It's not pretty but it allowed me to test the buttons before I added some real front end to it.



Here is an example of the payments I made before I created a working page that pulled from the cart. It allowed me to test the paypal buttons and make sure they worked before I pulled them into the rest of the application. Any problems that arose throughout creation of the application I just went back to the pages and amended the problems at the time they arose. I tried to get rid of bugs as they arose rather than wait until the end and allocate time to bugfixes. This meant that the application didn't gradually get more and more bugs as the development went on and instead were solved as they were discovered.

5.2 Integration Testing

Integration testing is what follows directly after unit testing. The main goal of this is to select individual parts of the code that are to be tested. The functions of the code are tested and reviewed to see if they run how they were intended to. This is also where code is tested in parts and as a whole. We test functions to see how they run as a single chunk of code. Then we see how it runs as part of the entire code block. This was done, as stated prior, by creating classes to test small chunks of code and also to test whole classes to see how they work. In short, it is to test the chunks of code when they are combined. This is to avoid bugs that could not be caught in the unit testing phase.

The best way to test the application was to allow outside users to use the application. As the developer of the application I realised that I know where every feature is and how to navigate the application flawlessly. By delegating the work to others who do not have experience in app development but instead are experienced in the food service industry, my target audience, I was able to test the features of the application better. Due to the coronavirus my choices were

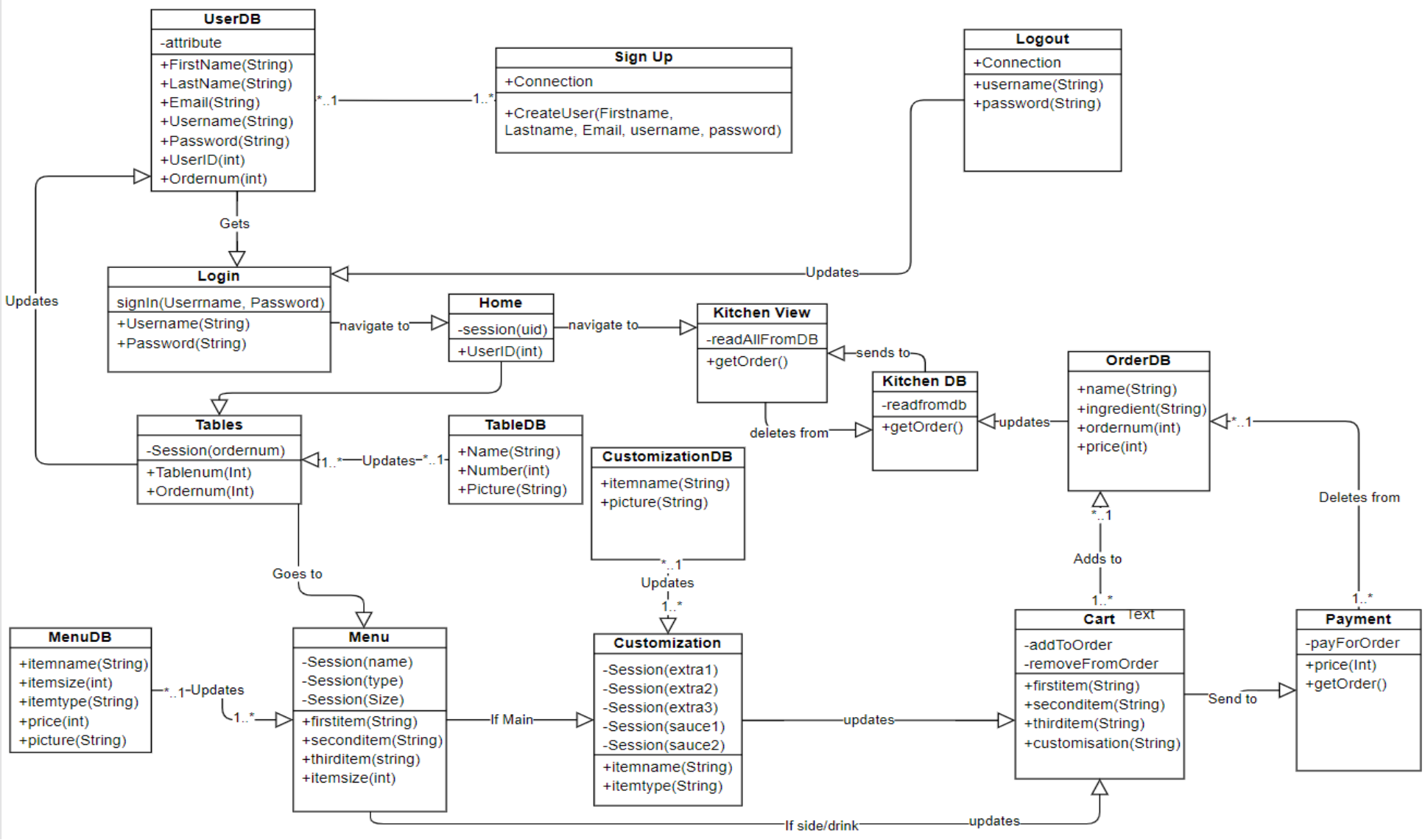
limited compared to when I first started, which I had planned to test the application in a restaurant.

I selected 5 people to navigate through a specific feature of my website to see how easily understandable it was. I allocated the first user to create and log into an account. The second user I asked to create an order and customise it. The third user I asked to delete orders and create a new one. The fourth user I asked to create an order for one table then create one for a new table. The fifth user I asked to pay for an order using the payment feature. Overall, the tests were a success. The only problems I had were people trying to navigate back through the website so I amended this by adding a button to jump them back to the start of the creation process for an order.

5.3 Architecture

The entire system was made using PhP as the main language and the database was created using SQLi with phpMyAdmin for management. I have created the application in a way that it is scalable in the future and can be expanded upon.

5.3.1 System Class Diagram



5.3.1.1 Class Diagram

5.3.1.1.1 Sign Up Class

Before the user can access the features of the application, the user must sign up for an account. To do this they will need to enter in their first name, last name, username, email and password. Their information, after they click the sign-up button, will be sent to the database after the password is hashed for security.

5.3.1.1.2 Login Class

The login class connects to the same database as the sign-up class. The user is prompted to enter in their username/email and password. When they enter in the details the database searches for the corresponding username and then checks if the password entered matches the column that the username is in. If this is correct then the user is logged into the system and the navigation swaps to the home screen.

5.3.1.1.3 UserDB

The UserDB is the user database where the information is stored. The database stores all of the signed-up users and when a login is attempted it is pulled from this database.

5.3.1.1.4 Home Class

This class is used as the hub to connect to other classes, it only really stores the session as an active login is needed to access the home class.

5.3.1.1.5 Tables Class

This class is where the tables menu is. It is pulled from the database and automatically created from there. The user can select from the menu one of the many tables available which will create the session variable that will be used later for the order. The connection to the SQL database is required as it is needed for the menu to appear.

5.3.1.1.6 TableDB

This is the table in the database where the tables information is stored, it is pulled from when the tables class is accessed. This creates the menus that the user will see.

5.3.1.1.7 Menu Class

This class is where the menu is. It is also pulled from the database and automatically created. The user can choose between mains, sides and drinks and choose whichever item they wish to add to their cart. If the user selects a main they are asked to customize, if they choose a side or drink they will be brought straight to the cart.

5.3.1.1.8 MenuDB

This is the table in the database where the menu is pulled from. It is required otherwise the information could not be obtained and therefore the order could not be made.

5.3.1.1.9 Customization Class

This class is how the order is customized. When the main order is selected they are asked to choose up to 3 extras and 2 sauces for their item of choice. These create session variables which will be used later by the cart to update the order and add it to the database.

5.3.1.1.10 CustomizationDB

This database is used to create the menus for the customization pages. The connection is needed to create the pages seen by the user and to create the variables needed to add to cart.

5.3.1.1.11 Cart Class

This class is where the order can be viewed from. This page allows users to add new items to their cart for the specific table that they have chosen. When they select a table, they are assigned a number which will correspond with the cart they have chosen. The order is added to the cart when they click to add it, they can also remove items from the cart if they wish to do so.

5.3.1.1.12 Payment Class

This class is used to pay for the items in the cart. It is similar to the cart class and pulls from the same tables. The user can select the payment option, when selected they will be prompted to enter in their paypal details. When they enter in their details they will be navigated to a success page which will navigate them back to the tables page to start the next order. When the payment is confirmed the current order is deleted from the database.

5.3.1.1.13 OrderDB

This is the table where all of the orders are stored. It is updated throughout the application with multiple orders and can be removed from at any time. It is pulled from to read at multiple occasions throughout the application too.

5.3.1.1.14 Kitchen View Class

This class is used to see all current orders. It is stored in its own separate table so that when an order is being created they do not have to worry about deleting from it when they have fulfilled an order.

5.3.1.1.15 KitchenDB

This table is used to display the kitchen view class. It is read from and deleted from as well as added to when an order is created. The kitchen staff can remove orders as they are made.

5.3.1.1.16 Logout Class

This class is used to log the user out from the application. When the user selects the logout button they are logged out of the system and the session is terminated.

5.4 Conclusion

Overall, I felt that the results for my application were to a level that I could be content with and allowed me to develop my application further. Each user that tested the application was surprised at how easy it was to use and enjoyed creating orders. Even the users that were not tech savvy were able to navigate the application with ease. From the feedback of the testing I tried to make the GUI more engaging by adding better pictures for the buttons so that they were not bland but other than this I think it was a success.

The class diagram was a huge help for the application as it let me see what I needed to create for each class. Although some of the variables changed throughout the basic structure of the application was still similar to what was in the class diagram in the end.

6 Project Plan

The project plan evolved somewhat throughout development and was difficult to stick to. When I was creating the plan, I tried to keep the phases divided. I first focused on documentation as this was to be done before I even began creating the application. Secondly, I focused on early development and prototyping of the application, which, it went through many iterations as I first started on cloud 9, then moved to android studio, then finally to creating in visual studio code using php. Finally, I tested my code thoroughly in the end to make sure that it was free of bugs and that it ran smoothly and without crashes. I have a Gantt chart created to show my expectations of the work I assumed I could get done versus the reality of what the process was actually like.

6.1 Gantt Chart

Fig 16 is my original Gantt chart that I had planned before the virus. I assumed I would be able to get the work done in the time I allocated myself but I hit a lot of roadbumps along the way and in the end, I was able to work through it despite restarting the project. Fig 17 is how the work looked after I had properly completed the project.

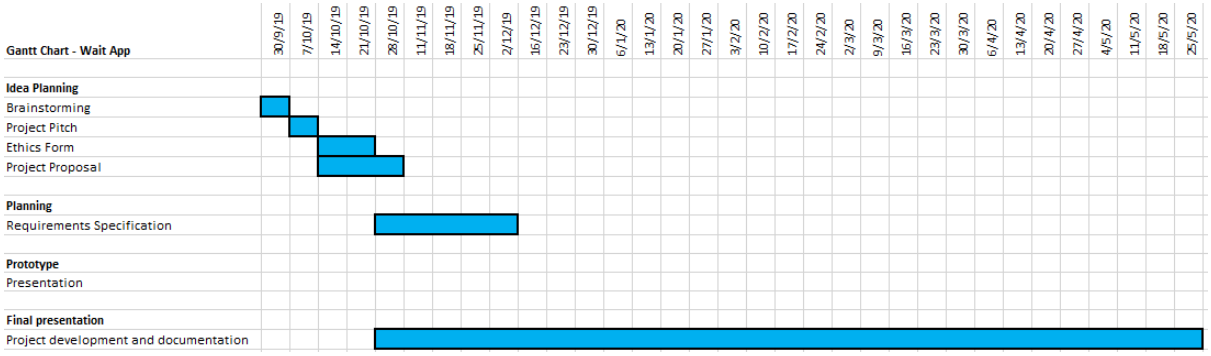


Fig 16. Initial Gantt Chart

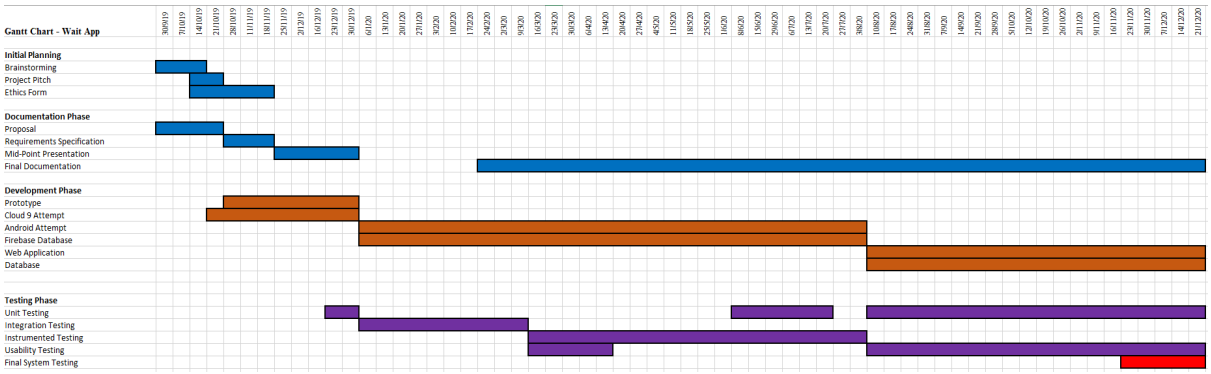


Fig 17. Final Gantt Chart

7 System Evolution

I got all of the key features that I set out to get into my project. I achieved these within the timeframe that I was given and have successfully created a working application. I have created features in the application that will allow for it to be improved upon and will create a scalable project that I can, in the future, add in more features if wanted. I feel that the application will continue to grow as I want to create an application that will help others develop similar apps and will also show the possibilities of expanding upon existing ideas. For extra features I already had some features planned that had to be cut due to the complexity involved. I want to add in a way to manage the stock for the application. I also wanted to create a way to create visuals of the sales. Another interesting feature to add would be to launch the application to a server but I would need to figure out a way to link the databases to the users.

In short, the features that I would hope to add in the future are as follows:

- A stock management system
- A means to visualize sales
- Deployment to a server

With additional resources and more times on my hands, and without the problems of the current global climate with the virus, I would like to expand on the idea of my Wait App.

8 User Manual

To use the application there are a few steps that need to be followed prior to launching the application.

1. The user will need to have xampp and mysql downloaded to use the application as this is dependent on the databases.
2. The user will need to create the two databases of “login” and “menu”, all of the tables will be automatically created when the user uses the application.
3. The project folder will need to be placed in the htdocs folder of xampp in order to launch.
4. Navigate to the http://localhost/project/home.php which should automatically put you to the login page
5. The user can create their own account to view the website and log in.
6. Beyond this the user can create orders and view current orders and go through the application as well as make fake payments.
7. To make payments they can use the fake paypal email and password as this is a sandbox account.

sb-vytxz1337812@personal.example.com	mypass123
sb-e6lw431594718@business.example.com	mypass123

If the user logs into the business account or the personal account they can see the transactions from the personal account into the business account

8.1 Link to GitHub

A link to my GitHub and the project repository can be found here.

https://github.com/Beelzfaster/project

9 References

Php.net. 2020. *PHP: Hypertext Preprocessor*. [online] Available at: <https://www.php.net/> [Accessed 22 September 2020].

W3schools.com. 2020. *PHP Tutorial*. [online] Available at: <https://www.w3schools.com/php/DEFAULT.asp> [Accessed 8 October 2020].

W3schools.com. 2020. *PHP Introduction*. [online] Available at: <https://www.w3schools.com/php/php_intro.asp> [Accessed 9 October 2020].

Tutorialspoint.com. 2020. *PHP Tutorial - Tutorialspoint*. [online] Available at: <https://www.tutorialspoint.com/php/index.htm> [Accessed 15 November 2020].

Code Institute. 2020. *What Is PHP Programming? - Code Institute*. [online] Available at: <https://codeinstitute.net/blog/what-is-php-programming/> [Accessed 17 December 2020].

W3schools.com. 2020. *PHP Sessions*. [online] Available at: <https://www.w3schools.com/php/php_sessions.asp> [Accessed 19 November 2020].

SkillForge. 2020. *How To Create A Database Using Phpmyadmin And XAMPP | Skillforge*. [online] Available at: <https://skillforge.com/how-to-create-a-database-using-phpmyadmin-xampp/> [Accessed 16 December 2020].

Swcarpentry.github.io. 2020. *Creating And Modifying Data – Databases And SQL*. [online] Available at: <https://swcarpentry.github.io/sql-novice-survey/09-create/index.html> [Accessed 14 October 2020].

Mark Otto, a., 2020. *Bootstrap*. [online] Getbootstrap.com. Available at: <https://getbootstrap.com/> [Accessed 15 November 2020].

W3schools.com. 2020. *PHP Mysql Insert Data*. [online] Available at: <https://www.w3schools.com/php/php_mysql_insert.asp> [Accessed 13 November 2020].

10 Appendix

10.1 Appendix 1 – Project Proposal

Project Proposal

Wait App

Gary Lynch,

X17140382

X17140382@student.ncirl.com

BSc (Hons) in Computing

Software Development

21/10/2019

Objectives

The objective of Wait App is to replace the standard ticket/docket system that restaurants currently use and replace it with an application. It will allow the wait staff to take orders from the customers and send the orders to the kitchen. It will separate tables by number and will have submenus for drinks, starters, mains etc. The system will allow the wait staff to create a digital receipt that can be sent to the kitchen in place of the standard written up tickets that they currently use.

Background

The idea for this project arose when I encountered problems with getting a job as wait staff due to my poor hand-writing. Even in my current job my handwriting causes issues and this is where the base idea came from. I originally thought of just creating an online notepad of sorts that would just let the wait staff create a receipt digitally after clicking all the options they wanted but after more thought the idea of sending it directly to the kitchen, effectively cutting out a journey to and from the kitchen.

Technical Approach

Research

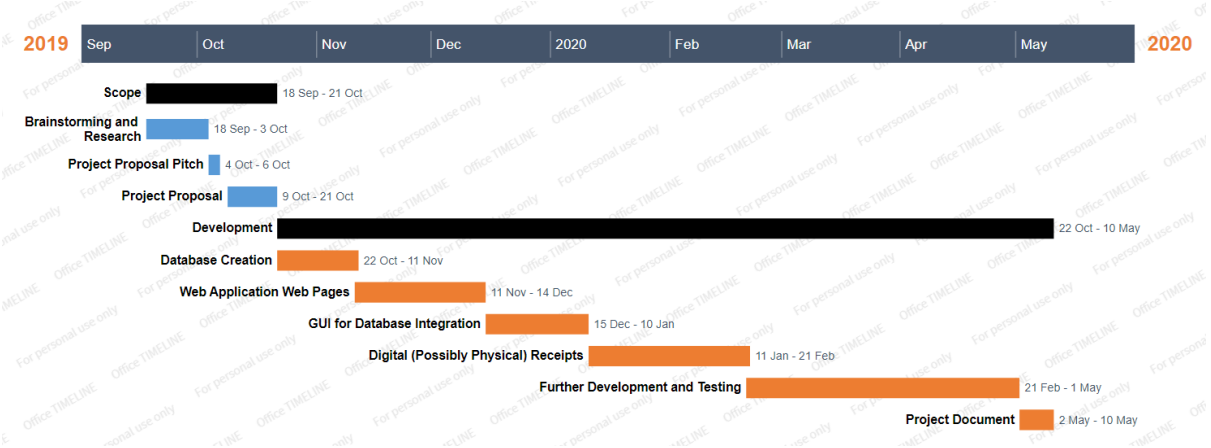
I have to research the use of Receipt printing machines to see if this is a viable option for my application, if this is the case this would give a different spin on how to send orders to the kitchen as this would allow the receipts to be physical instead of solely digital which is the current plan, I also need to see if integrating this with existing systems is an option however from research this seems unlikely as existing systems range from brand new to severely outdated.

Development

As of now I am hoping to make this a web application to make it a viable option on all phones, initially I had planned to use android studio but that would alienate non-android devices. Creating a web application means it is open to more devices and overall will make the project more manageable as my experience with android studio is almost non-existing. I plan to develop the web application using HTML/CSS/Javascript, using PhP for the database

integration. It will use a MySQL database to store the orders and create receipts. This is subject to change as I research more into the development.

Project Plan



Evaluation

I will test each new feature as it is implemented. I will attempt to create the database and a barebones application that will take basic orders with no customisation to test the base concept then add features and test each of them as they are implemented. This will hopefully mean that at the core the concept will work and each iteration from then on will have more features. The main goal is to be able to select a table, select items from menu, send items from menu to a different page. If this is achieved then the base concept works. Gary Lynch

Signature of student and date

10.2 Appendix 2 – Monthly Journals

Reflective Journal – October

I was pondering over a few ideas since the start of third year and really struggled to find one that I clicked with. I was worried my "best" idea would be too similar to a previous idea I had done but the one that stuck out to me the most was an application to somehow remove the problem of handwriting in the waiting industry. I got the idea first talking to my friend about how I could never be a waiter because of how unintelligible my handwriting is and from there I just expanded on the idea. I was worried that this one would be too similar to a bar app I had done previously but looking back that application was insanely basic and was begging to be expanded upon. Improvement on existing technology is innovation and I feel that expanding on my own ideas would be a good way to learn from the many mistakes and regrets I have existing with my old application.

Most of my family has been in the service industry for many years, my mom has for most of her life so I began to question her and ask why she doesn't currently use a point of sale system. When I asked she laughed and said "There was a day we had to write the orders on napkins because we didn't even have paper and you think the boss will pay to test a system". Another friend told me that the register they currently use is from two owners ago. That was where I came up with the idea of somehow try to not replace the systems that are currently in place but to work alongside them. In short, I just want my idea to make their job easier, not replace anything they currently have. I also want it to be testable without having to purchase much, if any, new technology. Ideally this will just be testable on the waiting staffs' phones with another system for the kitchen to see orders.

After I thought of a few ideas for the system I had to think of a name. At first, I thought of Pocket Docket as this is going to be a phone docket system, however there's already a docket that's pocket sized called that. Next, I thought of wait4me but this is already taken. I tried multiple other names that were already taken then I decided to try to get a name that can be shortened into something snappy. That's where I came up with the name "Waiting Application Integrating Tickets". This can be shortened down into WAIT and it tells you exactly what the app is for so I like it for now but if I think of something better in the future I'll leave it open to change.

I still haven't met with my supervisor yet but I have gotten feedback on my video pitch. Some of the things I received feedback on I don't agree with really as I repeated a few times

in the video. For example, one of the notes was to integrate payments such as cheques, credit cards and tips. I feel like once I start to handle the money side of things this app drifts away from the whole point where I want this to just help make a wait staffs life easier. I feel like if I'm adding payment then I should just allow the customer to order stuff themselves and then the wait staff just have to wait and bring it to them. I asked my friend and my mom about taking payments and they both said all payments are taken at the end when they're brought up to the till, I feel at the point I'm allowing for my application to handle payments they will need to update their systems to accommodate. Again, I always keep in mind the very first thing I asked "Why don't you currently use one" and their answers were basically they didn't want to upgrade. In theory extra features to accommodate multiple scenarios is great but I find that if I just put in features just for the sake of putting in features the application will become bloated and lose focus. My bar application is an example of this, I remember Keith Maycock was our tutor at the time and he said exactly that. "More features are nice but I don't want you to take away from the goal of the app was just to make your life easier". I didn't understand what he meant until I had finished adding in all the extra features, the app worked and did what I set out but it felt bloated. I'll talk to my supervisor about it soon and update next journal but hopefully I can somehow convey these points appropriately.

Gary Lynch – x17140382 (student signature and number)

Reflective Journal – November

For this month I tried to look at how I planned to code this. At first, I was stubborn and thought that android studio would never be an option for me because I had zero experience with it. However, after doing some of the Mobile Applications class I have found that I actually really enjoy android studio. Parts of it are clunky, especially the emulator and testing but the feedback the application gives is so helpful, it makes it easier to practice unfeasible things in my mind and actually figure out a way to do them so I might change to android studio in the near future. But for now, my requirement specification and my knowledge lie with basic coding so I am attempting to make a login that connects to a database. I am attempting to learn about password hashing alongside this.

I have started looking at how I wish to design the app. I don't really have a good eye for design so I am going to ask others for opinions but the general consensus, against my own preference, is to make the app light themed rather than dark. Most of the people in the field that are older say they find dark themes very difficult to use so I need to keep this in mind. Dark theme is just more comfortable for me where light theme is almost a necessity for them.

Even if I do end up swapping to android studio the effort with my login and my requirement specification will not be in vain. The requirement specification at least gave me an idea of the structure to my application and how it will all connect together rather than just the jumbled mess I had pictured in my head. Also, the login will at least give me an idea of how to hash passwords, retrieve and store all the information. I hope to at least have a login done before the midpoint presentation and a few basic pages with some navigation but as of now figuring out databases is proving to be very difficult.

Gary Lynch – x17140382 (student signature and number)

Reflective Journal – December

This month I decided to tackle my idea. Previously I had been working on the idea as a web application over the end of November but I found it was inefficient due to problems and swapped to making a phone app. I got the login and register done using a firebase database. I also made a home screen and got the basic rooms page done and I am quite pleased with my progress.

After the mid-point presentation, I received feedback that my idea seems to be a little basic and that it will eventually be incorporated here so I am spending this month doing research as I am now doubting my idea as a whole from the feedback. In the presentation I was told to look at other ideas and try to incorporate more features that each of them has. With this in mind I began to look at similar apps. A lot of apps were solely based on the billing process so I may need to look into that. I would have to look into how to take payments and such but it seems like a good idea to make the payment easier in terms of looking at charging people and taking care of tips and split bills in the matter. Another application looks at the amount of each product you use which seemed like an amazing idea. If I could save stats of that and allow the kitchen staff to pull that up it would help drastically with stock management. These

are two features that I think would help push my idea to the next level but I need to meet with my co-ordinator and ask him about these and how feasible these ideas are. These features do not seem to be a problem like I thought earlier with bloating my idea and when I started implementing features I realised how little it was in practicality.

Gary Lynch – x17140382 (student signature and number)

Reflective Journal – January

For this month I have begun working slowly on my idea. I have settled on adding payments to the application as well as attempting to manage stock. My idea for stock is to either automatically update it or to have it where we have a complete sales history and we can just review this to see the count. I am unsure as to how I can tackle the stock or the payments but for now I will focus on the core idea I had at first. I have been able to make a tables page, I have reworked the login to try and include a Facebook API. I will try to tie that into the application as it will make signing up easier. Overall, I have to work more on the architecture of the application as it is confusing how to handle it. I have begun looking at assets for my menu and I'm wondering how to drag them in. I have found a few tutorials on how to add in a menu for a fast food service and I can maybe rework these for my application. The reason these are somewhat awkward to use is because they are made for deliveries whereas my app is used in a restaurant to attend to customers. If I can find a way to use these to my advantage it would be a great help. For the next month I will look at creating my menu as I have focused on reworking the login somewhat this month.

Gary Lynch – x17140382 (student signature and number)

Reflective Journal – February

For February I worked on the menu. I have managed to pull in the menu using an XML created online, it turns out I could use the menu part of the tutorials I have found. The menu is very basic and I am unsure how to even extract variables from them as of now but the menus are there and they look nice which is what matters. The login for facebook is extremely confusing and I'm beginning to get more confused as to how to tackle the orders

for each table, as it stands I need to be able to have multiple orders active at the same time and they need to be assigned under the table. I thought I would be able to assign the variables to each table but it appears to be more difficult than that. As this is my first time attempting to use firebase with android studio this is proving to be more of a challenge than I initially anticipated so I need to work to figure that out. After meeting with my supervisor, I decided that despite my initially concerns I do need to attempt to add in payments. I am unsure as to how this will work with my app as it is made for the waiting staff and not the customers so I need to think of a way to tackle this too. I am looking into stripe at the moment. Overall, I think I have a plan for next month, I need to attempt to look more into how I will handle payments. I do however have most of my assignments due next month so I need to create time for this as well.

Gary Lynch – x17140382 (student signature and number)

Reflective Journal – March

Didn't get a lot done this month as I had most of my other assignments due at this time so I used the time to focus on them. I had a meeting with my supervisor to discuss the progress I had made and it seemed positive, I seem to be going in the right direction for the project but it is hard going. This month I looked into how to organize the cart and whether I should create a giant database to process all transactions or separate them into smaller databases which are not only more manageable but more readable. Either way I find that they will be clustered either in number of tables or the size of the tables. I have to look at which would be more efficient for the machine and which would slow it down. I also have to look at which could be useful in the end as this will be hard to change once I begin to do it. I also need to begin to think how I am going to incorporate the kitchen into this. I need to think of whether they will have an active or passive role in this application, where they will be able to interact with the app at all or whether we let that update automatically. The real-world example is that they have 2 tickets, one for the wait staff and one for the kitchen so that they both can read the tickets at the same time, perhaps having a database for each side will be a way of doing things or maybe have the database automatically update the kitchen side when an order

is confirmed. For next month I will try to have a better understanding of my database architecture.

Gary Lynch – x17140382 (student signature and number)

Reflective Journal – April

For April I didn't get much at all done on my project. With the virus situation happening I have lost my job so I had to try to figure out my personal situation among other personal problems encountered. Due to this I spoke with my supervisor and we decided it was probably best to defer until the end of summer to give myself time to adjust to the sudden changes that came with the virus. I have pushed back all of my classes so when I figure out how to deal with the situations at hand I will still have the same amount of work to deal with as I did prior to the virus. I will try to allocate time into the project when I get the chance but for now I will try to improve my current situation.

Gary Lynch – x17140382 (student signature and number)

Reflective Journal – May

Very little work done this month as I have encountered another problem, my pc is not working and due to losing my job I have no means to fix it at the moment. I am unsure as to how I can amend this situation but I am talking to the college about the laptop grant to maybe assist me or looking to get a loan of a laptop off of a friend or relative. It's quite difficult but I will still use the time to try and work out the database architecture as it is an integral part of the application. With this in mind I will be able to at least visualize how I plan to go moving forward and through this I will be able to continue progressing when I improve the situation.

Prior to my computer breaking I used the time to work a little on just increasing the api and pulling a more diverse menu through it. With this in mind I think having a complicated menu

will make things more confusing so I need to look at a more specified menu specializing in a single food, maybe pizzas or burgers.

Gary Lynch – x17140382 (student signature and number)

Reflective Journal – June

I have a working computer again as my friend has generously donated their laptop to me in their spare time but it is very slow at running android studio and makes it extremely difficult to test the code in a timely fashion. Each launch of the application can take between 2-5 minutes to test even minor changes, even colour changes which is a huge let down but I am trying to make the best of the situation and I am grateful that I have a laptop that I can use in the meantime. I have also hit a huge roadblock with my project, the Facebook login I was using has ceased support. This means that my entire login system is not working anymore and anything I had tied to this is also causing problems, specifically the tables and orders I had tied to users. This is a huge setback and I am unsure how to deal with this as of now. I know that firebase has its own login and I feel a bit down on myself for trying to spread my work and using the Facebook login method over the firebase method now but it is completely demoralizing to see my code break like this, I am unsure how to amend this at the moment and I feel that I need to redo almost my entire project at this point because it seems to be falling apart. I will try to have these problems fixed by the end of July but this is a huge setback.

Gary Lynch – x17140382 (student signature and number)

Reflective Journal – July

I have spoken to my tutor about this and explained my situation and have decided to defer the deadline of my final project until December, I will have to continue to work on my other projects and complete them in August but for just the software project I have extended the deadline until the end of the year. I find it extremely difficult to work at home as I share my house with 6 other people and don't have the best internet connection so it's difficult without the resources of the college at my disposal and also the workspace. I have also decided that I am going to have to scrap the android studio project and start entirely from scratch in a different way as the laptop is just too frustrating to work on. Trying to test and fix problems because a huge pain when I have to wait so long in between launches of the application, on

top of this with the login stopping support this means that my project is going to have to be redone in one way so I feel the best option is to just use this as a way to try steer away from android studio and approach a lighter framework as android studio is very intensive on the computers processing power. With all of this in mind I don't think I will be doing much next month as every single one of my other classes final projects are due in August and I feel with the extended deadline of this I need to give the other classes my full attention.

Gary Lynch – x17140382 (student signature and number)

Reflective Journal – August

As I figured not much was done this month as I had all of my other deadlines due and honestly became extremely burnt out on coding, I also have to try help support my family through the coronavirus crisis but the college seem to be very supportive and understanding with this in regards to referring me to the counselling and student assistance fund. With other classes I received support and it helped me through them so I at least feel a lot more confident with this project and a lot less disheartened than I did before I feel more motivated to get restarted on the project than I did initially and with this in mind I will attempt to get a start on the project again next month, I'm going to look into what application to use to program in and what language to use. I initially thought of using Java as it is the language I am the most familiar with but the frameworks for java are very limited for an application like this so I will look into using javascript maybe as it has good framework support. I need to research what would be best to use and what IDE to use, where would be best to host it for example before I get started because twice I had to restart my project due to things dropping support so I think this time I will finally look into hosting locally, months ago with cloud9 I relied on the hosting of online but due to them requiring payments it is difficult, especially due to the financial situation I am in. But I will research hard before diving into this application this time.

Gary Lynch – x17140382 (student signature and number)

Reflective Journal – September

So not a lot of coding was done this month, I researched a lot about localhosting and it is nowhere near as scary as it initially seemed. I am going to use PhP as I feel that I will also be using JavaScript for functions at some point. PhP seems like a good option as I will be

able to find tutorials easier due to the popularity of the language. I have also settled on the framework I will use and will be using bootstrap. This will make the design part easier which I have always lacked in due to not having the best eye for design. This will give me more time to focus on the database, which for this I have chosen to use SQL. I looked into reusing XML and importing again but upon my research people said that SQL is better as it is less slow and overall better supported than XML, XML is considered more "limited" than SQL so I find that I will be best off using this. It also has the support of phpmyadmin which will be better at supporting the databases and allowing me to see the updates to and from the databases in real time. Overall, I feel a lot happier about doing the web app over the android app as the android app was bulky and limited to just android phones, which I do not have, so I will be able to test my application easier on my computer and on any smartphone or tablet.

Gary Lynch – x17140382 (student signature and number)

Reflective Journal – October

For this month I focused on getting the localhost setup as well as the databases. It was way more confusing than I initially realized and I kept having a problem with xampp where the SQL would not start no matter what I tried. It turns out that if I clicked the start button too fast on my initial launch of xampp for the Apache and MySQL it would cause an error that could only be terminated in the task manager, this took a long time to figure out, longer than I hoped really. But I am glad that I at least know how to host the server now. I also looked into bootstrap templates and found one for restaurants and bars that I really like. It is somewhat bulky and has a lot of useless flavor features that I will probably remove to keep the app more concise but it is a good start for now. I have also began creating my SQL databases. It is a huge change from firebase which was extremely simple to setup in comparison to this as firebase allowed you to just drag in anything you wanted but this one seems somewhat easier to manage. I have begun creating my login and signup pages that for some reason just will not connect to the database. I am inexperienced really with SQL as I have only done very basic SQL statements in the past in my PLC course which we wrote into a word document for an understanding and most of our databases were done in Microsoft Access. The only other exposure I had to this was in third year of NCI but for that project it was a team members job to handle the login and signup.

Gary Lynch – x17140382 (student signature and number)

Reflective Journal – November

I got quite a bit done this month. I got my login and signup working, it turns out I misspelled my databases name the entire time and didn't understand how to run SQL statements either. But with this in mind I was able to get a good barebones of the website working. I have created a menu and worked hard on making it add and delete from the database on button presses. This involved learning about html forms and how to use PhP in tandem with HTML. For this it seems I have to declare php tags all the time which gets confusing very fast. I am trying to figure out how to get PhP variables to transfer between pages and I also need to figure out how to divide the database so that if multiple people are ordering at once all of their orders will not show at the same time. The architecture of my database is probably the most important part of this and I need to draw out a plan correctly before I really begin doing it. For now, I am just trying to learn the basics of the language and how to get basic functions to work, once I figure this out I can begin to change the website around and add in the functions in a more complex way. With how little time I have I am considering scrapping the idea of stock and payments seems impossible to do so I am unsure as to what I should do in this regard. For now, I have created the menu page and I need to just create the rest of the menu, I have settled on making it just burgers as this will be a good test of concept for the website. For next month I need to work hard on it.

Gary Lynch – x17140382 (student signature and number)

Reflective Journal – December

So, crunch time has ended and I am delighted with how things worked out. I found a lot of problems with my architecture multiple times and even with my plans I didn't foresee some of the problems that could occur. I didn't understand how I could delete orders for certain tables or how to update orders on the go until the end of the month. I also spoke to a tutor about my idea, what I had done and what I thought I could get done. I was told that I really needed to focus on trying to get payments done as this is a new feature for the application which would showcase a new feature when I was prioritizing trying to get stock done which would be another example of reading from the database and updating the database.

So, I got payments working. For the application I figured out how global variables worked. And allowed the application to create the order throughout all of the pages. My tutor told me that the way I had created my menu was somewhat bad as I had hardcoded the values into it

so I reworked it similar to the original way of pulling the values from the database and creating the menus through this which is way better, it means any menu can be swapped in for my current one. I created pictures for my menus myself. I made it create the tables on startup to make it easier for the user as they will not need to create any of the tables. I figured out how to update all of my databases correctly I made multiple databases for multiple situations all that create, read, update and delete in various situations. Each time a table is selected it creates a main, side and drink table named after that order that can be used for data warehousing. This means that if in the future I wanted to expand on the app I could track the sales for each table easier. I created a table that stores all sales that happened, I created the cart that keeps all orders and displays all of the current orders for each table, tied to the table number. I also created a table for the kitchen to view that they can see all incoming orders, in order, and delete from it freely without it affecting the wait staff. Also, the payments automatically clear the database on successful transaction so in the end I am overall very happy with my application. I feel that with the circumstances of covid I have done quite well and I am proud of the work that I have put in. I have learned lot about code, PhP and how databases worked which I was always very worried about as my only exposure to this had been in my PLC course. I also achieved all of the original features I planned to get in and I have expanded to add the payments that my supervisor wanted originally. The only feature I didn't manage to get in really was stock but I feel that the database has the work there I just needed the SQL statements to pull and update accordingly.

Gary Lynch – x17140382 (student signature and number)

11 Showcase Poster