COMP 3613

Software Engineering 2

Restaurant Ordering Application

R.O.A

Group Name: 

Group Member:

Matthew Cyrus - 816000799

Jada Gooding - 816001706

Lydelle Corraspe - 816002983

Laurel Jackson - 816004212

Dalton Brown - 816117716

Table of Contents

[1. Project Specifications: 3](#_Toc25870888)

[1.1 Problem Description: 3](#_Toc25870889)

[1.2 Project Description: 3](#_Toc25870890)

[1.3 Goal(s) of the System: 4](#_Toc25870891)

[1.4 Stakeholders and Target Users: 4](#_Toc25870892)

[2. Requirements Determination: 4](#_Toc25870893)

[2.1 User Requirements: 4](#_Toc25870894)

[2.2 System Requirements: 4](#_Toc25870895)

[2.3 Functional Requirements 5](#_Toc25870896)

[2.4 Non-Functional Requirements 6](#_Toc25870897)

[3. UML Design Diagrams: 7](#_Toc25870898)

[3.1 UML Use Case Diagram: 7](#_Toc25870899)

[3.2 Activity Diagram: 8](#_Toc25870900)

[3. Architecture 10](#_Toc25870901)

[3.2. Architectural Design Pattern 10](#_Toc25870902)

[3.3. Architecture Diagram 10](#_Toc25870903)

[4. Software Design 11](#_Toc25870904)

[4.2. Entity Relations Diagram 11](#_Toc25870905)

[4.3. Class diagram 12](#_Toc25870906)

[4.4. Proposed Technology 13](#_Toc25870907)

[4.4.1. Limitations of Technology 13](#_Toc25870908)

[5. Methodology 14](#_Toc25870909)

[6. Low-fidelity Prototype 15](#_Toc25870910)

[7. Implementation 16](#_Toc25870911)

[7.1 Product Backlog 16](#_Toc25870912)

[7.2 Roles 17](#_Toc25870913)

[7.3 Sprint Backlog Summary 17](#_Toc25870914)

[7.4 Sprint 1 19](#_Toc25870915)

[7.4.1 Change Report 19](#_Toc25870916)

[7.4.2 Conflicts and Solutions 19](#_Toc25870917)

[7.4.3 User Testing 20](#_Toc25870918)

[7.4.4 Review 20](#_Toc25870919)

[7.5 Sprint 2 20](#_Toc25870920)

[7.5.1 Change Report 21](#_Toc25870921)

[7.5.2 Conflicts and Solutions 21](#_Toc25870922)

[7.5.3 Integration Testing 21](#_Toc25870923)

[7.5.4 Review 21](#_Toc25870924)

[7.6 Sprint 3 22](#_Toc25870925)

[7.6.1 Change Report 22](#_Toc25870926)

[7.6.2 Conflict and solutions 22](#_Toc25870927)

[7.6.3 Defect testing & debugging 22](#_Toc25870928)

[7.6.4 Review 23](#_Toc25870929)

[7.7 Sprint 4 23](#_Toc25870930)

[7.7.1 Change Report 23](#_Toc25870931)

[7.7.2 Conflict and solutions 23](#_Toc25870932)

[7.7.3 Defect testing & debugging (performed again) 23](#_Toc25870933)

[7.7.4 Review 23](#_Toc25870934)

[8. Risks 24](#_Toc25870935)

[Appendix 25](#_Toc25870936)

[Sprint Backlog Cycle #1 25](#_Toc25870937)

[Sprint Backlog Cycle #2 26](#_Toc25870938)

[Sprint Backlog Cycle #3 27](#_Toc25870939)

Table of Figures

[Figure 1 Use Case Diagram 6](#_Toc25156679)

[Figure 2 Activity Diagram 1 7](#_Toc25156680)

[Figure 3 Activity Diagram 2 8](#_Toc25156681)

[Figure 4 Software Design 9](file:///C:\Users\816001706\Desktop\M4.docx#_Toc25156682)

[Figure 5 Entity Relational Diagram 10](#_Toc25156683)

[Figure 6 Class Diagram 11](file:///C:\Users\816001706\Desktop\M4.docx#_Toc25156684)

[Figure 7 Main Screen (Home) & Figure 8 Menu items 14](#_Toc25156685)

[Figure 9 Bill Details & Figure 10 Order Confirmation 14](#_Toc25156686)

[Figure 11 Burn down Chart from all sprints 17](#_Toc25156687)

Restaurant Ordering Application

# Project Specifications:

## Problem Description:

When Customers go out to eat, they expect to have tasty food as well as good service. Most servers try to do their best and do a decent job, but customer complaints might be justified if the wait staff is rude or does a poor job. In this case, Customers are justified in bringing their complaints to the manager and the manager would have to deal with the situation.

In addition, waiter staff is constantly checking up on guest in case they need their services. Currently, in the local market, they isn’t a mechanism is a place for servers to know if a table requires their assistants without disturbing the guest or having the customer signal to the waiter for their services.

Another problem that was observed is that many restaurants store all of their data manually. As a result, these businesses have large volumes of information and it becomes a challenge to maintain, store the records accurately. It takes a lot of time to enter these large volumes of information and there are chances of mistakes.

## Project Description:

The Restaurant Ordering Application (ROA) aims to replace the conventional menu ordering system which is time-consuming and old fashioned. Meal orders can be taken on an Tablet, which will have the complete menu offered by the restaurant. Each menu item, under any of the categories, comes with a description of the dish as well as ratings given by previous customers, a large image for better knowledge of the dish and the cost of the item.

When the Customer places an order it would go directly to the kitchen to be prepared. Customers would also be able to call the waiter with the touch of a button as well as to view their outstanding bill. When the Restaurant is out of a particular stock, those items will be removed from across all devices by the restaurant and won't be visible to the customer.

## Goal(s) of the System:

We have designed a solution that can make a difference in restaurants with the main goal to maintain the restaurant's functions in an effective and accurate manner as well as the day to day food records in the system. We hope that this system would impact the restaurant owners in an effective and change the experience of the customer.

## Stakeholders and Target Users:

The team identified Restaurants owners and or chains as our main stakeholders; with that being said, visitors to these establishments are our primary user population, while the waiter and kitchen staff are our secondary users (of the application).

# Requirements Determination:

## User Requirements:

UR1: users should require minimum knowledge to operate the system

UR2: users should be able to clearly read items

UR3: users should be able to clearly identify menu options based on the pictures provided

UR4: users should be able to call a waiter when it is needed.

UR5: users should be able to see how long their order will be

UR6: users should be able to rely on the application

UR7: the user interface should be menu-driven

## System Requirements:

SR1: The system should be able to show the customers the food options that are available at the restaurant.

SR2: The system should be able to remove a particular menu item when it is out of stock.

SR3: The system should be able to notify a waiter when there needed at their table.

SR4: The system should allow the customer to view the image, ratings, cost of a particular dish before the customer orders that item.

SR4: The system should send the order to the kitchen staff or bar staff.

SR5: The system should allow kitchen staff to update the order to ready and then notify the waiter when the order is ready.

SR6: The system should allow customers to click “view Bill” and the system would show the updated current standing of their bill.

SR7: The System should allow kitchen staff to reopen and cancel orders if required.

SR8: The system should allow the cashier to login to the system and accept Payments.

SR9: The system would show a list of categories for the user to select from.

## Functional Requirements

FR1: A user should be able to order a meal or meals from the menu.

FR2: A user should be able to navigate through the restaurant’s menu.

FR3: A user should be able to review their order before confirmation

FR4: A user should be able to remove a meal or meals on their order

FR5: A user should be able to request the service of a waiter

FR6: A user should be able to view their bill

## Non-Functional Requirements

NFR1: Usability - The Restaurant Ordering system should be user-friendly even for novice users and easy to learn.

NFR2: Real-time Updates: The system should be updated after a menu record is added or if a menu item is no longer available

NFR3: Time Constraints: The system should be able to respond to user interactions within 2 seconds.

NFR4: Fault-tolerant: The system should continue its normal operations despite the presence of system or hardware faults

NFR5: Scalability: The system should continue to function well when a change has occurred in size or volume in order to meet the user’s needs or demands.

# UML Design Diagrams:

## UML Use Case Diagram:

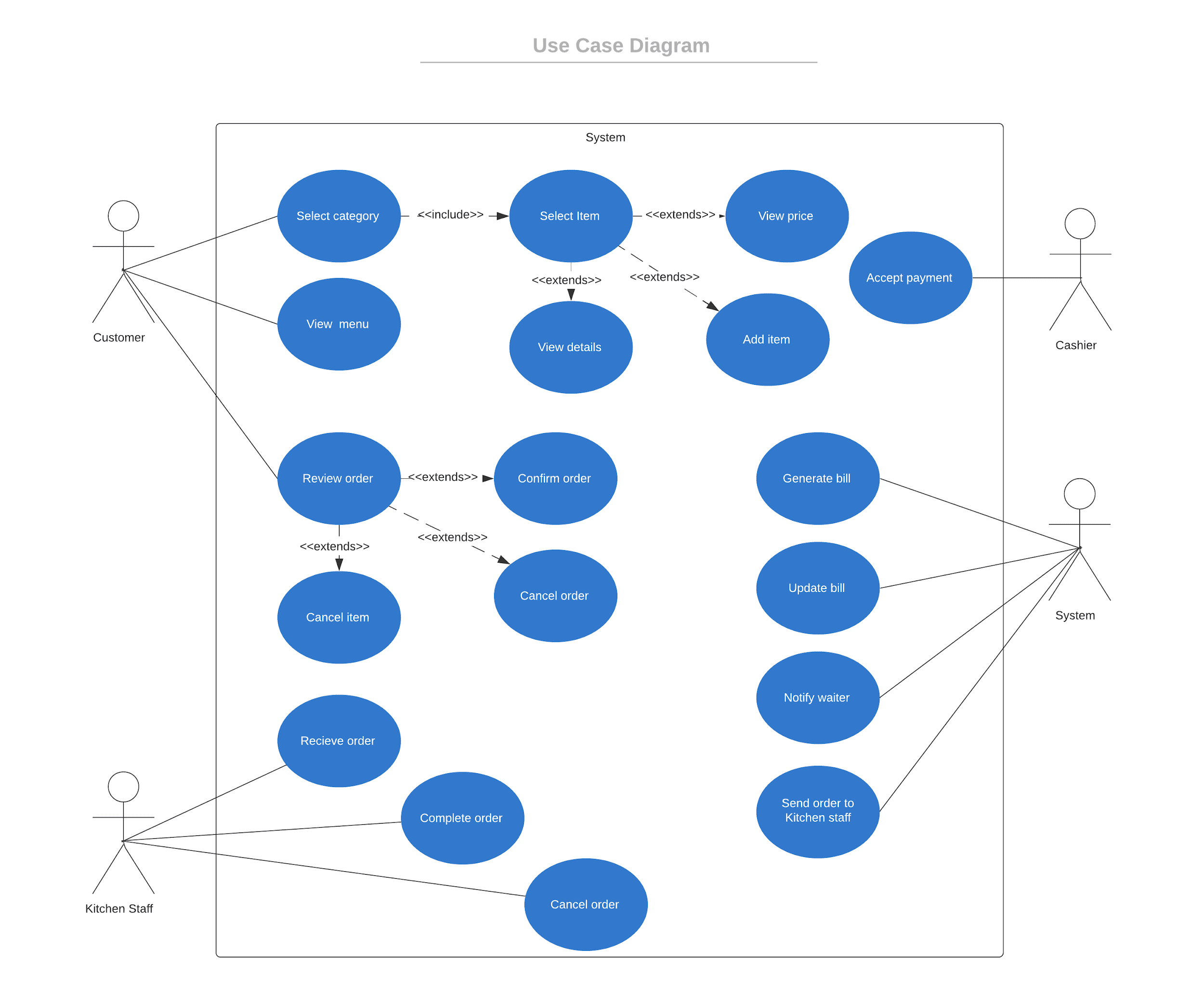


Figure 1 Use Case Diagram

## Activity Diagram:

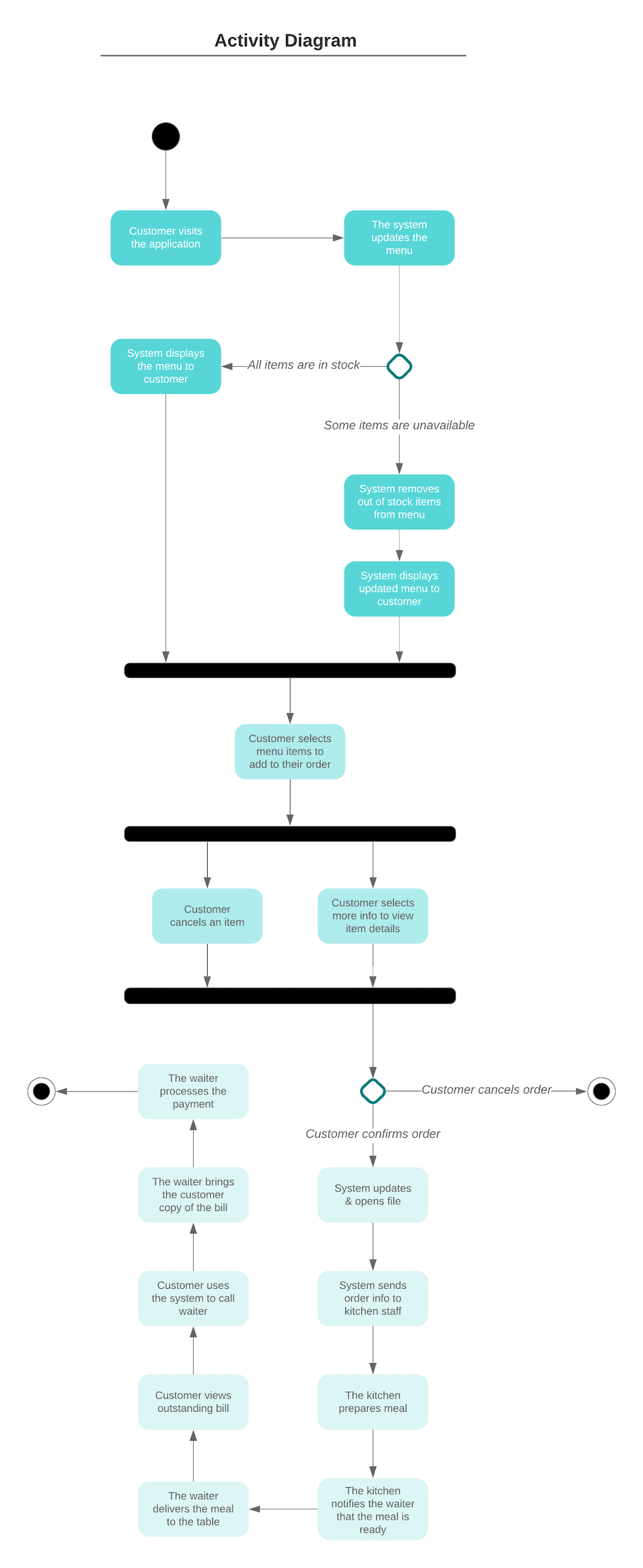


Figure 2 Activity Diagram 1

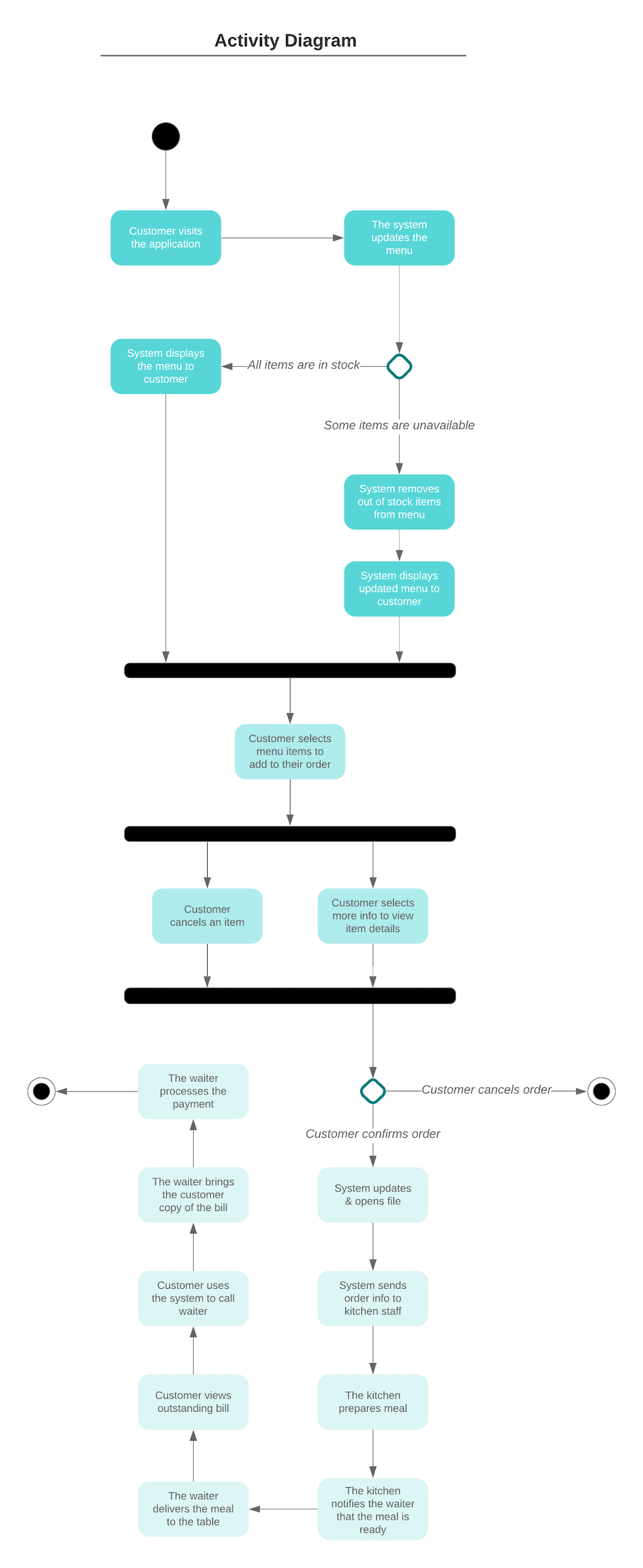


Figure 3 Activity Diagram 2

# Architecture

## Architectural Design Pattern

In object-oriented software development, design patterns are solutions to general problems plaguing many developers. The design patterns ensure the best practices are used. The patterns in this project were used to ensure maintainability, extensibility and ease in refactoring code within ROA.

The Architectural design pattern which best suits the ROA is the Client-server pattern. We chose this pattern as it is used when data in a shared database has to be accessed from a range of locations, such as a chain of restaurants having multiple locations, because servers can be replicated, and it may also be used when the load on a system is variable.

The principal advantage of this model is that servers can be distributed across a network. General functionality can be available to all clients and does not need to be implemented by all services.

## Architecture Diagram

Figure 1. Shows how the cloud server and mobile application interacts with the other components of the system.

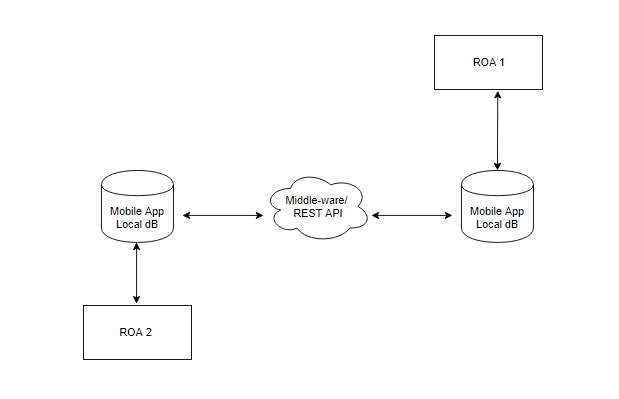
* The mobile application inserts, updates, and queries the database located on the device’s local storage.
* The middleware acts as a connector between the application server and the mobile application database. It allows the systems to communicate with each other and share information. (This acts as a backup mechanism)

Figure 4 Software Design

# Software Design

## Entity Relations Diagram

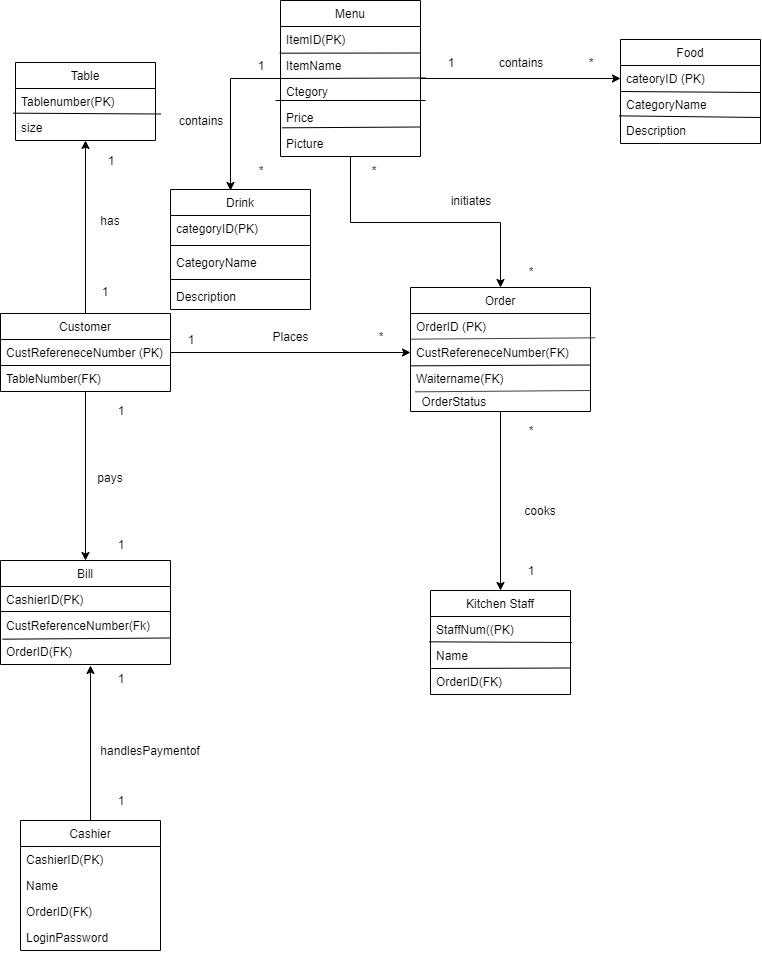


Figure 5 Entity Relational Diagram

## Class diagram

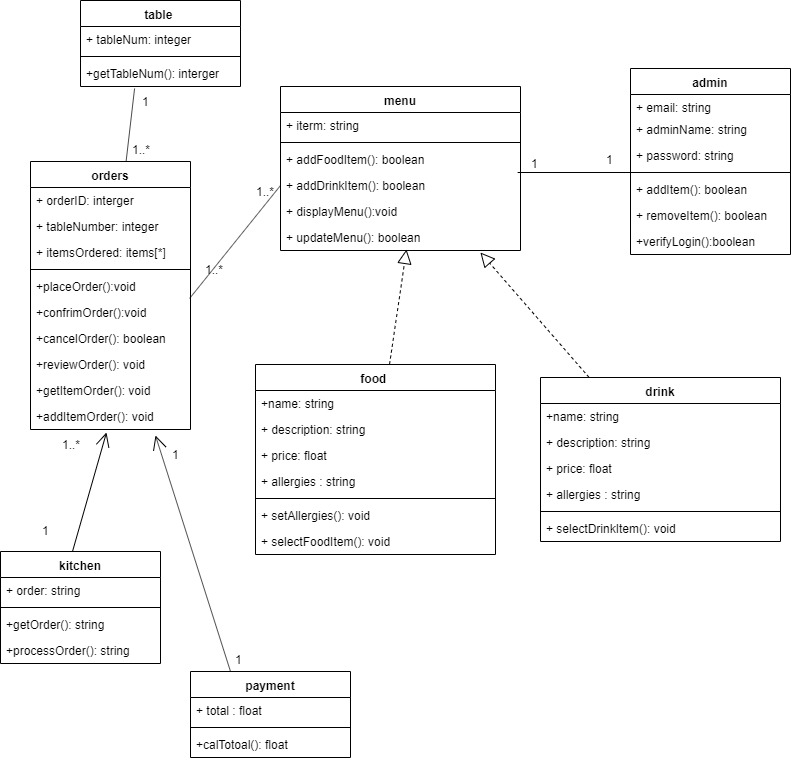


Figure 6 Class Diagram

## Proposed Technology

The Android Operating System is an open-source software development platform developed by Google and Open Handset Alliance. It is based on the Linux kernel and developed for smartphones and tablets with touchscreens.

The Mobile Application will be developed in Java using the official IDE for Android application development, Android Studio v3.5.1 where the application’s minimum SDK version is 21 and the target SDK version is 28. Building a native application will allow forward-compatibility since changes to the framework’s API are additive. This ensures ROA will be compatible with later versions of Android Platform and higher API levels.

ROA is a standalone client-side application. This is enabled by the mobile device’s local SQLite database. SQLite database facilitates the same functionality of the REST API to perform GET, POST, PUT, and DELETE operations, without causing a heavy dependency on the server. Thus, users are not forced to be connected to the internet in order to use the application.

### Limitations of Technology

The application only works on devices with the Android Operating System. Reports based on the performance of users via certain demographics are unavailable. Lastly, the application does not spill over into the purchasing department to assist logistics.

# Methodology

This section of the project will provide a detailed explanation of the methodology used to plan, analyze, develop and implement the project objectives listed in Milestone 2. The methodology was based on the SCRUM approach that included a product owner (that remains constant throughout the entire project), a SCRUM master (interchanges throughout each sprint) and a development team. Taking up the SCRUM method of project management, we would also be working closely with the product owner to ensure that all software and hardware requirements are met.

For the group members of 3G2B, we researched on software that could solve a business problem and concluded to build the Restaurant Ordering Application (ROA) that serves to replace the conventional menu ordering system. Each member of the 3G2B group researched on how the ROA system can be implemented into restaurants rather than the traditional waiter method.

For each milestone given, tasks were given to group members based on the best knowledge of the topic at hand. A buddy system was also introduced, where each group member checked on another member to ensure that they understood the given task and that progress was going at an optimum speed.

The development of ROA will be developed on either Android Studio or Flutter according to the programming skills and knowledge of our code developers.

# Low-fidelity Prototype

The development team produced a low fidelity prototype based on the requirements specified.

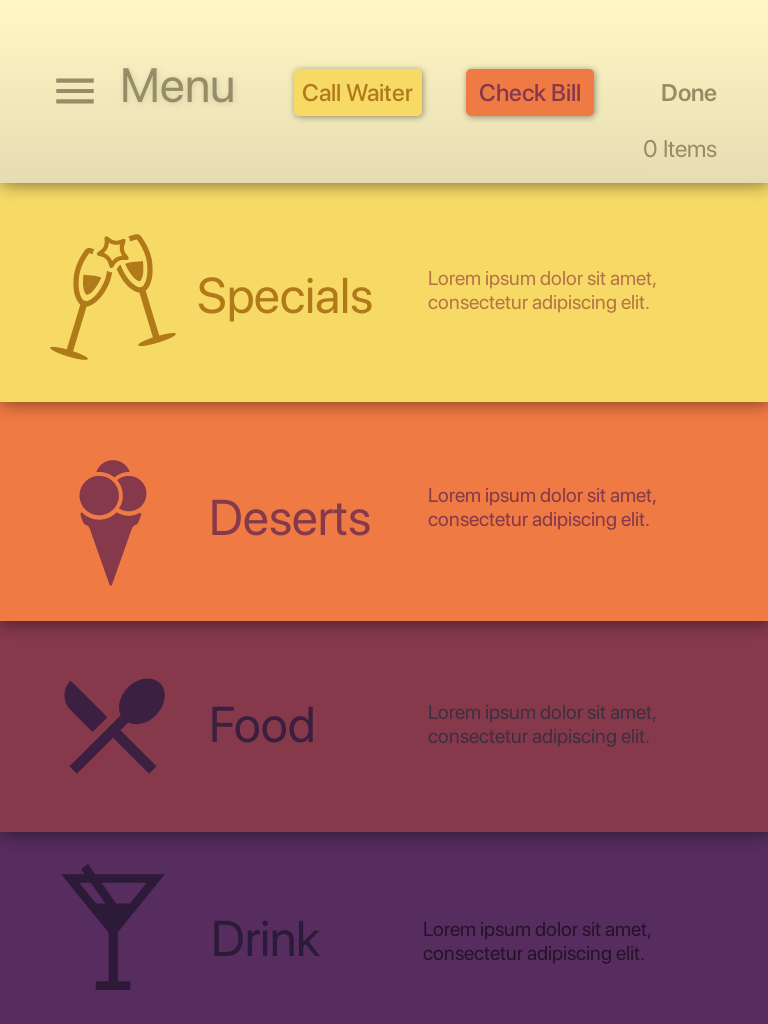
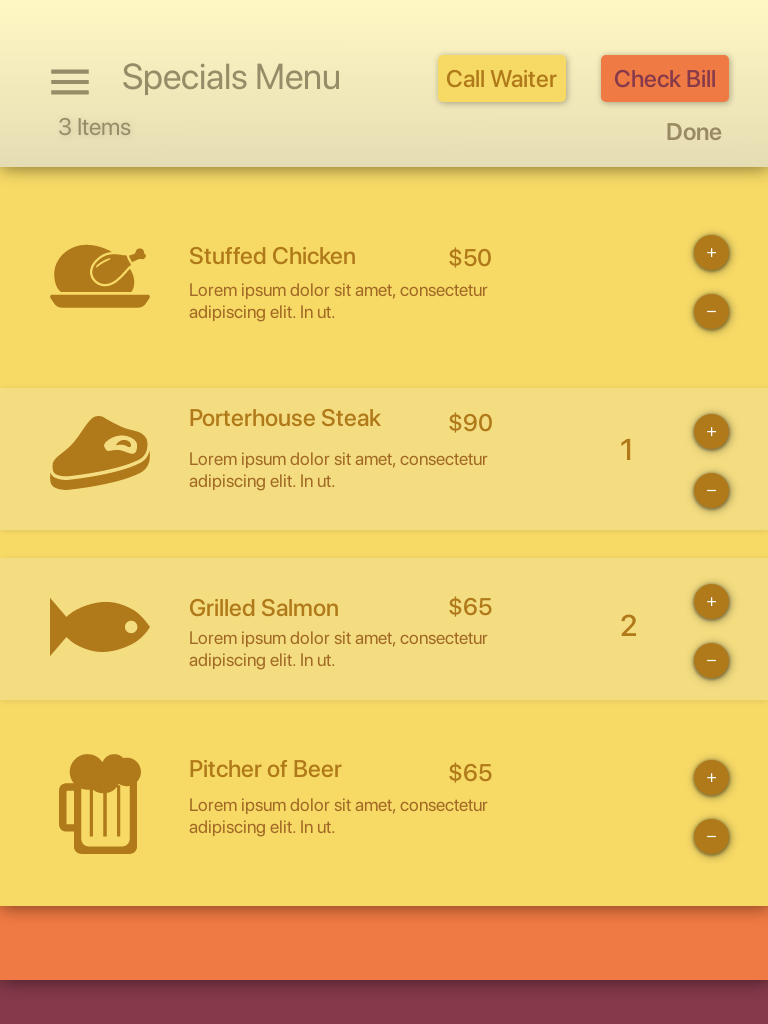
 

Figure 7 Main Screen (Home) Figure 8 Menu items

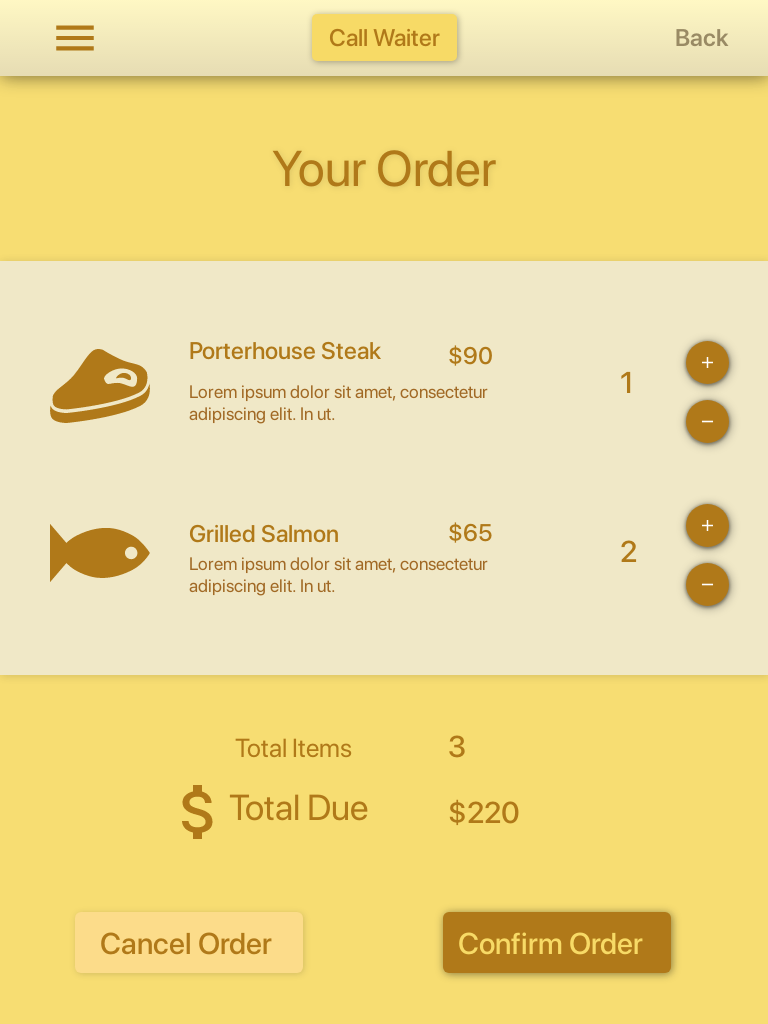
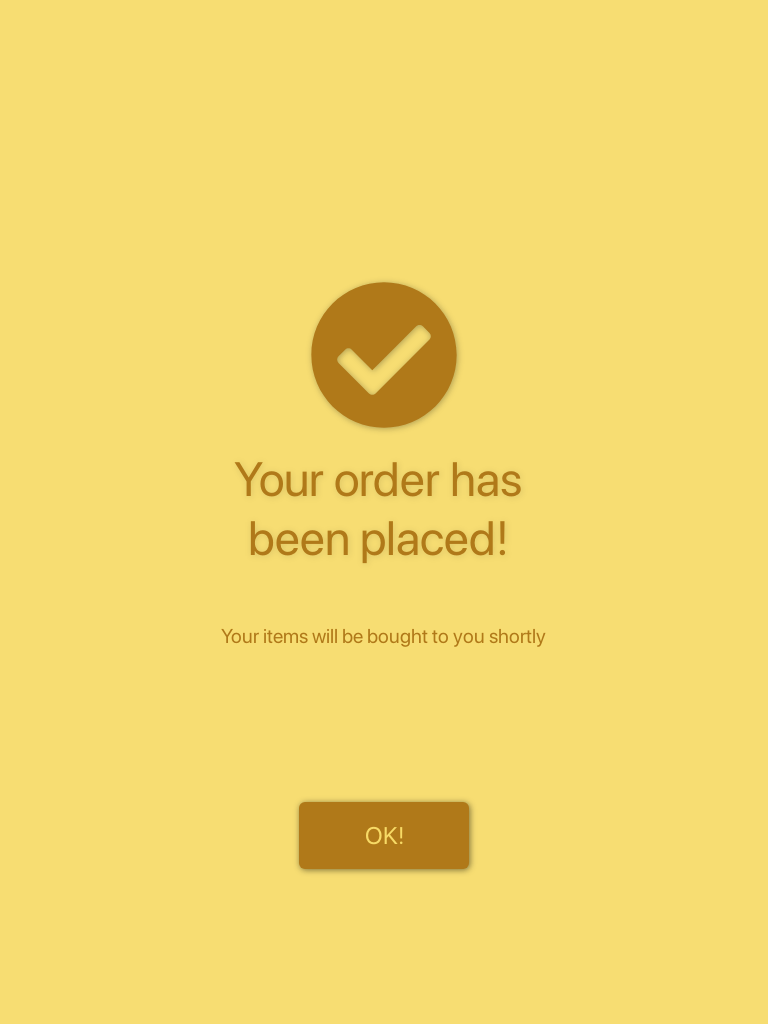
 

Figure 9 Bill Details Figure 10 Order Confirmation

# Implementation

Scrum Repo can be viewed [here.](https://github.com/jada-gooding/SWE2-32GB.git)

## 7.1 Product Backlog

The product owner constructed the product backlog from the requirements previously elicited. These requirements were converted into functional tasks which were categorized based on their importance of the features to the application’s functionality. From these, the sprint backlogs were to be composed. The SCRUM guide was consulted so as to ensure that the product backlog creation process was in accordance with the SCRUM methodology. The following is the product backlog as at the beginning of the first sprint:

1. Construct the UI to view the menu items:

The laying of the project foundation, the planning of the model classes to give us an idea of how information will be stored and handled in our application.

2.1 Adding and or deleting menu items from the dB:

Creating a database and its controller based off the models and schema developed, adding and creating views to interact with our users and get user input (via tap) to perform the necessary update and store the required data.

2.2 Load Information from database:

Loading the data from the database and displaying it into the respective views.

2.3 Display the user’s Data:

Ensuring that there is seamless communication between views and controllers and controllers, models and database.

1. Call waiter:

Creating a view to interact with our users based on a timing mechanism via a random number generator.

1. Improve upon the Menus’ Sections UI:

Improving user experience and intuitiveness.

## 7.2 Roles

The following are the assignment of roles for the three sprints that were conducted. The scrum guide was also used to ensure that each member knew what their role entails as well as what is expected of them.

#### First Sprint

Product Owner: Matthew Cyrus

Scrum Master: Laurel Jackson

Development Team: Lydelle Corraspe, Jada Gooding, and Dalton Brown

#### Second Sprint

Product Owner: Matthew Cyrus

Scrum Master: Jada Gooding

Development Team: Laurel Jackson, Lydelle Corraspe, and Dalton Brown

#### Third Sprint

Product Owner: Matthew Cyrus

Scrum Master: Lydelle Corraspe

Development Team: Laurel Jackson, Jada Gooding, and Dalton Brown

#### Forth Sprint

Product Owner: Matthew Cyrus

Scrum Master: Dalton Brown

Development Team: Laurel Jackson, Jada Gooding, Lydelle Corraspe

## 7.3 Sprint Backlog Summary

#### Sprint Backlog 1

In this first sprint, the team implemented the shell of the application. The team focused on constructing the navigational path(s) an intended user would utilize; whilst adding the item details for the various sections of the menu such as, the price, the item description and a photo thus, giving the user all the information they are seeking on a particular item.

#### Sprint Backlog 2

The team continued using Scrumy.com in the second sprint. The team focused on implementing one major feature, which was the adding or deleting items from the order. The time frame for this sprint was initially 4 days based on our scrum schedule developed at our teams’ initial scrum planning meeting; however, the goal was not met, thus an extension of 3 days was given by the scrum master.

#### Sprint Backlog 3

The team focused on implementing the last major feature, which was the calling the waiter to the table. The team also discussed finalizing the application layout and design at our scrum meeting. The time frame for this sprint was initially 2 days; however, an impromptu meeting was also held and an extension of 1 day was added on to the schedule by the scrum master.

#### Sprint Backlog 4

The team focused on updating the user experience by updating the layout and design of the application as previously discussed during sprint 3. The scrum master decided to implement human computer interaction concepts to the application. Therefore, guiding the development team accordingly on colour scheme, placement of buttons, and labelling to name a few. The team also revised the call waiter button functionality by reducing the timer length to a shorter duration.

You can view the progresses of each sprint cycle [here](https://myuwi-my.sharepoint.com/:x:/g/personal/jada_gooding_my_uwi_edu/ETGC-OoXt6dFp_-gPpINAN4BqcP-d_nboAaWYgq_yj9Q-A?e=VrhfO8).

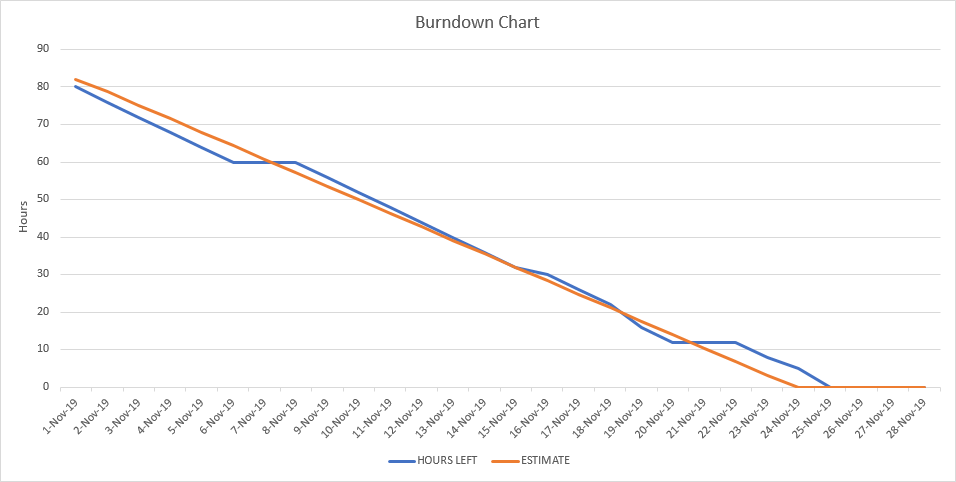


Figure 11 Burn down Chart from all sprints

## 7.4 Sprint 1

The first sprint began on the 1st of November and ended on the 5th of November. During this sprint only two “daily” scrum meeting times were decided upon, they were: Friday at 10am and Sunday at 6pm. During this sprint, the team covered the first item on the product backlog list.

The main task in the sprint was further broken down into smaller tasks and assigned to a team member. As seen on the scrum sheet, the scrum master prioritized the sub task based on a scale of high importance to low importance and also kept track of task progress to completion.

During the first sprint, which spanned 5 days, an initial sprint meeting, and two in person meetings were held. For the first half of the sprint, the tasks were completed in a timely manner and the scrum master notified the team that we were on track with our scrum schedule.

### 7.4.1 Change Report

No major changes were implemented at this stage of development. The team identified that the layout and design of the application needed to be improved on.

### 7.4.2 Conflicts and Solutions

Conflicts were inevitable, and the solutions were not always swift. During this sprint the major conflicts were working with android studio, and git.

As the first week progressed and everyone chose a task or two to handle, some members experienced blockades as the working code developed by on developer only worked on later versions of the software application. Solution for this problem included the development team utilizing their personal laptops only in order to further develop the application as the Lab facility provided ran an older version.

Git issues involved an error whilst pushing to git. Thus, our contingency plan was to check all files being committed to master before pushing. However, after removing files from the staging area and redoing the commit process ensuring the paths were correct, the non-fast forward error still appeared and was not solved as of the end of sprint 1. A google search also did not help eliminate this error.

### 7.4.3 User Testing

A cognitive walkthrough was conducted for the application with the current low fidelity design at that time. The product owner carried out our user tests. There was an incentive to this approach, as it allowed for a good relationship, and speedy response from the tester. This was because the user would feel more compelled to support the application.

From this test changes were made to the proposed software interface. The team rearranged the main application to utilize a grid layout using image buttons instead of a list view. Thus, striving for smooth use of the user interfaces.

### 7.4.4 Review

There was a product backlog change where the priority of the Tasks was changed as it was identified that a latte task carded to be completed in sprint 3 (View bill) should be merged with adding and deleing from the dB which is carded to be completed in Sprint 2.

## 7.5 Sprint 2

The second sprint took place from the 9th of November to the 15th of November. During this sprint an additional meeting time was added so that communication between team members can remain optimal. The new time added was Wednesdays at 1:00pm. Therefore, this made a total of three SCRUM meetings per week, however, some impromptu meetings were also held. The team only planned to implement two major features for the application, these features were adding and deleting items from the order and viewing the bill.

The tasks were completed relatively behind schedule, this was because of implementation issue between the application and SQL lite. The development team had to conduct research in order to eliminate the connection error between the database and our application. The scrum master took control of the situation and added 3 days to the scrum guide to allow for this set back. The development team succeed in utilizing the database thus, adding and deleting from one menu section was successful. As a result, implementing functionality of the dB for all sections was added to sprint 3 by the scrum master as the duration of sprint 2 came to an end. The development team also got to apply the alter quantity feature during this sprint.

### 7.5.1 Change Report

One major change as of sprint 2 was to swap firebase dB for SQL lite db. This changed occurred as it was best known to the development team. The sprint was originally carded to be completed on the 12nd of November; however, due to the SQL issue aforementioned, the sprint had to be extended by the scrum master to allow all of the tasks to be completed. The scrum master was careful to ensure that this extension was in accordance with SCRUM methodologies. Therefore, he consulted the SCRUM guide for confirmation. Although it is not advised, the decision to extend the sprint was made due to flexible scheduled created which allowed for unforeseen delays.

### 7.5.2 Conflicts and Solutions

The main conflict that arose during the sprint was where many of the functionalities that had to be tested were closely connected with the SQL Lite db. Therefore, the tests that were to be implemented on those features had to be delayed until a solution was devised. Thankfully, one of the developers made a breakthrough and the application functionality was up and running.

### 7.5.3 Integration Testing

Integration testing was conducted to ensure that the database connected smoothly to the application when a user selects an item to add to their order or if the user deletes an item from their order. The team was able to identify if the database responded with the updates. This was important as these updates are to be shown on the bill. Therefore, giving the development team the confidence knowing an item which appears on the bill supposed to be there and is not a rogue element.

### 7.5.4 Review

Although, we were behind schedule with sprint two we were still on track to meet our completion deadline. This is because our original scrum guide developed contained four scrum cycles. As a result of moving the main task from sprint 3 into sprint 2, we were left with playing room on the scheduled. This also enabled the team to complete he product backlog in three cycles instead of four cycles previously planned.

## 7.6 Sprint 3

The third sprint took place from the 16th of November to the 19th of November. During this sprint the team developed the last two remaining activities from sprint 2, in addition, to the new feature of allowing a user to call on a waiter via a tap of a button. The team discussed several ways to implement this feature at our carded scrum meeting.

The team agreed upon a pop-up message when the button is selected. This button displays a message to the user that a waiter is on their way to the table; this is done through the use of a timing mechanism (utilizing a random number generator between specified targets) developed by the team. And to increase the user experience, given the circumstance where the waiter hasn’t shown up at the table already, another message is shown when the timer reaches halfway.

The git error which occurred in sprint 1 was solved in this sprint cycle, which would explain why the project repository was not updated since the first sprint. The team utilized GitHub Desktop to perform the commits.

### 7.6.1 Change Report

No changes we implemented during this sprint.

### 7.6.2 Conflict and solutions

No major conflicts occurred during this sprint; the team agreed upon a way forward at our scrum meeting. Therefore, eliminating any questions or misunderstanding or misinterpretation the team may have had.

### 7.6.3 Defect testing & debugging

The team having developed a full functional prototype began to *break the system* to discover any bugs which were not previously known and ensured all navigational paths were intact. Any bugs found were resolved. User test were also carried out to ensure the database reflected the updates made by the user.

### 7.6.4 Review

The team agreed upon having a functional prototype rather than focusing on making the application look pretty, therefore a basic layout and design was applied and an improved version would be implemented for the final delivery.

## 7.7 Sprint 4

The fourth sprint took place from the 22nd of November to the 25th of November. During this sprint the team revised the design and layout of the application, to increase the user experience and usability of the application. The scrum master detailed a way forward at our scrum meeting to implement these changes based on concepts and principles from human computer interaction. As well as the team finalized the way each features was implemented, making only 1 change. (Discussed below)

### 7.7.1 Change Report

The team decided to adjust the call waiter functionality which was a minor change.

### 7.7.2 Conflict and solutions

No major conflicts occurred during this sprint.

### 7.7.3 Defect testing & debugging (performed again)

The team decided to perform defect testing and debugging one last time as to ensure all paths were satisfactory and the application behaved as expected as the upgrade in design occurred and we cannot assume the application was 100% bug free. During this session although it was not an error the team decided to adjust the call waiter functionality to a shorter duration.

### 7.7.4 Review

The team agreed upon having a functional prototype rather than focusing on making the application look pretty, therefore a basic layout and design was applied and an improved version would be implemented for the final delivery.

# Risks

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Risk** | **Probability** | **Effects** | **Type** | **Category** | **Strategies** |
| Requirements change by the product owner | Low | Serious | Project & product | Minimization | Implement extreme programming techniques. Therefore, project requirements will be left open for alterations if need be |
| Lack of conflict resolution within the team | Low | Catastrophic | Project | Avoidance | Maintain high level of communication within the team. |
| Wrong estimation and scheduling of product dev. | Moderate | Serious | Project & product | Avoidance | Keep members informed  about the details of each stage of development. |
| Productivity issues | Moderate | Catastrophic | Project | Avoidance | Constantly inform members of their roles and responsibilities. |
| No proper subject training | Low | Serious | Project & product | Avoidance | Use small, well trained team that is familiar with the requirements and programming languages involved. |
| Software tools cannot work together in an integrated way | Moderate | Tolerable | Project & Product | Minimization | Investigate the possibility of acquiring other software tools that can be easily integrated. |
| Staff members not following instructions correctly and therefore leads to inconsistencies in the project | Moderate | Serious | Project | Avoidance | Have a talk with the employee about what they are doing incorrectly  Exchange the roles of that employee with someone else on the team. |

# Appendix

## Sprint Backlog Cycle #1

A. Task Description: Viewing the Menu Items & more info details such as ingredients, price etc.

**Task Activities:**

(1) Add Menu Appetizer items to local db.

(2) Add Menu Entrees items to local db.

(3) Add Menu Desserts items to local db.

(4) Add Menu Drinks items to local db.

(5) Add functionality to view appetizers.

(6) Add functionality to view main courses.

(7) Add functionality to view desserts.

(8) Add functionality to view drinks.

B. Task Description: Create the main screen and add functionality to its components.

**Task Activities:**

(9) Create home/main screen.

(10) Add functionality to the home screen appetizers button.

(11) Add functionality to the home screen main course button.

(12) Add functionality to the home screen desserts button.

(13) Add functionality to the home screen drinks button.

(14) Add functionality to the home screen call waiter button.

(15) Add functionality to the home screen view bill.

(16) Add functionality to the home screen my order button.

(17) Adjust the home screen layout.

## Sprint Backlog Cycle #2

A. Task Description: Adding and or deleting items to/from the db. (Utilizing 1 menu section).

**Task Activities:**

(1) Database package:

Setup dB

(2) Database creation:

(a) Connect dB

(b) Create table

(3) Database Helper:

Create DB Helper

(4) Connect dB to the application.

(5) Adding a menu item to the db.

(6) Adding multiple items to the db.

(7) Deleting a menu item from the db.

(8) Deleting multiple items from the db.

(9) Integration testing to ensure the dB connect smoothly and performed as expected.

(10) Allow user input to alter the quantity of an item to an order.

(11) Adjusting the layout of the application.

## Sprint Backlog Cycle #3

A. Task Description: Adding and or deleting from all menu sections to/from the db.

**Task Activities:**

(1) Adding items to dB from all menu screens.

(2) Deleting items to dB from all menu screens.

B. Task Description: Loading the data from the dB and displaying the updates to the user

**Task Activities:**

(3) User test were conducted by the development team to ensure the data shown on the bill reflected what the user selected or updated on their order.

C. Task Description: Call waiter functionality

**Task Activities:**

(4) Generate a response when the user selects the call waiter button.

(5) Developer the time mechanism.

(6) Perform user test on the call waiter button to ensure the timer performs as expected.

(7) Optional: Food tracker progress bar.