Software Design Specification Document (CS360)

Campus Eat

Group Number: 10

Momina Haider
Syed Hamza Ahmad
Mohammad Razi Ul Haq
Mohammad Raza Khawaja
Muhammad Usama

Course: Software Engineering CS360

Instructor: Suleman Shahid

University: Lahore University of Management Sciences (LUMS)

Version: 1.0
Date: (29/03/2018)
Number of hours spent on this document: 108

Contents

1 Change Log

1.1 Project Scope

The purpose of Campus Eat is to make the food ordering system within LUMS by the LUMS Community more efficient and reliable. Our web application aims to address the problems associated with the current food ordering system in place. These problems are an aggregation of the issues that customers face while ordering food through phone calls, and issues that restaurant staff face to manage these orders. Some of the top issues highlighted through surveys and interviews with students and restaurant owners were:

- Social anxiety while talking to people on the phone.
- No way of comparing prices.
- Difficulty in exploring the menu.
- Difficulty to provide extra details within the order.
- Tedious for restaurants to write down and track orders made over the phone.
- Chances of noting down incorrect information over the phone are very high.

Campus Eat aims to solve these issues to make it easier for customers to view menus, prices, place orders, add extra order details and avail any promotions. They shall be more connected with the restaurants because of the ability to view the status of their order and make a better decision to eat based on a public restaurant rating. The application would also streamline the restaurant's process of tracking their orders. They would be able to view and process the orders in chronological order and have accurate order data provided in a presentable format - the customer's exact location, some personal preferences they would like in the food etc. Moreover, the ability to view an accurate log of their orders enables the restaurants to perform data analysis and offer promotions and price adjustments accordingly.

1.2 Change log

Change Made	**Reasoning**
A new use case (#23) was added, that the user will be able to logout of the web application.	A user might not want to stay logged on if they access the web application from a public machine.
New Functional Requirement (#23) and Use Case (#24) were added for Forgot Password.	The user should be able to get a new password if they have forgotten their current password.
Promotional Deals will not have a separate screen	It is easier to update deals for the manager and easier for a customer to order deals if they are part of the already existing menu
A new use Functional requirement (#24) was added, that the manager will be able to view their restaurant rating.	A manager might want to view their restaurant ratings.

2 Introduction

2.1 Document Purpose

This document primarily specifies the software design and architecture of an online food ordering system, a web application called "Campus Eat" which aims to address the problems associated with the current manual food ordering system in LUMS. The purpose of Campus Eat is to allow LUMS Community to order food from the on-campus restaurants in a smooth and efficient way. This document mainly focuses on specifying the high-level view of the architecture of our web application and the interactions between the user and the system. Moreover, the document provides the list of requirements (including functional and non-functional requirements) as mentioned in the SRS document in a more technical design manner which will form the basis of our system. This is the first version of the SDS document which aims to provide a high-level design framework from which our system will be built.

2.2 Product Scope

The purpose of Campus Eat is to make the food ordering system within LUMS by the LUMS Community more efficient and reliable. Our web application aims to address the problems associated with the current food ordering system in place. These problems are an aggregation of the issues that customers face while ordering food through phone calls, and issues that restaurant staff face to manage these orders. Some of the top issues highlighted through surveys and interviews with students and restaurant owners were:

- Social anxiety while talking to people on the phone.
- No way of comparing prices.
- Difficulty in exploring the menu.
- Difficulty to provide extra details within the order.
- Tedious for restaurants to write down and track orders made over the phone.
- Chances of noting down incorrect information over the phone are very high.

Campus Eat aims to solve these issues to make it easier for customers to view menus, prices, place orders, add extra order details and avail any promotions. They shall be more connected with the restaurants because of the ability to view the status of their order and make a better decision to eat based on a public restaurant rating.

The application would also streamline the restaurant's process of tracking their orders. They would be able to view and process the orders in chronological order and have accurate order data provided in a presentable format - the customer's exact location, some personal preferences they would like in the food etc. Moreover, the ability to view an accurate log of their orders enables the restaurants to perform data analysis and offer promotions and price adjustments accordingly.

2.3 Intended Audience and Document Overview

The document is intended for the following people:

- Dr. Suleman Shahid
- Teaching Assistants of CS360 Software Engineering
- Mr. Faisal, Owner of Jammin Java cafe.
- Mr. Mushtaq, Owner of Zakir Tikka.
- Mr. Saad, Owner of Chop Chop.
- Developers, testers and documentation writers of Campus Eat - which includes the entirety of Group 10.

This document acts as a blueprint for the realization of both functional and non-functional requirements that were previously highlighted, in the Software Requirements Specification document, providing an overview of the interface design, architectural design, database design etc. The document consists of six sections. The first briefly describes the scope of the project and accounts for any changes made since the SRS document. The second section entails the purpose, scope and intended audience of this document. The third provides an overview of the system, showing the architectural design and strategies used and the constraints of the system. The fourth section describes the system and subsystem architecture in depth, along with the database design and the interface requirements. Section five illustrates and explains the interface that will be output by the system to the users. Non-functional system requirements are then discussed in section six.

Readers should begin by gaining the project overview by reading section 1 and 2 and then understand the context and constraints of the system in section 3. Section 6 should be read next to gain a basic understanding of the non-functional requirements of the system. The interface design should be looked at next, by reading section 5, followed by going into the details of the system and subsystem architecture in section 4.

2.4 Definitions, Acronyms and Abbreviations

- **GUI:** Graphical User Interface. It is a type of interface that allows users to communicate through text, icons and buttons.
- **HTTPS:** Hypertext Transfer Protocol Secure
- **JSON:** Javascript Object Notation. It is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types.
- **SQL:** Structured Query Language – used for making database queries.

2.5 References and Acknowledgments

3 Overall Description

3.1 System overview

The system being developed is a web-based food ordering system, designed to replace the current call-based food ordering method in place at LUMS. It aims to streamline and automate the process of placing delivery orders at various eateries within the LUMS campus. The system would allow users to create and login to their accounts to place orders. It would enable them to view restaurant menus and compare prices of items across different restaurants before placing an order through the system. The system would also enable restaurant owners to update their menu items, prices and view orders currently in process. The restaurant cashiers would also be required to login to their accounts. Once logged in, they would be able to view orders placed by students, specify a delivery time for those orders and also change the status of orders from 'processing' to 'delivered'.

Our system is divided into many modules as part of our modular design approach. Each module performs some specific function. Each module compliments the overall structure and functionality of Campus Eat. For instance, on the main app webpage the render module triggers the screen layout to be displayed to the customer/cashier/manager. Upon submitting the login credentials, the log in module triggers and verifies the details by invoking the database module. This database module on successful verification gives a heads up to the login module which in turn render module to display the screen according to that login. The overall architecture follows the classic client-server design. All processing is to be done on the server, which means that the server is always the ultimate destination of all responses by all clients. The clients will only be able to render the data from the render module onto the screen.

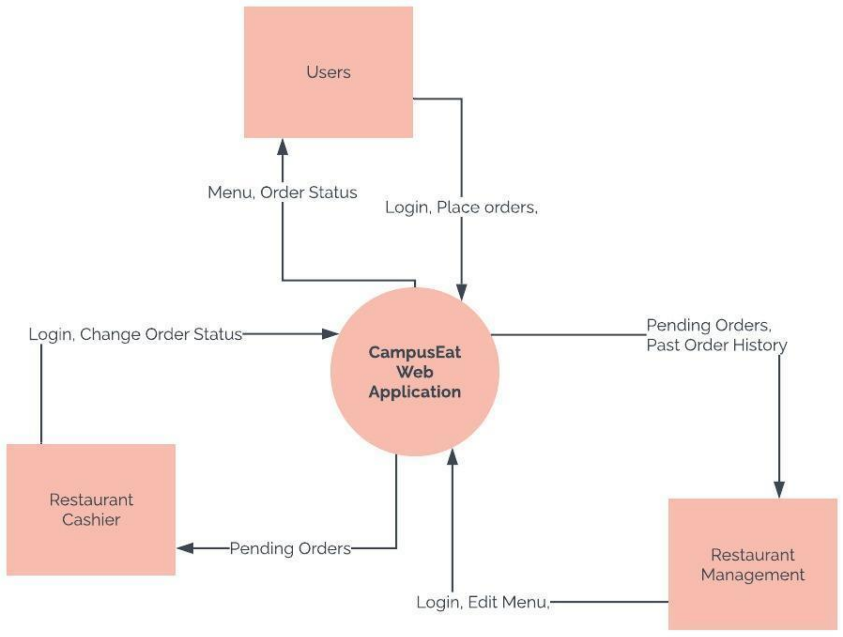


Figure 3.1.1: General Diagram

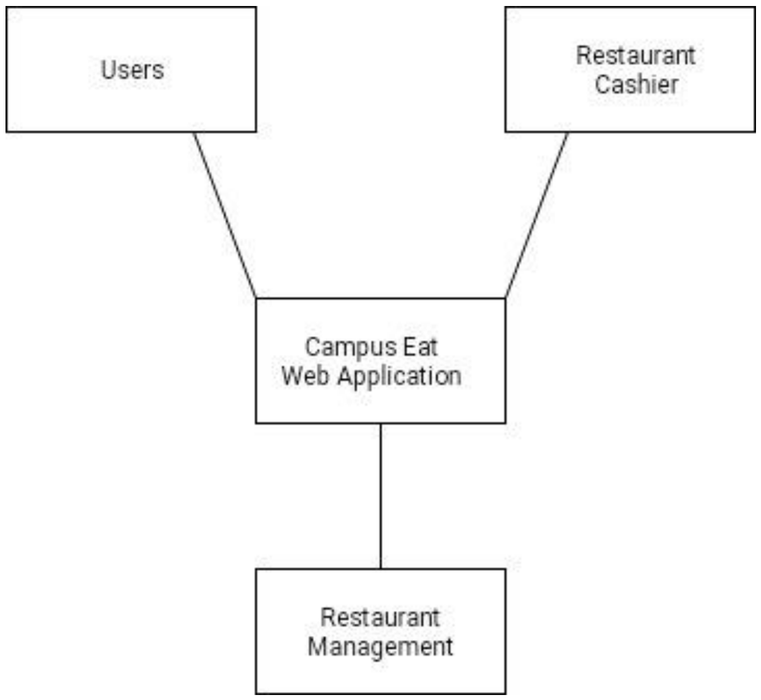


Figure 3.1.2: Context Diagram

3.2 System constraints

3.2.1 Performance

- The system should be a Progressive web application. It should load in minimal time even on flaky networks, should have icons on home screen and loads as a top-level, full screen experience. This would be particularly useful for students, since not all of them own high-end mobile phones capable of installing applications on them. 4. The system should be able to return a search query within
- Since we would be using an external database (MongoDB) to store our data we would have to write code to integrate it in our JavaScript code. Using an external Database may lead to the database becoming a bottleneck in processing and displaying of data. Moreover, scalability might be an issue since we would be using a free version of MongoDB, which only provides 500MB database space, which might run out.
- The search feature should be fast and must return results within 3 seconds. If search takes any longer, it might make the users avoid using the feature, causing them to potentially miss out on an important feature of the app

3.2.2 Availability

- The application is dependent on availability of Internet for it to function. Without an internet connection, the users would not be able to place orders, nor will the restaurants be able to receive orders.
- The application is being hosted on a third-party hosting site (Heroku), hence it is dependent on its servers for the amount of traffic it can handle. Since the free version of Heroku won't support a large amount of traffic, the system may crash if a large number of users try to simultaneously access it.

3.2.3 Time

- Since it is a complex system with many interdependent components, it would be difficult to add all the desired functionalities as mentioned in the SRS within the given time. The system may lack some of the bonus features due to the time constraint limiting its usefulness to the restaurant owners in particular.

3.2.4 Interface requirements

- The interface of the system should be simple to understand and use. Since we have assumed that the restaurant staff at LUMS should be able to use the web application without any excessive training, we will have to keep the interface self-explanatory and less technical.
- Smaller images, more white or negative space, one font instead of different fonts are some of the go-light-for-minimalism to design approach that will be used in our interface to increase the web page loading speed.

3.2.5 Minimum Possible Manual Input and Data validation

- The system designed should take minimum possible manual input from the user to ensure error-free data processing. Lesser the manual input from the user, lesser will be a chance of error occurrence.
- The system should be able to validate the user input and display error messages in case of invalid input. For instance, the designed system should be able to check if the user input is of the correct data type to ensure correct/complete food order processing.

3.2.6 Security

- The system would need to ensure that the user information is stored in a secure way and care should be taken to prevent any possible breach of information privacy.
- The system should be secured against possible DDOS attacks that might make it unavailable for use. Repeated such attacks will render the application as useless forcing the users to switch back to the old call-based ordering system.

- The system should be protected against both Stored and Reflected Cross Site Scripting (XSS) attacks. Since the application requires input from users at repeated places, care must be taken to prevent these attacks which may propagate to the large user base of the application.

3.3 Architectural strategies

- A client-server architecture will be used, where the user will be the client and will send requests to the server and receive responses.
- Existing software for Creating accounts and Login will be reused, but it is subject to availability, which will be confirmed during the development phase of the project.
- Restaurant managers might want to view order history categorized according to certain parameters. Extensions like these will be catered through the modification of the code.
- HTTPS will be used as it is widely used and supported.
- The website application will be based on the following:
    1. **MongoDB:** This is the database that will be used. The motivation of the MongoDB language is to implement a data store that provides high performance, high availability, and automatic scaling.
    2. **NodeJS:** This is a JavaScript runtime environment which is asynchronous and event-driven instead of waiting for a task to complete before moving on to another task. This will be used in the back-end. This will enable the performance of multiple operations in parallel.
    3. **Express:** This is the web application framework that will be used to run the back-end application. Express sends and receives queries and responses from the Database.
    4. **ReactJS:** This JavaScript library will be used to create user interfaces that will be interactive. React is known to be fast, scalable and simple.

4 System Architecture

4.1 System Architecture

4.1.1 Component Diagram

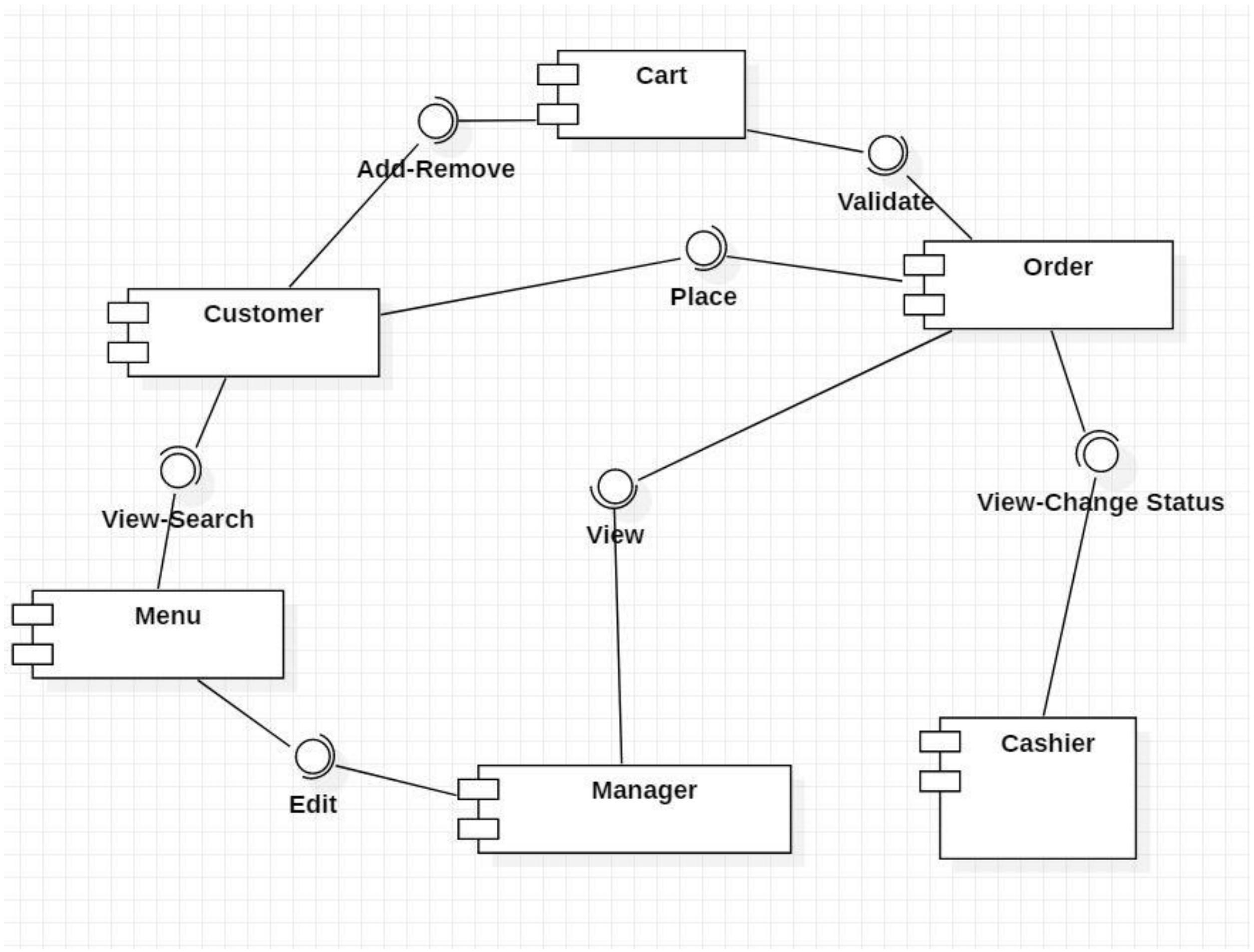


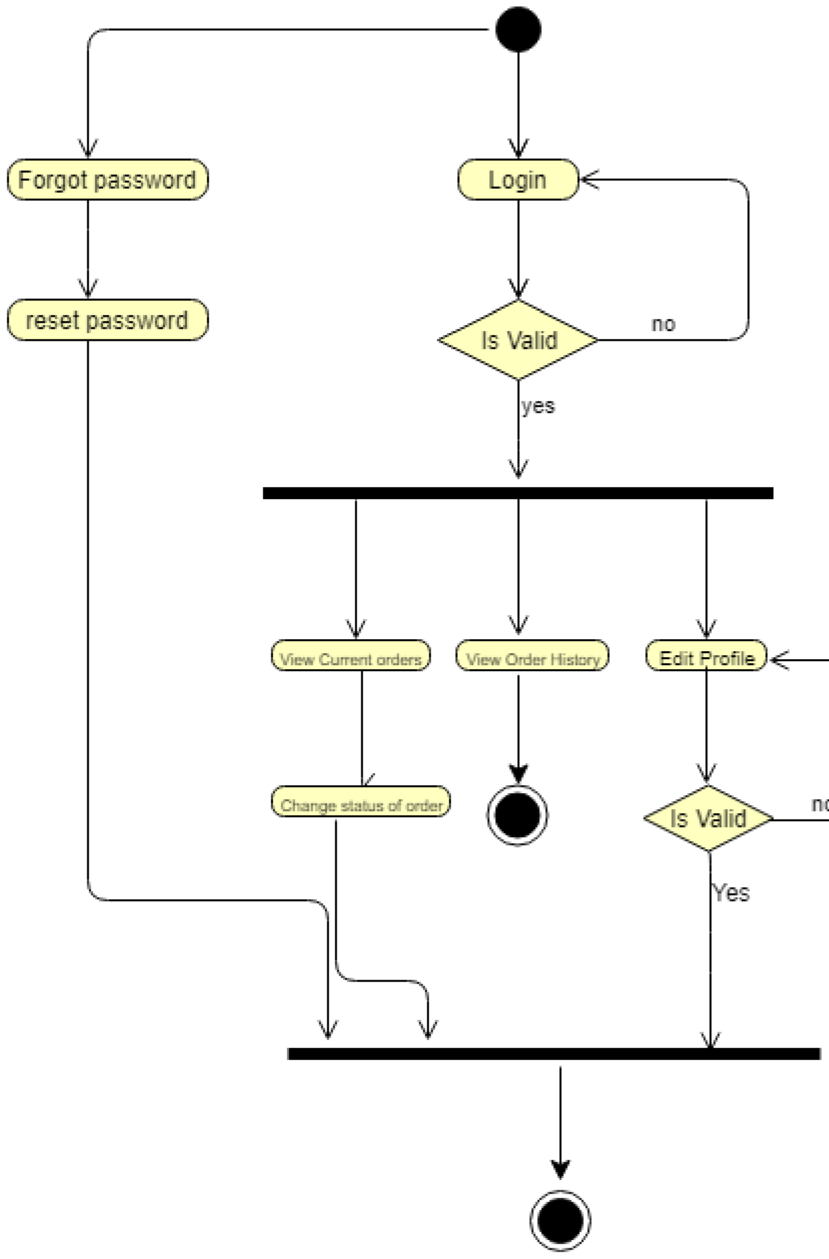
Figure 4.1.1.1: Component Diagram

- **<u>Customer:</u>** Upon signing up, a user becomes a customer. Customers can view the menu items for each restaurant and place their orders based on the menu items selected from each restaurant. These items are first added to shopping cart. Once a customer confirms their order their order is placed.
- **<u>Manager:</u>** Managers can edit the menu items for their restaurant. They can view order placed at their own restaurants only.
- **<u>Cashier:</u>** Cashier can view all the placed orders. They can change the status of these orders as well.
- **<u>Menu:</u>** Menu is a list of menu items of a restaurant. Customers view the menu to add items to their shopping cart.

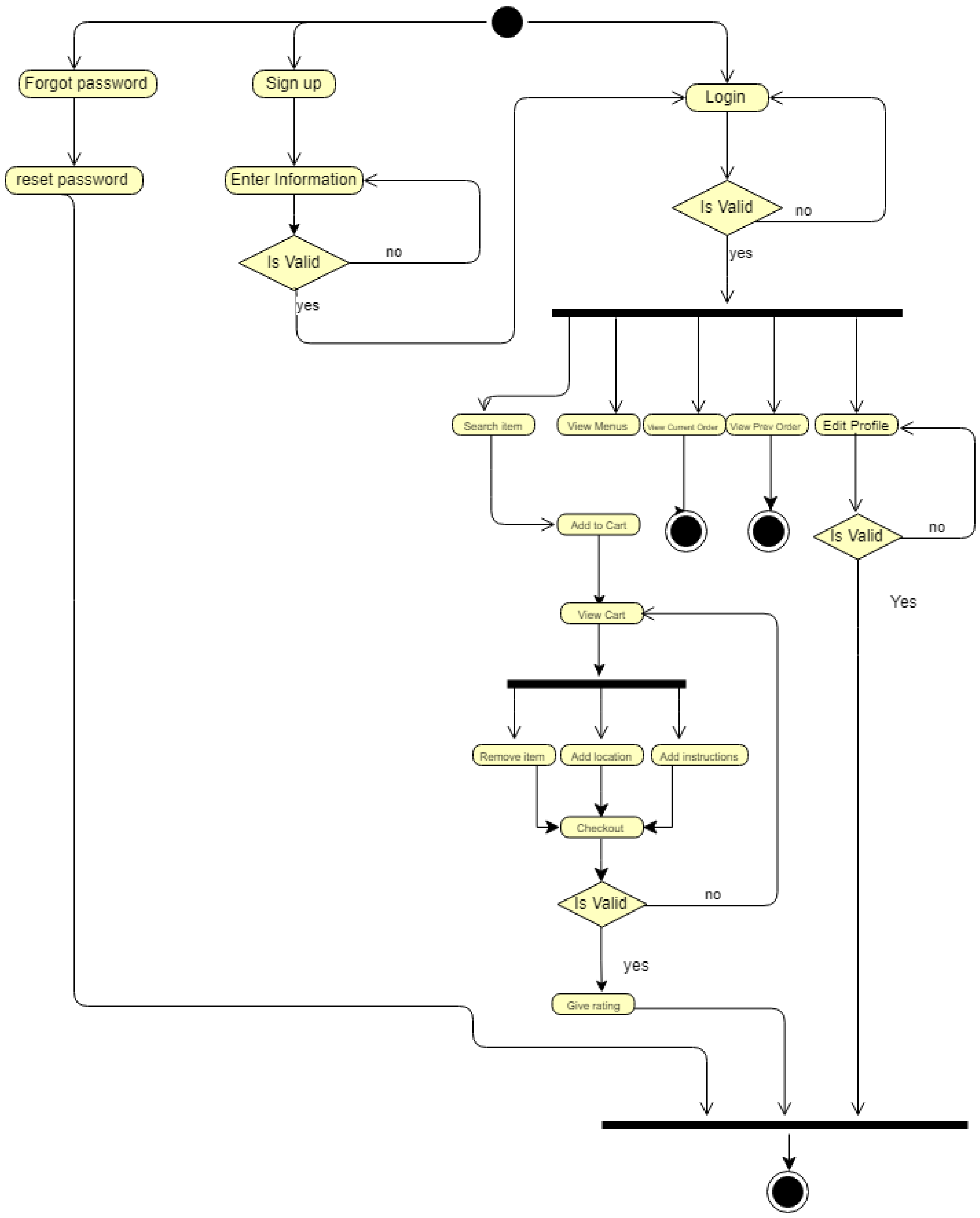
- **Cart:** Customers can add items from the menu to their shopping cart. Once an item is in the shopping cart, they can change the quantity of these items or remove items from the cart before placing the final order.
- **Order:** The items finalized by the customers' form part of an order that is visible to the manager and the cashier. The order also contains the customer details, special instructions by the customer and the delivery location.

4.1.2 Activity Diagrams for each component

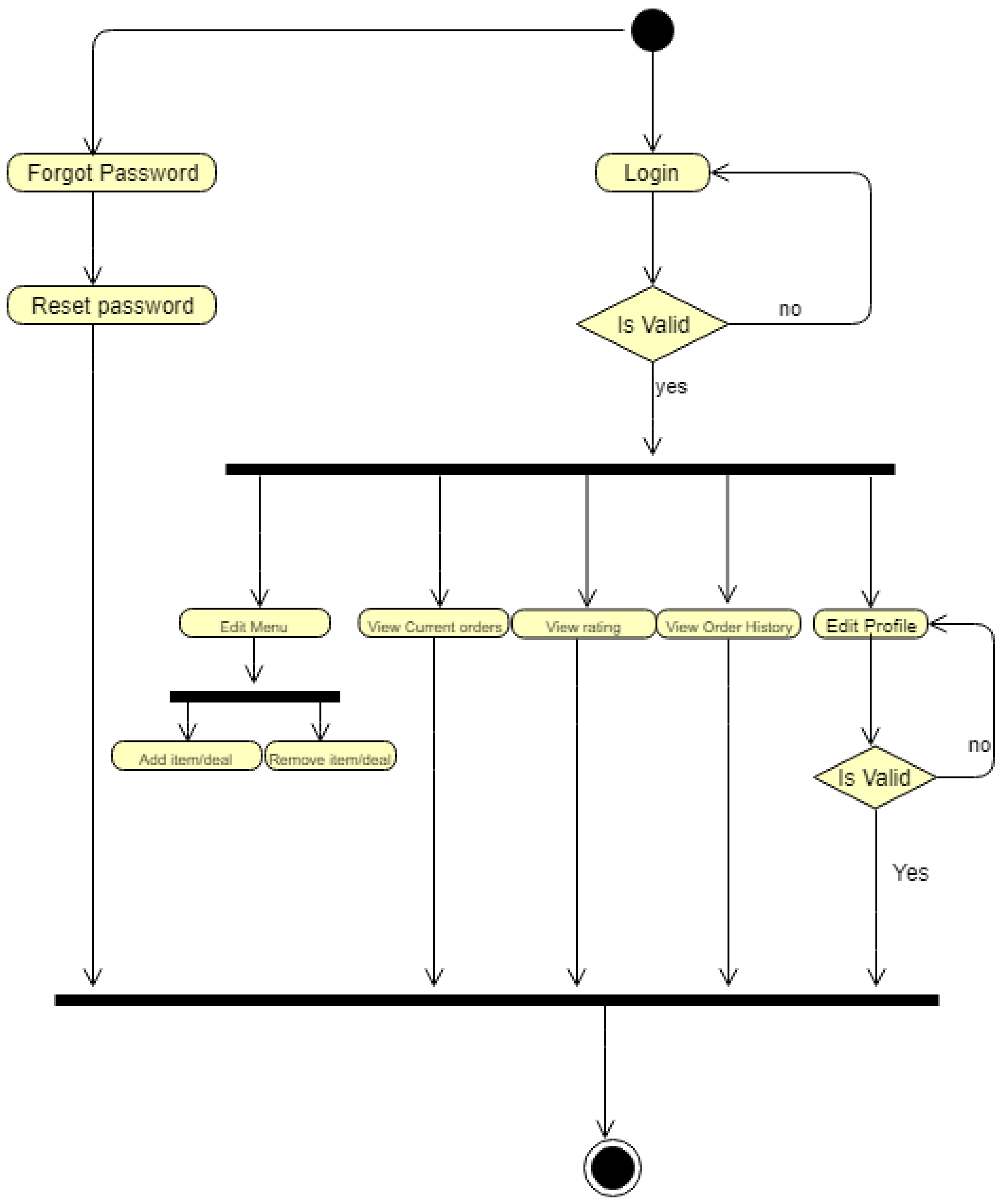
4.1.2.1 Cashier



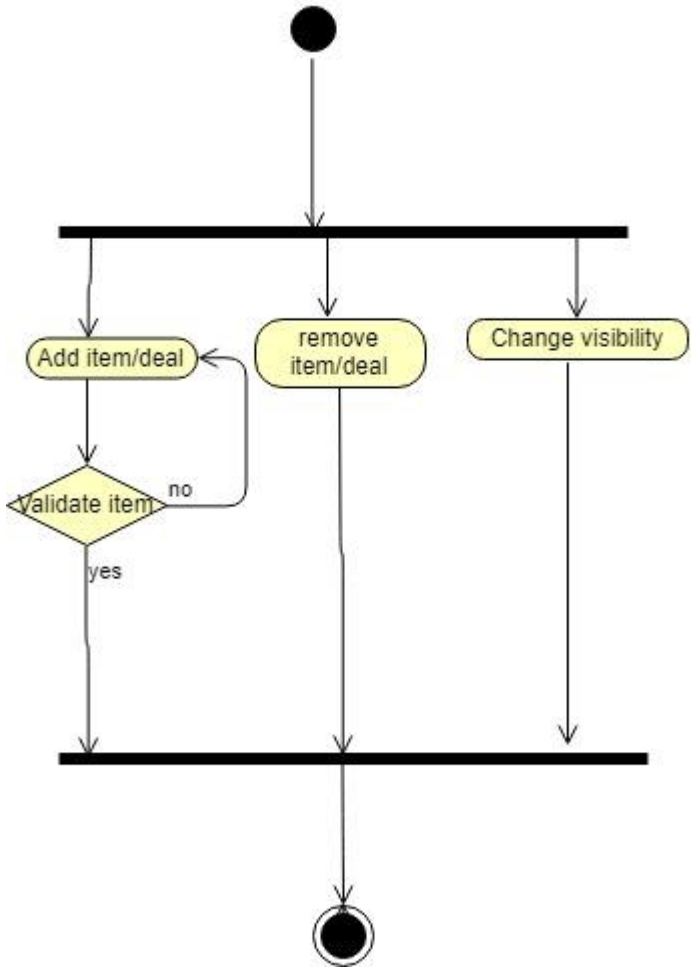
4.1.2.2 Customer



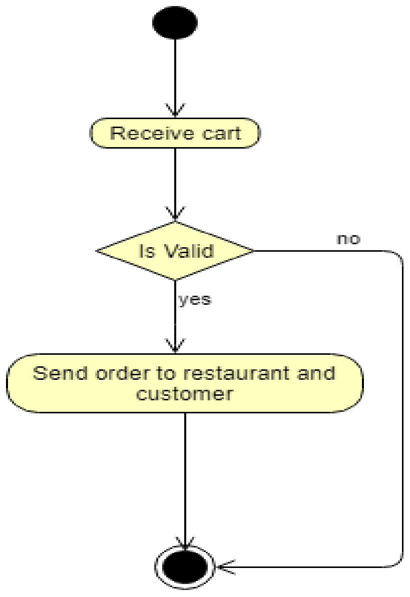
4.1.2.3 Manager



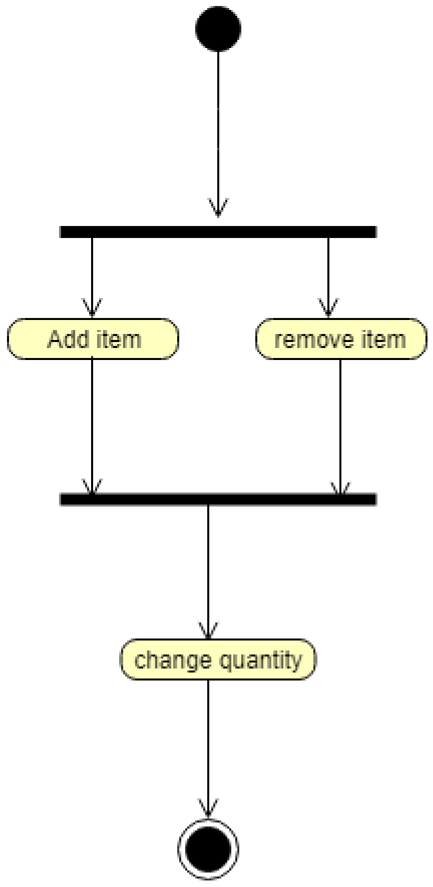
4.1.2.4 Menu



4.1.2.5 Orders



4.1.2.6 Shopping Cart

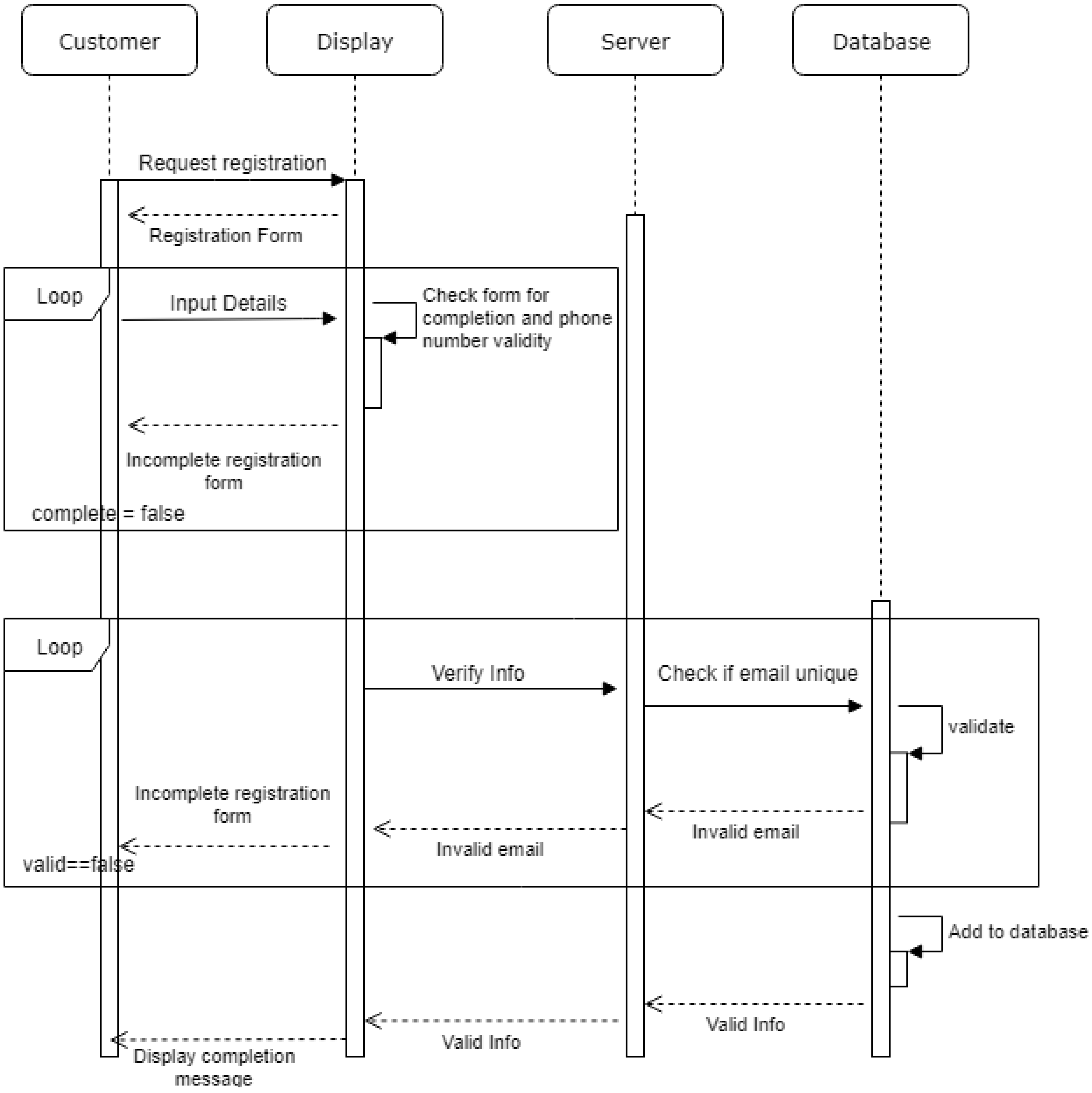


4.2 Subsystem Architecture

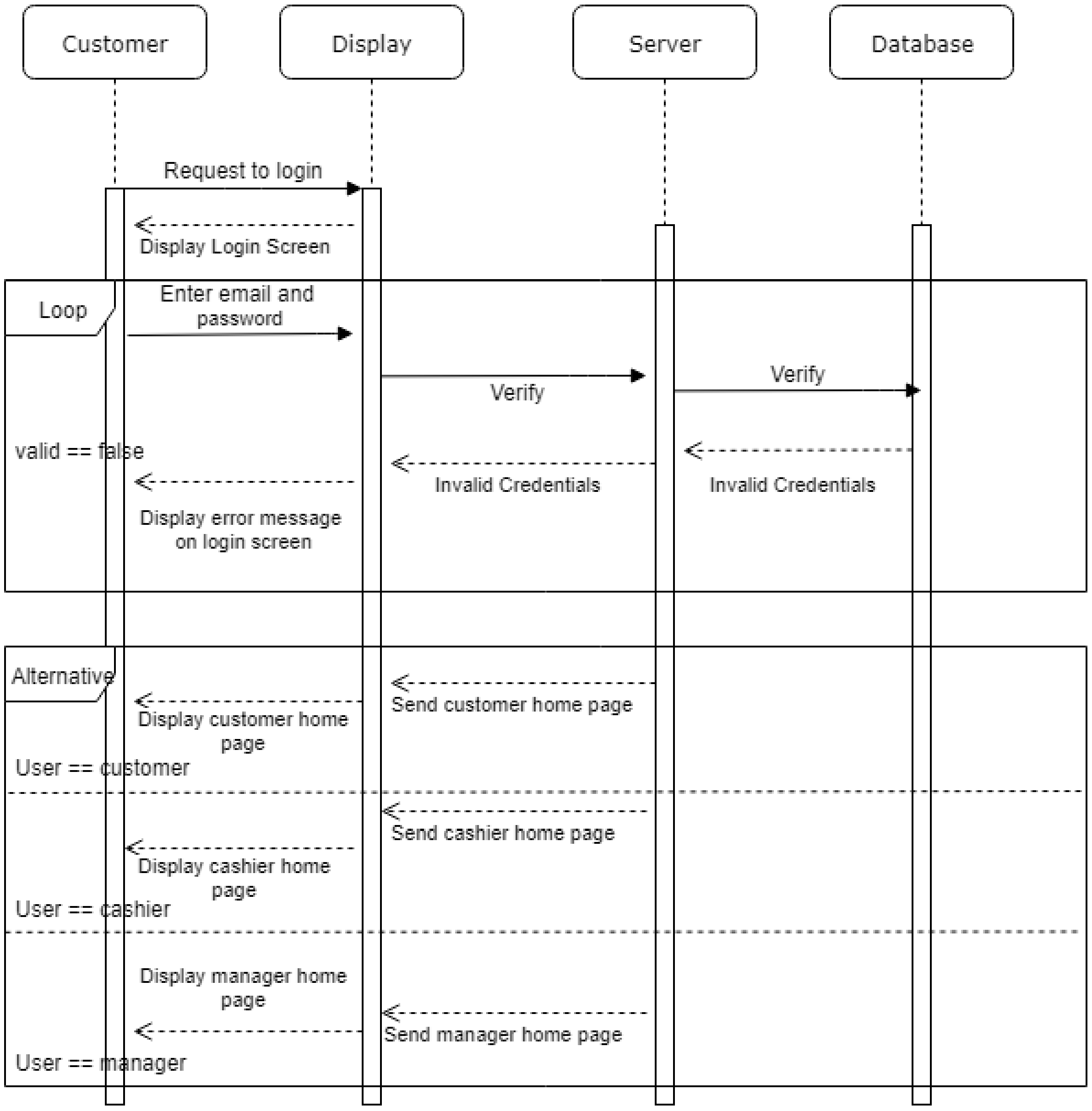
4.2.1 Sequence Diagrams

The following sequence diagrams cater to each of the use cases stated in the SRS document:

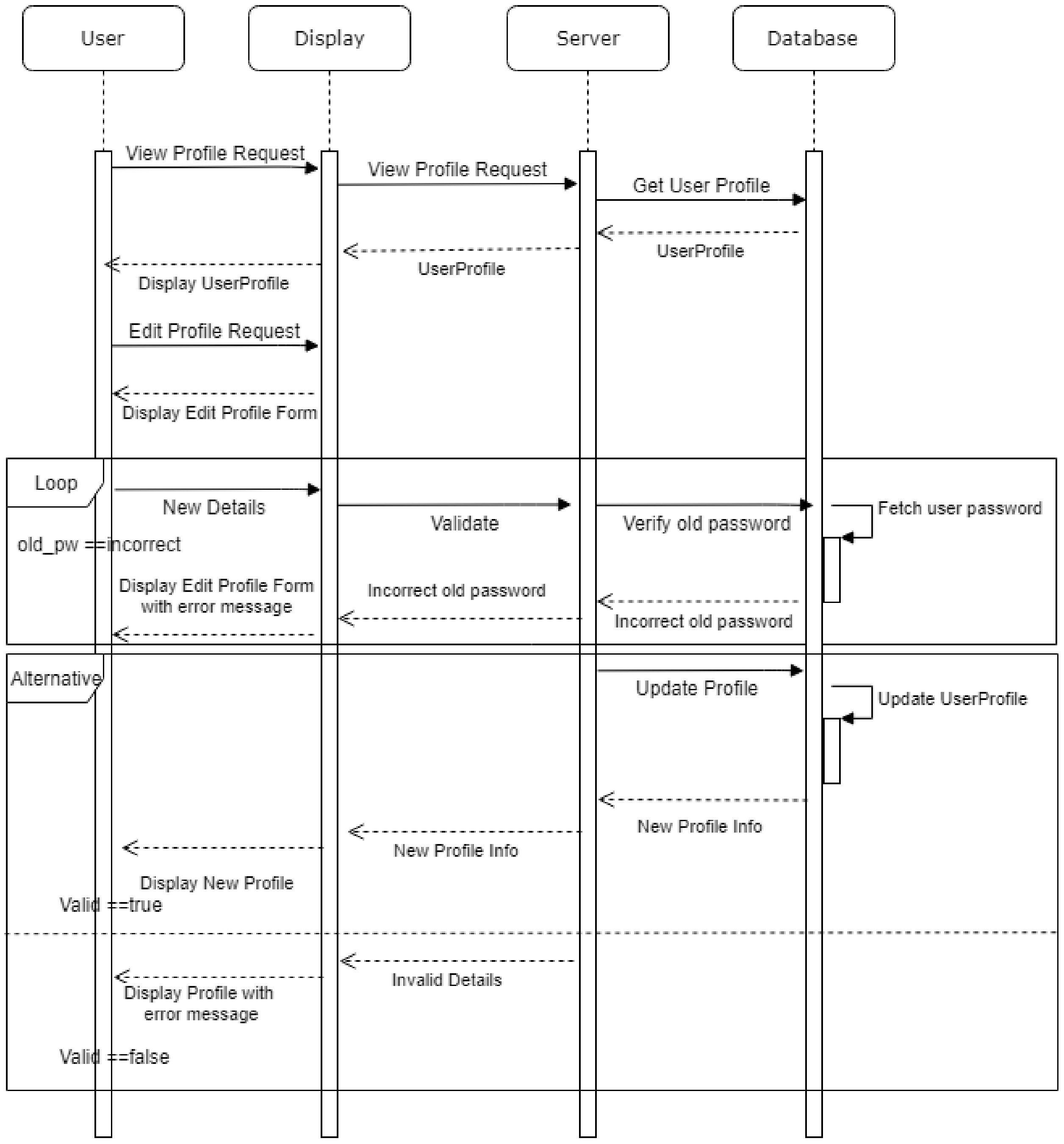
4.2.1.1 Use Case 1: Create Account



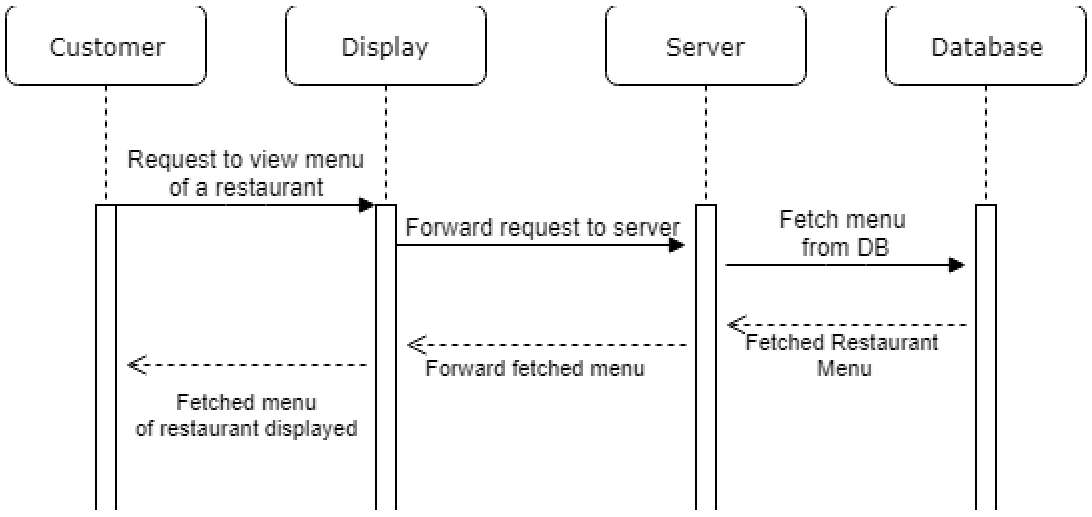
4.2.1.2 Use Case 2: Login



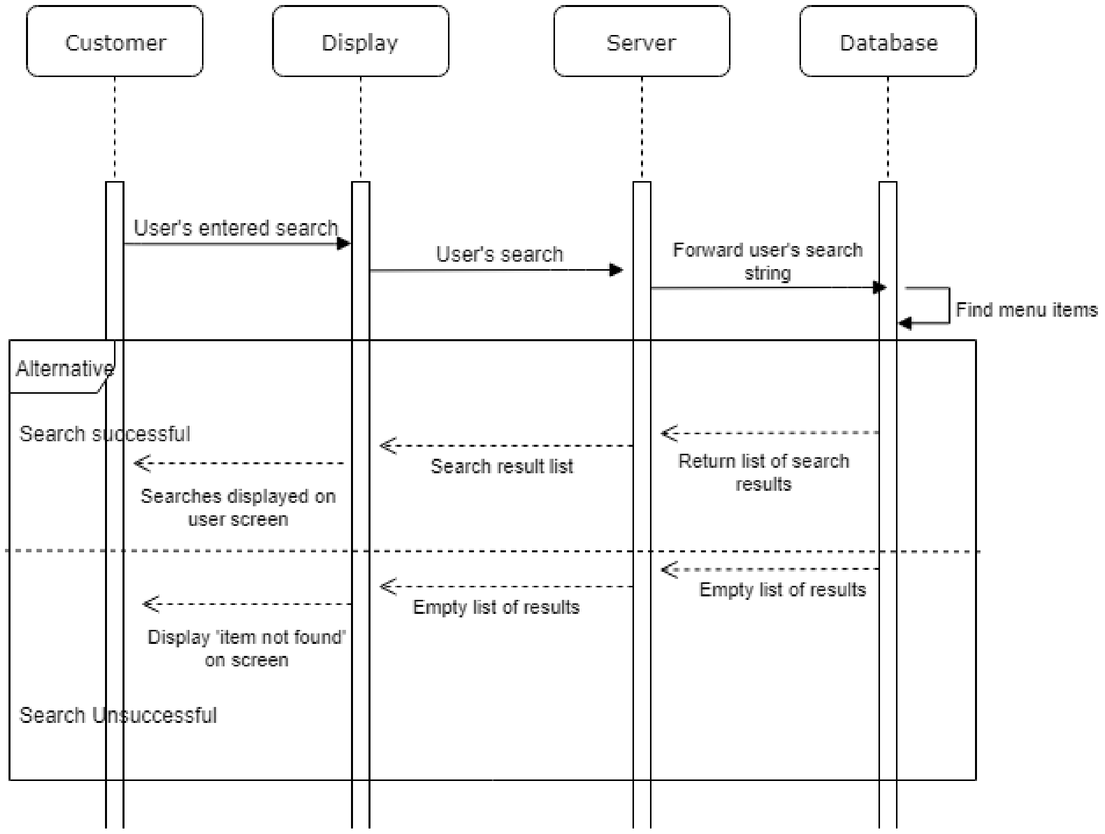
4.2.1.3 Use Case 3: Edit Profile



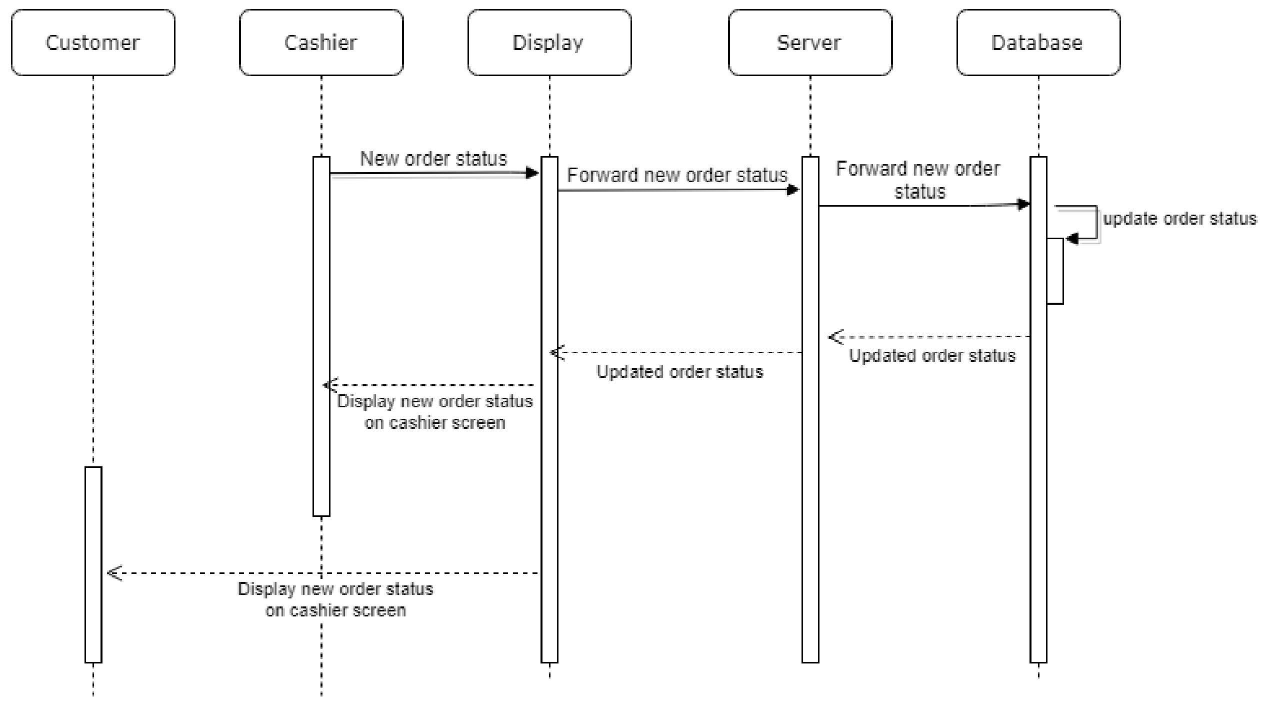
4.2.1.4 Use Case 4: View Menu


Customer
Display
Server
Database
Request to view menu of a restaurant
Forward request to server
Fetch menu from DB
Fetched Restaurant Menu
Forward fetched menu
Fetched menu of restaurant displayed

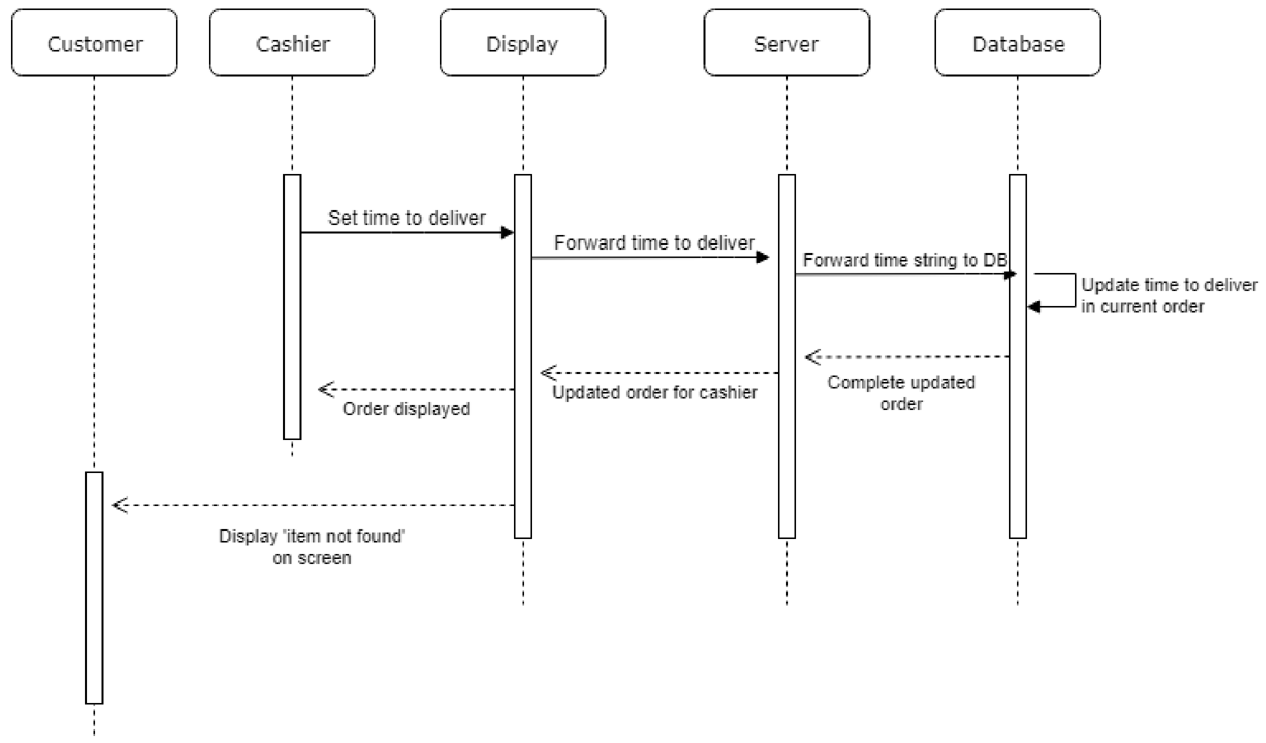
4.2.1.5 Use Case 5: Search Menu Items



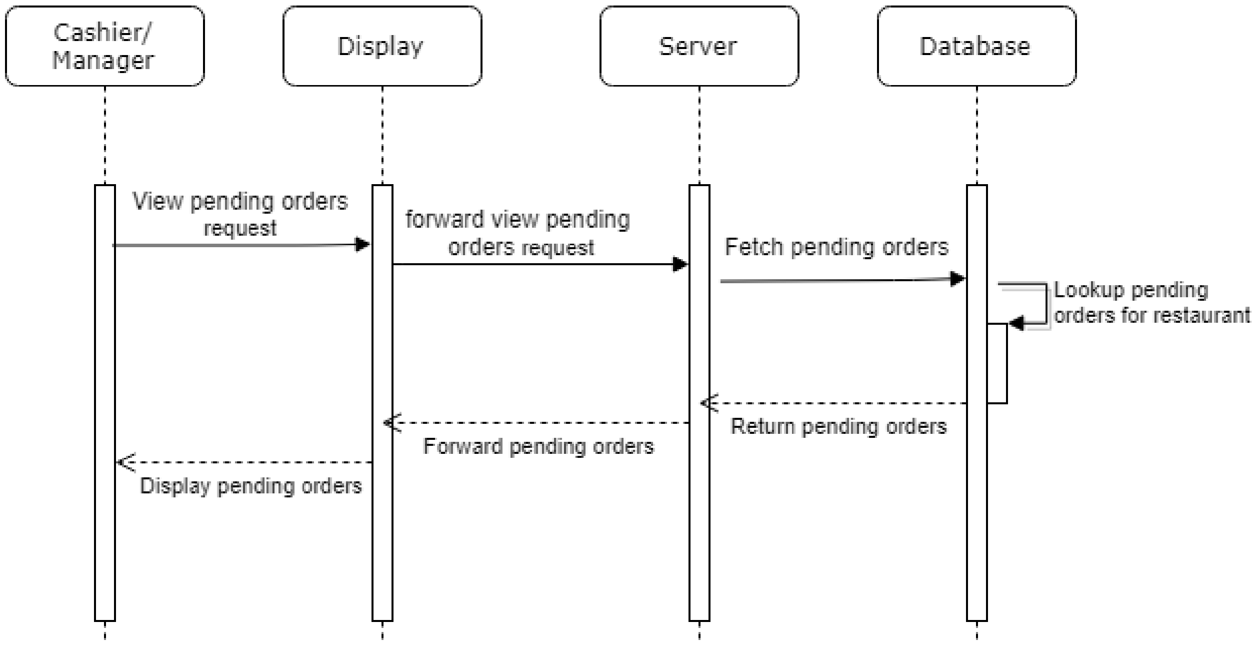
4.2.1.6 Use Case 6: Change Order Status



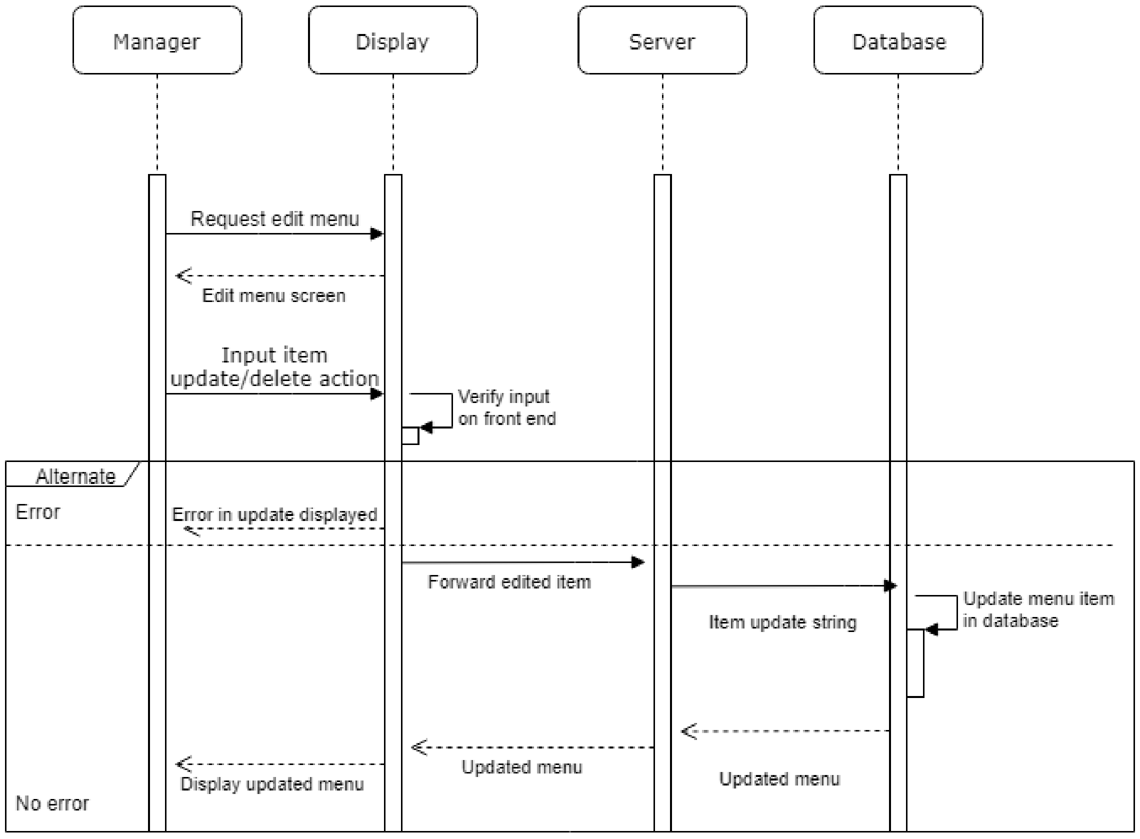
4.2.1.7 Use Case 7: Specify Time to Deliver


Customer
Cashier
Display
Server
Database
Set time to deliver
Forward time to deliver
Forward time string to DB
Update time to deliver in current order
Complete updated order
Updated order for cashier
Order displayed
Display 'item not found' on screen

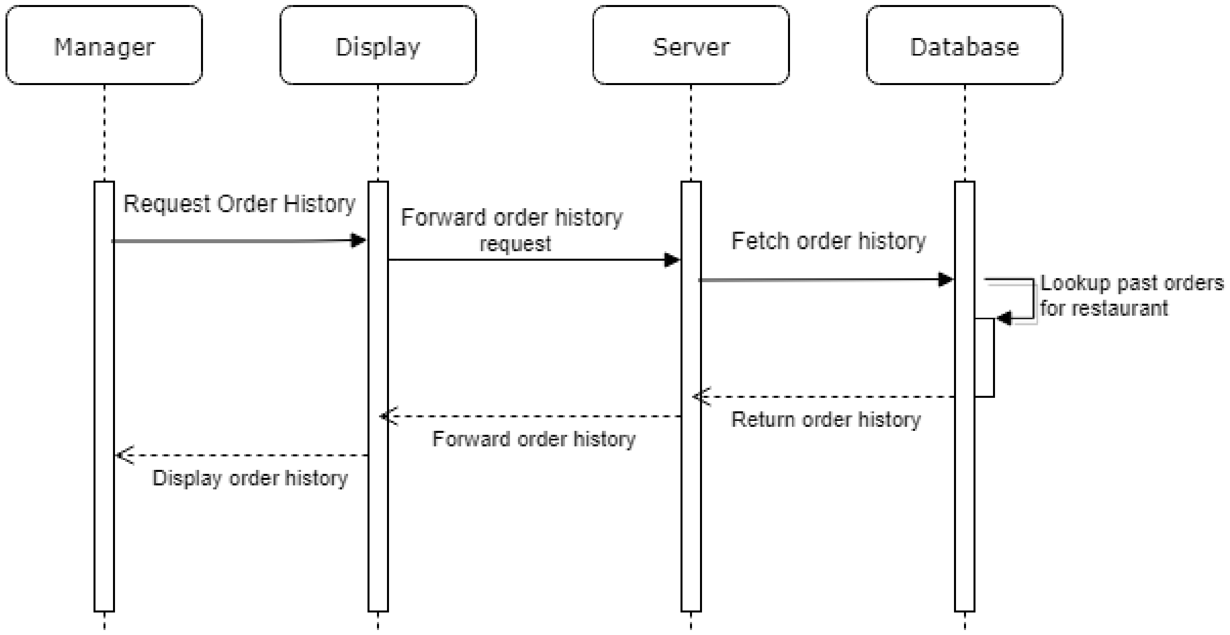
4.2.1.8 Use Case 8: View Pending Orders



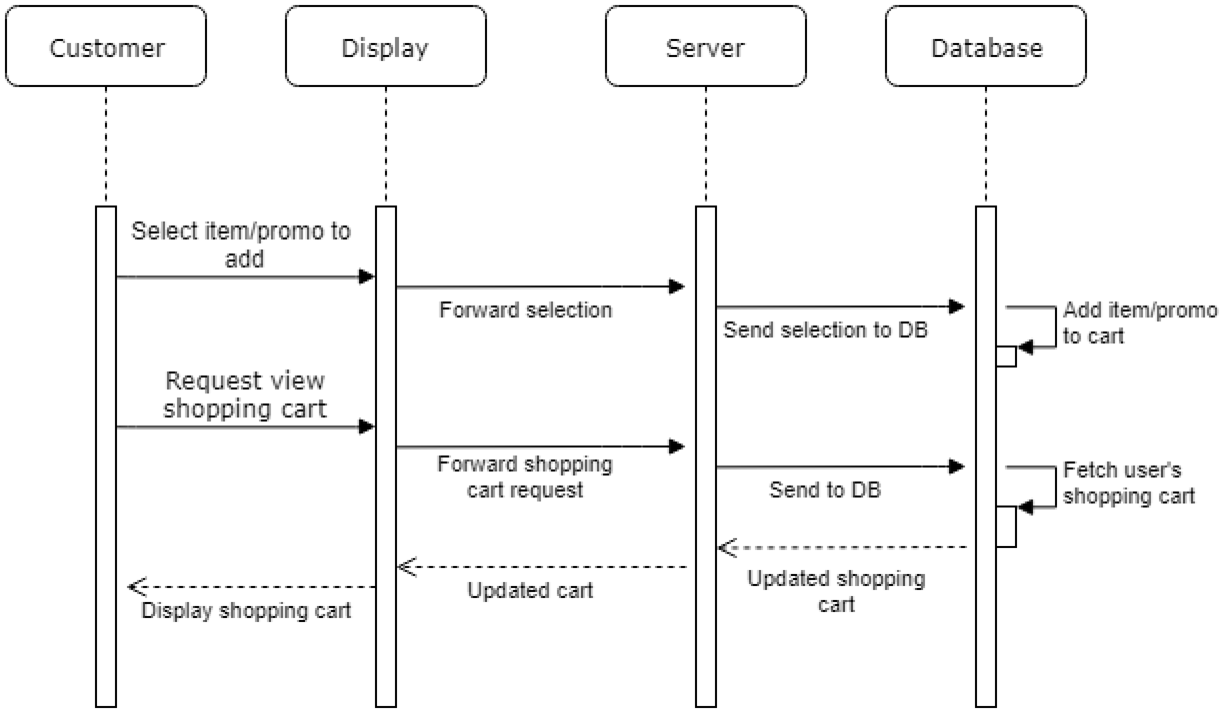
4.2.1.9 Use Case 9 + Use Case 11: Edit Menu + Add/Remove Promos



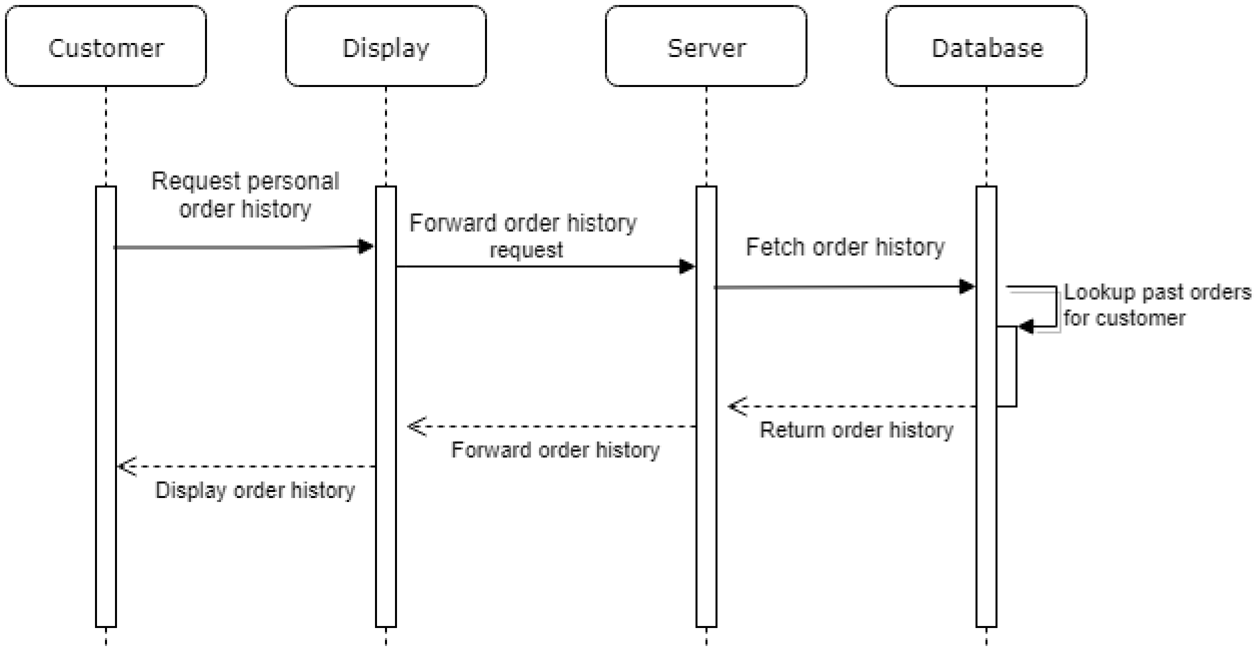
4.2.1.10 Use Case 10: View Restaurant Order History



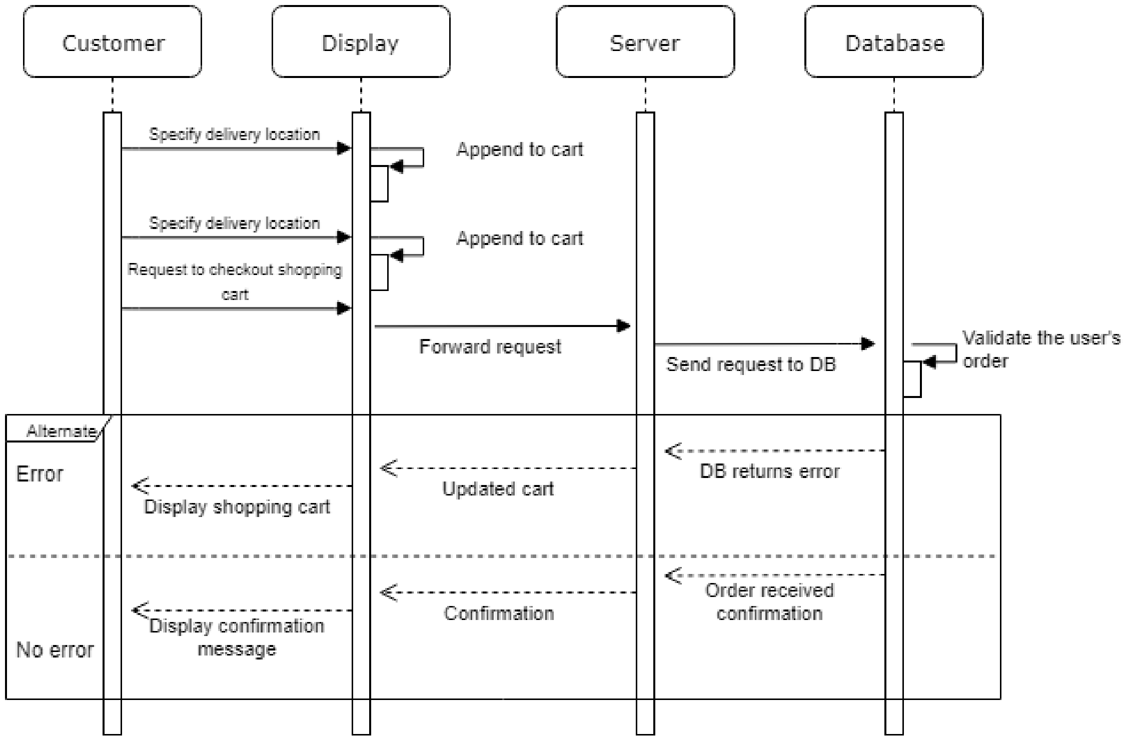
4.2.1.11 Use Case 12 + Use Case 14: Add item to shopping cart + Avail Promo



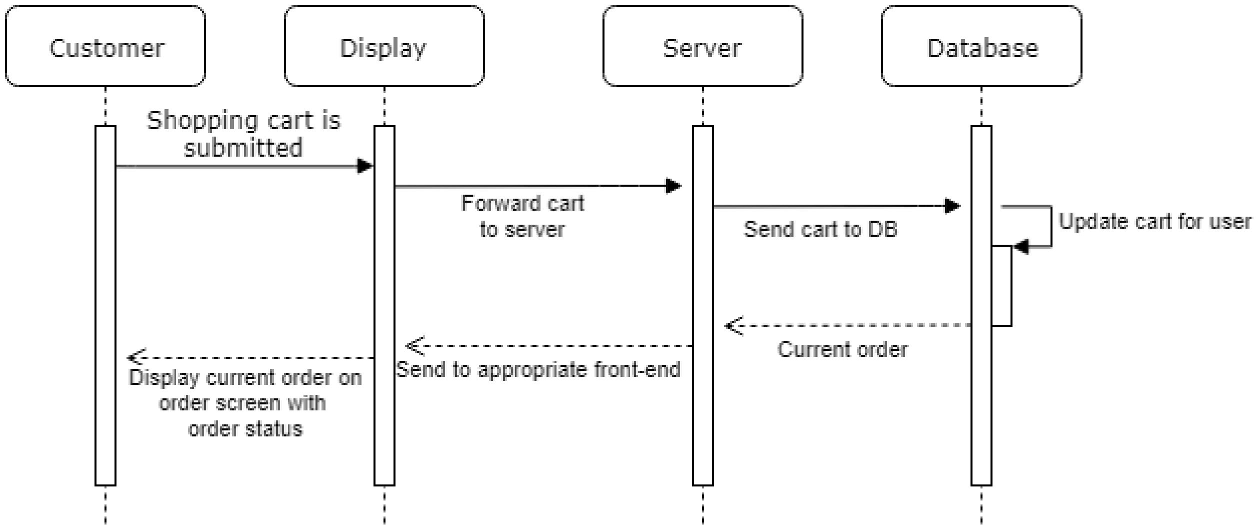
4.2.1.12 Use Case 15: View Personal Order History



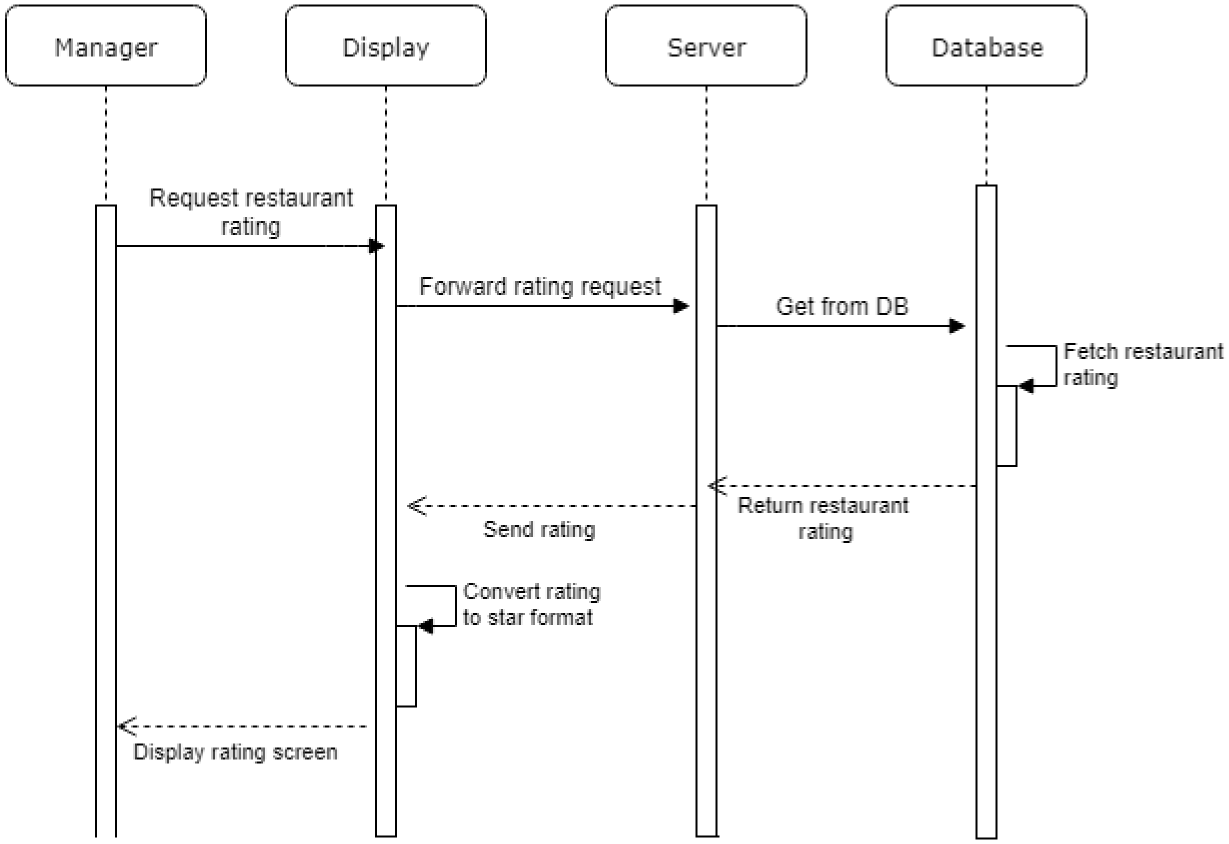
4.2.1.13 Use Case 13 + Use Case 21 + Use Case 22: Place Order(s) + Specify Delivery Instructions + Specify Delivery Location



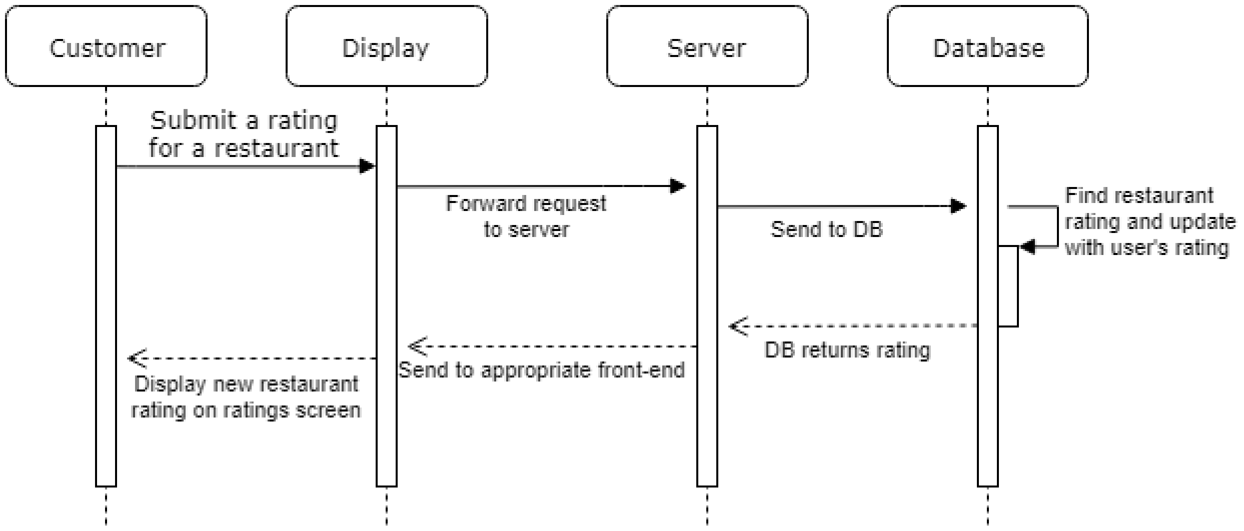
4.2.1.14 Use Case 16: View Order Status


Customer
Display
Server
Database
Shopping cart is submitted
Forward cart to server
Send cart to DB
Update cart for user
Current order
Send to appropriate front-end
Display current order on order screen with order status

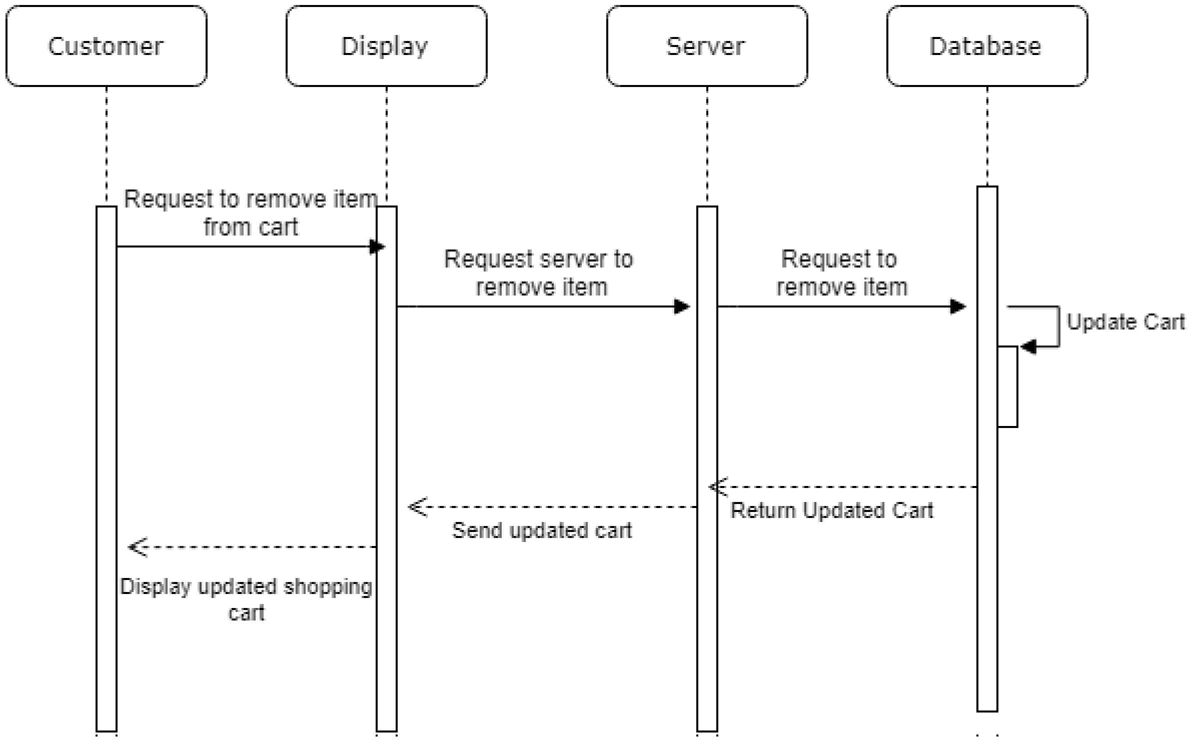
4.2.1.15 Use Case 17: View Restaurant Rating



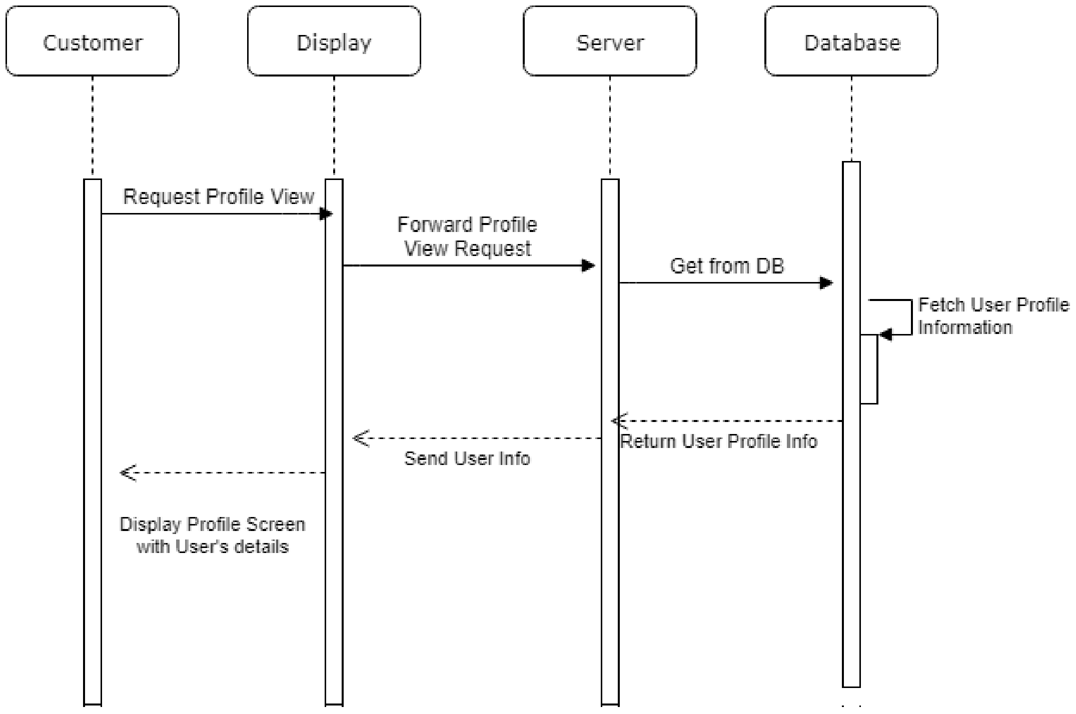
4.2.1.16 Use Case 18: Give rating to restaurant


Customer
Display
Server
Database
Submit a rating for a restaurant
Forward request to server
Send to DB
Find restaurant rating and update with user's rating
DB returns rating
Send to appropriate front-end
Display new restaurant rating on ratings screen

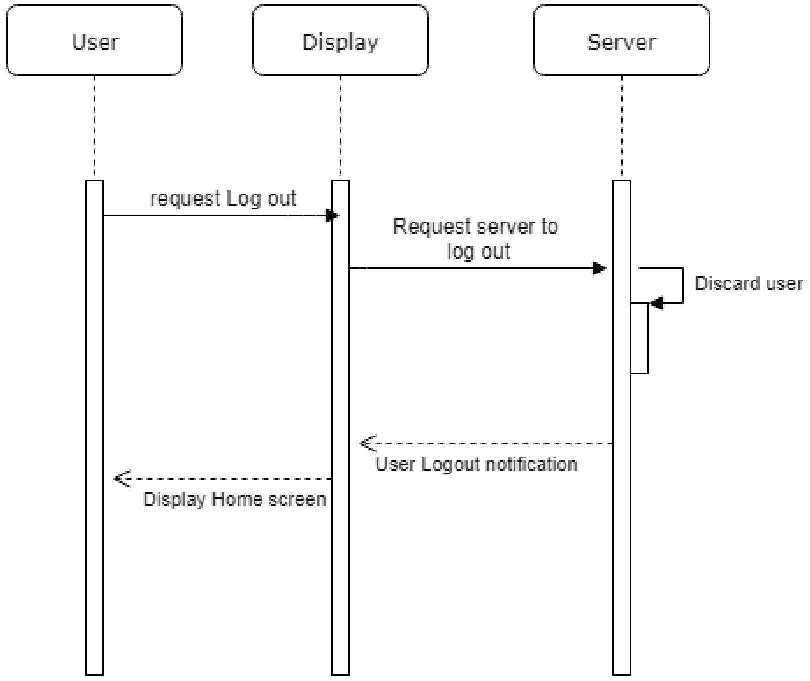
4.2.1.17 Use Case 19: Remove Item from Shopping Cart



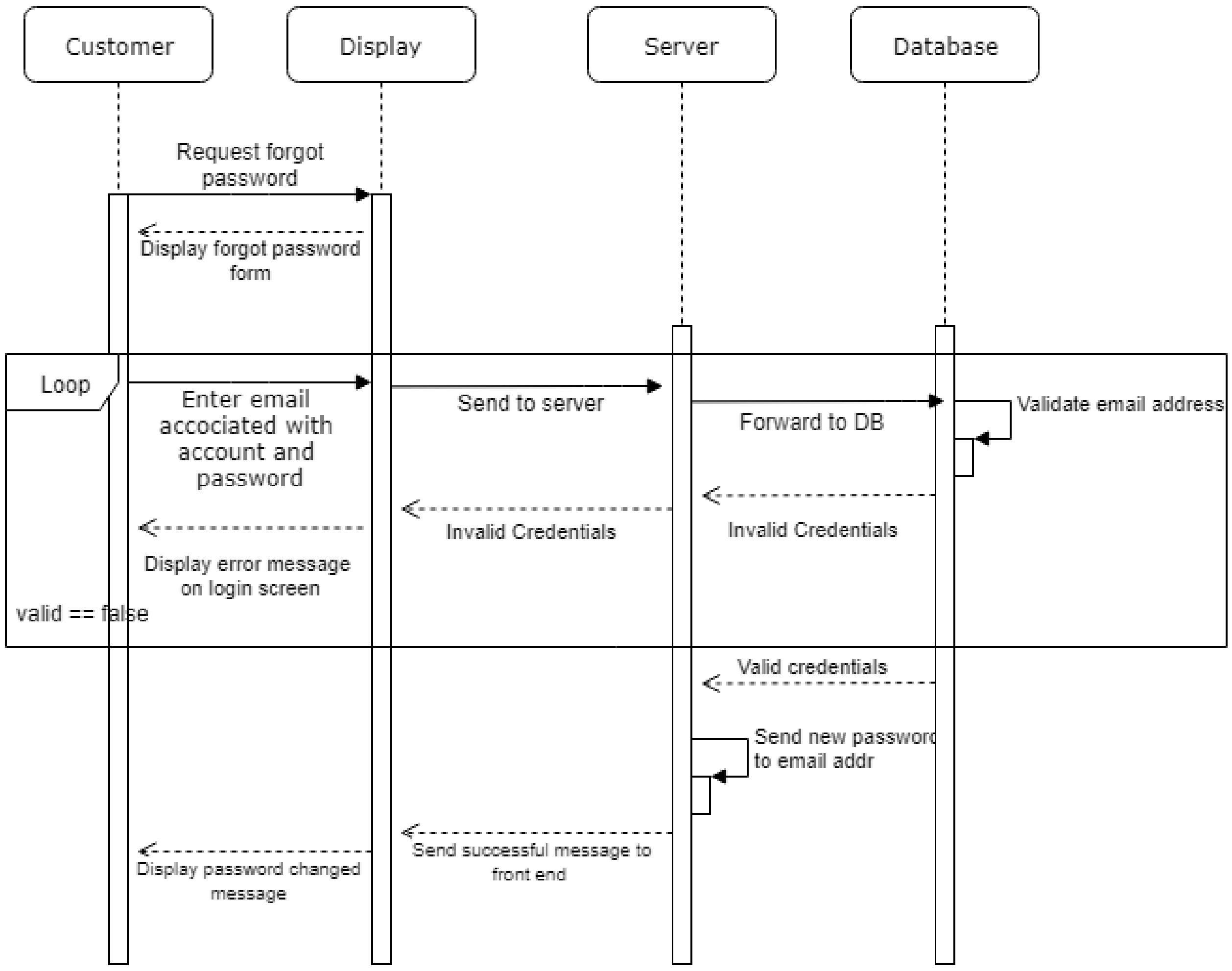
4.2.1.18 Use Case 20: View Profile


Customer
Display
Server
Database
Request Profile View
Forward Profile View Request
Get from DB
Fetch User Profile Information
Return User Profile Info
Send User Info
Display Profile Screen with User's details

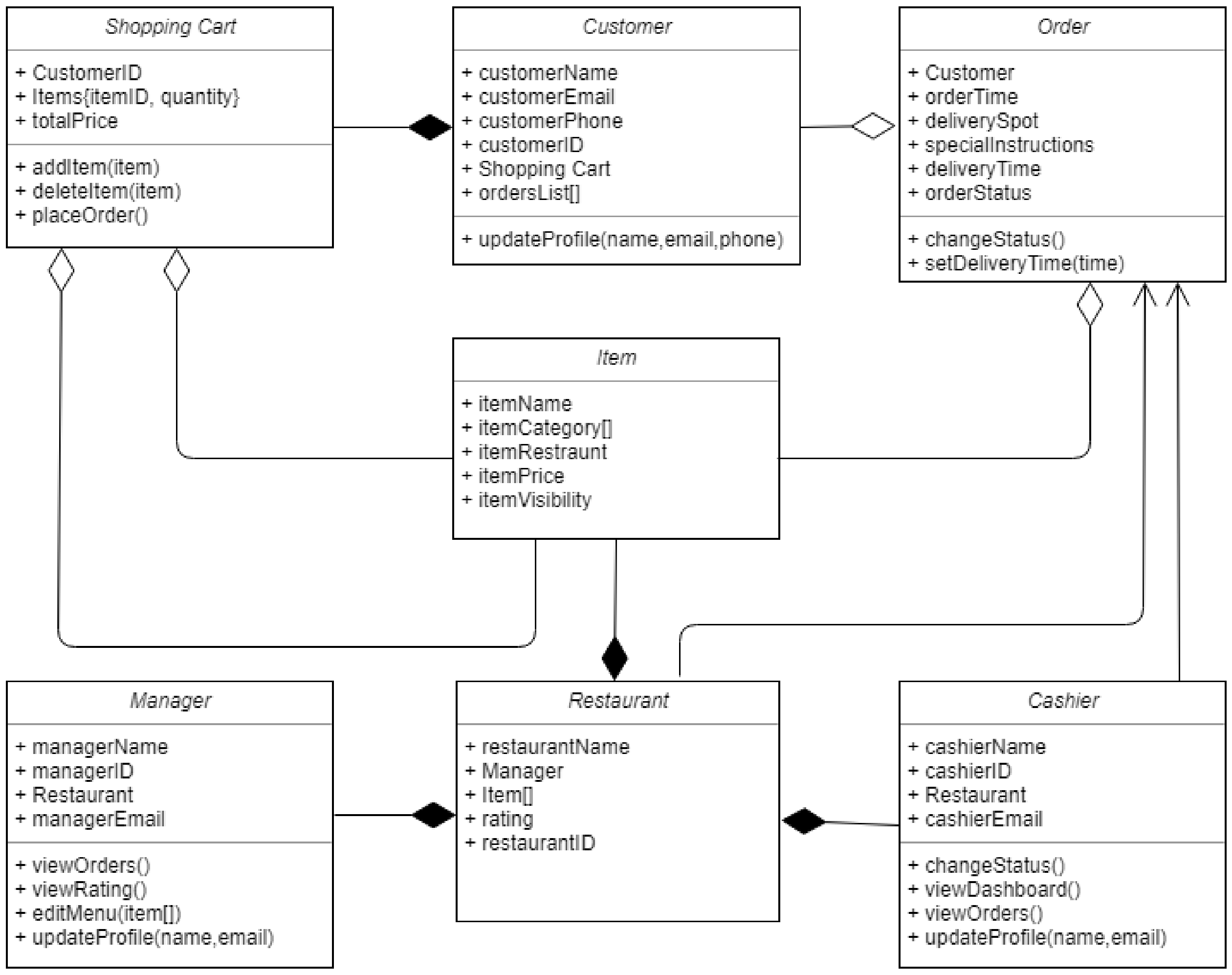
4.2.1.19 Use Case 23: Log out



4.2.1.20 Use Case 23: Forgot Password


Customer
Display
Server
Database
Request forgot password
Display forgot password form
Loop
Enter email accociated with account and password
Send to server
Forward to DB
Validate email address
Invalid Credentials
Invalid Credentials
Display error message on login screen
valid == false
Valid credentials
Send new password to email addr
Send successful message to front end
Display password changed message

4.2.2 Class Diagram



4.3 Data Structure

4.3.1 Internal software data structure
Data structures that are passed among components of the software are described.

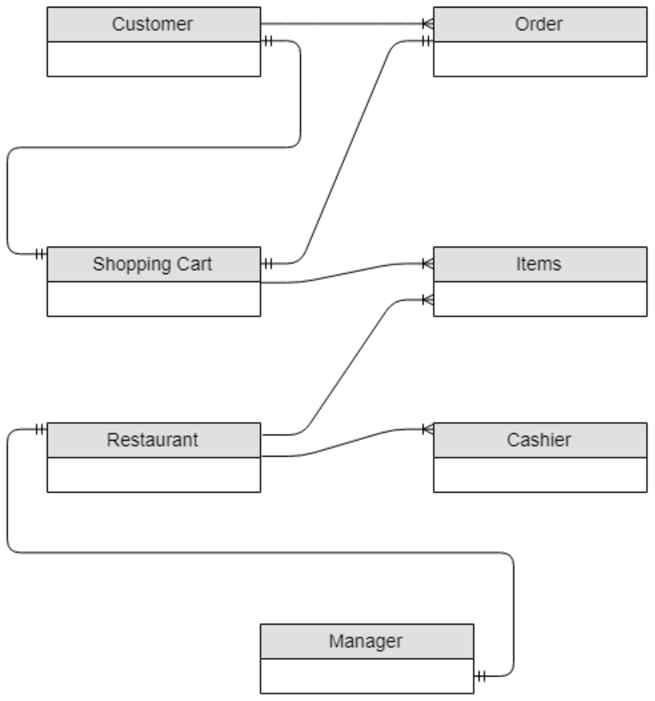
4.3.2 Global data structure
Data structured that are available to major portions of the architecture are described.

4.3.3 Temporary data structure
Files created for interim use are described.

4.4 Database Model

Campus Eat maintains a NoSQL MongoDB. This scheme uses JSON objects to store data. Our application relies heavily on the data it stores from its users and service providers. Our database will be keeping the following information:

- Customer Details
- Restaurant's Details
- Updated Menus for Restaurants
- Order Details



4.4.1 Database scheme and detailed description

The customer JSON object would have a unique identifier as "custID" for each customer. This object would also contain a string entry "custName", an integer "custPhone". Each customer object would also maintain a current state of the shopping cart for that customer. All previous orders are stored in "ordersList".

```
customer= {
        "custID":
        "custName":
        "custPhone":
        "shoppingCart":
        "ordersList":
}
```

The menu JSON object would have a unique key value as “itemID”. It contains a string “name”, an integer “price”, a “restaurantID” which is the key of the restaurant which that specific item belongs to. A string “description” holds the description for that item. A category is a list of keywords to make search easier. It also contains a rating variable kept as a float datatype.

```
menuItem= {
        "itemList":
        "name":
        "price":
        "restaurantID":
        "description":
        "category":
        "rating"
}
```

The order JSON object has a unique key to ensure lookup of a specific order. It also contains the “custID” which has the ID of the customer that particular order belongs to. The “restaurantID” is the key for the restaurant that order belongs to. The content is the actual order along with its quantity and menu item. The instructions is a string which hold the specific instructions of the order as wanted by the customer. Order time is the time that order was placed at and stored as a string. “orderDate” variable hold the date that order was placed on. The cost is the total amount of all the items on that order. The “deliveryTime” holds the estimated time of delivery for that particular order as updated by the cashier. The “deliverySpot” is the string that the customer enters for delivery.

```
order= {
        "orderID":
        "custID":
        "restaurantID":
        "content":
        "instructions":
        "orderTime":
        "orderDate":
        "cost":
        "deliveryTime":
        "deliverySpot":
}
```

The cashier JSON object has its own unique identifier to find that specific cashier. The string variable “cashierName” has the name of that cashier and “cashierEmail” string containns the email used to log in to the application. The “restaurantID” has the key ID of that restaurant at which this cashier works at.

```
cashier= {
        "cashierID":
        "cashierName":
        "cashierEmail":
        "restaurantID":
}
```

The manager JSON object has its own unique identifier. The string variable “managerName” has the name of that manager and “managerEmail” string contains the email used to log in to the application. The “restaurantID” has the key ID of that restaurant which this manager supervises.

```
manager= {
        "managerName":
        "managerID":
        "managerEmail":
}
```

The restaurant JSON object contains a string “restaurantName” to hold the name of that restaurant. This object also has its own unique key in “restaurantID”. An integer variable contains the phone number in “restaurantPhone”. This object contains the unique key of the manager which supervises that restaurant. It also contains a float variable to hold the rating of that restaurant out of 5. It also contains the menu of that restaurant as advertised respresented as a list of menu items.

```
restaurant= {
        "restaurantName":
        "restaurantID":
        "restaurantPhone":
        "managerID":
        "rating":
        "menu":
}
```

4.4.2 Database

MongoDB was chosen as our data storage architecture because it is far simpler and more flexible than using a SQL database. Because it uses JSON object format to store data, data is easily accessed, updated, and linkage of data does not require memory-intensive join operations.

Additional field can easily be added without any change to the rest of the schema - this is important because future accommodation to changing requirements becomes simpler. Some of us also have an experience in working with MongoDB.

The JSON document structure is extremely optimized for programming languages, especially JavaScript, which is the programming language we shall use. Iteration, data extraction, and other data operations like aggregation and map reduce are also optimized.

4.5 External Interface Requirements

4.5.1 User Interfaces

This system will be a web application that will utilize web browsers and web technologies to allow our users to perform tasks in real-time. Our web application will provide a graphical user interface. All operations on the application will use the functionality of buttons, forms, checkboxes, textboxes, dropdown menus and navigation bars. Three types of users will be using this application and the characteristics of each interface between our web application and the respective user are described below:

4.5.1.1 User 1: Customers; LUMS Community who will be using our application to order food

Upon visiting the Campus Eat website, options will be available to login to an existing account or create a new one. If the user chooses to create a new account, then a screen asking for various details of the user such as name, contact number, email etc. will be visible to the them. The user shall enter relevant details and then proceed to login to their newly created account. Once the user has logged in, a webpage shall open with a list of restaurants are open and ready to deliver. The names and logos of these restaurants shall be clearly visible allowing users to easily recognize the restaurant. As soon as the user selects a specific restaurant, another scrollable web page will appear on the user's screen showing the entire menu of that restaurant. Each menu item will have a name and price to identify it. The user will be able to add individual items to their shopping cart by making selections on the menu screen. This shopping cart will be loaded alongside the user making their selections and shall compute a total bill. The user will also be able to add any additional details in their order within the shopping cart. From here on, the user will be able to confirm or cancel the order. Upon confirmation, the status of the order shall be visible, as and when it is updated by the restaurant. Upon cancellation, the menu screen shall be loaded again.

4.5.1.2 User 2: Restaurant Cashier

Campus Eat team will be making accounts for all the restaurants in LUMS and will be giving them their respective account details. When the cashier will log in, a web page with a list of all the orders with the details (the food items ordered, customers details etc.) will be visible. This user will be able to update the status of every individual order on this screen. The status can be 'Processing' - to signal that the order has started to be prepared, or 'Delivered' - to signal that the order has been sent out for delivery.

4.5.1.3 User 3: Restaurant Manager

The interface for the restaurant management will be similar to the interface for the cashier, with a web page containing a list of all the orders with the details will be visible, but the user will not be able to update the status of any order. It will come along with an extra feature of editing the menu. They will be able to update prices, add new menu items and set the daily menu of the restaurant through the page. Upon saving these changes, the changes will be updated on the system and they will be redirected back to the main web page showing the respective restaurant's orders.

4.5.2 Hardware Interfaces

Any digital device which will allow our users to access our web application via Internet, is needed. A web browser, preferably google chrome, will be required. A keyboard for data entry and mouse for selecting operations will be required. However, touch-screen devices such as smartphones and tablets can be used as well.

5 User Interface Design

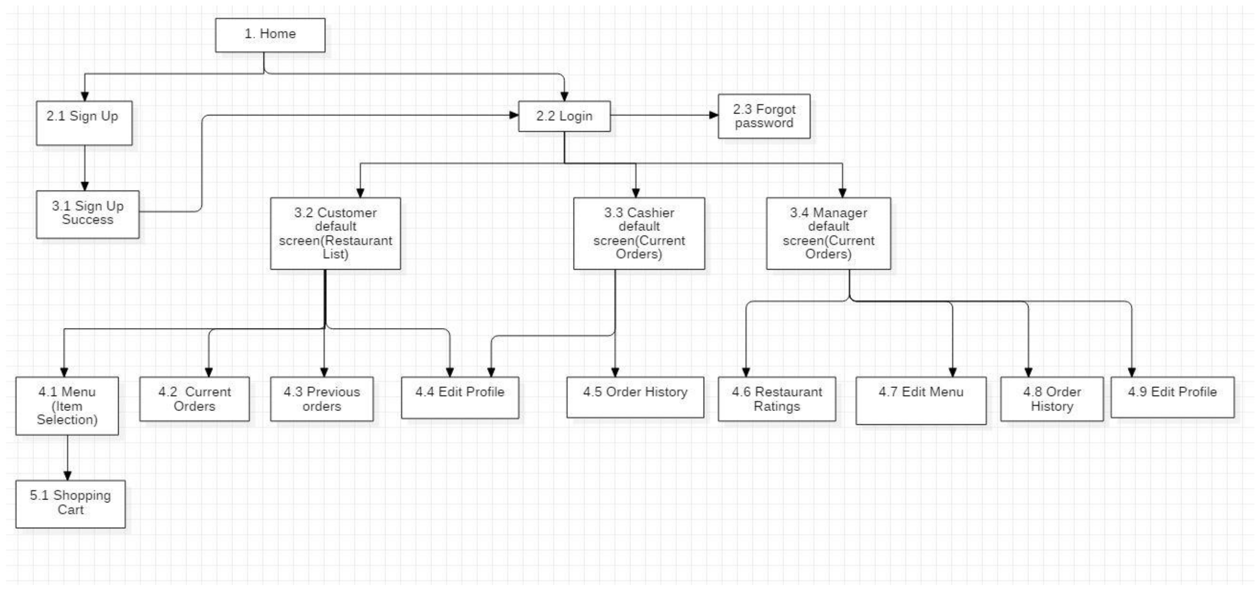
5.1 Description of the user interface

JavaScript library ReactJS will be used for the front end development of the system. It enables developers to declaratively describe their user interfaces and model their state. The developers do not need to describe the transactions on interfaces but only the final state said interaction leaves the system and interface in. Effectively, we only need to tell React what to do and it will create the interface itself. We do not need to worry about how to reflect these changes, or even manage when to make the changes. React will simply react to the state changes and automatically update the DOM when needed. Its integration with HTML makes it an attractive option for front end development of a web application, along with the advantage of having reusable components in the form of functions.

We used the online application, MarvelApp (https://marvelapp.com) to design all our screen layouts. The website had many prebuilt components that we could simply drag and drop to our screens and make adjustments to them according to our design requirements.
Overall, the application has a clean, easy-to-use and minimalistic interface. We did not use many colors and made sure to keep high contrast between text and the background for easy readability.

Most of the menus and screens will have GUI components such as plain icons, text fields and radio buttons. Users will be able to navigate between screens and options in a very efficient manner. The flow of screen pages and the details of each screen are presented in the upcoming sections. Users can navigate between the screens using the menu icon on the top left corner.

5.2 Information architecture

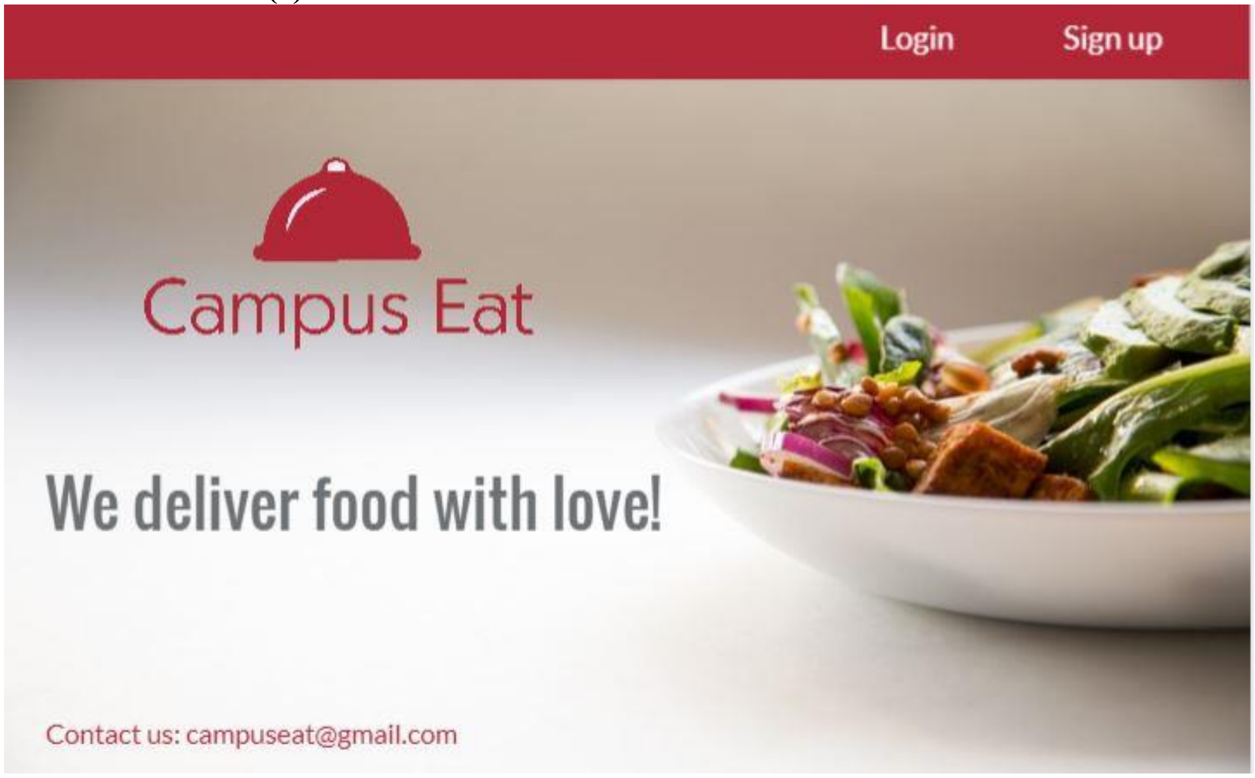


Upon accessing the website, each user is presented with the same home page. They are given an option to login or sign up. Users choosing to sign up will be allowed to create student level accounts only where they can place orders. The accounts for restaurant manager and cashiers will be prebuilt by us and these login details will be shared with them. Upon login, different screens will be displayed to each of the users (students, managers and cashiers) based on their account type. After login, every screen has a side options menu with links to all the pages which that user will need. These side menu options too are different for each user depending on their type.

5.3 Screens

To do: **Screen images** with description – explain **each item on the screen** e.g. button, text field, etc.
Explain how the screen is mapped on one of the user requirements/use cases

5.3.1 Home Screen (1)

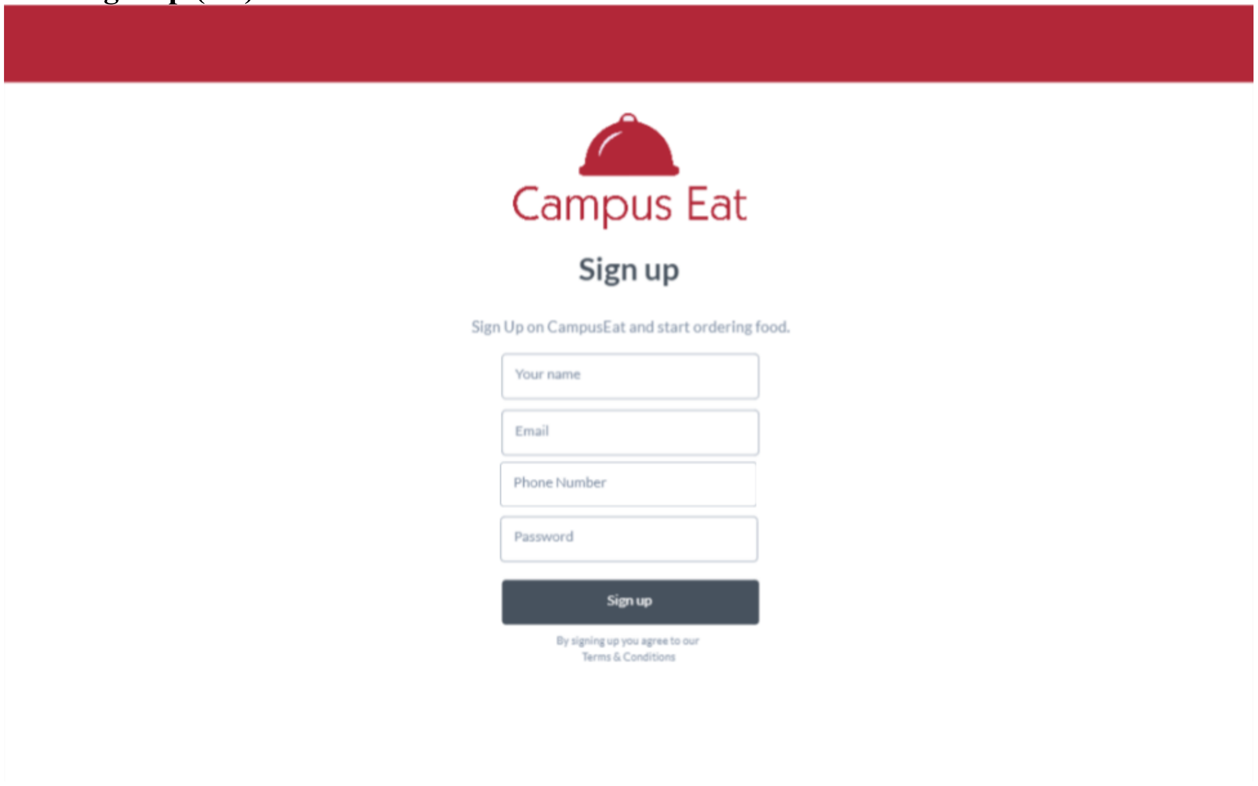


This screen presents a basic layout of our web application. It gives users the option to *Login* (in case the user already has an account) or *Sign up* (in case the user has no existing account and wants to create an account to order food). The Sign up option is for the customers only as the team of Campus Eat will be making an account for Restaurant Manager and the Cashiers before-hand. The bottom of the home screen provides the contact information (email) of the *Campus Eat* team, in case a user has any queries or complaints.

This screen provides the basis for our two major functional requirements:

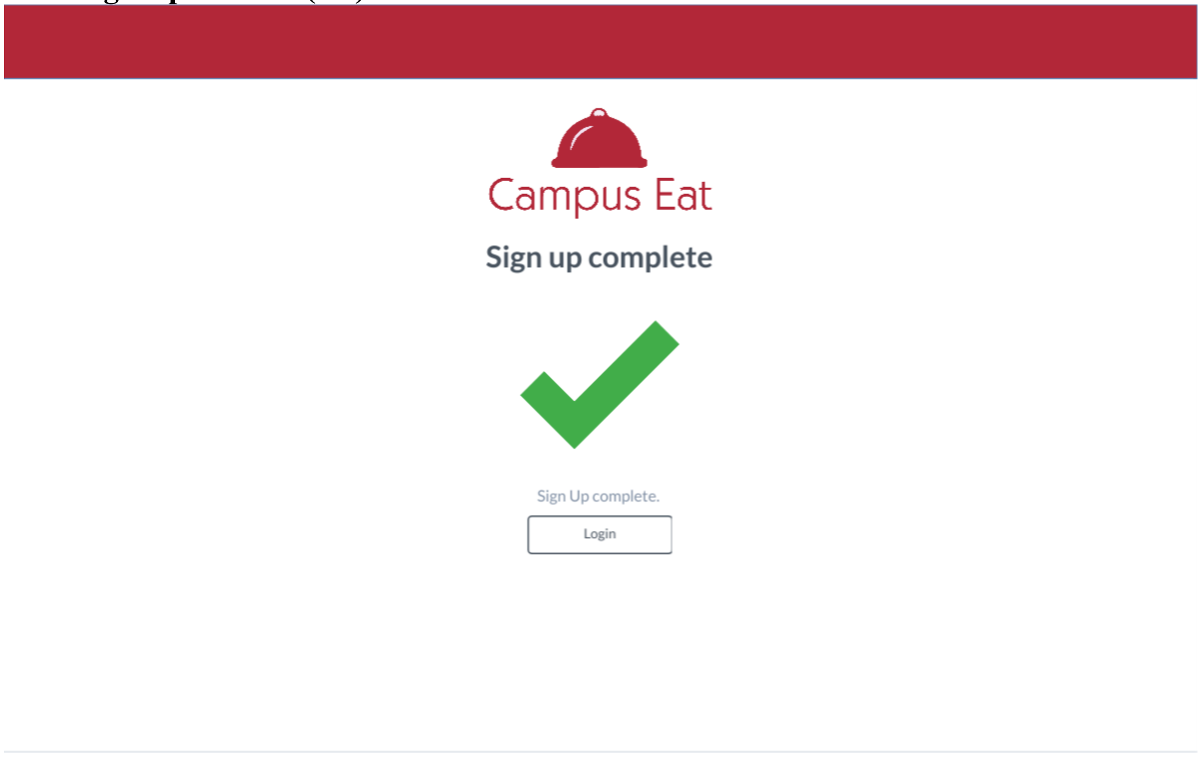
1. RQ<1>-Customer Registration
2. RQ<2>-Login

5.3.2 Sign Up (2.1)


Campus Eat
Sign up
Sign Up on CampusEat and start ordering food.
Your name
Email
Phone Number
Password
Sign up
By signing up you agree to our Terms & Conditions

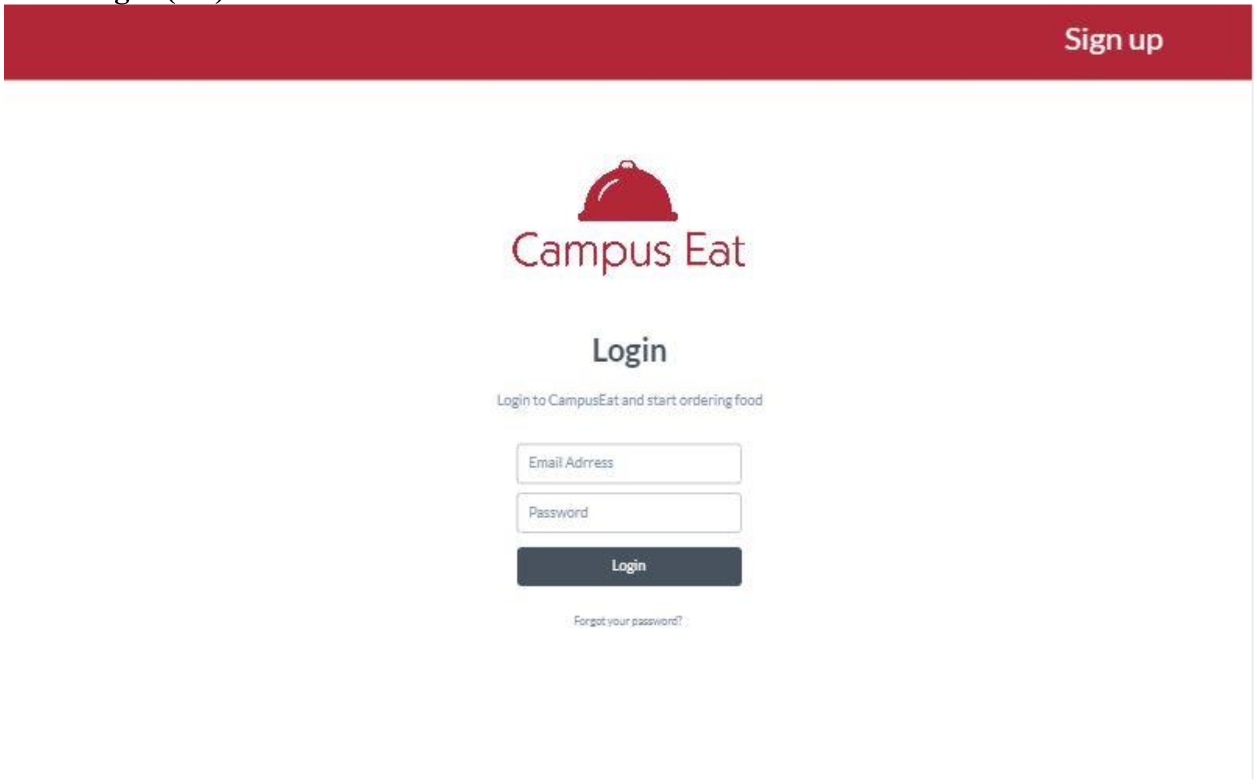
Upon choosing the option to sign up the users will be presented with this page. The users will be prompted for their details including their Name, Email, Phone number and a password. If an account already exists, the user will be presented with the same page again with the error. If sign up details are valid the users will be directed to a sign up successful screen (3.1).
This screen caters to our RQ<1>: *Customer Registration.*

5.3.3 Sign Up Success (3.1)



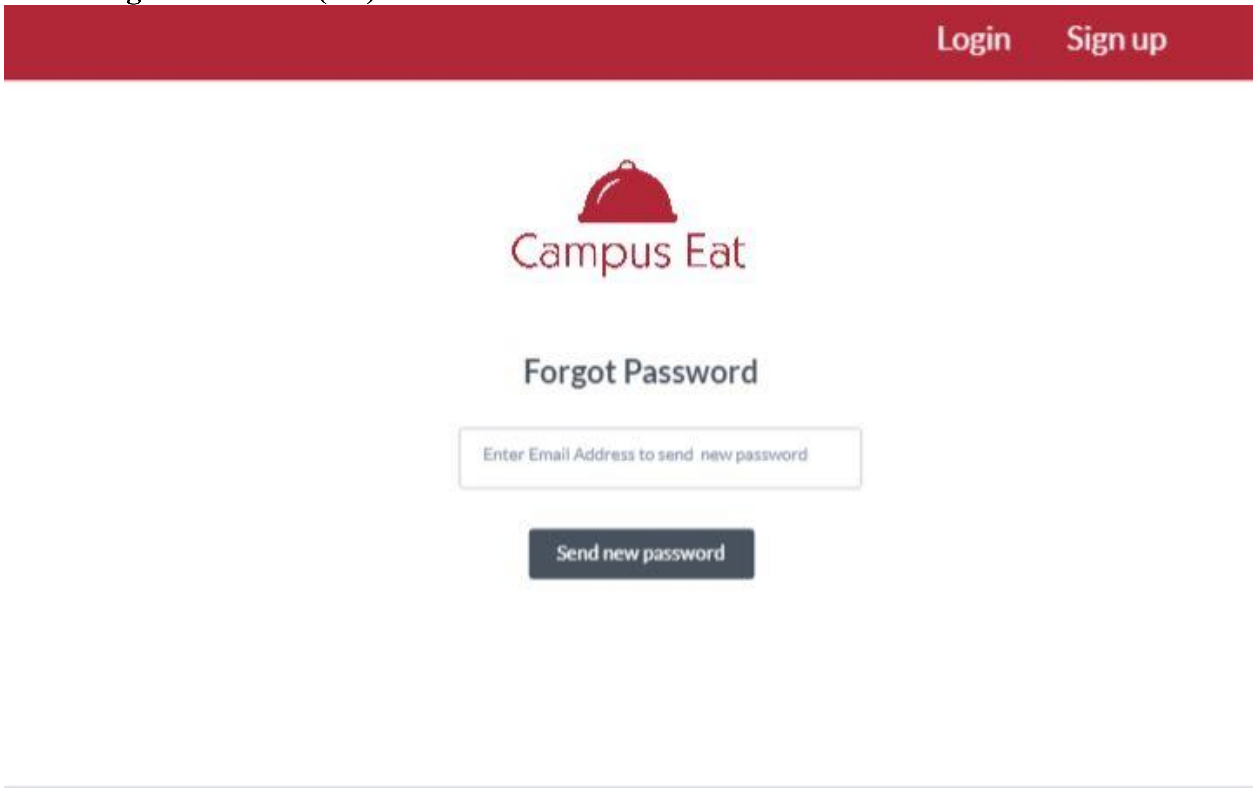
This screen is a static screen loaded after a user successfully signs up. It contains a link to login page to allow the user to login to their newly created account.

5.3.4 Login (2.2)



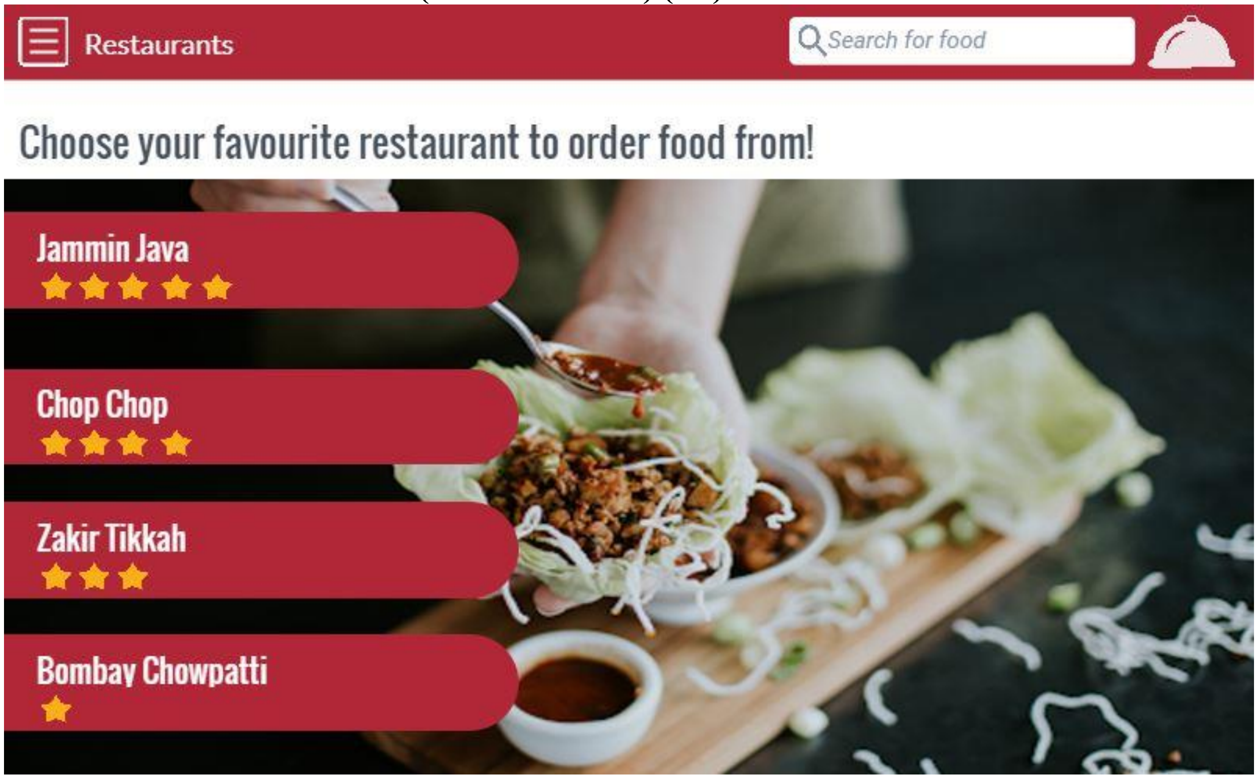
This screen refers to our second functional requirement which allows the Customer, Restaurant Cashier and the Restaurant Manager to login if the user has an existing Campus Eat account. The user will be required to enter the email address and the password in the two input fields in this screen to login. The user can press the login button on the screen after entering the email address and the password to process the login request. This screen also provides the user a “Forgot your password?” option. This option can be clicked in case the user forgot password. The Sign up option on the top right gives user an option to be navigated to the Sign up screen if they don’t have an existing account and pressed the Login option on the home screen accidently.
This screen fulfills our RQ<2>: Login

5.3.5 Forgot Password (2.3)



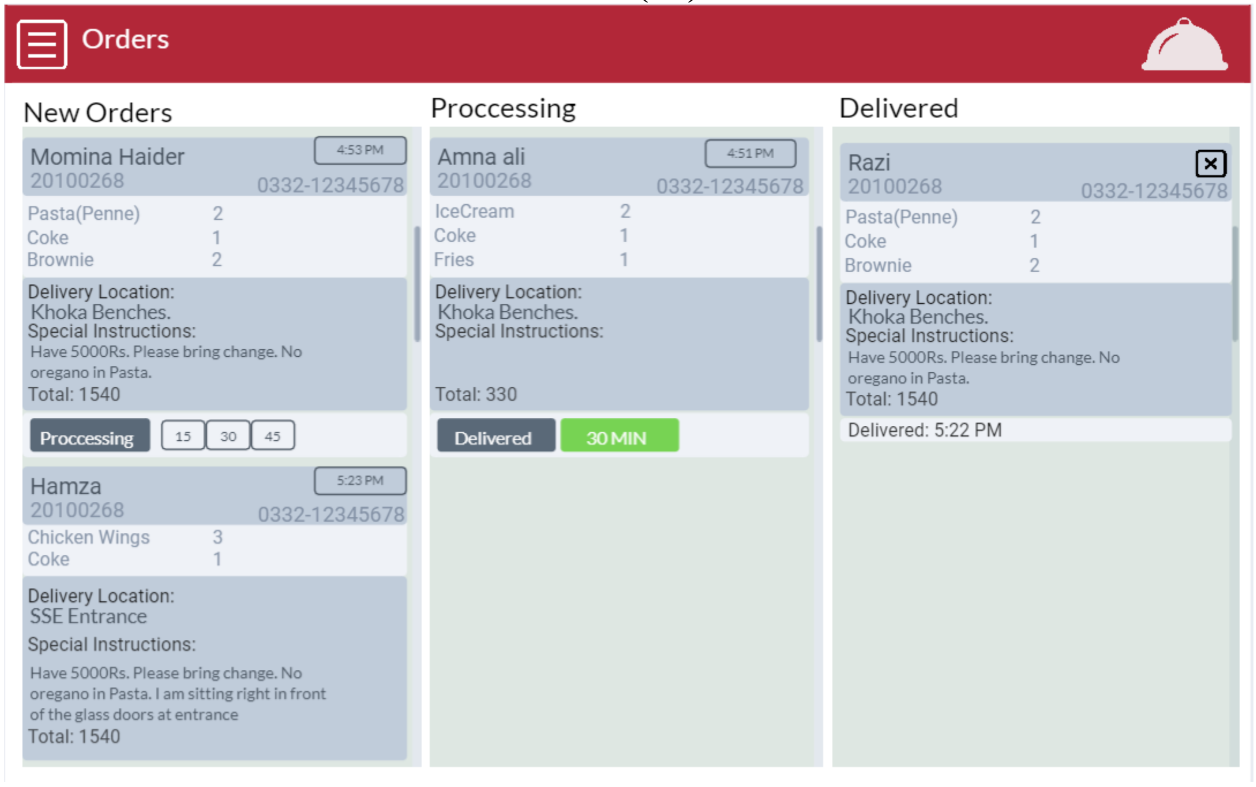
The user is directed to this screen if he/she clicks on the Forget your Password option on the Login Screen. The "Login" and "Sign up" clickable links on the top right allows the user to go back to the "Login" or "Sign up" screen respectively in case the user remembers password or was directed to this screen accidently. This screen has only one input field which requires the user to enter the Email address. Once the user has entered the email address, the "Send new password" button below allows the user to confirm so that the request can be processed. The new password will be sent to user's email address and this way the user will be able to login. This screen presents our new functionality that we decided to add – RQ<24> in this document; it does not refer to any of the functional requirement in the SRS document.

5.3.6 Customer default screen (Restaurant List) (3.2)



After successful sign up or login, our Customer will be navigated to this screen. This is our customer's default screen which will allow them to choose their desired restaurant in LUMS they want to order food from. The "Restaurants" written on the top left gives customers the name of the screen. The *Search for food* input search field allows the customer to type any food dish the customer wants after which the restaurant which offers that food item along with other details such as price will appear. This screen provides the Customers with four major clickable options basically the names of the restaurants in LUMS that deliver. There are four clickable bars and each bar contains the name of the restaurant and it's rating below. The restaurant with the highest rating is shown on the top while the one with the lowest below.
This screen primarily fulfills RQ<7>-Search and RQ<16>-View Ratings Description.

5.3.7 Current Orders- Cashier Default Screen (3.3)


Orders

New Orders

Momina Haider 4:53 PM
20100268 0332-12345678
Pasta(Penne) 2
Coke 1
Brownie 2
Delivery Location:
Khoka Benches.
Special Instructions:
Have 5000Rs. Please bring change. No oregano in Pasta.
Total: 1540
Proccessing 15 30 45

Hamza 5:23 PM
20100268 0332-12345678
Chicken Wings 3
Coke 1
Delivery Location:
SSE Entrance
Special Instructions:
Have 5000Rs. Please bring change. No oregano in Pasta. I am sitting right in front of the glass doors at entrance
Total: 1540

Proccessing

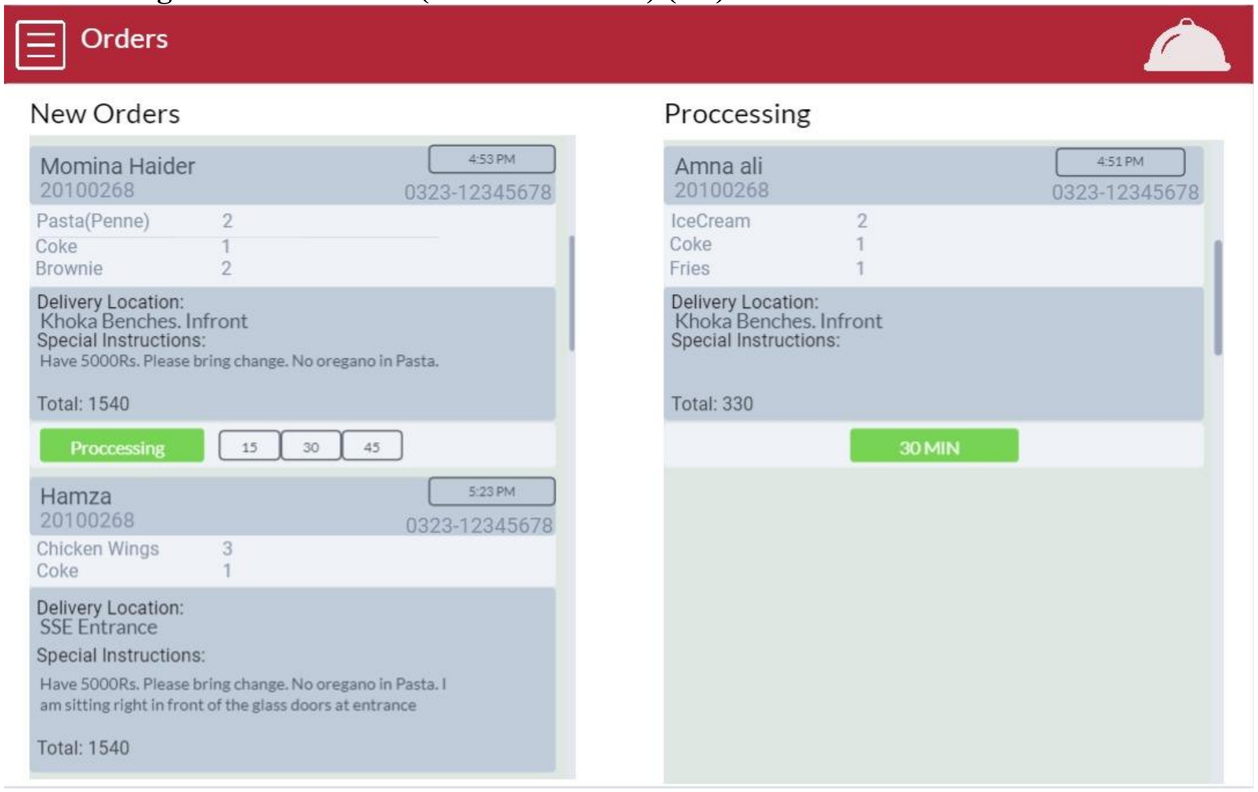
Amna ali 4:51 PM
20100268 0332-12345678
IceCream 2
Coke 1
Fries 1
Delivery Location:
Khoka Benches.
Special Instructions:
Total: 330
Delivered 30 MIN

Delivered

Razi
20100268 0332-12345678
Pasta(Penne) 2
Coke 1
Brownie 2
Delivery Location:
Khoka Benches.
Special Instructions:
Have 5000Rs. Please bring change. No oregano in Pasta.
Total: 1540
Delivered: 5:22 PM

This screen has 3 separate scrollable sections. The leftmost section has new orders from the user. The cashier can click on estimated delivery time for each order. Then, on clicking the processing button, the order will move to the middle sections which contains the orders currently in process. In the processing section, once the order gets prepared and delivered the cashier clicks the delivered button. Upon doing this, the order disappears from the user's current orders screen and appears in the past orders screen for customer. In this screen the order moves to the rightmost screen. The delivered section is empty each time the page is loaded.
This screen caters to our RQ<13>: *Receive Order(s),* RQ<14>: *Update Order Status,* RQ<15>: *Specify Time To Deliver.*

5.3.8 Manager Default Screen (Current Orders) (3.4)


Orders

New Orders

Momina Haider
20100268
4:53 PM
0323-12345678

Pasta(Penne)	2
Coke	1
Brownie	2

Delivery Location:
Khoka Benches. Infront
Special Instructions:
Have 5000Rs. Please bring change. No oregano in Pasta.

Total: 1540

Hamza
20100268
5:23 PM
0323-12345678

Chicken Wings	3
Coke	1

Delivery Location:
SSE Entrance
Special Instructions:
Have 5000Rs. Please bring change. No oregano in Pasta. I am sitting right in front of the glass doors at entrance

Total: 1540

Proccessing

Amna ali
20100268
4:51 PM
0323-12345678

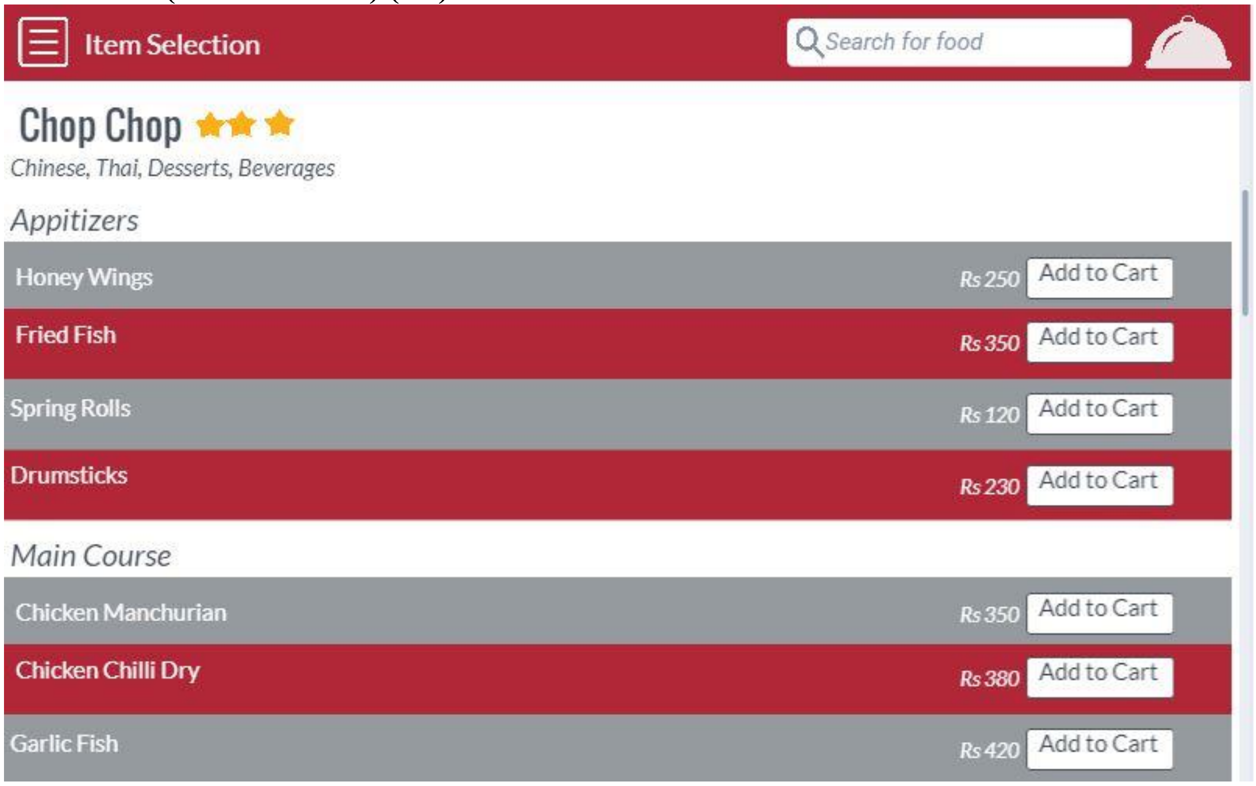
IceCream	2
Coke	1
Fries	1

Delivery Location:
Khoka Benches. Infront
Special Instructions:

Total: 330

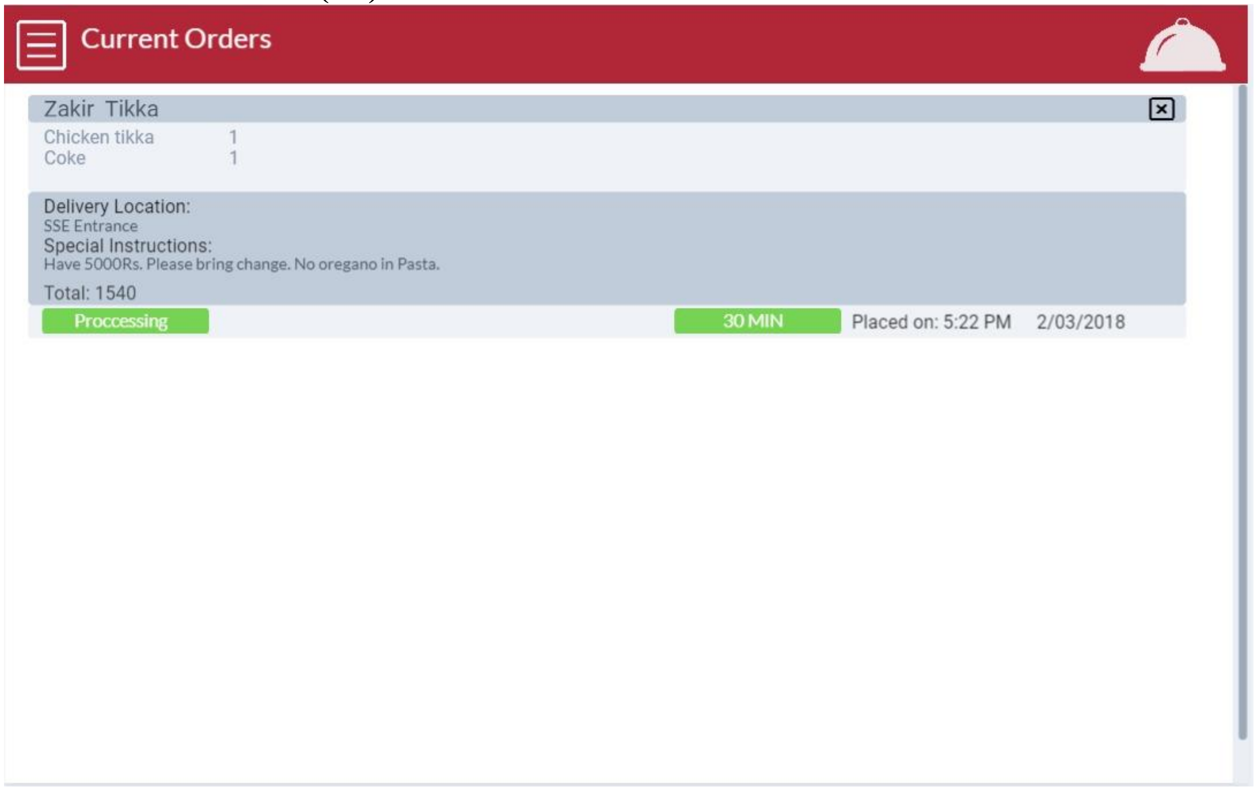
This is the Restaurant's Manager Default screen which displays the *New Orders* and the *Processing* side by side. These are two sub-scrollable screens in this screen which will allow the Manager to see the orders that have been placed but not processed and the other one which displays the orders that have been processed. The new orders sub screen displays the details of the customer and the description of the orders: food items, delivery location and the total amount. The Processing sub-screen will give the Manager the details of the order that have been processed and updated by the Cashier and the estimate delivery time the Cashier has assigned to this order. This is majorly a display screen where the Manager can only view the orders that have been placed by the customer and the details of the order that have been processed by the Cashier. The top left side navigation will allow the Manager to navigate to other web pages.
This screen fulfills our RQ<13>-*Receive Orders.*

5.3.9 Menu (Item Selection) (4.1)


Item Selection
Search for food
Chop Chop
Chinese, Thai, Desserts, Beverages
Appitizers
Honey Wings Rs 250 Add to Cart
Fried Fish Rs 350 Add to Cart
Spring Rolls Rs 120 Add to Cart
Drumsticks Rs 230 Add to Cart
Main Course
Chicken Manchurian Rs 350 Add to Cart
Chicken Chilli Dry Rs 380 Add to Cart
Garlic Fish Rs 420 Add to Cart

This screen will appear after the Customer selects a particular restaurant to order food from. The top left shows the name of the screen "Item Selection" and the Search input field on the top right allows the user to type any particular dish that the customer is looking for in this particular restaurant. The name, ratings and the specialty of the restaurant will be mentioned on top. This is a scrollable screen which will display the menu of that particular restaurant. The menu will be divided into different categories as shown in the screen above. For every category, there will be different food items represented in bars. The extreme right of each bar will show user the price of that particular item and the "Add to Cart" button will allow the user to order that particular food item. This is a scrollable that will allow the users to view the restaurant's entire menu and select the food they want to get delivered. The end of the screen will contain a "Process" button which will navigate the customer to the Shopping Cart screen. This screen caters to our RQ<5>: *Displaying Menus Description*, RQ<8>: *Add items to Shopping Cart Description*.

5.3.10 Current Orders (4.2)



Current Orders

Zakir Tikka

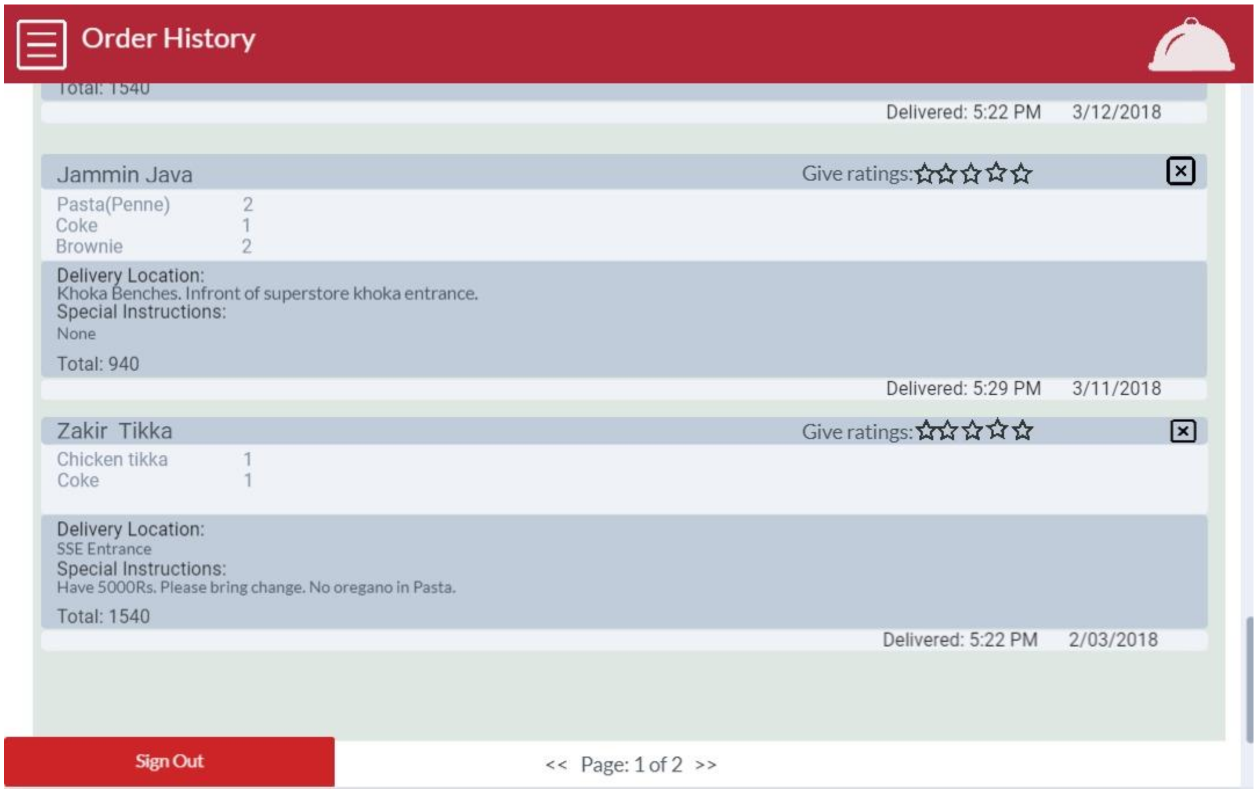
Chicken tikka 1
Coke 1

Delivery Location:
SSE Entrance
Special Instructions:
Have 5000Rs. Please bring change. No oregano in Pasta.
Total: 1540

Placed on: 5:22 PM 2/03/2018

This screen allows the user to view the orders that are placed by the customer and right now are in process. The top left shows the name of the screen “Current Orders” and the side navigation option allows the user to navigate to other screens. This is a scrollable screen which displays the user the details of the current order including the restaurant, food items ordered, delivery location, special instructions and the total bill. The screen also provides the detail of the current status and when the order was placed: the exact time and date.
This screen fulfills our RQ<12>-View Order Status

5.3.11 Order History (4.3)



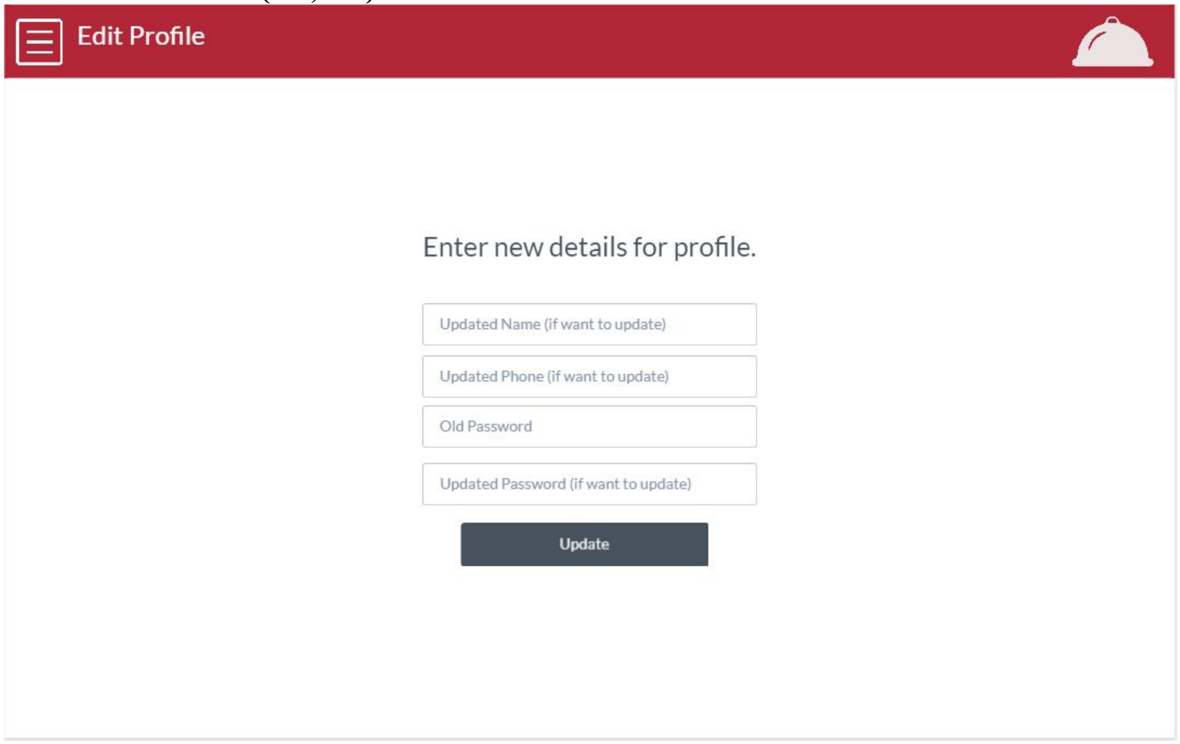
Total: 1540
Delivered: 5:22 PM 3/12/2018

Jammin Java
Pasta(Penne) 2
Coke 1
Brownie 2
Delivery Location:
Khoka Benches. Infront of superstore khoka entrance.
Special Instructions:
None
Total: 940
Delivered: 5:29 PM 3/11/2018

Zakir Tikka
Chicken tikka 1
Coke 1
Delivery Location:
SSE Entrance
Special Instructions:
Have 5000Rs. Please bring change. No oregano in Pasta.
Total: 1540
Delivered: 5:22 PM 2/03/2018

This screen allows the Customer to view the Order History. This is a scrollable screen which can be scrolled down and the customer can change the page horizontally as well. In order to ensure the efficiency of our system, we have made two screens which can be changed by either moving left or right as shown in the bottom of our screen. The user can also move this screen up and down by scrolling. This screen is divided into several bars with each bar illustrating a particular order. The details of every order are described in that bar including the food items ordered, location and the total bill. The bottom right of every bar shows user the delivery details: the time and the date when the order was delivered. Every bar contains a “Give ratings” option. The user can select the stars in this option and rate the restaurant out of 5. In the bottom left is the “Sign Out” option which allows the Customer to sign out.
This screen fulfills our RQ<18>-*View Customer Order History* and RQ<17>-*Give Ratings.*

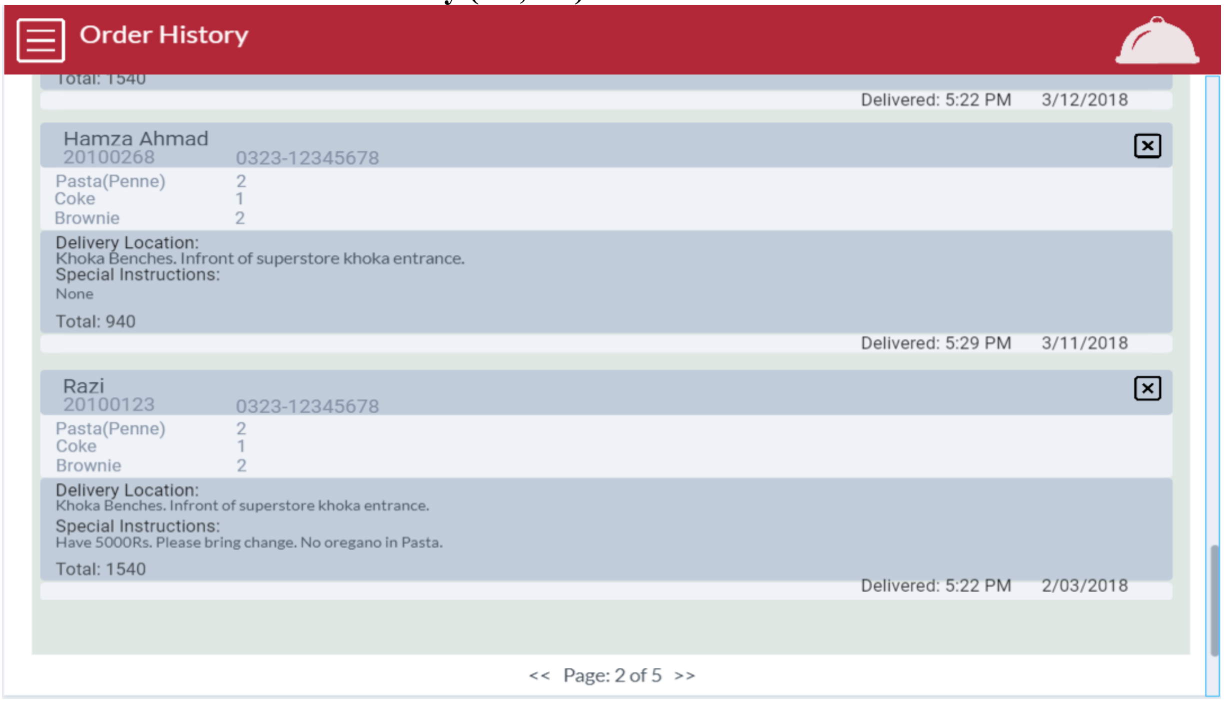
5.3.12 Edit Profile (4.4, 4.9)


Edit Profile
Updated Name (if want to update)
Updated Phone (if want to update)
Old Password
Updated Password (if want to update)
Update

Enter new details for profile.

This screen has 4 text fields allowing the user to update their profile details. The users can leave the input fields they do not want to change, blank. Once the user clicks the "Update" button, these changes will be reflected in the database. The screen is non-scrollable.
This screen caters to our RQ<4>: *Edit Profile.*

5.3.13 Restaurants Order History (4.5, 4.8)



Total: 1540
Delivered: 5:22 PM 3/12/2018

Hamza Ahmad
20100268 0323-12345678

Pasta(Penne)	2
Coke	1
Brownie	2

Delivery Location:
Khoka Benches. Infront of superstore khoka entrance.
Special Instructions:
None
Total: 940
Delivered: 5:29 PM 3/11/2018

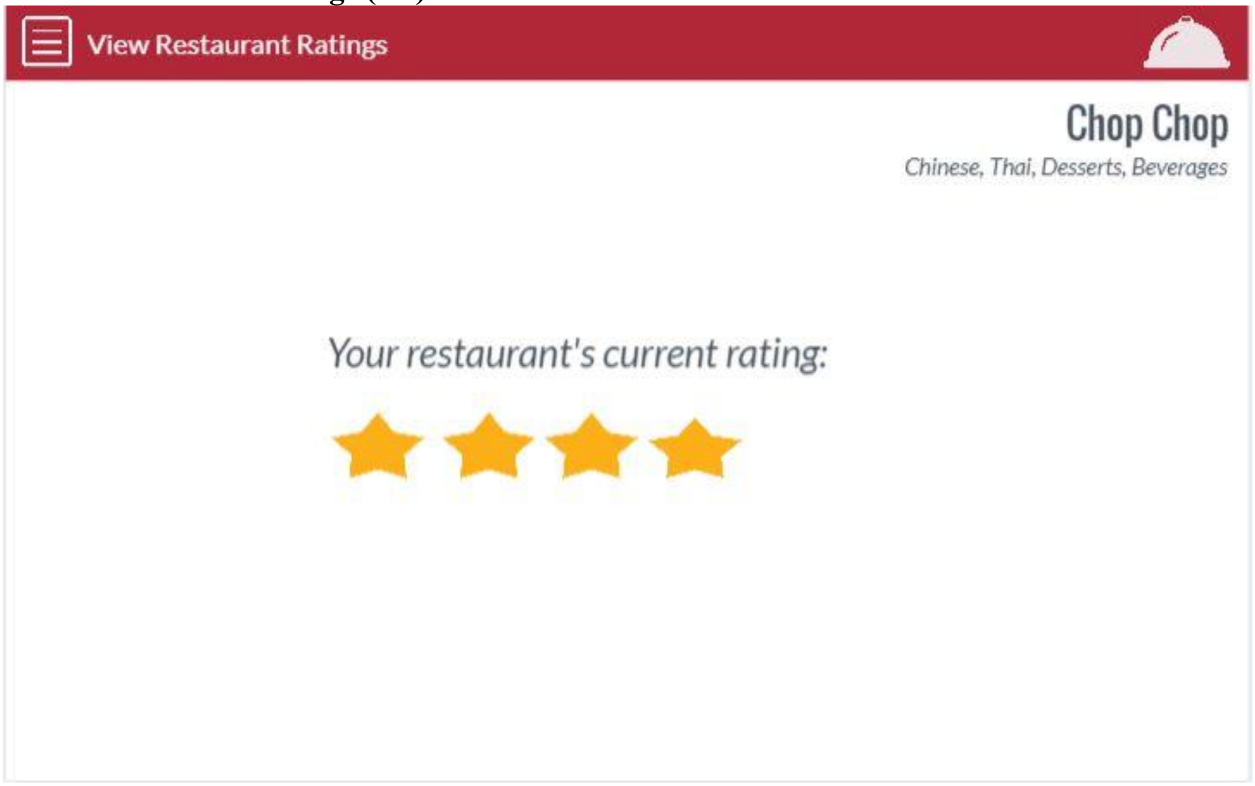
Razi
20100123 0323-12345678

Pasta(Penne)	2
Coke	1
Brownie	2

Delivery Location:
Khoka Benches. Infront of superstore khoka entrance.
Special Instructions:
Have 5000Rs. Please bring change. No oregano in Pasta.
Total: 1540
Delivered: 5:22 PM 2/03/2018

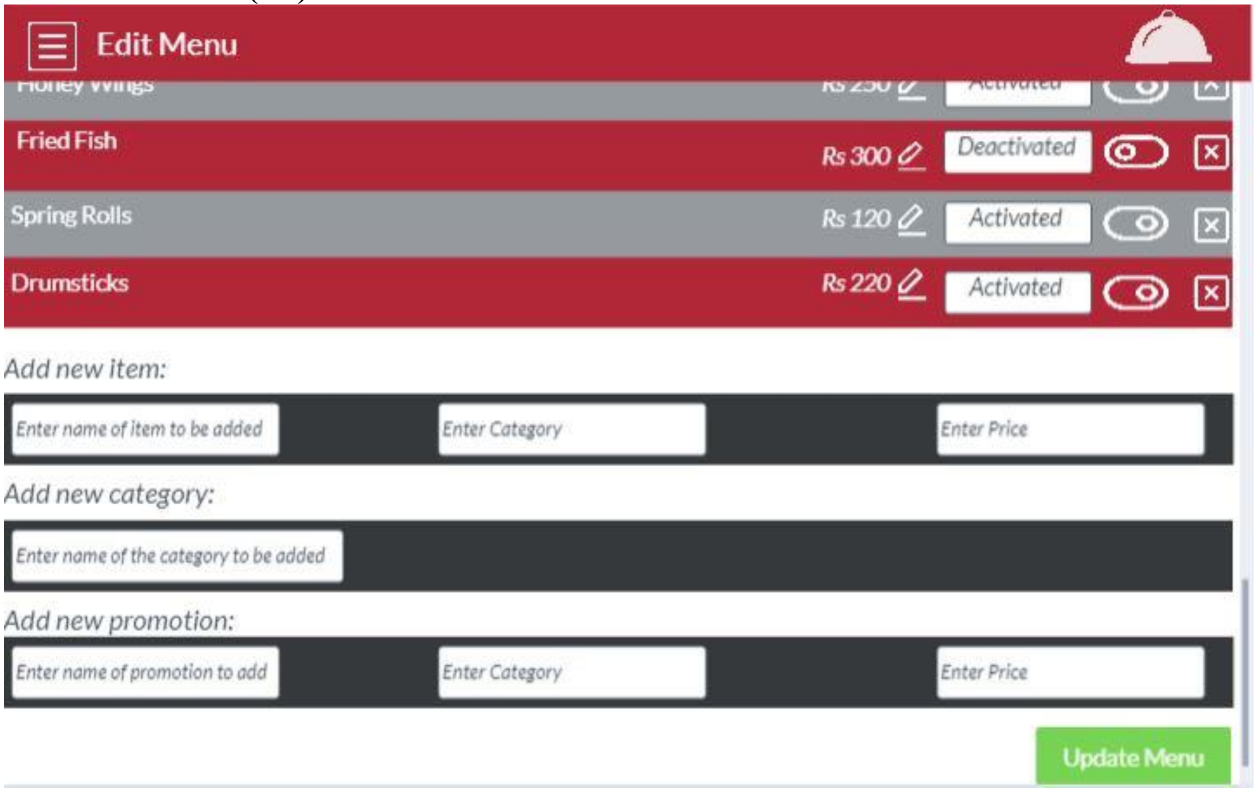
The screen name is visible on the left of the top bar. This is a scrollable screen and is visible to both restaurant cashier and restaurant manager. Each page displays the last 20 orders for that restaurant. To view older orders, the user can navigate to next page.
This screen caters to our RQ<19>: *View Restaurant Order History.*

5.3.14 Restaurant Ratings (4.6)


View Restaurant Ratings
Chop Chop
Chinese, Thai, Desserts, Beverages
Your restaurant's current rating:

This screen will allow the Restaurant Managers to view the current rating of their respective restaurants. The top right will of the screen will display the name of their restaurant along with the description of the items served. The top left of the screen will show the Manager the name of this screen, "View Restaurant Ratings" and the side navigation bar clickable option will allow the Manager to navigate to other pages. The center of the screen will present the ratings of the restaurant indicated by yellow stars with 5 as the maximum stars as mentioned in our SRS. This screen presents our new functionality that we decided to add; hence it is referred to RQ<23> as mentioned in the Change log.

5.3.15 Edit Menu (4.7)

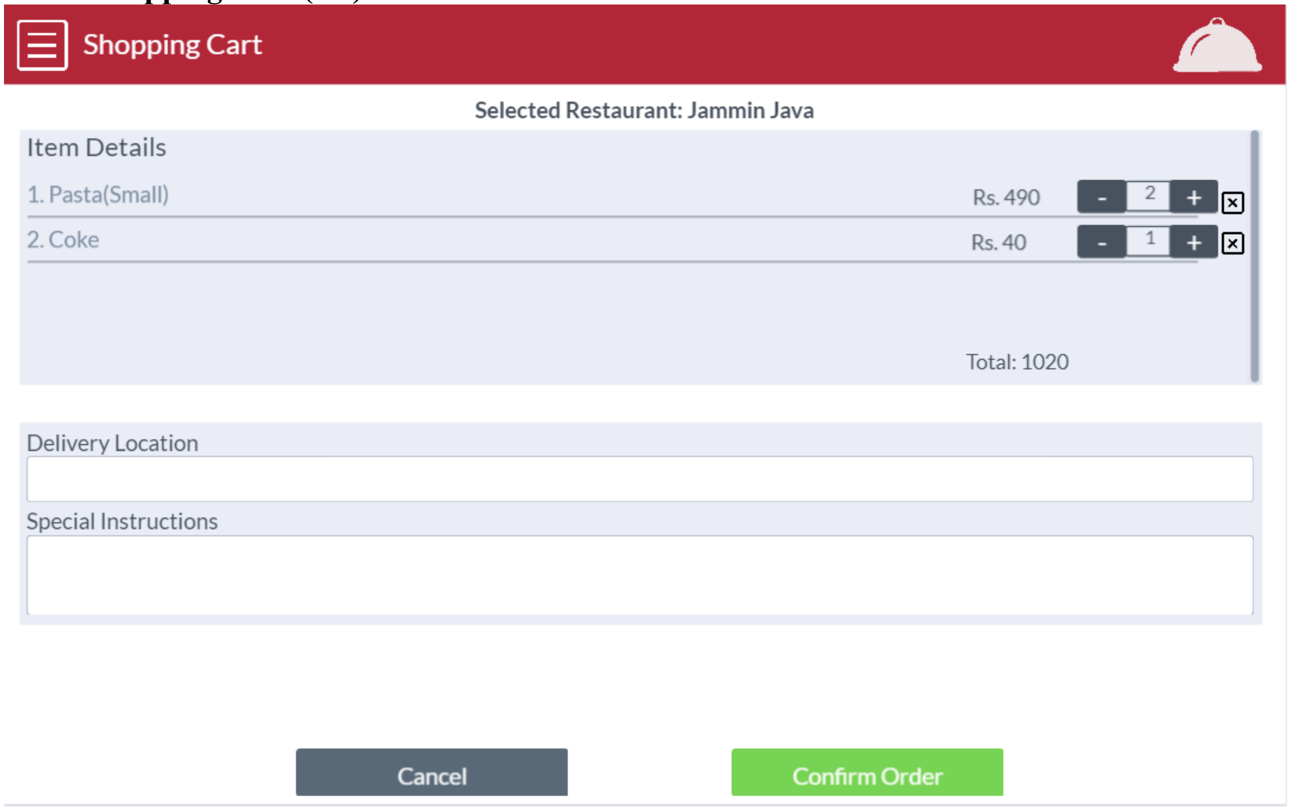

Edit Menu
Honey Wings Rs 250 Activated
Fried Fish Rs 300 Deactivated
Spring Rolls Rs 120 Activated
Drumsticks Rs 220 Activated
Add new item:
Enter name of item to be added
Enter Category
Enter Price
Add new category:
Enter name of the category to be added
Add new promotion:
Enter name of promotion to add
Enter Category
Enter Price
Update Menu

This screen will be available to The Restaurant Manager only which will allow the Manager to edit the menu of the restaurant. This is a scrollable screen with the top left showing the name of the screen "Edit Menu" and the side navigation option which will allow the Manager to navigate to another web page. The screen will be first divided into bars with each bar representing a food item and its details. The edit pencil shown in the screen with the price gives the Manager an option to edit the price of the item.

Moreover, the icon allows the Manager to either activate or deactivate. Activate means that the food item is available right now where as deactivate means that the food item isn't available. The cross option on the extreme right of every bar allows the Manager to remove this particular item from the menu. The bottom of the screen allows the Manager to either add new item, category or promotion. The new item has three input fields which allow the Manager to type the name, category and price of the item. The new category has one input only which allows the Manager to type new category. The new promotion has three input fields where the Manager is asked to type the name, category and price of the promotion. The right below of the screen there is a "Update Menu" button which the Manager can press after making the editions in the menu.

This screen fulfills our RQ<6>: Edit Menu and RQ<21>: Create Promotional Offers.

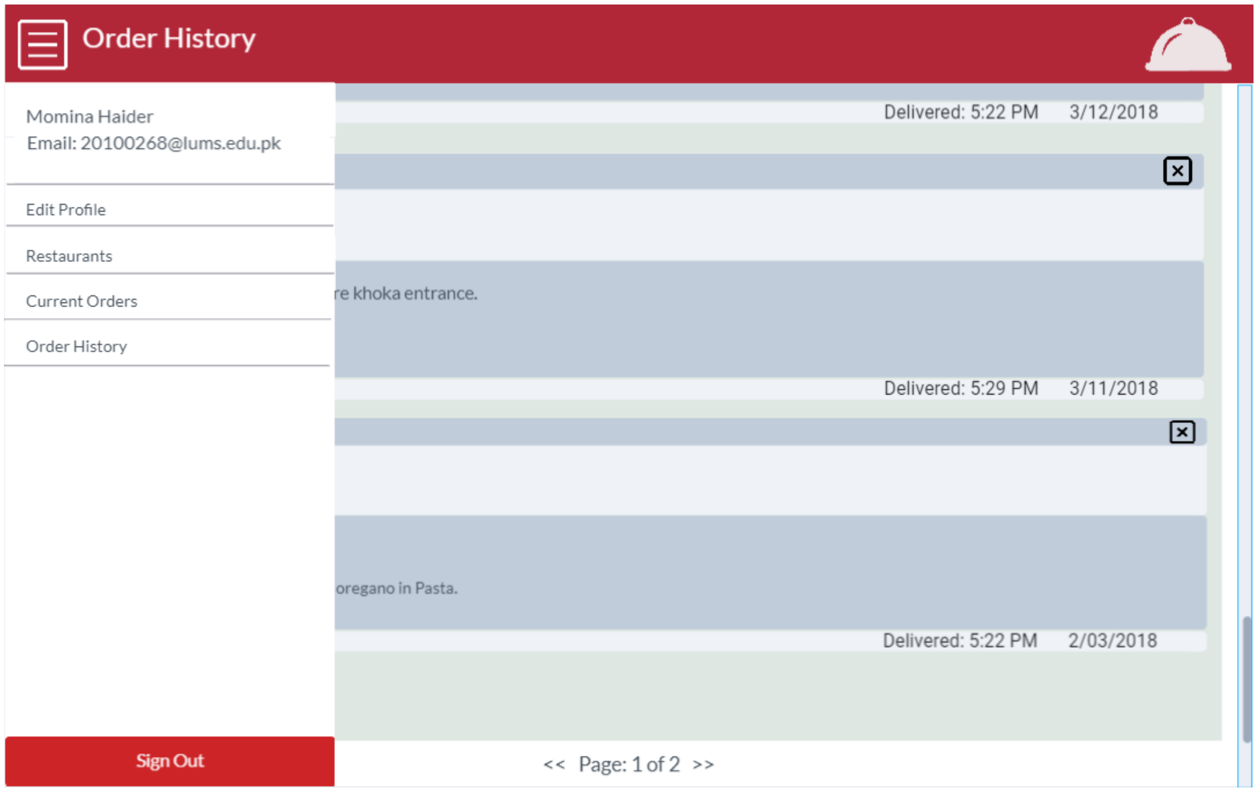
5.3.16 Shopping Cart (5.1)



Upon choosing the "proceed to shopping cart" option in the Menu(Item selection) 4.1 screen the user will be directed to this page. The name of the screen is visible in the left side of the top bar. The screen is divided into 2 main sections. The first section is scrollable. Here the user can select the quantities of selected items, remove items from the shopping cart through clickable buttons. The total payable cash is also displayed to the users on this screen. The second section of the screen presents the users with text fields to specify their delivery location and any special instructions that they might want to convey to the restaurant. Once the users press confirm order, their orders will be sent to that particular restaurant.
This screen caters to our RQ<9>: *Remove Items from Shopping Cart*, RQ<10>: *Delivery Instructions*, RQ<11>: *Checkout.*

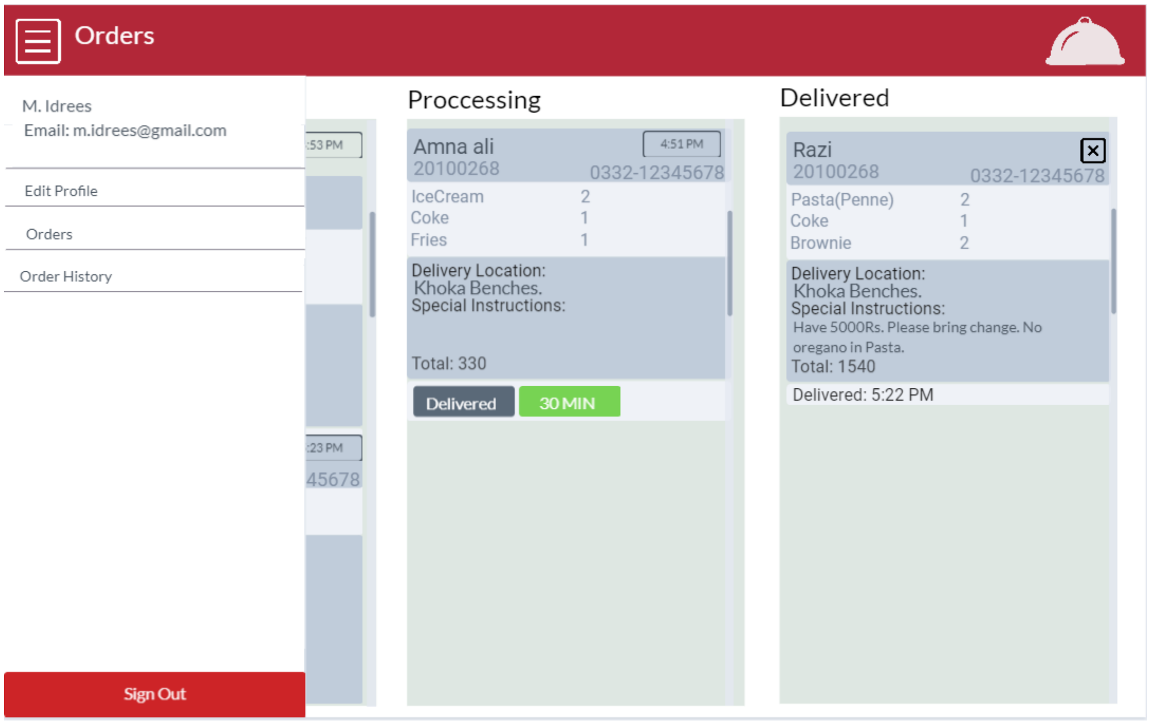
5.3.17 Options menu sidebar

For customers:



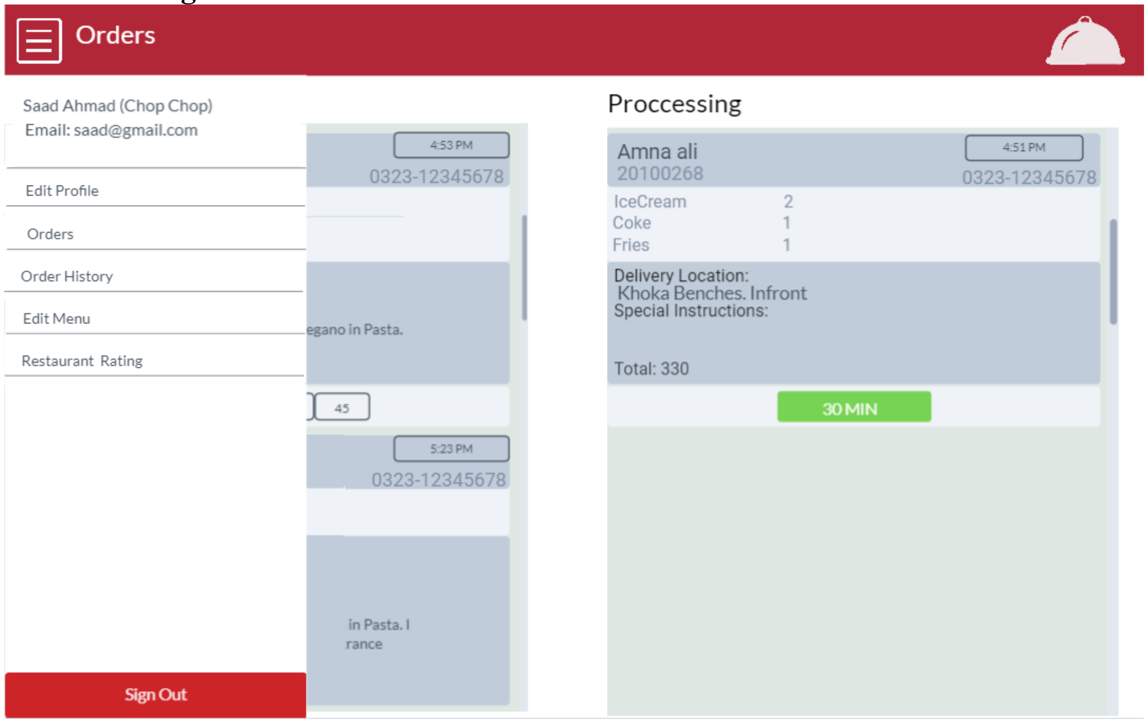
This side menu is available to the customers. They can view their profile details within it. The menu bar also includes the links to Edit Profile, Restaurants, Current Orders and Order History. The students also have the option to sign out from their accounts. This menu is available to users on every screen, once they have signed in.

To the Cashier:



This side menu is available to the cashiers. They can view their profile details within it. The menu bar also includes the links to Edit Profile, Current Orders and Order History. The cashiers too have the option to sign out from their accounts. This menu is available to cashiers on every screen, once they have signed in.

To the Manager:



Saad Ahmad (Chop Chop)
Email: saad@gmail.com

4:53 PM
0323-12345678
egano in Pasta.
45
5:23 PM
0323-12345678
in Pasta. I
rance

Amna ali
20100268
4:51 PM
0323-12345678

IceCream	2
Coke	1
Fries	1

Delivery Location:
Khoka Benches. Infront
Special Instructions:

Total: 330

This side menu is available to the managers. They can view their profile details within it. The menu bar also includes the links to Edit Profile, Orders, Order History, Edit Menu and Restaurant Ratings. The managers also have the option to sign out from their accounts. This options menu is available to users on every screen, once they have signed in.

5.4 User interface design rules

The following rules and standards were kept in mind while designing the UI and the screens:

- **Consistency:** The icons, menu and button positions were kept consistent throughout the screens to make the application easy to use. Similarly, text color and font too is kept consistent throughout all the pages.
- **Readability:** The colors used on screens are catchy and offer high contrast with white text. This has massively improved the readability of the text being displayed on the screen.
- **Visibility:** All the screens available for a user are conveniently placed within the menu on the top left corner for easy access. At every screen, the user can navigate to all possible screens available to them through the menu.
- **Affordance:** It's easy to figure out intuitively that how to use the various options provided on our screens. The users will be able to perform the tasks they want and use the options as soon as they see the screens. For instance, the user will know that in order to avail the option written on the button, they will have to press it.
- **Feedback:** When the user will interact with the screen, there will be an instant reaction. For example, when the user clicks on the button then a new web page will be reloaded,

- The names of the screens are kept simple and meaningful to allow for easy to interface and navigation between the screens. The design is kept minimalistic overall to improve the loading time of the web application
- Information is presented in logical order and the application relies on user's expectations derived from their real-world experiences, thus reducing cognitive strain and making the application easier to use.

6 Other Non-functional Requirements

6.1 Performance Requirements

6.1.1 Orders from each user should appear on the restaurant's screen as soon as the user places the order in real time. This would allow the restaurant owner to keep track of the orders in a chronological order and avoid delays.

6.1.2 The restaurant menus should load as quickly as possible. Preferably, the menus should load within 3 seconds, but can take more time if the menu is really long as in case of Superstore's menu.

6.1.3 The system should be a Progressive web application. It should load in minimal time even on flaky networks, should have icons on home screen and loads as a top-level, full screen experience. This would be particularly useful for students, since not all of them own high-end mobile phones capable of installing applications on them.

6.1.4 The system should be able to return a search query within a window of 3 seconds.

6.1.5 Any changes in the menu and pricing performed by the restaurant management will be visible to the customer in real time.

6.1.6 In case of an internet connection or power failure, no information should be lost; on reboot or reset of connection, all orders will be rendered again instantly.

6.2 Safety and Security Requirements

Campus Eat is a food ordering web application and hence does not require a particularly high level of security. Our primary focus is to keep the user information safe from hackers. User accounts will be password protected and these passwords will be stored in encrypted form in the database. Some of the basic safety and security requirements that we identified in our meetings with our stakeholders are as follows:

6.2.1 Secure protocols, such as HTTPS, should be used to ensure the personal information of a user are not disclosed to anyone else.

6.2.2 No third party will be able to access the passwords of users through the database.

6.2.3 The system should permit only staff members who are on the list of authorized administrators to create or edit menu. One user cannot tamper with the access rights of another user. Restaurant staff, other than the restaurant managers, will only be allowed to view orders on login. Customers will not be able to edit restaurant menus but will be able to view menus and place orders.

6.2.4 If a user tries to log in to the system with an account that is non-existent, they should not be logged in.

6.2.5 If a customer tries to create an account with an email address that already has an existing account, the customer should be prompted that the account already exists.

6.2.6 Users will be forced to set a password having a minimum of 8 characters. These passwords will then be stored in database in form of a hash. This will ensure that even if there is a breach in database access, the hacker will not be able to see the users' passwords.

6.3 Software Quality Attributes

6.3.1 Availability:

1. The system should be available for use when it is needed by the users. While, ideally it should work 100% of the time, it should at least be available for 98% of the time.
2. As long as internet connection is available the system should be able to connect to database for user authentication.

6.3.2 Maintainability:

1. The restaurant managers should be able to regularly update their menus and prices. In case of restaurants where the menu changes daily (for example Chop Chop) this would be necessary to display the correct daily menu for the students.
2. The application should be easy to extend. Implementing new functionality in the future should be made easier by writing code that favors it.
3. To accommodate for extensibility, explanatory comments should be present in the code.

6.3.3 Scalability:

1. The restaurant management should be able to add as many menu items as they want to.
2. The system should be able to scale to a large number of customers (around 10,000).

6.3.4 Testability:

1. A proper test environment should be setup to allow the testing of the various functionalities the application will carry.
2. The system should be easily divisible into multiple modules to facilitate testing.

6.3.5 Usability:

1. The GUI should be easy to use and understand for all the users.
2. The onboarding process of creating an account should be simple. No unnecessary information will be collected for the formation of an extensive profile. Essential details such as Email Address, Name and Phone Number will suffice.

Appendix A - Group Log

The Group met on 20/03/2019 for approximately 4 hours, discussed the document, created a Google doc to simultaneously work on and devised a plan on how to distribute the tasks between the group members.

From 22/03/2019 to 29/03/2019, the group met everyday to discuss progress and work on parts of the documents together. Topics for future meetings were also discussed.

Appendix B – Contribution Statement

Name	*Contributions in this phase*	*Approx. Number of hours*	*Remarks*
M. Razi Ul Haq	*Parts of Section 2 and 3. Section 5 including screens*	*23*	
Syed Hamza Ahmad	*Section 6, Section4 and Parts of section 2.*	*23*	
Momina Haider	*Parts of Section 2 and 3. Section 5 including screens*	*22*	
M. Usama	*Section 4, Database, Class Diagram design and definition*	*20*	
M. Raza Khawaja	*Section 1, Section 4.*	*20*	