LONDON METROPOLITAN UNIVERSITY

Final Year Project on:

FoodBite

Food Ordering Application

Student Name: Niruta Devkota

Student ID: 17030729

Course: BSc (Hons) Computing

First supervisor: Mr. Sandeep Gurung

Second supervisor: Mr. Sushil Paudel

ABSTRACT

The purpose of the this project FoodBite is to develop an ordering food application based on Android with Food Menu, Cart, Ordering, and Setting Profile features. The research method used in this research is water model of System Development Life Cycle (SDLC) method with following phases: requirement definition, analysing and determining the features needed in developing application and making the detail definition of each features, system and software design, designing user experience design, Unified Modelling Language (UML) design, and database structure design, implementation, making database and translating the result of designs to programming language code then doing testing and if any changes and reparations needed then the previous phases could be back. The result of this project is a food ordering application based on Android for customer. The conclusion of this project is to help customer in making order easily, to give detail information needed by customer and provide payment facilities.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to several individuals and organizations for supporting me throughout my Undergraduate study. First and foremost, I wish to express my sincere thanks to my supervisors, Mr. Sandeep Gurung and Mr. Sushil Paudel, for their patience, suggestions, helpful information, practical advice and ideas which have always helped me tremendously in my research and writing of this project. Their vast knowledge, profound experience and professional expertise in the subject field has helped me to complete this project successfully. I am thankful to them for their precious time in guiding me, answering my queries, correcting and improving me during the research and completion of the project. Without their guidance and help, this project would not have been possible.

I am thankful to and fortunate enough to get constant encouragement, support and guidance from my family and friends which helped me in successfully completing this project work.

Niruta Devkota

17030729

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Project Introduction

Online food ordering is the process of ordering food through the restaurant's website or mobile app, or through a multi restaurants website or app. A customer can choose to have the food delivered or for pick-up. The process consists of a customer choosing the restaurant of their choice, scanning the menu items, choosing an item, and finally choosing for pick-up or delivery. Payment is then administered by paying with a credit card or debit card through the app or website or in cash at the restaurant when going to pick up. The website and app inform the customer of the food quality, duration of food preparation, and when the food is ready for pick-up or the amount of time it will take for delivery (Revolvy, 2018).

Food delivery is one of the most commonplace services in the modern world. Combining professionally made food with convenient access from home, food delivery is more popular today than at any other time in history. Interestingly, food delivery may be enormously popular now, but this idea has existed in rudimentary form for thousands of years (Harvey, 2019). It would not be an exaggeration to say that food delivery applications have made the restaurant industry democratic With applications becoming a mainstream utility on mobile devices, and the GPS systems being made available to everyone, it is no longer a hassle to deliver food to the exact location of a customer (william, 2019).The online food ordering market has increased in the U.S with 40 per cent of U.S adults having ordered their food online once. The online food ordering market includes foods prepared by restaurants, prepared by independent people, and groceries being ordered online and then picked up or delivered.

The first online food order was a pizza from Pizza Hut in 1994. The first online food ordering service, World Wide Waiter (now known as Waiter.com), was founded in 1995. The site originally serviced only northern California, later expanding to several additional cities in the United States. GrubHub was founded in 2004. By the late 2000s, major pizza chains had created their mobile applications and started doing 20-30 per cent of their business online. With increased smartphone penetration, and the growth of both Uber and the sharing economy, food delivery start-ups started to receive more attention. In 2010, Snapfinger, who is a multi-

restaurant ordering website, had a growth in their mobile food orders by 17 per cent in one year. Instacart was founded in 2012. In 2013, Seamless and Grubhub merged. Uber Eats launched in Los Angeles, California in 2014. By 2015, online ordering began overtaking phone orders. In 2015, China's online food ordering and the delivery market grew from 0.15 billion Yuan to 44.25 billion Yuan. As of September 2016, online delivery accounted for about 3 per cent of the 61 billion U.S. restaurant transactions. (Revolvy, 2018)

1.3. Problem Statement

As industries are fast expanding, people are seeking more ways to purchase products with much ease and still maintain cost-effectiveness. The vendors need to purchase the products to sell to end-users. The manual method of going to their local food sales outlets to purchase food is becoming obsolete and more tasking. Food can be ordered through the internet and payment made without going to the restaurant or the food vendor. So there is a need for a wide range of publicity and enabling direct order, processing and delivering of food through an online system.

1.4. Project as Solution

For this system, there will be a system administrator who will have the rights to enter the menu with current prevailing prices. This application will help people to order food easily excluding hassle.

1.5. Aim of the Project

To develop a food ordering system using the mobile application, which will help the user to order food anywhere anytime easily.

1.6. Objectives of the Project

- To allow a user to view food catalogue and search for food items.
- To allow a user to pay through credit card and have cash on delivery services.
- To allow customers to order using the application.

1.7. Structure of the Report

Introduction: This chapter contains project introduction, problem statement, the project as a solution, aim and objectives of the project and structure of the report.

Background: This chapter contains project elaboration, review of similar applications and journals, comparison with similar applications and its conclusion.

Development: This chapter contains methodologies considered for development process and reasons for selecting the methodologies and description of the methodologies.

Design: This chapter contains the design of the system. It contains Wireframe design and UML Designs of the developed application.

Testing: This chapter contains the testing of the system to ensure that all the functionality of the application works as intended or not.

Limitation: This chapter describes the limitation of the application.

Conclusion: This chapter contains the conclusion of the report which sums up the entire prospect of the developed applciation.

Reference List: This chapter includes all the reference list referenced in the project.

Appendix: This chapter contains all the proofs / relevant documents such as Gantt chart and survey report.

2. Background

This chapter includes the concepts, findings and research from the literature review. Relevant literatures were only reviewed and related information to the project's topic from the reviewed journals, articles and documents is summarized here.

2.1. Project Elaboration

FoodBite is an android application whose main motive is that it greatly simplifies the ordering process for both the customer and the restaurant. People are so confused in their daily life they rarely have time to have a proper meal. In this day and age, everyone is so consumed in their mobile devices it was a no brainer that it would be the most convenient means for the busy people to order food in no time. This application’s main motive is to make the daily life of people hassle-free.

The user can view food menu, go through the menu to add the item to the cart, delete and update the food items on the cart, add delivery details and they can choose through two methods of payment i.e. cash on delivery and online payment. User can pay online using a digital payment system Khalti. They should have an account in Khalti if they want to pay through the system.

The application which I’m about to develop is based on Android Operating System. The app will be made compatible with Android OS version 5 (Lollipop) and above based mobile phones. The project is developed in Java and XML. PHP is used as the server-side language. The database is maintained in MySQL. It is an online application and since its database is also an online database so it cannot be used without the internet.

2.2. Review of Similar Applications

In the Google Play Store, I've found some similar applications which provide somehow similar features as my application does. Here, I've mentioned those applications.

UberEats

Uber Eats is an American online food ordering and delivery platform launched by Uber in 2014 and based in San Francisco, California. This application lets user browse nearby restaurants and search for food by cuisine, restaurant name, dish, meal, and diet. User can choose from a variety of food to order: Pizza. Burritos. Burgers. Sushi. Chinese food. It is the app choice of millions and Uber Eats has delivered over 1 billion orders.

The Uber Eats app was selected as a Google Play "Editor's Choice" and was part of the "Best of 2018 Awards" for User's Choice (UberEats, n.d.).

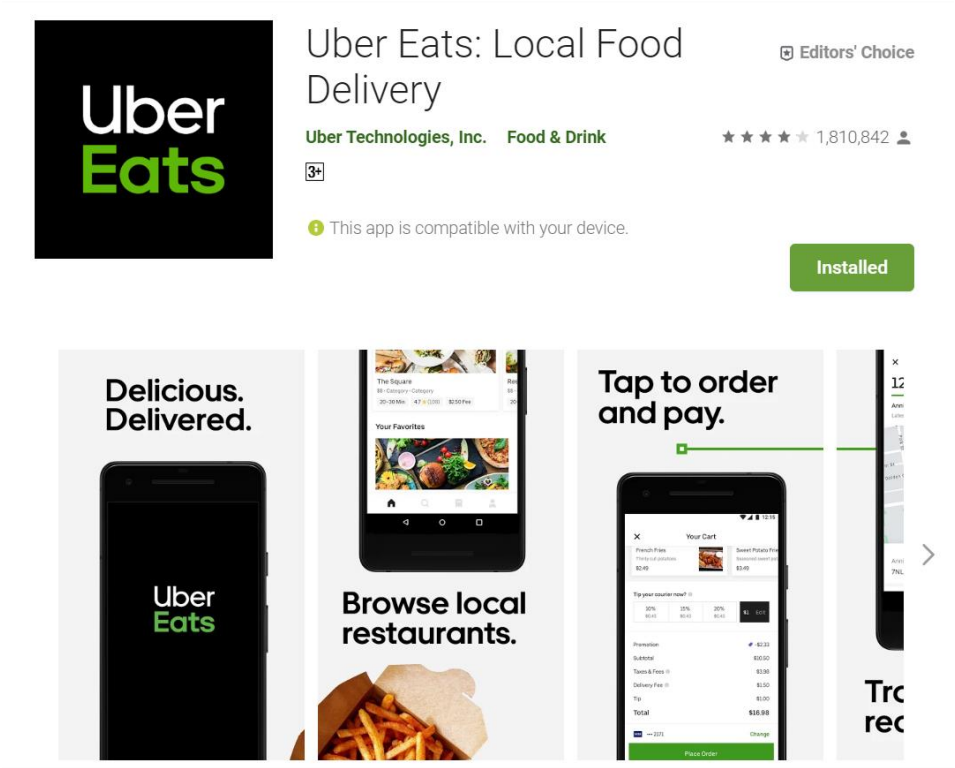


Figure 1: UberEats (Play Store, n.d.)

Swiggy Food Order & Delivery

Swiggy is a food order and delivery application which allows a user to order food and beverages online from restaurants near and around you. Swiggy delivers food from neighbourhood local joints, cafes, luxurious and elite restaurants. It delivers from chains like Dominos, KFC, Burger King, Pizza Hut, FreshMenu, Mc Donald's, Subway, Taco Bell and many more. They don't have any minimum order restrictions. Their features include long-distance lightning-fast orders, live order tracking, freebies, cashback, offers and discounts (Swiggy, n.d.).

Figure 2: Swiggy (Play Store, n.d.)

Foodpanda – Local Food Delivery

Foodpanda is a German mobile food delivery marketplace headquartered in Berlin, German. It is a local food delivery application that chooses local favourites; the best near you. They have cuisine and a dish to suit every moment and they claim to help a user make the first bite last. First, enter your address (home/ office/ treehouse). Then, choose your favourite restaurant and place an order. They'll prepare your food and once it's ready, our courier brings it to you. If you need something to watch, you can track your rider in real-time (foodpanda, n.d.).

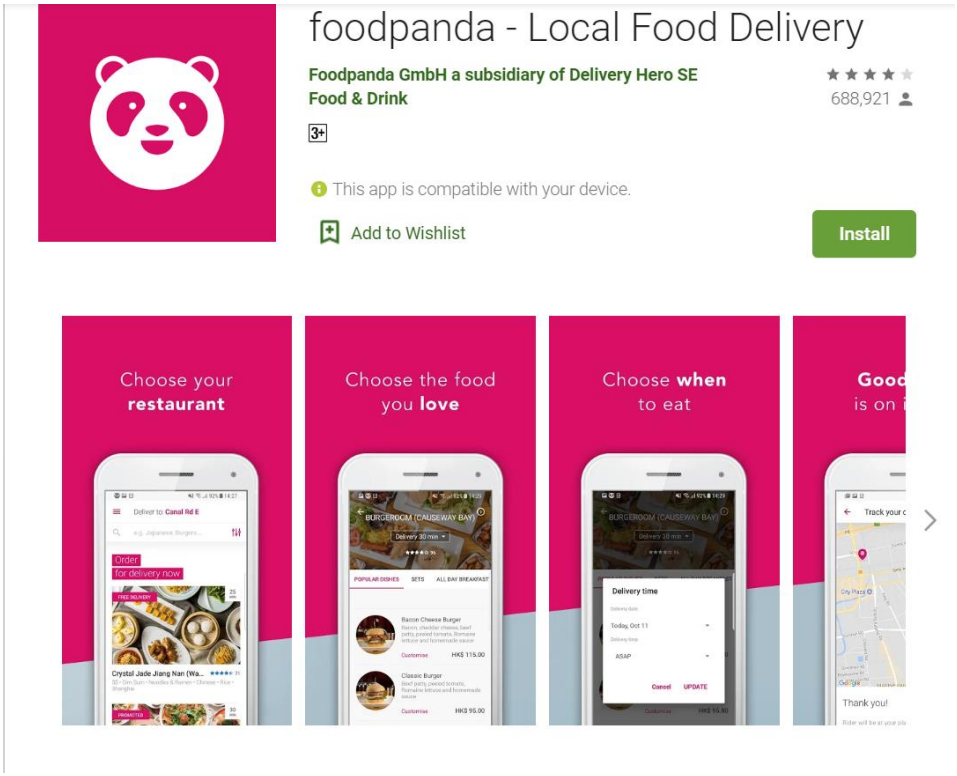


Figure 3: foodpanda – Local Food Delivery (Play Store, n.d.)

Zomato – Restaurant Finder Delivery App

Zomato is a restaurant finder and food delivery application that lets you search for and discover restaurants to eat out at or order in from. It lets you browse through restaurant menus, photos, user reviews and ratings to decide where you want to eat and use the map feature to guide you there. If users are in India, UAE, or Lebanon, you can also order food online for delivery, with thousands of great food delivery restaurants to choose from. The features of the application include searching restaurants, cafes, pubs and bars by location, cuisine and name, viewing menus, pictures, phone numbers, direction and user reviews and all the other information needed to choose a restaurant for dining out, food delivery, nightlife or takeaway. They have theme-based curated lists to discover places for the best burgers, perfect date spots, or the top trending restaurants. This application lets you rate and review restaurants a user have been to and share photos of your foodie moments directly from the app (Zomato, n.d.).

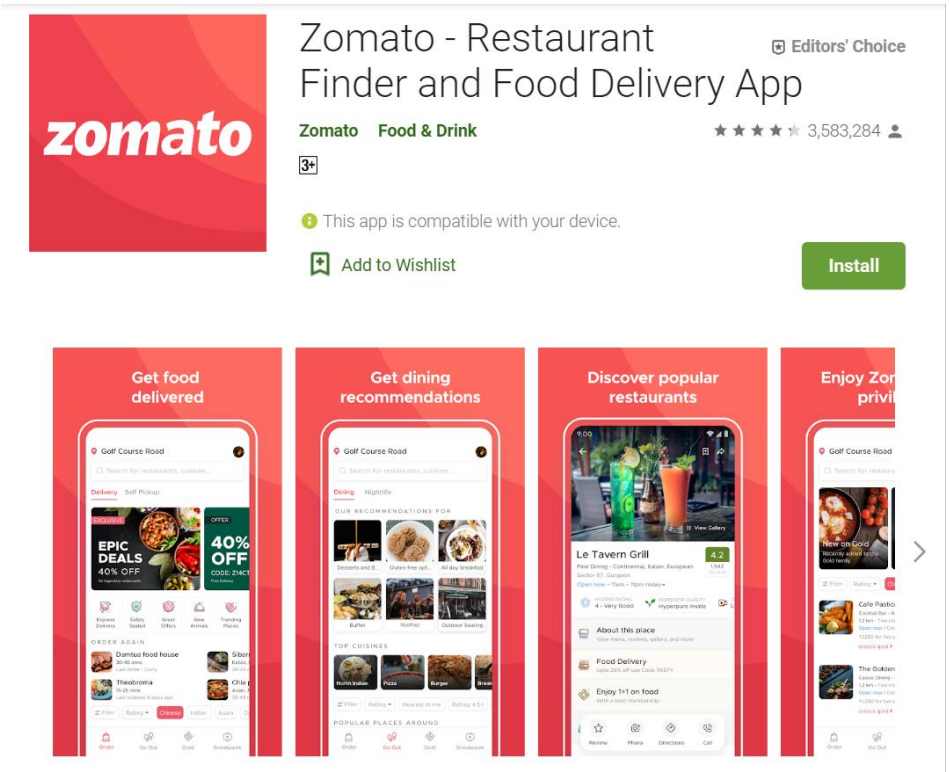


Figure 4: Zomato (Play Store, n.d.)

Deliveroo: Restaurant Delivery

Deliveroo is a restaurant delivery application that allows the user to order from the best restaurants in the business from – local hotspots to national favourites – and get the food delivered fast, right to user door. User can choose from hundreds of fine restaurants around the neighbourhood, including Busaba Eathai, Carluccio's, Gourmet Burger Kitchen, MEATliquor, and other top-quality independent businesses. The features of the application includes searching food by cuisines, category or restaurants, browsing by fastest delivery time, viewing menu and reading dish details, easily adding and removing items from basket in a single tap, placing an order for now later or tomorrow, paying directly from phone, tracking the progress of order, receiving notifications on delivery status (Deliveroo, n.d.).

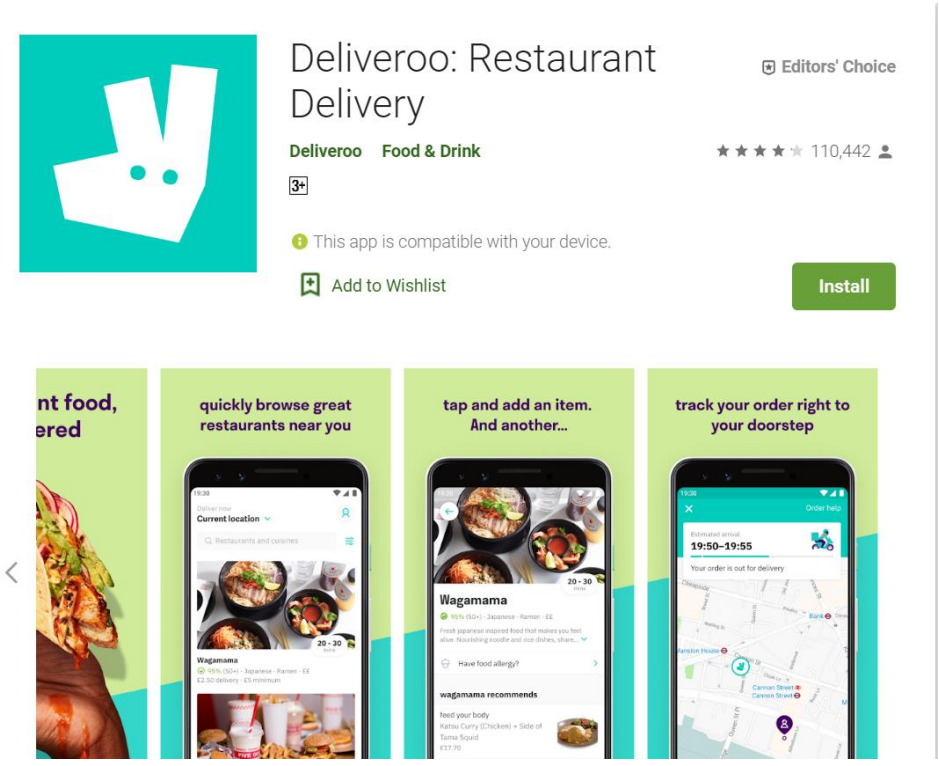


Figure 5: Deliveroo (Play Store, n.d.)

2.3. Review of Similar Journals

- **Android Application for Local Food Ordering System**

  The Rampant growth of wireless technology and Mobile devices in this era is creating a great impact on our lives. Some early efforts have been made to combine and utilize both of these technologies in advancement of hospitality industry. This research work aims to automate the food ordering process in near vicinity and also improve the dining experience of customers. In this report we discuss about the design & implementation of automated food ordering system with real time customer feedback for vendors.

  This system, implements wireless data access to servers. The android application on user's mobile will have all the menu details. The order details from customer's mobile are wirelessly updated in central database and subsequently sent to kitchen and cashier respectively. The vendor can manage the menu modifications easily. The wireless application on mobile devices provide a means of convenience, improving efficiency and accuracy for vendors by saving time, reducing human errors and real-time customer feedback (S.M. Takalkar, 2016).

- **Online Food Ordering System**

  Our proposed system is an online food ordering system that enables ease for the customers. It overcomes the disadvantages of the traditional queueing system. Our proposed system is a medium to order online food hassle free from restaurants as well as mess service. This system improves the method of taking the order from customer. The online food ordering system sets up a food menu online and customers can easily place the order as per their wish. Also with a food menu, customers can easily track the orders.

  This system also provides a feedback system in which user can rate the food items. Also, the proposed system can recommend hotels, food, based on the ratings given by the user, the hotel staff will be informed for the improvements along with the quality. The payment can be made online or pay-on-delivery system. For more secured ordering separate accounts are maintained for each user by providing them an ID and a password (R.Adithya, 2017).

- **Mobile Food Ordering Application using Android OS Platform**

  The purpose of this research is making an ordering food application based on Android with New Order, Order History, Restaurant Profile, Order Status, Tracking Order, and Setting Profile features. The research method used in this research is water model of System Development Life Cycle (SDLC) method with following phases: requirement definition, analysing and determining the features needed in developing application and making the detail definition of each features, system and software design, designing the flow of developing application by using storyboard design, user experience design, Unified Modelling Language (UML) design, and database structure design, implementation an unit testing, making database and translating the result of designs to programming language code then doing unit testing, integration and System testing, integrating unit program to one unit system then doing system testing, operation and maintenance, operating the result of system testing and if any changes and reparations needed then the previous phases could be back.

  The result of this research is an ordering food application based on Android for customer and courier user, and a website for restaurant and admin user. The conclusion of this research is to help customer in making order easily, to give detail information needed by customer, to help restaurant in receiving order, and to help courier while doing delivery (Ricky, 2014).

- **Design and Implementation of Digital Dining in Restaurants using Android**

  The growth of wireless technology and Mobile devices in this era is creating a great impact on our lives. Some early efforts have been made to combine and utilize both of these technologies in advancement of hospitality industry. This paper presents an easy and more subtle way of communicating to realize a wireless food ordering system. This system, implements wireless data access to the servers and food ordering functions through both desktops and mobile devices such as tablets over a wirelessly integrated local area network. In this paper we discuss about the design & implementation of automated food

ordering system for restaurants. This system, implements wireless data access to servers. The android application on user's mobile will have all the menu details. The order details from customer's mobile are wirelessly updated in central database and subsequently sent to kitchen and cashier respectively. The restaurant owner can manage the menu modifications easily. The wireless application on mobile devices provides a means of convenience, improving efficiency and accuracy for restaurants by saving time and real-time customer feedback (Shobhit Goyal, 2011).

- **FoodForCare: An Android application for self-care with healthy food**
  Recently, there are many people who ignore health concerns especially in their eating habits. This has made the number of diseases found in society, especially non-communicable diseases (NCDs) such as diabetes, hypertension, and cancer drastically increase every year. The number of patients having these diseases, could be reduced by paying more attention to the food that they eat and nutrition that they receive. Accordingly, the researchers would like to propose FoodForCare, an Android application for self-care with healthy food. The main purpose is to help users have better eating habits and a healthier lifestyle. FoodForCare provides functions for users to keep their daily personal health and food records of food intake. The users can see an analysis of nutrition and calories per day and whether it is sufficient or not. This application can give an overview on food calories and nutrition so that they can eat wisely. Finally, the development of this application hopes to help Thai people in order to manage their total nutrition and calories taken for a healthier lifestyle and will directly decrease the number of people who are getting diseases caused from a disorder of food and nutrition (Natnicha Suthumchai, 2016).

2.4. Comparison with Similar Applications

This is the comparison table between my application (FoodBite) and other applications (UberEats, Swiggy, Foodpanda and Zomato) that I choose to compare my application with.

Table 1: Comparison table with similar applications

Features/Application	**FoodBite**	**UberEats**	**Swiggy**	**Foodpanda**	**Zomato**
Detailed description of meals	Yes	Yes	Yes	Yes	Yes
Custom search tool	No	Yes	Yes	Yes	Yes
Order Placement	Yes	Yes	Yes	Yes	Yes
Multiple payment solutions	Yes	Yes	Yes	Yes	Yes
Order History	No	Yes	Yes	Yes	Yes
Manage Order	Yes	Yes	Yes	Yes	Yes
Geolocation	No	Yes	Yes	Yes	Yes
Schedule an order	No	Yes	Yes	Yes	Yes
Cancel order	Yes	Yes	N/A	N/A	Yes

2.5. Conclusion

From the comparison between similar applications, I found out that a lot of features of my application and the applications I choose to compare my application with had many similar features and some features were not available in my application that was present on those applications and vice versa.

3. Development

3.1. Methodologies considered

In order to develop a proper system, a developer need to follow the System Development Life Cycle. There are various methodologies that can be chosen in order to develop a complete system. For example; Waterfall Model, Spiral Model, Agile, Prototype Model, Iterative Model and so on. Every model is somehow related but have different aspects of developing the system. Here are some of the methodologies that were considered before developing the system.

3.1.1. Waterfall Model

This SDLC model is the oldest and most straightforward. With this methodology, we finish one phase and then start the next. Each phase has its mini-plan and each phase “waterfalls” into the next. The biggest drawback of this model is that small details left incomplete can hold up the entire process. (STACKIFY, 2017).

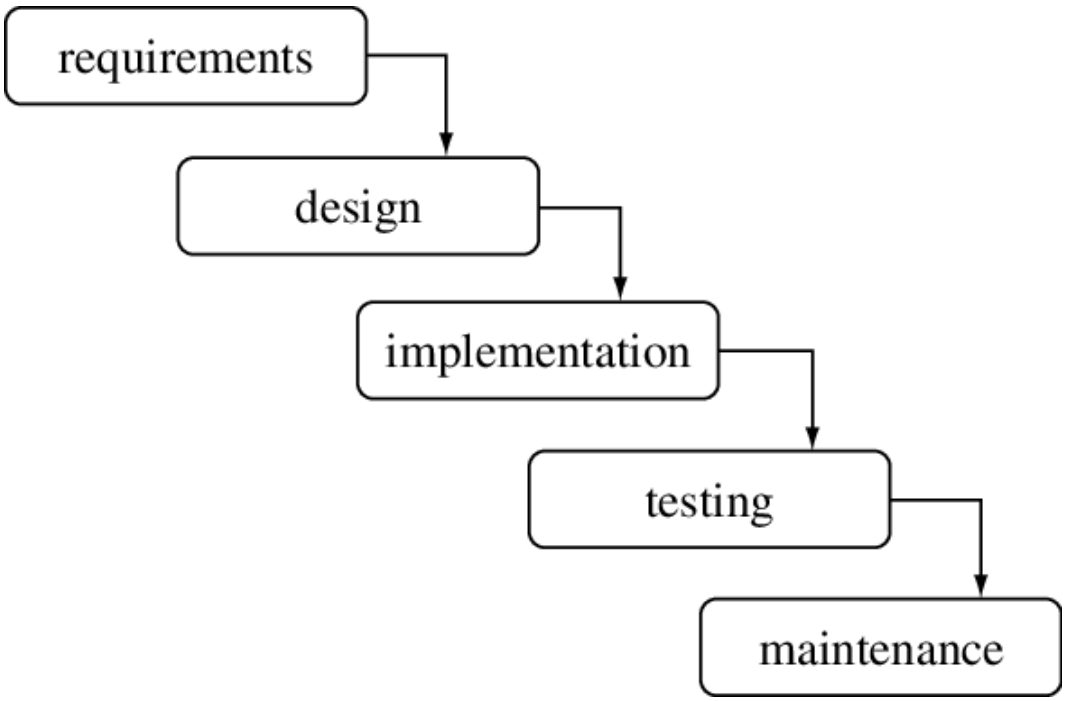


Figure 6: Waterfall model

Source: (Gary Marseden, 2008)

3.1.2. V-shaped Model

An extension of the waterfall model, this SDLC methodology tests at each stage of development. As with the waterfall, this process can run into roadblocks. (STACKIFY, 2017).

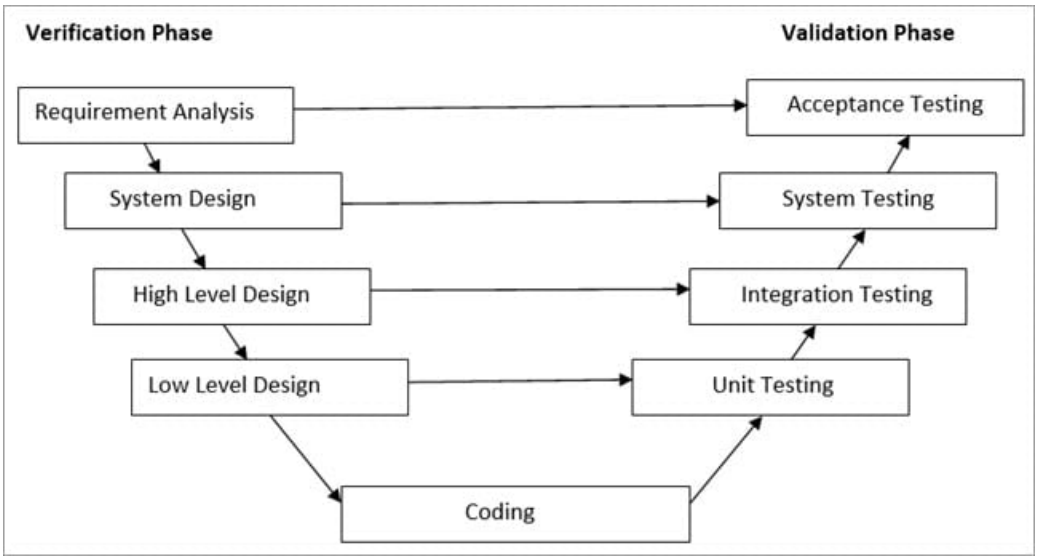


Figure 7: V shaped model

Source: (softwaretestinghelp, 2020)

3.1.3. Iterative Model

This SDLC model emphasizes repetition. Developers create a version very quickly and for relatively little cost, then test and improve it through rapid and successive versions. One big disadvantage here is that it can eat up resources fast if left unchecked. (STACKIFY, 2017).

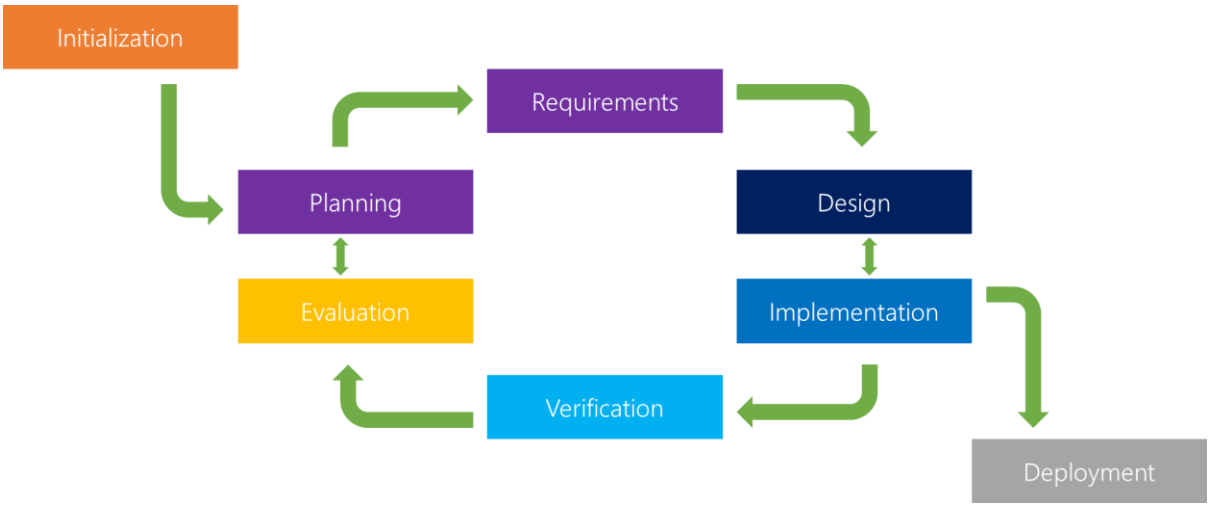


Figure 8: Iterative model

Source: (Powell-Morse, 2016)

3.1.4. Spiral Model

The most flexible of the SDLC models, the spiral model is similar to the iterative model in its emphasis on repetition. The spiral model goes through the planning, design, build and test phases over and over, with gradual improvements at each pass. (STACKIFY, 2017).

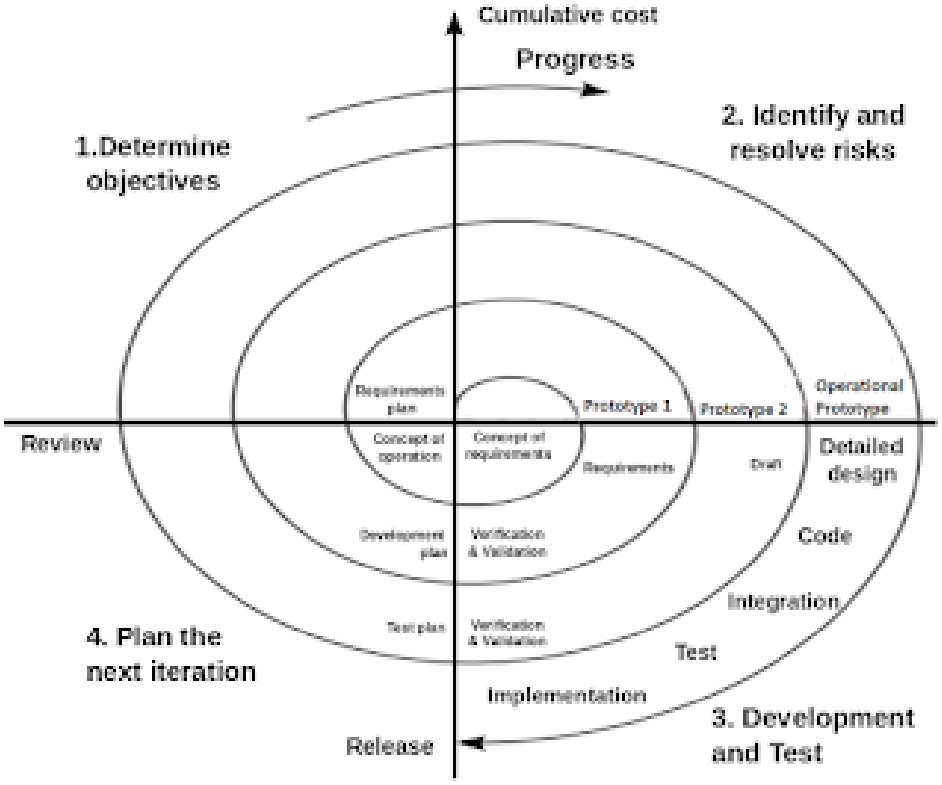


Figure 9: Spiral model

Source: (studytonight, 2020)

3.1.5. Agile

The Agile SDLC model separates the product into cycles and delivers a working product very quickly. This methodology produces a succession of releases. Testing of each release feeds back info that's incorporated into the next version. The drawback of this model is that the heavy emphasis on customer interaction can lead the project in the wrong direction in some cases. (STACKIFY, 2017).

Figure 10: Agile model

Source: (Sohail, 2019)

3.2. Reason for selecting methodology

The methodology that I am using for the final year project is Iterative methodology.

- It is easily adaptable to the ever changing needs of the project as well as the client**.**
- It is more cost effective to change the scope or requirements in Iterative model.
- One can get reliable user feedback, when presenting sketches and blueprints of the product to users for their feedback.
- Risks are identified and resolved during iteration; and each iteration is an easily managed.
- Testing and debugging during smaller iteration is easy.
- If a new iteration fails, a previous iteration can be implemented or “rolled back” with minimal losses.

3.3. Selected Methodology (Iterative Methodology)

There are lots of methodologies that can considered while developing the overall system. The old and famous methodology, Waterfall methodology is easy to understand but cannot be bound with this project as it needs to be perfect with each any every stage while developing. In case of this project, there can't be a perfect stage in order to move to another stage. The system developer needs to maintain the development with improving the features in every stage so I choose Iterative Model which I think is suitable for the project.

The iterative model differs from the traditional waterfall model in that it is more of a cyclical process, rather than a hard step-by-step process of stages. Once the initial planning phase is complete, a handful of other stages are repeated, creating cycles. As each cycle is completed, the software is improved and iterated on.

The initial stage is a planning stage, used to map out any specific details including hardware or software requirements as well as preparation for the other stages to follow.

The second stage is analysis, which is performed to set in place the database models, business logic and so on that will be necessary for this stage. The design stage also takes place here, wherein technical requirements are established that are necessary to meet any needs determined in the analysis stage.

Next, implementation and coding processes begin. Any specification, planning and design docs are implemented and coded at this point.

Fourth is the testing stage after the current build iteration is coded and implemented. These testing procedures are in place to identify any issues or bugs that have shown up.

Finally, once the previous stages are completed, a thorough evaluation is necessary of all development up to this stage. The team and clients or other parties are able to examine the project and offer feedback regarding what needs to or can change.

Once these stages are finished, the most recently built iteration of the software, as well as any evaluation feedback, are returned to the planning and development stage at the top for the process to repeat itself over again (Planbox, 2019).

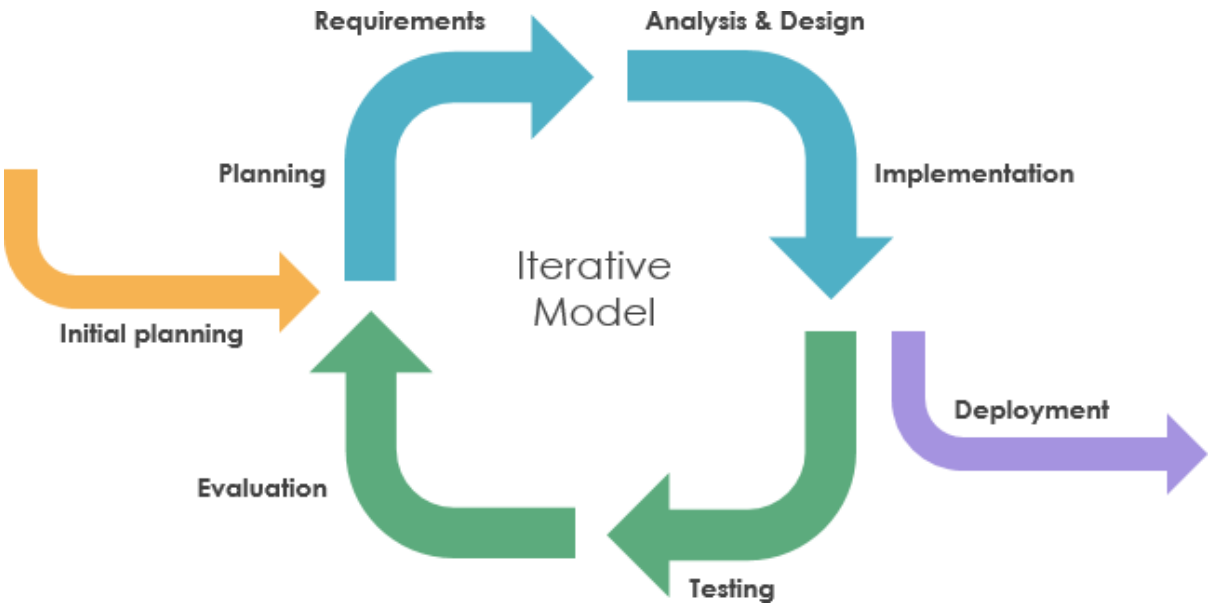


Figure 11: Iterative model

Following are the iterations divided for developing the application;

- 1s iteration: Logo design, User registration, login and food menu display
- 2nd iteration: Adding to cart, displaying items in cart
- 3rd iteration: Proceeding to checkout, updating and deleting item from cart
- 4th iteration: Online payment and cash on delivery

All the iteration are completed and are shown in the design part of the report.

Stages of Iterative Methodology

Following the iterative methodology, it contains various stages to complete a fully functional system. Those steps are briefly explained below.

1. Planning Phase

   This is the first stage of the iterative model, where proper planning is done, which helps them in mapping out the specifications documents, establish software or hardware requirements and generally prepare for the upcoming stages of the cycle.

   During the initial phase of the system development, the requirement for the needs aspects for system are to be gathered. At first, I researched about the market of the application on why I should build it and about the similar applications and website that are available publicly. It was found that there were very few food delivery app based on Pokhara. The details about those similar application was already described about entitling Similar Applications on **Chapter 2.** For this application, I conducted a survey among my fellow friends and within my circle using Google Forms. The detailed report of the survey for the study are shown in **Appendix** at the last of this document.

2. Analysis and Design Phase

   Once the planning is complete for the cycle, an analysis is performed to point out the appropriate business logic, database models and to know any other requirements of this particular stage. Moreover, the design stage also occurs in this phase of iterative model, where the technical requirements are established that will be utilized in order to meet the need of analysis stage.

   In order to make a clear concept on how the application will look like, a UI based prototype needs to be designed. Wire framing Mock-up can also be a prototype for it. After reviewing all the available similar application, the initial wire framing was designed. It is just a visualization of the system that is to be developed. After getting review of the developed wireframe, a UI based working prototype was designed on Balsmiq Mock-up. This looks like and works as same as the application which is just a design. This allows the system

developer to collect the required feedback from the people on how the system should be designed. The wireframe and UI design of the system are shown below at **Chapter 4** of the document.

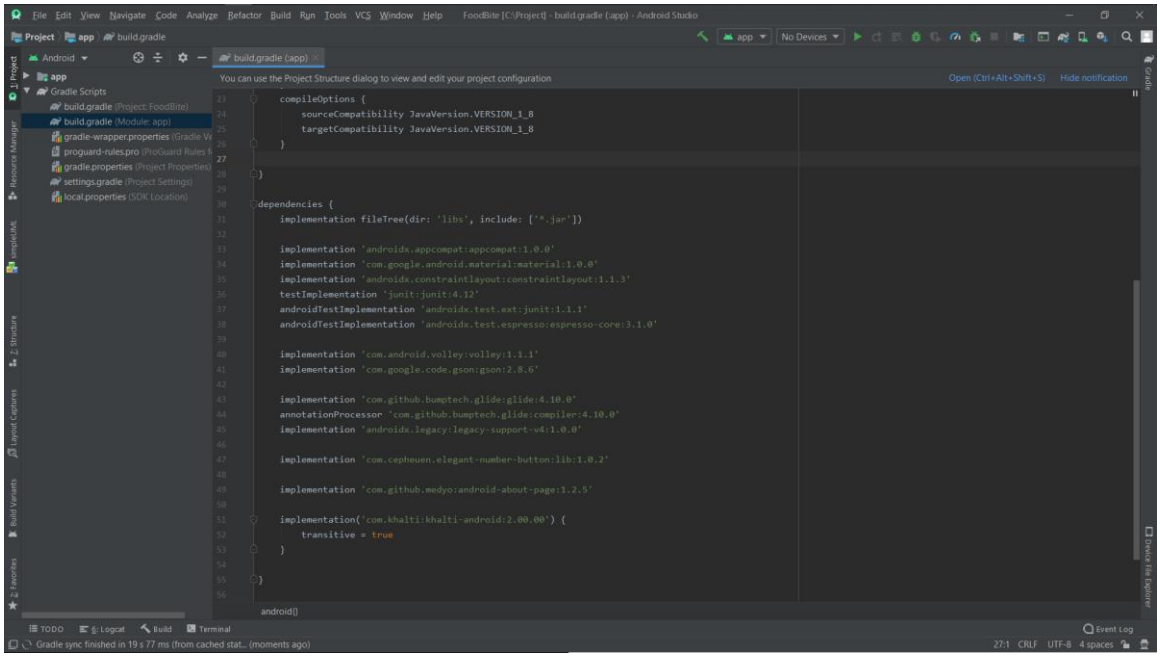
3. Implementation Phase

This is the third and the most important phase of the iterative model. Here, the actual implementation and coding process is executed. All planning, specification, and design documents up to this point are coded and implemented into this initial iteration of the project.

The system is based on Android OS and Java was chosen as the main programming language of the application. In Android Studio, XML is used to design the layout of the pages. Java provides the live function of the designed XML layouts.

Different aspects such as Fragments, Adapters, and Models and so on are used to classify the main Java class files.

Dependencies used:



These are dependencies I implemented while developing the project I used glide because it is a fast and efficient open source media management and

image loading framework for Android. It supports fetching, decoding, and displaying video stills, images, and animated GIFs.
I implemented elegant number button for creating a counter with increment and decrement to add food quantity.
I implemented khalti for online payment.

Some important codes that were crucial in development of the project are attached in **Appendix** section.

4. Testing Phase
   After the current build iteration is coded and implemented, testing is initiated in the cycle to identify and locate any potential bugs or issues that may have been in the software.

   Debugging is done to remove the bugs of the system whereas testing is done to verify the qualify control of the application. Testing is one of the important tasks that is to be performed before taking the system to the market or making it live. The detail of the testing cases is below at **Chapter 5** of this document.

5. Evaluation Phase
   The final phase of the Iterative life cycle is the evaluation phase, where the entire team along with the client, examine the status of the project and validate whether it is as per the suggested requirements.

4. Design

In Prototype methodology, the design phase starts right after requirement gathering. After gathering all required data, the initial design of the purposed system should start. This helps in giving visualization to purposed system. Designing is all about the User Interface. The initial design decides on what the system will look like. It is as like a blueprint of the system that is to be prepared. The design should be simple and clear in order to make user interactive.

4.1. User Interface Design

Logo

Logo is a symbol which lets people identify your brand and it lets people what kind of brand it is.

Figure 12: Logo

Splash Screen

This is first page that appears when a user opens the application. It contains application name with chosen colour theme of application.

Figure 13: Splash Screen

Login Page

This page appears right after splash screen. In this page user can log in to the application if they have already registered to the application or they can click on register now to register to the application

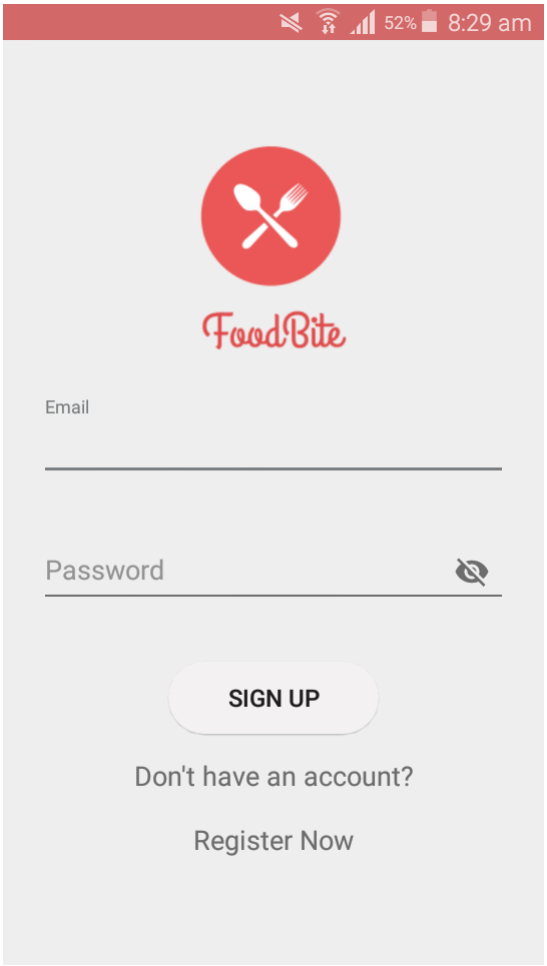


Figure 14: Login Page

Register Page

This page appears when a user clicks on register now. Here the user can fill up the form in order to register themselves to login to the application.

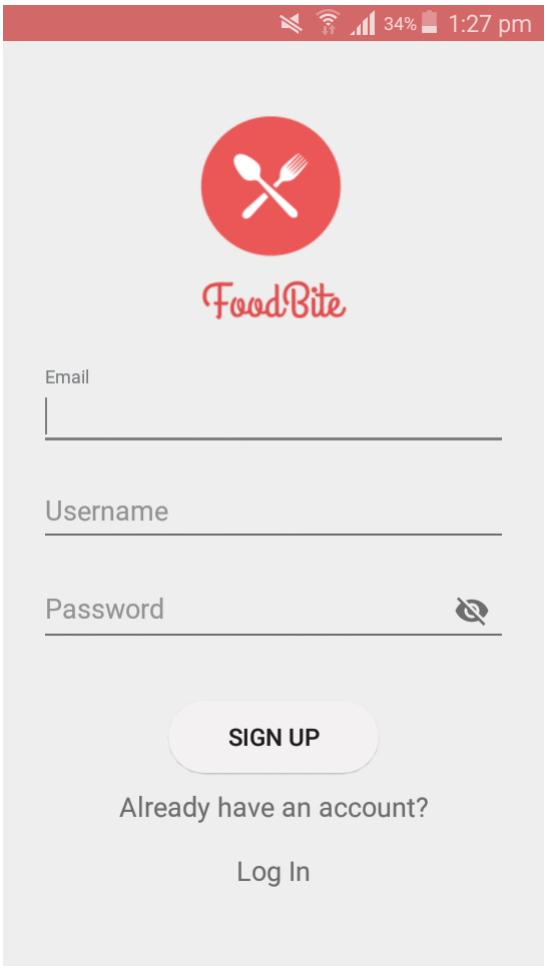


Figure 15: Register Page

Home Page

This page appears after a user is successfully logged in to the application. This page shows the list of menu which a user can select to order.

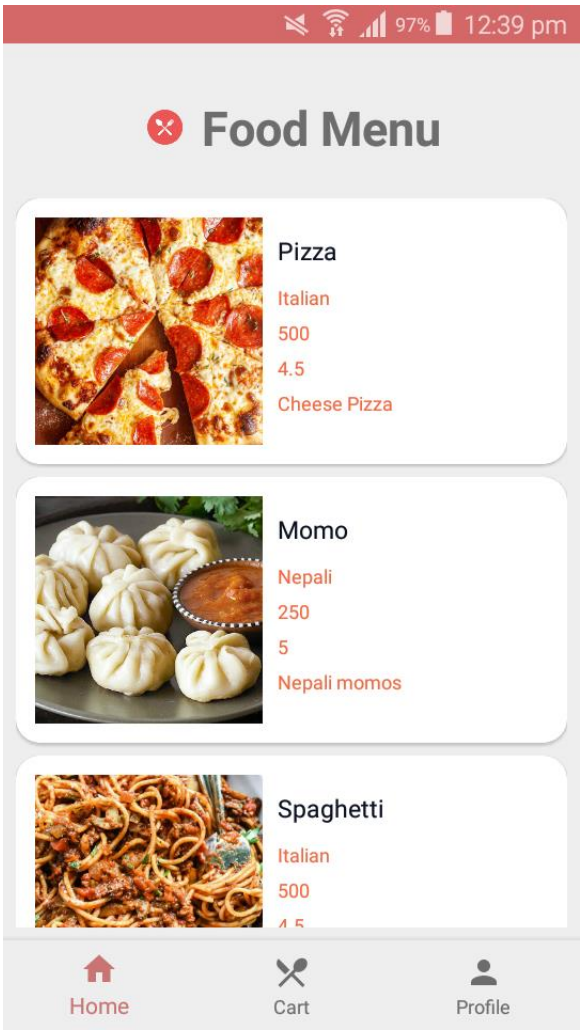


Figure 16: Home page

Add to Cart Page

This page appears when a user clicks on the menu item. Here user can click on the number button to select the number of food item they want to add to cart to order and click on the cart to add the desired amount of food to cart.

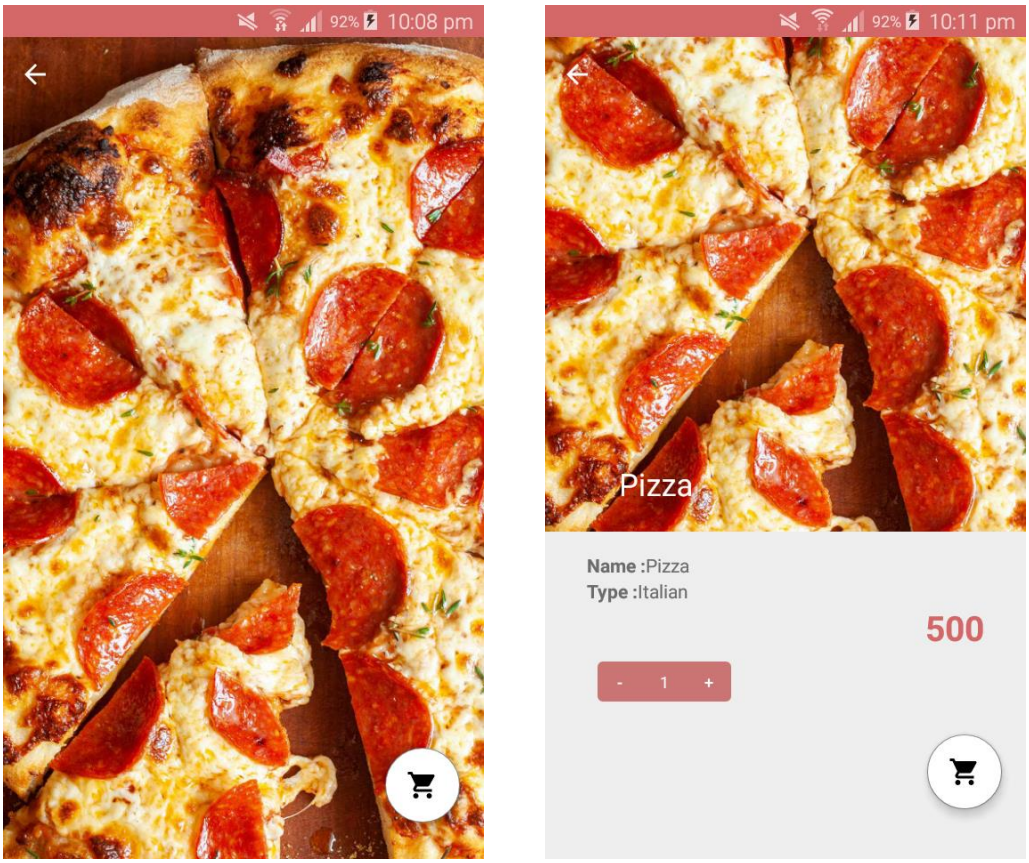


Figure 17: Order page

Cart Page

This page shows the list of food items added to cart.

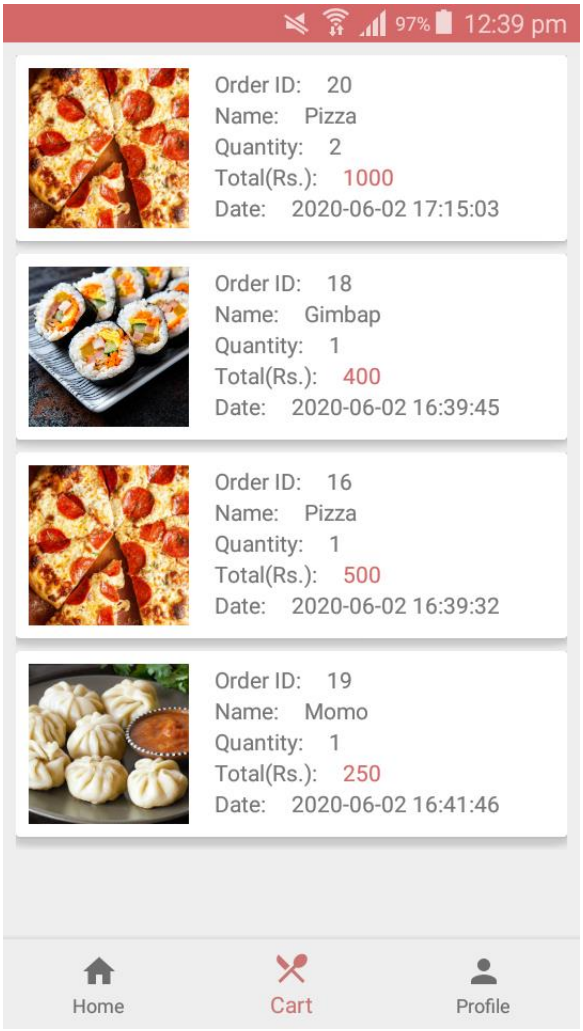


Figure 18: Cart Page

Order Page

This is the order page which is displayed after clicking the item on the cart. In this page user can cancel the order, update the order and choose payment method to order food.

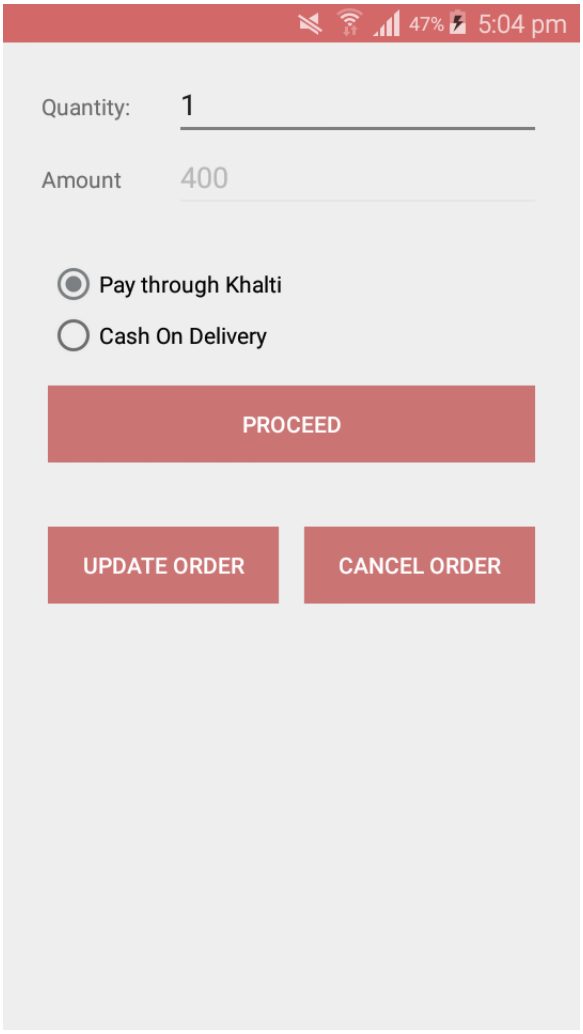


Figure 19: Order Page

Delivery Details Page

This page appears after you select a method of payment and click on proceed. Here user should put on their delivery address or contact number and any requests they may have for delivery.

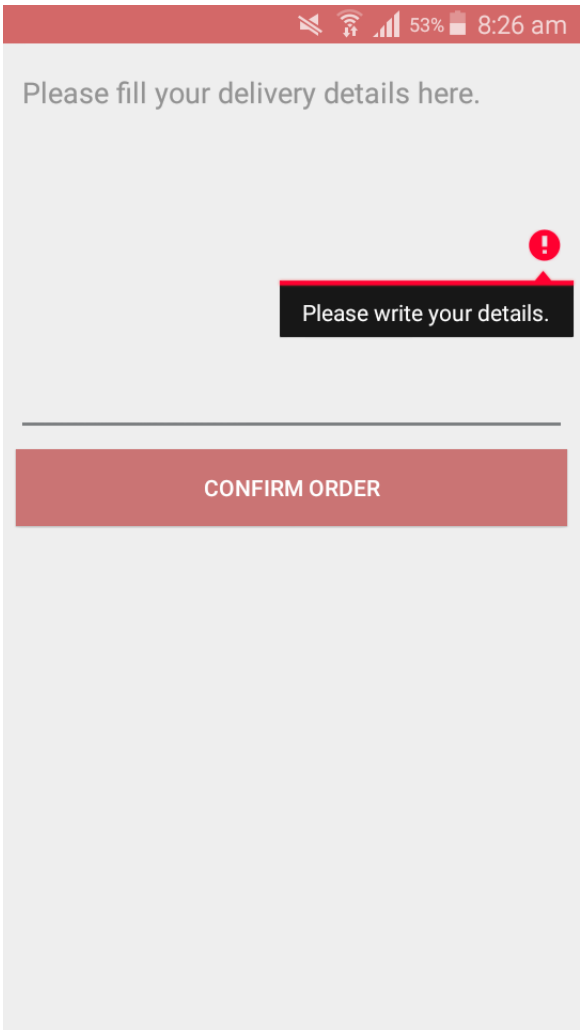


Figure 20: Delivery Details Page

Profile Page

This pages shows user's name and email information and user can update their password in this page they can view information of the application and logout from the application.

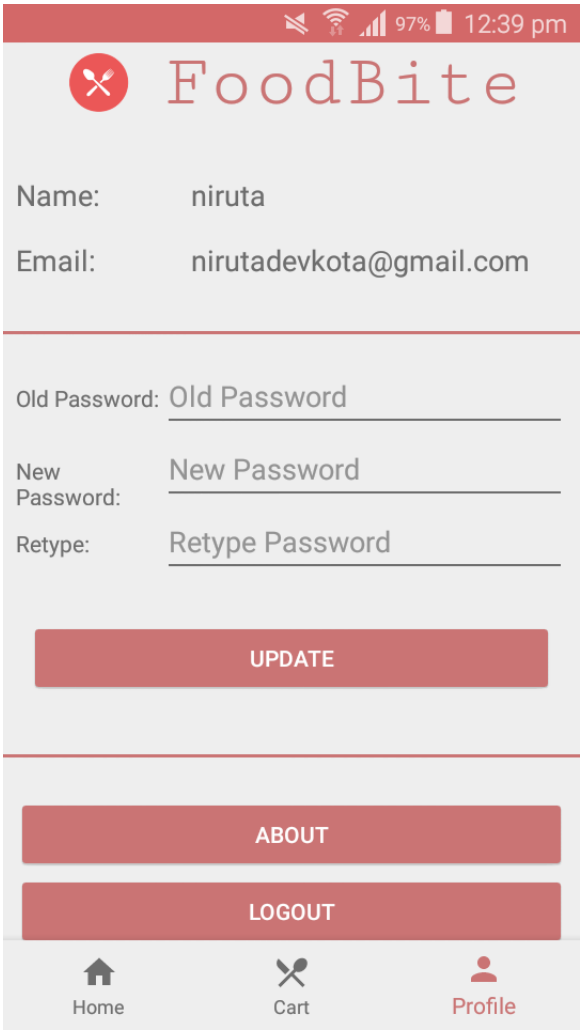


Figure 21: Profile Page

About Page

This page displays information of application and contact details of the developer.

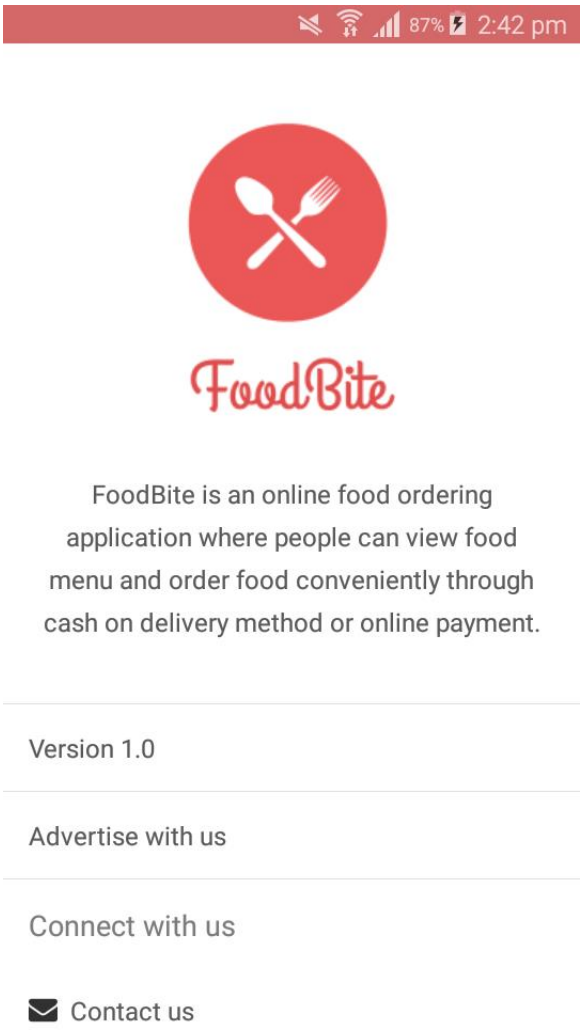


Figure 22: About Page

4.2. Use Case Design

A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse (Ambler, 2004).

Actor can be anyone who interacts with the system. An actor is a person, organization or external system that plays a role in one or more interactions with the system. Actors are drawn as stick figures. The actors interact to perform certain tasks with the system to represent the end result.

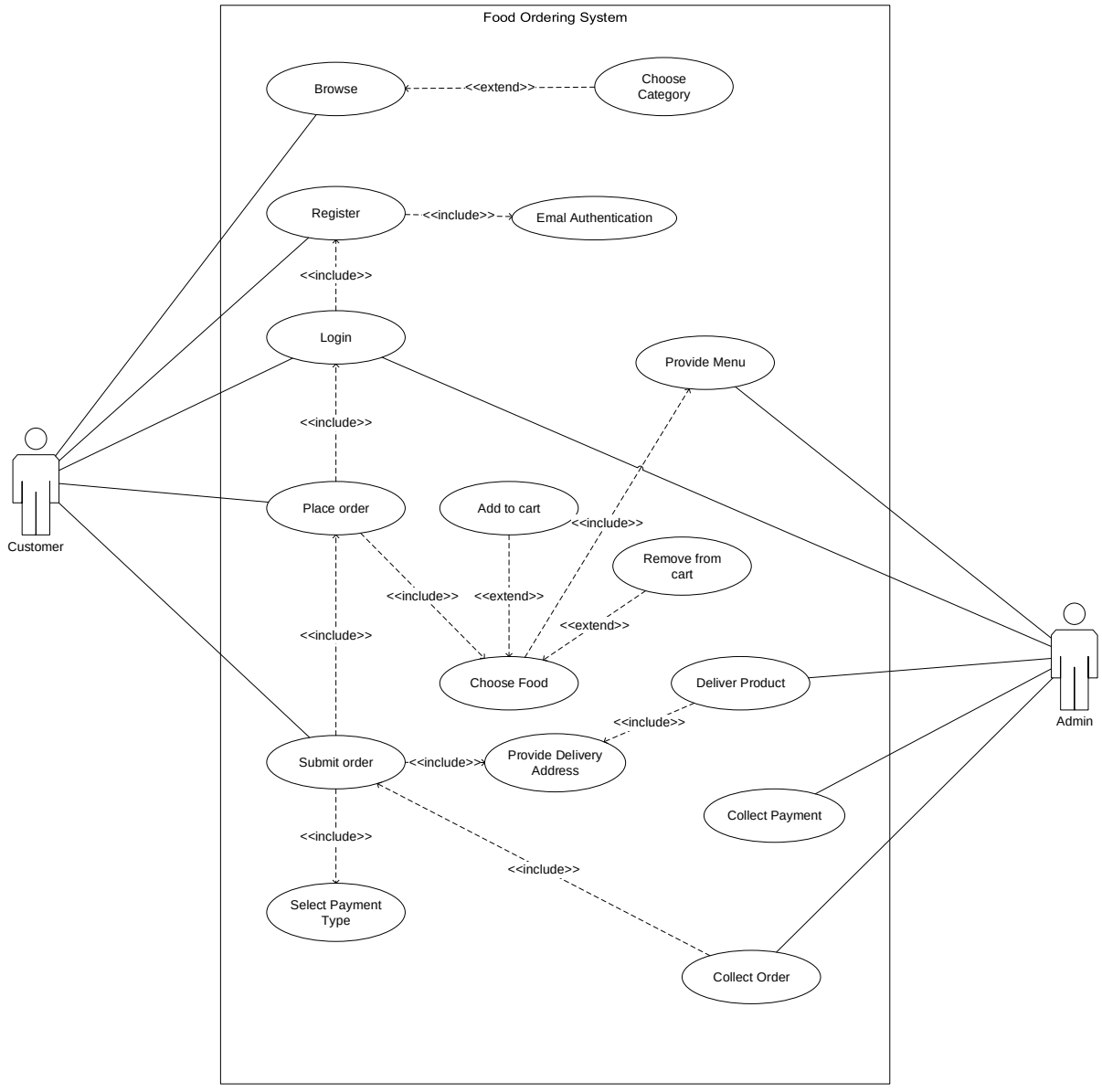


Figure 23: Use Case Diagram

4.3. Extended Use Case

Login/Register

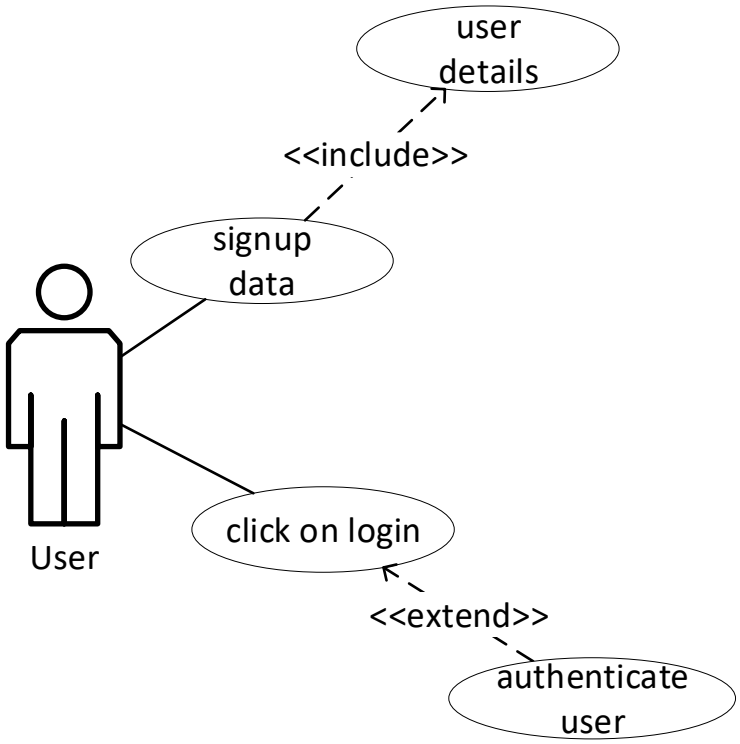


Figure 24: Extended use case for login/register

Select Food Item

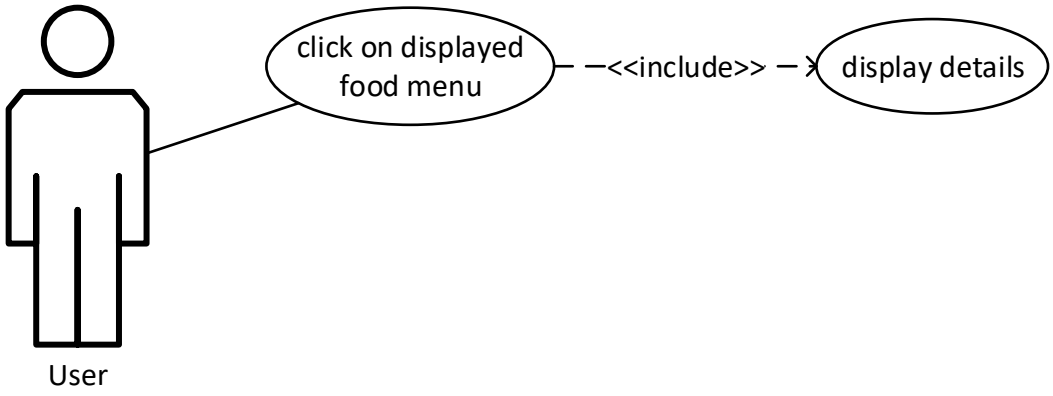


Figure 25: Extended use case for select food item

Add to cart

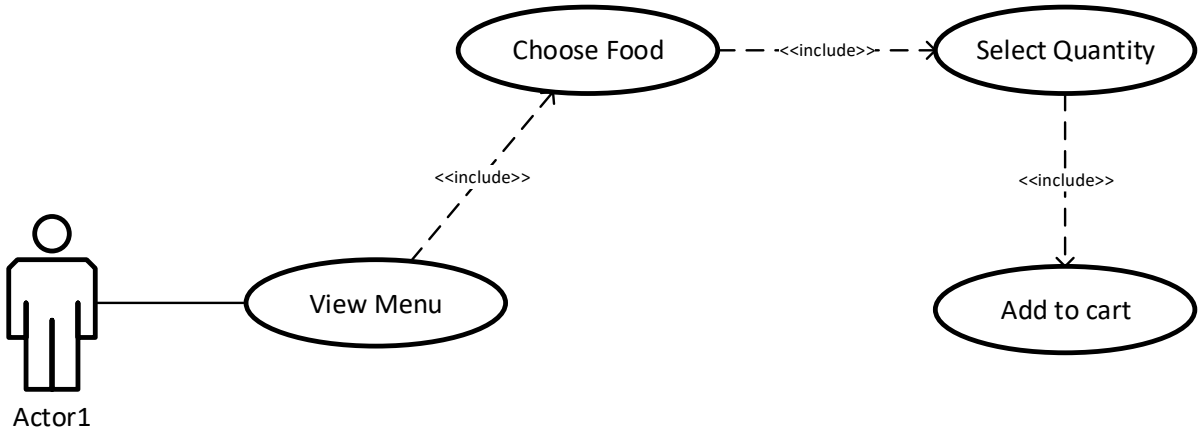


Figure 26: Extended use case for add to cart

Order Food

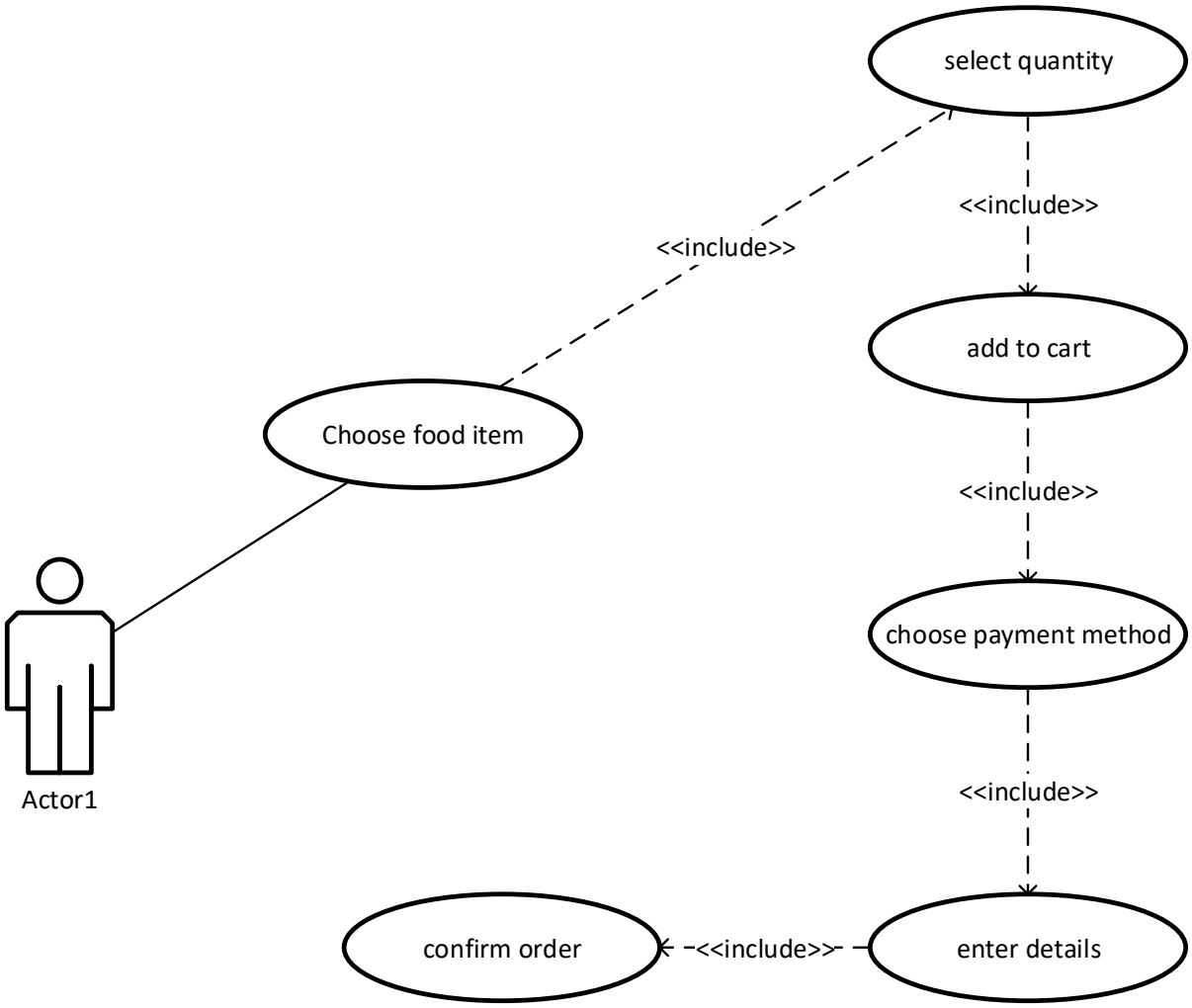


Figure 27: Extended use case for order food

Logout

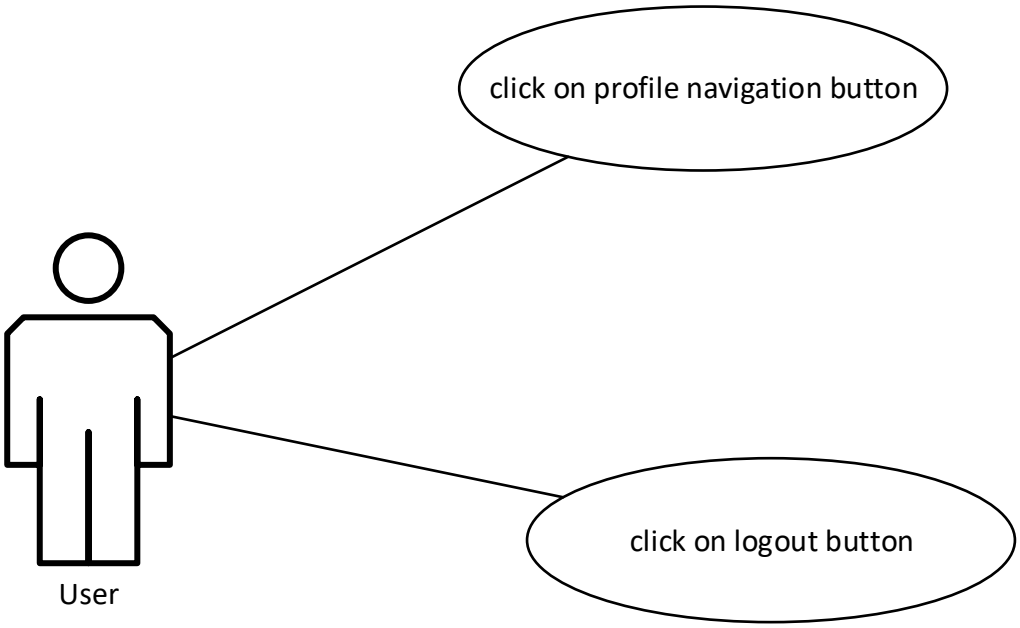


Figure 28: Extended use case for logout

4.4. Entity Relationship Diagram (ERD)

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs (Lucidchart, 2020).

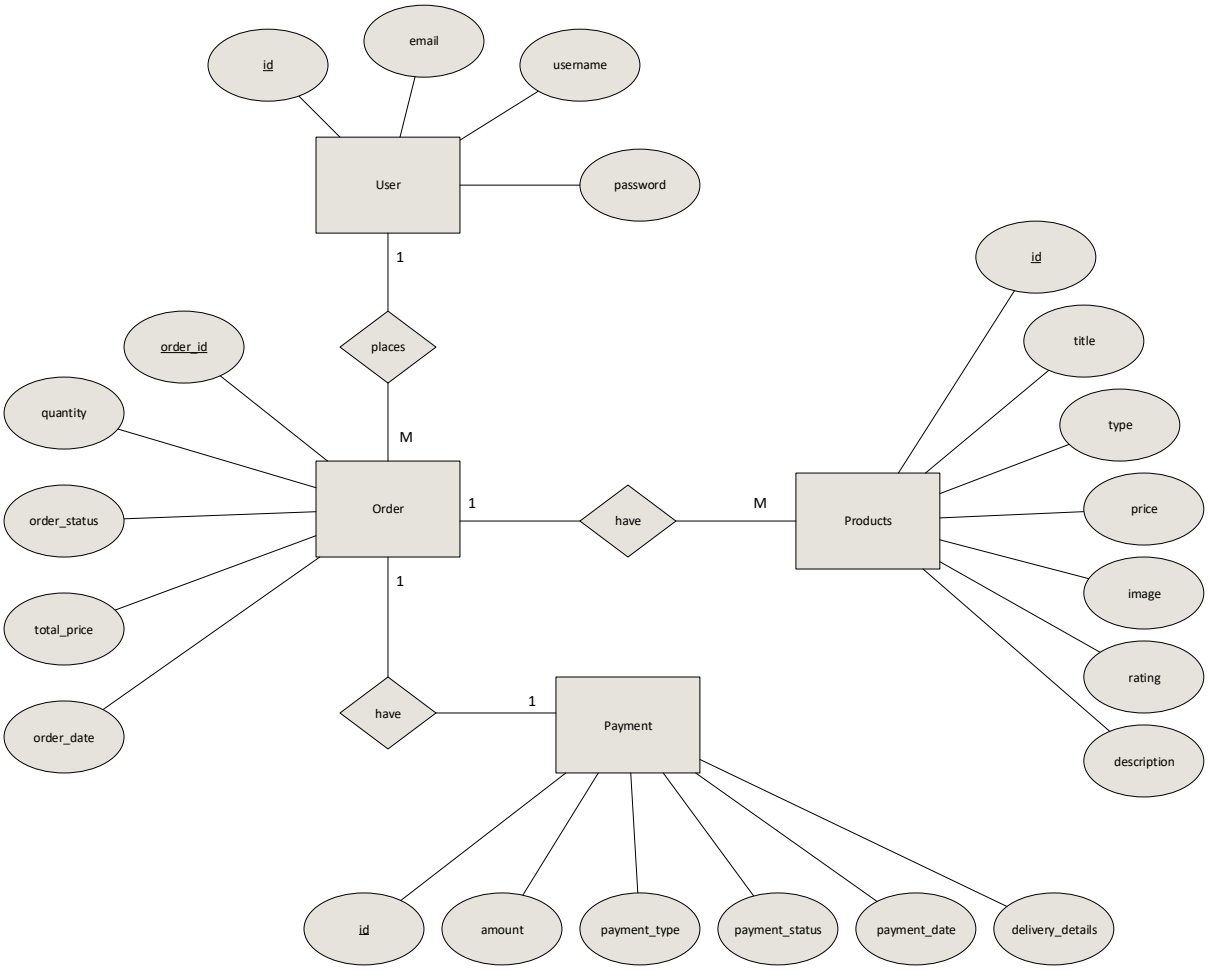


Figure 29: Entity Relationship Diagram

4.5. Class Diagram

Class diagram is a static diagram. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram (tutorialspoint, 2020).

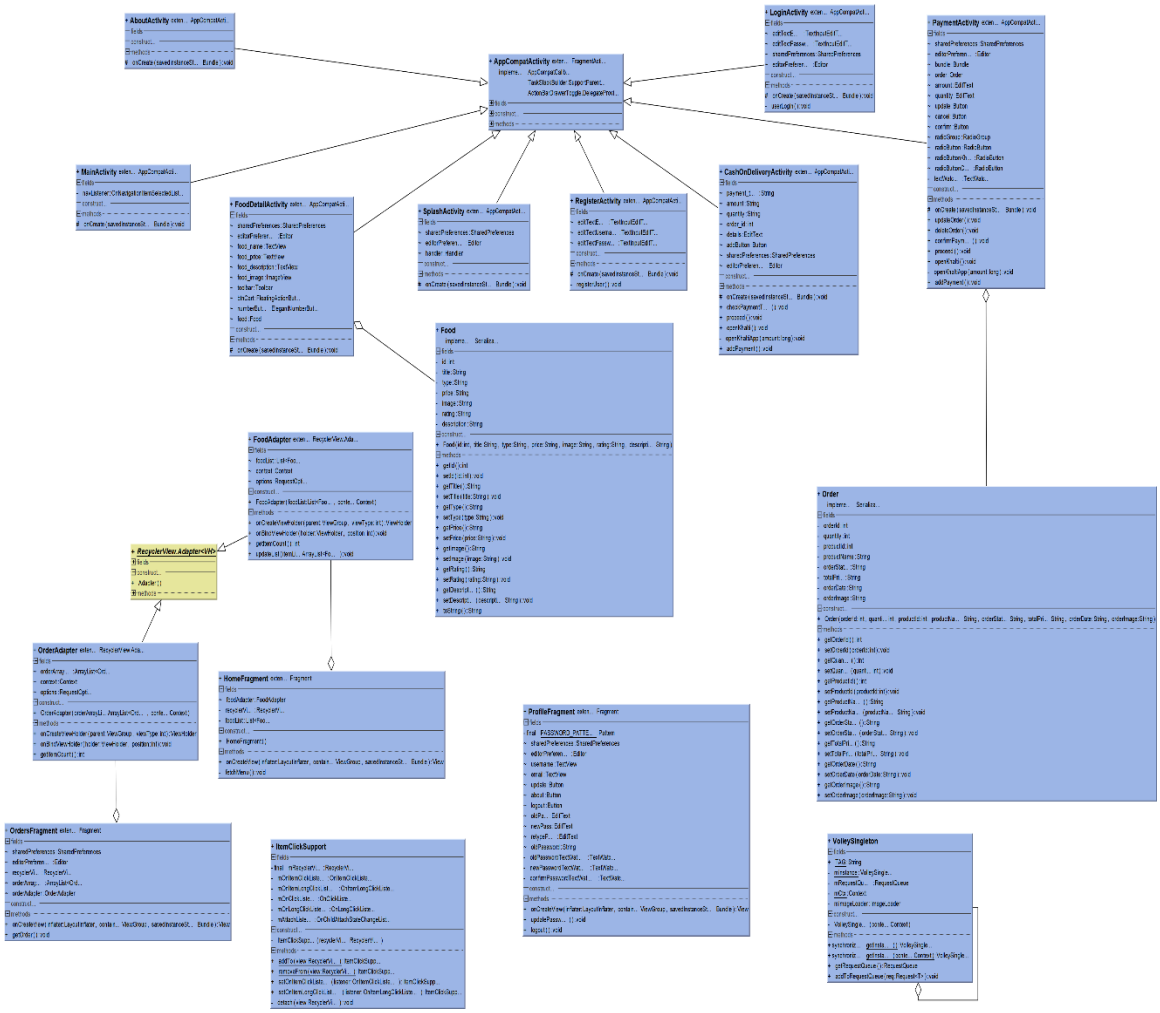


Figure 30: Class Diagram

4.6. Activity Diagram

An activity diagram visually presents a series of action or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modelling. They can also describe the steps in a use case diagram. Activities modelled can be sequential and concurrent. In both cases an activity diagram will have a beginning (an initial state) and an end (a final state) (smartdraw, 2020).

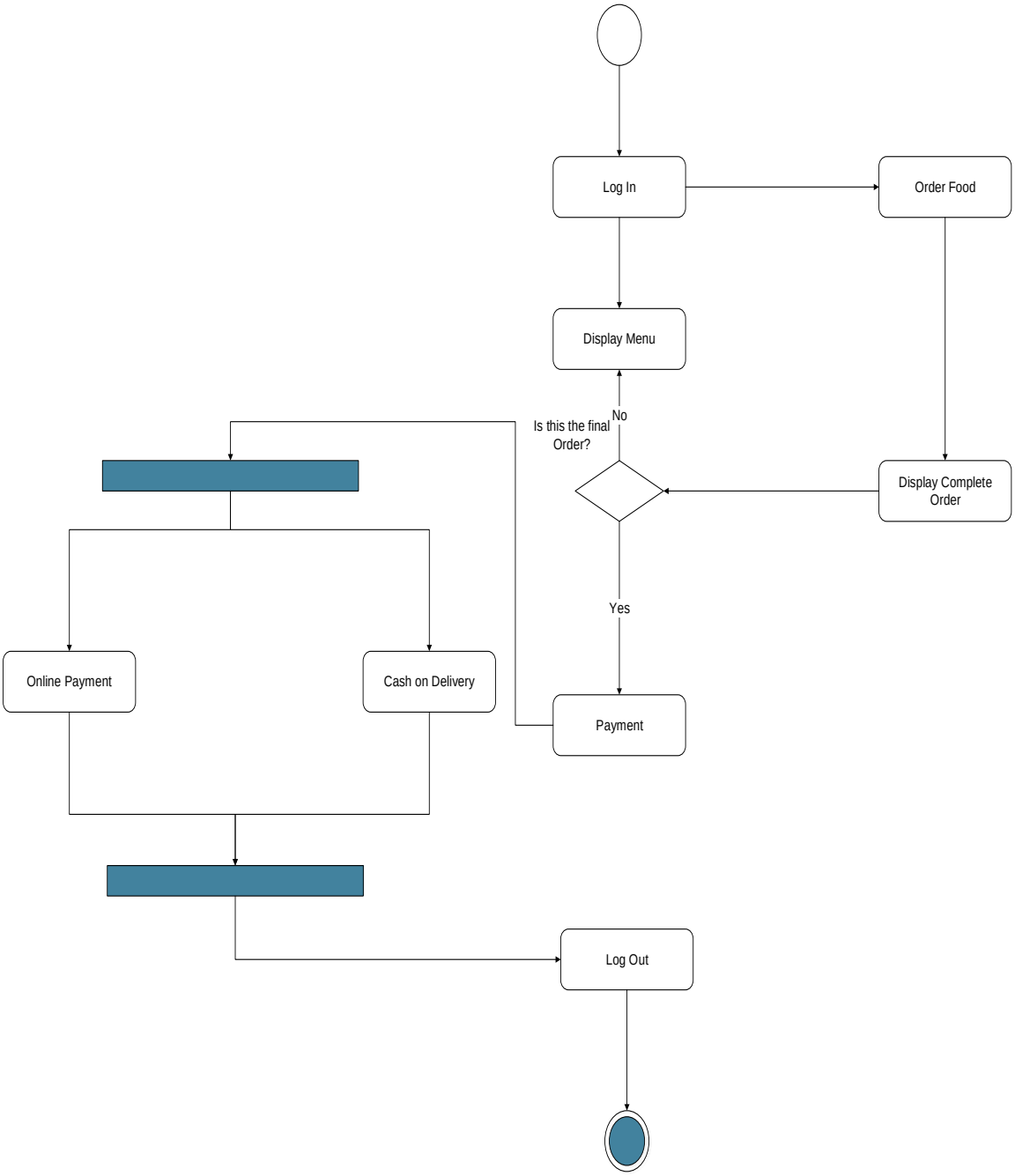


Figure 31: Activity Diagram

Activity diagram for Login/Register

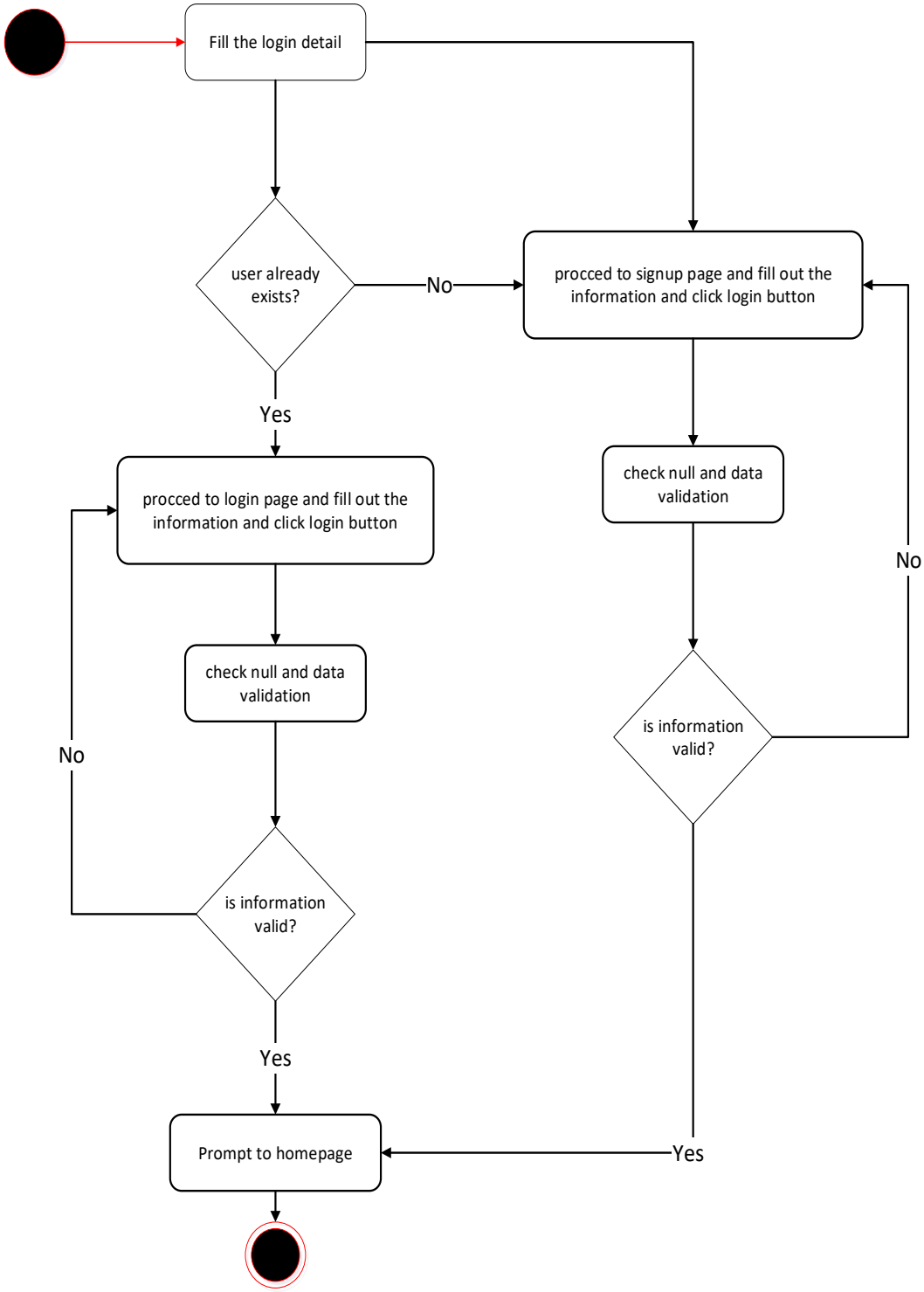


Figure 32: Activity diagram for login/register

Activity diagram for select food

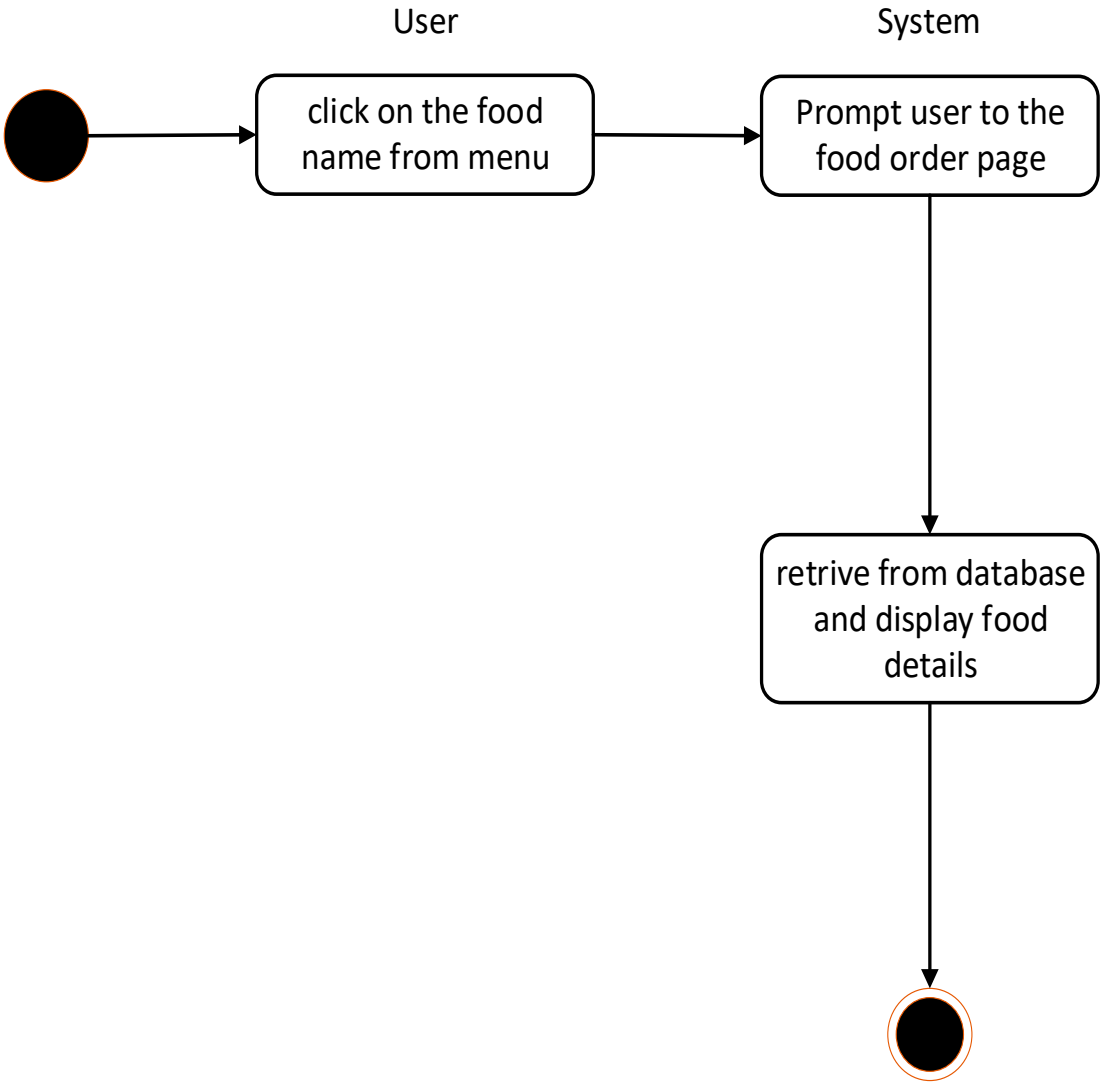
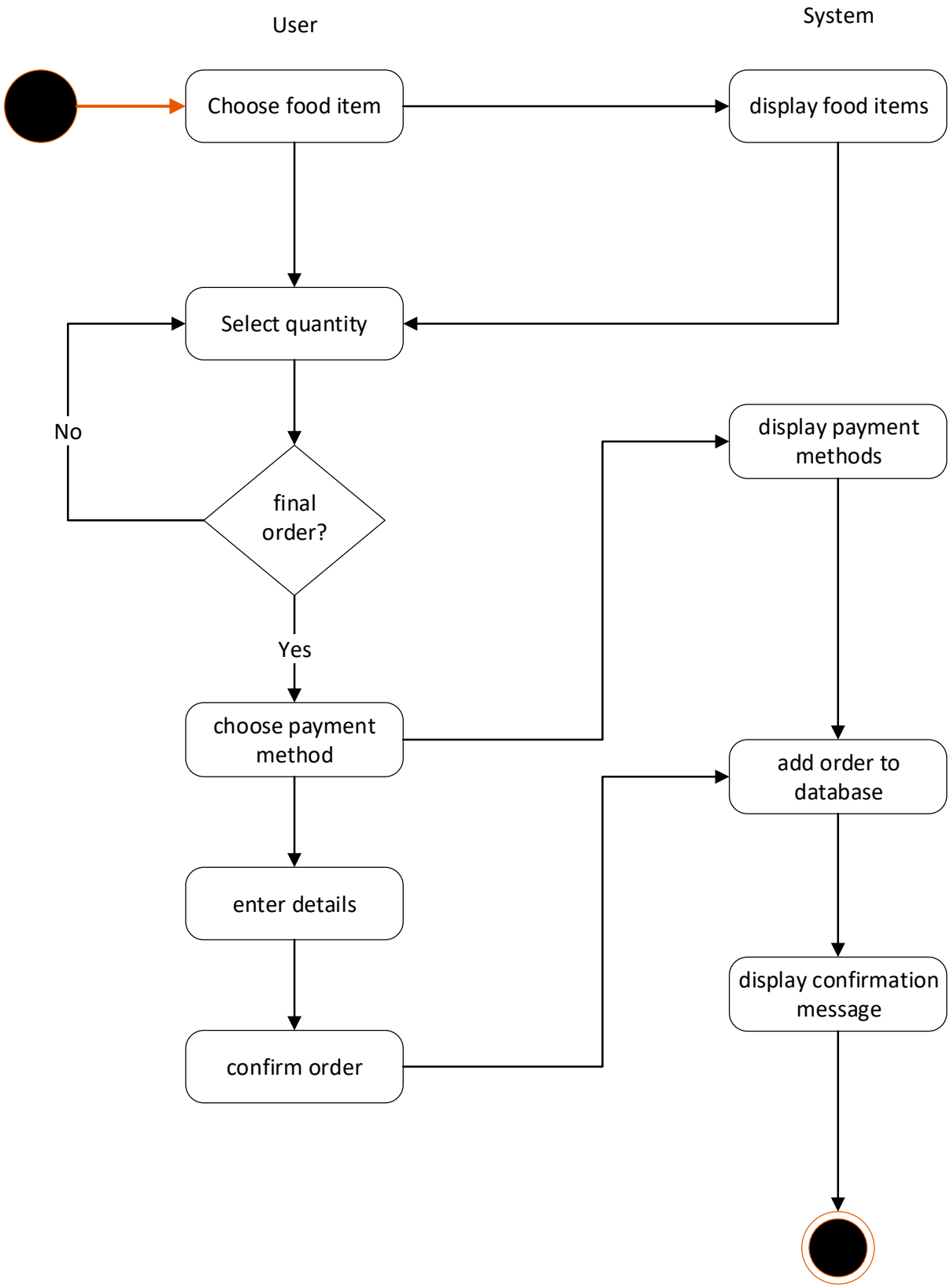


Figure 33: Activity diagram for change password

Order Food


User
System
Choose food item
display food items
Select quantity
No
final order?
Yes
display payment methods
choose payment method
add order to database
enter details
display confirmation message
confirm order

Activity diagram for change password

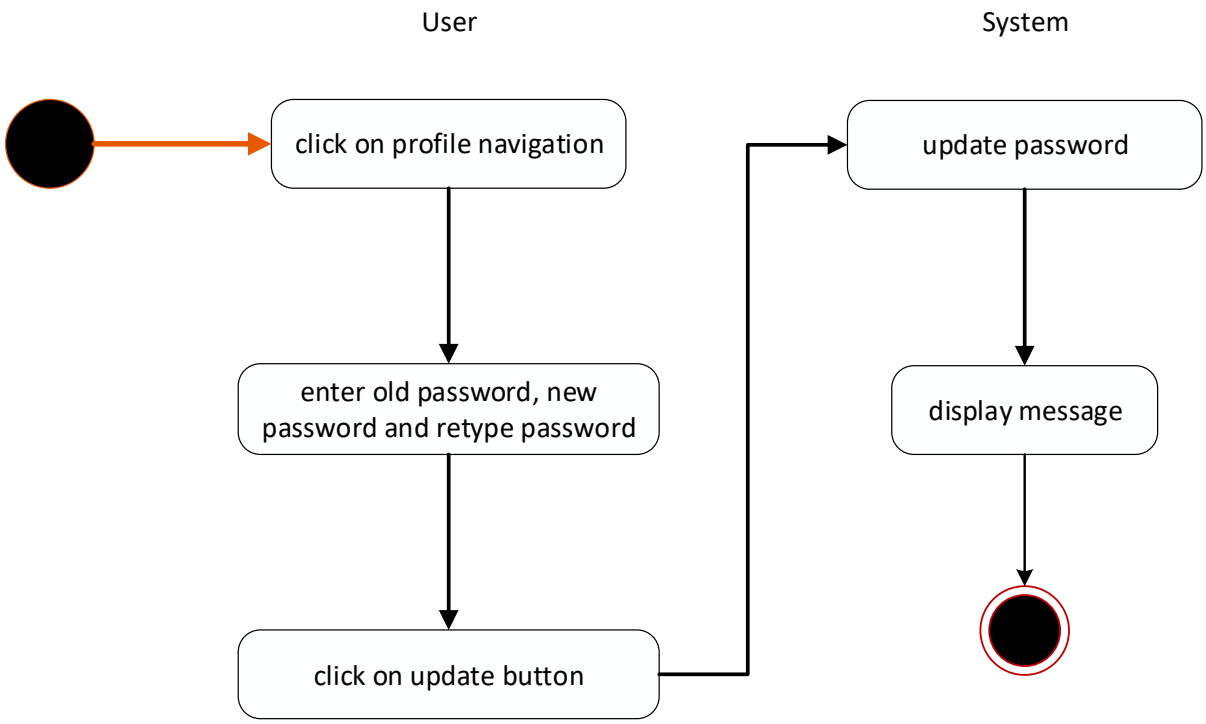


Figure 34: Activity diagram for change password

Activity diagram for logout

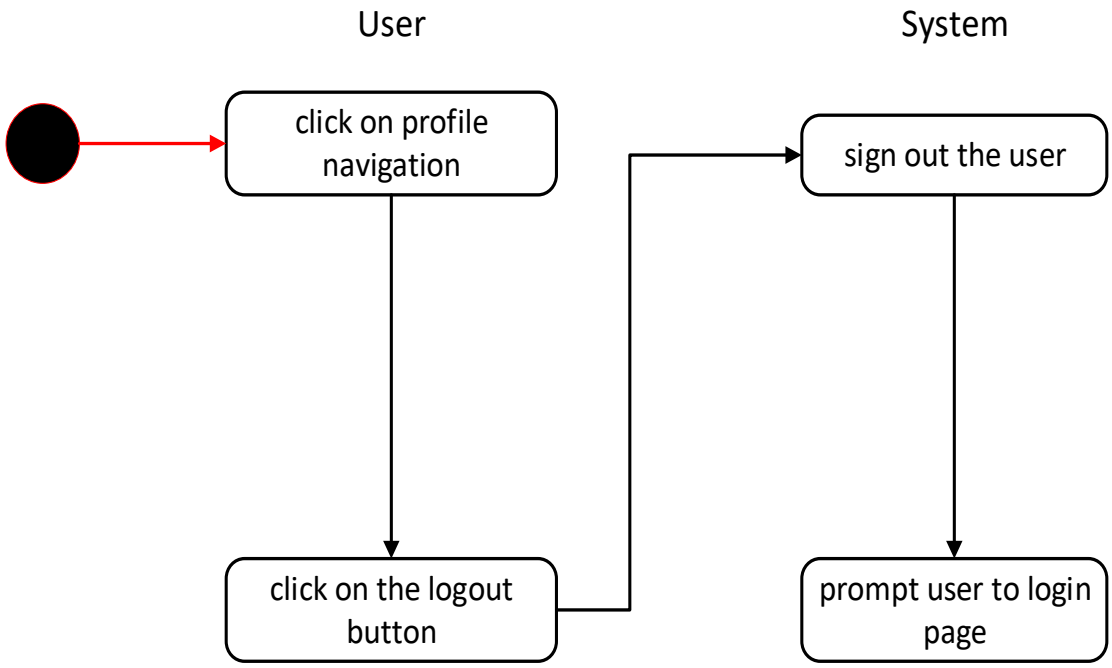


Figure 35: Activity diagram for logout

4.7. Data Dictionary

A data dictionary is a file or a set of files that contains a database's metadata. The data dictionary contains records about other objects in the database, such as data ownership, data relationships to other objects, and other data (technopedia, 2011).

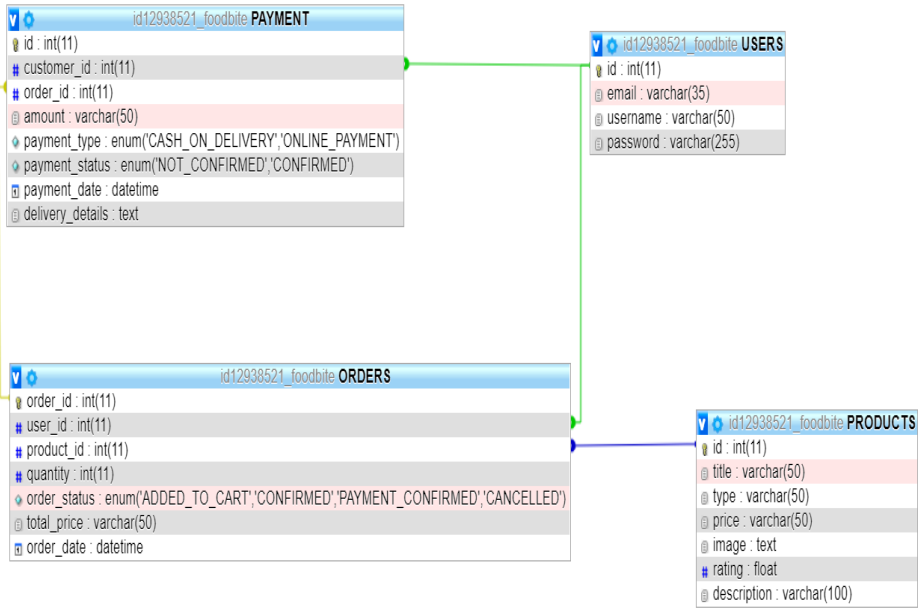


Figure 36: Data Dictionary

4.8. Sequence Diagram

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when (Visual Paradigm, 2020).

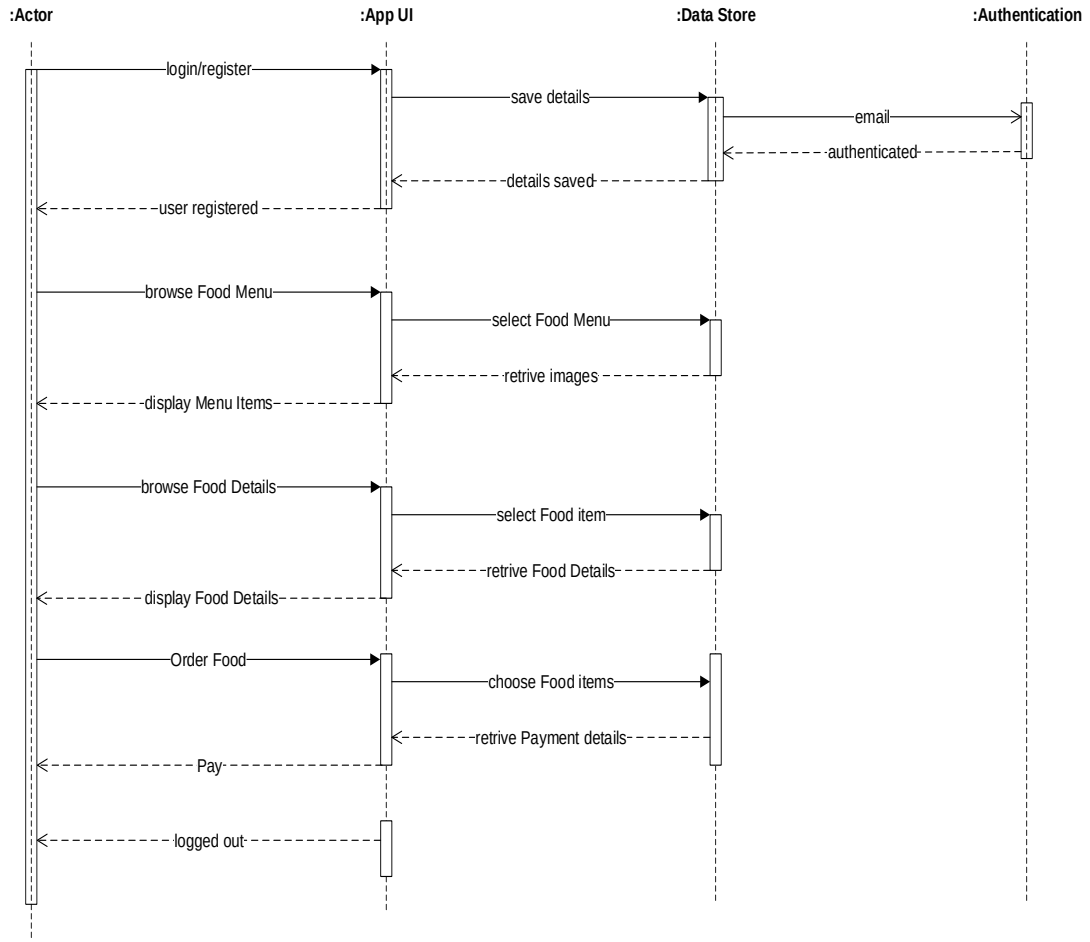


Figure 37: Sequence Diagram

5. Testing

In order to complete the whole system development, the quality of the application should be checked and should be verified whether the system works smoothly or not. It is one of the important stages in Software Development Life Cycle. During the testing phase, the developer ensures whether the system has any further bugs or not.

There are various testing methods that can be implemented while developing the system. These testing methods have their own advantages and disadvantages. White box testing, Black box testing, Unit testing, Functional testing are some of the popular testing methods.

For this project, I have decided to do Black box testing because it is easy and this method can be performed at any time during the development of the system.

5.1. Testing Plan

Table 2: Test Plan

Test Case No.	Objective
1.	To test whether splash screen is displayed while installing the application.
2.	To test whether login screen is properly shown are not.
3.	To test whether register page shows after clicking on Register Now.
4.	To check whether registration takes null value or not.
5.	To check whether registration takes invalid email.
6.	To check whether registration takes previously entered email.
7.	To check whether login takes null value or not.
8.	To check whether email is sent for email validation or not.
9.	To check whether home page displays list of food items fetched from server or not.
10.	To check whether all the navigation button works or not.
11.	To check whether clicking on a food item opens page to order the food item or not.
12.	To check whether clicking on cart adds item to cart or not.

13.	To check whether clicking on item on the cart opens page to proceed the order checkout or not.
14.	To check whether the item added on the cart can be updated or not.
15.	To check whether the item added on the cart can be deleted or not.
16.	To check whether clicking on cash on delivery radio button and clicking on proceed opens the page to add delivery information or not.
17.	To check whether order is confirmed and saved on database or not.
18.	To check whether clicking on pay through khalti opens khalti or not.
19.	To check whether payment is confirmed through khalti or not.
20.	To check whether password can be updated or not.
21.	To check whether about page is shown or not.
22.	To check whether the links in the about page are working or not.
23.	To check whether logout button works or not.

5.2. Test Cases

Table 3: Test Table 1

Test Case No.	1
Objective	To test whether splash screen is displayed while installing the application.
Expected Result	Splash screen should be displayed.
Actual Result	Splash screen is displayed.
Conclusion	Test Successful.

Table 4: Test Table 2

Test Case No.	2
Objective	To test whether login screen is properly shown are not.
Expected Result	Login page should be displayed.
Actual Result	Login page was displayed.
Conclusion	Test Successful.

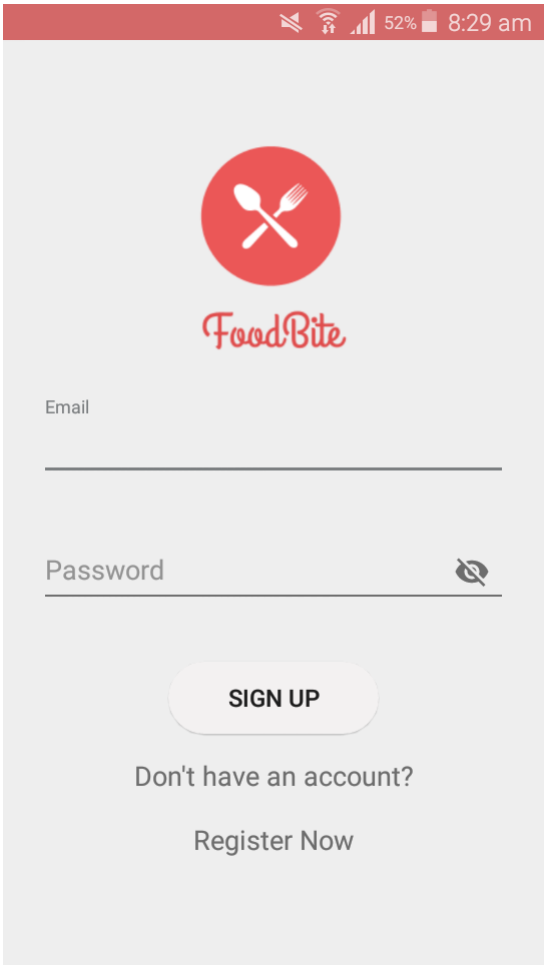


Table 5: Test Table 3

Test Case No.	3
Objective	To test whether register page shows after clicking on Register Now.
Expected Result	Register page should be displayed.
Actual Result	Register page was displayed.
Conclusion	Test Successful.

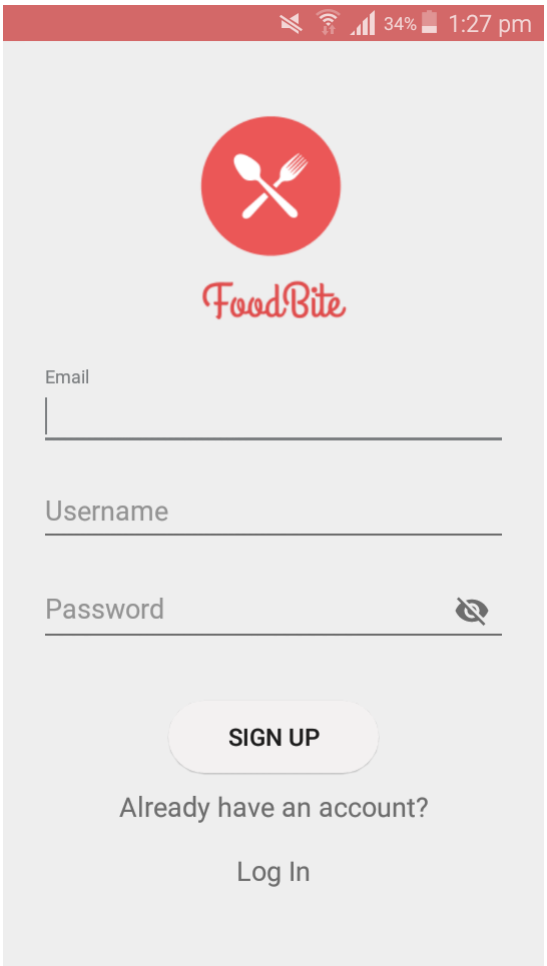


Table 6: Test Table 5

Test Case No.	5
Objective	To check whether registration takes null value or not.
Expected Result	Toast message should be displayed.
Actual Result	Toast message was displayed.
Conclusion	Test Successful.

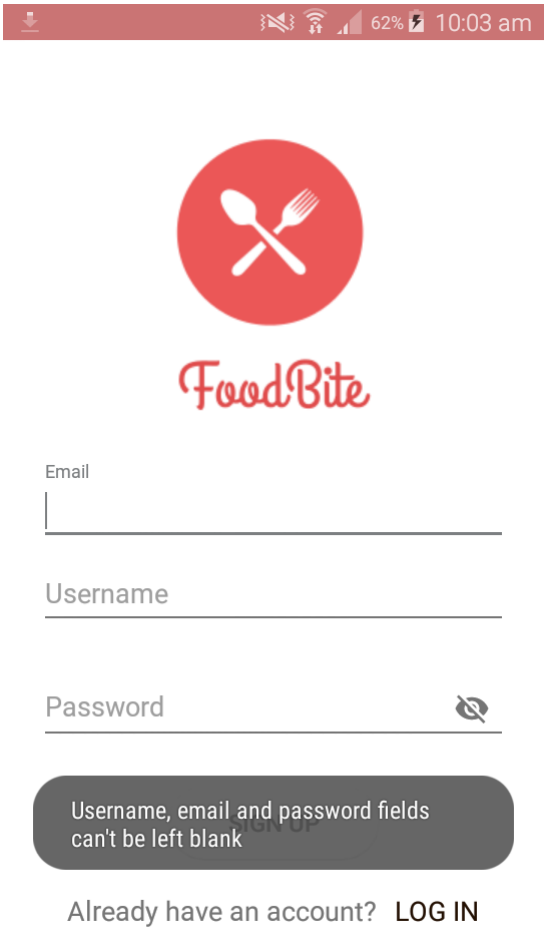


Table 7: Test Table 6

Test Case No.	6
Objective	To check whether registration takes invalid email.
Expected Result	Toast message should be shown and login page should be opened.
Actual Result	Toast message was shown and login page was opened.
Conclusion	Test Successful.

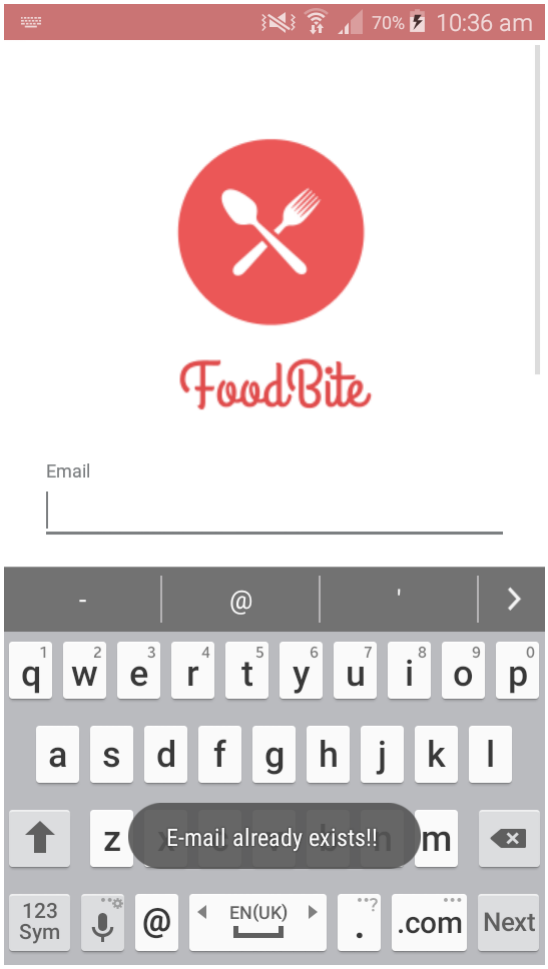


Table 8: Test Table 7

Test Case No.	7
Objective	To check whether registration takes previously entered email.
Expected Result	Toast message should be displayed and login page should be opened.
Actual Result	Toast message was shown and login page was opened.
Conclusion	Test Successful.

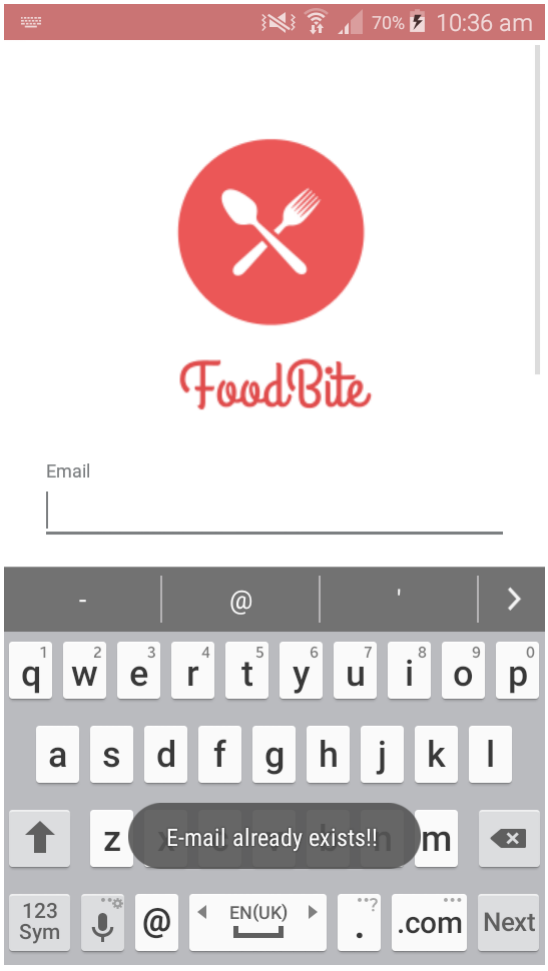


Table 9: Test Table 8

Test Case No.	8
Objective	To check whether login takes null value or not.
Expected Result	Toast message should be displayed.
Actual Result	Toast message was displayed.
Conclusion	Test Successful.

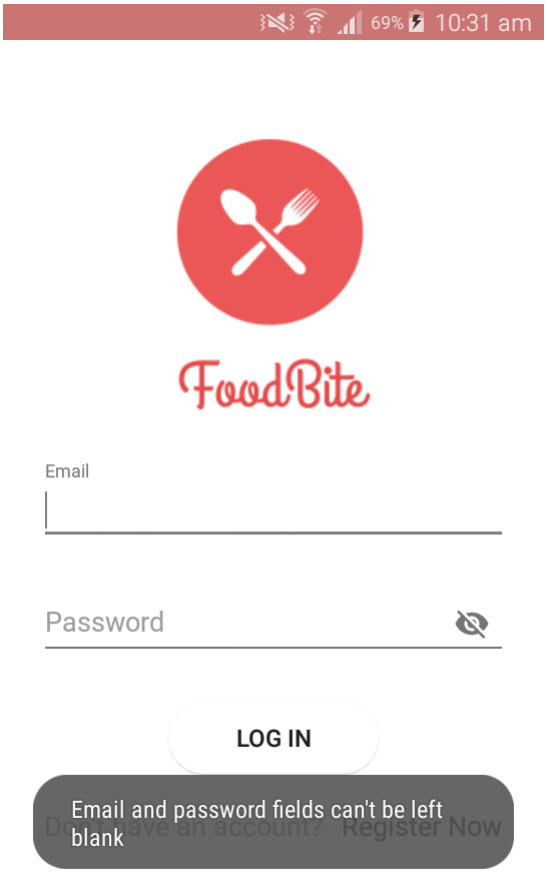
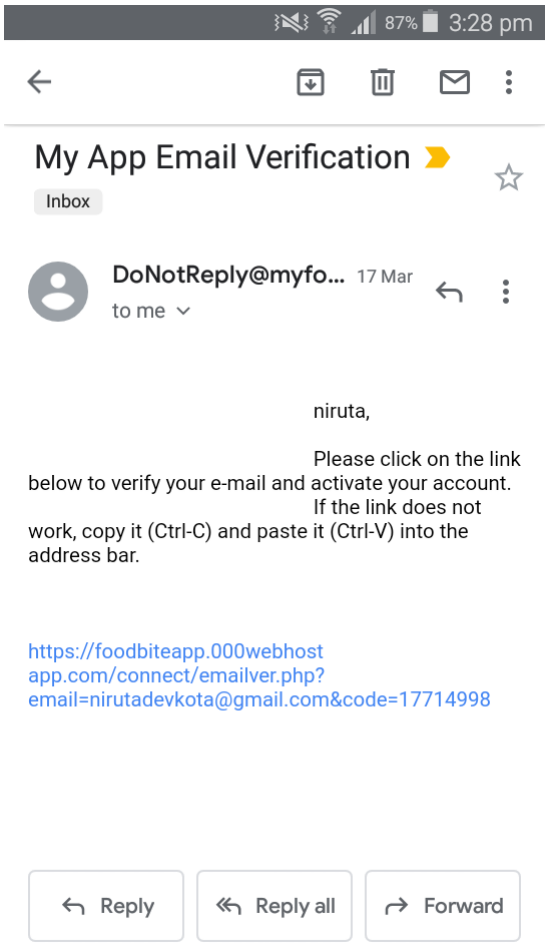
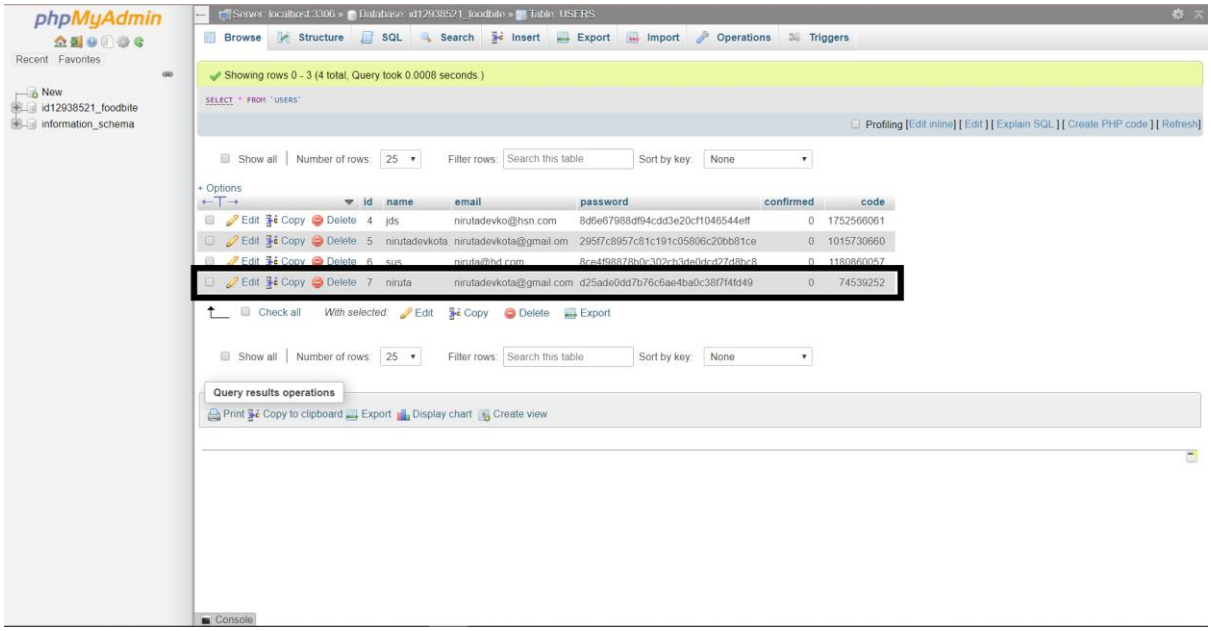


Table 10: Test Table 10

Test Case No.	9
Objective	To check whether email is sent for email validation or not.
Expected Result	Email should be sent to the designated email.
Actual Result	Email was received to valid the email address.
Conclusion	Test Successful.




phpMyAdmin
Recent Favorites
New
id12938521_foodbite
information_schema
Server: localhost:3306 » Database: id12938521_foodbite » Table: USERS
Browse Structure SQL Search Insert Export Import Operations Triggers
Showing rows 0 - 3 (4 total, Query took 0.0008 seconds.)
SELECT * FROM `USERS`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
+ Options
id name email password confirmed code
Edit Copy Delete 4 jds nirutadevko@hsn.com 8d6e67988df94cdd3e20cf1046544eff 0 1752566061
Edit Copy Delete 5 nirutadevkota nirutadevkota@gmail.com 295f7c8957c81c191c05806c20bb81ce 0 1015730660
Edit Copy Delete 6 sus niruta@hd.com 8ce4f98878b0c302cb3de0dcd27d8bc8 0 1180860057
Edit Copy Delete 7 niruta nirutadevkota@gmail.com d25ade0dd7b76c6ae4ba0c38f7f4fd49 0 74539252
Check all With selected: Edit Copy Delete Export
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Query results operations
Print Copy to clipboard Export Display chart Create view
Console

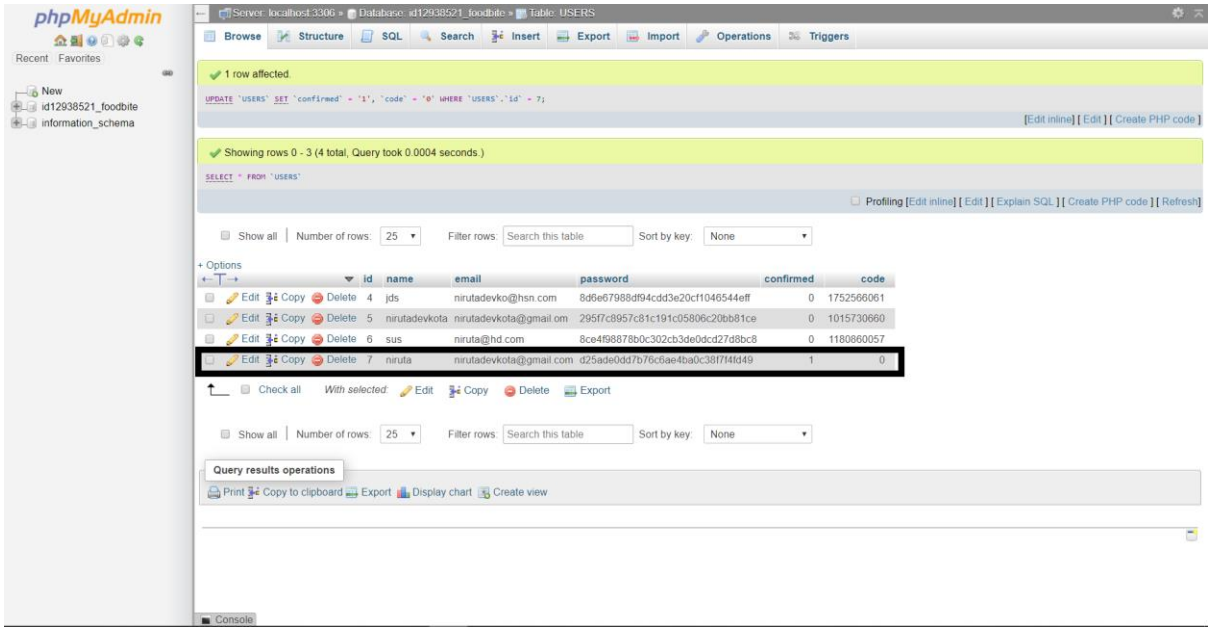

phpMyAdmin
Recent Favorites
New
id12938521_foodbite
information_schema
Server: localhost:3306 » Database: id12938521_foodbite » Table: USERS
Browse Structure SQL Search Insert Export Import Operations Triggers
1 row affected.
UPDATE `USERS` SET `confirmed` = '1', `code` = '0' WHERE `USERS`.`id` = 7;
[Edit inline] [Edit] [Create PHP code]
Showing rows 0 - 3 (4 total, Query took 0.0004 seconds.)
SELECT * FROM `USERS`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
+ Options
id name email password confirmed code
Edit Copy Delete 4 jds nirutadevko@hsn.com 8d6e67988df94cdd3e20cf1046544eff 0 1752566061
Edit Copy Delete 5 nirutadevkota nirutadevkota@gmail.com 295f7c8957c81c191c05806c20bb81ce 0 1015730660
Edit Copy Delete 6 sus niruta@hd.com 8ce4f98878b0c302cb3de0dcd27d8bc8 0 1180860057
Edit Copy Delete 7 niruta nirutadevkota@gmail.com d25ade0dd7b76c6ae4ba0c38f7f4fd49 1 0
Check all With selected: Edit Copy Delete Export
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Query results operations
Print Copy to clipboard Export Display chart Create view
Console

Table 11: Test Case 10

Test Case No.	10
Objective	To check whether home page displays list of food items fetched from server or not.
Expected Result	The food items are expected to be shown on home page.
Actual Result	The food items were successfully fetched and shown on the home page.
Conclusion	Test Successful.

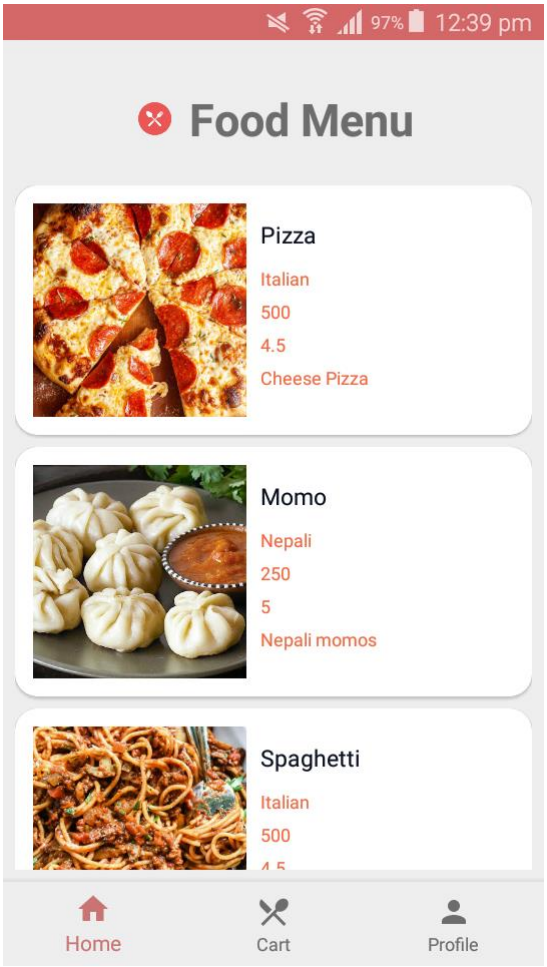


Table 12: Test Case 11

Test Case No.	11
Objective	To check whether all the navigation button works or not.
Expected Result	The navigation buttons are expected to be working.
Actual Result	The buttons worked as expected.
Conclusion	Test Successful.

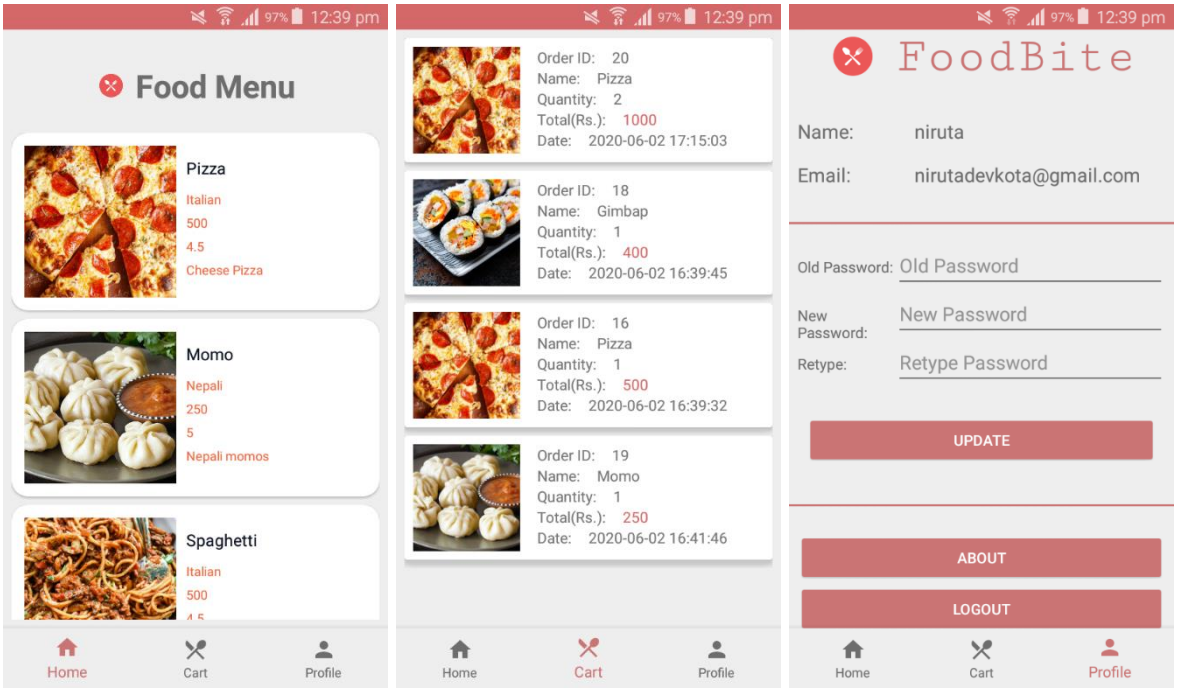


Table 13: Test Case 12

Test Case No.	12
Objective	To check whether clicking on a food item opens page to order the food item or not.
Expected Result	The food order page is expected to be shown after clicking on a food item.
Actual Result	The food order page was shown after clicking on food item.
Conclusion	Test Successful.

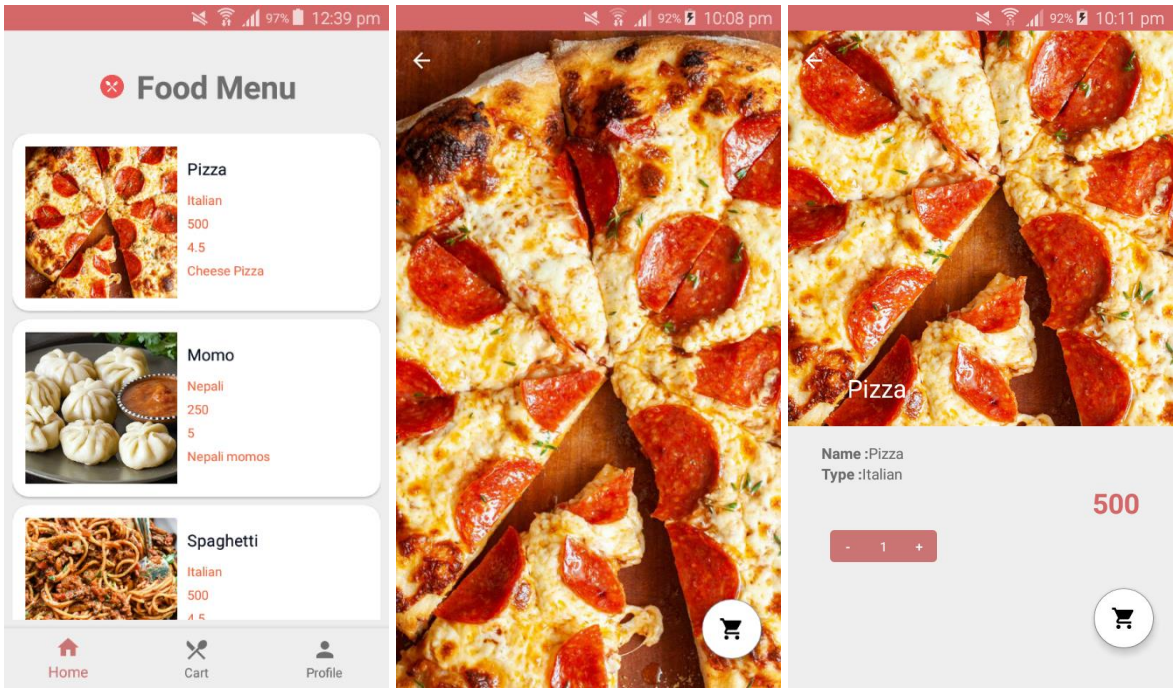
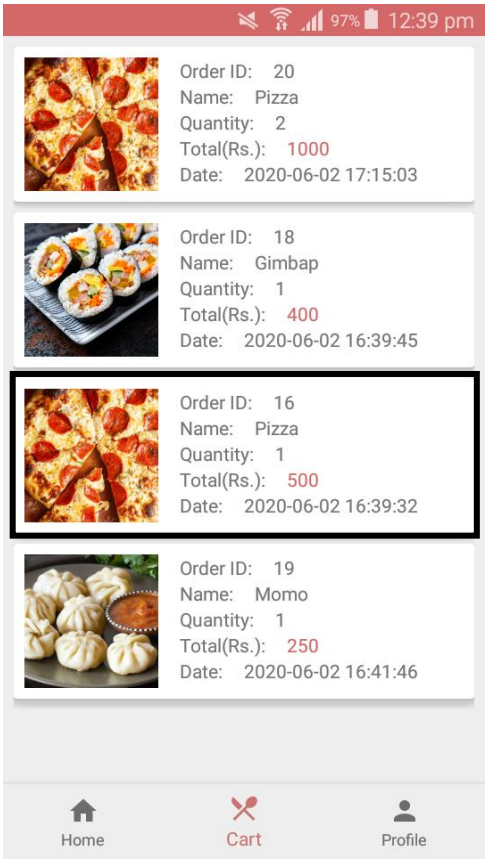


Table 14: Test Case 13

Test Case No.	13
Objective	To check whether clicking on item on the cart opens page to proceed the order checkout or not.
Expected Result	The page to select method of payment should be opened.
Actual Result	The page to select method of payment was opened.
Conclusion	Test Successful.



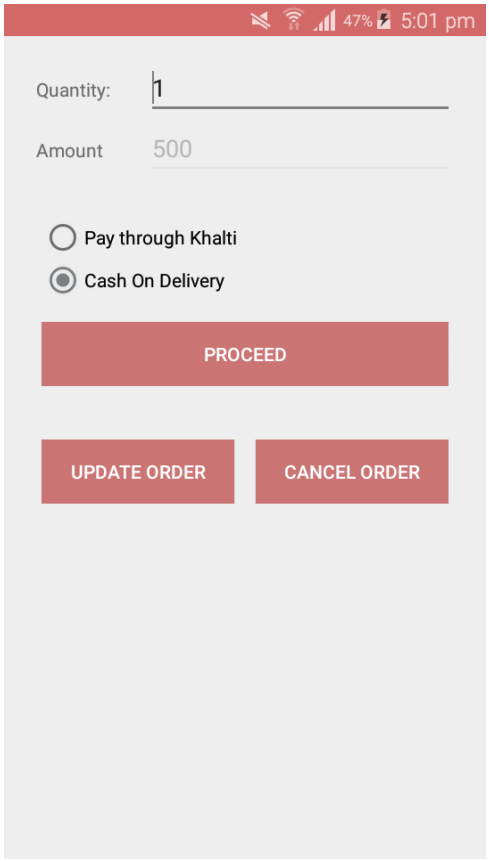


Table 15: Test Case 14

Test Case No.	14
Objective	To check whether the item added on the cart can be updated or not.
Expected Result	The item added on the cart should be updated.
Actual Result	Item added on cart was updated.
Conclusion	Test Successful.

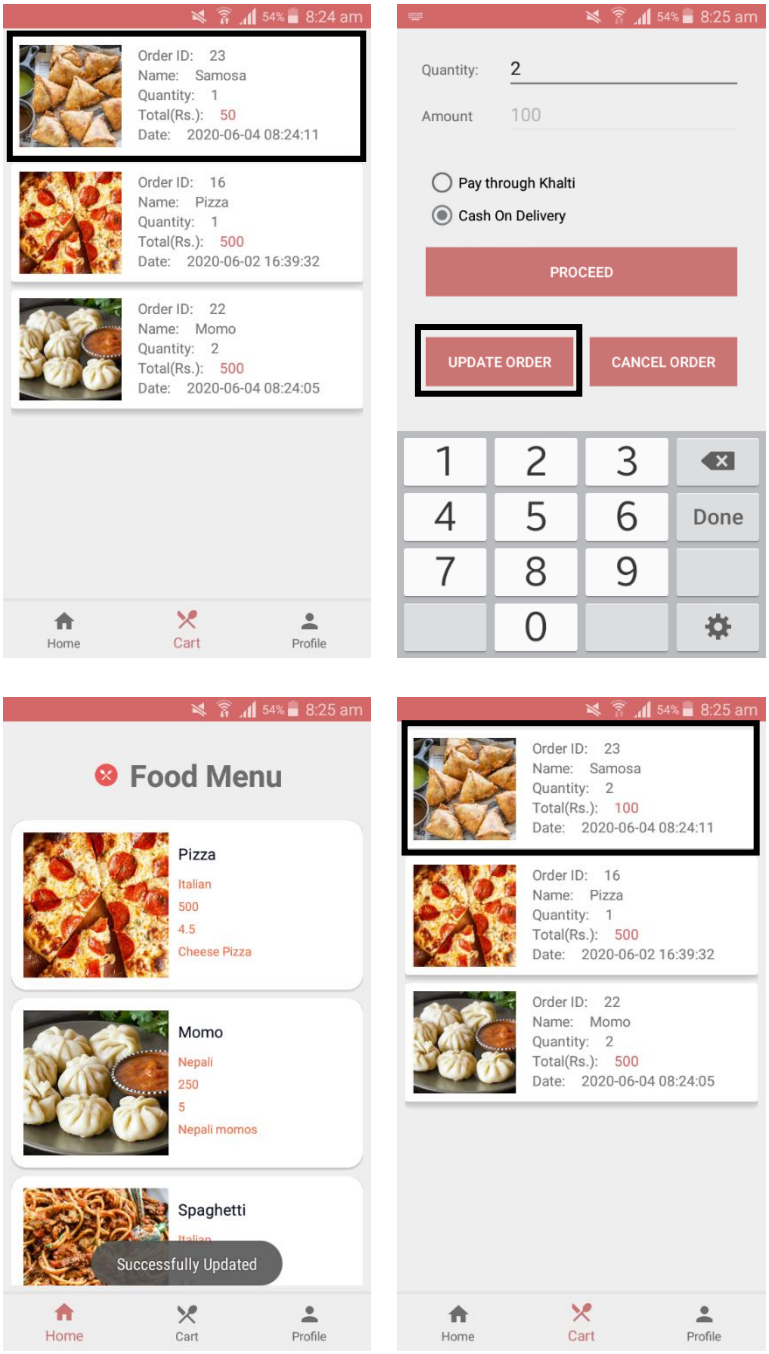
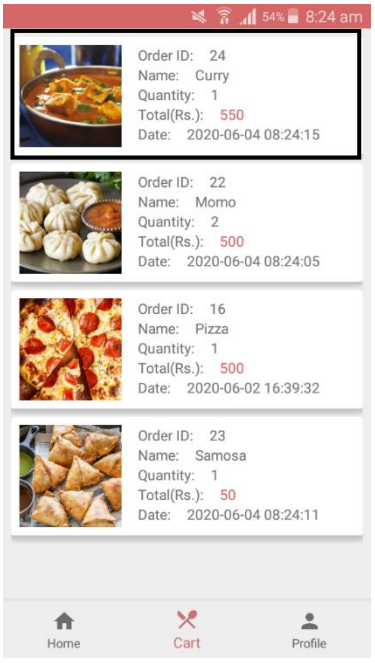
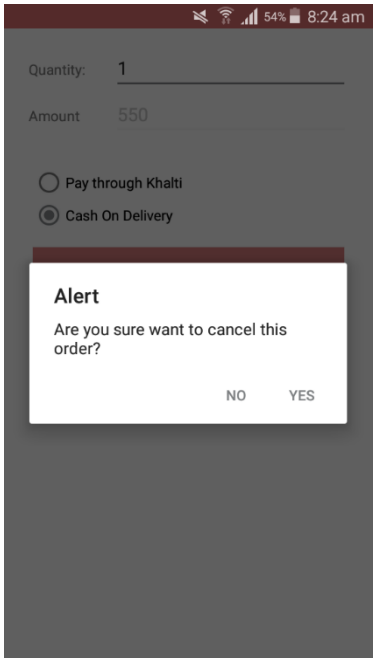
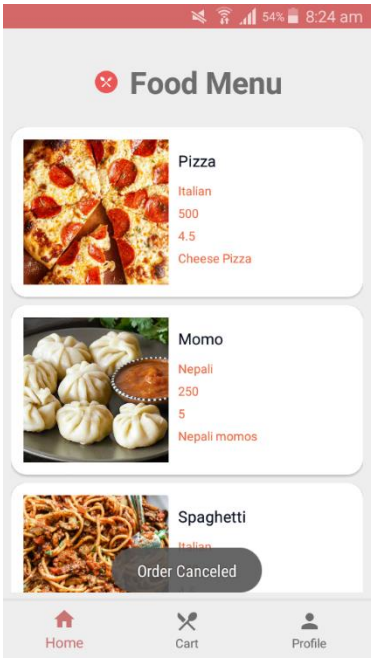


Table 16: Test Case 15

Test Case No.	15
Objective	To check whether the item added on the cart can be deleted or not.
Expected Result	The item should be deleted from cart.
Actual Result	Item was deleted from cart.
Conclusion	Test Successful.







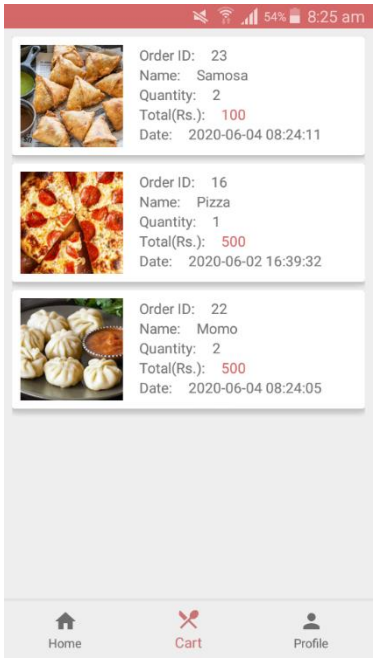


Table 17: Test Case 16

Test Case No.	16
Objective	To check whether clicking on cash on delivery radio button and clicking on proceed opens the page to add delivery information or not.
Expected Result	The delivery information page should be opened.
Actual Result	The delivery information page was opened.
Conclusion	Test Successful.

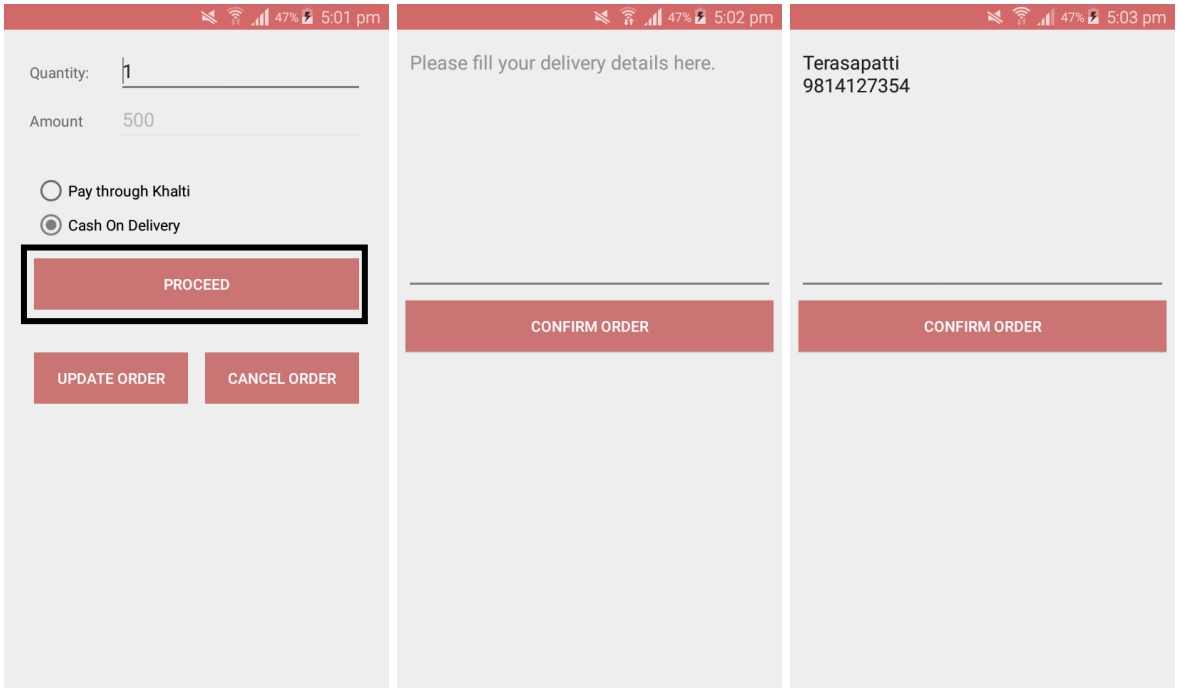
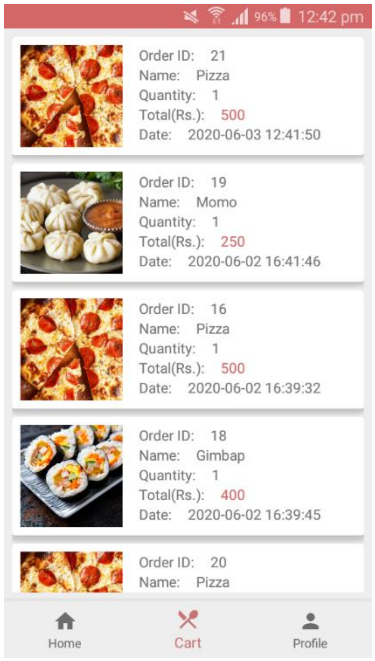
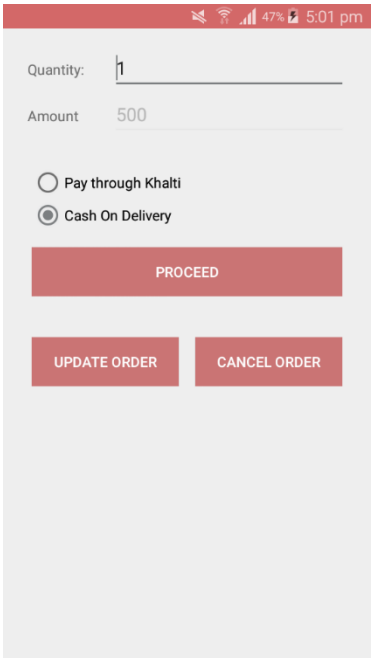
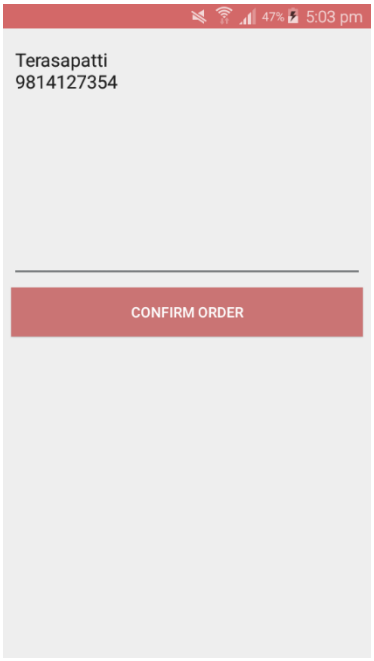


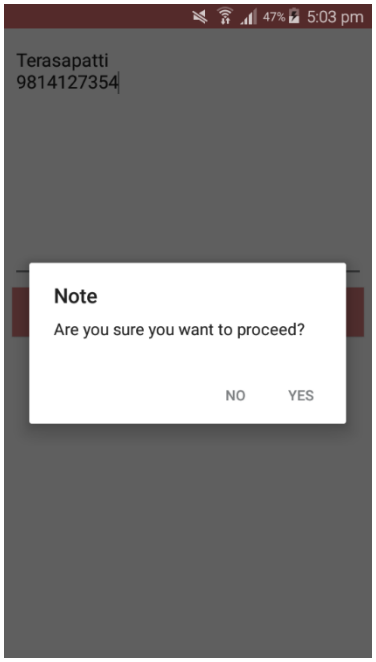
Table 18: Test Case 17

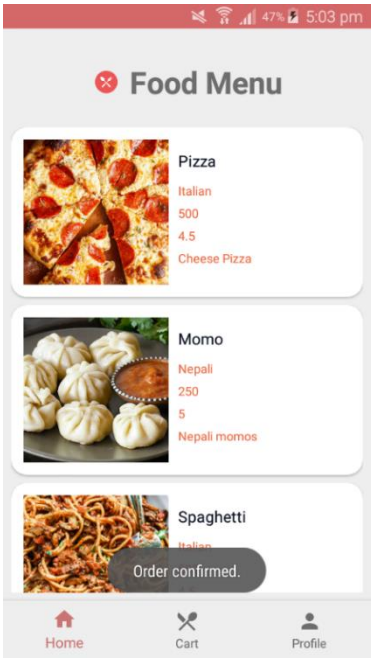
Test Case No.	17
Objective	To check whether order is confirmed and saved on database or not.
Expected Result	The order should be confirmed and should be saved on database.
Actual Result	The order was confirmed and saved in database.
Conclusion	Test Successful.

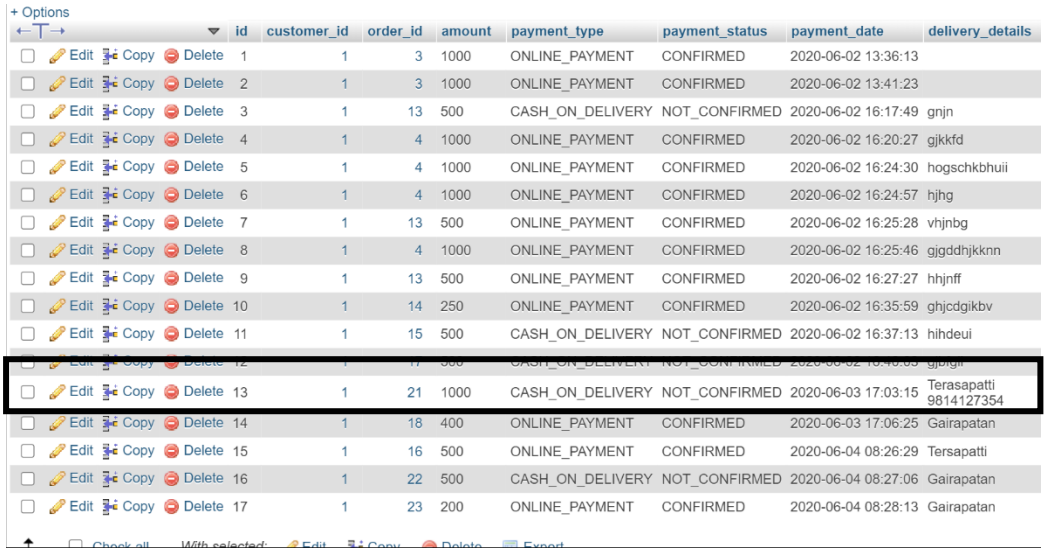






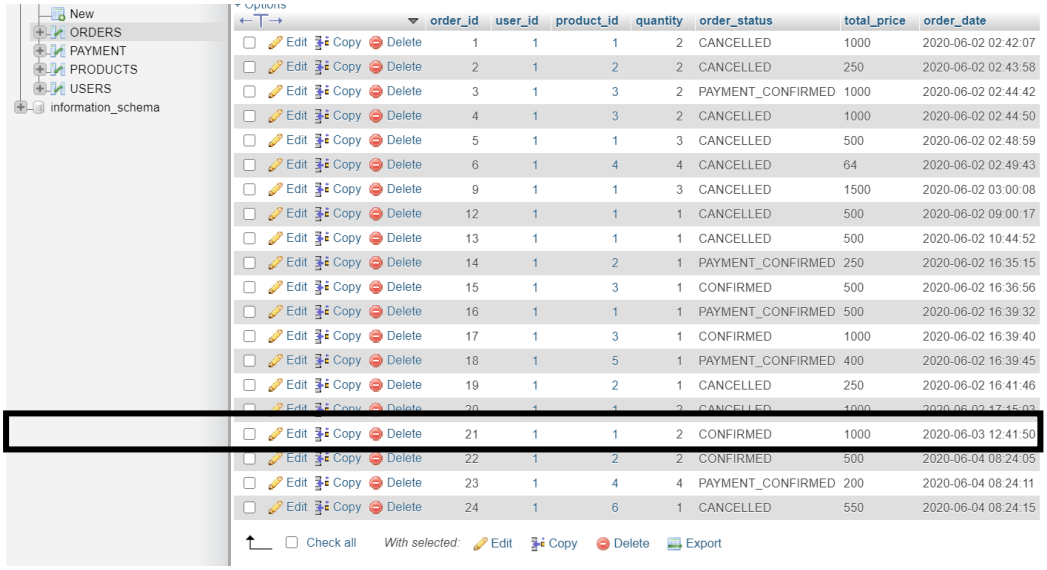






+ Options

	id	customer_id	order_id	amount	payment_type	payment_status	payment_date	delivery_details
Edit Copy Delete	1	1	3	1000	ONLINE_PAYMENT	CONFIRMED	2020-06-02 13:36:13	
Edit Copy Delete	2	1	3	1000	ONLINE_PAYMENT	CONFIRMED	2020-06-02 13:41:23	
Edit Copy Delete	3	1	13	500	CASH_ON_DELIVERY	NOT_CONFIRMED	2020-06-02 16:17:49	gnjn
Edit Copy Delete	4	1	4	1000	ONLINE_PAYMENT	CONFIRMED	2020-06-02 16:20:27	gjkkfd
Edit Copy Delete	5	1	4	1000	ONLINE_PAYMENT	CONFIRMED	2020-06-02 16:24:30	hogschkbhuii
Edit Copy Delete	6	1	4	1000	ONLINE_PAYMENT	CONFIRMED	2020-06-02 16:24:57	hjhg
Edit Copy Delete	7	1	13	500	ONLINE_PAYMENT	CONFIRMED	2020-06-02 16:25:28	vhjnbg
Edit Copy Delete	8	1	4	1000	ONLINE_PAYMENT	CONFIRMED	2020-06-02 16:25:46	gjgddhjkknn
Edit Copy Delete	9	1	13	500	ONLINE_PAYMENT	CONFIRMED	2020-06-02 16:27:27	hhjnff
Edit Copy Delete	10	1	14	250	ONLINE_PAYMENT	CONFIRMED	2020-06-02 16:35:59	ghjcdgikbv
Edit Copy Delete	11	1	15	500	CASH_ON_DELIVERY	NOT_CONFIRMED	2020-06-02 16:37:13	hihdeui
Edit Copy Delete	13	1	21	1000	CASH_ON_DELIVERY	NOT_CONFIRMED	2020-06-03 17:03:15	Terasapatti 9814127354
Edit Copy Delete	14	1	18	400	ONLINE_PAYMENT	CONFIRMED	2020-06-03 17:06:25	Gairapatan
Edit Copy Delete	15	1	16	500	ONLINE_PAYMENT	CONFIRMED	2020-06-04 08:26:29	Tersapatti
Edit Copy Delete	16	1	22	500	CASH_ON_DELIVERY	NOT_CONFIRMED	2020-06-04 08:27:06	Gairapatan
Edit Copy Delete	17	1	23	200	ONLINE_PAYMENT	CONFIRMED	2020-06-04 08:28:13	Gairapatan



New
ORDERS
PAYMENT
PRODUCTS
USERS
information_schema

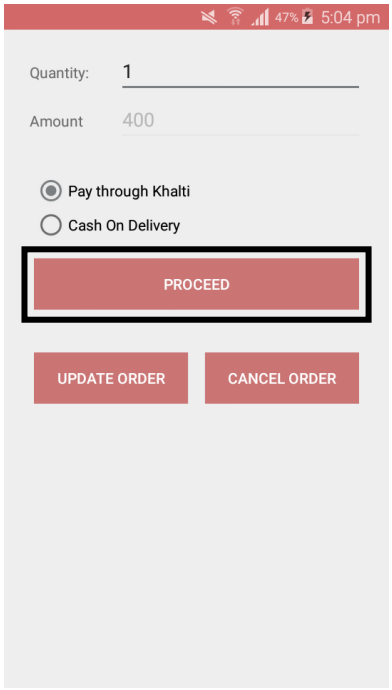
+ Options

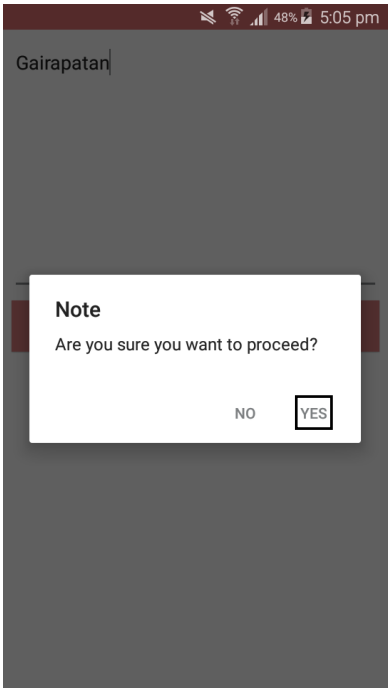
	order_id	user_id	product_id	quantity	order_status	total_price	order_date
Edit Copy Delete	1	1	1	2	CANCELLED	1000	2020-06-02 02:42:07
Edit Copy Delete	2	1	2	2	CANCELLED	250	2020-06-02 02:43:58
Edit Copy Delete	3	1	3	2	PAYMENT_CONFIRMED	1000	2020-06-02 02:44:42
Edit Copy Delete	4	1	3	2	CANCELLED	1000	2020-06-02 02:44:50
Edit Copy Delete	5	1	1	3	CANCELLED	500	2020-06-02 02:48:59
Edit Copy Delete	6	1	4	4	CANCELLED	64	2020-06-02 02:49:43
Edit Copy Delete	9	1	1	3	CANCELLED	1500	2020-06-02 03:00:08
Edit Copy Delete	12	1	1	1	CANCELLED	500	2020-06-02 09:00:17
Edit Copy Delete	13	1	1	1	CANCELLED	500	2020-06-02 10:44:52
Edit Copy Delete	14	1	2	1	PAYMENT_CONFIRMED	250	2020-06-02 16:35:15
Edit Copy Delete	15	1	3	1	CONFIRMED	500	2020-06-02 16:36:56
Edit Copy Delete	16	1	1	1	PAYMENT_CONFIRMED	500	2020-06-02 16:39:32
Edit Copy Delete	17	1	3	1	CONFIRMED	1000	2020-06-02 16:39:40
Edit Copy Delete	18	1	5	1	PAYMENT_CONFIRMED	400	2020-06-02 16:39:45
Edit Copy Delete	19	1	2	1	CANCELLED	250	2020-06-02 16:41:46
Edit Copy Delete	21	1	1	2	CONFIRMED	1000	2020-06-03 12:41:50
Edit Copy Delete	22	1	2	2	CONFIRMED	500	2020-06-04 08:24:05
Edit Copy Delete	23	1	4	4	PAYMENT_CONFIRMED	200	2020-06-04 08:24:11
Edit Copy Delete	24	1	6	1	CANCELLED	550	2020-06-04 08:24:15

Check all With selected: Edit Copy Delete Export

Table 19: Test Case 18

Test Case No.	18
Objective	To check whether clicking on pay through khalti opens khalti or not.
Expected Result	The khalti payment should be opened.
Actual Result	Khalti payment was opened.
Conclusion	Test Successful.





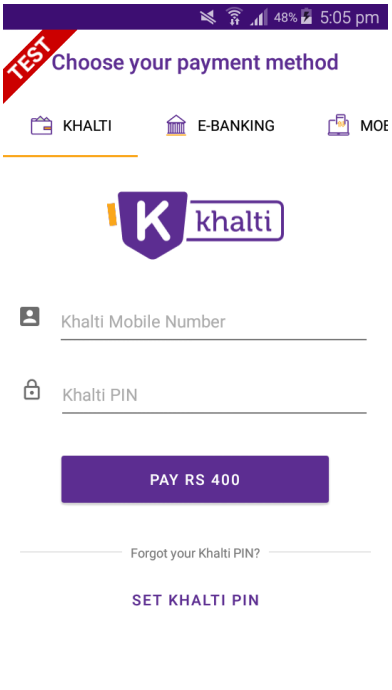
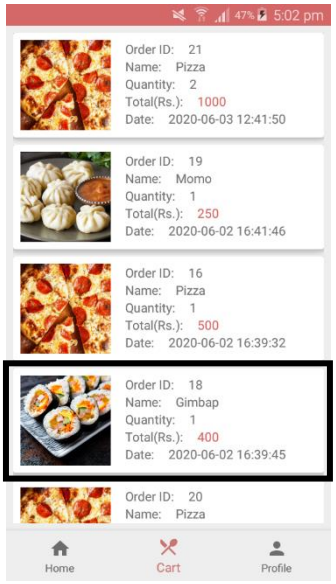
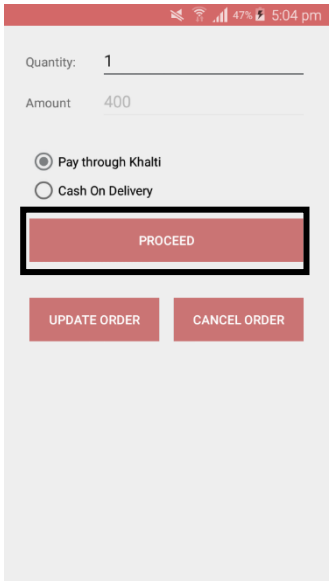
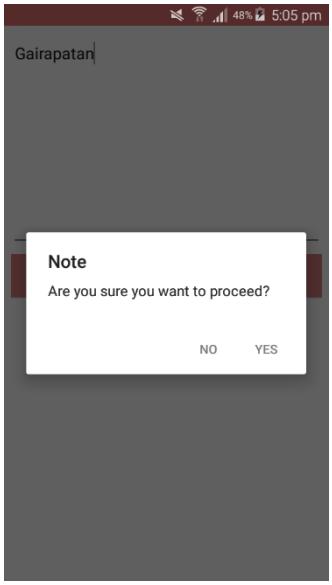


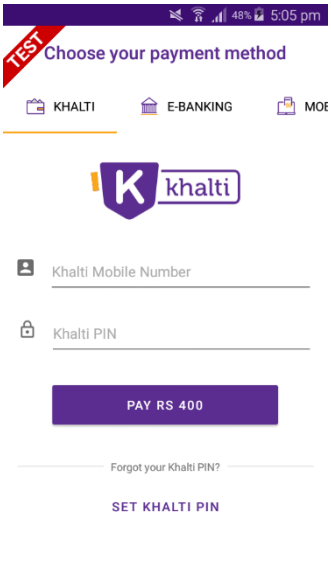
Table 20: Test Case 19

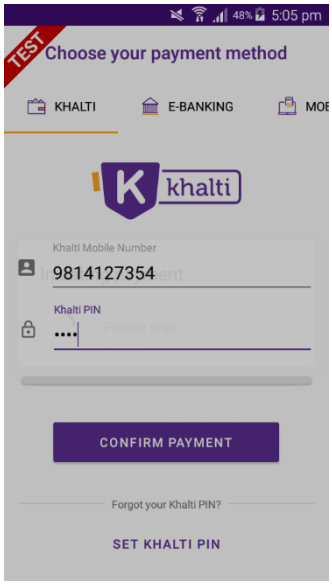
Test Case No.	19
Objective	To check whether payment is confirmed through khalti or not.
Expected Result	After filling the required fields payment should be confirmed and homepage should be opened with a toast message.
Actual Result	Payment was successful and homepage with toast message was opened.
Conclusion	Test Successful.











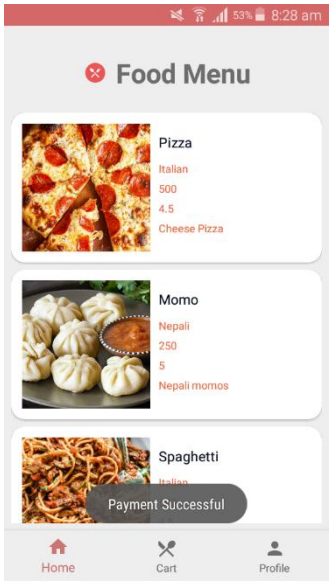


Table 21: Test Case 20

Test Case No.	20
Objective	To check whether password can be updated or not.
Expected Result	Password should be change and toast message should be shown.
Actual Result	Password was changed and toast message was shown.
Conclusion	Test Successful.

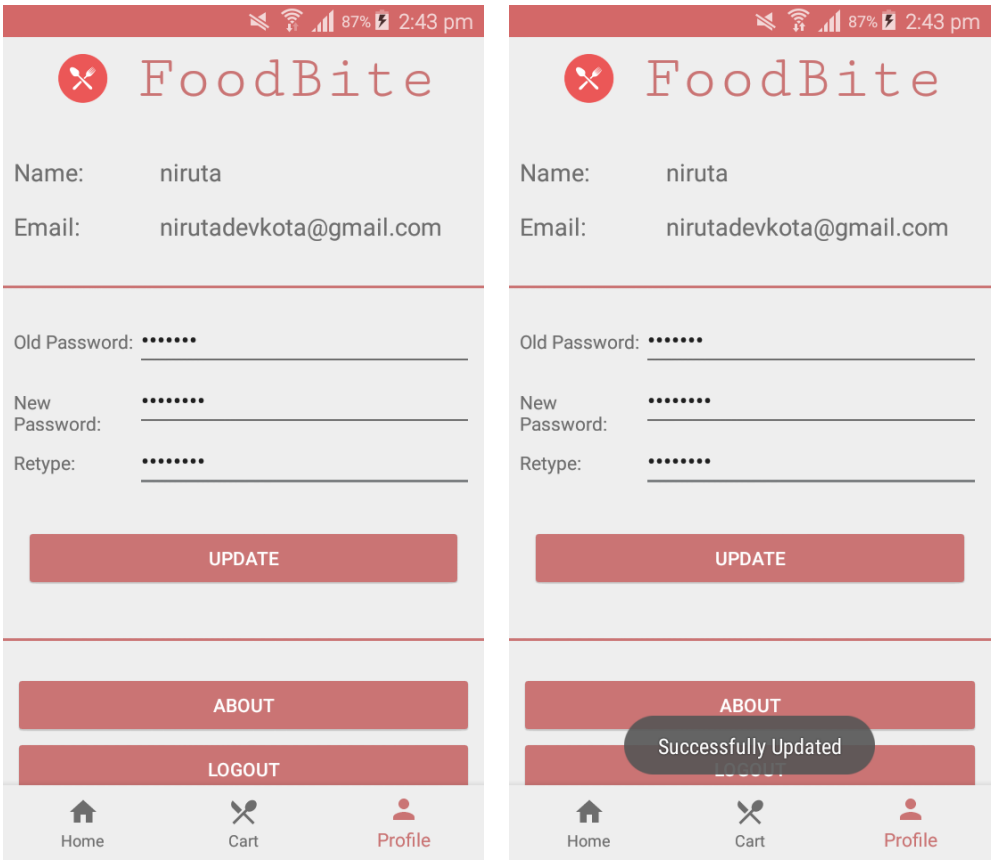
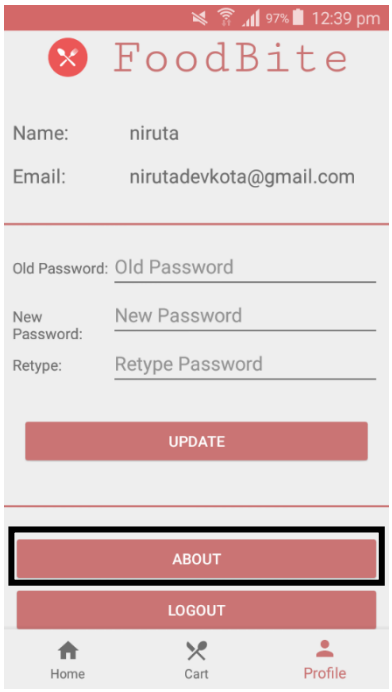
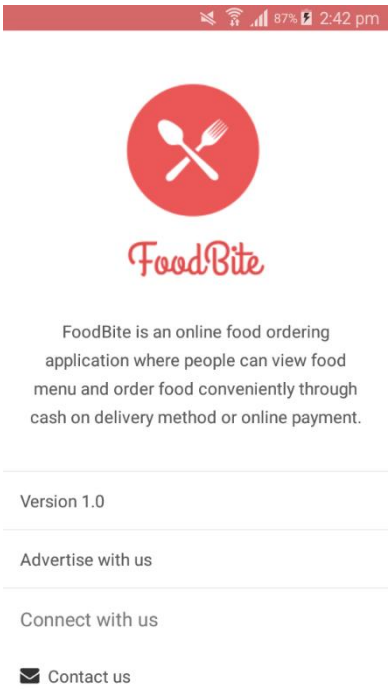


Table 22: Test Case 21

Test Case No.	21
Objective	To check whether about page is shown or not.
Expected Result	On clicking the about button about page should be shown.
Actual Result	About page was shown.
Conclusion	Test Successful.





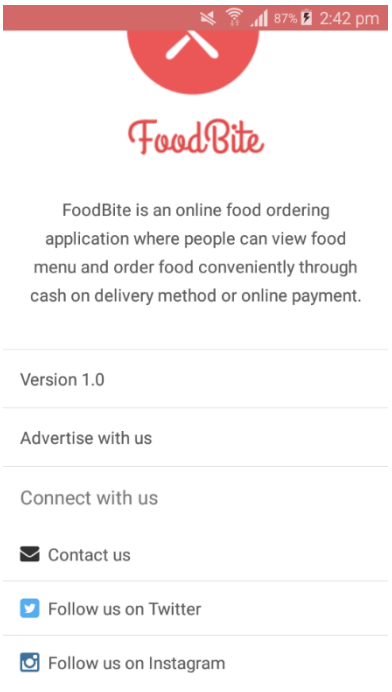
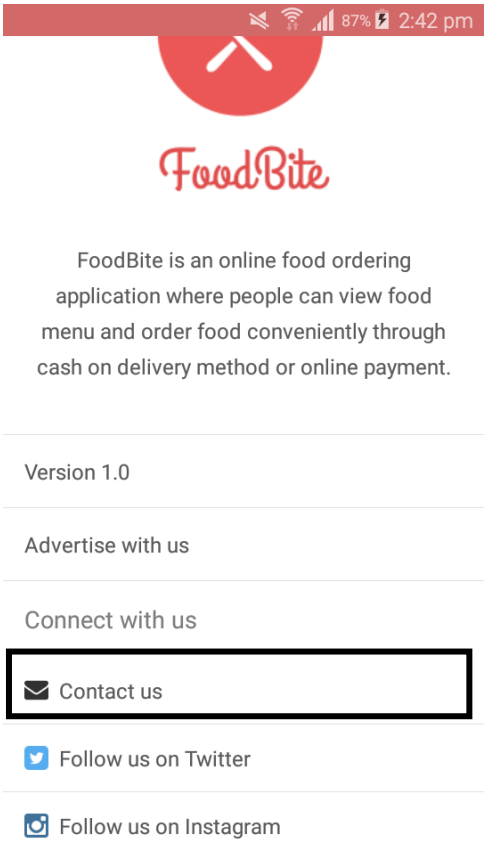


Table 23: Test Case 22

Test Case No.	22
Objective	To check whether the links in the about page are working or not.
Expected Result	If the link is clicked it should be directed to the particular link address.
Actual Result	Link was directed to the link address.
Conclusion	Test Successful.



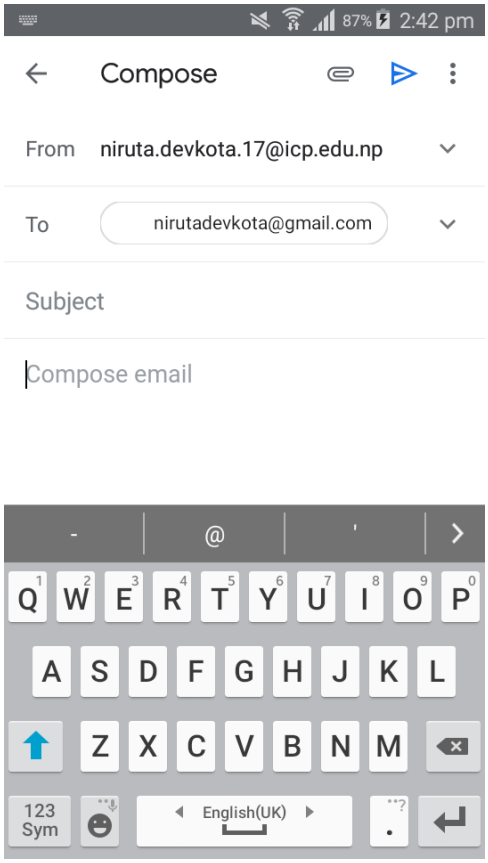
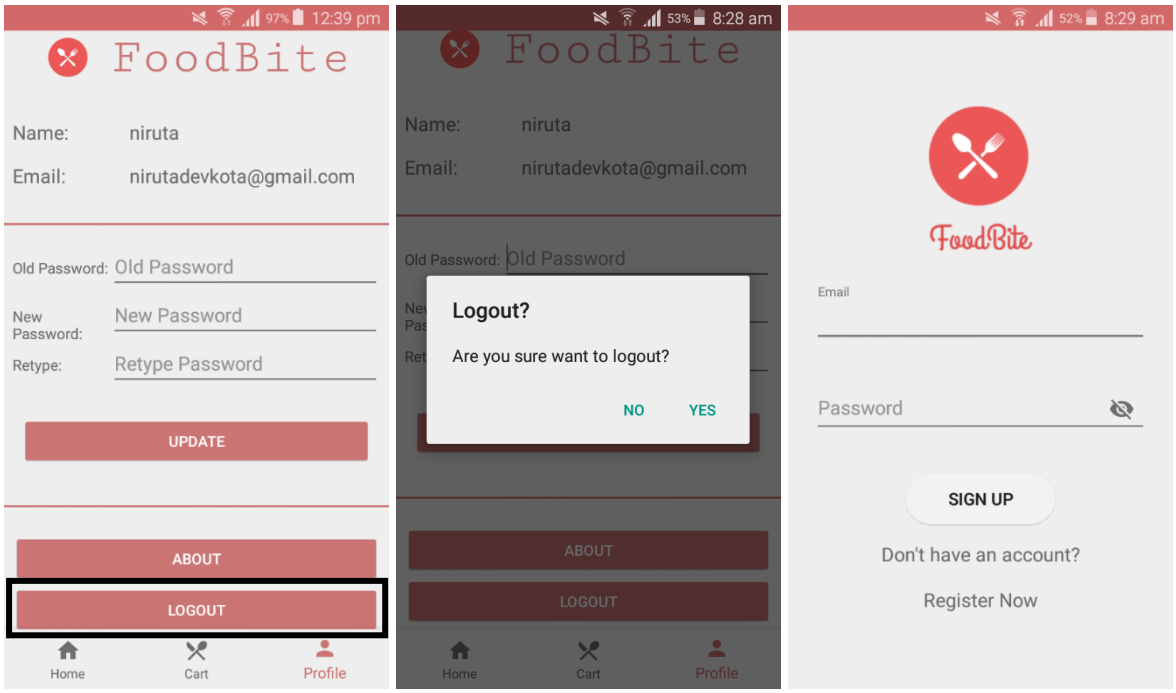


Table 24: Test Case 23

Test Case No.	23
Objective	To check whether logout button works or not.
Expected Result	A dialog box should be shown to ask whether a user really wants to logout, if yes is clicked user should be logged out and sent to login screen.
Actual Result	Dialog box was shown and clicking on yes login page was shown.
Conclusion	Test Successful.



6. Limitation

The various things I had planned to do for this project were not implemented in this application due to various reasons. So, the project is simple to use and lacks the following features;

- The application requires internet to function. It is an online application.
- Favourite and order history functions are not implemented.

7. Conclusion

With the increase in use of technology, everyone is craving for convenient and reliable methods to perform their daily tasks. They want very thing to be accessible with the touch of their fingers and with the increase in reliability on smartphones people want to order their food with as much less effort as possible. The developed application is just the solution to the problem.

The application FoodBite is easy to use for both beginners and advance users. Users can register to the application, view food menu add the food item to the cart, update and delete the order and pay for the order through cash on delivery and online payment method. It is a simple food ordering application which is developed to make food ordering and consumption easy and reliable.

The development of the project was a big challenge but I am very satisfied with the final outcome of the project. It required a lot of research and learning to make the application a success and the final outcome is the result of assistance and guidance from many people who helped me understand difficult topics and helped me understand the problems I was facing more clearly. All the supervision, guidance and research helped me complete the project in time and achieve the required result. I would like to express my gratitude to my supervisors, family and friends for their constant support. This wouldn’t be possible without them. Overall, the development of the project was informative with some pressure and I was able to learn a lot during the span of the development.

8. Reference List

Ambler, S. W., 2004. Chapter 5: Usage Modeling. In: S. W. Ambler, ed. *The Object Primer 3rd Edition: Agile Model Driven Development with UML 2.0.* s.l.:Cambridge University Press; 3 edition, p. 572.

Applikey, 2018. *Complete Guide on Food Delivery App Development.* [Online] Available at: https://applikeysolutions.com/blog/complete-guide-on-food-delivery-app-development
[Accessed 18 November 2019].

Boulanger, J.-L., 2017. *Learn more about Iterative Approach.* [Online] Available at: https://www.sciencedirect.com/topics/computer-science/iterative-approach
[Accessed 19 November 2019].

Deliveroo, n.d. s.l.:s.n.

foodpanda, n.d. s.l.:s.n.

Gary Marseden, A. M. M. P., 2008. People are people, but technology. *Philosophical transactions. Series A, Mathematical, physical, and engineering,* Volume 366, pp. 3795-804.

Graham, J., 2015. *USA TODAY.* [Online] Available at: https://www.usatoday.com/story/tech/2015/10/15/comparing-food-delivery-apps-uber-yelp-others/73979148/
[Accessed 12 November 2019].

Harvey, I., 2019. *Food Delivery: The Epic History of Humanity's Greatest Convenience.* [Online] Available at: https://www.thevintagenews.com/2019/01/08/food-delivery/
[Accessed 27 November 2019].

Lucidchart, 2020. *What is an Entity Relationship Diagram (ERD)?.* [Online] Available at: https://www.lucidchart.com/pages/er-diagrams#section_0
[Accessed 3 May 202].

Mattia, D., 2018. *THE zebra.* [Online] Available at: https://www.thezebra.com/insurance-news/5623/on-demand-food-

delivery-services/
[Accessed 12 November 2019].

Natnicha Suthumchai, S. T. P. Y. S. T., 2016. FoodForCare: An Android application for self-care with healthy food. In: *2016 Fifth ICT International Student Project Conference (ICT-ISPC).* s.l.:IEEE, pp. 89-92.

Planbox, 2019. *What is Iterative Development and How does it Work?.* [Online] Available at: https://www.planbox.com/what-is-iterative-development-and-how-does-it-work/
[Accessed 20 March 2020].

Play Store, n.d. s.l.:s.n.

Powell-Morse, A., 2016. *Iterative Model: What Is It And When Should You Use It?.* [Online]
Available at: https://airbrake.io/blog/sdlc/iterative-model
[Accessed 18 November 2019].

R.Adithya, A. S. S. P. V. K., 2017. Online Food Ordering System. *International Journal of Computer Applications,* 180(6), pp. 22-24.

Revolvy, 2018. *Online food ordering.* [Online] Available at: https://www.revolvy.com/page/Online-food-ordering?cr=1
[Accessed 29 July 2019].

Ricky, M. Y., 2014. Mobile food ordering application using android os platform. In: *EPJ Web of Conferences.* s.l.:EDP Sciences, p. 00041.

S.M. Takalkar, D. P. K. A. S. H. R. H. B., 2016. Android Application for Local Food Ordering System. *IJCSN International Journal of Computer Science and Network,* Volume 5, pp. 215-217.

Sharma, S., 2018. *CREDENCYS.* [Online] Available at: https://www.credencys.com/blog/food-delivery-app-development/
[Accessed 11 November 2019].

Shobhit Goyal, M. B., 2011. Design and Implementation of Digital Dining in Restaurants using Android. In: s.l.:s.n., pp. 684-686.

smartdraw, 2020. *Activity Diagram.* [Online]
Available at: https://www.smartdraw.com/activity-diagram/#whatisActivityDiagram
[Accessed 20 April 2020].

softwaretestinghelp, 2020. *SDLC (Software Development Life Cycle) Phases, Methodologies, Process, And Models.* [Online]
Available at: https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/
[Accessed 18 November 2019].

Sohail, A., 2019. *Agile Project Management — Part 1: Understanding and Benefits.* [Online]
Available at: https://medium.com/@masads/agile-project-management-part-1-understanding-and-benefits-eb079b563288
[Accessed 19 November 2019].

STACKIFY, 2017. *What is SDLC? Understand the Software Development Life Cycle.* [Online]
Available at: https://stackify.com/what-is-sdlc/
[Accessed 18 November 2019].

studytonight, 2020. *https://airbrake.io/blog/sdlc/iterative-model.* [Online]
Available at: https://www.studytonight.com/3d-game-engineering-with-unity/game-development-models
[Accessed 19 November 2019].

Swiggy, n.d. s.l.:s.n.

technopedia, 2011. *Data Dictionary.* [Online]
Available at: https://www.techopedia.com/definition/27752/data-dictionary
[Accessed 20 May 2020].

tutorialspoint, 2020. *UML- Class Diagram.* [Online]
Available at: https://www.tutorialspoint.com/uml/uml_class_diagram.htm
[Accessed 1 June 2020].

UberEats, n.d. s.l.:s.n.

Visual Paradigm, 2020. *What is sequence Diagram?.* [Online] Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/
[Accessed 22 May 2020].

william, 2019. *The Relevance And Growth of Food Delivery Business.* [Online] Available at: https://hackernoon.com/the-relevance-and-growth-of-food-delivery-business-0zx32nj
[Accessed 2 December 2019].

WOXAPP, 2018. *WOXAPP.* [Online] Available at: https://woxapp.com/our-blog/to-create-an-application-the-similar-ubereats/
[Accessed 13 November 2019].

Zomato, n.d. s.l.:s.n.

9. Appendix

9.1. Wireframe

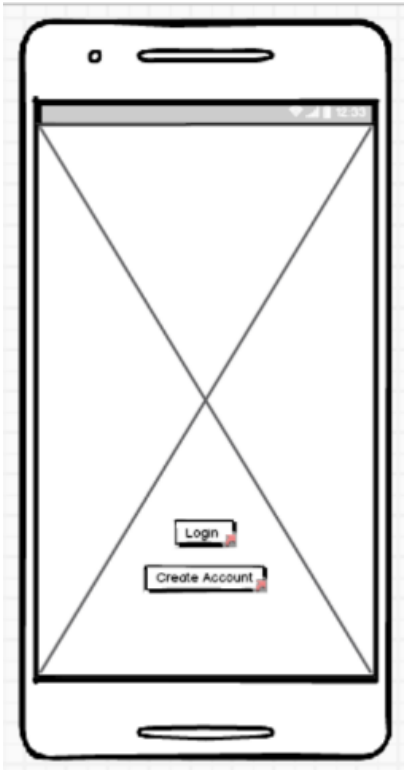


Figure 38: Register Wireframe

Figure 39: Register Wireframe

Figure 40: Login Wireframe

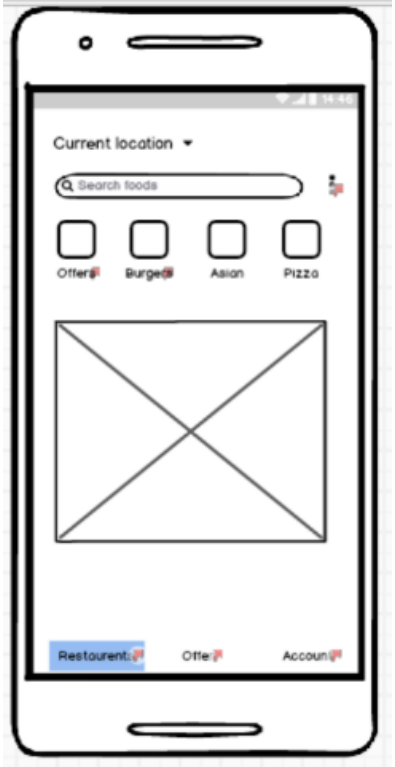


Figure 41: Homepage wireframe

Figure 42: Forgot password Wireframe

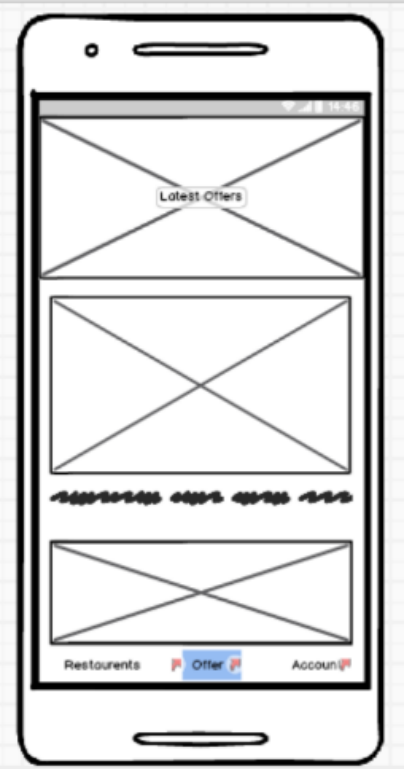


Figure 43: Offer Wireframe

Figure 44: Cart Wireframe

Figure 45: Order Details Wireframe

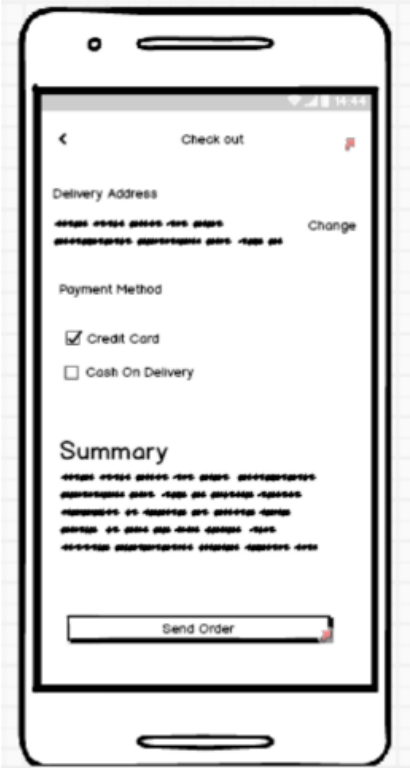


Figure 46: Check out Wireframe

Figure 47: Account Wireframe

9.2. Software Requirement Specification

Functional Requirements

Registration and Login:

If customer wants to view menu and order the food then they must be registered, unregistered user can't order. The customer login to the system by entering valid username and password. If invalid username or password is entered error message is shown. The user should also be connected to the internet to use the application.

Display Food Menu:

In the application, all the food items are displayed with their image, name, category, description, rating and price from the database.

Select Food Item:

Customer can click on the menu item and select the desired quantity of food item and click on cart icon to add the food item to cart.

Change requirements:

Customer can change the quantity of the food item or delete the food item anytime.

Order:

Customer can select the food item they want to order and choose the method of payment to order.

Delivery Details:

Customer need to feel up delivery details before confirming payment.

Change Password:

Customer can change their password anytime.

About:

Customer can view the about page and contact the developer if they have any queries.

Logout:

Customer can logout from the application after payment or surf the products more.

Non Functional Requirements

Performance

When user opens the application the application load food menu within 5 seconds with all the thumbnail images.

Responsiveness

When user selects any food they can easily add the food to cart and mention the quantity of the food, and can customize with few touches instantaneously.

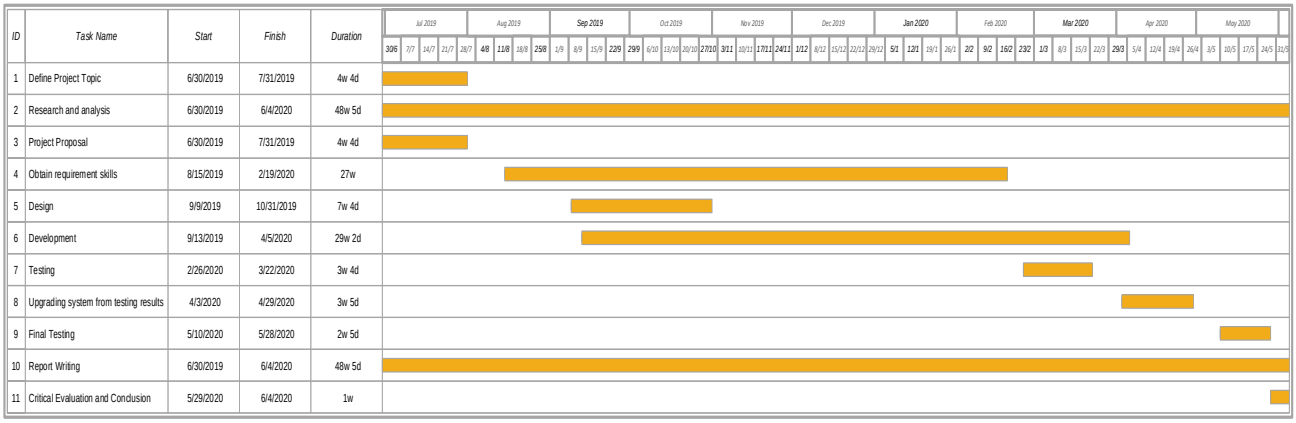
Availability

As we know mobile phones have become basic needs for most of the people, this app will be easy to get in the phones.

User Friendly

The application can be used easily beginner and advance users.

9.3. Gantt chart



ID	Task Name	Start	Finish	Duration
1	Define Project Topic	6/30/2019	7/31/2019	4w 4d
2	Research and analysis	6/30/2019	6/4/2020	48w 5d
3	Project Proposal	6/30/2019	7/31/2019	4w 4d
4	Obtain requirement skills	8/15/2019	2/19/2020	27w
5	Design	9/9/2019	10/31/2019	7w 4d
6	Development	9/13/2019	4/5/2020	29w 2d
7	Testing	2/26/2020	3/22/2020	3w 4d
8	Upgrading system from testing results	4/3/2020	4/29/2020	3w 5d
9	Final Testing	5/10/2020	5/28/2020	2w 5d
10	Report Writing	6/30/2019	6/4/2020	48w 5d
11	Critical Evaluation and Conclusion	5/29/2020	6/4/2020	1w

Figure 48: Gantt chart

9.4. Source Codes

FoodAdapter.java

```
package com.niruta.foodbite;

import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.RelativeLayout;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;
import com.bumptech.glide.request.RequestOptions;

import java.util.ArrayList;
import java.util.List;


public class FoodAdapter extends RecyclerView.Adapter<FoodAdapter.ViewHolder>
{

    List<Food> foodList;
    Context context;
    RequestOptions options;

    public FoodAdapter(List<Food> foodList, Context context) {
        this.foodList = foodList;
        this.context = context;

        options = new
RequestOptions().centerCrop().placeholder(R.drawable.loading_shape).error(R.dr
awable.loading_shape);
    }


    @NonNull
    @Override
    public FoodAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, final int viewType) {
        final Context context = parent.getContext();
        LayoutInflater inflater = LayoutInflater.from(context);

        View view = inflater.inflate(R.layout.food_list, parent, false);

        final ViewHolder viewHolder = new ViewHolder(view);
        viewHolder.relativeLayout.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                //Food food = foodList
                Intent intent = new Intent(context, FoodDetailActivity.class);
                //Bundle bundle = new Bundle();

                intent.putExtra("food_id",
foodList.get(viewHolder.getAdapterPosition()).getId());

intent.putExtra("food_detail_name",foodList.get(viewHolder.getAdapterPosition(
)).getTitle());

intent.putExtra("food_detail_category",foodList.get(viewHolder.getAdapterPosit
ion()).getType());

intent.putExtra("food_detail_price",foodList.get(viewHolder.getAdapterPosition
()).getPrice());

intent.putExtra("food_detail_image",foodList.get(viewHolder.getAdapterPosition
()).getImage());

                context.startActivity(intent);
            }
        });


        return viewHolder;
    }

    @Override
    public void onBindViewHolder(@NonNull FoodAdapter.ViewHolder holder, int
position) {
        Food food = foodList.get(position);

        holder.name.setText(food.getTitle());
        holder.type.setText(food.getType());
        holder.price.setText(food.getPrice());
        holder.rating.setText(food.getRating());
        holder.description.setText(food.getDescription());


Glide.with(context).load(food.getImage()).apply(options).into(holder.imageView
);



    }

    @Override
    public int getItemCount() {
        return foodList.size();
    }

    public class ViewHolder extends RecyclerView.ViewHolder {
        TextView name, type, price, rating, description;
        ImageView imageView;
        RelativeLayout relativeLayout;

        public ViewHolder(@NonNull View itemView) {
            super(itemView);

            name = itemView.findViewById(R.id.name);
```

```
            type = itemView.findViewById(R.id.type);
            price = itemView.findViewById(R.id.price);
            rating = itemView.findViewById(R.id.rating);
            description = itemView.findViewById(R.id.description);
            imageView = itemView.findViewById(R.id.food_image);
            relativeLayout = itemView.findViewById(R.id.relative);
        }
    }
    public void updateList(ArrayList<Food> itemList){
        foodList.clear();
        foodList = itemList;
        notifyDataSetChanged();
    }
}
```

PaymentActivity.java

```
package com.niruta.foodbite;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.text.Editable;
import android.text.InputType;
import android.text.TextWatcher;
import android.util.Log;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;

import com.android.volley.AuthFailureError;
import com.android.volley.NetworkError;
import com.android.volley.NoConnectionError;
import com.android.volley.ParseError;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.ServerError;
import com.android.volley.TimeoutError;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.khalti.checkout.helper.Config;
import com.khalti.checkout.helper.KhaltiCheckOut;
import com.khalti.checkout.helper.OnCheckOutListener;
```

```
import java.util.HashMap;
import java.util.Map;

public class PaymentActivity extends AppCompatActivity {
    SharedPreferences sharedPreferences;
    SharedPreferences.Editor editorPreferences;

    Bundle bundle;
    Order order;
    EditText amount, quantity;
    Button update, cancel, confirm;
    RadioGroup radioGroup;
    RadioButton radioButton;
    RadioButton radioButtonKhalti, radioButtonCash;
    private TextWatcher textWatcher = new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count,
int after) {

        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before, int
count) {
            try {
                String changedQuantity = quantity.getText().toString().trim();
                if (Integer.parseInt(changedQuantity) <= 0) {
                    quantity.setError("Quantity should be greater than
zero.");
                    confirm.setEnabled(false);
                    update.setEnabled(false);
                } else {
                    quantity.setError(null);
                    confirm.setEnabled(true);
                    update.setEnabled(true);

                    int changedQuantity2 =
Integer.parseInt(quantity.getText().toString().trim());
                    int singlePrice = Integer.parseInt(order.getTotalPrice())
/ order.getQuantity();
                    int finalPrice = singlePrice * changedQuantity2;
                    amount.setText(String.valueOf(finalPrice));
                }
            } catch (Exception e) {
                quantity.setError(null);
                amount.setText("");
                update.setEnabled(false);
                confirm.setEnabled(false);
            }
        }

        @Override
        public void afterTextChanged(Editable s) {

        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_payment);

        amount = findViewById(R.id.amount);
        quantity = findViewById(R.id.quantity);
        update = findViewById(R.id.update_order);
        cancel = findViewById(R.id.cancel_order);
        confirm = findViewById(R.id.confirm);
        radioGroup = findViewById(R.id.radioGroup);
        radioButtonKhalti = findViewById(R.id.onlinePayment);
        radioButtonCash = findViewById(R.id.cashOnDelivery);

        sharedPreferences = getSharedPreferences("LoginActivity",
MODE_PRIVATE);
        editorPreferences = sharedPreferences.edit();

        bundle = getIntent().getExtras();
        order = (Order) bundle.getSerializable("order_details");

        amount.setEnabled(false);
        amount.setInputType(InputType.TYPE_NULL);
        amount.setFocusableInTouchMode(false);

        quantity.setText(String.valueOf(order.getQuantity()));
        amount.setText(order.getTotalPrice());

        quantity.setInputType(InputType.TYPE_CLASS_NUMBER);

        quantity.addTextChangedListener(textWatcher);

        confirm.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                confirmPayment();
            }
        });

        update.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                updateOrder();
            }
        });

        cancel.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(PaymentActivity.this);
                builder.setTitle("Alert");
                builder.setMessage("Are you sure want to cancel this order?");
                builder.setCancelable(false);

                builder.setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int
i) {
                        deleteOrder();
```

```
                    }
                });
                builder.setNegativeButton("No", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int
i) {
                        dialogInterface.cancel();
                    }
                });

                AlertDialog alertDialog = builder.create();
                alertDialog.show();
            }
        });

    }

    public void updateOrder() {
        String url =
"https://foodbiteapp.000webhostapp.com/connect/update_order.php";
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE,
                WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
        final RequestQueue requestQueue =
Volley.newRequestQueue(PaymentActivity.this);
        StringRequest stringRequest = new StringRequest(Request.Method.POST,
url, new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                if (response.trim().equals("Success")) {
                    //progress_bar_verification.setVisibility(View.GONE);

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(PaymentActivity.this, "Successfully
Updated", Toast.LENGTH_SHORT).show();
                    Intent intent = new Intent(PaymentActivity.this,
MainActivity.class);
                    startActivity(intent);
                    finish();
                } else if (response.trim().equals("Error")) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(PaymentActivity.this, "Error while
updating. Try again.", Toast.LENGTH_SHORT).show();
                }
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                if (error instanceof NetworkError) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(getApplicationContext(), "Cannot connect to
Internet...Please check your connection!", Toast.LENGTH_SHORT).show();
                } else if (error instanceof ServerError) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(getApplicationContext(), "The server could
```

```
not be found. Please try again after some time!!", Toast.LENGTH_SHORT).show();
                } else if (error instanceof AuthFailureError) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(getApplicationContext(), "Cannot connect to
Internet...Please check your connection!", Toast.LENGTH_SHORT).show();
                } else if (error instanceof ParseError) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(getApplicationContext(), "Parsing error!
Please try again after some time!!", Toast.LENGTH_SHORT).show();
                } else if (error instanceof NoConnectionError) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(getApplicationContext(), "Cannot connect to
Internet...Please check your connection!", Toast.LENGTH_SHORT).show();
                } else if (error instanceof TimeoutError) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(getApplicationContext(), "Connection
TimeOut! Please check your internet connection.", Toast.LENGTH_SHORT).show();
                }
            }
        }) {
            @Override
            protected Map<String, String> getParams() throws AuthFailureError
{
                Map<String, String> params = new HashMap<>();
                params.put("user_id",
String.valueOf(sharedPreferences.getInt("id", 0)));
                params.put("product_id",
String.valueOf(order.getProductId()));
                params.put("quantity", quantity.getText().toString().trim());
                params.put("total_price", amount.getText().toString().trim());
                params.put("order_id", String.valueOf(order.getOrderId()));

                return params;
            }
        };
        requestQueue.add(stringRequest);
    }

    public void deleteOrder() {
        String url =
"https://foodbiteapp.000webhostapp.com/connect/delete_order.php";
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE,
                WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
        final RequestQueue requestQueue =
Volley.newRequestQueue(PaymentActivity.this);
        StringRequest stringRequest = new StringRequest(Request.Method.POST,
url, new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                if (response.trim().equals("Success")) {
                    //progress_bar_verification.setVisibility(View.GONE);

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(PaymentActivity.this, "Order Canceled",
Toast.LENGTH_SHORT).show();
```

```
                    Intent intent = new Intent(PaymentActivity.this,
MainActivity.class);
                    startActivity(intent);
                    finish();
                } else if (response.trim().equals("Error")) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(PaymentActivity.this, "Error while
deleting. Try again.", Toast.LENGTH_SHORT).show();
                }
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                if (error instanceof NetworkError) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(getApplicationContext(), "Cannot connect to
Internet...Please check your connection!", Toast.LENGTH_SHORT).show();
                } else if (error instanceof ServerError) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(getApplicationContext(), "The server could
not be found. Please try again after some time!!", Toast.LENGTH_SHORT).show();
                } else if (error instanceof AuthFailureError) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(getApplicationContext(), "Cannot connect to
Internet...Please check your connection!", Toast.LENGTH_SHORT).show();
                } else if (error instanceof ParseError) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(getApplicationContext(), "Parsing error!
Please try again after some time!!", Toast.LENGTH_SHORT).show();
                } else if (error instanceof NoConnectionError) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(getApplicationContext(), "Cannot connect to
Internet...Please check your connection!", Toast.LENGTH_SHORT).show();
                } else if (error instanceof TimeoutError) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(getApplicationContext(), "Connection
TimeOut! Please check your internet connection.", Toast.LENGTH_SHORT).show();
                }
            }
        }) {
            @Override
            protected Map<String, String> getParams() throws AuthFailureError
{
                Map<String, String> params = new HashMap<>();
                params.put("user_id",
String.valueOf(sharedPreferences.getInt("id", 0)));
                params.put("product_id",
String.valueOf(order.getProductId()));
                params.put("order_id", String.valueOf(order.getOrderId()));

                return params;
            }
```

```
        };
        requestQueue.add(stringRequest);
    }

    public void confirmPayment() {
        int radioId = radioGroup.getCheckedRadioButtonId();
        radioButton = findViewById(radioId);
        String paymentType = radioButton.getText().toString().trim();
        if (paymentType.equals("Pay through Khalti")) {
            String quantityKhalti, amountKhalti;
            quantityKhalti = quantity.getText().toString().trim();
            amountKhalti = amount.getText().toString().trim();
            editorPreferences.putInt("order_id", order.getOrderId());
            editorPreferences.putString("amount", amountKhalti);
            editorPreferences.putString("quantity",quantityKhalti);
            editorPreferences.putString("payment_type","ONLINE_PAYMENT");
            editorPreferences.apply();
            Intent intent = new Intent(PaymentActivity.this,
CashOnDeliveryActivity.class);
            startActivity(intent);
           // proceed();
        }
        else {
            String quantityKhalti, amountKhalti;
            quantityKhalti = quantity.getText().toString().trim();
            amountKhalti = amount.getText().toString().trim();
            editorPreferences.putInt("order_id", order.getOrderId());
            editorPreferences.putString("amount", amountKhalti);
            editorPreferences.putString("quantity",quantityKhalti);
            editorPreferences.putString("payment_type","CASH_ON_DELIVERY");
            editorPreferences.apply();
            Intent intent = new Intent(PaymentActivity.this,
CashOnDeliveryActivity.class);
            startActivity(intent);

        }
    }
    public void proceed() {
        AlertDialog.Builder builder = new
AlertDialog.Builder(PaymentActivity.this);
        builder.setTitle("Note");
        builder.setMessage("Are you sure you want to proceed?");
        builder.setCancelable(false);
        builder.setPositiveButton("Yes", new DialogInterface.OnClickListener()
{
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                openKhalti();
            }
        });
        builder.setNegativeButton("No", new DialogInterface.OnClickListener()
{
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                dialogInterface.cancel();
            }
        });
        AlertDialog alertDialog = builder.create();
        alertDialog.show();
```

```
    }

    public void openKhalti() {
        long amount2 = Long.parseLong(amount.getText().toString().trim());
        openKhaltiApp(amount2);

    }

    private void openKhaltiApp(long amount) {
        amount *= 100;
        Config.Builder builder = new
Config.Builder("test_public_key_db85ad7c754047db8199dd27d74d5ff6", "Product
Id", "Product Name", amount, new OnCheckOutListener() {

            @Override
            public void onSuccess(@NonNull Map<String, Object> data) {
                Log.i("success", data.toString());
                Toast.makeText(PaymentActivity.this, "Success",
Toast.LENGTH_SHORT).show();

            }

            @Override
            public void onError(@NonNull String action, @NonNull Map<String,
String> errorMap) {
                Log.e("hello", errorMap.toString());
                addPayment();

            }

        });

        Config config = builder.build();

        KhaltiCheckOut khaltiCheckOut = new
KhaltiCheckOut(PaymentActivity.this, config);
        khaltiCheckOut.show();
    }

    public void addPayment() {
        String quantityKhalti, amountKhalti;
        quantityKhalti = quantity.getText().toString().trim();
        amountKhalti = amount.getText().toString().trim();

        String url =
"https://foodbiteapp.000webhostapp.com/connect/payment_khalti.php";
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE,
                WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
        final RequestQueue requestQueue =
Volley.newRequestQueue(PaymentActivity.this);
        StringRequest stringRequest = new StringRequest(Request.Method.POST,
url, new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                if (response.trim().equals("Success")) {
                    //progress_bar_verification.setVisibility(View.GONE);

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(PaymentActivity.this, "Payment Successful",
```

```
Toast.LENGTH_SHORT).show();
                    Intent intent = new Intent(PaymentActivity.this,
MainActivity.class);
                    startActivity(intent);
                    finish();
                } else if (response.trim().equals("Error1")) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(PaymentActivity.this, "Insert error. Try
again.", Toast.LENGTH_SHORT).show();
                } else if (response.trim().equals("Error2")) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(PaymentActivity.this, "Update error. Try
again.", Toast.LENGTH_SHORT).show();
                }
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                if (error instanceof NetworkError) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(getApplicationContext(), "Cannot connect to
Internet...Please check your connection!", Toast.LENGTH_SHORT).show();
                } else if (error instanceof ServerError) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(getApplicationContext(), "The server could
not be found. Please try again after some time!!", Toast.LENGTH_SHORT).show();
                } else if (error instanceof AuthFailureError) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(getApplicationContext(), "Cannot connect to
Internet...Please check your connection!", Toast.LENGTH_SHORT).show();
                } else if (error instanceof ParseError) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(getApplicationContext(), "Parsing error!
Please try again after some time!!", Toast.LENGTH_SHORT).show();
                } else if (error instanceof NoConnectionError) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(getApplicationContext(), "Cannot connect to
Internet...Please check your connection!", Toast.LENGTH_SHORT).show();
                } else if (error instanceof TimeoutError) {

getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
                    Toast.makeText(getApplicationContext(), "Connection
TimeOut! Please check your internet connection.", Toast.LENGTH_SHORT).show();
                }
            }
        }) {
            @Override
            protected Map<String, String> getParams() throws AuthFailureError
{
                Map<String, String> params = new HashMap<>();
                params.put("user_id",
String.valueOf(sharedPreferences.getInt("id", 0)));
```

```
                params.put("amount", amountKhalti);
                params.put("order_id", String.valueOf(order.getOrderId()));
                params.put("quantity", quantityKhalti);

                return params;
            }
        };
        requestQueue.add(stringRequest);
    }
}
```

9.5. Google Form Survey

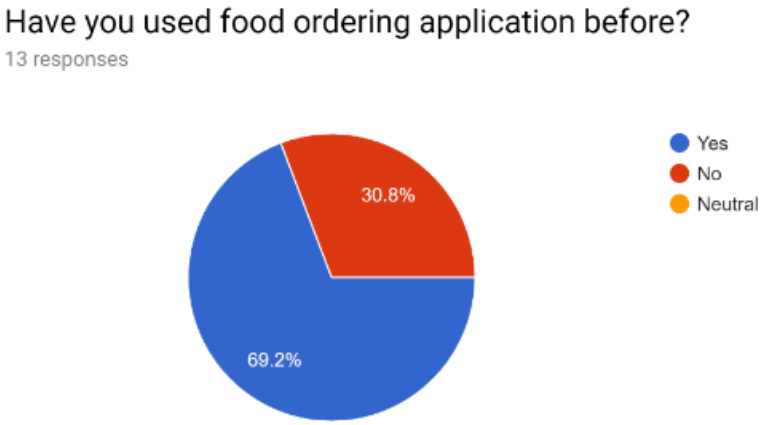


Figure 49: 1st survey question

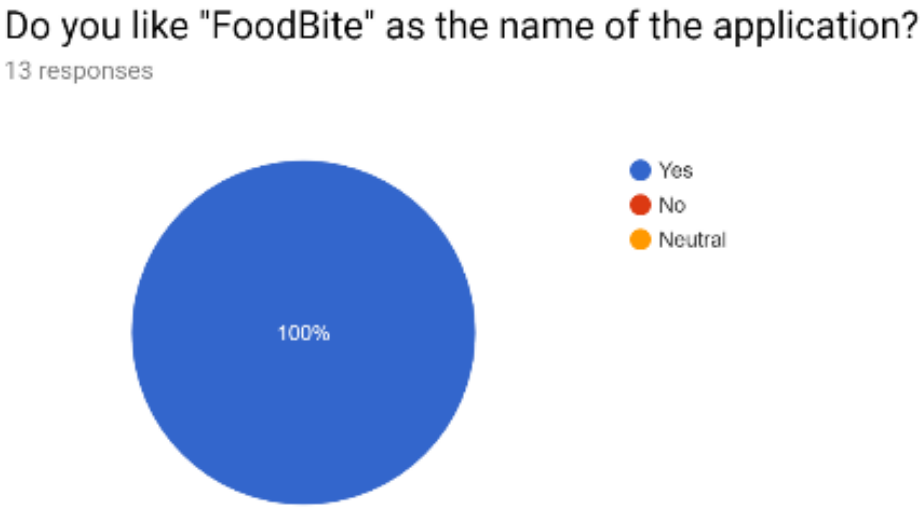


Figure 50: 2nd survey question

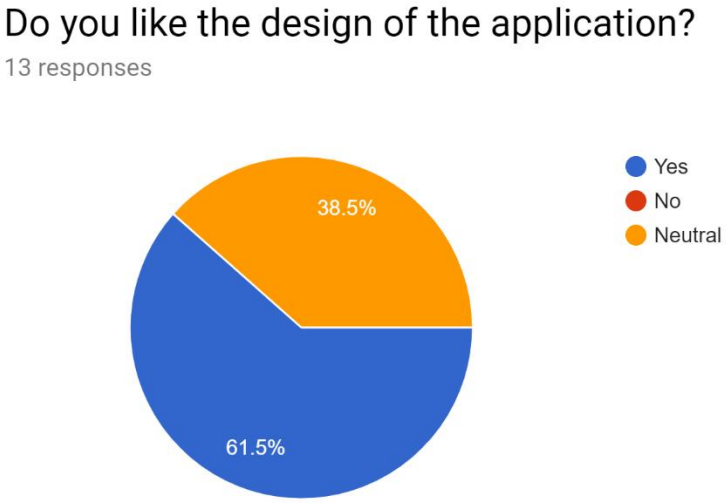


Figure 51: 3rd survey question

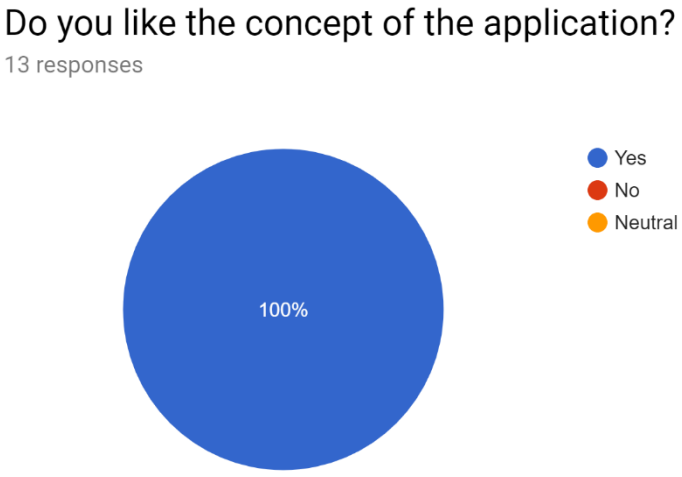


Figure 52: 4th survey question

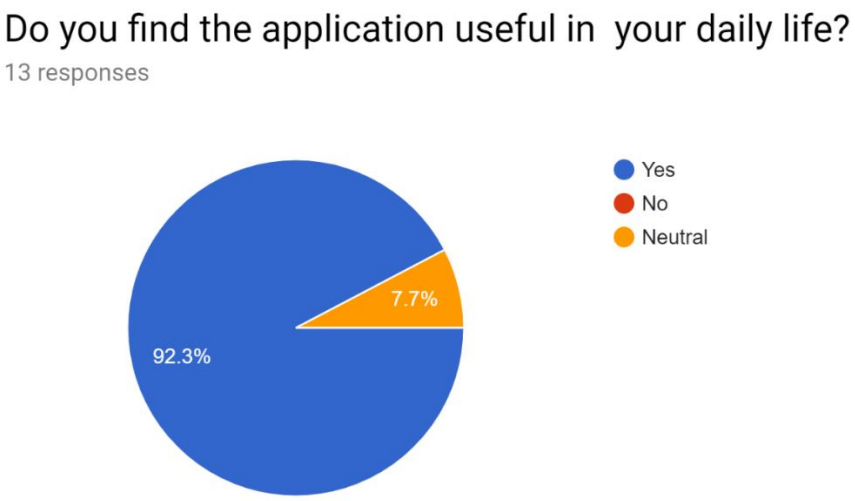


Figure 53: 5th survey question

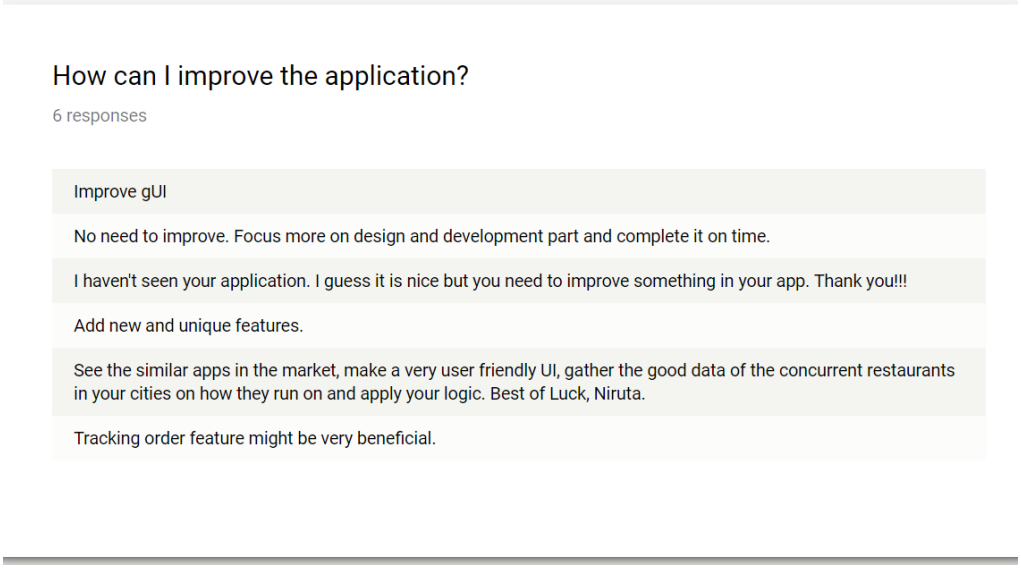


Figure 54: Suggestion survey