SOFTWARE ENGINEERING PROJECT

FINAL REPORT

CSE3001

FOODZAPP

Team Structure:

- Yashasvi Asthana (15BCE1161)
- Ritwik Kala (15BCE1114)
- Ritwik Gupta (15BCE1059)
- Suryansh Bhardwaj (15BCE1047)

Faculty: Prof. Anusha K.

Abstract:

Foodzapp is an online food ordering android based application. It is a third party application which can be integrated into any restaurant system. This will allow that restaurant or food chain to take orders through this app. There will be an interactive graphical user interface through which the user can select anything from the restaurant menu and can place order by giving his/her name, phone number and address. The user has an option to pay the bill online through this app or at the time of delivery.

Functional Requirements:

- Total bar (displays total amount)
- Type of food item
- Menu (List in that type)
- Amount selecting dropdown
- My plate button
- Add or delete option against food items in My plate
- Address field
- Phone number field
- Name field
- Make payment button
- Cash on delivery option
- Credit card details

Non-functional Requirements:

- Welcome screen
- Total amount
- List of food items
- Details of each food item
- My plate screen

FOODZAPP WBS

	A	B	C	D	E	F	G
1	Task name			Duration (Days)	Start	Finish	Resource Names
2	FOODZAPP			80	01-Aug-16	22-Oct-16	Ritwik Kala, Yashasvi Asthana, Suryansh Bhardwaj, Ritwik Gupta
3	1. Analysis			20	01-Aug-16	20-Aug-16	
4		1.1 User requirements report		20	01-Aug-16	20-Aug-16	
5			1.1.1 Interview Users	6	01-Aug-16	06-Aug-16	Ritwik Kala
6			1.1.2 Documents Requirements	14	07-Aug-16	20-Aug-16	Ritwik Gupta, Suryansh Bhardwaj
7			1.1.3 Milestone: user's sign off on requirements report				
8		1.2 Milestone: Anlaysis complete					
9							
10	2. Design			55	21-Aug-16	13-Oct-16	
11		2.1 XML Designing		15	21-Aug-16	04-Sep-16	
12			2.1.1 Menu module	7	21-Aug-16	27-Aug-16	Yashasvi Asthana
13			2.1.2 User credentials module	5	28-Aug-16	01-Sep-16	Suryansh Bhardwaj, Ritwik Gupta
14			2.1.3 Payment module	3	02-Sep-16	04-Sep-16	Ritwik Kala
15		2.2 JAVA coding		40	05-Sep-16	13-Oct-16	
16			2.2.1 Menu module	15	05-Sep-16	19-Sep-16	Yashasvi Asthana, Suryansh Bhardwaj
17			2.2.2 User credentials module	10	20-Sep-16	29-Sep-16	Yashasvi Asthana, Ritwik Kala
18			2.2.3 Payment module	10	30-Sep-16	08-Oct-16	Suryansh Bhardwaj, Ritwik Gupta
19			2.2.4 Relating modules	5	09-Oct-16	13-Oct-16	ALL
20							
21	3. Testing			5	14-Oct-16	18-Oct-16	
22		3.1 App Testing		5	14-Oct-16	18-Oct-16	
23			3.1.1 Alpha Testing	2	14-Oct-16	15-Oct-16	Yashasvi Asthana, Ritwik Kala
24			3.1.2 Beta Testing	3	16-Oct-16	18-Oct-16	Suryansh Bhardwaj, Ritwik Gupta
25							
26	4. Presentation			5	18-Oct-16	22-Oct-16	
27		4.1 Final presentation		2	18-Oct-16	19-Oct-16	ALL
28		4.2 Presentation in Exhibition		3	20-Oct-16	22-Oct-16	ALL
29		4.3 Milestone: Presentation complete					

Software Requirements Specification

for

Foodzapp

Prepared by:

1. Yashasvi Asthana 15BCE1161

2.Ritwik Gupta 15BCE1059

3. Suryansh Bhardwaj 15BCE1047

4. Ritwik Kala 15BCE1114

VIT University

7/9/2016

Table of Contents

1. Introduction

1.1 Purpose

Foodzapp is an Online Food Ordering Android based Application. It is a third party application which can be integrated into any restaurant system. This application therefore enables restaurants and food chains to take orders online in a convenient and organized manner. The document provides information about the application's requirements, both functional and non-functional as well as provides details regarding the User Interface(UI) and features of the application.

1.2 Document Conventions

No such conventions have been used to show special significance. However, the document provides:
(a) A description of the environment in which the application is expected to operate.
(b) A definition of the application's features.
(c) A specification of the application's functional and nonfunctional requirements.

1.3 Intended Audience and Reading Suggestions

This document is intended for users, developers, testers as well as the project coordinators. The SRS has been organized as seen in the Table of Contents. The SRS includes an introduction, an overview of the product as well as functional and non-functional requirements and UI. Users can refer to the overview get a basic idea of the product whereas developers can hover over to the requirements to get more in depth knowledge about the application system.

1.4 Product Scope

Foodzapp is an online food ordering android based application. It is a third party application which can be integrated into any restaurant system. This will allow that restaurant or food chain to take orders through this app. There will be an interactive graphical user interface through which the user can select anything from the restaurant menu and can place order by giving his/her name, phone number and address. The user has an option to pay the bill online through this app or at the time of delivery. The main selling point of the application is its exclusivity. An organization has to register with us before they can put their menu on the application which ensures quality. The application will provide the following capabilities:

- *The application can be accessed via Wifi or Data Plan in cellphones.*
- *Users will be able to manage their own account.*
- *Search function for food items based on the categories and item name.*

The project's clients have determined that this application will provide the following benefits:

- *Provide a more organized way to manage the food cart.*
- *Provide better reliability and security to the users on transactions.*
- *Provide flexibility and convenience to the users.*

The scope of this product is that it is available to anyone with v5.0.1 and above android device at any time in any place.

2. Overall Description

2.1 Product Perspective

This product is a derivative of the giants in this genre namely, Food Panda and/or Zomato. We differ from such applications primarily in the way we make our application accessible to restaurants and food chain. In our application we maintain exclusivity in terms of merchant access ensuring that only the most trusted and reliable brands can use the service.

2.2 Product Functions

- *Show the Menu*
- *Take the Order*
- *Online Payment*

2.3 User Classes and Characteristics

- *General User: This is the class of users who can simply see the menu and then make order of their will. This is the class which is most important to satisfy.*
- *Merchant: This class will be the class of merchants which can see what all are the orders and their payment methods too. This class is the least important to satisfy.*
- *Executive User: The class is especially for the Developers and Project Managers. They would definitely be provided with some more features and accessibility.*

2.4 Operating Environment

We are planning to release the application on android versions above v4.0 but we are also planning to release it on IOS as well as Windows depending on the kind of response we get after the application is released on android.

2.5 Design and Implementation Constraints

The application acts as a connecting point between the merchant and the user. If for some reason the merchant decides to cancel a particular user order, we are powerless to do anything about it. The application is only android based so far and thus IOS and Windows users do not have access to the service we provide.

2.6 User Documentation

We will be providing on-line help and tutorials on the application and also if any issue/error occurs then the customers can contact and solve their queries via email or contacting us on the given number.

2.7 Assumptions and Dependencies

The following is a list of assumptions and dependencies that would affect the software requirements if they turned out to be false.

- *Android version that we catered the application around is updated which can cause discrepancies within the application. To avoid it, regular updates have to be pushed out with each new android software update.*
- *Users don't have enough basic understanding to Android, IOS or windows phone and Internet.*

- *We assume that traffic would be well under the range of our server's capability and thus we assume that the system wouldn't crash.*

3. External Interface Requirements

3.1 User Interfaces

Usability

Interfaces are a critical class of components within the DML that will provide the means by which users interact with the system. All interfaces should provide easy access to help as well as clearly indicate the current state of the user's transaction when the user isn't idle.
The first screen for the user would be a login or sign up screen. User can create an account with our application and/or login to an already existing account. He is then redirected to choose the restaurant from which he wants to make an order which is followed by a screen showing the menu of that particular restaurant. User can choose the food items he wants to order and add them to the cart using the 'Add to Cart' Button. User can then review the cart, make changes and finally click 'Confirm' to move the payment portal where he can choose a payment method and be redirected to the bank's payment portal to complete the transaction after which he is redirected back to the application with an order confirmation form.

Administrative

Administrative interfaces will assist Administrators and Developers in maintaining database and controlling access to them. Because of the complexity of the data model, administrators will be able to edit multiple records simultaneously and create links between them.

3.2 Hardware Interfaces

The application is supported on android devices with v5.0.1 and above. The communication protocol followed here is basically within the application itself. The merchant can login to the application and if any orders have been placed for his restaurant, he will get alerts for the same there.

3.3 Software Interfaces

A database of orders placed would be maintained using SQL and/or other such DB query languages and as mentioned earlier we support android v 5.0.1 and above. Once the user places an order it gets stored in the database and a message is sent to the restaurant where the order is placed. Once the order is delivered the user entry is deleted from the database.

3.4 Communications Interfaces

Communication function includes the electronic forms which will be filled out by the customers when he is checking out his order to put in his credentials. Email and Web Browser support is also provided for contacting if any issues occur. HTTP Protocol will be used for network server communication protocol.

4. System Features

4.1 Add-To-Cart Feature

4.1.1 Description and Priority

Users can add food items to the cart. They can add multiple food items directly and can avail various offers and coupons while placing an order. It is of High priority as it provides users a systematic way to see the selected items and without the cart, maintaining the data would be difficult.

4.1.2 Stimulus/Response Sequences

The user picks the items that he wants to order from the restaurant menu and they are all added to cart as a response from the system. The user can check the items he picked so far by clicking on the add to cart icon on the screen which also shows the total amount for the entire order and a payment portal option.

4.1.3 Functional Requirements

REQ-1:<Remove Item> User can drop a food item from the cart. The total amount will automatically readjust.
REQ-2:<Add Item>User is redirected to the menu page where he can add more items to the cart.
REQ-3:<Payment>User is redirected to the payment portal.

4.2 Payment-Portal Feature

4.1.1 Description and Priority

Payment Portal redirects the user to secured page where he/she can choose the payment option of his choosing. User can use from a variety of options such as Debit Card, Credit Card, Paytm, Net Banking and Cash on Delivery.

4.1.2 Stimulus/Response Sequences

User chooses the payment option of his choice and then adds the credentials for the same. The page then redirects him to the bank's particular payment portal where he can complete the payment. Once the payment is complete he is redirected back to the application with an order confirmation.

4.1.3 Functional Requirements

REQ-1:<Pay> User can use this option to make payment for the order and he/she is redirected to the bank's secure payment portal.
REQ-2:<Go Back>User can use this option to go back to the 'add to cart' menu.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The application requires at least 512 MB of RAM and an android version 4.0 or above. The minimum space required is 30 MB only for the application.

5.2 Safety Requirements

The customers are ensured safety by backing up the data at the moment it is saved in the Database. Developers and Project Managers must take precautions before editing/deleting any saved data.

5.3 Security Requirements

A third party organization would certify and provide a more secure environment to the customers/client. HTTPS Network Server Connection will be used for security/privacy issues. The transactions made, would have 128-bit encryption level.

5.4 Software Quality Attributes

The customers would be provided with a flexible, portable, reliable and a robust application. It will also be user-friendly and an easy to use application.

5.5 Business Rules

Developer: *Developer has overall control over the application. He can modify the roles of other individuals as well.*

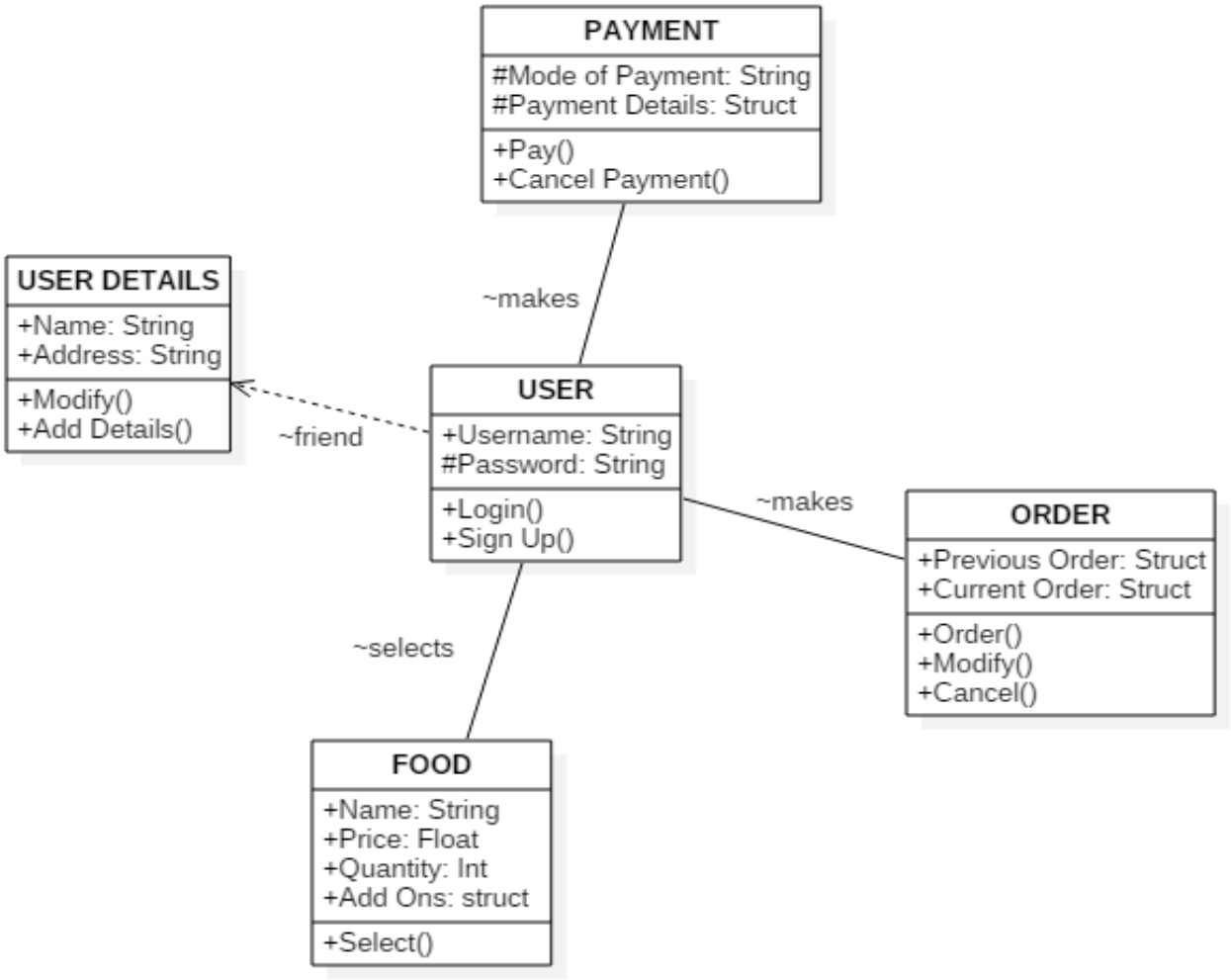
User*: User is a client and can only access the parts which the developer gives him access to.*

Merchant*: Merchant is the seller and can only access the parts which the developer gives him access to.*

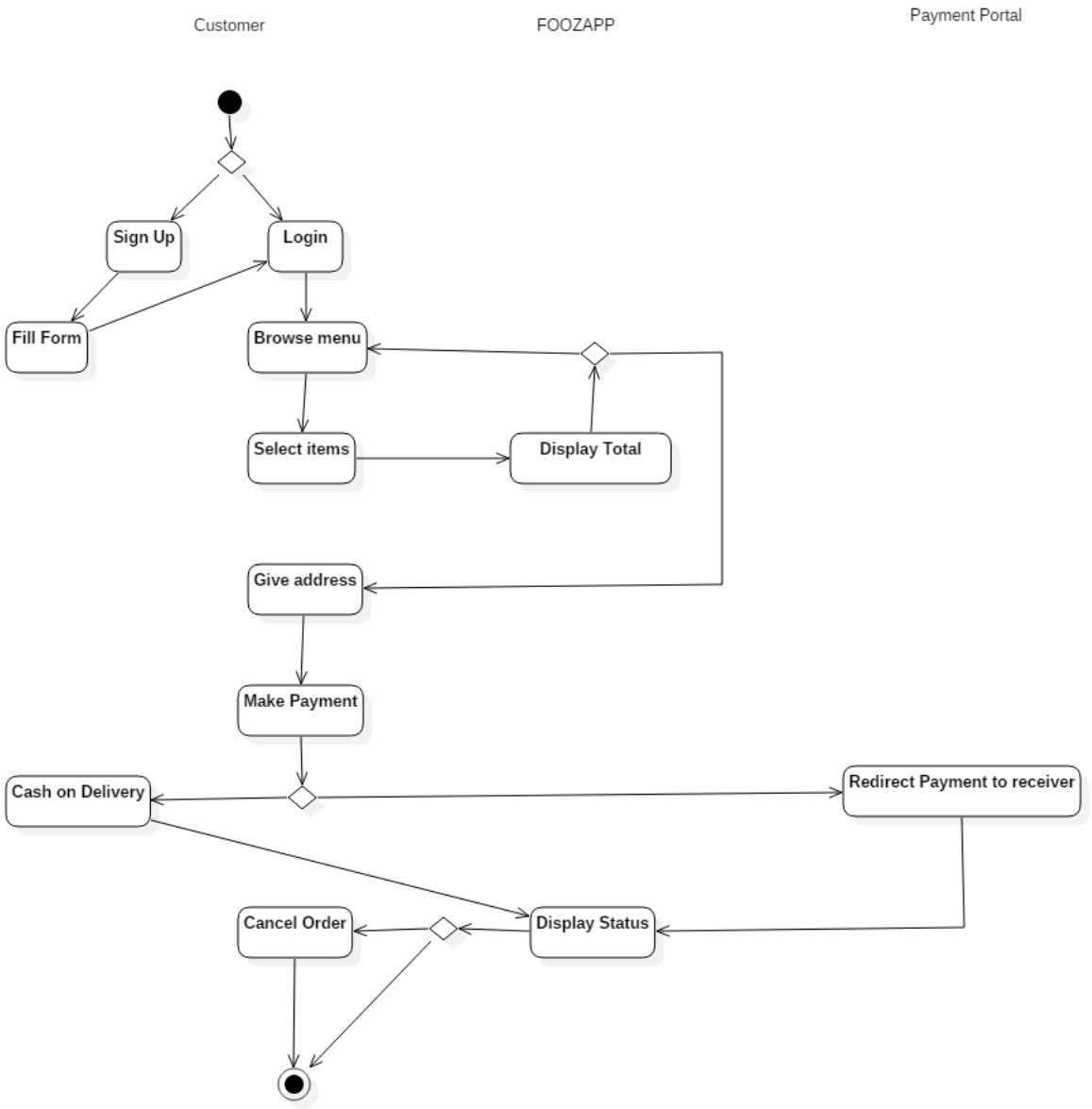
Appendix A: Glossary

No such necessary terms required to interpret the SRS.

Appendix B: Analysis Models:

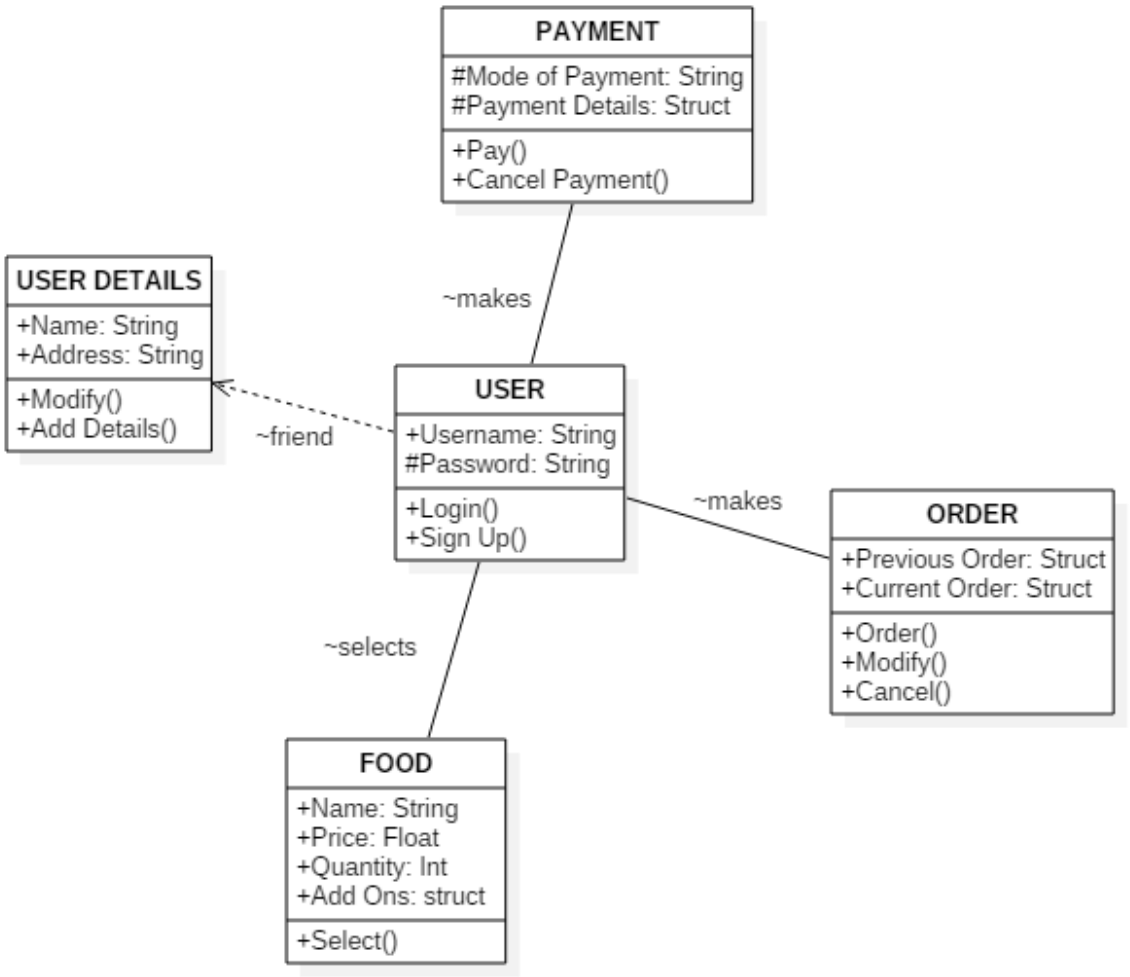


UML DIAGRAMS

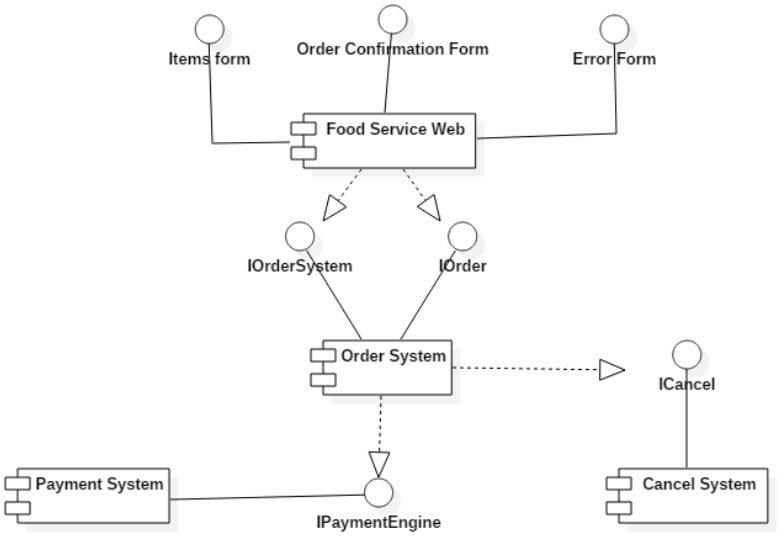


ACTIVITY DIAGRAM

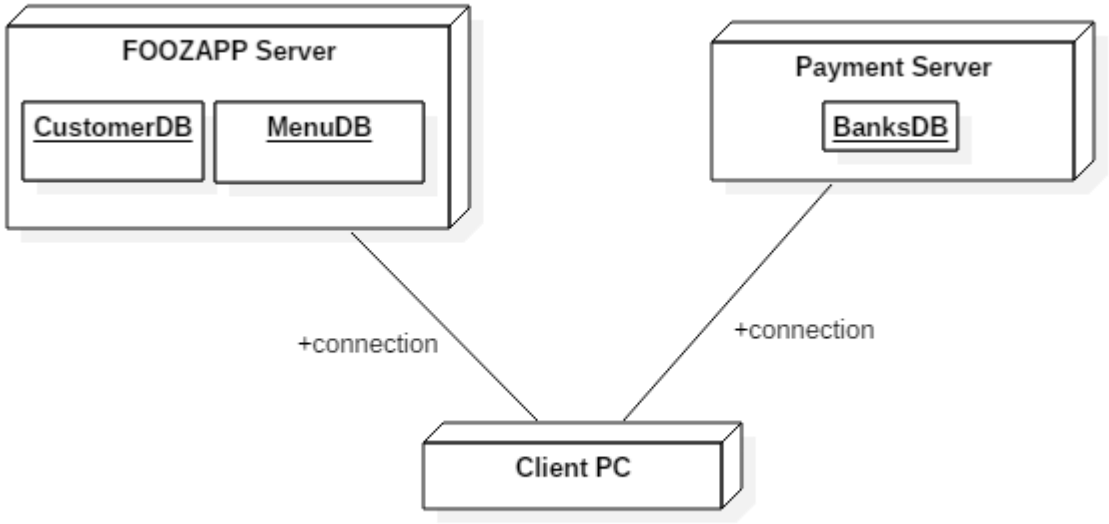
CLASS DIAGRAM



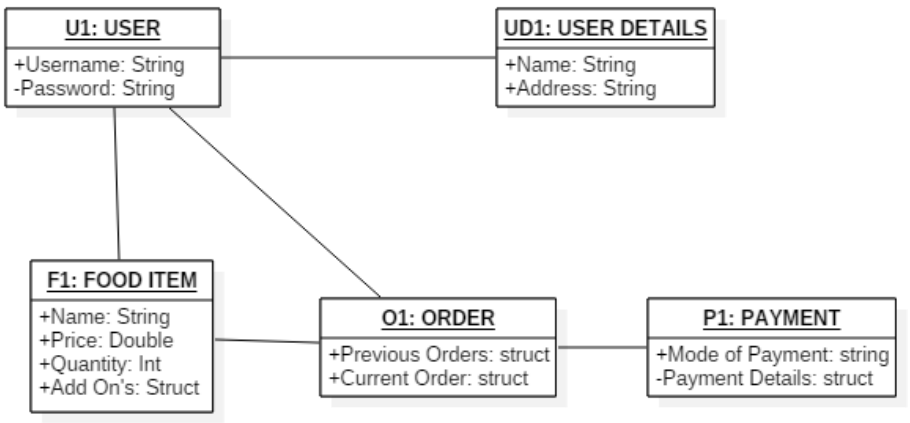
COMPONENT DIAGRAM



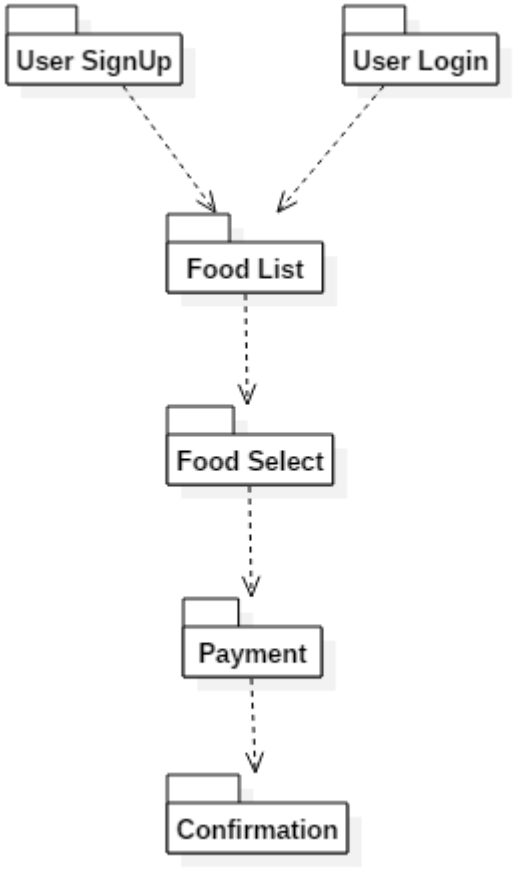
DEPLOYMENT DIAGRAM



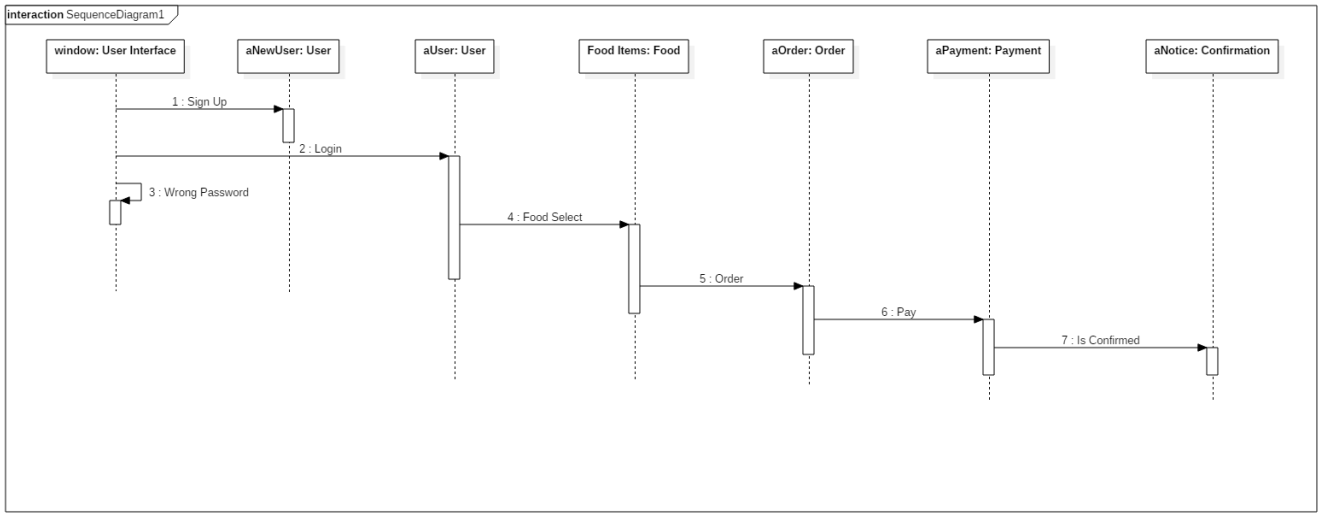
OBJECT DIAGRAM



PACKAGE DIAGRAM

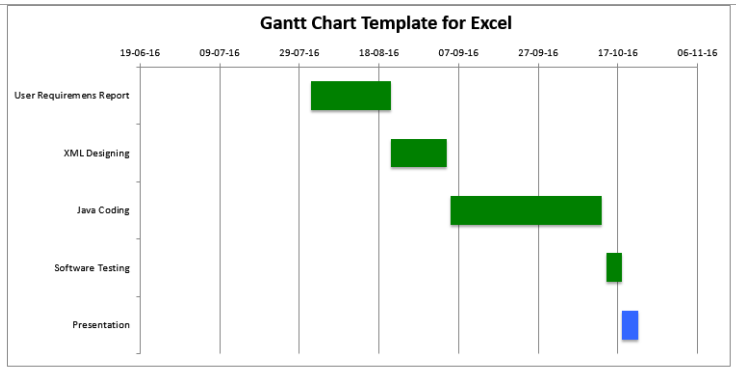


SEQUENCE DIAGRAM



GANTT CHART

Task Name	Start	End	Duration (days)
User Requiremens	01-08-16	20-08-16	20
XML Designing	21-08-16	04-09-16	14
Java Coding	05-09-16	13-10-16	38
Software Testing	14-10-16	18-10-16	4
Presentation	18-10-16	22-10-16	4



A Software Design Specification Template

For FOODZAPP

BY - Ritwik Gupta(15BCE1059)

Suryansh Bhardwaj(15BCE1047)

Yashasvi Asthana(15BCE1161)

Ritwik Kala(15BCE1114)

Table of Contents

6. Introduction

The following is a basic template for the specification of software designs. Wherever possible, guidelines (instead of prescribing requirements) for the contents of various sections and subsections of the document have been provided. To obtain more detailed subsections of a particular section, choose one or more of the subsection topics from the list of guidelines provided.

In devising this template, information from many sources have been gleamed, including various texts on Software Engineering (Pressman and Somerville), Android development, various SEI reports and documentation requirements.

This software design specification is designed to meet the following criteria:

- It adequately serves as training material for new project members, imparting to them enough information and understanding about the project implementation, so that they are able to understand what is being said in design meetings.

- It serves as "objective evidence" that the designers and/or implementers are following through on their commitment to implement the functionality described in the requirements specification.

- It needs to be as detailed as possible, while at the same time not imposing too much of a burden on the designers and/or implementers that it becomes overly difficult to create or maintain.

NOTE: The sections in this document are concerned solely with the design of the software.

6.1 Document Outline

- Introduction
- System Overview
- Design Considerations
    - Assumptions and Dependencies
    - General Constraints
    - Goals and Guidelines
- Detailed System Design
    - module-1 Login module
    - module-2 Food-Menu module
    - module-3 Add-to-Cart module
    - module-4 Payment module
- Glossary
- Bibliography

6.2 Document Description

Here is the description of the contents (by section and subsection) of the proposed template for software design specifications:

6.2.1 Introduction

This document is a reference for developers and testers to give them an idea about the development and design procedures.

The purpose of this document is to provide a detailed description about the application FOODZAPP. This document is intended for different types of readers such as developers, project managers, and testers as well as the project coordinators providing architectural and system designing.

The document will provide the document outline regarding the FOODZAPP 1.0. The SRS of FOODZAPP is also referenced to this document. There are no such pre-requisite for this document.

6.2.2 System Overview

There will be an interactive graphical user interface through which the user can choose the restaurant and then select the food items. The user then uses the ADD-TO-CART feature to add the selected food items in the cart. The order placed is shown only once in the process. Then the user is redirected to the Payment Portal.

7. Design Considerations

This section describes many of the issues which need to be addressed or resolved before attempting to devise a complete design solution.

7.1 Assumptions and Dependencies

The following assumptions and Dependencies are taken regarding the application:

- Android Studio is recommended for the development procedure.
- Android Version 4.0 or above is required on the user's end.
- The design of login page and other pages may differ in design.
- The application would be able to handle the traffic.
- Space required to store the data in database is enough.
- The user should have an internet access via Wi-Fi or Data Plan.

7.2 General Constraints

- Minimum 30MB space on drive
- 512MB ram
- Android version 4.0 or later
- Internet Connection via Wi-Fi or Data Plan
- Updated version of the app
- The application needs to be certified by a third party security validation organization.
- App is not supported on Windows or IOS devices.

7.3 Goals and Guidelines

- The KISS principle ("Keep it simple stupid!") is followed throughout the design.
  This is required so that the development is more efficient and streamlined.

- Balance between performance and quality is maintained to make sure that the processing is easy and the interface is visually aesthetic.

7.4 Development Methods

The software interface of the application and all its features have been designed in JAVA and the backend database is being designed using MySQL query language. XML is used for developing the User Interface (UI) of the application.

8. Architectural Strategies

The model followed in the development of the software is Incremental. The incremental model is a method of software development where the product is designed, implemented and tested incrementally (a little more is added each time) until the product is finished. This model was chosen so that all the packages can be built separately by the team members and then combined as single software.

Java and XML will be used to develop the android program. The tool used is android studio. SQL will be used to create and maintain the database which will be accessible by the application.

Existing modules in android studio are used. Some of them are – OS, view, widget and many more.

This is just a prototype application. In future the application can be modified according to the customer restaurant who buys the application. The database will then be taken from the restaurant server (if existent). The customer restaurant can also demand some additional modules which will be designed then accordingly.

This application will require internet connection to work. The changes in database will be in real time. Multiple servers will be established to remove any lag in the client server communication.

The user should have at least 512mb ram and android 4.0 or later to run the application.

9. System Architecture

There are 3 modules – Login/Signup Module, Food Select Module and My Cart Module.

The signup option opens a link in the browser and where all the necessary fields should be filled to signup.

The functions/methods available to the user include Add Details, Modify User Details, Login, Signup, Select Food, Order, Modify Order, Cancel Order, Pay, Cancel Payment, etc.

10. Policies and Tactics

The project revolves around the use of Android Studio. Compiling of the Java Code was done using Eclipse and Android Studio. The database was designed using MySql and Sql Plus.

No specific coding conventions were followed however; the lines of code were minimalized to the maximum extent. The use of Brute Force Algorithms was also minimalized wherever possible to decrease the lines of code.

A prototype of the android application will be created. This would be made available to our clients and end users for their inputs on user interface, time managements and usability. The users get a peek of our app after completion of every module and their feedbacks are well received.

A beta version of the app would be made available to our clients. The app would be launched and tested by the end users. However, before that a detailed testing of the same would be carried out in our environment.

Once a restaurant wishes for our app to be customized for them, the orders would be directed directly to them through the app. However, weekly maintenance of the application would be carried out by us. New features as well as improvements will also be provided.

11. Detailed System Design

11.1 Classification

The whole project was divided into the following modules – Login/Signup Module, Food Select Module and My Cart Module.

The Modules were broken down into sub modules which involves –

User requirements report and its analysis, XML Designing, Java Coding, Alpha and Beta Testing as well as Final Presentation.

The functions/methods available to the user include Add Details, Modify User Details, Login, Signup, Select Food, Order, Modify Order, Cancel Order, Pay, Cancel Payment, etc.

11.2 Definition and Responsibilities

The Login Module is the most basic module. Through it, the user can make an account which will be registered in our database. The user's choice and history of orders will be stored through the account he logs in through.

Food Select module would contain a huge list of food items of that particular restaurant for which the app is modified and designed. It would enable the user to choose the food items he desires from the list. The user can select as many items de deems necessary and add it to "My Cart".

All the food items that the user wishes to order, does so by adding it to My Cart. Hence, My Cart would include the users order. The user can modify the order by changing the quantity of items, adding another food item or dropping a selected item from his order list. This module would also include the history of previous orders made by the user for his reference.

After the user finalizes his choice and places an order, he is asked for Payment options. The user can pay through inline banking or has the option for paying through Cash on Delivery Method. After the payment is completed the user gets an order confirmation mail or text message as the restaurant's acknowledgement.

11.3 Constraints

Through the Login Module, the user logs into his account. However, the user can login to only one account at a time, i.e., the orders can be placed through a single account per phone. The password length should be minimum 8 characters.

The My Cart Module can display only the history of 3 previous orders placed using the application, i.e., the previous orders placed would fade away. While placing an order there might be items that might not get selected, that would happen only if the restaurant considers that item as unavailable or out of season. During season or it's availability, the item can then be selected.

The user can select a food item as well as its amount in the Food Select Module, however the quantity has a limit. The limit is decided by the restaurant and a quantity limit is added to that particular item.

Also, humongous party orders cannot be placed using the application, the reason being a lot of time to prepare the order as well as problems in delivering it. A huge order will be decided by the total quantity of food items selected or through a price limit.

Some other constraints for the android application is -

- Minimum 30MB space on drive
- 512MB ram
- Android version 4.0 or later

- Internet Communication via Wi-Fi or Data Plan
- Updated version of the app
- The application needs to be certified by a third party security validation organization.
- App is not supported on Windows or IOS devices.

11.4 Composition

The Modules were broken down into sub modules which involves –

User requirements report and its analysis:

Basic user requirements were gathered through thorough research on previous such mobile applications and considering factors like user interface, ease and flexibility of the application. Each of these requirements were deeply analyzed and a report entailing every detail was prepared.

XML Designing:

The layout of the application as well as its contents were designed using XML in Android Studio. The design and interface was given flexibility and updation and other modifications were done easily using XML.

Java Coding:

On obtaining a well-developed design and structure of the application numerous links and responses were done using JAVA. The XML Designing and Java Coding were done parallel to each other.

Testing:

On completion of each module, testing of the application took place. The feedbacks received were considered and modifications were done. This completes that particular module. In such a way one module after another was tested and completed. On the completion of the application as a whole, alpha and beta testing was done. The changes and modifications, if any, were done to the whole project while fixing all the bugs and issues. The mobile application was then deployed.

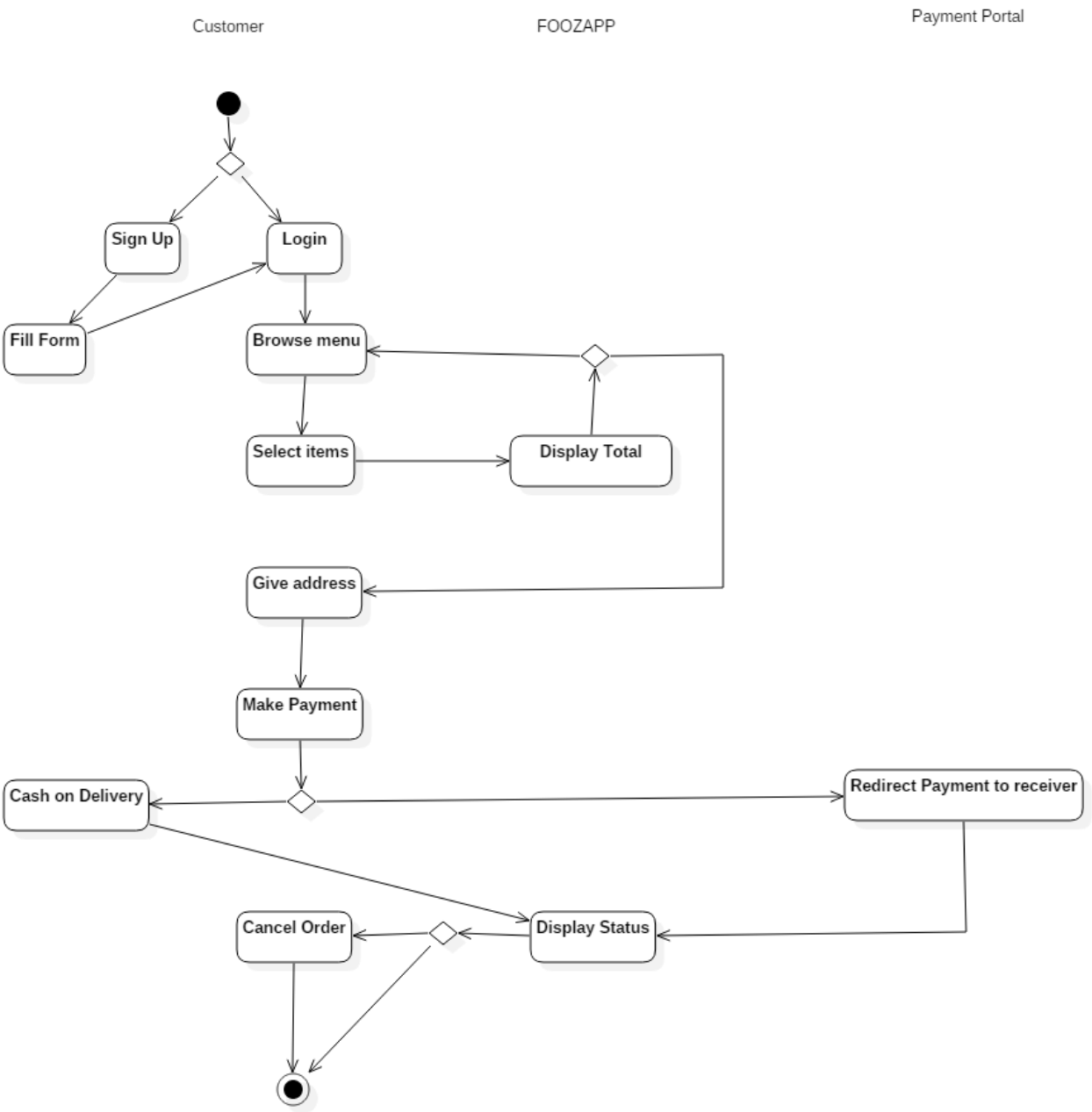
11.5 Uses/Interactions

The user logs in through the Log in Module and the id and password is checked with the database. The user is notified in case of a wrong password or id.

A huge database consisting of food items of a particular restaurant's menu is created and the user selects food from it. When the user orders the food he wants, the order is placed through the account he logs into in the Login Module. The history of such orders is present in the My Cart menu.

After Payment for the order, the user receives a notification as a sign of acknowledgement on the account he logs into.

11.6 Detailed Subsystem Design



12. Glossary

No such terms or abbreviations have been made.

13. Bibliography

No such references are made.

FOODZAPP

Test Plan

Version 1.0

Dated: 24th October, 2016

Authors

Ritwik Gupta 15BCE1059

Suryansh Bhardwaj 15BCE1047

Ritwik Kala 15BCE1114

Yashasvi Asthana 15BCE1161

Revision History

Date	Version	Description	Author
24th October, 2016	1.0	First version	Test Team

14. BACKGROUND

Foodzapp is an online food ordering android based application. It is a third party application which can be integrated into any restaurant system. This will allow that restaurant or food chain to take orders through this app. There will be an interactive graphical user interface through which the user can select anything from the restaurant menu and can place order by giving his/her name, phone number and address. The user has an option to pay the bill online through this app or at the time of delivery.

15. INTRODUCTION

This test plan is a basic guideline for future testing in the FOODZAPP Application. It mainly focuses on two problems: what we will test and how we will test.

16. TEST ITEMS

16.1 GUI test

16.2 Basic function test

16.3 Database test

16.4 Network test

17. FEATURES TO BE TESTED

17.1 GUI test

System should provide a GUI for restaurant employees to interface with the backend FOODZAPP database

17.2 Database test

17.2.1 Basic operations: add/update/delete/query records in each table

17.2.2 Advanced operations: start/stop, backup, recover the database

17.3 Basic function test

17.3.1 Add to cart

17.3.2 Update/delete cart items

17.3.3 Search for a particular food item

17.3.4 View item price

17.3.5 View item detail

17.3.6 Payment portal

17.4 Network test

Check connectivity of the system in the LAN environment

18. FEATURES NOT TO BE TESTED

18.1 Maximum simultaneous online-users is no more than 50

Reason: This is only an assumption condition in requirement specification

18.2 Licensing requirement

Reason: Since they are free software, they need not be tested.

19. APPROACH

19.1.1 Unit testing (class testing)

19.1.2 Integrity testing (thread-based testing)

19.1.3 Validation testing

20. ITEM PASS CRITERIA

20.1 GUI test

Pass criteria: Restaurant employees could use this GUI to interface with the backend FOODZAPP database without any difficulties

20.2 Database test

Pass criteria: Results of all basic and advanced operations are normal (refer to section 4)

20.3 Basic function test

Add to cart

Pass criteria:

- User can choose and add a particular item to his cart using this function.
- The retrieved item information should appear in the cart form.

20.3.1 Update/delete cart items

Pass criteria:

20.3.2 The record would be selected using the item ID

20.3.3 Updates can be made on the cart form or the menu form.

20.3.4 The item can be deleted from the cart in the cart form or the menu form.

- If an item is deleted, it will not appear in further totaling and cart search queries.

20.3.5 Search for a particular food item

Pass criteria:

- Each food item shall have following attributes: Name, Item ID.
- The retrieved food item information should contain these attributes.

20.3.6 View item price

Pass criteria:

- The item item can be retrieved using the item ID
- The food item can be deleted only if the user has not already paid for it.
- If items were deleted, they would not appear in further cart search queries.
- Item price should be equal to the value associated with the Item ID times the no of those items.

20.3.7 View item detail

Pass criteria:

- The product shall let the restaurant employee/customer query food item' detail information by their item ID.
- The search results would produce a list of items, which match the search parameters with following Details: Item ID, Name.
- The display would also provide the stock which is available for further orders.
- The search display will be restricted to 20 results per page and there would be means to navigate from sets of search results.
- The user can perform multiple searches before finally selecting a set of items for checkout. These should be stored across searches.

20.3.8 Payment portal

Pass criteria:

- Customers can check out the orders using a specific order ID.
- The checkout is initiated using the make payment button on the cart form.
- The User ID who is making the transaction would be entered
- The checkout date and time would automatically reflect the current system date.
- An order once made cannot be cancelled.
- The maximum order a user can make is up to Rs.8000. The system should not allow checkout of orders beyond this limit.

20.4 Network test

Pass criteria: Results of operations (ping, ftp and ODBC connectivity check) are normal

21. TEST DELIVERABLES

21.1 Test plan

21.2 Test report

22. ENVIRONMENTAL NEEDS

- Hardware: Minimum 250 MB RAM and 25 MB Disk Space required.
- Software: Android Version 4.0 or above.

23. RESPONSIBILITIES

- Mr. Ritwik Gupta and Mr. Suryansh Bhardwaj are involved in test plan document
- All members of testing group are involved in testing process
- Mr. Ritwik Kala and Mr. Yashasvi Asthana are involved in test report document

24. STAFFING

Mr. Yashasvi Asthana, leader of testing group

Mr. Ritwik Gupta, member of testing group

Mr. Ritwik Kala, member of testing group

Mr. Suryansh Bhardwaj, member of testing group

25. SCHEDULE

- Test plan is due on October 25
- Test report is due on October 25

26. RESOURCES

- Developers of the system are involved in testing process (debugging, unit testing, even integrity testing)
- Users of the system are involved in testing process (integrity testing)

27. RISKS CONTINGENCIES

Should prototype of the system not be due before October 12, test report could not be guaranteed to be available on October 25

28. APPROVALS

No such approvals.

TEST-CASE

S.no	Req.No	Test Case no	Description of test case	Expected Result	Actual Result	Priority	Severity	Status	Comments
1	1	1	Verify if the application has 5 forms	Application has 5 forms	Application has 5 forms	High	High	Completed	
1.LOGIN PAGE									
2	2.1	2.1.1	verify form1 has 5 fields	Form 1 has 5 fields	Form 1 has 5 fields	High	High	Completed	
	2.2	2.2.1	Verify 1st field is Username	Form has Username field	Form has Username field	High	High	Completed	
	2.3	2.3.1	Verify 2nd field is Password	Form has Password field	Form has Password field	High	High	Completed	
	2.4	2.4.1	Verify 3rd field is Forgot Password?	Form has forgot password field	Form has forgot password field	High	High	Completed	
	2.5	2.5.1	Verify 4th field is Login Button	Form has login button	Form has login button	High	High	Completed	
	2.6	2.6.1	Verify 5th field is Sign Up button	Form has Sign up button	Form has Sign up button	High	High	Completed	
3	3.1	3.1.1	Verify username has 10 characters	Username has 10 characters	Username has 10 characters	Low	Medium	Completed	
		3.1.2	Verify username 98732065111 is incorrect	Username is incorrect:Must have only 10 digits	Username is incorrect:Must have only 10 digits	Low	Medium	Completed	
		3.1.3	Verify username 987320651 is incorrect	Username is incorrect,must have 10 digits	Username is incorrect,must have 10 digits	Low	Medium	Completed	
		3.1.4	[illegible] correct	Username is correct	Username is correct	Low	Medium	Completed	
	3.2	3.2.1	Verify username doesn't have alphabets	Username doesn't have alphabets	Username doesn't have alphabets	Low	Medium	Completed	
	3.3	3.3.1	Verify Special characters are not allowed in username	Special characters are not allowed	Special characters are not allowed	Low	Medium	Completed	
4	4.1	4.1.1	Verify password has any characters	Password can have any character	Password can have any character	Low	Medium	Completed	
			Verify Password "Suryan12" is correct	Password is correct	Password is correct	Low	Medium	Completed	
			Verify Password "Sur@123"' is	Password is correct	Password is correct	Low	Medium	Completed	
	4.2	4.2.1	Verify Password must have minimum 6 length	Password has minimum 6 length	Password has minimum 6 length	Low	Medium	Completed	
		4.2.2	Verify Password "sur12" is incorrect	Password sur12 is incorrect	Password sur12 is incorrect	Low	Medium	Completed	
		4.2.3	Verify Password "surya12" is correct	Password surya12 is correct	Password surya12 is correct	Low	Medium	Completed	
		4.2.4	Verify Password "Suryansh" is	Password Suryansh is	Password Suryansh is	Low	Medium	Completed	
5	5.1	5.1.1	Verify Forgot Password button works	Forgot Password button works properly	Forgot Password button works properly	High	High	Completed	
	5.2	5.2.1	FORGOT PASSWORD Form get's opened	Forgot password Form is opened	Forgot password Form is opened	High	High	Completed	
6	6.1	6.1.1	Verify Login button works	Login button works properly	Login button works properly	High	High	Completed	
		6.1.2	Next form is opened	Next Form2 gets opened	Next Form2 gets opened	High	High	Completed	
7	7.1	7.1.1	Verify Sign Up button works	Sign up button works properly	Sign up button works properly	High	High	Completed	
		7.1.2	SIGN UP Form get's opened	SIGN UP Form is opened	SIGN UP Form is opened	High	High	Completed	
2.SIGN UP PAGE									
8	8.1	8.1.1	Verify FORM has 5 fields	Form has 5 fields	Form has 5 fields	High	High	Completed	
	8.2	8.2.1	Verify 1st field is Name	Form has a Name field	Form has a Name field	High	High	Completed	
	8.3	8.3.1	Verify 2nd field is Phone number	Form has a Phone number field	Form has a Phone number field	High	High	Completed	
	8.4	8.4.1	Verify 3rd field is Password	Form has a Password field	Form has a Password field	High	High	Completed	
	8.5	8.5.1	Verify 4th field is Confirm Password	Form has a Confirm Password field	Form has a Confirm Password field	High	High	Completed	
	8.6	8.6.1	Verify 5th field is Submit Button	Form has a submit button	Form has a submit button	High	High	Completed	
9	9.1	9.1.1	Verify Name has atmost 30 characters	Name has atmost 30 characters	Name has atmost 30 characters	Low	Medium	Completed	
			Verify Name "Moja" is allowed	Name Moja is allowed	Name Moja is allowed	Low	Medium	Completed	

9	9.1	9.1.1	Verify Name has atmost 30 characters	Name has atmost 30 characters	Name has atmost 30 characters	Low	Medium	Completed
			Verify Name "Moja" is allowed	Name Moja is allowed	Name Moja is allowed	Low	Medium	Completed
	9.2	9.2.1	Verify Name doesnot have any numbers	Name doesn't have any numbers	Name doesn't have any numbers	Low	Medium	Completed
			Verify Name YASH123 is incorrect	Name YASH123 is incorrect and is not accepted	Name YASH123 is incorrect and is not accepted	Low	Medium	Completed
			Verify Name Suryansh is correct	Name Suryansh is accepted	Name Suryansh is accepted	Low	Medium	Completed
10	10.1	10.1.1	Verify Phone number has 10 digits	Phone number has 10 digits	Phone number has 10 digits	Low	Medium	Completed
			Verify username 98732065111 is incorrect	Username is incorrect:Must have only 10 digits	Username is incorrect:Must have only 10 digits	Low	Medium	Completed
			Verify username 987320651 is incorrect	Username is incorrect,must have 10 digits	Username is incorrect,must have 10 digits	Low	Medium	Completed
			Verify username 9873206511 is	Username is correct	Username is correct	Low	Medium	Completed
	10.2	10.2.1	Verify Phone number has only digits	Phone number has only digits	Phone number has only digits	Low	Medium	Completed
			Verify Phone number 98997080A2 is incorrect	Phone number is not allowed	Phone number is not allowed	Low	Medium	Completed
			Verify number 9899708081 is correct	Number is correct and accepted	Number is correct and accepted	Low	Medium	Completed
11	11.1	11.1.1	Verify password has any characters	Password can have any character	Password can have any character	Low	Medium	Completed
			Verify Password "Suryan12" is correct	Password is correct	Password is correct	Low	Medium	Completed
			Verify Password "Sur@123"' is	Password is correct	Password is correct	Low	Medium	Completed
		11.2.1	Verify Password must have minimum 6 length	Password has minimum 6 length	Password has minimum 6 length	Low	Medium	Completed
			Verify Password "sur12" is incorrect	Password sur12 is incorrect	Password sur12 is incorrect	Low	Medium	Completed
			Verify Password "surya12" is correct	Password surya12 is correct	Password surya12 is correct	Low	Medium	Completed
			Verify Password "Suryansh" is	Password Suryansh is	Password Suryansh is	Low	Medium	Completed

12	12.1	12.1.1	should be same as the above password	Passwords match and Password gets confirmed	Passwords match and Password gets confirmed	High	High	Completed
13	13.1	13.1.1	Submit button works	Submit button works properly	Submit button works properly	High	High	Completed
		13.1.2	Next Form gets opened	MENU Form get's opened	MENU Form get's opened	High	High	Completed
3.FORGOT PASSWO								
14	14.1	14.1.1	Verify FORM has 4 fields	Form has 4 fields	Form has 4 fields	High	High	Completed
	14.2	14.2.1	Verify 1st field is Phone number	field	Form has a Phone number	High	High	Completed
	14.3	14.3.1	Verify 2nd field is "Send Password" button	Form has a Send Password button	Form has a Send Password button	High	High	Completed
	14.4	14.4.1	Verify 3rd field is Key	Form has a field named Key	Form has a field named Key	High	High	Completed
	14.5	14.5.1	Verify 4th field is "VERIFY" button	Form has a VERIFY button	Form has a VERIFY button	High	High	Completed
15	15.1	15.1.1	Verify "Send Password" button works	"Send Password" button works properly	"Send Password" button works properly	High	High	Completed
	15.2	15.2.1	Verify the button sends New Password at the registered phone	New Password is being sent to the Registered number	New Password is being sent to the Registered number	High	High	Completed
16	16.1	16.1.1	Verify Key has any characters	Key can have any character	Key can have any character	Low	Medium	Completed
			Verify key "RITW134" is correct	Key RITW134 is accepted	Key RITW134 is accepted	Low	Medium	Completed
			Verify key "YASH@12" is correct	Key YASH@12 is correct	Key YASH@12 is correct	Low	Medium	Completed
	16.2	16.2.1	Key can have any number of characters	Key can have any number of characters	Key can have any number of characters	Low	Medium	Completed
			Verify key "qwertyuiop" is correct	Key is correct	Key is correct	Low	Medium	Completed
			Verify key 1234asd@#$ is correct	Key is correct	Key is correct	Low	Medium	Completed

17	17.1	17.1.1	Verify if "VERIFY" Button works	VERIFY Button works properly	VERIFY Button works properly	High	High	Completed	
		17.1.2	Verify if Key is matched with Password	Key get's matched with password	Key get's matched with password	High	High	Completed	
		17.1.3	Verify, error is created when key doesn't matches with password	Error is created in that situation	Error is created in that situation	High	High	Completed	
4.MENU PAGE									
18	18.1	18.1.1	Verify if Form has 2 fields	Form has 2 fields	Form has 2 fields	High	High	Completed	
	18.2	18.2.1	Verify the 1st field is List of all the food items with corressponding	Form has a List	Form has a List	High	High	Completed	
	18.3	18.3.1	Verify the 2nd field is TOTAL	Form has a field named TOTAL	Form has a field named TOTAL	High	High	Completed	
19	19.1	19.1.1	Verify all the available checkboxes are clickable	Available checkboxes are clickable	Available checkboxes are clickable	High	High	Completed	
	19.2	19.2.1	Verify that all the drop down "Quantity" menu have 10 values	All the drop down Quantity menu have 10 values	All the drop down Quantity menu have 10 values	High	High	Completed	
20	20.1	20.1.1	Verify that the TOTAL field updates as per the number of checkboxes selected and their corressponding quantities	Total field updates accordingly	Total field updates accordingly	High	High	Completed	
		20.1.2	Verify that the TOTAL field is clickable and links to the CART	Total Field links to the CART Page	Total Field links to the CART Page	High	High	Completed	
5.CART									
21	21.1	21.1.1	Verify if Form has 4 fields	Form has 4 fields	Form has 4 fields	High	High	Completed	
	21.2	21.2.1	Verify the 1st field is the List of food items selected in the MENU Page	List has all the food items selected in the MENU Page	List has all the food items selected in the MENU Page	High	High	Completed	
	21.3	21.3.1	Verify the 2nd field is TOTAL and is same as the TOTAL in MENU Page	Form has a TOTAL field	Form has a TOTAL field	High	High	Completed	
	21.4	21.4.1	Verify the 3rd field is an Address box	Form has an Address box	Form has an Address box	High	High	Completed	
	21.5	21.5.1	Verify the 4th field is MAKE PAYMENT button	Form has a MAKE PAYMENT Button	Form has a MAKE PAYMENT Button	High	High	Completed	
22	22.1	22.1.1	Verify the Address box can take input upto 255 characters	Address box's character limit is 255	Address box's character limit is 255	High	High	Completed	
23	23.1	23.1.1	Verify that the MAKE PAYMENT Button works	MAKE PAYMENT Button works properly	MAKE PAYMENT Button works properly	High	High	Completed	
	23.2	23.2.1	Verify that the MAKE PAYMENT button links to the 3rd Party Payment portal	MAKE PAYMENT Button links to the 3rd party Payment portal successfully	MAKE PAYMENT Button links to the 3rd party Payment portal successfully	High	High	Completed	

FOODZAPP Testing

Group Members:

- Yashasvi Asthana (15BCE1161)
- Ritwik Kala (15BCE1114)
- Suryansh Bhardwaj (15BCE1047)
- Ritwik Gupta (15BCE1059)

FOODzapp is a third party android application developed using Android Studio.

Android Studio can include some testing simulators as an API. The Android Testing Support Library provides an extensive framework for testing Android apps. This library provides a set of APIs that allow you to quickly build and run test code for your apps, including JUnit 4 and functional user interface (UI) tests. You can run tests created using these APIs from the Android Studio IDE or from the command line.

Such an API is UI Automator:

The UI Automator testing framework provides a set of APIs to build UI tests that perform interactions on user apps and system apps. The UI Automator APIs allows you to perform operations such as opening the Settings menu or the app launcher in a test device. The UI Automator testing framework is well-suited for writing black box-style automated tests, where the test code does not rely on internal implementation details of the target app.
The key features of the UI Automator testing framework include:

- A viewer to inspect layout hierarchy.
- An API to retrieve state information and perform operations on the target device.
- APIs that support cross-app UI testing.

This tool requires Android 4.3 (API level 18) or higher.

CODING:

XML:

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.foodzapp.yash.foodzapp.MainActivity"
    android:backgroundTint="#b0527dd4"
    android:background="#b25f61d1">

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Phone Number"
        android:maxLength="10"
        android:id="@+id/phone"
        android:layout_marginTop="136dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:inputType="phone"
        android:textColor="#c73737" />
```

```
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="Password"
    android:id="@+id/password"
    android:inputType="textPassword"
    android:textStyle="normal"
    android:layout_below="@+id/phone"
    android:layout_alignLeft="@+id/phone"
    android:layout_alignStart="@+id/phone"
    android:textColor="#c73737"
    android:selectAllOnFocus="true"
    android:focusable="true" />

<TextView
    android:layout_width="120dp"
    android:layout_height="60dp"
    android:textAlignment="center"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="LOGIN"
    android:id="@+id/LOGX"
    android:textStyle="bold"
    android:textSize="30dp"
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:textColor="@color/light_font"
    android:shadowColor="@color/text_shadow"
    android:shadowDx="1"
    android:shadowDy="1"
    android:shadowRadius="2" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Login"
    android:id="@+id/login_button"
    android:layout_marginTop="43dp"
    android:layout_below="@+id/password"
    android:layout_centerHorizontal="true"
    android:clickable="true" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Forgot Password?"
    android:id="@+id/forgotpass"
    android:clickable="true"
    android:textSize="17dp"
    android:textStyle="bold"
    android:layout_marginBottom="24dp"
    android:layout_alignParentBottom="true"
    android:layout_toRightOf="@+id/password"
    android:layout_toEndOf="@+id/password" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="SignUp"
    android:id="@+id/signup"
    android:clickable="true"
    android:textSize="17dp"
    android:textStyle="bold"
```

```
        android:layout_marginBottom="27dp"
        android:layout_above="@+id/forgotpass"
        android:layout_toRightOf="@+id/password"
        android:layout_toEndOf="@+id/password" />

</RelativeLayout>
```

activity_forgotpass.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.foodzapp.yash.foodzapp.forgotpass">




    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="phone"
        android:ems="10"
        android:id="@+id/phn1"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="103dp"
        android:textAlignment="center"
        android:hint="Phone Number" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="GO"
        android:id="@+id/gobtn"
        android:layout_below="@+id/phn1"
        android:layout_centerHorizontal="true"
```

```
    android:layout_marginTop="39dp" />

  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Verify"
    android:id="@+id/verifybtn"
    android:layout_alignParentBottom="true"
    android:layout_alignLeft="@+id/gobtn"
    android:layout_alignStart="@+id/gobtn" />

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:id="@+id/textView1"
    android:layout_below="@+id/gobtn"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="37dp" />

  <EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="number"
    android:ems="10"
    android:id="@+id/key"
    android:textAlignment="center"
    android:hint="Enter the key"
    android:layout_above="@+id/verifybtn"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="41dp" />



</RelativeLayout>
```

activity_register.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context=".register">

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="Fill in the form for Registeration"
    android:id="@+id/textView4"
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:textSize="25dp" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Register"
    android:id="@+id/reg"
    android:onClick="ret"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="57dp"
    android:textSize="25dp" />

<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:ems="10"
    android:id="@+id/pswd"
    android:hint="Password"
    android:layout_below="@+id/textView4"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginTop="138dp" />

<EditText
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:inputType="textEmailAddress"
    android:ems="10"
    android:id="@+id/email"
    android:hint="E-mail Address"
    android:layout_below="@+id/editText"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:ems="10"
    android:id="@+id/editText"
    android:hint="Confirm Password"
    android:layout_below="@+id/pswd"
    android:layout_alignRight="@+id/pswd"
    android:layout_alignEnd="@+id/pswd" />

<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="phone"
    android:ems="10"
    android:id="@+id/editText2"
    android:hint="Phone"
    android:layout_marginBottom="48dp"
    android:layout_alignBottom="@+id/pswd"
    android:layout_alignRight="@+id/pswd"
    android:layout_alignEnd="@+id/pswd" />

</RelativeLayout>
```

row.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
android:orientation="horizontal"
>


<ImageView
   android:id="@+id/img"
   android:layout_width="150dp"
   android:layout_height="100dp"
   android:layout_alignParentTop="true"
   android:layout_alignParentLeft="true"
   android:layout_alignParentStart="true" />

<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:textAppearance="?android:attr/textAppearanceLarge"
   android:text="item name"
   android:id="@+id/item"
   android:layout_below="@+id/img"
   android:layout_alignParentLeft="true"
   android:layout_alignParentStart="true" />

<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:textAppearance="?android:attr/textAppearanceLarge"
   android:text="price"
   android:id="@+id/price"
   android:textSize="20dp"
   android:layout_alignParentTop="true"
   android:layout_alignLeft="@+id/additem"
   android:layout_alignStart="@+id/additem" />

<Button
   style="?android:attr/buttonStyleSmall"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Add Item"
   android:id="@+id/additem"
   android:layout_marginRight="31dp"
   android:layout_marginEnd="31dp"
   android:layout_alignBottom="@+id/img"
   android:layout_alignParentRight="true"
```

```
    android:layout_alignParentEnd="true"
    android:textSize="15dp" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="null"
    android:id="@+id/quantity"
    android:layout_alignBottom="@+id/item"
    android:layout_alignRight="@+id/additem"
    android:layout_alignEnd="@+id/additem"
    android:textSize="20dp" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Rs."
    android:id="@+id/textView2"
    android:textSize="20dp"
    android:layout_alignBottom="@+id/price"
    android:layout_toLeftOf="@+id/price"
    android:layout_toStartOf="@+id/price" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Quantity:"
    android:id="@+id/textView3"
    android:textSize="20dp"
    android:layout_alignTop="@+id/quantity"
    android:layout_alignRight="@+id/price"
    android:layout_alignEnd="@+id/price" />

<Button
    android:layout_width="30dp"
    android:layout_height="wrap_content"
    android:text="-"
    android:id="@+id/delete"
    android:layout_alignTop="@+id/additem"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:textSize="20dp" />
```

```
</RelativeLayout>
```

activity_menu2.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.foodzapp.yash.foodzapp.Menu"
    android:windowSoftInputMode="adjustNothing">



    <ListView
        android:id="@+id/list_menu"
        android:layout_width="350dp"
        android:layout_height="350dp"
        android:layout_centerVertical="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        tools:listitem="@layout/row"
        android:divider="@android:color/transparent"
        android:dividerHeight="40sp">

    </ListView>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="MENU"
        android:id="@+id/textView"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:textSize="30dp" />
```

```
    <TextView
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:text="TOTAL = 0"
        android:id="@+id/total"
        android:textSize="26dp"
        android:layout_alignBottom="@+id/order"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Order"
        android:id="@+id/order"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />
</RelativeLayout>
```

activity_order.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.foodzapp.yash.foodzapp.order">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Your total bill amount is "
        android:id="@+id/textView5"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
```

```
    android:layout_alignParentStart="true"
    android:layout_marginTop="113dp"
    android:textSize="20dp" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="0"
    android:id="@+id/urtotal"
    android:layout_alignBottom="@+id/textView5"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:textSize="25dp" />

<Button
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="PAY NOW"
    android:id="@+id/pay"
    android:layout_marginBottom="85dp"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true" />

<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="textMultiLine"
    android:ems="8"
    android:id="@+id/address"
    android:layout_centerVertical="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Address"
    android:id="@+id/textView6"
    android:layout_alignBottom="@+id/address"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
```

```
    android:textSize="25dp" />
</RelativeLayout>
```

JAVA:

MainActitvity:

```
package com.foodzapp.yash.foodzapp;

import android.app.ActionBar;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.EditText;
import android.widget.Toast;


public class MainActivity extends AppCompatActivity {

    Button login;
    EditText phone,password;
    TextView forgotpass,signup;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


        login=(Button)findViewById(R.id.login_button);
        phone=(EditText)findViewById(R.id.phone);
        password=(EditText)findViewById(R.id.password);
        forgotpass=(TextView)findViewById(R.id.forgotpass);
```

```
signup=(TextView)findViewById(R.id.signup);


final Intent forgot=new Intent(this,forgotpass.class);
forgotpass.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(forgot);
    }
});
signup.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(MainActivity.this,register.class));
    }
});




login.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        /*
        Bundle extras = getIntent().getExtras();
        String ph=extras.getString("username");
        //String ps=extras.getString("pass");
        if(extras!=null){
            ph="notset";
            //ps="notset";
        }
        */
        if(phone.getText().toString().equals("9958792078") &&
                password.getText().toString().equals("yash")){
            Toast.makeText(getApplicationContext(),"WELCOME ADMIN...",Toast.LENGTH_SHORT).show();
        }
        else if(phone.getText().toString().equals("9958446074") &&
                password.getText().toString().equals("mak")){
            Toast.makeText(getApplicationContext(),"WELCOME USER...",Toast.LENGTH_SHORT).show();
            startActivity(new Intent(MainActivity.this,Menu.class));
        }
```

```
            /*else if(phone.getText().toString().equals(ph)&& password.getText().toString().equals(0)){
                Toast.makeText(getApplicationContext(),"WELCOME USER...",Toast.LENGTH_SHORT).show();
                startActivity(new Intent(MainActivity.this,Menu.class));
            }*/
            else{
                Toast.makeText(getApplicationContext(),"Invalid Credentials",Toast.LENGTH_SHORT).show();
            }
        }
    });


}
}
```

forgotpass:

```
package com.foodzapp.yash.foodzapp;

import java.util.concurrent.TimeUnit;

import android.app.ActionBar;
import android.app.PendingIntent;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.os.CountDownTimer;
import android.widget.EditText;


public class forgotpass extends AppCompatActivity {


    EditText phone1;
    EditText key;
    TextView resend;
```

```
Button go;
Button verify;
TextView _tv;
CountDownTimer countdowntimer;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_forgotpass);
    phone1=(EditText)findViewById(R.id.phn1);
    key=(EditText)findViewById(R.id.key);
    go=(Button)findViewById(R.id.gobtn);
    verify=(Button)findViewById(R.id.verifybtn);
    _tv = (TextView) findViewById( R.id.textView1);



    class CountDownTimerClass extends CountDownTimer {

        public CountDownTimerClass(long millisInFuture, long countDownInterval) {

            super(millisInFuture, countDownInterval);

        }

        @Override
        public void onTick(long millisUntilFinished) {

            int progress = (int) (millisUntilFinished/1000);

            _tv.setText(Integer.toString(progress)+" seconds left to enter the key");

        }

        @Override
        public void onFinish() {

            _tv.setText("TIME UP !");
            key.setEnabled(false);

        }
    }
```

```
    // ADD THE CODE TO SEND A MESSAGE ON PHONE
        go.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                key.setEnabled(true);
                go.setEnabled(false);
                SmsManager smsManager = SmsManager.getDefault();
                smsManager.sendTextMessage("9790718874", null, "Your key to login is 4533", null, null);
                countdowntimer = new CountDownTimerClass(300000,1000);
                countdowntimer.start();

            }
        });

        verify.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(forgotpass.this,Menu.class));
            }
        });




    }
}
```

register:

```
package com.foodzapp.yash.foodzapp;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class register extends AppCompatActivity {
```

```
    Button register;
    EditText newphone;
    EditText newpass;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        newphone=(EditText)findViewById(R.id.phone);
        newpass=(EditText)findViewById(R.id.pswd);
        register=(Button)findViewById(R.id.reg);
        register.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent i=new Intent(getApplicationContext(),MainActivity.class);
                //i.putExtra("username",newphone.getText().toString());
                //i.putExtra("pass",newpass.getText().toString());
                startActivity(i);
            }
        });

    }
}
```

CustomList:

```
package com.foodzapp.yash.foodzapp;

/**
 * Created by Yashasvi on 31-10-2016.
 */

import android.app.Activity;
import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
```

```
import java.util.ArrayList;
import java.util.HashMap;

public class CustomList extends ArrayAdapter<String>{

   private final Activity context;
   private final String[] web;
   private final String[] price;
   private final Integer[] imageId;
   private final String[] quantity;
   private final Integer[] additem;


   public CustomList(Activity context,
                     String[] web, String[] price, Integer[] imageId,String[] quantity,Integer[] additem) {
      super(context, R.layout.row, web);
      this.context = context;
      this.web = web;
      this.price = price;
      this.imageId = imageId;
      this.quantity=quantity;
      this.additem=additem;

   }




   @Override
   public View getView(final int position, View view, ViewGroup parent) {
      LayoutInflater inflater = context.getLayoutInflater();

      View rowView= inflater.inflate(R.layout.row, null, true);
      TextView txtTitle = (TextView) rowView.findViewById(R.id.item);
      TextView pri=(TextView) rowView.findViewById(R.id.price);
      ImageView imageView = (ImageView) rowView.findViewById(R.id.img);
      Button additem=(Button) rowView.findViewById(R.id.additem);
      final TextView quan=(TextView) rowView.findViewById(R.id.quantity);
      Button delete=(Button) rowView.findViewById(R.id.delete);
      quan.setText(quantity[position]);
      additem.setOnClickListener(new View.OnClickListener() {
         @Override
```

```
        public void onClick(View v) {
            int a=Integer.valueOf(quantity[position]);
            a++;
            quantity[position]=String.valueOf(a);
            quan.setText(quantity[position]);
        }
    });
    delete.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            int a=Integer.valueOf(quantity[position]);
            a--;
            if(a==-1){
                a=0;
            }
            quantity[position]=String.valueOf(a);
            quan.setText(quantity[position]);
        }
    });
    txtTitle.setText(web[position]);
    pri.setText(price[position]);
    imageView.setImageResource(imageId[position]);
    return rowView;
  }

}
```

menu:

```
package com.foodzapp.yash.foodzapp;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.app.Activity;
```

```
public class Menu extends Activity {
    TextView totaltxt;
    int total;
    Button order;
    ListView list;

    String[] web = {
            "Butter Chicken","Kadai Chicken",
            "Kadai Paneer","Shahi Paneer","Butter Naan","Coke"
    };

    String[] price = {
            "200","210","170","165","30","35"
    };

    Integer[] imageId = {
            R.drawable.download,R.drawable.kadai,
            R.drawable.kadaipaneer,
            R.drawable.shahi,
            R.drawable.naan,
            R.drawable.coke
    };

    String[] quantity= {
            "0","0","0","0","0","0"
    };

    Integer[] additem={
            R.id.additem
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_menu2);
        totaltxt=(TextView)findViewById(R.id.total);
        order=(Button)findViewById(R.id.order);
        CustomList adapter = new CustomList(Menu.this, web, price, imageId,quantity,additem);
        list=(ListView)findViewById(R.id.list_menu);
        list.setAdapter(adapter);
        list.setOnItemClickListener(new AdapterView.OnItemClickListener() {
```

```
        @Override
        public void onItemClick(AdapterView<?> parent, final View view,
                                final int position, long id) {
            Toast.makeText(Menu.this, "You Clicked at " +web[+ position], Toast.LENGTH_SHORT).show();

        }
    });
    totaltxt.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            total=0;
            for(int i=0;i<quantity.length;i++){
                total=total + Integer.valueOf(quantity[i])*Integer.valueOf(price[i]);
                totaltxt.setText("TOTAL =      "+total);
            }
        }
    });
    order.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent i=new Intent(getApplicationContext(),order.class);
            i.putExtra("total",total);
            startActivity(i);
        }
    });




  }

}
```

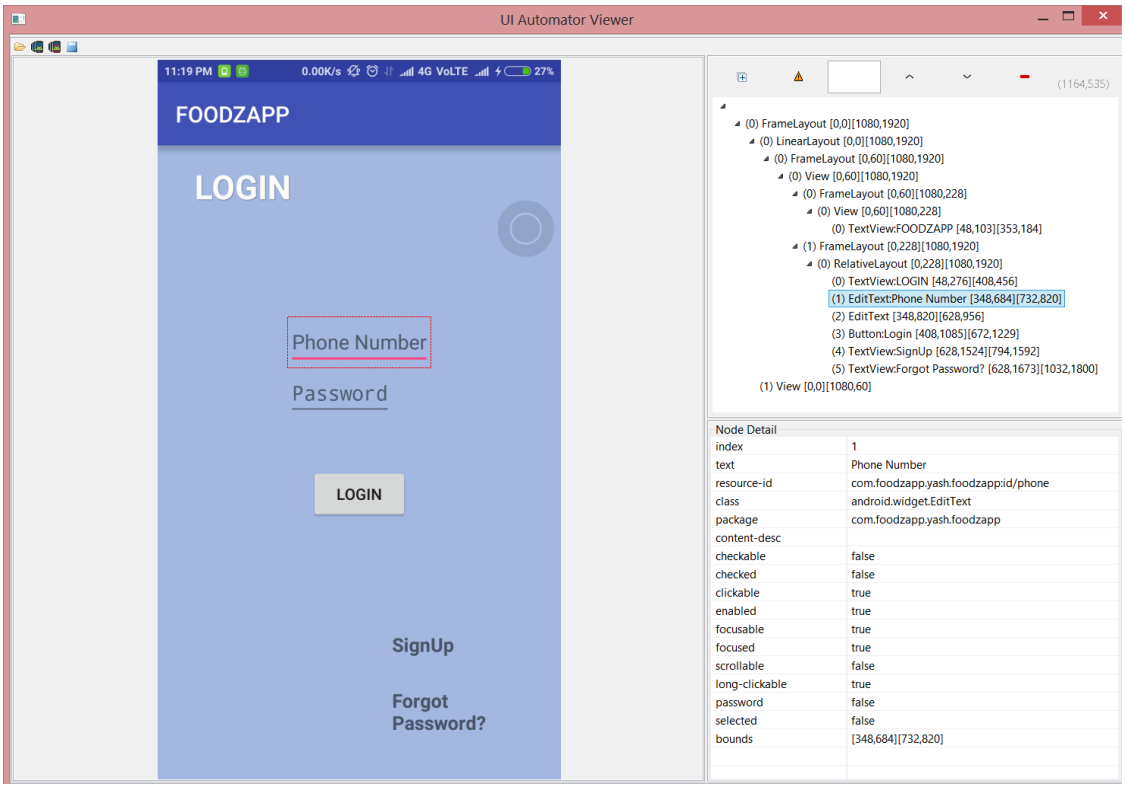
order:

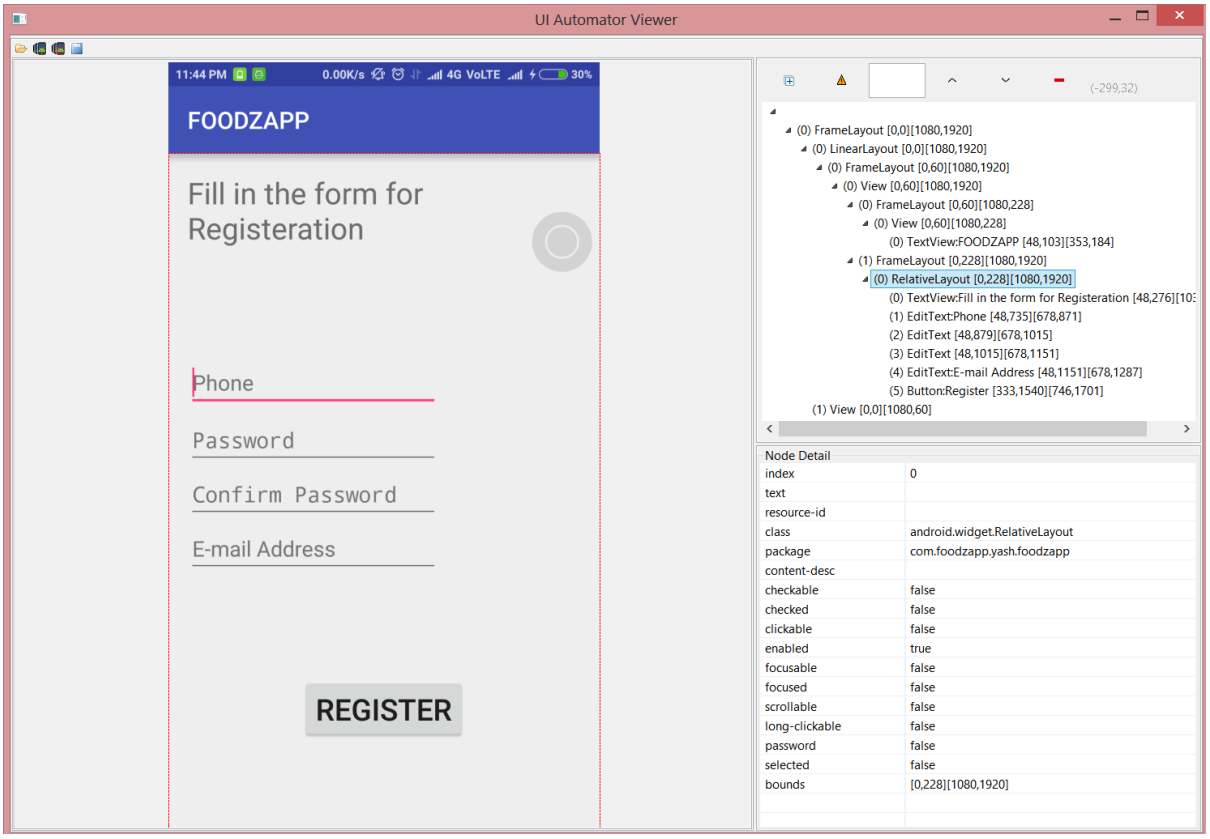
```
package com.foodzapp.yash.foodzapp;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
```

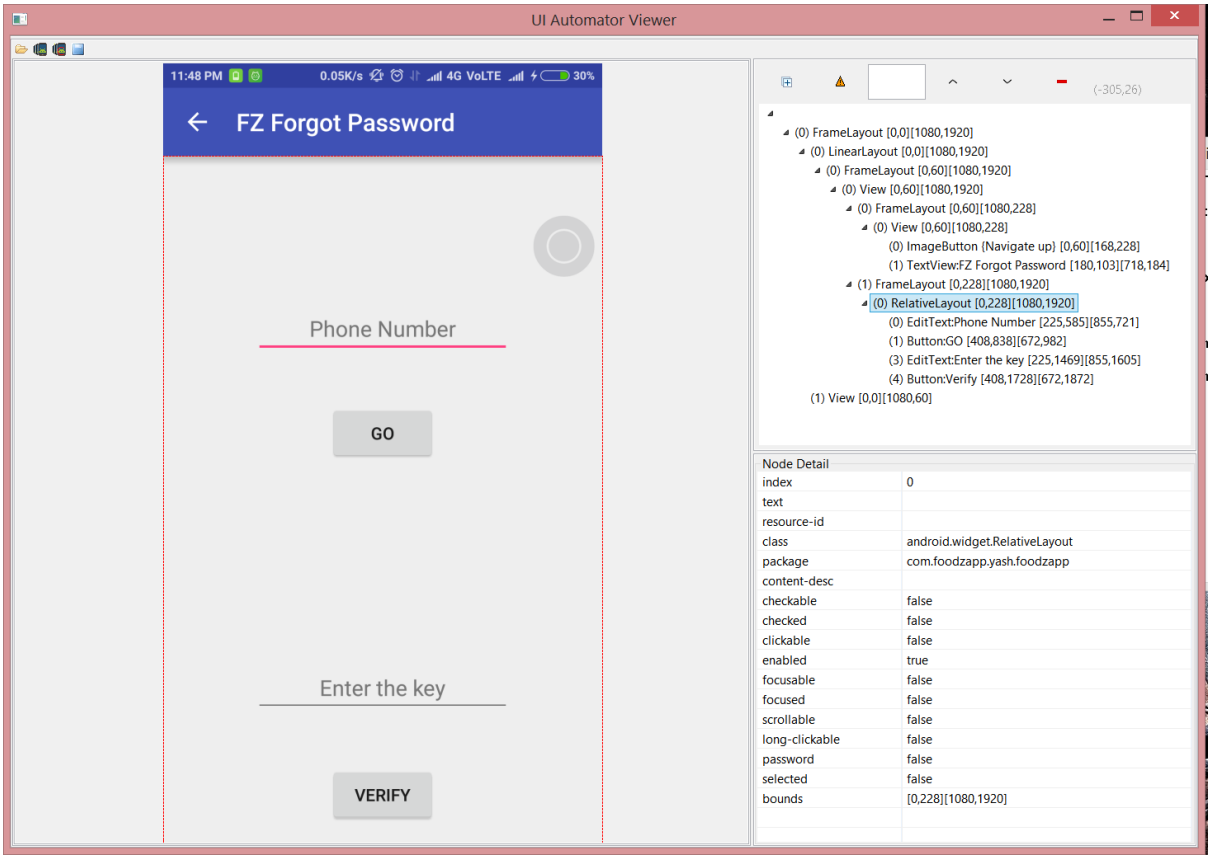
```
import android.widget.Toast;

public class order extends AppCompatActivity {



    TextView urtotal;
    EditText address;
    Button pay;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_order);
        urtotal=(TextView)findViewById(R.id.urtotal);
        int b;
        b=0;
        Bundle extras = getIntent().getExtras();
        if(extras!=null){
            b=extras.getInt("total");
        }

        urtotal.setText("Rs."+b);

        address=(EditText)findViewById(R.id.address);
        pay=(Button)findViewById(R.id.pay);
        pay.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(address.getText()==null){
                    Toast.makeText(order.this, "Please fill the Address!",
                            Toast.LENGTH_LONG).show();
                }
                Toast.makeText(order.this, "Payment portal not active!",
                        Toast.LENGTH_LONG).show();
            }
        });
    }
}
```

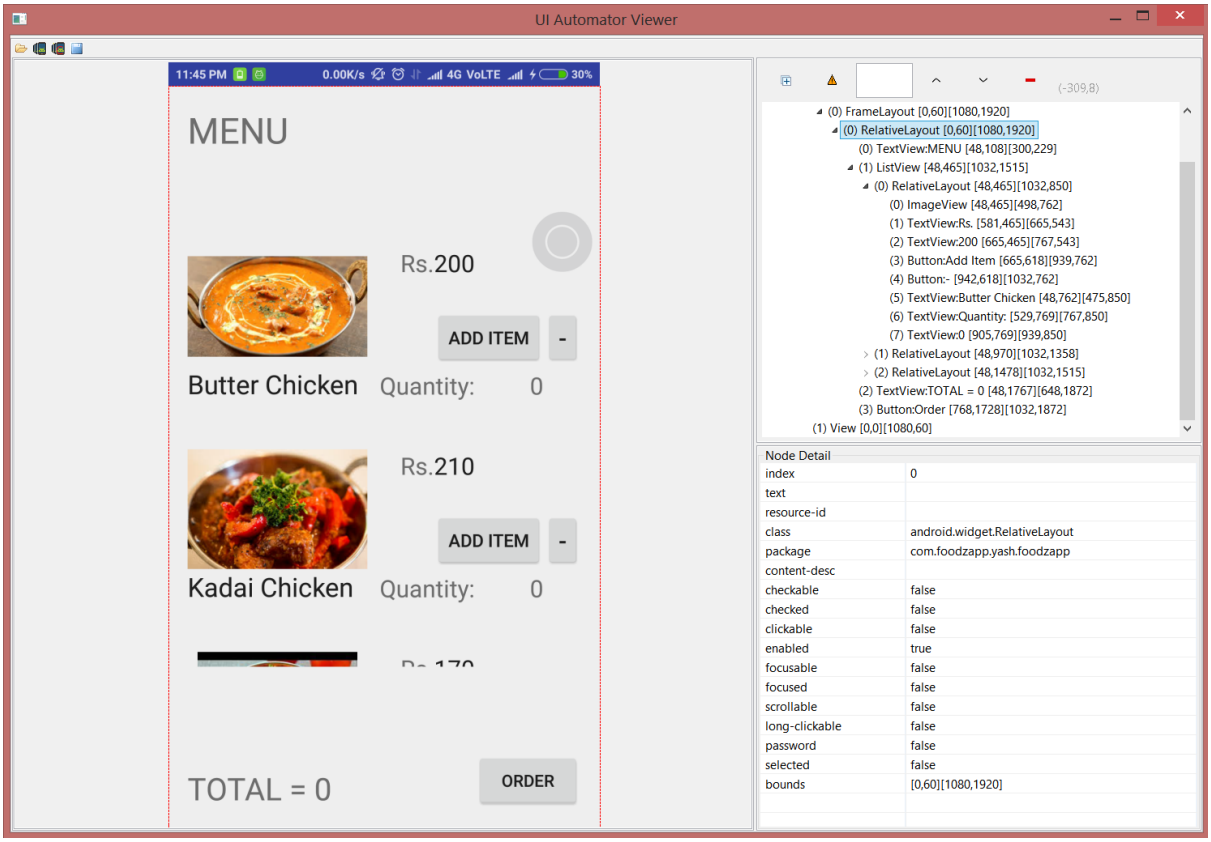
TESTING:

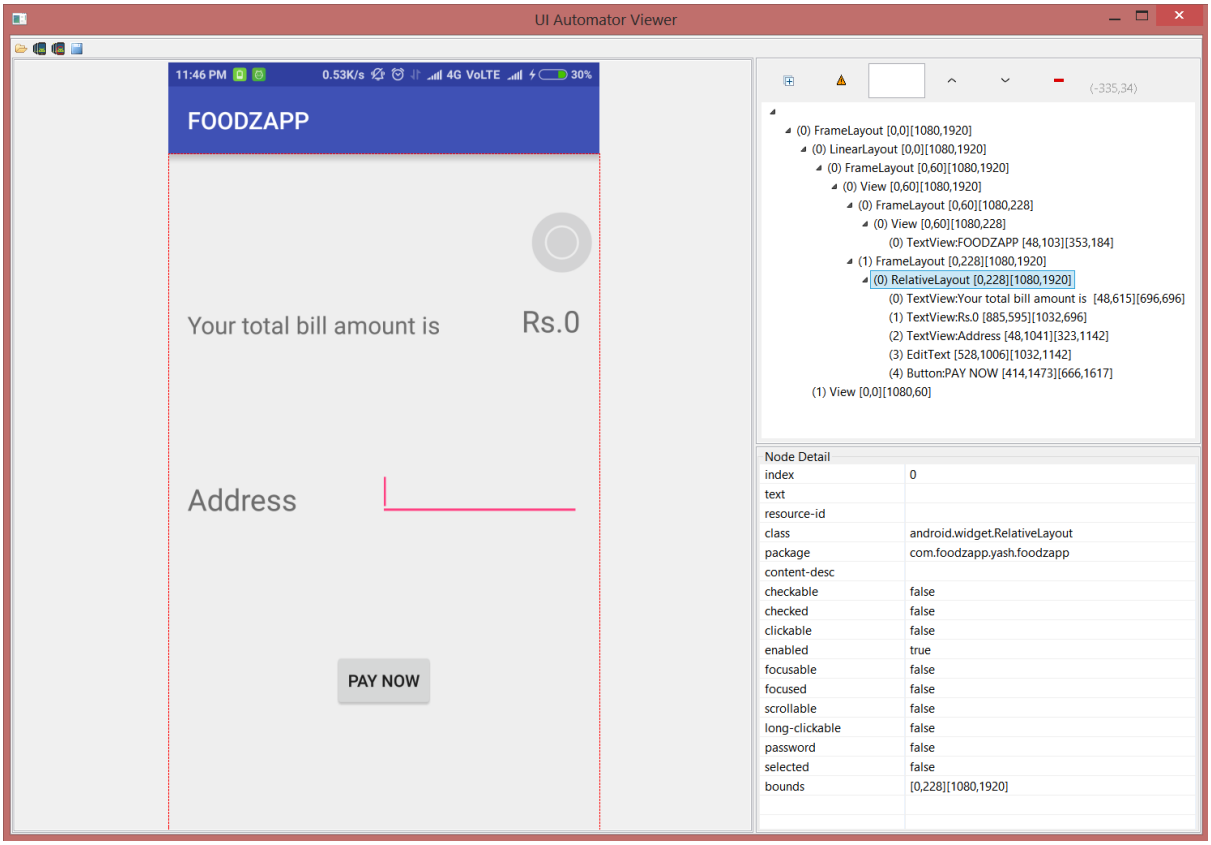
UI Automator Viewer:




UI Automator Viewer
11:44 PM 0.00K/s 4G VoLTE 30%
FOODZAPP
Fill in the form for Registeration
Phone
Password
Confirm Password
E-mail Address
REGISTER
(-299,32)
(0) FrameLayout [0,0][1080,1920]
(0) LinearLayout [0,0][1080,1920]
(0) FrameLayout [0,60][1080,1920]
(0) View [0,60][1080,1920]
(0) FrameLayout [0,60][1080,228]
(0) View [0,60][1080,228]
(0) TextView:FOODZAPP [48,103][353,184]
(1) FrameLayout [0,228][1080,1920]
(0) RelativeLayout [0,228][1080,1920]
(0) TextView:Fill in the form for Registeration [48,276][10
(1) EditText:Phone [48,735][678,871]
(2) EditText [48,879][678,1015]
(3) EditText [48,1015][678,1151]
(4) EditText:E-mail Address [48,1151][678,1287]
(5) Button:Register [333,1540][746,1701]
(1) View [0,0][1080,60]
Node Detail
index 0
text
resource-id
class android.widget.RelativeLayout
package com.foodzapp.yash.foodzapp
content-desc
checkable false
checked false
clickable false
enabled true
focusable false
focused false
scrollable false
long-clickable false
password false
selected false
bounds [0,228][1080,1920]


UI Automator Viewer
11:48 PM 0.05K/s 4G VoLTE 30%
FZ Forgot Password
Phone Number
GO
Enter the key
VERIFY
(-305,26)
(0) FrameLayout [0,0][1080,1920]
(0) LinearLayout [0,0][1080,1920]
(0) FrameLayout [0,60][1080,1920]
(0) View [0,60][1080,1920]
(0) FrameLayout [0,60][1080,228]
(0) View [0,60][1080,228]
(0) ImageButton (Navigate up) [0,60][168,228]
(1) TextView:FZ Forgot Password [180,103][718,184]
(1) FrameLayout [0,228][1080,1920]
(0) RelativeLayout [0,228][1080,1920]
(0) EditText:Phone Number [225,585][855,721]
(1) Button:GO [408,838][672,982]
(3) EditText:Enter the key [225,1469][855,1605]
(4) Button:Verify [408,1728][672,1872]
(1) View [0,0][1080,60]
Node Detail
index 0
text
resource-id
class android.widget.RelativeLayout
package com.foodzapp.yash.foodzapp
content-desc
checkable false
checked false
clickable false
enabled true
focusable false
focused false
scrollable false
long-clickable false
password false
selected false
bounds [0,228][1080,1920]


UI Automator Viewer
11:45 PM 0.00K/s 4G VoLTE 30%
MENU
Rs.200
ADD ITEM
-
Butter Chicken
Quantity:
0
Rs.210
ADD ITEM
-
Kadai Chicken
Quantity:
0
TOTAL = 0
ORDER
(0) FrameLayout [0,60][1080,1920]
(0) RelativeLayout [0,60][1080,1920]
(0) TextView:MENU [48,108][300,229]
(1) ListView [48,465][1032,1515]
(0) RelativeLayout [48,465][1032,850]
(0) ImageView [48,465][498,762]
(1) TextView:Rs. [581,465][665,543]
(2) TextView:200 [665,465][767,543]
(3) Button:Add Item [665,618][939,762]
(4) Button:- [942,618][1032,762]
(5) TextView:Butter Chicken [48,762][475,850]
(6) TextView:Quantity: [529,769][767,850]
(7) TextView:0 [905,769][939,850]
(1) RelativeLayout [48,970][1032,1358]
(2) RelativeLayout [48,1478][1032,1515]
(2) TextView:TOTAL = 0 [48,1767][648,1872]
(3) Button:Order [768,1728][1032,1872]
(1) View [0,0][1080,60]
Node Detail
index 0
text
resource-id
class android.widget.RelativeLayout
package com.foodzapp.yash.foodzapp
content-desc
checkable false
checked false
clickable false
enabled true
focusable false
focused false
scrollable false
long-clickable false
password false
selected false
bounds [0,60][1080,1920]


UI Automator Viewer
11:46 PM 0.53K/s 4G VoLTE 30%
FOODZAPP
Your total bill amount is
Rs.0
Address
PAY NOW
(0) FrameLayout [0,0][1080,1920]
(0) LinearLayout [0,0][1080,1920]
(0) FrameLayout [0,60][1080,1920]
(0) View [0,60][1080,1920]
(0) FrameLayout [0,60][1080,228]
(0) View [0,60][1080,228]
(0) TextView:FOODZAPP [48,103][353,184]
(1) FrameLayout [0,228][1080,1920]
(0) RelativeLayout [0,228][1080,1920]
(0) TextView:Your total bill amount is [48,615][696,696]
(1) TextView:Rs.0 [885,595][1032,696]
(2) TextView:Address [48,1041][323,1142]
(3) EditText [528,1006][1032,1142]
(4) Button:PAY NOW [414,1473][666,1617]
(1) View [0,0][1080,60]
Node Detail
index 0
text
resource-id
class android.widget.RelativeLayout
package com.foodzapp.yash.foodzapp
content-desc
checkable false
checked false
clickable false
enabled true
focusable false
focused false
scrollable false
long-clickable false
password false
selected false
bounds [0,228][1080,1920]

Foodzapp: Online Food Ordering App

Ritwik Kala, Yashasvi Asthana, Ritwik Gupta, Suryansh Bhardwaj

Prof. Anusha K.

School of Computing Science and Engineering

Introduction

The main objective behind this project was to provide a reliable online food ordering system that is robust and flexible. We have created an android application as platform for this purpose. The interactive graphical user interface and the overall simplicity of the application makes it easier for customers/users to place and pay for orders online.

Scope of Project

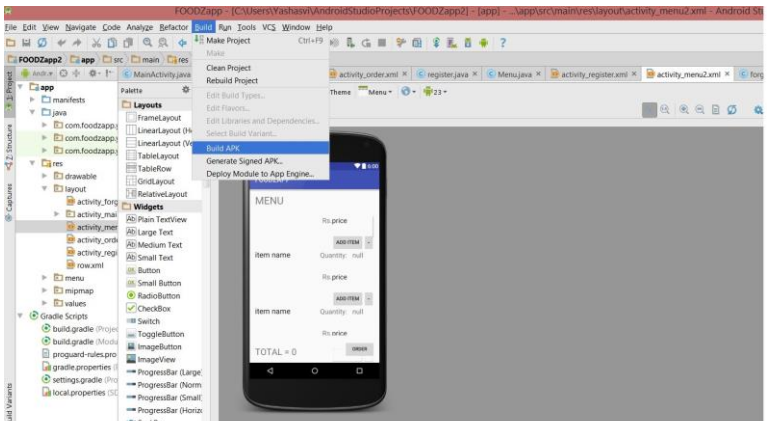
The scope our project is to provide an online food ordering android based application. It is a third party application which can be integrated into any restaurant system. This will allow that restaurant or food chain to take orders through this app. Users can select anything from the restaurant menu and can place order by giving his/her name, phone number and address. The user has an option to pay the bill online through this app or at the time of delivery.

Methodology

Foodzapp is an online food ordering android based application. The application was created using Android Studio.

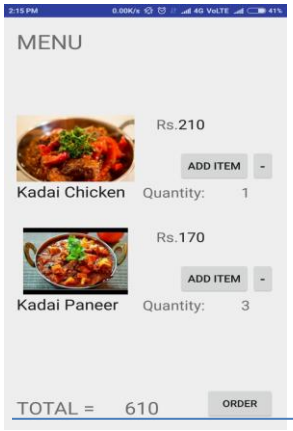
The interface of the android applications was designed through efficient XML Coding while the linking of pages and functionalities were designed using JAVA Coding.

An APK was build using Android Studio through which the app can be installed.



Results

This is the web application for Foodzapp which includes the functions as stated above. The screenshot includes interface through which we can login and place orders on the application.



This screenshot includes all the basic food ordering facilities of the application.

Conclusion

This project was not possible without the help and support from the faculties who assigned us and guided us during this project. We were successfully able to implement the basic functionality and provide an interesting application for both consumers and sellers.

Contact Details

ritwik.gupta2015@vit.ac.in
ritwik.kala2015@vit.ac.in
yashasvi.ashthana2015@vit.ac.in
suryansh.bhardwaj2015@vit.ac.in