Final Year Project Template

Dear Students

This guide is prepared to help the students in preparing their final year project reports. Report writing is one of the primary professional responsibilities of a practising software engineer. The final report of any project is not just a formality. It is a primary product of the engineering efforts and is often the basis for evaluation of the software engineer's professional abilities. The report is also a service to the software community who needs the information regarding that particular software. The report should stand on its own and it should include all the necessary sections, targeting at a reader who does not necessarily have any prior knowledge about the project or the technology involved in it.

How to follow this guideline?

The sections in italics of this format report are instructions that you are supposed to follow. The other sections are meant as information that will help you satisfy the minimum requirement of the report. The first portion of this report elaborates on the generic issues involved in report writing such as intellectual property, plagiarism, formatting etc.

Who is the Target Reader?

The target audience of these project reports is bachelor level CS student who has only basic knowledge of the computer science field. You should, therefore, assume that the reader is someone who does not know anything about your project but who may be interested in replicating your work in his or her own project. Your report should, therefore, include all the details about the 'what?', 'why?' and 'how?' that would enable someone to do this.

To ensure the reports are easy to read with a consistent format, it is very important that students strictly follow the instructions during report preparation. You have taken several courses to improve your English writing skills during the BSCS program. You are supposed to follow the elements and techniques you have learnt in these courses especially the knowledge you have gained in "Technical Report Writing" should be evident in all versions of your reports.

Submission Process

It is the reasonability of the project supervisor to make sure that the template is followed during the documentation of the different phases of the project. This is a generic document and it can be changed as per the project requirement but *only with the approval of the project supervisor. You should focus on completing the artifacts provided in this guideline without worrying about the size of the report. This document must be submitted to the project coordinator on the due date with proper approval from the supervisor and co-supervisor. Failure to follow this guideline will result in the cancellation of the project. The document will be reviewed by the supervisor as well as your English instructor and accepted only when its soundness satisfies both of them. Following portions of the report must be completed at each step of the projects.*

Chapter No	***Title***	***Completion Stage***
1 of V 1.0	*Introduction*	*Proposal Defense*
2,3 of V 1.0	*Software Requirement specification System Design*	*Part 1 Mid-Term*
V 1.0 Complete		*Part 1 Final*
V 2.0 Complete		*Part II Mid-Term*
V 3.0 Complete		*Part II Final*

Project Types

There are two types of Final Year Project: the Development Project and the Research project.

- Development Projects
  The objective of this type of the project is to develop a system that meets a set of user needs. This may take many forms such as a server, a program, a library, a collection of programs, an embedded system, plug-ins, modification to existing software etc.
  The focus of this project is on the sound software engineering principles and functionality of the software you have produced. Your project will be evaluated in terms of how well it meets the user needs, how well is it tested and the user-friendliness of the interface.
- **Research Project**
  The objective of the research-oriented project is to solve a research problem. This may take the form of evaluating the effectiveness of existing solutions, modify existing solutions or develop a new solution to the problem.

The focus of this project is on the sound experimental technique and evaluating the solution thoroughly. You will do a background research on the domain and develop a basis of your work. The success of your solution will be evaluated on the basis of understanding and use of experimental methods as well as evaluation methodology of your solution.

You should discuss this question with your supervisor at the proposal defence stage of the project to decide whether your project is a research-oriented or a development project.

What is intellectual property?

Intellectual Property is the term used to describe the outputs of creative endeavour in literary, artistic, industrial, scientific and engineering fields that can be protected under legislation. During the course of your project, you may generate some novel work, therefore, it is necessary to understand the concept of Intellectual Property.

Plagiarism – What is it?

a) Unauthorized act of copying/reproducing or attempt to copy an idea, writing or invention of another person
b) Extraction of academic data which are the results of research undertaken by another person, such as findings of the research, data obtained, whether published or unpublished, without giving due acknowledgement to the original source.
c) Unauthorized translation of the writing of another person from one language to another whether wholly or partly.

What must be done to avoid plagiarism?

a) ***Citation and references.***
Following are the main items that require citation.

- ***Direct quotes:*** *phrases, sentences, or sections copied directly from a text; cite with quotation marks (use a limited amount of text, not a full text)*
  Example: "*Failure to reference appropriately will be considered unethical academic behaviour and could result in allegations of misconduct*." [1]

The [1] symbol at the end of the quote refers to the first entry in a list of references, as shown at the end of this template under a heading 'references'.

- ***Paraphrased text:*** *sections of your writing that are based on research (not common knowledge) but written in your own words (not in quotes*
- ***Facts and Figures:*** *numbers, percentages, and facts that have been collected by an exclusive source (such as during an experiment or poll)*
- ***Theories, methods, and ideas****: an original idea or thought that you find during your research and present in your writing*
- ***Images, graphs, illustrations:*** *always follow copyright rules when using images, including those you find online*

  Paraphrased text, facts and figures, theories, methods and ideas, Images, graphs and illustrations should be referenced with symbols [ ] as shown in above example.

  In order to have an accurate record of what you have researched and therefore an accurate reference, it is important that you write down the details of your sources as you study. You should keep a complete list of references as presented in the last section of this report template.

b) ***Be familiar with the area that you are talking about.*** *By understanding the subject, you are more likely to write in your own words, rather than restate someone else's definition of this subject. Look for information on the topic you want to write about. This can be on the Internet or in books, although books are almost always more authoritative than the Internet.*

c) ***When in doubt, give credit.*** *Mention the source inside your paraphrase: "According to Richard Feynman, quantum electrodynamics can be described using path integral formulations."*

Spelling

There is no excuse for spelling mistakes in any report as spelling errors create a bad impression. Always use a spell checker; they are invaluable for picking up typographical errors as well as genuine spelling mistakes. Note that spelling checkers cannot detect cases where the wrong word happens to be a real word e.g. from the – form. So a careful proofread is necessary.

Writing in the third person

We would strongly advise writing FYP report in the third person. This provides a greater sense of objectivity and distance as the focus is on what is being said rather than who is saying it. To write in the third person, you write as if you are an outsider reporting on the aims, methods and outcomes of your project, rather than writing as though they are happening to you. First person pronouns such as "I" and "We" are replaced with third person pronouns such as "the project leader/team", "he/she, "it" and "they".

General Formatting Guidelines

Here are some general formatting guidelines that apply to the entire report:

- *Use 1- or 1-1/2-inch margins for all four margins of the report. You might want to use a 1-1/2-inch margin at the top and 1-inch margins for the left, right, and bottom.*
- *Use a 1-1/2-inch left margin if your binding uses a lot of space*
- *Generally use double-spaced typing except in those areas where single spacing is shown (for example, in the transmittal letter, descriptive abstract, figure titles, short vertical lists, and items in the information-sources list).*
- *Use one side of the paper only.*

Headings: Specific Format and Style

First-Level Headings

Follow these guidelines for first-level headings:

- *Capitalize each word of first level heading except preposition and article but if they appear as the first word of heading then capitalize them as well.*
- *Use Roman OR Arabic numerals with first-levels.*
- *Either underline the words but not the Roman/Arabic numeral, OR bold the entire heading including the numeral.*
- *Make first-levels centred on the page.*
- *Start a new page whenever you have a first-level heading.*
- *Begin first-levels on the standard first text line of a page.*
- *Leave 3 blank lines between first-levels and the first line of text.*

- *Use 18-font size.*
- *Use decimal numbering system for headings.*

Second-Level Headings

Follow these guidelines for second-level headings:

- *Capitalize each word in second-level heading.*
- *Use 16-font size and bold.*
- *Make second-levels flush left.*
- *Leave 2 blank lines between previous text and second-levels.*
- *Leave 1 blank line between second-levels and the following text.*

Third-Level Headings

Follow these guidelines for third-level headings:

- *Make third-levels sentence-style.*
- *Use bold for third-levels.*
- *Do not make third-levels a grammatical part of sentences that follow.*
- *Use the standard spacing between paragraphs for paragraphs that contain third-levels.*

Page-Numbering Style

- *All pages within the front and back covers are, but the page number is not always displayed.*
- *All pages coming before page 1 of the introduction use lowercase Roman numerals.*
- *All pages beginning with page 1 of the introduction use with Arabic numerals.*
- *Page numbers are not displayed on the transmittal letter, title page, and first page of the table of contents, page 1 of the introduction, and the appendix divider page.*
- *There are several choices of pagination style for the main-text pages:*
    - *Center page numbers at the bottom (halfway between the last text line and the bottom edge of the paper).*
    - *Place page numbers in the top right corner (on the right margin, halfway between the top text line and the top edge of the paper). Do not display page numbers on any page with a centred (first-level) heading (display it centred at the bottom).*

- *Some word-processing software causes problems in implementing these pagination guidelines; let your instructor know.*

Figures and Table Labels

- *Every figure and table must be labelled and referenced in the text. The label of the figure is placed on the bottom of the figure and list of figures must also be generated. A figure without its description cannot convey the intended meaning to the reader. Label of the table is placed at the top of the tables and list of tables must also be generated.*

Menu Drive (Project Title) V 1.0

Font: Impact

Size: 24

Font: Arial

Size: 14

STUDENT NAME 1

STUDENT NAME 2

Font: Times

Size: 16

Fall-2017

Department of Computer Science

Capital University of Science & Technology, Islamabad

Submission Form for Final-Year

PROJECT REPORT

Version	V 1.0	**NUMBER OF MEMBERS**	

TITLE	

SUPERVISOR NAME	

MEMBER NAME	**REG. NO.**	**EMAIL ADDRESS**

MEMBERS' SIGNATURES

____

____

____

Supervisor's Signature

Note 1: This paper must be signed by your supervisor
Note 2: The soft-copies of your project report, source codes, schematics, and executable should be delivered in a CD

APPROVAL CERTIFICATE

This project, entitled as “Menu Drive (Insert Your Project Title Here) ” has been approved for the award of

Bachelors of Science in Computer Science

Committee Signatures:

Supervisor: ___________________________

(Mr. Qamar Uz Zaman)

Project Coordinator: __________________________

(Mr. Abrar Arshad)

Head of Department: __________________________

(Dr. Nayyer Masood)

DECLARATION

I/We, hereby, declare that “No portion of the work referred to, in this project has been submitted in support of an application for another degree or qualification of this or any other university/institute or other institution of learning”. It is further declared that this undergraduate project, neither as a whole nor as a part thereof has been copied out from any sources, wherever references have been provided.

MEMBERS’ SIGNATURES

ACKNOWLEDGEMENTS

It is usual to thank those individuals who have provided particularly useful assistance, technical or otherwise, during your project. Your supervisor will obviously be pleased to be acknowledged as he or she will have invested quite a lot of time overseeing your progress.

Executive Summary

An abstract summarizes, usually in one paragraph of 300 words or less, the major aspects of the entire project report in a prescribed sequence. Anyone unfamiliar with your project can have a good idea of what it's about having read the abstract alone and will know whether it will be of interest to them.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter provides a brief summary of project scope, project specification, a comparison study with the available existing solutions and existing tools and technologies may be used for the development of this project. This chapter also includes a project work breakdown structure and a proposed timeline.

1.1. Project Introduction

Nowadays, restaurants use paper-based menu cards to collect orders from customers. The current proposed system will generate a digital menu card can be viewed from a web-based and/or an android mobile. The customer can place the order using this menu card. The restaurant management can track and manage the order automatically. A restaurant manager, waiter, customer and delivery boy can be considered as main actors of the proposed system.

A customer can place the order from the restaurant website or use his android mobile. He can reserve a table in the dining hall.

A waiter can view the items of an order. He can take an order from the customer using his mobile and can forward the order to the management for cooking.

A restaurant manager can assign a newly placed order to a chef for cooking. A restaurant manager can view the orders. He can track the status of an order either it is completed or in a process to completion. He can assign a delivery boy to a completed order for home delivery and can track the location of a delivery boy. The location of a delivery boy will be shown to the restaurant manager using Google map.

A restaurant manager can communicate with the waiter through his/her mobile by sending a text message for the completed orders to be served to the customers in the dining hall.

He can view the log about the table reservation, customers reserved online and can confirm or cancel the reservation.

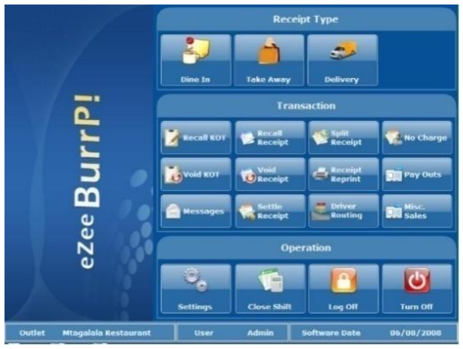
A kitchen manager can view the items to be cooked for an order and can update the status when the order is ready.

This can facilitate any restaurant and make the order processing fast and efficient.

1.2. Existing Examples / Solutions

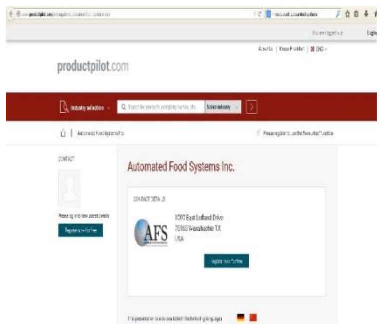
There are a number of application that provide such kind of functionalities. In the following section, some of these are listed.

REAL-TIME FULLY AUTOMATED RESTAURANT MANAGEMENT AND COMMUNICATION SYSTEM "RESTO" is a well-reputed system in the world that has been appreciated and described by the IJRET (International Journal of Research in Engineering and Technology) in their latest research study.



eZee BurrP! - A fully integrated Intuitive Restaurant POS Software is suitable to manage the services of restaurants, bars, nightclubs, quick service restaurant, delivery and other operations. It is a simple approach, easy to use and rock-solid reliability makes eZee BurrP! A great system for your POS needs. Whether your restaurant is a single store or a part of a nationwide chain, this system works perfectly.

Product Pilot is an automated system app in which different food items and products will



be selected online and then their home delivery will be done by the Delivery boy. On this application, Customer must have to register-to-register the order.

E-Delivery is another app, which works on the principles of home delivery.

Table 1.1 presents the comparison of features between Resto, Product Pilot, E-Delivery, Ezee BurrP! And Menu Drive.

Table 1.1: Existing Solutions features comparison

Sr no.	Characteristics	Resto	Product Pilot	E-Delivery	Ezee Burrp!	MENU DRIVE (Proposed System)
1	E-Menu card	√	√	√	√	√
2	Waiter will handle the android device					√
3	Real time DBMS	√	√	√	√	
4	Real-time Feedback	√	√		√	
5	Tracing					√
6	Arranging tables and chairs order					√
7	QR code ordering					√

1.3. Business Scope

The restaurant business is growing day by day. Efficient order management, tracking order and easy communication with customers can facilitate this business. Restaurant customers are well aware of the web and mobile application. A software application that may assist in above-said areas can make a place in the market and can attract the customers to place and interact with the business around the country.

1.4. Useful Tools and Technologies

J2EE, HTML, CSS, JavaScript, Android studio can be used to develop this application.

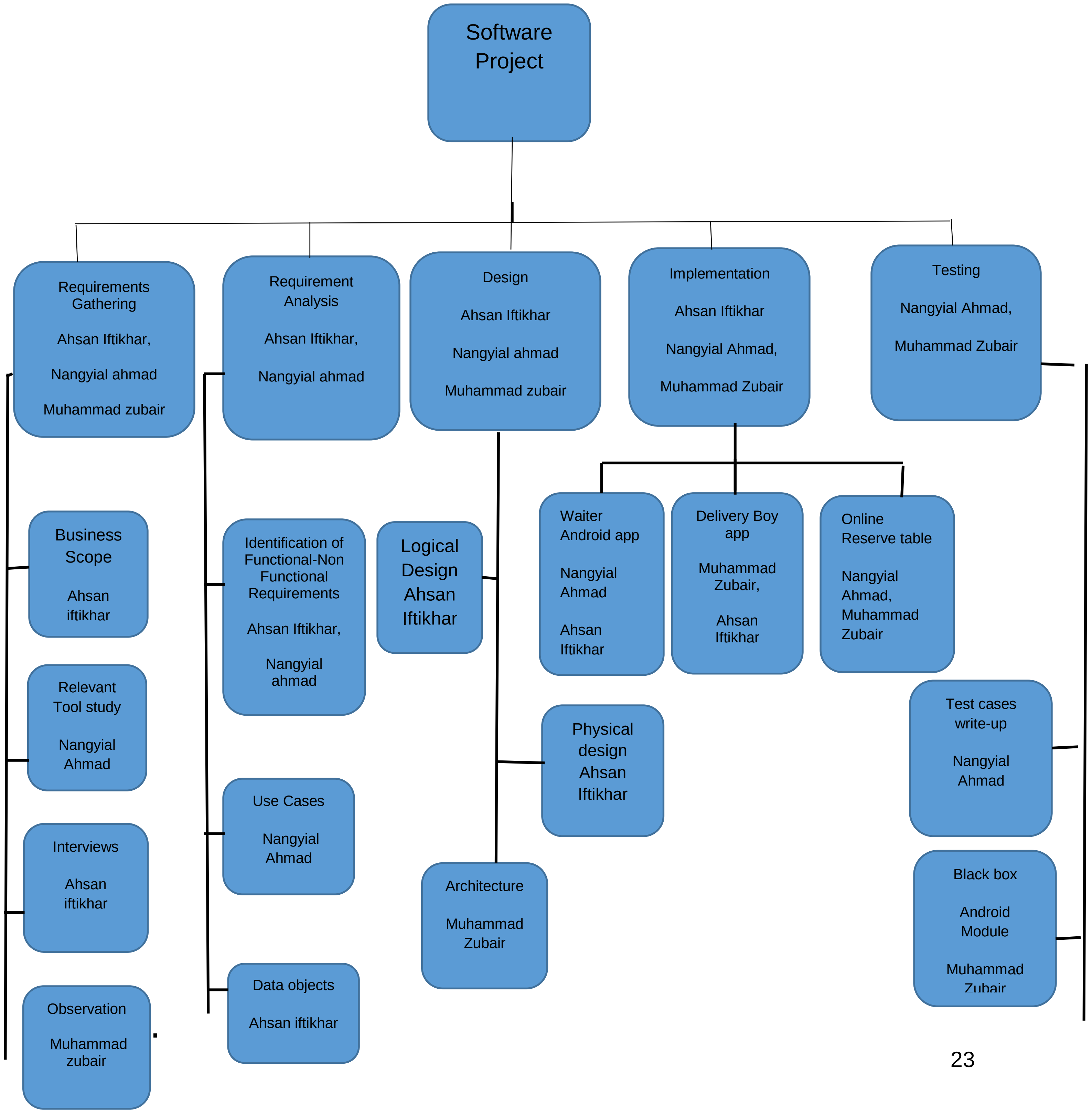
Android Studio provides portability to build Android apps.

IntelliJ provides Supports different languages like JSP, HTML and JAVA.

PostgreSQL database management system is a freeware environment and can be used on android devices

1.5. Project Work Break Down

The project breakdown structure is shown in Figure 1.1.



Software Project
Requirements Gathering Ahsan Iftikhar, Nangyial ahmad Muhammad zubair
Requirement Analysis Ahsan Iftikhar, Nangyial ahmad
Design Ahsan Iftikhar Nangyial ahmad Muhammad zubair
Implementation Ahsan Iftikhar Nangyial Ahmad, Muhammad Zubair
Testing Nangyial Ahmad, Muhammad Zubair
Business Scope Ahsan iftikhar
Relevant Tool study Nangyial Ahmad
Interviews Ahsan iftikhar
Observation Muhammad zubair
Identification of Functional-Non Functional Requirements Ahsan Iftikhar, Nangyial ahmad
Use Cases Nangyial Ahmad
Data objects Ahsan iftikhar
Logical Design Ahsan Iftikhar
Physical design Ahsan Iftikhar
Architecture Muhammad Zubair
Waiter Android app Nangyial Ahmad Ahsan Iftikhar
Delivery Boy app Muhammad Zubair, Ahsan Iftikhar
Online Reserve table Nangyial Ahmad, Muhammad Zubair
Test cases write-up Nangyial Ahmad
Black box Android Module Muhammad Zubair

1.7. Project Time Line

The proposed project timeline is shown in the following the figure.

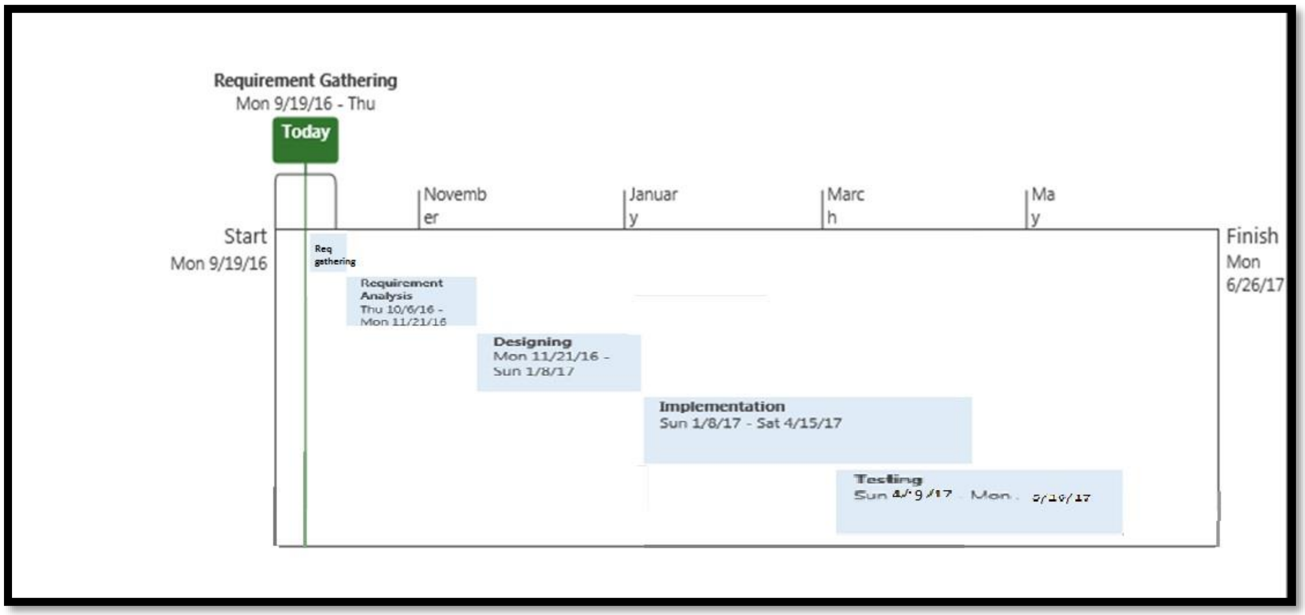


Fig 1.2: Project Timeline

Chapter 2

Requirement Specification and Analysis

Requirements analysis is a process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications. In Chapter 2 we will enlist the functional and non-functional requirements and model functional requirements in the form of use case model.

2.1. Functional Requirements

A functional requirement defines a function of a system or its component. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish.

Table 2.1: Functional Requirements

S. No.	Functional Requirement	Type	Status
1	The waiter takes order from a customer on the mobile app.	Core	Pending
2	Waiter views Menu.	Core	Pending
3	Waiter views Prices of the meal on the Mobile app.	Core	Pending
4	Waiter sends the order to the Kitchen Manager.	Core	Pending
5	Waiter checks order's status by receiving notification from kitchen manager.	intermediate	Pending
7	Waiter views food item.	intermediate	Pending
8	Waiter finalizes the order.	Core	Pending

9	Kitchen manager views food order status.	Core	Pending
10	Kitchen manager updates food order’s status as pending or completed category.	Core	Pending
11	The customer selects the different items, which he/she wants to order.	Core	Pending
12	The customer also removes ads or updates his/her order.	intermediate	Pending
13	According to his /her order bill, Detail and time of delivery will be shown to the Customer.	Core	Pending
14	The customer signs up for the first time he/she used the system online.	Core	Pending
15	Customer placing the order by using QR code.	Core	Pending
16	Customer login to the system.	Core	Pending
17	Customer selects a table from available tables	Intermediate	Pending
18	Customer modifies his account information.	Intermediate	Pending
19	The customer selects the time and date of the reservation.	Core	Pending
20	Kitchen manager sign into the system.	Core	Pending
21	Kitchen manager sign out from the system	Intermediate	Pending
22	The manager will be provided/allocated an account from the Web Admin.	Core	Pending
23	Manager login to the Website.	Core	Pending
24	Manager updates his profile information.	Intermediate	Pending
25	Hall and table information uploaded by the customer.	Intermediate	Pending
26	Manager views mostly booked table, Sales, Which delivery boy delivers most orders and mostly ordered food.	Intermediate	Pending

27	Manager views the table’s status.	Intermediate	Pending
28	Manager manages food information.	Intermediate	Pending
29	The manager adds updates or deletes food category.	Intermediate	Pending
30	Manager views delivery boy’s location.	Core	Pending
31	Manager assigns food delivery order to the Delivery boy.	Core	Pending
32	Manager generates QR code.	Core	Pending
33	Manager signs out from the system.	Intermediate	Pending

2.2. Non-Functional Requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. They are contrasted with functional requirements that define specific behaviour or functions.

Table 2.2: Non-Functional Requirement

S. No.	Non-Functional Requirements	Category
1	The system should give possible suggestions if the user enters wrong input.	Usability
2	Visiting customer can place an order by himself only by scanning QR code.	Security
3	The system should verify the information correctly.	Security
4	The system should keep and retrieve record correctly.	Reliability

2.3. Selected Functional Requirements

Following is the list of the requirements selected for the current iteration.

Table 2.3: Selected Set of Requirements for Current Iteration

S. No.	Functional Requirement	Type
1	The customer selects the different items, which he/she wants to order.	New
2	The customer also removes ads or updates his/her order.	New
3	According to his /her order bill, Detail and time of delivery will be shown to the Customer.	New
4	The customer signs up for the first time he/she used the system online.	New
5	Customer placing the order by using QR code.	New
6	Customer login to the system.	New
7	Customer selects a table from available tables	New
8	Customer modifies his account information.	New
9	The customer selects the time and date of the reservation.	New

2.4. System Use Case Modeling

A use case is a list of actions or event steps, typically defining the interactions between a role (known in the Unified Modeling Language as an actor) and a system, to achieve a goal. The actor can be a human or other external system. Customer's use cases are shown in the following the figure.

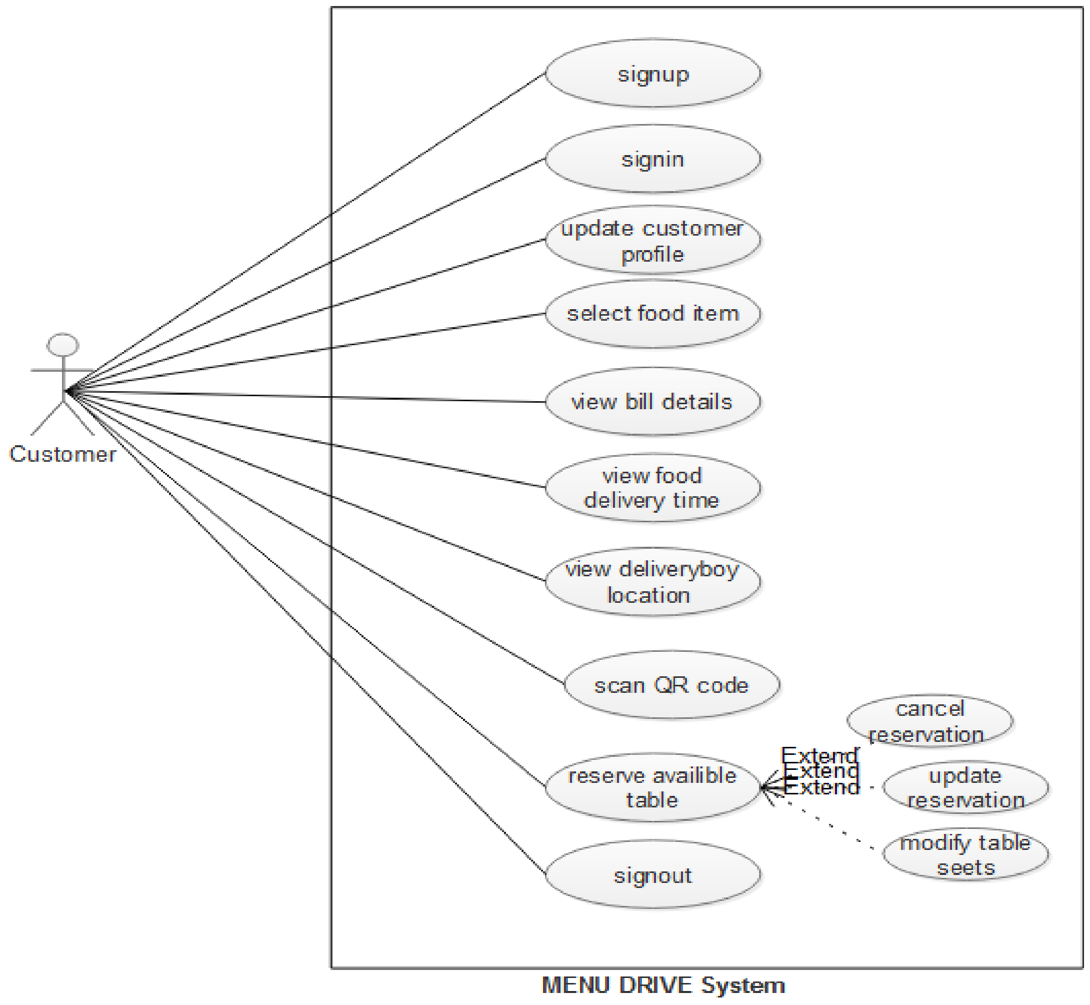


Figure 2.1: Customer's use case Diagram

Table 2.4: Customer Signup

Use Case ID:	Uc1		
Use Case Name:	Signup		
Created By:	Ahsan Iftikhar	Last Updated By:	Nangyial Ahmad
Date Created:	10/22/2016	Last Revision Date:	4/11/2016
Actors:	Customer		
Description:	The customer can sign up by the first time he/she used the system by providing a username, password, address and mobile number.		
Trigger:	Signup button		
Preconditions:	Customer provides username, password, address and mobile number to sign up and click on sign up button.		
Postconditions:	Customer will be signed up to the system and now he/she will able to use the system.		
Normal Flow:	Customer	System	
	1: Customer clicks signup button to request for sign up	The system provides Customer sign-up form.	
	2: Customer fills in form by providing username, password, address and mobile number.	System signs up the Customer.	
Alternative Flows:	Customer cancels the current form.		
Exceptions:	2.The database is not responding. 2. The customer has not filled the form correctly. 1. The system is not responding.		

Table 2.5: Customer Sign in

Use Case ID:	**Uc2**		
Use Case Name:	Sign in		
Created By:	Ahsan Iftikhar	Last Updated By:	Nangyial Ahmad
Date Created:	22/10/2016	Last Revision Date:	4/11/2016
Actors:	Customer		
Description:	Customer will sign into the system by providing Username and password.		
Trigger:	Sign in button		
Preconditions:	Customer provides username, password and then click on the sign in button.		
Post conditions:	Customer will be signed in to the system.		
Normal Flow:	**Customer**	**System**	
	1: Customer will click sign in button to request for sign in	The system will provide Customer sign in form.	
	2: Customer will fill form by providing username, password.	System will allow Customer to log in to the system.	
	3: Customer will be signed in for the system.		
Alternative Flows:	Customer will cancel the current form.		
Exceptions:	2. Database is not responding. 2. Customer has not filled the form correctly. 1, 3. System is not responding.		

Table 2.6: Update Customer Profile

Use Case ID:	Uc3		
Use Case Name:	Update Customer Profile		
Created By:	Ahsan Iftikhar	Last Updated By:	Nangyial Ahmad
Date Created:	22/10/2016	Last Revision Date:	2/11/2016
Actors:	Customer		
Description:	Customer will Update his/her profile.		
Trigger:	Update profile button		
Preconditions:	Customer will have a profile. Customer will be signed in		
Post conditions:	Customer will update his/her profile.		
Normal Flow:	**Customer**	**System**	
	1: Customer will click Update profile button to request for Modifying profile.	System will provide Customer with update profile form.	
	2: Customer will fill form of update profile.	System updates profile of current customer in database.	
Alternative Flows:	Customer will cancel the current form.		
Exceptions:	2. Database is not responding. 2. Customer has not filled the form correctly. 1, 2. System is not responding.		

Table 2.7: Select food items

Use Case ID:	**Uc4**		
Use Case Name:	Select food items		
Created By:	Nangyial Ahmad	Last Updated By:	Nangyial Ahmad
Date Created:	22/10/2016	Last Revision Date:	2/11/2016
Actors:	Customer		
Description:	Customer will select different food items that are available on the website after signing in to the system		
Trigger:	Select items		
Preconditions:	Customer must be signed in. Customer must select different food categories.		
Post conditions:	Customer will select desirable food items available on the website.		
Normal Flow:	**Customer**	**System**	
	1: Customer will click Select items button to request for choosing items.	System will provide Customer Select food item form.	
	2: Customer will choose food items and submit his/her selection.	System will verify food items and process customer's request.	
Alternative Flows:	Customer will cancel the current form.		
Exceptions:	2.Database is not responding. 1, 2. System is not responding. 2.Customer have chosen out of stock item.		

Table 2.8: view bill details

Use Case ID:	**Uc5**		
Use Case Name:	view bill details		
Created By:	Nangyial Ahmad	Last Updated By:	Ahsan Iftikhar
Date Created:	22/10/2016	Last Revision Date:	4/11/2016
Actors:	Customer		
Description:	Customer will view his/her bill details after selecting items.		
Trigger:	view bill details button		
Preconditions:	Customer must be signed in. Customer must have selected some items.		
Post conditions:	Customer will view his/her bill details.		
Normal Flow:	**Customer**	**System**	
	1: Customer will click view bill details button to request for viewing his/her bill.	System will show bill details of the current items selected by the customer.	
Alternative Flows:	Customer will cancel the current form.		
Exceptions:	1. Customer has not chosen any item. 1.Database is not responding. 1. System is not responding.		

Table 2.9: view food Delivery time

Use Case ID:	**Uc6**		
Use Case Name:	View food Delivery time		
Created By:	Ahsan Iftikhar	Last Updated By:	Ahsan Iftikhar
Date Created:	22/10/2016	Last Revision Date:	4/11/2016
Actors:	Customer		
Description:	Customer will view Delivery time for the food items he/she have chosen.		
Trigger:	View Delivery time button		
Preconditions:	Customer must be signed in. Customer must have selected some items.		
Post conditions:	Customer will view Delivery time of food items.		
Normal Flow:	**Customer**	**System**	
	1: Customer will click view Delivery time button to request for viewing Delivery time.	System will display Delivery time.	
Alternative Flows:	Customer will cancel the current form.		
Exceptions:	1. Customer have not chosen any item. 1. Database is not responding. 1. System is not responding.		

Table 2.10: view ordered items

Use Case ID:	**Uc7**		
Use Case Name:	View ordered items		
Created By:	Nangyial Ahmad	Last Updated By:	Ahsan Iftikhar
Date Created:	22/10/2016	Last Revision Date:	4/11/2016
Actors:	Customer		
Description:	Customer will view ordered food items.		
Trigger:	View ordered items button		
Preconditions:	Customer must be signed in. Customer must have selected one or more items to see the details.		
Post conditions:	Customer will view Details of the items he/she desires.		
Normal Flow:	**Customer**	**System**	
	1: Customer will click view ordered items button to request for viewing ordered items.	System will display ordered items.	
Alternative Flows:	Customer will cancel the current form.		
Exceptions:	1. Database is not responding. 1. System is not responding. 1. Customer have not chosen any item.		

Table 2.11: view Delivery Boy's location

Use Case ID:	**Uc7**		
Use Case Name:	View Delivery boy's location		
Created By:	Nangyial Ahmad	Last Updated By:	Nangyial Ahmad
Date Created:	22/10/2016	Last Revision Date:	4/11/2016
Actors:	Customer		
Description:	Customer can view delivery boy's location.		
Trigger:	View location		
Preconditions:	Customer must be signed in. Customer must have selected items to place order.		
Post conditions:	Customer will view Delivery boy's location.		
Normal Flow:	**Customer**	**System**	
	1: Customer will click View location button to request for viewing delivery boy's location.	System will allow Customer to view Delivery boy location.	
	2: Customer will view Delivery Boy's location.		
Alternative Flows:	Customer will cancel the current form.		
Exceptions:	1. Database is not responding. 1. System is not responding. 1. Customer have not chosen any item. 1,2.Customer order is not dispatch yet.		

Table 3: Scan QR code

Use Case ID:	**Uc8**		
Use Case Name:	Scan QR code		
Created By:	Nangyial Ahmad	Last Updated By:	Ahsan Iftikhar
Date Created:	22/10/2016	Last Revision Date:	4/11/2016
Actors:	Customer		
Description:	Customer can scan QR code to place order through his/her mobile, that QR code will be placed on the table.		
Trigger:	Scan QR button		
Preconditions:	Customer have selected items to place order.		
Post conditions:	Customer will Scan QR code.		
Normal Flow:	**Customer**	**System**	
	1: Customer will click Scan QR button to request for Scanning QR code and placing order.	System will allow Customer to scan QR code.	
	2: Customer will Scan QR code.	Customer's order will be placed.	
Alternative Flows:	Customer will cancel the current form.		
Exceptions:	1.Customer have not chosen any item. 2. QR code is not verified by the system. 2. QR code is not correct.		

Table 2.13: reserve available table

Use Case ID:	**Uc9**		
Use Case Name:	Reserve available table		
Created By:	Muhammad Zubair	Last Updated By:	Nangyial Ahmad
Date Created:	22/10/2016	Last Revision Date:	4/11/2016
Actors:	Customer		
Description:	Customer can reserve available table.		
Trigger:	available table button		
Preconditions:	Customer must be signed in. Customers must see which tables are booked.		
Post conditions:	Customer will select/reserve available table.		
Normal Flow:	**Customer**	**System**	
	1: Customer will click available table button to request for booking table (reserve table).	System will display tables to the customer by showing red colored table as booked.	
	2: Customer will reserve available table, which are not red.	Customer's table will be reserved.	
Alternative Flows:	Customer will cancel the current form.		
Exceptions:	1, 2. Database is not responding. 1. System is not responding. 2. Customer have selected booked table.		

Table 2.14: track delivery

Use Case ID:	**Uc10**		
Use Case Name:	Modify table Seats		
Created By:	Muhammad Zubair	Last Updated By:	Nangyial Ahmad
Date Created:	22/10/2016	Last Revision Date:	4/11/2016
Actors:	Customer		
Description:	Customer can Modify his Seats that are already booked.		
Trigger:	Modify table seats button		
Preconditions:	Customer must be signed in. Customer must have selected / booked seats.		
Post conditions:	Customer will Modify table seats.		
Normal Flow:	**Customer**	**System**	
	1: Customer will click modify table seats button to request for modifying seats.	System will allow Customer to modify seats.	
	2: Customer will modify seats.	System will update information in database.	
Alternative Flows:	Customer will cancel the current form.		
Exceptions:	1, 2.Database is not responding. 1, 2.System is not responding.		

Table 4: cancel reservation

Use Case ID:	**Uc11**		
Use Case Name:	Cancel reservation		
Created By:	Muhammad Zubair	Last Updated By:	Ahsan Iftikhar
Date Created:	22/10/2016	Last Revision Date:	4/11/2016
Actors:	Customer		
Description:	Customer can cancel his/her reservation.		
Trigger:	c-reservation button		
Preconditions:	Customer must be signed in. Customer must have selected / booked seats.		
Post conditions:	Customer will cancel his/her reservation.		
Normal Flow:	**Customer**	**System**	
	1: Customer will click c-reservation button to request for cancelling reservation.	System will allow Customer to cancel the reservation.	
	2: Customer will cancel reservation.	System will cancel reservation	
Alternative Flows:	Customer will cancel the current form.		
Exceptions:	1. Database is not responding. 1, 2. System is not responding. 2. Customer have not book any reservation.		

Table 2.16: update reservation

Use Case ID:	**Uc12**		
Use Case Name:	update reservation		
Created By:	Muhammad Zubair	Last Updated By:	Ahsan Iftikhar
Date Created:	22/10/2016	Last Revision Date:	4/11/2016
Actors:	Customer		
Description:	Customer can update his/her reservation.		
Trigger:	Update reservation button		
Preconditions:	Customer must be signed in. Customer must have selected / booked seats.		
Post conditions:	Customer will update his/her reservation.		
Normal Flow:	**Customer**	**System**	
	1: Customer will click Update reservation button to request for updating reservation.	System will allow Customer to update the reservation.	
	2: Customer will update reservation.	Customer’s reservation will be updated in the system.	
Alternative Flows:	Customer will cancel the current form.		
Exceptions:	1. Database is not responding. 1, 2. System is not responding. 2. Customer have not booked any reservation.		

Table 2.17: Sign out

Use Case ID:	**Uc13**		
Use Case Name:	Sign out		
Created By:	Ahsan Iftikhar	Last Updated By:	Ahsan Iftikhar
Date Created:	22/10/2016	Last Revision Date:	4/11/2016
Actors:	Customer		
Description:	Customer will sign out of the system.		
Trigger:	Sign out button		
Preconditions:	Customer must be signed in.		
Post conditions:	Customer will be logged out of the system.		
Normal Flow:	Customer	System	
	1: Customer will click sign out button to request for log out.	System will process Customer's request and allow customer to sign out.	
	2: Customer will be signed out of the system.		
Alternative Flows:	Customer will cancel the current form.		
Exceptions:	System is not responding.		

2.5. System Sequence diagram

System sequence diagram (SSD) is a sequence diagram that shows, for a particular scenario of a use case, the events that external actors generate their order, and possible inter-system events.

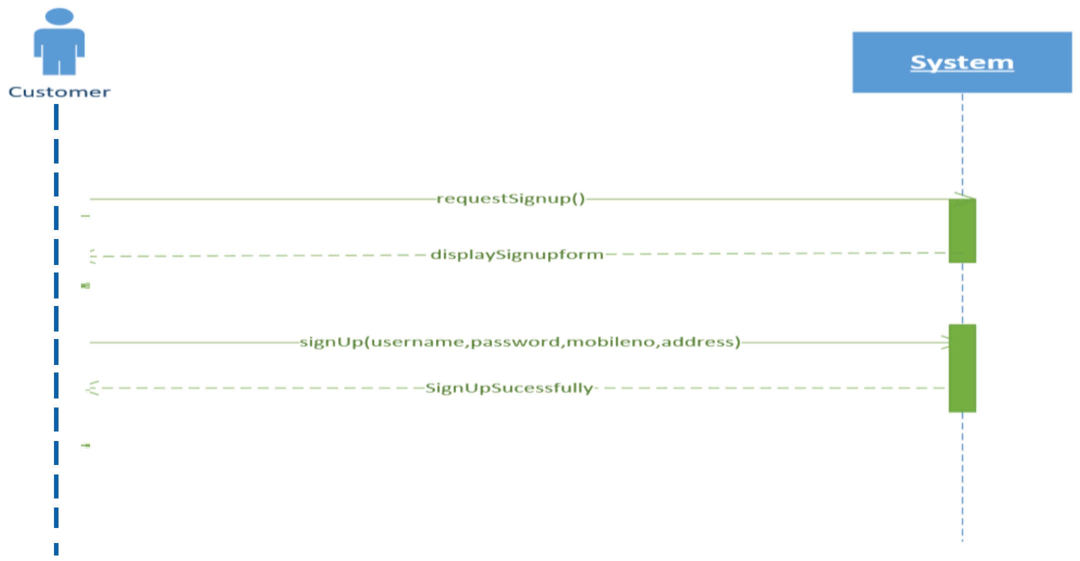
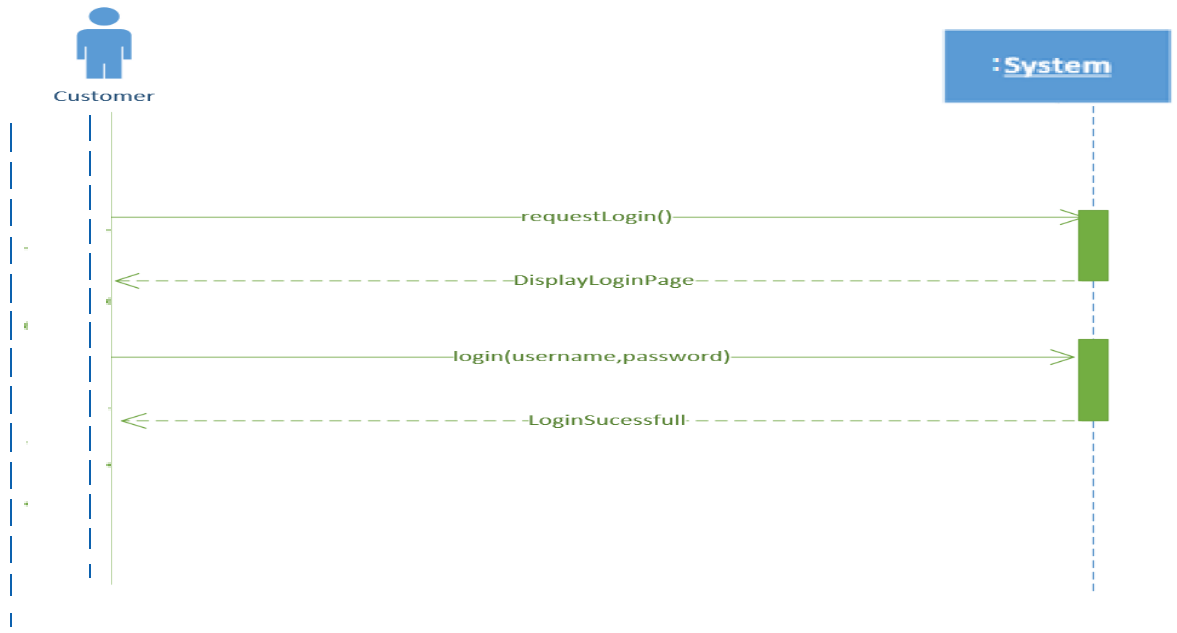


Figure 2.2: SSD (Customer Signup)

Figure 2.3: SSD (Customer login)



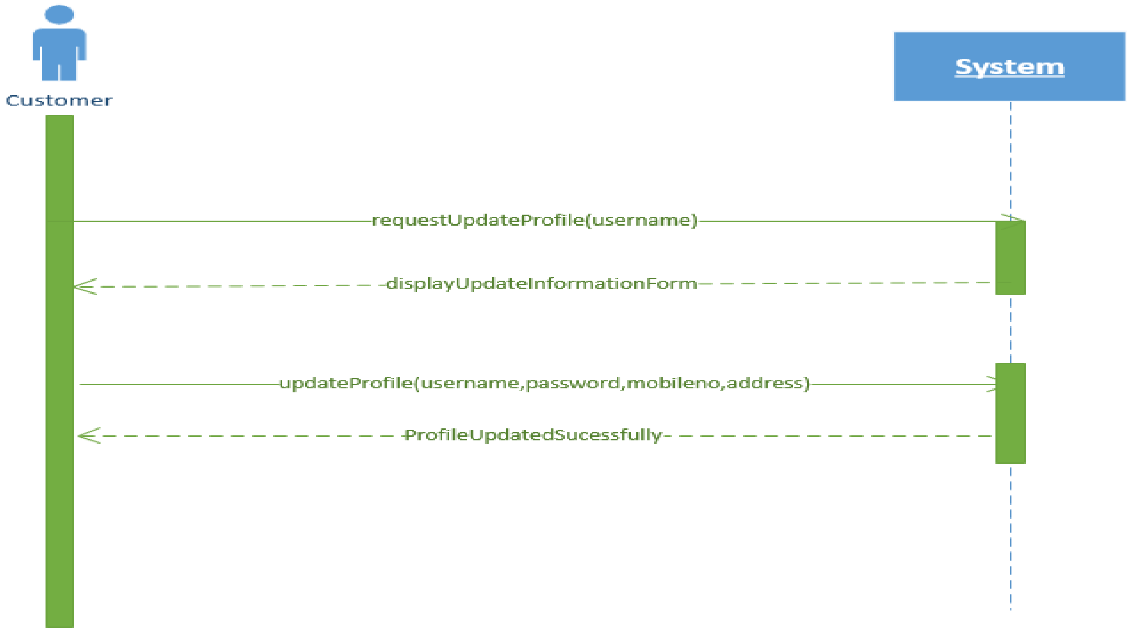
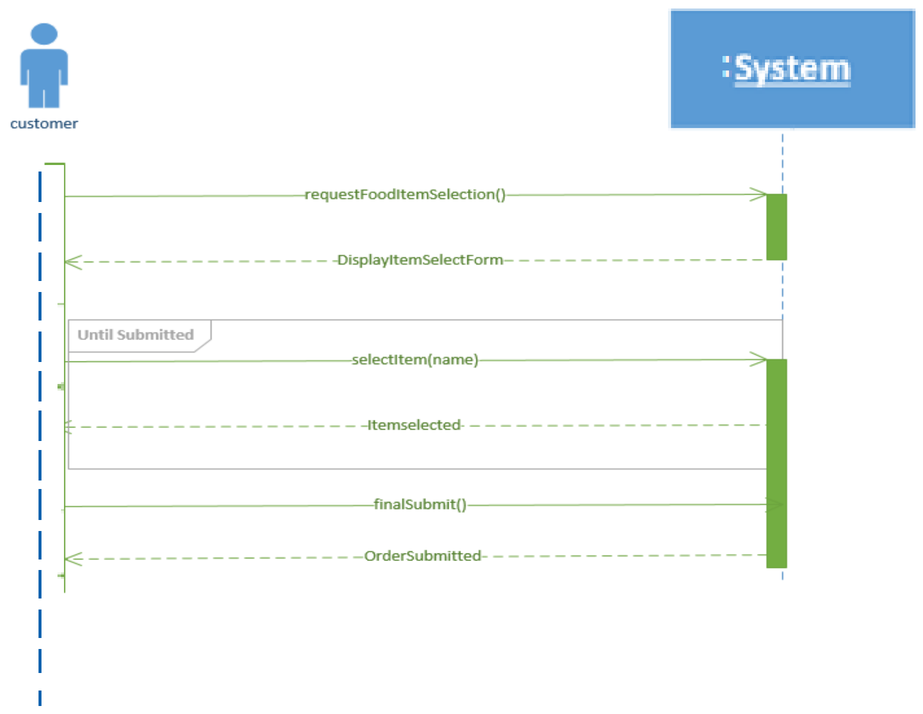

Customer
System
requestUpdateProfile(username)
displayUpdateInformationForm
updateProfile(username,password,mobileno,address)
ProfileUpdatedSucessfully

Figure 2.4: SSD (Customer Update profile)



.

Figure 2.5: SSD (Select item)

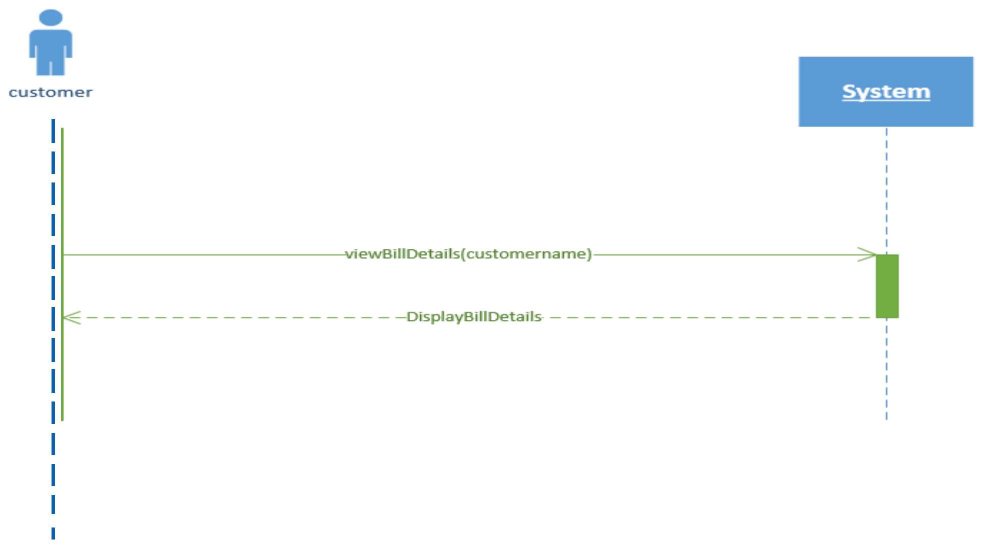


Figure 2.6: SSD (Display bill details)

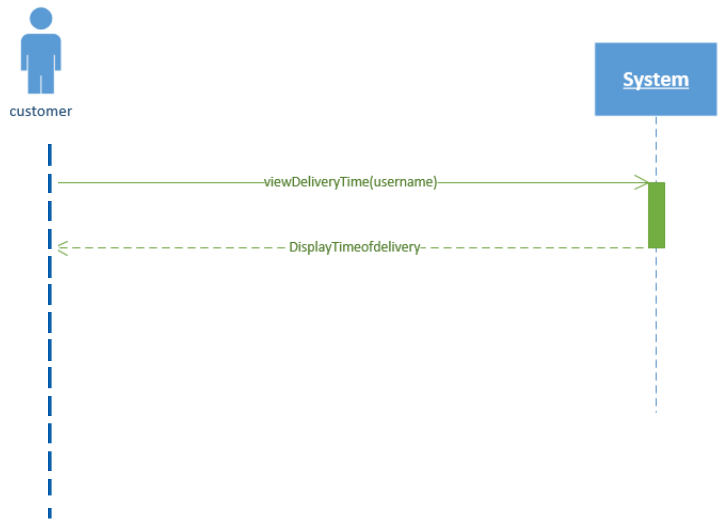


Figure 2.7: SSD (View Delivery Time)

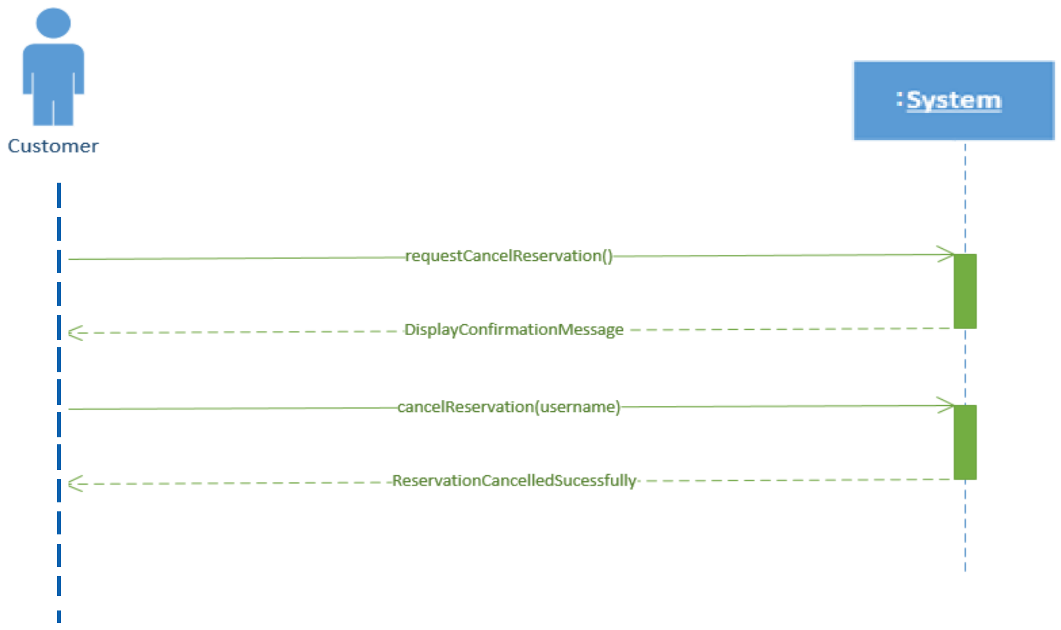

Customer
:System
requestCancelReservation()
DisplayConfirmationMessage
cancelReservation(username)
ReservationCancelledSucessfully

Figure 2.8: SSD (Cancel Reservation)

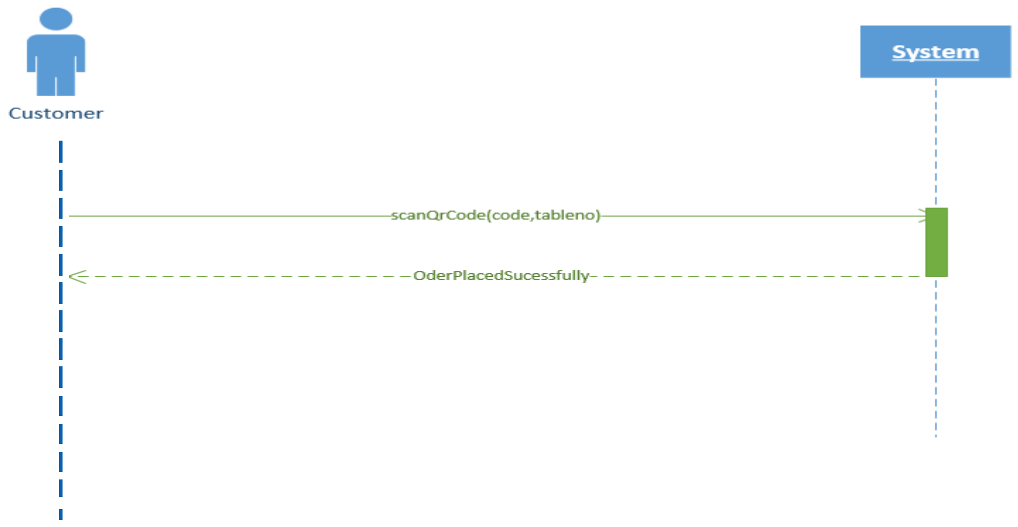


Figure 2.9: SSD (Location of Delivery Boy)

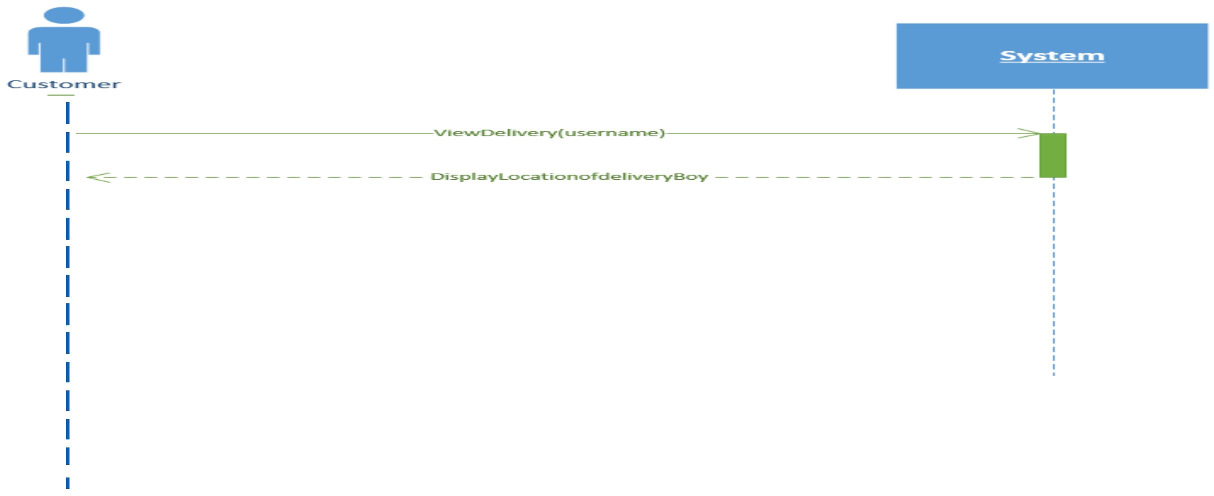


Figure 2.10: SSD (Scan QR code)

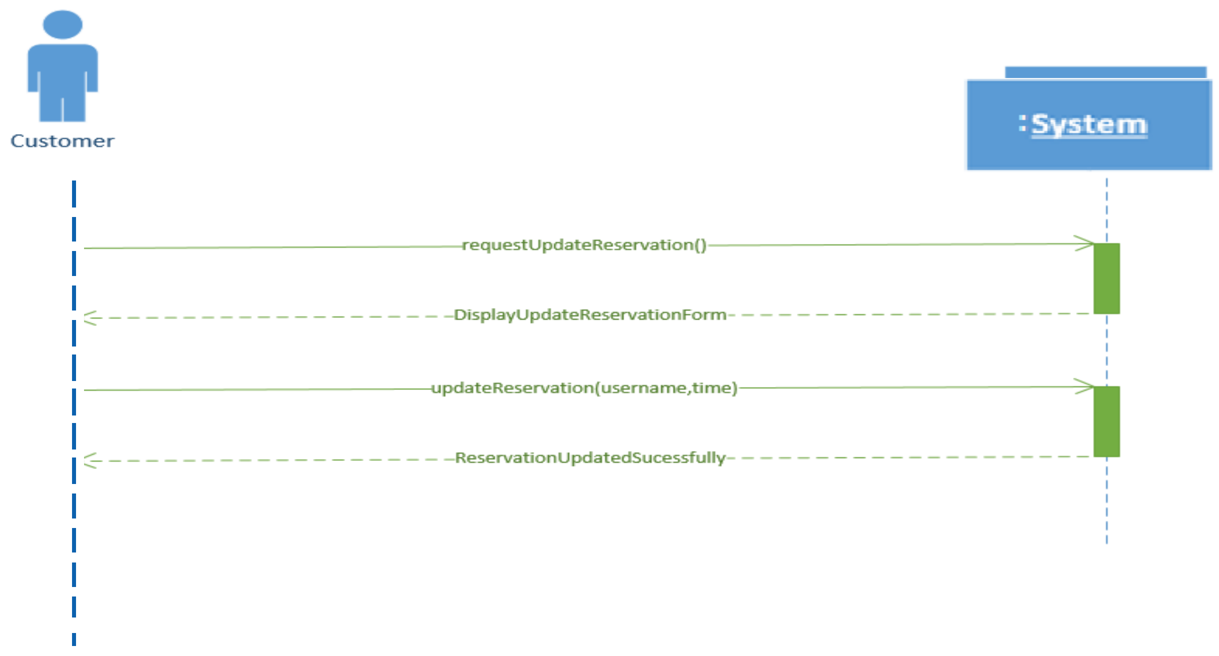

Customer
:System
requestUpdateReservation()
DisplayUpdateReservationForm
updateReservation(username,time)
ReservationUpdatedSucessfully

Figure 2.11: SSD (Update Reservation)

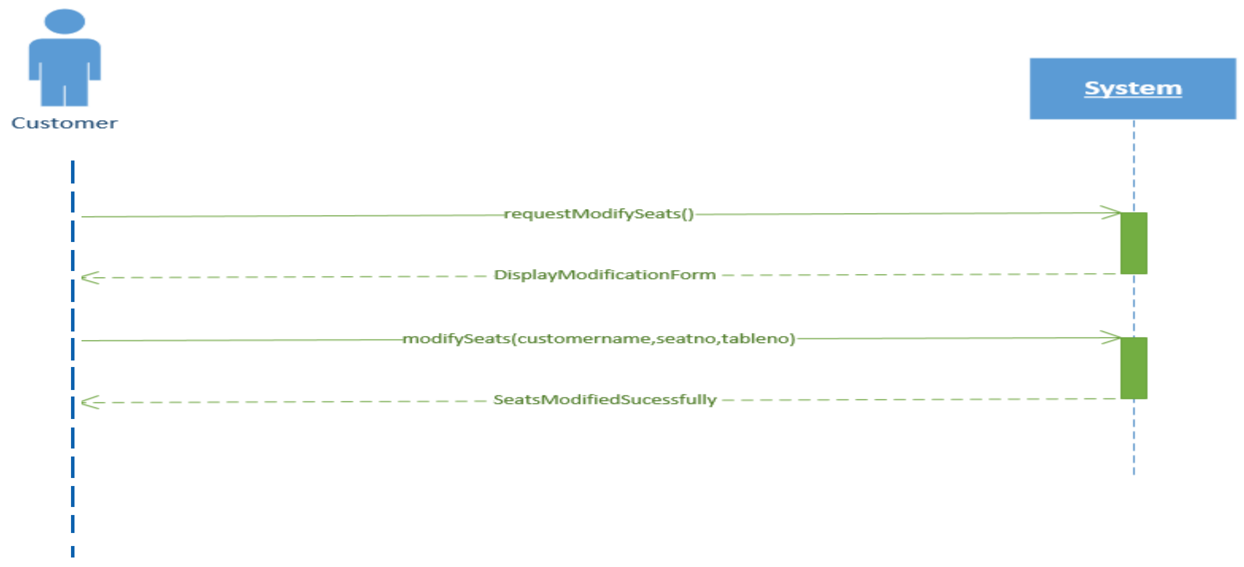


Figure 12: SSD (Modify seats)

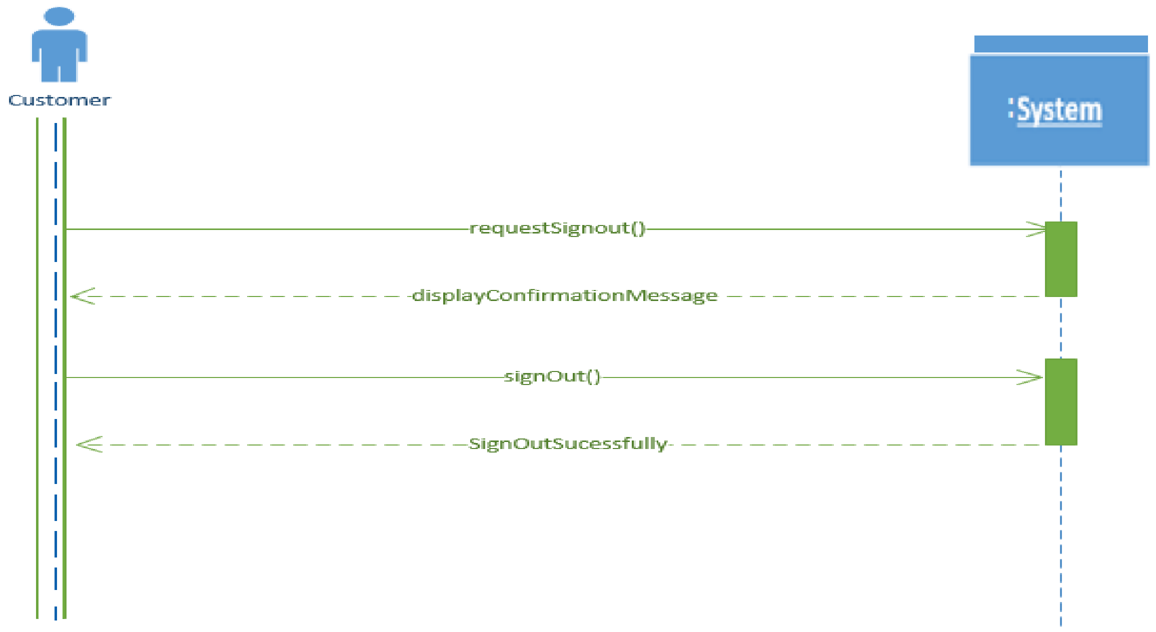


Figure 2.13: SSD (Sign out)

2.6. Domain Model

The basic concepts of the domain are customer, order, food item and dining table are shown in the following figure representing domain model.

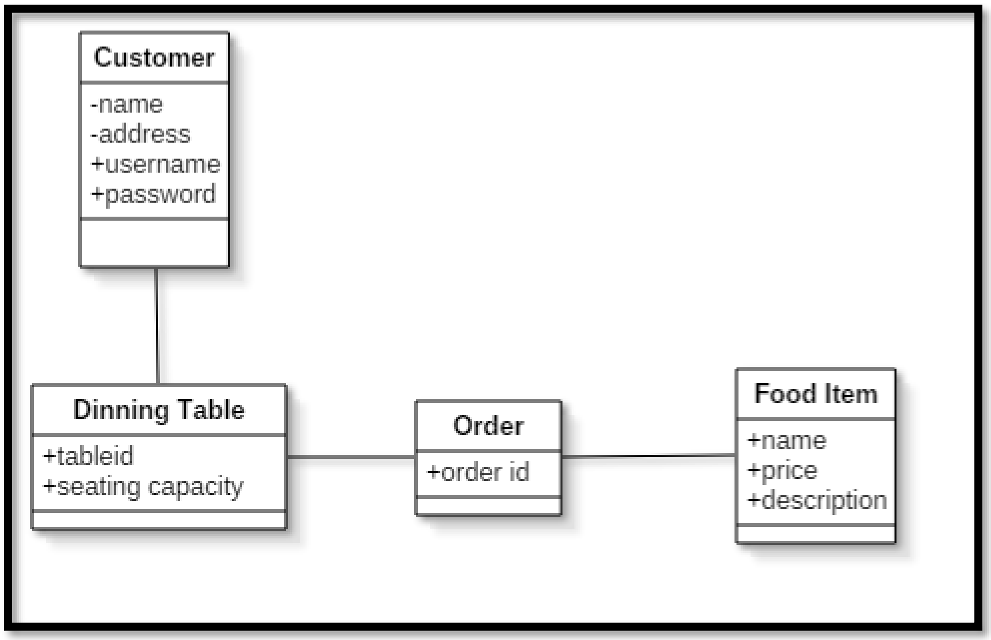


Figure 2.14: Domain Model

Chapter 3

System Design

The purpose of this chapter is to provide information that is complementary to the development phase. Without an adequate design, that delivers required function as well as quality attributes, the project will fail. However, communicating architecture to its stakeholders is as important a job as creating it in the first place.

3.1. Layer Definition

Table3.1: Layers Definition

Layers	Description
Presentation Layer	This layer will be used for the interaction with the user through a graphical user interface.
Business Logic Layer	This layer contains the business logic. All the constraints and majority of the functions reside under this layer.
Database Layer	This layer contains the database of the application being developed.

3.1.1. Presentation Layer:

Occupies the top level and displays information related to services available on a website. This tier communicates with other tiers by sending results to the browser and other tiers in the network.

3.1.2. Business Logic Layer:

Application Layer also called the middle tier, logic tier, business logic or logic tier, this tier is pulled from the presentation tier. It controls application functionality by performing detailed processing.

3.1.3. Database Layer:

Database layer includes database servers where information is stored and retrieved. Data in this tier is kept independent of application servers or business logic.

3.2. Software Architecture

Software architecture is described as the organization or structure of a system, where the system represents a collection of components that accomplish a specific function or set of functions. Below is the architecture diagram of the system:

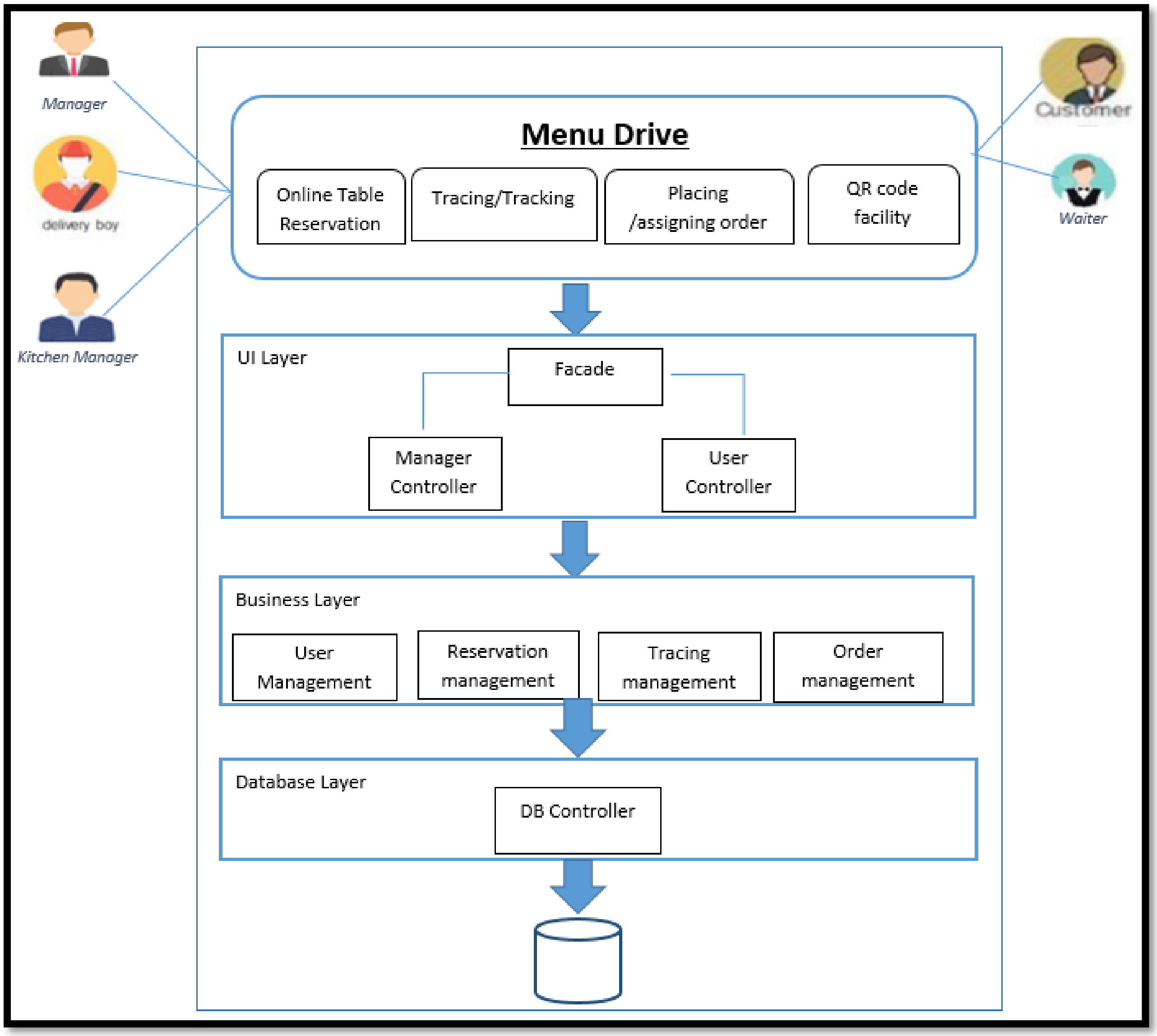


Figure 3.1: Software Architecture Diagram

3.3. Class Diagram

The class diagram describes the attributes and operations of a class and the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages:

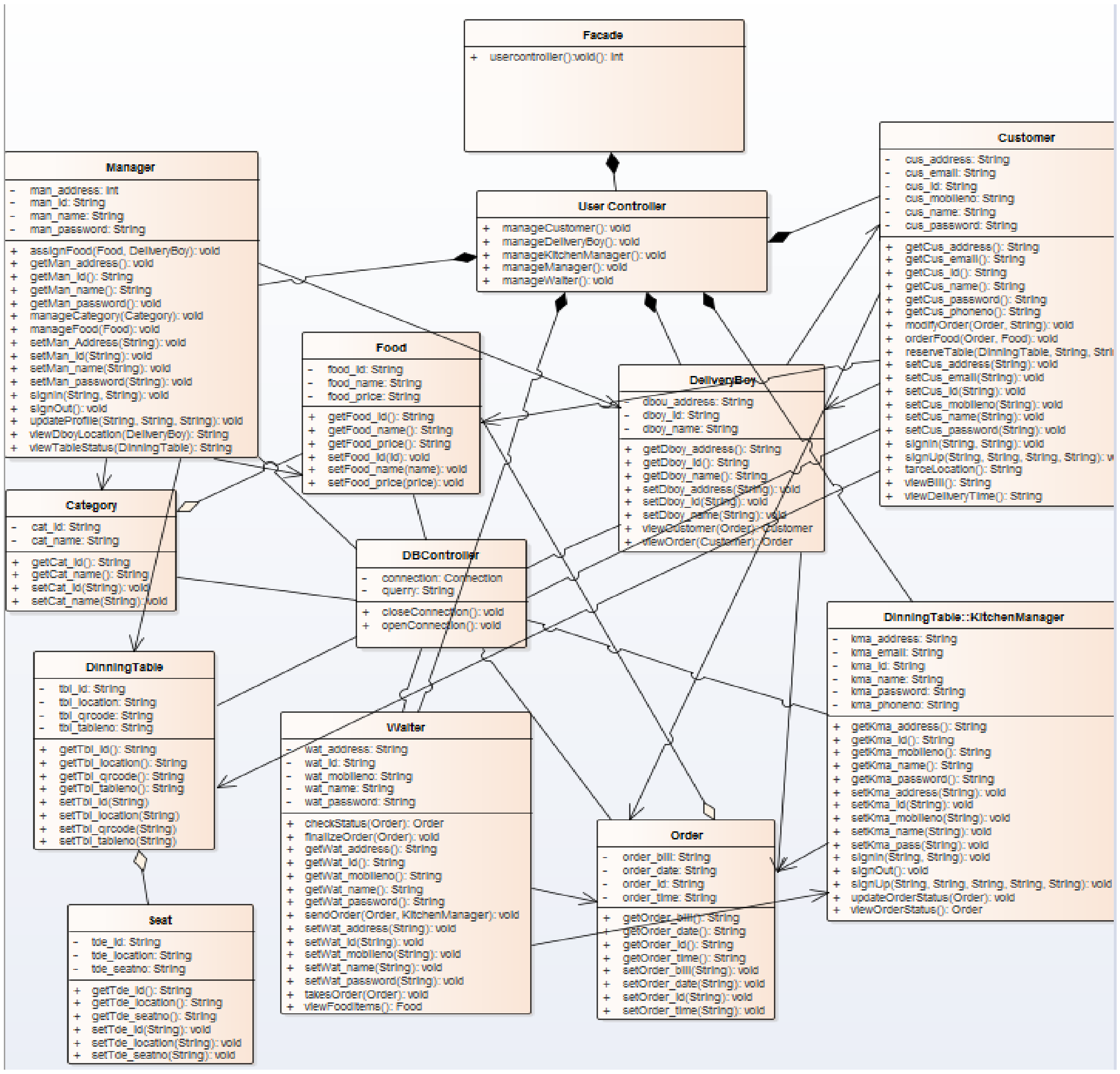

Facade
+ usercontroller():void(): int
Manager
- man_address: int
- man_id: String
- man_name: String
- man_password: String
+ assignFood(Food, DeliveryBoy): void
+ getMan_address(): void
+ getMan_id(): String
+ getMan_name(): String
+ getMan_password(): void
+ manageCategory(Category): void
+ manageFood(Food): void
+ setMan_Address(String): void
+ setMan_id(String): void
+ setMan_name(String): void
+ setMan_password(String): void
+ signin(String, String): void
+ signOut(): void
+ updateProfile(String, String, String): void
+ viewDboyLocation(DeliveryBoy): String
+ viewTableStatus(DinningTable): String
User Controller
+ manageCustomer(): void
+ manageDeliveryBoy(): void
+ manageKitchenManager(): void
+ manageManager(): void
+ manageWaiter(): void
Customer
- cus_address: String
- cus_email: String
- cus_id: String
- cus_mobileno: String
- cus_name: String
- cus_password: String
+ getCus_address(): String
+ getCus_email(): String
+ getCus_id(): String
+ getCus_name(): String
+ getCus_password(): String
+ getCus_phoneno(): String
+ modifyOrder(Order, String): void
+ orderFood(Order, Food): void
+ reserveTable(DinningTable, String, Stri
+ setCus_address(String): void
+ setCus_email(String): void
+ setCus_id(String): void
+ setCus_mobileno(String): void
+ setCus_name(String): void
+ setCus_password(String): void
+ signin(String, String): void
+ signUp(String, String, String, String): v
+ targeLocation(): String
+ viewBill(): String
+ viewDeliveryTime(): String
Food
- food_id: String
- food_name: String
- food_price: String
+ getFood_id(): String
+ getFood_name(): String
+ getFood_price(): String
+ setFood_id(id): void
+ setFood_name(name): void
+ setFood_price(price): void
DeliveryBoy
- dbou_address: String
- dboy_id: String
- dboy_name: String
+ getDboy_address(): String
+ getDboy_id(): String
+ getDboy_name(): String
+ setDboy_address(String): void
+ setDboy_id(String): void
+ setDboy_name(String): void
+ viewCustomer(Order): Customer
+ viewOrder(Customer): Order
Category
- cat_id: String
- cat_name: String
+ getCat_id(): String
+ getCat_name(): String
+ setCat_id(String): void
+ setCat_name(String): void
DBController
- connection: Connection
- querry: String
+ closeConnection(): void
+ openConnection(): void
DinningTable::KitchenManager
- kma_address: String
- kma_email: String
- kma_id: String
- kma_name: String
- kma_password: String
- kma_phoneno: String
+ getKma_address(): String
+ getKma_id(): String
+ getKma_mobileno(): String
+ getKma_name(): String
+ getKma_password(): String
+ setKma_address(String): void
+ setKma_id(String): void
+ setKma_mobileno(String): void
+ setKma_name(String): void
+ setKma_pass(String): void
+ signin(String, String): void
+ signOut(): void
+ signUp(String, String, String, String, String): void
+ updateOrderStatus(Order): void
+ viewOrderStatus(): Order
DinningTable
- tbl_id: String
- tbl_location: String
- tbl_qrcode: String
- tbl_tableno: String
+ getTbl_id(): String
+ getTbl_location(): String
+ getTbl_qrcode(): String
+ getTbl_tableno(): String
+ setTbl_id(String)
+ setTbl_location(String)
+ setTbl_qrcode(String)
+ setTbl_tableno(String)
Waiter
- wat_address: String
- wat_id: String
- wat_mobileno: String
- wat_name: String
- wat_password: String
+ checkStatus(Order): Order
+ finalizeOrder(Order): void
+ getWat_address(): String
+ getWat_id(): String
+ getWat_mobileno(): String
+ getWat_name(): String
+ getWat_password(): String
+ sendOrder(Order, KitchenManager): void
+ setWat_address(String): void
+ setWat_id(String): void
+ setWat_mobileno(String): void
+ setWat_name(String): void
+ setWat_password(String): void
+ takesOrder(Order): void
+ viewFoodItems(): Food
Order
- order_bill: String
- order_date: String
- order_id: String
- order_time: String
+ getOrder_bill(): String
+ getOrder_date(): String
+ getOrder_id(): String
+ getOrder_time(): String
+ setOrder_bill(String): void
+ setOrder_date(String): void
+ setOrder_id(String): void
+ setOrder_time(String): void
Seat
- tde_id: String
- tde_location: String
- tde_seatno: String
+ getTde_id(): String
+ getTde_location(): String
+ getTde_seatno(): String
+ setTde_id(String): void
+ setTde_location(String): void
+ setTde_seatno(String): void

Figure 3.2: UML Class Diagram

DB Controller:

In this class, data of admin and user will be stored. In DB controller; there is two attributes connection and query attribute and have two functions close connection and open connection, which will be used to open and close the connection of database. DB Controller class has association relationship with admin and user.

Facade:

In Facade class, there is two functions manage Kitchen and manage waiter, manage Customer and manage manager. Façade class has aggregation relationship with Customer, waiter, kitchen manager and manager.

Manager:

Manager have address, name, id and password in it. Manager assigns food, manages category, manages food, sign in, sign out etc.

Kitchen Manager:

Kitchen manager views food status and manages order status.

Waiter:

Waiter can check status of order and finalize order.

Delivery Boy:

Delivery boy views order details and customer details.

CUSTOMER:

Customer can sign in, sign out, select food item, update order, reserve table and update customer profile.

3.4. Sequence Diagram

Sequence Diagram model the flow of logic within your system in a visual manner enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes.

3.4.1. Customer

This Sequence diagram tells about the flow of Customer actions how he interacts with the system and how he performs general tasks e.g. signup, sign in, update profile, select food item, scan QR code etc.

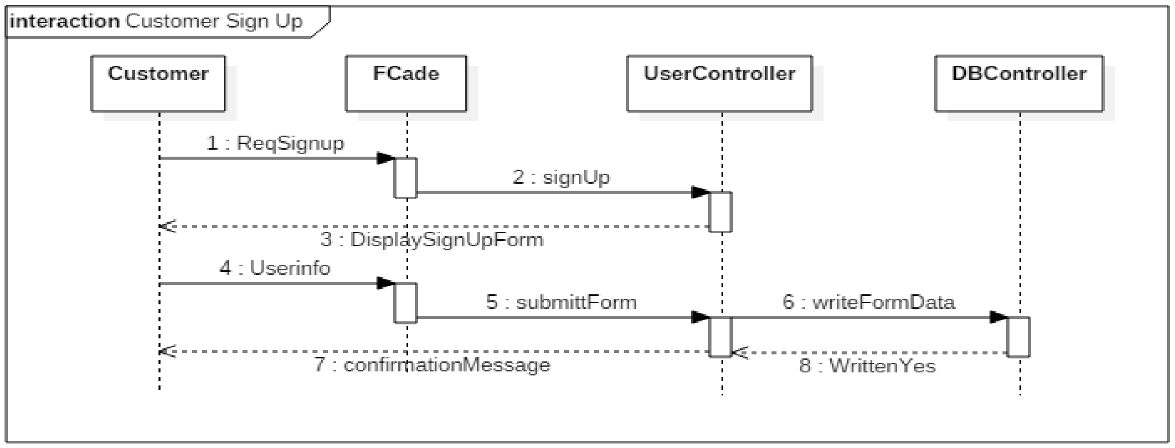


Figure 2.4: SD Customer Sign Up

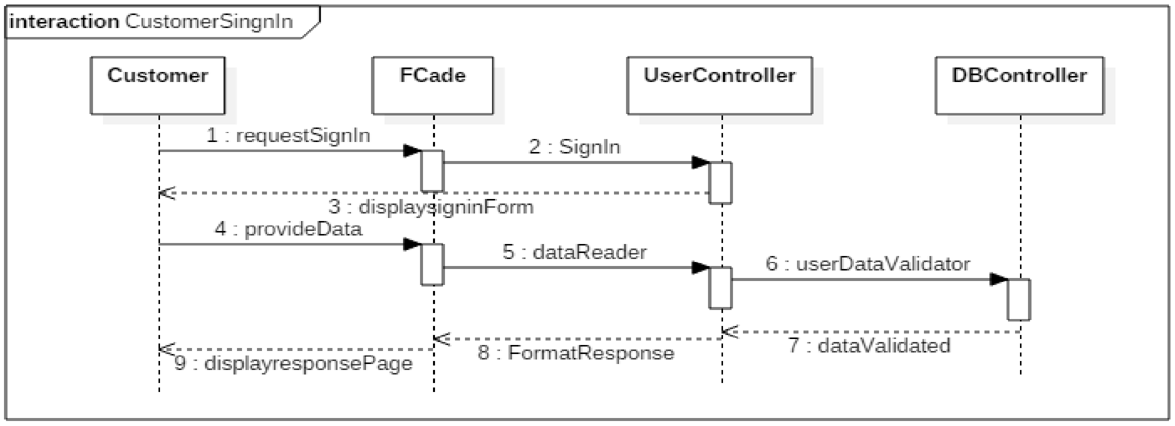


Figure 3.5: SD Customer Sign In

3.5. Entity Relationship Diagram

The entity-relationship model (or ER model) is a way of graphically representing the logical relationships of entities (or objects) in order to create a database. An entity–relationship model (ER model) describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types.

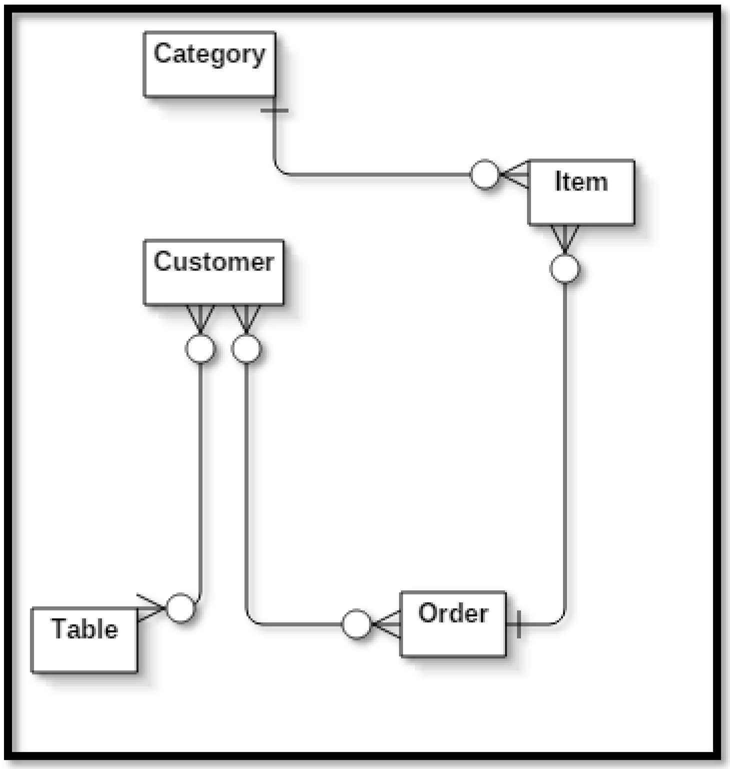


Figure 3.6: Entity Relationship Diagram

3.6. Database Schema

A database schema represents the logical configuration of all or part of a relational database. It can exist both as a visual representation and as a set of rules known as integrity constraints that govern a database. These rules are expressed in a data definition language, such as SQL.

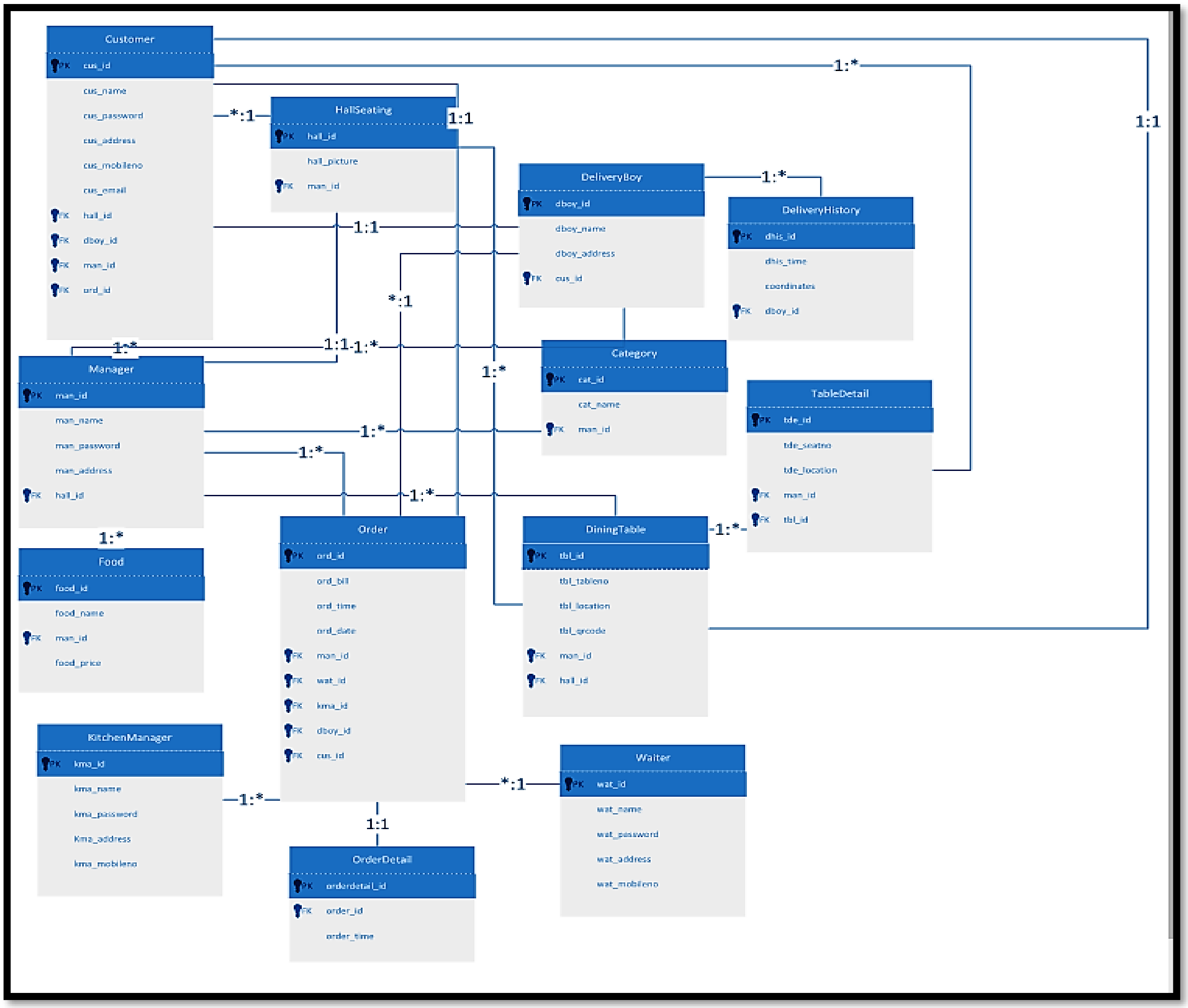


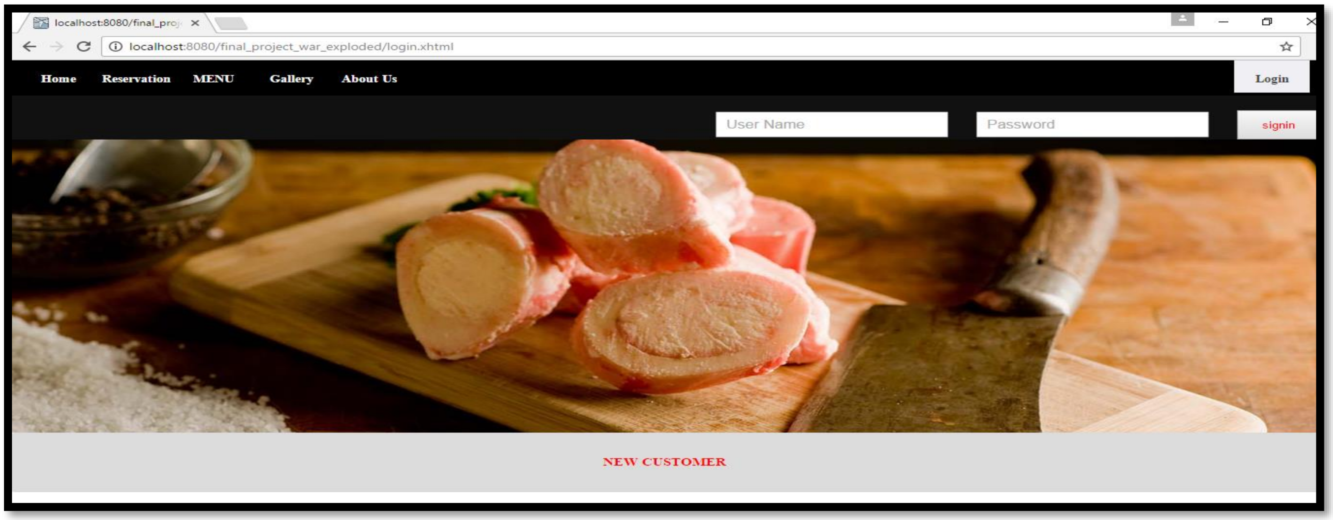
Figure 3.7: Database Schema

3.7. User Interface Design

User Interface (UI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions. UI brings together concepts from interaction design, visual design, and information architecture.

<u>User Sign in:</u>

User can order food by getting login to the system. This is the login screen for the user in which user enter username and password. User have to enter his/her unique username in "username" column. In "password" field user, enter his/her password of length 20 or less than 20.

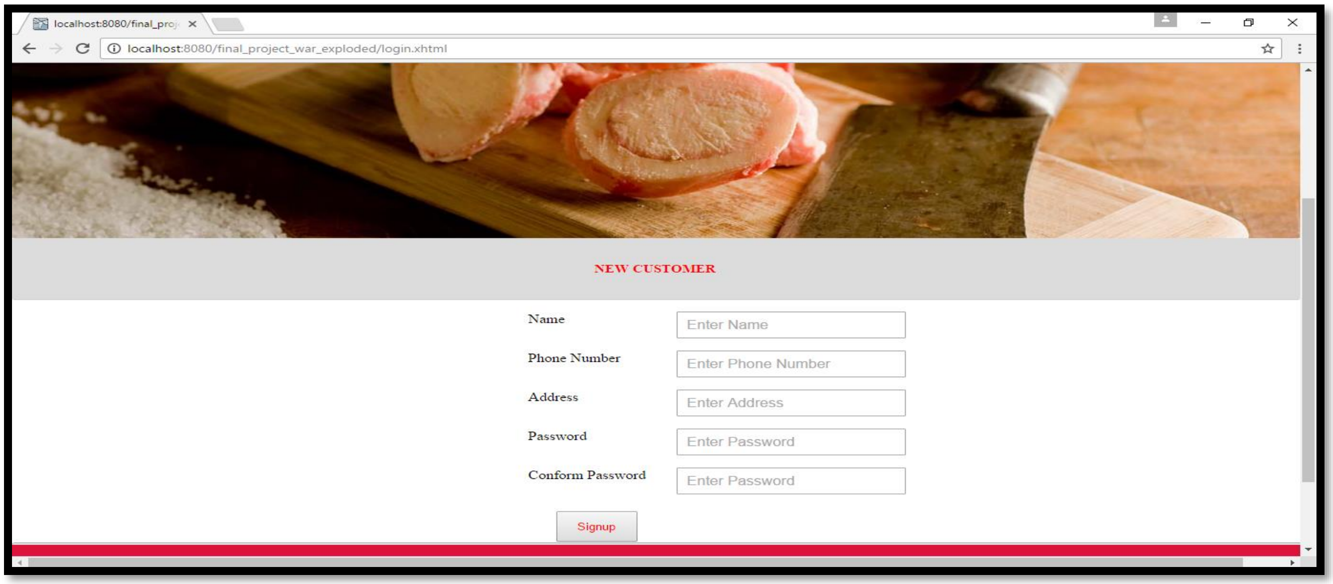


localhost:8080/final_proj
localhost:8080/final_project_war_exploded/login.xhtml
Home
Reservation
MENU
Gallery
About Us
Login
User Name
Password
signin
NEW CUSTOMER

Figure 3.8: Sign in GUI

<u>Signup:</u>

For first time user have to create their account, so this is the signup page in which user first enter their detail name, phone number, address, password. In name column user have to enter his/her name which contain of first and last name. In phone column user, enter their mobile of length 11 0r less than 11 digit.

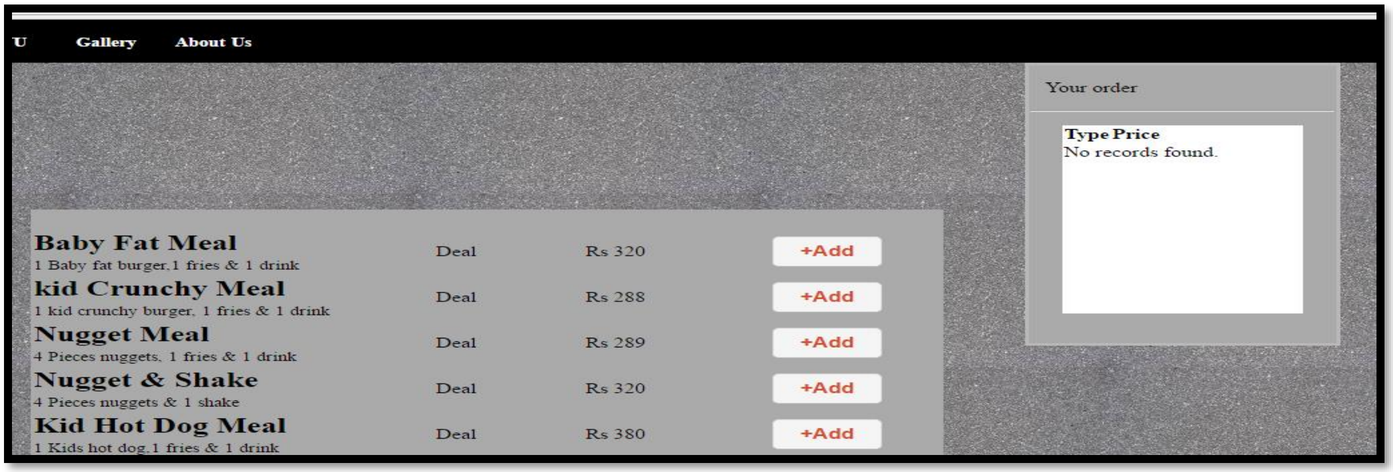


localhost:8080/final_proj
localhost:8080/final_project_war_exploded/login.xhtml
NEW CUSTOMER
Signup

Name	Enter Name
Phone Number	Enter Phone Number
Address	Enter Address
Password	Enter Password
Conform Password	Enter Password

Figure 3.9:Signup GUI

After signing to system user can order food from this page in which he/she select their food item which detail are show in order list and total bill also. Food can be selected by click on “Add” button food can be added to order list.



Type Price
No records found.

Baby Fat Meal 1 Baby fat burger,1 fries & 1 drink	Deal	Rs 320	+Add
kid Crunchy Meal 1 kid crunchy burger, 1 fries & 1 drink	Deal	Rs 288	+Add
Nugget Meal 4 Pieces nuggets, 1 fries & 1 drink	Deal	Rs 289	+Add
Nugget & Shake 4 Pieces nuggets & 1 shake	Deal	Rs 320	+Add
Kid Hot Dog Meal 1 Kids hot dog,1 fries & 1 drink	Deal	Rs 380	+Add

Figure 3.10: order Placement

Chapter 4

Software Development

This chapter will provide the details about the coding standard, we adopted during implementation phase.

4.1. Coding Standards

The adopted coding standards are discussed in the following subsections.

4.1.1. Indentation

Four spaces are used as the unit of indentation. The indentation pattern should be consistently followed throughout.

4.1.2. Declaration

One declaration per line is used to enhances the clarity of code. The order and position of declaration is as follows:

- First the static/class variables is placed in the sequence: First public class variables, protected,
- package/default level i.e. with no access modifier and then the private. As far as possible static or class fields are explicitly instantiated.
- Instance variables are placed in the sequence: First public instance variables, protected,
- package level with no access modifier and then private.
- Next the class constructors are declared.
- Class methods are grouped by functionality rather than by scope or accessibility to make reading and understanding the code easier.
- Declarations for local variables are only at the beginning of blocks e.g. at the beginning of a try/catch construct.

4.1.3. Statement Standards

Each line contains at most one statement. While compound statements are statements that contain lists of statements enclosed in braces. The enclosed statements are indented one more level than the compound statement. The opening brace at the end of the line that begins the compound statement. The closing brace to begin a line and be indented to the beginning of the compound statement. Braces are used around all statements, even single statements, when they are part of a control structure, such as a if-else or for statement. A Boolean expression / function is compared to a Boolean constants.

4.1.4. Naming Convention

Naming conventions make programs more understandable by making them easier to read. Following conventions are followed while naming a class or a member:

We used full English descriptors that accurately describe the variable, method or class. For example, use of names like totalSales, currentDate instead of names like x1, y1, or fn.

Terminology applicable to the domain is used. Implying that if user refers to clients as customers, then the term Customer is used for the class, not Client.

Mixed case is used to make names readable with lower case letters in general capitalizing the first letter of class names and interface names.

4.2. Development Environment

Android Studio is the official integrated development environment (IDE) for the Android platform. It was announced on May 16, 2013 at the Google I/O conference. Android Studio is freely available under the Apache License 2.0.

The reason for using android studio was that it provides a very interactive and easy to understand interface to work with android devices. In this tool, User can test the written code on android device and that results in better outcomes. Different services were made by us related our final year project in android studio that are related to tracking

Which calculates latitudes and longitudes and provide location and there are two apps one for delivery boy and one for waiter, which are developed in this tool.

Few alternatives for android studio are AIDE (Android IDE), Application Craft, Basic4Android and Cordova.

Intellij Idea:

IntelliJ IDEA is a Java integrated development environment (IDE) for developing computer software. It is developed by Jet Brains (formerly known as IntelliJ), and is available as an Apache 2 Licensed community edition, and in a proprietary commercial edition.

4.3. Database management System

PostgreSQL:

PostgreSQL (pronounced "post-gresQL") is an open source relational database management system (DBMS) developed by a worldwide team of volunteers. PostgreSQL is not controlled by any corporation or other private entity and the source code is available free of charge. This is a database tool also known as pg admin which we used in our project that stores information and data coming from different java server faces. It is a sql database in which we have tested our sql queries. Alternatives for PostgreSQL are SQLite, MySQL Community edition, MongoDB and Microsoft sql server.

4.4. Software Description

Main modules of our project are

- Order delivery management module.
- Online table management module.
- Online order management module.
- Trace and track management module.
- Hall management module.

Input:

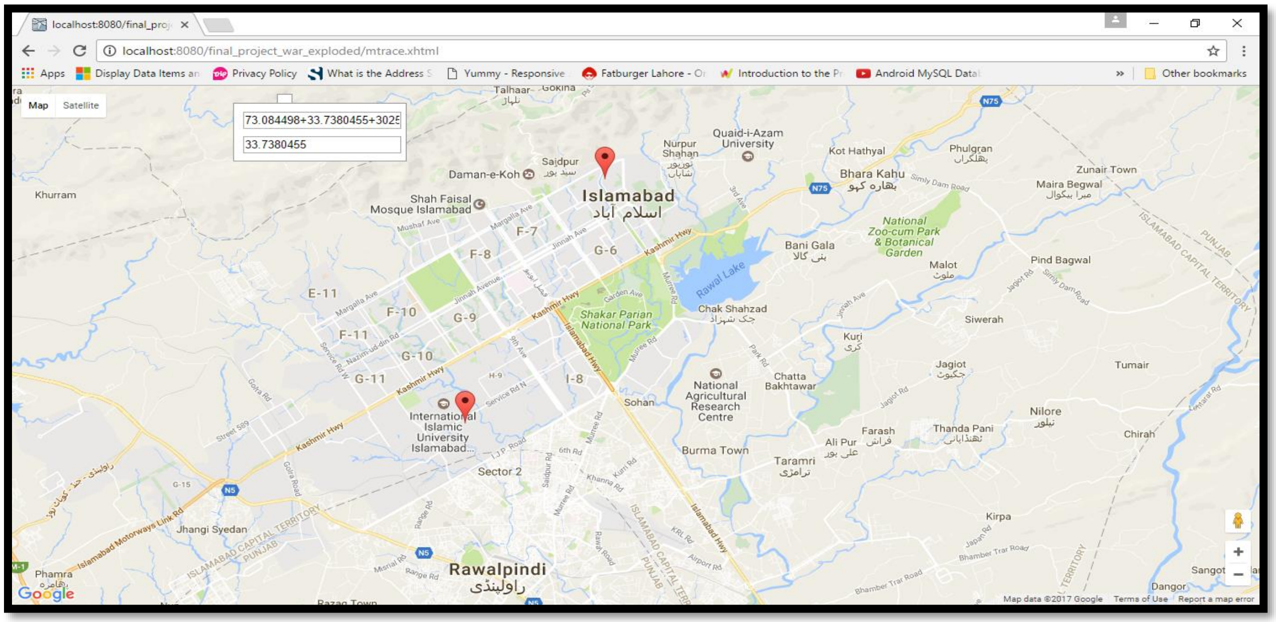
In this module, location of the given coordinates will be found. User will first open the website and he/she will be asked to enter the longitudes and the latitudes for the desired location. User will input or enter the longitude and latitude of the desired location.

Output:

In the output Location on the map will be discovered and location is shown for the given latitudes and longitudes.

```
longitude = gps.getLongitude();
latitude = gps.getLatitude();
Toast.makeText(context, "Longitude:" + Double.toString(longitude) + "\nLatitude:" +
Double.toString(latitude)+ "\nphone:" + MobileNumber.mob, Toast.LENGTH_SHORT).show();
Background task = new Background();
task.execute();

private Location getLocation() {

try {
locationManager = (LocationManager) mContext
.getSystemService(LOCATION_SERVICE);

// getting GPS status
checkGPS = locationManager
.isProviderEnabled(LocationManager.GPS_PROVIDER);

// getting network status
checkNetwork = locationManager
.isProviderEnabled(LocationManager.NETWORK_PROVIDER);

if (!checkGPS && !checkNetwork) {
            Toast.makeText(mContext, "No Service Provider Available", Toast.LENGTH_SHORT).show();

        } else {
this.canGetLocation = true;
// First get location from Network Provider
if (checkNetwork) {
```

```
try {
locationManager.requestLocationUpdates(
                            LocationManager.NETWORK_PROVIDER,
MIN_TIME_BW_UPDATES,
MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
                    Log.d("Network", "Network");
if (locationManager != null) {
loc = locationManager
.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);

                    }

if (loc != null) {
latitude = loc.getLatitude();
longitude = loc.getLongitude();
                    }
                } catch (SecurityException e) {

                }
            }
        }
// if GPS Enabled get lat/long using GPS Services
if (checkGPS) {

if (loc == null) {
try {
locationManager.requestLocationUpdates(
                                LocationManager.GPS_PROVIDER,
MIN_TIME_BW_UPDATES,
MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
                        Log.d("GPS Enabled", "GPS Enabled");
if (locationManager != null) {
loc = locationManager
.getLastKnownLocation(LocationManager.GPS_PROVIDER);
if (loc != null) {
latitude = loc.getLatitude();
longitude = loc.getLongitude();

                            }
                        }
                    } catch (SecurityException e) {


                }
            }
        }

    } catch (Exception e) {
        e.printStackTrace();
    }

return loc;
}
public String getallonlinedata(){
    System.out.println("1");
    String datainarry="";
try {
        String json ="https://finalorder.000webhostapp.com/getdataall.php";
        URL url = new URL(json);
        System.out.println("2");
        HttpsURLConnection httpURLConnection = (HttpsURLConnection)url.openConnection();
        httpURLConnection.setRequestMethod("POST");
        httpURLConnection.setDoOutput(true);
        InputStream inputStream = httpURLConnection.getInputStream();
        System.out.println("3");
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream));
        System.out.println("4");
        StringBuilder stringBuilder = new StringBuilder();
        System.out.println("5");
while ((data= bufferedReader.readLine())!=null)
        {
            System.out.println("data"+data);
data = data.replaceAll("[^.?0-9]+", " ");
            System.out.println(Arrays.asList(data.trim().split(" ")));
            String location[] = data.split(" ");
for(int i=1; i<location.length;i++){
                {if(datainarry==""){
                    datainarry=location[i];
                }else {
                    datainarry=datainarry+"+"+location[i];}}

            }
            System.out.println("longitude" + location[1]);
            System.out.println("latitude" + location[2]);
obj.setLongi(datainarry);
obj.setLatit(location[2]);
        }bufferedReader.close();
        inputStream.close();
        httpURLConnection.disconnect();
return data; }catch (Exception e)
    {System.out.print(e);
    }return data;}
```



73.084498+33.7380455+3025

33.7380455

Figure 4.1 trace and track management module

In this function through longitude and latitude, location of the mobile will be traced.

By using the GPS of the mobile phone, the location of the mobile phone will be calculated. First location manager will access location service after that GPS status of the mobile phone will be checked whether it is activated or not activated Than network status will be checked means whether internet is activated or not. If either of these are disconnected than No Service Provider available message will be shown else it will first get location from network provider by requesting location updates if location manager is not null than get last known location and if location is not null than get latitude and longitude of location. GPS will be checked if GPS is available than latitude and longitude will be accessed using GPS service.

In this code data about location i.e. latitude and longitude will be accessed from database in an array from the link that is given in above code. First A URL object is created than connection is opened through URL object by doing url.openconnection () after that object of Input Stream will be created and of BufferedReader too. After by using a for loop location data will be saved in an array and latitude and longitudes will be displayed.

Online Order Management Module

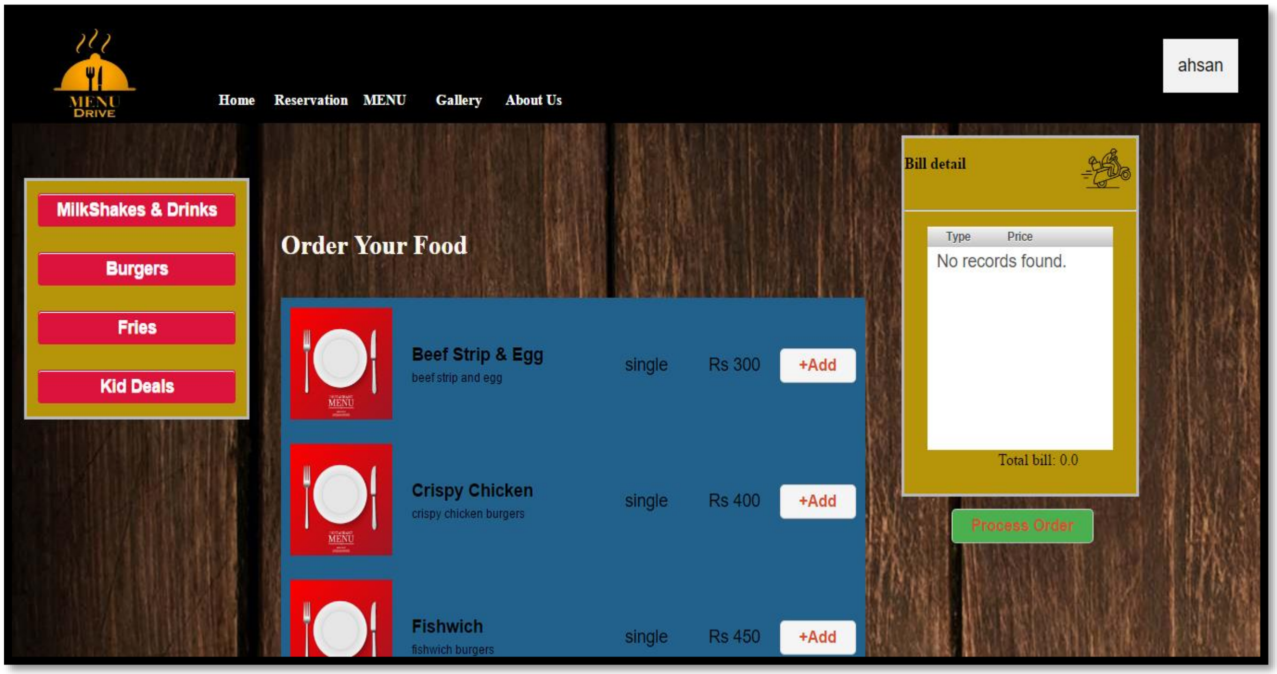
Input:

Food will be shown on the website after login and user can add the food or order by clicking the add button and that order will be shown in bill details and from there bill will be calculated for the customer.

Output:

Order will be finalized and saved in the database.

```
public void finalize_order() throws SQLException {
    Statement stmt=null;
    stmt = c.createStatement();
int customerid= Integer.parseInt(getLoggedUser());
    String status="uncooked";
    String Dstatus="no";
    String deliveryboy="no";
int searchListLength = lis.size();
for (int i = 0; i < searchListLength; i++) {
int foodid =lis.get(i).getId();
int foodquantity =lis.get(i).getFoodquantity();
        String sql = "INSERT INTO
online_order_detail(on_cus_id,on_food_id,on_food_quantity,on_or_status,deliveryboy,delivered_stat
us)  VALUES
('"+customerid+"','"+foodid+"','"+foodquantity+"','"+status+"','"+deliveryboy+"','"+Dstatus+"')";
        stmt.executeUpdate(sql);
    }

lis.clear();
totalbill=0;
}
```



Type	Price
No records found.	

Total bill: 0.0

Beef Strip & Egg beef strip and egg	single	Rs 300	+Add
Crispy Chicken crispy chicken burgers	single	Rs 400	+Add
Fishwich fishwich burgers	single	Rs 450	+Add

Figure 4.2: order management module

In this code function of finalize order will deal with when user finalizes the order. Customerid a variable will be created which will get current logged user id than status of that user's order will be set to uncooked and status of delivery and delivery boy to no. after that by checking the list length size the insert query will be executed which will save the user's order information in the database.

Order delivery management module

Input:

User will input phone number and then he/she can trace order's location or get order detail.

Output:

Order’s location will be traced or order details will be shown.

```
{"server_response":[
<%
String pno="no";

        pno= request.getParameter("Phone");
// String pno="03025155721";
Connection c = null;
try {
      Class.forName("org.postgresql.Driver");

    c = DriverManager.getConnection("jdbc:postgresql://postgres127595-env-
5696204.jelastic.regruhosting.ru/postgres", "webadmin", "YIRlbv15558");


} catch (Exception e) {
      e.printStackTrace();
      System.err.println(e.getClass().getName()+": "+e.getMessage());
      System.exit(0);
   }
   System.out.println("Opened database successfully");
   Statement stmt=null;
   stmt = c.createStatement();

   Statement stmt2=null;
   stmt2 = c.createStatement();
   JSONObject obj;

   String sql="select on_cus_id,on_or_status,deliveryboy from online_order_detail WHERE on_or_status = 'cooked'
AND deliveryboy='"+pno+"'  GROUP BY on_cus_id,on_or_status,deliveryboy " ;
   ResultSet rs=stmt.executeQuery(sql);
while (rs.next()) {
      String cid=rs.getString("on_cus_id");
      obj=new JSONObject();
      obj.put("CID",cid);

      String sql2="SELECT * FROM customerdetail WHERE id='"+cid+"' ";
      ResultSet rs2=stmt2.executeQuery(sql2);
      rs2.next();

      obj.put("CName",rs2.getString("cus_name"));
      obj.put("CAddress",rs2.getString("cus_address"));
      obj.put("CPhone",rs2.getString("cus_phone"));
      out.print(obj+",");}%>]}
```

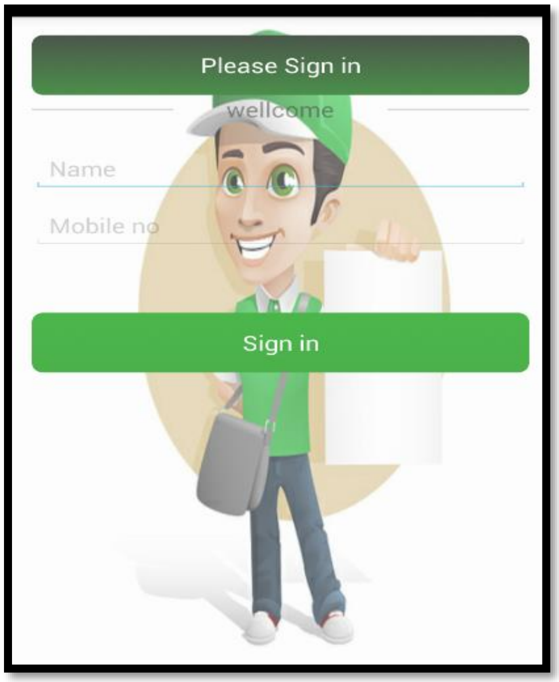


Figure 4.3: Order delivery management module

Online Table management module

Input:

User can reserve the table by clicking the Add table button and then by selecting the table. Reservation details will be shown in view Reservation details. Table can also be added and deleted.

Output:

Table will be reserved and shown by a red square.

```
public void reserveTable() {

    try {

        Connection con = null;
```

```
            Class.forName("org.postgresql.Driver");

            con = DriverManager.getConnection("jdbc:postgresql://localhost:5432/fyp", "postgres", "a");

            Statement statement = con.createStatement();

            String q = "insert into
reservations(cname,phno,seats,tableid,datee,stime,etime)values('"+getUsername()+"','"+getPhoneno()+"','"+getSeats
()+"','"+getId()+"','"+getDate()+"','"+getStarting_time()+"','"+getEnding_time()+"')";



           statement.executeUpdate(q);

        } catch (Exception ex) {

        }

    }
```

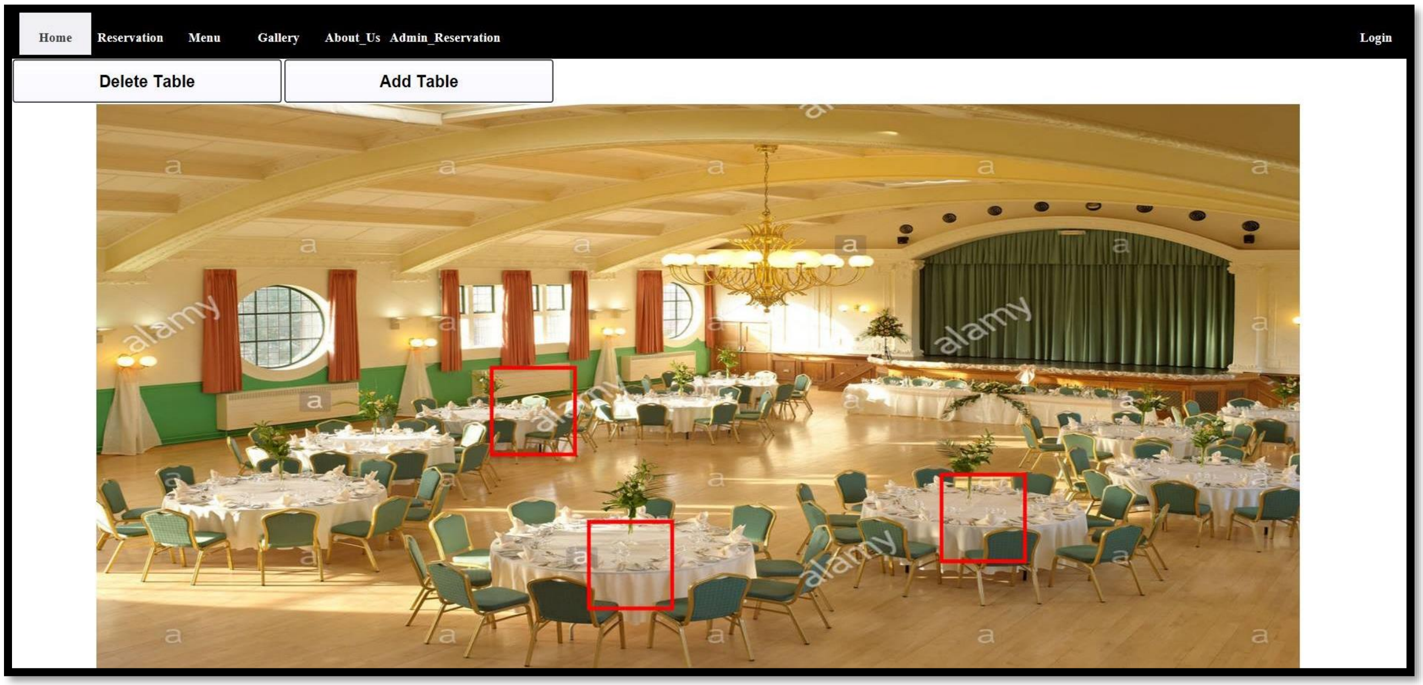


Figure 4.4: online table management module

In the above code, the user can reserve table. When the user reserves the table than through insert database query his/her name, phone number, required seats, id of the table, date and time of reservation and start time and end time will be saved in the database. And user's order will be booked.

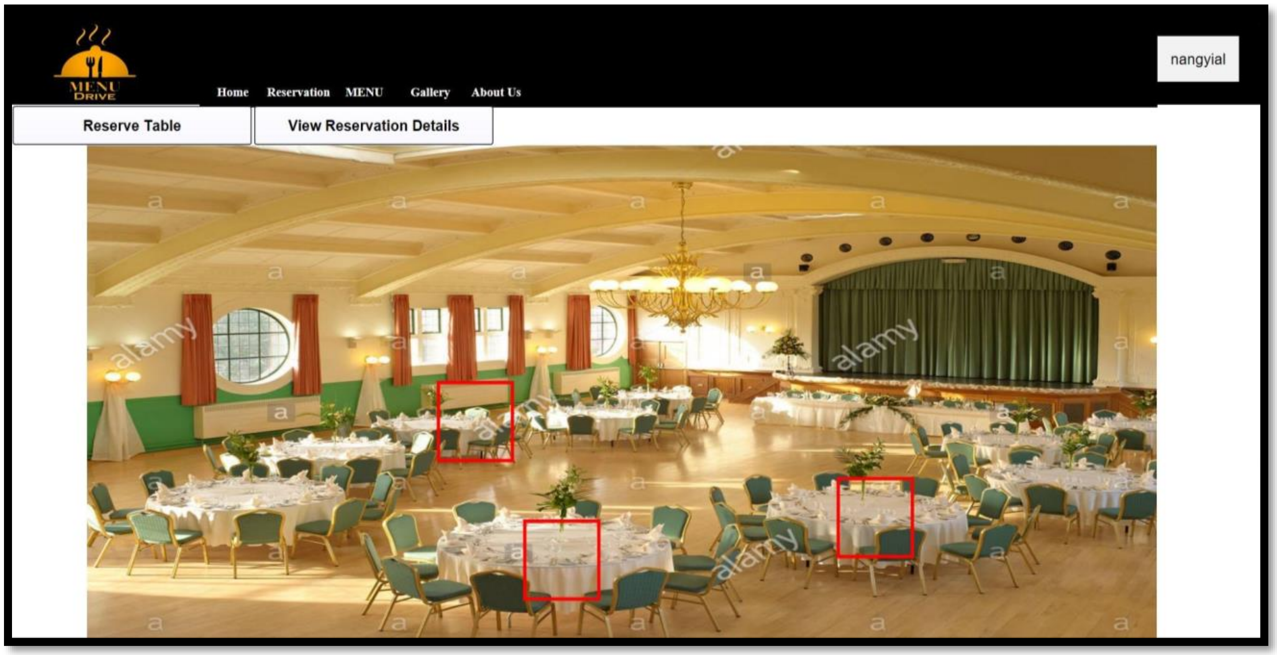


Figure 4.5:3 online table management module

Online Table Management Module

Input:

User will select the image by clicking the browse button and then after submitting that image will be saved in the database and shown on the website for reserving table.

Output:

Image will be shown to the customer on website.

```
public StreamedContent viewImage()
   {
      try {
         Connection con;
         Class.forName("org.postgresql.Driver");
```

```
        con = DriverManager.getConnection("jdbc:postgresql://localhost:5432/fyp", "postgres",
"a");

        Statement statement = con.createStatement();

        FileOutputStream fos;

        byte b[]=new byte[1024];

      PreparedStatement ps=con.prepareStatement("select * from upload_image");

        ResultSet rs=ps.executeQuery();

        while(rs.next()){

         b=rs.getBytes(3);

        }

        InputStream is=new ByteArrayInputStream(b);

        return new DefaultStreamedContent(is,"image/png");

      }catch (Exception ex)

      {}

return null;}
```

In the above code, first connection to the database will be established after that by using a query image saved in database will be accessed and shown on the website. User will select the image by clicking the browse button and then after submitting that image will be shown on the website for reserving table.

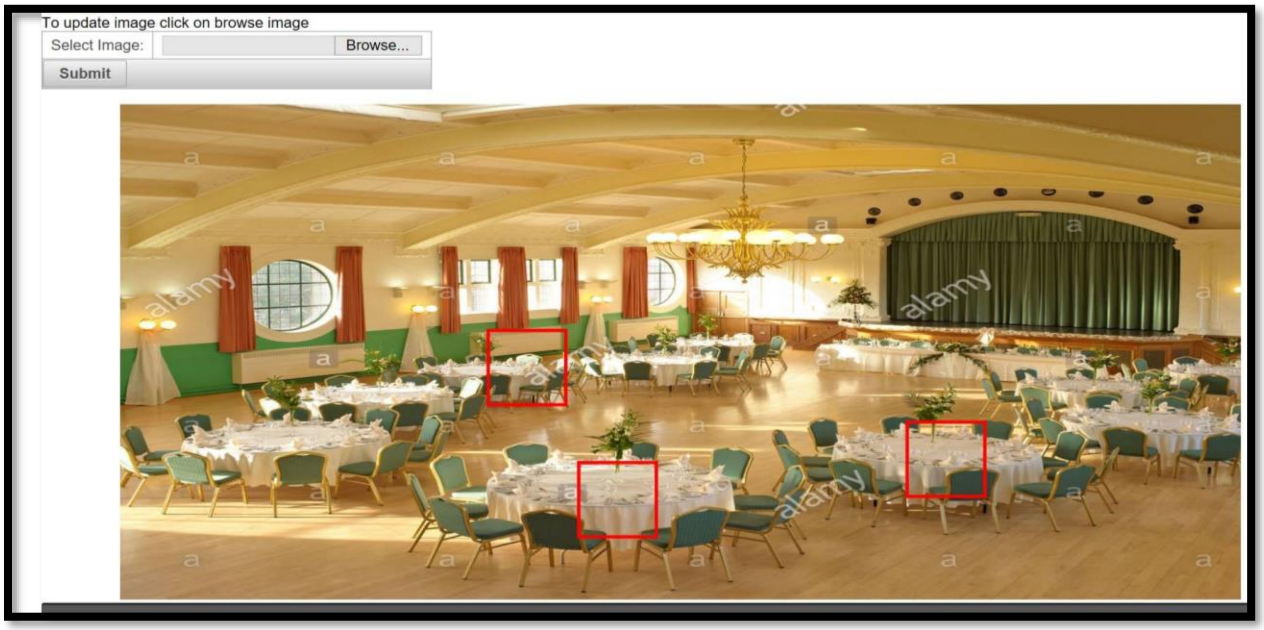


Figure 4.6:4 online table management module

Upload image to database

Input:

User will select the image by clicking the browse button and then after submitting that image will be save in the database and shown on the website for reserving table.

Output:

Image will be saved to the database and shown to the customer on website.

```
public void upload() {

    System.out.println(file.getFileName().toString());

    String type="";

     if(file.getFileName().length()<4){

        FacesContext.getCurrentInstance().addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR, "Error!",
"Please select an image."));
```

```
    }

    type=file.getFileName().toString().substring(file.getFileName().toString().length()-3,file.getFileName().toString().length());

    if(type.equals("png")||type.equals("jpg")){

      try {

        System.out.println(file.getFileName());

        InputStream fin2 = file.getInputstream();

       Connection con = null;

        System.out.println(file.getFileName().toString());

        Class.forName("org.postgresql.Driver");

        con = DriverManager.getConnection("jdbc:postgresql://localhost:5432/fyp", "postgres", "a");

        Statement statement=con.createStatement();

        deleteFile();

        statement.executeUpdate("DELETE from hallseating");

        statement.executeUpdate("DELETE from reservations");

        PreparedStatement pre = con.prepareStatement("insert into upload_image (image_name,image) values(?,?)");

        pre.setString(1, file.getFileName().toString());

        pre.setBinaryStream(2, fin2, file.getSize());

        pre.executeUpdate();

        System.out.println("Inserting Successfully!");

        pre.close();

        FacesContext.getCurrentInstance().addMessage(null, new FacesMessage(FacesMessage.SEVERITY_INFO, "Image
Inserted", "Image Inserted Sucessfully"));} catch (Exception e) {

        System.out.println("Exception-File Upload." + e.getMessage());

      }}

    else {

      System.out.println(type);

      FacesContext.getCurrentInstance().addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR, "Error!", "The
image should be in jpg or png format.")) }
```

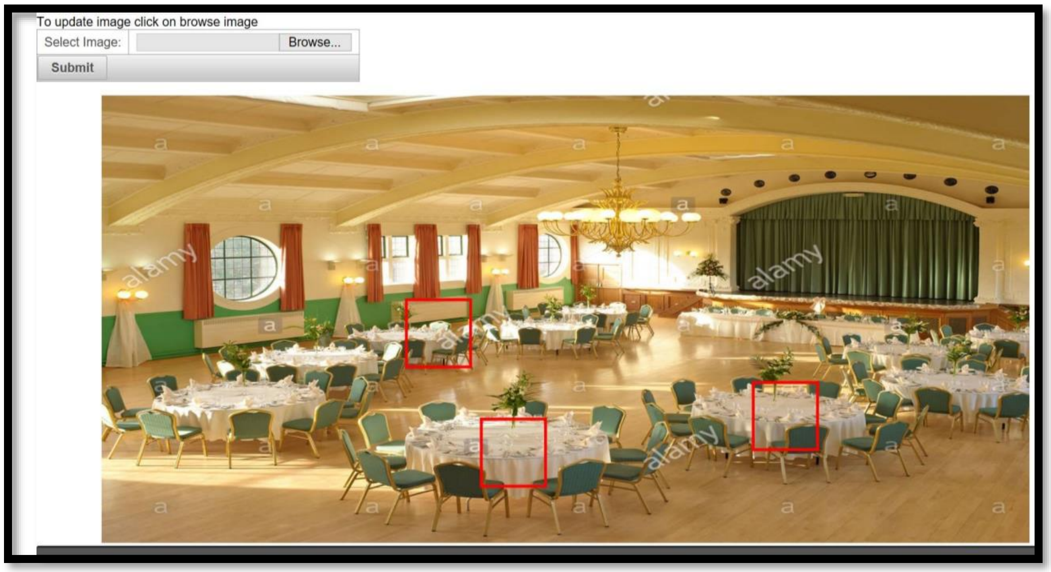


Figure 4.7: online table management module

In the above code image of the hall can be inserted into the database. And that image can be retrieved by using database queries. First of all user will be asked to enter the image or select the image after selecting the image the system will check the format of the picture whether png or jpg. After verifying the format image will be uploaded to the database. After insertion, a message is shown, saying image is inserted.

Inserting table quadrants in the database:

Input:

Quadrants of the table will be selected upon selection of table by using convex and then those quadrants will be inserted in to the database.

Output:

Quadrants will be saved to the database.

```
public void insertDB(){

    try {

        Connection con = null;
        Class.forName("org.postgresql.Driver");

        con = DriverManager.getConnection("jdbc:postgresql://localhost:5432/fyp", "postgres", "a");

        Statement statement = con.createStatement();

        int a=Integer.parseInt(getQuardx());

        int b=Integer.parseInt(getQuardy());

        int x=a-45;

        int y=b-45;

        int xend=a+45;

        int yend=b+45;

        String sql="insert into hallseating(x,y,seat,xend,yend) VALUES('"+x+"','"+y+"','"+getSeat()+"','"+xend+"','"+yend+"')";

        statement.executeUpdate(sql);

        System.out.println("insert called");

    }catch (Exception ex){}  }
```

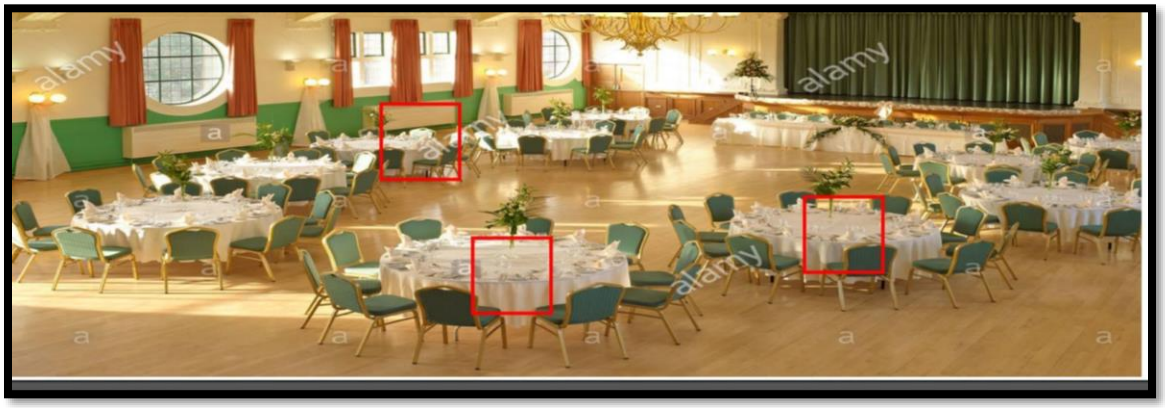


Figure 4.10 online table management module

In the above code user will click on a table to reserve that table a convex will be shown on that table colored as red if it is booked otherwise user can book that table with his/her desired preferences.

Chapter 5

Software Testing

This chapter provides a description about the adopted testing procedure. This includes the selected testing methodology, test suite and the test results of the developed software.

5.1. Testing Methodology

We have used black box testing in our testing phase. We used black box testing because it is very efficient and it contains following benefits. Black box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied virtually to every level of software testing: unit, integration, system and acceptance. Black box unit testing is used in our project.

Unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use:

- Black box tests are reproducible.
- Find software bugs early.
- Facilitates change.
- The environment the program is running is also tested.
- The invested effort can be used multiple times.
- More effective on larger units of code than glass box testing.
- Tester needs no knowledge of implementation, including specific programming languages.
- Tests are done from a user's point of view.
- Will help to expose any ambiguities or inconsistencies in the specifications.

5.2. Testing Environment

Junit:

JUnit is a unit-testing framework for the Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks, which is collectively known as xUnit that originated with SUnit.

NetBeans IDE

NetBeans is a software development platform written in Java. The NetBeans Platform allows applications to be developed from a set of modular software components called modules. ... The NetBeans IDE is primarily intended for development in Java, but also supports other languages, in particular PHP, C/C++ and HTML5.

There are 36 use cases in our project to test all those use cases we have written test cases to test the functionality of the system.

Steps to perform:

- Run Junit through Command Prompt.
- Select java code to make test cases.
- Export those test cases as a code file.
- Test cases will be tested using NetBeans IDE.

5.2.1. Test Case: Sign in

Table 5.1: Customer Sign in

Date: 06 June 2017	
System: Menu Drive	
Objective: Sign in	*Test ID:* 2
Version: 1	*Test Type:* Unit testing

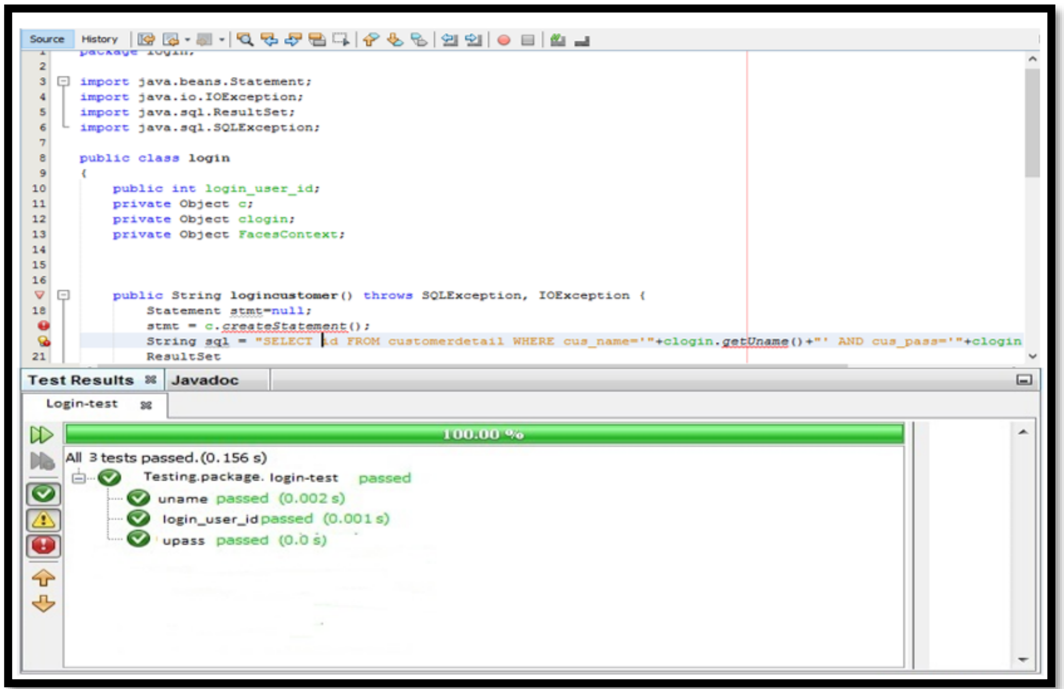
Input:
uname=Ahsan
upass=12345	

Expected Result: User will be signed in to the system.	
Actual Result: passed	

Description:

In this figure, test cases are built using Junit automated creation tool and then those test cases are tested using NetBeans IDE. First Junit automated test creation tool is started through Command Prompt Console by providing command after that it will be started. We will have to load .java code file in it for which we will have to make test cases. It will be loaded as new test. Different test cases will be built through test editor in Junit. Than these test cases will be exported as a code file, which is later loaded in NetBeans IDE to execute test cases and to check whether they are passed or failed.

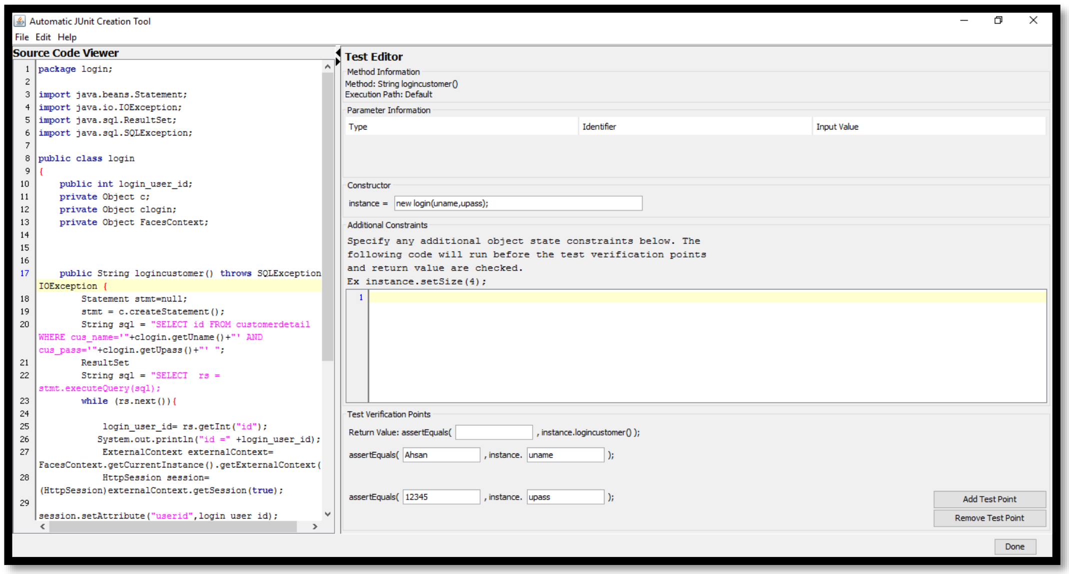
In this figure, Test cases that are created in Junit are being tested in the NetBeans IDE.



```java
package login;

import java.beans.Statement;
import java.io.IOException;
import java.sql.ResultSet;
import java.sql.SQLException;

public class login
{
    public int login_user_id;
    private Object c;
    private Object clogin;
    private Object FacesContext;



    public String logincustomer() throws SQLException, IOException {
        Statement stmt=null;
        stmt = c.createStatement();
        String sql = "SELECT id FROM customerdetail WHERE cus_name='"+clogin.getUname()+"' AND cus_pass='"+clogin
        ResultSet
```

```
All 3 tests passed.(0.156 s)
  Testing.package.login-test  passed
    uname passed (0.002 s)
    login_user_id passed (0.001 s)
    upass passed (0.0 s)
```

Figure 5.1: Sign in NetBeans IDE



```java
package login;

import java.beans.Statement;
import java.io.IOException;
import java.sql.ResultSet;
import java.sql.SQLException;

public class login
{
    public int login_user_id;
    private Object c;
    private Object clogin;
    private Object FacesContext;



    public String logincustomer() throws SQLException
IOException {
        Statement stmt=null;
        stmt = c.createStatement();
        String sql = "SELECT id FROM customerdetail
WHERE cus_name='"+clogin.getUname()+"' AND
cus_pass='"+clogin.getUpass()+"' ";
        ResultSet
        String sql = "SELECT  rs =
stmt.executeQuery(sql);
        while (rs.next()){

            login_user_id= rs.getInt("id");
           System.out.println("id =" +login_user_id);
            ExternalContext externalContext=
FacesContext.getCurrentInstance().getExternalContext(
            HttpSession session=
(HttpSession)externalContext.getSession(true);

session.setAttribute("userid",login user id);
```

Method Information
Method: String logincustomer()
Execution Path: Default

Parameter Information

Type	Identifier	Input Value

Constructor
instance = new login(uname,upass);

Additional Constraints

```
Specify any additional object state constraints below. The
following code will run before the test verification points
and return value are checked.
Ex instance.setSize(4);
```

Test Verification Points
Return Value: assertEquals(, instance.logincustomer());
assertEquals(Ahsan , instance. uname);
assertEquals(12345 , instance. upass);

Figure 5.2: Sign in JUnit

In this figure Test cases of sign in are being created by giving uname and upass.

5.2.2. Test Case: Trace location

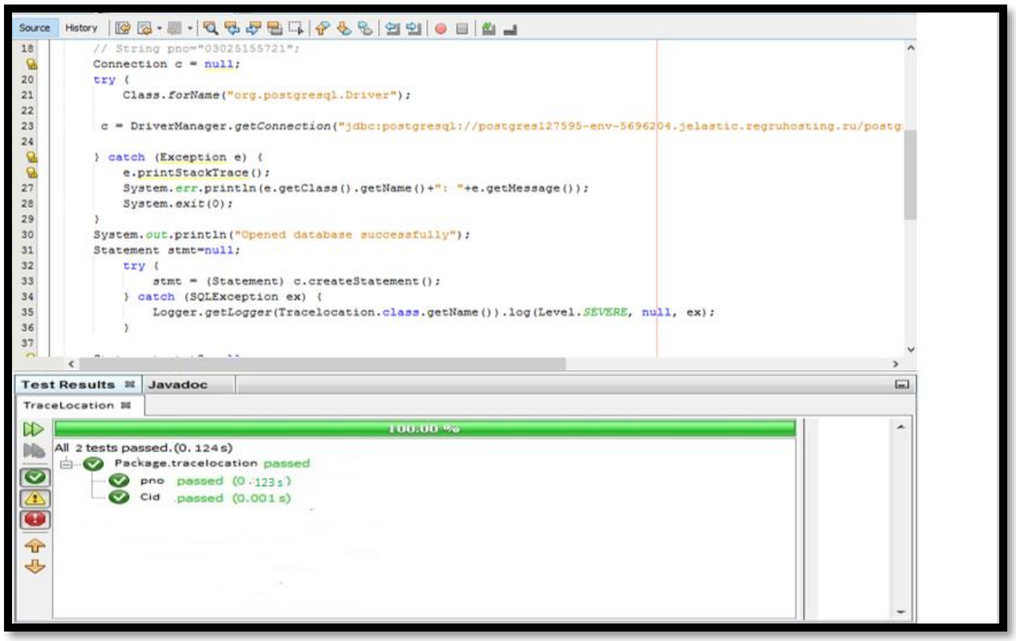
Table 5.2: Trace location

Date: 06 June 2017	
System: Menu Drive	
Objective: Trace location	Test ID : 3
Version : 1	Test Type: Unit testing
Input: pno=03325264321	
Expected Result: Delivery boy will trace location of customer.	
Actual Result: passed	

Description:

In this figure, test cases are built using Junit automated creation tool and then those test cases are tested using NetBeans IDE. First Junit automated test creation tool is started through Command Prompt Console by providing command after that it will be started. We will have to load .java code file in it for which we will have to make test cases. It will be loaded as new test. Different test cases will be built through test editor in Junit. Than these test cases will be exported as a code file, which is later loaded in NetBeans IDE to execute test cases and to check whether they are passed or failed.

In this figure, Test cases that are created in Junit are being tested in the NetBeans IDE. In this figure test cases of trace location of customer are being tested in NetBeans IDE.

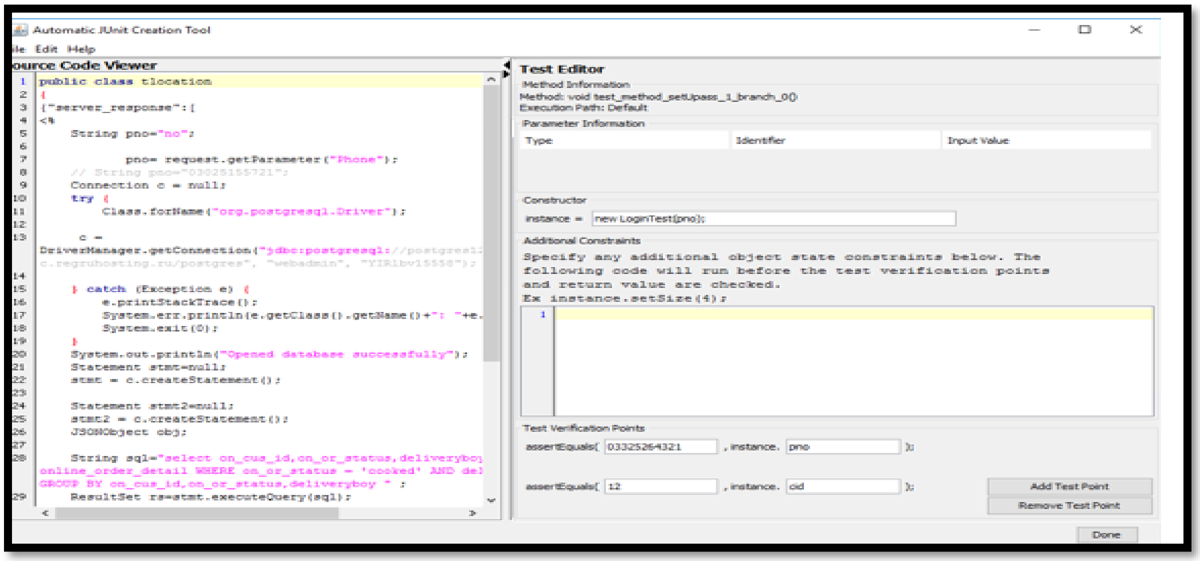


```
        // String pno="03025155721";
        Connection c = null;
        try {
            Class.forName("org.postgresql.Driver");

         c = DriverManager.getConnection("jdbc:postgresql://postgres127595-env-5696204.jelastic.regruhosting.ru/postg

        } catch (Exception e) {
            e.printStackTrace();
            System.err.println(e.getClass().getName()+": "+e.getMessage());
            System.exit(0);
        }
        System.out.println("Opened database successfully");
        Statement stmt=null;
            try {
                stmt = (Statement) c.createStatement();
            } catch (SQLException ex) {
                Logger.getLogger(Tracelocation.class.getName()).log(Level.SEVERE, null, ex);
            }
```

```
All 2 tests passed.(0. 124s)
    Package.tracelocation passed
        pno   passed  (0.123 s)
        Cid   passed  (0.001 s)
```

Figure 5.3: Trace location NetBeans IDE

In this figure test case of trace location of customer are being written in Junit.



```
public class tlocation
{
{"server_response":[
<%
    String pno="no";

            pno= request.getParameter("Phone");
    // String pno="03025155721";
    Connection c = null;
    try {
        Class.forName("org.postgresql.Driver");

     c =
DriverManager.getConnection("jdbc:postgresql://postgres1
c.regruhosting.ru/postgres", "webadmin", "YIRlbv15558");

    } catch (Exception e) {
        e.printStackTrace();
        System.err.println(e.getClass().getName()+": "+e.
        System.exit(0);
    }
    System.out.println("Opened database successfully");
    Statement stmt=null;
    stmt = c.createStatement();

    Statement stmt2=null;
    stmt2 = c.createStatement();
    JSONObject obj;

    String sql="select on_cus_id,on_or_status,deliveryboy
online_order_detail WHERE on_or_status = 'cooked' AND del
GROUP BY on_cus_id,on_or_status,deliveryboy " ;
    ResultSet rs=stmt.executeQuery(sql);
```

Method Information
Method: void test_method_setUpass_1_branch_0()
Execution Path: Default

Parameter Information

Type	Identifier	Input Value

Constructor
instance = new LoginTest(pno);

Additional Constraints

```
Specify any additional object state constraints below. The
following code will run before the test verification points
and return value are checked.
Ex instance.setSize(4);
```

Test Verification Points

assertEquals(03325264321 , instance. pno);

assertEquals(12 , instance. cid);

Figure 5.4: Trace location Junit

5.2.3. Test Case: Get order details

Table 5.3: get order details

Date: 06 June 2017	
System: Menu Drive	
Objective: Get order details	Test ID: 4
Version: 1	Test Type: Unit testing

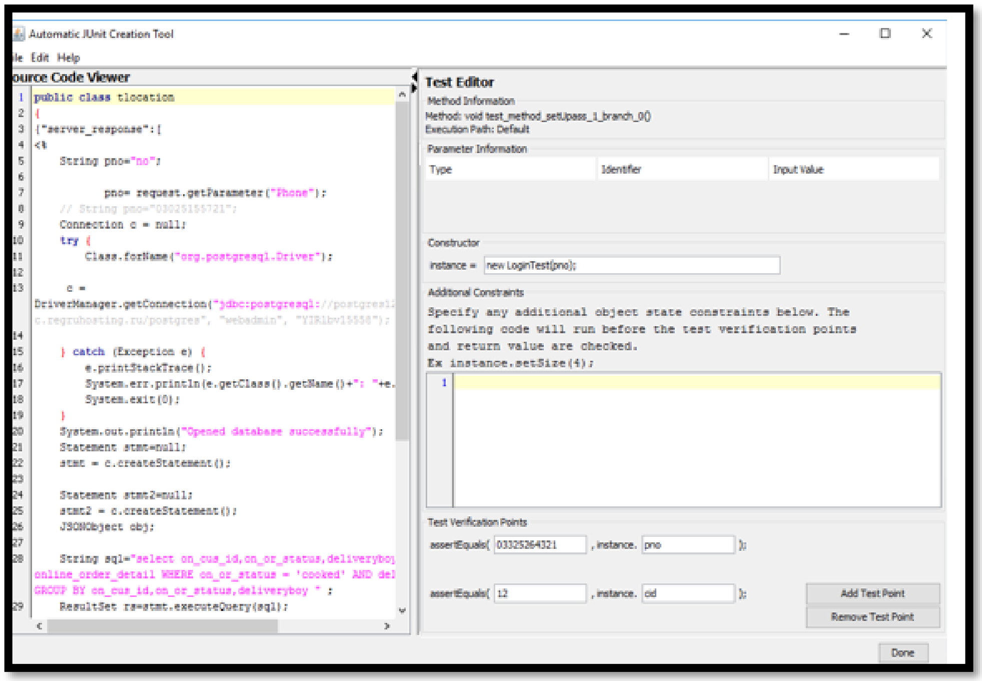
Input:
Pno=03325264321
cid=12	

Expected Result: Delivery boy will order details of customer.	
Actual Result: passed	

Description:

In this figure, test cases are built using Junit automated creation tool and then those test cases are tested using NetBeans IDE. First Junit automated test creation tool is started through Command Prompt Console by providing command after that it will be started. We will have to load .java code file in it for which we will have to make test cases. It will be loaded as new test. Different test cases will be built through test editor in Junit. Than these test cases will be exported as a code file, which is later loaded in NetBeans IDE to execute test cases and to check whether they are passed or failed.

In the following figure test case of getting order details of customer is being written in Junit.



```
public class tlocation
{
{"server_response":[
<%
    String pno="no";

            pno= request.getParameter("Phone");
    // String pno="03025155721";
    Connection c = null;
    try {
        Class.forName("org.postgresql.Driver");

     c =
DriverManager.getConnection("jdbc:postgresql://postgres12
c.regruhosting.ru/postgres", "webadmin", "YIRlbv15558");

    } catch (Exception e) {
        e.printStackTrace();
        System.err.println(e.getClass().getName()+": "+e.
        System.exit(0);
    }
    System.out.println("Opened database successfully");
    Statement stmt=null;
    stmt = c.createStatement();

    Statement stmt2=null;
    stmt2 = c.createStatement();
    JSONObject obj;

    String sql="select on_cus_id,on_or_status,deliveryboy
online_order_detail WHERE on_or_status = 'cooked' AND del
GROUP BY on_cus_id,on_or_status,deliveryboy " ;
    ResultSet rs=stmt.executeQuery(sql);
```

Method Information
Method: void test_method_setUpass_1_branch_0()
Execution Path: Default

Parameter Information

Type	Identifier	Input Value

Constructor
instance = new LoginTest(pno);

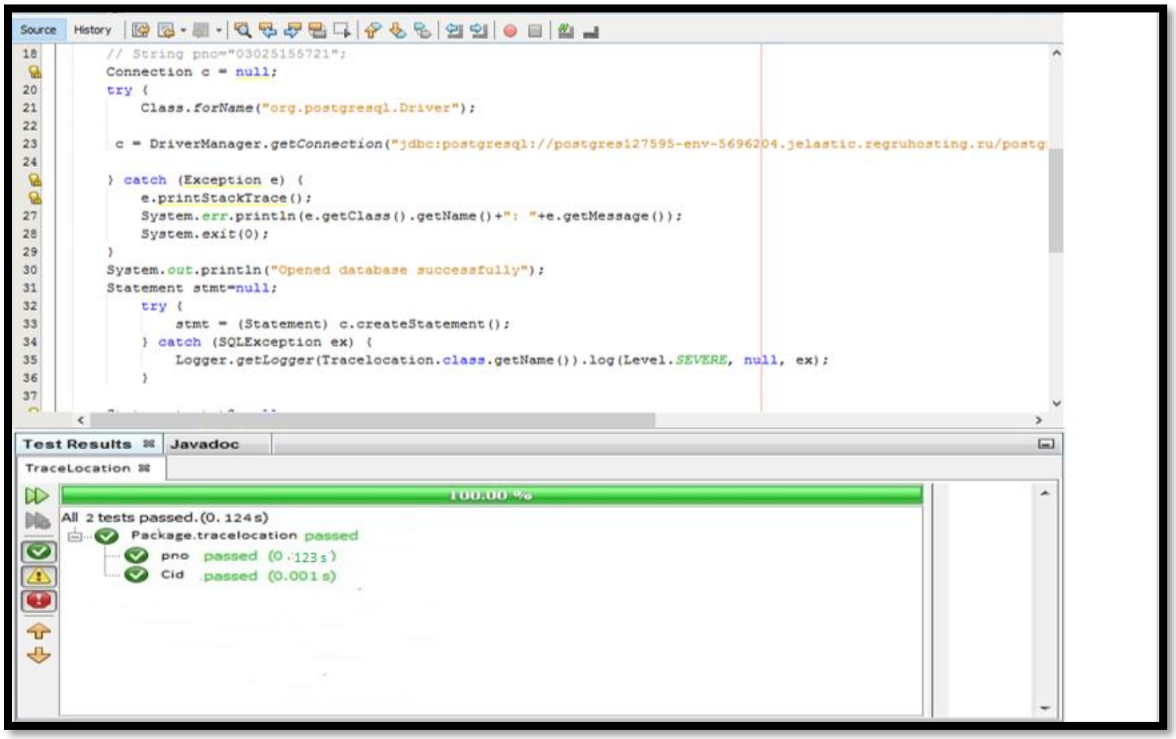
Additional Constraints

```
Specify any additional object state constraints below. The
following code will run before the test verification points
and return value are checked.
Ex instance.setSize(4);
```

Test Verification Points

```
assertEquals( 03325264321 , instance. pno );
assertEquals( 12 , instance. cid );
```

Figure 5.5:get order details JUnit



```
    // String pno="03025155721";
    Connection c = null;
    try {
        Class.forName("org.postgresql.Driver");

     c = DriverManager.getConnection("jdbc:postgresql://postgres127595-env-5696204.jelastic.regruhosting.ru/postg

    } catch (Exception e) {
        e.printStackTrace();
        System.err.println(e.getClass().getName()+": "+e.getMessage());
        System.exit(0);
    }
    System.out.println("Opened database successfully");
    Statement stmt=null;
        try {
            stmt = (Statement) c.createStatement();
        } catch (SQLException ex) {
            Logger.getLogger(Tracelocation.class.getName()).log(Level.SEVERE, null, ex);
        }
```

```
All 2 tests passed.(0.124 s)
    Package.tracelocation passed
        pno  passed (0.123 s)
        Cid  passed (0.001 s)
```

Figure 5.6:get order details Junit

In this figure, test cases of getting order details of customer are being tested in NetBeans IDE.

5.2.4. Test Case: Reserve Table

Table 5.4: Reserve table

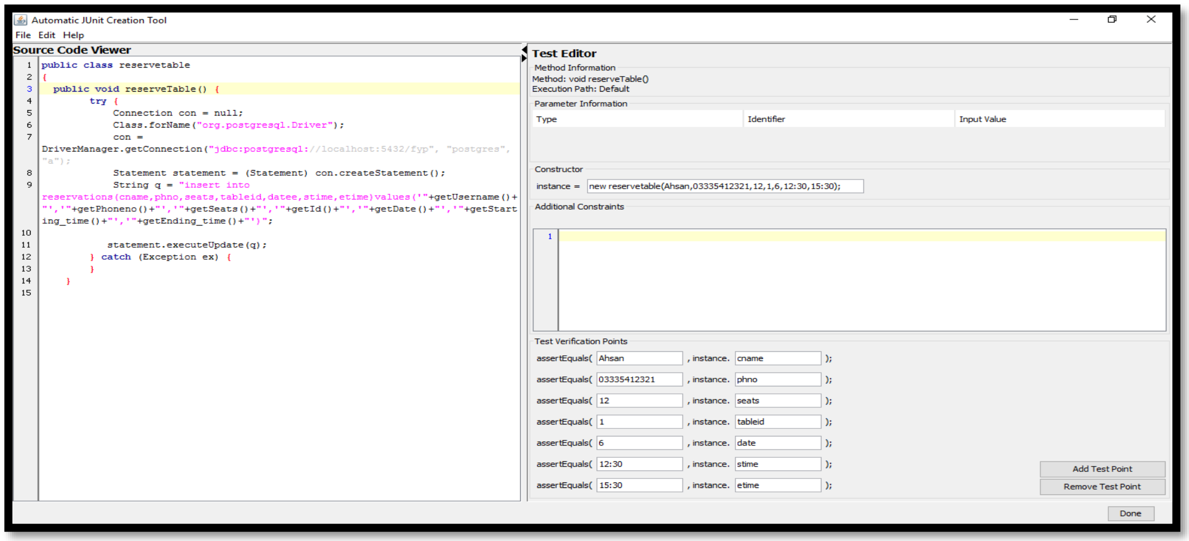
Date: 06 June 2017	
System: Menu Drive	
Objective: Table Reservation	Test ID: 5
Version: 1	Test Type: Unit testing

Input:
uname=Ahsan
pno=03325264321
seats=12
date=6 June 2017
stime=12:30
etime=15:30	

Expected Result: User will reserve table online.	
Actual Result: passed	

Description:

In this figure, test cases are built using Junit automated creation tool and then those test cases are tested using NetBeans IDE. First Junit automated test creation tool is started through Command Prompt Console by providing command after that it will be started. We will have to load .java code file in it for which we will have to make test cases. It will be loaded as new test. Different test cases will be built through test editor in Junit. Than these test cases will be exported as a code file, which is later loaded in NetBeans IDE to execute test cases and to check whether they are passed or failed. In this figure test case for the reserve table are being written in Junit.



```
public class reservetable
{
  public void reserveTable() {
        try {
            Connection con = null;
            Class.forName("org.postgresql.Driver");
            con =
DriverManager.getConnection("jdbc:postgresql://localhost:5432/fyp", "postgres",
"a");
            Statement statement = (Statement) con.createStatement();
            String q = "insert into
reservations(cname,phno,seats,tableid,datee,stime,etime)values('"+getUsername()+
"','"+getPhoneno()+"','"+getSeats()+"','"+getId()+"','"+getDate()+"','"+getStart
ing_time()+"','"+getEnding_time()+"')";

           statement.executeUpdate(q);
        } catch (Exception ex) {
        }
    }
```

Method Information
Method: void reserveTable()
Execution Path: Default

Parameter Information

Type	Identifier	Input Value

Constructor
instance = new reservetable(Ahsan,03335412321,12,1,6,12:30,15:30);

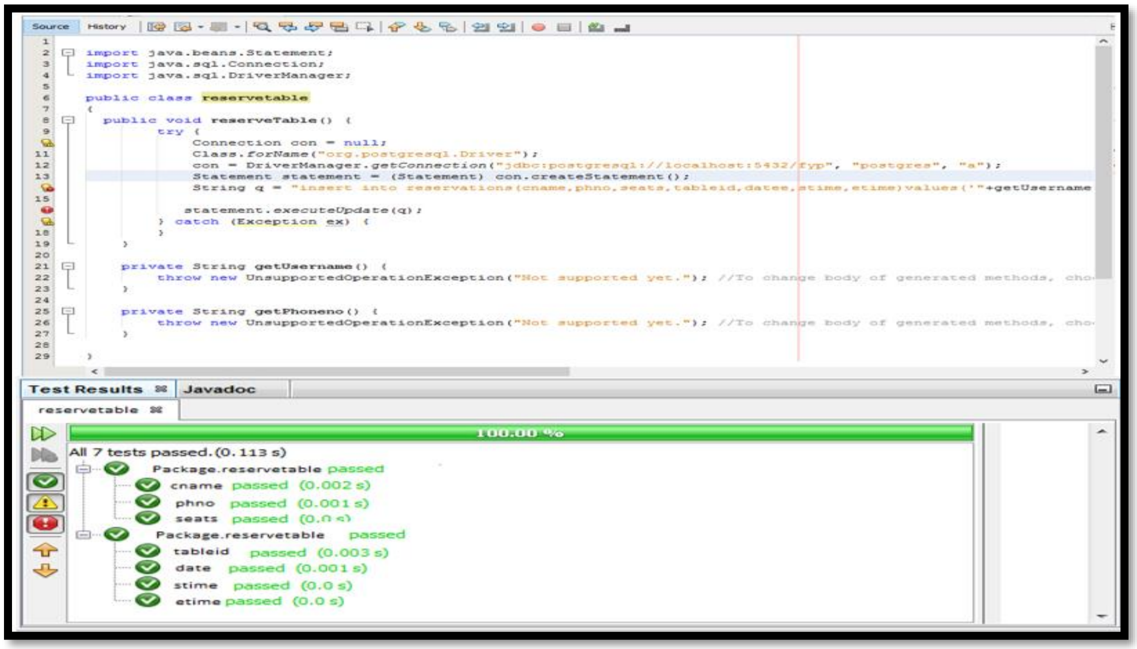
Additional Constraints

Test Verification Points

```
assertEquals( Ahsan       , instance. cname   );
assertEquals( 03335412321 , instance. phno    );
assertEquals( 12          , instance. seats   );
assertEquals( 1           , instance. tableid );
assertEquals( 6           , instance. date    );
assertEquals( 12:30       , instance. stime   );
assertEquals( 15:30       , instance. etime   );
```

Figure 5.7: reserve table JUnit

In the following figure, test case of reserve table is being executed and tested in NetBeans IDE.



```
import java.beans.Statement;
import java.sql.Connection;
import java.sql.DriverManager;

public class reservetable
{
    public void reserveTable() {
            try {
                Connection con = null;
                Class.forName("org.postgresql.Driver");
                con = DriverManager.getConnection("jdbc:postgresql://localhost:5432/fyp", "postgres", "a");
                Statement statement = (Statement) con.createStatement();
                String q = "insert into reservations(cname,phno,seats,tableid,datee,stime,etime)values('"+getUsername

               statement.executeUpdate(q);
            } catch (Exception ex) {
            }
        }

        private String getUsername() {
            throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, cho

        }

        private String getPhoneno() {
            throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, cho
        }

    }
```

```
100.00 %
All 7 tests passed.(0.113 s)
  Package.reservetable passed
    cname passed (0.002 s)
    phno passed (0.001 s)
    seats passed (0.0 s)
  Package.reservetable passed
    tableid passed (0.003 s)
    date passed (0.001 s)
    stime passed (0.0 s)
    etime passed (0.0 s)
```

Figure 5.8: reserve table Junit

Chapter 6

Software Deployment

6.1. Installation / Deployment Process Description

For the deployment, we will provide user with the .war files and two apk's of android applications. The .war file will contain XHTML web pages; database connectivity details and database script lets.

The .war file will be enclosed with in a .zip file. We will provide .war file to the user which he/she will upload on any of the desired web server and get it running. We will provide apk's of three applications one of which will be Delivery boy app through this application Delivery boy can trace location of customer and Delivery boy can get order details of the customer. The other mobile application will be of waiter through that a waiter can order from the customer.

Chapter 7

Project Evaluation

This chapter includes the examiners evaluation report, including the points to be revised/included along with the selected requirements in the next iteration.

7.1. Project Evaluation Report

Examiner Name:	
S. No.	**Suggestion**
1	Entity relation should be labeled.
2	Sequence diagram should be split.
3	English sentence structure should be improved.
4	Formatting should be improved.
5	Naming conventions are not followed in code artifacts.
6	Indention rules are not followed as described in chapter 4
7	Test cases for the incorrect behavior to be included.
8	Test cases should be added to test remaining functionalities.

Other Comments (If any):

____

____

____

____

Signature