SEAT ME!
(Online Table Booking System)

Submitted in partial fulfillment of the requirements
of the degree of

B. E. Information Technology

By

Indraneel Das	**04**
Mitesh Gupta	**09**
Kunal Naickar	**15**
Anjali Pai	**17**

Supervisor(s):

Mr. Bhavesh Pandya
Assistant Professor

Department of Information Technology
St. Francis Institute of Technology
(Engineering College)

University of Mumbai
2016-2017

CERTIFICATE

This is to certify that the project entitled **"SEAT ME (Online Table Booking System)"** is a bonafide work of **Indraneel Das (Roll No. 04), Mitesh Gupta (Roll No. 09), Kunal Naickar (Roll No. 15), Anjali Pai (Roll No. 17)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of B.E. in Information Technology.

Mr. Bhavesh Pandya
(Project Guide)

Dr. Joanne Gomes
(Head of Department)

Dr. Sincy George
(Principal)

Project Report Approval for B.E.

This project report entitled "SEAT ME (Online Table Booking System)" by Indraneel Das (Roll No. 04), Mitesh Gupta (Roll No. 09), KunalNaickar (Roll No. 11), Anjali Pai (Roll No. 17) is approved for the degree of **B.E. in Information Technology.**

Examiners

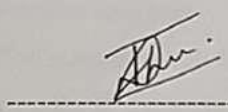
1. Anagha Patil
2. Bhavesh Pandya BNPandya

Date: 29/4/2017

Place: MUMBAI

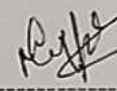
Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.



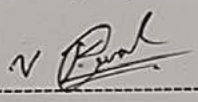
(Signature)

Indraneel Das (Roll no. 04)



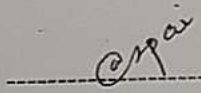
(Signature)

Mitesh Gupta (Roll no. 09)



(Signature)

Kunal Naickar (Roll no. 15)



(Signature)

Anjali Pai (Roll No. 17)

Date: 29/4/2017

Abstract

The proposed system involves building a user interface for the end user who wants to book a table in a particular restaurant. The proposed system provides a list of restaurants in the vicinity of the location the user enters. Then it provides a layout of the seating arrangement of the selected restaurant and allows the user to book the table if it is unoccupied. The tables that are already booked will be marked with a red indicator on the layout. The tables that are not booked will be marked with green indicator. The system will very well cater to the requirements of the end user and will also help the user to save time. If the user is found booking falsely, the restaurant manager will be allowed to blacklist the customer with particular username. The manager or the admin can also have weekly promotions for the customer. The customer can also view the history of all the restaurants visited by him in the period of one month. If the customer has visited the restaurant and has been accepted by the restaurant manager he will be prompted to make a quick review of his experience.

Contents

List of Figures

List of Tables

List of Abbreviations

Sr. No.	Abbreviation	Expanded form
1	DFD	Data Flow Diagram
2	UI	User Interface
3	WBS	Work Breakdown Structure

Chapter 1

Introduction

Over the recent years, technology has started to play a significant role in the modern human lifestyle. In today's world, where just about everything is more convenient and accessible due to advances in technology across almost all sectors. With efficient use of available resources, it is possible to do any task right at your finger tip with minimal effort and within fraction of seconds.

Most of the people in our society have heavy work hours and busy lifestyle, and when they plan to go for a snack, lunch or dinner, they happen to spend more time on searching the restaurant and then waiting for the tables to vacant during rush hours which is very frustrating. The aim of this project is to solve this problem by efficiently using technology collaborated with the real world by letting the customers search for the restaurant of their choice with various filters and then viewing the ambience and deciding over the table right from their home /workplace and reserving the table with just a single click on their android phones added with much more features for user experience. Therefore, the project aims in replacing the traditional waiting queues outside the restaurant by giving a visual view of the seating inside the restaurant.

1.1 Description

The application is developed on android platform. Most of the smart phones today use Android OS, therefore the application will be able to cater the maximum number of users. The android platform is easy to use and there are various tools available to develop applications which would run on this platform. So by using this platform we aim to develop an application that would help the user to book a table at a restaurant at the tip of his finger. In today's fast moving world where time is money, this application will help the user to save his/her time which they waste by going to the restaurant and waiting for the table to be free. The application aims to collaborate the real world with technology efficiently to help the user search for the restaurants around him and also book a table in the restaurant.

The user will sign up for the first time on this application then for the next times he/she is only required to log in. Then with the help of the GPS or manually entering the location the user will be able to see the restaurants around him/her and also will be able to see the menu and the reviews given by other customers and book a table accordingly. The application will maintain a list of the reviews given by the customer and also display the same. The user who has just visited the restaurant will also be able to write a review.

1.2 Problem Formulation

Presently, most of the websites like Zomato, Foodpanda, Swiggy provide only with facilities such as menu viewing, contact number and food ordering. Usually in restaurants, to book a table in advance the customer needs to call the restaurant manager to check for seats available and then if they are, the manager reserves the seats. The drawback of this system is that the customer cannot view the seat which has been booked for him sitting at home. Also some of the restaurants do not consider reserving tables during rush hours.

1.3 Motivation:

The world has evolved with technology. Time is very expensive to be wasted just by waiting for seat outside a restaurant. Keeping in mind the problems incurred we came up with the idea of online table reservation from android phones as everyone has a phone with them , this system will perhaps help our users to efficiently find for restaurants nearest, check their menu online, view the seating of the restaurant and then book the table of their choice. This project can further be expanded with better user experience by providing the user the facility to review a particular restaurant he has visited, by providing privilege discounts on the special occasion of the user, inviting a friend with a click on the application etc. Thus enhancing the overall user experience.

1.4 Proposed Solution

The proposed project is a smart restaurant reservation system that provides customers an easy way of reserving a table and getting a customized search result.

Restaurant discovery:

Location-based search, history of bookings, and user-generated reviews are the features that the app uses to help its customers discover and book tables at restaurants.

Search:

To use mobile app, first of all, you need to create an account using your email address and a password, or you can simply sign in with your Facebook account. Then, the very first screen will give you suggestions based on your current location. The suggestions include a list of restaurants in a given area, their ratings based on users' reviews, and available reservation time. You can search for a specific place, city, neighborhood, and filter suggestions by location, cuisine.

Once you click on the restaurant of your choice you will get to view the seating area of the restaurant where you can click on the seat you wish to reserve and then reserve the seat. You will have to choose and also update the time and number of people accompanying the restaurant with you .It will also provide the venue's location, ratings, and menu, the app also provides other information on a restaurant such as its phone numbers, prices, open hours, parking availability, and a short description of a place.

History of bookings:

Apart from a restaurant search, users can see the history of bookings they have made before. This feature provides users with a quicker access to the places they have already visited. It may really come in handy in case you have forgotten the name of a restaurant you liked.

Also the application will capture the birth date of family members of the guests visiting and flashes a special offer to the customer on their special date by notifying the user. Restaurant promotions are offered to the users. Reward points can be created for future use.

User-generated reviews:

Users can post a review on the app. They can leave comments and rate a restaurant's food, ambience, and service from 1 to 5. The app won't let just anybody post a review, you have to be at the restaurant that you wish to tell the world about.

For the Restaurant Manager:

Manager can see the occupied tables, block the tables for offline booking in peak hours. The manager can blacklist the cheaters. He can check the breakup of the sales session.

1.5 Scope of Project:

Since the mobile technology continuous to grow up, each and every thing is being done with the help of mobiles. Therefore we aim to use the mobile to help the user with the very basic need of booking a table whenever the user goes out for having food. For now the user would only be able to see the restaurants around him/her, book a table and write a review for the same. In the future we also aim to build a feature such that the user would also be able to order food and make payment for the same.

In this system, the inputs are the location of the user and the user gets the restaurants which are near the user, the user then books a table in the restaurant.

Chapter 2

Review of Literature

2.1 Review of Literature

[1] By referring to first paper we became aware of the already existing systems. The potential for the use of Information Technology (I.T.) in the field of hotel industry was identified in 2010 by Courtney McTavish and Suresh Sankarnarayanan in their paper "Intelligent agent based hotel search & booking system", Electro/Information Technology (EIT), IEEE International Conference . The system here would use an intelligent agent (instead of human agent) to perform similar search and booking activities that can improve the speed of search and reduce the cost significantly. The paper mentioned above talks on the development of a system based on the outdated technologies like JADE-LEAP agent development kit. The system (application) that will be developed by us would be an Android application which is the latest and the fast growing technology.

The number of users of this application would be greater as ever one now-a-days carries a smartphone. The system which was developed had limited features i.e. it cannot give you the restaurants you are looking for in your surroundings or more precisely according to your location. The application which will be developed by us would give the real time availability of tables in a restaurant .The user would be able to book the tables according to his/her preference .The user would also be able to see a 360 degree view of the tables present in the particular restaurant . Also the application would have a feedback mechanism for both the customer as well as the manager of the restaurants. The feedback system would help us to blacklist the customers as well as the restaurants. The restaurants will be blacklisted if they do not provide the customer with what they promised and the customers would be blacklisted if they make a booking for the table in a restaurant and do not reach the restaurant.

With this application we aim to eliminate the time for which the customer has to wait at the restaurant and also it will be a very easy task for the restaurants to manage their customers at times of rush.

[2] This was the most helpful paper of all and a base for our current project idea. This paper has helped us to understand the concepts of RDBMS, OODBMS and ODBMS. The use of object oriented database system and operational database is useful to maintain data at the server side. Using this helps us in making the application online and independent of the underlying operating system.

Chapter 3

System Analysis

3.1 Functional Requirements

Functional requirements describes all the required functionality or system services. Functionality is described as a set of inputs, the behavior, and outputs expected from the system. They are functions or features that must be included in a system in order to satisfy the business needs and must be acceptable to the system users.

The proposed system should provide following functional requirements:

- The system requires the user to register by providing the details and then login to the system by providing the username and password.
- The system provides the user with a menu. The user can Book the table, view current booking and can also cancel the reserved table
- The system provides with location, date & time and the number of covers user wants to book. And when the user reserves a table an ID will be generated for that table
- The system also provides the user to view the live booking seating areas
- The system tracks the booked slots by checking the ID.

3.2 Non-Functional Requirements

The non-functional requirements define the properties and constraints of the system. Non-functional requirements impose constraints on the design or implementation. It is used to judge the operation of a system

The proposed system should provide following non-functional requirements:

- Acceptable – Verified as meeting the stated objectives
- Accessible – Accessible from different devices
- Availability – 24 x 7 x 365. (Booking tables will work only during workings hours of the restaurant)
- Secured and Restorable – System and data backups. Business Continuity / Disaster Recovery
- Maintainability-The ease with which the system can be changed, whether for bug fixes or to add new functionality.
- Modifiable & Extensible – Ease and cost to make ongoing changes.
- Responsive / Performance – Response times. Speed of page loads and calculations.
- Reliable – Consistent and dependable quality of service.
- Usable – Easy to use by target users.

3.3 Specific Requirements

Hardware Requirements

- Hard Disk : 1 GB hard disk space and above.
- RAM : Minimum 512MB and above.
- 1GHZ processor

Software Requirements

- The system will run on any device with android operating system
- Coding Language : Java, PHP
- Software : Android Studio, Text editor.
- Backend : MYSQL

An internet connection is required.

3.4 Use-Case Diagram and description

Diagram:

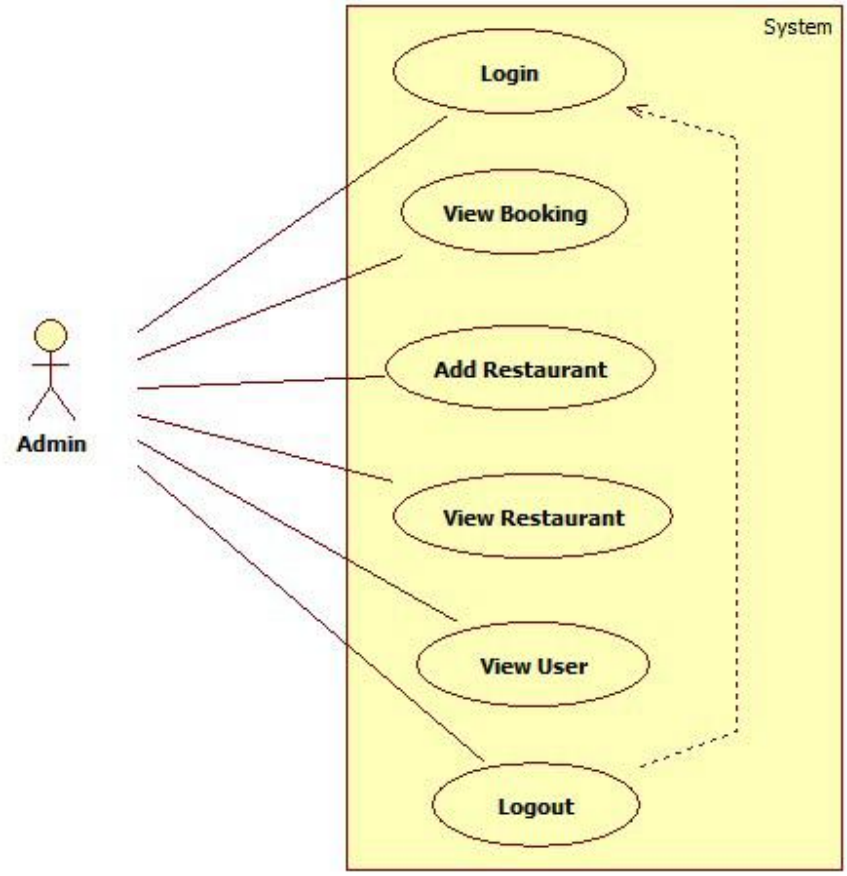


Fig. 3.4.1: Use Case Diagram of Admin

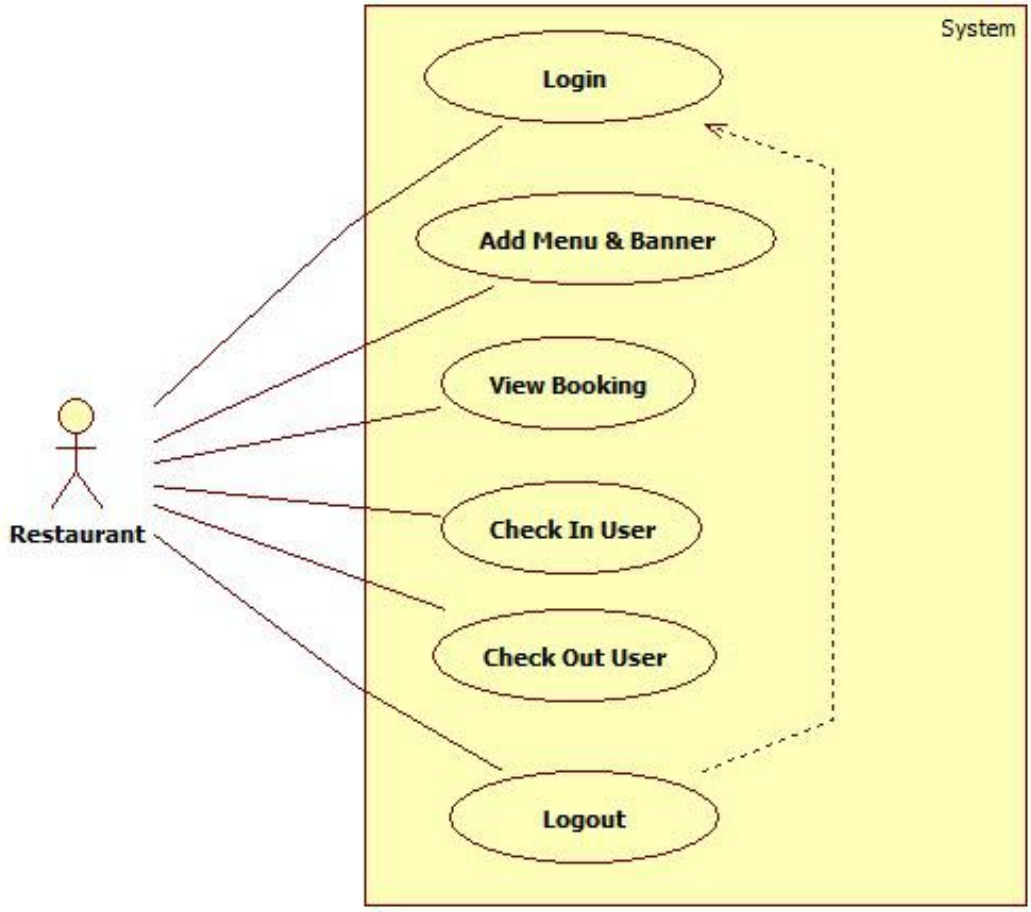


Fig. 3.4.2: Use Case Diagram of Restaurant Manager

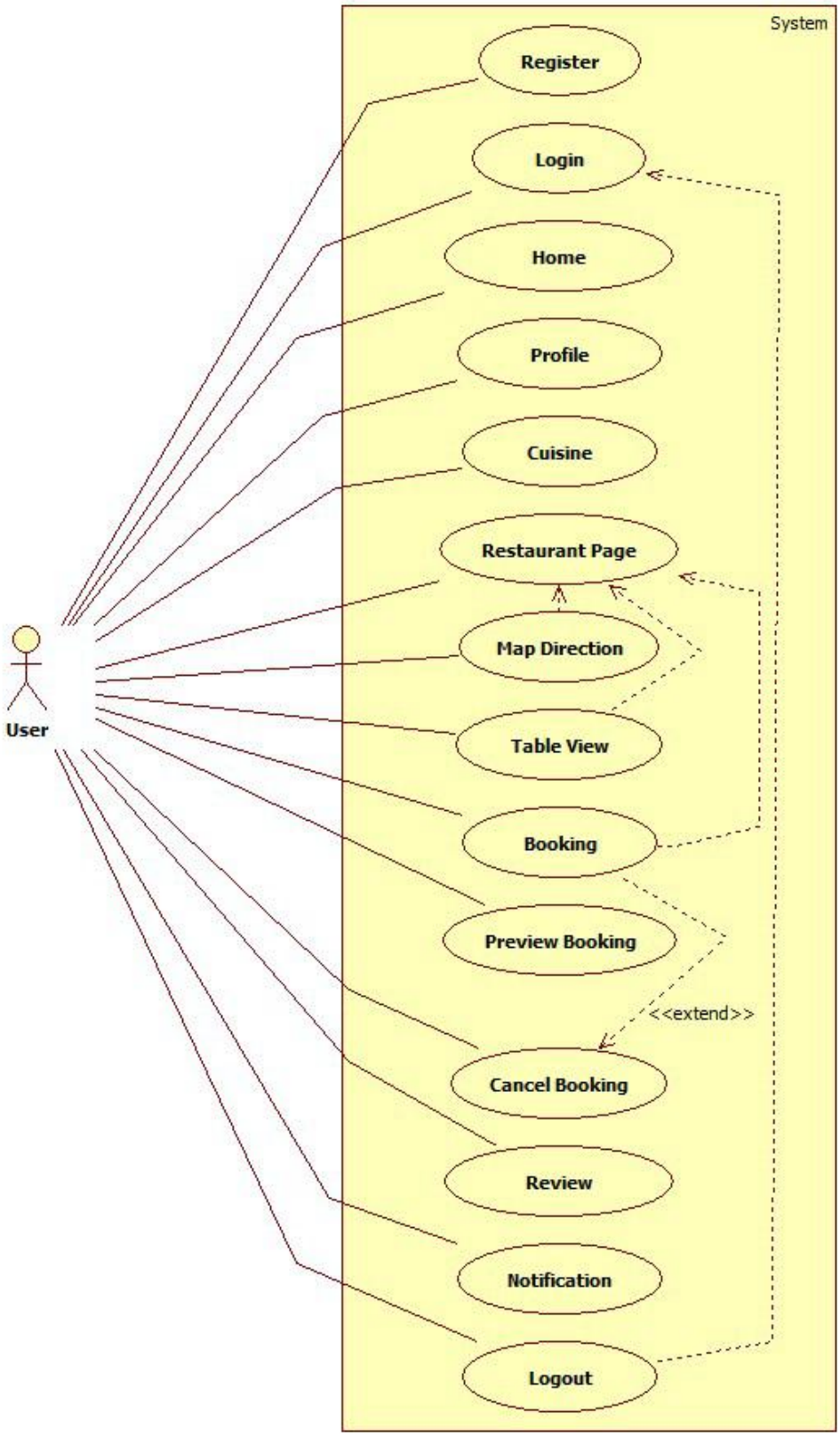


Fig. 3.4.3: Use Case Diagram of Android User

Use Case	Register
Primary Actor	Customer
Goal in Context	Allows user to enter the details and to login to the system.
Preconditions	User must use Online table booking system
Trigger	On entering the details the user gets logged in.
Scenario	User observes the UI and enters the corresponding details in the text box.
Exceptions	User does not enter the correct information.
Secondary Actor	Admin

Table 3.4.1: Use Case Description of "Register & Provide details"

Use Case	Select operations
Primary Actor	Customer
Goal in Context	To prompt the user with the menu.
Preconditions	User must be logged in to the system.
Trigger	On completion of registration process, user selects an appropriate operation.
Scenario	User registers, system performs certain operations.
Exceptions	Operation is not selected.

Table 3.4.2: Use Case Description of "Select Operations"

Use Case	Book
Primary Actor	Customer
Goal in Context	To book a table.
Preconditions	User must be logged in to the system.
Trigger	On selecting book operation, the system provides with the location, date & time, no. of tables vacant
Scenario	User selects this operation and the user can book table for given no of covers.
Exceptions	User does not book any table.

Table 3.4.3: Use Case Description of "Book"

Use Case	Cancel
Primary Actor	Customer
Goal in Context	To cancel one or more table
Preconditions	User must be logged in to the system.
Trigger	On selecting the cancel operation, the user cancels the reserved table
Scenario	User selects this operation and the user can cancel reserved table
Exceptions	User does not cancel any table

Table 3.4.4: Use Case Description of "Cancel"

Use Case	View Current Booking
Primary Actor	Customer
Goal in Context	To view representation of the seating space
Preconditions	User must be logged in to the system.
Trigger	On selecting view current booking operation, the system provides the representation of parking where red represents the occupied tables and green represents the remaining tables.
Scenario	User selects this operation and the user can view the current booking scenario.

Table 3.4.5: Use Case Description of "View Current Booking"

Use Case	Make Payment
Primary Actor	Customer
Goal in Context	To pay for booking tables or order and pay for at least one item
Preconditions	User must book a table
Scenario	User selects this operation to pay for the table that he/she wants to book.
Exceptions	URL is not entered.

Table 3.4.6: Use Case Description of "Make Payment"

Use Case	Login
Primary Actor	Admin
Goal in Context	Allows admin to enter the details and to login to the system.
Preconditions	Admin's details must be stored in the database.
Trigger	On entering the details the admin gets logged in.
Scenario	Admin observes the UI and enters username and password.
Exceptions	Admin does not enter the correct information.
Secondary Actor	Customer

Table 3.4.7: Use Case Description of "Login"

Chapter 4

Analysis Modeling

4.1 Data Modeling

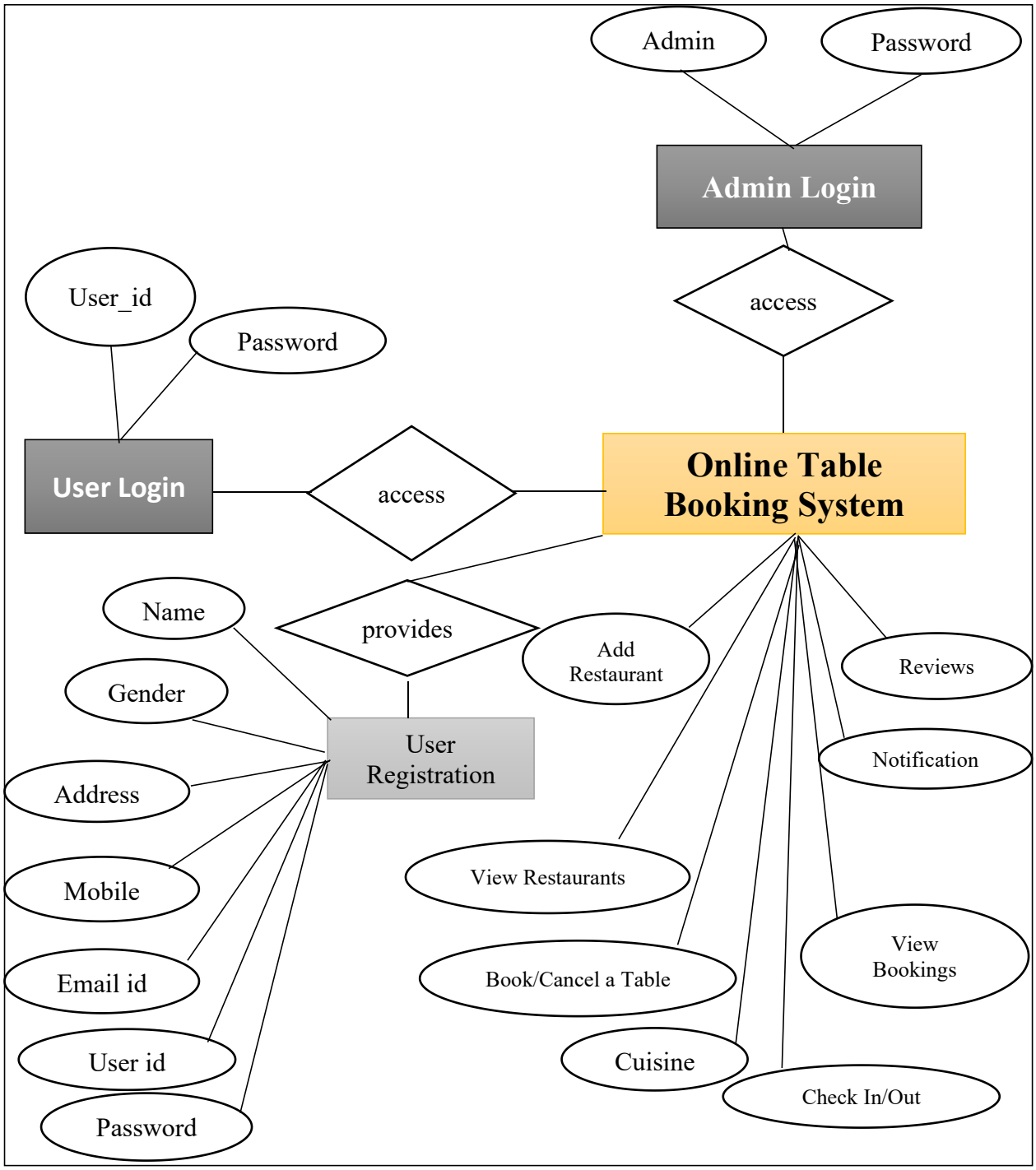


Fig 4.1.1 :ER Diagram

4.2 Class Diagram

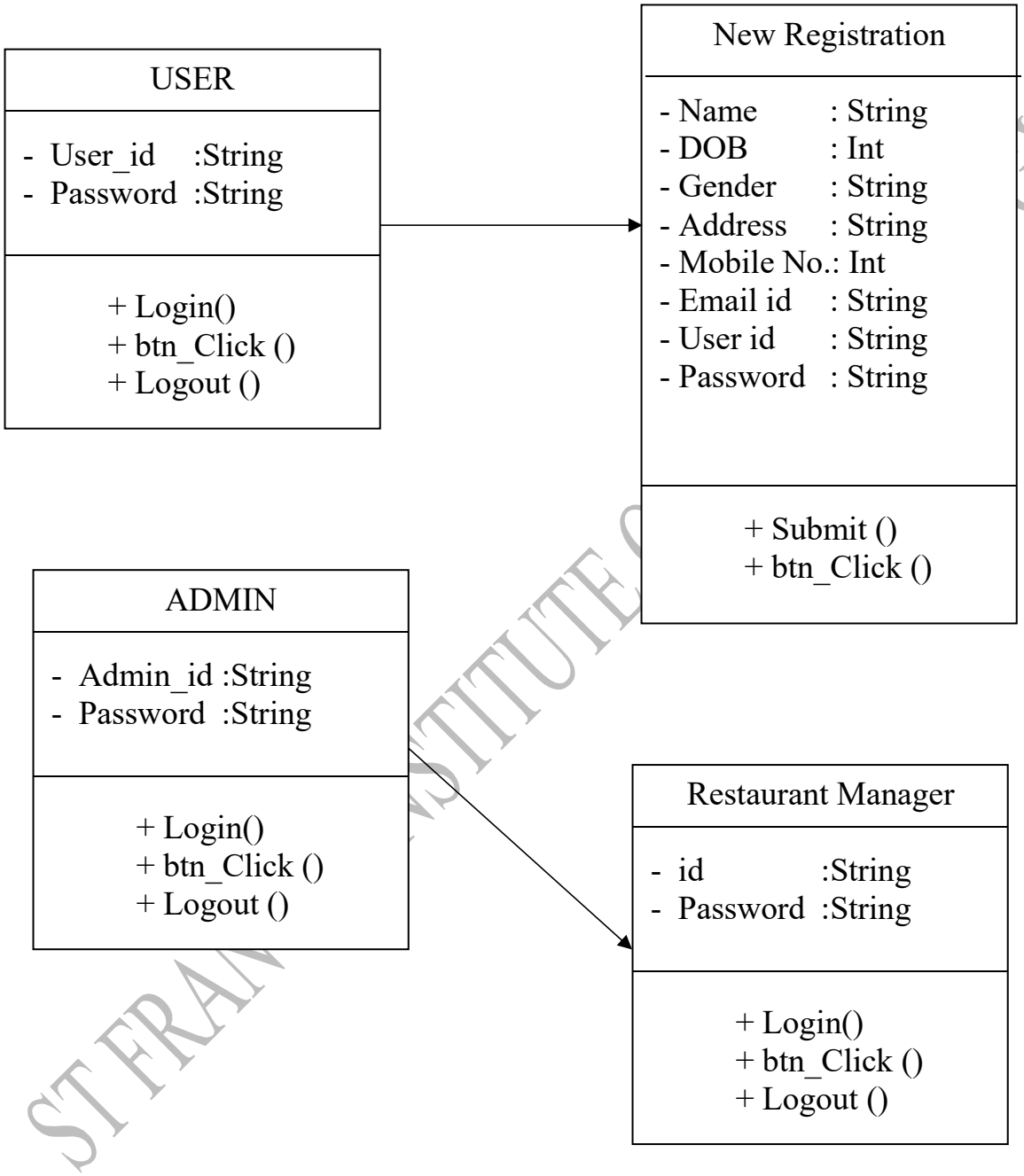


Fig 4.2.1 :Class Diagram

4.3 Sequence Diagram

Fig. 4.3.1: Sequence Diagram of Admin

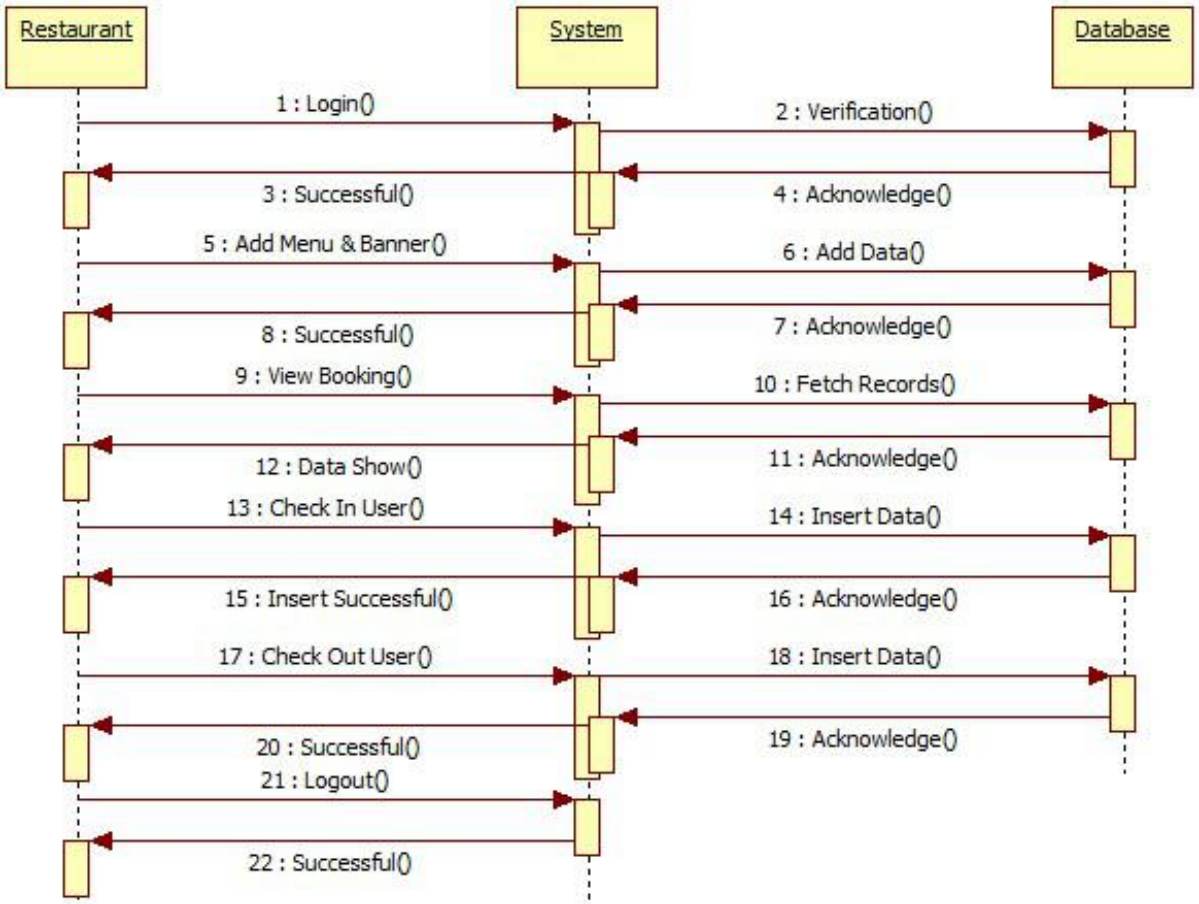


Fig. 4.3.2: Sequence Diagram of Restaurant Manager

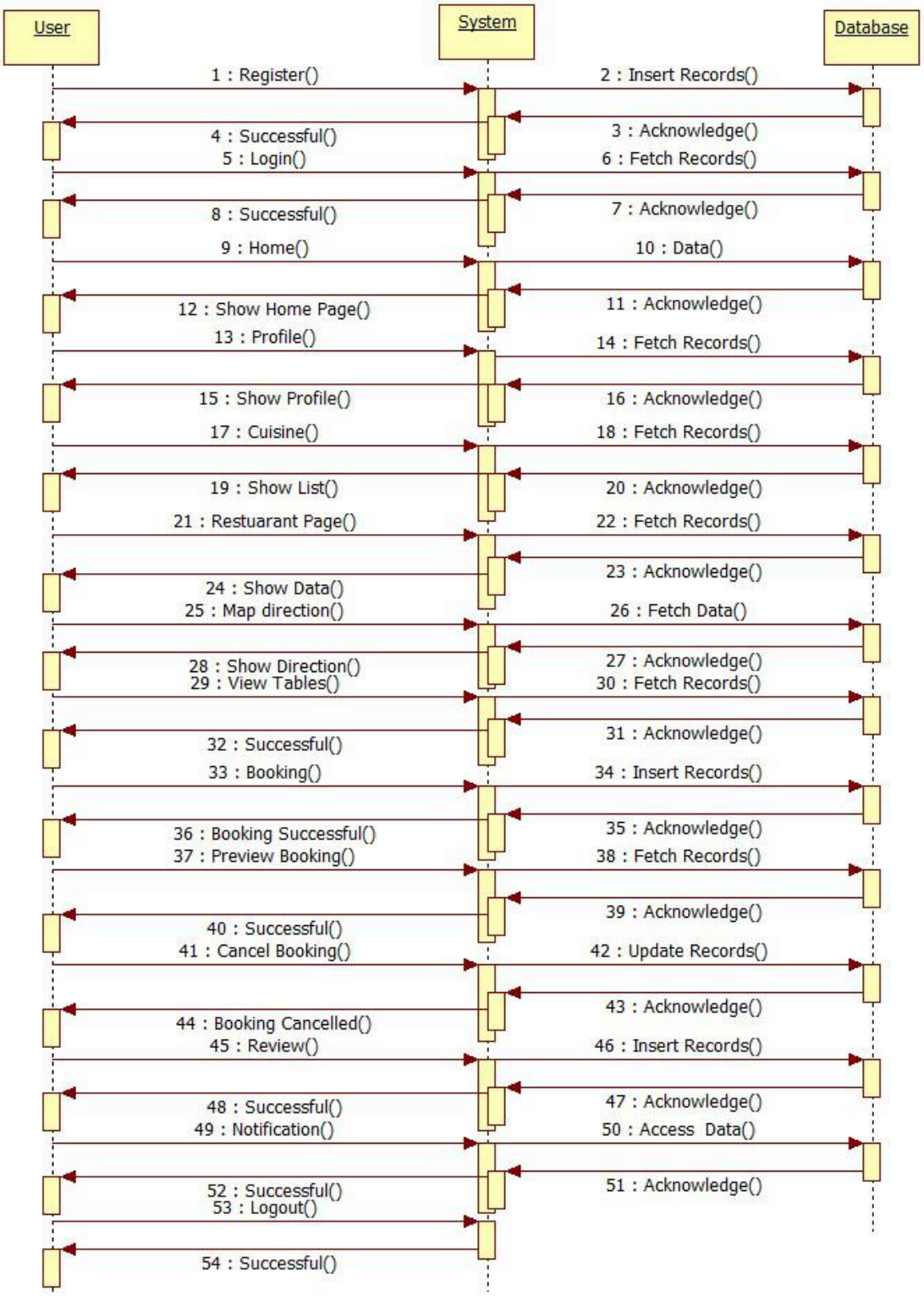
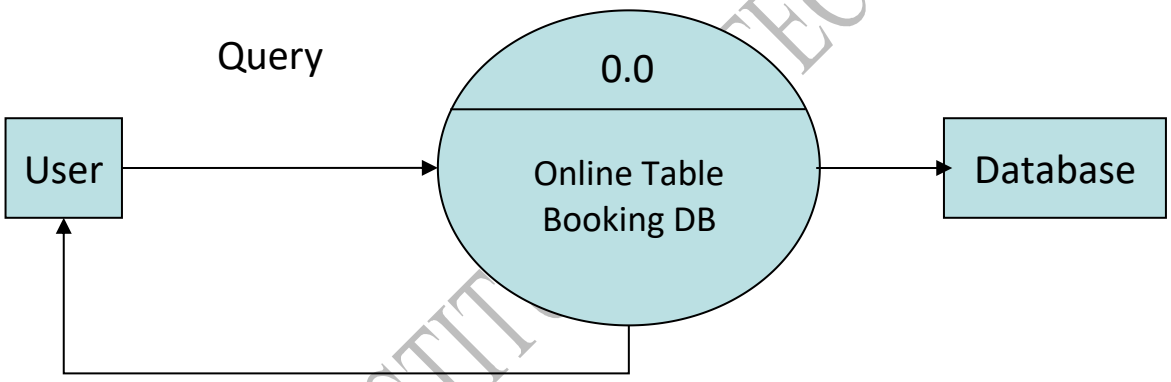


Fig. 4.3.3: Sequence Diagram of Android User

4.4 Functional Modeling

Data Flow Diagram

Data flow diagrams provide a graphical representation of how information moves between processes in a system. A Data Flow Diagram shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored.

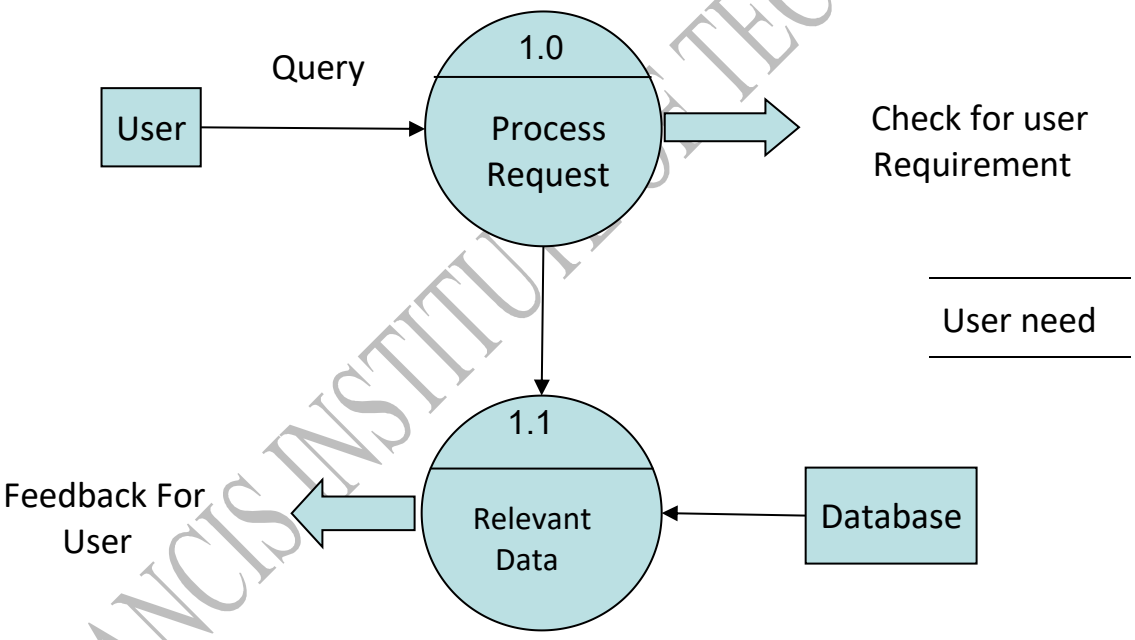


Level 0 DFD & Level 1 DFD

Fig. 4.4.1: Level 0 DFD

In this level 0 data flow diagram, the whole system is represented with the help of input, processing and output. The input to the online table booking system is the data entered by the end user. The Online table booking system should select a particular option and perform appropriate operation

In level 1 data flow diagram, the bubble “Online Table Booking System” is shown in more detail by various processes. The user needs to register to the system by providing the details and then



login to the system by providing the username and password. The user is provided with menu. The user can book the table, view current booking and can also cancel the reserved table. And when the user books a table an ID will be generated for that table. The admin can check the reserved tables of the booked tables with the help of the ID.

Fig. 4.4.2 : Level 1 DFD

Level 2 DFD

In level 2 data flow diagram, we will elaborate “Online Table booking system” in more detail. The user needs to register to the system by providing the details and then login to the system by providing the username and password. The user is provided with menu. The user can book the table, view current booking and can also cancel the reserved table. And when the user books a table an ID will be generated for that table. The user can view the live booked tables. The admin can check the booking details of the reserved tables with the help of the ID.

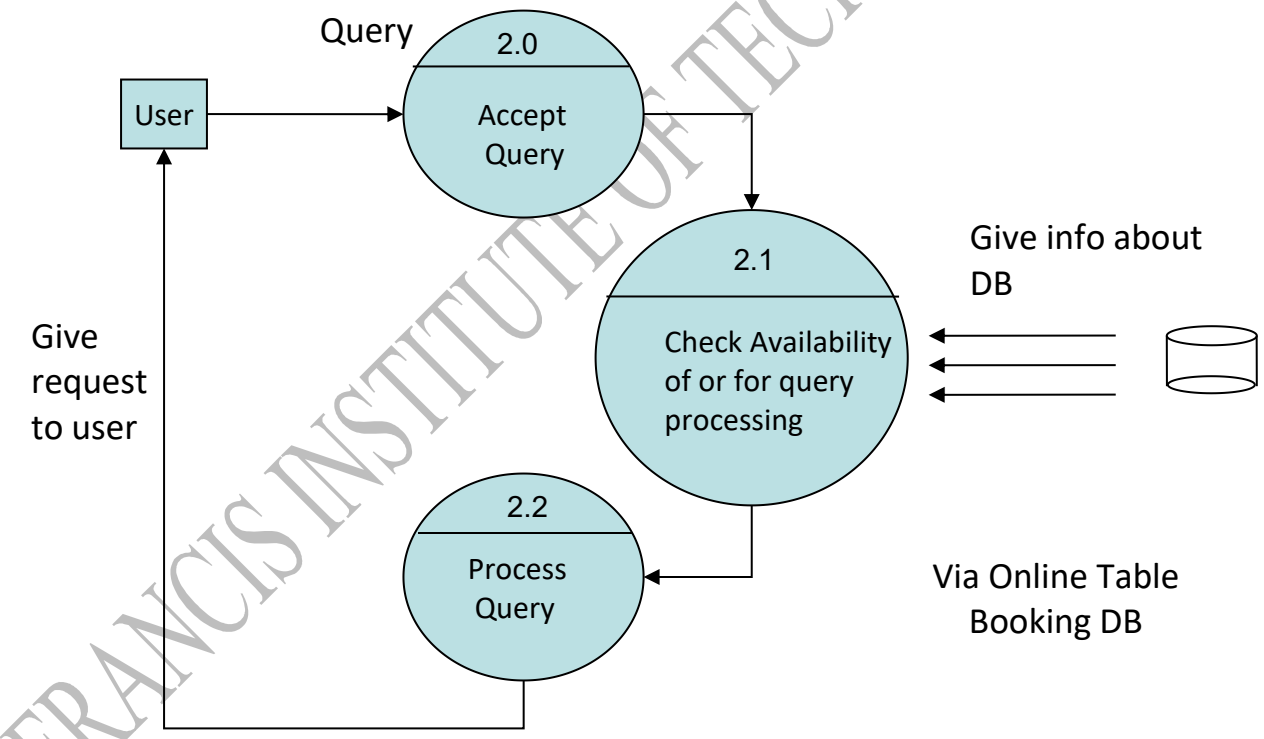


Fig. 4.4.3: Level 2 DFD

4.5 Timeline chart and WBS

Good planning spreads the necessary work over a reasonable period of time and allows everyone to work at a consistent, sustainable pace.

A Timeline is a clean and concise visual representation of a series of events. It helps to arrange large chunks of time and see the overall plan easily. A timeline is typically divided into chunks, each ending with a milestone.

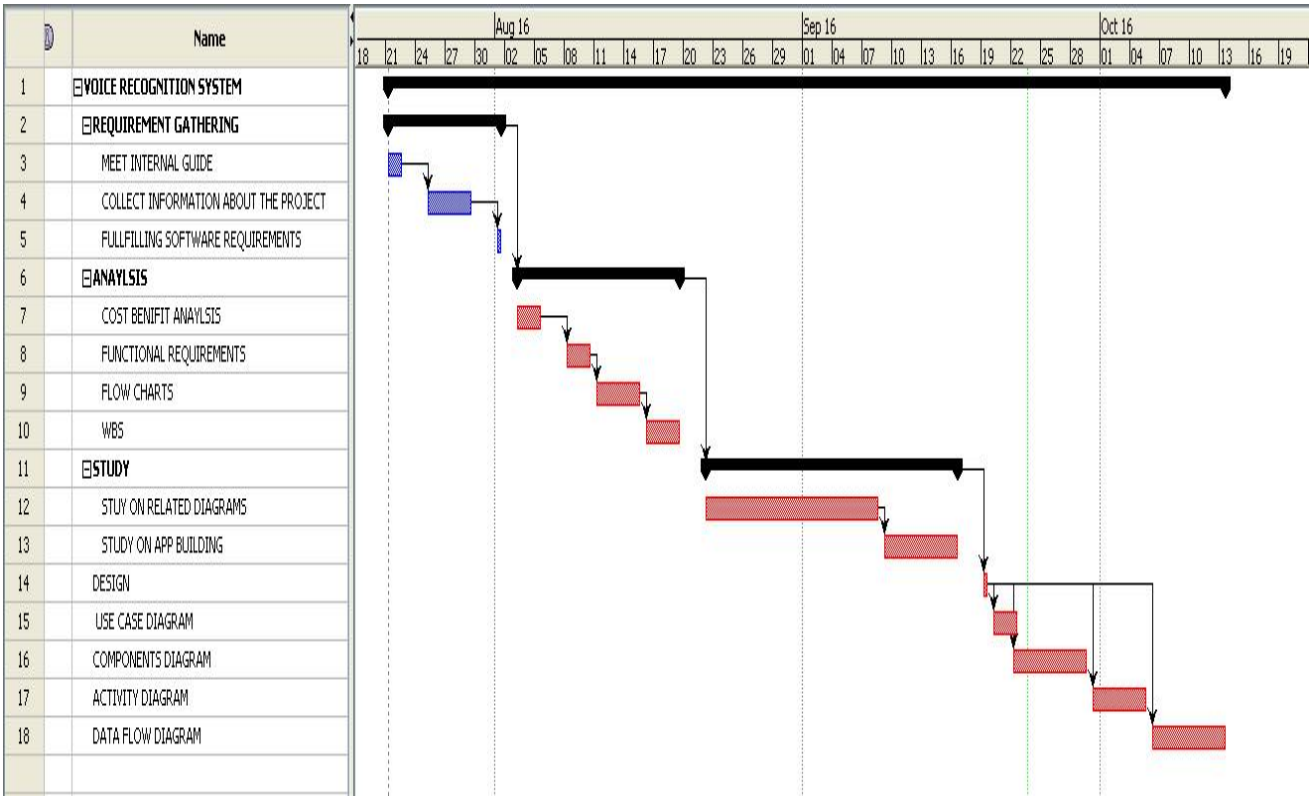


Fig. 4.5.1: Timeline Chart

Chapter 5

Design

5.1 Architectural Design:

The Architectural Design of the proposed system gives an overview of how the application will work. The system is divided into two parts for the customers and for the restaurant managers and the admin. Customers will access the system through android application and the managers of restaurant get to access through the website. It begins when the user logs in his credentials into the application. Once the user is logged in the user has to click on the button Book Table on the home page which will navigate the user to the page consisting of various cuisine's, the user on selecting one particular cuisine is then navigated to the map where the user can view the names of the restaurant that are located in the area where the user is placed. Once the user chooses the restaurant he will be asked to reserve a table with appropriate date and time. The user can also cancel the booked table and review the restaurant if the restaurant manager has checked the user out in the system more detailed explanation of how the application will work.

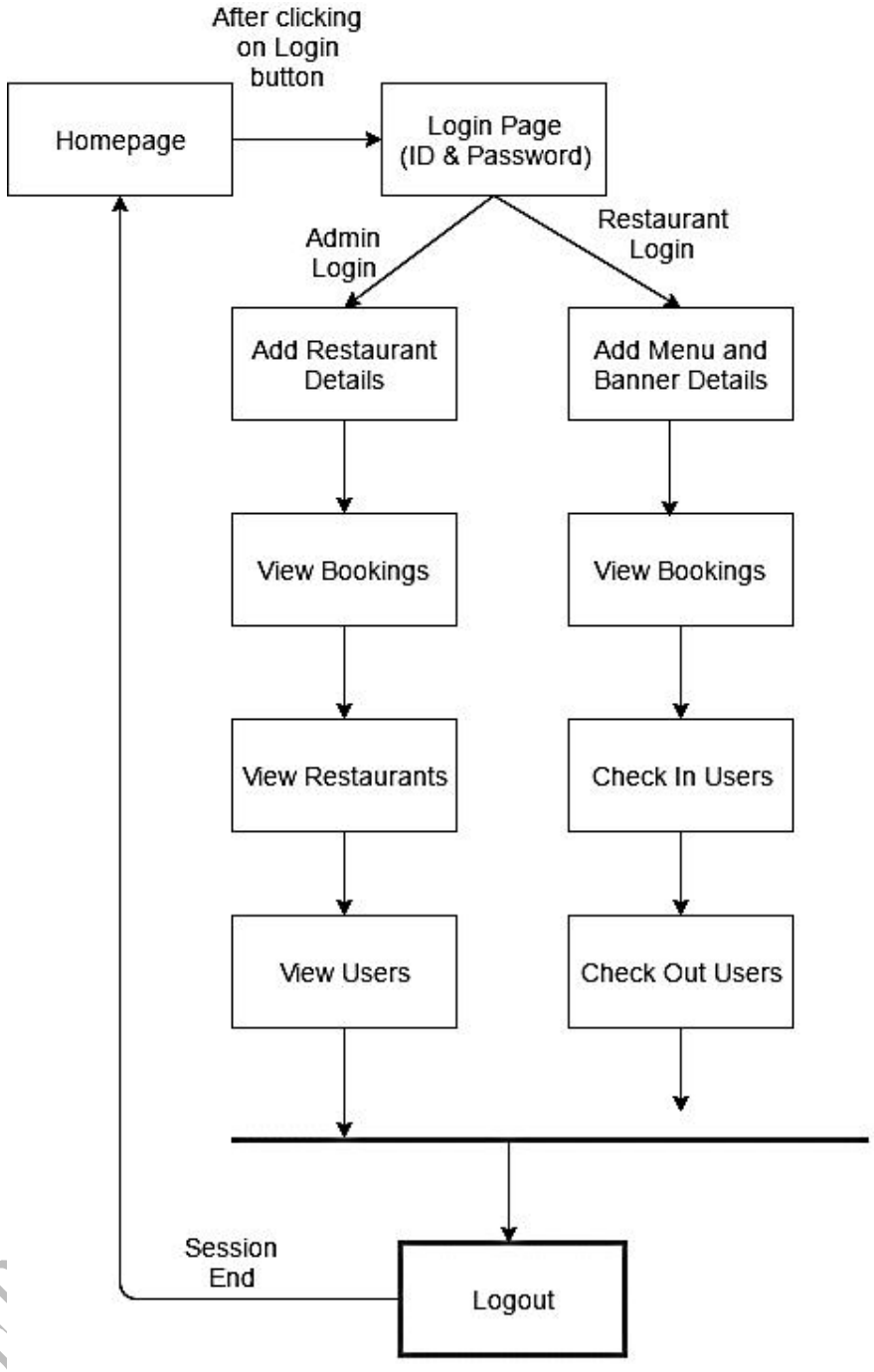


Fig.5.1.1 Flow Chart of Admin and Restaurant Manager

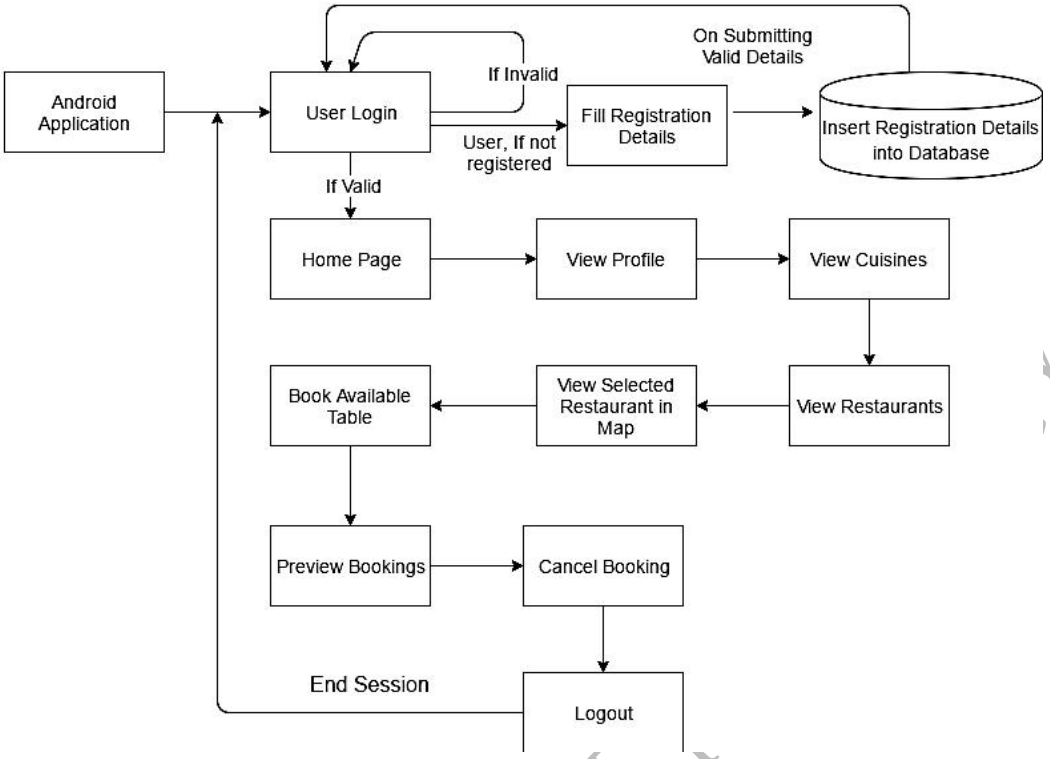


Fig.5.1.2 Flow Chart of Android User

Description of the Flowchart:

On launching the application, the user is asked to enter the username and password. If the user enters correct details than a successful login attempt is made, if not the details are checked with the details whether the user is user is registered or not. If not the user will be prompted to sign up. After the successful login attempt, the user can view the menu which contains: reserve table, Cancel reservation and View Current booking. If the user selects book table, the user needs to select various parameters such as location, date and time, no. of covers. After booking the table, the user gets an ID for that particular slot. If the user selects View Current booking, the user can view the representation of the seating space where a red spot represents that the table is reserved and a green spot represents that the table is not reserved and is free for booking. The admin is asked to just enter the username and password, the system verifies whether the details are present in the database or not. The admin just verifies the tables booked by a particular user and accordingly provides with the same. The admin verifies the user with the same ID generated by the user for booking a particular table. The admin can blacklist the users found playing again the policy.

5.2 User Interface Design:

For Android User:

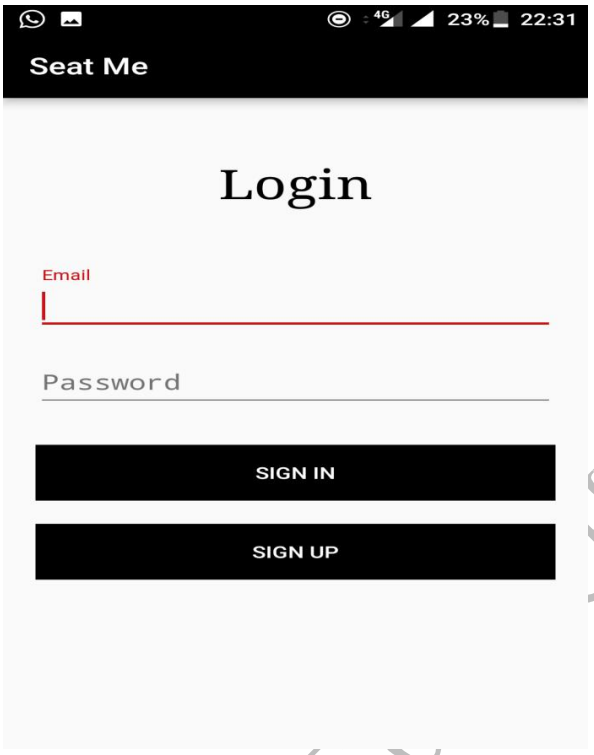


Fig 5.2.1: Sign In Screen

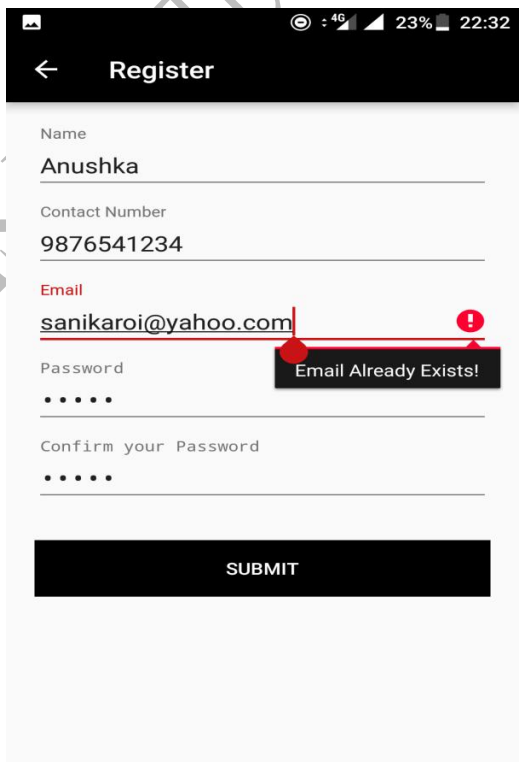


Fig 5.2.2 : Registration Page

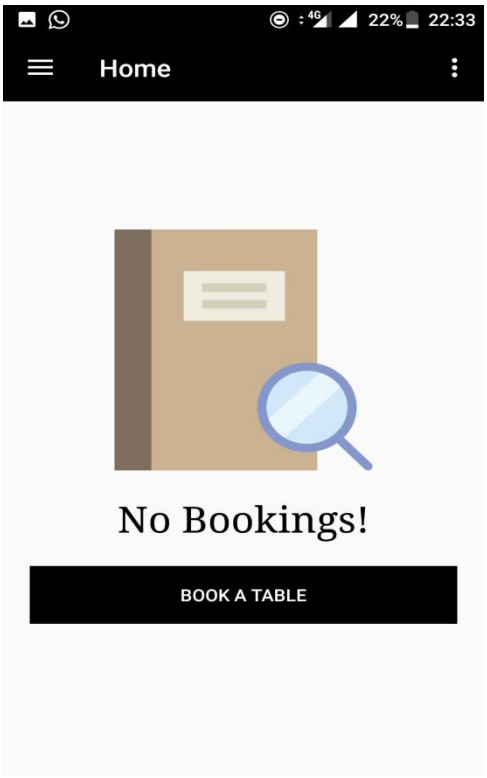


Fig 5.2.3 :Home Page

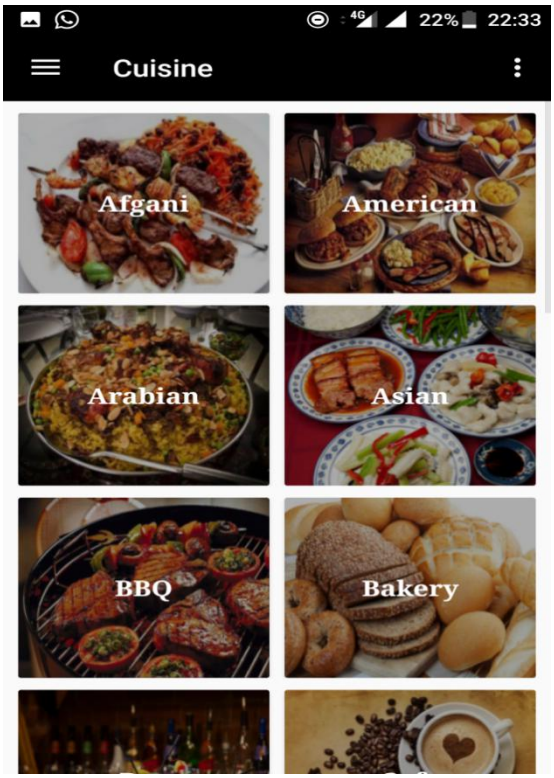


Fig 5.2.4: Choose a cuisine

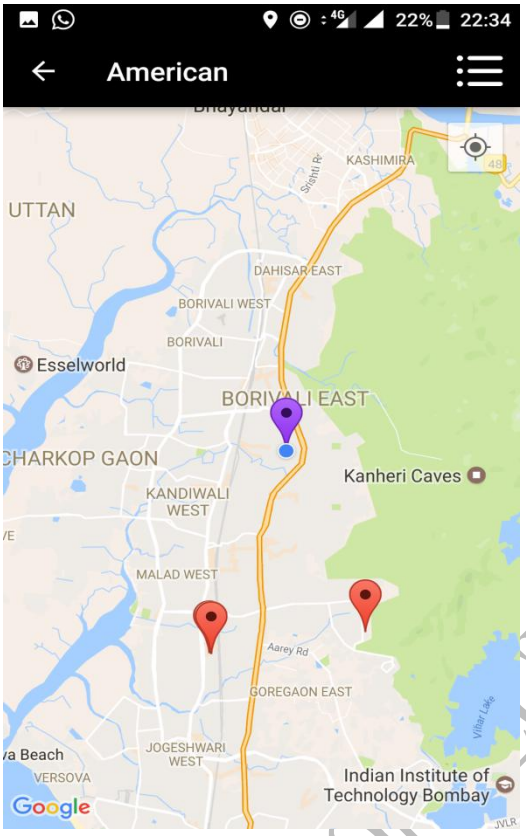


Fig 5.2.5: The restaurants that are in the location are plotted

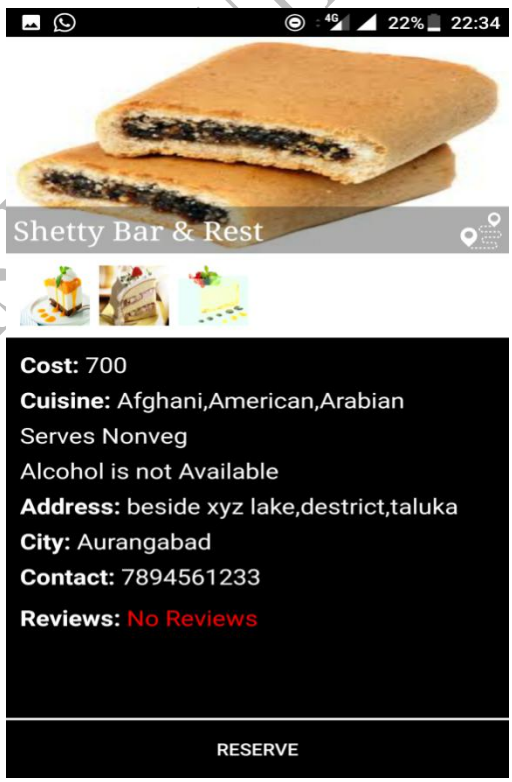


Fig 5.2.6: The user is navigated to restaurant page on clicking the restaurant in the map

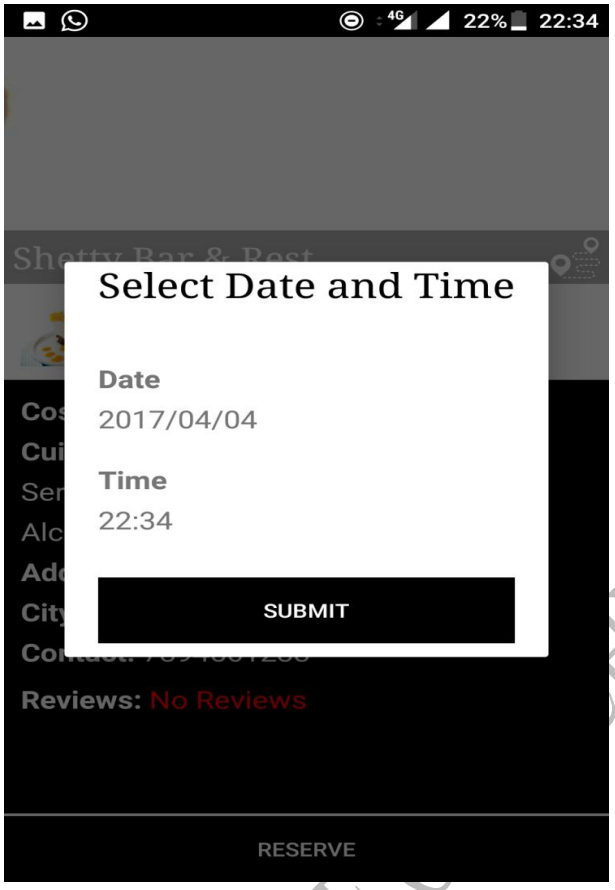


Fig 5.2.7: The user reserves the table with appropriate date and time

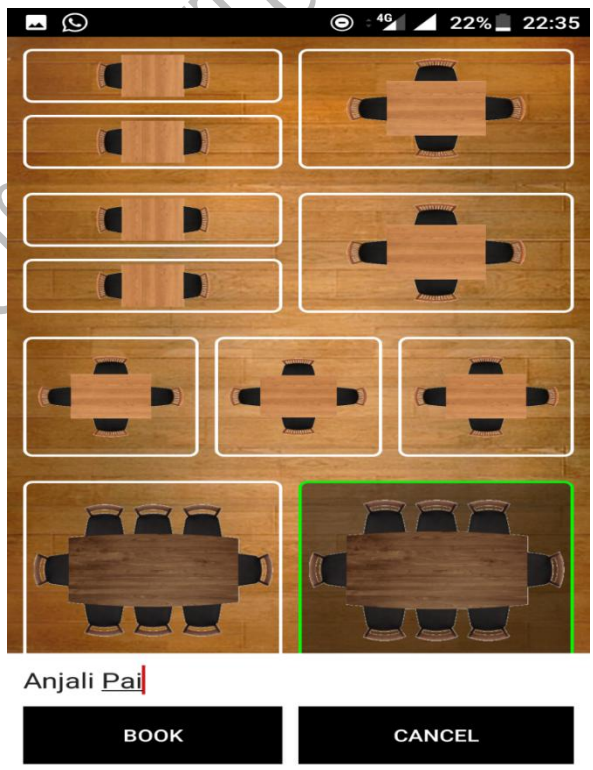


Fig 5.2.8: The user is asked to select the table and reserve the table with appropriate name.

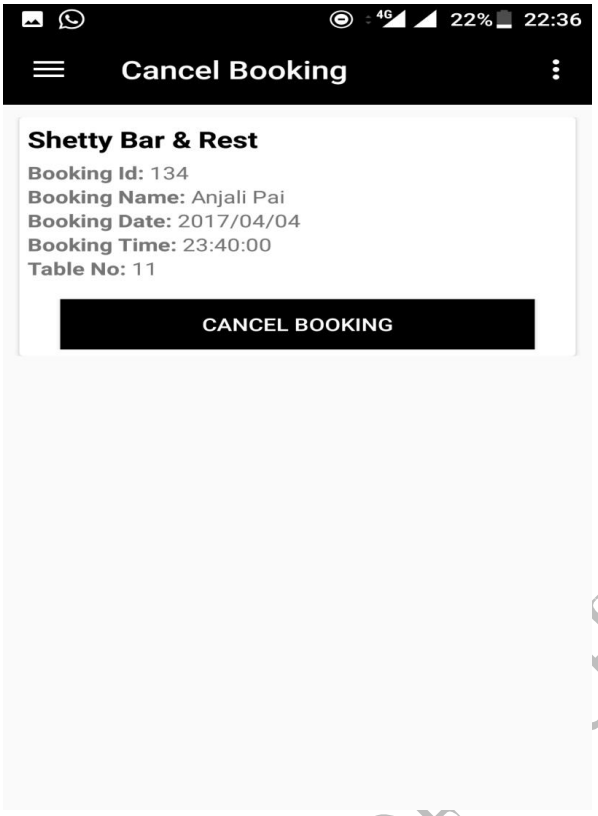


Fig 5.2.9: The user can also cancel the booking within one hour

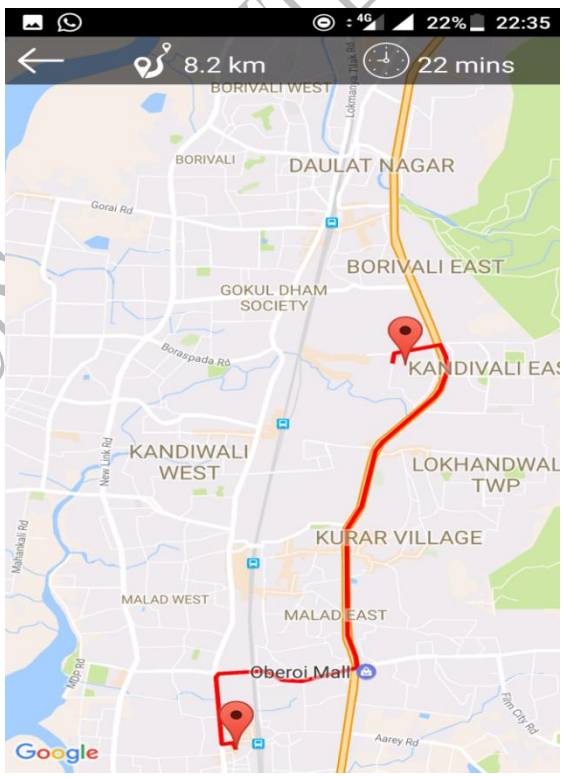


Fig 5.2.10: On clicking on navigate button the user is redirected to the map where he can view the distance between the restaurant from the current position.

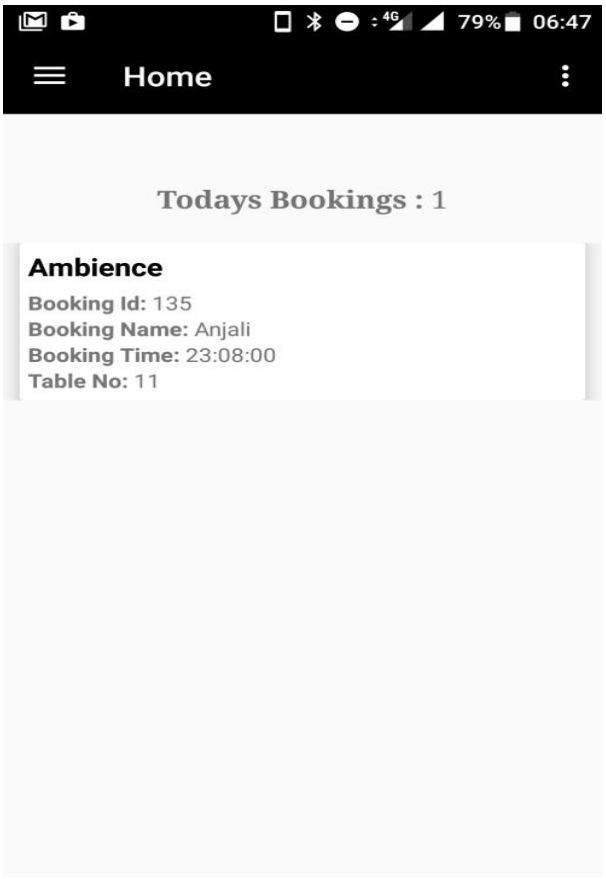


Fig 5.2.11 : Current bookings

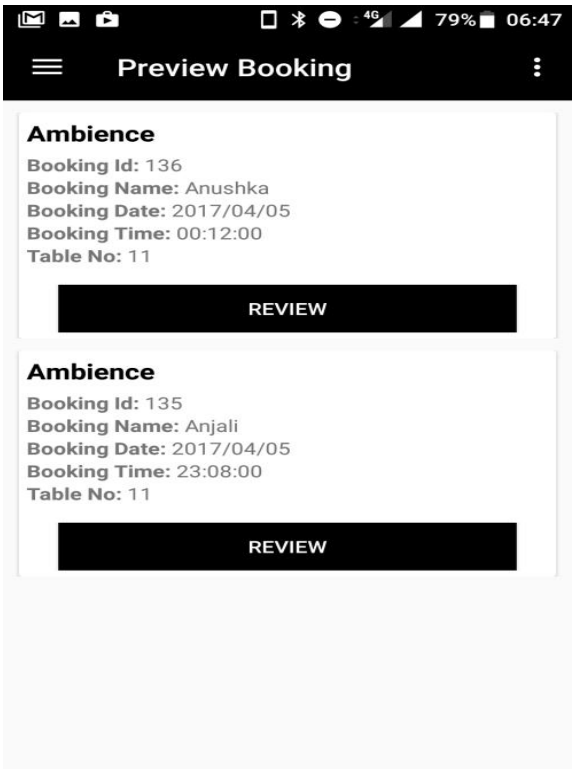


Fig 5.2.12 : Preview booking

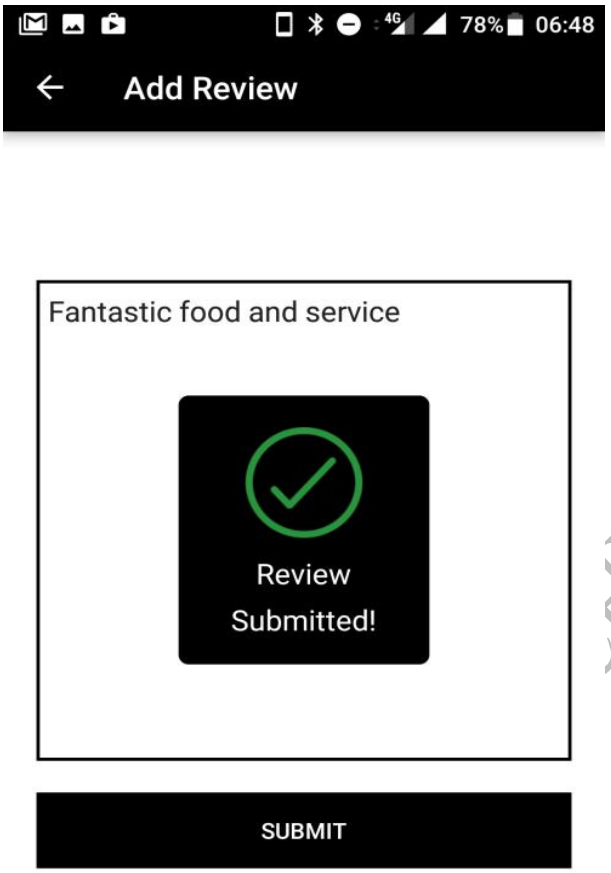


Fig 5.2.13: Review submitted

For Web Application

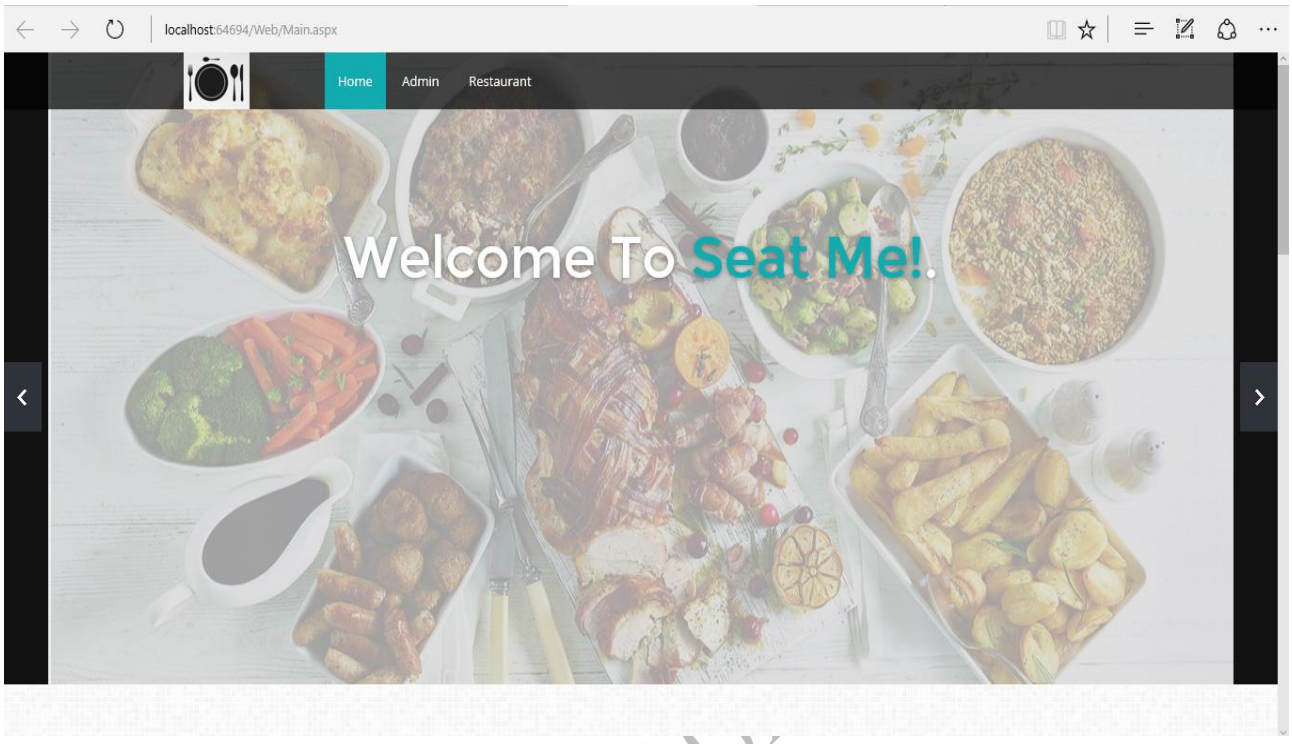


Fig 5.2.14 : Homepage

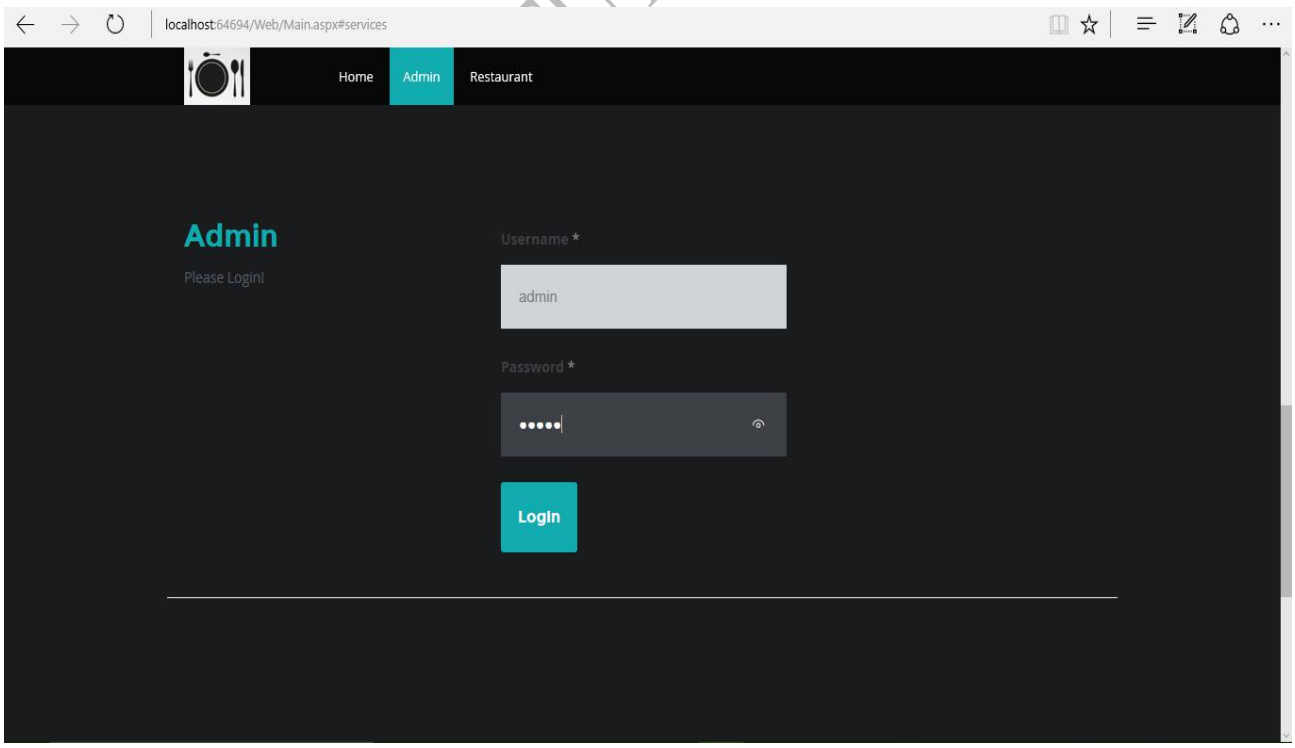


Fig 5.2.15 : Admin Log in page

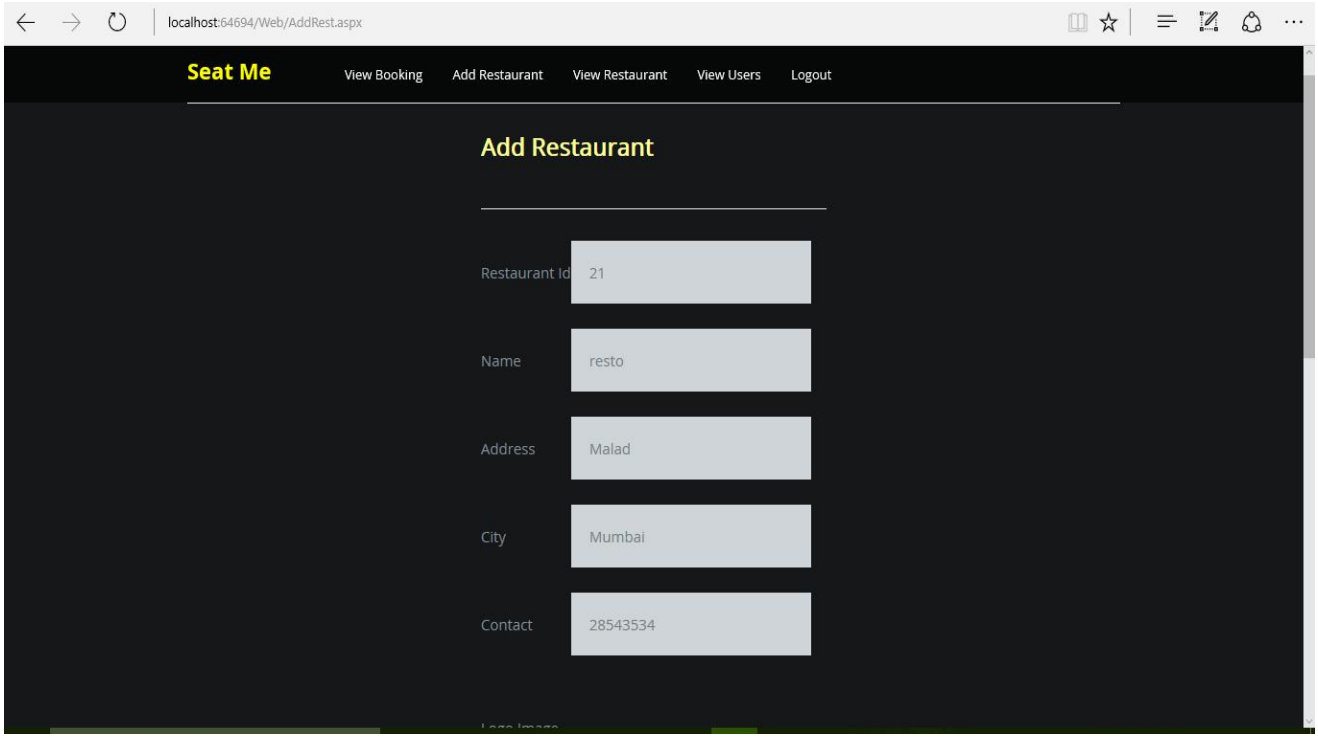


Fig 5.2.16 : Admin can add the restaurant

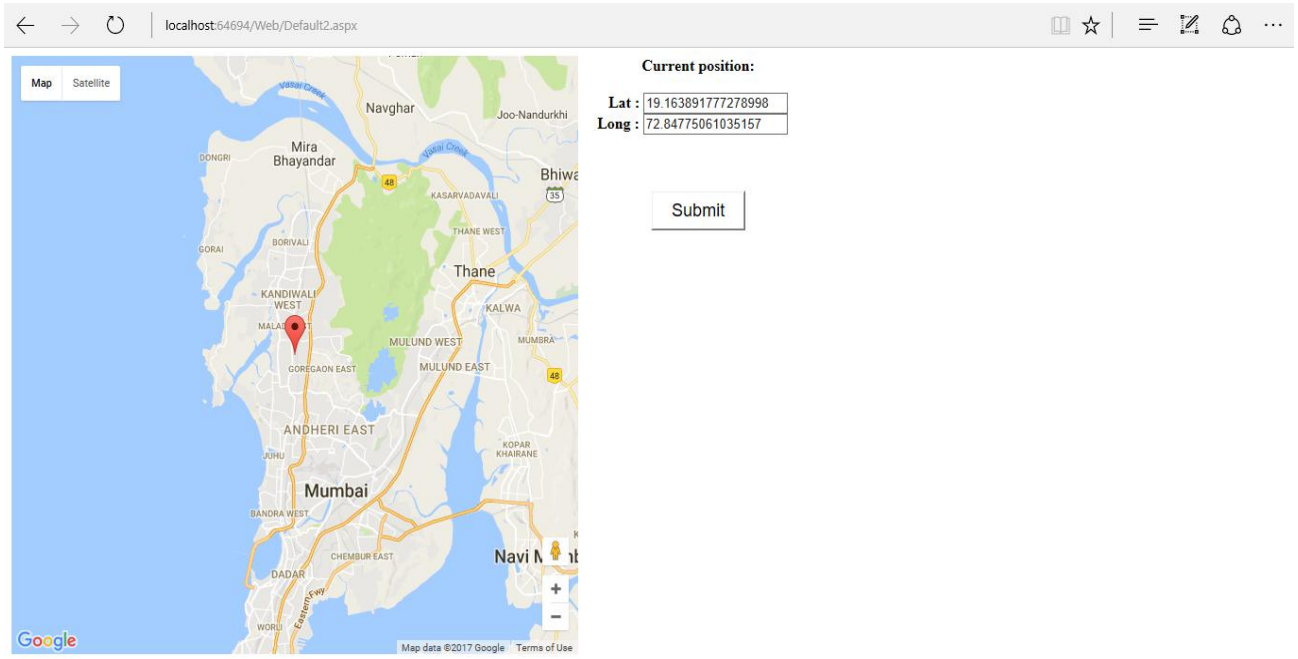


Fig 5.2.17 : Mapping of the restaurant on google map

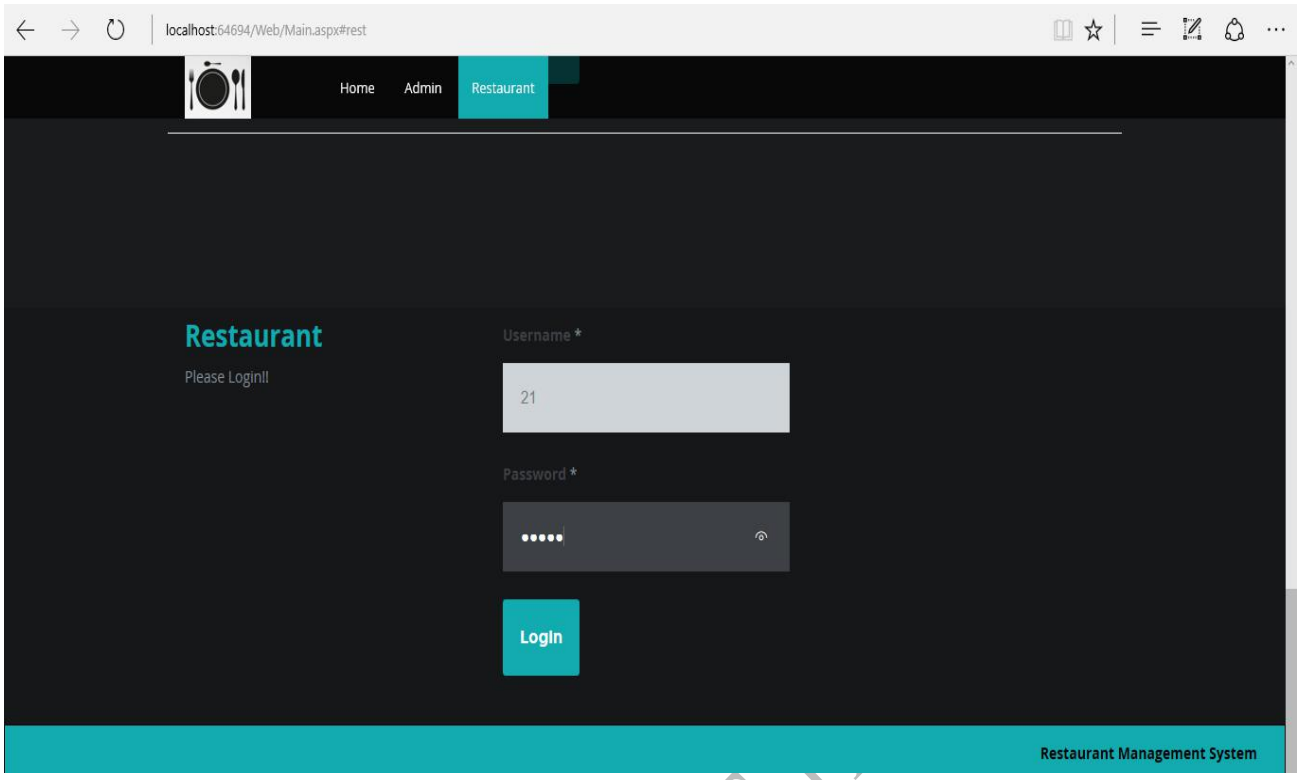


Fig 5.2.18 : Restaurant Manager log in

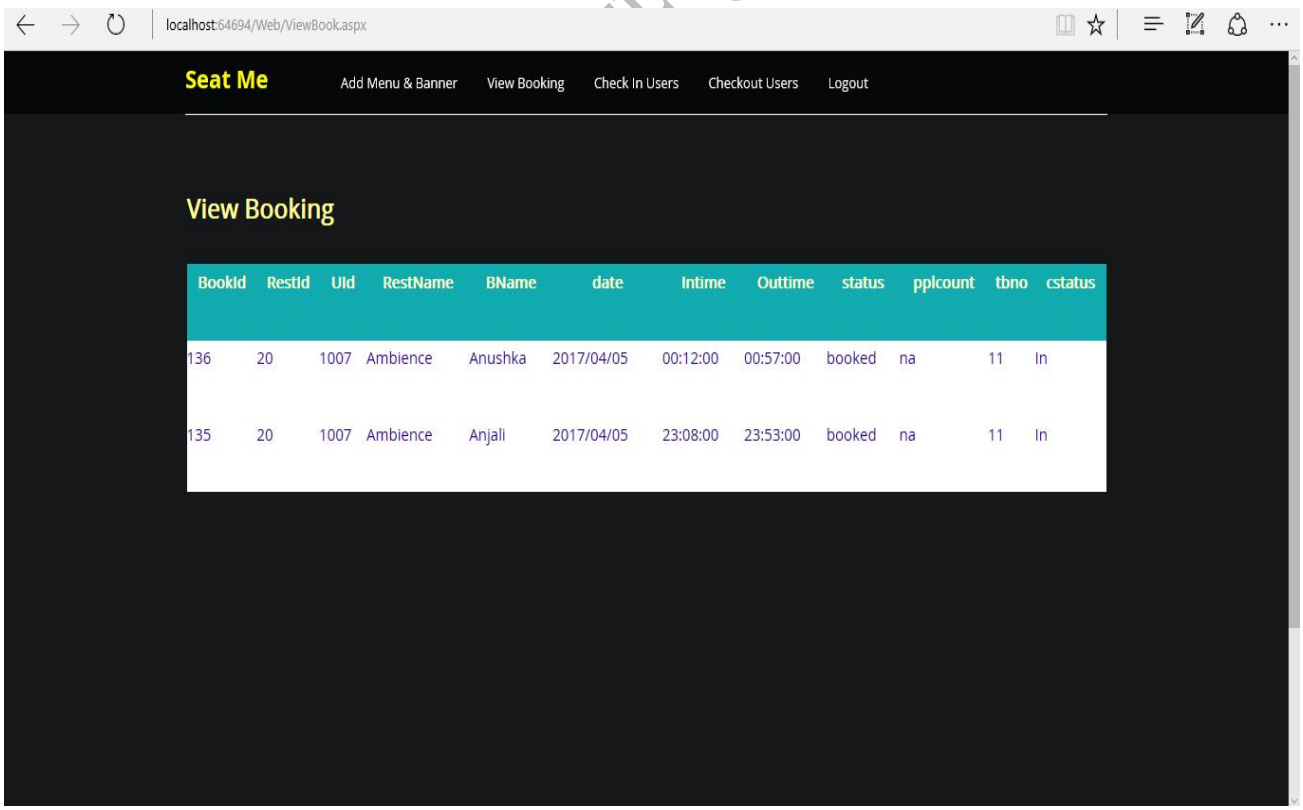


Fig 5.2.19 : Restaurant Manager can view the current booking

Chapter 6
Implementation

6.1 Project Implementation Technology

The Project Website is loaded in Visual Studio 2010 and application is loaded in Android Studio. We used Visual Studio and Android Studio for Design and coding of project. Created and maintained all databases into SQL Server 2008, in that we create tables, write query for store data or record of project.

OVERVIEW OF TECHNOLOGIES USED

Front End Technology

Microsoft .NET Framework

- The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet. The .NET Framework is designed to fulfill the following objectives:
- To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that guarantees safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.

Features of the Common Language Runtime

The common language runtime manages memory, thread execution, code execution, code safety verification, compilation, and other system services. These features are intrinsic to the managed code that runs on the common language runtime.

With regards to security, managed components are awarded varying degrees of trust, depending on a number of factors that include their origin (such as the Internet, enterprise network, or local computer). This means that a managed component might or might not be able to perform file-access operations, registry-access operations, or other sensitive functions, even if it is being used in the same active application.

The runtime enforces code access security. For example, users can trust that an executable embedded in a Web page can play an animation on screen or sing a song, but cannot access their personal data, file system, or network.

The runtime also accelerates developer productivity. For example, programmers can write applications in their development language of choice, yet take full advantage of the runtime, the class library, and components written in other languages by other developers. Any compiler vendor who chooses to target the runtime can do so. Language compilers that target the .NET Framework make the features of the .NET Framework available to existing code written in that language, greatly easing the migration process for existing applications.

.NET Framework Class Library

The .NET Framework class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is object oriented, providing types from which your own managed code can derive functionality. This not only makes the .NET Framework types easy to use, but also reduces the time associated with learning new features of the .NET Framework. In addition, third-party components can integrate seamlessly with classes in the .NET Framework.

For example, the Windows Forms classes are a comprehensive set of reusable types that vastly simplify Windows GUI development. If you write an ASP.NET Web Form application, you can use the Web Forms classes.

Active Server Pages.NET

ASP.NET is a programming framework built on the common language runtime that can be used on a server to build powerful Web applications. ASP.NET offers several important advantages over previous Web development models:

- **Enhanced Performance.**
- **World-Class Tool Support.**
- **Power and Flexibility**.
- **Simplicity.**
- **Security.**
- **Language Support**

BACK END TECHNOLOGY:

About Microsoft SQL Server

Microsoft SQL Server is a Structured Query Language (SQL) based, client/server relational database. Each of these terms describes a fundamental part of the architecture of SQL Server.

Database

A database is similar to a data file in that it is a storage place for data. Like a data file, a database does not present information directly to a user; the user runs an application that accesses data from the database and presents it to the user in an understandable format.

Relational Database

There are different ways to organize data in a database but relational databases are one of the most effective. Relational database systems are an application of mathematical set theory to the problem of effectively organizing data. In a relational database, data is collected into tables (called relations in relational theory).

When organizing data into tables, you can usually find many different ways to define tables. Relational database theory defines a process, normalization, which ensures that the set of tables you define will organize your data effectively.

Client/Server:

In a client/server system, the server is a relatively large computer in a central location that manages a resource used by many people. When individuals need to use the resource, they connect over the network from their computers, or clients, to the server.

Examples of servers are: In a client/server database architecture, the database files and DBMS software reside on a server. A communications component is provided so applications can run on separate clients and communicate to the database server over a network. The SQL Server communication component also allows communication between an application running on the server and SQL Server.

Server applications are usually capable of working with several clients at the same time. SQL Server can work with thousands of client applications simultaneously. The server has features to prevent the logical problems that occur if a user tries to read or modify data currently being used by others.

Structured Query Language (SQL)

To work with data in a database, you must use a set of commands and statements (language) defined by the DBMS software. There are several different languages that can be used with relational databases; the most common is SQL.

Both the American National Standards Institute (ANSI) and the International Standards Organization (ISO) have defined standards for SQL. Most modern DBMS products support the Entry Level of SQL-92, the latest SQL standard (published in 1992).

Middleware Technology

Active Data Objects.Net Overview

ADO.NET is an evolution of the ADO data access model that directly addresses user requirements for developing scalable applications. It was designed specifically for the web with scalability, statelessness, and XML in mind.

A Data Adapter is the object that connects to the database to fill the Dataset. Then, it connects back to the database to update the data there, based on operations performed while the Dataset held the data. In the past, data processing has been primarily connection-based. Now, in an effort to make multi-tiered apps more efficient, data processing is turning to a message-based approach that revolves around chunks of information.

When dealing with connections to a database, there are two different options: SQL Server .NET Data Provider (System.Data.SqlClient) and OLE DB .NET Data Provider (System.Data.OleDb). In these samples we will use the SQL Server .NET Data Provider. These are written to talk directly to Microsoft SQL Server. The OLE DB .NET Data Provider is used to talk to any OLE DB provider (as it uses OLE DB underneath).

1. ADO.NET is the next evolution of ADO for the .Net Framework.
2. ADO.NET was created with n-Tier, statelessness and XML in the forefront. Two new objects, the Dataset and Data Adapter, are provided for these scenarios. ADO.NET can be used to get data from a stream, or to store data in a cache for updates.

3. There is a lot more information about ADO.NET in the documentation.
4. Remember, you can execute a command directly against the database in order to do inserts, updates, and deletes. You don't need to first put data into a Dataset in order to insert, update, or delete it.

Introduction to Android

Android provides a rich application framework that allows you to build innovative apps and games for mobile devices in a Java language environment. The documents listed in the left navigation provide details about how to build apps using Android's various APIs.

If you're new to Android development, it's important that you understand the following fundamental concepts about the Android app framework:

Apps provide multiple entry points

Android apps are built as a combination of distinct components that can be invoked individually. For instance, an individual *activity* provides a single screen for a user interface, and a *service* independently performs work in the background.

From one component you can start another component using an *intent*. You can even start a component in a different app, such as an activity in a maps app to show an address. This model provides multiple entry points for a single app and allows any app to behave as a user's "default" for an action that other apps may invoke.

Apps adapt to different devices

Android provides an adaptive app framework that allows you to provide unique resources for different device configurations. For example, you can create different XML layout files for different screen sizes and the system determines which layout to apply based on the current device's screen size.

Application Fundamentals

Android apps are written in the Java programming language. The Android SDK tools compile your code along with any data and resource files into an APK: an Android package, which is an archive file with an .apk suffix. One APK file contains all the contents of an Android app and is the file that Android-powered devices use to install the app.

Once installed on a device, each Android app lives in its own security sandbox:

- The Android operating system is a multi-user Linux system in which each app is a different user.
- By default, the system assigns each app a unique Linux user ID (the ID is used only by the system and is unknown to the app). The system sets permissions for all the files in an app so that only the user ID assigned to that app can access them.
- Each process has its own virtual machine (VM), so an app's code runs in isolation from other apps.

App Components

App components are the essential building blocks of an Android app. Each component is a different point through which the system can enter your app. Not all components are actual entry points for the user and some depend on each other, but each one exists as its own entity and plays a specific role—each one is a unique building block that helps define your app's overall behavior. There are four different types of app components. Each type serves a distinct purpose and has a distinct lifecycle that defines how the component is created and destroyed.

Here are the four types of app components:

- **Activities**
- **Services**
- **Activating Components**
- **The Manifest File**

6.2 Working of the project:

Android

Login.java

```
public class Login extends AppCompatActivity{
    ProgressBar dprog;
    ImageView dimg;
    TextView dtxt1,dtxt2;
    Dialog dl;
    EditText user,pass;
    Button signin,signup;
    RelativeLayout ray;
    SharedPreferences sp;
    SharedPreferences.Editor editor;
    GPS_Tracker gps;
    ConnectivityManager cm;
    Boolean dialog_status=false;
    Timer timer;
    TimerTask timerTask;
    Handler handler=new Handler();
    Dialog int_d;
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        sp=getSharedPreferences("seatme", Context.MODE_PRIVATE);
        String str=sp.getString("uid","");
        stopService(new Intent(Login.this,Back_Service.class));
        startService(new Intent(Login.this,Back_Service.class));
        if(str.compareTo("")!=0)
        {
            Intent i=new Intent(Login.this,MainActivity.class);
            startActivity(i);
            finish();
        }
        else
```

```
{
    setContentView(R.layout.login);
    gps=new GPS_Tracker(this,Login.this);
    Boolean ans=weHavePermissionforGPS();
    if(!ans)
    {
        requestforGPSPermissionFirst();
    }
    timer=new Timer();
    init_timer();
    timer.schedule(timerTask,0,1000);
    user = (EditText) findViewById(R.id.user);
    pass = (EditText) findViewById(R.id.pass);
    signin = (Button) findViewById(R.id.signin);
    signup = (Button) findViewById(R.id.signup);
    ray = (RelativeLayout) findViewById(R.id.loginray);
    signin.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (user.getText().toString().compareTo("") != 0 ||
pass.getText().toString().compareTo("") != 0)
            {
                if (user.getText().toString().compareTo("") != 0)
                {
                    if (pass.getText().toString().compareTo("") != 0)
                    {
                        if(gps.canGetLocation())
                        {
                            new login().execute(user.getText().toString(),
pass.getText().toString());
                        }
                        else
                        {
                            Snackbar snack = Snackbar.make(v, "Enable
```

```
GPS & click on Sign In", Snackbar.LENGTH_SHORT);
                                        View vs = snack.getView();
                                        TextView txt = (TextView)
vs.findViewById(android.support.design.R.id.snackbar_text);
                                        txt.setTextColor(Color.RED);
                                        snack.show();
                                    }
                                }
                                else
                                {
                                    pass.setError("Enter Password");
                                    pass.requestFocus();
                                }
                            }
                            else
                            {
                                user.setError("Enter Email");
                                user.requestFocus();
                            }
                        }
                        else
                        {
                            Snackbar snack = Snackbar.make(v, "Enter Email &
Password to Proceed", Snackbar.LENGTH_SHORT);
                            View vs = snack.getView();
                            TextView txt = (TextView)
vs.findViewById(android.support.design.R.id.snackbar_text);
                            txt.setTextColor(Color.RED);
                            snack.show();
                        }
                    }
                });
                signup.setOnClickListener(new View.OnClickListener() {
                    @Override
```

```
                public void onClick(View v) {
                    Intent i = new Intent(Login.this, Register.class);
                    startActivity(i);
                }
            });
        }}
    public void load()
    {
        dl=new Dialog(Login.this,R.style.AppTheme);
        dl.requestWindowFeature(Window.FEATURE_NO_TITLE);
        dl.setContentView(R.layout.loading_dailog);
        dl.setCancelable(false);
        dl.getWindow().setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));
        dprog= (ProgressBar) dl.findViewById(R.id.ld_progess);
        dimg= (ImageView) dl.findViewById(R.id.ld_img);
        dtxt1= (TextView) dl.findViewById(R.id.ld_text1);
        dtxt2= (TextView) dl.findViewById(R.id.ld_text2);
        dprog.setVisibility(View.VISIBLE);
        dimg.setImageResource(R.drawable.tick);
        dimg.setVisibility(View.GONE);
        dtxt1.setText("Loading..");
        dtxt2.setText("Please Wait!");
        dl.show();
    }
    public class login extends AsyncTask<String,JSONObject,String>
    {
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            load();
        }
        @Override
        protected String doInBackground(String... params) {
```

```
        String a="back";
        RestAPI api=new RestAPI();
        try {
            JSONObject json=api.login(params[0],params[1]);
            JSONParse jp=new JSONParse();
            a=jp.parse(json);
        } catch (Exception e) {
            a=e.getMessage();
        }
        return a;
    }
    @Override
    protected void onPostExecute(String s) {
        super.onPostExecute(s);
        if(s.compareTo("false")==0)
        {
            dprog.setVisibility(View.GONE);
            dimg.setImageResource(R.drawable.wrong);
            dimg.setVisibility(View.VISIBLE);
            dtxt1.setText("Invalid");
            dtxt2.setText("Credentials!");
            new Handler().postDelayed(new Runnable() {
                @Override
                public void run() {
                    dl.cancel();
                    user.setText("");
                    pass.setText("");
                    user.requestFocus();
                }
            },3000);
        }
        else
        {
            if(s.contains("*"))
```

```
                {
                    String temp[]=s.split("\\*");
                    editor =sp.edit();
                    editor.putString("uid",temp[0]);
                    editor.putString("name",temp[1]);
                    editor.commit();
                    Intent i=new Intent(Login.this,MainActivity.class);
                    startActivity(i);
                    finish();
                }
                else {
                    dl.cancel();
                    Toast.makeText(Login.this, s, Toast.LENGTH_SHORT).show();
                }
            }
        }
    }

    private boolean weHavePermissionforGPS()
    {
        return ContextCompat.checkSelfPermission(this,
android.Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED;
    }
    private void requestforGPSPermissionFirst()
    {
        if (ActivityCompat.shouldShowRequestPermissionRationale(this,
android.Manifest.permission.ACCESS_FINE_LOCATION))
        {
            requestForResultContactsPermission();
        }
        else
        {
            requestForResultContactsPermission();
```

```
        }
    }
    private void requestForResultContactsPermission()
    {
        ActivityCompat.requestPermissions(this, new
String[]{android.Manifest.permission.ACCESS_FINE_LOCATION}, 111);
    }
    public Boolean checkinternet()
    {
        try{
        cm = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
        boolean isConnected = activeNetwork != null &&
activeNetwork.isConnectedOrConnecting();
        if (isConnected) {
            return   true;
        } else {
            return false;
        }
        }
        catch (Exception e)
        {
            return true;
        }
    }
    public void call_dailog()
    {
        AlertDialog.Builder ad=new AlertDialog.Builder(Login.this);
        ad.setCancelable(false);
        ad.setTitle("NO Internet Connection");
        ad.setMessage("Please check your Interent Connection!..Cannot Proceed
without it!");
        ad.setNeutralButton("Try Again", new DialogInterface.OnClickListener() {
```

```
            @Override
            public void onClick(DialogInterface dialog, int which) {
                    dialog.cancel();
                    dialog_status=false;
            }
        });
        int_d=ad.create();
        int_d.show();
    }
    public void init_timer()
    {
        timerTask=new TimerTask() {
            @Override
            public void run() {
                handler.post(new Runnable()
                {
                    @Override
                    public void run() {
                        if (checkinternet())
                        {
                            try {
                                int_d.cancel();
                            } catch (Exception e) {
                            }
                            dialog_status = false;
                        }
                        else
                        {
                            if(!dialog_status)
                            {
                                dialog_status = true;
                                call_dailog();
                            }
                        }
```

```
                    }
                });
            }
        };
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        try
        {
            timer.cancel();
        }
        catch (Exception e){}
    }
}
```

Booking.java

```
public class Bookings extends Fragment{

    public static TabLayout tabLayout;
    public static ViewPager viewPager;
    public static int int_items = 3 ;
    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View v=inflater.inflate(R.layout.bookings,container,false);
        tabLayout = (TabLayout) v.findViewById(R.id.tabs);
        viewPager = (ViewPager) v.findViewById(R.id.viewpager);
        viewPager.setAdapter(new MyAdapter(getChildFragmentManager()));
        tabLayout.post(new Runnable() {
            @Override
            public void run() {
                tabLayout.setupWithViewPager(viewPager);
            }
```

```
        });
        return v;
    }
    class MyAdapter extends FragmentPagerAdapter {
        public MyAdapter(FragmentManager fm)
        {
            super(fm);
        }
        @Override
        public Fragment getItem(int position)
        {
            switch (position)
            {
                case 0 : return new Current_Frag();
                case 1 : return new Previous_Frag();
                case 2 : return new Cancel_Frag();
            }
            return null;
        }
        @Override
        public int getCount() {

            return int_items;

        }
        @Override
        public CharSequence getPageTitle(int position) {
            switch (position){
                case 0 :
                    return "Current";
                case 1 :
                    return "Previous";
                case 2 :
                    return "Cancelled";
```

```
            }
            return null;
        }
    }
}
```

Main_Activity.java

```
public class MainActivity extends AppCompatActivity
{
    SharedPreferences sp;
    SharedPreferences.Editor editor;
    private DrawerLayout mDrawerLayout;
    private ListView mDrawerList;
    private ActionBarDrawerToggle mDrawerToggle;
    private CharSequence mDrawerTitle;
    private CharSequence mTitle;
    ArrayList<String> Newdraweritems;
    private DrawerlistAdapter adapter;
    String[] titlearray=new String[]{"Home","Profile","Cuisine","Bookings","Preview
Booking","Cancel Booking","Reviews","Logout"};
    int[] titleicon=new
int[]{R.drawable.home,R.drawable.user,R.drawable.cuisine,R.drawable.bookings,R.drawabl
e.previewb,R.drawable.cancelb,R.drawable.review,R.drawable.logout};
    Timer timer;
    TimerTask task;
    Handler handler=new Handler();
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sp=getSharedPreferences("seatme", Context.MODE_PRIVATE);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
```

```
mTitle = mDrawerTitle = getTitle();
mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
mDrawerList = (ListView) findViewById(R.id.listView1);
Boolean ans=weHavePermissionforGPS();
if(!ans)
{
    requestforGPSPermissionFirst();
}
else
{
    GPS_Tracker gps=new GPS_Tracker(this,MainActivity.this);
    gps.getLocation();
    if(!gps.canGetLocation())
    {
        Toast.makeText(this, "Enable GPS", Toast.LENGTH_SHORT).show();
    }
}
timer=new Timer();
init_timer();
timer.schedule(task,0,500);
Newdraweritems = new ArrayList<String>();
for(int i=0;i<titlearray.length;i++)
{
    Newdraweritems.add(titlearray[i]+","+titleicon[i]);
}
mDrawerList.setOnItemClickListener(new SlideMenuClickListener());
// setting the nav drawer list adapter
adapter = new DrawerlistAdapter(getApplicationContext(),
        Newdraweritems);
mDrawerList.setAdapter(adapter);
// enabling action bar app icon and behaving it as toggle button
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
getSupportActionBar().setHomeButtonEnabled(true);
```

```
        mDrawerToggle = new ActionBarDrawerToggle(this, mDrawerLayout,toolbar,
                R.string.app_name, // nav drawer open - description for accessibility
                R.string.app_name // nav drawer close - description for accessibility
        ) {
            public void onDrawerClosed(View view) {
                try {
                    getSupportActionBar().setTitle(mTitle);
                    // calling onPrepareOptionsMenu() to show action bar icons
                    invalidateOptionsMenu();
                }catch (Exception e){}
            }

            public void onDrawerOpened(View drawerView) {
                try {
                    getSupportActionBar().setTitle(mDrawerTitle);
                    // calling onPrepareOptionsMenu() to hide action bar icons
                    invalidateOptionsMenu();
                }catch (Exception e){}
            }
        };
        mDrawerLayout.setDrawerListener(mDrawerToggle);

        if (savedInstanceState == null) {
            // on first time display view for first nav item
            displayView(0);
        }
    }

    private class SlideMenuClickListener implements
            ListView.OnItemClickListener {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position,
                                long id) {
            // display view for selected nav drawer item
```

```
            displayView(position);
        }
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // toggle nav drawer on selecting action bar app icon/title
        if (mDrawerToggle.onOptionsItemSelected(item)) {
            return true;
        }
        // Handle action bar actions click
        switch (item.getItemId()) {
            case R.id.logout:
                logout();
            default:
                return super.onOptionsItemSelected(item);
        }
    }

    @Override
    public boolean onPrepareOptionsMenu(Menu menu) {
        // if nav drawer is opened, hide the action items
        boolean drawerOpen = mDrawerLayout.isDrawerOpen(mDrawerList);
        menu.findItem(R.id.logout).setVisible(!drawerOpen);
        return super.onPrepareOptionsMenu(menu);
    }
    private void displayView(int position) {
        // update the main content by replacing fragments
        Fragment fragment = null;
        switch (position) {
```

```
            case 0:
                fragment = new Home();
                break;
            case 1:
                fragment = new Profile();
                break;
            case 2:
                fragment = new Cuisine();
                break;
            case 3:
                fragment = new Bookings();
                break;
            case 4:
                fragment = new Preview_Booking();
                break;
            case 5:
                fragment = new Cancel_Booking();
                break;
            case 6:
                fragment = new Review_List();
                break;
            case 7:
                logout();
                break;
            default:
                break;
        }
        if (fragment != null) {
            FragmentManager fragmentManager = getSupportFragmentManager();
            fragmentManager.beginTransaction()
                    .replace(R.id.frameid, fragment).commit();

            // update selected item and title, then close the drawer
            mDrawerList.setItemChecked(position, true);
```

```
            mDrawerList.setSelection(position);
            setTitle(titlearray[position]);
            mDrawerLayout.closeDrawer(mDrawerList);
        } else {
            // error in creating fragment
            Log.e("MainActivity", "Error in creating fragment");
        }
    }
    @Override
    public void setTitle(CharSequence title) {
        mTitle = title;
        getSupportActionBar().setTitle(mTitle);
    }

    @Override
    protected void onPostCreate(Bundle savedInstanceState) {
        super.onPostCreate(savedInstanceState);
        // Sync the toggle state after onRestoreInstanceState has occurred.
        mDrawerToggle.syncState();
    }
    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        // Pass any configuration change to the drawer toggls
        mDrawerToggle.onConfigurationChanged(newConfig);
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        try{
        timer.cancel();}
        catch (Exception e){}
    }
    public    void logout()
```

```
        {
            editor=sp.edit();
            editor.putString("uid","");
            editor.putString("name","");
            editor.commit();
            Intent i=new Intent(MainActivity.this,Login.class);
            startActivity(i);
            finish();
        }
        private boolean weHavePermissionforGPS()
        {
            return ContextCompat.checkSelfPermission(this,
android.Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED;
        }
        private void requestforGPSPermissionFirst()
        {
            if (ActivityCompat.shouldShowRequestPermissionRationale(this,
android.Manifest.permission.ACCESS_FINE_LOCATION))
            {
                requestForResultContactsPermission();
            }
            else
            {
                requestForResultContactsPermission();
            }
        }
        private void requestForResultContactsPermission()
        {
            ActivityCompat.requestPermissions(this, new
String[]{android.Manifest.permission.ACCESS_FINE_LOCATION}, 111);
        }
        public void init_timer()
        {
```

```
            task=new TimerTask() {
                @Override
                public void run() {


                    handler.post(new Runnable() {
                        @Override
                        public void run() {
                            String str=sp.getString("newbook","");
                            if(str.compareTo("")!=0)
                            {
                                editor=sp.edit();
                                editor.putString("newbook","");
                                editor.commit();
                                Fragment fragment = new Cuisine();
                                FragmentManager fragmentManager =
getSupportFragmentManager();
                                fragmentManager.beginTransaction().replace(R.id.frameid,
fragment).commit();
                                mDrawerList.setItemChecked(2, true);
                                mDrawerList.setSelection(2);
                                setTitle(titlearray[2]);
                                mDrawerLayout.closeDrawer(mDrawerList);
                            }
                        }
                    });


                }
            };
        }}
```

6.3 Feasibility Report

Feasibility Study is a high level capsule version of the entire process intended to answer a number of questions like: What is the problem? Is there any feasible solution to the given problem? Is the problem even worth solving? Feasibility study is conducted once the problem clearly understood.

The following feasibilities are considered for the project in order to ensure that the project is variable and it does not have any major obstructions. Feasibility study encompasses the following things:

- Technical Feasibility
- Economic Feasibility
- Operational Feasibility

In this phase, we study the feasibility of all proposed systems, and pick the best feasible solution for the problem. The feasibility is studied based on three main factors as follows.

Technical Feasibility

In this step, we verify whether the proposed systems are technically feasible or not. i.e., all the technologies required to develop the system are available readily or not.

The system can be feasible because of the following grounds:

- All necessary technology exists to develop the system.
- This system is too flexible and it can be expanded further.
- This system can give guarantees of accuracy, ease of use, reliability and the data security.
- This system can give instant response to inquire.

Our project is technically feasible because, all the technology needed for our project is readily available.

Operating System	**:**	Windows 7(For PC) & Android v4.4 or Higher (For Android Devices)
Languages	**:**	Asp.Net with C# & Java
Database System	**:**	MS-SQL Server 2008
Documentation Tool	**:**	MS - Word 2013

Economic Feasibility

Economically, this project is completely feasible because it requires no extra financial investment and with respect to time, it's completely possible to complete this project in 6 months.

In this issue, we should consider:

- The cost to conduct a full system investigation.
- The cost of h/w and s/w for the class of application being considered.
- The development tool.
- The cost of maintenance etc...

Our project is economically feasible because the cost of development is very minimal when compared to financial benefits of the application.

Operational Feasibility

In this step, we verify different operational factors of the proposed systems like man-power, time etc., whichever solution uses less operational resources, is the best operationally feasible solution. The solution should also be operationally possible to implement.

- The methods of processing and presentation are completely accepted by the clients since they can meet all user requirements.
- The clients have been involved in the planning and development of the system.
- The proposed system will not cause any problem under any circumstances.

Our project is operationally feasible because the time requirements and personnel requirements are satisfied. We are a team of four members and we worked on this project for three working months.

Chapter 7

Testing

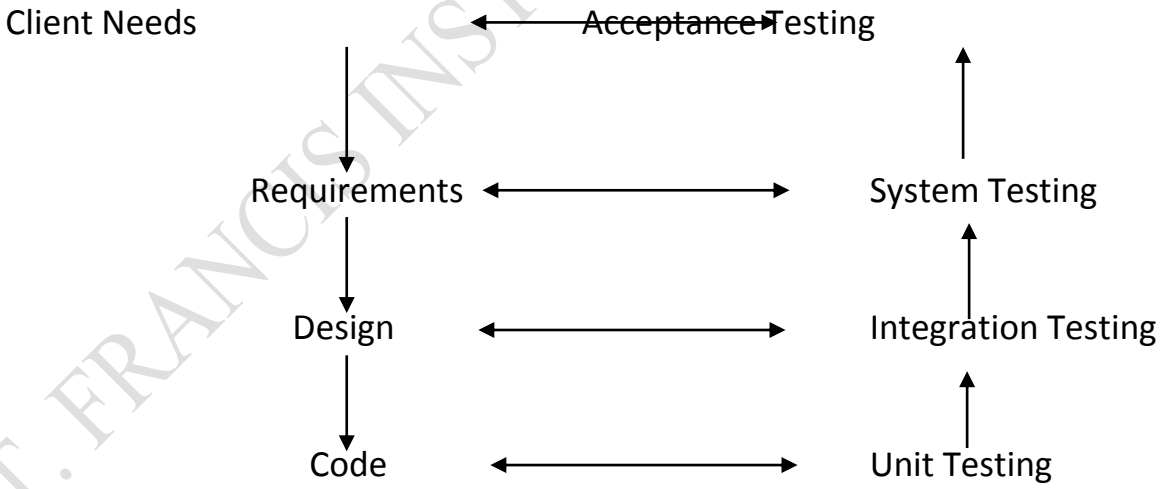
As the project is on bit large scale, we always need testing to make it successful. If each components work properly in all respect and gives desired output for all kind of inputs then project is said to be successful. So the conclusion is-to make the project successful, it needs to be tested.

The testing done here was System Testing checking whether the user requirements were satisfied. The code for the new system has been written completely using ASP .NET and Android Studio as the coding language, C# and Android Studio as the interface for front-end designing. The new system has been tested well with the help of the users and all the applications have been verified from every nook and corner of the user.

Although some applications were found to be erroneous these applications have been corrected before being implemented. The flow of the forms has been found to be very much in accordance with the actual flow of data.

Levels of Testing

In order to uncover the errors present in different phases we have the concept of levels of testing. The basic levels of testing are:



7.1 Test cases

User Login/Registration: To begin with login, user need to register by filling up basic registration details. There are multiple fields in registration page and every field has to fill by user. User cannot use character in the login id field.

Login: -login id and password is kept compulsory fields, and if the id or password doesn't match then it will show an error message.

VALIDATION CRITERIA

1. In each form, no field which is not null able should be left blank.
2. All numeric fields should be checked for non-numeric values. Similarly, text fields like names should not contain any numeric characters.
3. All primary keys should be automatically generated to prevent the user from entering any existing key.
4. Use of error handling for each Save, Edit, delete and other important operations.
5. Whenever the user Tabs out or Enter from a text box, the data should be validated and if it is invalid, focus should again be sent to the text box with proper message.

7.2 Type of Testing used

Unit Testing

Unit testing focuses verification efforts on the smallest unit of the software design**,** the module**.** This is also known as "Module Testing"**.** The modules are tested separately**.** This testing carried out during programming stage itself**.** In this testing each module is found to be working satisfactorily as regards to the expected output from the module**.**

Integration Testing

Data can be grossed across an interface**;** one module can have adverse efforts on another**.** Integration testing is systematic testing for construction the program structure while at the same time conducting tests to uncover errors associated with in the interface. The objective is to take unit tested modules and build a program structure**.** All the modules are combined and tested as a whole**.** Here correction is difficult because the isolation of cause is complicate by the vast expense of the entire program. Thus in the integration testing stop**,** all the errors uncovered are corrected for the text testing steps**.**

System testing

System testing is the stage of implementation that is aimed at ensuring that the system works accurately and efficiently for live operation commences. Testing is vital to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, then goal will be successfully achieved.

User Acceptance Testing

User acceptance of a system is the key factor of the success of any system. The system under study is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required.

Chapter 8

Results and Discussions

8.1 Results

This shall form the penultimate chapter of the report and shall include a thorough evaluation of the investigation carried out and bring out the contributions from the study.

The system was also tested after adding a restaurant by the admin.It was checked if the new restaurant appears on the interface of the application and the log in credentials of the restaurant manager were accepted.

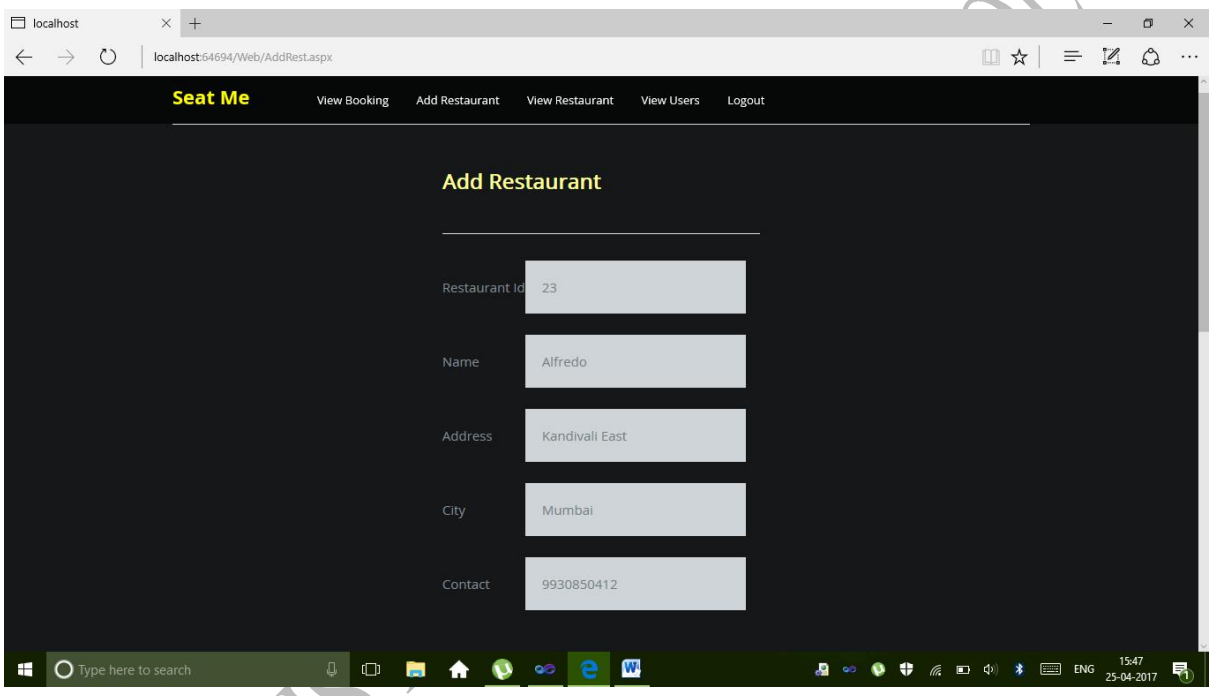


Fig 8.1.1: Add a restaurant

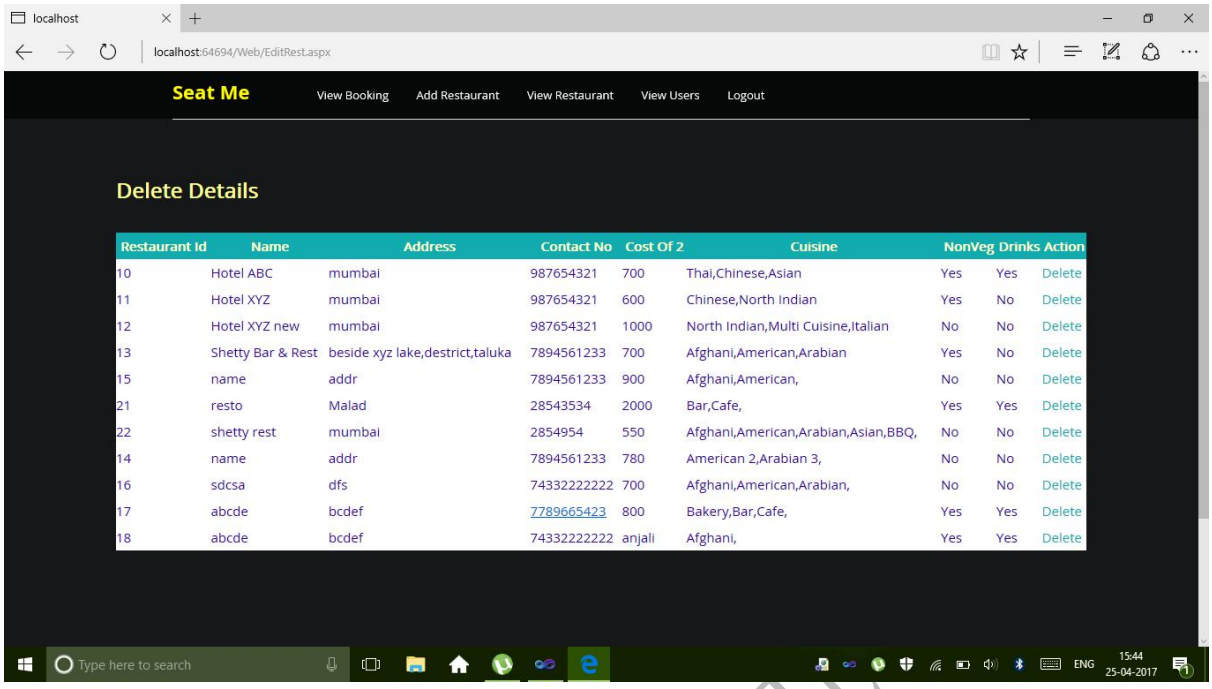


Fig 8.1.2: Before adding a restaurant

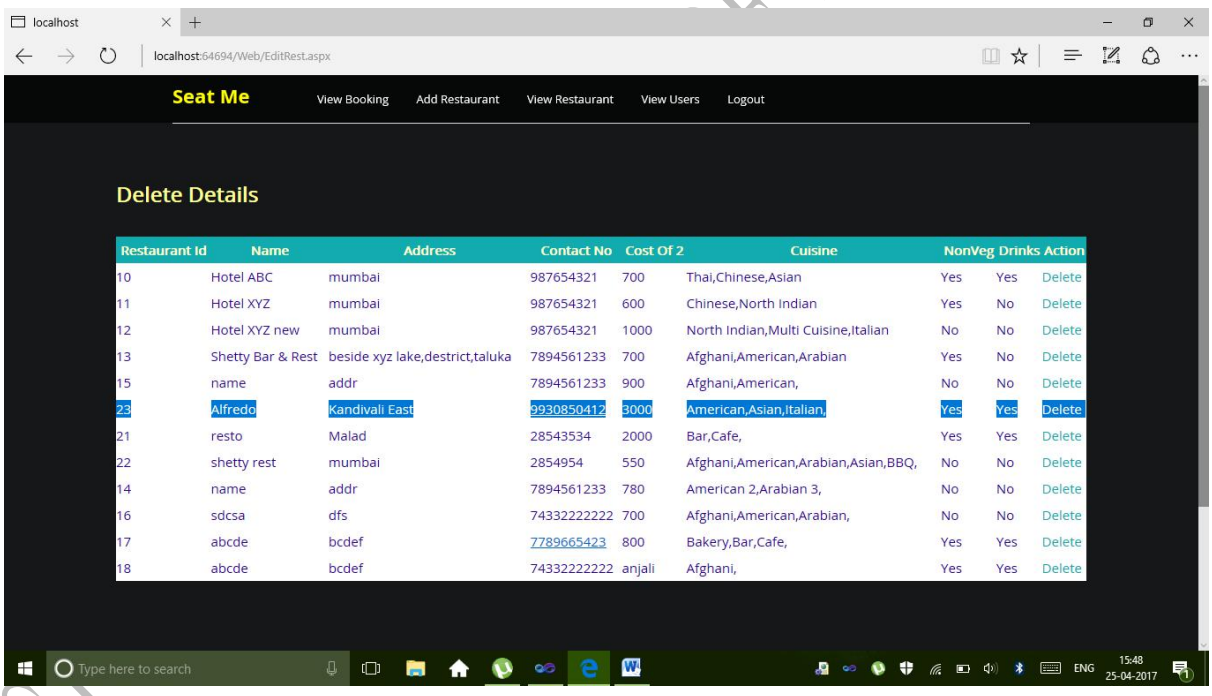


Fig 8.1.3: After adding a restaurant

To test if the system is working as per expectations a seat was booked in a restaurant.After booking the seat it was checked if the user, the restaurant manager as well as the admin panel of the application were receiving an update and confirmation of the same.

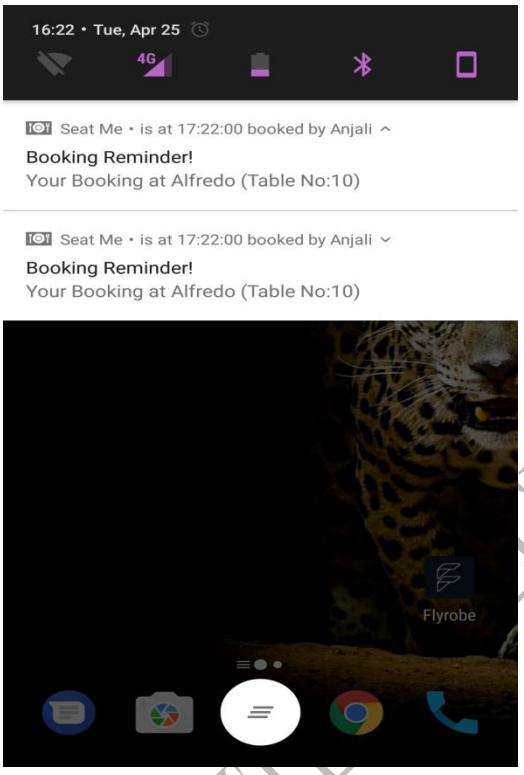


Fig 8.1.4: Notification of the booking to the user

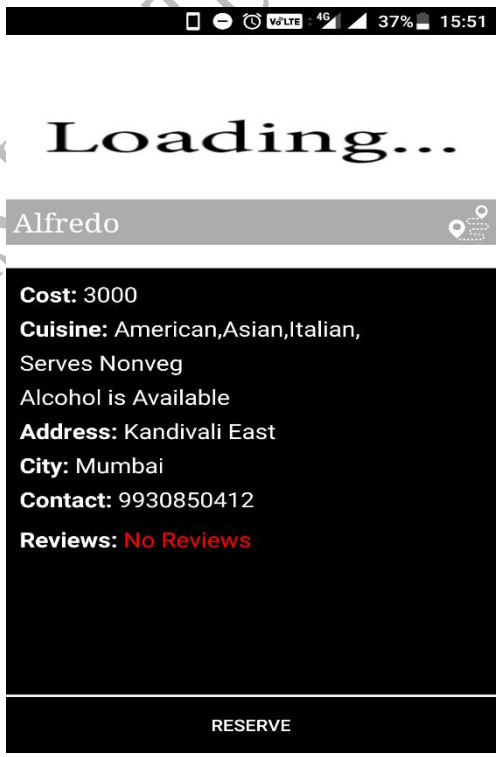


Fig 8.1.5: Booking confirmation

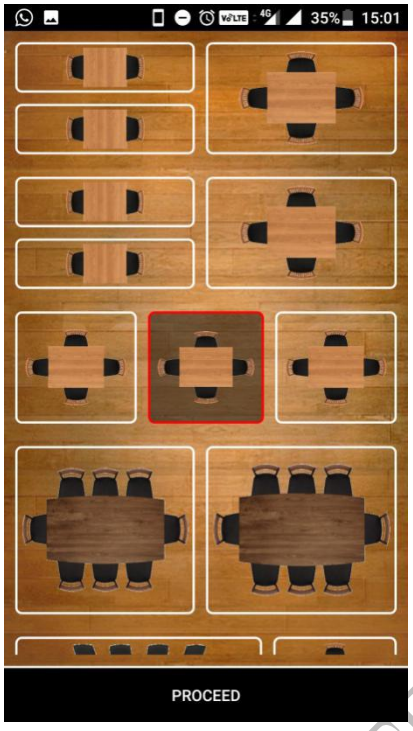


Fig 8.1.6: Table booked

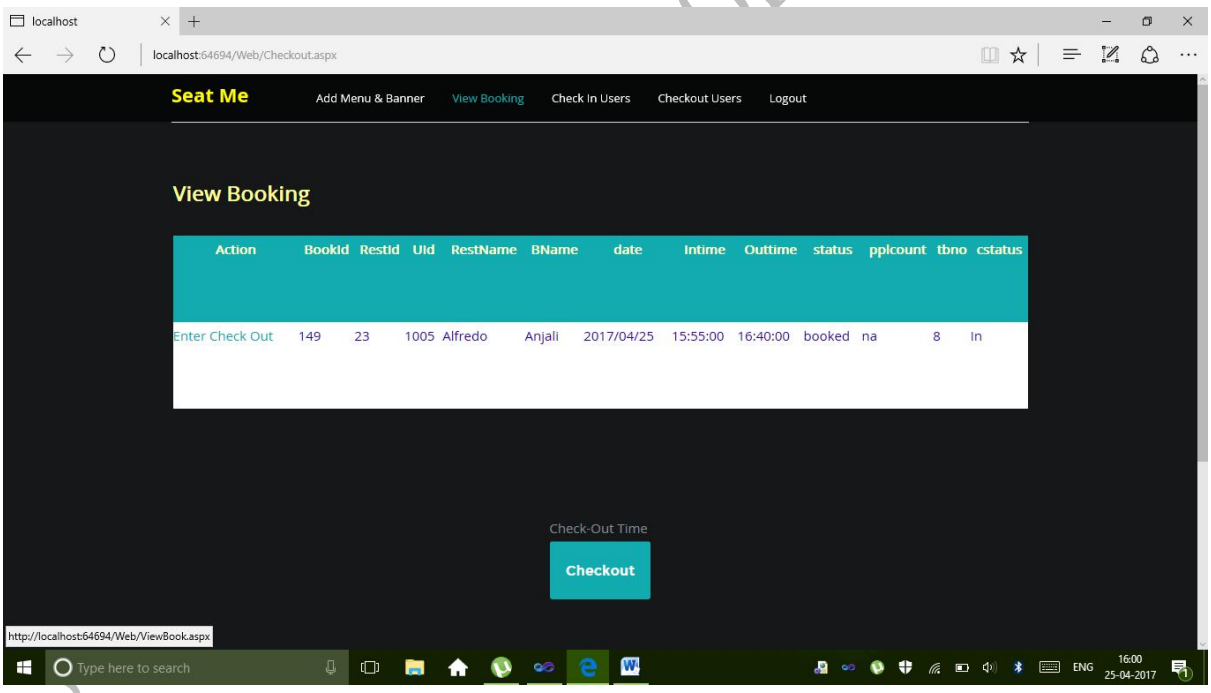


Fig 8.1.7: Table booking viewed by restaurant manager

8.2 Discussions

The effectiveness of the application was discussed with the help of a survey.The results of the survey are as follows:

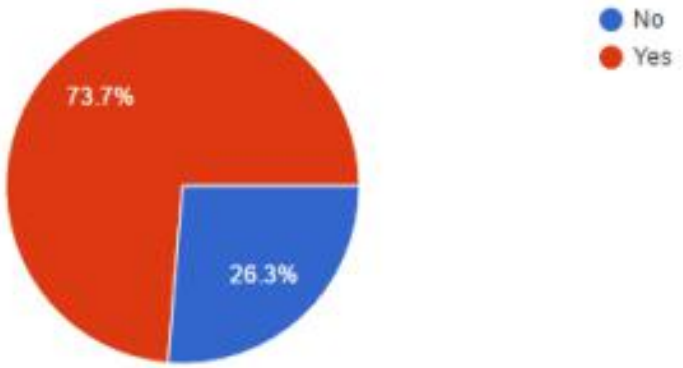


Fig 8.2.1: Effectiveness of the application

The discussion shall logically lead to inferences and conclusions as well as scope for possible further future work.

Chapter 9

Conclusion & Future scope

This was our project of System Design about "**Online Table Booking**" Android as well as web application based on Java and Asp .Net language. The Development of this system takes a lot of efforts from us. We think this system gave a lot of satisfaction to all of us. Though every task is never said to be perfect in this development field even more improvement may be possible in this application. We learned so many things and gained a lot of knowledge about development field. We hope this will prove fruitful to us.

This application does the following things: enable the customer to sign-up, log in, search for restaurants, book a table, cancel a booking(done by the manager of the restaurant), enables the admin of the application to add more restaurants to the application, remove a restaurant from the application, the customer can a write a review for the visited restaurant. This application would save time of the customer and give the customer pleasure to book a table with the help of the application.

Apart from these functions a certain number of improvements are possible in this application. It can also be used to place order so that more time of the customer is saved. Apart from placing the order the application can also get a payment option through which the customer can also pay for the food ordered by him/her. There can be a provision where in the customer would be able to book a table a day prior to the visit to the restaurant.

Literature Cited

[1] Courtney McTavish, Suresh Sankaranarayanan “Intelligent agent based hotel search & booking system”, Electro/Information Technology (EIT), 2010 IEEE International Conference on Page(s): 1-6

[2] Seema V. Kedar, Akkshay T. Shinde, Dhanshree V. Chandgude, Department of Information Technology University of Pune, “Precise Approach to perform decision making on medical database”

[3] Reto Meier (2009) “Professional android application development” – Wiley Publishing Inc.

[4] Satya Komatineni (2009) “Pro Android” – Apress Publications.

Acknowledgement

We are extremely grateful to our college **St. Francis Institute of Technology** for the confidence bestowed in us and entrusting our project entitled **Seat Me (Online Table Booking System).**

We express our sincere gratitude to our respected director **Bro. Melchior Tom**, our principal **Dr. Sincy George** and our HOD **Dr. Joanne Gomes** for encouragement and facilities provided to us.

We owe our profound gratitude to our project guide **Mr. Bhavesh Pandya**, who guided and supported us at every stage of our project.

Many people, especially our team members and classmates, have made valuable comment suggestions on this proposal which gave us an inspiration to improve our assignment.