VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI, KARNATAKA – 590018

A

MINI PROJECT REPORT

ON

“RESTAURANT RESERVATION SYSTEM”

Submitted in partial fulfillment of the requirements as a part of the ***DBMS Laboratory with Mini Project (18CSL58)*** *for the V semester of degree of* ***Bachelor of Engineering in Computer Science and Engineering*** *of Visvesvaraya Technological University, Belagavi*

Submitted by:

ABHAY TM - 1JS19CS003
ADITHYA S - 1JS19CS010

Under the Guidance of

Mrs. Rajeshwari K.S,
Assistant Professor, Dept of CSE,
JSSATE, Bengaluru

JSS ACADEMY OF TECHNICAL EDUCATION
Dr.Vishnuvardhan Rd, Srinivasapura Post,
Bengaluru – 560060 (2021-2022)

JSS MAHA VIDYAPEETHA, MYSORE

JSS ACADEMY OF TECHNICAL EDUCATION

JSS Campus, Uttarahalli - Kengeri main road, Bengaluru – 560060

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project work entitled "RESTAURANT RESERVATION SYSTEM" is a bona fide work carried out by **Mr. Abhay TM (1JS19CS003)** and **Mr. Adithya S (1JS19CS010)** in partial fulfillments of the requirements for DBMS Laboratory with Mini Project (18CSL58) of V semester **Bachelor of Engineering in Computer Science and Engineering** of the academic year 2021-2022. It is certified that all the corrections and suggestions indicated for Internal Assessments have been incorporated in the report deposited in the department Library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Mrs. Rajeshwari K.S,	**Dr. Naveen N.C,**
Assistant Professor, Dept of CSE	Professor and Head, Dept of CSE
JSSATE, Bengaluru	JSSATE, Bengaluru

Name of the Examiners:	**Signature with Date:**
1. …………………………	……………………………
2. …………………………	……………………………

ABSTRACT

A database management system stores, organizes and manages a large amount of information within a single software application. The use of this system increases efficient of business operations and reduce overall costs.

"Restaurant reservation system" is a system used to automate the working of restaurants by using various DBMS attributes. This system is to make table reservations in a restaurant easier for the customers and to make the jobs of the managers easier by reducing the number of people waiting for the tables.

It also gives the manager the ability to store the items that are ordered by a particular user after he or she visits the restaurant as in the current situation remembering the orders are so difficult.

The manager has the facility to edit the tables available and to alert the valets when the customer is ready to leave. The customer must register and while doing so has to enter all his required credentials to access the reservations system and security is taken care of as the system also has a Login page – where the customer and managers can login using separate credentials.

The Restaurant Reservation system is the system for managing the restaurant business. After successful login, the customer can access the menu page with the items listed according to the desired time. The main point of developing this system is to help restaurant administrators manage the restaurant business and help customers with reserving tables.

We also have the facility to add new upcoming restaurants and user can delete his or booking, update his or her booking after the booking has been completed.
This gives this project all the CRUD operations of the database design and makes it an apt representation of the requirements given to us.

It also has the GUI which can be used by the user to register, login, check the restaurants, check the menu , book a table , update, and delete bookings.

ACKNOWLEDGEMENT

We express our humble pranams to His Holiness **Jagadguru Sri Sri Sri Shivaratri Deshikendra mahaswamiji** for showering his blessings on us to receive good education and have a successful career.

The completion of any project involves the effort of many people. We have been very lucky to have received support and guidance from all kinds of sources to complete our project. So, we take this opportunity to express our gratitude to all those who gave us guidance and encouragement to successfully complete this project.

We express a sincere thanks to our beloved principal **Dr. Mrityunjaya V Latte** for having supported us in all academic endeavors.

We are also forever grateful to **Dr. Naveen N.C** Head of Department, Computer science and Engineering for his unending support, guidance and encouragement in all are ventures.

We are very thankful for the resourceful guidance and timely assistance and graceful gesture of our guide **Mrs. Rajeshwari K.S** Assistant Professor, **Mr. Rohitaksha K** Assistant Professor Department of computerscience and engineering who has helped us in every aspect of our project work.

Last but not the least, we would be immensely pleased to express our heartfelt thanks to all the teaching and non-teaching staff of the Department of CSE and our friends for the timely help support and guidance.

ABHAY TM
ADITHYA S

Table of Contents

Chapter 2 Requirement Specifications

Chapter 3 System Design and Implementation

LIST OF FIGURES

Chapter 1

Preamble

1.1 Introduction

A database is an organized collection of data. A relational database, more restrictively, is a collection of schemas, tables, queries, reports, views, and other elements. A database management system (DBMS) is a computer-software application that interacts with end users, other applications, and the database itself to capture and analyze data. A general-purpose DBMS allows the definition, creation, querying, update, and administration of databases. There is a need for an application to make it easy for industries and trading companies to maintain their records and have a track of goods. With an effective MULTI RESTURANT MANAGEMENT SYSTEM, restaurants can automate the working of restaurants correctly. MULTI RESTURANT MANAGEMENT SYSTEM plays an important role in managing time efficiently.

Database Management System (DBMS)

Following the technology progress in the areas of processors, computer memory, computer storage, and computer networks, the sizes, capabilities, and performance of databases and their respective DBMSs have grown in orders of magnitude. The development of database technology can be divided into three eras based on data model or structure: navigational, SQL/relational, and post-relational. The two main early navigational data models were the hierarchical model, epitomized by IBM's IMS system, and the CODASYL model (network model), implemented in a number of products such as IDMS.

The relational model employs set of ledger-style tables, each used for a different type of entity. Only in the mid-1980s did computing hardware become powerful enough to allow the wide deployment of relational systems (DBMSs plus applications). By the early 1990s, however, relational systems dominated in all large-scale data processing applications, and as of 2015 they remain dominant: IBM DB2, Oracle, MySQL, and Microsoft SQL Server are the top DBMS. The dominant database language, standardized SQL for the relational model, has influenced database languages for other data models.

1.1.1 Hypertext Markup Language (HTML)

The Hypertext Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets.

1.1.2 Cascading Style Sheets (CSS)

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

1.1.3 JavaScript

JavaScript often abbreviated JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. Over 97% of websites use JavaScript on the client side for web page behavior, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices.

JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard. It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).

1.1.4 Database MySQL

MySQL is free and open-source software under the terms of the GNU General Public License, and is also available under a variety of proprietary licenses. MySQL was owned and sponsored by the Swedish company MySQL AB, which was bought by Sun Microsystems (now Oracle Corporation). In 2010, when Oracle acquired Sun, Widenius forked the open-source MySQL project to create MariaDB.

MySQL has stand-alone clients that allow users to interact directly with a MySQL database using SQL, but more often, MySQL is used with other programs to implement applications that need relational database capability.

1.1.5 Django

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

- Ridiculously fast.

Django was designed to help developers take applications from concept to completion as quickly as possible.

- Reassuringly secure.

Django takes security seriously and helps developers avoid many common security mistakes.

- Exceedingly scalable.

Some of the busiest sites on the web leverage Django's ability to quickly and flexibly scale.

Django officially supports the following databases:

- PostgreSQL
- MariaDB
- MySQL
- Oracle
- SQLite

1.1.6 Normalization

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. To overcome these anomalies, we need to normalize the data. There are 4 basic types of normalizations. They are:

- First normal form (INF)
- Second normal form(2NF)
- Third normal form(3NF)
- Boyce & Codd normal form (BCNF)

First normal form (INF) is defined as per rule as: an attribute (column) of a table cannot hold multiple values. It should hold only atomic values. This means that there shouldn't be repetition of data in the tables.

A table is said to be in 2NF if the two conditions stated are satisfied. The table is in First normal form and all the non-prime attribute are dependent on the proper subset of any candidate key of table. The attribute that is not part of any candidate key are known as non-prime attribute.

A table design is said to be in 3NF if the table is in 2NF and Transitive functional dependency of non-prime attribute on any super key are removed.

Boyce Codd normal form (BCNF) is the advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency X-BY, X should be the super key of the table.

1.2 Objectives

Objectives of “Restaurant management system” are

- To Minimize the Time for waiting.
- Managing the restaurant efficiently.
- Shortening the time for booking.
- Booking optimization
- Valet parking
- Efficiency
- Easy Login

1.3 Organization of Report

Chapter 1 provides the information about the basics of HTML, CSS, JavaScript, Django, and MySQL. In **Chapter 2**, we discuss the software and hardware requirements to run the above applications. **Chapter 3** gives the idea of the project and its actual implementation. **Chapter 4** discusses about the results and discussions of the program. It concludes by giving the direction for future enhancement.

1.4 Summary

The chapter discussed before is an overview about the Django Application and MySQL Database DBMS. The scope of study and objectives of the project are mentioned clearly. The organization of the report has been pictured to increase the readability. Further, coming up chapters depicts the use of various queries to implement various changes like insert, update, delete and also triggers to perform various functions.

Chapter 2

Requirement Specifications

2.1 SOFTWARE SPECIFICATION

- Operating System: Windows 10
- Front End: HTML, CSS, and JavaScript
- Rear End: MySQL, Django

2.2 HARDWARE SPECIFICATION

- Processor: x86 compatible processor with 1.7 GHz Clock Speed
- RAM: 512 MB or greater
- Hard Disk: 20 GB or grater
- Monitor: VGA/SVGA
- Keyboard: 104 keys standard
- Mouse: 2/3 button. Optical/Mechanical.

2.3 USER CHARACTERISTICS

Every user:

- Should be comfortable with basic working of the computer
- Must have basic knowledge of English
- Must carry a Login username and password used for authentication

Chapter 3

System Design and Implementation

3.1 Introduction

Systems design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development.

This Project is implemented using Django, which is proven to be a very efficient tool in the field of python programming. It is done under Windows 10 platform. HTML, CSS, and Python programming language is used to implement the entire code.

Interface to the program is provided with the help of MySQL Database.

3.2 ER Diagram

An entity–relationship model or the ER Diagram describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types and specifies relationships that can exist between instances of those entity types.

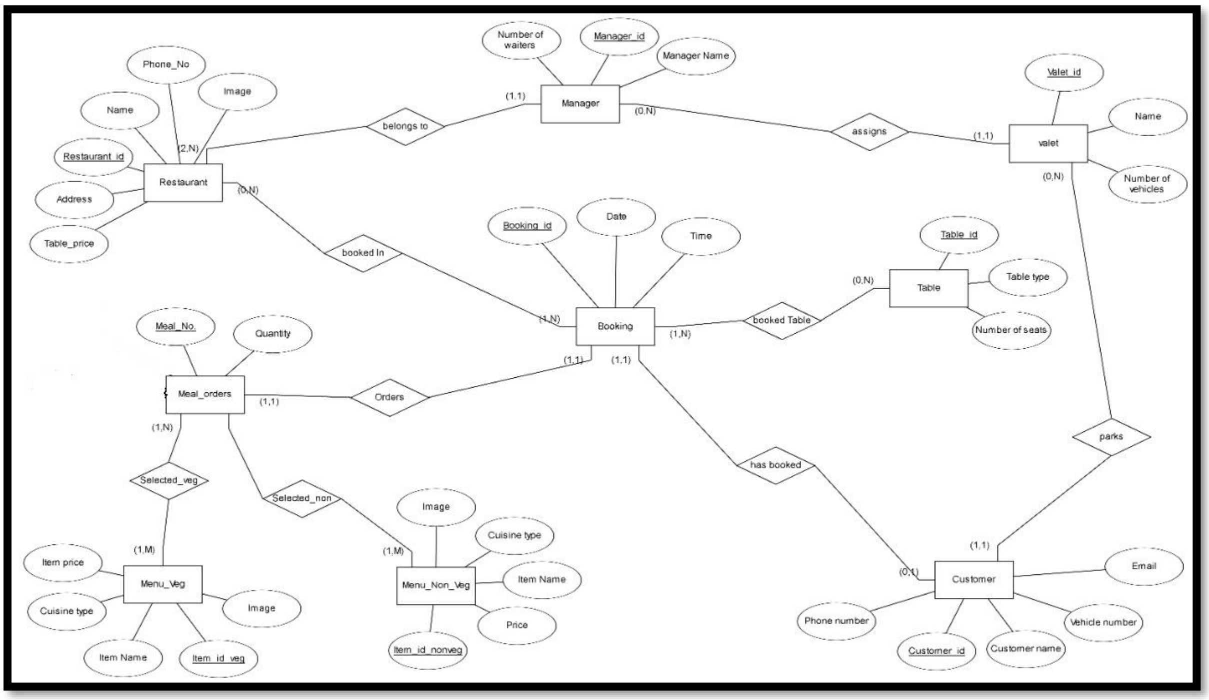


Figure 3.2 ER Diagram

3.2 ENTITIES

1. Restaurant – Restaurant_ID – Primary Key , Name , Address , Table_Price , Phone_No , Image (Image name stored in Database)
2. Manager – Manager_ID , Manager_Name , No_Of_Waiters (Restaurant_ID Foreign Key)
3. Valet – Valet_ID , Valet_Name , No_Of_Vehicles , (Manager_ID Foreign Key)
4. Table – Table_ID , Table_Type , No_Of_Seates
5. Menu_Veg – Item_ID_Veg , Item_Name , Item_Price , Cuisine_Type, Image
6. Menu_NonVeg – Item_ID_NonVeg , Item_Name , Item_Price , Cuisine_Type, Image
7. Meals_Orders – Meal_No , Quantity , (Item_Veg,Item_NonVeg,Booking_ID)
8. Customer – Customer_ID , Customer_Name , Vehicle_Number , Email , Phone_No , (Valet_ID Foreign Key)
9. Booking – Booking_ID, Date , Time , (Customer_ID, Restaurant_ID,Manager_ID,Table_ID Foreign Key)

3.3 Schema Diagram

The schema diagram of a database system is its structure described in a formal language supported by the database management system (DBMS). The formal definition of a database schema is a set of formulas called integrity constraints imposed on a database.

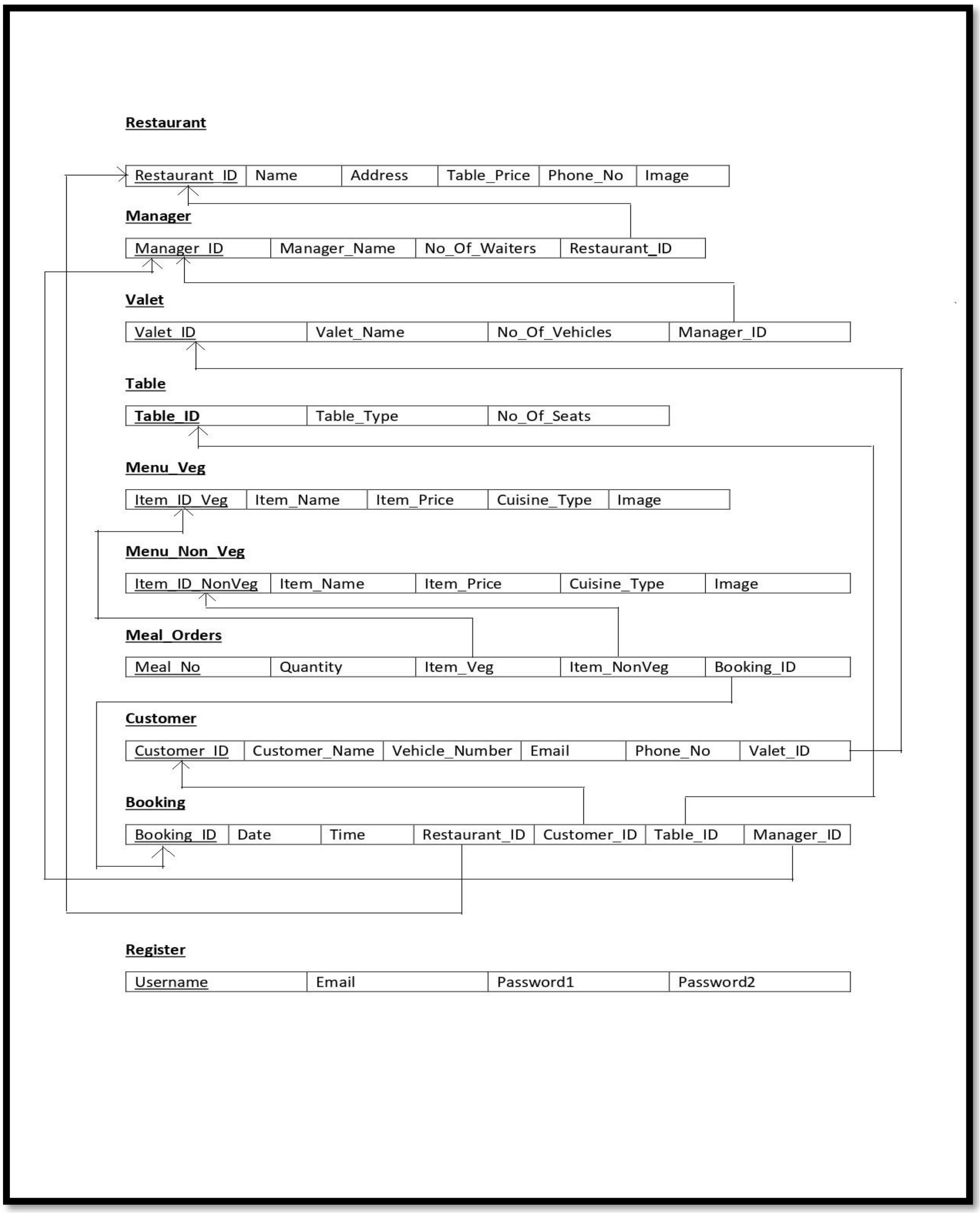

Restaurant
Restaurant_ID | Name | Address | Table_Price | Phone_No | Image
Manager
Manager_ID | Manager_Name | No_Of_Waiters | Restaurant_ID
Valet
Valet_ID | Valet_Name | No_Of_Vehicles | Manager_ID
Table
Table_ID | Table_Type | No_Of_Seats
Menu_Veg
Item_ID_Veg | Item_Name | Item_Price | Cuisine_Type | Image
Menu_Non_Veg
Item_ID_NonVeg | Item_Name | Item_Price | Cuisine_Type | Image
Meal_Orders
Meal_No | Quantity | Item_Veg | Item_NonVeg | Booking_ID
Customer
Customer_ID | Customer_Name | Vehicle_Number | Email | Phone_No | Valet_ID
Booking
Booking_ID | Date | Time | Restaurant_ID | Customer_ID | Table_ID | Manager_ID
Register
Username | Email | Password1 | Password2

Figure 3.3 Relationship Schema Diagram

The term "schema" refers to the organization of data as a blueprint of how the database is constructed. These integrity constraints ensure compatibility between parts of the schema. All constraints are expressible in the same language. A database can be considered a structure in realization of the database language. The states of a created conceptual schema are transformed into an explicit mapping, the database schema. This describes how real-world entities are modeled in the database.

Figure 3.2 Schema is defined for an Inventory Management System. All the various table used are described in the following schema. The necessary Primary keys and the corresponding foreign keys are also represented.

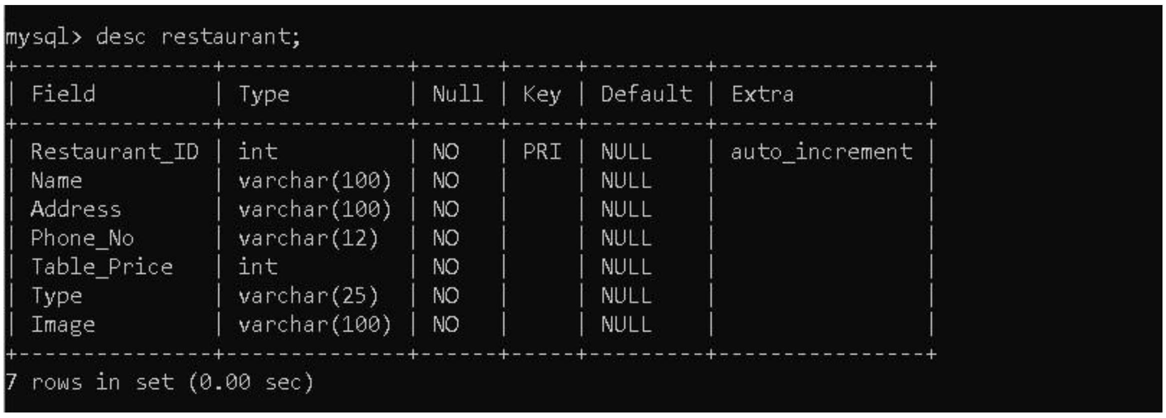
3.4 Queries

The below mentioned are all the queries used to perform various tasks in MySQL and DJANGO such as insert, delete, update. A short description of the query is also provided. Updating can also be done using the GUI provided in the admin page and the webapp.

3.4.1 Creating Tables

RESTAURANT TABLE

```
CREATE TABLE RESTAURANT
(
        RESTAURANT_ID INTEGER,
        NAME VARCHAR (100),
        ADDRESS VARCHAR (100),
        PHONE_NO NUMBER (12),
        TABLE_PRICE INTEGER,
        TYPE VARCHAR (25),
        IMAGE VARCHAR(100)
        PRIMARY KEY (RESTAURANT_ID)
);
```



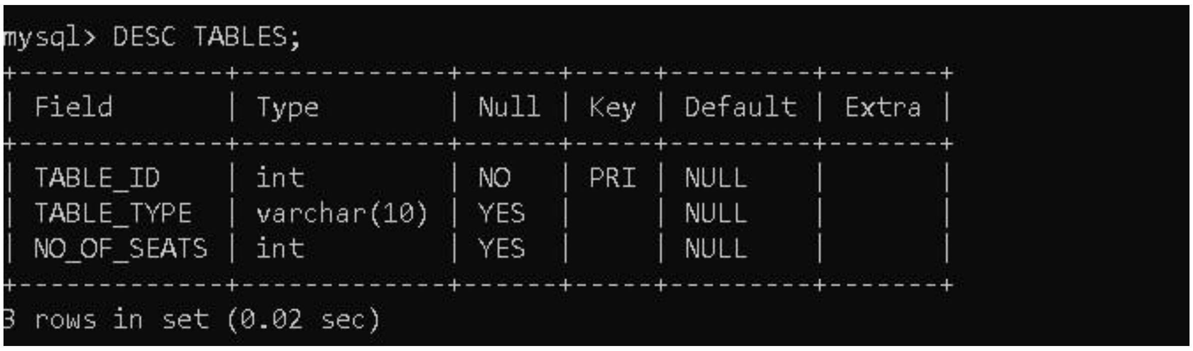
```
mysql> desc restaurant;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| Restaurant_ID | int          | NO   | PRI | NULL    | auto_increment |
| Name          | varchar(100) | NO   |     | NULL    |                |
| Address       | varchar(100) | NO   |     | NULL    |                |
| Phone_No      | varchar(12)  | NO   |     | NULL    |                |
| Table_Price   | int          | NO   |     | NULL    |                |
| Type          | varchar(25)  | NO   |     | NULL    |                |
| Image         | varchar(100) | NO   |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

Figure 3.4 Restaurant Table Description

Figure 3.4 shows the description of the table shown in the MySQL command line after creation of table . It shows the attributes in the table and their Types with the constraints.

TABLES TABLE

```
CREATE TABLE TABLES
(
        TABLE_ID INTEGER,
        TABLE_TYPE VARCHAR (10),
        NO_OF_SEATS INTEGER,
        PRIMARY KEY (TABLE_ID)
);
```



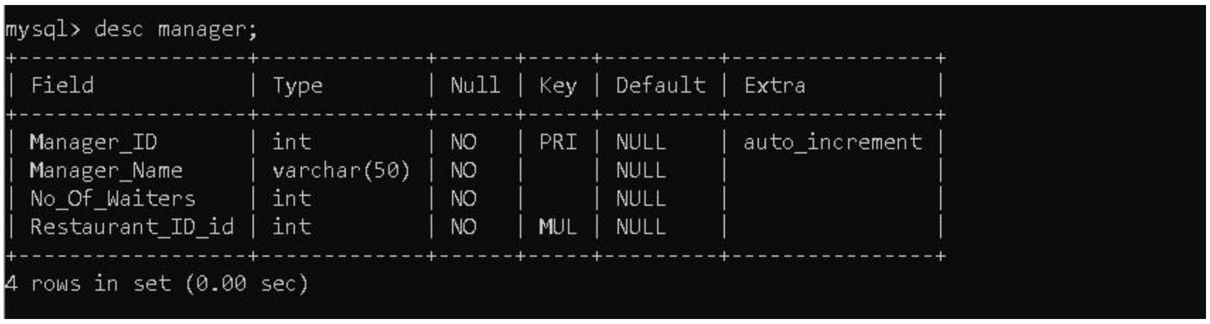
```
mysql> DESC TABLES;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| TABLE_ID    | int         | NO   | PRI | NULL    |       |
| TABLE_TYPE  | varchar(10) | YES  |     | NULL    |       |
| NO_OF_SEATS | int         | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.02 sec)
```

Figure 3.5 Tables Table Description

Figure 3.5 shows the description of the table shown in the MySQL command line after creation of table . It shows the attributes in the table and their Types with the constraints.

MANAGER TABLE

```
CREATE TABLE MANAGER
(
        MANAGER_ID INTEGER,
        MANAGER_NAME VARCHAR (50),
        NO_OF_WAITERS INTEGER,
        RESTAURANT_ID INTEGER,
        PRIMARY KEY ('MANAGER_ID'),
        FOREIGN KEY (RESTAURANT_ID) REFERENCES RESTAURANT
        (RESTAURANT_ID) ON DELETE CASCADE
);
```



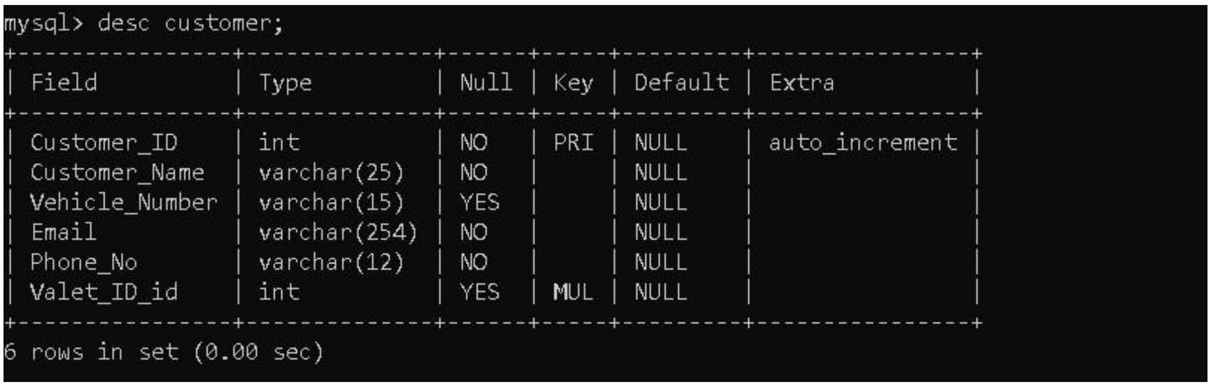
```
mysql> desc manager;
+------------------+-------------+------+-----+---------+----------------+
| Field            | Type        | Null | Key | Default | Extra          |
+------------------+-------------+------+-----+---------+----------------+
| Manager_ID       | int         | NO   | PRI | NULL    | auto_increment |
| Manager_Name     | varchar(50) | NO   |     | NULL    |                |
| No_Of_Waiters    | int         | NO   |     | NULL    |                |
| Restaurant_ID_id | int         | NO   | MUL | NULL    |                |
+------------------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

Figure 3.6 Manager Table Description

Figure 3.6 shows the description of the table shown in the MySQL command line after creation of table .It shows the attributes in the table and their Types with the constraints.

CUSTOMER TABLE

```
CREATE TABLE CUSTOMER
(
        CUSTOMER_ID INTEGER,
        CUSTOMER_NAME VARCHAR (25),
        VEHICLE_NUMBER VARCHAR (15),
        EMAIL VARCHAR(254)
        PHONE_NO NUMBER (12),
        VALET_ID INTEGER,
        PRIMARY KEY (CUSTOMER_ID),
        FOREIGN KEY (VALET_ID) REFERENCES VALET(VALET_ID)
);
```



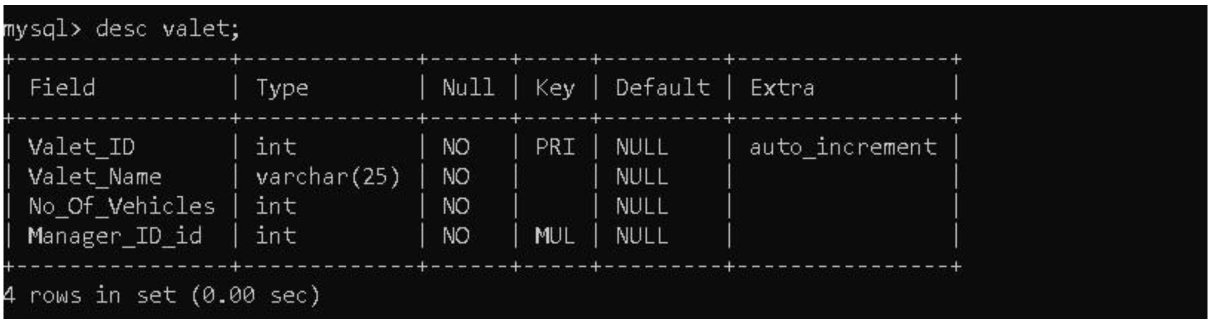
```
mysql> desc customer;
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| Customer_ID    | int          | NO   | PRI | NULL    | auto_increment |
| Customer_Name  | varchar(25)  | NO   |     | NULL    |                |
| Vehicle_Number | varchar(15)  | YES  |     | NULL    |                |
| Email          | varchar(254) | NO   |     | NULL    |                |
| Phone_No       | varchar(12)  | NO   |     | NULL    |                |
| Valet_ID_id    | int          | YES  | MUL | NULL    |                |
+----------------+--------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

Figure 3.7 Customer Table Description

Figure 3.7 shows the description of the table shown in the MySQL command line after creation of table . It shows the attributes in the table and their Types with the constraints.

VALET TABLE

```
CREATE TABLE VALET
(
        VALET_ID INTEGER,
        VALET_NAME VARCHAR (20),
        NO_OF_VEHICLES INTEGER,
        MANAGER_ID INTEGER,
        PRIMARY KEY (VALET_ID),
        FOREIGN KEY (MANAGER_ID) REFERENCES MANAGER (MANAGER_ID)
        ON DELETE SET NULL
);
```



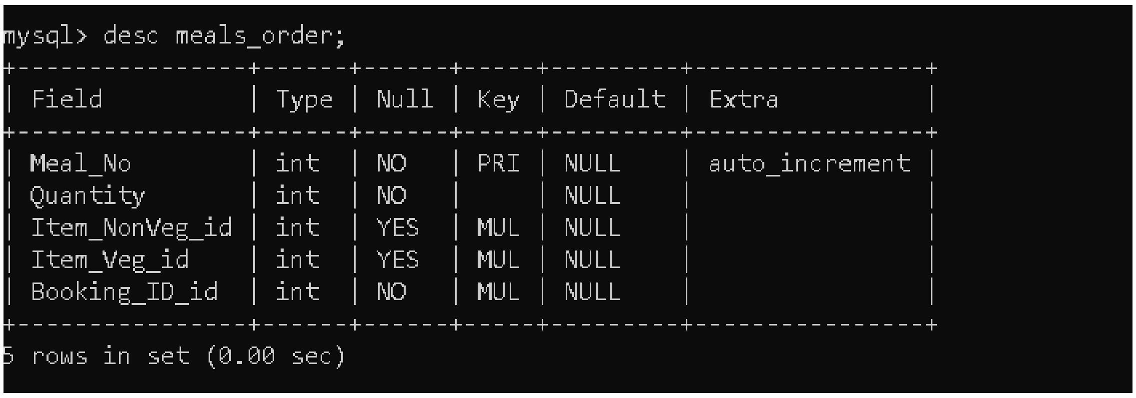
```
mysql> desc valet;
+----------------+-------------+------+-----+---------+----------------+
| Field          | Type        | Null | Key | Default | Extra          |
+----------------+-------------+------+-----+---------+----------------+
| Valet_ID       | int         | NO   | PRI | NULL    | auto_increment |
| Valet_Name     | varchar(25) | NO   |     | NULL    |                |
| No_Of_Vehicles | int         | NO   |     | NULL    |                |
| Manager_ID_id  | int         | NO   | MUL | NULL    |                |
+----------------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

Figure 3.8 Valet Table Description

Figure 3.8 shows the description of the table shown in the MySQL command line after creation of table . It shows the attributes in the table and their Types with the constraints.

MEAL_ORDERS TABLE

```
CREATE TABLE MEAL_ORDER
(
        MEAL_NO INTEGER,
        QUANTITY INTEGER,
        ITEM_VEG INTEGER,
        ITEM_NONVEG INTEGER,
        BOOKING_ID INTEGER
        PRIMARY KEY ('MEAL_NO'),
        FOREIGN KEY (BOOKING_ID) REFERENCES BOOKING (BOOKING_ID));
        FOREIGN KEY (ITEM_VEG) REFERENCES MENU_VEG (ITEM_ID_VEG));
        FOREIGN KEY (ITEM_NONVEG) REFERENCES MENU_NON_VEG
        (ITEM_ID_NONVEG)
);
```



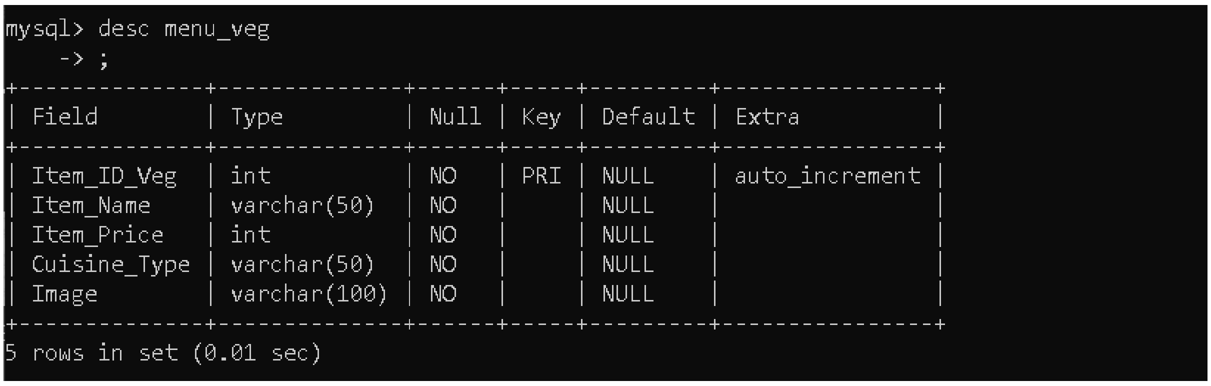
```
mysql> desc meals_order;
+----------------+------+------+-----+---------+----------------+
| Field          | Type | Null | Key | Default | Extra          |
+----------------+------+------+-----+---------+----------------+
| Meal_No        | int  | NO   | PRI | NULL    | auto_increment |
| Quantity       | int  | NO   |     | NULL    |                |
| Item_NonVeg_id | int  | YES  | MUL | NULL    |                |
| Item_Veg_id    | int  | YES  | MUL | NULL    |                |
| Booking_ID_id  | int  | NO   | MUL | NULL    |                |
+----------------+------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

Figure 3.9 Meals_Order Table Description

Figure 3.9 shows the description of the table shown in the MySQL command line after creation of table . It shows the attributes in the table and their Types with the constraints.

MENU_VEG TABLE

```
CREATE TABLE MENU_VEG
(
        ITEM_ID_VEG INTEGER,
        ITEM_PRICE INTEGER,
        ITEM_NAME VARCHAR (20),
        CUISINE_TYPE VARCHAR (10),
        IMAGE VARCHAR(100),
        PRIMARY KEY (ITEM_ID_VEG)
);
```



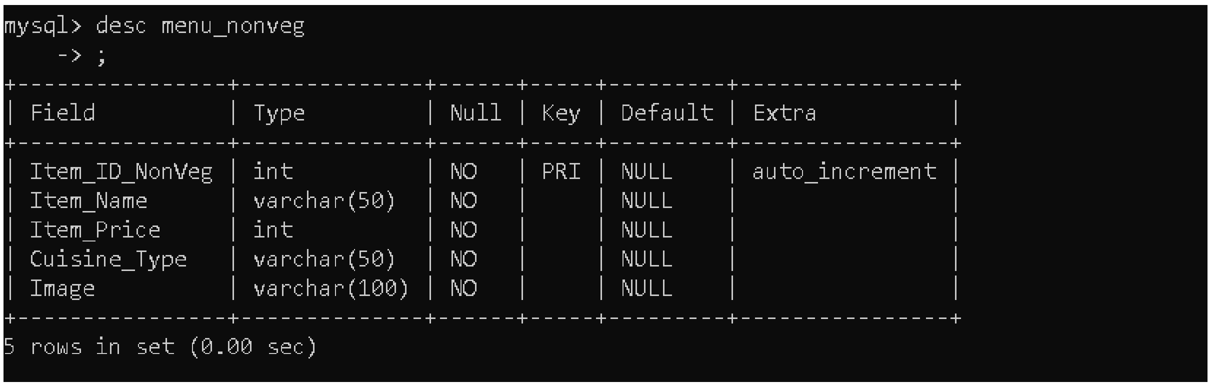
```
mysql> desc menu_veg
    -> ;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| Item_ID_Veg  | int          | NO   | PRI | NULL    | auto_increment |
| Item_Name    | varchar(50)  | NO   |     | NULL    |                |
| Item_Price   | int          | NO   |     | NULL    |                |
| Cuisine_Type | varchar(50)  | NO   |     | NULL    |                |
| Image        | varchar(100) | NO   |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```

Figure 3.10 Menu_Veg Table Description

Figure 3.10 shows the description of the table shown in the MySQL command line after creation of table . It shows the attributes in the table and their Types with the constraints.

MENU_NONVEG TABLE

```
CREATE TABLE MENU_VEG
(
        ITEM_ID_VEG INTEGER,
        ITEM_PRICE INTEGER,
        ITEM_NAME VARCHAR (20),
        CUISINE_TYPE VARCHAR (10),
        IMAGE VARCHAR(100),
        PRIMARY KEY (ITEM_ID_VEG)
);
```



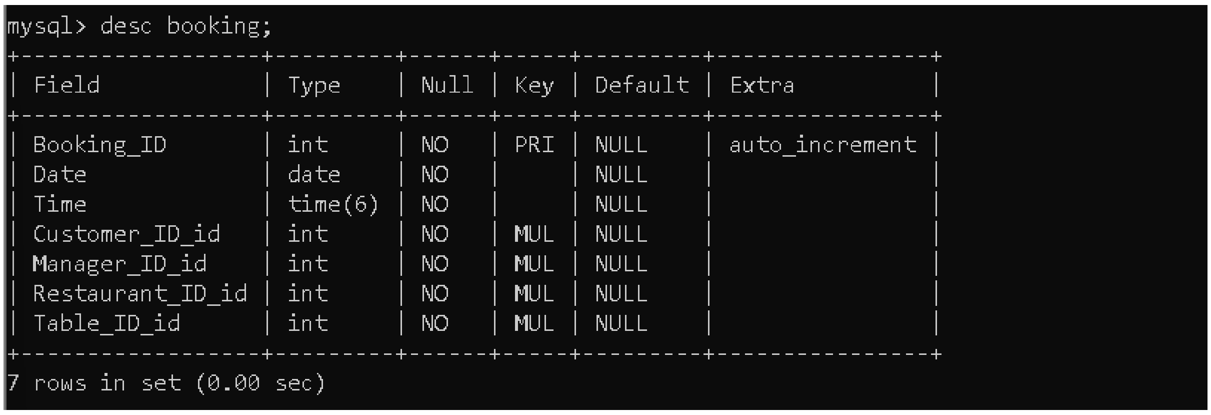
```
mysql> desc menu_nonveg
    -> ;
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| Item_ID_NonVeg | int          | NO   | PRI | NULL    | auto_increment |
| Item_Name      | varchar(50)  | NO   |     | NULL    |                |
| Item_Price     | int          | NO   |     | NULL    |                |
| Cuisine_Type   | varchar(50)  | NO   |     | NULL    |                |
| Image          | varchar(100) | NO   |     | NULL    |                |
+----------------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

Figure 3.11 Menu_NonVeg Table Description

Figure 3.11 shows the description of the table shown in the MySQL command line after creation of table . It shows the attributes in the table and their Types with the constraints.

BOOKING TABLE

```
CREATE TABLE BOOKING
(
        BOOKING ID INTEGER,
        DATE DATE,
        TIME TIME(6),
        CUSTOMER_ID INTEGER,
        MANAGER_ID INTEGER,
        RESTAURANT_ID INTEGER,
        TABLE_ID INTEGER,
        PRIMARY KEY(BOOKING_ID),
        FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER
        (CUSTOMER_ID) ON DELETE CASCADE,
        FOREIGN KEY (MANAGER_ID) REFERENCES MANAGER (MANAGER_ID)
        ON DELETE CASCADE,
        FOREIGN KEY (RESTAURANT_ID) REFERENCES RESTAURANT
        (RESTAURANT_ID) ON DELETE CASCADE,
        FOREIGN KEY (TABLE_ID) REFERENCES TABLE (TABLE_ID) ON DELETE
        CASCADE
);
```



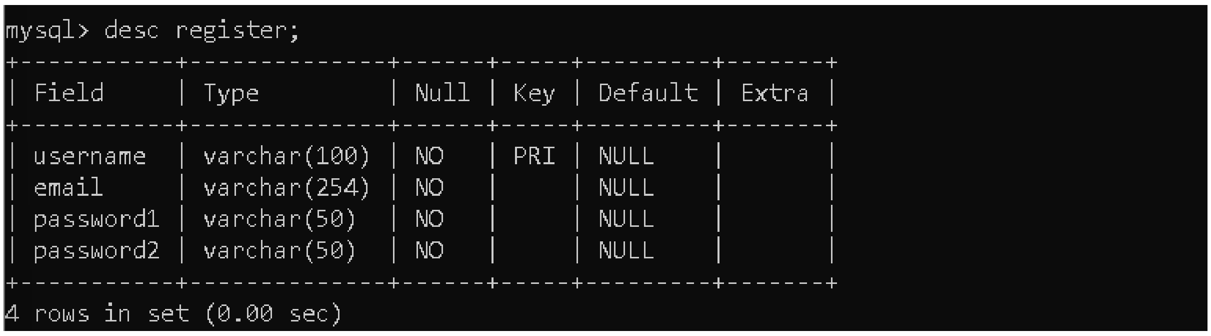
```
mysql> desc booking;
+------------------+---------+------+-----+---------+----------------+
| Field            | Type    | Null | Key | Default | Extra          |
+------------------+---------+------+-----+---------+----------------+
| Booking_ID       | int     | NO   | PRI | NULL    | auto_increment |
| Date             | date    | NO   |     | NULL    |                |
| Time             | time(6) | NO   |     | NULL    |                |
| Customer_ID_id   | int     | NO   | MUL | NULL    |                |
| Manager_ID_id    | int     | NO   | MUL | NULL    |                |
| Restaurant_ID_id | int     | NO   | MUL | NULL    |                |
| Table_ID_id      | int     | NO   | MUL | NULL    |                |
+------------------+---------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

Figure 3.12 Booking Table Description

Figure 3.12 shows the description of the table shown in the MySQL command line after creation of table . It shows the attributes in the table and their Types with the constraints.

REGISTER TABLE

```
CREATE TABLE REGISTER
(
        USERNAME VARCHAR(100) PRIMARY KEY,
        EMAIL VARCHAR(254),
        PASSWORD1 VARCHAR(50),
        PASSWORD2 VARCHAR(50)
);
```



```
mysql> desc register;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| username  | varchar(100) | NO   | PRI | NULL    |       |
| email     | varchar(254) | NO   |     | NULL    |       |
| password1 | varchar(50)  | NO   |     | NULL    |       |
| password2 | varchar(50)  | NO   |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

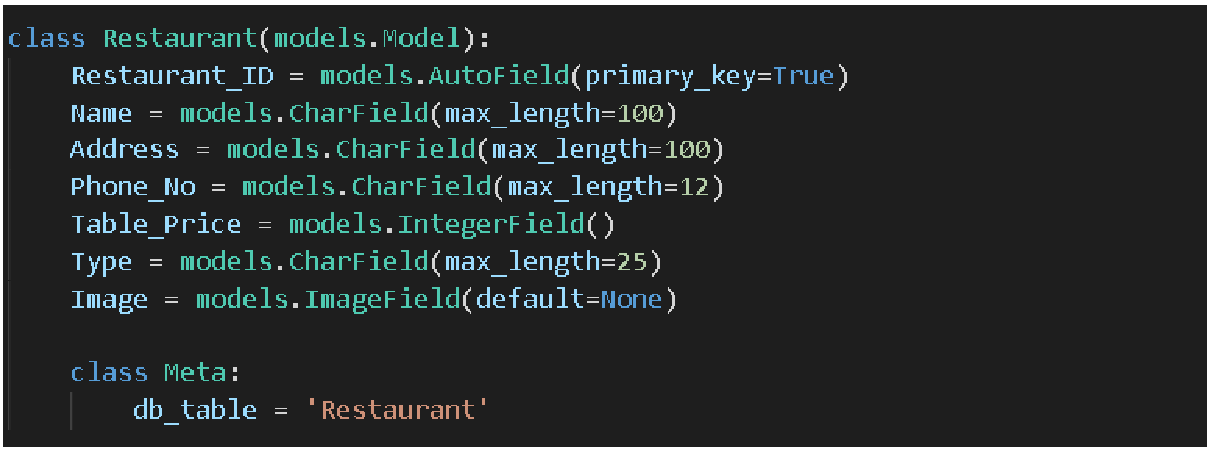
Figure 3.13 Register Table Description

Figure 3.13 shows the description of the table shown in the MySQL command line after creation of table . It shows the attributes in the table and their Types with the constraints.

3.4.2 Create Tables Using Django

Tables are created in Django using classes and models in the models.py file .The above tables are implemented as shown in the code snippets attached below.

RESTAURANT MODEL

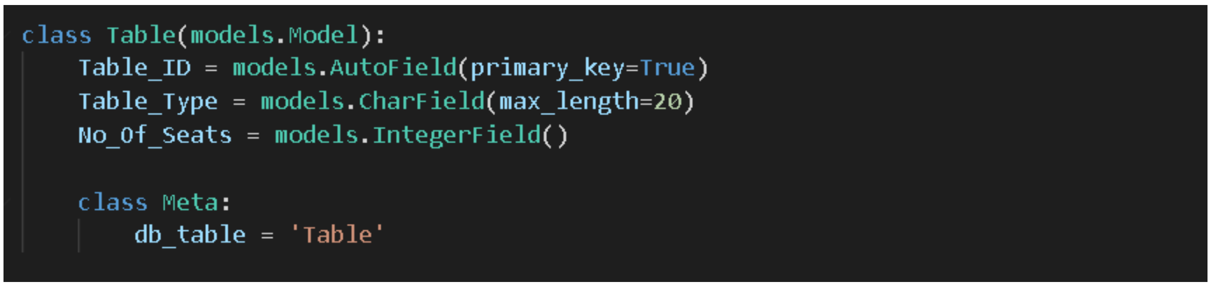


```
class Restaurant(models.Model):
    Restaurant_ID = models.AutoField(primary_key=True)
    Name = models.CharField(max_length=100)
    Address = models.CharField(max_length=100)
    Phone_No = models.CharField(max_length=12)
    Table_Price = models.IntegerField()
    Type = models.CharField(max_length=25)
    Image = models.ImageField(default=None)

    class Meta:
        db_table = 'Restaurant'
```

Figure 3.14 Restaurant Model

Figure 3.14 shows the implementation of this table as class and model in Django.

TABLES MODEL

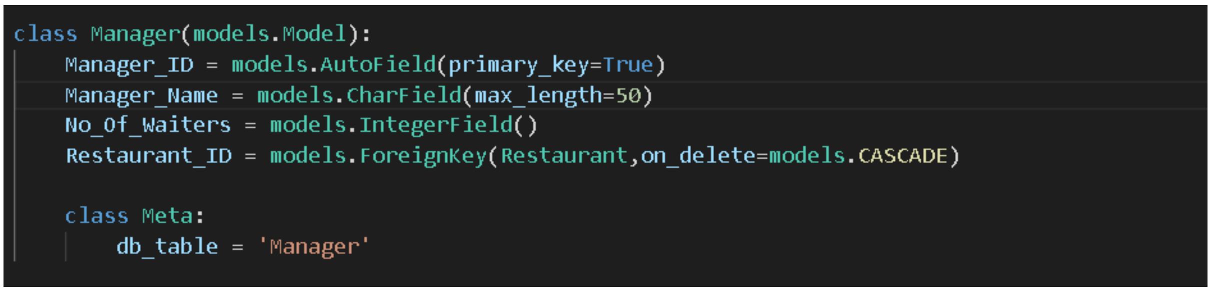


```
class Table(models.Model):
    Table_ID = models.AutoField(primary_key=True)
    Table_Type = models.CharField(max_length=20)
    No_Of_Seats = models.IntegerField()

    class Meta:
        db_table = 'Table'
```

Figure 3.15 Tables Model

Figure 3.15 shows the implementation of this table as class and model in Django.

MANAGER MODEL

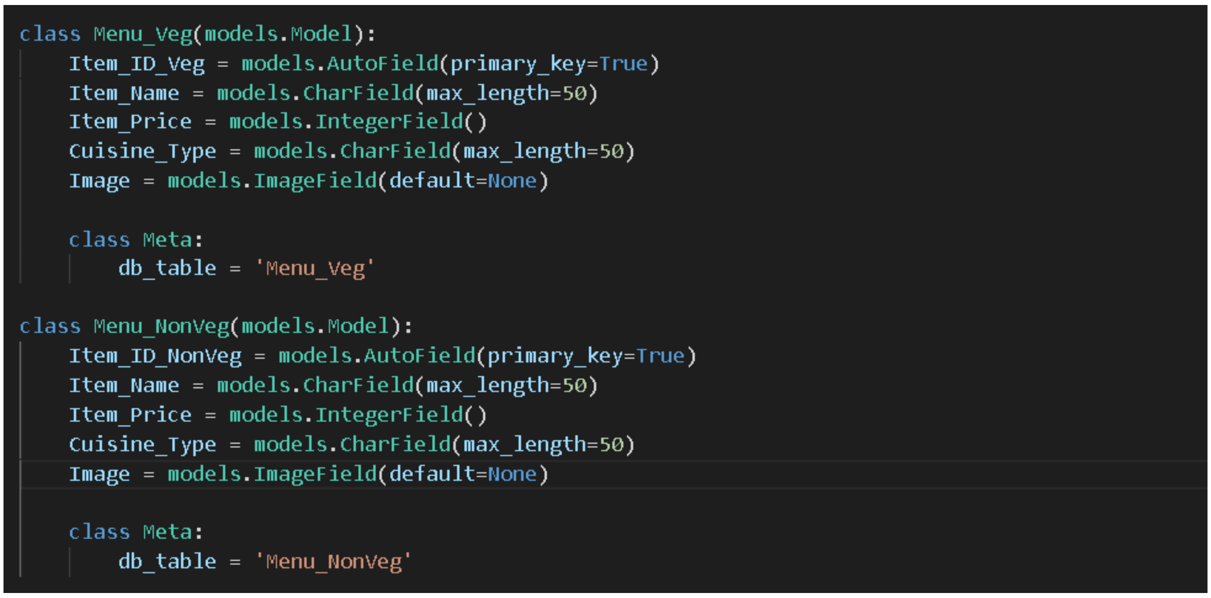


```
class Manager(models.Model):
    Manager_ID = models.AutoField(primary_key=True)
    Manager_Name = models.CharField(max_length=50)
    No_Of_Waiters = models.IntegerField()
    Restaurant_ID = models.ForeignKey(Restaurant,on_delete=models.CASCADE)

    class Meta:
        db_table = 'Manager'
```

Figure 3.16 Manager Model

Figure 3.16 shows the implementation of this table as class and model in Django.

MENU_VEG AND MENU_NONVEG MODELS

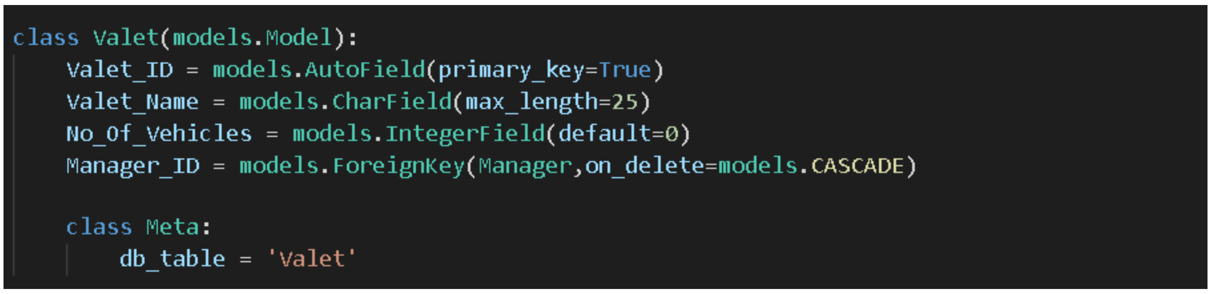


```python
class Menu_Veg(models.Model):
    Item_ID_Veg = models.AutoField(primary_key=True)
    Item_Name = models.CharField(max_length=50)
    Item_Price = models.IntegerField()
    Cuisine_Type = models.CharField(max_length=50)
    Image = models.ImageField(default=None)

    class Meta:
        db_table = 'Menu_Veg'

class Menu_NonVeg(models.Model):
    Item_ID_NonVeg = models.AutoField(primary_key=True)
    Item_Name = models.CharField(max_length=50)
    Item_Price = models.IntegerField()
    Cuisine_Type = models.CharField(max_length=50)
    Image = models.ImageField(default=None)

    class Meta:
        db_table = 'Menu_NonVeg'
```

Figure 3.17 Menu Veg and NonVeg Model

Figure 3.17 shows the implementation of this table as class and model in Django.

VALET MODEL

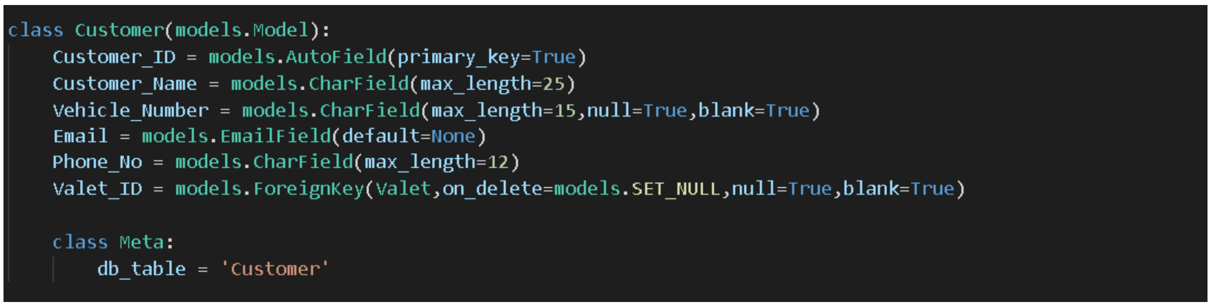


```python
class Valet(models.Model):
    Valet_ID = models.AutoField(primary_key=True)
    Valet_Name = models.CharField(max_length=25)
    No_Of_Vehicles = models.IntegerField(default=0)
    Manager_ID = models.ForeignKey(Manager,on_delete=models.CASCADE)

    class Meta:
        db_table = 'Valet'
```

Figure 3.18 Valet Model

Figure 3.18 shows the implementation of this table as class and model in Django.

CUSTOMER MODEL

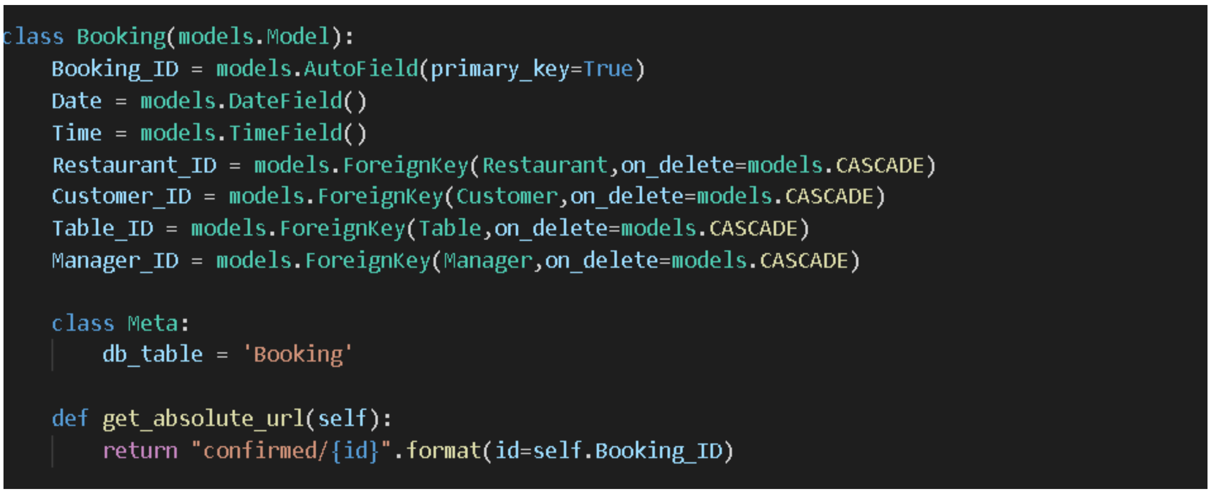


```python
class Customer(models.Model):
    Customer_ID = models.AutoField(primary_key=True)
    Customer_Name = models.CharField(max_length=25)
    Vehicle_Number = models.CharField(max_length=15,null=True,blank=True)
    Email = models.EmailField(default=None)
    Phone_No = models.CharField(max_length=12)
    Valet_ID = models.ForeignKey(Valet,on_delete=models.SET_NULL,null=True,blank=True)

    class Meta:
        db_table = 'Customer'
```

Figure 3.19 Customer Model

Figure 3.19 shows the implementation of this table as class and model in Django.

BOOKING MODEL

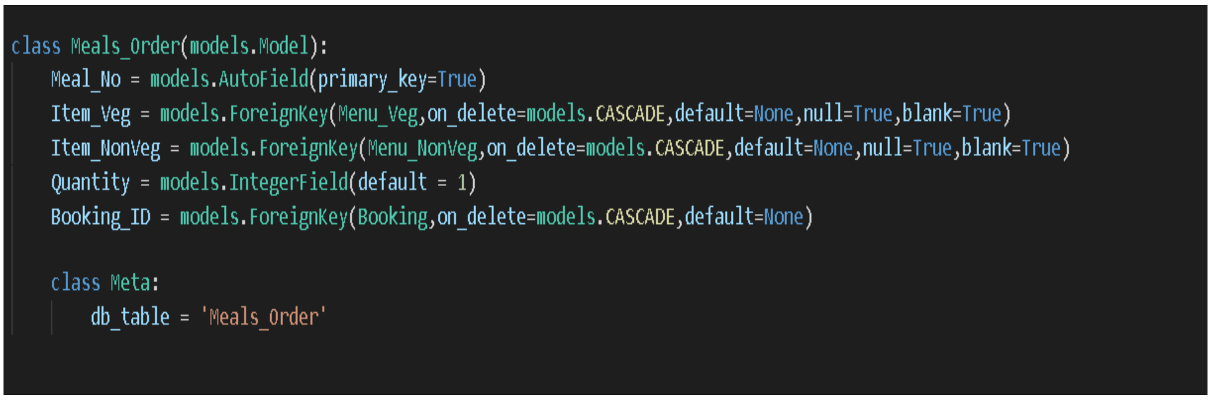


```
class Booking(models.Model):
    Booking_ID = models.AutoField(primary_key=True)
    Date = models.DateField()
    Time = models.TimeField()
    Restaurant_ID = models.ForeignKey(Restaurant,on_delete=models.CASCADE)
    Customer_ID = models.ForeignKey(Customer,on_delete=models.CASCADE)
    Table_ID = models.ForeignKey(Table,on_delete=models.CASCADE)
    Manager_ID = models.ForeignKey(Manager,on_delete=models.CASCADE)

    class Meta:
        db_table = 'Booking'

    def get_absolute_url(self):
        return "confirmed/{id}".format(id=self.Booking_ID)
```

Figure 3.20 Booking Model

Figure 3.20 shows the implementation of this table as class and model in Django.

MEALS_ORDER MODEL



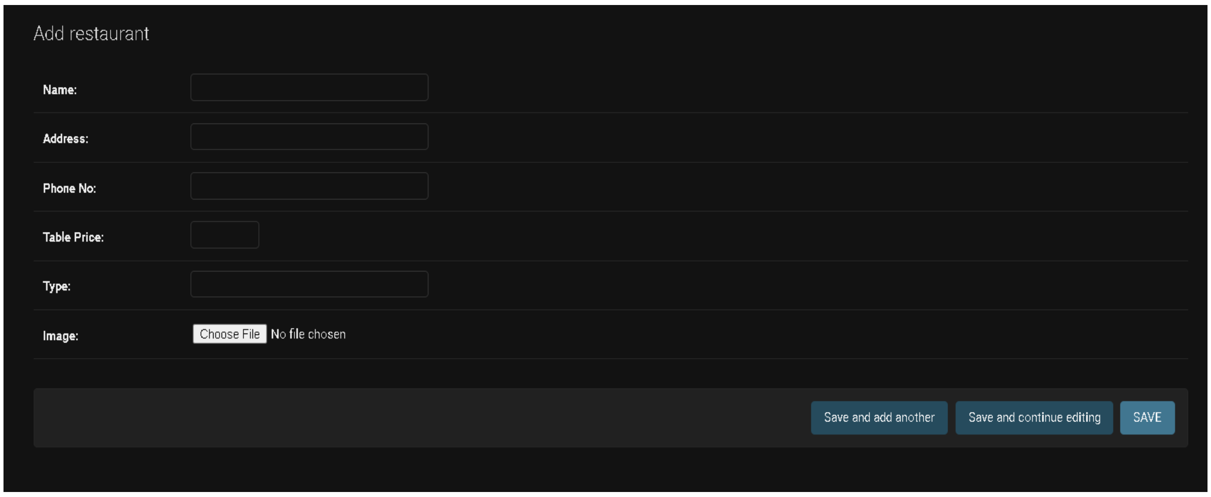
```
class Meals_Order(models.Model):
    Meal_No = models.AutoField(primary_key=True)
    Item_Veg = models.ForeignKey(Menu_Veg,on_delete=models.CASCADE,default=None,null=True,blank=True)
    Item_NonVeg = models.ForeignKey(Menu_NonVeg,on_delete=models.CASCADE,default=None,null=True,blank=True)
    Quantity = models.IntegerField(default = 1)
    Booking_ID = models.ForeignKey(Booking,on_delete=models.CASCADE,default=None)

    class Meta:
        db_table = 'Meals_Order'
```

Figure 3.21 Meals_Order Model

Figure 3.21 shows the implementation of this table as class and model in Django.

3.4.3 Insertion

In our system the insertion is implemented using the admins page and not by using SQL commands and screenshots are as shown below. You can see all the fields of the Restaurant table which need to be inserted into the table and this system gives a nice GUI for you to interact and insert the required fields for insertion of a new restaurant into the Database.



Add restaurant

Name:

Address:

Phone No:

Table Price:

Type:

Image: Choose File No file chosen

Figure 3.22 Insertion using admin page

Figure 3.22 the image for insertion of restaurant and is done using the GUI done using DJANGO.

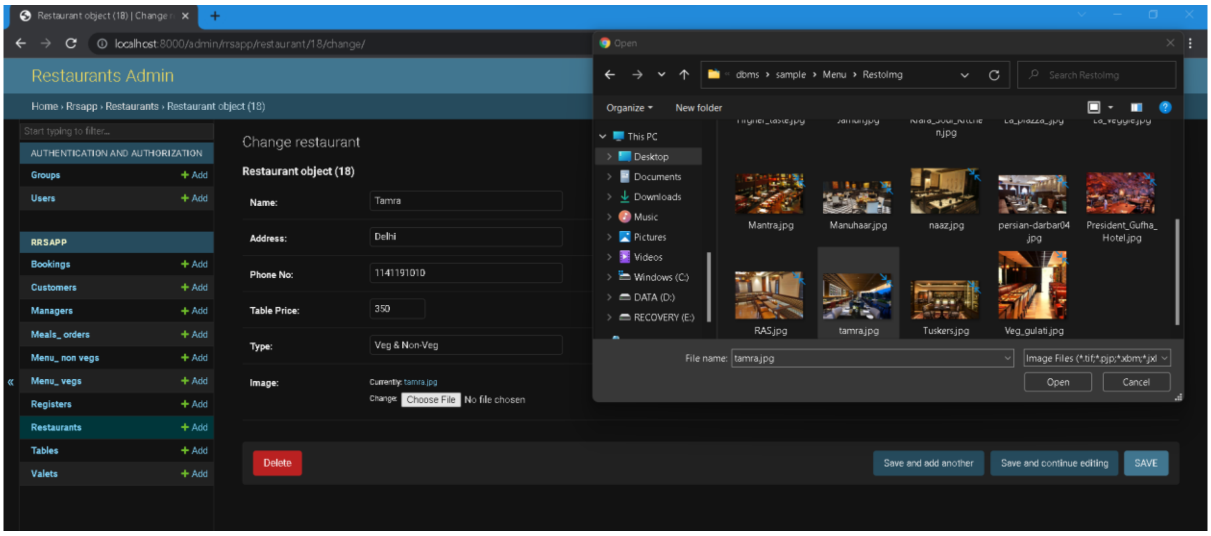
Similarly, all the other table insertions can be done using the admin page.

Exceptions are the Booking and the BOOKING , CUSTOMER and REGISTER tables which can be inserted from the front end and the admin page GUI provide.

Some table insertion can also be done using SQL commands like for tables like Manager, Valet, Table etc. using the commands as shown below.

- INSERT INTO MANAGER VALUES (&MANAGER_ID,"&MANAGER_NAME",&No_Of_Waiters, &RESTAURANT_ID);
- INSERT INTO VALET VALUES (&VALET_ID,"&VALET_NAME",&No_Of_Vehicles,&MANAGER_ID);
- INSERT INTO TABLES VALUES (&TABLE_ID,"&TABLE_TYPE",&NO_OF_SEATS);

Some tables have images to be inserted in them and can be best inserted using the admin GUI as selecting the images from the PC becomes very easy as shown in the snapshot below for the Restaurant Insertion.



Change restaurant

Restaurant object (18)

Name: Tamra

Address: Delhi

Phone No: 1141191010

Table Price: 350

Type: Veg & Non-Veg

Image: Currently: tamra.jpg
Change: Choose File No file chosen

File name: tamra.jpg
Image Files (*.tif;*.pjp;*.xbm;*.jxl

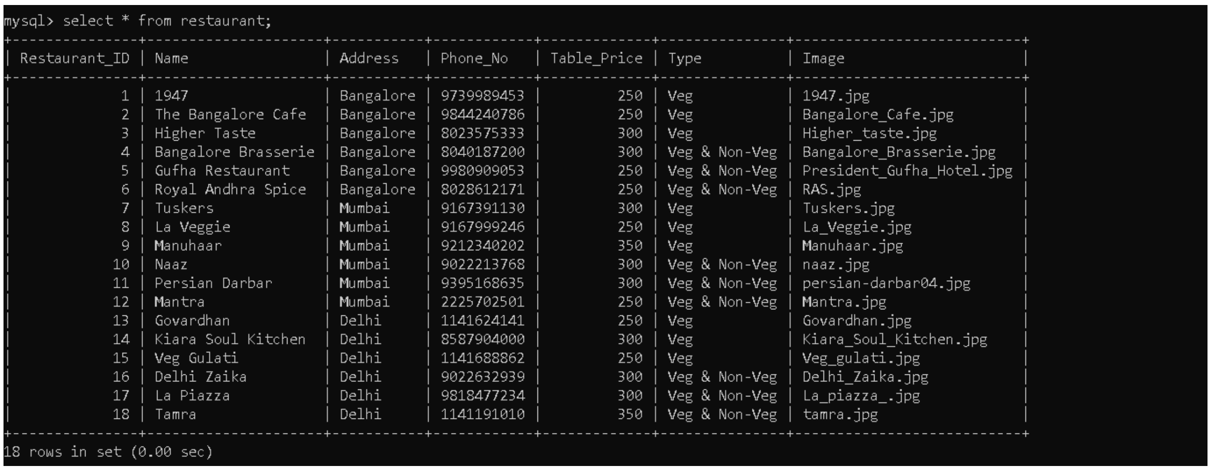
Figure 3.23 Insertion of Image

Figure 3.23 shows the insertion of an image to the restaurant table done from the admin page.

3.4.3.1 Tables Example after Insertions

Here is an example of the Restaurant table , Menu Tables after insertion from the GUI admin page.

Restaurant Table

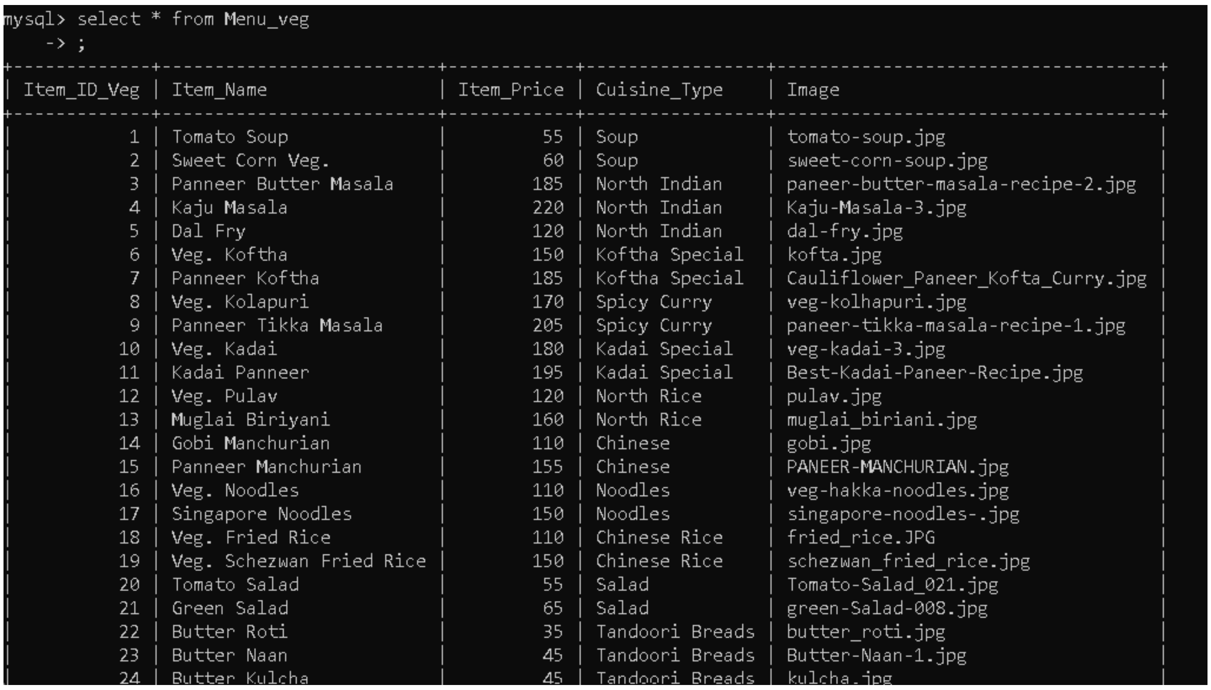


```
mysql> select * from restaurant;
+---------------+-----------------------+-----------+------------+-------------+---------------+----------------------------+
| Restaurant_ID | Name                  | Address   | Phone_No   | Table_Price | Type          | Image                      |
+---------------+-----------------------+-----------+------------+-------------+---------------+----------------------------+
|             1 | 1947                  | Bangalore | 9739989453 |         250 | Veg           | 1947.jpg                   |
|             2 | The Bangalore Cafe    | Bangalore | 9844240786 |         250 | Veg           | Bangalore_Cafe.jpg         |
|             3 | Higher Taste          | Bangalore | 8023575333 |         300 | Veg           | Higher_taste.jpg           |
|             4 | Bangalore Brasserie   | Bangalore | 8040187200 |         300 | Veg & Non-Veg | Bangalore_Brasserie.jpg    |
|             5 | Gufha Restaurant      | Bangalore | 9980909053 |         250 | Veg & Non-Veg | President_Gufha_Hotel.jpg  |
|             6 | Royal Andhra Spice    | Bangalore | 8028612171 |         250 | Veg & Non-Veg | RAS.jpg                    |
|             7 | Tuskers               | Mumbai    | 9167391130 |         300 | Veg           | Tuskers.jpg                |
|             8 | La Veggie             | Mumbai    | 9167999246 |         250 | Veg           | La_Veggie.jpg              |
|             9 | Manuhaar              | Mumbai    | 9212340202 |         350 | Veg           | Manuhaar.jpg               |
|            10 | Naaz                  | Mumbai    | 9022213768 |         300 | Veg & Non-Veg | naaz.jpg                   |
|            11 | Persian Darbar        | Mumbai    | 9395168635 |         300 | Veg & Non-Veg | persian-darbar04.jpg       |
|            12 | Mantra                | Mumbai    | 2225702501 |         250 | Veg & Non-Veg | Mantra.jpg                 |
|            13 | Govardhan             | Delhi     | 1141624141 |         250 | Veg           | Govardhan.jpg              |
|            14 | Kiara Soul Kitchen    | Delhi     | 8587904000 |         300 | Veg           | Kiara_Soul_Kitchen.jpg     |
|            15 | Veg Gulati            | Delhi     | 1141688862 |         250 | Veg           | Veg_gulati.jpg             |
|            16 | Delhi Zaika           | Delhi     | 9022632939 |         300 | Veg & Non-Veg | Delhi_Zaika.jpg            |
|            17 | La Piazza             | Delhi     | 9818477234 |         300 | Veg & Non-Veg | La_piazza_.jpg             |
|            18 | Tamra                 | Delhi     | 1141191010 |         350 | Veg & Non-Veg | tamra.jpg                  |
+---------------+-----------------------+-----------+------------+-------------+---------------+----------------------------+
18 rows in set (0.00 sec)
```

Figure 3.24 Restaurant table data

Figure 3.24 shows the data in the restaurant table after insertion. It contains the data of all the restaurants available in the database.

Menu_Veg Table

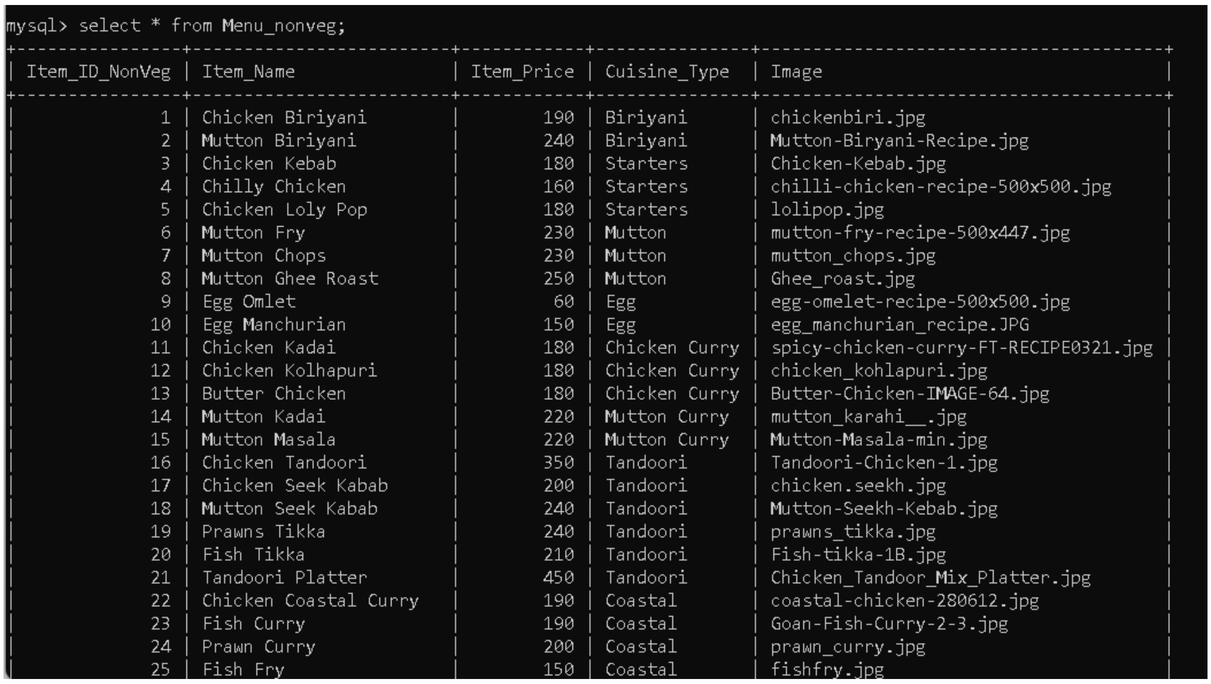


```
mysql> select * from Menu_veg
    -> ;
+-------------+--------------------------+------------+-----------------+------------------------------------+
| Item_ID_Veg | Item_Name                | Item_Price | Cuisine_Type    | Image                              |
+-------------+--------------------------+------------+-----------------+------------------------------------+
|           1 | Tomato Soup              |         55 | Soup            | tomato-soup.jpg                    |
|           2 | Sweet Corn Veg.          |         60 | Soup            | sweet-corn-soup.jpg                |
|           3 | Panneer Butter Masala    |        185 | North Indian    | paneer-butter-masala-recipe-2.jpg  |
|           4 | Kaju Masala              |        220 | North Indian    | Kaju-Masala-3.jpg                  |
|           5 | Dal Fry                  |        120 | North Indian    | dal-fry.jpg                        |
|           6 | Veg. Koftha              |        150 | Koftha Special  | kofta.jpg                          |
|           7 | Panneer Koftha           |        185 | Koftha Special  | Cauliflower_Paneer_Kofta_Curry.jpg |
|           8 | Veg. Kolapuri            |        170 | Spicy Curry     | veg-kolhapuri.jpg                  |
|           9 | Panneer Tikka Masala     |        205 | Spicy Curry     | paneer-tikka-masala-recipe-1.jpg   |
|          10 | Veg. Kadai               |        180 | Kadai Special   | veg-kadai-3.jpg                    |
|          11 | Kadai Panneer            |        195 | Kadai Special   | Best-Kadai-Paneer-Recipe.jpg       |
|          12 | Veg. Pulav               |        120 | North Rice      | pulav.jpg                          |
|          13 | Muglai Biriyani          |        160 | North Rice      | muglai_biriani.jpg                 |
|          14 | Gobi Manchurian          |        110 | Chinese         | gobi.jpg                           |
|          15 | Panneer Manchurian       |        155 | Chinese         | PANEER-MANCHURIAN.jpg              |
|          16 | Veg. Noodles             |        110 | Noodles         | veg-hakka-noodles.jpg              |
|          17 | Singapore Noodles        |        150 | Noodles         | singapore-noodles-.jpg             |
|          18 | Veg. Fried Rice          |        110 | Chinese Rice    | fried_rice.JPG                     |
|          19 | Veg. Schezwan Fried Rice |        150 | Chinese Rice    | schezwan_fried_rice.jpg            |
|          20 | Tomato Salad             |         55 | Salad           | Tomato-Salad_021.jpg               |
|          21 | Green Salad              |         65 | Salad           | green-Salad-008.jpg                |
|          22 | Butter Roti              |         35 | Tandoori Breads | butter_roti.jpg                    |
|          23 | Butter Naan              |         45 | Tandoori Breads | Butter-Naan-1.jpg                  |
|          24 | Butter Kulcha            |         45 | Tandoori Breads | kulcha.jpg                         |
```

Figure 3.25 Menu_Veg table data

Figure 3.25 shows the data in the Menu_Veg table after insertion. It contains the data of all the items available in the database.

Menu_NonVeg Table



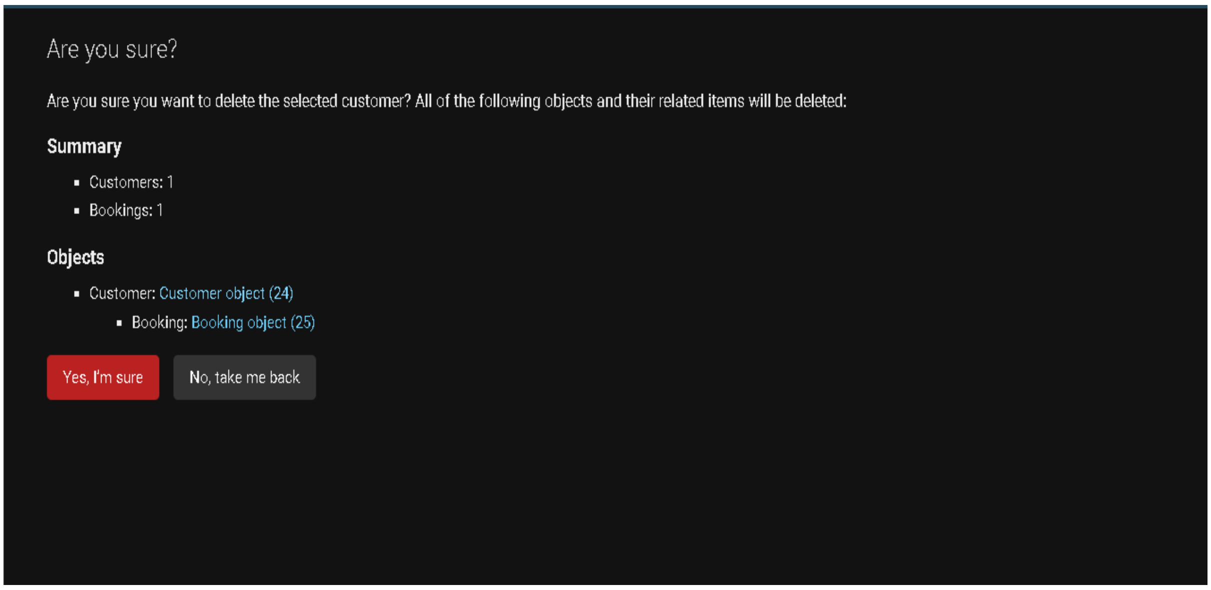
```
mysql> select * from Menu_nonveg;
+----------------+-------------------------+------------+---------------+---------------------------------------+
| Item_ID_NonVeg | Item_Name               | Item_Price | Cuisine_Type  | Image                                 |
+----------------+-------------------------+------------+---------------+---------------------------------------+
|              1 | Chicken Biriyani        |        190 | Biriyani      | chickenbiri.jpg                       |
|              2 | Mutton Biriyani         |        240 | Biriyani      | Mutton-Biryani-Recipe.jpg             |
|              3 | Chicken Kebab           |        180 | Starters      | Chicken-Kebab.jpg                     |
|              4 | Chilly Chicken          |        160 | Starters      | chilli-chicken-recipe-500x500.jpg     |
|              5 | Chicken Loly Pop        |        180 | Starters      | lolipop.jpg                           |
|              6 | Mutton Fry              |        230 | Mutton        | mutton-fry-recipe-500x447.jpg         |
|              7 | Mutton Chops            |        230 | Mutton        | mutton_chops.jpg                      |
|              8 | Mutton Ghee Roast       |        250 | Mutton        | Ghee_roast.jpg                        |
|              9 | Egg Omlet               |         60 | Egg           | egg-omelet-recipe-500x500.jpg         |
|             10 | Egg Manchurian          |        150 | Egg           | egg_manchurian_recipe.JPG             |
|             11 | Chicken Kadai           |        180 | Chicken Curry | spicy-chicken-curry-FT-RECIPE0321.jpg |
|             12 | Chicken Kolhapuri       |        180 | Chicken Curry | chicken_kohlapuri.jpg                 |
|             13 | Butter Chicken          |        180 | Chicken Curry | Butter-Chicken-IMAGE-64.jpg           |
|             14 | Mutton Kadai            |        220 | Mutton Curry  | mutton_karahi__.jpg                   |
|             15 | Mutton Masala           |        220 | Mutton Curry  | Mutton-Masala-min.jpg                 |
|             16 | Chicken Tandoori        |        350 | Tandoori      | Tandoori-Chicken-1.jpg                |
|             17 | Chicken Seek Kabab      |        200 | Tandoori      | chicken.seekh.jpg                     |
|             18 | Mutton Seek Kabab       |        240 | Tandoori      | Mutton-Seekh-Kebab.jpg                |
|             19 | Prawns Tikka            |        240 | Tandoori      | prawns_tikka.jpg                      |
|             20 | Fish Tikka              |        210 | Tandoori      | Fish-tikka-1B.jpg                     |
|             21 | Tandoori Platter        |        450 | Tandoori      | Chicken_Tandoor_Mix_Platter.jpg       |
|             22 | Chicken Coastal Curry   |        190 | Coastal       | coastal-chicken-280612.jpg            |
|             23 | Fish Curry              |        190 | Coastal       | Goan-Fish-Curry-2-3.jpg               |
|             24 | Prawn Curry             |        200 | Coastal       | prawn_curry.jpg                       |
|             25 | Fish Fry                |        150 | Coastal       | fishfry.jpg                           |
```

Figure 3.26 Menu_NonVeg table data

Figure 3.26 shows the data in the Menu_NonVeg table after insertion. It contains the data of all the items available in the database.

3.4.4 Deletion & Updating

Deletion can be done for some tables using commands similar to the one shown below
DELETE FROM VALET WHERE VALET_ID=”………………”;
Updating can be done in the admin page and the webapp provided to the restaurant and the users using the Django database .delete() and .update() tools.



Are you sure?

Are you sure you want to delete the selected customer? All of the following objects and their related items will be deleted:

Summary

- Customers: 1
- Bookings: 1

Objects

- Customer: Customer object (24)
    - Booking: Booking object (25)

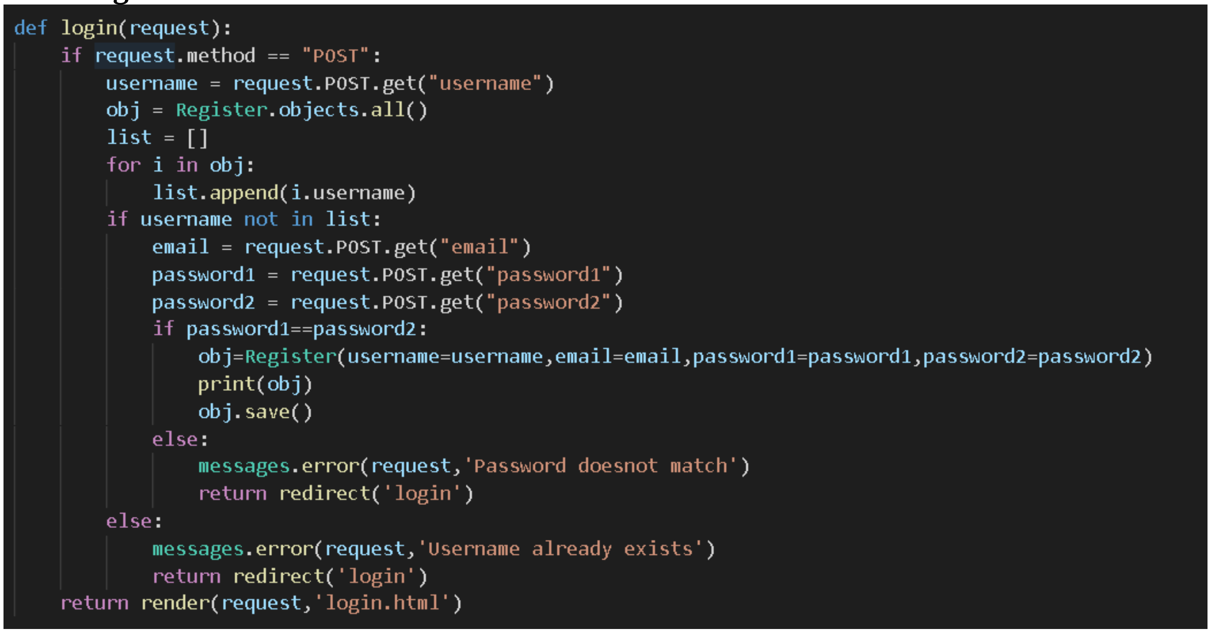
Figure 3.27 Deletion

Figure 3.26 Here is an example of deleting the Customer from admin page which in turn deletes the Booking done by that customer as ON DELETE CASCADE is used.

3.5 Functions

In Django the logic like Stored procedures or the Triggers and all other logical implementations are done using the views.py file by writing appropriate python functions to carry out the respective tasks as shown in the snapshots below for further information or source code visit the GitHub link provided at the end of the report.
Given below are few of the functional implementations of our project .

3.5.1 Registration

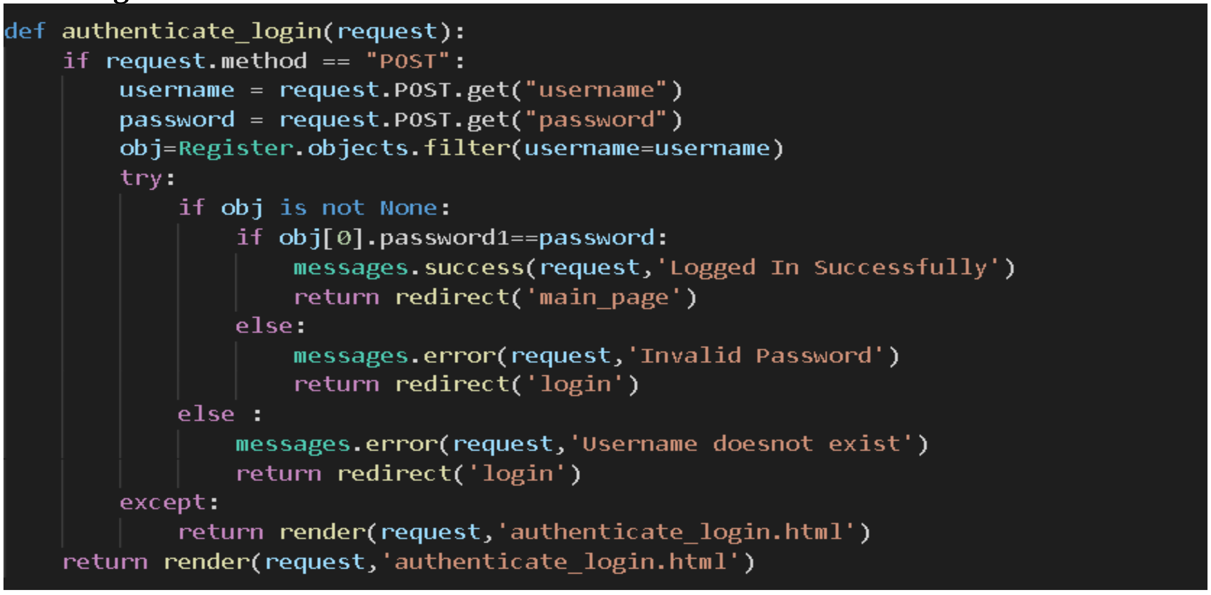


```python
def login(request):
    if request.method == "POST":
        username = request.POST.get("username")
        obj = Register.objects.all()
        list = []
        for i in obj:
            list.append(i.username)
        if username not in list:
            email = request.POST.get("email")
            password1 = request.POST.get("password1")
            password2 = request.POST.get("password2")
            if password1==password2:
                obj=Register(username=username,email=email,password1=password1,password2=password2)
                print(obj)
                obj.save()
            else:
                messages.error(request,'Password doesnot match')
                return redirect('login')
        else:
            messages.error(request,'Username already exists')
            return redirect('login')
    return render(request,'login.html')
```

Figure 3.28 Registration function

Figure 3.27 Shows the function used for the implementation in the webpage for the given topic

3.5.2 Login

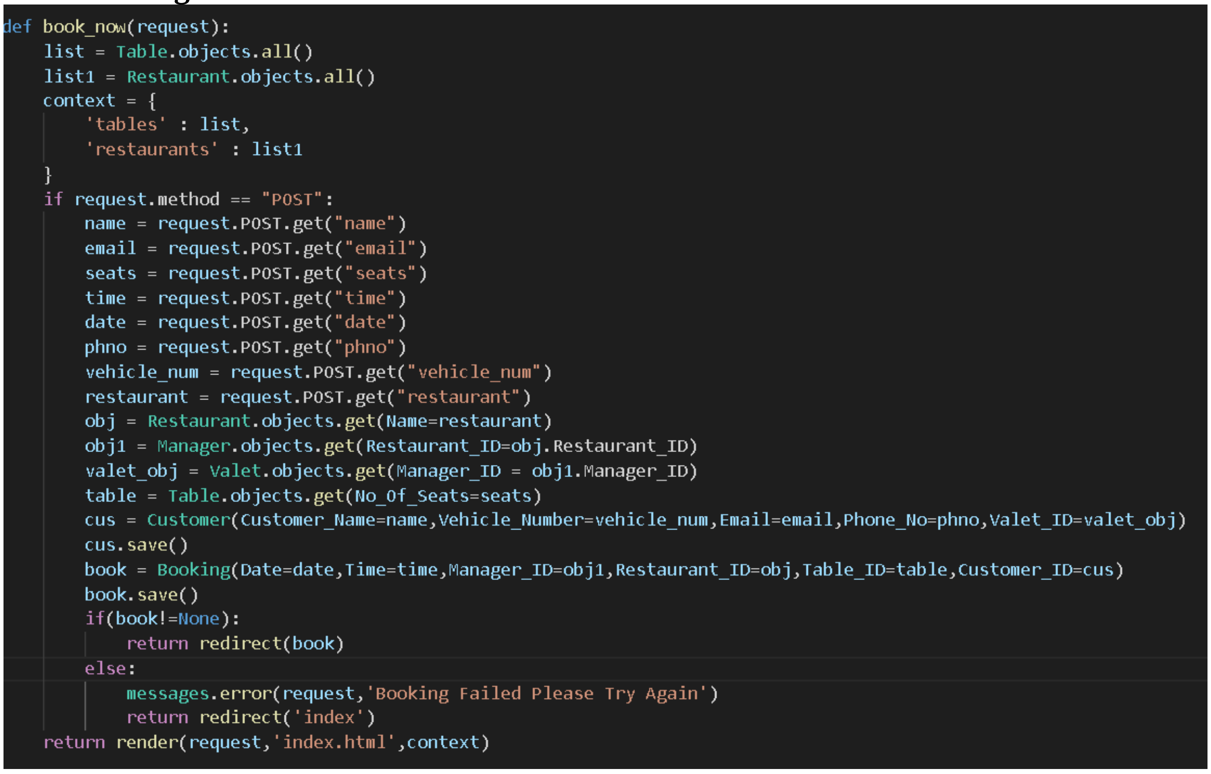


```python
def authenticate_login(request):
    if request.method == "POST":
        username = request.POST.get("username")
        password = request.POST.get("password")
        obj=Register.objects.filter(username=username)
        try:
            if obj is not None:
                if obj[0].password1==password:
                    messages.success(request,'Logged In Successfully')
                    return redirect('main_page')
                else:
                    messages.error(request,'Invalid Password')
                    return redirect('login')
            else :
                messages.error(request,'Username doesnot exist')
                return redirect('login')
        except:
            return render(request,'authenticate_login.html')
    return render(request,'authenticate_login.html')
```

Figure 3.29 Login function

Figure 3.28 Shows the function used for the implementation in the webpage for the given topic

3.5.3 Booking

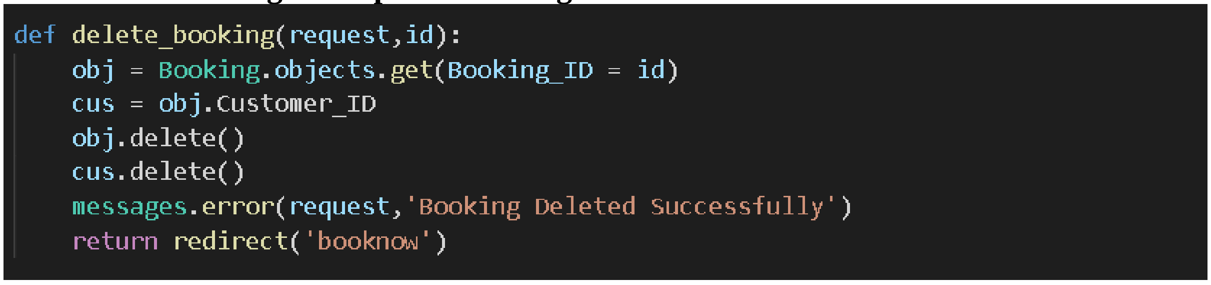


```
def book_now(request):
    list = Table.objects.all()
    list1 = Restaurant.objects.all()
    context = {
        'tables' : list,
        'restaurants' : list1
    }
    if request.method == "POST":
        name = request.POST.get("name")
        email = request.POST.get("email")
        seats = request.POST.get("seats")
        time = request.POST.get("time")
        date = request.POST.get("date")
        phno = request.POST.get("phno")
        vehicle_num = request.POST.get("vehicle_num")
        restaurant = request.POST.get("restaurant")
        obj = Restaurant.objects.get(Name=restaurant)
        obj1 = Manager.objects.get(Restaurant_ID=obj.Restaurant_ID)
        valet_obj = Valet.objects.get(Manager_ID = obj1.Manager_ID)
        table = Table.objects.get(No_Of_Seats=seats)
        cus = Customer(Customer_Name=name,Vehicle_Number=vehicle_num,Email=email,Phone_No=phno,Valet_ID=valet_obj)
        cus.save()
        book = Booking(Date=date,Time=time,Manager_ID=obj1,Restaurant_ID=obj,Table_ID=table,Customer_ID=cus)
        book.save()
        if(book!=None):
            return redirect(book)
        else:
            messages.error(request,'Booking Failed Please Try Again')
            return redirect('index')
    return render(request,'index.html',context)
```

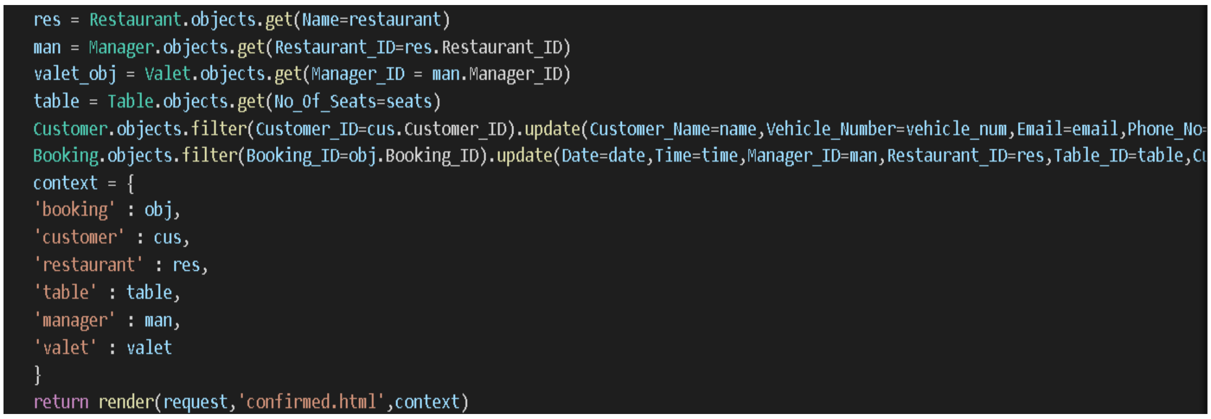
Figure 3.30 Booking function

Figure 3.29 Shows the function used for the implementation in the webpage for the given topic

3.5.4 Delete Booking and Update Bookings



```
def delete_booking(request,id):
    obj = Booking.objects.get(Booking_ID = id)
    cus = obj.Customer_ID
    obj.delete()
    cus.delete()
    messages.error(request,'Booking Deleted Successfully')
    return redirect('booknow')
```



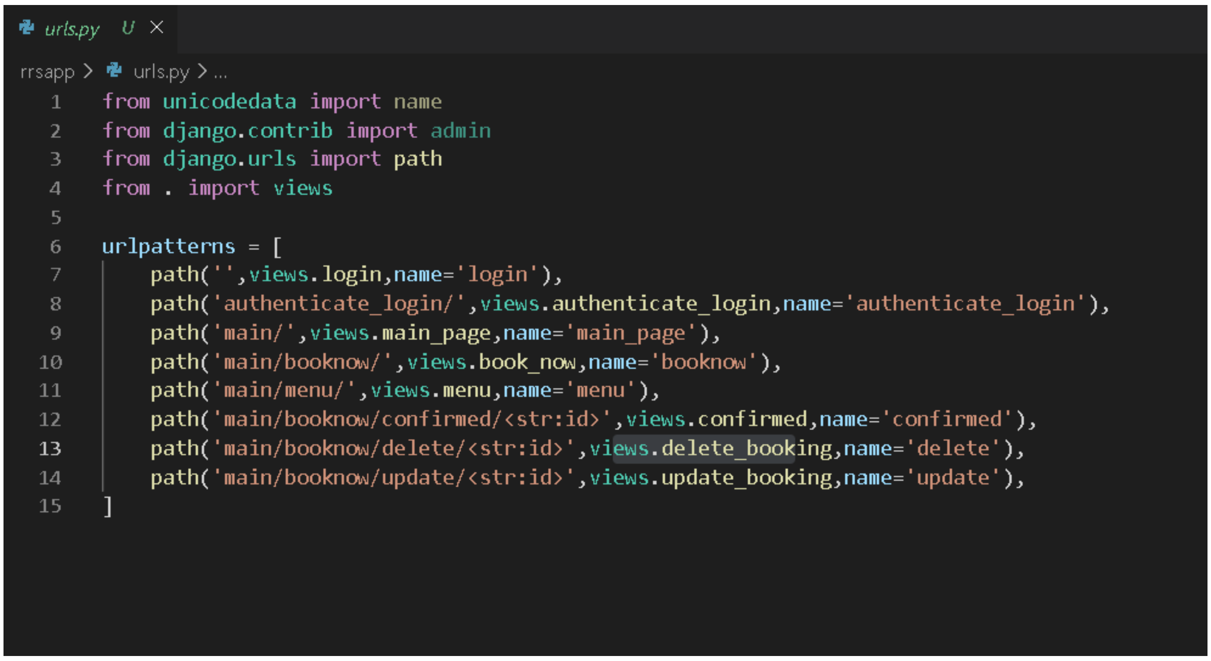
```
    res = Restaurant.objects.get(Name=restaurant)
    man = Manager.objects.get(Restaurant_ID=res.Restaurant_ID)
    valet_obj = Valet.objects.get(Manager_ID = man.Manager_ID)
    table = Table.objects.get(No_Of_Seats=seats)
    Customer.objects.filter(Customer_ID=cus.Customer_ID).update(Customer_Name=name,Vehicle_Number=vehicle_num,Email=email,Phone_No=
    Booking.objects.filter(Booking_ID=obj.Booking_ID).update(Date=date,Time=time,Manager_ID=man,Restaurant_ID=res,Table_ID=table,Cu
    context = {
    'booking' : obj,
    'customer' : cus,
    'restaurant' : res,
    'table' : table,
    'manager' : man,
    'valet' : valet
    }
    return render(request,'confirmed.html',context)
```

Figure 3.31 Delete and Update function

Figure 3.30 Shows the function used for the implementation in the webpage for the given topic

3.6 URL Mappings

As this is a web application, we need URL mappings to handle all the different html templates rendering using regex and dynamic URL mappings and this functionality is implemented using the urls.py file.



urls.py U ×

rrsapp > urls.py > ...

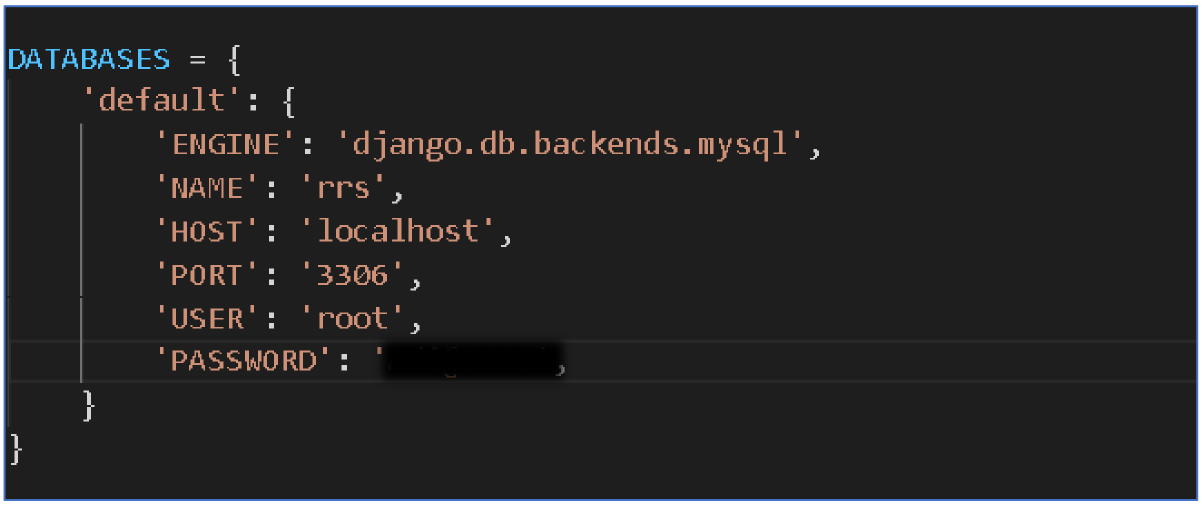
```python
from unicodedata import name
from django.contrib import admin
from django.urls import path
from . import views

urlpatterns = [
    path('',views.login,name='login'),
    path('authenticate_login/',views.authenticate_login,name='authenticate_login'),
    path('main/',views.main_page,name='main_page'),
    path('main/booknow/',views.book_now,name='booknow'),
    path('main/menu/',views.menu,name='menu'),
    path('main/booknow/confirmed/<str:id>',views.confirmed,name='confirmed'),
    path('main/booknow/delete/<str:id>',views.delete_booking,name='delete'),
    path('main/booknow/update/<str:id>',views.update_booking,name='update'),
]
```

Figure 3.32 URL routings

Figure 3.31 as you can see various paths are given for various URL routings.

3.7 Database connection to Django

The database is connected using the settings.py file and migrations are done to the database using ‘python manage.py makemigrations’ followed by ‘python manage.py migrate’ commands.



```python
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'rrs',
        'HOST': 'localhost',
        'PORT': '3306',
        'USER': 'root',
        'PASSWORD': '[redacted]',
    }
}
```

Figure 3.33 Database connection

Figure 3.32 This snippet shows the commands used to link the database to the Django app.

Chapter 4

Results and Discussions

The project is compiled and executed using Python and Django . We have put in few screen shots here to show the working of our application.

Main Webapp Screen Shots

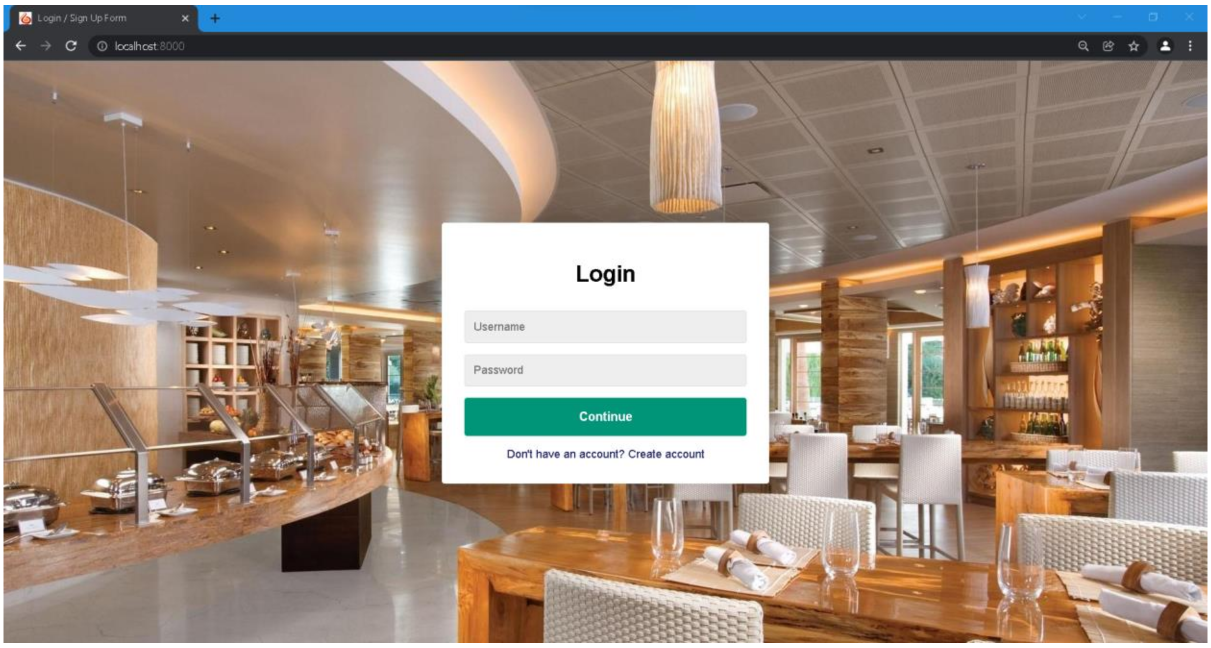


Figure 4.1 Login Page

Figure 4.1 : Here the user can login to see the main page of the webapp. This page also consists of a link to create a new account for the user if they don’t have the account.

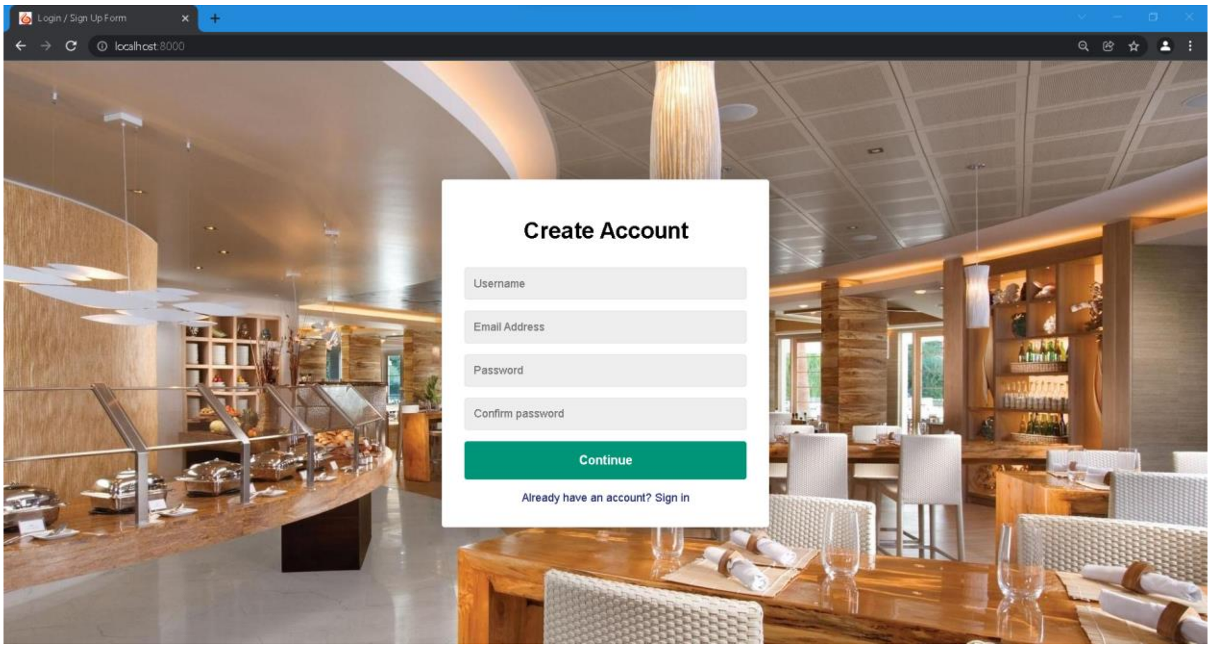


Figure 4.2 User Registration Page

Figure 4.2 : Here the user can register and create a new account for himself to access the pages after login.

Main page SS

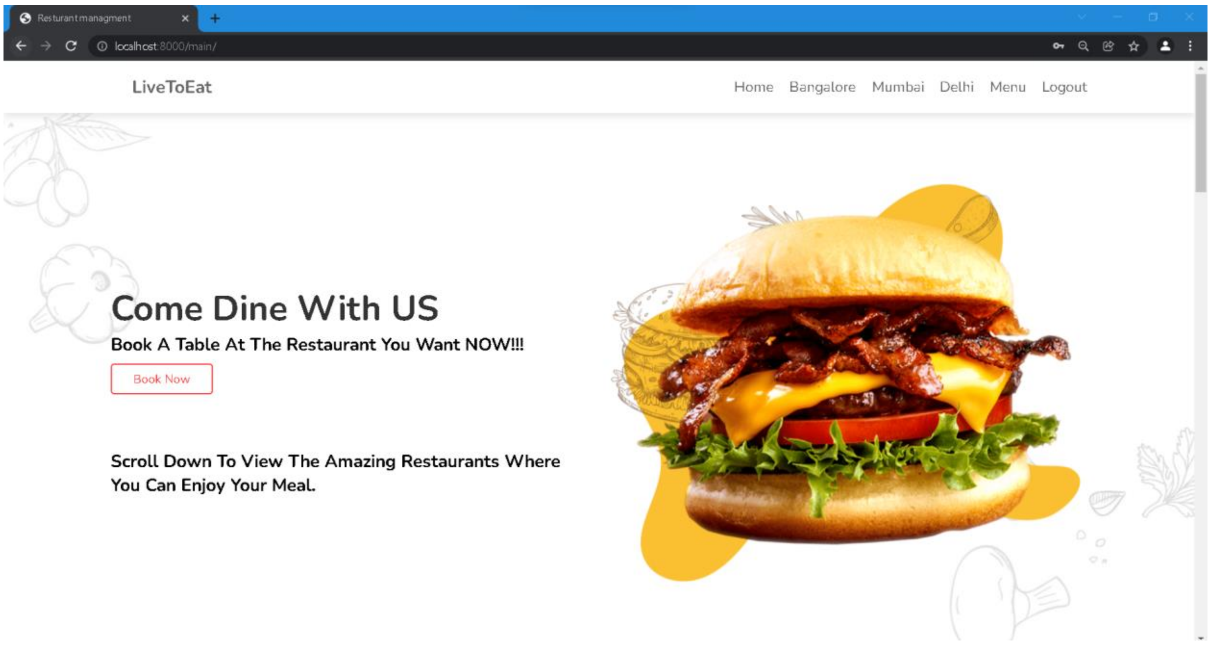
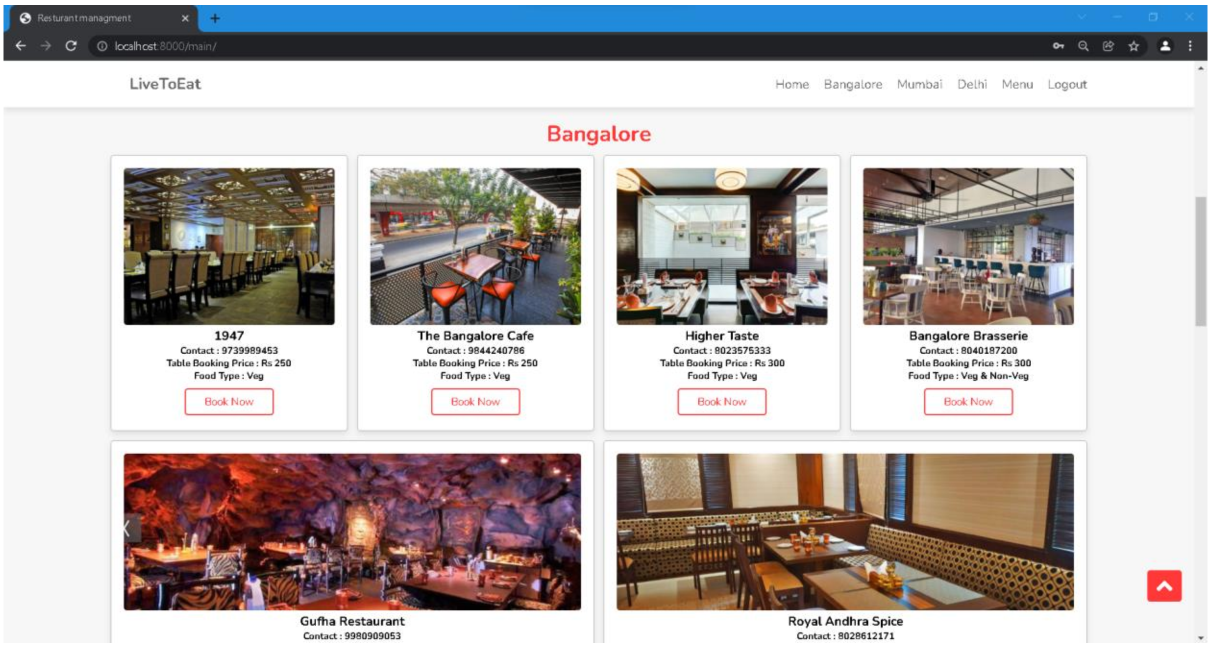

Resturant managment
localhost:8000/main/
LiveToEat
Home
Bangalore
Mumbai
Delhi
Menu
Logout
Come Dine With US
Book A Table At The Restaurant You Want NOW!!!
Book Now
Scroll Down To View The Amazing Restaurants Where You Can Enjoy Your Meal.

Figure 4.3 Main Page

Figure 4.3 is the main page that is displayed as soon as the user logs in to the webapp.



Bangalore

1947
Contact : 9739989453
Table Booking Price : Rs 250
Food Type : Veg

The Bangalore Cafe
Contact : 9844240786
Table Booking Price : Rs 250
Food Type : Veg

Higher Taste
Contact : 8023575333
Table Booking Price : Rs 300
Food Type : Veg

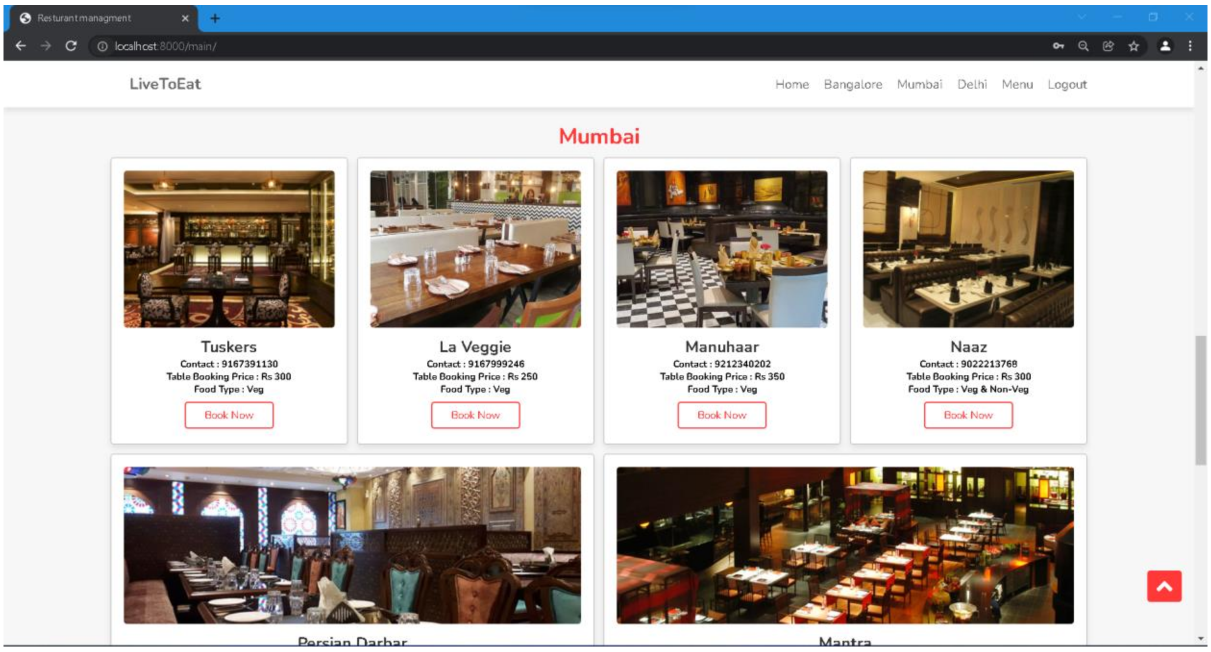
Bangalore Brasserie
Contact : 8040187200
Table Booking Price : Rs 300
Food Type : Veg & Non-Veg

Gufha Restaurant
Contact : 9980909053

Royal Andhra Spice
Contact : 8028612171

Figure 4.4 Main Page Bangalore Restaurants

Figure 4.4 contains the restaurants that is queried from the database using the address of the restaurants and this contains the restaurants that are in Bangalore.



Mumbai

Tuskers
Contact : 9167391130
Table Booking Price : Rs 300
Food Type : Veg

La Veggie
Contact : 9167999246
Table Booking Price : Rs 250
Food Type : Veg

Manuhaar
Contact : 9212340202
Table Booking Price : Rs 350
Food Type : Veg

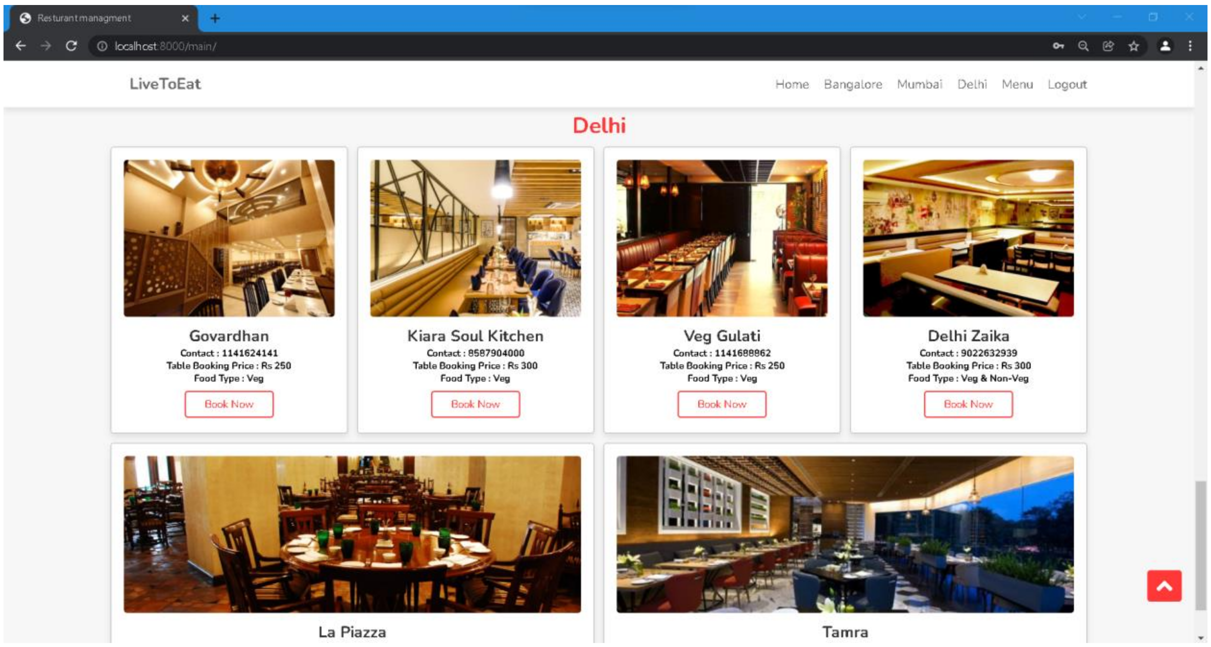
Naaz
Contact : 9022213768
Table Booking Price : Rs 300
Food Type : Veg & Non-Veg

Persian Darbar

Mantra

Figure 4.5 Main Page Mumbai Restaurants

Figure 4.5 contains the restaurants that is queried from the database using the address of the restaurants and this contains the restaurants that are in Mumbai.



Delhi

Govardhan
Contact : 1141624141
Table Booking Price : Rs 250
Food Type : Veg

Kiara Soul Kitchen
Contact : 8587904000
Table Booking Price : Rs 300
Food Type : Veg

Veg Gulati
Contact : 1141688862
Table Booking Price : Rs 250
Food Type : Veg

Delhi Zaika
Contact : 9022632939
Table Booking Price : Rs 300
Food Type : Veg & Non-Veg

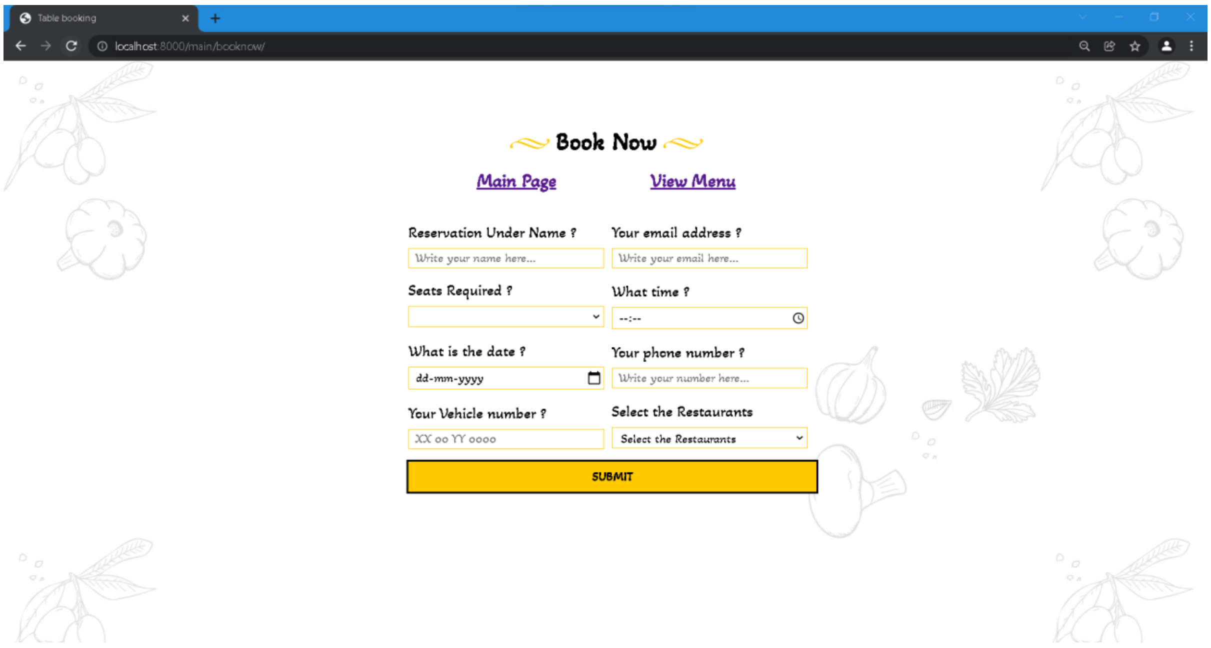
La Piazza

Tamra

Figure 4.6 Main Page Delhi Restaurants

Figure 4.6 contains the restaurants that is queried from the database using the address of the restaurants and this contains the restaurants that are in Delhi.

Booking Page

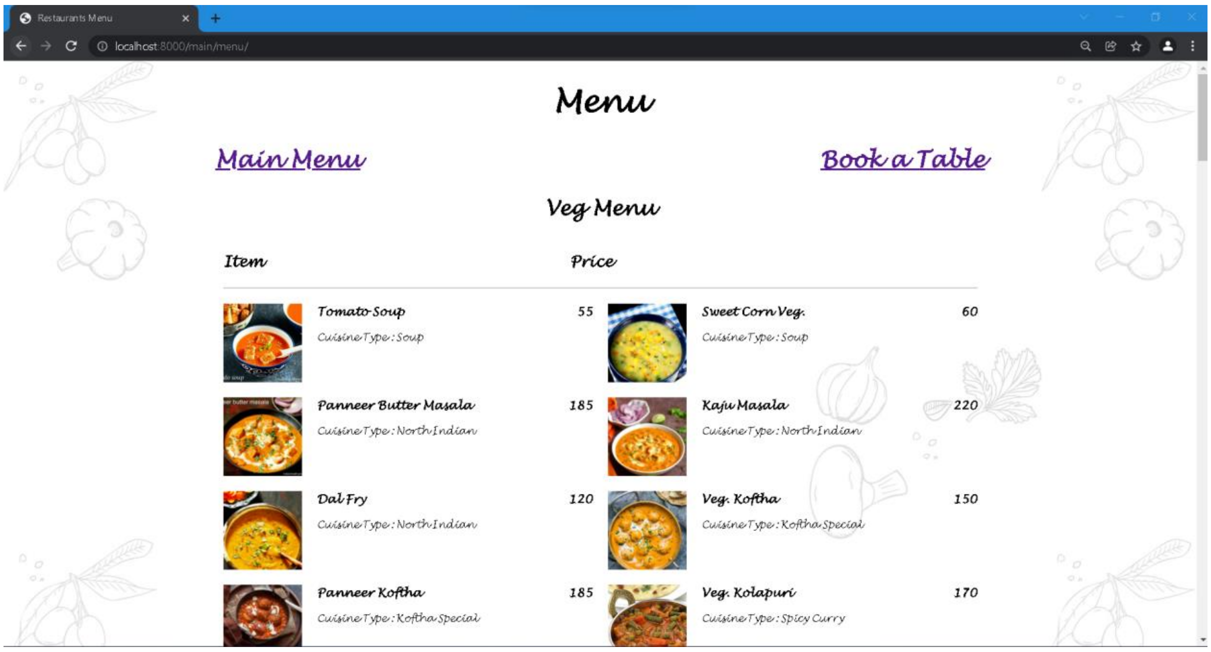


Reservation Under Name ?	Your email address ?
Write your name here...	Write your email here...
Seats Required ?	What time ?
	--:--
What is the date ?	Your phone number ?
dd-mm-yyyy	Write your number here...
Your Vehicle number ?	Select the Restaurants
XX oo YY oooo	Select the Restaurants

Figure 4.7 Booking Page

Figure 4.7 displays a form where user can select the restaurants and give all of his required details to make the booking in the restaurants.

Menu Page



Restaurants Menu

localhost:8000/main/menu/

Menu

Main Menu

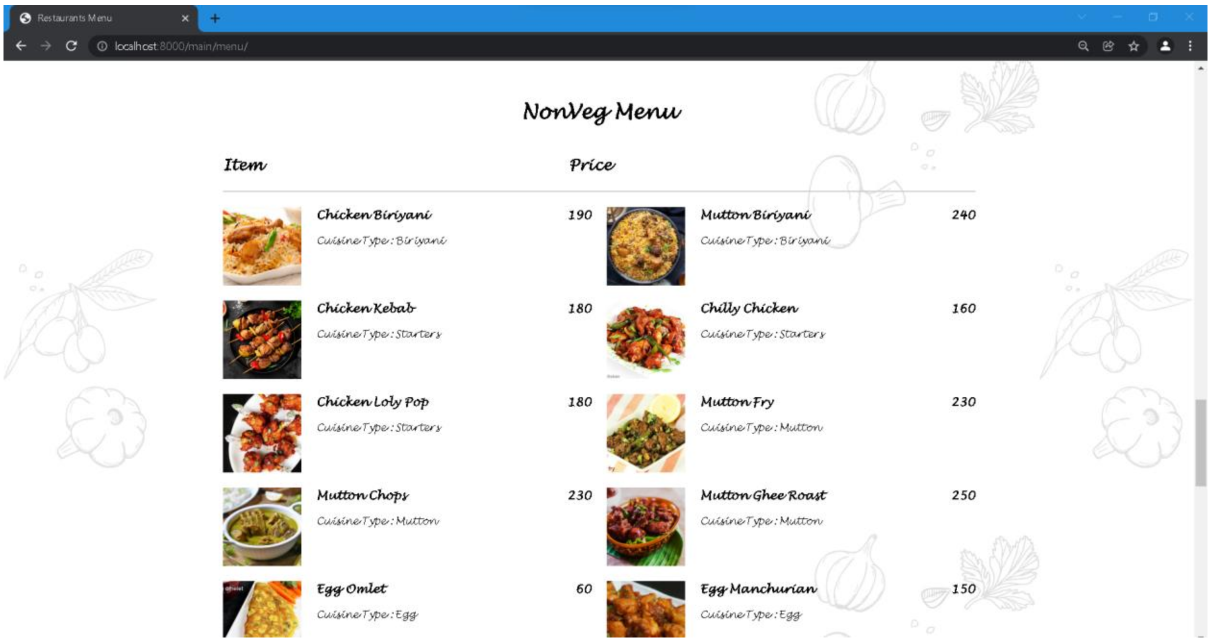
Book a Table

Veg Menu

Item	Price		
Tomato Soup CuisineType : Soup	55	Sweet Corn Veg. CuisineType : Soup	60
Panneer Butter Masala CuisineType : NorthIndian	185	Kaju Masala CuisineType : NorthIndian	220
Dal Fry CuisineType : NorthIndian	120	Veg. Koftha CuisineType : Koftha Special	150
Panneer Koftha CuisineType : Koftha Special	185	Veg. Kolapuri CuisineType : Spicy Curry	170

Figure 4.8 Menu Page Veg Content

Figure 4.8 has the Menu contents that are displayed in the webapp as an example of the items displayed when the user goes to this website. This is for the Veg Items



Restaurants Menu

localhost:8000/main/menu/

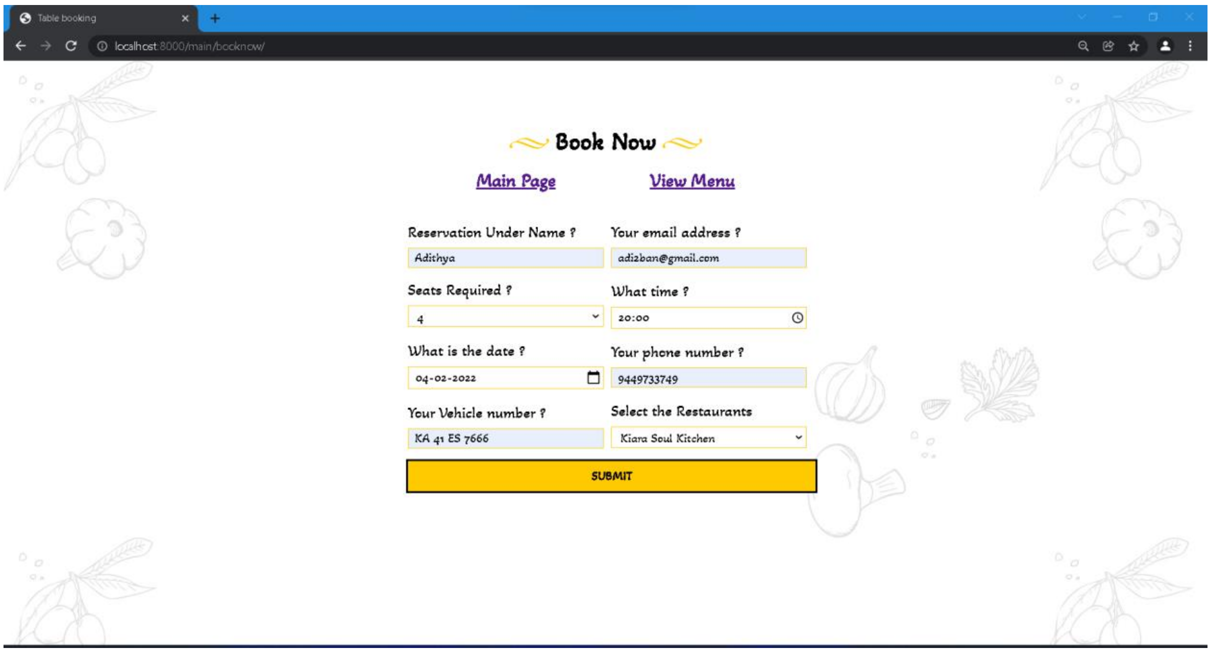
NonVeg Menu

Item	Price		
Chicken Biriyani CuisineType : Biriyani	190	Mutton Biriyani CuisineType : Biriyani	240
Chicken Kebab CuisineType : Starters	180	Chilly Chicken CuisineType : Starters	160
Chicken Loly Pop CuisineType : Starters	180	Mutton Fry CuisineType : Mutton	230
Mutton Chops CuisineType : Mutton	230	Mutton Ghee Roast CuisineType : Mutton	250
Egg Omlet CuisineType : Egg	60	Egg Manchurian CuisineType : Egg	150

Figure 4.9 Menu Page Non-Veg Content

Figure 4.9 has the Menu contents that are displayed in the webapp as an example of the items displayed when the user goes to this website. This is for the Non-Veg Items

Booking Page Working



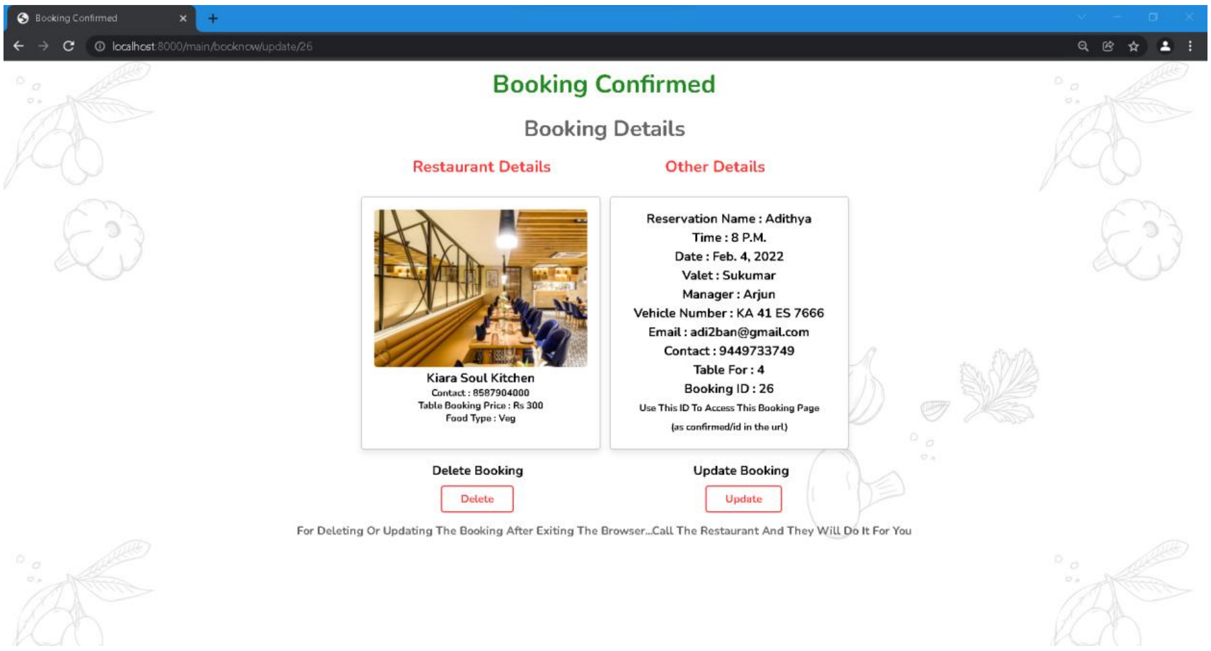
Book Now

Main Page   View Menu

Reservation Under Name ?	Your email address ?
Adithya	adi2ban@gmail.com
Seats Required ?	What time ?
4	20:00
What is the date ?	Your phone number ?
04-02-2022	9449733749
Your Vehicle number ?	Select the Restaurants
KA 41 ES 7666	Kiara Soul Kitchen

SUBMIT

Figure 4.10 Data Filled in Booking Page
Figure 4.10 shows the working of the Booking page as the customer enters the data into the website before submission.



Booking Confirmed

Booking Details

Restaurant Details   Other Details

Kiara Soul Kitchen
Contact : 8587904000
Table Booking Price : Rs 300
Food Type : Veg

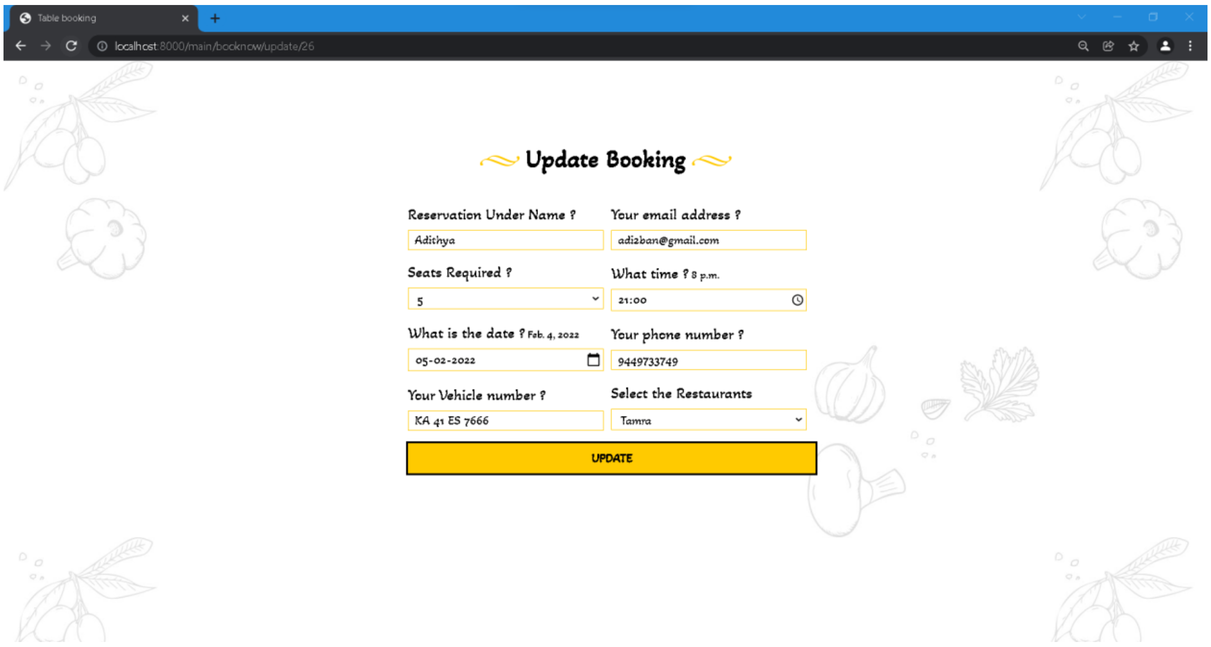
Reservation Name : Adithya
Time : 8 P.M.
Date : Feb. 4, 2022
Valet : Sukumar
Manager : Arjun
Vehicle Number : KA 41 ES 7666
Email : adi2ban@gmail.com
Contact : 9449733749
Table For : 4
Booking ID : 26
Use This ID To Access This Booking Page
(as confirmed/id in the url)

Delete Booking   Update Booking

Delete   Update

For Deleting Or Updating The Booking After Exiting The Browser...Call The Restaurant And They Will Do It For You

Figure 4.11 Booking Confirmation Page
Figure 4.11 shows the confirmation page that is displayed with all the booking details that are given by the user and this page is displayed as soon as the user submits the booking.

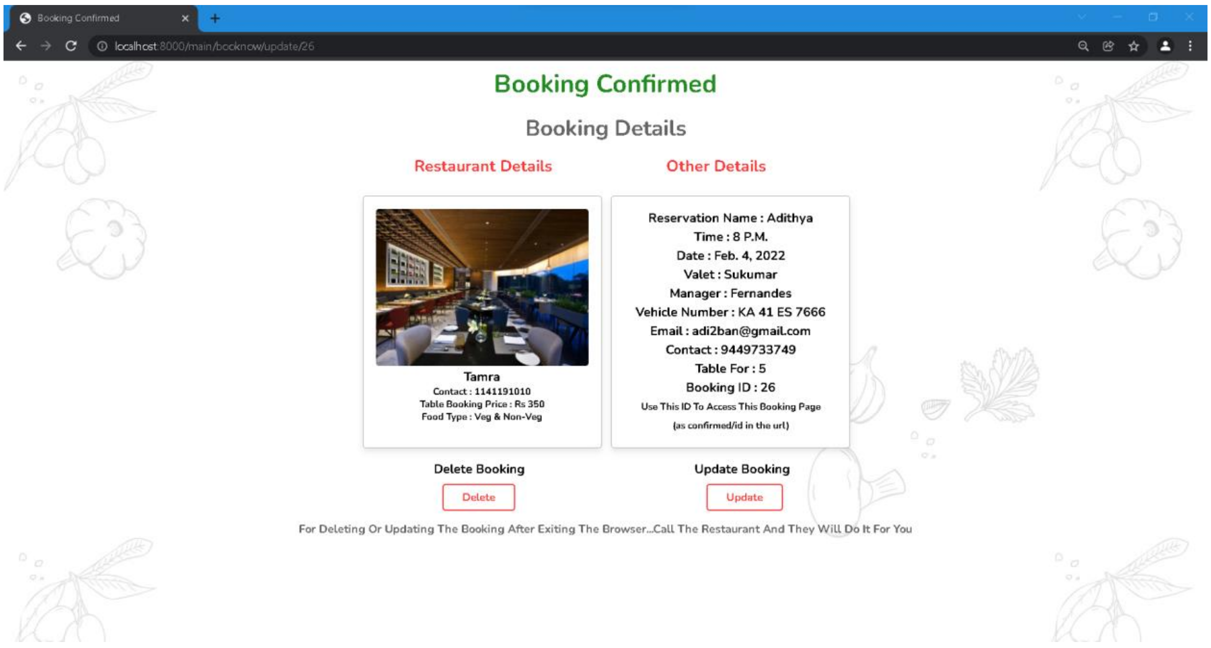


Update Booking

Reservation Under Name ?	Your email address ?
Adithya	adi2ban@gmail.com
Seats Required ?	What time ? 8 p.m.
5	21:00
What is the date ? Feb. 4, 2022	Your phone number ?
05-02-2022	9449733749
Your Vehicle number ?	Select the Restaurants
KA 41 ES 7666	Tamra

Figure 4.12 Booking Updating Page

Figure 4.12 shows the update page that can be used to make the changes to the booking that the user has already created.



Booking Confirmed

Booking Details

Restaurant Details

Tamra
Contact : 1141191010
Table Booking Price : Rs 350
Food Type : Veg & Non-Veg

Other Details

Reservation Name : Adithya
Time : 8 P.M.
Date : Feb. 4, 2022
Valet : Sukumar
Manager : Fernandes
Vehicle Number : KA 41 ES 7666
Email : adi2ban@gmail.com
Contact : 9449733749
Table For : 5
Booking ID : 26
Use This ID To Access This Booking Page
(as confirmed/id in the url)

Delete Booking

Update Booking

For Deleting Or Updating The Booking After Exiting The Browser...Call The Restaurant And They Will Do It For You

Figure 4.13 Booking Updating Confirmation Page

Figure 4.13 shows updated view of the confirmation page after the user updates the detail in the Update Booking page.

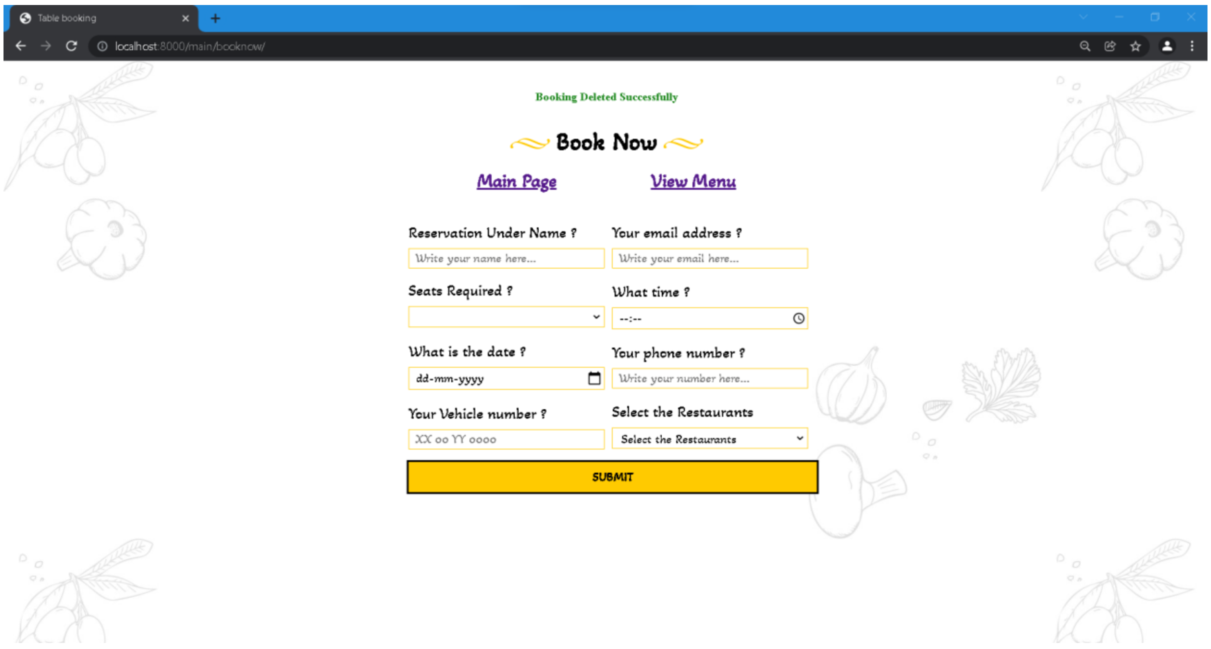


Figure 4.14 Page after Deletion of Booking

Figure 4.14 shows the page that is displayed to the user after deletion is completed.

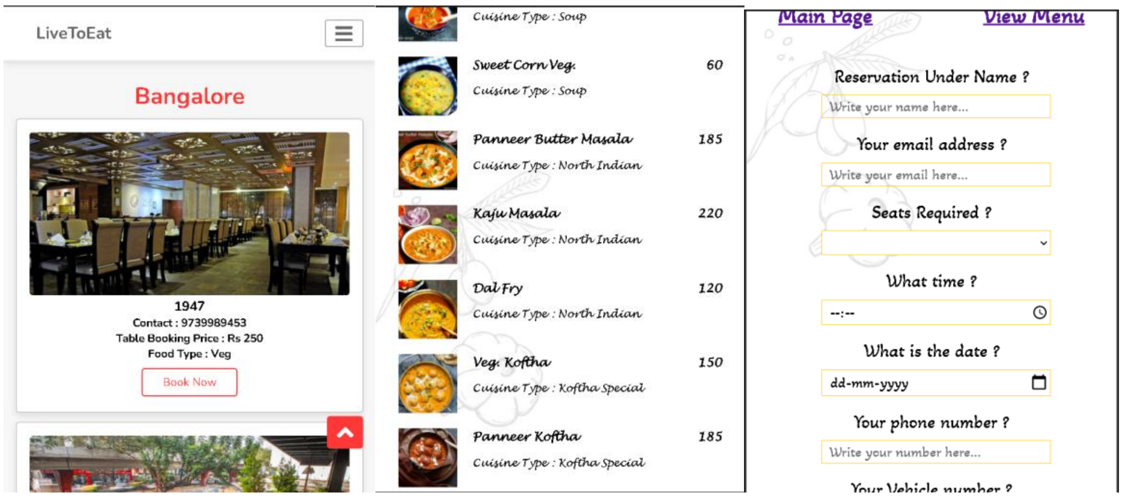


Figure 4.15 Responsive Webpage Screen Shots

Figure 4.15 shows the website in mobile resolutions and as you can see the webpages adapt themselves to fit the resolutions of the screen.

Restaurant Admin Page Screen Shots

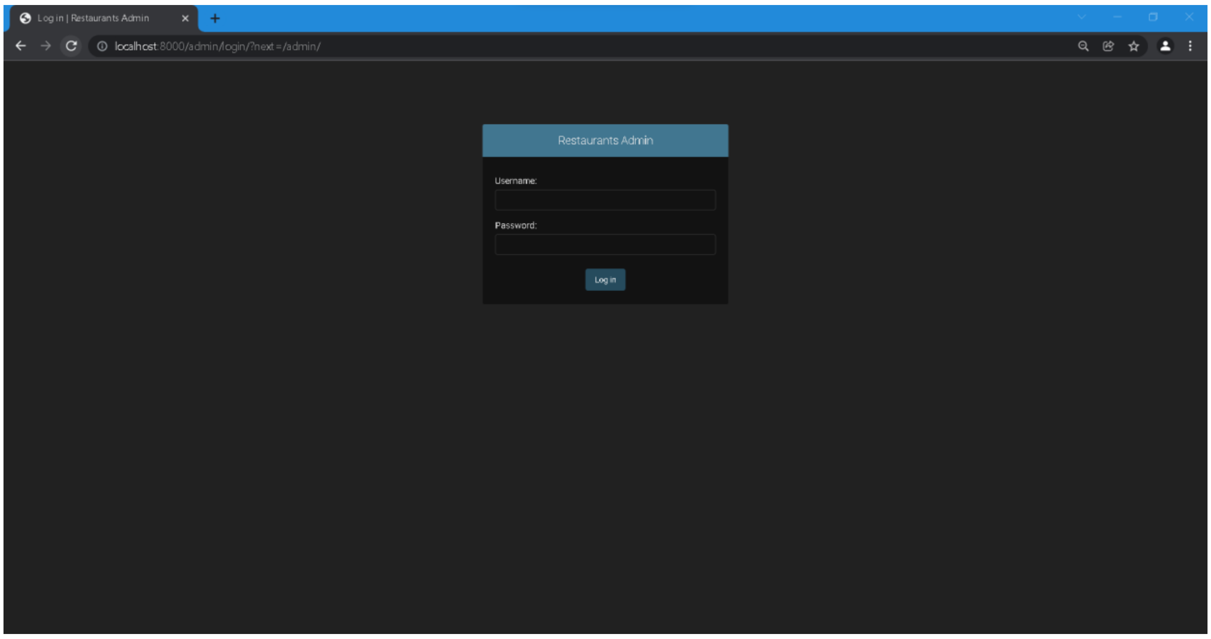
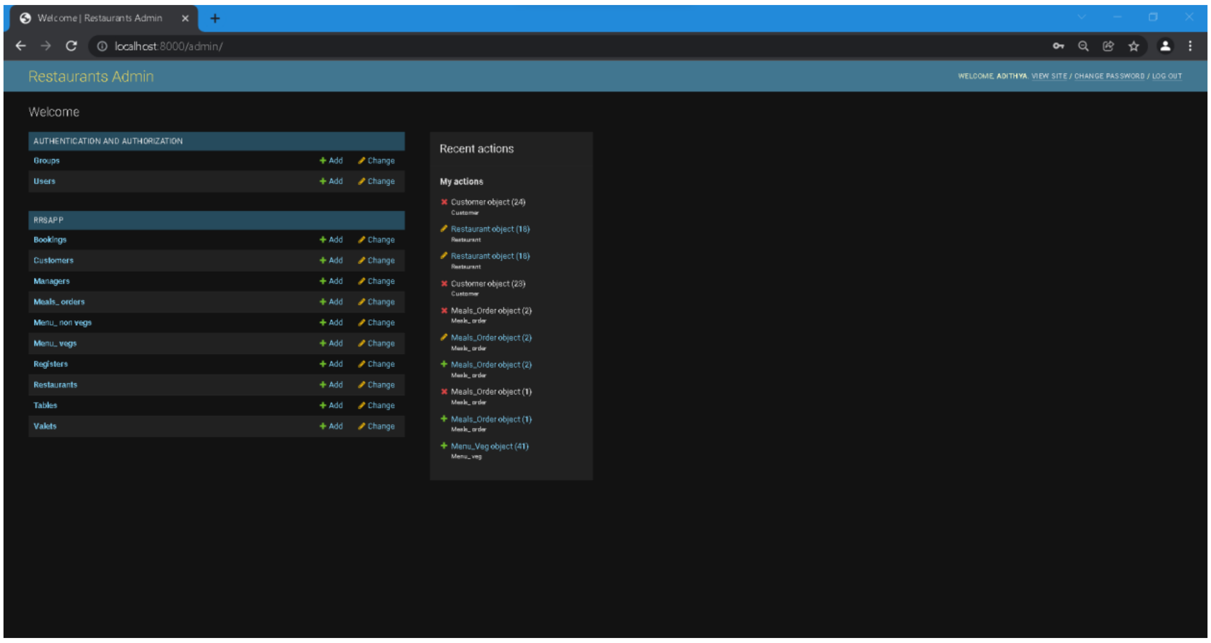

Log in | Restaurants Admin
localhost:8000/admin/login/?next=/admin/
Restaurants Admin
Username:
Password:
Log in

Figure 4.16 Restaurant Admin Login page

Figure 4.16 This is the admin page login which provides some features for the restaurant , managers.


Welcome | Restaurants Admin
localhost:8000/admin/
Restaurants Admin
WELCOME, ADITHYA. VIEW SITE / CHANGE PASSWORD / LOG OUT

Welcome

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

RRSAPP		
Bookings	+ Add	Change
Customers	+ Add	Change
Managers	+ Add	Change
Meals_ orders	+ Add	Change
Menu_ non vegs	+ Add	Change
Menu_ vegs	+ Add	Change
Registers	+ Add	Change
Restaurants	+ Add	Change
Tables	+ Add	Change
Valets	+ Add	Change

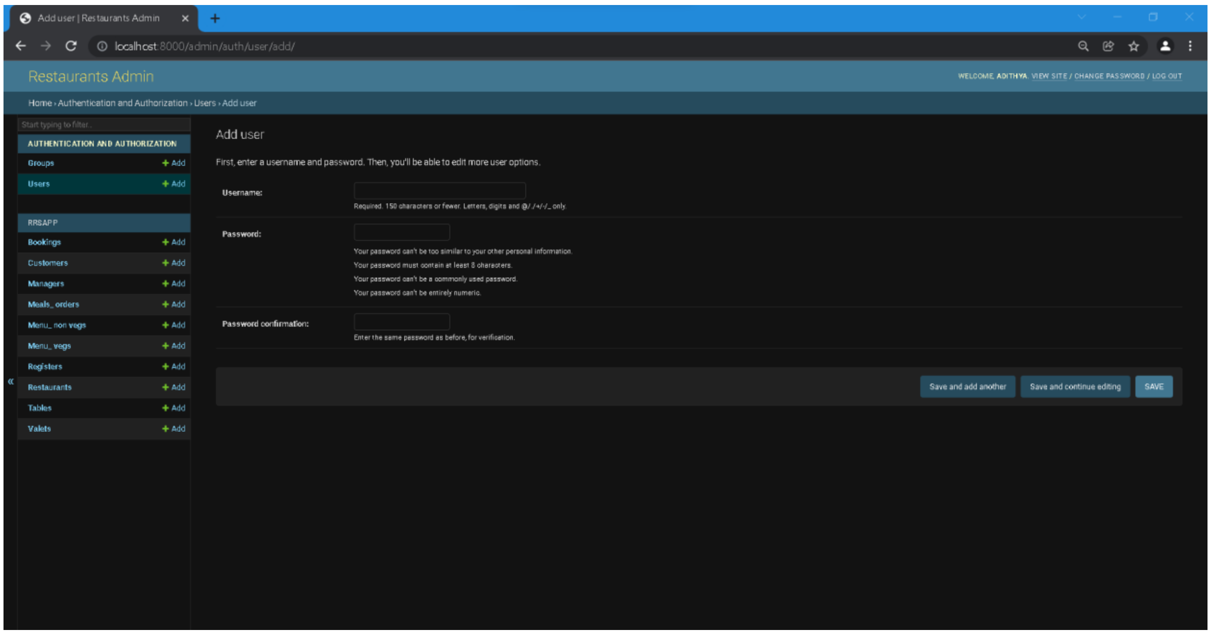
Recent actions

My actions

- ✗ Customer object (24) Customer
- Restaurant object (18) Restaurant
- Restaurant object (18) Restaurant
- ✗ Customer object (23) Customer
- ✗ Meals_Order object (2) Meals_order
- Meals_Order object (2) Meals_order
- + Meals_Order object (2) Meals_order
- ✗ Meals_Order object (1) Meals_order
- + Meals_Order object (1) Meals_order
- + Menu_Veg object (41) Menu_veg

Figure 4.17 Admin page after login

Figure 4.17 shows the admin page and all the models or tables that we have created in the database for the manager to view.


Add user | Restaurants Admin
localhost:8000/admin/auth/user/add/
Restaurants Admin
WELCOME, ADITHYA. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Authentication and Authorization › Users › Add user
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
RRSAPP
Bookings + Add
Customers + Add
Managers + Add
Meals_ orders + Add
Menu_ non vegs + Add
Menu_ vegs + Add
Registers + Add
Restaurants + Add
Tables + Add
Valets + Add
Save and add another
Save and continue editing
SAVE

Add user

First, enter a username and password. Then, you'll be able to edit more user options.

Username:

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password:

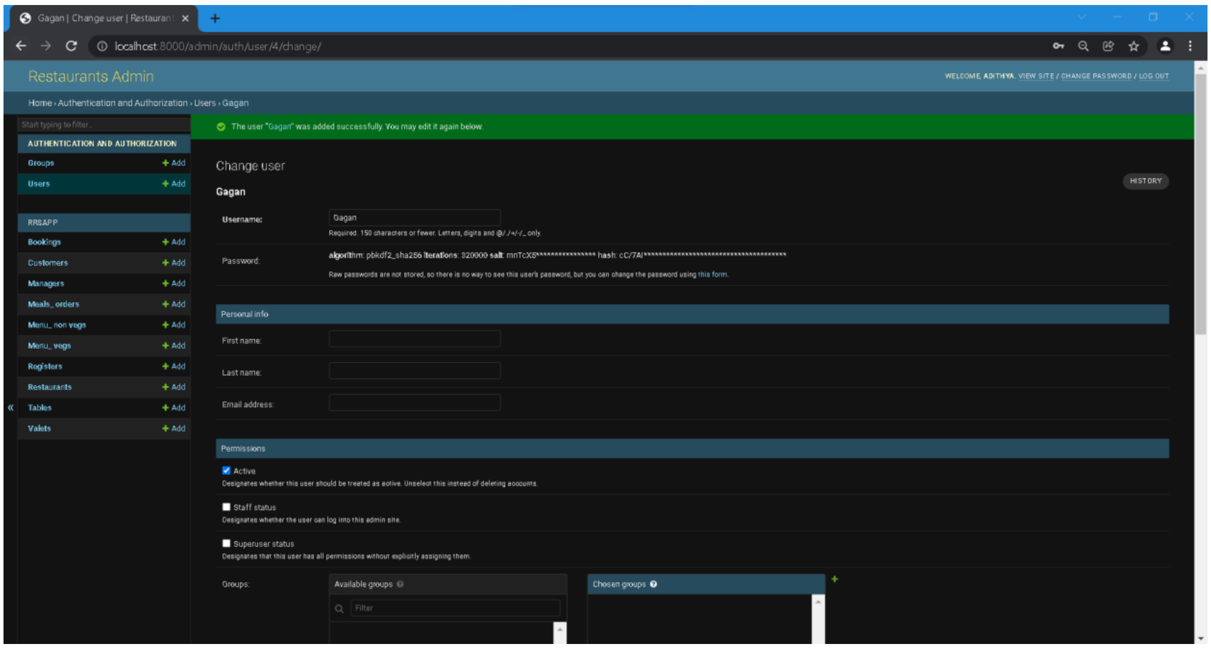
Your password can't be too similar to your other personal information.
Your password must contain at least 8 characters.
Your password can't be a commonly used password.
Your password can't be entirely numeric.

Password confirmation:

Enter the same password as before, for verification.

Figure 4.18 Page to add new staff for restaurant

Figure 4.18 shows the page to add the new staff like manager or restaurant owner etc.


Gagan | Change user | Restaurant
localhost:8000/admin/auth/user/4/change/
Restaurants Admin
WELCOME, ADITHYA. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Authentication and Authorization › Users › Gagan
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
RRSAPP
Bookings + Add
Customers + Add
Managers + Add
Meals_ orders + Add
Menu_ non vegs + Add
Menu_ vegs + Add
Registers + Add
Restaurants + Add
Tables + Add
Valets + Add
HISTORY

The user "Gagan" was added successfully. You may edit it again below.

Change user

Gagan

Username: Gagan

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password: algorithm: pbkdf2_sha256 iterations: 320000 salt: mnTcX5**************** hash: cC/7AI**************************************

Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.

Personal info

First name:

Last name:

Email address:

Permissions

[✓ Active]
Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

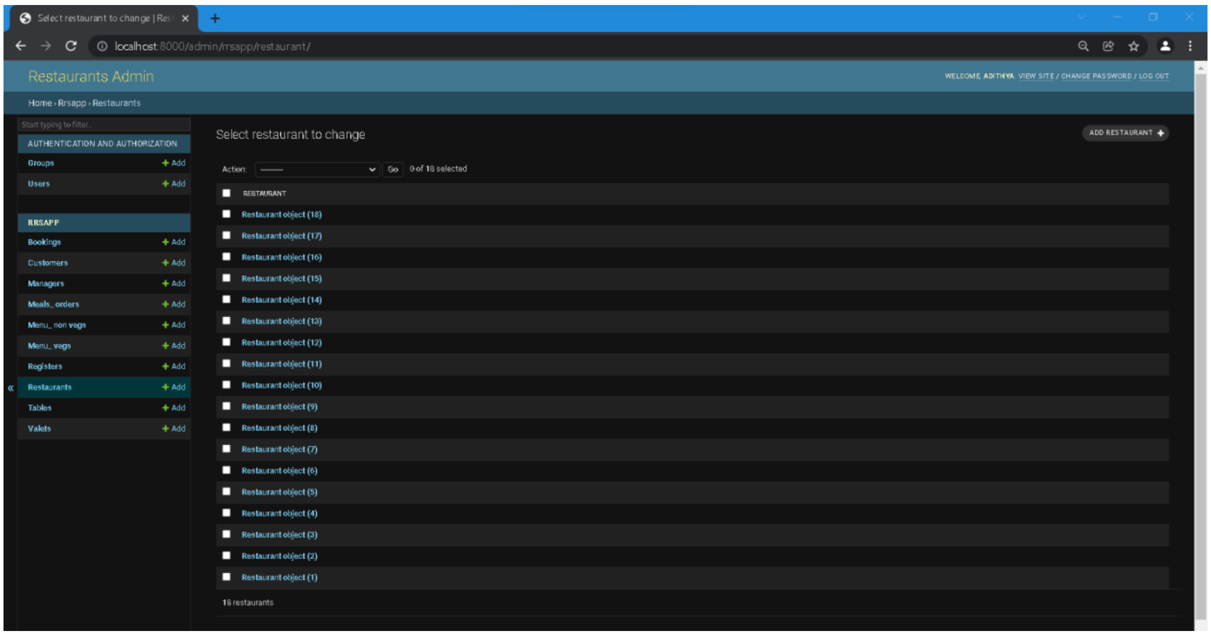
Staff status
Designates whether the user can log into this admin site.

Superuser status
Designates that this user has all permissions without explicitly assigning them.

Groups: Available groups | Filter | Chosen groups

Figure 4.19 User added successfully and status

Figure 4.19 shows the editing option available for the superuser to edit the information of the staff members or the restaurant owners.



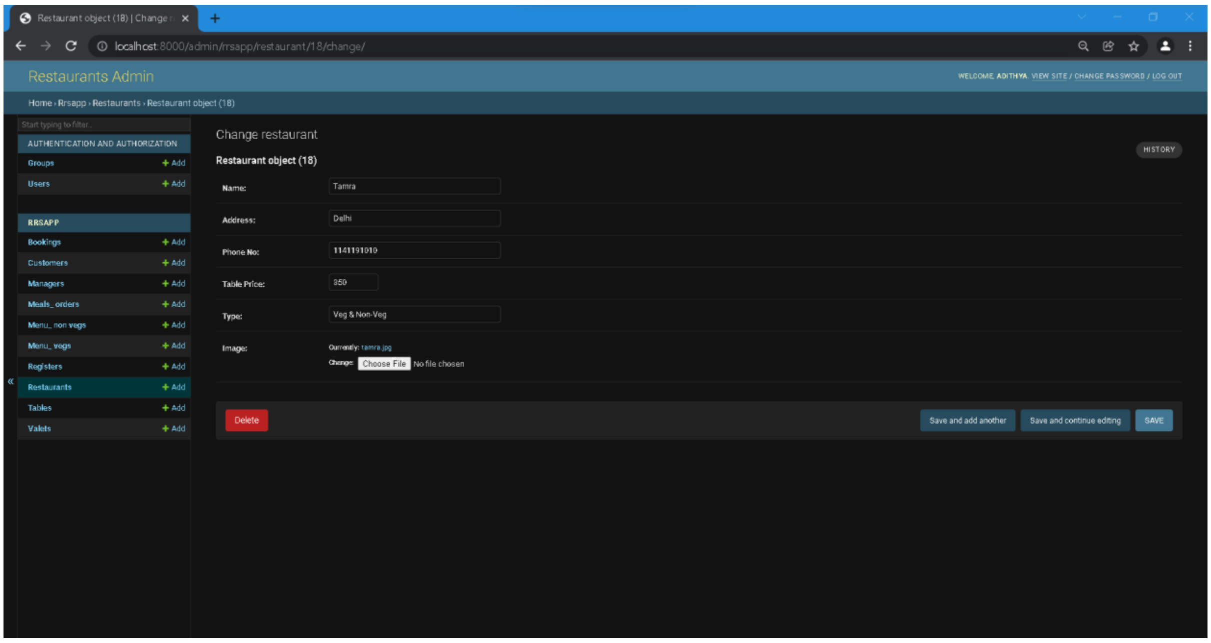
Select restaurant to change

RESTAURANT
Restaurant object (18)
Restaurant object (17)
Restaurant object (16)
Restaurant object (15)
Restaurant object (14)
Restaurant object (13)
Restaurant object (12)
Restaurant object (11)
Restaurant object (10)
Restaurant object (9)
Restaurant object (8)
Restaurant object (7)
Restaurant object (6)
Restaurant object (5)
Restaurant object (4)
Restaurant object (3)
Restaurant object (2)
Restaurant object (1)

18 restaurants

Figure 4.20 Page Showing Data from Database

Figure 4.20 This page shows an example of the view provided by Django admin GUI to show the available restaurants.


Restaurant object (18) | Change r
localhost:8000/admin/rrsapp/restaurant/18/change/
Restaurants Admin
WELCOME, ADITHYA. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Rrsapp › Restaurants › Restaurant object (18)
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
RRSAPP
Bookings + Add
Customers + Add
Managers + Add
Meals_orders + Add
Menu_non vegs + Add
Menu_vegs + Add
Registers + Add
Restaurants + Add
Tables + Add
Valets + Add
HISTORY
Delete
Save and add another
Save and continue editing
SAVE

Change restaurant

Restaurant object (18)

Name:	Tamra
Address:	Delhi
Phone No:	1141191010
Table Price:	350
Type:	Veg & Non-Veg
Image:	Currently: tamra.jpg Change: Choose File No file chosen

Figure 4.21 Data from a single row after insertion

Figure 4.21 shows the data of the insertion of one row in the restaurant table or model which can be deleted or updated.

Here all CRUD operations can be done that is Creation , Retrieval , Updating ,and Deletion.

Conclusion and Future Enhancements

Conclusion

RRS or the Restaurant reservation system developed here is a sophisticated system developed using various tools to help in betterment and ease for people to book and enjoy meals at the restaurants of their choice.
The system tries its level best to meet the requirements of the customer.
It also becomes an important part of every restaurant as it improves effectiveness, efficiency, management of resources, etc. It also establishes good and prominent relations with the stake holders like suppliers, customers, etc. It integrates and combines the entire business activities and take care of each and every step that ultimately helps in satisfaction of the customers and goals of the company.

Future Enhancements

The future scope of our project is vast.
We can make the app large scale so that all restaurants from various part of the country can be managed by this single system and improve the given facilities. The system can be enhanced in other different ways and these enhancements can also be implemented in the future so as to meet the growing or changing demand of the people. We can also do some changes like give the user some booking options to book a meal so that it is ready when he/she arrives at the restaurant.
These are some of the many future enhancements that can be made to our current system

References

[1].Fundamentals of Database Systems, Ramez Elmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.

[2].Database management systems , Ramkrishnan and Gehrke , 3rd Edition , 2014 , McGraw Hill

Website References

- https://www.youtube.com/watch?v=UmljXZIypDc&list=PL-osiE80TeTtoQCKZ03TU5fNfx2UY6U4p
- https://www.google.co.in/?gws_rd=ssl
- https://docs.djangoproject.com/en/4.0/
- https://www.w3schools.com/ for HTML , CSS and JavaScript

Visit the GitHub link below for the source code of the project.
https://github.com/Adithya2441/dbms-mini

.