BANGALORE UNIVERSITY

UNIVERSITY VISVESVARAYA COLLEGE OF ENGINEERING

K R Circle, Bengaluru - 560001

FOR QUALITY AND EXCELLENCE IN HIGHER EDUCATION

★ ★ ★ ★ ★

ACCREDITED BY NAAC

Department of Computer Science and Engineering

A DBMS Mini Project Report on

"DOOR DELIGHT"

A Food Ordering System

Submitted By

Nithin Bharghav R(18GAEI6032)

Pavan Kumar N (18GAEI6033)

V SEM, B.Tech (ISE)

Under the Guidance of

Smt. Samyama Gunjal G H

Assistant Professor

Dept of CSE

UVCE

YEAR 2020-21

Bangalore University

University Visvesvaraya College of Engineering

K R Circle, Bengaluru - 560001

FOR QUALITY AND EXCELLENCE IN HIGHER EDUCATION

★ ★ ★ ★ ★

ACCREDITED BY NAAC

Department of Computer Science and Engineering

Certificate

This is to certify that Mr. Nithin Bharghav R of V Semester, B. Tech (Information Science and Engineering) bearing the register number 18GAEI6032 has submitted the Database Management System Mini Project Report on **" DOOR DELIGHT "-** A Food Ordering System in partial fulfilment for the Database Management System Lab prescribed by the Bangalore University for the academic year 2020-21.

Guide:	**Chairperson:**
Smt. Samyama Gunjal G H	**Dr. Dilip Kumar S M**
Assistant Professor	**Professor &Chairperson**
Department of CSE	**Department of CSE**
UVCE	**UVCE**

Examiner 1: ___________________________

Examiner 2: ___________________________

Bangalore University

University Visvesvaraya College of Engineering

K R Circle, Bengaluru - 560001

Department of Computer Science and Engineering

Certificate

This is to certify that Mr. Pavan Kumar N of V Semester, B. Tech (Information Science and Engineering) bearing the register number 18GAEI6033 has submitted the Database Management System Mini Project Report on **" DOOR DELIGHT "-** A Food Ordering System in partial fulfilment for the Database Management System Lab prescribed by the Bangalore University for the academic year 2020-21.

Guide:	**Chairperson:**
Smt. Samyama Gunjal G H **Assistant Professor** **Department of CSE** **UVCE**	**Dr. Dilip Kumar S M** **Professor &Chairperson** **Department of CSE** **UVCE**

Examiner 1: ________________________

Examiner 2: ________________________

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task is incomplete without the mention of people who made it possible, whose constant guidance and encouragement crown all efforts and success.

We are grateful to acknowledge **Dr. Venugopal K R**, Vice Chancellor, Bangalore University, for his valuable suggestions and relentless support, which has sustained me throughout the course of the project.

We express our gratefulness to **Dr. H N Ramesh**, Principal, UVCE who has been leading our college with a brighter vision in technical education.

We express our profound gratitude to **Dr. Dilip Kumar S M**, Chairperson, Department of Computer Science and Engineering, UVCE who has been a constant source of inspiration to the students work.

We express our gratitude to our guide **Smt. Samyama Gunjal G H**, Assistant Professor, Department of Computer Science and Engineering, UVCE for her valuable guidance and supervision in this course of the project.

We also thank our guide **Miss. Veena Hosamani**, Lab Incharge, Department of Computer Science and engineering, UVCE.

We express our sincere thanks to all teaching and non-teaching staff, Department of Computer Science and Engineering, UVCE for all the facilities that they provided us to successfully complete this project.

We also thank our parents and friends for their continuous support and encouragement.

THANK YOU

NITHIN BHARGHAV R
18GAEI6032
PAVAN KUMAR N
18GAEI6033

ABSTRACT

Increased demand of restaurant-goers generated the need for much attention for the hospitality industry. Providing much option with ease of ordering and delivering is the need of the hour. Technological interference has become mandatory to improve the quality of the service and business in this industry. Our proposed system is an online food ordering system that enables ease for the customers. It overcomes the disadvantages of the traditional queueing. Our proposed system is a medium to order online food hassle free from restaurants. The online food ordering system sets up a food menu online and customers can easily place the order as per their wish. This is a website designed primarily for use in the food delivery industry. This system will allow hotels and restaurants to increase scope of business by reducing the labour cost involved. The system also allows to quickly and easily manage an online menu which customers can browse and use to place orders with just few clicks.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The project sets to develop an online ordering system for restaurant. Many industries now quickly adopting technologies. Restaurant industry also embraces different types of technologies which make daily processes easier and faster. But the adoption of technology in restaurant industry is slower compared to other industries.

Restaurants can use technology in different ways. One of them is to build an online presence by a web application which may also help in online ordering. Because the use of internet-based ordering system is in rise in today's world. When it comes to buying goods or foods online, customers want to be ensured about the quality of the foods that she/he is going to buy. There are also some problems of traditional food ordering system, which results wasting time and making conflicts. In existing system there are few problems:

- For placing orders, customers have to visit restaurants to know about food items and then place order and pay.

In this method time and manual work is required. Some restaurants take order via phone. While placing an order over the phone, customer lacks the copy of available menu items in restaurant.

To solve these issues, an Online Food Order System has been developed which is originally designed for small scale business. But this system is applicable in any restaurant. The main advantage of this online system is that it greatly simplifies the food ordering process for both of the customer and the restaurant.

The anticipated benefits of the project are:

1. This will speed up the ordering process.
2. The system will help to reduce labour cost involved.
3. This will avoid long queues at the counter due to the speed of execution and number of optimum screens to accommodate the maximum throughput.
4. The system will be less probable to make mistake, since it's a machine.
5. The top benefit of online ordering was a savings in labour, since employees are not tied up on the phone or at the counter.
6. Order accuracy was another benefit for restaurant.

CHAPTER 2
LITERATURE SURVEY

PAST WORKS ON THIS DOMAIN

A computerised restaurant system is “an integrated IT system that supervises, manages and facilitates the planning operations in restaurant” (Tan, 2013). Before the automated system has been introduced, the orders and payments were managed manually using register books etc. The Point-of-Sales (POS) was introduced in the era of 1990. After the system got popularity, it started being used in restaurant industry. After 1990, the internet and wireless technology moved on so much fast. Restaurants also started using different technology-based system to increase the efficiency of the system (Sullivan, 2015). Researchers also started introducing different types of solutions for restaurants. Lots of work have been done for automating the operations of the restaurants. Every researcher chose an aspect or problem and tried to make good solution for that. The solutions provided by researchers can be categorised in different categories.

PDA-BASED SYSTEM

Xiang, Zhou and Chowdhury (2004) developed a PDA based in-house meal ordering system for restaurant named iMenu. It is one of the first systems that introduced PDA based ordering in restaurant industry. The developers used .NET platform for developing the system. Patel, Patel and Obersnel (2007) discussed about implementation of PDA-based ordering. The proposed system by Patel, Patel and Obersnel (2007) was a web-based stand-alone wireless application. Hongzhen, Bin and Wenlin (2009) developed a web service and wireless system which can be used through desktop and personal digital assistant (PDA). In this system, the access of data to the servers can be wired and wireless. The food ordering functions served through both desktop computers and PDAs over a wired/wireless integrated local area network.

RFID BASED SYSTEM

Many researchers also proposed and implemented wireless communication based system within the restaurant. Ngai, Suk and Lo (2008) developed a radio frequency identification (RFID) based sushi management system in a conveyor-belt sushi restaurant. Their case study showed that RFID technology helps improve food safety, inventory control, service quality, operational efficiency, and data visibility in sushi restaurants. Unfortunately, this system does not support customer-centred service because it cannot actively identify customers. To overcome this problem, Tan, Chang and Chen (2012) proposed a RFID based smart food recommender system, where customers will have a RFID based card. The system integrates radio frequency identification (RFID), wireless local area network, database technologies, and a menu recommender. This system enables waiters to immediately identify customers via RFID-based membership cards and then actively recommend the most appropriate menus through menu recommender for customers. The customer will be identified by this RFID based card and recommended food menu based on his/her past choice of food. For taking orders, the waiters use PDAs that are connected the system via local area network (LAN). Gayatri, Chaitanya and Harikrishna (2014) also proposed a RFID based model almost the same system like Tan et al. (2012). But they added an extra feature to the system, where customers can order food from their own mobile through short text message (SMS).

MOBILE APPLICATION BASED SYSTEM

Samsudin et al. (2011) recognised the limitations of PDA based food ordering system and developed a food ordering system with Microsoft Access database and ASP.NET and VB.NET for Microsoft Windows based smartphones. Customers can order foods remotely from this application. As Android based smartphones took over Windows based smartphones, Shinde et al. (2014) developed an application for Android operating system based smartphones where they used Java and SQLite. The order processing speed of this Android application is faster than PDA based ordering system. The application can play a media between customer, restaurant manager and kitchen. Dhore et al. (2014) also proposed an Android based system to automate ordering process of the restaurant, through which one or more customers can order food, book table, and also make payment remotely. Both of these to works implemented almost same functionalities. Leong (2016) also developed a system for Android operating system, which can be used for ordering food online. In this system, computers need to be installed in kitchen and manager's desk, so they can view and check orders and payment.

WEB APPLICATION BASED SYSTEM

To overcome the device and operating system centric approach for automating restaurant operation, K et al. (2016) proposed an online web application which can be accessed through internet. For using this application, computer screen will be placed on each table of restaurant for customers to order. Customers can order food for take away and make payment through the application. However, the idea is innovative, but a little bit weird as computers need to be installed in every table. Sometimes it may create complex problem rather than making a solution. Patel (2015) developed an online food ordering system with Java in backend system and HTML with CSS in frontend. It was a robust and nice application. But the responsive development method was not used in the application. Hence, the application cannot view properly with a smart phone. Tan (2013) developed a web-based application with ASP .Net and Bootstrap to automate the ordering process of the restaurant and make it paperless. But the system is only focused on the management of the restaurant.

EVALUATION OF THE PAST WORKS

The previous lines illustrate that many researchers worked for implementing IT based systems in restaurants to gain more revenue and reduce order processing time. But most of them did not focus on the online based restaurant system where customers can order food over internet from their home. With the expansion of internet technology in this era, it is not feasible to stick with only technologies demonstrated above as most of the proposed system only tried to automate the in-house ordering system.

On the other hand, the systems which is developed to order food remotely are device or operating systems oriented. For obtaining the benefits of the system, customers need to use a smartphone with specific operating system like Windows or Android. No one tried to find a solution which does not depend on any specific device, while a web application can be accessed by through kind of devices with internet connection from anywhere. Moreover, there is no need for installing extra software for browsing a web application. If the web application is developed using modern mobile friendly manner properly, it will give a good performance on any kind and size of device.

The proposed solutions also lack the use of the power of social community, which can be a big marketing medium for the restaurant and help influencing new customers to have a food experience in the restaurant. The earlier part of this paper stated that the digital restaurant ordering is growing 300 percent faster than dine-in traffic (Beltis, 2016). So, the restaurants should embrace internet technology to make a solid appearance on the internet and provide the service in a wider range of customers within minimum cost and time.

CHAPTER 3
SOFTWARE ENVIRONMENT

3.1 Database Management System (DBMS)

DBMS is a collection of programs that enables users to create and maintain a database The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating and sharing databases among various users and applications.

A Relational database is a database that has a collection of tables of data items, all of which is formally described and organized according to the relational model. Data in a single table represents a relation, from which the name of the database type comes. In typical solutions, tables may have additionally defined relationships with each other. In the relational model, each table schema must identify a column or group of columns, called the primary key, to uniquely identify each row. A relationship can then be established between each row in the table and a row in another table by creating a foreign key, a column or group of columns in one table that points to the primary key of another table.

3.1.1 Characteristics of Database Management Systems

- Self-describing nature.
- Keeps a tight control on data redundancy.
- Enforces user defined rules to ensure that integrity of table data.
- Provides insulation between Programs and data, Data abstraction.
- Supports multiple views of the data.
- Helps sharing of data and Multi-user transaction processing.

3.1.2 Advantages of using the DBMS approach

- Controlling the redundancy.
- Restricting unauthorized access.
- Providing persistent storage for program objects.
- Providing storage structures for efficient query processing.
- Providing backup and recovery.
- Providing multiple users interfaces.
- Enforcing Integrity Constraints.
- Representing Complex Relationships among Data.

3.2 Structured Query Language (SQL)

SQL is a comprehensive database language. SQL is a standard language for storing, manipulating and retrieving data in databases. The ANSI standard SQL provides basic functions for data manipulation, transaction control, and record retrieval from the database. However, most end users interact with Oracle through application that provides an interface that hides the underlying SQL and its complexity. SQL uses the terms table, row, and column for relation, tuple, and attribute, respectively.

3.2.1 Applications of SQL

SQL is one of the most widely used query language over the databases that:

- Allows users to access data in the relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in a database and manipulate that data.
- Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures and views.

3.2.2 SQL Commands

The standard SQL commands to interact with relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP. These commands can be classified into the following groups based on their nature –

DDL - Data Definition Language

SL.No.	Command & Description
1	**CREATE** Creates a new table, a view of a table, or other object in the database.
2	**ALTER** Modifies an existing database object, such as a table.
3	**DROP** Deletes an entire table, a view of a table or other objects in the database.

DML - Data Manipulation Language

SL.No.	Command & Description
1	**SELECT** Retrieves certain records from one or more tables.
2	**INSERT** Creates a record.
3	**UPDATE** Modifies records.
4	**DELETE** Deletes records.

DCL - Data Control Language

SL.No.	Command & Description
1	**GRANT** Gives a privilege to user.
2	**REVOKE** Takes back privileges granted from user.

3.2.3 SQL Data Type

SQL Data Type is an attribute that specifies the type of data of any object. Each column, variable and expression have a related data type in SQL. You can use these data types while creating your tables. You can choose a data type for a table column based on your requirement.

Numeric data types:

1. INT(size) - A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295.
2. INTEGER(size) - Equal to INT(size).

3. SMALLINT(size) - A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535.
4. FLOAT*(*size, d*)* - A floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter.
5. DOUBLE*(*size, d*)* - A normal-size floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter.
6. DECIMAL*(*size, d*)* - An exact fixed-point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter.

Character-String data types:

1. CHAR(size) - A FIXED length string (can contain letters, numbers, and special characters). The size parameter specifies the column length in characters - can be from 0 to 255. Default is 1.
2. VARCHAR(size) - A VARIABLE length string (can contain letters, numbers, and special characters). The size parameter specifies the maximum column length in characters - can be from 0 to 65535.
3. CHAR VARYING(n) – similar to VARCHAR(size).

Bit-String data types:

1. BIT(size) - A bit-value type. The number of bits per value is specified in size. The size parameter can hold a value from 1 to 64. The default value for size is 1.
2. BIT VARYING(n) - similar to BIT(size) where n is the maximum number of bits. The default for n ,the length of a character string or bit string is 1.
3. BINARY(size) - Equal to CHAR(), but stores binary byte strings. The size parameter specifies the column length in bytes. Default is 1.

Boolean data types:

1. BOOL - Zero is considered as false, nonzero values are considered as true.
2. BOOLEAN - Equal to BOOL.

Date and Time data types:

1. DATE - A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'.

2. DATETIME(fsp) - A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.
3. TIME(fsp) - A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'.
4. TIMESTAMP(fsp) - TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss.

3.2.4 Aggregate Functions in SQL

Following aggregate functions are provided by the SQL.

1. COUNT - Returns number of tuples.
2. SUM - Returns sum of entries in a column.
3. MAX - Returns Maximum value from an entire column.
4. MIN - Returns Minimum value from an entire column.
5. AVG - Returns Average of all the entries in a column.

3.2.5 Constraints in SQL

Following constraints are provided by the SQL.

1. NOT NULL - Column should contain some value.
2. PRIMARY KEY - Should not allow duplicate and null values to a column.
3. UNIQUE - Each value of a column should be unique.
4. DEFAULT - Provides a default value for a column when none is specified.
5. FOREIGN Key - Uniquely identifies a row/record in any of the given database table.
6. CHECK - The CHECK constraint ensures that all the values in a column satisfies certain conditions.
7. INDEX - Used to create and retrieve data from the database very quickly.

3.2.6 Triggers in SQL

A trigger is a special type of stored procedure that automatically runs when an event occurs in the database server. DML triggers run when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view. These triggers fire when any valid event fires, whether table rows are affected or not.

CHAPTER 4
SYSTEM REQUIREMENTS

4.1 Hardware Requirements

- System : Intel i3 2.4 GHz.
- Hard Disk : 256 GB
- Monitor : 15 VGA Color.
- Ram : 8GB.

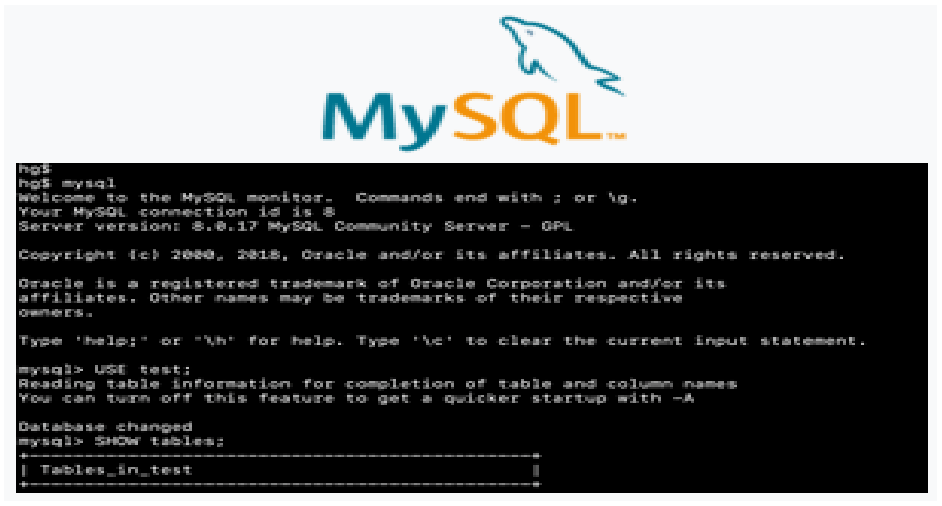
4.2 Software Requirements

- **OS:** No particular operating system, platform independent. Windows 10 used.
- **Backend :**
  - Django version 3+ required. To run the application, locate to folder path and hit python manage.py runserver. Django is used to create tables and link databases; Python is at the heart of Django. Models handles data, database. Views contains python code, Templates is made of HTML, CSS.
  - Database: MySQL.
- **Front End :** HTML, CSS, Bootstrap, JS.

4.3 Overview of Tools/Software

❖ MySQL

MYSQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data. SQL is a language programmer use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups.



```
hg$
hg$ mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.17 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW tables;
+---------------------------------------------------------+
| Tables_in_test                                          |
+---------------------------------------------------------+
```

Figure 4.1 : Screenshot of the default MySQL command-line banner and prompt

MySQL's Logical Architecture:

The topmost layer contains the services that aren't unique to MySQL. They're services most network-based client/server tools or servers need: connection handling, authentication, security, and so forth.

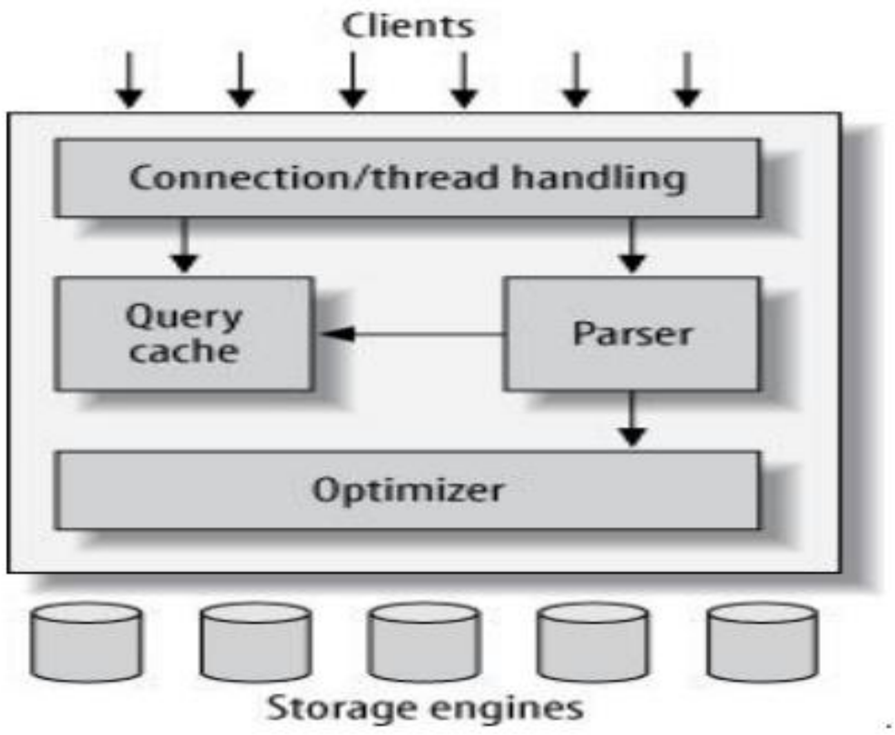


Fig. 4.2: MySQL's Logical Architecture

The third layer contains the storage engines. They are responsible for storing and retrieving all data stored "in" MySQL. Like the various file systems available for GNU/Linux, each storage engine has its own benefits and drawbacks. The server communicates with them through the storage engine API. This interface hides differences between storage engines and makes them largely transparent at the query layer. The API contains a couple of dozen low-level functions that perform operations such as begin a transaction" or "fetch the row that has this primary key. The storage engines don't parse SQL or communicate with each other; they simply respond to requests from the server.

❖ DJANGO

Django is a Python-based free and open-source web framework, which follows the model view-template (MVT) architectural pattern. It is maintained by the Django Software Foundation (DSF), an independent organization established as a non-profit. Django's primary goal is to ease the creation of complex, database-driven websites.

Django follows a Model-View-Template(MVT) architecture, which is split up into three different parts:

- The Model is the logical data structure behind the entire application and is represented by a database(generally relational databases such as MySQL, Postgres).
- The View is the user interface — what you see in your browser when you visit a website. These are represented by HTML/CSS/JavaScript files.
- The Template's main goal is to keep everything that browser renders. The model's data that's coming from the server in different parts while integrating the same when the user interacts with the website. Here, the template layer in Django is more similar to the views layer in MVC pattern.

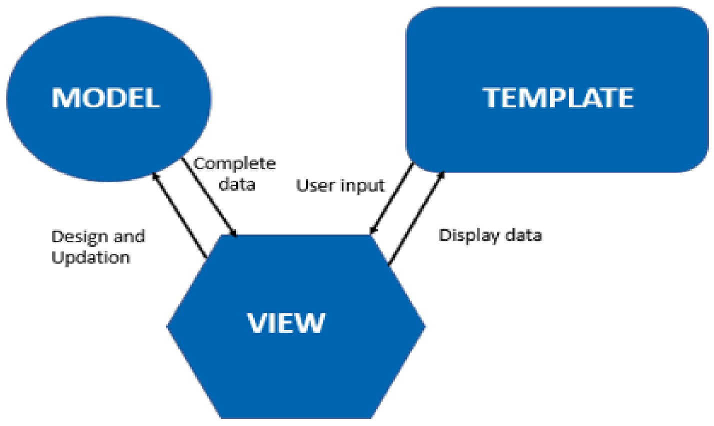


Figure 4.3 : MVT Django Architecture

Model: The Model is the part of the web-app which acts as a mediator between the website interface and the database. In technical terms, it is the object which implements the logic for the application's data domain. There are times when the application may only take data in a particular dataset, and directly send it to the view (UI component) without needing any database then the dataset is considered as a model. Although today if we want any kind of website we need to have some sort of database as we must be requiring some user input even if we are creating a simple blog site. The Model is the component which contains Business Logic in Django architecture.

For example: When you sign up on any website you are actually sending information to the controller which then transfers it to the models which in turn applies business logic on it and stores in the database.

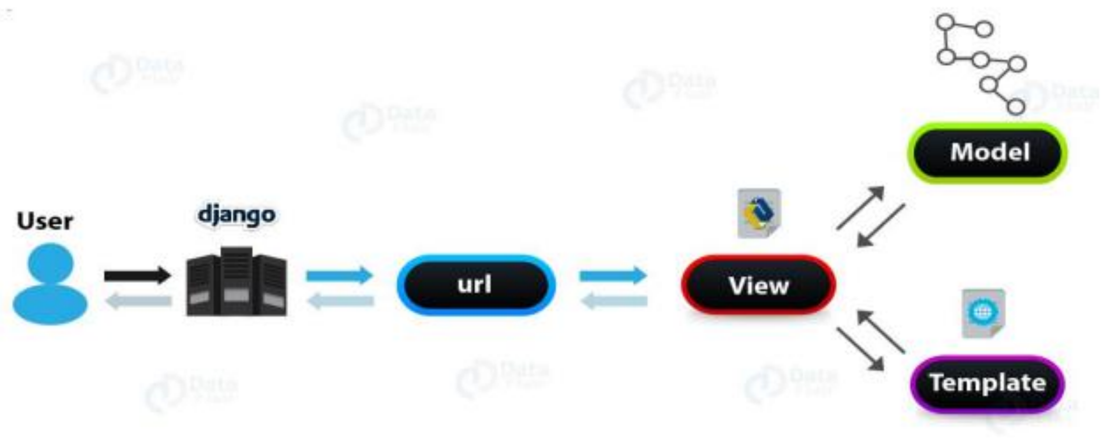


Figure 4.4 Django Architecture

View: This component contains the UI logic in the Django architecture. View is actually the User Interface of the web-application and contains the parts like HTML, CSS and other frontend technologies. Generally, this UI creates from the Models component, i.e., the content comes from the Models component.

For example: When you click on any link or interact with the website components, the new webpages that website generates is actually the specific views that stores and generates when we are interacting with the specific components.

Template: Django provides a convenient way to generate dynamic HTML pages by using its template system. A template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted. In HTML file, we can't write python code because the code is only interpreted by python interpreter not the browser. We know that HTML is a static mark-up language, while Python is a dynamic programming language. Django template engine is used to separate the design from the python code and allows us to build dynamic web pages.

Advantages of Django Architecture: The Django Framework is based on this architecture and it actually communicates between all these three components without needing to write complex code. That's why Django is gaining popularity. This architecture in Django has various advantages like:

1**. Rapid Development:** Actually, this Django architecture that separates in different components makes it easy for multiple developers to work on different aspects of the same application simultaneously. That is also one of the features of Django.

2. **Loosely Coupled:** This architecture of Django has different components which require each other at certain parts of the application, at every instant, that increases the security of the overall website. As the model file will now only save on our server rather than saving on the webpage.

3. **Ease of Modification:** This is an important aspect of development as there are different components in Django architecture. If there is a change in different components, we don't have to change it in other components. This is actually one of the special features of Django, as here it provides us with much more adaptability of our website than other frameworks.

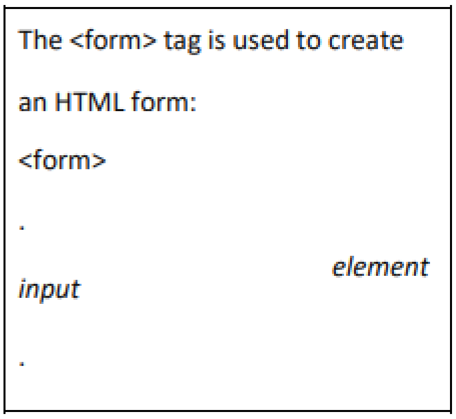
❖ HTML

HTML or Hyper Text Mark-up Language is the standard mark-up language used to create web pages.HTML was created in 1991 by Tim Berners-Lee at CERN in Switzerland. It was designed to allow scientists to display and share their research.

HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like <html>). HTML tags most commonly come in pairs like<h1> and </h1> , although some tags represent empty elements and so are unpaired, for example <img> first tag in a pair is the start tag, and the second tag is the end tag (they are also called opening tags and closing tags).

The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML

tags, but uses the tags to interpret the content of the page. HTML describes the structure of a website semantically along with cues for presentation, making it a mark-up language rather than a programming language.
HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as Java Script which affect the behaviour of HTML web pages. HTML is descriptive mark-up language. Library of various mark-up languages is defined in various browsers.
HTML Images: The **<img>** Tag and the Src Attribute: In HTML, images are defined with the <img> tag. The <img> tag is empty, which means that it contains attributes only, and has no closing tag. To display an image on a page, you need to use the src attribute. Src stands for "source". The value of the src attribute is the URL of the image you want to display.
Syntax for defining an image:<imgsrc=”url” alt=”some_text”>
HTML FORMS: HTML forms are used to pass data to a server. An HTML form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select lists, text area, field set, legend, and label elements.


The <form> tag is used to create

an HTML form:

<form>

.

input *element*

.

Figure 4.5: HTML form tag

❖ CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a mark-up language. While most often used to style web pages and user interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL. CSS is a cornerstone specification of the web and almost all web pages' use CSS style sheets to describe their presentation. CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colours, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content (such as by allowing for table less web design).

The major points of CSS are given below:

• CSS stands for Cascading Style Sheet.

• CSS is used to design HTML tags.

• CSS is a widely used language on the web.

• HTML, CSS and JavaScript are used for web designing. It helps the web designers to apply style on HTML tags.

With plain HTML you define the colours and sizes of text and tables throughout your pages. If you want to change a certain element you will therefore have to work your way through the document and change it. With CSS you define the colours and sizes in "styles". Then as you write your documents you refer to the styles. Therefore: if you change a certain style it will change the look of your entire site. Another big advantage is that CSS offers much more detailed attributes than plain HTML for defining the look and feel of your site.

❖ JAVASCRIPT

JavaScript (JS) is a dynamic computer programming language. It is most commonly used as part of web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed. It is also being used in server-side network programming (with Node.js), game development and the creation of desktop and mobile applications. JavaScript is a prototype-based scripting language with dynamic typing and has first- class functions. Its syntax was influenced by C. JavaScript copies many names and naming conventions from Java, but the two languages are otherwise unrelated and have very different semantics. The key design principles within JavaScript are taken from the Self and Scheme programming languages. It is a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles. The application of JavaScript in use outside of web pages—for example, in PDF documents, site-specific browsers, and desktop widgets—is also significant. Newer and faster JavaScript VMs and platforms built upon them (notably Node.js) have also increased the popularity of JavaScript for server-side web applications. On the client side, JavaScript was traditionally implemented as an interpreted language but just-in-time compilation is now performed by recent (post-2012) browsers.

❖ BOOTSTRAP

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS and JavaScript based design templates for typography, forms, buttons, navigation, and other interface components. Bootstrap is a HTML, CSS & JS Library that focuses on simplifying the development of informative web pages (as opposed to web apps). The primary purpose of adding it to a web project is to apply Bootstrap's choices of colour, size, font and layout to that project. As such, the primary factor is whether the developers in charge find those choices to their liking. Once added to a project, Bootstrap provides basic style definitions for all HTML elements.

CHAPTER 5
SYSTEM MODEL

The Online Food Order System application is a web-based system. The main advantage of this system is that it greatly simplifies the ordering process for both the customer and the restaurant and also greatly lightens the load on the restaurant's end, as the entire process of taking orders is automated. Anticipated Benefits are:

- This will minimize the number of employees at the back of the counter.
- The system will help to reduce labour cost involved.
- The system will be less probable to make mistake, since it's a machine.
- This will avoid long queues at the counter due to the speed of execution and number of optimum screens to accommodate the maximum throughput.

The structure of the system can be divided into 3 main logical components:

- Web Ordering System- provides the functionality for customers to place their order and supply necessary details.
- Menu Management-allows the restaurant to control what can be ordered by the customers
- Order Retrieval System-This is a final logical component. Allows restaurant to keep track of all orders placed. This component takes care of order retrieving and displaying order information.

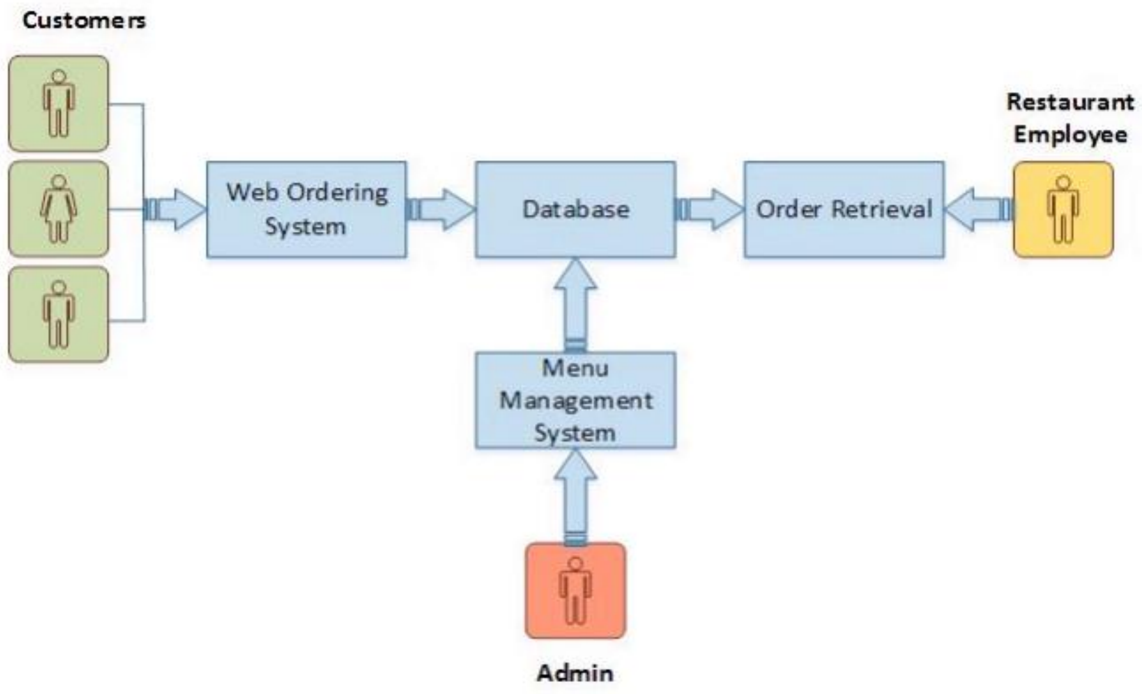


Figure 5.1: System Model

5.1 Web Ordering System Module

This module provides the functionality for customers to place their order and supply necessary details. Users of the system, namely restaurant customers, must be provided the following functionality:

- Create an account.
- Manage their account.
- Log in to the system.
- Navigate the restaurant's menu.
- Select an item from the menu.
- Add an item to their current order.
- Review their current order.
- Remove an item/remove all items from their current order.
- Provide payment details.
- Place an order.
- Receive confirmation in the form of an order number.
- View order placed.

5.2 Menu Management System Module

This module provides functionality for the power user-Administrator only. It will not be available to any other users of the system like Restaurant Employees or Customers. Using a graphical interface, it will allow an Admin to manage the menu that is displayed to users of the web ordering system:

- Add/update/delete food category to/from the menu.
- Add /update/delete food item to/from the menu.
- Update price for a given food item.

Before customers can actually use this system, functionality provided by this component will have to be configured first. Once the initial configuration is done, this will be the least likely used component as menu updates are mostly seasonal and do not occur frequently.

5.3 Order Retrieval System Module

This is the simplest module out of all 3 modules. It is designed to be used only by restaurant employees, and provides the following functions:

- Retrieve new orders from the database.
- Display the orders in an easily readable, graphical way.

CHAPTER 6
SYSTEM DESIGN

6.1 ENTITIES AND ATTRIBUTES

An entity may be an object with a physical existence (for e.g. A particular person, car or employee) or it may be an object with a conceptual existence (e.g., company, university).

Each entity has attributes i.e., the particular properties that describe it. The attribute values that describe each entity become a major part of the data store in the database. Whenever an attribute of one entity type refers to another entity type, a relationship exists. In the initial design of entity types, relationships are typically captured in the form of attributes. As the design is refined these attributes get converted into relationships between entity types. In the ER diagrams, the emphasis is on representing the schemas rather than the instances. This is more useful in the database design because a database schema changes rarely, whereas contents of the entity sets change frequently.

6.1.1 Types of Attributes

1. **Simple attribute: -** An attribute is classified as a simple attribute if it cannot be partitioned into smaller components. For example, age and sex of a person. A simple attribute is represented by an oval.
2. **Composite attribute: -** A composite attribute can be subdivided into smaller components which further form attributes. For example, ‘name’ attribute of an entity “person” can be broken down into first name and last name which further form attributes. Grouping of these related attributes forms a composite attribute.
3. **Single valued attribute: -** If an attribute of a particular entity represents single value for each instance, then it is called a single-valued attribute. For example, Ramesh, Kamal and Suraj are the instances of entity ‘student’ and each of them is issued a separate roll number. A single oval is used to represent this attribute.
4. **Multi valued attribute: –** An attribute which can hold more than one value, it is then termed as multi-valued attribute. For example, phone number of a person. Symbol of multi- valued attribute is shown below

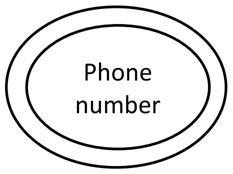


Figure 6.1:Symbol of Multivalued attribute

5. **Derived attribute:** A derived attribute calculates its value from another attribute. For example, 'age' is a derived attribute if it calculates its value from 'current date' & 'birth date' attributes. A derived attribute is represented by a dashed oval.
6. **Stored attribute:** In some cases ,two or more attribute values are related for example, the Age and Birth_date attributes of a person. For a particular person entity, the value of Age can be determined from the current date and the value of the person's Birth_date. The Age attribute hence called derived attribute and is said to be derivable from the Birth_date attribute, which is called a Stored attribute.
7. **Complex attribute**: is a Composite and Multivalued attribute.

6.2 RELATIONSHIPS

A relationship is defined as bond or attachment between 2 or more entities. Normally, a verb in a sentence signifies a relationship. For example,

- An employee assigned a project.
- Teacher teaches a student.
- Author writes a book.

A diamond is used to symbolically represent a relationship in the ERdiagram.

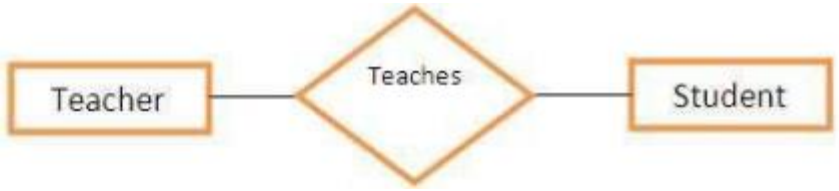


Figure 6.2 :Relationship connecting two Entities.

Various terms related to relationships

6.2.1 Degree of relationship: - It signifies the number of entities involved in a relationship. Degree of a relationship can be classified into following types:

- Unary relationship: - If only single entity is involved in a relationship then it is a unary relationship. For example, An employee (manager) supervises another employee.
- Binary relationships: - when two entities are associated to form a relation, then it is known as a binary relationship. For example, A person works in a company. Most of the times we use only binary relationship in an e-r diagram.
- Other types of relationships are ternary and quaternary. As the name signifies, a ternary relationship is associated with three entities and a quaternary relationship is associated with four entities.

6.2.2 Connectivity of a relationship: - Connectivity of a relationship describes, how many instances of one entity type are linked to how many instances of another entity type. Various categories of connectivity of a relationship are:

- **One to One (1:1) –** "Restaurant has menu" signifies a one-to-one relationship because only one instance of an entity is related with exactly one instance of another entity type.

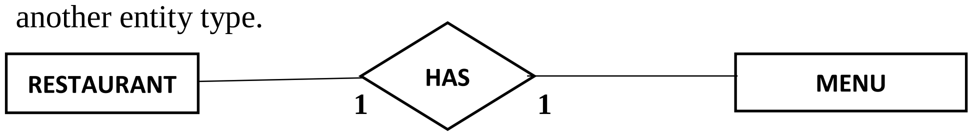


Figure 6.3: One to One Relationship

- **One to Many (1:M) –** "A Customer places Order" is a one-to-many relationship because a Customer can place more than one order, but a order is related to only one Customer.

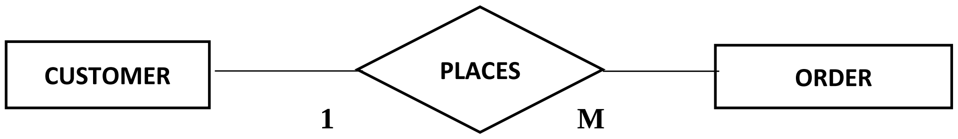


Figure 6.4 : One to Many Relationship

- **Many to One (M:1) –** "Menu has items" is a many-to-one relationship .

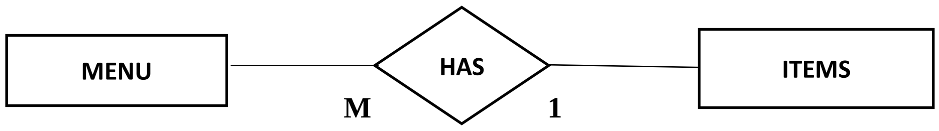


Figure 6.5 : Many to One Relationship

- **Many to Many (M: N) –** "Author writes books" is a many-to-many relationship because an author can write many books and a book can be written by many authors.

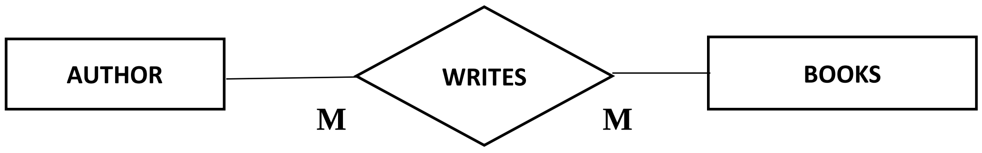


Figure 6.6 :Many to Many Relationship

6.2.3 Total Participation and Partial Participation

- **Total Participation:** It specifies that each entity in entity set must compulsorily participate in at least one relationship instance in that relationship set. It is also called as Mandatory Participation. The total participation is represented using a double line between the entity set and the relationship set.

- **Partial Participation:** It specifies that each entity in entity set may or may not participate in the relationship instance in that relationship set. It is also called as Optional Participation. The partial participation is represented using a single line between the entity set and the relationship set.

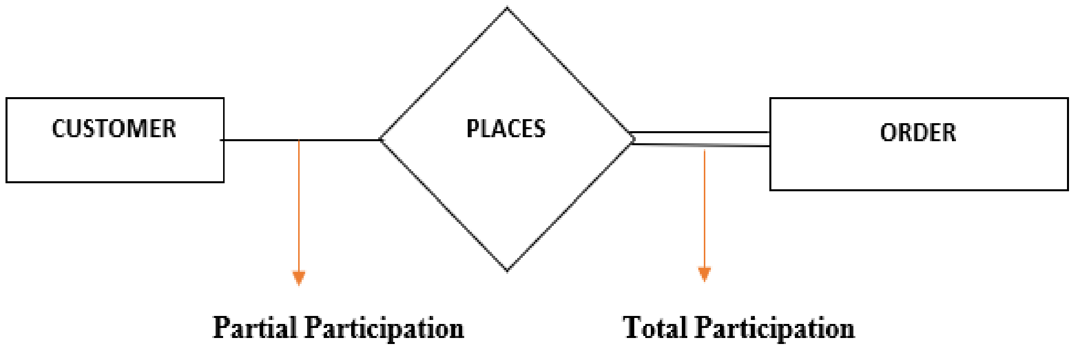


Figure 6.7: Total and Partial Participations

6.2.4 Types of Entities

Strong entity: A strong entity has a primary key attribute which uniquely identifies each entity. Symbol of strong entity is same as an entity.

Figure 6.8:Strong Entity

Weak entity: A weak entity does not have a primary key attribute and depends on other entity via a foreign key attribute.

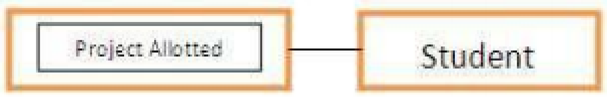


Figure 6.9 :Weak Entity

6.3 ER DIAGRAM

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. An Entity-Relationship (ER) model is an abstract way to describe a database. It is a popular high-level conceptual data model. Entity relationship diagrams (ER diagrams) are used to present the diagrammatic notations associated with ER model.

6.3.1 Notations for ER diagram

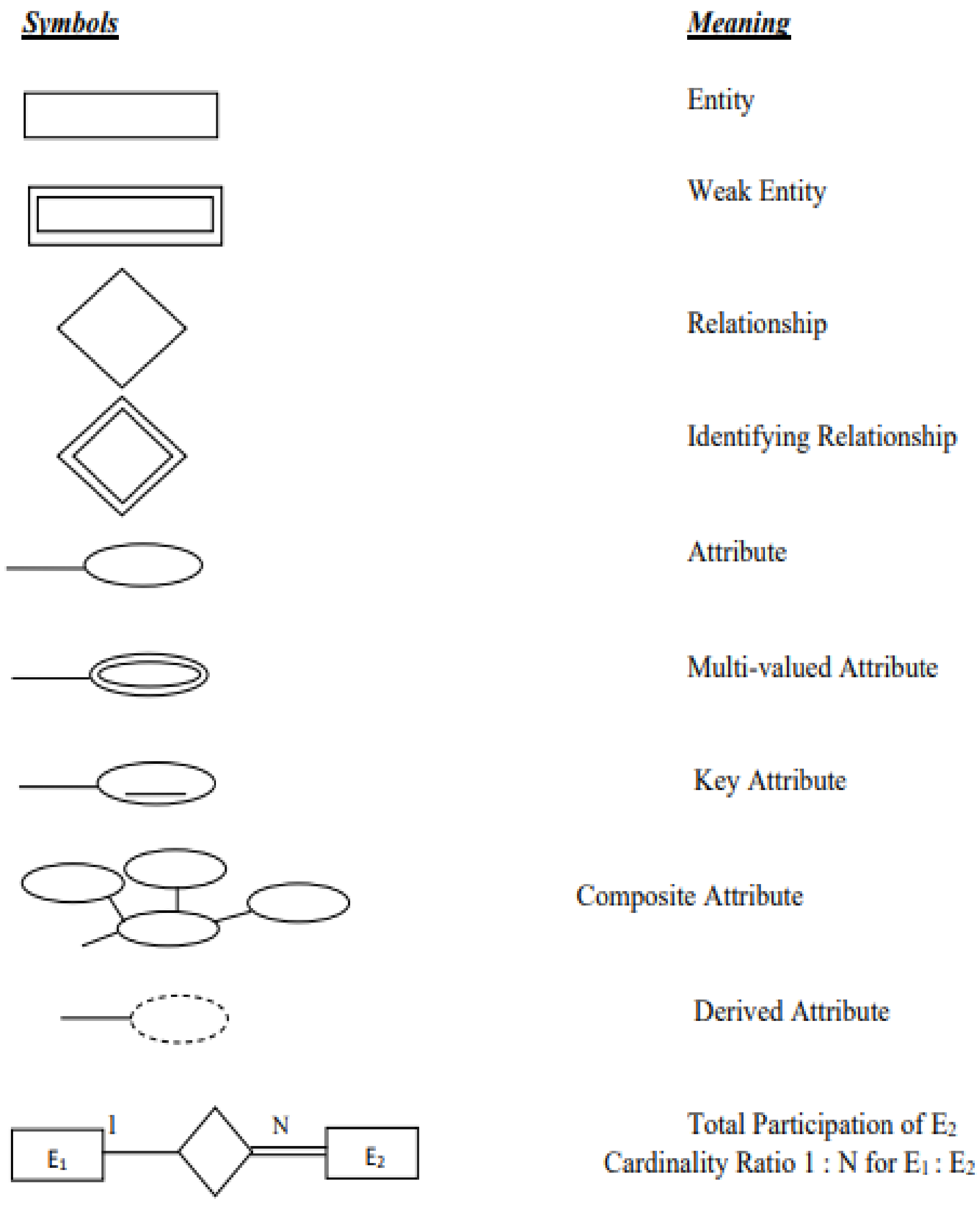


Figure 6.10: Notations for ER diagram

6.3.2 ER Diagram for Food Ordering System

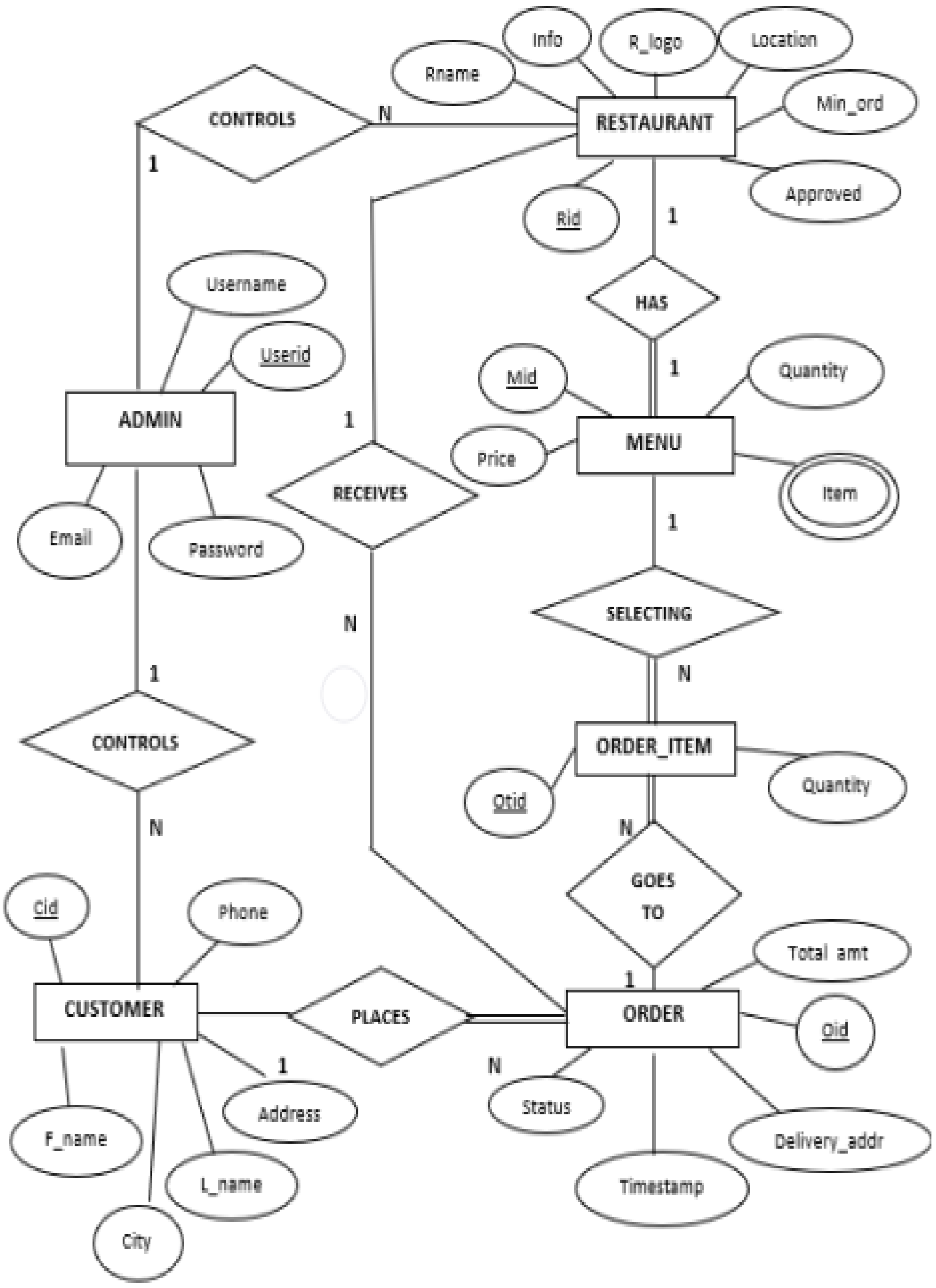


Figure 6.11 :ER Diagram for Food Ordering System

6.3.3 ER Diagram for Food Ordering System Explanation

The system that we have implemented has 6 entities.
They are as follows:

1. **Admin:** This entity contains the email address, password in encrypted format of all the users using the web application and some log entries such as last used date and time, also specifies whether the user is admin or not, if he is a customer or a restaurant user. The primary attribute is the Userid which is underlined.
2. **Restaurant:** This entity contains the list of restaurants registered with our web application, has attributes which describe whether the restaurant is open or not, info about the restaurant, if it is approved. Also contains a restaurant logo to display for the customer. Rid is the primary key in this entity.
3. **Customer:** Designed to contain the basic details of the person who orders food from the restaurant. These include name, phone number, address. These details are provided to the restaurant when the customer's order.
4. **Menu:** Menu from the restaurant is an entity aimed to contain the details like items in that particular restaurant, quantity of that particular item available, its price. Mid(menuid) is the attribute that is used to distinguish a particular row.
5. **Order Item:** This is an entity that is designed to add features such as cart, when the customer selects item, the items along with its quantities are shown to the customer again to confirm his order.
6. **Order:** As expected it is the entity that contains the final order or selection of items and its quantity made by the customer. Also has details such as timestamp of the order, total amount of the order, delivery address for the order made, and the current status. Orderid(Oid) is the primary key.
   - The Item attribute is made multi-valued to accommodate different types. It is resolved as per the rules in DBMS and hence becomes a separate table.
   - The relationship "controls" has a cardinality ratio of 1: N, since an admin can control 'n' customers and restaurants.
   - The relationship "places" has a cardinality ratio of 1: N, as the same customer can make multiple orders.
   - The relationship "has" has a cardinality ratio of 1:1, since a restaurant can have only 1 menu containing all items.

- The relationship “receives” has a cardinality ratio of 1: N, as one restaurant can receive multiple orders.
- The relationship “goes to” has a cardinality ratio of N:1, as n items can contain in a cart.
- The relationship “selecting” has a cardinality ratio of 1: N since n items can be selected from one menu.

6.4 RELATIONAL DATABASE SCHEMA

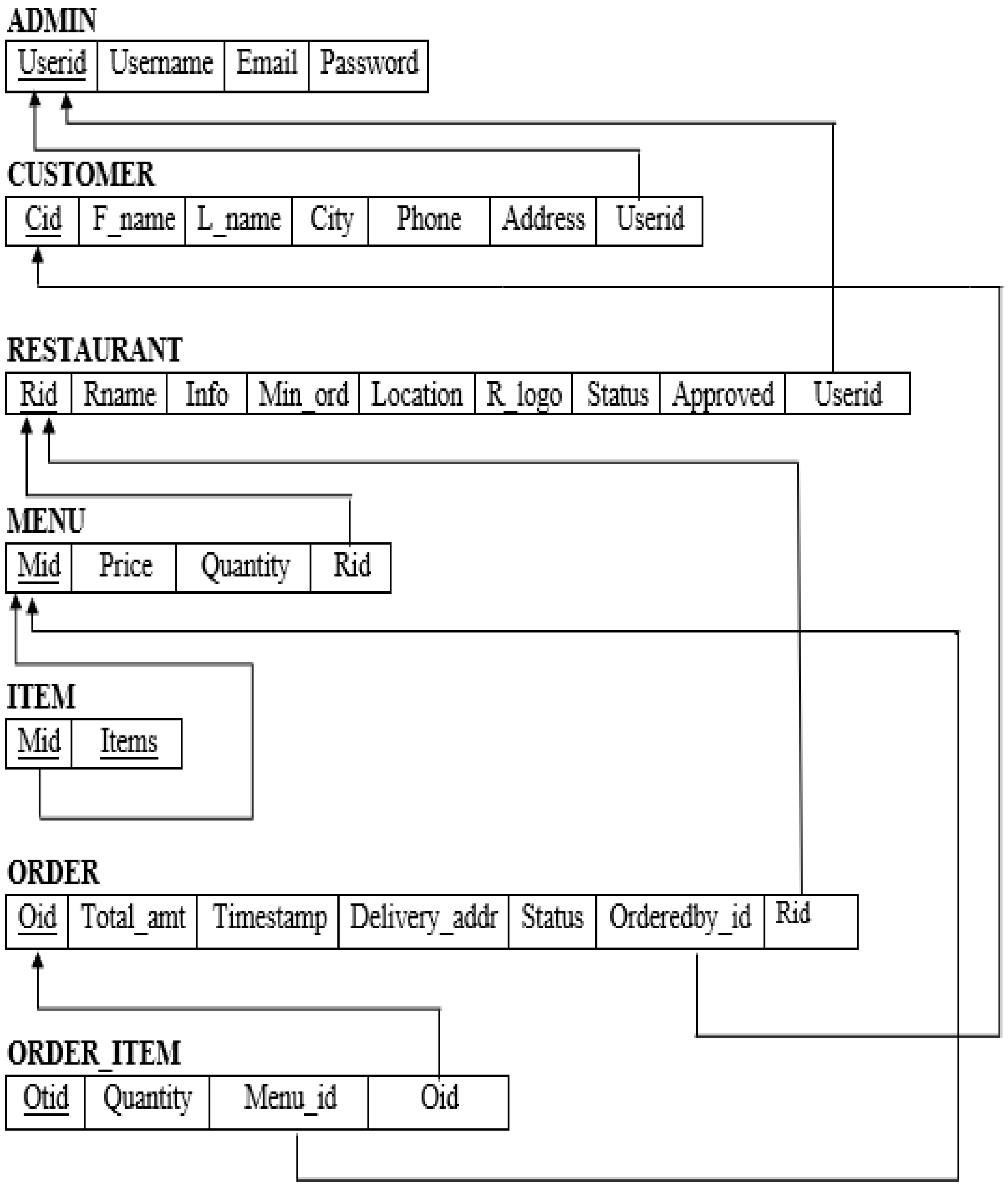

ADMIN
Userid | Username | Email | Password
CUSTOMER
Cid | F_name | L_name | City | Phone | Address | Userid
RESTAURANT
Rid | Rname | Info | Min_ord | Location | R_logo | Status | Approved | Userid
MENU
Mid | Price | Quantity | Rid
ITEM
Mid | Items
ORDER
Oid | Total_amt | Timestamp | Delivery_addr | Status | Orderedby_id | Rid
ORDER_ITEM
Otid | Quantity | Menu_id | Oid

Figure 6.12 : Relational Database Schema for Food Ordering System

6.4.1 Relational Mapping

A relational schema for a database is an outline of how data is organized. It can be a graphic illustration or another kind of chart used by programmers to understand how each table is laid out, including the columns and the types of data they hold and how tables connect. A database schema usually specifies which columns are primary keys in tables and which other columns have special constraints such as being required to have unique values in each record. In this section we describe the steps of an algorithm for ER to relational mapping. The above ER diagram is used to derive the respective relational schema. The mapping will create tables with simple single valued attributes.

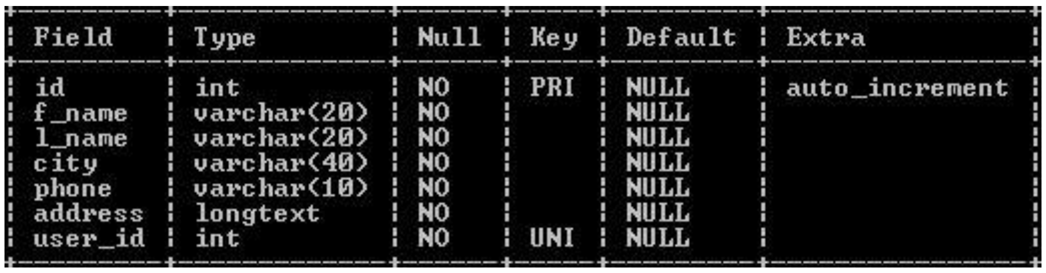
The steps are:

1. Mapping of Regular Entity Types.
2. Mapping of Weak Entity Types.
3. Mapping of Binary 1: 1 Relationship Types.
4. Mapping of Binary 1: N Relationship Types.
5. Mapping of Binary M : N Relationship Types.
6. Mapping of Multivalued Attributes.
7. Mapping of N-ary Relationship Types.

- ❖ **Mapping of 1: N Relationship type:** For each regular binary 1: n relationship type R, identify the relation S that represents the participating entity type at the N-side of the relationship type. Include as the foreign key in S the primary key of relation T that represents other entity type in R. We have to do this because each entity instance on n side is related to at most one entity instance on 1-side of relationship type.
- ❖ **Mapping of M: N Relationship type:** For each m:n relationship type R, create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity type, their combination will form the primary key of S. Also include any simple attributes of m: n relationship type.
- ❖ **Mapping of 1:1 Relationship type:** For each 1:1 relationship type R can be migrated to any participating entity types. This relationship type ensures that each user in the database can lodge one complaint

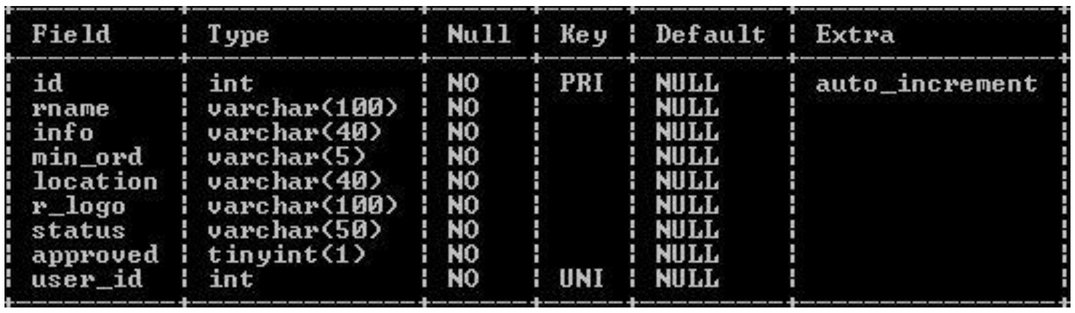
6.5 Database Tables/Relations

CUSTOMER TABLE



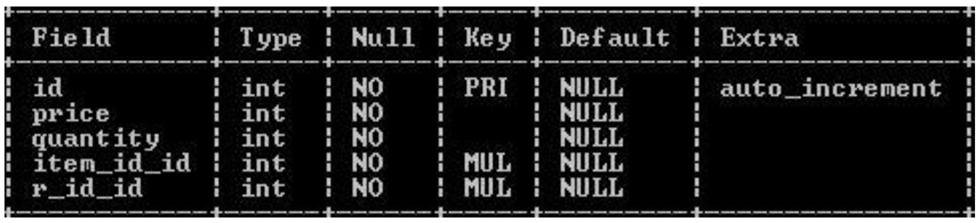
```
+---------+-------------+------+-----+---------+----------------+
| Field   | Type        | Null | Key | Default | Extra          |
+---------+-------------+------+-----+---------+----------------+
| id      | int         | NO   | PRI | NULL    | auto_increment |
| f_name  | varchar(20) | NO   |     | NULL    |                |
| l_name  | varchar(20) | NO   |     | NULL    |                |
| city    | varchar(40) | NO   |     | NULL    |                |
| phone   | varchar(10) | NO   |     | NULL    |                |
| address | longtext    | NO   |     | NULL    |                |
| user_id | int         | NO   | UNI | NULL    |                |
+---------+-------------+------+-----+---------+----------------+
```

RESTAURANT TABLE



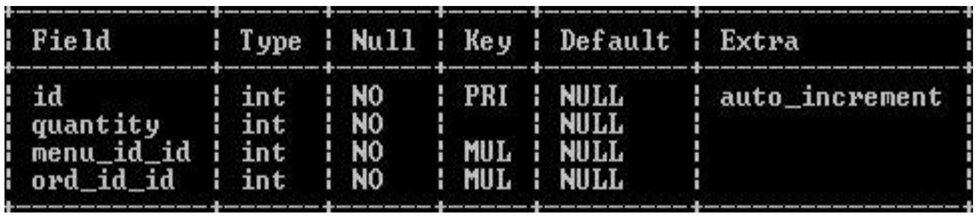
```
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| id       | int          | NO   | PRI | NULL    | auto_increment |
| rname    | varchar(100) | NO   |     | NULL    |                |
| info     | varchar(40)  | NO   |     | NULL    |                |
| min_ord  | varchar(5)   | NO   |     | NULL    |                |
| location | varchar(40)  | NO   |     | NULL    |                |
| r_logo   | varchar(100) | NO   |     | NULL    |                |
| status   | varchar(50)  | NO   |     | NULL    |                |
| approved | tinyint(1)   | NO   |     | NULL    |                |
| user_id  | int          | NO   | UNI | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
```

MENU TABLE



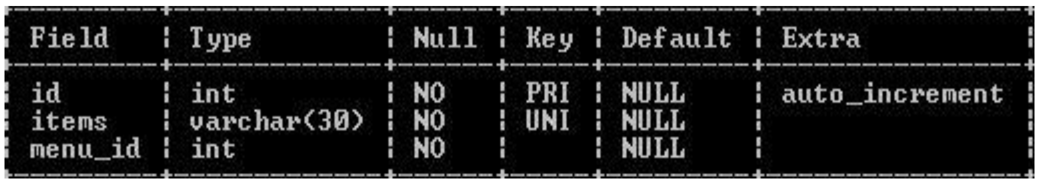
```
+------------+------+------+-----+---------+----------------+
| Field      | Type | Null | Key | Default | Extra          |
+------------+------+------+-----+---------+----------------+
| id         | int  | NO   | PRI | NULL    | auto_increment |
| price      | int  | NO   |     | NULL    |                |
| quantity   | int  | NO   |     | NULL    |                |
| item_id_id | int  | NO   | MUL | NULL    |                |
| r_id_id    | int  | NO   | MUL | NULL    |                |
+------------+------+------+-----+---------+----------------+
```

ITEM TABLE



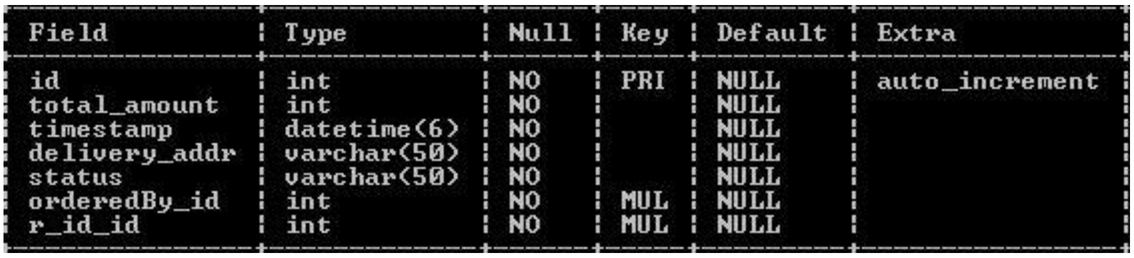
```
+------------+------+------+-----+---------+----------------+
| Field      | Type | Null | Key | Default | Extra          |
+------------+------+------+-----+---------+----------------+
| id         | int  | NO   | PRI | NULL    | auto_increment |
| quantity   | int  | NO   |     | NULL    |                |
| menu_id_id | int  | NO   | MUL | NULL    |                |
| ord_id_id  | int  | NO   | MUL | NULL    |                |
+------------+------+------+-----+---------+----------------+
```

ORDER_ITEM TABLE



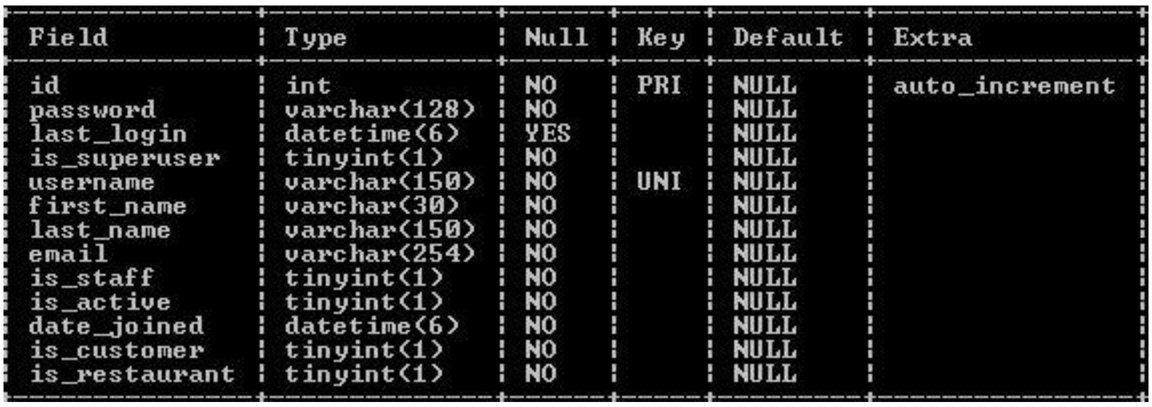
```
+---------+-------------+------+-----+---------+----------------+
| Field   | Type        | Null | Key | Default | Extra          |
+---------+-------------+------+-----+---------+----------------+
| id      | int         | NO   | PRI | NULL    | auto_increment |
| items   | varchar(30) | NO   | UNI | NULL    |                |
| menu_id | int         | NO   |     | NULL    |                |
+---------+-------------+------+-----+---------+----------------+
```

ORDER TABLE



```
+----------------+-------------+------+-----+---------+----------------+
| Field          | Type        | Null | Key | Default | Extra          |
+----------------+-------------+------+-----+---------+----------------+
| id             | int         | NO   | PRI | NULL    | auto_increment |
| total_amount   | int         | NO   |     | NULL    |                |
| timestamp      | datetime(6) | NO   |     | NULL    |                |
| delivery_addr  | varchar(50) | NO   |     | NULL    |                |
| status         | varchar(50) | NO   |     | NULL    |                |
| orderedBy_id   | int         | NO   | MUL | NULL    |                |
| r_id_id        | int         | NO   | MUL | NULL    |                |
+----------------+-------------+------+-----+---------+----------------+
```

USER/ADMIN TABLE



```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| id            | int          | NO   | PRI | NULL    | auto_increment |
| password      | varchar(128) | NO   |     | NULL    |                |
| last_login    | datetime(6)  | YES  |     | NULL    |                |
| is_superuser  | tinyint(1)   | NO   |     | NULL    |                |
| username      | varchar(150) | NO   | UNI | NULL    |                |
| first_name    | varchar(30)  | NO   |     | NULL    |                |
| last_name     | varchar(150) | NO   |     | NULL    |                |
| email         | varchar(254) | NO   |     | NULL    |                |
| is_staff      | tinyint(1)   | NO   |     | NULL    |                |
| is_active     | tinyint(1)   | NO   |     | NULL    |                |
| date_joined   | datetime(6)  | NO   |     | NULL    |                |
| is_customer   | tinyint(1)   | NO   |     | NULL    |                |
| is_restaurant | tinyint(1)   | NO   |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
```

6.6 NORMALIZATION OF RELATIONS

6.6.1 Functional Dependency

The Functional Dependency denoted by X →Y, between two sets of attributes X and Y that are subsets of R species a constraint on the possible tuples that can form a relation stare r of R. The constraints is that for any two tuples t1 and t2 in r that have t1[X] = t2[X]. This means that the values of the Y component of a tuple in r depend on, or determined by the values of the X components. Alternatively, the values of the X component of a tuple uniquely determine the values of the Y component. Consider the following schema,

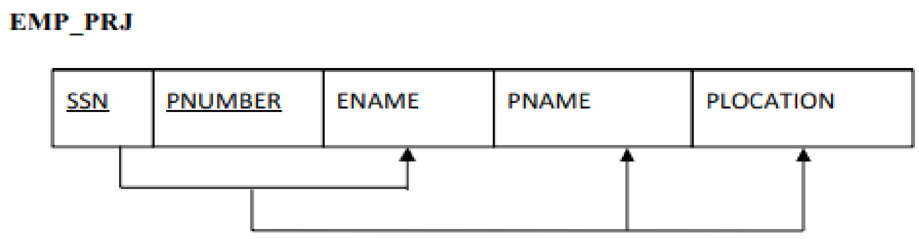


Figure 6.13: Example of Functional Dependency

In the above schema (Fig 6.13) the functional dependencies are:

1. SSN → ENAME The value of an employee's social security number (SSN) uniquely determines the employee name (ENAME).

2. PNUMBER →{ PNAME, PLOCATION} These values of project's number uniquely determine the project name (PNAME) and project locations (PLOCATIONS).

6.6.2 Normalization

The normalization process was proposed by Codd, it takes a relation schema through a series of tests to certify whether if satisfies a certain normal form. The process proceeds in a top-down fashion by evaluating each relation against the criteria for normal forms and decomposing relations as necessary can thus be considered a relational design by analysis. Normalization of data can be looked upon as a process of analysing the given relation schemas based on their Functional Dependencies and primary keys to achieve the desirable properties of:

- Minimizing redundancy.
- Minimizing the insertion, deletion, and update anomalies.

Normal form of a relation refers to the highest normal form condition that meets, and hence indicates the degree to which it has been normalized.

6.6.3 Definitions of keys and Attributes Participating in Keys

- ❖ **Super Key:** A super key of a relation schema R= {A1, A2… An} is asset of attributes S C R with the property that no two tuples t1 and t2 in any legal relation state r of R will have t1[S] = t2[S]. A key K is a super key with the additional property that removal of any attribute from K will cause K not to be a super key anymore.
- ❖ **Candidate Key:** If a relation schema has more than one key, each key is called candidate key.
- ❖ **Prime Attribute:** An attribute of relation schema R is called a prime attribute of R if it is a member of some candidate key of R.
- ❖ **Non-Prime Attribute:** An attribute is called nonprime if it is not a prime attribute or it is not a member of any candidate.

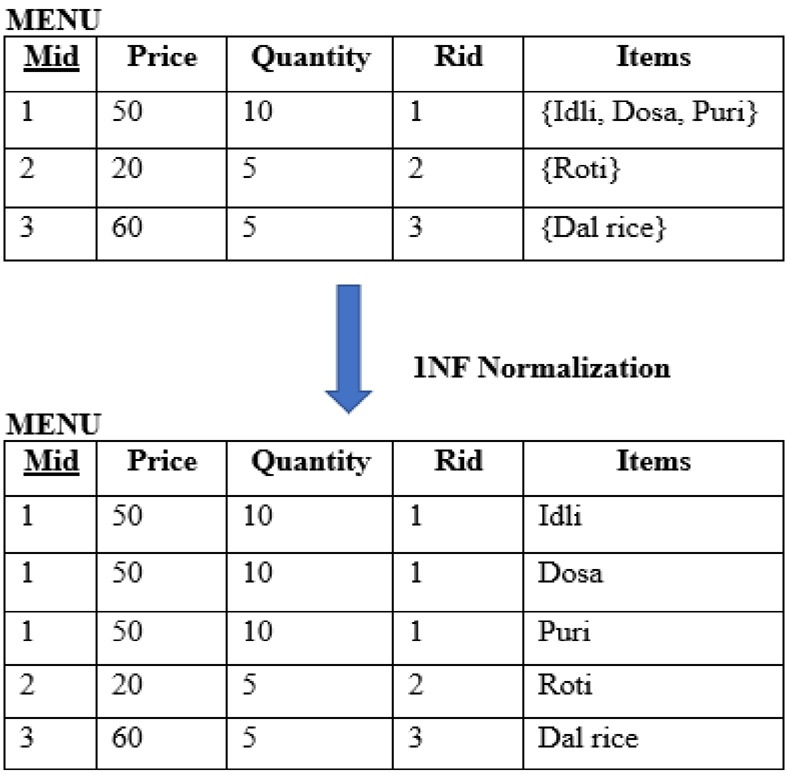
6.6.4 Normal Forms

There are three normal forms

- First normal form
- Second normal form
- Third normal form

These were proposed by Codd as a sequence to achieve the desirable state of 3NF relations by progressing through the intermediate states of 1Nf and 2NF if needed.

• **First Normal Form (1NF):** It states that the domain of attribute must include only atomic be values and that the value of any attribute in a tuple must be a single value from the domain of the attribute. Hence 1NF disallows having a set of values a tuple of values or a combination of both as an attribute value for a single tuple. In other words, 1NF disallows relations within relations or relations as attribute values within tuples.



MENU

Mid	Price	Quantity	Rid	Items
1	50	10	1	{Idli, Dosa, Puri}
2	20	5	2	{Roti}
3	60	5	3	{Dal rice}

1NF Normalization

MENU

Mid	Price	Quantity	Rid	Items
1	50	10	1	Idli
1	50	10	1	Dosa
1	50	10	1	Puri
2	20	5	2	Roti
3	60	5	3	Dal rice

Figure 6.14: 1NF version of the MENU relation

• **Second Normal Form (2NF):** This normal form is based on the full functional dependency. A functional dependency X →Y is full functional dependency if removal for any attribute A from X means that dependency does not hold any more.

A relation schema R is in 2NF if every nonprime attribute A in R is fully functionally dependent on the primary key of R

If a relation schema is not in 2NF, it can be second normalized or 2NF normalized into number of 2NF relations in which nonprime attributes are associated only with the part of the primary key on which they are fully functionally dependent. The test for 2NF involves testing for functional dependencies whose left-hand side attributes are part of the primary key.

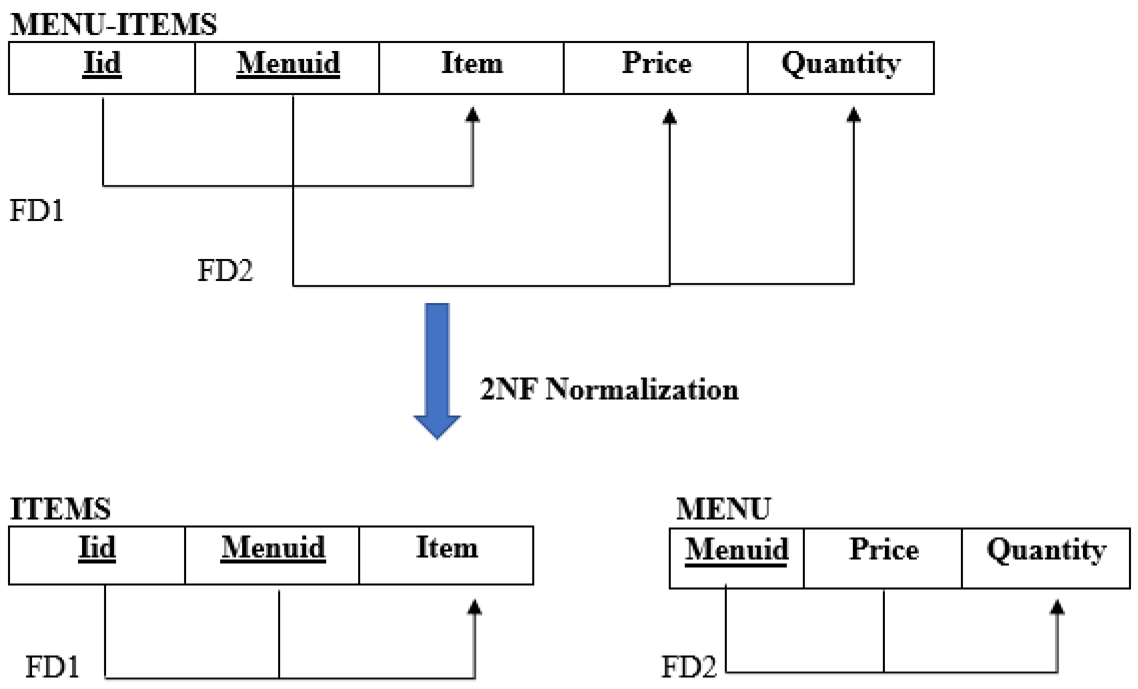


Figure 6.15: Normalizing MENU-ITEM into 2NF

• Third Normal Form: Third normal form (3NF) is based on the concept of transitive dependency. A functional dependency X→Y in a relation schema R is a transitive dependency if there is a set of attributes Z that is neither a candidate key nor a subset of any key of R and both X→Z and Z→Y hold.

A relation schema R is in 3NF if it satisfies 2NF and no nonprime attribute of R is transitively dependent on the primary key.

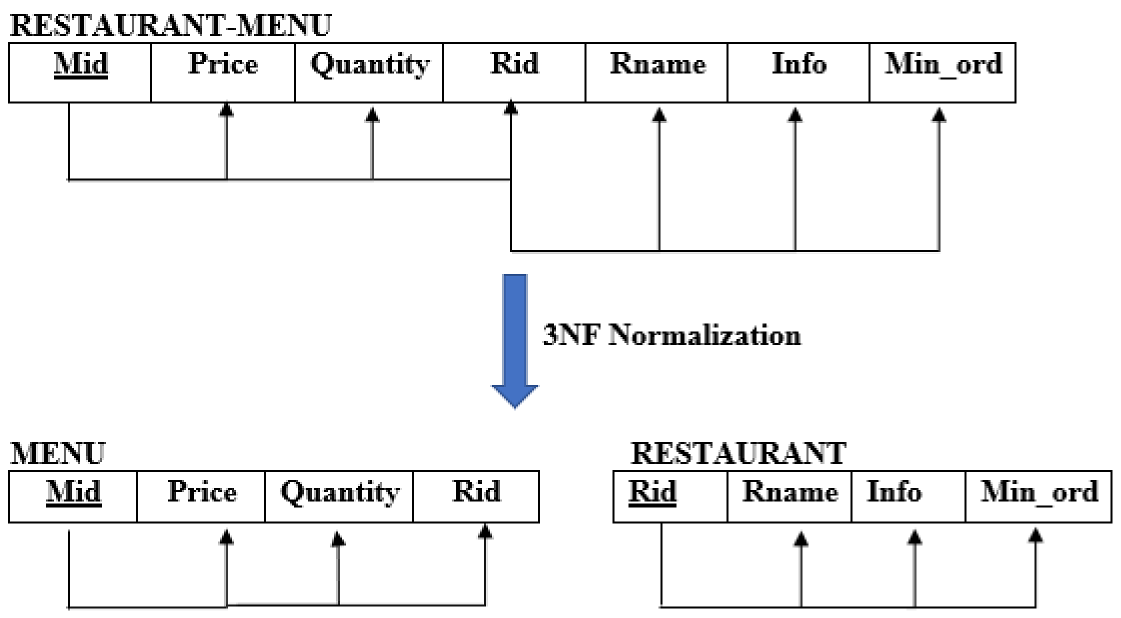


Figure 6.16: Normalizing RESTAURANT-MENU into 3NF

CHAPTER 7

IMPLEMENTATION

7.1 DJANGO USER DEFINED FUNCTIONS

Besides the built-in Django functions, we can create our own functions. A function is a block of statements that can be used repeatedly in a program. A function will not execute immediately when a page load. A function will be executed by a call to the function.

Create a User Defined Function in Django

A user-defined function declaration starts with the word function:

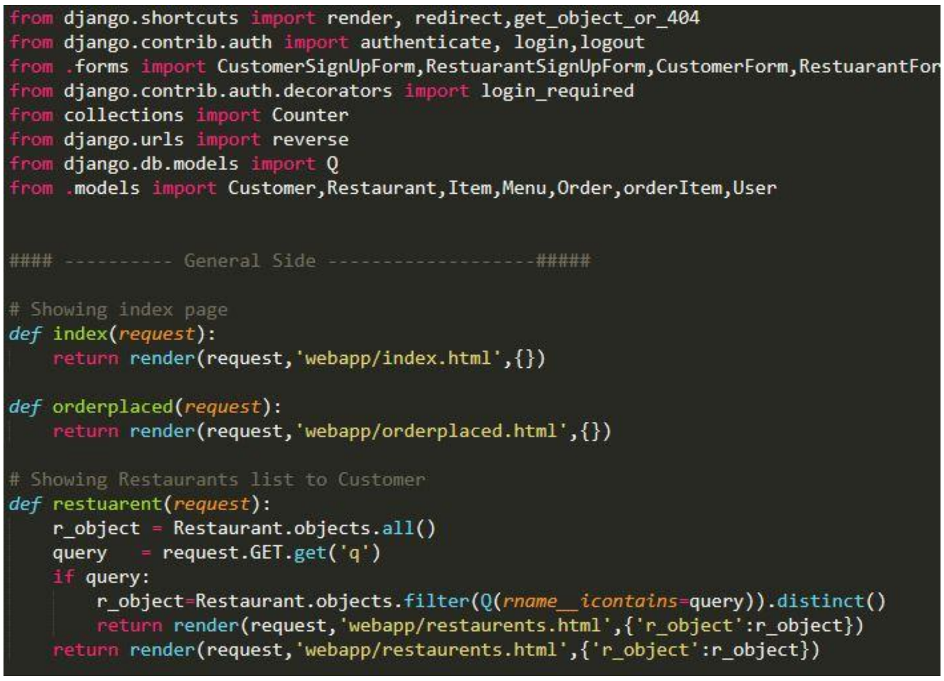
Syntax:

def functionName():

code to be executed

7.1.1 Django Views.py Functions

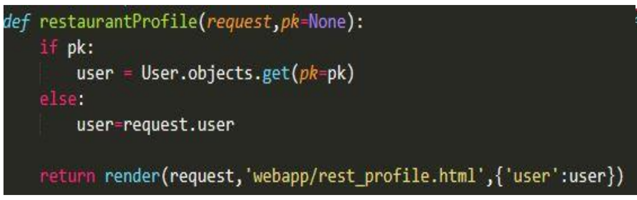
In views.py, we create functions which renders html pages, a function will not execute immediately when a page loads. A function will be executed by a call to the function.



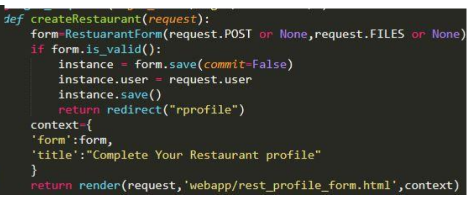
```
from django.shortcuts import render, redirect,get_object_or_404
from django.contrib.auth import authenticate, login,logout
from .forms import CustomerSignUpForm,RestuarantSignUpForm,CustomerForm,RestuarantFor
from django.contrib.auth.decorators import login_required
from collections import Counter
from django.urls import reverse
from django.db.models import Q
from .models import Customer,Restaurant,Item,Menu,Order,orderItem,User


#### ---------- General Side -------------------#####

# Showing index page
def index(request):
    return render(request,'webapp/index.html',{})

def orderplaced(request):
    return render(request,'webapp/orderplaced.html',{})

# Showing Restaurants list to Customer
def restuarent(request):
    r_object = Restaurant.objects.all()
    query    = request.GET.get('q')
    if query:
        r_object=Restaurant.objects.filter(Q(rname__icontains=query)).distinct()
        return render(request,'webapp/restaurents.html',{'r_object':r_object})
    return render(request,'webapp/restaurents.html',{'r_object':r_object})
```

Figure 7.1 :Snapshot of Views.py

The user defined functions written are index, orderplaced, restaurant, logout, customer register, customer Login, customer Profile, create Customer, update Customer, restaurant menu, checkout, rest register, rest login, restaurant profile, create restaurant, update restaurant, menu manipulation, orderlist.



```python
def restaurantProfile(request,pk=None):
    if pk:
        user = User.objects.get(pk=pk)
    else:
        user=request.user

    return render(request,'webapp/rest_profile.html',{'user':user})
```

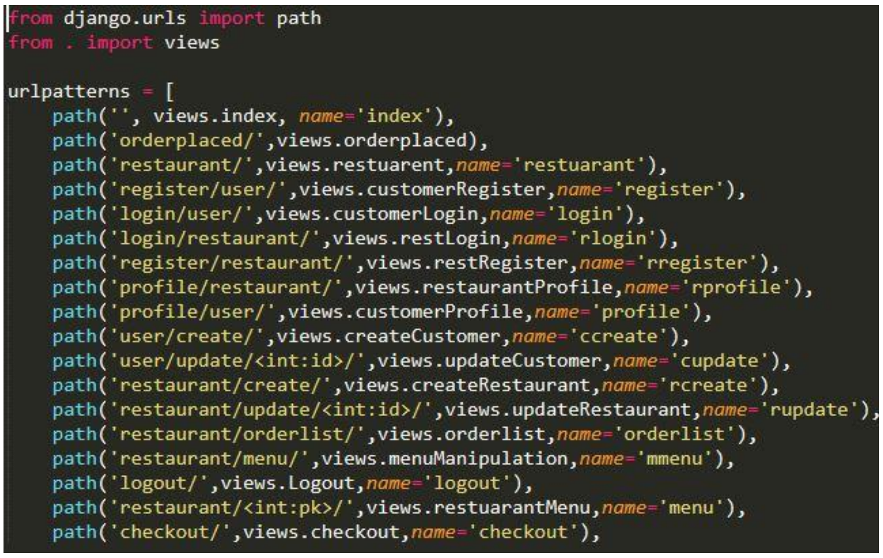


```python
def createRestaurant(request):
    form=RestuarantForm(request.POST or None,request.FILES or None)
    if form.is_valid():
        instance = form.save(commit=False)
        instance.user = request.user
        instance.save()
        return redirect("rprofile")
    context={
    'form':form,
    'title':"Complete Your Restaurant profile"
    }
    return render(request,'webapp/rest_profile_form.html',context)
```

Figure 7.2: User defined Functions restaurantProfile and createRestaurant

7.1.2 Django-urls.py

In urls.py we define the path of the pages to be rendered by the views.py functions

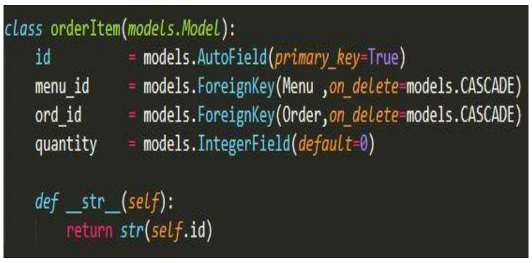


```python
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('orderplaced/',views.orderplaced),
    path('restaurant/',views.restuarent,name='restuarant'),
    path('register/user/',views.customerRegister,name='register'),
    path('login/user/',views.customerLogin,name='login'),
    path('login/restaurant/',views.restLogin,name='rlogin'),
    path('register/restaurant/',views.restRegister,name='rregister'),
    path('profile/restaurant/',views.restaurantProfile,name='rprofile'),
    path('profile/user/',views.customerProfile,name='profile'),
    path('user/create/',views.createCustomer,name='ccreate'),
    path('user/update/<int:id>/',views.updateCustomer,name='cupdate'),
    path('restaurant/create/',views.createRestaurant,name='rcreate'),
    path('restaurant/update/<int:id>/',views.updateRestaurant,name='rupdate'),
    path('restaurant/orderlist/',views.orderlist,name='orderlist'),
    path('restaurant/menu/',views.menuManipulation,name='mmenu'),
    path('logout/',views.Logout,name='logout'),
    path('restaurant/<int:pk>/',views.restuarantMenu,name='menu'),
    path('checkout/',views.checkout,name='checkout'),
```

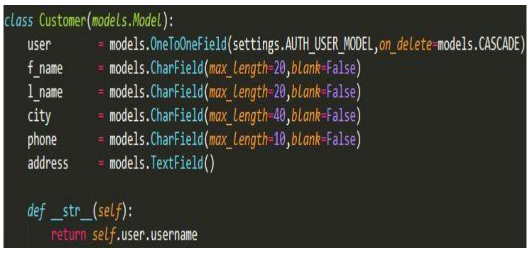
Figure 7.3: Snapshot of urls.py

7.1.3: Django-models.py

Here we define the database and the tables to be added to the database, the tables are created by writing python classes.



```python
class orderItem(models.Model):
    id          = models.AutoField(primary_key=True)
    menu_id     = models.ForeignKey(Menu ,on_delete=models.CASCADE)
    ord_id      = models.ForeignKey(Order,on_delete=models.CASCADE)
    quantity    = models.IntegerField(default=0)

    def __str__(self):
        return str(self.id)
```

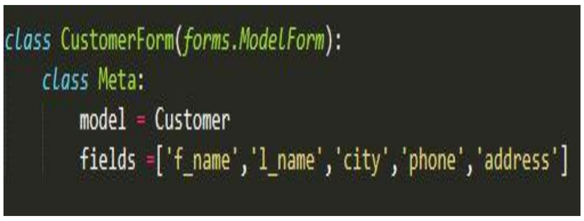


```python
class Customer(models.Model):
    user        = models.OneToOneField(settings.AUTH_USER_MODEL,on_delete=models.CASCADE)
    f_name      = models.CharField(max_length=20,blank=False)
    l_name      = models.CharField(max_length=20,blank=False)
    city        = models.CharField(max_length=40,blank=False)
    phone       = models.CharField(max_length=10,blank=False)
    address     = models.TextField()

    def __str__(self):
        return self.user.username
```

Figure 7.4: Snapshots of models.py

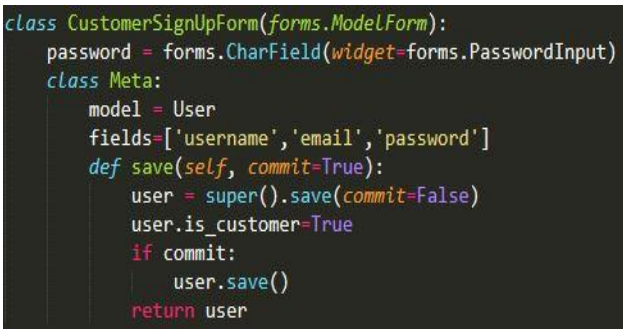
7.1.4: Django-forms.py

In forms.py we define the forms to add to the database externally.



```python
class CustomerForm(forms.ModelForm):
    class Meta:
        model = Customer
        fields =['f_name','l_name','city','phone','address']
```

Figure 7.5: Snapshot of CustomerForm code



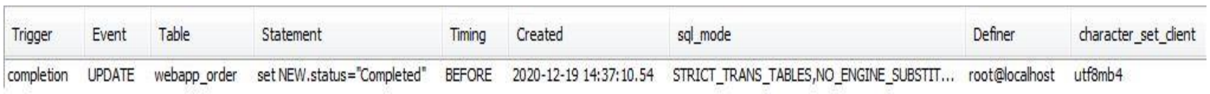
```python
class CustomerSignUpForm(forms.ModelForm):
    password = forms.CharField(widget=forms.PasswordInput)
    class Meta:
        model = User
        fields=['username','email','password']
        def save(self, commit=True):
            user = super().save(commit=False)
            user.is_customer=True
            if commit:
                user.save()
            return user
```

Figure 7.6 : Snapshot of forms.py

7.2 IMPLEMENTATION OF TRIGGER

Since our implementation does not include the delivery module, we managed to create a trigger which sets the status of the order to completed by default, but if it needs to be modified we just have to remove the trigger and set the new status. The implementation of the trigger is as follows:

create trigger completion
before update
on webapp_order
for each row
set NEW.status="Completed";



Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer	character_set_client
completion	UPDATE	webapp_order	set NEW.status="Completed"	BEFORE	2020-12-19 14:37:10.54	STRICT_TRANS_TABLES,NO_ENGINE_SUBSTIT...	root@localhost	utf8mb4

Figure 7.7: Snapshot of Trigger Implementation

CHAPTER 8

SNAPSHOTS

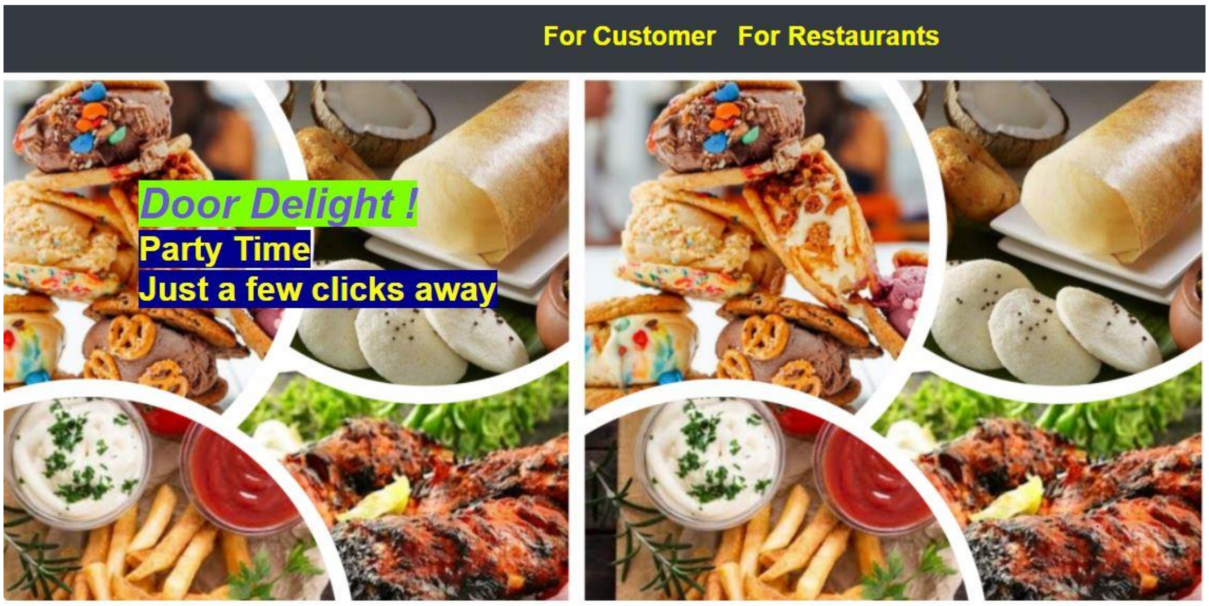
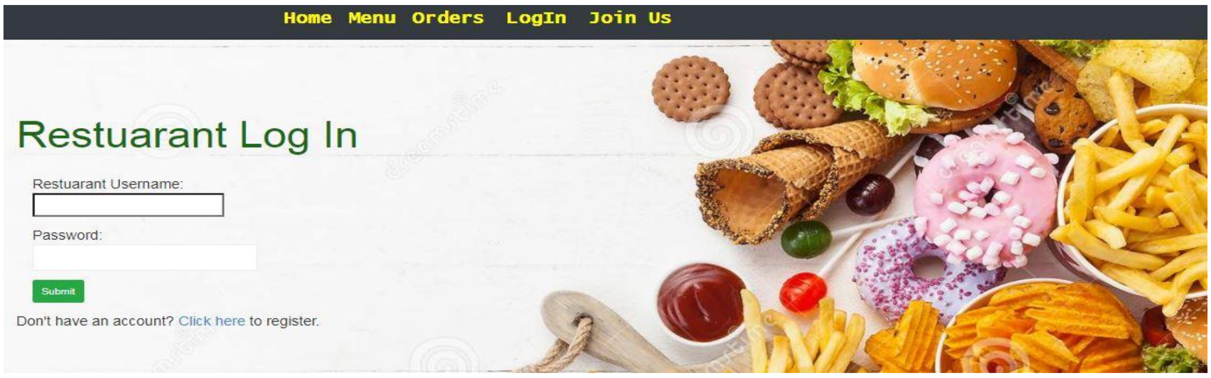


Figure 8.1: Homepage



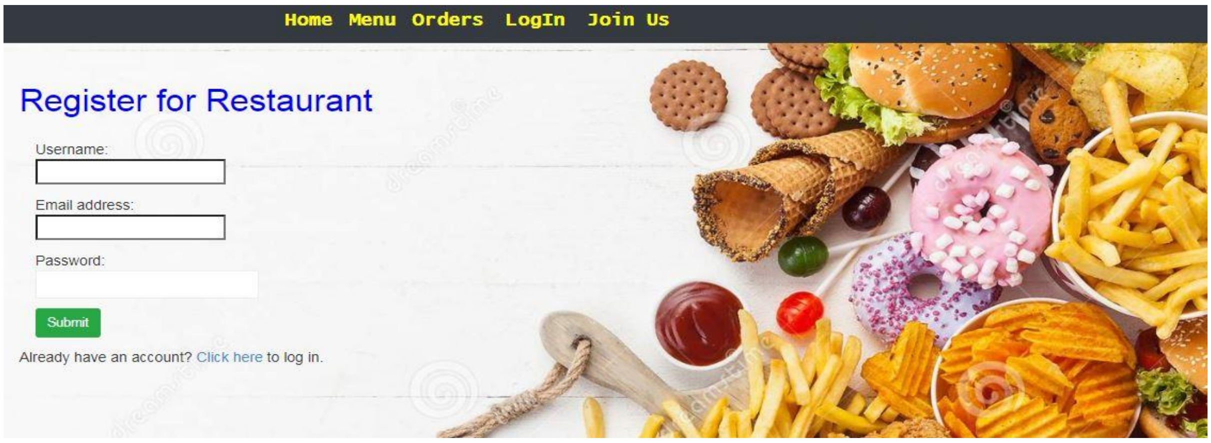
Restuarant Log In

Restuarant Username:

Password:

Don't have an account? Click here to register.

Figure 8.2: Restaurant Login Page



Register for Restaurant

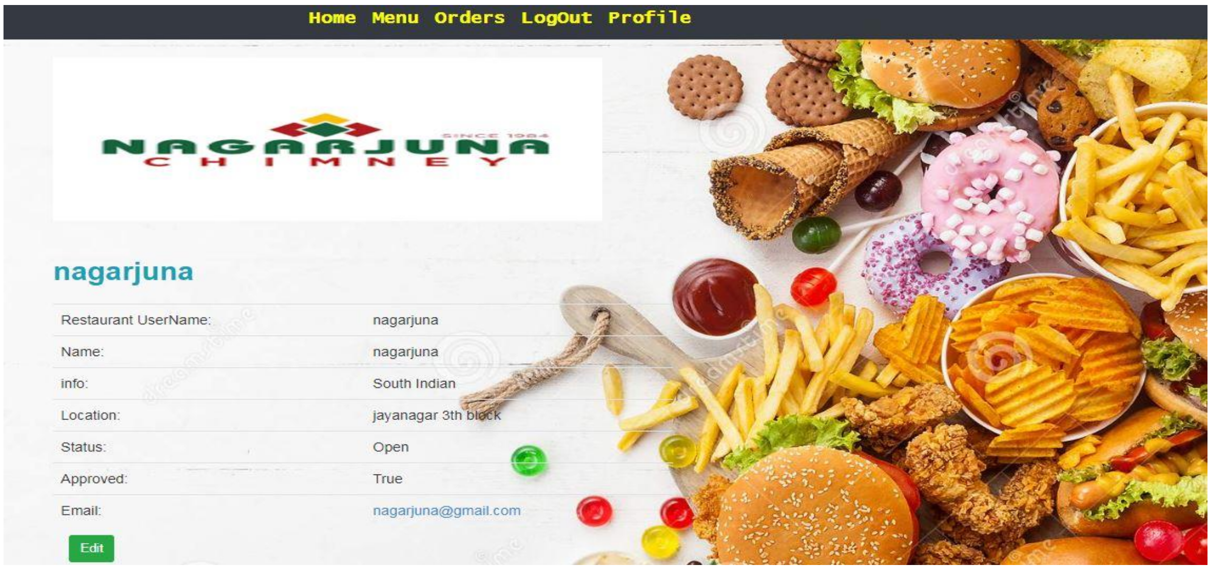
Username:

Email address:

Password:

Already have an account? Click here to log in.

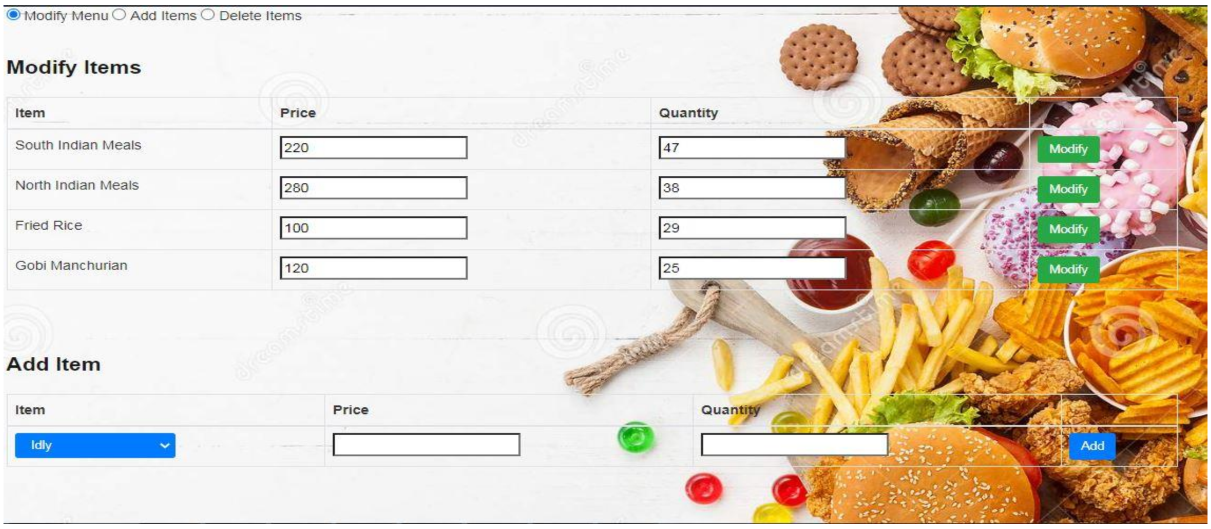
Figure 8.3: Restaurant Signup Page



nagarjuna

Restaurant UserName:	nagarjuna
Name:	nagarjuna
info:	South Indian
Location:	jayanagar 3th block
Status:	Open
Approved:	True
Email:	nagarjuna@gmail.com

Figure 8.4: Restaurant Dashboard



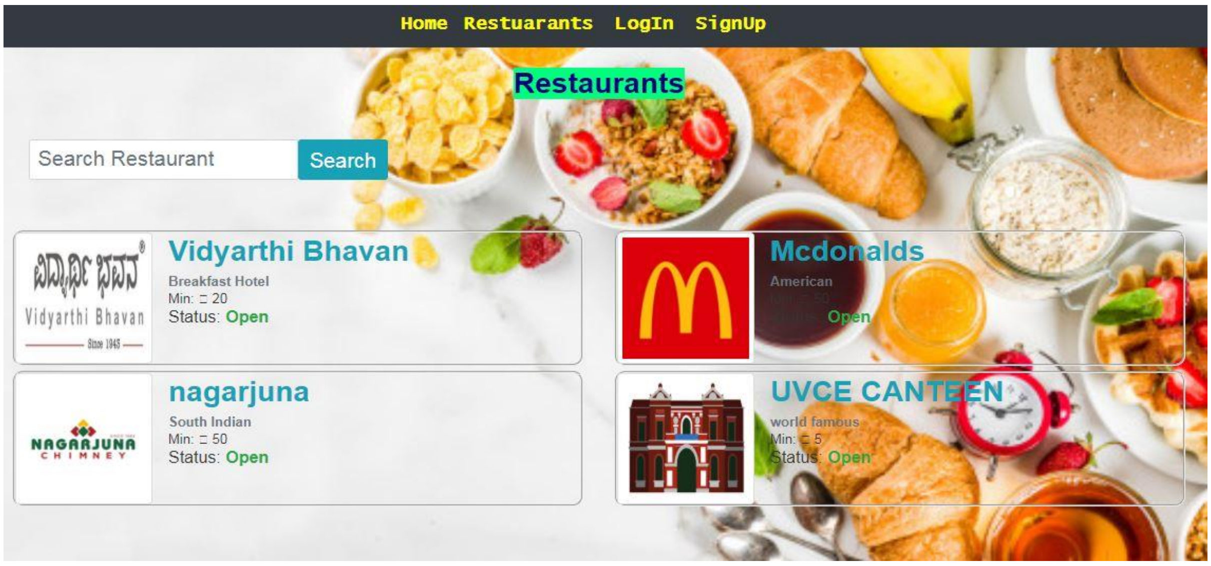
Modify Items

Item	Price	Quantity
South Indian Meals	220	47
North Indian Meals	280	38
Fried Rice	100	29
Gobi Manchurian	120	25

Add Item

Item	Price	Quantity
Idly		

Figure 8.5: Modify, add, deleting items from menu



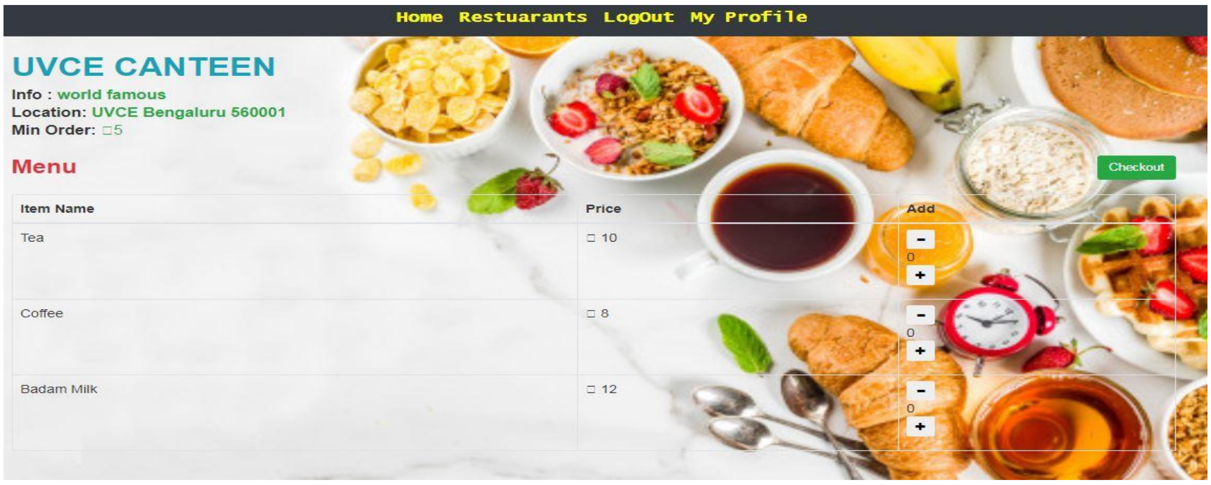
Vidyarthi Bhavan
Breakfast Hotel
Min: ₹ 20
Status: Open

Mcdonalds
American
Open

nagarjuna
South Indian
Min: ₹ 50
Status: Open

UVCE CANTEEN
world famous
Min: ₹ 5
Status: Open

Figure 8.6: Restaurants List



Home Restuarants LogOut My Profile

UVCE CANTEEN

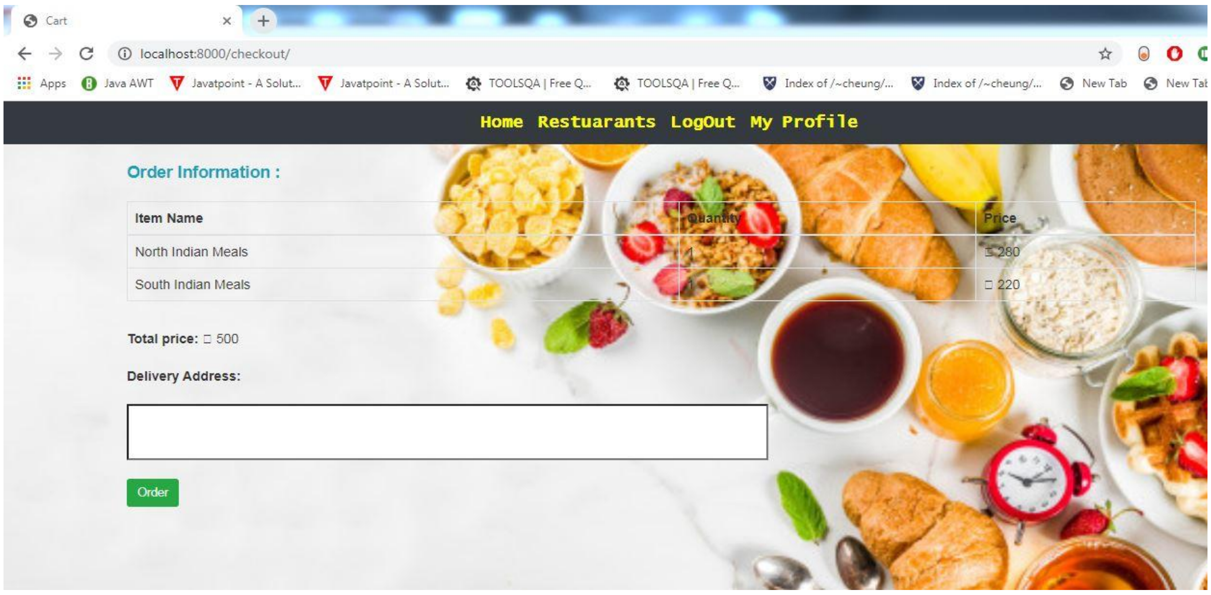
Info : world famous
Location: UVCE Bengaluru 560001
Min Order: □5

Menu

Checkout

Item Name	Price	Add
Tea	□ 10	- 0 +
Coffee	□ 8	- 0 +
Badam Milk	□ 12	- 0 +

Figure 8.7 : Selecting items



Cart
localhost:8000/checkout/
Apps Java AWT Javatpoint - A Solut... Javatpoint - A Solut... TOOLSQA | Free Q... TOOLSQA | Free Q... Index of /~cheung/... Index of /~cheung/... New Tab New Tab

Home Restuarants LogOut My Profile

Order Information :

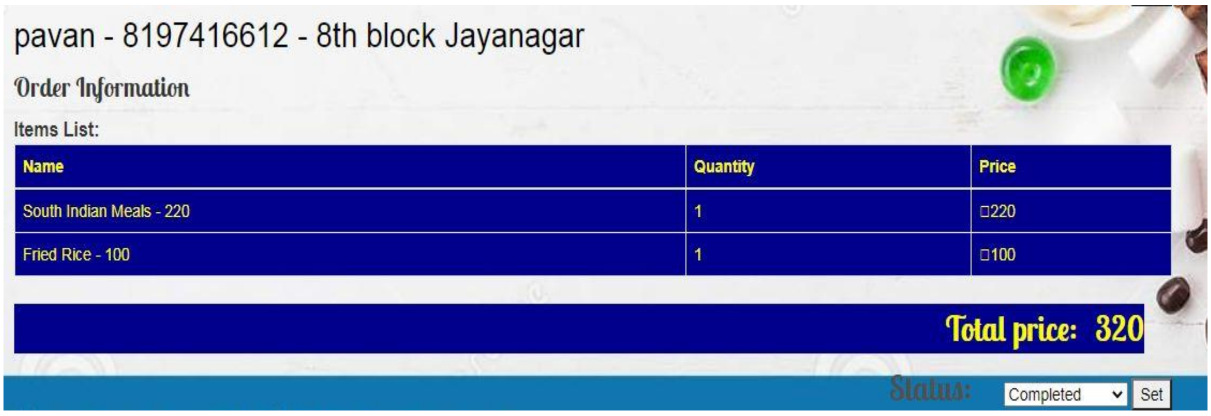
Item Name	Quantity	Price
North Indian Meals	1	□ 280
South Indian Meals	1	□ 220

Total price: □ 500

Delivery Address:

Order

Figure 8.8 : Cart



pavan - 8197416612 - 8th block Jayanagar

Order Information

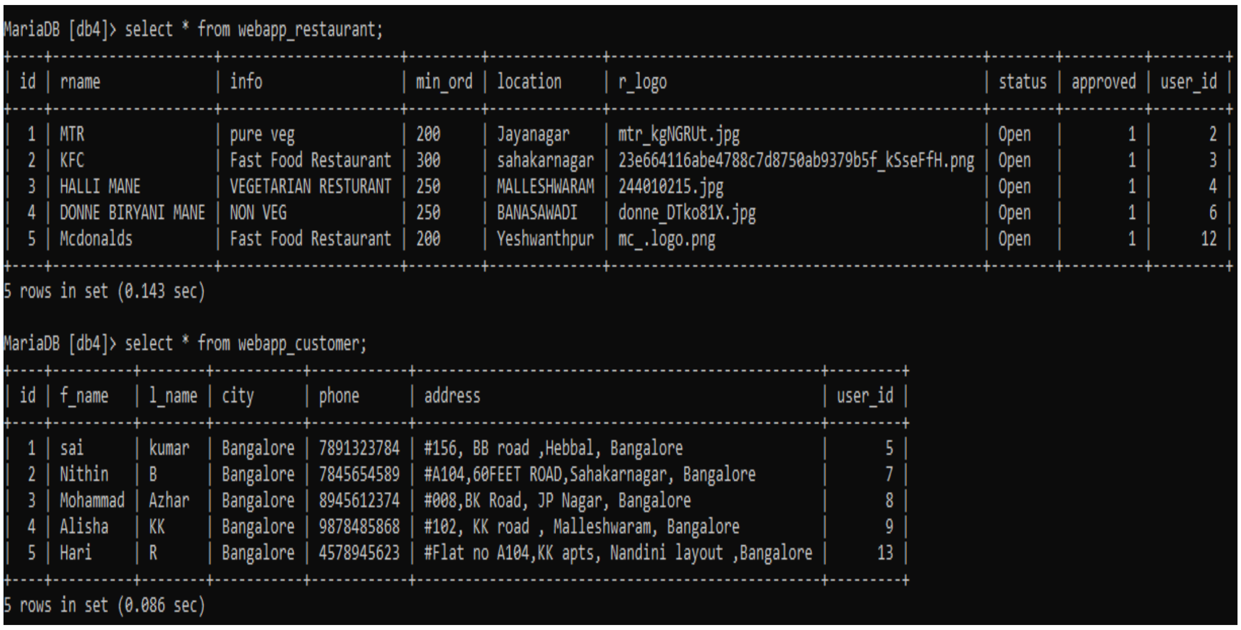
Items List:

Name	Quantity	Price
South Indian Meals - 220	1	□220
Fried Rice - 100	1	□100

Total price: 320

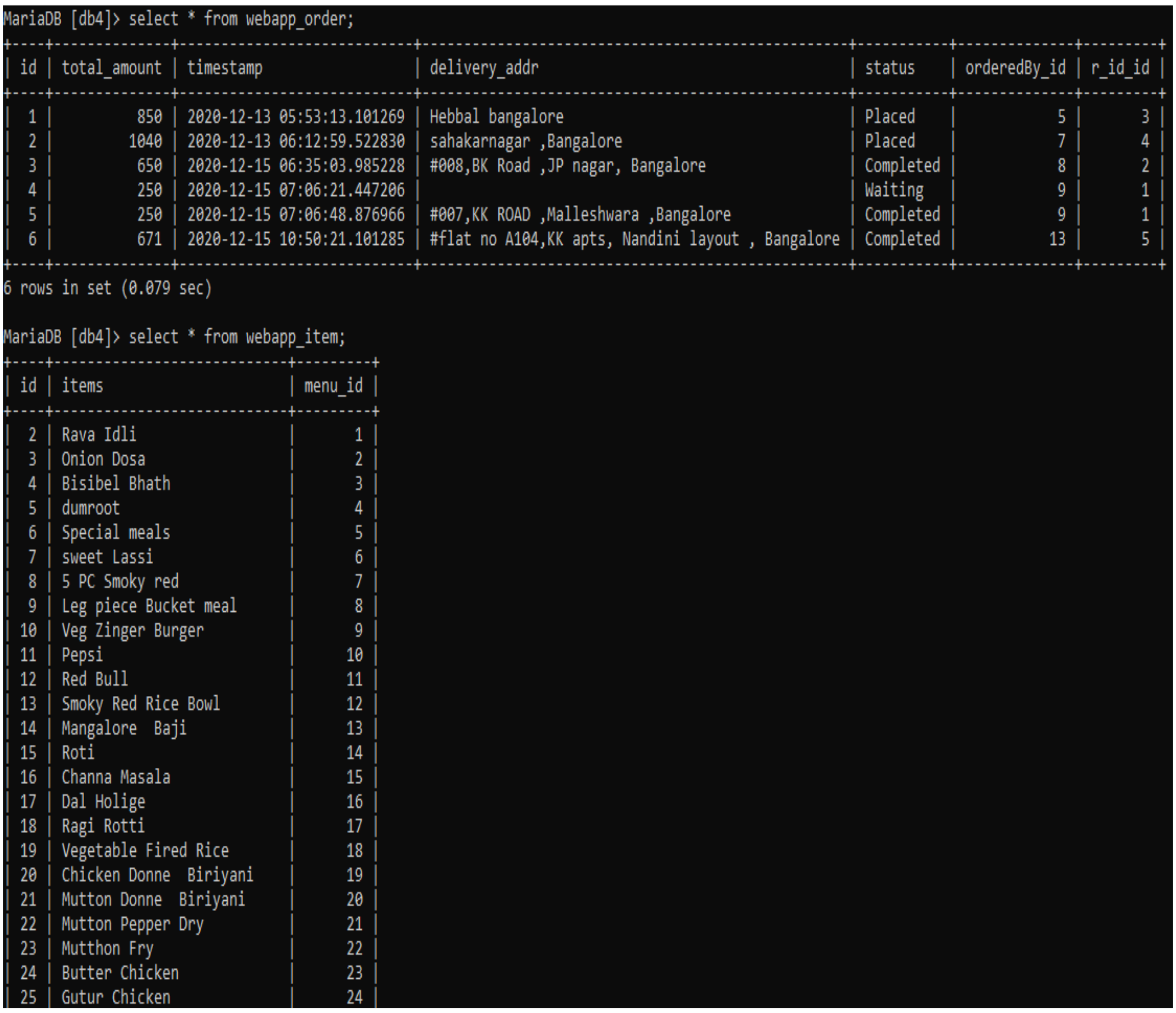
Status: Completed Set

Figure 8.9: Order Information



```
MariaDB [db4]> select * from webapp_restaurant;
+----+--------------------+----------------------+---------+--------------+------------------------------------------------+--------+----------+---------+
| id | rname              | info                 | min_ord | location     | r_logo                                         | status | approved | user_id |
+----+--------------------+----------------------+---------+--------------+------------------------------------------------+--------+----------+---------+
|  1 | MTR                | pure veg             | 200     | Jayanagar    | mtr_kgNGRUt.jpg                                | Open   |        1 |       2 |
|  2 | KFC                | Fast Food Restaurant | 300     | sahakarnagar | 23e664116abe4788c7d8750ab9379b5f_kSseFfH.png   | Open   |        1 |       3 |
|  3 | HALLI MANE         | VEGETARIAN RESTURANT | 250     | MALLESHWARAM | 244010215.jpg                                  | Open   |        1 |       4 |
|  4 | DONNE BIRYANI MANE | NON VEG              | 250     | BANASAWADI   | donne_DTko81X.jpg                              | Open   |        1 |       6 |
|  5 | Mcdonalds          | Fast Food Restaurant | 200     | Yeshwanthpur | mc_.logo.png                                   | Open   |        1 |      12 |
+----+--------------------+----------------------+---------+--------------+------------------------------------------------+--------+----------+---------+
5 rows in set (0.143 sec)

MariaDB [db4]> select * from webapp_customer;
+----+----------+--------+-----------+------------+-------------------------------------------------+---------+
| id | f_name   | l_name | city      | phone      | address                                         | user_id |
+----+----------+--------+-----------+------------+-------------------------------------------------+---------+
|  1 | sai      | kumar  | Bangalore | 7891323784 | #156, BB road ,Hebbal, Bangalore                |       5 |
|  2 | Nithin   | B      | Bangalore | 7845654589 | #A104,60FEET ROAD,Sahakarnagar, Bangalore       |       7 |
|  3 | Mohammad | Azhar  | Bangalore | 8945612374 | #008,BK Road, JP Nagar, Bangalore               |       8 |
|  4 | Alisha   | KK     | Bangalore | 9878485868 | #102, KK road , Malleshwaram, Bangalore         |       9 |
|  5 | Hari     | R      | Bangalore | 4578945623 | #Flat no A104,KK apts, Nandini layout ,Bangalore |      13 |
+----+----------+--------+-----------+------------+-------------------------------------------------+---------+
5 rows in set (0.086 sec)
```

Figure 8.10: Restaurant and Customer Tables



```
MariaDB [db4]> select * from webapp_order;
+----+--------------+----------------------------+---------------------------------------------------+-----------+--------------+---------+
| id | total_amount | timestamp                  | delivery_addr                                     | status    | orderedBy_id | r_id_id |
+----+--------------+----------------------------+---------------------------------------------------+-----------+--------------+---------+
|  1 |          850 | 2020-12-13 05:53:13.101269 | Hebbal bangalore                                  | Placed    |            5 |       3 |
|  2 |         1040 | 2020-12-13 06:12:59.522830 | sahakarnagar ,Bangalore                           | Placed    |            7 |       4 |
|  3 |          650 | 2020-12-15 06:35:03.985228 | #008,BK Road ,JP nagar, Bangalore                 | Completed |            8 |       2 |
|  4 |          250 | 2020-12-15 07:06:21.447206 |                                                   | Waiting   |            9 |       1 |
|  5 |          250 | 2020-12-15 07:06:48.876966 | #007,KK ROAD ,Malleshwara ,Bangalore              | Completed |            9 |       1 |
|  6 |          671 | 2020-12-15 10:50:21.101285 | #flat no A104,KK apts, Nandini layout , Bangalore | Completed |           13 |       5 |
+----+--------------+----------------------------+---------------------------------------------------+-----------+--------------+---------+
6 rows in set (0.079 sec)

MariaDB [db4]> select * from webapp_item;
+----+--------------------------+---------+
| id | items                    | menu_id |
+----+--------------------------+---------+
|  2 | Rava Idli                |       1 |
|  3 | Onion Dosa               |       2 |
|  4 | Bisibel Bhath            |       3 |
|  5 | dumroot                  |       4 |
|  6 | Special meals            |       5 |
|  7 | sweet Lassi              |       6 |
|  8 | 5 PC Smoky red           |       7 |
|  9 | Leg piece Bucket meal    |       8 |
| 10 | Veg Zinger Burger        |       9 |
| 11 | Pepsi                    |      10 |
| 12 | Red Bull                 |      11 |
| 13 | Smoky Red Rice Bowl      |      12 |
| 14 | Mangalore  Baji          |      13 |
| 15 | Roti                     |      14 |
| 16 | Channa Masala            |      15 |
| 17 | Dal Holige               |      16 |
| 18 | Ragi Rotti               |      17 |
| 19 | Vegetable Fired Rice     |      18 |
| 20 | Chicken Donne  Biriyani  |      19 |
| 21 | Mutton Donne  Biriyani   |      20 |
| 22 | Mutton Pepper Dry        |      21 |
| 23 | Mutthon Fry              |      22 |
| 24 | Butter Chicken           |      23 |
| 25 | Gutur Chicken            |      24 |
```

Figure 8.11: Order and Item Tables

CHAPTER 9

CONCLUSION AND FUTURE WORK

9.1 CONCLUSION

This project is carried out to analyse the internet applications on database management through the use of MySQL and the features of Django and the way it can be used and interfaced with MySQL as a standalone database application.

This project was made to make it easier for the customers to order food online instead of waiting in long queues outside the restaurants, especially during the pandemic times. So, the application is designed to promote food industry and overcome the losses occurred during the pandemic.

We were successful in separating customer and restaurant side in the website, and we also made sure that all the passwords are encrypted. We made sure that the restaurant gets the order details made for their products. Easy updation and deletion of menu makes the project useful.

A Trigger has been implemented to make the delivery look complete. We've made sure that the website looks intuitive and we hope it is understood by all sectors of the population.

9.2 FUTURE WORK

Our project has wide scope of further improvements such as addition of payment gateway, add delivery option, large scale deployment, containerization. We hope to create master and slave database structure to reduce the overload of the database queries. We also hope to add a bill printing feature.

REFERENCES

- Ramez Elmasri and Shamkant B Navathe, Fundamentals of Database Systems, Fifth Edition, AddisonWesley, 2012.
- Database System Concepts, Sixth Edition, Abraham Silberschatz, Henry F. Korth, S. Sudarshan : Tata McGraw-Hill, 2010.
- An Introduction to Database Systems by C.J. Date, A. Kannan, S. Swamynathan, 8th Edition, Pearson Education, 2006.
- Database Systems: The Complete Book, Second Edition, Hector Garcia-Molina,Jeffrey D. Ullman, Jennifer Widom, Pearson Education, 2001.
- https://www.w3schools.com/html/
- https://www.w3schools.com/css/
- https://www.tutorialspoint.com/mysql/
- https://tutorial.djangogirls.org/en/
- https://docs.djangoproject.com/en/2.2/
- http://getbootstrap.com/
- https://cs.uwaterloo.ca/~tozsu/courses/CS338/lectures/4%20Basic%20SQL.pdf
- www.cis.gsu.edu/dmcdonald/cis3730/SQL.pdf
- https://www.academia.edu/.../Ramakrishnan_Raghu.