

**A PROJECT REPORT**  
**ON**  
**CAFETERIA ORDERING SYSTEM**

Submitted in partial fulfilment of the requirement For the award of degree of

Of

BACHELOR OF COMPUTER APPLICATIONS Of  
BANGALORE UNIVERSITY

2019

**By**

**Manish Bharti**

(16NCSB7017)

Under the guidance of

**Prof. ANNIE CHRISTELA**



DEPARTMENT OF COMPUTER APPLICATIONS

ST. FRANCIS DE SALES COLLEGE

BENGALURU-560100

## DECLARATION

I, MANISH BHARTI, hereby declare that the Project Report titled **“CAFETERIA ORDERING SYSTEM”** submitted to Bangalore University, Bengaluru in partial fulfilment of the requirements for the award of the Degree of Bachelor of Computer Applications is a report of work done by me under the supervision of **Prof. ANNIE CHRISTALA** I also declare that this report any part of it has not been submitted to any other University/Institute for the award of any degree.

Signature of student

## **ACKNOWLEDGEMENT**

I would like to express my sincere gratitude and regards to my external guide Mrs. Rhicha Tiwari for her constant inspiration, supervision and invaluable guidance during the training. I would also like to thank Mrs. Jayshree of Digitage Technologies for giving me such an opportunity to continue my training in Digitage Technologies and providing the facility. At last I would also like to extend my sincere gratitude to all my faculty members and specially Mrs. Annie for giving their valuable suggestions.

Signature of Student

Sign of HOD

Sign of Internal Guide

## **ABSTRACT**

Nowadays, many restaurants manage their business by manual especially take customer ordering. . In traditional booking system,. restaurant waiter takes the customer ordering by manual system with using paper. Customer does some formal conversation like hello, hi, etc. It takes 5 to 10 minutes to book the order and waiter book the order on paper so there is probability of lost and duplication of order information. Restaurant management system puts the order in a queue with specific priority according to time and quantity, and then the cook is assigned for the specific order to complete it. Besides, the restaurant waiter information also by manual system kept use paper and this is difficult for restaurant administrator to find waiter information, probability missing the paper and difficult to arrange the schedule. The chances of committing mistakes at the restaurant side in providing a menu list for a specific time would be more. This restaurant menu and management system will replace the paper waste, is more maintainable, and allows for greater order engagement.

<b>SL.N O</b>	<b>CONTENTS</b>		<b>PAGE NO.</b>
<b>01.</b>	INTRODUCTION		
	1.1	Basic Introduction of Project	2
	1.2	Objective and Scope	2
	1.3	Tools and Technologies used	3
	1.4	Existing System	6
	1.5	Proposed Systems	7
<b>02.</b>	SYSTEM ANALYSIS		
	2.1	Preliminary Analysis & Information Gathering	8
	2.2	Objective and Scope	11
	2.3	System Requirements Specification	13
	2.4	Software Engineering Model Used	13
<b>03.</b>	SYSTEM DESIGN		

	3.1	Project Planning	17
	3.2	Modules	19
	3.3	Data Flow Diagram	20
	3.4	E-R Diagram	21
	3.5	Database Design	22
	3.6	Screen Shorts	25
<b>04.</b>	TESTING		
	4.1	Introduction to Testing	35
	4.2	Types of Testing	35
<b>05.</b>	IMPLEMENTATION & MAINTAINANCE		40
<b>06.</b>	CONCLUSION		84
<b>07.</b>	FUTURE ENHANCEMENT		86
<b>08.</b>	REFERENCES		88

# CHAPTER 1

## INTRODUCTION

### **1.1. Basic introduction of Project:**

This Cafeteria Ordering System can be used by employees in a cafeteria to handle the clients, their orders and can help them easily find free tables or place orders. The services that are provided is food ordering and reservation table management by the customer through the system online, and waiter information management, menu information management and report. The restaurant menu is organized by categories (appetizers, soups, salads, entrees, sides and drinks) of menu items. The traditional system is a cafeteria paper menu and ordering system is replaced with an electronic medium i.e. a digital tablet. Due to a digitalized system, the risk of manual errors is eliminated, thus eliminating the communication barrier. The tablet displays all the information the manager needs to know about the order he has placed. Each menu item has a name, price and associated recipe. A recipe for a menu item has a chef, preparation instruction and associated ingredients. With this system online, ordering and reservation management will become easier and systematic to replace traditional system where are still using paper. The main point of developing this system is to help restaurant administrator manage the restaurant food orders.

### **1.2. Objective and Scope:**

The main objective of the cafeteria ordering system is to provide a software that can be used by employees in a cafeteria to handle the clients, their orders and can help them easily find free tables or place orders.

#### **OBJECTIVES:**

1. This will minimize the number of employees at the back of the counter.
2. The system will help to reduce the cost of labor.
3. 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

#### **SCOPE:**



1.The interfaces of a cafeteria ordering system are made for ease of usage, an excellent experience for restaurant staff. Also, thanks to the simple to use restaurant software , it is easier to train new joiners and helps reduce training costs in terms of time and money.

## 2. Support Availability

Good cafeteria ordering systems provide you with quick support when you need them, The vendor team will have available personnel who can quickly support you in case you wish to resolve an issue or need help in troubleshooting support.

## 3. Analytics & Reporting

With the advent of data into analytics and insights reporting, restaurants are able to make much more informed decisions. Restaurants define a few key metrics and analyses traffic from there.

# 1.3 Tools and Technologies

**Python**-Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a

segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

**Django**-Django is a free and open source web application framework written in Python. A framework is nothing more than a collection of modules that make development easier. They are grouped together, and allow you to create applications or websites from an existing source, instead of from scratch.

This is how websites - even simple ones designed by a single person - can still include advanced functionality like authentication support, management and admin panels, contact forms, comment boxes, file upload support, and more. In other words, if you were creating a website from scratch you would need to develop these components yourself. By using a framework instead, these components are already built, you just need to configure them properly to match your site.

The official project site describes Django as "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." Django offers a big collection of modules which you can use in your own projects. Primarily, frameworks exist to save developers a lot of wasted time and headaches and Django is no different.

You might also be interested in learning that Django was created with front-end developers in mind. "Django's template language is designed to feel comfortable and easy-to-learn to those used to working with HTML, like designers and front-end developers. But it is also flexible and highly extensible, allowing developers to augment the template language as needed."

**SQLite 3**-SQLite is a software library that provides a relational database management system. The lite in SQLite means light weight in terms of setup, database administration, and required resource.SQLite has the following noticeable features: self-contained, serverless, zero-configuration, transactional.SQLite database is integrated with the application that accesses the database. The applications interact with the SQLite database read and write directly from the database files stored on disk.SQLite uses dynamic types for tables. It means you can store any value in any column, regardless of the data type.SQLite allows a single database connection to access multiple database files simultaneously. This brings many nice features like joining tables in different databases or copying data between databases in a single command.SQLite is capable of creating in-memory databases which are very fast to work with.

**HTML and CSS**-HTML is the language for describing the structure of Web pages.

HTML gives authors the means to:

- Publish online documents with headings, text, tables, lists, photos, etc.
- Retrieve online information via hypertext links, at the click of a button.
- Design forms for conducting transactions with remote services, for use in searching for information, making reservations, ordering products, etc.
- Include spread-sheets, video clips, sound clips, and other applications directly in their documents.

With HTML, authors describe the structure of pages using markup. The elements of the language label pieces of content such as “paragraph,” “list,” “table,” and so on.

CSS is the language for describing the presentation of Web pages, including colors, layout, and fonts. It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers. CSS is independent of HTML and can be used with any XML-based markup language. The separation of HTML from CSS makes it easier to maintain sites, share style sheets across pages, and tailor pages to different environments.

## 1.4 Existing System

- i. The existing system is manual in nature, where they do all calculation manually large volumes of paper work are done and stored which has the possibility of getting up.
- ii. Maintaining data is a tedious one and updating the records is not an easy job in the existing system.
- iii. If any information required then the user should refer the files and ledgers where the details are stored
- iv. Daily updating is too difficult. So the existing system is to be changed and put for computerization.

### **Drawbacks of existing system**

- i. The study of existing system has thrown a light on various activities involved in the organization.
- ii. Existing system is maintained manually.
- iii. The work done by the existing system is separate and tedious to maintain.
- iv. The registers are maintained separately for all the function of the concern.
- v. Reports are not easily generated whenever it is needed.
- vi. Security is less, Lack of accuracy.

## 1.5 Proposed System

- i. It has been proposed to computerize the above mentioned existing manual system.
- ii. The computerized system gives good solution for the above mentioned problems.
- iii. This proposed system has been computerized with Python-Django as the front-end tool and SQLite 3 as backend tool.
- iv. This new software package has been developed completely on menu driven basis.
- v. The various inputs given to the proposed system are order details.
- vi. The above information is maintained and can be retrieved from the proposed system.
- vii. This system has been developed to generate daily and periodical reports.

### **Aadvantages of proposed system**

- i. Data security
- ii. High processing speed
- iii. Excellent data linkage.
- iv. Use of more constrains.
- v. Duplication work is avoided.
- vi. Generations of reports are very quick.
- vii. Unlimited data size
- viii. User friendly
- ix. Time consuming

# CHAPTER 2

## SYSTEM ANALYSIS

## **SYSTEM ANALYSIS**

### **2.1 Preliminary Analysis and Information Gathering**

The first and foremost strategy for development of a project starts from the thought of designing a simple form that accepts order and product details for a small firm in which it is easy and convenient for taking order, there is a search engine report generation and also editing options for placing orders and adding products. When it is approved by the organization and our project guide, the first activity, i.e. preliminary investigation begins. The activity has three parts:

- **Request Clarification**
- **Feasibility Study**
- **Request Approval**

#### **Request Clarification**

After the approval of the request to the organization and project guide, with an investigation being considered, the project request must be examined to determine precisely what the system requires. Here our project is basically meant for management of restaurant orders. In today's busy schedule, man needs everything should be provided in a readymade manner. So taking into consideration of the vast use of net in day to day life, the corresponding development of the portal came into existence.

## **Feasibility Study**

An important outcome of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resource and time. The different feasibilities that have to be analyzed are

- i. Operational Feasibility
- ii. Economic Feasibility
- iii. Technical Feasibility

### **Operational Feasibility:**

Operational Feasibility deals with the study of prospects of the system to be developed. This system operationally eliminates all the tensions of the Admin and helps him in effectively tracking the project progress. This kind of automation will surely reduce the time and energy, which previously consumed in manual work. Based on the study, the system is proved to be operationally feasible.

### **Economic Feasibility:**

Economic Feasibility or Cost-benefit is an assessment of the economic justification for a computer based project. As hardware was installed from the beginning & for lots of purposes thus the cost on project of hardware is low. The Virtual Private Network is to be developed using the existing resources of the organization. So the project is economically feasible.

### **Technical Feasibility:**

According to Roger S. Pressman, Technical Feasibility is the assessment of the technical resources of the organization. The organization needs IBM compatible machines with a graphical web browser connected to the Internet and Intranet. The system is developed for platform Independent environment.

Python-Django, HTML, SQL server and Web Logic Server are used to develop the system. The technical feasibility has been carried out. The system is technically feasible for development and can be developed with the existing facility



## **Request Approval**

Not all request projects are desirable or feasible. Some organization receives so many project requests from client users that only few of them are pursued. However, those projects that are both feasible and desirable should be put into schedule. After a project request is approved, its cost, priority, completion time and personnel requirement is estimated and used to determine where to add it to any project list. Truly speaking, the approval of those above factors, development works can be launched

## **2.2 Input/Output**

### **Input Design**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

What data should be given as input?

How the data should be arranged or coded?

The dialog to guide the operating personnel in providing input.

Methods for preparing input validations and steps to follow when error occur

### **Objectives**

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free

from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in mire of instant. Thus the objective of input design is to create an input layout that is easy to follow.

### **Output Design**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making, Confirm an action

1. Designing computer output should proceed in an organized, well thought out manner the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements

2. Select methods for presenting information

3. Create document, report, or other formats that contain information produced by the system

The output form of an information system should accomplish one or more of the following objectives

Convey information about past activities, current status or projections of the future

Signal important events, opportunities, problems, or warnings

Trigger an action

## 2.3 System Requirement Specification:

### HARDWARE REQUIREMENTS

Processor : Intel i3 and above

System Bus : 64 bits

RAM : 4GB and above

HDD : 500GB and above

### SOFTWARE REQUIREMENTS

Operating System : Microsoft windows 10 and above

Coding Language : Django ,HTML, CSS

Front End : Python

Back End : SQLite3

## 2.4 Software Engineering Model:

### SDLC METHODOLOGIES

This document plays a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system it means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article. "A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models. As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project

The steps for Spiral Model can be generalized as follows:

The new system requirements are defined in as much details as possible. This usually

involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.

A preliminary design is created for the new system. A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product. A second prototype is evolved by a fourfold procedure: Evaluating the first prototype in terms of its strengths, weakness, and risks. Defining the requirements of the second prototype. Planning and designing the second prototype. Constructing and testing the second prototype. At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involve

development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product. The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above. The preceding steps are iterated until the customer is satisfied that the

refined prototype represents the final product desired. The final system is constructed, based on the refined prototype. The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

### SPIRAL MODEL

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. This Spiral model is combination of iterative development process model and sequential linear development model i.e, the waterfall model with a very high emphasis on risk analysis. It allows incremental releases of the product or incremental refinement through each iteration around the spiral

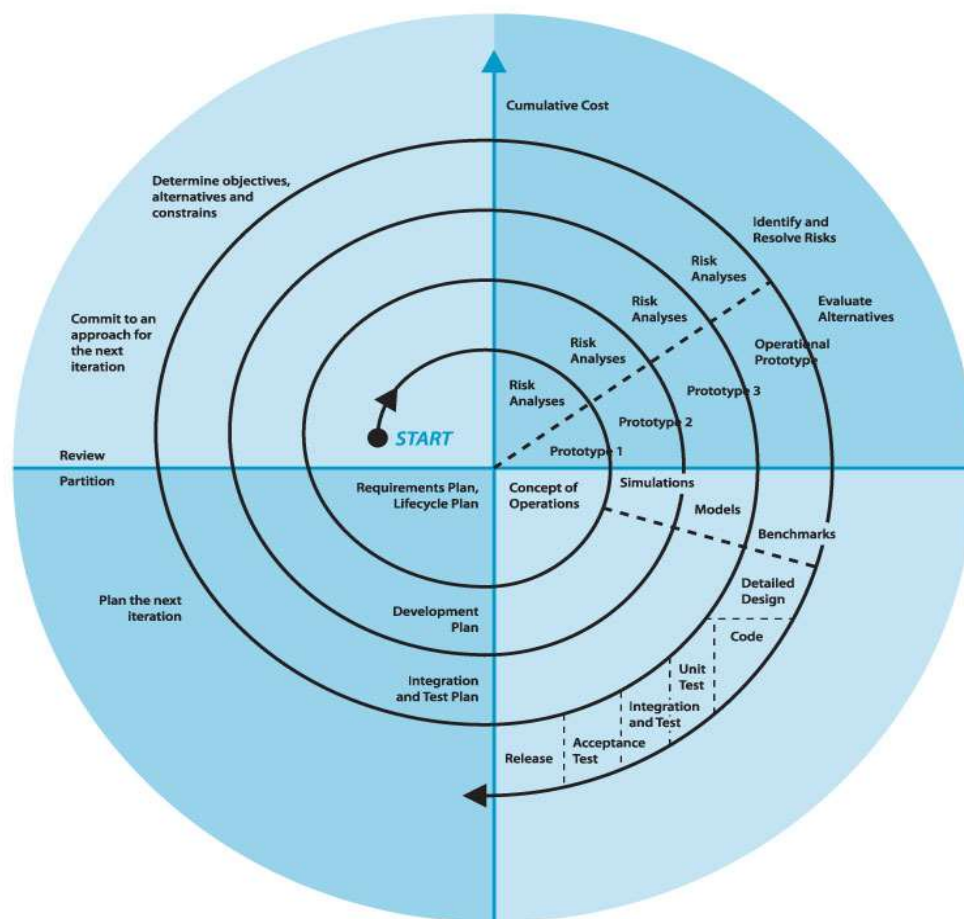
Advantages:

Estimates (i.e, budget, schedule etc.) become more realistic as work progresses, because important issues discovered earlier

It is more able to cope with the changes that are software development generally entails

Software engineers can get their hands in and start worrying on the core of a project earlier

The following diagram shows how a spiral model acts like:



# CHAPTER 3

## SYSTEM DESIGN

## SYSTEM DESIGN

### 3.1 Project Planning

Work Assigned	Due date	Description
Basic Introduction of Project,  Objective and Scope,  Tools and Technologies used,  Proposed Systems,  Limitations of Existing System,	16-01-2019	The project was chosen and analyzed to find its scope and limitations.
Preliminary Analysis & Information Gathering,  Input/Output,  System Design and Development,  System Requirements Specification,  Software Engineering Model Used	30-01-2019	Important information regarding the project was collected and requirements for the system were specified.
Project Planning,	11-02-2019	Project schedule table was drawn ,also database was collected and designed.

Modules,  Data Flow Diagram,  E-R Diagram,  Database Design,  Screen Shorts		
Introduction to Testing,  Types of Testing	27-02-2019	Testing was implemented.
Implementation & Maintainance	05-03-2019	Coding of the project got completed
Conclusion	30-03-2019	Project conclusion was decided.
Future  Enhancement	04-04-2019	Next level of the project was discussed.
References	11-04-2019	References of the project were note down.



## 3.2 Modules

After careful analysis of the system it has been identified to have the following modules:

- i. Order Management Form
- ii. Product Management Form

### **Modules Description**

#### **Order Management Form:**

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

- i. Retrieve new orders from the database.
- ii. Display the orders in an easily readable format.
- iii. It allows to update order details.

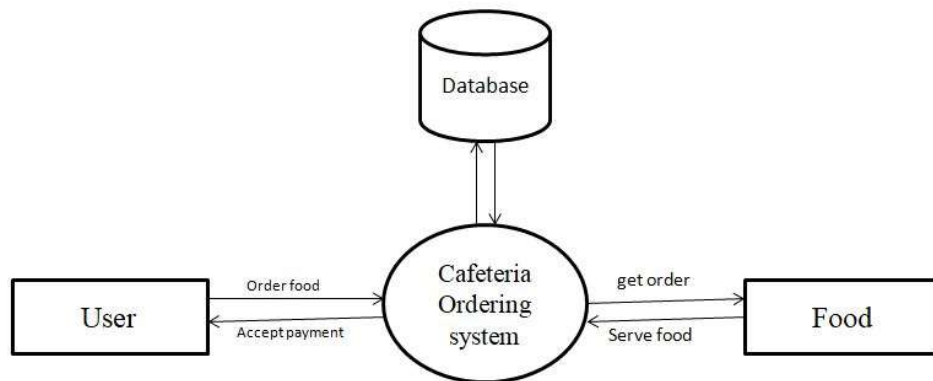
#### **Product Management Form:**

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 user interface, it will allow an Admin to manage the menu

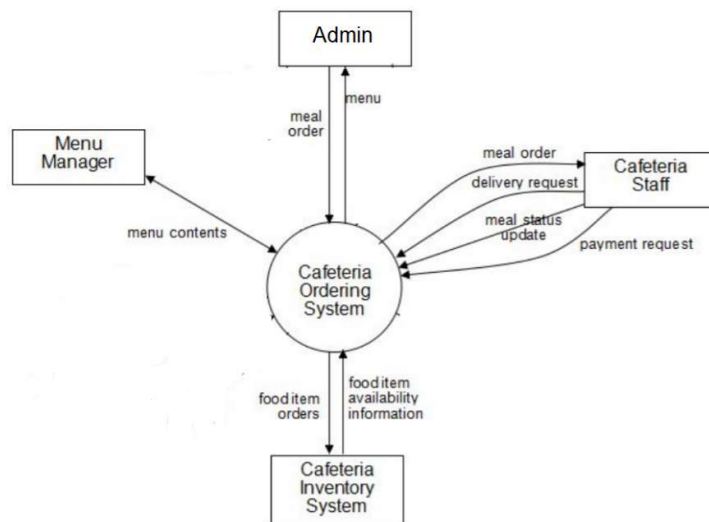
- i. Add /update/delete food item to/from the menu.
- ii. Update price for a given food item.
- iii. Update additional information (description, order status, etc.) for a given food item.

## 1.3 Data Flow Diagram

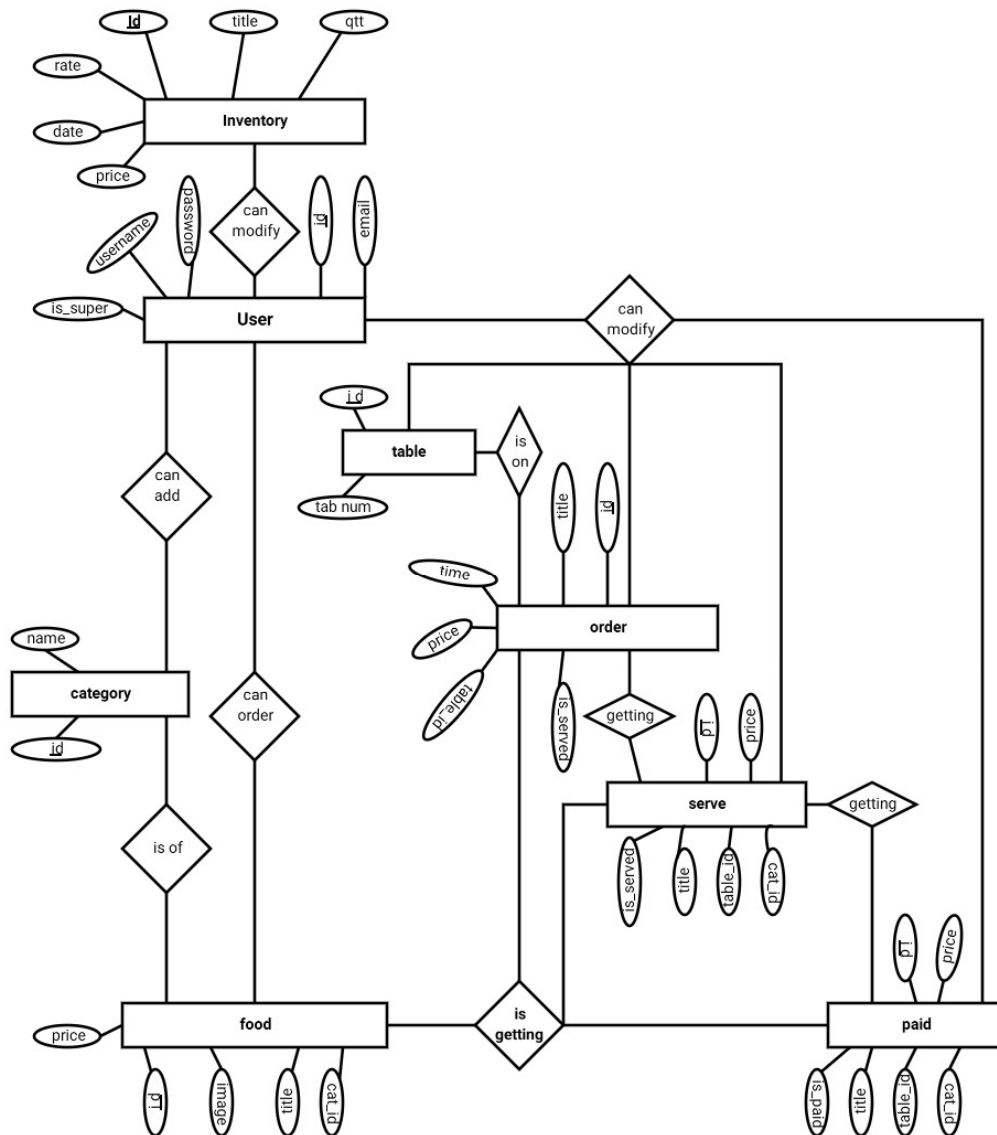
### Level 0:



### Level 1:



### 3.4 E-R Diagram



### 3.5 Database Design

#### Auth\_user

Name	Type	NN	PK	AI	U	Default	Check
id	integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
password	varchar(128)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
last_login	datetime	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
is_superuser	bool	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
first_name	varchar(30)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
last_name	varchar(30)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
email	varchar(254)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
is_staff	bool	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
is_active	bool	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
date_joined	datetime	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
username	varchar(150)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		

#### category

Name	Type	NN	PK	AI	U	Default	Check
id	integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
name	varchar(100)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

#### Public\_food

Name	Type	NN	PK	AI	U	Default	Check
id	integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
title	varchar(200)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
price	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
category_id	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
image	varchar(100)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

## Public\_order

Name	Type	NN	PK	AI	U	Default	Check
id	integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
price	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
time	datetime	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
category_id	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
table_id	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
is_served	bool	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
title	varchar(200)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

## inventory

Name	Type	NN	PK	AI	U	Default	Check
id	integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
title	varchar(200)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
qtt	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
rate	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
price	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
date	datetime	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

## Public\_order

Name	Type	NN	PK	AI	U	Default	Check
id	integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
price	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
time	datetime	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
category_id	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
table_id	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
title	varchar(200)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

## Public\_paid

Name	Type	NN	PK	AI	U	Default	Check
id	integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
price	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
time	datetime	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
is_paid	bool	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
category_id	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
table_id	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
title	varchar(200)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

## table

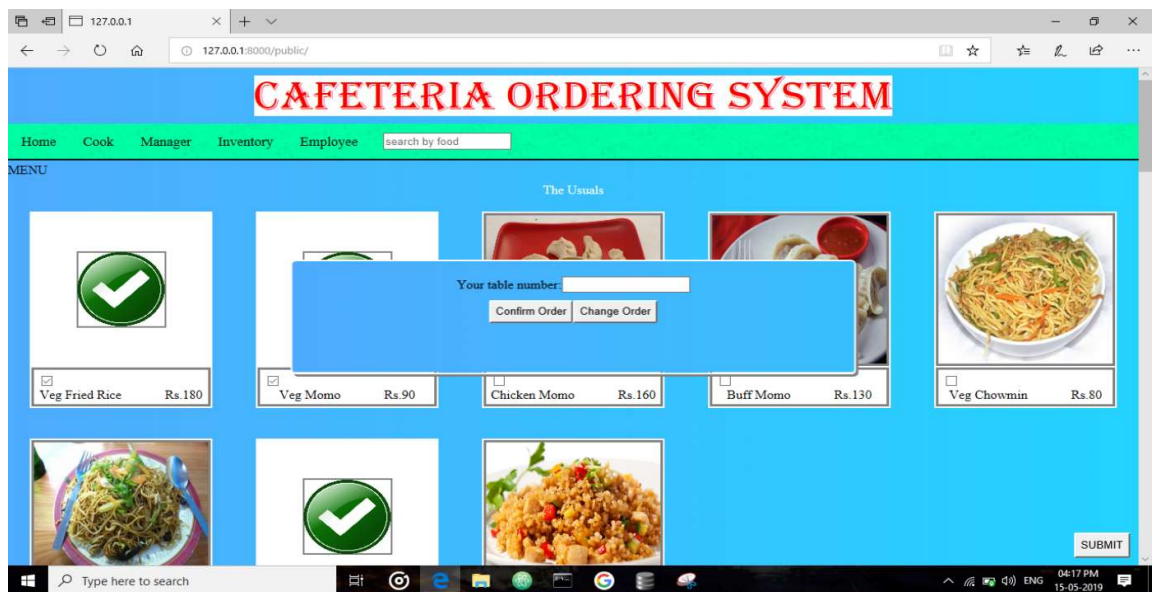
Name	Type	NN	PK	AI	U	Default	Check
id	integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
tableNum	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

### 3.6 Screenshots

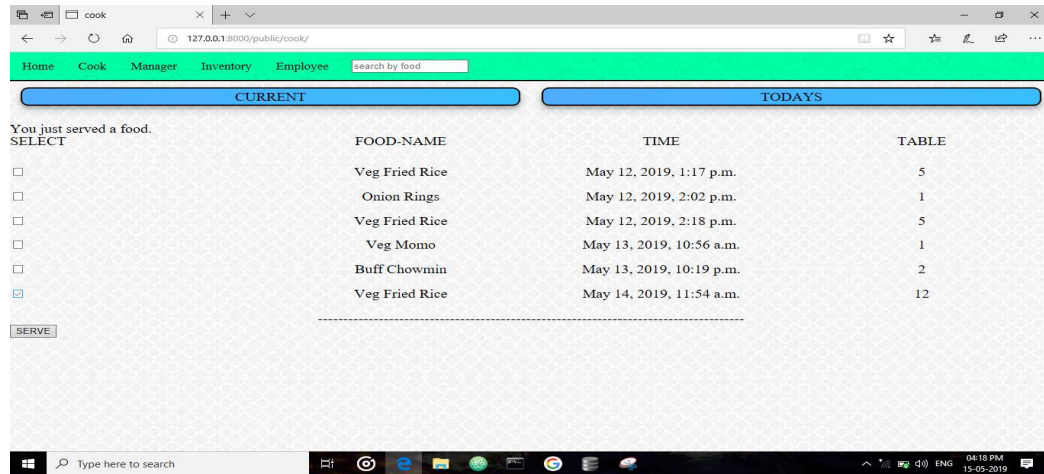
#### Home page



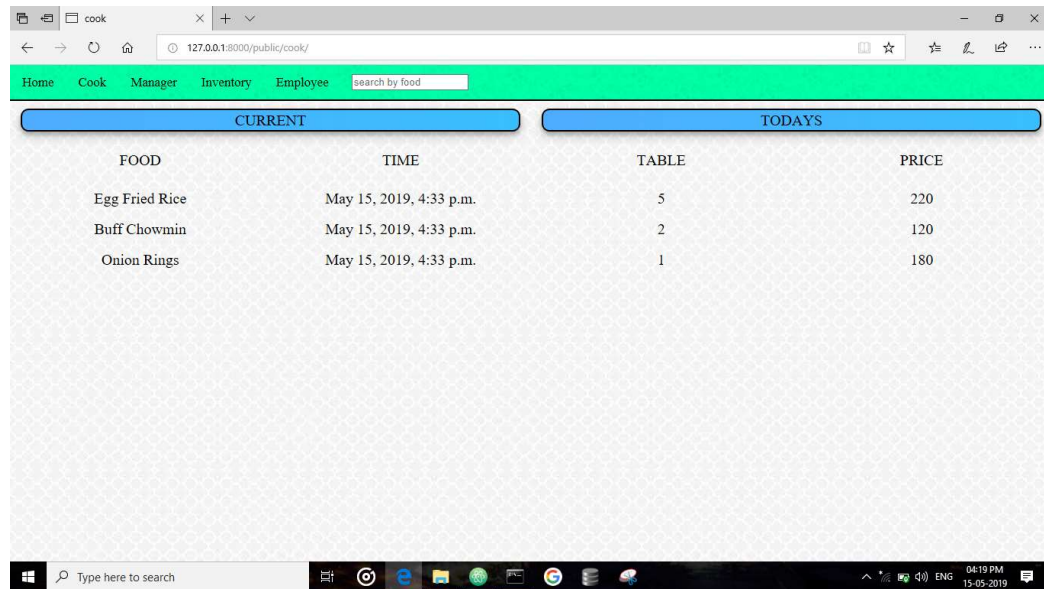
#### Home page during placing order:



## Cook serving food:



## Cook Today's served food:





## Manager Current transaction:

Home Cook Manager Inventory Employee search by food

CURRENT TRANSACTION TODAY'S TRANSACTION

You just paid for a food.  
SELECT

FOOD	TABLE	TIME	PRICE
Veg Fried Rice	2	May 11, 2019, 9:15 p.m.	180
Veg Chowmin	2	May 13, 2019, 10:18 p.m.	80
Veg Fried Rice	11	May 13, 2019, 10:18 p.m.	180
Veg Fried Rice	5	May 14, 2019, 11:04 a.m.	180
Egg Fried Rice	5	May 15, 2019, 4:33 p.m.	220
Buff Chowmin	2	May 15, 2019, 4:33 p.m.	120
Onion Rings	1	May 15, 2019, 4:33 p.m.	180

pay

http://127.0.0.1:8000/public/pay/

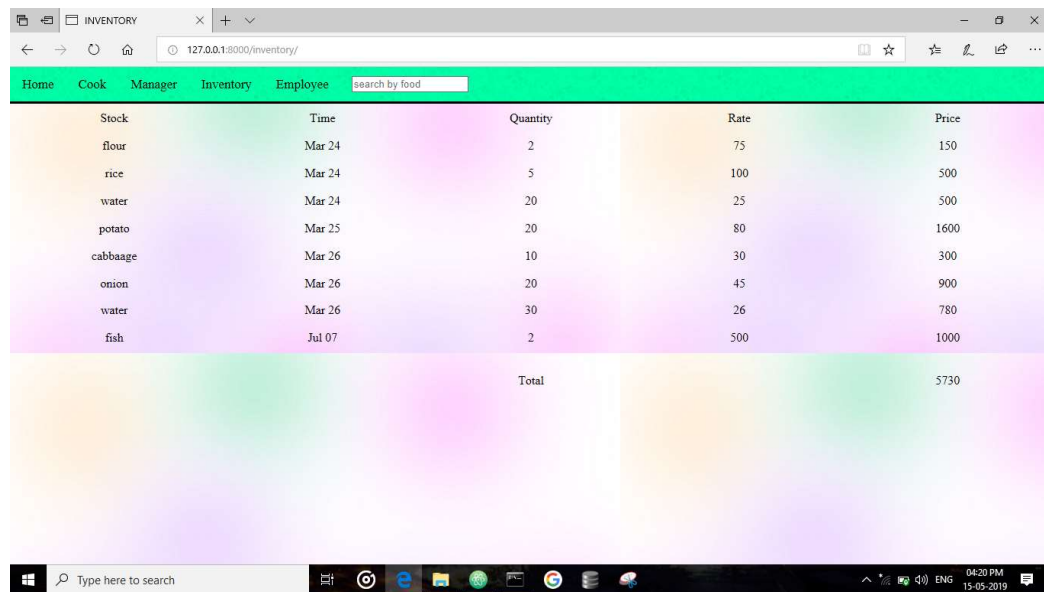
## Manager Today's transactin:

Home Cook Manager Inventory Employee search by food

CURRENT TRANSACTION TODAY'S TRANSACTION

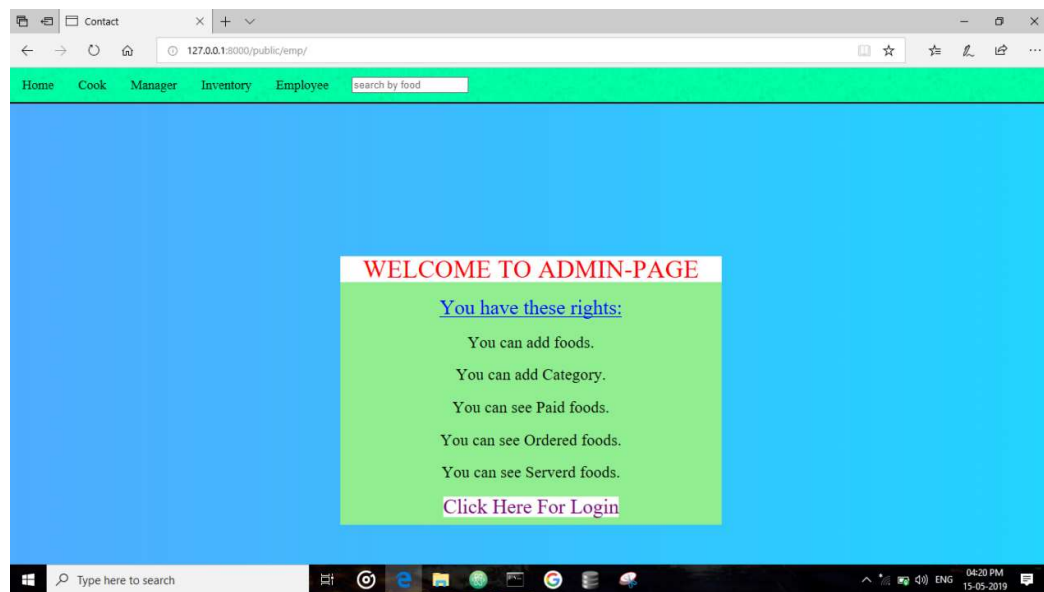
FOOD	TIME	TABLE	PRICE
Veg Momo	May 15, 2019, 4:34 p.m.	10	90
Egg Fried Rice	May 15, 2019, 4:34 p.m.	5	220
Buff Chowmin	May 15, 2019, 4:34 p.m.	2	120
Onion Rings	May 15, 2019, 4:34 p.m.	1	180
Total Transactions			10560

## Inventory:



Stock	Time	Quantity	Rate	Price
flour	Mar 24	2	75	150
rice	Mar 24	5	100	500
water	Mar 24	20	25	500
potato	Mar 25	20	80	1600
cabbaage	Mar 26	10	30	300
onion	Mar 26	20	45	900
water	Mar 26	30	26	780
fish	Jul 07	2	500	1000
Total				5730

## Employee (Admin):



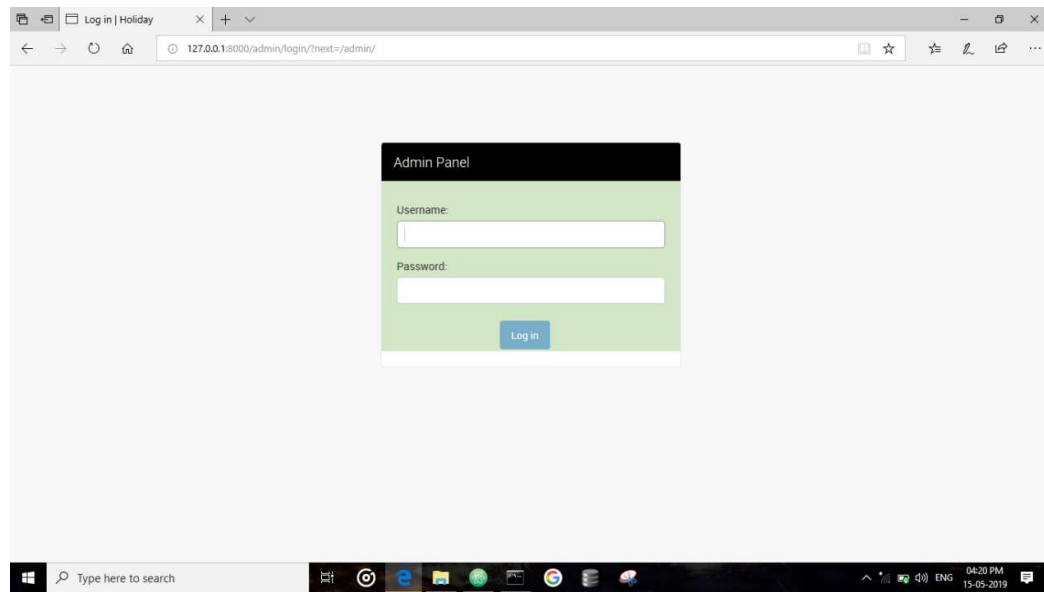
**WELCOME TO ADMIN-PAGE**

You have these rights:

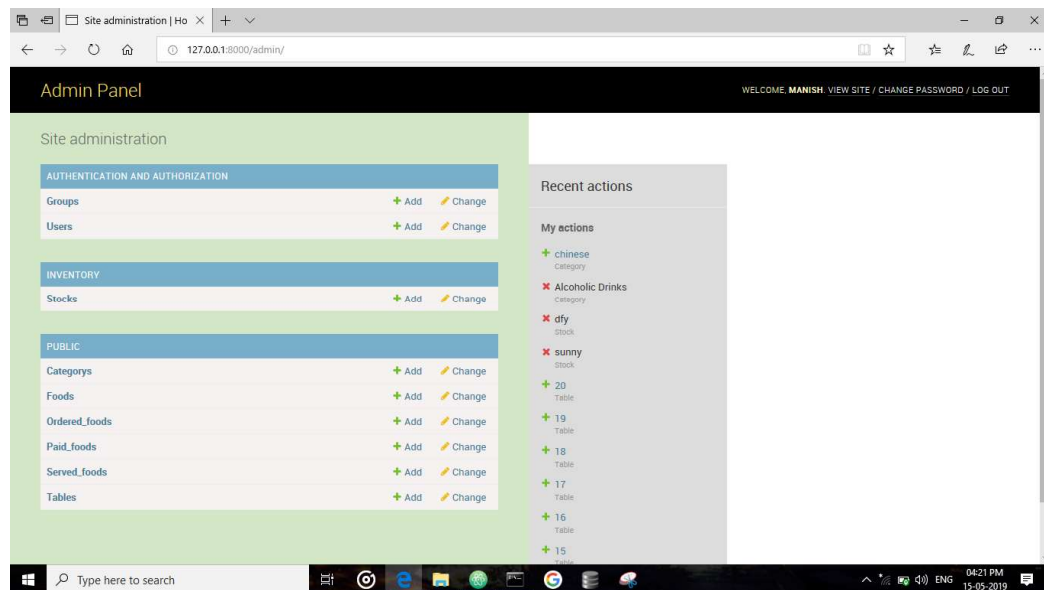
- You can add foods.
- You can add Category.
- You can see Paid foods.
- You can see Ordered foods.
- You can see Serverd foods.

[Click Here For Login](#)

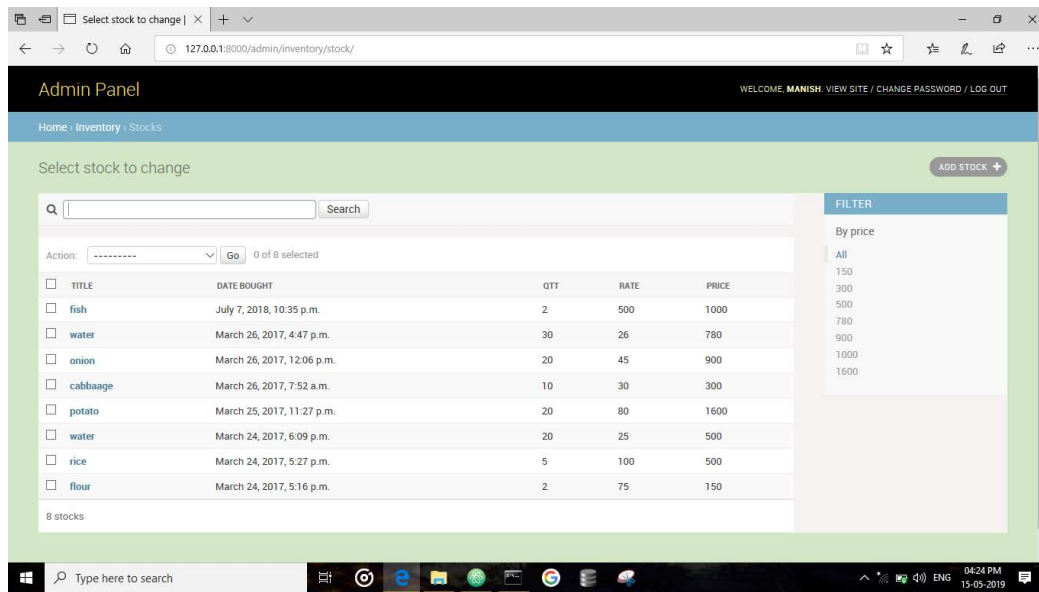
## Admin Login:



## Admin Home:



## Admin Stock form:



The screenshot shows the 'Admin Panel' interface for managing stocks. The page title is 'Select stock to change'. The breadcrumb navigation is 'Home > Inventory > Stocks'. The page includes a search bar, a filter sidebar, and a table of stock items.

**Admin Panel** WELCOME, MANISH, VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Inventory > Stocks

Select stock to change ADD STOCK +

Search

Action: ----- Go 0 of 8 selected

<input type="checkbox"/>	TITLE	DATE BOUGHT	QTT	RATE	PRICE
<input type="checkbox"/>	fish	July 7, 2018, 10:35 p.m.	2	500	1000
<input type="checkbox"/>	water	March 26, 2017, 4:47 p.m.	30	26	780
<input type="checkbox"/>	onion	March 26, 2017, 12:06 p.m.	20	45	900
<input type="checkbox"/>	cabbaage	March 26, 2017, 7:52 a.m.	10	30	300
<input type="checkbox"/>	potato	March 25, 2017, 11:27 p.m.	20	80	1600
<input type="checkbox"/>	water	March 24, 2017, 6:09 p.m.	20	25	500
<input type="checkbox"/>	rice	March 24, 2017, 5:27 p.m.	5	100	500
<input type="checkbox"/>	flour	March 24, 2017, 5:16 p.m.	2	75	150

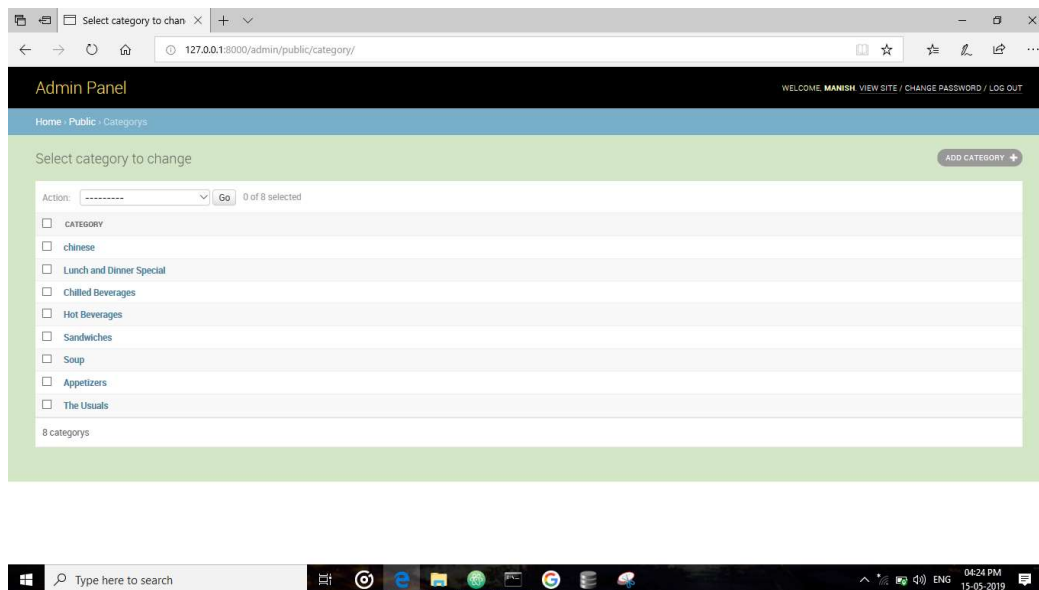
8 stocks

**FILTER**

By price

- All
- 150
- 300
- 500
- 780
- 900
- 1000
- 1600

## Admin Catageory:



The screenshot shows the 'Admin Panel' interface for managing categories. The page title is 'Select category to change'. The breadcrumb navigation is 'Home > Public > Categories'. The page includes a search bar, a filter sidebar, and a list of category items.

**Admin Panel** WELCOME, MANISH, VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Public > Categories

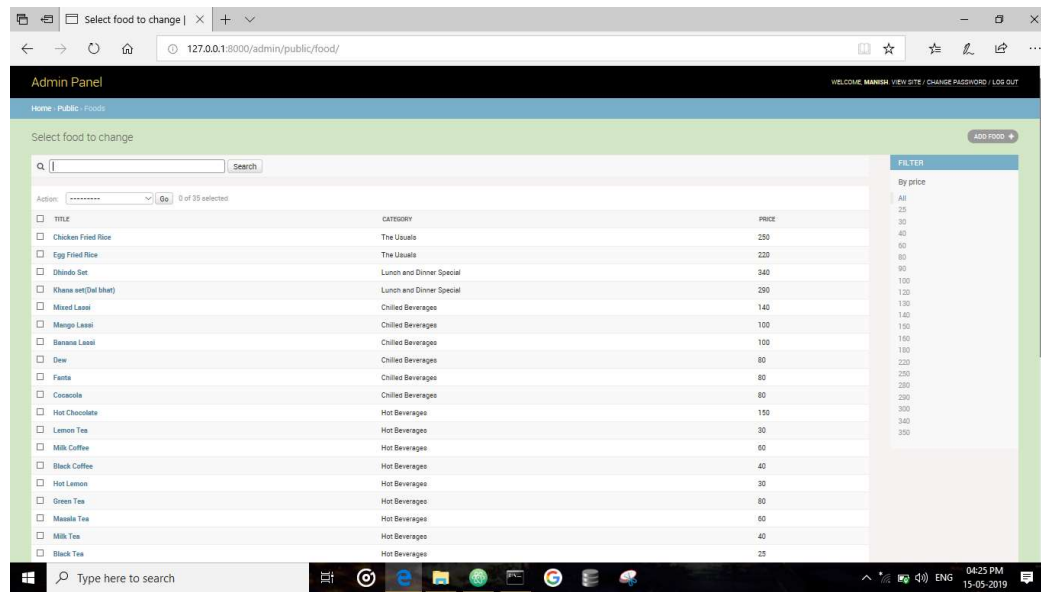
Select category to change ADD CATEGORY +

Action: ----- Go 0 of 8 selected

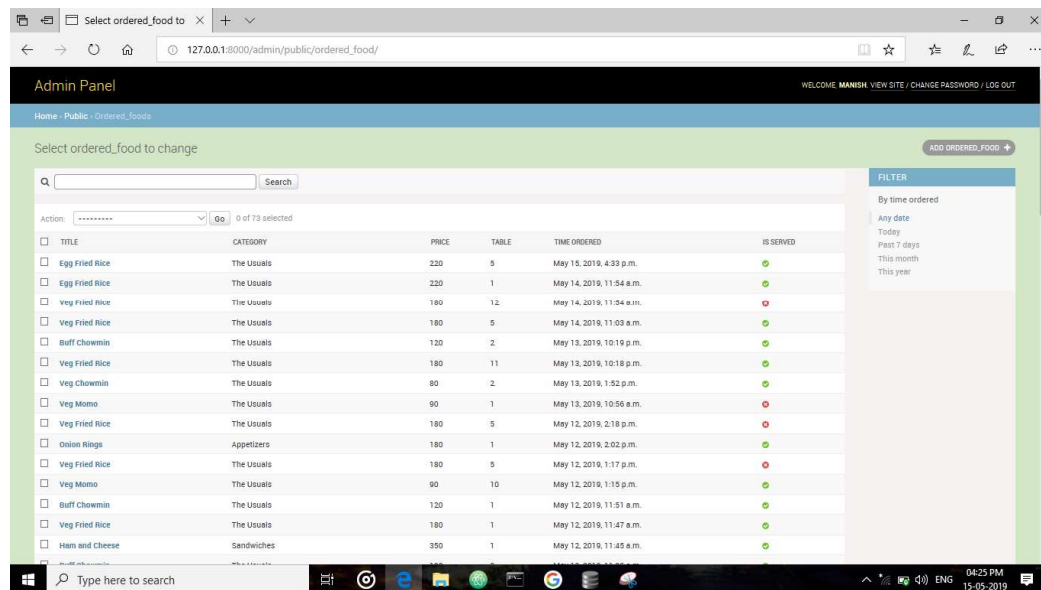
- ☐ CATEGORY
- ☐ chinese
- ☐ Lunch and Dinner Special
- ☐ Chilled Beverages
- ☐ Hot Beverages
- ☐ Sandwiches
- ☐ Soup
- ☐ Appetizers
- ☐ The Usuals

8 categories

## Admin Food:



## Admin Food order:



## Admin served food:

**Admin Panel** WELCOME MANISH VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Public > Served\_Foods

Select served\_food to change

ADD SERVED\_FOOD +

Search

Action: \*\*\*\*\* 0 of 66 selected

TITLE	CATEGORY	PRICE	TABLE	TIME SERVED	IS PAID
<input type="checkbox"/> Onion Rings	Appetizers	180	1	May 15, 2019, 4:33 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Buff Chowmin	The Usuals	120	2	May 15, 2019, 4:33 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Egg Fried Rice	The Usuals	220	5	May 15, 2019, 4:33 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Egg Fried Rice	The Usuals	220	1	May 14, 2019, 11:55 a.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Veg Fried Rice	The Usuals	180	5	May 14, 2019, 11:04 a.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Veg Fried Rice	The Usuals	180	11	May 13, 2019, 10:18 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Veg Chowmin	The Usuals	80	2	May 13, 2019, 10:18 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Veg Momo	The Usuals	90	10	May 13, 2019, 10:57 a.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Buff Chowmin	The Usuals	120	1	May 12, 2019, 2:28 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Veg Fried Rice	The Usuals	180	1	May 12, 2019, 11:48 a.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Buff Chowmin	The Usuals	120	2	May 12, 2019, 11:46 a.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Ham and Cheese	Sandwiches	350	1	May 12, 2019, 11:46 a.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Chicken Lollipop	Appetizers	280	6	May 12, 2019, 11:54 a.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Buff Chowmin	The Usuals	120	1	May 12, 2019, 11:34 a.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Veg Fried Rice	The Usuals	180	2	May 11, 2019, 9:15 p.m.	<input checked="" type="checkbox"/>

Filter

By time served

Any date

Today

Past 7 days

This month

This year

## Admin food paid:

**Admin Panel** WELCOME MANISH VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Public > Paid\_Foods

Select paid\_food to change

ADD PAID\_FOOD +

Search

Action: \*\*\*\*\* 0 of 66 selected

TITLE	CATEGORY	PRICE	TABLE	TIME PAID	IS PAID
<input type="checkbox"/> Onion Rings	Appetizers	180	1	May 15, 2019, 4:34 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Buff Chowmin	The Usuals	120	2	May 15, 2019, 4:34 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Egg Fried Rice	The Usuals	220	5	May 15, 2019, 4:34 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Veg Momo	The Usuals	90	10	May 15, 2019, 4:34 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Egg Fried Rice	The Usuals	220	1	May 14, 2019, 11:55 a.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Veg Fried Rice	The Usuals	180	2	May 13, 2019, 10:18 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Buff Chowmin	The Usuals	120	1	May 13, 2019, 10:18 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Chicken Lollipop	Appetizers	280	6	May 12, 2019, 2:48 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Buff Chowmin	The Usuals	120	1	May 12, 2019, 1 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Veg Momo	The Usuals	90	6	May 12, 2019, 1 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Ham and Cheese	Sandwiches	350	1	May 12, 2019, 12:31 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Veg Fried Rice	The Usuals	180	1	May 12, 2019, 12:31 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Buff Chowmin	The Usuals	120	2	May 12, 2019, 11:46 a.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Veg Momo	The Usuals	90	6	May 11, 2019, 1:36 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Veg Momo	The Usuals	90	5	May 11, 2019, 1:30 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Chicken Fried Rice	The Usuals	250	2	May 11, 2019, 1:30 p.m.	<input checked="" type="checkbox"/>
<input type="checkbox"/> Ham and Cheese	Sandwiches	350	5	May 11, 2019, 1:30 p.m.	<input checked="" type="checkbox"/>

Filter

By time paid

Any date

Today

Past 7 days

This month

This year

## Admin Table:



# CHAPTER 4

## TESTING



# TESTING

## 4.1 Introduction To Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies. assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

## 4.2 Types Of Testing

### Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated is the testing of individual software units of the application. It is done after the completion of an individual with integration. This structural testing that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration Unit tests unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

### Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, a shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is

specifically aimed at exposing the problems that arise from the combination of components.

### **Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements. System documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

invalid Input : identified classes of invalid input must be rejected

Functions : identified functions must be exercised

Output : identified classes of applications must be exercised.

Systems Procedures : interfacing system or procedures must be invoked

Organization and preparation of functional test is used on requirements key functions, or special test cases. In addition systematic coverage pertaining to identify Business process flows data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

### **System Testing**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows emphasizing pre-driven process links and integration points.

### **White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests must be written from a definitive source document, such as specification or requirements document such specification or requirements document is a test in which the software under test is treated as a black box you cannot “see” into it. The test provides inputs and responds to cut without considering how the software works.

**Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

- i. All field entries must work properly.
- ii. Pages must be activated from the identified link.
- iii. The entry screen, messages and responses must not be delayed.

**Features to be tested**

- i. Verify that the entries are of the correct format.
- ii. No duplicate entries should be allowed.
- iii. All links should take the user to the correct page.

**Integration Testing**

Software integration testing is the incremental Integration testing of two or more integrated software components in a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, Cafeteria Ordering System

e.g. components in a software system or one step up software applications at the company level interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

### **Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

# **CHAPTER 5**

## **IMPLEMENTATION AND MAINTENANCE**

## CODING:

### **manage.py**

```
import os

import sys

if __name__ == "__main__":

    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "rms.settings")

    try:

        from django.core.management import execute_from_command_line

    except ImportError:

        try:

            import django

        except ImportError:

            raise ImportError(

                "Couldn't import Django. Are you sure it's installed and "

                "available on your PYTHONPATH environment variable? Did you "

                "forget to activate a virtual environment?"

            )

        raise

    execute_from_command_line(sys.argv)
```

**urls.py**

```
from django.conf.urls import url, include

from django.contrib import admin

from django.conf import settings

from django.conf.urls.static import static

from django.contrib.auth.views import login, logout
```

```
urlpatterns = [

    url(r'^admin/', admin.site.urls),

    url(r'^public/', include('public.urls')),

    url(r'^inventory/', include('inventory.urls')),

    url(r'^employee/', include('employee.urls')),

    url(r'^login/$', login, name='login'),

    url(r'^logout/$', logout, name='logout'),

    url(r'^', include('public.urls')),

]
```

```
if settings.DEBUG:
```

```
    urlpatterns += static(settings.STATIC_URL,
document_root=settings.STATIC_ROOT)

    urlpatterns += static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

**settings.py**

```
"django==1.11.5"
```

```
import os
```

```
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
```

```
SECRET_KEY = 'jb3zuvfv7(9aw!#izsrn&$5ux&j3=w#ay%69x0zg=#yv=3zwm%'
```

```
DEBUG = True
```

```
ALLOWED_HOSTS = ['*']
```

```
INSTALLED_APPS = [  
    'public.apps.PublicConfig',  
    'inventory.apps.InventoryConfig',  
    'employee.apps.EmployeeConfig',  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',
```



```
]
```

```
MIDDLEWARE = [  
  
    'django.middleware.security.SecurityMiddleware',  
  
    'django.contrib.sessions.middleware.SessionMiddleware',  
  
    'django.middleware.common.CommonMiddleware',  
  
    'django.middleware.csrf.CsrfViewMiddleware',  
  
    'django.contrib.auth.middleware.AuthenticationMiddleware',  
  
    'django.contrib.auth.middleware.SessionAuthenticationMiddleware',  
  
    'django.contrib.messages.middleware.MessageMiddleware',  
  
    'django.middleware.clickjacking.XFrameOptionsMiddleware',  
  
]
```

```
ROOT_URLCONF = 'rms.urls'
```

```
TEMPLATES = [  
  
    {  
  
        'BACKEND': 'django.template.backends.django.DjangoTemplates',  
  
        'DIRS': [],  
  
        'APP_DIRS': True,  
  
        'OPTIONS': {  
  
            'context_processors': [  
  
                'django.template.context_processors.debug',  
  
                'django.template.context_processors.request',  
  
                'django.contrib.auth.context_processors.auth',
```

```
        'django.contrib.messages.context_processors.messages',

    ],

    },

    },

]
```

```
WSGI_APPLICATION = 'rms.wsgi.application'
```

```
DATABASES = {
```

```
    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),

    }

}
```

```
AUTH_PASSWORD_VALIDATORS = [
```

```
    {

        'NAME':

'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',

    },

    {

        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',

    },

    {

        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
```

```

    },

    {

        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',

    },

]

```

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'Asia/NewDelhi'

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

STATIC\_URL = '/static/'

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')

MEDIA\_URL = '/media/'

### **Public\_ Templates:**

#### **Base.html**

```
<!DOCTYPE html>
```

```
<html>
```

```
    <head>
```

```
        <meta charset="UTF-8">
```

```
        <meta name = "viewport" content = "width = device-width, initial-scale =
1">
```

```
        <title>{% block title %}RMS{% endblock %}</title>
```

```

        {% load staticfiles %}

        <link rel="stylesheet" type="text/css" href="{% static
'public/css/cssresets.css'%}" />

        <link rel="stylesheet" type="text/css" href="{% static
'public/css/style.css'%}" />


        <script type="text/javascript" src="{% static 'public/js/jquery-3.1.1.min.js'
%}" "></script>

        <script type="text/javascript" src="{% static 'public/js/home.js'
%}" "></script>

        <script type="text/javascript" src="{% static 'public/js/home-jquery.js'
%}" "></script>

        {% block header %} {% endblock %}

    </head>

    <body>

        <!-- List items for the navigation bar start here -->

        <table align="center">

            <tr><td height="10"></td></tr>

            <tr>

                <td colspan="10" style="font:bold;font-family:Algerian;font-
size:40pt;color:red;background-color:white;">

                    CAFETERIA ORDERING  SYSTEM</td></tr>

```

```

<tr><td height="10"></td></tr>

</table>

<nav style="width: 100% height:10%;">

    <ul class="topnav" id="myTopnav">

        <li><a href="{% url 'public:index' %}">Home</a></li>

        <li><a href="{% url 'public:cook' %}">Cook</a></li>

        <li><a href="{% url 'public:manager' %}">Manager</a></li>

        <li><a href="{% url 'inventory:index' %}">Inventory</a></li>

        <li><a href="{% url 'public:emp' %}">Employee</a></li>

        <li class="icon">

            <a href="javascript:void(0);" onclick="iconShow()">&#9776;</a>

        </li>

        <li id = "search-bar">

            <form role="search" method="get" action="{% url 'public:index' %}">

                <input type="text" class="form-control" placeholder="search by food"
autocomplete="on" name="q" value="{{ request.GET.q }}">

            </form>

        </li>

    </ul>

</nav>

<!-- List item for the navigation bar end here -->

```

```
<div class = "message" >

    {% if error_message %}<p><strong>{{ error_message }}</strong></p>{%
endif %}

    {% if messages %}

        {% for message in messages %}

            <p><strong>{{ message }}</strong></p>

        {% endfor %}

    {% endif %}

</div>
```

```
{% block body %}

{% endblock %}
```

```
</body>
```

```
</html>
```

### **Cook.html:**

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<meta name = "viewport" content = "width = device-width, initial-scale = 1">
```

```
<title>cook</title>
```

```
{% load staticfiles %}

<link rel="stylesheet" type="text/css" href="{% static
'public/css/cssresets.css'%}" />

<link rel="stylesheet" type="text/css" href="{% static 'public/css/cook.css'%}" />

<script type="text/javascript" src="{% static 'public/js/jquery-3.1.1.min.js'
%}"></script>
```

```
<script>
```

```
$(document).ready(function(){

    $("#todays").click(function(){

        $(".todays").show();

        $(".current").css("display","none");

        $("#todays").addClass("show-border");

        $("#current").removeClass("show-border");

    });

    $("#current").click(function(){

        $(".current").show();

        $(".todays").css("display","none");

        $("#current").addClass("show-border");

        $("#todays").removeClass("show-border");

    });

});
```

```
</script>

</head>

<body>

    <!-- List items for the navigation bar start here -->

    <nav style="width: 100%;">

        <ul class="topnav" id="myTopnav">

            <li><a href="{% url 'public:index' %}">Home</a></li>

            <li><a href="{% url 'public:cook' %}">Cook</a></li>

            <li><a href="{% url 'public:manager' %}">Manager</a></li>

            <li><a href="{% url 'inventory:index' %}">Inventory</a></li>

            <li><a href="{% url 'public:emp' %}">Employee</a></li>

            <li class="icon">

                <a href="javascript:void(0);" onclick="iconShow()">&#9776;</a>

            </li>

            <li id = "search-bar">

                <form role="search" method="get" action="{% url 'public:index' %}">

                    <input type="text" class="form-control" placeholder="search by food"
autocomplete="on" name="q" value="{{ request.GET.q }}">

                </form>

            </li>

        </ul>

    </nav>
```



```
<!-- List item for the navigation bar end here -->
```

```
<div class="headerdiv">
```

```
    <div id="current">
```

```
        CURRENT
```

```
    </div>
```

```
    <div id="todays">
```

```
        TODAYS
```

```
    </div>
```

```
</div>
```

```
<!-- code for the current section goes here -->
```

```
<div class="current">
```

```
    {% if messages %}
```

```
    {% for message in messages %}
```

```
        <strong>{{ message }}</strong>
```

```
    {% endfor %}
```

```
    {% endif %}
```

```
    <form action="{% url 'public:serve' %}" method="post">
```

```
        {% csrf_token %}
```

```
        <table>
```

```
            <tr class="table-row">
```

```

        <th align="left">Select</th>

        <th>Food-Name</th>

        <th>Time</th>

        <th>Table</th>

    </tr>

    {% for food in foods %}

    <tr class="table-row">

        <td><input type="checkbox" name="food" id="food{{
forloop.counter }}" value="{{ food.id }}" /></td>

        <td><label for="food{{ forloop.counter }}">{{ food
}}</label></td>

        <td>{{ food.time }}</td>

        <td>{{ food.table }}</td>

    </tr>

    {% endfor %}

    <tr><td colspan="4">-----
-----</td></tr>

</table>

    <input type="submit" value="SERVE"/>

    </form>

</div>

<!-- code for the current section ends here -->

```

```
<!-- code for the todays section goes here -->

<div class="todays">

    <table>

        <tr class="table-row">

            <th>Food</th>

            <th>Time</th>

            <th>Table</th>

            <th>Price</th>

        </tr>

        {% for served_food in served_foods %}

            {% if served_food.served_today == True %}

                <tr class="table-row">

                    <td>{{ served_food }}</td>

                    <td>{{ served_food.time }}</td>

                    <td>{{ served_food.table }}</td>

                    <td>{{ served_food.price }}</td>

                </tr>

            {% endif %}

        {% endfor %}

    </table>

</div>

<!-- code for the todays section ends here -->
```

```
</body>
```

```
</html>
```

**Emp.html:**

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <meta charset="UTF-8">
```

```
    <meta name = "viewport" content = "width = device-width, initial-scale = 1">
```

```
    <title>Contact</title>
```

```
    {% load staticfiles %}
```

```
    <link rel="stylesheet" type="text/css" href="{% static 'public/css/cssresets.css'%}" />
```

```
    <link rel="stylesheet" type="text/css" href="{% static 'public/css/emp.css'%}" />
```

```
    <script type="text/javascript" src="{% static 'public/js/jquery-3.1.1.min.js'%}"></script>
```

```
    <script type="text/javascript" src="{% static 'public/js/home.js'%}"></script>
```

```
    <script type="text/javascript" src="{% static 'public/js/home-jquery.js'%}"></script>
```

```
  </head>
```

```
  <body>
```

```

<!-- List items for the navigation bar start here -->

<nav style="width: 100%;">

    <ul class="topnav" id="myTopnav">

        <li><a href="{% url 'public:index' %}">Home</a></li>

        <li><a href="{% url 'public:cook' %}">Cook</a></li>

        <li><a href="{% url 'public:manager' %}">Manager</a></li>

        <li><a href="{% url 'inventory:index' %}">Inventory</a></li>

        <li><a href="{% url 'public:emp' %}">Employee</a></li>

        <li class="icon">

            <a href="javascript:void(0);" onclick="iconShow()">&#9776;</a>

        </li>

        <li id = "search-bar">

            <form role="search" method="get" action="{% url 'public:index' %}">

                <input type="text" class="form-control" placeholder="search by food"
autocomplete="on" name="q" value="{{ request.GET.q }}">

            </form>

        </li>

    </ul>

</nav>

<!-- List item for the navigation bar end here -->

<!--<form action="{% url 'public:emp' %}" method="post">

    {% csrf_token %}

```

```

    {{ form.as_p }}

    <input type="submit" value="Send" />

</form>-->

<table align="center" >

    <tr>

        <td height="200"> </td>

    </tr>

    <tr>

        <td height="10" width="500" align="center">

            <form style="background-color:lightgreen;">

                <h1 style="font:bold;font-family:#;font-size:25pt;color:red;background-
color:white;">

                    WELCOME TO ADMIN-PAGE</h1><br/>

                <ul type="circle" style="font:bold;font-family:#;font-size:16pt;">

                    <h4 style="font:#;font-family:#;font-size:20pt;color:blue;text-
decoration:underline;">You have these rights:</h4><br/>

                    <li>You can add foods.</li><br/>

                    <li>You can add Category.</li><br/>

                    <li>You can see Paid foods.</li><br/>

                    <li>You can see Ordered foods.</li><br/>

                    <li>You can see Serverd foods.</li><br/>

```

```

</ul>

<table align="center" border="5">

<tr>

<td style="font:bold;font-size:20pt;background-
color:white;border-bottom: 10px solid lightgreen;">

<a href="/admin">Click Here For Login</a> <br/>

</td>

</tr>

</table>

</form>

</td>

</tr>

</table>

<!-->

</body>

</html>

```

**Index.html:**

```
{% extends 'public/base.html' %}
```

```
{% block title %}{% endblock %}
```

```
{% block body %}
```

```
<!-- menu begins here-->
```

```
<div class = "Menu">
```

```
<h1 style="padding-top: 5px; padding-bottom: 5px;">MENU</h1>
```

```
<form name="form" action="{% url 'public:order' %}" method="post">
```

```
{% csrf_token %}
```

```
{% for category in categories %}
```

```
<h2>{{ category }}</h2>
```

```
{% for food in foods %}
```

```
{% if food.category == category %}
```

```
<div class="food-item">
```

```
<div class="image">
```

```
{% if food.image %}
```

```

```

```
{% else %}
```

```
<h3>No image to display</h3>
```

```
{% endif %}
```

```

```



```

<div class="container">

    <input name="food" type="checkbox" id="food{{
forloop.counter }}" value="{{ food.id }}" ">

    <p>{{ food }} &emsp;&emsp;&emsp;<strong>Rs. {{
food.price }}</strong></p>

</div>

</div>

</div>

{% endif %}

{% endfor %}

{% endfor %}

</div>

<!-- menu ends here -->

```

```

<!-- table selection code starts here -->

<div class="for-table" id = "table-info">

    <div class="table-info">

        <p>Your table number:</p>

        <input name="table" type="numbers" id="{{ table }}" value="{{ table
    }}"><br>

    </div>

    <div id="table-buttons">

```

```

        <button id = "confirm-order" type="submit" value="submit">Confirm
Order</button>

```

```

    </form>

```

```

        <button id = "change-order">Change Order</button>

```

```

    </div>

```

```

</div>

```

```

<!-- table selection code ends here -->

```

```

<button id = "submit" type="submit" ><h1> SUBMIT</h1> </button>

```

```

<script>

```

```

    $(document).ready(function(){

```

```

        $("#submit").click(function(){

```

```

            $("body > *").not("body > .for-table,#submit").css("z-index","-1");

```

```

            $(".for-table").css("z-index","99");

```

```

            $(".for-table").show(500).focus();

```

```

        });

```

```

        $("#change-order").click(function(){

```

```

            $(".for-table").hide(500);

```

```

            $("body > *").css("z-index","auto")

```

```

        });

```

```

        $("#confirm-order").click(function(){

```

```

        $(".for-table").hide(500);

        $("body > *").css("z-index","auto")

    });

});

</script>

{% endblock %}

```

### Manager.html:

```

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name = "viewport" content = "width = device-width, initial-scale = 1">

    <title>manager</title>


    {% load staticfiles %}

    <link rel="stylesheet" type="text/css" href="{% static
'public/css/cssresets.css'%}" />

    <link rel="stylesheet" type="text/css" href="{% static
'public/css/manager.css'%}" />

    <script type="text/javascript" src="{% static 'public/js/jquery-3.1.1.min.js'
'%}" "></script>


<script>

```

```
$(document).ready(function(){

    $("#todays").click(function(){

        $(".todays-transaction").show();

        $(".current-transaction").css("display","none");

    });

    $("#current").click(function(){

        $(".current-transaction").show();

        $(".todays-transaction").css("display","none");

    });

});

</script>

</head>

<body>

<!-- List items for the navigation bar start here -->

<nav style="width: 100%;">

    <ul class="topnav" id="myTopnav">

        <li><a href="{% url 'public:index' %}">Home</a></li>

        <li><a href="{% url 'public:cook' %}">Cook</a></li>

        <li><a href="{% url 'public:manager' %}">Manager</a></li>
```

```

<li><a href="{% url 'inventory:index' %}">Inventory</a></li>

<li><a href="{% url 'public:emp' %}">Employee</a></li>

<li class="icon">

    <a href="javascript:void(0);" onclick="iconShow()">&#9776;</a>

</li>

<li id = "search-bar">

    <form role="search" method="get" action="{% url 'public:index' %}">

        <input type="text" class="form-control" placeholder="search by food"
autocomplete="on" name="q" value="{{ request.GET.q }}">

    </form>

</li>

<!--      <ul class="navbar-right">

        <li id = "sign-up" class = "sign-up-button" ><a href="#">Sign
Up</a></li>

        <li id = "log-in" class = "log-in-button" ><a
href="#">Login</a></li>

    </ul>-->

</ul>

</nav>

<!-- List item for the navigation bar end here -->

<div class="headerdiv">

    <div id="current">

```

## CURRENT TRANSACTION

```
</div>
```

```
<div id="todays">
```

```
TODAYS TRANSACTION
```

```
</div>
```

```
</div>
```

```
<!-- code for the current section goes here -->
```

```
<div class="current-transaction">
```

```
{% if messages %}
```

```
{% for message in messages %}
```

```
<strong>{{ message }}</strong>
```

```
{% endfor %}
```

```
{% endif %}
```

```
<form action="{% url 'public:pay' %}" method="post">
```

```
{% csrf_token %}
```

```
<table>
```

```
<tr class="table-row">
```

```
<th align="left">Select</th>
```

```
<th>Food</th>
```

```
<th>Table</th>
```

```
<th>Time</th>
```

```

        <th>Price</th>

        <!--<th>Details</th>-->

    </tr>

    {% for food in served_foods %}

    <tr class="table-row">

        <td><input type="checkbox" name="food" id="food{{
forloop.counter }}" value="{{ food.id }}" /></td>

        <td><label for="food{{ forloop.counter }}">{{ food
}}</label></td>

        <td>{{ food.table }}</td>

        <td>{{ food.time }}</td>

        <td>{{ food.price }}</td>

        <!--<td><a href="">Details</td>-->

    </tr>

    {% endfor %}

    <tr><td></td></tr>

    <tr><td></td></tr>

    <tr><td></td></tr>

    <tr><td></td></tr>

    <tr><td></td></tr>

    <tr><td></td><td colspan="2">-----
-----</td></tr>

    <tr>

```

```
</td><td>

<td></td>

<td><input type="submit" value="pay"/></td>

<td></td>

</tr>

</table>

</form>

</div>

<!-- code for the current section ends here -->

<!-- code for the todays section goes here -->

<div class="todays-transaction">

    <table border="1">

        <tr class="table-row">

            <th>Food</th>

            <th>Time</th>

            <th>Table</th>

            <th>Price</th>

        </tr>

        {% for paid_food in paid_foods %}

            {% if paid_food.paid_today == True %}

                <tr class="table-row">
```



```
<td>{{ paid_food }}</td>

<td>{{ paid_food.time }}</td>

<td>{{ paid_food.table }}</td>

<td>{{ paid_food.price }}</td>

</tr>

{% endif %}

{% endfor %}

<tr><td></td></tr>

<tr><td></td></tr>

<tr><td></td></tr>

<tr><td></td></tr>

<tr><td></td></tr>

<tr>

<td></td>

<td>Total Transactions</td>

<td></td>

<td>{{ total }}</td>

</tr>

</table>

</div>

<!-- code for the todays section ends here -->

</body>
```

```
</html>
```

**Inventory\_index.html:**

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
    <meta charset="UTF-8">
```

```
    <title>INVENTORY</title>
```

```
    {% load staticfiles %}
```

```
    <link rel="stylesheet" type="text/css" href="{% static  
'inventory/css/cssresets.css'%}" />
```

```
    <link rel="stylesheet" type="text/css" href="{% static  
'inventory/css/inventory.css'%}" />
```

```
    <script type="text/javascript" src="{% static 'inventory/js/jquery-3.1.1.min.js'  
%}" "></script>
```

```
</head>
```

```
<body>
```

```
<!-- List items for the navigation bar start here -->
```

```
    <nav style="width: 100%;">
```

```
        <ul class="topnav" id="myTopnav">
```

```
            <li><a href="{% url 'public:index' %}">Home</a></li>
```

```
            <li><a href="{% url 'public:cook' %}">Cook</a></li>
```

```
            <li><a href="{% url 'public:manager' %}">Manager</a></li>
```

```
            <li><a href="{% url 'inventory:index' %}">Inventory</a></li>
```

```
            <li><a href="{% url 'public:emp' %}">Employee</a></li>
```

```

        <li class="icon">

        <a href="javascript:void(0);" onclick="iconShow()">&#9776;</a>

        </li>

        <li id = "search-bar">

        <form role="search" method="get" action="{% url 'public:index' %}">

        <input type="text" class="form-control" placeholder="search by food"
autocomplete="on" name="q" value="{{ request.GET.q }}">

        </form>

        </li>

        <!-- <ul class="navbar-right">

        <li id = "sign-up" class = "sign-up-button" ><a href="#">Sign
Up</a></li>

        <li id = "log-in" class = "log-in-button" ><a
href="#">Login</a></li>

        </ul>-->

        </ul>

        </nav>

        <!-- List item for the navigation bar end here -->

        <!-- code for the inventory section goes here -->

        <div class="inventory">

        <table>

        <tr class="table-row">

        <th>Stock</th>

```

```
<th>Time</th>

<th>Quantity</th>

<th>Rate</th>

<th>Price</th>

</tr>

{% for stock in stocks %}

<tr class="table-row">

    <td>{{ stock.title }}</td>

    <td>{{ stock.date|date:'M d' }}</td>

    <td>{{ stock.qtt }}</td>

    <td>{{ stock.rate }}</td>

    <td>{{ stock.price }}</td>

</tr>

{% endfor %}

<tr><td></td></tr>

<tr>

    <td></td>

    <td colspan=3 ><strong>Total</strong></td>

    <td><strong>{{ total }}</strong></td>

</tr>

<tr><td></td></tr>

<tr><td></td></tr>
```

```

<tr>

    <td></td>

    <td></td>

    <!--<td><button id = "add-stock">Add Stock</button></td>-->

    <td></td>

    <td></td>

</tr>

</table>

```

```

<div class="quantity-form" style="display:none">

    <form action="{% url 'inventory:index' %}" method="post">

        {% csrf_token %}

        {{ form.as_p }}

        <button id = "submit-detail">Submit</button>

        <!-- <input type="submit" value="Submit" /> -->

    </form>

</div>

<!-- code for the inventory section ends here -->

```

```

<script>

    $(document).ready(function(){

        $("#add-stock").click(function(){

```

```
        $("body > *").not("body > .quantity-form").css("opacity","0.2");

        $(".quantity-form").show(500);

    });

    $("#submit-detail").click(function(){

        $(".quantity-form").hide(500);

        $("body > *").not("body > .quantity-form").css("opacity","1");

    });

});

</script>

</body>

</html>
```

**Public\_admin.py:**

```
from django.contrib import admin

from .models import Table, Constant, Category, Food, Ordered_food, Served_food,
Paid_food


class FoodAdmin(admin.ModelAdmin):

    list_display = ('title', 'category', 'price')

    list_filter = ['price']

    search_fields = ['title']
```

```
class OrderedFoodAdmin(admin.ModelAdmin):

    list_display = ('title', 'category', 'price', 'table', 'time', 'is_served')

    list_filter = ['time']

    search_fields = ['title']
```

```
class ServedFoodAdmin(admin.ModelAdmin):

    list_display = ('title', 'category', 'price', 'table', 'time', 'is_paid')

    list_filter = ['time']

    search_fields = ['title']
```

```
class PaidFoodAdmin(admin.ModelAdmin):

    list_display = ('title', 'category', 'price', 'table', 'time')

    list_filter = ['time']
```

```
admin.site.register(Food, FoodAdmin)

admin.site.register(Ordered_food, OrderedFoodAdmin)

admin.site.register(Served_food, ServedFoodAdmin)

admin.site.register(Paid_food, PaidFoodAdmin)

admin.site.register(Category)

admin.site.register(Table)

#admin.site.register(Constant, ConstantAdmin)
```

### **Public\_models.py:**

```
import datetime
```

```
from django.db import models

from django.utils import timezone

class Table(models.Model):

    tableNum = models.IntegerField()

    def __str__(self):

        return str(self.tableNum)

class Constant(models.Model):

    serviceCharge = models.IntegerField()

    vat = models.IntegerField()

    def __str__(self):

        return str(self.vat)

class Category(models.Model):

    name = models.CharField(max_length=100)

    def __str__(self):

        return self.name


class Food(models.Model):

    title = models.CharField(max_length=200)

    category = models.ForeignKey(Category, on_delete=models.CASCADE)

    price = models.IntegerField()

    image = models.ImageField(default=False)
```



```
def __str__(self):

    return self.title

class Ordered_food(models.Model):

    title = models.CharField(default='null', max_length=200)

    category = models.ForeignKey(Category, default='null',
on_delete=models.CASCADE)

    price = models.IntegerField(default=0)

    table = models.ForeignKey(Table, default=1, on_delete=models.CASCADE)

    time = models.DateTimeField('time ordered')

    is_served = models.BooleanField(default=False)

def __str__(self):

    return self.title

def ordered_today(self):

    return self.time >= timezone.now() - datetime.timedelta(days=1)

class Served_food(models.Model):

    title = models.CharField(default='null', max_length=200)

    category = models.ForeignKey(Category,
default='null',on_delete=models.CASCADE)

    price = models.IntegerField(default=0)

    table = models.ForeignKey(Table, default=1, on_delete=models.CASCADE)

    time = models.DateTimeField('time served')
```

```
is_paid = models.BooleanField(default=False)
```

```
def __str__(self):
```

```
    return self.title
```

```
def served_today(self):
```

```
    return self.time >= timezone.now() - datetime.timedelta(days=1)
```

```
class Paid_food(models.Model):
```

```
    title = models.CharField(default='null', max_length=200)
```

```
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
```

```
    price = models.IntegerField(default=0)
```

```
    table = models.ForeignKey(Table, default=1, on_delete=models.CASCADE)
```

```
    time = models.DateTimeField('time paid')
```

```
def __str__(self):
```

```
    return self.title
```

```
def paid_today(self):
```

```
    return self.time >= timezone.now() - datetime.timedelta(days=1)
```

**public\_viwes.py:**

```
from django.shortcuts import render, get_object_or_404
```

```
from django.http import HttpResponseRedirect, HttpResponse

from django.urls import reverse

from django.contrib import messages

from django.utils import timezone

from .models import Category, Food, Ordered_food, Served_food, Paid_food, Table

from django.db.models import Q

from .forms import ContactForm


def index(request):

    foods = Food.objects.all()

    categories = Category.objects.all()

    tables = Table.objects.all()

    query = request.GET.get("q")

    if query:

        foods = foods.filter(

            Q(title__icontains=query)

        ).distinct()

        categories = Category.objects.filter(food=foods)

        return render(request, 'public/index.html', {'foods': foods, 'categories':categories,
'tables':tables})

    else:

        return render(request, 'public/index.html', {'foods': foods, 'categories':categories,
'tables':tables})
```

```
def emp(request):

    if request.method == 'POST':

        form = ContactForm(request.POST)

        if form.is_valid():

            return HttpResponseRedirect(reverse('public:index'))

    else:

        form = ContactForm()

    return render(request, 'public/emp.html', {'form': form})

    #return render(request, 'employee/main.html', {'form': form})

def cook(request):

    foods = Ordered_food.objects.filter(is_served=False)

    served_foods = Served_food.objects.all()

    return render(request, 'public/cook.html', {'foods' : foods, 'served_foods' :
served_foods})

def manager(request):

    served_foods = Served_food.objects.filter(is_paid=False)

    paid_foods = Paid_food.objects.all()

    total = 0

    for paid_food in paid_foods:

        total = total + paid_food.price
```

```
    return render(request, 'public/manager.html',{'served_foods': served_foods,
'paid_foods': paid_foods, 'total': total})
```

```
def order(request):
```

```
    foods = Food.objects.all()
```

```
    categories = Category.objects.all()
```

```
    tables = Table.objects.all()
```

```
    orderd_food = Ordered_food()
```

```
    try:
```

```
        selected_food = foods.get(pk=request.POST['food'])
```

```
        selected_table = tables.get(pk=int(request.POST['table']))
```

```
    except (KeyError, Food.DoesNotExist):
```

```
        return render(request, 'public/index.html', {
```

```
            'foods': foods,
```

```
            'categories': categories,
```

```
            'tables': tables,
```

```
            'error_message': "You didn't select a food."
```

```
        })
```

```
    else:
```

```
        orderd_food.title = selected_food.title
```

```
        orderd_food.price = selected_food.price
```

```
        orderd_food.category = selected_food.category
```

```
        orderd_food.table = selected_table
```

```
        orderd_food.time = timezone.now()
```

```
        orderd_food.save()

    messages.success(request, "You just ordered a food.")

    return HttpResponseRedirect(reverse('public:index'))

def serve(request):

    foods = Ordered_food.objects.filter(is_served=False)

    served_foods = Served_food.objects.all()

    served_food = Served_food()

    try:

        selected_food = foods.get(pk=request.POST['food'])

    except (KeyError, Food.DoesNotExist):

        return render(request, 'public/cook.html', {

            'foods': foods,

            'served_foods': served_foods,

            'error_message': "You didn't select a food."

        })

    else:

        selected_food.is_served = True

        selected_food.save()

        served_food.title = selected_food.title

        served_food.price = selected_food.price
```

```
served_food.category = selected_food.category

served_food.table = selected_food.table

served_food.time = timezone.now()

served_food.save()


messages.success(request, "You just served a food.")

return HttpResponseRedirect(reverse('public:cook'))

def pay(request):

    foods = Served_food.objects.filter(is_paid=False)

    paid_foods = Paid_food.objects.all()

    paid_food = Paid_food()

    try:

        selected_food = foods.get(pk=request.POST['food'])

    except (KeyError, Food.DoesNotExist):

        return render(request, 'public/manager.html', {

            'served_foods' : foods,

            'paid_foods' : paid_foods,

            'error_message': "You didn't select a food."

        })

    else:

        selected_food.is_paid = True

        selected_food.save()
```

```
paid_food.title = selected_food.title

paid_food.price = selected_food.price

paid_food.category = selected_food.category

paid_food.table = selected_food.table

paid_food.time = timezone.now()

paid_food.save()


messages.success(request, "You just paid for a food.")

return HttpResponseRedirect(reverse('public:manager'))
```

**cook.css:**

```
:root{

    --color-black:black;

    --color-red:red;

    --color-background:rgb(171, 147, 145);

    --color-tabs-background:rgb(171, 109, 80);

    --color-white:white;

    --color-blue:blue;

}

body{

    width: 100%;

    background-image: url("../images/background1.jpg");

    font-size: 20px;
```



```
}
```

```
*{
```

```
    box-sizing: border-box;
```

```
}
```

```
/*CODE FOR THE NAVIGATION BAR START HERE*/
```

```
nav{
```

```
    max-width: 100%;
```

```
}
```

```
/* Remove margins and padding from the list, and add a black background color */
```

```
ul.topnav {
```

```
    list-style-type: none;
```

```
    margin: 1;
```

```
    padding: 1;
```

```
    overflow: hidden;
```

```
    background-image:url("../images/background2.jpg");
```

```
    border-bottom: 2px solid black;
```

```
}
```

```
/* Float the list items side by side */
```

```
ul.topnav li {float: left;}
```

```
ul.topnav ul.navbar-right {  
  
    float: right;  
  
}
```

```
/* Style the links inside the list items */
```

```
ul.topnav li a {  
  
    display: inline-block;  
  
    color: black;  
  
    text-align: center;  
  
    padding: 14px 16px;  
  
    text-decoration: none;  
  
    transition: 0.3s;  
  
    font-size: 17px;  
  
}
```

```
ul.navbar-right li a {  
  
    display: inline-block;  
  
    color: black;  
  
    text-align: center;  
  
    padding: 14px 16px;  
  
    text-decoration: none;  
  
    transition: 0.3s;
```

```
font-size: 17px;

}
```

```
#search-bar input {

margin:12px 14px;

/*min-width: 25%;*/

}
```

```
/* Change background color of links on hover */
```

```
ul.topnav li a:hover {background-color: #555;}
```

```
/*Change the background color when the link is active*/
```

```
ul.topnav li a:hover {color: blue;}
```

```
/* Hide the list item that contains the link that should open and close the topnav on
small screens */
```

```
ul.topnav li.icon {display: none;}
```

```
/*style for the header div start here*/
```

```
div.headerdiv{

width: 100%;

margin-top: 10px;
```

```
display: block;

background-image: url("images/background.jpg");

}
```

```
#current,#todays{

width: 48%;

height:30px;

float:left;

padding:0% 5% 0% 5%;

margin-left: 1%;

margin-right: 1%;

margin-bottom: 2%;

align-items: center;

justify-content: center;

display: flex;

background-image: url("../images/background.jpg");

border: 2px solid var(--color-black);

border-radius: 10px;

box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);

}

#current:active, #todays:active{

transform: translateY(-4px);

}
```

```
/*style for the header div end here*/
```

```
/*style for the table data starts here*/
```

```
.current{
```

```
    width: 100%;
```

```
}
```

```
.current table{
```

```
    width: 100%;
```

```
}
```

```
.table-row{
```

```
    height: 40px;
```

```
}
```

```
.current table tr{
```

```
    width: 100%;
```

```
}
```

```
.current table tr th{
```

```
    text-transform: uppercase;
```

```
}
```

```
.current table td{
```

```
    width: 25%;
```

```
        text-align: center;

        padding-top: 10px;
    }

```

```
td button{

    height: 25px;

    border-radius:3px;

}

```

```
/*style for the table data ends here*/

```

```
/*style for the todays section starts here*/

```

```
.todays{

    display:none;

}

```

```
.todays table{

    width: 100%;

}

```

```
.todays table-row{

    height: 40px;

```

```
}
```

```
.todays table tr{  
  
    width: 100%;  
  
}
```

```
.todays table tr th{  
  
    text-transform: uppercase;  
  
}
```

```
.todays table td{  
  
    width: 25%;  
  
    text-align: center;  
  
    padding-top: 10px;  
  
}
```

```
td button{  
  
    height: 25px;  
  
    border-radius:3px;  
  
}
```

```
/*style for the todays section ends here*/
```

```
/*ADD ALL THE MEDIA QUERIES HERE*/
```

```
/*When the screen is less than 680 pixels wide the body width is 100%*/
```

```
@media screen and (max-width:680px) {
```

```
  body{
```

```
    max-width: 100%;
```

```
  }
```

```
}
```

```
/* When the screen is less than 680 pixels wide, hide all list items, except for the first  
one ("Home"). Show the list item that contains the link to open and close the topnav  
(li.icon) */
```

```
@media screen and (max-width:680px) {
```

```
  ul.topnav li:not(:first-child) {display: none;}
```

```
  ul.topnav ul.navbar-right {display:none;}
```

```
  ul.topnav li.icon {
```

```
    float: right;
```

```
    display: inline-block;
```

```
  }
```

```
}
```

```
/* The "responsive" class is added to the topnav with JavaScript when the user clicks  
on the icon. This class makes the topnav look good on small screens */
```

```
@media screen and (max-width:680px) {
```

```
  ul.topnav.responsive {position: relative;}
```



```
ul.topnav.responsive li.icon {  
  
    position: absolute;  
  
    right: 0;  
  
    top: 0;  
  
}  
  
ul.topnav.responsive li {  
  
    float: none;  
  
    display: inline;  
  
}  
  
ul.topnav.responsive ul.navbar-right {  
  
    float: none;  
  
    display: inline;  
  
}  
  
ul.topnav.responsive li a {  
  
    display: block;  
  
    text-align: left;  
  
}  
}
```

# **CHAPTER 6**

# **CONCLUSION**

## **Conclusion**

The project entitled “CAFETERIA ORDERING SYSTEM “ has been proposed to be implementing to replace the manual system. The developed system accomplishes all the objectives stated for the need for the change of the system. The proposed online restaurant management system is time saving and error free as compared to the traditional system. This system attracts customers and also adds the efficiency of maintaining the restaurant’s ordering and billing. Hence it is the modern way to grow up the business using Ecommerce. This system entirely reduces the unnecessary time. The outputs produced seem to satisfy all the users but it will definitely take to look forwarded for the real consequences the new system could produce. This project was made user friendly by the use of Django enabling the user to interact easily with the database. It’s also enabled the platform to serve the needs of emerging information technology trends and needs.

# CHAPTER 7

## FUTURE ENHANCEMENT

### **Future Enhancement**

The system is developed with the intrusion of future expansion of the concern. The concern can go for any future developments, as per their needs. If so they can easily develop the modules and integrated with the existing system. New menus can easily added since it is highly user friendly. The system would receive notification of orders placed by customer, customer can place order using mobile, table can be selected automatically, there may be separate user id for cooks and managers, order can be placed only if there will be stock and so on. Totally the system can be accommodated for any type modification.

# CHAPTER 8

# REFERENCES

## References

- i. RogerS.PressmanSoftwareEngineering7thedition
- ii. Using SQLite- by JayA. Kreibich
- iii. Javascript: the definitive guide- by David Flanagan
- iv. Mastering html, css & javascript- by Rafe Colburn BPB publishings
- v. Developers community for queries and doubts <https://stackoverflow.com>
- vi. Article referred <https://designmodo.com/design/>
- vii. <http://financesonline.com>
- viii. <http://www.slideshare.net>
- ix. <https://www.python.org>
- x. <https://stackoverflow.com>
- xi. <https://www.quora.com>
- xii. Django Documentation Release1.10.6.dev20170211184251Django Software FoundationFebruary11,2017