

# **AXIS BUSINESS SCHOOL**



## **PROJECT REPORT**

**ON**

## **ADVANCE FOOD ORDERING SYSTEM**

Submitted to the Department of MCA in partial fulfillment of the requirements For the Degree of

## **MASTER OF COMPUTER APPLICATION (MCA)**

**During the Academic Year 2021-2022**

**Submitted By:**

**Rohit shukla**

**ROLL NO:**

**2007220140035**

**Carried out at**



# **PROJECT REPORT**

**ON**

## **ADVANCE FOOD ORDERING SYSTEM USING Django**

A project Submitted in Partial Fulfillment of the Requirement for the Degree

### **MASTER OF COMPUTER APPLICATION**

By

**Rohit shukla**

(Roll No : 2007220140035)

Under the Supervision of

**Mr. SANJEEV KUMAR SHUKLA**



**Dr. A.P.J. Abdul Kalam Technical University (AKTU), Lucknow**

## **CERTIFICATE**

Certified that **Rohit shukla** (2007220140035) has carried out the Project work presented in this entitled “**ADVANCE FOOD ORDERING SYSTEM**” for the award of Master of Computer Application from Dr .A.P.J Abdul Kalam Technical University (APJ, AKTU), Lucknow under my supervision.

Signature:

**Mr. SANJEEV KUMAR SHUKLA**

Head of the Department

Dept. Of Master of Computer Application

Date:

# **Master of Computer Application**

## **AXIS BUSINESS SCHOOL**

### **Certificate**

Certified that the Project Report entitled “**ADVANCE FOOD ORDERING SYSTEM USING Django**” is work done by **Rohit shukla** (2007220140035) Student of MCA Final year (2021-22), in partial Fulfillment of requirements for the award of Degree of Master of Computer Application under the (Abdul Kalam Technical University (AKTU) ).

**Internal Examiner**

**Head of Department**

**External Examiner**

## **ACKNOWLEDGEMENT**

We are extremely amazed to come out with our project “**ADVANCE FOOD ORDERING SYSTEM USING Django**” under the guidance of **Mr. Sanjeev Kumar Shukla**, We are Grateful to our experienced in programming department for being a source of Inspiration and for his consistent guidance. This Web Application ‘Advance Food Ordering System Using Django’ is fully dedicated to bring Revolution in the field of **Rohit shukla** (Roll No : 2007220140035) I am committed to provide the best solution for think solution for bringing this Web app into existence.

With gratitude,

**ROHIT SHUKLA**

**MCA(2020-22)**

## **DECLARATION**

We hereby declare that this submission is our own work and that, to the best of Our Knowledge and belief, it contains no material previously published or written by another person nor material which to a Substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgement has been made in the text.

**NAME:** ROHIT SHUKLA

**SIGNATURE:**

**ROLL NO:** 2007220140035

## **Abstract**

The Online Food Ordering System described in this document has been designed to fill a specific niche in the market by providing small restaurants with the ability to offer their customers an online ordering option without having to invest large amounts of time and money in having custom web application designed specifically for them. The system, which is highly customizable, allows the restaurant employees to easily manage the site content, most importantly the menu, themselves through a very intuitive graphical interface.

The website, which is the only component seen by the restaurant customers, is then built dynamically based on the current state of the system, so any changes made are reflected in real time. Visitors to the site, once registered, are then able to easily navigate this menu, add food items to their order, and specify delivery options with only a few clicks, greatly simplifying the ordering process. Back in the restaurant, placed orders are promptly retrieved and displayed in an easily readable format for efficient processing.

The purpose of this document is to provide in-depth descriptions of design and implementation details of the system, as well as descriptions of all available functionality and plans for evolution. In addition, user manuals and trouble-shooting tips have been included for all three components to give the reader a clear idea of intended typical use cases for the system.

# Table of Contents

<b>Chapter 1: Justification &amp; Feasibility</b> .....	10
Justification .....	10
Feasibility .....	11
<b>Chapter 2: Requirements Specification</b> .....	55
System Model .....	55
Functional Requirements .....	55
The Web Ordering System .....	56
Menu Management System .....	56
Order Retrieval System .....	57
User Interface Specifications .....	57
Web Ordering System .....	58
Menu Management System .....	58
Order Retrieval System .....	58
Non-functional Requirements .....	59
System Evolution .....	59
<b>Chapter 3: System Design</b> .....	<b>Error! Bookmark not defined.</b>
System Design .....	60
Level 1: The Database & the 3 Components .....	60
Level 2: Web Ordering System Components .....	60
Level 2: Menu Management System Components .....	62
Level 2: Order Retrieval System Components .....	63
User Interface Design .....	64
Help System Design .....	64
<b>Chapter 4: Testing Design</b> .....	65
Testing .....	65
Phases .....	65
Requirements Traceability .....	66
Testing Schedule .....	68
Recording Procedures .....	68



Hardware and Web application Requirements .....	69
<b>Chapter 5: User Manual</b> .....	70
Introductory Manual .....	70
Using the Desktop Application .....	70
Using the Web Ordering System .....	74
Using the Help System.....	75
System Reference Manual.....	75
Services (Alphabetical) .....	75
Error Recovery.....	76
Installation.....	77

## **Chapter 1: Justification & Feasibility**

### **Justification**

In today's age of fast food and take-out, many restaurants have chosen to focus on quick preparation and speedy delivery of orders rather than offering a rich dining experience. Until very recently, all of these delivery orders were placed over the phone, but there are many disadvantages to this system. First, the customer must have a physical copy of the restaurant's menu to look at while placing their order and this menu must be up to date. While this expectation is not unreasonable, it is certainly inconvenient.

Second, the orders are placed using strictly oral communication, which makes it far more difficult for the customer to receive immediate feedback on the order they have placed. This often leads to confusion and incorrect orders. The current system is also inconvenient for the restaurant itself, as they must either have a dedicated staff member to answer the phone and take orders, or some employees must perform double-duty, distracting them from their regular tasks.

What I propose is an online ordering system, originally designed for use in college cafeterias, but just as applicable in any food delivery industry. The main advantage of my system is that it greatly simplifies the ordering process for both the customer and the restaurant. When the customer visits the ordering webpage, they are presented with an interactive and

up-to-date menu, complete with all available options and dynamically adjusting prices based on the selected options. After making a selection, the item is then added to their order, which the customer can review the details of at any time before checking out. This provides instant visual confirmation of what was selected and ensures that items in the order are, in fact, what was intended.

The system also greatly lightens the load on the restaurant's end, as the entire process of taking orders is automated. Once an order is placed on the webpage, it is placed into the database and then retrieved, in pretty much real-time, by a desktop application on the restaurant's end. Within this application, all items in the order are displayed, along with their corresponding options and delivery details, in a concise and easy to read manner. This allows restaurant employees to quickly go through the orders as they are placed and produce the necessary items with minimal delay and confusion.

While there are already a few systems like this in existence, all those I have encountered have been designed specifically for one restaurant, and thus cater only to their unique needs. Perhaps the greatest advantage of my system is its flexibility. The web order forms are built dynamically from the database, which can be managed using a graphical user interface. This allows the restaurant employees to not only set up and customize the system on their own, but also allows them to make changes to the menu in real time. For this reason, the exact same system can be used by numerous businesses with absolutely no modification to the code itself, which greatly increases its usefulness.

## **Feasibility**

At the present moment, the system is entirely functional, save the few minor bugs which are bound to present themselves during more extensive testing. A user is currently able to register and log in to the website and place an order. That order is then displayed, correctly and completely, in the order retrieval desktop application.

Much of what is left to do focuses not on improving functionality, but rather on improving user experience by creating richer graphical interfaces for the user to interact with and modifying the application's icons and color schemes to make them more pleasing to look at and use. For this reason, I feel that completing the project in the required timeframe is very feasible, particularly if I am able to adhere to the dates outlined in Figure 1, below.

In addition to time, a second factor influencing feasibility is resources, which also should not be a concern here. The online ordering system is structured like a fairly standard web application, and as such requires no special hardware and only basic web application, namely web and database servers, to function properly. Therefore, I anticipate finishing all of the required work on time or, ideally, ahead of schedule, leaving me with time to investigate a few additional features I would like to add but are not integral to the system.

## **Project Aim & Objectives**

The project aims to build a web-based system for restaurant, which automates food ordering system. It will also help the management to manage the online orders and view the status. The management can add menus and take orders with the system. The system also has a simple mobile-friendly user interface which can be used through different types of devices and screens. Razorpay API will be integrated with the application so customers can login with their Razorpay account and like or share menus which can work a new customer as word of mouth. In order to achieve the mentioned aim, following objectives should be achieved:

1. An extensive literature review will be conducted to find out the past work done to automate the food ordering process of restaurant.
2. Appropriate methods and methodologies will be used for designing and developing the whole system in systematic approach.
3. Django 5.4 and Bootstrap 4 will be used for developing the backend and frontend of the system.
4. Payment Media will be integrated to the system. So, customer can login and register through their payment media account and share their views about menu.
5. The system will be tested properly to ensure the quality of the system.

# **SYSTEM DEVELOPMENT ENVIRONMENT**

## **WHAT IS HTML?**

To publish information for global distribution, one needs a universal-understood language, a kind of publishing mother tongue that all computers may potentially understand. The publishing language used by the World Wide Web is HTML (Hyper Text Markup Language)

### **HTML (Gives Authors The Means To:)**

Publish online documents with headings, text, tables, list, photos etc.

1. Retrieve online information via hypertext links, at the click of a button
2. Design forms for conducting transactions with remote services, for use in searching information, making reservation, ordering products etc.;
3. Includes spreadsheets, video clips, sound clips, and other applications directly in the documents.
4. This particular feature ensures that the form is not submitted if an input box with required attribute is empty. The required attribute can also be used as a selector to style the element.
5. For verifying any particular textbox, we can insert regular expression directly into an element. We can check a textbox value against the regular expression specified in the pattern attribute.
6. The default value is always read-only and read-write value of -webkit-user-modify allows the user to edit the content.

### **Some HTML Tags**

<HTML>	: Starting an HTML tag
<HEAD>	: Creating a web page's head
<TITLE>	: Giving a web page's body
</HEAD>	: Ending a web pages head
</BODY>	: Ending a web pages body
</HTML>	: Ending a web page
<FORM>	: Creating a HTML forms
<INPUT TYPE=BUTTON>	: Creating a buttons
<INPUT TYPE=CHECKBOX>	: Creating a checkboxes
<INPUT TYPE=SUBMIT>	: Creating a submit button
<INPUT TYPE=TEXT>	: Creating a text fields

### **HTML 4.0**

HTML 4.0 extends with mechanisms for style sheets, scripting, frames embedding objects, improved support for right to left and mixed direction texts, richer tables and enhancements to form, offering improved accessibilities for people with disability

## WHAT IS CSS?

CSS stands for Cascading Style Sheets, it is fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.

CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

## Advantages of CSS

CSS saves time – You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.

Pages load faster – If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.

Easy maintenance – To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.

Superior styles to HTML – CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.

Multiple Device Compatibility – Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.

Global web standards – Now HTML attributes are being deprecated and it is being recommended to use CSS. So its a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

## WHAT IS JAVA SCRIPT?

JavaScript, originally supported by Netscape Navigator, is the most popular Web scripting language today. JavaScript lets you embed programs right in your Web pages and run these programs using the Web browser. You place these programs in a <SCRIPT> element. If you want the script to write directly to the Web page, place it in the <BODY> element.

EX: <HTML>

<HEAD>

<TITLE></TITLE>

```
</HEAD>

<BODY>

  <SCRIPT LANGUAGE="JavaScript">

  </SCRIPT>

</BODY></HTML>
```

## JAVASCRIPTS OBJECTS

JavaScript is an object-oriented language. JavaScript comes with a number of predefined objects.

### Objects of the JavaScript

1. Document: Corresponds to the current Web page's body. Using this object, you have access to the HTML of the page itself, including the all links, images and anchors in it.
2. Form: Holds information about HTML forms in the current page.
3. Frame: Refers to a frame in the browser's window.
4. History: Holds the records of sites the Web browser has visited before reaching the current page.
5. Location: Holds information about the location of the current web page.
6. Navigator: Refers to the browser itself, letting you determine what browser the user has.
7. Window: Refers to the current browser window.

## JAVASCRIPTS EVENTS

Some of the events of JavaScript

1. on Change: Occurs when data in a control, like a text field, changes.
2. on Click: Occurs when an element is clicked.
3. on Focus: Occurs when an element gets the focus.
4. on Mouse Down: Occurs when a mouse button goes down.
5. on Reset: Occurs when the user clicks the reset button.

## JAVASCRIPTS FUNCTIONS

Declaration of function:

*Syntax*: function function name ()

```
{  
  
...  
  
...  
  
}
```

Write these functions in <SCRIPT> tag.

## What is Bootstrap?

Bootstrap is a free and open source front end development framework for the creation of websites and web apps. The Bootstrap framework is built on HTML, CSS, and JavaScript (JS) to facilitate the development of responsive, mobile-first sites and apps.

Responsive design makes it possible for a web page or app to detect the visitor's screen size and orientation and automatically adapt the display accordingly; the mobile first approach assumes that smartphones, tablets and task-specific Mobile apps are employees' primary tools for getting work done and addresses the requirements of those technologies in design.

Bootstrap includes user interface components, layouts and JS tools along with the framework for implementation. The software is available precompiled or as source code.

Mark Otto and Jacob Thornton developed Bootstrap at Twitter as a means of improving the consistency of tools used on the site and reducing maintenance. The software was formerly known as Twitter Blueprint and is sometimes referred to as Twitter Bootstrap.

In computers, the word bootstrap means to boot: to load a program into a computer using a much smaller initial program to load in the desired program (which is usually an operating system).

In the physical world, a bootstrap is a small strap or loop at the back of a leather boot that enables you to pull the entire boot on and in general usage, bootstrapping is the leveraging of a small initial effort into something larger and more significant. There is also a common expression, "pulling yourself up by your own bootstraps," meaning to leverage yourself to success from a small beginning.

## Bootstrap Advantages:

- ❖ Fewer Cross browser bugs
- ❖ A consistent framework that supports major of all browsers and CSS compatibility fixes
- ❖ Lightweight and customizable
- ❖ Responsive structures and styles
- ❖ Several JavaScript plugins using the jQuery

- ❖ Good documentation and community support
- ❖ Loads of free and professional templates, WordPress themes and plugins
- ❖ Great grid system

## What is 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. . I have used python language in project.

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.

## Python Advantages

- ❖ Easy to Read, Learn and Write
- ❖ Improved Productivity
- ❖ Interpreted Language
- ❖ Dynamically Typed
- ❖ Free and Open-Source
- ❖ Vast Libraries Support
- ❖ Portability

## What is Django?

Django is a free and open source web application framework, written in Python. A web framework is a set of components that helps you to develop websites faster and easier.



When you're building a website, you always need a similar set of components: a way to handle user authentication (signing up, signing in, signing out), a management panel for your website, forms, a way to upload files, etc.

Luckily for you, other people long ago noticed that web developers face similar problems when building a new site, so they teamed up and created frameworks (Django being one of them) that give you ready-made components to use. . I have used Django in project.

Frameworks exist to save you from having to reinvent the wheel and to help alleviate some of the overhead when you're building a new site. To understand what Django is actually for, we need to take a closer look at the servers. The first thing is that the server needs to know that you want it to serve you a web page.

Imagine a mailbox (port) which is monitored for incoming letters (requests). This is done by a web server. The web server reads the letter and then sends a response with a webpage. But when you want to send something, you need to have some content. And Django is something that helps you create the content.

## **Django Advantages**

- ❖ Implemented in Python
- ❖ Better CDN connectivity and Content Management
- ❖ Batteries Included Framework
- ❖ Fast Processing
- ❖ Offers Rapid-development
- ❖ Scalable
- ❖ Security

## **What is Razorpay?**

Razorpay is a payment mode founded by the group of IIT Roorkee alumni. It's a kind of payment mode that combines more than one banking system. It is an online payment system that can access all your credit, debit cards, UPI, and popular mobile wallets. It makes nuisance-free payments in India for others. In this payment system, different banks and wallets get connected to it to make payments easier for everyone. . I have used Rozorpay in project.

Razorpay is the in-between mode of payment which transact the payments to the other banks which are connected and available in all kinds of banks, credit, debit, wallets (Netflix, Airtel, Swiggy, and others).

Razorpay subscription also brings with it the useful feature of UPI autopay, nowadays transactions via cards are decreasing day by day due to digital payments. It helps in registered business also

where payments get easier for others from different apps. After the pandemic and big break from the direct interaction, people have started to get more comfortable in digital payments where all the business has got a lot of upstream.

This platform of digital payment makes a link between many other apps in which one of the levels is crossed by Razorpay. Apart from the payment gateway, it provides more other benefits to the customers like Razorpay provides where it provides the merchant to automate bank transfer, invoices, and the account for the payroll.

Razorpay capital provides a loan for the business to avoid your cash flow. It benefits all the small business and cashier flow for the people. It helps businesses to make their market place and make available working capital rolls.

What is VsCode?

Visual Studio Code is a code editor in layman's terms. Visual Studio Code is "a free-editor that helps the programmer write code, helps in debugging and corrects the code using the intelli-sense method ". In normal terms, it facilitates users to write the code in an easy manner. Many people say that it is half of an IDE and an editor, but the decision is up to to the coders. Any program/software that we see or use works on the code that runs in the background. Traditionally coding was used to do in the traditional editors or even in the basic editors like notepad! These editors used to provide basic support to the coders. I have used VSCode in project.

Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging.

First and foremost, it is an editor that gets out of your way. The delightfully frictionless edit-build-debug cycle means less time fiddling with your environment, and more time executing on your ideas.

## **What is ReportLab?**

The ReportLab Toolkit is an Open Source Python library for generating PDFs and graphics. or ReportLab is the time-proven, ultra-robust open-source engine for creating complex, data-driven PDF documents and custom vector graphics. It's free, open-source , and written in Python. A charts and widgets library for creating reusable data graphics.

## **What is Data?**

Data is a collection of a distinct small unit of information. It can be used in a variety of forms like text, numbers, media, bytes, etc. it can be stored in pieces of paper or electronic memory, etc.

Word 'Data' is originated from the word 'datum' that means 'single piece of information.' It is plural of the word datum.

In computing, Data is information that can be translated into a form for efficient movement and processing. Data is interchangeable.

## What is Database?

A database is an organized collection of data, so that it can be easily accessed and managed. You can organize data into tables, rows, columns, and index it to make it easier to find relevant information. Database handlers create a database in such a way that only one set of software program provides access of data to all the users. The main purpose of the database is to operate a large amount of information by storing, retrieving, and managing data.

There are many dynamic websites on the World Wide Web nowadays which are handled through databases. For example, a model that checks the availability of rooms in a hotel. It is an example of a dynamic website that uses a database.

There are many databases available like MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server, etc.

Modern databases are managed by the database management system (DBMS).

SQL or Structured Query Language is used to operate on the data stored in a database. SQL depends on relational algebra and tuple relational calculus.

A cylindrical structure is used to display the image of a database.

## What is SQLite ?

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures. These features make SQLite a popular choice as an Application File Format. SQLite database files are a recommended storage format by the US Library of Congress. Think of SQLite not as a replacement for Oracle but as a replacement for fopen(). I have used SQLite Database in project.

SQLite is a compact library. With all features enabled, the library size can be less than 750KiB, depending on the target platform and compiler optimization settings. (64-bit code is larger. And some compiler optimizations such as aggressive function inlining and loop unrolling can cause the object code to be much larger.) There is a tradeoff between memory usage and speed. SQLite generally runs faster the more memory you give it. Nevertheless, performance is usually quite good even in low-memory environments. Depending on how it is used, SQLite can be faster than direct filesystem I/O.

## **WaterFall Model**

The waterfall model is a well-known structured methodology for web application development. The whole process of system development is divided into distinct phases. The model has been introduced in 1970s. Every phase has a unique output. It was the first SDLC model to be used widely. So that, sometime it is referred to Waterfall by SDLC. The waterfall model is used when the system requirements are well known, technology is understood and the system is a new version of an existing product (Dennis, Wixom and Roth, 2012).

Mainly there are six phases in Waterfall model. If there is a problem faced in any phase of the cycle, the system goes to the previous phase. The phases of Waterfall method are:

Requirements Gathering & Analysis: In this phase, all possible requirements of the system are captured and documented in a requirement specification doc.

System Design: The requirements documented in previous phase are studied in this phase and the system design is prepared.

Implementation: With inputs from system design, the system is developed in several unites. Then the units are tested.

Integration & Testing: The units of the program developed in previous phase are integrated into a system. Then the whole system is tested.

Deployment of the System: When the all kind of testing is done, the product is deployed in the customer environment.

Maintenance: There are some issues which are found in the client environment. Patches are released to fix those issues.

## **Requirements Elicitation**

The requirements of a system is characteristics of a system it needs to have. The requirements has been collected in the planning phase of the SDLC. Different kinds of data collection methods have been utilised to obtain the requirements of the system, which are explained in Chapter 3.1.2.

## **Functional Requirements**

According to International Institute of Business Analysis (IIBA), functional requirements are “the product capabilities, or things that the product must do for its users” (Dennis, Wixom and Roth, 2012). Following are the functional requirements of the project:

- The application must have user registration and login option.

- The application must have registration and sign in option with Razorpay API.
- The Application must have a shopping cart for ordering foods.
- The application must have admin registration and login system.
- The application must have password recovery system with email address for users and admins.
- The application must have menu add and edit options for admin.

## **Non-Functional Requirements**

International Institute of Business Analysis (IIBA) defines non-functional requirements as “the quality attributes , design, and implementation constraints, and external interfaces which a product must have” (Dennis, Wixom and Roth, 2012). Following are the non-functional requirements of the project.

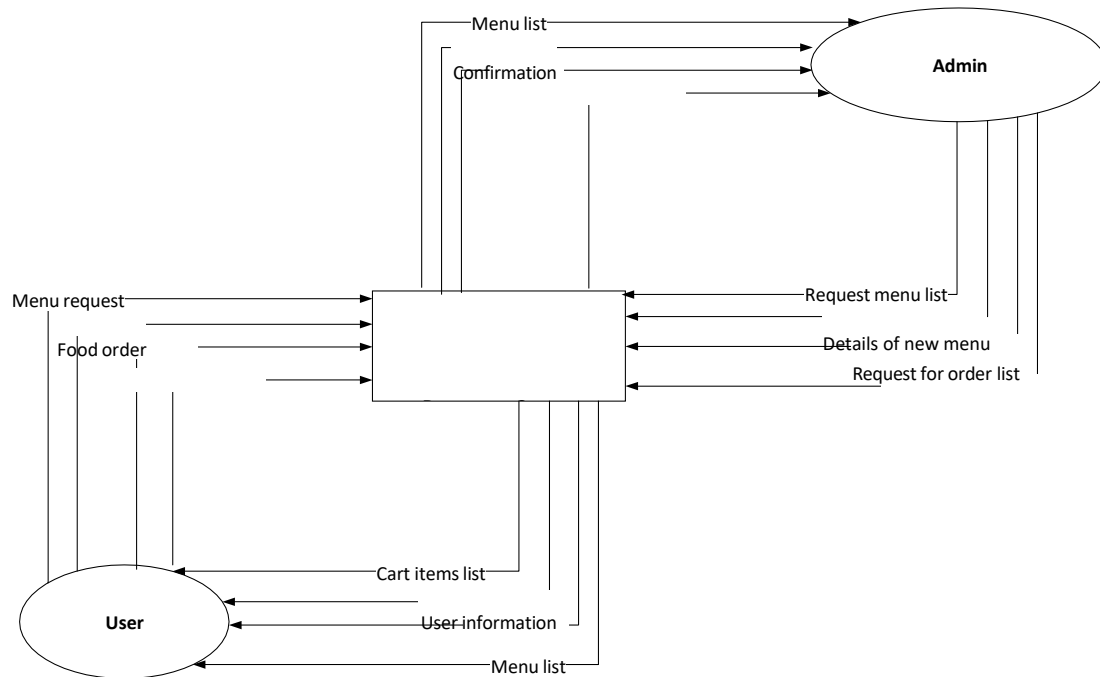
- The application must have a user interface.
- The user interface must be mobile-friendly.
- Exception handling methods must be used.
- Users should get confirmation and warning message.

## **Process Modelling**

Process modelling is used in a project to depict the processes of data in an application. The restaurant application will be developed and implemented using Model-View-Controller (MVC) design pattern. This processes are mostly implemented as business logic in application controllers. There are different tools for process modelling in SSADM. Context diagram and data flow diagram will be used to model the processes of the system.

## **Context Diagram (CD)**

Context diagrams define “how the business process or computer system interacts with its environment” (Dennis, Wixom and Roth, 2012). Context diagrams are used early in a project to describe the entities of the system. It shows the external entities and data flows into and out of the system. The processes and data stores are not shown in context diagram. It will be shown in data flow diagram.



The above context diagram shows the processes of two entities, User and Admin, with the restaurant system. Both of the entities have four processes with the system.

## Data Flow Diagram (DFD)

This is a data flow diagram (DFD), which shows the process of ordering food through online restaurant system. The oval shape means an entity or user. The boxes in the middle of the diagram are process box. The process box shows the number of process and its description. The boxes in the right side of diagram are data store.

The box represents when a piece of data is stored. The between the boxes and oval shape shows the flow of data. The main purpose of a DFD is to show the flow of data while doing a process. Figure 6 shows the DFD of ordering foods process for users

### Data flow diagram levels

Data flow diagrams are also categorized by level. Starting with the most basic, level 0, DFDs get increasingly complex as the level increases. As you build your own data flow diagram, you will need to decide which level your diagram will be.

**Level 0 DFDs**, also known as context diagrams, are the most basic data flow diagrams. They provide a broad view that is easily digestible but offers little detail. Level 0 data flow diagrams show a single process node and its connections to external entities.

**Level 1 DFDs** are still a general overview, but they go into more detail than a context diagram. In a level 1 data flow diagram, the single process node from the context diagram is broken down into subprocesses. As these processes are added, the diagram will need additional data flows and data stores to link them together.

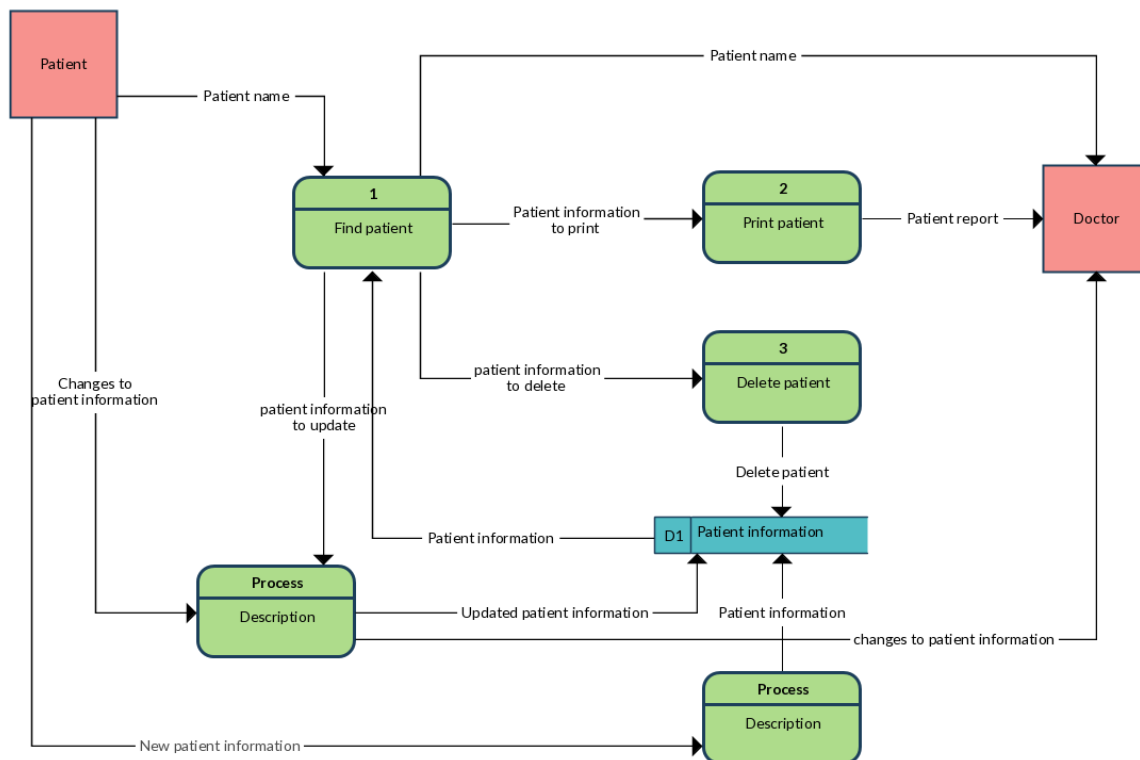
**Level 2+ DFDs** simply break processes down into more detailed subprocesses. In theory, DFDs could go beyond level 3, but they rarely do. Level 3 data flow diagrams are detailed

enough that it doesn't usually make sense to break them down further.

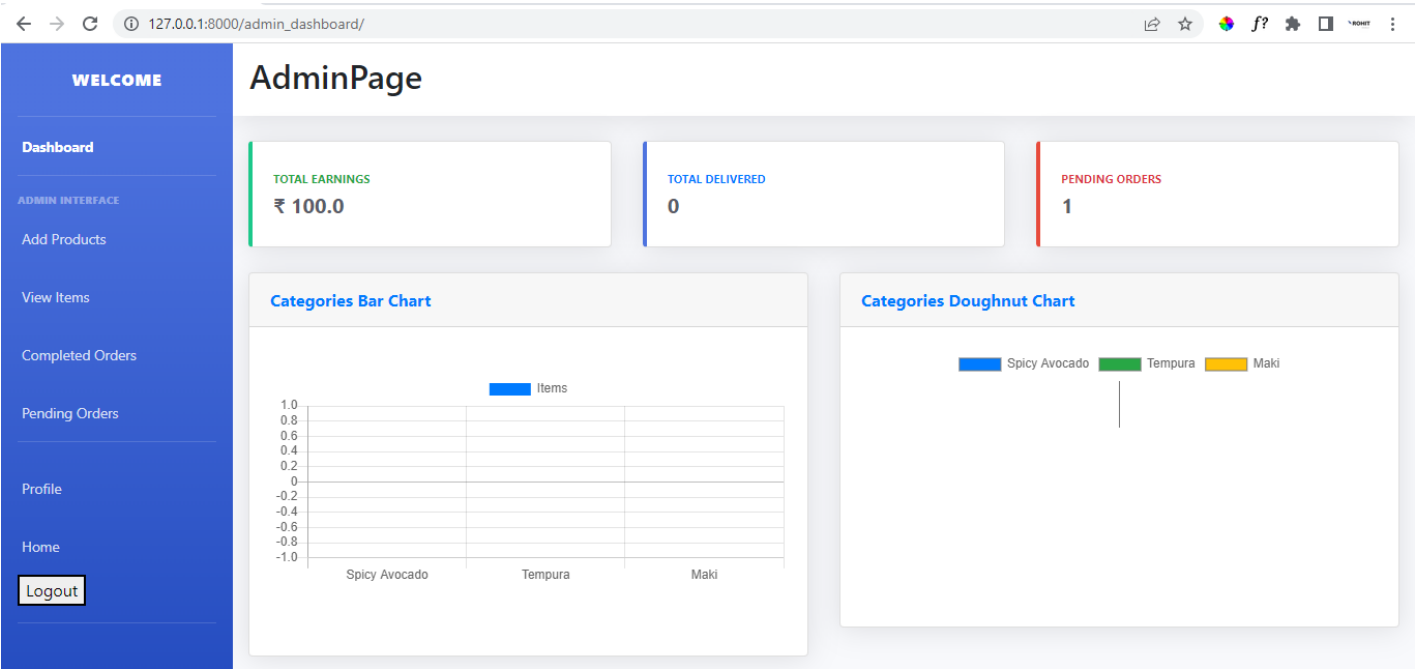
## Why DFD?

DFD graphically representing the functions, or processes, which capture, manipulate, store, and distribute data between a system and its environment and between components of a system. The visual representation makes it a good communication tool between User and System designer. Structure of DFD allows starting from a broad overview and expand it to a hierarchy of detailed diagrams. DFD has often been used due to the following reasons:

- Logical information flow of the system
- Determination of physical system construction requirements
- Simplicity of notation
- Establishment of manual and automated systems requirements



# ADMIN DASHBOARD



← → ↻ ⓘ 127.0.0.1:8000/item\_list/

WELCOME

Dashboard

ADMIN INTERFACE

Add Products

View Items

Completed Orders

Pending Orders

Profile

Home

Logout

AdminPage

Item No.	Title	Decription	Instructions	Price	Pieces		
10	Potbelly A Wreck Sandwich	Potbelly's signature sandwich, A Wreck, is also its most popular one. this is a true carnivore's delight	Jain Option Available	150.0	0 pieces	Update	Delete
11	Jack in the Box Tacos	A Report said the chain sells more than 554 million of items per year. making tacos the chain's best-selling item.	Jain Option Available	180.0	1 pieces	Update	Delete
12	Wendy's Dave's Single	According to a Wendy's rep, Dave's Single is a best-seller. This burger truly hits the spot when you have a fast-food craving.	Jain Option Available	100.0	1 pieces	Update	Delete
17	Zaxby's Chicken Finger Plate	It is available in both Original and Buffalo Chicken—it's listed as one of its most-ordered items on the company's website.	Jain Option Available	250.0	1 pieces	Update	Delete
19	Subway	it's turkey sub is most popular sandwich, according to a rep for chain. Specifically, sub with a bag of chips on the side.	Cash on: Option Available	240.0	1 pieces	Update	Delete
20	Chicken Honey	Church's Chicken has two clear menu stars: the	Cash on: Option	210.0	1	Update	Delete



WELCOME

## AdminPage

Dashboard

ADMIN INTERFACE

Add Products

View Items

Completed Orders

Pending Orders

Profile

Home

Logout

### Add Item

Title\*

Image\*

Choose File No file chosen

Description

Price\*

Pieces\*

6

Instructions\*

Jain Option Available

WELCOME

## AdminPage

Dashboard

ADMIN INTERFACE

Add Products

View Items

Completed Orders

Pending Orders

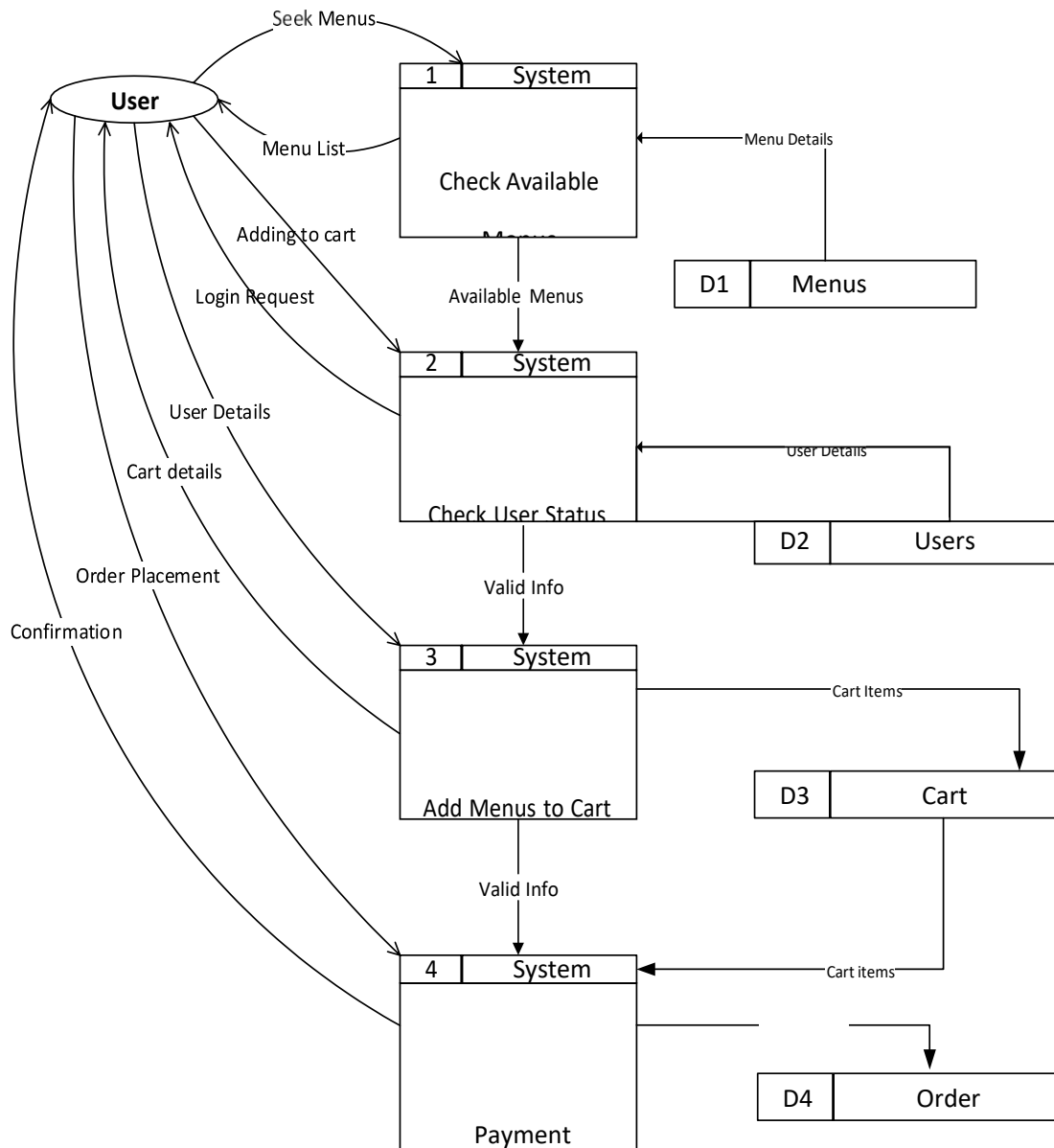
Profile

Home

Logout

### Completed Orders

Order No.	User	User Email	Name	Description	Specification	Price	Total Pieces	Status	Delivery Date
35	rohitshukla001	rawstar1090@gmail.com	Wendy's Dave's Single	According to a Wendy's rep, Dave's Single is a best-seller. This burger truly hits the spot when you have a fast-food craving.	Jain Option Available	100.0	1 pieces	Delivered	May 21, 2022



Process 1 shows that whenever a visitor request the menus page, the system makes a query in the menu table of the database and acquires available menus and returns the menu list to visitor. If visitor likes to add any menu the menu list (shopping cart), s/he hits "Add to Menu" button. Then in Process 2, the systems checks if the user is logged in. If the user is not logged in, the system requests user to log in by redirecting him/her to the log in page. After getting user's credential, the system compares the credential against the user database. In Process 3, if the user credential matched with the credentials saved in User table

in database, the system adds the menu to Cart table and redirects to the cart page.  
In Process 4, if user orders the menu, the order details saved in the Order table.

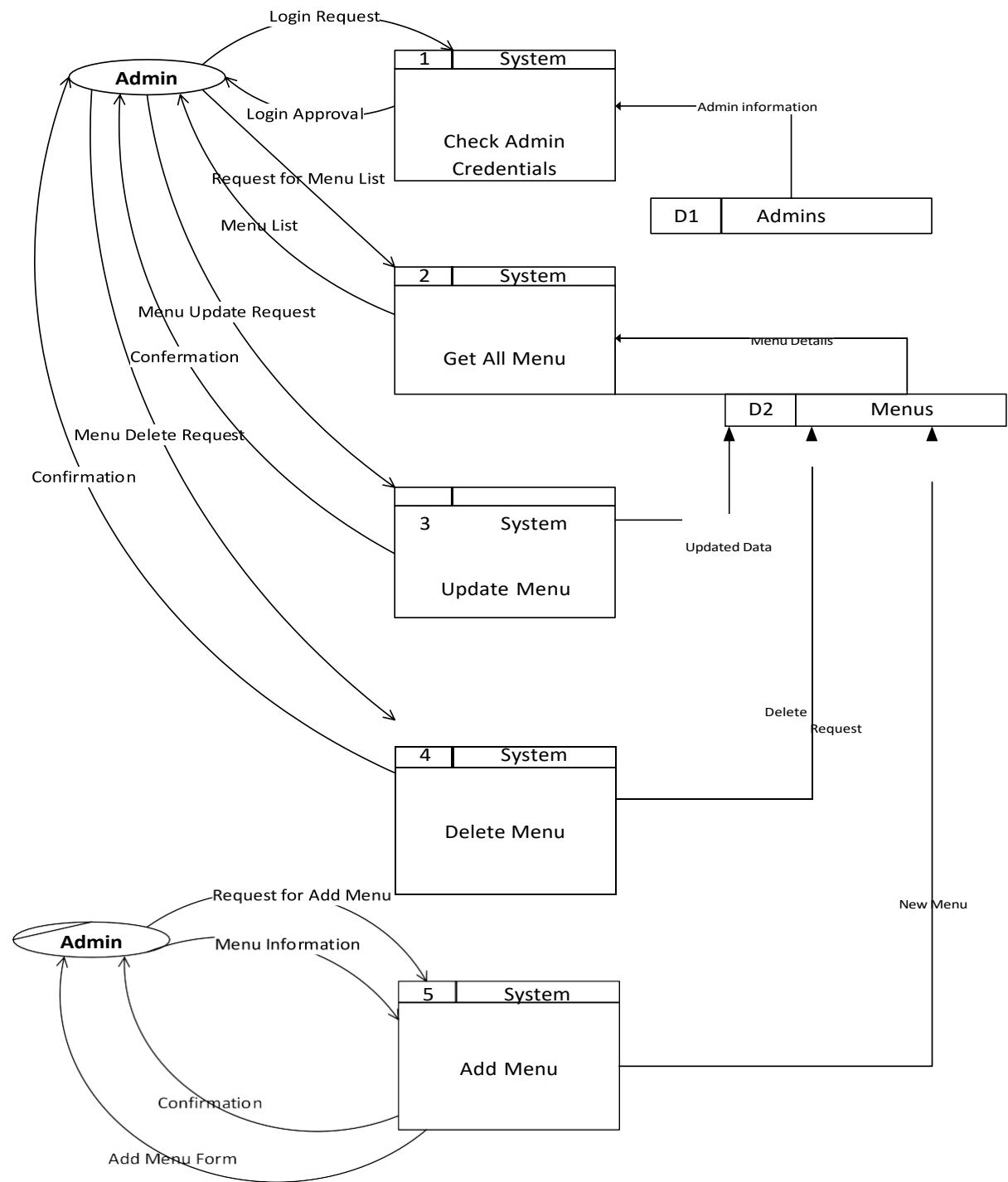


Figure 7 illustrates the processes of admin in for manipulating Menus table of the database. There are five processes for admin in this application. Process 1 depicts the login of admin to the system. When admin gives credentials and send login request to system, the system checks the credentials against the Admins table of the database and gives access. Process 2 illustrates the acquisition of available menu list for admin. In Process 3, the update process of menu has been depicted. In Process 4, deletion of menu has been illustrated. In Process 5, adding process of new menu is shown.

## **Software Development Life Cycle (SDLC)**

it is a process used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

SDLC is the acronym of Software Development Life Cycle.

It is also called as Software Development Process.

SDLC is a framework defining tasks performed at each step in the software development process.

ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

### **What is SDLC?**

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC.

### **SDLC Models**

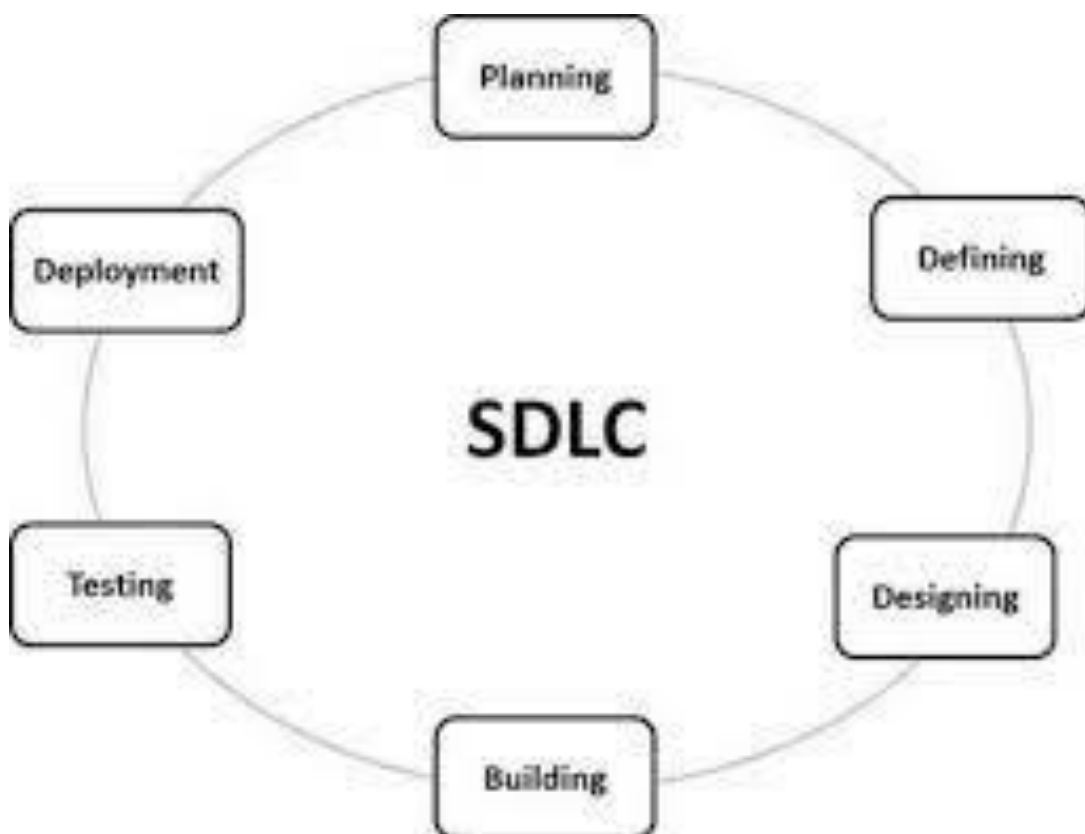
There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred

as Software Development Process Models". Each process model follows a Series of steps unique to its type to ensure success in the process of software development.

Following are the most important and popular SDLC models followed in the industry –

- Waterfall Model
- Iterative Model
- Spiral Model
- V-Model
- Big Bang Model

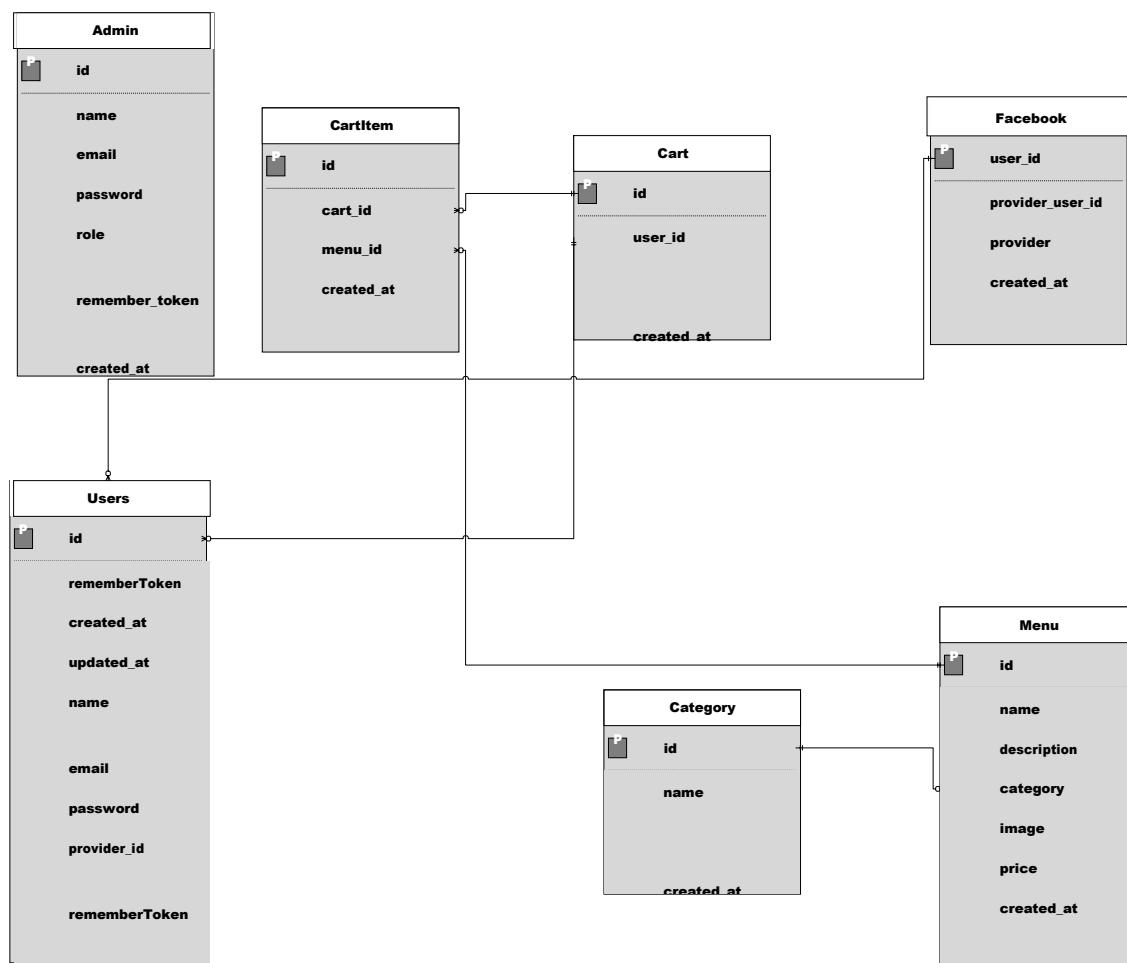
Other related methodologies are Agile Model, RAD Model, Rapid Application Development and Prototyping Models.



# Data Modelling

## Entity Relationship Diagram (ERD)

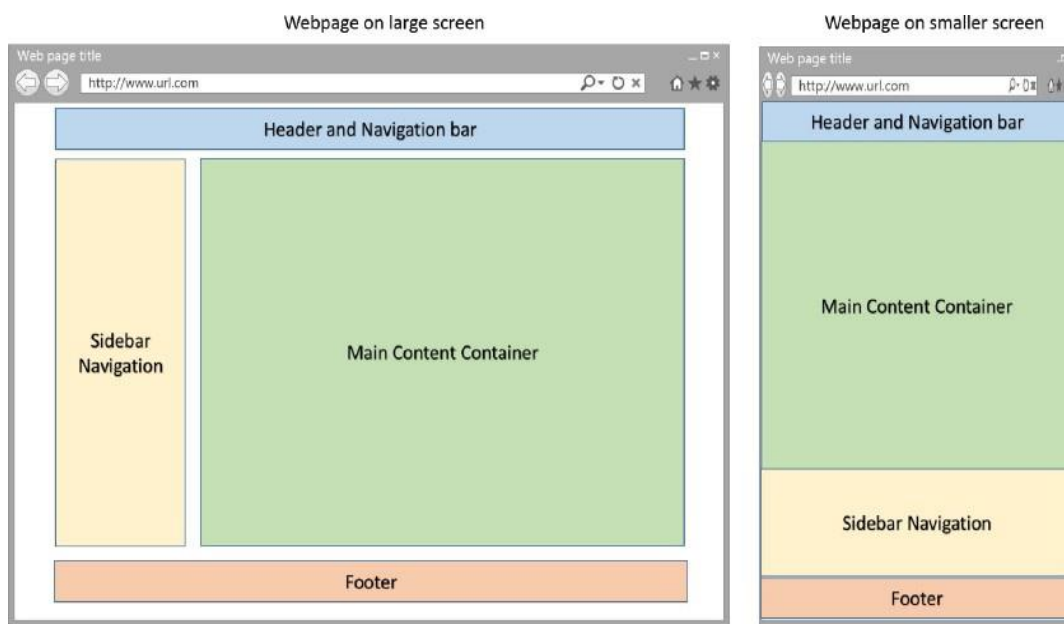
Entity Relationship Diagram (ERD) is a visual presentation which “facilitates database design by allowing specification of an enterprise schema that represents the overall logical structure of a database” (Korth, Sudarshan and Silberschatz, 2010). The following ERD illustrates the database design for restaurant application.



The crow foots shows the links between entities. A line with crowfoot in one side and normal line in another side means many-to one or one-to- many relationship. The ERD shows that, there is not any relationship between admin table and other tables in the database. User table has relationship with Razorpay and Cart tables. The Menu table has a relationship with CartItem and Category tables. The Cart table has a relationship with CartItem and User tables.

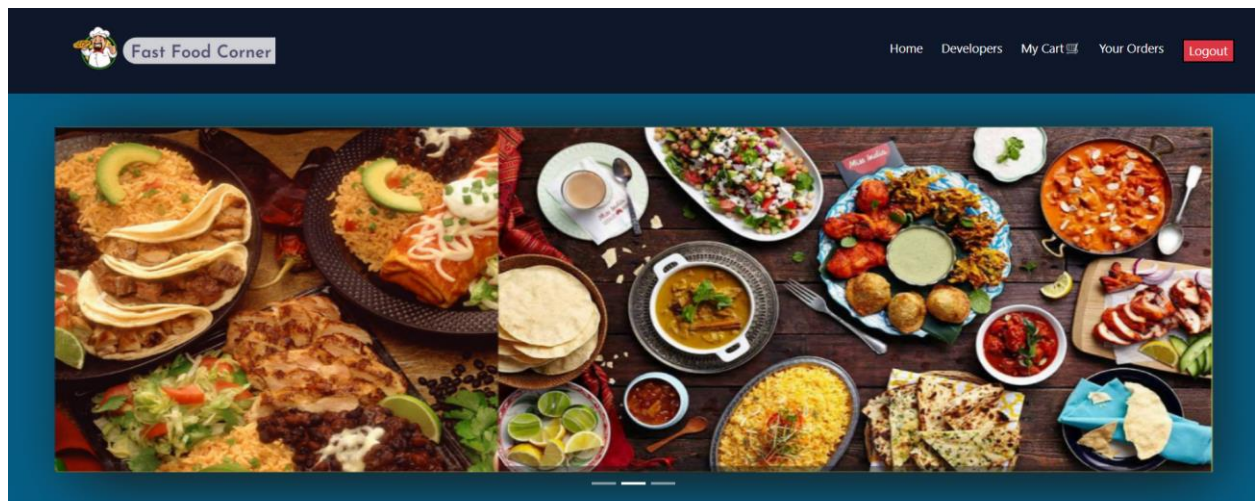
## User Interface Design

As specified earlier, the user interface of the system should be responsive. So, the interface can adopt any screen size. As the system can be accessed by devices with large screen (e.g. desktop computer, smart television browser) as well as devices with small screen (e.g. smart phone, tablet), every page of the application is designed for two types of screen differently.



## Homepage:

The homepage is the index page of the application. This is the page that is shown to visitor when s/he accessed the website through URL. Figure 10 illustrates the layout of the homepage in large screen



The homepage of the application consists of a menu bar with company name, a Jumbotron, featured food item and footer. The login and registration links appear in the top right side of the menu bar, if the user is logged in. If the user is logged in, there will be a dropdown button for logging out with displaying the username.

When this page appears in a small screen, the layout of the page will change to show the page to viewers more effectively. Figure 13 shows the changes of the menu list page when it appears in a small screen.

The homepage layout will change when someone visits the website from a smaller screen. When the website is visited from a device with a small screen, the menu bar is toggled under a button. If the toggle button is pressed, the menu bar comes out in a stacked manner. The header and footer layout is constant for all of the pages of the site.

## Menu List

The purpose of the menu list page is to display all of the menu items to visitors. The page shows the list of the menus with image and price. Visitors can add menus to cart from this page or can view the details of the menu. Figure 12 shows the layout of the menu in large screen.



## Our Dishes



### Potbelly A Wreck Sandwich

Potbelly's signature sandwich, A Wreck, is also its most popular one. this is a true carnivore's delight

₹ 150.0

Add to cart



### Jack in the Box Tacos

A Report said the chain sells more than 554 million of items per year. making tacos the chain's best-selling item.

₹ 180.0

Add to cart

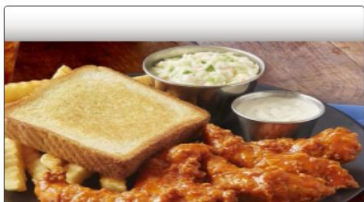


### Wendy's Dave's Single

According to a Wendy's rep, Dave's Single is a best-seller. This burger truly hits the spot when you have a fast-food craving.

₹ 100.0

Add to cart



### Zaxby's Chicken Finger Plate

It is available in both Original and Buffalo Chicken—it's listed as one of its most-ordered items on the company's website.

₹ 250.0

Add to cart



### Subway

it's turkey sub is most popular sandwich, according to a rep for chain. Specifically, sub with a bag of chips on the side.

₹ 240.0

Add to cart



### Chicken Honey Butter Biscuits

Church's Chicken has two clear menu stars: the Hand-Battered Fried Chicken and the Honey Butter Biscuits.

₹ 210.0

Add to cart

## **Implementation**

This chapter provides an overview of the implementation process. In first, the implementation of the frontend part is described. After that, different backend functionalities have been discussed and evaluated.

## **Views**

The user interface of the application is responsive. So, the interface can occupy the full screen in any kind of device. For making the user interface of the application mobile friendly, Bootstrap 4 is used. Django uses blade templating engine for making views and presentation layer of applications. Blade templating engine is used to develop and present the views. There is a main frontend view is developed, which has been extended for different views of different pages. The following code snippet shows the main view of the application.

```
<!DOCTYPE html>

@include('layouts.header')

<body>

@include('layouts.navbar')


@yield('content')
```

We can see that the main view extends and includes three other views which are header, navbar and footer. The header file is the HTML header files, which contains metatags of page. The navbar blade contains the navigation bar of the application. The footer blade contains the footer of the application. The footer blade also contains some JavaScript files needed for the application. For example, it contains the JavaScript file for connecting to Razorpay API.

The following code snippet shows the header blade. The header blade contains the metatags and CSS links.



### JACK IN THE BOX TACOS

A Report said the chain sells more than 554 million of items per year. making tacos the chain's best-selling item.

180.0 ₹ / piece

Add to cart

## Reviews

Enter Your Review

Submit

rohitshukla001 May 14, 2022

nice



[Home](#) [Developers](#) [My Cart](#) [Your Orders](#) [Logout](#)

### Active Orders

Total Bill Amount: 100.0 for 1 boxes.

Please pay either via PayTM or Upi or Cash at time of delivery!

Ordered	Name	Specification	Price	Total Pieces	Order Status
May 13, 2022	Wendy's Dave's Single	Jain Option Available	100.0	1 pieces	Active

### Past Orders

Sorry, you have not ordered any item yet!!!

© 2021-22 RohitShukla.me



[Home](#) [Developers](#) [My Cart](#) [Your Orders](#) [Logout](#)

Added to Cart!! Continue Shopping!!

### Your Cart

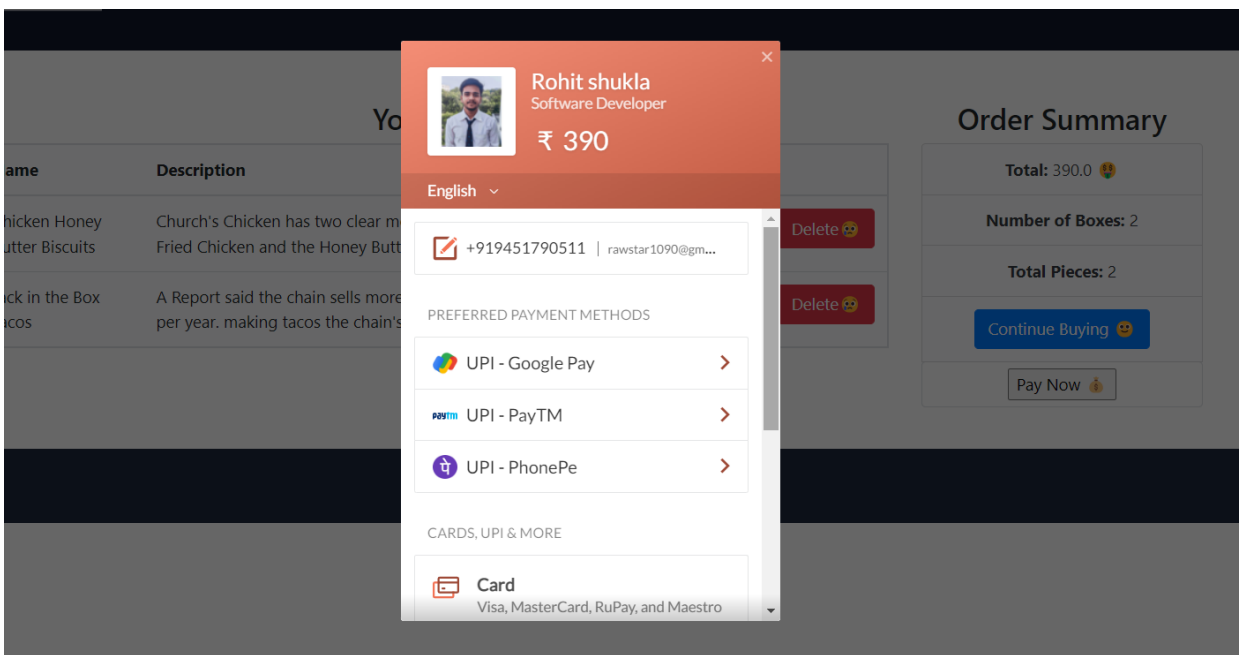
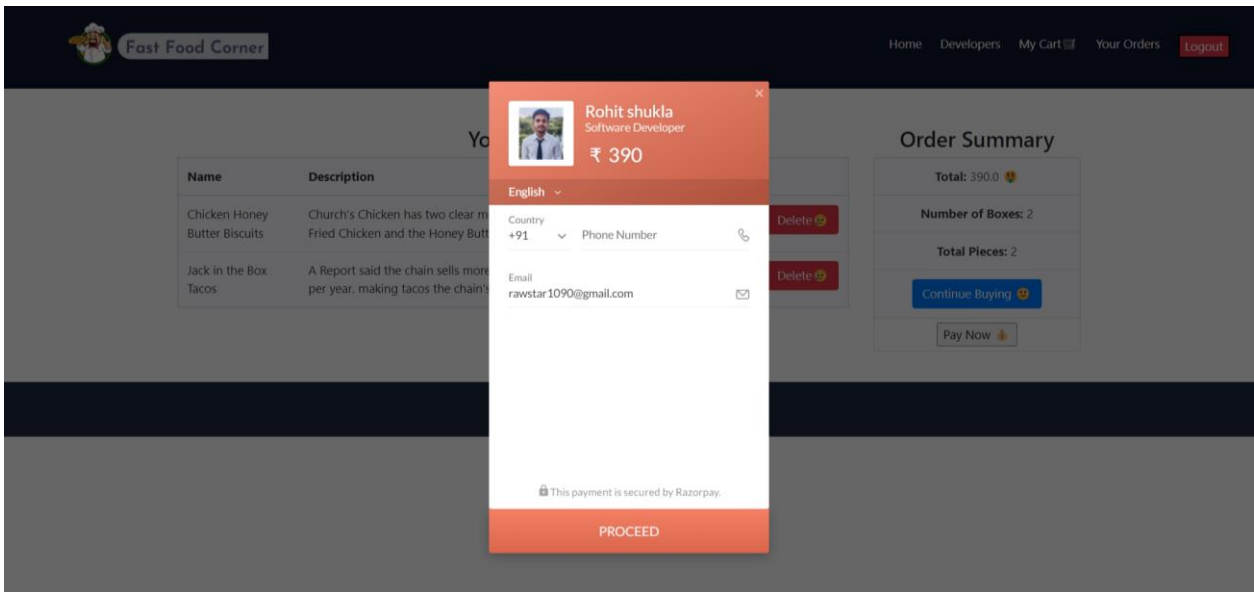
Name	Description	Price	Pieces	
Jack in the Box Tacos	A Report said the chain sells more than 554 million of items per year. making tacos the chain's best-selling item.	180.0	1	<a href="#">Delete</a>
Chicken Honey Butter Biscuits	Church's Chicken has two clear menu stars: the Hand-Battered Fried Chicken and the Honey Butter Biscuits.	210.0	1	<a href="#">Delete</a>
Jack in the Box Tacos	A Report said the chain sells more than 554 million of items per year. making tacos the chain's best-selling item.	180.0	1	<a href="#">Delete</a>

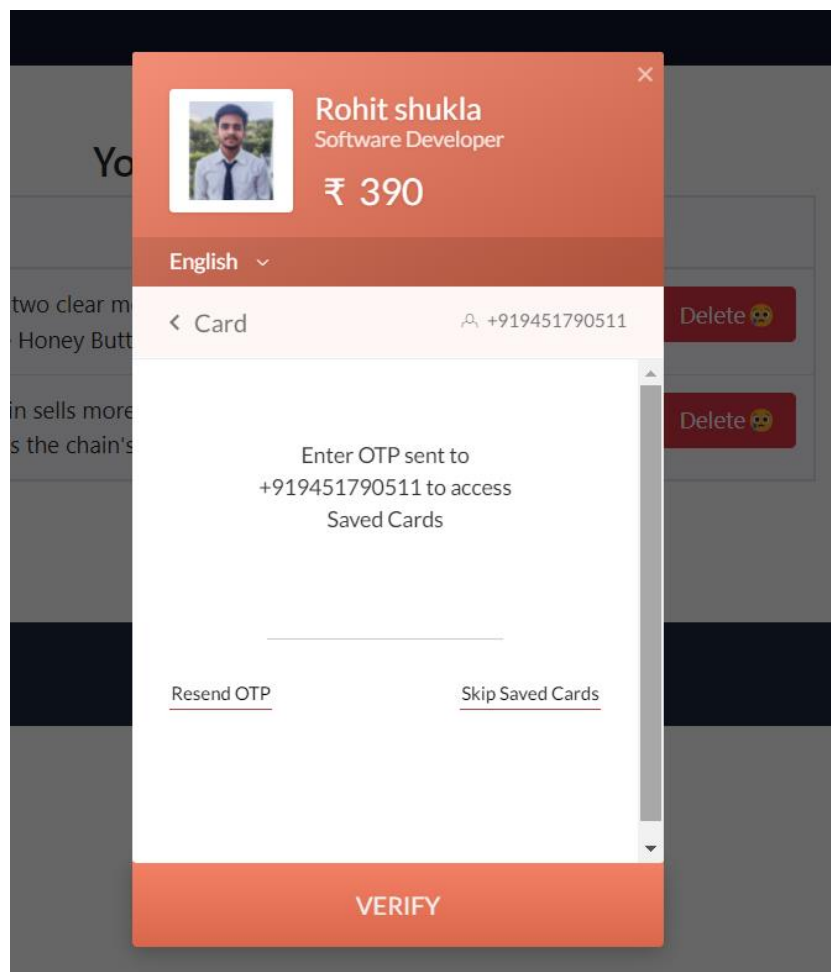
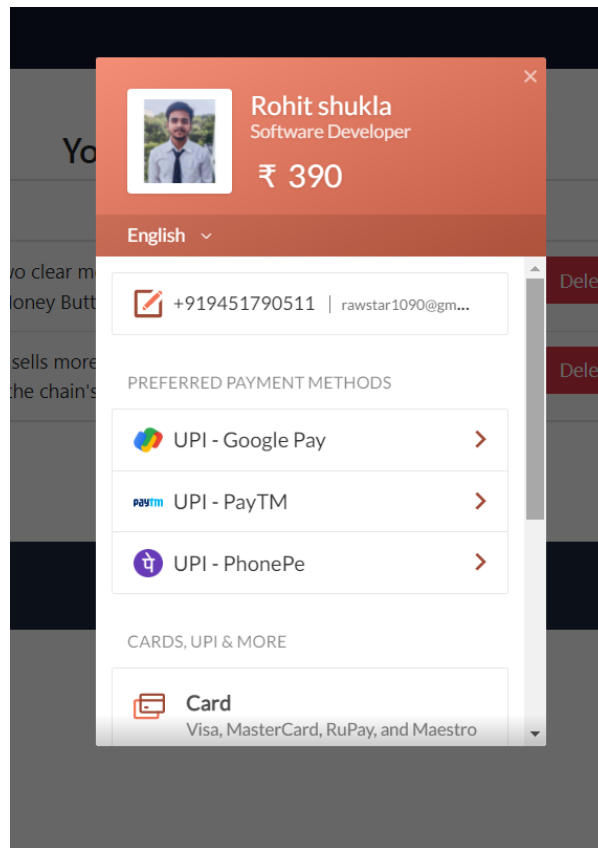
### Order Summary

Total: 570.0 🍔
Number of Boxes: 3
Total Pieces: 3
<a href="#">Continue Buying</a>
<a href="#">Pay Now</a>

© 2021-22 RohitShukla.me

# Payment System





```

<html lang="{{ app()->getLocale() }}">

<head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1,
    shrink-to-fit=no">

    <meta name="description" content="">

    <meta name="author" content="">

    <title> @yield('title')</title>

    <!-- Required meta tags -->


    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1,
    shrink-to-fit=no">


    <!-- Bootstrap CSS -->

```



## DEVELOPER


Fast Food Corner

[Home](#)
[Developers](#)
[DashBoard](#)
[Logout](#)



Rohit shukla

[View Portfolio »](#)

© 2021-22 RohitShukla.me

## Testing

This chapter elaborates the test process of the application and shows the result of testing.

### Black Box Testing

Testing can be divided into two types broadly: functional testing and structural testing. Structural testing, also known as white-box testing, involves examining the internal implementation. It tests the design used by the implementation to verify its correctness. In contrast, Functional testing, sometimes referred to as black box testing, is testing on the functionality of the system based on the specified requirement. The test itself has little knowledge about the testing target's internal structure. The following table shows the test cases for black box testing.

Test Code	Test Case	Test Steps	Expected Result	Actual Result	Pass /Fail
Test 1	Check Razorpay login functionality with logged in Razorpay user	1. Go to registration page 2. Click on "Login with Razorpay" button	The window is redirected to Razorpay, then to homepage with showing the Razorpay timeline name on the topright of the bar. The name and details also saved in database.	As expected	Pass

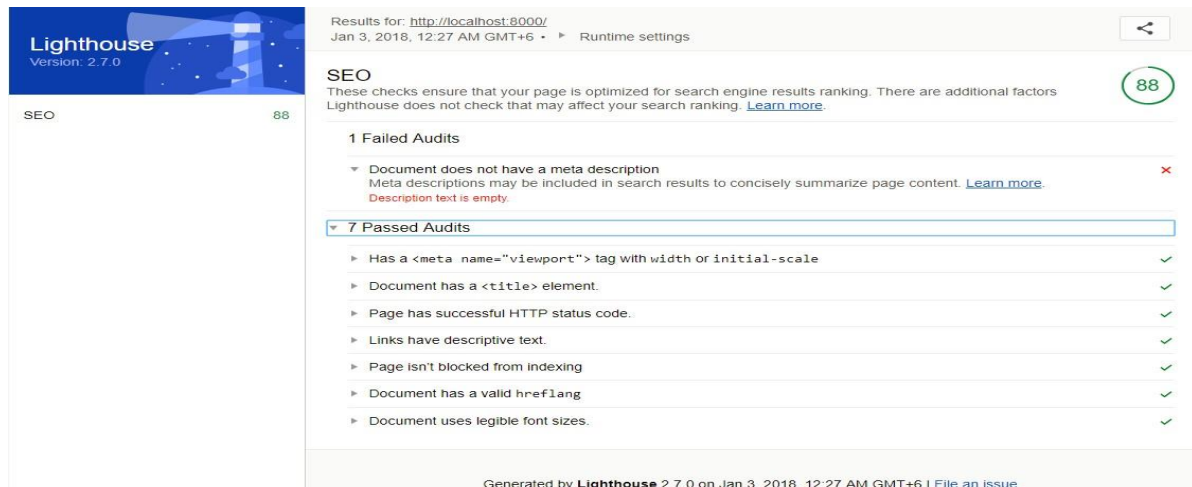
Test 2	Check Razorpay login functionality with non-logged in Razorpay user	<ol style="list-style-type: none"> <li>1. Go to registration page</li> <li>2. Click on "Login with Razorpay" button</li> <li>3. Enter Razorpay username and password on the next page</li> <li>4. Press sign in button</li> </ol>	The window is redirected to homepage with showing the Razorpay timeline name on the top right of the bar.	As expected	Pass
Test 3	Check adding to cart functionality for logged in user	<ol style="list-style-type: none"> <li>1. Go to any menu details page</li> <li>2. Click on add to menu button</li> </ol>	The user is redirected to cart page where the subtotal and total price shown.	As expected	Pass
Test 4	Check adding a menu twice to cart	<ol style="list-style-type: none"> <li>1. Go to any menu details page</li> <li>2. Click on add to menu button</li> <li>3. Get back to menu page</li> <li>4. Add the same menu to cart</li> </ol>	The user is redirected to cart page where the updated subtotal and total price shown.	As expected	Pass



Test 5	Add new menu into database	1. Fill up the add menu form 2. Press add button	A confirmation message will be shown. New menu added to database. The image of the menu uploaded to images folder with menu name.	As expected	Pass
Test 6	Edit existing menu	1. Navigate to menu index page 2. Click edit menu 3. Change existing value 4. Click update	The menu description should be changed in menu page and admin menu list.	As expected	Pass

## Search Engine Optimisation (SEO) Testing

There are different types of tools which automate testing processes and give results about the application. Lighthouse is an open-sourced tool made by Google for web applications which can be used to test Search Engine Optimisation (SEO) Compatibility of web application (Google, 2017). Search Engine Optimisation (SEO) is very important for web applications. A well-designed application for SEO can be found in search engines more easily and accurately. Lighthouse provides a SEO testing system for web applications and gives suggestions about potential problems.



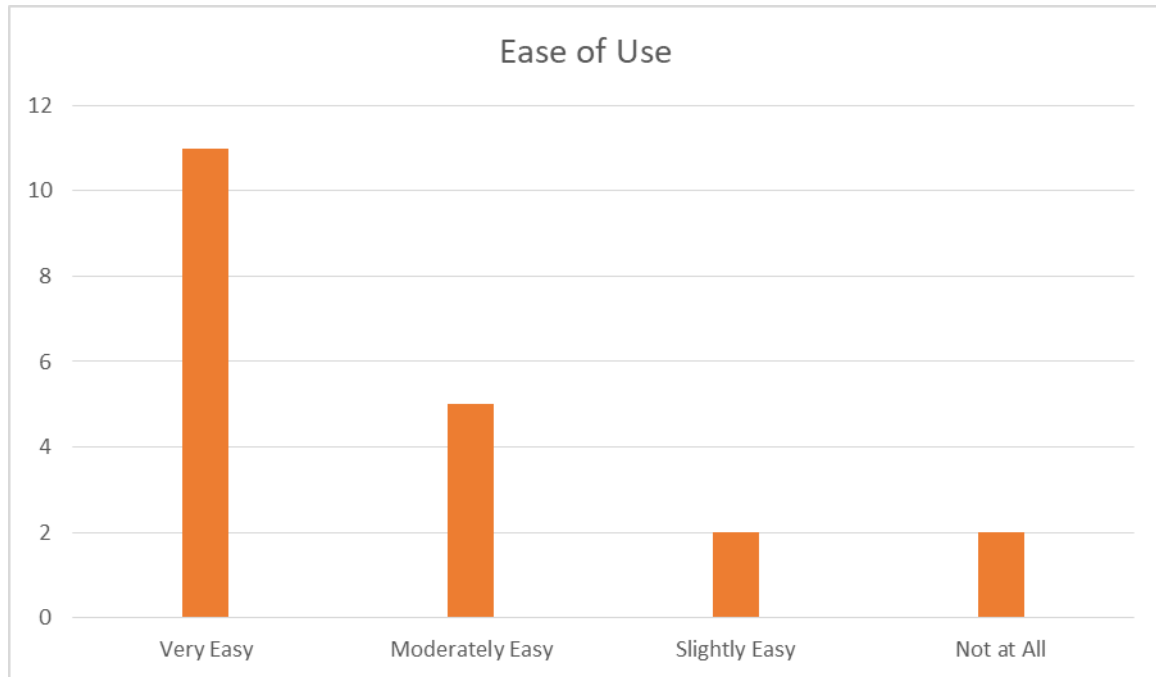
Above image that the restaurant app passed 7 tests out of 8 tests. The test result shows that the metadata for page is not used in header section of the page which may cause inappropriate result in search engine. The application gets overall 88 percent marks.

## User Acceptance Testing

User acceptance testing of the application is used to determine if the application meets the requirements of focused people. For user acceptance testing, random selection process is used. A questionnaire is made to obtain user evaluation of the system. The questionnaire consists of five questions, which are set to gain user perspective on the ease of use, user friendliness, and responsiveness of UI and performance of the application. The answers have been taken anonymously from 20 random persons. The questionnaire can be found in Appendix B.

## Ease of use

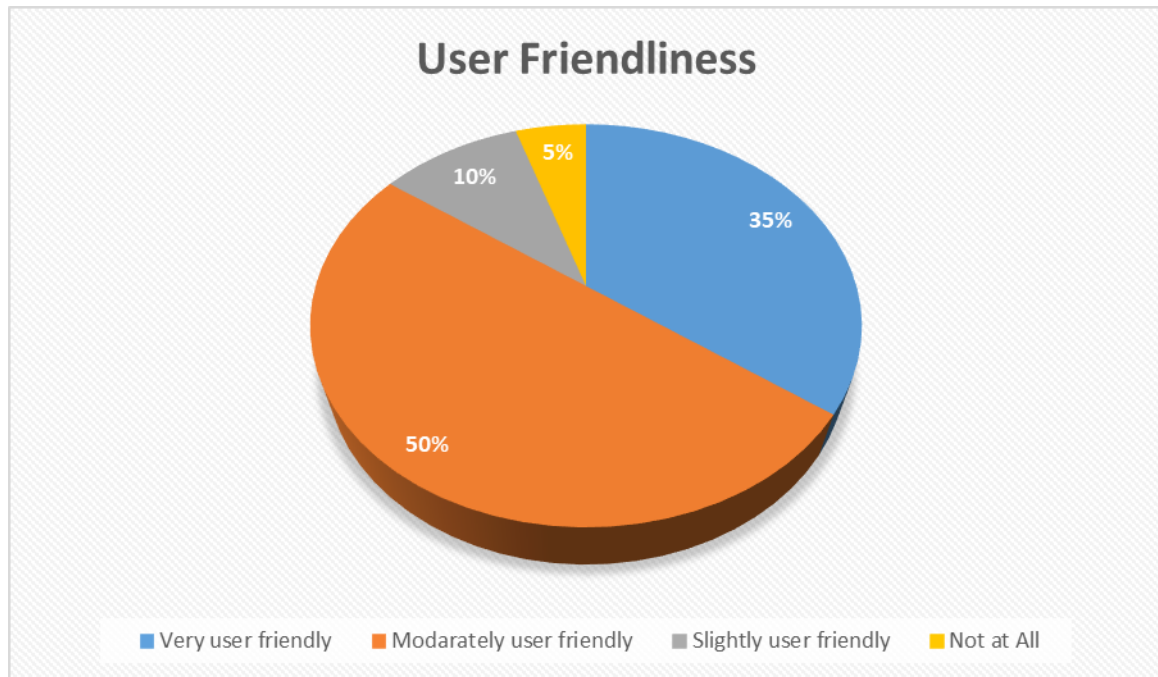
The first question of the questionnaire is for addressing the ease of use of the system. Figure 35 illustrates the result of this question.



As the bar chart shows, 11 persons of 20 is ticked that the application is very easy and 5 found the application is moderately easy. Only 4 persons found the application is slightly easy or not at all.

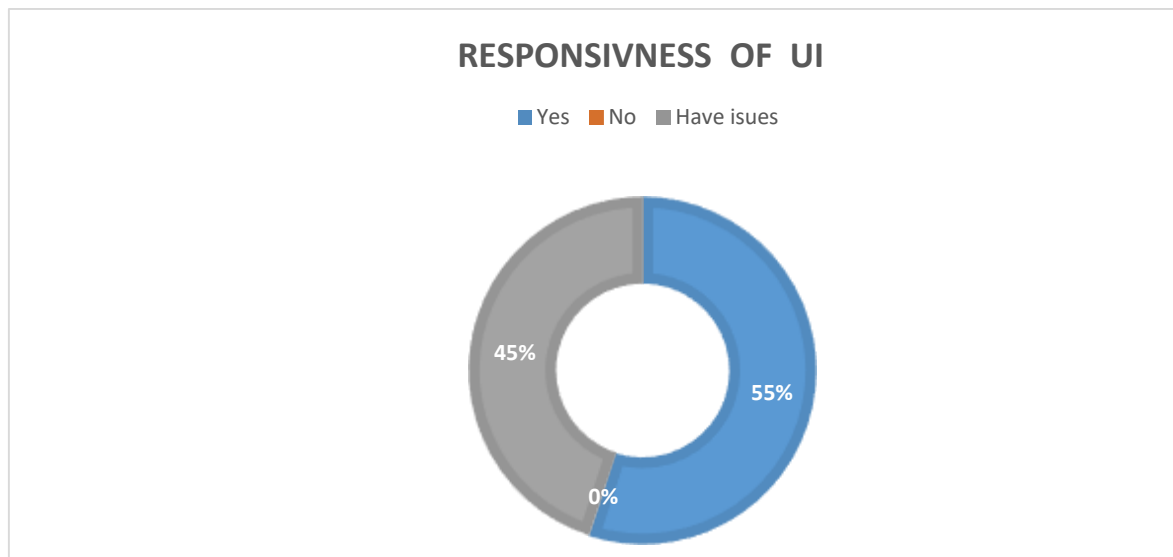
### **User Friendly**

The pie chart in Figure 36 illustrates the percentage of views of respondents on user friendliness of the application. Half of all respondents agreed that the system is moderately user friendly and 35% of them found the system very user friendly. So, it can be said that the website is user friendly. But there is a room for improvement as 15% of respondents do not find it so much user friendly.

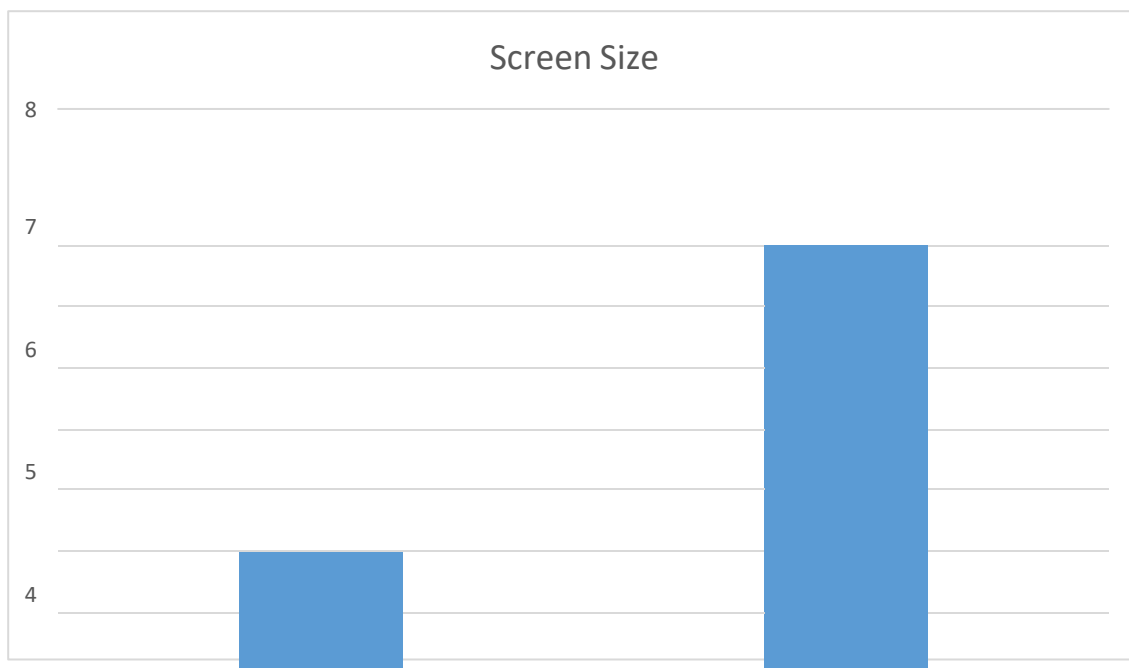


### Responsiveness of Interface

The responsiveness of UI means that the website UI is optimised properly for different size of screens. The following donut pie chart shows that all respondent found the website responsive. But there are many issues to improve the responsiveness of the UI as 45% of respondent faced problem with their device.



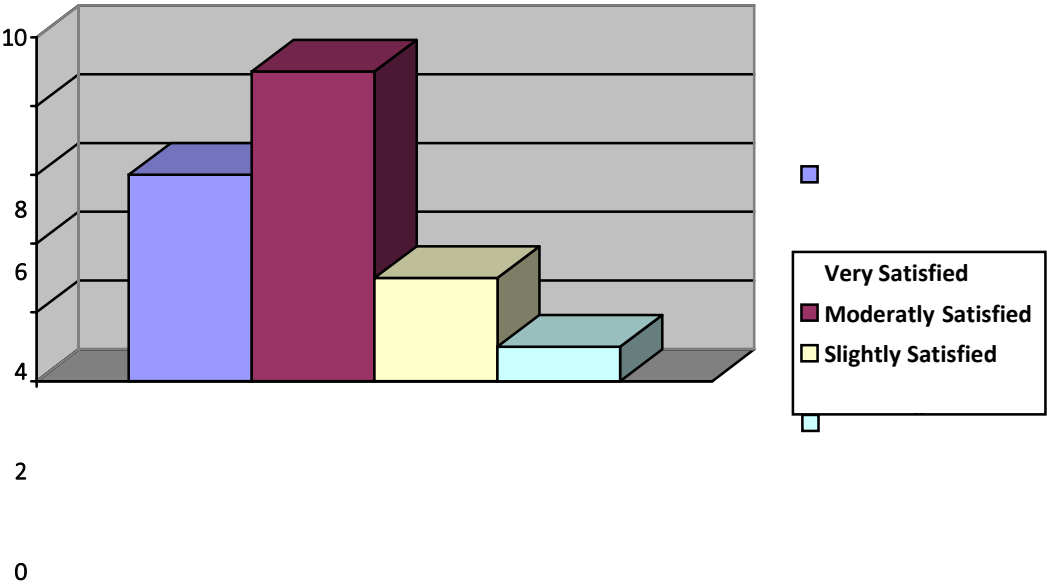
There can be many issues that can make problem in responsiveness. It has been tried to find some of them. Who have problem with UI of the website, their screen size data collected and analysed. The result is shown in the bar chart in Figure 38.



We can see, most of respondents who face problem in responsiveness use a device with a screen smaller than 5 inch. So, UI should be optimised for smaller screens.

**Speed & Performance**

The pie chart in Figure 39 illustrates the percentage of satisfaction with speed and performance of the system.



According to pie chart, 47% of respondent is very satisfied with the performance and speed and 32% of respondent is moderately satisfied with the system.

## **Results & Evaluations**

This chapter provides an overview of the project and its outcomes. The outcomes of the project also is also analysed and evaluated.

### **Security of the System**

Security is one of the most important thing for a web application, especially if the system contains transection and user personal data. As it is an online food ordering system, transactions will be made through the app and the application also saves the different kind of personal information. In regards of ensuring the security of the application different techniques have been used. For ensuring the security of the password, default hashing function of Django has been used. Django handles all the passwords by encrypting and decrypting automatically, which has been hashed based on the secret key of the application. One the biggest problem for web applications is Cross-site request forgery (CSRF) attack. CSRF is a type of malicious exploit whereby unauthorized commands are performed on behalf of an authenticated user when s/he clicked on malicious link. Django automatically generates a CSRF "token" for each active user session managed by the application. This token is used to verify that the authenticated user is the one actually making the requests to the application. Anytime a HTML form in the application is defined, a CSRF hidden token field is included in the form so that the default CSRF protection middleware of Django can validate the request.

### **Responsive User Interface**

The responsiveness of the application in any kind of screen was a big requirement for the system. In regards of achieving this objective, Bootstrap 4 front end framework has been used to develop the user interface of the system. This enables the UI of the application to glue any kind of screen whether it be smart phone, tablet PC, laptop or desktop PC. It also responsive for the proximity of the device. The application interface matches the size of the device automatically whether it is kept horizontally or vertically. Bootstrap 4 also reorganise the elements of the page based on the screen size.

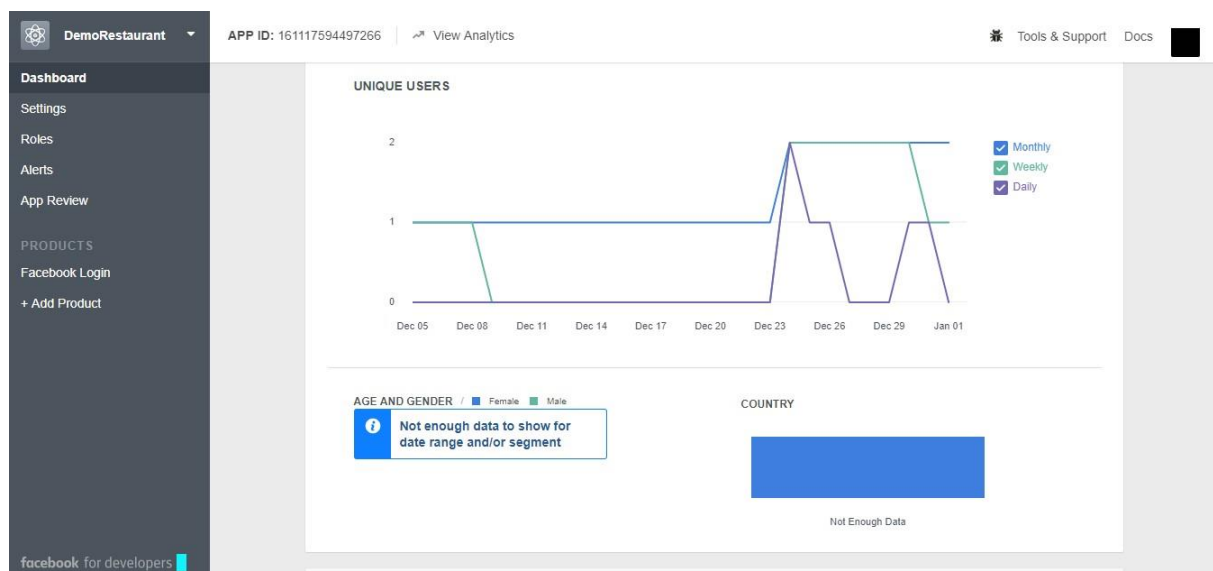
### **Payment API Integration**

According to Statista, a Hamburg based leading statistics company, Razorpay is

the most popular payment website by number of active users (Statista, 2017). On the other hand, it has been explained based on the different studies that integration of payment media online buying portals increase the trust of the users on the website. The comments and likes of friends and other relatives on payment media influence people to order online. Hence, Razorpay API has been added to menu details pages. Users can

like, share and comment through Razorpay payment media plugin. The like button shows the friend names of user who liked the menu previously. Users can directly share the menu in personal timeline by clicking on the share button or can comment to express their thought about the menu. This helps the management about the impression of the food menu on the users. Then, the management can take steps to improve the quality of the food.

The integration of Razorpay API also helps the management in keeping track of users and get different kinds of insights about the users. As the developed restaurant application has not any in-app business intelligence tool, the Razorpay analytics can give the management a good overview of the online customers. In-app analytics application development can make expand the length of the project. But it is good tool for management. Restaurant owners can make decisions more informatively if they have a business intelligence. The lacking of in-app business analytics tool can be minimised by utilising the Razorpay analytics tool.





The Razorpay API also eased user registration and login process. New users can register for ordering food by tapping one button. New users do not need to fill up registration form which is a boring thing for most people. On the other hand, existing users can login by pressing one button.

### **Payment Sharing Options**

Different payment media sharing options are implemented in the system. The users can tweet about the food menus in Twitter, like and share them in Razorpay. Users can also make comments on the food items page to express their review on the food item.

Whenever a user uses Razorpay or Twitter for sharing food menus, their friends and followers can know about the restaurant. As they know the restaurant through their friends, they will feel trust about the quality of the restaurant.

### **Conclusions and Recommendations**

This chapter concludes the report of this project. This chapter starts with discussing the achievements of this project. Following that, it describes the limitations in the system. It then proposes and recommends some features to be added to the system. Finally, the chapter ends by concluding remarks.

### **Achievement of the Project**

The project has gone through a series of activities to develop a complex solution for the online food ordering system. After analysis of the project's goal and research direction, a set of objectives were established, as specified in Chapter 1.2. All the activities done during the project were attempts to realise these objectives. At the end of the project, the developed prototype web application has fulfilled these objectives by the following means:

- Objective #1 was satisfied by reviewing the past works for automating the restaurant food ordering process. Along with this, the web application development technologies is briefly discussed.

- Objective #2 was addressed by utilising Extreme Programming method of Agile Development. Along with this, SSADM tools are used to analyse and design the system.
- Objective #3 was satisfied by developing the system with Django 5.4 and Bootstrap 4.
- Objective #4 was satisfied by integrating Razorpay API to the system.
- Objective #5 was addressed with various testing approaches to ensure the prototype system is as robust as possible.

The project was time-consuming. It has been tried to implement as many features as possible within the very limited timeframe. It has successfully satisfied the Functional Requirements. Some Non-functional Requirements of the system is not implemented. These requirements have top priority and reflect the most needed features. Some requirements are not implemented due to time constraints. However, their absence would not result in major operational issues as they are the lower priority features. These features could be implemented in the future.

### **Limitations of the System**

There are also some limitations of the system. The shopping cart of the system has basic functionalities and does not support advanced cart modification features. Along with this, validation functionalities and almost all functionalities of the application are handled with server side programming. It makes extra load on the server, especially when the application gets lots of viewers. This limitation can be minimised by validating data using client side language like JavaScript or HTML 5. Along with this, the order model has been developed. But the controllers and functions for pushing data into order table is not written. So, the placed orders cannot be viewed.

### **Future Recommendations**

In addition to the unfinished requirements, there are other possibilities of further improving the project. The respondent of user acceptance testing also suggest some improvement ideas. The improvements may include:

- Overcoming the limitations mentioned in the Section 7.3.
- Secured payment system with various payment methods.
- Presenting graphical floor plan for table management and reservation.
- Adding support for food order delivery tracking.
- Advanced inventory control with material storage and expiry information.
- Managing customer loyalty membership and discount voucher.
- Converting the system to progressive web application.

## **System Prototyping**

Prototyping is a Rapid Action Development (RAD) method. In this method, the analysis, design and implementation phases performed concurrently and repeatedly in a cycle until the system is completed. With this methodology, the basics of analysis and design are completed. Then the work on the system prototype begins immediately. So that, many bugs and problems remained on the system. After that, the users or project sponsors provide comment on the system. Then, the system is reanalysed, redesigned and implemented based on feedbacks. This process continues until the users or project sponsors satisfied with the system (Gould, 2016).

## **Agile Methodology**

Agile is an iterative and incremental development model. The Agile methodology is initiated in 2001 at a conference held in Utah, USA. The required web application starts with a simple design, then to code small functions and modules. The work on these functions and modules is done in weeks for each life cycle which is called increment or sprint. In these sprints, errors to be recognised, and customer feedbacks to be incorporated into next design of the next increment (Ben-Zahia and Jaluta, 2014).

There are many models related to this methodology. The most famous ones include Scrum, lean, and Extreme Programming (XP).

## **Rational for Selected SDLC**

There are different criteria for selecting appropriate SDLC for developing an IT system, because none of the SDLCs is best. Every SDLC has own advantages and disadvantages. So, the SDLC should be chosen carefully based on the type of the project. Dennis, Wixom and Roth (2012) discussed about different criteria for selecting the appropriate development methodology for a project. SDLC selection criteria provided by them is shown in the table below.

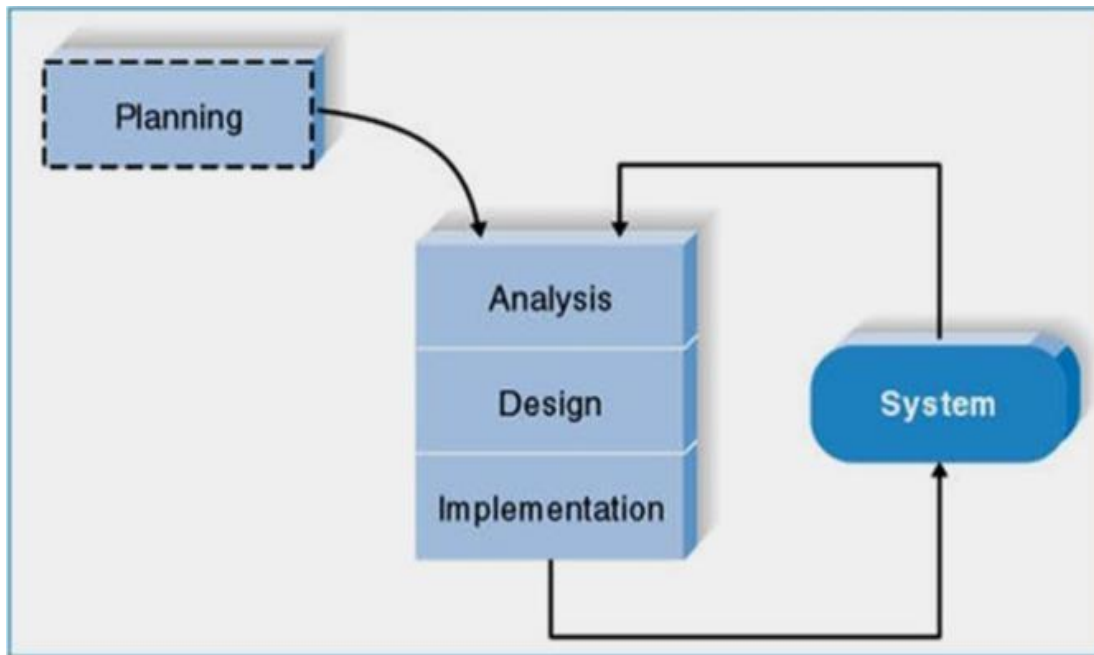
Usefulness in Developing System	Waterfall	System Prototyping	Agile Development
With unclear user requirements	Poor	Excellent	Excellent
With unfamiliar technology	Poor	Poor	Good

That are complex	Good	Poor	Poor
That are reliable	Good	Poor	Good
With short time schedule	Poor	Excellent	Excellent
With schedule visibility	Poor	Excellent	Good

Table 2 illustrates that agile development is the best solution when the time is short, deadline is visible and the system should be reliable. It also has excellent performance when the requirements of system is not clear. Its performance is poor when the system complex. However, the system that is going to be built is not a complex system and the system should be developed within a short time. Therefore, it has been considered that agile methodology is the best SDLC for making the system.

On the other hand, Waterfall methodology is good for developing complex and reliable system. But its performance is poor when the schedule is short and deadline is visible. The System Prototyping method has an excellent performance with unclear requirements and within short time schedule. So, it could be a good choice for building this restaurant system. But the reliability of product developed with System Prototyping is poor. No one wants to build a system that he or she cannot rely on. In these criteria, Agile Development provides good result. So, the Agile Methodology has been chosen as the web application development methodology for the system.

There are many models of implementing Agile Methodology. Extreme Programming (XP) has been chosen. Extreme Programming (XP) model has four stages for developing a project, as the figure illustrates below.



As the figure above shows, there are mainly four phases in Extreme Programming method. These phases are discussed below:

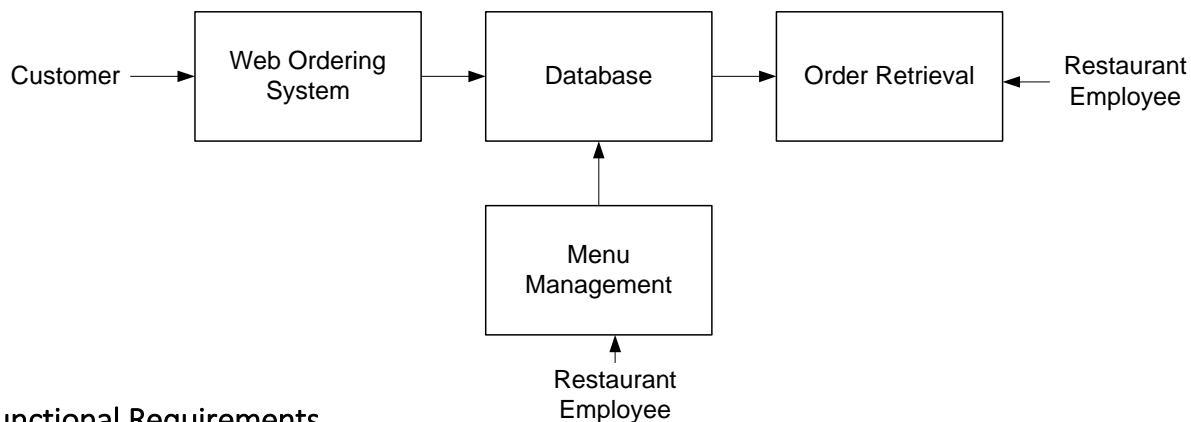
- **Planning:** Extreme Programming starts with the planning stage. In this phase, the requirements for the system have been collected and documented. In this step, the plan, time, and costs of carrying out the iterations is prepared.
- **Analysis:** In this phase, the logical model of the system has been developed. SSADM is used to make the logical structure of the system. A bottom-up approach is used for analysis of the system, as there is not any previous system. The system will be developed from scratch. Along with these, the user interface requirements are also analysed.
- **Design:** In this phase of the SDLC, the logical model of database and the interface of the system is designed. The normalisation of the database schema is done in this phase.
- **Implementation:** In this phase, the system has been implemented through coding. As extreme programming is an iterative method, it is

possible to use test driven development method using unit testing. After the system has been developed, the end-to-end testing (black box testing, user acceptance testing) will be used to evaluate the system. It should be kept in mind that analysis, design and implementation phases are iterative phases. After completing one iteration, feedback has been taken. Then all the phases began for the next iteration.

## **Requirements Specification**

### **System Model**

The structure of the system can be divided into three main logical components. The first component must provide some form of menu management, allowing the restaurant to control what can be ordered by customers. The second component is the web ordering system and provides the functionality for customers to place their order and supply all necessary details. The third and final logical component is the order retrieval system. Used by the restaurant to keep track of all orders which have been placed, this component takes care of retrieving and displaying order information, as well as updating orders which have already been processed.



### **Functional Requirements**

As can be seen in the system model diagrammed above, each of the three system components essentially provides a layer of isolation between the end user and the database. The motivation behind this isolation is twofold. Firstly, allowing the end user to interact with the system through a rich interface provide a much more enjoyable user experience, particularly for the non-technical users which will account for the majority of the system's users. In addition, this isolation layer also protects the integrity of the

database by preventing users from taking any action outside those which the system is designed to handle. Because of this design pattern, it is essential to enumerate exactly which functions a user will be presented and these functions are outlined below, grouped by component.

### **The Web Ordering System**

Users of the web ordering 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.
- Customize options for a selected item.
- Add an item to their current order.
- Review their current order.
- Remove an item/remove all items from their current order.
- Provide delivery and payment details.
- Place an order.
- Receive confirmation in the form of an order number.

As the goal of the system is to make the process of placing an order as simple as possible for the customer, the functionality provided through the web ordering system is restricted to that which most pertinent to accomplish the desired task. All of the functions outlined above, with the exceptions of account creation and management, will be used every time a customer places an order. By not including extraneous functions, I am moving towards my goal of simplifying the ordering process.

### **Menu Management System**

The menu management system will be available only to restaurant employees and will, as the name suggests, allow them to manage the menu that is displayed to users of the web ordering system. The functions afforded by the menu management system provide user with the ability to, using a graphical interface:



- Add a new/update/delete vendor to/from the menu.
- Add a new/update/delete food category to/from the menu.
- Add a new/update/delete food item to/from the menu.
- Add a new/update/delete option for a given food item.
- Update price for a given food item.
- Update default options for a given food item.
- Update additional information (description, photo, etc.) for a given food item.

It is anticipated that the functionality provided by this component will be one of the first things noted by the restaurant user, as they will have to go through it to configure their menu, etc. before beginning to actually take orders. Once everything is initially configured, however, this component will likely be the least used, as menu updates generally do not occur with great frequency.

### **Order Retrieval System**

Of the three components, the order retrieval system is functionally the simplest. Like the menu management system, 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.
- Mark an order as having been processed and remove it from the list of active orders.

### **User Interface Specifications**

Each of the system components will have their own unique interface. These are described below.

## **Web Ordering System**

Users of the web ordering system will interact with the application through a series of simple forms. Each category of food has its own form associated with it which presents a drop down menu for choosing which specific item from the category should be added to the order, and a series of check boxes and radio buttons for selecting which options are to be included. Adding an item to the order is accomplished by a single button click. Users select which category of food they would like to order, and therefore which form should be displayed, by navigating a menu bar, an approach which should be familiar to most users.

Entering delivery and payment details is done in a similar manner. The user is presented with a form and must complete the required fields, which include both drop down and text boxes, before checking out and receiving a confirmation number. One thing worth noting here is that whenever possible drop down boxes and buttons were used over freeform input in order to both simplify the ordering process and reduce the possibility of an SQL injection attempt.

## **Menu Management System**

User interaction with the menu management system is similar to that with the web ordering system. Users navigate a tree structure to find the vendor, category, or specific food item that they would like to modify and after making their selection they are presented with a form which displays all of the current fields and values associated with that item, all of which can be modified or removed. The form also presents buttons which allow the addition of new fields and values. Unlike the web ordering system, however, most of the input here will be freeform, specifically in the form of text boxes, since there is no finite set of fields which could be added. This does not raise a major concern though, as input sanitation will be performed, and the user, who is assumed to be a restaurant employee, is less likely to be malicious than a web user.

## **Order Retrieval System**

User interaction with the order retrieval will be very simple. The application will automatically fetch new orders from the database at regular intervals and display the order numbers, along with delivery time, in a panel on the left hand side of the application. To view the details of an order, the user must simply click on that order

number, which will populate the right-hand panel with the details, displayed in an easy to read and navigate tree structure. This structure can intuitively be expanded and collapsed to display only the desired information. Finally, once an order is processed, the user clicks a single button, labeled "Processed", to remove it from the list of active orders.

### **Non-functional Requirements**

Because the design patterns of the Online Ordering System are pretty much the standard for a web application, the non-functional requirements of the system are very straightforward. Although written using Google Web Toolkit, the application is cross-compiled to HTML and JavaScript, along with a PYTHON backend, all of which are supported by any reasonably well maintained web server, although I would recommend Apache2, and particularly the free XAMPP distribution.

All of the application data is stored in a PostgreSQL database, and therefore a PostgreSQL server must also be installed on the host computer. As with Apache2, this web application is freely available and can be installed and run under most operating systems.

The server hardware can be any computer capable of running both the web and database servers and handling the expected traffic. For a restaurant that is not expecting to see much web traffic, or possibly doing only a limited test run, an average personal computer may be appropriate. Once the site starts generating more hits, though, it will likely be necessary to upgrade to a dedicated host to ensure proper performance. The exact cutoffs will need to be determined through a more thorough stress testing of the system.

### **System Evolution**

As mentioned in the system model, at the heart of the entire ordering system is the database. In fact, the system could be completely operational using nothing but the database and an appropriate shell utility, assuming that all users are well-versed in SQL and enjoy using it to order food. While this would be a bit extreme, it does illustrate the point that the one part of the system which will stay relatively constant is the database.

On the other hand, it is very probable that the other components will continue to evolve with time. For example, with the booming popularity of mobile applications, I would really like to make the web interface available as a phone application as well. Also it may

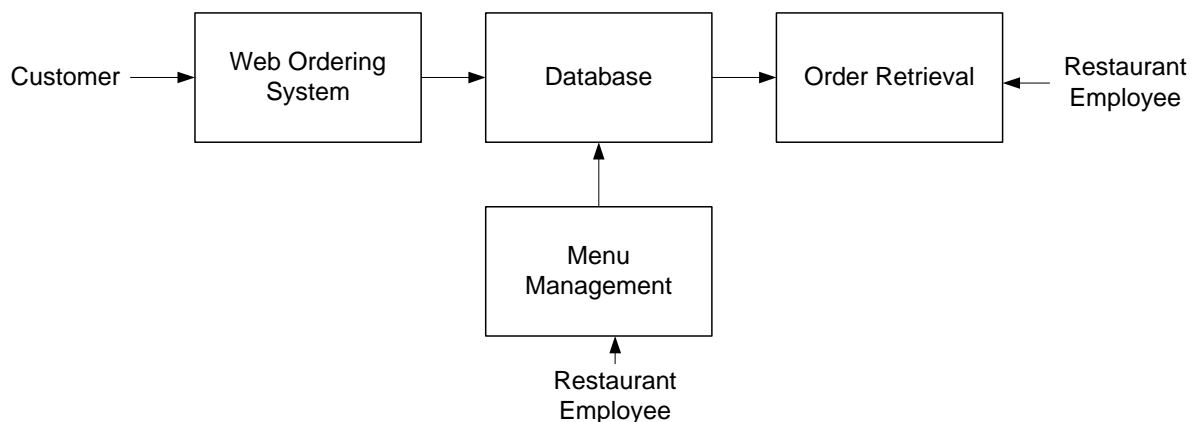
make sense to at some point migrate the menu management and order retrieval systems to web, or even mobile, applications as well, as some users may prefer to use them as such.

I am also certain that if this system goes into actual use, many requests will arise for additional features which I had not previously considered, but would be useful to have. For this reason, I feel as though the application can be constantly evolving, which I consider a very good thing.

## **System Design**

### **Level 1: The Database & the 3 Components**

The structure of the system can be divided into three main logical components. The first component must provide some form of menu management, allowing the restaurant to control what can be ordered by customers. The second component is the web ordering system and provides the functionality for customers to place their order and supply all necessary details. The third and final logical component is the order retrieval system. Used by the restaurant to keep track of all orders which have been placed, this component takes care of retrieving and displaying order information, as well as updating orders which have already been processed.

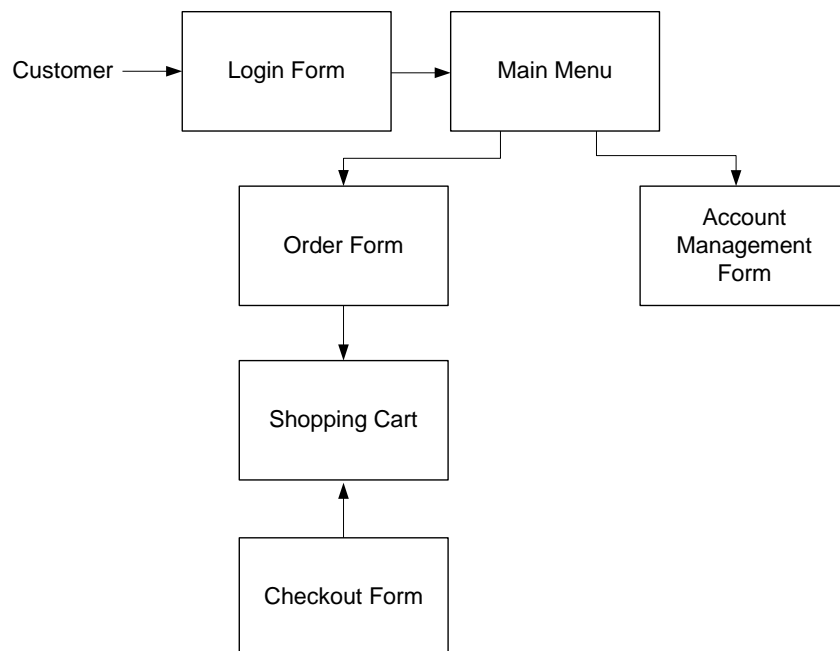


### **Level 2: Web Ordering System Components**

The web ordering system is comprised of 6 major components. These are the login form, the main menu, the account management form, the order form, the shopping cart, and

the checkout form. When the customer first arrives at the site, they are presented with the login form. After either signing in or, if they do not yet have an account, first registering and then signing in, the user is taken to a welcome page with the main menu.

From here, they have two options – they can either change their password and other preferences through the account management form, or they can select an order form and begin adding items to their order. If they choose the second path, they can navigate the numerous order forms using the main menu, each of which corresponds to a specific category of order items, adding items to their shopping cart along the way. At any time they can view and modify their shopping cart and when they are finally ready to place their order, they can proceed to the checkout form. The checkout form uses the contents of the shopping cart to present a summary of the order and to calculate the total cost, in addition to allowing the user to specify all of the necessary delivery details.



### Level 3: The Login Form

The login form is standard for a form of this type. It provides text fields for username and password, which the user must enter before signing in. This form also gives the option for a user to register for the site if they have not yet done so.

### **Level 3: The Main Menu**

The main menu, found at the top of the screen like in most applications, presents the user with two levels of selections. They must first choose the vendor they would like to view and then choose a category of food. Once they make these two selections, the application generates an order form specifically for that type of food, and displays this form to the user.

### **Level 3: The Account Management Form**

Currently the account management form only offers the user the option to change their password.

### **Level 3: The Order Form**

The order form, which is dynamically generated based on selections from the main menu,

### **Level 3: The Shopping Cart**

The shopping cart performs much like a shopping cart in any other application. After an item is added to the order, it is displayed, along with its price, in the shopping cart. The shopping cart also keeps a running total of the current price of the whole order. By clicking on an item in the shopping cart, the user can review all of the details for that particular item. Finally, the shopping cart contains a button for the user to proceed to checkout.

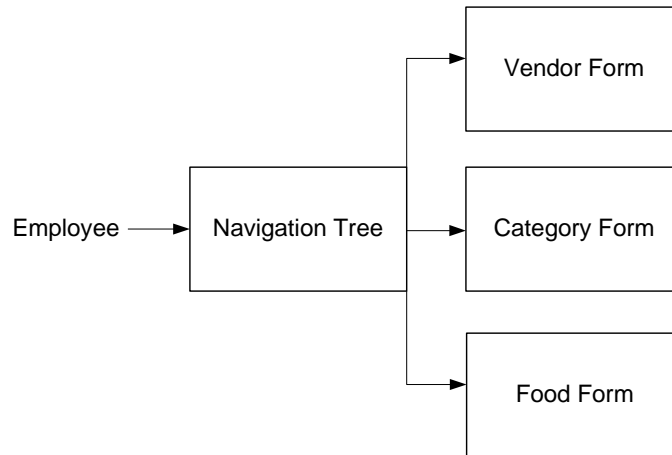
### **Level 3: The Checkout Form**

The checkout form is the user's last chance to verify that the contents of their order are correct before actually placing it. This form also provides fields for the user to supply all of the necessary checkout and delivery details (payment type, delivery address, etc.).

### **Level 2: Menu Management System Components**

In order to make use of the menu management system, the user must interact with the navigation tree, which uses a hierarchical tree structure to display all of the vendors, categories of foods, and specific food items stored in the system. When the user selects an item from this tree, they are able to edit the item using the appropriate form – a

Vendor Form if a vendor is selected, a Category Form if a category of foods is selected, and a Food Form if an individual food item is selected.



### **Level 3: The Navigation Tree**

The navigation tree is a 3-level (excluding the root) hierarchical arrangement, with each leaf corresponding to a form. At the first level are vendors, at level two categories of food, and at level 3 individual food items. When a leaf is selected, it brings up a form corresponding to the item at that leaf.

#### **Level 3.1: The Forms**

There are three types of forms in the menu management system - Vendor Forms, Category Forms, and Food Forms. The three forms are all similar, allowing the user to add, edit, and remove information relevant to the selected item. Where they differ is in the specific fields that the user is able to edit. After changes to any of the forms are saved, the necessary records in the database are updated.

### **Level 2: Order Retrieval System Components**

The simplest of the three components, the order retrieval system can be broken down into just two components. They are the summary panel, which displays a list of all currently active orders, and the order detail panel, which highlights just a single order. When the application first starts, the order details for the first order in the list are displayed. In order to view the details of a different order, the user must simply select it from the list in the summary panel.



### **Level 3: Summary Panel**

The summary panel, located on the left side of the screen, displays a list of all currently active orders, along with their delivery times and statuses. By changing the selected item in this list, the user is able to control the contents of the order detail panel.

### **Level 3: Order Detail Panel**

The order detail panel which contains a hierarchical tree structure for viewing all of the details related to the order which is currently selected in the summary panel. This component also contains a button to mark the order as processed and remove it from the list of active orders.

### **User Interface Design**

The user interface design principles can be broken into two groups. The interface in the web application is designed to limit free form user input, using mostly drop down menus, radio buttons and check boxes. This is done for two reasons – to simplify the ordering process as much as possible, and to limit SQL injection attempts. Free form input is necessary in the menu management component, however, as all of the values must be user supplied. The interface for this component contains traditional forms comprised of text fields and corresponding labels, along with save and discard buttons for each form.

### **Help System Design**

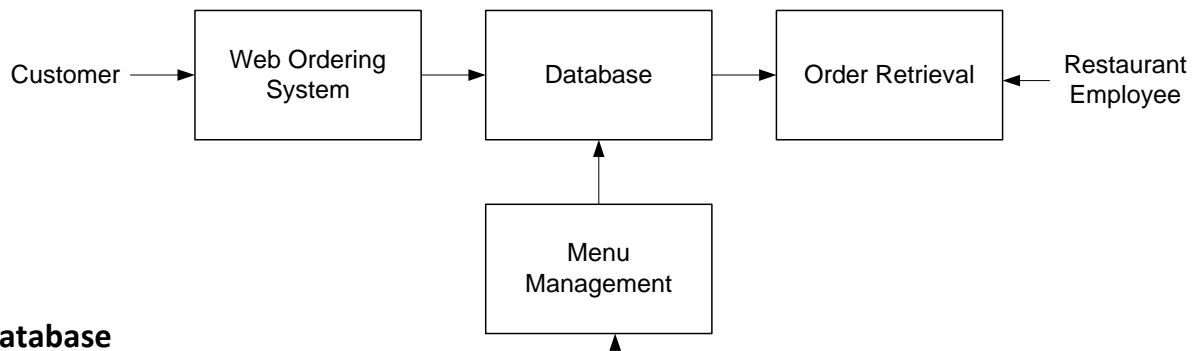
Due to the form-based nature of the applications, the design of the help systems will be minimal. In both the desktop and web applications, it will be accessed from the application's main menu and will open in a new window. Modeled after the typical help system design, it will be both searchable and include a navigation tree highlighting common topics. There will be a help page for each form type, describing the significance of each field on the page.



## **Chapter 4: Testing Design**

### **Testing Phases**

The structure of the system can be divided into three main logical components, plus the database, which is invisible to the end user. Each of these components must be tested individually, and the approaches which will be used for each component are described in the following sections.



### **Database**

Testing of the database component is very straightforward, and has actually already been mostly completed. The database was the first component designed and before beginning work on any of the applications, I wrote all of the SQL statements I expected to need and executed them directly, essentially isolating the database, using the psql client. By doing this I was able to reveal, and promptly fix a large percentage of the errors within the database itself.

### **Web Ordering System**

Testing of the web ordering system will be the most strenuous, as it is the component that will see the highest frequency of use and will be exposed to the most users, which leads to a higher potential of failure. Testing here will be divided into two phases. During normal use case testing I will execute all of the functions available through the web interface using a broad spectrum of reasonable values that a user would be expected to input. In addition to simply observing the in-application effects, I will also be monitoring and inspecting the JSON requests and responses (using Firebug) to make sure that everything is sent and received correctly.

In phase two I will perform exceptional use case testing, where I will artificially generate cases that shouldn't arise, but possibly could, and monitor how the system handles these cases. These cases fall into one of two categories – when the mistake happens in the browser and the server has to deal with it, or the other way around. I have tried to place appropriate checks on all values being sent back and forth so the system realizes something is wrong before going to the database and potentially changing the state of the system, but it will very important to see if there is anything I have not accounted for.

### **Menu Management System**

Testing of the menu management system will be very similar to that of the web ordering system, as I will first run test cases where the user supplies acceptable values, and afterwards test how the system responds to unexpected input.

### **Order Retrieval System**

Of all the components, testing of the order retrieval system will be the simplest. Since it is simply an interface to display the results of database queries and has no potential to change the state of the system, the only thing that really needs to be tested is how the system responds when a result set is not in the form it is expecting. This will be done by intentionally corrupting the database and analyzing the response of the order retrieval system.

### **Requirements Traceability**

In the requirements document, I specified the following functional requirements:

For the Web Ordering System:

- Create an account.
- Manage their account.
- Log in to the system.
- Navigate the restaurant's menu.
- Select an item from the menu.
- Customize options for a selected item.
- Add an item to their current order.
- Review their current order.

- Remove an item/remove all items from their current order.
- Provide delivery and payment details.
- Place an order.
- Receive confirmation in the form of an order number.

For the Menu Management System:

- Add a new/update/delete vendor to/from the menu.
- Add a new/update/delete food category to/from the menu.
- Add a new/update/delete food item to/from the menu.
- Add a new/update/delete option for a given food item.
- Update price for a given food item.
- Update default options for a given food item.
- Update additional information (description, photo, etc.) for a given food item.

For the Order Retrieval System:

- Retrieve new orders from the database.
- Display the orders in an easily readable, graphical way.
- Mark an order as having been processed and remove it from the list of active orders.

In order to assure thorough testing, I will have to generate cases in which each of these functions is performed, not just with a single input value, but an example of each possible class of input. This may seem tedious, but is absolutely necessary since the system is so heavily dependent on user input and must respond appropriately to anything the user may do. The good news is that because the system was design to contain only the absolutely necessary functionality, the testing of the different functions should flow smoothly into one another.

## **Testing Schedule**

Throughout all of the design and development phases, I have been performing unit tests on each component, assuring that it works properly before introducing it into the rest of the system, but I plan on beginning rigorous testing of the system starting in the first week of December. I will begin by putting the system through its normal paces as a normal user would be expected to. I will test the entire functionality of the system, but will do so following the normal logical flow, and only providing reasonable values for user input.

Once I am happy with how the system performs for normal use cases (which hopefully will be completed by December 8<sup>th</sup>), I will move on to testing the exceptional use cases. These are the use cases that would never be encountered by the average user, but, whether through confusion or malice, may come up. Two examples would be adding hundreds of items to an order or attempting to supply an SQL statement as a login credential.

In the final phase of testing, which I will begin on December 15<sup>th</sup>, I will carry out stress and performance testing. In this phase, I will bombard the system with an increasing number of HTTP requests and then measure response times and see at what point things begin to break down.

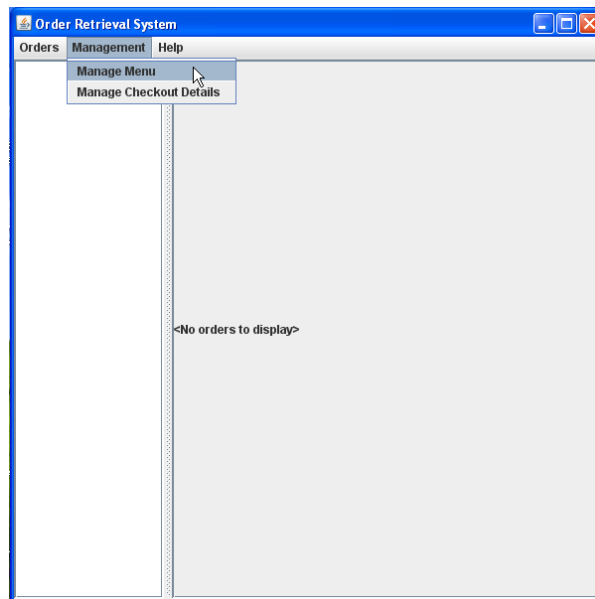
## **Recording Procedures**

The recording procedures I will use can be divided into two categories. For the use case testing, the tests will be designed in a “Pass/Fail” manner, making recording the results very simple. Each time a series of tests is run, the results will be recorded in a spreadsheet, where each outcome can be identified by the name of the test case along with the execution date and time.

For the stress and performance testing, the procedure will be similar. However, since these tests involve numerical results rather than simply “Pass/Fail”, I will be able to not only record the results, but to also perform analysis (both numerically and graphically) on them to get a better idea exactly what the numbers mean.

## Hardware and Web application Requirements

The testing to be performed will require no special hardware, but some specially designed web application may be used, particularly when performing stress testing. I plan on writing and utilizing a multi-threaded Java program to bombard the system with an increasing number of HTTP requests and then measure response times and see at what point things begin to break down. I also will be using the Firefox plug-in Firebug to monitor the JSON requests and responses.

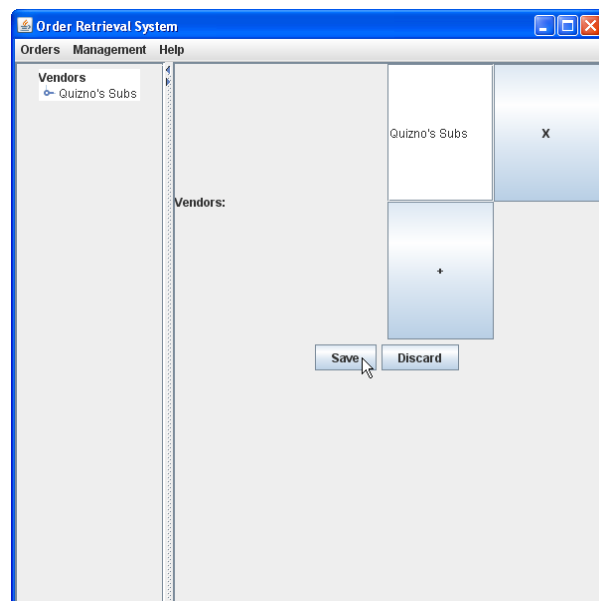
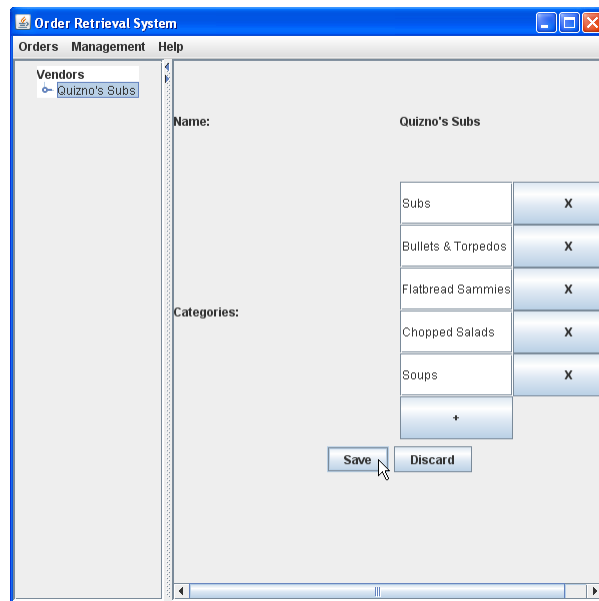


## Chapter 5: User Manual

### Introductory Manual

#### Using the Desktop Application

When the system is first installed, it contains no menu information. Therefore, the first step in using the system is to add vendor information. Start by launching the desktop client.



## **Managing Vendors**

Navigate to “Management” > “Manage Menu” in the toolbar at the top of the screen. This will bring up the menu management view. The menu management view provides a simple graphical interface for editing the content of the menu displayed to the customer.

Click the “+” button to add a new vendor, then supply the name of the vendor and click “Save”. This adds the vendor to the navigation tree on the left. If there will be multiple vendors using the system, repeat this process for each of them. If at any point you would like to modify the vendor names, do so from this screen.

### **Managing Categories**

Categories represent a related group of foods, usually what would be found as a menu heading. Categories must be associated with a vendor, so to add a new category select the appropriate vendor from the navigation tree on the left. In the category field, click the “+” button to add a new category then supply the category name and click “Save”. Now select the category you just created from the navigation tree. This will bring up a new form. If items from the category are available in multiple sizes, add these sizes to the “Sizes” field. Some categories allow customers to specify additional options. These options are specified in the “Options” field. Finally, the food items within this category must be added. This is done by clicking the “+” button in the “Food Items” field and supplying the name of the item. When you are finished, click “Save”.

## **Managing Foods**

To edit food details, select the food you wish to modify from the navigation tree. From this screen you can now modify the prices and default options for the selected food item. Repeat this process for all items you have added. The ordering system is now ready for use. If at any point you need to edit menu information you can return to these screens through the “Management” tab on the application toolbar.

## Managing Orders

To manage placed orders, navigate to “Orders” > “View Orders” on the application toolbar. You will now see a list of all currently active orders on the left side of the application.

The screenshot shows the 'Order Retrieval System' window with the 'Orders' menu selected. The left sidebar shows a tree view under 'Vendors' with 'Quizno's Subs' expanded, and 'Soups' selected. The main area is titled 'Soups' and contains a table with columns for 'Name' and a checkbox. The table lists various soup options with their respective sizes and checkboxes.

Name	
Cup	<input checked="" type="checkbox"/>
Bowl	<input checked="" type="checkbox"/>
+	
Crackers	<input checked="" type="checkbox"/>
+	
Broccoli Cheese Soup	<input checked="" type="checkbox"/>
Chicken Noodle Soup	<input checked="" type="checkbox"/>
Chili	<input checked="" type="checkbox"/>
+	

At the bottom of the main area are 'Save' and 'Discard' buttons.

The screenshot shows the 'Order Retrieval System' window with the 'Orders' menu selected. The left sidebar shows a list of orders. The main area displays the details for 'Order 1'.

**Order 1**

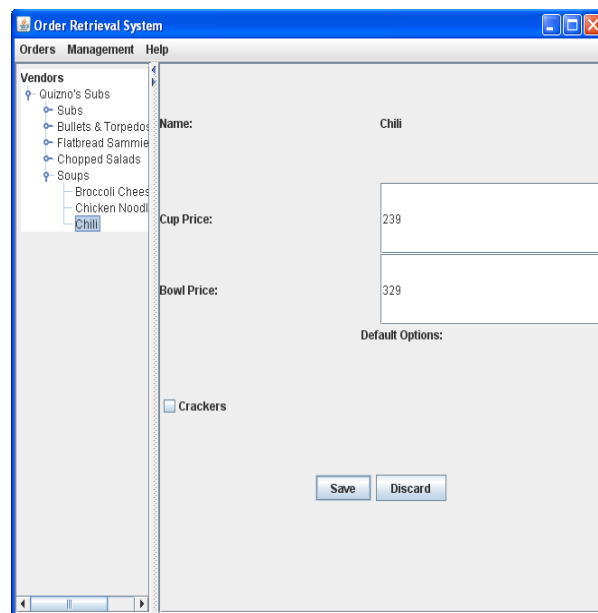
- Customer: Danny Jackowitz
- Payment Type: Cash
- Delivery Type: Delivery
- Delivery Time: 12:15
- Address: 123 Fictitious Lane
- Notes:

At the bottom right of the main area is a 'Processed' button.

**Items**

- Quizno's Subs Broccoli Cheese Soup
  - Vendor: Quizno's Subs
  - Category: Soups
  - Item: Broccoli Cheese Soup
  - Options
    - [None]





To view the details of a specific order, select that order from the list. Once an order has been prepared, mark it as processed by clicking the “Processed” button. As new orders are placed, they will be added to the list of active orders.

## Using the Web Ordering System

### Signing Up & Signing In

When a customer first visits the website, they must register before they can begin using the system. To do this, click on the “Sign Up” button on the main page. Then fill out the required fields and click “Register”. An e-mail containing your password will be sent to the specified account and you can then begin using the system.

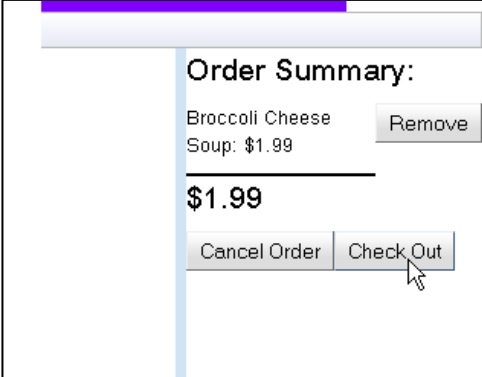
### Selecting Food Items

After signing in on the main page, click “Check Out Tonight’s Menu...” and select the category of food you would like to order. This will bring up an order form specific to that category.

On this form, select the food you would like to order, specifying the desired size and selecting any appropriate options, then click add to order. This will add the item to your “shopping cart”, located on the far right of the screen. Repeat this process for as many items as you would like to order.

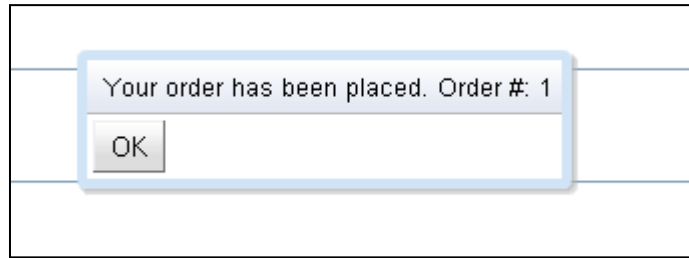
### Placing an Order

When you are finished adding items to your order, click “Check Out” from within the shopping cart.

A screenshot of a web application's "Order Summary" form. The form has a light blue header bar. Below the header, the text "Order Summary:" is displayed in bold. Underneath, the item "Broccoli Cheese Soup: \$1.99" is listed, followed by a "Remove" button. A horizontal line separates the item list from the total price, which is "\$1.99" in bold. At the bottom of the form, there are two buttons: "Cancel Order" and "Check Out". A mouse cursor is pointing at the "Check Out" button.

Order Summary:	
Broccoli Cheese Soup: \$1.99	<a href="#">Remove</a>
<hr/>	
<b>\$1.99</b>	
<a href="#">Cancel Order</a>	<a href="#">Check Out</a>

You will then be presented with one final form to complete. After supplying the required payment and delivery details, click “Place Order”.



You should then receive confirmation that your order was placed successfully, along with your order number, completing the ordering process.

### **Using the Help System**

The help systems in both the desktop application and online ordering system can be accessed through the “Help” tab on the application toolbar. In both cases, accessing this tab will present the user with a tutorial very similar to those presented above, instructing them, step-by-step, on how to complete the desired action. In addition to this in-application help, the help document for the desktop application will also be distributed with the system as a .pdf to make employee training easier.

- ✓ System Reference Manual
- ✓ Services (Alphabetical)
- ✓ Desktop Application
- ✓ Adding Categories
- ✓ Adding Foods
- ✓ Adding Vendors
- ✓ Deleting Categories
- ✓ Deleting Foods
- ✓ Deleting Vendors

- ✓ Marking Orders as Processed
- ✓ Modifying Categories
- ✓ Modifying Foods
- ✓ Modifying Vendors
- ✓ Retrieving Orders
- ✓ Web Ordering System
- ✓ Adding an Item to Your Order
- ✓ Checking Out
- ✓ Modifying an Item within Your Order
- ✓ Placing an Order
- ✓ Removing an Item from Your Order
- ✓ Selecting a Food Item
- ✓ Signing In
- ✓ Signing Up
- ✓ Viewing the Menu
- ✓ Error Recovery

Because the all of the information relevant to the functionality of the system is stored in the database, which is designed to maintain consistent state even when an application error occurs, the most effective way to recover from an error is to restart the application, either by closing and restarting the application (desktop application) or by refreshing the browser (web ordering system). In either case, a restart should have no negative effects on the database and the application should function normally.

If, however, the same error continually occurs, this likely indicates that the database has somehow been corrupted. Although measures have been taken to protect the consistency of the database, there may still be a slight chance of this corruption occurring through either user error or intentional tampering. If the database has become corrupted, a clean (destructive) reinstallation of the system must be performed. This can be done following the same process used to initially install the system.

## Installation

The installation process consists mainly of the creation of the PostgreSQL database that coordinates the entire system. At the present moment, this creation is handled by a script which must be run from within a PostgreSQL client (such as psql). This is not ideal, however, and when the time comes for the system to actually be distributed to a restaurant for testing, this script will be wrapped in an executable (most likely platform dependent), which the user will simply be able to double-click to complete the installation process.

## Glossary

**Web application** – an application, often comparable in functionality to a desktop application, which is accessed over a network, often via a web browser, rather than hosted on the user's machine

**Navigation tree** – an expandable and collapsible structure used to display large amounts of information in an easily navigable manner

**Application toolbar** – usually located at the very top of the window, the application toolbar provides the user with quick access to many common functions

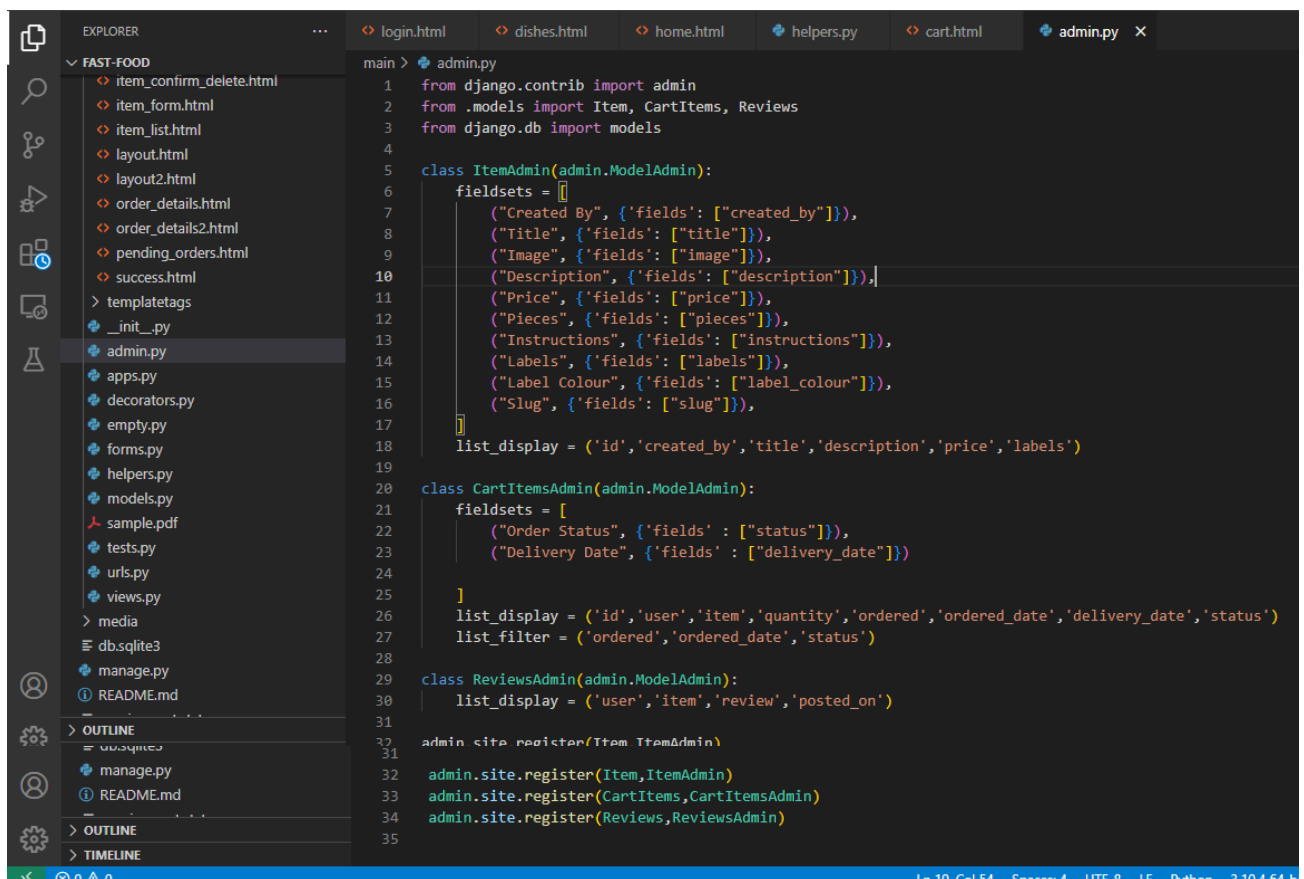
**In-application help** – most commonly accessed through the “Help” tab on the application toolbar, in-application help provides the user with a quick reference from directly within the application. This is contrasted with more in-depth offline documentation.

**Navigation tree** – an expandable and collapsible structure used to display large amounts of information in an easily navigable manner

**Destructive reinstallation** – this reverts the system back to its initial state, destroying all of the user's data. Generally this is a “when all else fails” scenario.

## Source code

### Admin.py



```
1 from django.contrib import admin
2 from .models import Item, CartItems, Reviews
3 from django.db import models
4
5 class ItemAdmin(admin.ModelAdmin):
6     fieldsets = [
7         ("Created By", {'fields': ["created_by"]}),
8         ("Title", {'fields': ["title"]}),
9         ("Image", {'fields': ["image"]}),
10        ("Description", {'fields': ["description"]}),
11        ("Price", {'fields': ["price"]}),
12        ("Pieces", {'fields': ["pieces"]}),
13        ("Instructions", {'fields': ["instructions"]}),
14        ("Labels", {'fields': ["labels"]}),
15        ("Label Colour", {'fields': ["label_colour"]}),
16        ("Slug", {'fields': ["slug"]}),
17    ]
18    list_display = ('id', 'created_by', 'title', 'description', 'price', 'labels')
19
20 class CartItemsAdmin(admin.ModelAdmin):
21     fieldsets = [
22         ("Order Status", {'fields': ["status"]}),
23         ("Delivery Date", {'fields': ["delivery_date"]})
24     ]
25     list_display = ('id', 'user', 'item', 'quantity', 'ordered', 'ordered_date', 'delivery_date', 'status')
26     list_filter = ('ordered', 'ordered_date', 'status')
27
28 class ReviewsAdmin(admin.ModelAdmin):
29     list_display = ('user', 'item', 'review', 'posted_on')
30
31 admin.site.register(Item, ItemAdmin)
32 admin.site.register(CartItems, CartItemsAdmin)
33 admin.site.register(Reviews, ReviewsAdmin)
```

## Apps.py

```
Terminal  Help  apps.py - fast-food - Visual Stud

login.html  dishes.html  home.html  helpers.py

main > apps.py > ...
1  from django.apps import AppConfig
2
3
4  class MainConfig(AppConfig):
5      name = 'main'
6
```

## Decorators.py

```
Terminal  Help  decorators.py - fast-food - Visual Studio Code

login.html  dishes.html  home.html  helpers.py  cart.html  decorators.py X

main > decorators.py > admin_required > wrapper_func
1  from django.http import HttpResponse
2  from django.shortcuts import redirect
3  from django.contrib.auth.models import Group
4  from .models import *
5
6  def admin_required(view_func):
7      def wrapper_func(request, *args, **kwargs):
8          group = Group.objects.get(user = request.user)
9          if group.name == 'admin_owner':
10             return view_func(request, *args, **kwargs)
11         else:
12             return redirect('/')
13     return wrapper_func
```

## Models.py

```
login.html  dishes.html  home.html  helpers.py  models.py X
in > models.py > ...
1  from django.db import models
2  from django.conf import settings
3  from django.shortcuts import reverse
4  from django.utils import timezone
5  from django.contrib.auth.models import User
6
7  class Item(models.Model):
8      LABELS = (
9          ('BestSeller', 'BestSeller'),
10         ('New', 'New'),
11         ('Spicy🔥', 'Spicy🔥'),
12     )
13
14     LABEL_COLOUR = (
15         ('danger', 'danger'),
16         ('success', 'success'),
17         ('primary', 'primary'),
18         ('info', 'info')
19     )
20     title = models.CharField(max_length=150)
21     description = models.CharField(max_length=250, blank=True)
22     price = models.FloatField()
23     pieces = models.IntegerField(default=6)
24     instructions = models.CharField(max_length=250, default="Jain Option Available")
25     image = models.ImageField(default='default.png', upload_to='images/')
26     labels = models.CharField(max_length=25, choices=LABELS, blank=True)
27     label_colour = models.CharField(max_length=15, choices=LABEL_COLOUR, blank=True)
28     slug = models.SlugField(default="sushi_name")
29     created_by = models.ForeignKey(User, on_delete=models.CASCADE)
30
31     def __str__(self):
32         return self.title
```



```
login.html  dishes.html  home.html  helpers.py  models.py X
main > models.py > ...
34     def get_absolute_url(self):
35         return reverse("main:dishes", kwargs={
36             'slug': self.slug
37         })
38
39     def get_add_to_cart_url(self):
40         return reverse("main:add-to-cart", kwargs={
41             'slug': self.slug
42         })
43
44     def get_item_delete_url(self):
45         return reverse("main:item-delete", kwargs={
46             'slug': self.slug
47         })
48
49     def get_update_item_url(self):
50         return reverse("main:item-update", kwargs={
51             'slug': self.slug
52         })
53
54     class Reviews(models.Model):
55         user = models.ForeignKey(User, on_delete = models.CASCADE)
56         item = models.ForeignKey(Item, on_delete = models.CASCADE)
57         rslug = models.SlugField()
58         review = models.TextField()
59         posted_on = models.DateField(default=timezone.now)
60
61         class Meta:
62             verbose_name = 'Review'
63             verbose_name_plural = 'Reviews'
64
65     def str (self):
```

## urls.py

```
login.html dishes.html home.html helpers.py models.py tests.py
main > urls.py > ...
1  from django.urls import path
2  from . import views
3  from .views import (
4      MenuListView,
5      menuDetail,
6      add_to_cart,
7      get_cart_items,
8      order_item,
9      CartDeleteView,
10     order_details,
11     admin_view,
12     item_list,
13     homee,
14     homeee,
15     pending_orders,
16     ItemCreateView,
17     ItemUpdateView,
18     ItemDeleteView,
19     update_status,
20     add_reviews,
21 )
22
23 app_name = "main"
24
25 urlpatterns = [
26     path('', MenuListView.as_view(), name='home'),
27     path('dishes/<slug>', views.menuDetail, name='dishes'),
28     path('item_list/', views.item_list, name='item_list'),
29     path('item/new/', ItemCreateView.as_view(), name='item-create'),
30     path('item-update/<slug>', ItemUpdateView.as_view(), name='item-update'),
31     path('item-delete/<slug>', ItemDeleteView.as_view(), name='item-delete'),
32     path('add-to-cart/<slug>', views.add_to_cart, name='add-to-cart'),
33     path('cart/', views.get_cart_items, name='cart')
```

```

3 app_name = "main"
4
5 urlpatterns = [
6     path('', MenuListView.as_view(), name='home'),
7     path('dishes/<slug>', views.menuDetail, name='dishes'),
8     path('item_list/', views.item_list, name='item_list'),
9     path('item/new/', ItemCreateView.as_view(), name='item-create'),
10    path('item-update/<slug>', ItemUpdateView.as_view(), name='item-update'),
11    path('item-delete/<slug>', ItemDeleteView.as_view(), name='item-delete'),
12    path('add-to-cart/<slug>', views.add_to_cart, name='add-to-cart'),
13    path('cart/', views.get_cart_items, name='cart'),
14    path('developer/', views.developer),
15    path('index/', views.homeee),
16    path('success/', views.homeeee),
17    #path('profile', views.profile),
18    path('remove-from-cart/<int:pk>', CartDeleteView.as_view(), name='remove-from-cart'),
19    path('ordered/', views.order_item, name='ordered'),
20    path('order_details/', views.order_details, name='order_details'),
21    path('admin_view/', views.admin_view, name='admin_view'),
22    path('pending_orders/', views.pending_orders, name='pending_orders'),
23    path('admin_dashboard/', views.admin_dashboard, name='admin_dashboard'),
24    path('update_status/<int:pk>', views.update_status, name='update_status'),
25    path('postReview', views.add_reviews, name='add_reviews'),
26 ]
27

```

## Views.py

```

login.html  dishes.html  home.html  layout.html  views.py  X
h > views.py > order_details
@login_required
def order_details(request):
    items = CartItems.objects.filter(user=request.user, ordered=True, status="Active").order_by('-ordered_date')
    cart_items = CartItems.objects.filter(user=request.user, ordered=True, status="Delivered").order_by('-ordered_date')
    bill = items.aggregate(Sum('item__price'))
    number = items.aggregate(Sum('quantity'))
    pieces = items.aggregate(Sum('item__pieces'))

    # total = items.aggregate(Sum('item__price_sum' + 'item__price_sum'))
    total = bill.get("item__price_sum")
    count = number.get("quantity__sum")
    total_pieces = pieces.get("item__pieces__sum")
    context = {}
    context['items'] = items,
    context['cart_items'] = cart_items,
    context['total'] = total,
    context['count'] = count,
    context['total_pieces'] = total_pieces

    return render(request, 'main/order_details.html', context)

@login_required(login_url='/accounts/login/')
@admin_required
def admin_view(request):
    cart_items = CartItems.objects.filter(item__created_by=request.user, ordered=True, status="Delivered").order_by('-ordered_date')
    context = {}
    context['cart_items'] = cart_items,
    return render(request, 'main/admin_view.html', context)

@login_required(login_url='/accounts/login/')
@admin_required
def item_list(request):

```

```

login.html  dishes.html  home.html  layout.html  views.py  X
in > views.py > order_details
1  @login_required
2  def add_reviews(request):
3      if request.method == "POST":
4          user = request.user
5          rslug = request.POST.get("rslug")
6          item = Item.objects.get(slug=rslug)
7          review = request.POST.get("review")
8
9          reviews = Reviews(user=user, item=item, review=review, rslug=rslug)
10         reviews.save()
11         messages.success(request, "Thankyou for reviewing this product!!")
12         return redirect(f"/dishes/{item.slug}")
13
14     class ItemCreateView(LoginRequiredMixin, CreateView):
15         model = Item
16         fields = ['title', 'image', 'description', 'price', 'pieces', 'instructions', 'labels', 'label_colour', 'slug']
17
18         def form_valid(self, form):
19             form.instance.created_by = self.request.user
20             return super().form_valid(form)
21
22     class ItemUpdateView(LoginRequiredMixin, UserPassesTestMixin, UpdateView):
23         model = Item
24         fields = ['title', 'image', 'description', 'price', 'pieces', 'instructions', 'labels', 'label_colour', 'slug']
25
26         def form_valid(self, form):
27             form.instance.created_by = self.request.user
28             return super().form_valid(form)
29
30         def test_func(self):
31             item = self.get_object()
32             if self.request.user == item.created_by:

```

```

1  Help  views.py - fast-food - Visual Studio Code
login.html  dishes.html  home.html  layout.html  views.py  X
in > views.py > order_details
1  from django.shortcuts import render, get_object_or_404, redirect
2  from .models import Item, CartItems, Reviews
3  from django.contrib import messages
4  from django.views.generic import (
5      ListView,
6      DetailView,
7      CreateView,
8      UpdateView,
9      DeleteView,
10 )
11 from django.utils import timezone
12 from django.contrib.auth.decorators import login_required
13 from django.contrib.auth.mixins import LoginRequiredMixin, UserPassesTestMixin
14 from .decorators import *
15 from django.db.models import Sum
16
17 class MenuListView(ListView):
18     model = Item
19     template_name = 'main/home.html'
20     context_object_name = 'menu_items'
21
22 def menuDetail(request, slug):
23     item = Item.objects.filter(slug=slug).first()
24     reviews = Reviews.objects.filter(rslug=slug).order_by('-id')[:7]
25     context = {
26         'item' : item,
27         'reviews' : reviews,
28     }
29     return render(request, 'main/dishes.html', context)
30
31 @login_required
32 def add_reviews(request):

```

# Layout.html

```
login.html dishes.html home.html layout.html X
> templates > main > layout.html > html > body > nav.navbar.pt-2.bggg.navbar-expand-lg.navbar-light. > span.mx-5. > img
<body>
  <nav class="navbar pt-2 bggg navbar-expand-lg navbar-light ">
    <span class="mx-5 ">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNav">
    <ul class="navbar-nav mr-auto">
    </ul>
    <ul class="mx-2 nav navbar-nav ">
      <li class="mx-2 fa fa-home"><a href="{% url 'main:home' %}" class="mr-2 text-light">Home</a>
      <li class="mx-2 "><a href="/developer" class="mr-2 text-light">Developers</a></li>

      {% if user.is_authenticated %}
      {% if request.user|has_group:"admin_owner" %}
        <li class="mx-2"><a href="{% url 'main:admin_dashboard' %}" class="mr-2 text-light">Dash
        <li>
          <form class="logout-link bg-danger" action="{% url 'accounts:logout' %}" method="pos
            {% csrf_token %}
            <button type="submit" class="mr-2">Logout</button>
          </form>
        </li>
      {% else %}
        <li class="mx-2"><a href="{% url 'main:cart' %}" class="mr-2 text-light">My Cart</a>
        <li class="mx-2"><a href="{% url 'main:order_details' %}" class="mr-2 text-light">Your C
```

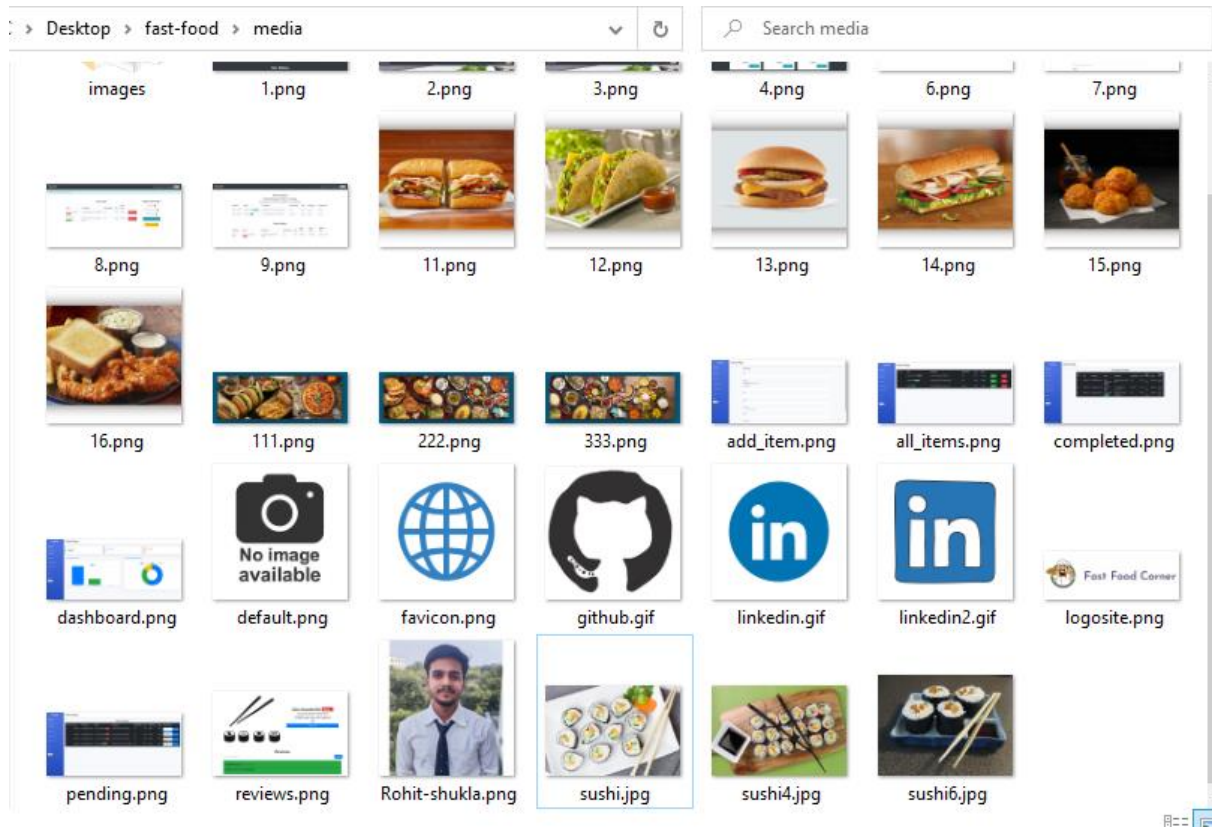
```
login.html dishes.html home.html layout.html X
main > templates > main > layout.html > html > body > nav.navbar.pt-2.bggg.navbar-expand-lg.navbar-light. > span.mx-5. > img
60      {% csrf_token %}
61      <button type="submit" class="mx-2 mr-2 text-white bg-danger">Logout</button>
62    </form>
63  </li>
64    {% endif %}
65  {% else %}
66  <li>
67    <form class="login-link " action="{% url 'accounts:login' %}" method="post">
68      {% csrf_token %}
69      <button type="submit" class="mx-2 mr-2 btn text-white bg-primary">Login</button>
70    </form>
71  </li>
72  <li>
73    <form class="logout-link" action="{% url 'accounts:signup' %}" method="post">
74      {% csrf_token %}
75      <button type="submit" class="mx-2 mr-2 btn text-white btn-primary">Signup</button>
76    </form>
77  </li>
78    {% endif %}
79  </ul>
80 </div>
81 </nav>
82 <main role="main">
83   {% if messages %}
84   {% for message in messages %}
85     <div class="alert alert-{{ message.tags }}">
86       {{ message }}
87     </div>
88   {% endfor %}
89   {% endif %}
90   {% block content %}{% endblock %}
91 </main>
```

## Cart.html

```
login.html dishes.html home.html index.html cart.html X
main > templates > main > cart.html > div.cart > div.container > div.row > div.col-sm-9.pl-8.mt-5 >
/
8   {% if cart_items %}
9   <div class="table-responsive">
10  <table class="border table table-light">
11  <thead class="thead-dark">
12  <tr>
13  <th scope="col">Name</th>
14  <th scope="col">Description</th>
15  <th scope="col">Price</th>
16  <th scope="col">pdf</th>
17  <th scope="col">Pieces</th>
18  <th scope="col"></th>
19  </tr>
20  </thead>
21  <tbody>
22  {% for cart in cart_items %}
23  <tr>
24  <td>
25  {{ cart.item.title }}
26  <span class="badge badge-{{ cart.item.label_colour }}">
27  >{{ cart.item.labels }}</span>
28  </td>
29  <td>{{ cart.item.description }}</td>
30  <td>{{ cart.item.price }}</td>
31  <td>{{ cart.item.pieces }}</td>
32  <td>
33  <a class="btn btn-danger" href="..\main\sample.pdf">Pdf</a>
34  </td>
35  <td>
36  <a
37  class="btn btn-danger"
38  href="{% url 'main:remove-from-cart' cart.id %}"
39
```

```
dishes.html home.html index.html cart.html X
main > cart.html > div.cart > div.container > div.row > div.col-sm-9.pl-8.mt-5 >
<b>Number of Boxes: </b>{{ count }}
</li>
<li
class="list-group-item list-group-item-light text-center text-dark"
>
<b>Total Pieces:</b> {{ total_pieces }}
</li>
<li class="list-group-item list-group-item-light text-center">
<a class="btn btn-primary text-white" href="/"
>Continue Buying 😊</a>
</li>
<form
class="text-center border btn text-white"
action="success"
method="POST"
>
{% csrf_token %}
<script
src="https://checkout.razorpay.com/v1/checkout.js"
data-key="rzp_live_35X6zFXeQ10yA9"
data-amount="100"
data-currency="INR"
data-order_id="{{ payment.id }}"
data-buttontext="Pay Now 💰"
data-name="Rohit shukla"
data-description="Software Developer"
data-image="..\media\Rohit-shukla.png"
data-prefill.name="Rohit shukla"
data-prefill.email="rawstar1090@gmail.com"
data-theme.color="#F37254"
></script>
</form>
```

## Images:



**Thank You**