FoodHack – Food Ordering Website

A PROJECT REPORT
Submitted By

Atif Ali
University Roll No- 2100290140040

Brahm Dev Pandey
University Roll No- 2100290140052

Anuj
University Roll No- 2100290140031

Nikhil Aggarwal
University Roll No- 2100290140095

Submitted in partial fulfillment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATIONS

Under the Supervision of
Dr. Vipin Kumar
Associate Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

CERTIFICATE

Certified that **Atif Ali (University Roll No.- 2100290140040)** have carried out the project work having "**FoodHack- Food Ordering Website**" for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Atif Ali
University Roll No. 2100290140040
Brahm Dev Pandey
University Roll No- 2100290140052
Anuj
University Roll No- 2100290140031
Nikhil Aggarwal
University Roll No- 2100290140095

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Vipin Kumar
Associate Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Signature of Internal Examiner **Signature of External Examiner**

Dr. Arun Kumar Tripathi
Head, Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

FoodHack is a website which will provide the facility to order the food online and get it delivered at your own place or receive it without standing in the queue for ordering your food.
This Website provides facility to order your meal online in advance and get it delivered at your own place.
The purpose of Online Food Ordering System is to automate the existing manual system by the help of computerized equipment's and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with.
It provides a lightweight, interactive interface so that it can be used easily on all types devices.
Customer can choose more than one item to make an order and can view order details before logging off. The order confirmation is sent to the customer. The order is placed in the queue and updated in the database and returned in real time. This system assists the staff to go through the orders in real time and process it efficiently with minimal errors.
Online Food Ordering System, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization in better utilization of resources. The organization can maintain computerized records without redundant entries. That means that one need not be distracted by information that is not relevant, while being able to reach the information.

ACKNOWLEDGEMENT

Success in life is never attained single handedly. My deepest gratitude goes to my project supervisor, **Dr. Vipin Kumar** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi, Professor and Head, Department of Computer Applications**, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Atif Ali (2100290140040)

Brahm Dev Pandey (2100290140052)

Anuj (2100290140031)

Nikhil Aggarwal (2100290140095)

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Overview

Every organization, whether big or small, has challenges to overcome and managing the information of Category, Food Item, Order, Payment, Confirm Order. Every Online Food Ordering System has different Food Item needs; therefore, we design exclusive employee management systems that are adapted to your managerial requirements. This is designed to assist in strategic planning, and will help you ensure that your organization is equipped with the right level of information and details for your future goals. Also, for those busy executive who are always on the go, our systems come with remote access features, which will allow you to manage your workforce anytime. These will ultimately allow you to better manage resources.

1.2 Objective

The main objective of this project is to manage the details of Food Item, Category, Customer, Order, Confirm Order. It manages all the information about Food Item, Payment, Confirm Order, Foot Item. The project is totally built at administrative end and thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work for managing the Food Item, Category, Payment, Customer. It tracks all the details about the Customer, Order, Confirm Order. The purpose of this project is to automate the existing manual system by the help of computerized equipment’s and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with.

It provides a lightweight, interactive interface so that it can be used easily on all types devices. Customer can choose more than one item to make an order and can view order details before logging off. The order confirmation is sent to the customer. The order is placed in the queue and updated in the database and returned in real time. This system assists the staff to go through the orders in real time and process it efficiently with minimal errors.

1.3 Scope

Our project aims at business process automation, i.e. we have tried to computerize various processes of Online Food Ordering System.

1.3.1 In computer system the person has to fill the various forms & number of copies of the forms can be easily generated at a time.

1.3.2 To assist the staff in reducing the efforts spent on their respective working areas.

1.3.3 To utilize resources in an efficient manner by increasing their productivity through automation.

1.3.4 The system generates types of reports that can be used for various purposes.

1.3.5 It satisfies the user requirement

1.3.6 Be easy to understand by the user and operator

1.3.7 Delivered on schedule withing the budget

1.4 Functionalities

Functionalities provided by Online Food Ordering System are as follows:

1.4.1 Provides the searching facilities based on various factors. Such as Food Item, Customer, Order, Confirm Order.

1.4.2 Online Food Ordering System also manage the Payment details online for Order details, Confirm Order details, Food Item.

1.4.3 It tracks all the information of Category, Payment, Order etc

1.4.4 Shows the information and description of the Food Item, Customer

1.4.5 To increase efficiency of managing the Food Item, Category

1.4.6 It deals with monitoring the information and transactions of Order

1.4.7 Manage the information of Food Item

1.4.8 Manage the information of Order

CHAPTER 2: REQUIREMENTS OF THE ANALYSIS OF THE SYSTEM

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.

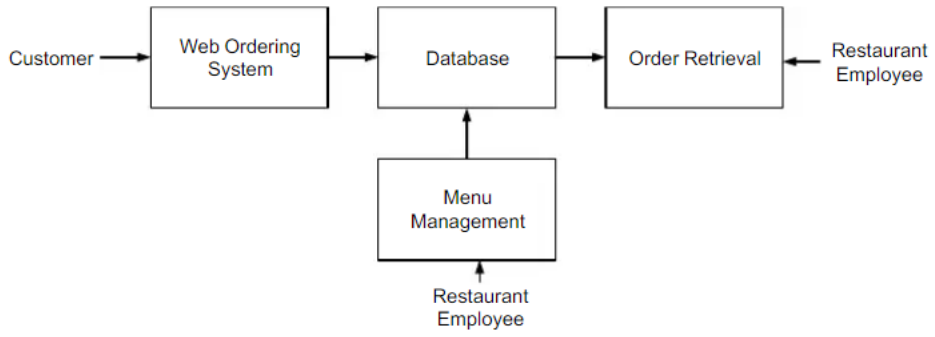

Figure: 2.1

FUNCTIONAL REQUIREMENTS

As can be seen in the system model diagramed 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.

Ordering System

- Create an Account

- Log into the Website
- Navigate the Restaurant menu
- Select an item from the menu
- Customized options for selected item
- Add item to current order
- Remove items/ remove all items from current order
- Provide delivery and payment details

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 additional information for a given food item

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

2.1 FEASIBILITY ANALYSIS

After studying and analyzing all the existing and requires functionalities of the system, the next task is to do the feasibility study for the project. Feasibility study includes consideration of all the possible ways to provide a solution to a given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirements.

2.1.1 Economical Feasibility

For the economic feasibility, Economic analysis or cost/benefits analysis is most frequently used technique the effectiveness of a proposed system. it is a procedure to determine the benefits and saving those are expected from the proposes system and compare them with cost. If the benefits outweigh the costs, a decision is taken to design and implement the system. otherwise, further justification or alternative in proposed system will have to be made if it is to have a chance of being approved this is ongoing effort that improves in accuracy at each phase of a system life cycle

2.1.2 Technical Feasibility

This included the study of function, performance and constraints that may affect the ability to achieve an acceptable system. For this feasibility study, we studied complete functionalities to be provided in the system, as described in the System Requirement Specification (SRS), and checked if everything was possible using different type of front end and backend platform.

The technical requirement for the system is economic and it does not use any other additional Hardware and software. Technical evaluation must also assess whether the existing systems can be upgraded to use the new technology and whether the organization has the expertise to use it.

2.1.3 Operational Feasibility

No doubt the technically growing world needs more enhancement in technology, this application is very user friendly and all inputs to be taken all self-explanatory even to a layman. As far our study is concerned, the clients will be comfortable and happy as the system has cut down their loads and bring the young generation to the same virtual world they are growing drastically.

The system working is quite easy to use and learn due to its simple but attractive interface. User requires no special training for operating the system. Technical performance include issues such as determining whether the system can provide the right information for the client and admin.

2.2 HARDWARE AND SOFTWARE REQUIREMENT

Hardware Details

1 GHz CPU

512 MB RAM

300 MB Disk Space

Software Details

Web Browser

Visual Studio Code

NodeJS

Google Firebase Firestore

2.3 THE WEB ORDERING SYSTEM

Users of the web ordering system, namely restaurant customers, must be provided the following functionality

- Create an 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 payment details
- Place an order

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.

2.3.1 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.

2.3.2 Order Retrieval System

Of these 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 funcitons:

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

2.3.3 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 isaccomplished 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 deals is done in a similar manner. The user is presented with a form and must complete the required fields, which include both dropdown 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 and SQL injection attempt.

2.4 NON-FUNCTIONAL REQUIREMENTS

2.4.1 Portability

System running on one platform can easily be converted to run on another platform.

2.4.2 Reliability

The ability of the system to behave consistently in a user-acceptable manner when operating within the environment for which the system was intended.

2.4.3 Availability

The system should be available 24x7, meaning the user can access it using a web browser, only restricted by the down time of the server on which the system runs.

2.4.4 Maintainability

A commercial database is used for maintaining the database and the application server takes care of the site.

2.4.5 Security

Secure access of confidential data

CHAPTER 3: SOFTWARE DEVELOPMENT LIFE CYCLE MODEL

Waterfall Model

The waterfall model is a well-known structured methodology for software 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 is:

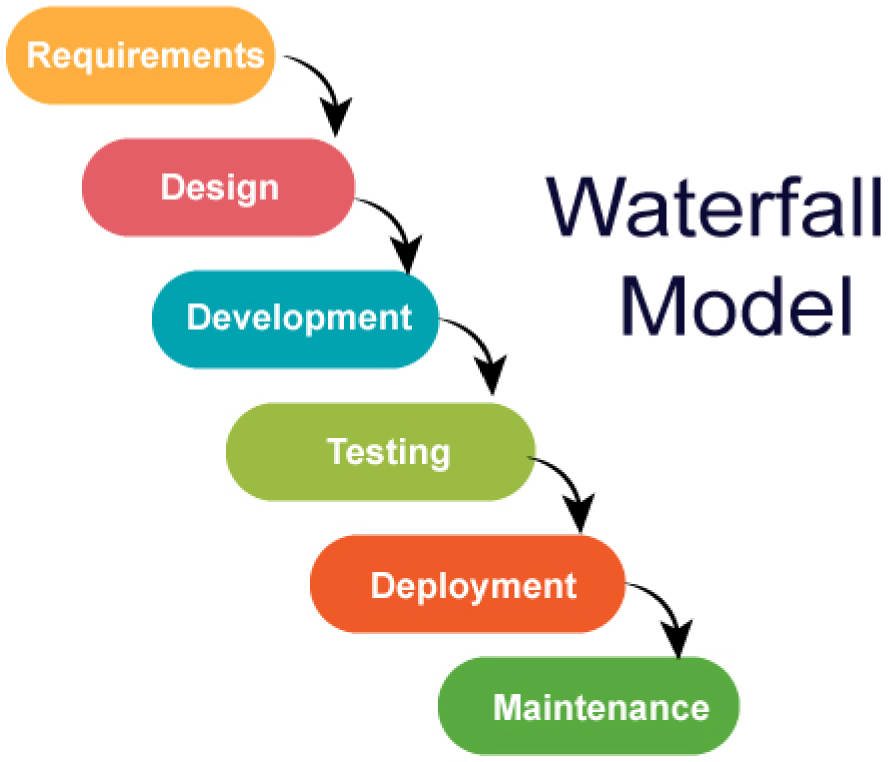

Figure: 3.1

Requirements Gathering & Analysis:

In this phase, the project title had been selected. The project title for the system was FoodHack. This project starting with brainstorming ideas with supervisor and proposed the title of the project. An abstract and description of the project module has also been done and attached. Besides, the Gantt chart also needed as a guideline and references for the project. This phase is to analyze the existing system and the article of the techniques or method that will be used for this project. In this phase also get all the requirements that are needed to design and develop the new system. Based on the collection of information through article, method and technique that is suitable been decided.

System Design:

The requirements documented in previous phase are studied in this phase and the system design is prepared. All the data or requirement obtained during planning and analysis phase transformed into the design

Implementation:

With inputs from system design, the system is developed in several units. Then the units are tested. This phase is where the design will implement into the coding. The system will develop regarding the user and system requirement. In this project, to develop the system will be use Visual Studio Code to code, Bootstrap as framework and Google Firebase Firestore as database and NodeJS for backend. This phase is a critical phase because user part needed to fulfil and to make sure the objectives accomplish.

Integration & Testing:

The units of the program developed in previous phase are integrated into a system. Then the whole system is tested. This testing phase will test the system to check the error and ensure the function run well as a whole system. Any error or bugs will be fixed and repeated testing the system until all the function can be use.

Deployment of the system:

This phase is when the system has successfully done and fulfil all the objective. The system can be deployed and finally the system will publish to the user for use as their need.

Maintenance:

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

CHAPTER 4: SYSTEM DESIGN

4.1 Introduction

System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system.

System Analysis is the process that decomposes a system into its component pieces for the purpose of defining how well those components interact to accomplish the set requirements.

The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

MVC Design Pattern

The Model View Controller (MVC) design pattern specifies that an application consist of a data model, presentation information, and control information.

MVC mostly relates to the user Interface/interaction layer of an application.

In the MVC pattern the user sees the View which is updated by the model which turn manipulated by the controller.

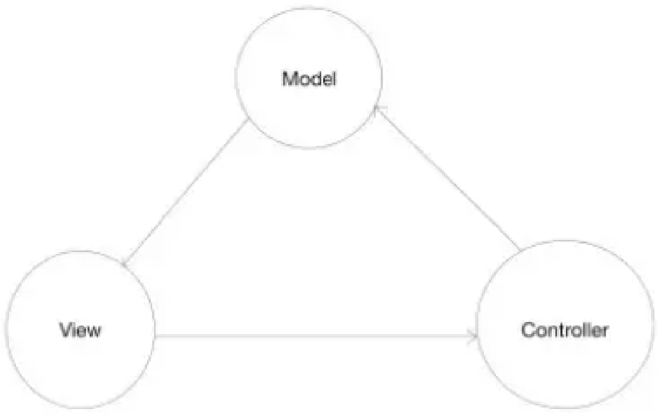

Figure: 4.1

MVC Pattern

- The **Model** contains only the pure application data, it contains no logic describing how to present the data to a user. They are the parts of the application that implement the logic for the application's data domain. They retrieve and store model state in a database.
- The **View** presents the model's data to the user. The view can only be used to access the model's data. They are components that display the application's user interface (UI).
- The **Controller** exists between the view and the model. It listens to events triggered by the view and executes the appropriate commands. They are the components that handle user interaction, work with the model, and ultimately select a view to render that displays UI.

4.2 System Flowchart

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

The flowchart shows the steps as boxes of various kinds, and their order byconnecting the boxes with arrows. This diagrammatic representation illustrates asolution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

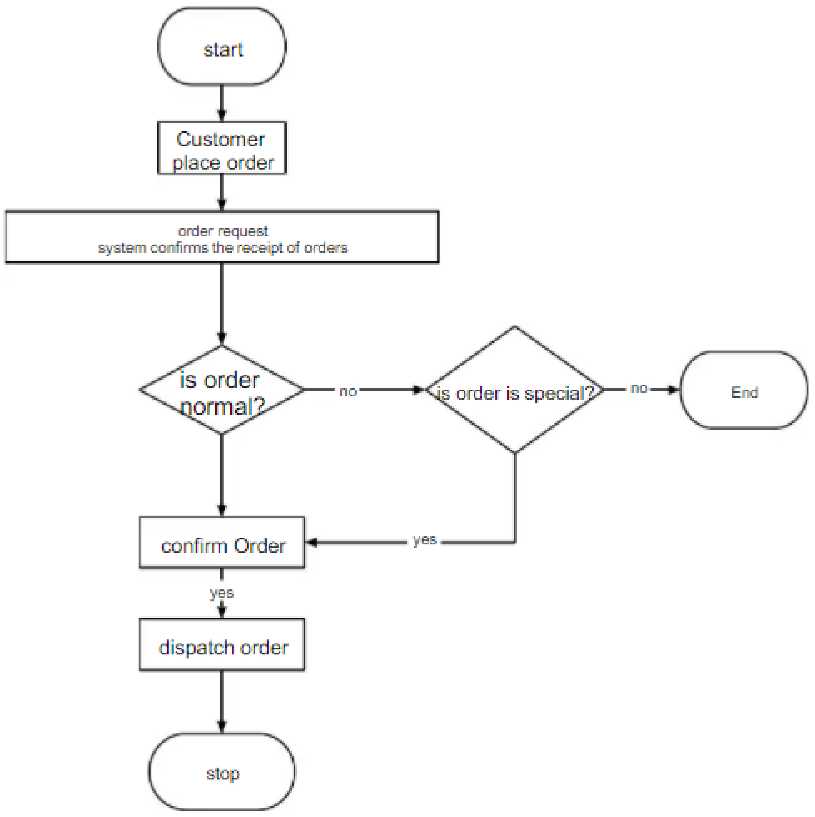

Figure: 4.2

4.3 ER DIAGRAM

Entity is represented by the rectangle shape. The entity will be our database table of Online Food Ordering System later on.

Attribute is represented by the oval shape. This will be the columns or fields of each table in the Online Food Ordering System.

Relationship is represented by diamond shape. This will determine the relationships among entities. This is usually in a form of primary key to foreign key connection.

We will follow the 3 basic rules in creating the ER Diagram.

1. Identify all the entities.
2. Identify the relationship between entities and
3. Add meaningful attributes to our entities.

Step 1. Entities

- User
- Product
- Order
- Payment
- Admin

Step 2. After we have specified our entities, it is time now to connect or establish a relationship among the entities.

- The user have Order
- Order have some products
- Order has some payment

Step 3. The last part of the ERD process is to add attributes to our entities.

User Entity has following attributes

- Email
- Password
- FName
- Id

Product Entity has following attributes

- PID
- PName
- Price
- Image

Order Entity has following attributes

- PName
- Email
- OID
- Price
- Quantity

Payment Entity has following attributes

- PID
- Email
- Bank
- Card
- CCV

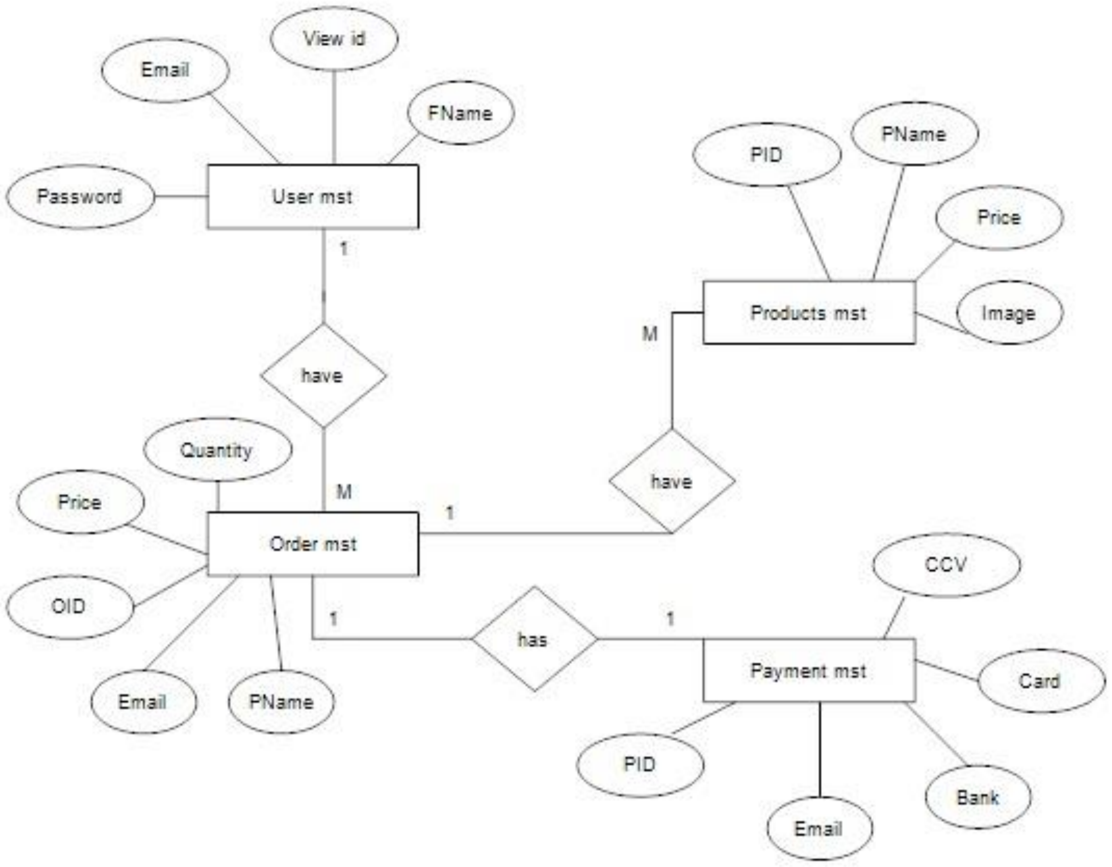
View id
Email
FName
Password
User mst
1
have
M
PID
PName
Price
Products mst
Image
M
Quantity
Price
Order mst
1
have
OID
Email
PName
1
has
1
CCV
Payment mst
Card
PID
Email
Bank

Figure: 4.3

4.4 USE CASE DIAGRAM

Use-case diagrams model the behavior of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors.

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

Purposes of a use case diagram given below:

1. It gathers the system's needs.
2. It depicts the external view of the system.
3. It recognizes the internal as well as external factors that influence the system.
4. It represents the interaction between the actors.

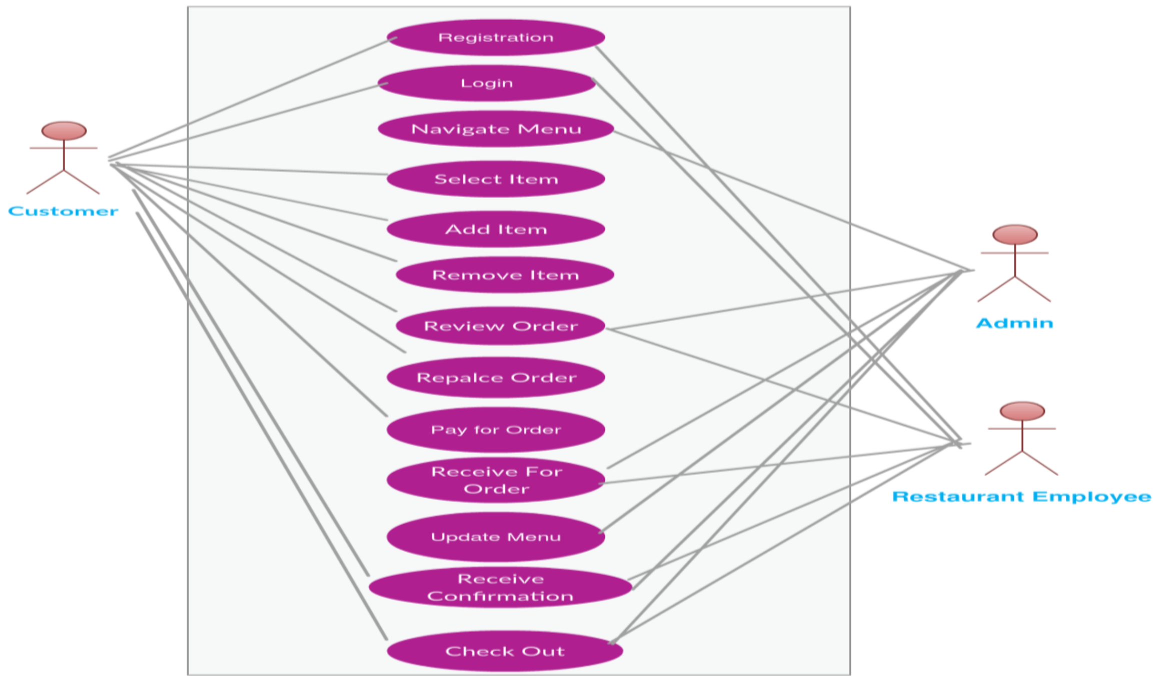

Figure: 4.4

4.5 ACTIVITY DIAGRAM

An Activity Diagram is a behavioural diagram. It depicts the behaviour of a system. Its primary use is to depict the dynamic aspects of a system. The dynamic aspect of a system specifies how the system operates to attain its function.

It is basically a flowchart to represent the flow from one activity to another activity. Activity Diagrams are not exactly flowcharts as they have some additional capabilities including branching, parallel flow, etc.

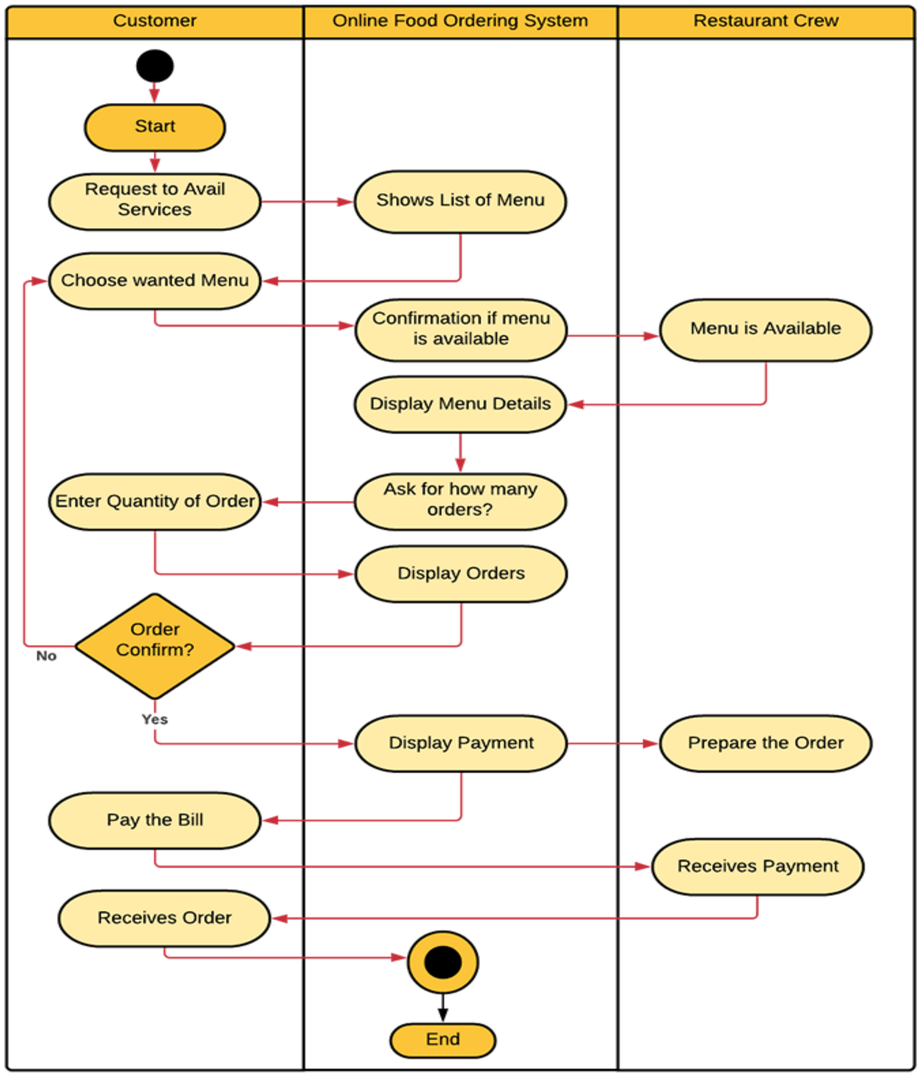

Figure: 4.5

4.6 SEQUENCE DIAGRAM

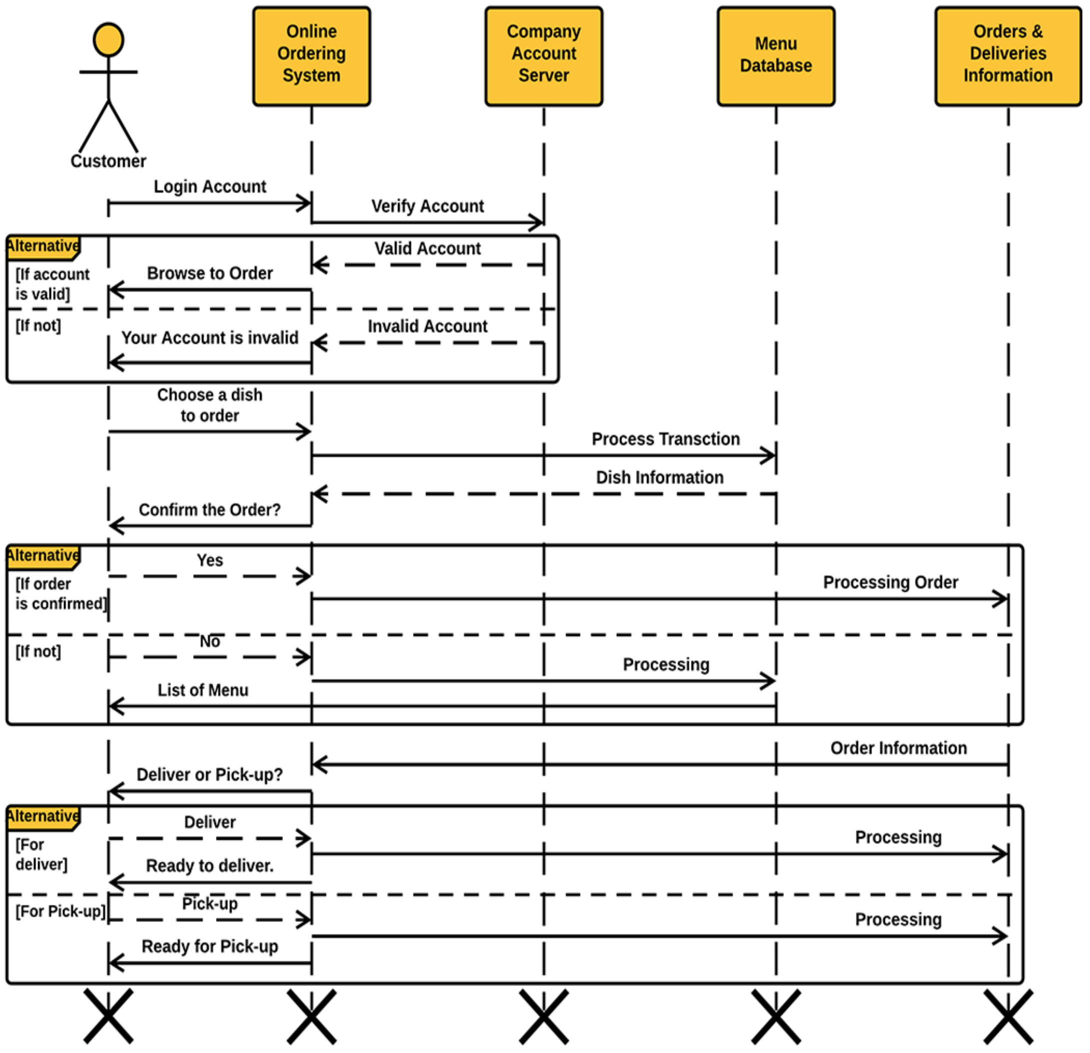

Figure: 4.6

The design shows the detailed illustration of events sequenced and happens in Food Ordering System. This designed sequence diagram is able to show programmers and readers about the sequence of messages between the actor and the objects.

As you can see through the illustration, the conditions and interactions are emphasized. These interactions are essential for the Online Food Ordering System development.

The series of messages are shown and labeled to guide you in building the System. You can modify the design if you have more ideas. You can also add more features to this design and use it as your project blueprint.

4.7 COLLABORATION DIAGRAM

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.

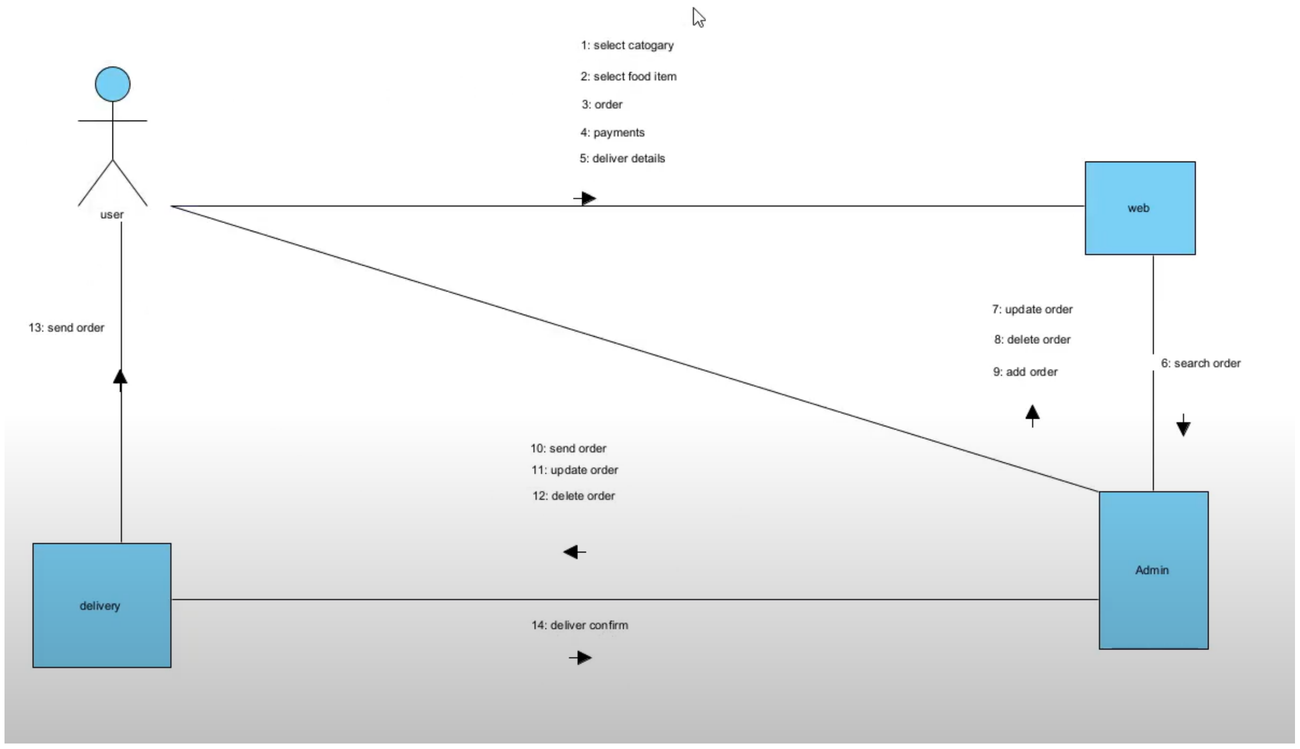

Figure: 4.7

4.8 STATE CHART DIAGRAM

The name of the diagram itself clarifies the purpose of the diagram and other details. It describes different states of a component in a system. The states are specific to a component/object of a system.

A State chart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.

State chart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. State chart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

State chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of State chart diagram is to model lifetime of an object from creation to termination.

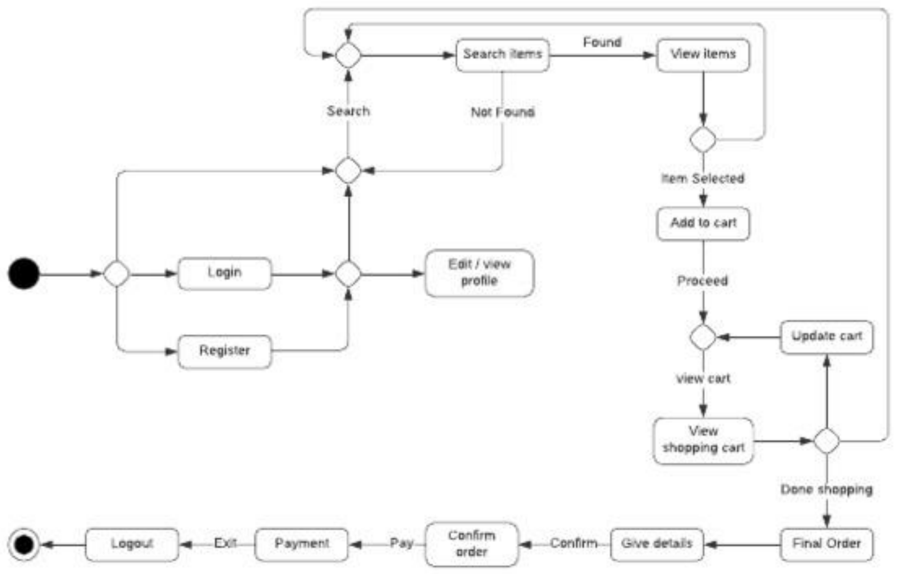

Figure: 4.8

4.9 COMPONENT DIAGRAM

A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.

It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces.

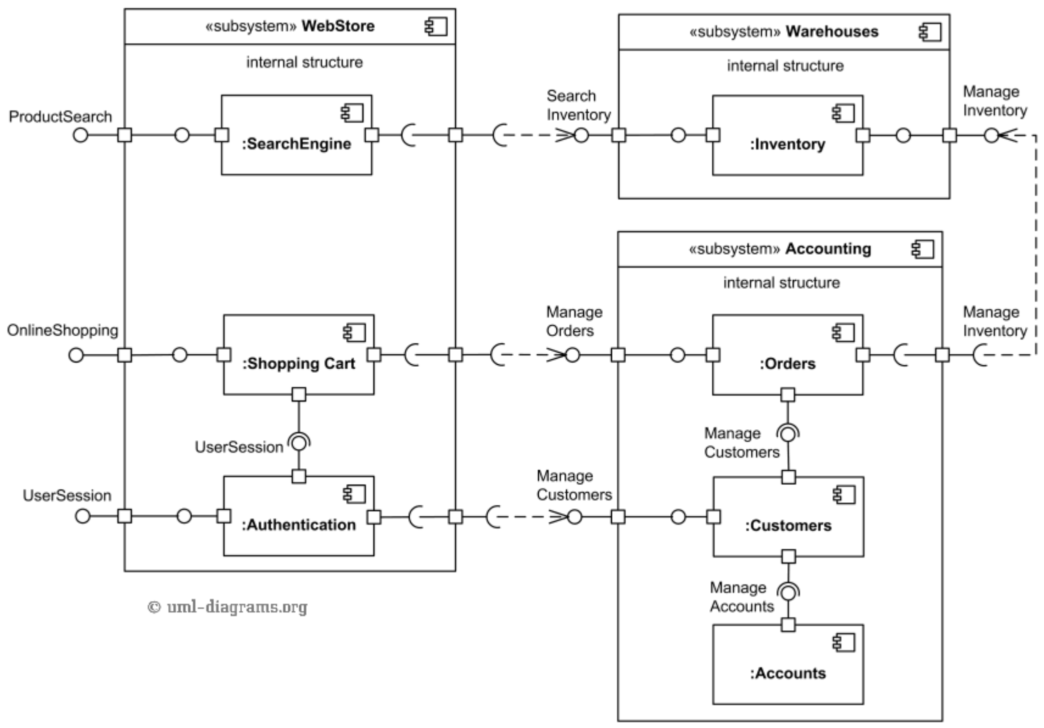

Figure:4.9

4.10 DEPLOYMENT DIAGRAM

Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed.

Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

The term Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components, where software components are deployed. Component diagrams and deployment diagrams are closely related.

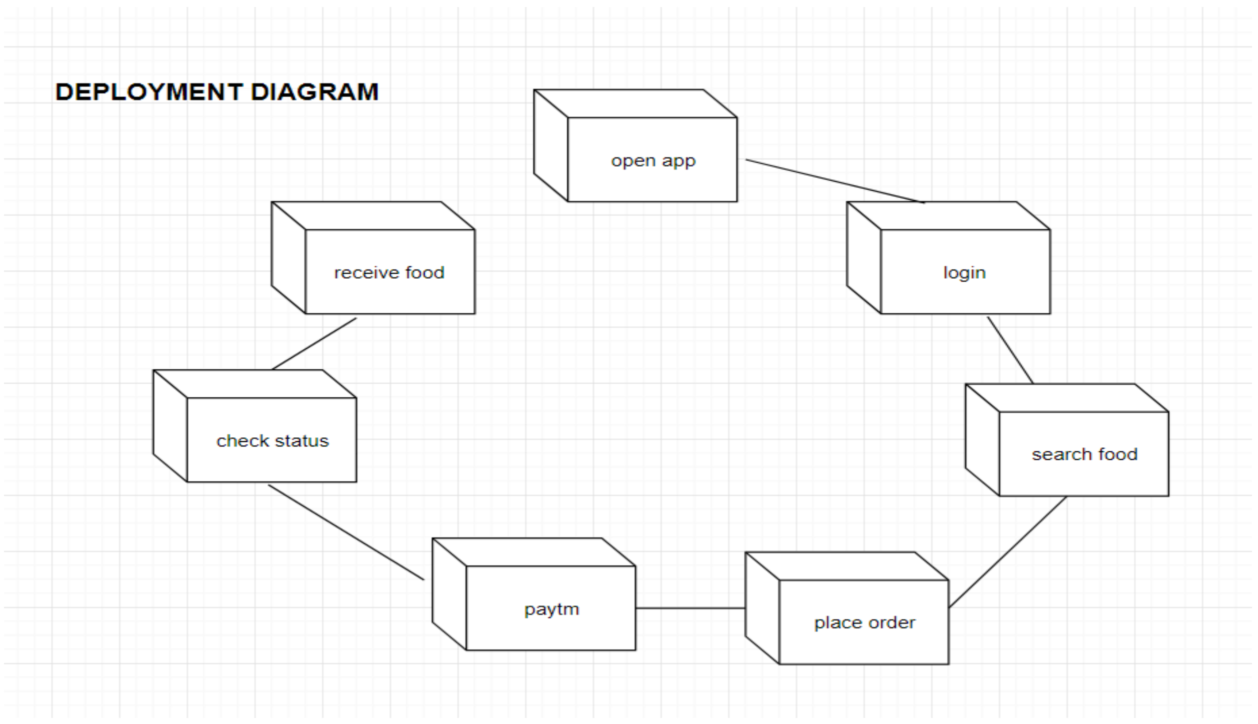

Figure: 4.10

CHAPTER 5: TESTING

Testing is vital for the success of any software. No system design is ever perfect. Testing is also carried in two phases, first is during the software engineering that is during the module creation, second phase is after the completion of software, this is system testing which verifies that the whole set of programs hanged together.

White Box Testing:

In this technique, the close examination of the logical parts through the software are tested by cases that exercise species sets of conditions or loops. All logical parts of the software checked once. Errors that can be corrected using this technique are typographical errors, logical expressions which should be executed once may be getting executed more than once and error resulting by using wrong controls and loops. White box testing tests all the independent parts within a module and logical decisions on their true and the false side are exercised. All loops and bound within their operational bounds were exercised and internal data structure to ensure their validity were exercised once.

Black Box Testing:

This method enables the software engineer to device sets of input techniques that fully exercise all functional requirements for a program. Black Box tests the input, the output and the external data. It checks whether the input data is correct and whether we are getting the desired output.

Alpha Testing:

Acceptance testing is also sometimes called alpha testing. Be spoke systems are developed for a single customer. The alpha testing proceeds until the system developer and the customer agree that the provided system is an acceptable implementation of the system requirements.

Beta Testing:

On the other hand, when a system is to be marked as a software product, another process called beta testing is often conducted. During beta testing, a system is delivered among a number of potential users who agree to use it. The customers then report problems to the developers. This provides the product for real use and detects errors which may not have been anticipated by the system developers.

Unit Testing:

Each module is considered independently. It focuses on each until of software as implemented in the source code. It is white box testing.

Integration Testing:

Integration Testing aims at constructing the program structure while at the same constructing tests to uncover errors associated with interfacing the modules. Modules are integrated by using the top down approach.

Validation Testing:

Validation testing was performed to ensure that all the functional and performance requirements are met.

System Testing:

It is executing programs to check logical changes made in it with intention of finding errors. A system is tested for online response, volume of transaction, recovery from failure etc. System testing is done to ensure that the system satisfies all the user requirements.

Implementation and Software Specification Testings

Detailed Design of implementation

This phase of the systems development life cycle refines hardware and software specifications, establishes programming plans, trains users and implements extensive testing procedures. To evaluate design and operating specifications and/or provide the basis for further modification.

Technical Design

This activity builds upon specifications produced during new system design, adding detailed technical specifications and documentation.

Test Specifications and Planning

This activity prepares detailed test specifications for individual modules and programs, job systems, subsystems, and for the system as a whole.

Programming and Testing

This activity encompasses actual development, writing, and testing of program units or modules.

User Training

This activity encompasses writing user procedure manuals, preparation of user training materials, conducting training programs, and testing procedures.

Acceptance Test

A final procedural review to demonstrate a system and secure user approval before a system becomes operational.

Installation Phase

In this phase the new Computerized system is installed the conversion to new procedures is fully implemented and the potential of the new system is explored.

System Installation

The process of starting the actual use of a system and training user personnel in its operation.

Review Phase

This phase evaluate the successes and failures during a systems development project, and to measure the results of a new Computerized Transystem in terms of benefits and savings projected at the start of the project.

Development Recap

A review of a project immediately after completion to find successes and potential problems in future work.

Post-Implementation Review

A review, conducted after a new system has been in operation for some time, to evaluate actual system performance against original expectations and projections for cost-benefit improvements. Also identifies maintenance projects to enhance or improve the system.

THE STEPS IN THE SOFTWARE TESTING

The steps involved during Unit Testing are as follows:

a. Preparation of the test cases.
b. Preparation of the possible test data with all the validation checks.
c. Complete code review of the module.
d. Actual testing done manually.
e. Modifications done for the errors found during testing.
f. Prepared the test result scripts.

The unit testing done included the testing of the following items:

1. Functionality of the entire module/forms.
2. Validations for user input.
3. Checking of the Coding standards to be maintained during coding.
4. Testing the module with all the possible test data.
5. Testing of the functionality involving all type of calculations etc.
6. Commenting standard in the source files.

After completing the Unit testing of all the modules, the whole system is integrated with all its dependencies in that module. While System Integration, We integrated the modules one by one and tested the system at each step. This helped in reduction of errors at the time of the system testing.

The steps involved during System testing are as follows:

- Integration of all the modules/forms in the system.
- Preparation of the test cases.
- Preparation of the possible test data with all the validation checks.
- Actual testing done manually.
- Recording of all the reproduced errors.

- Modifications done for the errors found during testing.
- Prepared the test result scripts after rectification of the errors.

The System Testing done included the testing of the following items:

- Functionality of the entire system as a whole.
- User Interface of the system.
- Testing the dependent modules together with all the possible test data scripts.
- Verification and Validation testing.
- Testing the reports with all its functionality.

After the completion of system testing, the next following phase was the Acceptance Testing Clients at their end this and accepted the system with appreciation. Thus, we reached the final phase of the project delivery.

There are other six tests, which fall under special category. They are described below:

- Peak Load Test: It determines whether the system will handle the volume of activities that occur when the system is at the peak of its processing demand. For example, test the system by activating all terminals at the same time.
- Storage Testing: It determines the capacity of the system to store transaction data on a disk or in other files.
- Performance Time Testing: It determines the length of time system used by the system to process transaction data. This test is conduced prior to implementation to determine how long it takes to get a response to an inquiry, make a backup copy of a file, or send a transmission and get a response.
- Recovery Testing: This testing determines the ability of user to recover data or re-start system after failure. For example, load backup copy of data and resume processing without data or integrity loss.

- Procedure Testing: It determines the clarity of documentation on operation and users of system by having users do exactly what manuals request. For example, powering down system at the end of week or responding to paper-out light on printer.
- Human Factors Testing: It determines how users will use the system when processing data or preparing reports.

CHAPTER 6: SNAPSHOTS

6.1 HOME PAGE

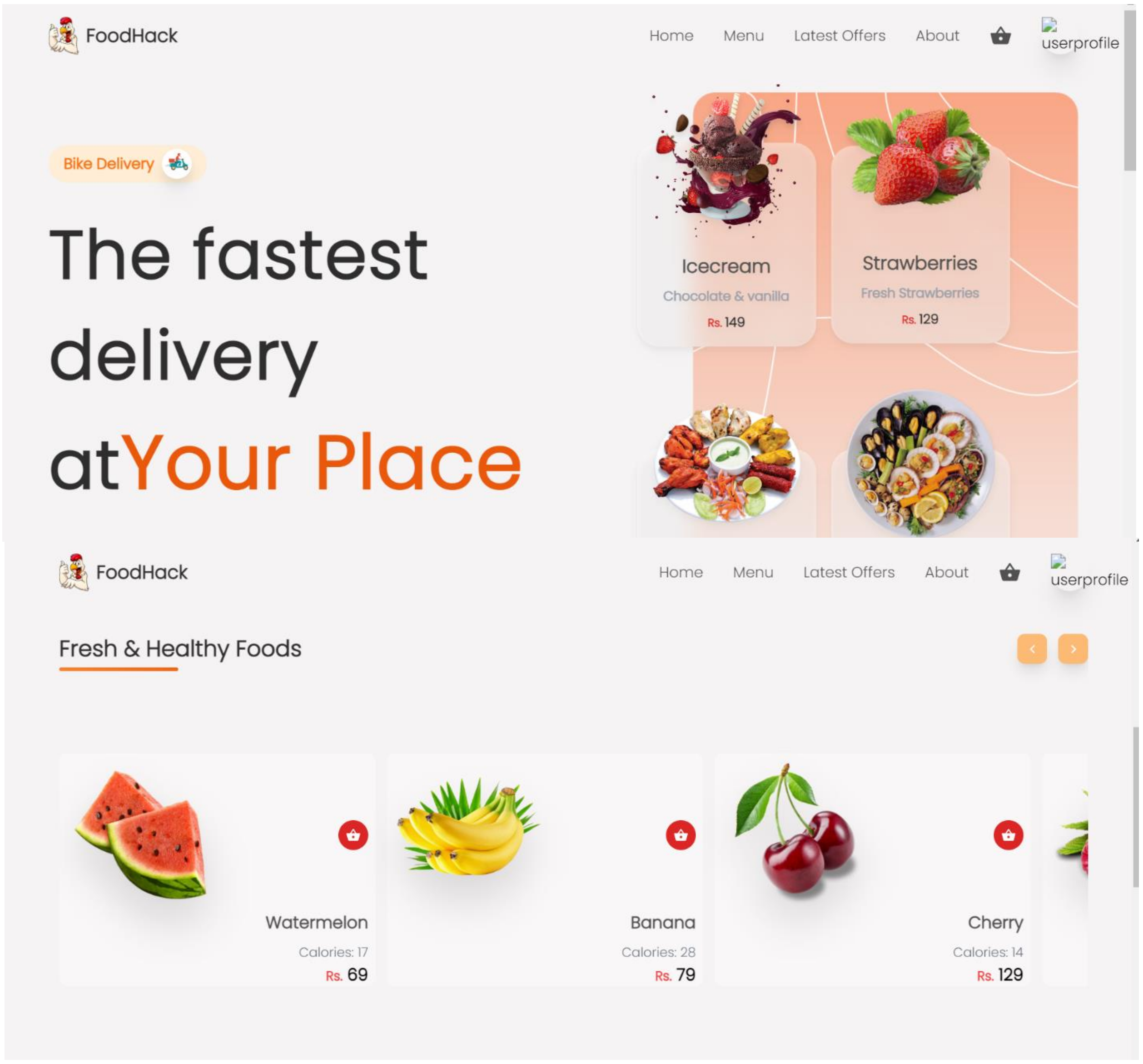

Figure: 6.1

6.2 MENUS

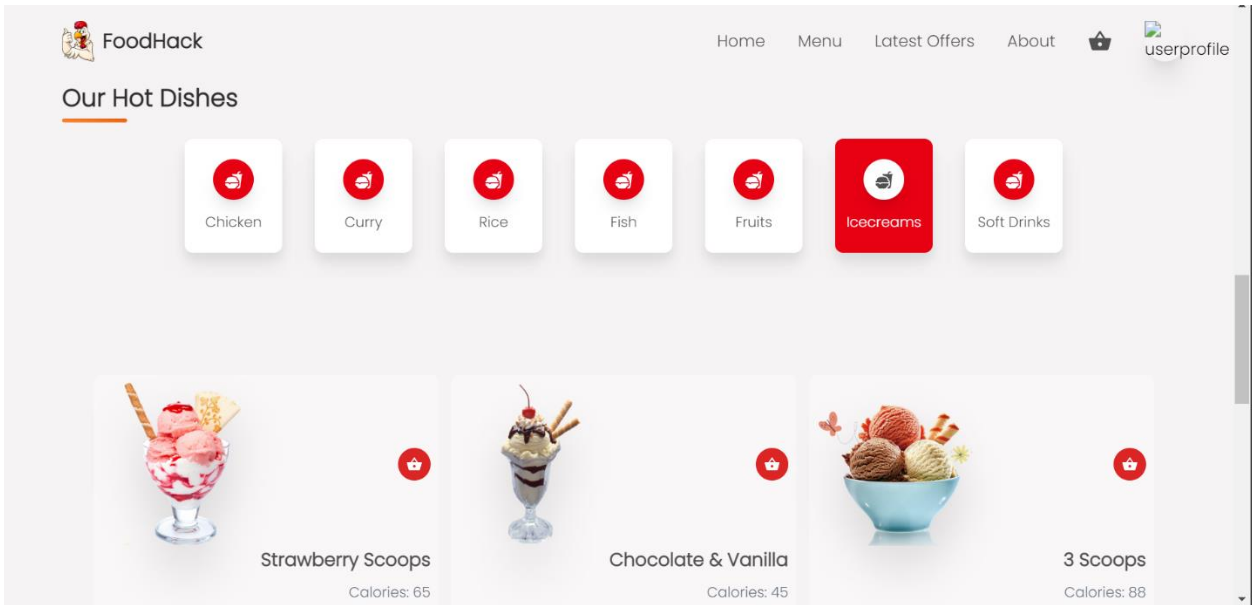

Figure: 6.2

6.3 CART

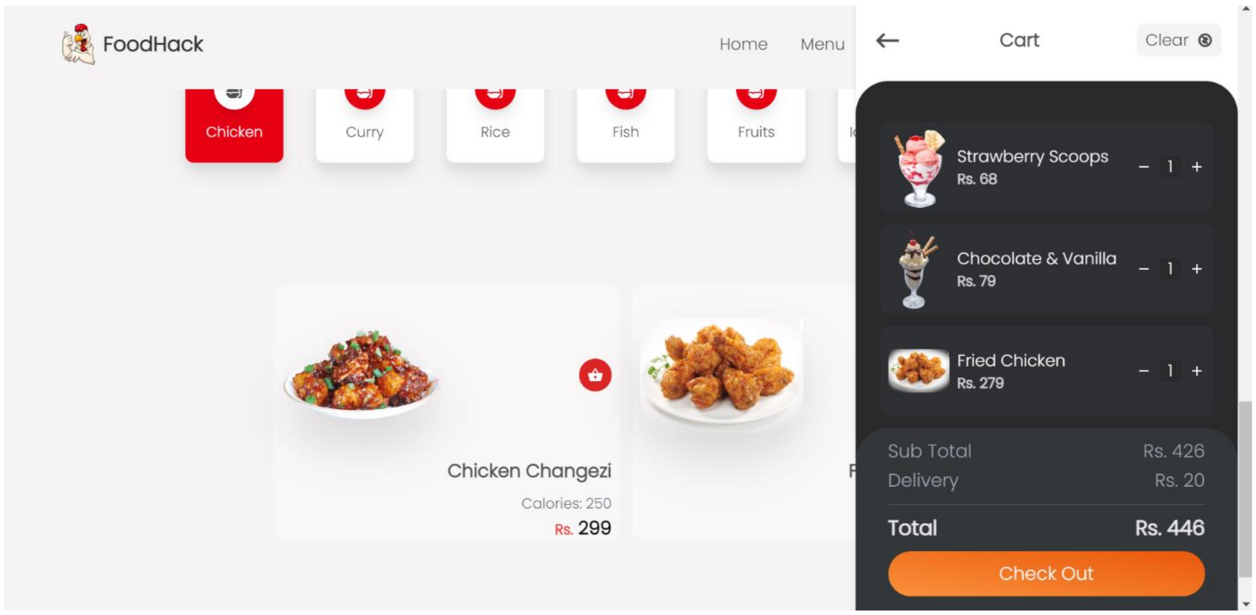

Figure: 6.3

CHAPTER 7: REPORTS

- It generates the report on food item, category, payment.
- Provide filter reports on customer, order, confirm order.
- You can easily export PDF for the food item, payment, order.
- It generates the report on the particular day sell, weekly sell, or monthly sale.
- It generates the report on the number of different users.

CHAPTER 8: CODING

Components:

index.html

```
<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />

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

    <meta name="theme-color" content="#000000" />

    <meta

      name="description"

      content="Web site created using create-react-app"

    />

    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />



    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />



    <title>FoodHack - Food Ordering Website</title>

  </head>

  <body>

    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>


  </body>
</html>
```

Index.js

```
export { default as Header } from "./Header";
export { default as MainContainer } from "./MainContainer";
export { default as CreateContainer } from "./CreateContainer";
export { default as HomeContainer } from "./HomeContainer";
export { default as Loader } from "./Loader";
export { default as RowContainer } from "./RowContainer";
export { default as MenuContainer } from "./MenuContainer";
export { default as CartContainer } from "./CartContainer";
export { default as CartItem } from "./CartItem";
```

app.js

```
import React, { useEffect } from "react";
import { CreateContainer, Header, MainContainer } from "./components";
import { Route, Routes } from "react-router-dom";
import { AnimatePresence } from "framer-motion";
import { getAllFoodItems } from "./utils/firebaseFunctions";
import { useStateValue } from "./context/StateProvider";
import { actionType } from "./context/reducer";


const App = () => {
  const [{ foodItems }, dispatch] = useStateValue();


  const fetchData = async () => {
    await getAllFoodItems().then((data) => {
      dispatch({
        type: actionType.SET_FOOD_ITEMS,
        foodItems: data,
      });
    });
  };


  useEffect(() => {
    fetchData();
  }, []);


  return (
    <AnimatePresence>
      {" "}
      /* if there are multiple animations then exitBeforeEnter will prevent it
      from collission*/
      <div className="w-screen h-auto flex flex-col bg-primary">
        <Header />

        <main className="mt-14 md:mt-20 px-4 md:px-16 py-4 w-full">
          <Routes>
            <Route path="/*" element={<MainContainer />} />
            <Route path="/createItem" element={<CreateContainer />} />
          </Routes>
        </main>
      </div>
    </AnimatePresence>
  );
};
export default App;
```

Header.jsx

```
import React, { useState } from "react";
import Logo from "../img/logo.png";
import { MdShoppingBasket, MdAdd, MdLogout } from "react-icons/md";
import Avatar from "../img/avatar.png";
import { motion } from "framer-motion";
import { getAuth, signInWithPopup, GoogleAuthProvider } from "firebase/auth";
import { actionType } from "../context/reducer";
import { useStateValue } from "../context/StateProvider";


import { Link } from "react-router-dom";

import { app } from "../firebase.config";


const Header = () => {

  const firebaseAuth = getAuth(app);

  const provider = new GoogleAuthProvider();


  const [{ user, cartShow, cartItems }, dispatch] = useStateValue();


  const [isMenu, setisMenu] = useState(false);


  const login = async () => {

    if (!user) {

      const {

        user: { refreshToken, providerData },

      } = await signInWithPopup(firebaseAuth, provider);

      dispatch({

        type: actionType.SET_USER,

        user: providerData[0],

      });

      localStorage.setItem("user", JSON.stringify(providerData[0]));
    } else {
      setisMenu(!isMenu);
    }
  };

  const logout = () => {
    setisMenu(false);
    localStorage.clear();
    dispatch({
      type: actionType.SET_USER,
      user: null,
    });
  };

  const showCart = () => {
    dispatch({
      type: actionType.SET_CART_SHOW,
      cartShow: !cartShow,
    });
  };

  return (
    <header className="fixed z-50 w-screen p-3 px-4 md:p-6 md:px-16 bg-primary">
```

```
{/* desktop and tablet*/}
<div className="hidden md:flex w-full h-full items-center justify-between">
  <Link to={"/"} className="flex items-center gap-2">
    <img src={Logo} className="w-8 object-cover" alt="logo" />
    <p className="text-headingColor text-xl font-bold">FoodHack</p>
  </Link>
  <div className="flex items-center gap-8">
    <motion.ul
      initial={{ opacity: 0, x: 200 }}
      animate={{ opacity: 1, x: 0 }}
      exit={{ opacity: 0, x: 200 }}
      className="flex items-center gap-8 "
    >
      <li className="text-base text-textColor hover:text-headingColor duration-100 transition-
all ease-in-out cursor-pointer">
        Home
      </li>
      <li className="text-base text-textColor hover:text-headingColor duration-100 transition-
all ease-in-out cursor-pointer">
        <a href="#menu">Menu</a>
      </li>
      <li className="text-base text-textColor hover:text-headingColor duration-100 transition-
all ease-in-out cursor-pointer">
```

```
              Latest Offers
            </li>
            <li className="text-base text-textColor hover:text-headingColor duration-100 transition-
all ease-in-out cursor-pointer">
              About
            </li>
          </motion.ul>


          <div
            className="relative flex items-center justify-center"
            onClick={showCart}
          >
            <MdShoppingBasket className="text-textColor text-2xl cursor-pointer" />
            {cartItems && cartItems.length > 0 && (
              <div className="absolute -top-2 -right-2 w-4 h-4 rounded-full flex items-center bg-
cartNumBg text-center justify-center">
                <p className="text-sm text-white font-semibold">
                  {cartItems.length}
                </p>
              </div>
            )}
          </div>
```

```
     <div className="relative ">
      <motion.img
        whileTap={{ scale: 0.6 }}
        src={user ? user.photoURL : Avatar}
        className="w-10 min-w-[40px] h-10 min-h-[40px] drop- shadow-xl cursor-pointer
rounded-full"
        alt="userprofile"
        onClick={login}
      />
      {isMenu && (
        <motion.div
          initial={{ opacity: 0, scale: 0.6 }}
          animate={{ opacity: 1, scale: 1 }}
          exit={{ opacity: 0, scale: 0.6 }}
          className="w-40 bg-gray-50 shadow-xl rounded-lg flex flex-col absolute top-12 right-
0"
        >
          {user && user.email === "atif.2124mca1052@kiet.edu" && (
            <Link to={"/createItem"}>
              <p className="px-4 py-2 flex items-center gap-3 cursor-pointer hover:bg-slate-100
transition-all duration-100 ease-in-out text-textColor text-base">
                New Item
                <MdAdd />
```

```
                  </p>
                </Link>
              )}
              <p
                className="px-4 py-2 flex items-center gap-3 cursor-pointer hover:bg-slate-100
transition-all duration-100 ease-in-out text-textColor text-base"
                onClick={logout}
              >
                Logout
                <MdLogout />
              </p>
            </motion.div>
          )}
        </div>
      </div>
    </div>

    {/* mobile */}
    <div className="flex items-center justify-between md:hidden w-full h-full">
      <div
        className="relative flex items-center justify-center"
        onClick={showCart}
      >
```

```
          <MdShoppingBasket className="text-textColor text-2xl cursor-pointer" />
          {cartItems && cartItems.length > 0 && (
            <div className="absolute -top-2 -right-2 w-4 h-4 rounded-full flex items-center bg-
cartNumBg text-center justify-center">
              <p className="text-sm text-white font-semibold">
                {cartItems.length}
              </p>
            </div>
          )}
        </div>

        <Link to={"/"} className="flex items-center gap-2">
          <img src={Logo} className="w-8 object-cover" alt="logo" />
          <p className="text-headingColor text-xl font-bold">
            KIET Amul Parlour
          </p>
        </Link>

        <div className="relative ">
          <motion.img
            whileTap={{ scale: 0.6 }}
            src={user ? user.photoURL : Avatar}
            className="w-10 min-w-[40px] h-10 min-h-[40px] drop- shadow-xl cursor-pointer
rounded-full"
            alt="userprofile"
            onClick={login}
          />
          {isMenu && (
            <motion.div
              initial={{ opacity: 0, scale: 0.6 }}
              animate={{ opacity: 1, scale: 1 }}
              exit={{ opacity: 0, scale: 0.6 }}
              className="w-40 bg-gray-50 shadow-xl rounded-lg flex flex-col absolute top-12 right-
0"
            >
              {user && user.email === "atif.2124mca1052@kiet.edu" && (
                <Link to={"/createItem"}>
                  <p
                    className="px-4 py-2 flex items-center gap-3 cursor-pointer hover:bg-slate-100
transition-all duration-100 ease-in-out text-textColor text-base "
                    onClick={() => setisMenu(false)}
                  >
                    New Item
                    <MdAdd />
                  </p>
          </Link>
        )}

        <ul className="flex flex-col">
          <li
            className="text-base text-textColor hover:text-headingColor duration-100 transition-
all ease-in-out cursor-pointer hover:bg-slate-100 px-4 py-2 "
            onClick={() => setisMenu(false)}
          >
            Home
          </li>
          <li
            className="text-base text-textColor hover:text-headingColor duration-100 transition-
all ease-in-out cursor-pointer hover:bg-slate-100 px-4 py-2 "
            onClick={() => setisMenu(false)}
          >
            Menu
          </li>
          <li
            className="text-base text-textColor hover:text-headingColor duration-100 transition-
all ease-in-out cursor-pointer hover:bg-slate-100 px-4 py-2 "
            onClick={() => setisMenu(false)}
          >
```

```
                  About
                </li>
                <li
                  className="text-base text-textColor hover:text-headingColor duration-100 transition-all ease-in-out cursor-pointer hover:bg-slate-100 px-4 py-2 "
                  onClick={() => setisMenu(false)}
                >
                  Services
                </li>
              </ul>

              <p
                className="m-2 p-2 rounded-md shadow-md flex items-center justify-center bg-gray-200  gap-3 cursor-pointer hover:bg-gray-300 transition-all duration-100 ease-in-out text-textColor text-base"
                onClick={logout}
              >
                Logout
                <MdLogout />
              </p>
            </motion.div>
          )}
        </div>
```

```
      </div>
    </header>
  );
};
export default Header;
```

HomeContainer.jsx

```
import React from "react";
import Delivery from "../img/delivery.png";
import HeroBg from "../img/heroBg.png";
import { heroData } from "../utils/data";

const HomeContainer = () => {
  return (
    <section className="grid grid-cols-1 md:grid-cols-2 gap-2 w-full" id="home">
      <div className="py-2 flex-1 flex flex-col items-start  justify-center gap-6">
        <div className="flex items-center gap-2 justify-center bg-orange-100 px-4 py-1 rounded-
full">
          <p className="text-base text-orange-500 font-semibold">
            Bike Delivery
          </p>
          <div className="w-8 h-8 bg-white rounded-full overflow-hidden drop-shadow-xl">
            <img
            src={Delivery}
            className="w-full h-full object-contain"
            alt="delivery"
          />
        </div>
      </div>

      <p className="text-[2.5rem] lg:text-[4.5rem] font-bold tracking-wide text-headingColor ">
        The fastest delivery at
        <span className="text-orange-600 text-[3rem] lg:text-[5rem]">
          Your Place
        </span>
      </p>

      <p className="text-base text-textColor text-center md:text-left md:w-[80%]">
        Save your time. Order your food and get it delivered at your own
        place.
      </p>

      <button
        type="button"
        className="bg-gradient-to-br from-orange-400 to-orange-500 w-full md:w-auto px-4 py-2
rounded-lg hover:shadow-lg transition-all ease-in-out duration-100"
```

```
    >
      Order Now
    </button>
  </div>
  <div className="py-2 flex-1 flex items-center relative">
    <img
      src={HeroBg}
      className="ml-auto h-400 w-full lg:w-auto lg:h-650"
      alt="hero-bg"
    />

    <div className="w-full h-full absolute top-0 left-0 flex items-center justify-center lg:px-16
py-4 gap-4 flex-wrap ">
      {heroData &&
        heroData.map((n) => (
          <div
            key={n.id}
            className="lg:w-120 p-4 bg-cardOverlay backdrop-blur-md rounded-3xl flex flex-col
items-center justify-center drop-shadow-lg"
          >
            <img
              src={n.imageSrc}
              className="w-20 lg:w-40 -mt-10 lg:-mt-20"
```

```
                  alt="I1"
                />
                <p className="text-base lg:text-xl font-semibold text-textColor mt-2 lg:mt-4">
                  {n.name}
                </p>

                <p className="text-[12px] lg:text-sm text-lighttextGray font-semibold my-1 lg:my-2
">
                  {n.decp}
                </p>

                <p className="text-sm font-semibold text-headingColor">
                  <span className="text-xs text-red-600">Rs. </span>
                  {n.price}
                </p>
              </div>
            ))}
          </div>
        </div>
      </section>
  );
};
export default HomeContainer;
```

CartContainer.jsx

```
import React, { useState } from "react";
import { MdOutlineKeyboardBackspace } from "react-icons/md";
import { motion } from "framer-motion";
import { RiRefreshFill } from "react-icons/ri";

import { useStateValue } from "../context/StateProvider";
import { actionType } from "../context/reducer";
import EmptyCart from "../img/emptyCart.svg";
import { CartItem } from ".";
import { useEffect } from "react";

const CartContainer = () => {
  const [{ cartShow, cartItems, user }, dispatch] = useStateValue();
  const [flag, setFlag] = useState(1);
  const [tot, setTotal] = useState(0);

  const showCart = () => {
    dispatch({
      type: actionType.SET_CART_SHOW,
      cartShow: !cartShow,
    });
```

```
};

useEffect(() => {
  let totalPrice = cartItems.reduce(function (accumulator, item) {
    return accumulator + item.qty * item.price;
  }, 0);
  setTotal(totalPrice);
}, [tot, flag]);

const clearCart = () => {
  dispatch({
    type: actionType.SET_CARTITEMS,
    cartItems: [],
  });
  localStorage.setItem("cartItems", JSON.stringify([]));
};

return (
  <motion.div
    initial={{ opacity: 0, x: 200 }}
    animate={{ opacity: 1, x: 0 }}
    exit={{ opacity: 0, x: 200 }}
```

```
      className="fixed top-0 right-0 w-full md:w-375 h-screen bg-white drop-shadow-md flex
flex-col z-[101]"
    >
      <div
        className="w-full flex items-center justify-between p-4"
        onClick={showCart}
      >
        <motion.div whileTap={{ scale: 0.75 }}>
          <MdOutlineKeyboardBackspace className="text-textColor text-3xl cursor-pointer" />
        </motion.div>

        <p className="text-textColor text-lg font-semibold">Cart</p>

        <motion.p
          whileTap={{ scale: 0.75 }}
          className="flex items-center gap-2 p-1 px-2 my-2 bg-gray-100 rounded-md
hover:shadow-md cursor-pointer text-textColor text-base"
        >
          Clear <RiRefreshFill />
        </motion.p>
      </div>

      {/* bottom section */}
      {cartItems && cartItems.length > 0 ? (
        <div className="w-full h-full bg-cartBg rounded-t-[2rem] flex flex-col">
          <div className="w-full h-340 md:h-42 px-6 py-10 flex flex-col gap-3 overflow-y-scroll
scrollbar-none">
            {/* cart item */}
            {cartItems &&
              cartItems.map((item) => (
                <CartItem
                  key={item.id}
                  item={item}
                  setFlag={setFlag}
                  flag={flag}
                />
              ))}
          </div>

          {/* cartTotal section */}

          <div className="w-full flex-1 bg-cartTotal rounded-t-[2rem] flex flex-col items-center
justify-evenly px-8 py-2">
            <div className="w-full flex items-center justify-between">
              <p className="text-gray-400 text-lg">Sub Total</p>
              <p className="text-gray-400 text-lg">Rs. {tot}</p>
```

```
</div>

<div className="w-full flex items-center justify-between">
  <p className="text-gray-400 text-lg">Delivery</p>
  <p className="text-gray-400 text-lg">Rs. 20</p>
</div>

<div className="w-full border-b border-gray-600 my-2"></div>

<div className="w-full flex items-center justify-between">
  <p className="text-gray-200 text-xl font-semibold">Total</p>
  <p className="text-gray-200 text-xl font-semibold">
    Rs. {tot + 20}
  </p>
</div>

{user ? (
  <motion.button
    whileTap={{ scale: 0.8 }}
    type="button"
    className="w-full p-2 rounded-full bg-gradient-to-tr from-orange-400 to-orange-600
text-gray-50 text-lg my-2 hover:shadow-lg"
  >
```

```
                Check Out
              </motion.button>
            ) : (
              <motion.button
                whileTap={{ scale: 0.8 }}
                type="button"
                className="w-full p-2 rounded-full bg-gradient-to-tr from-orange-400 to-orange-600
text-gray-50 text-lg my-2 hover:shadow-lg"
              >
                LogIn
              </motion.button>
            )}
          </div>
        </div>
      ) : (
        <div className="w-full h-full flex flex-col items-center justify-center gap-6">
          <img src={EmptyCart} className="w-300" alt="" />
          <p className="text-xl text-textColor font-semibold">
            Add some items to your cart
          </p>
        </div>
      )}
    </motion.div>
```

```
  );
};
export default CartContainer;
```

Context:

initialState.js

```
import { fetchCart, fetchUser } from "../utils/fetchLocalStorageData";


const userInfo = fetchUser();
const cartInfo = fetchCart();


export const initialState = {
  user: userInfo,
  foodItems: null,
  cartShow: false,
  cartItems: cartInfo,
};
```

Reducer.js

```
export const actionType = {
  SET_USER: "SET_USER",
  SET_FOOD_ITEMS: "SET_FOOD_ITEMS",
  SET_CART_SHOW: "SET_CART_SHOW",

  SET_CARTITEMS: "SET_CARTITEMS",

};


const reducer = (state, action) => {

  console.log(action);


  switch (action.type) {

    case actionType.SET_USER:

      return {

        ...state,

        user: action.user,

      };


    case actionType.SET_FOOD_ITEMS:

      return {

        ...state,

        foodItems: action.foodItems,

      };

    case actionType.SET_CART_SHOW:

      return {

        ...state,

        cartShow: action.cartShow,
      };
    case actionType.SET_CARTITEMS:
      return {
        ...state,
        cartItems: action.cartItems,
      };
    default:
      return state;
  }
};

export default reducer;
```

<u>stateProvider.js</u>

```
import React, { createContext, useContext, useReducer } from "react";


export const StateContext = createContext();
export const StateProvider = ({ reducer, initialState, children }) => (
  <StateContext.Provider value={useReducer(reducer, initialState)}>
    {children}
  </StateContext.Provider>
);
export const useStateValue = () => useContext(StateContext);
```

Utilities

Data.js

```
import I1 from "../img/i1.png";

import F1 from "../img/f1.png";

import C3 from "../img/c3.png";

import Fi1 from "../img/fi1.png";


export const heroData = [

  {

    id: 1,

    name: "Icecream",

    decp: "Chocolate & vanilla",

    price: "149",

    imageSrc: I1,

  },

  {

    id: 2,

    name: "Strawberries",

    decp: "Fresh Strawberries",

    price: "129",

    imageSrc: F1,
  },
  {
    id: 3,
    name: "Chicken Kabab",
    decp: "Mixed Kabab Plate",
    price: "289",
    imageSrc: C3,
  },
  {
    id: 4,
    name: "Fish Kabab",
    decp: "Mixed Kabab Plate",
    price: "269",
    imageSrc: Fi1,
  },
];

export const categories = [
  {
    id: 1,
    name: "Chicken",
    urlParamName: "chicken",
  },
  {
    id: 2,
    name: "Curry",
    urlParamName: "curry",
  },
  {
    id: 3,
    name: "Rice",
    urlParamName: "rice",
  },
  {
    id: 4,
    name: "Fish",
    urlParamName: "fish",
  },
  {
    id: 5,
    name: "Fruits",
    urlParamName: "fruits",
  },
  {
    id: 6,
    name: "Icecreams",
    urlParamName: "icecreams",
  },
  {
    id: 7,
    name: "Soft Drinks",
    urlParamName: "drinks",
  },
];
```

firebaseFunctions.js

```
import {
  collection,
  doc,
  getDocs,
  orderBy,
  query,
  setDoc,
} from "firebase/firestore";
import { firestore } from "../firebase.config";
export const saveItem = async (data) => {
  await setDoc(doc(firestore, "foodItems", `${Date.now()}`), data, {
    merge: true,
  });
};

export const getAllFoodItems = async () => {

  const items = await getDocs(

    query(collection(firestore, "foodItems"), orderBy("id", "desc"))

  );

  return items.docs.map((doc) => doc.data());

};
```

CHAPTER 9: CONCLUSION

Our project is only a humble venture to satisfy the needs to manage their project work. User friendly coding have also adopted. This package shall prove to be a powerful package in satisfying all the requirements of the school. The objective of software planning is to provide a frame word that enables the manager to make reasonable estimates made within a limited time frame at the beginning of the software project and should be updated regularly as the project progresses.

At the end it is concluded that we have made effort on following points

- A description of the background and context of the project and its relation to word already done in the area.
- Made statement of the aims and objectives of the project.
- The description of purpose, scope, and applicability.
- We define the problem on which we are working on the project.
- We describe the requirement specifications of the system and the actions that can be done on these things.
- We understand the problem domain and produce a model of the system, which describes operations that can be performed on the system.
- We included features and operations in detail, including screen layouts.
- We designed user interface and security issues related to system.

CHAPTER 10: FUTURE SCOPE

In a nutshell, it can be summarized that the future scope of the project circles around maintaining information regarding:

- We can add Printer in future
- We can give more advance software for Online Food Ordering System including more facilities
- We will host the platform on online servers to make it accessible worldwide
- Integrate multiple load balancers to distribute the loads of the system
- Create the master and slave database structure to reduce the overload of the database queries
- Implement the backup mechanism for taking backup of codebase and database on regular on different servers

The above-mentioned points are the enhancements which can be done to increase the applicability and usage of this project. Here we can maintain the records of Food Item and Category. Also, as it can be seen that now-a-days the players are versatile, i.e. so there is a scope for introducing a method to maintain the Online Food Ordering System. Enhancements can be done to maintain all the Food Item, Category, Customer, Order, Confirm Order.

We have left all the options open so that if there is any other future requirement in the system by the user for the enhancement of the system then it is possible to implement them. In the last we would like to thanks all the persons involved in the development of the system directly or indirectly. We hope that the project will serve its purpose for which it is develop there by underlining success of process.

CHAPTER 11: LIMITATION OF PROJECT

Although I have put my best efforts to make the software flexible, easy to operate but limitations cannot be ruled out even by me. Though the software presents a broad range of options to its users some intricate options could not be covered into it, partly because of logistic and partly due to lack of sophistication. Paucity of time was also major constraint, thus it was not possible to make the software foolproof and dynamic. Lack of time also compelled me to ignore some part such as storing old result of the candidate etc.

Considerable efforts have made the software easy to operate even for the people not related to the field of computers but it is acknowledged that a layman may find it a bit problematic at the first instance. The user is provided help at each step for his convenience in working with the software.

List of limitations which is available in the Online Food Ordering System:

- Excel export has not been developed for Food Item, Category due to some criticality.
- The transactions are executed in off-line mode, hence on-line data for Customer, Order capture and modification is not possible.
- Off-line reports of Food Item, Confirm Order, Customer cannot be generated due to batch mode execution.

CHAPTER 12: REFERENCES

- https://www.tutorialspoint.com
- https://www.javatpoint.com
- https://www.sqlite.org/index.html
- https://www.w3school.com
- https://www.w3schools.com/html/
- https://www.youtube.com
- https://www.google.com
- https://www.wikepedia.com
- https://www.geeksforgeeks.com
- https://www.studocu.com