

# ONLINE TIFFIN SERVICE

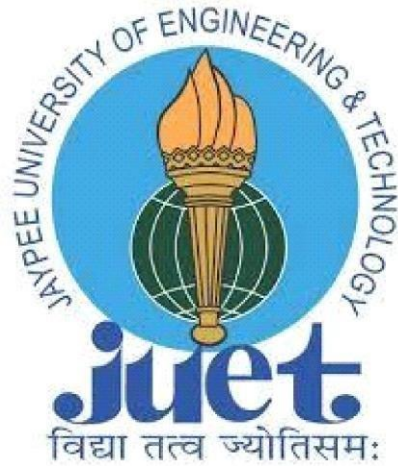
A Project Report

*Submitted by:*

Esha Mitra - 201b107  
Prince Katore - 201b197  
Navanshu – 201b164

*Under the guidance of:*

**Dr. Kunj Bihari Meena**



Jan 2024

*Submitted in partial fulfillment for the award of the degree*

*Of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

Department of Computer Science & Engineering

JAYPEE UNIVERSITY OF ENGINEERING & TECHNOLOGY,

AB ROAD, RAGHOGARH, DT. GUNA - 473226, M.P., INDIA

## DECLARATION

We hereby declare that the work reported in the B. Tech. project entitled as “**Online Tiffin Service**” in partial fulfillment for the award of degree of, Bachelor of Technology submitted at Jaypee University of Engineering and Technology, Guna, as per best of my knowledge and belief there is no infringement of intellectual property right and copyright. In case of any violation, we will solely be responsible.

Esha Mitra – 201b107

Navanshu – 201b164

Prince Katare – 201b197

Department of Computer Science and Engineering,  
Jaypee University of Engineering and Technology,  
Guna, M.P., India

Date:



## **JAYPEE UNIVERSITY OF ENGINEERING & TECHNOLOGY**

**Grade 'A+' Accredited with by NAAC & Approved U/S 2(f) of the UGC Act, 1956**

**A.B. Road, Raghogarh, Dist: Guna (M.P.) India, Pin-473226**

**Phone: 07544 267051, 267310-14, Fax: 07544 267011**

**Website: [www.juet.ac.in](http://www.juet.ac.in)**

### **CERTIFICATE**

This is to certify that the work titled **“Online Tiffin Service”** submitted by **“Esha Mitra(201b107), Navanshu (201b164) and Prince Katare (201b197)”** in partial fulfilment for the award of degree of Bachelor of Technology (Computer science Engineering) of Jaypee University of engineering & Technology, Guna has been carried under my supervision. As per best of my knowledge and belief, there is no infringement of intellectual property right and copyright. Also, this work has not been submitted partially or whole to any other university or Institute for the award of this or any other degree or diploma. In case of any violation concern students will solely be responsible.

(Dr. Kunj Bihari Meena)

Assistant Professor (SG), Dept. of CSE

Date:

## **ACKNOWLEDGEMENT**

When undergoing through the making of a certain project, there exists a lot of contribution and guidance of other people who support us in achieving what lies in front. Not only do they guide our path, but also play a vital role in the establishment and understanding of what we really want to achieve.

Thus, this is to show our gratitude towards our mentor, **Dr. K.B. Meena** for guiding us through our entire report. He supported us immensely in the better understanding of the project, guiding us towards the better knowledge of the project we were working on. The weekly interaction with him brought to us more clarification on the topic and thus simplifying our task.

**Thanking you ,**

**Esha Mitra (201b107)**

**Navanshu (201b164)**

**Prince Katare (201b197)**

Date:

## EXECUTIVE SUMMARY

Online Tiffin Service is a premium tiffin service aimed at providing wholesome and delicious meals conveniently delivered to customers' doorsteps. Our service caters to individuals and families seeking healthy, flavorful, and convenient dining options amidst their busy schedules. We prioritize freshness, quality ingredients, and customer satisfaction in every meal we prepare.

In this report, we describe the development of Online Tiffin Service, an online – tiffin service system using the MERN stack, a popular web development technology stack that includes MongoDB, Express, React, and Node.js. The system includes modules for new products, search products, filter & range of meal management, cart / wish-list, payment.

We begin by outlining the design and architecture of the system, including the database schema and API endpoints. We then describe the implementation of the front-end user interface using React, including forms for registration and login, scheduling, and a dashboard for doctors and administrators to manage appointments and medical records. Next, we detail the back-end implementation using Node.js and Express, including authentication and authorization for different user roles, validation of input data, and integration with external APIs for verification and processing of data.

Finally, we evaluate the performance and usability of the system, including load testing, user feedback, and future directions for development and improvement.

Overall, our Online Tiffin Service demonstrates the power and flexibility of the MERN stack for building complex web applications, and provides a useful tool for improving vendor and buyer experience and provides a platform for the local sellers to compete in the market.

## LIST OF FIGURES

S. No.	Figure No.	Figure Name	Page No.
1	Figure 3.1	Block Diagram of MERN Stack	10
2	Figure 3.2	Level-0 Data Flow Diagram	16
3	Figure 3.3	Level-1 Data Flow Diagram	16
4	Figure 3.4	Use Case Diagram	17
5	Figure 3.5	Schema Diagram	20
6	Figure 3.6	Sequence Diagram	21
7	Figure 3.7	User Schema	27
8	Figure 3.8	Product Schema (part 1)	28
9	Figure 3.9	Product Schema (part 2)	29
10	Figure 3.10	Subscription Schema	30
11	Figure 4.1	Customer Signup Page	31
12	Figure 4.2	Vendor Signup Page	31
13	Figure 4.3	Edit Details Page	32
14	Figure 4.4	Customer Login Page	33
15	Figure 4.5	Vendor Login Page	33
16	Figure 4.6	Home Page	34
17	Figure 4.7	List of Available Vendors ( by Pin-Code)	34

18	Figure 4.8	Add Meal Subscription	35
19	Figure 4.9	Approve / Cancel Subscription Request	35
20	Figure 4.10	See Reviews Page	36
21	Figure 4.11	Add Reviews Page	36
22	Figure 4.12	Revenue Chart Page	37

## Table of Contents

Cover Page	i
Declaration of the Student.	ii
Certificate of the Guide	iii
Acknowledgement.	iv
Executive Summary	v
List of Figures	vi- vii
Table of Content	viii
<b>1. INTRODUCTION</b>	<b>1</b>
1.1 Problem Definition	2
1.2 Project Overview	2
1.3 Hardware Specification	3
1.4 Software Specification	4-5
<b>2. LITERATURE SURVEY</b>	<b>6</b>
2.1 Existing System	6
2.2 Proposed System	6
2.3 Feasibility Study	7
<b>3. SYSTEM ANALYSIS AND DESIGN</b>	<b>10</b>
3.1 Components of MERN Stack	10
3.2 Requirement Specification	13
3.3 Diagrams	15
3.3.1 Data Flow Diagram	16
3.3.2 Use Case Diagram	17
3.3.3 Schema Diagram	19
3.3.4 Sequence Diagram	20
3.4 Design and Test Steps	21
3.5 Algorithms and Pseudo Code	25
3.6 Database Design	26
<b>4. RESULTS / OUTPUTS</b>	<b>31</b>
<b>5. CONCLUSIONS / RECOMMENDATIONS</b>	<b>38</b>
<b>6. REFERENCES</b>	<b>39</b>
<b>7. APPENDICES</b>	<b>40</b>
7.1 Appendix 1 – ReactJS	40
7.2 Appendix 2 – NodeJS.	41
<b>8.STUDENTS PROFILE</b>	<b>43</b>



# **CHAPTER 1**

## **INTRODUCTION**

Online Tiffin Service is a premium tiffin service aimed at providing wholesome and delicious meals conveniently delivered to customers' doorsteps. Our service caters to individuals and families seeking healthy, flavorful, and convenient dining options amidst their busy schedules. We prioritize freshness, quality ingredients, and customer satisfaction in every meal we prepare.

In this report, we describe Online Tiffin Service , built using the MERN stack, a popular web development technology stack that includes MongoDB, Express, React, and Node.js. Online Tiffin Service includes a dashboard, filters, price range facility, search products from the lot , register and login pages, including adding to cart and payment of the order. We begin by outlining the challenges and opportunities in traditional market, including how online retail can be available 24/7, easy management of cash flow, resources and products. We then describe the design and architecture of our online retail system, including the data model and API endpoints.

Next, we provide an overview of the key features,including the login, register, search , meal management , order , delivery , payment and filter of the meal plans.

Finally, we evaluate the performance and usability of our meal management system, including user feedback and future directions for development and improvement. We believe that our system provides a valuable tool for improving patient care and optimizing resources, and demonstrates the power and flexibility of the MERN stack for building complex web applications in this domain.

## 1.1 Problem Definition

The rise of busy lifestyles, long working hours, and the need for healthier, homemade meals has created a demand for convenient and reliable tiffin (meal) delivery services. In response to this demand, the development of an online tiffin management service aims to bridge the gap between consumers and homemade, nutritious meals. This service will provide a platform for users to order customized meals, manage their subscriptions, and track deliveries seamlessly.

## 1.2 Project Overview

The objective is to develop a comprehensive online tiffin management service that addresses the following challenges:

- **User Convenience:** Design a user-friendly interface accessible through web and mobile platforms that allows users to easily browse menus, customize meal plans, schedule deliveries, and manage subscriptions with minimal effort.
- **Meal Customization:** Implement a system that enables users to tailor their meal preferences, including dietary restrictions, portion sizes, and preferred cuisines, ensuring personalized and satisfying meal experiences.
- **Efficient Delivery Management:** Develop an efficient logistics system to optimize delivery routes, minimize delivery times, and ensure timely and accurate deliveries, maintaining the freshness and quality of the meals.
- **Payment and Billing:** Incorporate secure payment gateways and transparent billing systems to facilitate hassle-free transactions, including subscription renewals, refunds, and discounts.
- **Quality Assurance:** Establish quality control measures to maintain the consistency, freshness, and nutritional value of the meals, ensuring adherence to food safety standards and customer satisfaction.
- **Customer Support:** Provide responsive customer support channels, including live chat, email, and phone assistance, to address inquiries, resolve issues, and gather feedback for continuous improvement.
- **Scalability and Sustainability:** Design a scalable architecture that accommodates growth in user base and geographic expansion while maintaining cost-effectiveness, sustainability, and profitability.

- **Regulatory Compliance:** Ensure compliance with local regulations and food safety standards governing food preparation, packaging, handling, and delivery operations.
- **Data Security and Privacy:** Implement robust data security protocols to protect user information, including personal details, payment credentials, and order history, safeguarding against data breaches and unauthorized access.
- **Market Differentiation:** Identify unique selling propositions and competitive advantages to differentiate the service from existing tiffin delivery providers, attracting and retaining customers in a crowded market landscape.

By addressing these challenges, the online tiffin management service aims to revolutionize the way consumers access homemade, nutritious meals, offering convenience, customization, and quality while fostering healthy eating habits and lifestyle choices.

## 1.3 Hardware Specifications

This is a web-based application that can be deployed on a server and accessed through a web browser. The hardware specifications for the server hosting the application should be carefully considered to ensure optimal performance and reliability.

Since this is a project developed for educational purposes, the server used is online which allows to host for free. Therefore, the server hardware is generic.

In addition to the server hardware, each user accessing the Online Tiffin Service system will require a computer or mobile device with a web browser and an internet connection. Here are the recommended hardware specifications for these devices:

### 1.3.1 Computer

- Processor: Intel Core i5 or equivalent
- RAM: 8GB DDR4 or higher
- Storage: 256GB SSD or higher
- Operating System: Windows 10 or higher, macOS 10.14 or higher

### 1.3.2 Mobile Devices

- Processor: Generic Processor
- RAM: 3GB or higher
- Storage: 64GB or higher

- Operating System: iOS 14 or higher, Android 11 or higher

It is recommended that users have a stable internet connection with a minimum speed of 10Mbps for optimal performance of the web application. These hardware specifications are recommendations and can be adjusted based on the specific needs of the ecommerce and the number of users accessing the system. Adequate hardware resources are essential to ensure that the web- application operates efficiently and reliably.

## **1.4 Software Specifications**

The project Online Tiffin Service, is built using the MERN stack, which consists of MongoDB, Express, React, and Node.js. In addition to the MERN stack, the project utilizes several other software tools and libraries. Here are the software specifications for the system:

### **1.4.1 Front-End**

- React: A JavaScript library for building user interfaces
- Redux: A predictable state container for JavaScript apps
- Material UI: A popular UI component library for React

### **1.4.2 Back-End:**

- Node.js: A JavaScript runtime built on the Chrome V8 engine
- Express: A fast, unopinionated, and minimalist web framework for Node.js
- Passport: A popular authentication middleware for Node.js
- JWT: JSON Web Tokens for authentication

### **1.4.3 Database:**

- MongoDB: A NoSQL document-oriented database
- Mongoose: A MongoDB object modeling library for Node.js

### **1.4.4 Deployment:**

- Free to host online service to host both Frontend and Backend codebase.

### **1.4.5 Testing:**

- Unit testing by testing the software on different Edge Cases and Corner Cases.
- Used Thunderclient and Postman as the means.

These software specifications are the backbone of the project and have been carefully

selected to ensure optimal performance, security, and scalability. The software specifications may vary based on the specific needs of the hospital and the development team's expertise. It is essential to ensure that all software components are up-to-date and compatible with each other to ensure smooth operation of the system.

## CHAPTER 2

### LITERATURE SURVEY

#### 2.1 Existing System

The existing system of online tiffin services typically involves a platform where customers can order freshly prepared meals or tiffins online for delivery or pickup.

Below mentioned are some of the existing online tiffin services :

- **FreshMenu:** FreshMenu offers a variety of chef-prepared meals, including global cuisines and healthy options. They operate in several cities and offer both individual orders and subscription-based plans.
- **Box8:** Box8 specializes in Indian meals, offering a range of dishes including biryanis, wraps, and desserts. They focus on delivering freshly prepared meals with a quick delivery time.
- **Faasos:** Faasos provides a diverse menu of wraps, rice feasts, and desserts, with a focus on Indian flavors. They offer both individual orders and subscription-based meal plans.
- **Zomato Treats:** Zomato, a popular food delivery platform, also offers a subscription-based service called Zomato Treats. Subscribers can avail themselves of complimentary desserts with their meal orders from select restaurants.
- **Swiggy Daily:** Swiggy, another leading food delivery platform, has a service called Swiggy Daily that offers daily meal subscriptions. Customers can choose from a rotating menu of homestyle meals prepared by home chefs and professional kitchens.
- **Mumum Co:** Mumum Co offers nutritious, organic meals for babies and toddlers, delivered directly to customers' homes. They focus on providing healthy, preservative-free baby food options.
- **Yumist:** Yumist specializes in providing homestyle meals prepared with fresh ingredients. They offer a daily rotating menu and focus on delivering quality meals at affordable prices

#### 2.2 Proposed System

Overall, the system of online tiffin services aims to provide customers with convenient access to delicious and nutritious meals delivered to their doorstep, saving

them time and effort in meal planning and preparation. Some of the facilities and functionalities provided by the project are :

- **Online Ordering Platform:** Customers can browse through a variety of meal options available on the online platform. These options often include various cuisines, dietary preferences (like vegetarian, vegan, gluten-free), and meal plans (such as daily, weekly, or monthly subscriptions).
- **Menu Selection:** Customers can select the meals they want to order from the menu. They may have the option to customize their orders based on their preferences, such as selecting specific dishes, portion sizes, or adding extras like sides or beverages.
- **Subscription Options:** Many tiffin services offer subscription plans where customers can sign up for regular deliveries of meals. These subscriptions can be daily, weekly, or monthly, providing customers with convenience and a consistent supply of freshly prepared food.
- **Delivery or Pickup:** Customers can choose between delivery to their doorstep or opting for pickup from a designated location. Delivery options may include same-day delivery or scheduled deliveries based on the customer's preferences.
- **Payment:** Customers can pay for their orders securely through the online platform using various payment methods such as credit/debit cards, digital wallets, or cash on delivery, depending on the service provider's policies.
- **Feedback and Reviews:** Many online tiffin services encourage customers to provide feedback and reviews after receiving their orders. This helps in maintaining service quality and allows the service provider to improve their offerings based on customer preferences.
- **Customer Support:** Customer support channels are available for assistance with order inquiries, changes, or any issues that may arise during the ordering or delivery process. This can include online chat support, email, or phone assistance.
- **Quality and Freshness:** Online tiffin services focus on providing high-quality, freshly prepared meals to their customers. They often emphasize the use of fresh ingredients, healthy cooking techniques, and hygienic food preparation practices to ensure customer satisfaction and retention.

In the proposed system, apart from the above-mentioned facilities, the customer can also subscribe to a particular service provider for desired number of days and can cancel the subscription any time.

Also, the consumer can decide to take the monthly , yearly, or weekly subscriptions according to their choice and convenience. The consumer can also decide the plans for themselves such as veg/non-veg or the dinner, breakfast and lunch plans.

## **2.3 Feasibility Study**

The feasibility study is an essential step in determining the viability of a hospital managementsystem project. The study considers various factors, including technical, economic, operational, and legal, to assess whether the project is feasible and worthwhile. The following sections describe the feasibility study of the proposed meal management system project.

### **2.3.1 Technical Feasibility**

The technical feasibility of the project refers to whether the proposed system can be implemented using the available technology and resources. The proposed system is developed using the MERN stack, which is a widely used and popular web development stack. The MERN stack offers a robust and scalable architecture that can handle the complex requirements of a online meal management system. The system can be hosted on a cloud server or a local server, depending on the requirements.

### **2.3.2 Economic Feasibility:**

The economic feasibility of the project refers to whether the proposed system is financially viable and provides a positive return on investment. The proposed system offers several benefits, including increased efficiency, improved care, and reduced administrative workload. These benefits can translate into cost savings in the long run. The cost of implementing and maintaining the system is offset by the savings generated by the system. It is cheaper to enforce an online meal management system than to attract investment and gather capital for a traditional market space.

### **2.3.3 Operational Feasibility:**



The operational feasibility of the project refers to whether the proposed system is operationally viable and can be integrated into the existing operations. The proposed system is designed to be user-friendly and intuitive, requiring minimal training and support. The system can be customized to meet the user specific requirements and can be integrated with existing systems. The existing vendors can decide to expand their business with the help of the platform and in process will also provide variety for the customers.

#### **2.3.4 Legal Feasibility:**

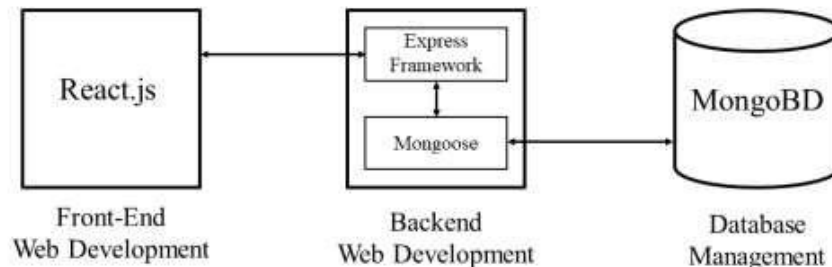
The legal feasibility of the project refers to whether the proposed system complies with the relevant laws and regulations. The proposed system is designed to comply with the relevant laws and regulations, such as HIPAA and GDPR. The system offers robust security features, including user authentication and access controls, to ensure the confidentiality and integrity of data.

Based on the feasibility study, it can be concluded that the proposed system is feasible and worthwhile. The system offers several benefits, including increased efficiency, improved care, and reduced administrative workload. The system can be customized to meet the specific requirements and can be integrated with existing systems. The system complies with the relevant laws and regulations and offers robust security features to ensure the confidentiality and integrity of data.

## CHAPTER 3

### SYSTEM ANALYSIS AND DESIGN

#### 3.1 Components of MERN stack



#### 3.1 Block Diagram of MERN stack

Block diagram of MERN stack is shown in Fig. 3.1. MERN stands for MongoDB, Express, React, Node, after the four key technologies that make up the stack.

- MongoDB — document database
- Express.js — Node.js web framework
- React.js — a client-side JavaScript framework
- Node.js — the premier JavaScript web server

Express and Node make up the middle (application) tier. Express.js is a server-side web framework, and Node.js is the popular and powerful JavaScript server platform. The MERN architecture allows you to easily construct a three-tier architecture (front end, back end, database) entirely using JavaScript and JSON.

These layers are as follows:

1. Web as front-end tier
2. Server as the middle tier
3. Database as backend tier

**The technologies used in this web application are :-**

- **Visual Studio Code**

Visual Studio Code provides developers with a new choice of developer tool that combines the simplicity and streamlined experience of a code editor with the best of what developers need for their core code-build-debug cycle. Visual Studio Code is the first code editor, and first cross-platform development tool – supporting OS X, Linux, and Windows – in the Visual Studio family.

At its heart, Visual Studio Code (VS Code) features a powerful, fast code editor great for day-to-day use. The Beta release of Code already has many of the features developers need in a code and text editor, including navigation, keyboard support with customizable bindings, syntax highlighting, bracket matching, auto indentation, and snippets, with support for dozens of languages.

- **React [1]**

ReactJS is JavaScript library used for building reusable UI components. React is a library for building composable user interfaces. It encourages the creation of reusable UI components, which present data that changes over time. Lots of people use React as the V in MVC. React abstracts away the DOM from you, offering a simpler programming model and better performance. React can also render on the server using Node, and it can power native apps using React Native. React implements one-way reactive data flow, which reduces the boilerplate and is easier to reason about than traditional data binding. React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies. React can be used as a base in the development of single-page, mobile, or server-rendered applications with frameworks like Next.js. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

- **Node.js [2]**

Node.js is a very powerful JavaScript-based platform built on Google Chrome's JavaScript V8 Engine. It is used to develop I/O intensive web applications like video streaming sites, single page applications, and other web applications. Node.js is open source, completely free, and used by thousands of developers around the world. As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications. Almost no function in Node.js directly performs I/O, so the process never blocks except when the I/O is performed using synchronous methods of Node.js standard library. Because nothing blocks, scalable systems are very reasonable to develop in Node.js.

- **MongoDB [4]**

MongoDB is a non-relational document database that provides support for JSON-like storage. The MongoDB database has a flexible data model that enables you to store unstructured data, and it provides full indexing support, and replication with rich and intuitive APIs. MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time. The main features of MongoDB are indexing, Ad-hoc queries , replication , load- balancing , file storage , aggregation , server-side JavaScript execution , capped collections , transactions. MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++. This tutorial will give you great understanding on MongoDB concepts needed to create and deploy a highly scalable and performance-oriented database.

- **Express.js [3]**

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications. Following are some of the core features of Express framework-

- Allows to set up middleware to respond to HTTP Requests.
- Defines a routing table which is used to perform different actions based on HTTP Method and URL.
- Allows to dynamically render HTML Pages based on passing arguments to templates.

Express.js, or simply Express, is a back-end web application framework for building RESTful APIs with Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.

- **API [5]**

The API that is being linked to the backend of the project to fetch the requests generated by the user and send a response to the server. Application Programming Interfaces (APIs) are constructs made available in programming

languages to allow developers to create complex functionality more easily. Client-side JavaScript has many APIs available to it — these are not part of the JavaScript language itself, rather they are built on top of the core JavaScript language, providing you with extra superpowers to use in your JavaScript code. They generally fall into two categories: Browser APIs are built into your web browser and can expose data from the browser and surrounding computer environment and do useful complex things with it. Third-party APIs are not built into the browser by default, and you generally must retrieve their code and information from somewhere on the Web.

## **3.2 Requirement Specifications**

The requirement specification outlines the functional and non-functional requirements of Online Tiffin Service. The system should meet the following requirements to provide an efficient and user-friendly system.

### **3.2.1 Functional Requirements :**

- a) User Authentication and Authorization:
  - User registration and login functionality.
  - Password recovery/reset options.
  - User roles (customer, administrator, vendor).
- b) Product Management:
  - Add, edit, delete products.
  - Categorization and tagging of products.
  - Product search and filtering options.
  - Inventory management.
- c) Shopping Cart:
  - Add, remove, and update items in the cart.
  - View total price and item count.
  - Save items for later.
- d) Checkout Process:
  - Secure and user-friendly checkout.
  - Multiple payment options .
  - Shipping address and payment details.
  - Order summary and confirmation.

- e) Order Management:
  - Order history
  - order confirmation and shipping updates.
  - Invoice generation.
- f) User Profile:
  - View and edit personal information.
  - Order history
  - Wishlist and favorites.
- g) Responsive Design:
  - Mobile-friendly design for various devices.
  - Cross-browser compatibility.

### 3.2.2 Non-Functional Requirements :

- a) **Performance:** The system should provide fast and responsive performance, even under heavy loads.
- b) **Security:** The system should offer robust security features, including user authentication and access controls, to ensure the confidentiality and integrity of data.
- c) **Usability:** The system should be user-friendly and intuitive, requiring minimal training and support.
- d) **Scalability:** The system should be scalable and able to handle the complex requirements of a meal management system.
- e) **Compatibility:** The system should be compatible with a wide range of web browsers and devices.
- f) **Accessibility:** The system should be accessible to users with disabilities, complying with relevant accessibility guidelines.

Based on the requirement specification, the system should provide an efficient and user-friendly solution for managing inventory/resources , tend to the customer needs and provide a platform for healthy competition between the providers. The system should meet the functional and non-functional requirements outlined above to ensure the system is scalable, secure, and easy to use.

### 3.3Diagrams

#### 3.3.1Data flow diagram

A data flow diagram (DFD) is a graphical representation of the flow of data through a system. It provides a clear and concise way to represent the various inputs, processes, outputs, and storage of data within a system. The primary goal of a DFD is to help in the design of an efficient system by breaking it down into smaller, more manageable pieces.

DFDs consist of a set of symbols and connectors that represent data sources, data flows, processes, data stores, and external entities. The symbols are used to represent the various components of the system, while the connectors indicate the flow of data between these components.

The four basic components of a DFD are as follows:

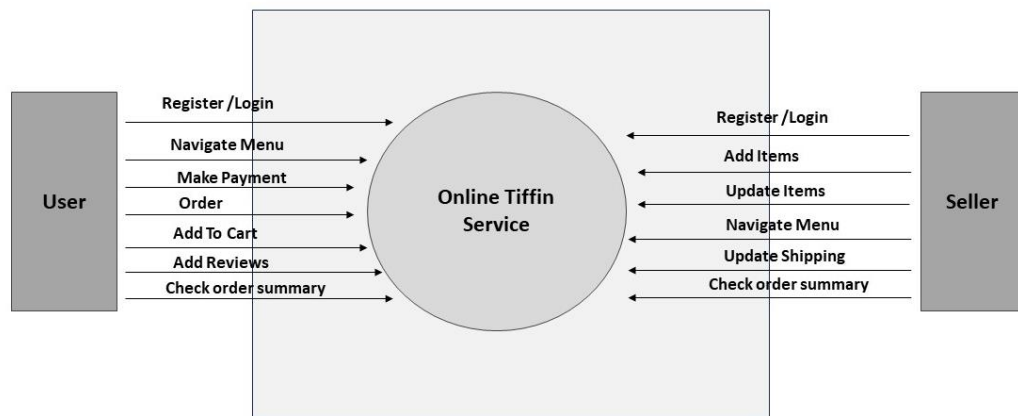
1. **Data Sources:** These represent the origin of the data that flows through the system. They may include internal or external sources.
2. **Data Flows:** These represent the movement of data from one component of the system to another. They may include both physical and logical flows.
3. **Processes:** These represent the transformation of data within the system. They may include calculations, comparisons, and other types of operations.
4. **Data Stores:** These represent the storage of data within the system. They may include databases, files, or other types of storage media.

DFDs can be used in a variety of contexts, including business process modeling, software engineering, and systems analysis and design. They are particularly useful in situations where complex systems need to be broken down into smaller, more manageable pieces for analysis and design purposes.

Figure 3.2 shows the level 0 DFD and Figure 3.3 shows level 1 DFD of the project.

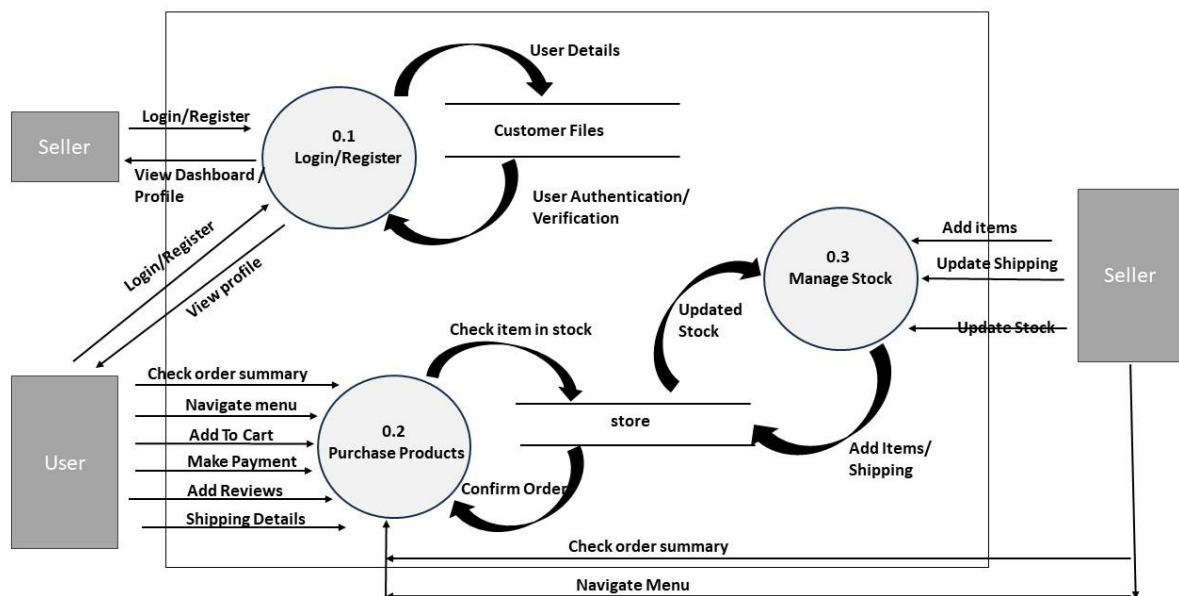
The following are the various DFDs in context of the project.

### Level 0 DFD :



**Fig 3.2 Level 0 Data Flow Diagram**

### Level 1 DFD :



**Fig 3.3 Level – 1 Data Flow Diagram**

### 3.3.2 Use Case Diagram

Figure 3.4 shows the use case diagram of the project. A use case diagram is a type of behavioral diagram in Unified Modeling Language (UML) that describes the behavior of a system from an external point of view. It provides a visual representation of the interactions between actors (users) and a system, highlighting the functionality that the system provides to its users.



The key components of a use case diagram are actors, use cases, and relationships. Actors represent the users of the system, while use cases represent the actions or tasks that the system can perform. Relationships connect actors to use cases, indicating the interactions between them.

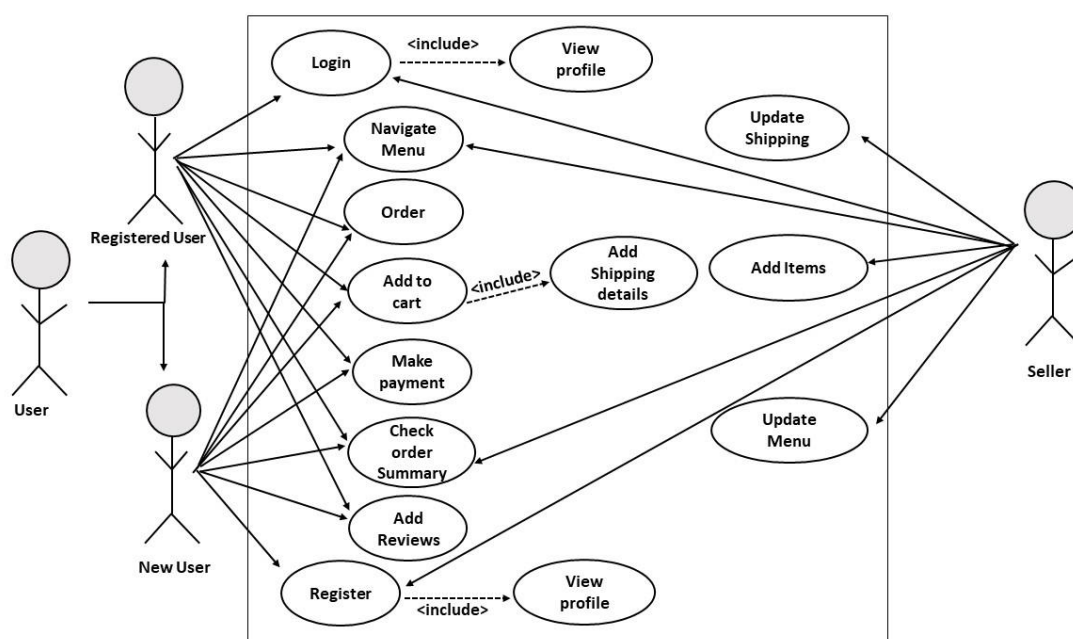
Use case diagrams can be used to model various systems, including software applications, business processes, and organizational structures. They are especially useful in identifying the functional requirements of a system, as well as the potential interactions between different actors and use cases.

Some common types of relationships that are typically represented in a use case diagram include:

1. Association: Indicates that an actor interacts with a use case.
2. Extend: Indicates that a use case can be extended by another use case under certain conditions.
3. Include: Indicates that a use case includes the functionality of another use case.

Use case diagrams are an important tool for system analysts, software engineers, and business analysts. They provide a clear and concise way to represent the functionality of a system, aiding in the design, development, and testing of software systems.

The following is the Use Case Diagram for the proposed system design:



### **Fig 3.4 Use Case Diagram**

#### **Use Case Description :**

##### **Actors :**

- Customer / User
- Admin

##### **Preconditions :**

- The user is either logged in or is a new user for the system.
  - The customer navigates to the home page to look for some listed products.
- a) The admin can add , delete , or update the category of products listed on the website.
  - b) The customer selects the "Shopping Cart" or "Checkout" option.
  - c) The system displays the contents of the shopping cart, including product details, quantities, and total price.
  - d) The customer reviews the items in the cart and decides to proceed to checkout.
  - e) The customer selects the "Proceed to Checkout" option.
  - f) The system prompts the customer to confirm the shipping address and provides the option to edit or add a new address.
  - g) The customer confirms the shipping address and proceeds to the payment section.
  - h) The system presents the available payment options (credit card, PayPal, etc.).
  - i) The customer selects a preferred payment method and enters the necessary payment details.
  - j) The system securely processes the payment and confirms the successful transaction.
  - k) The system generates an order confirmation page, displaying the order summary, transaction ID, and estimated delivery date.
  - l) The customer receives an email confirmation with the order details.

##### **Alternative Flows:**

- **Payment Failure:**

If the payment fails, the system provides an error message and allows the customer to retry or choose an alternative payment method.

- **Address Edit:**

If the customer decides to edit the shipping address, the system allows them to make changes before proceeding to payment.

- **Out-of-Stock Items:**

If any item in the shopping cart is out of stock, the system notifies the customer and allows them to remove or replace the item.

**Postconditions:**

- a) The customer's order is successfully placed.
- b) The inventory is updated to reflect the purchased items.
- c) The customer receives an email confirmation with order details.
- d) The order is visible in the customer's order history.

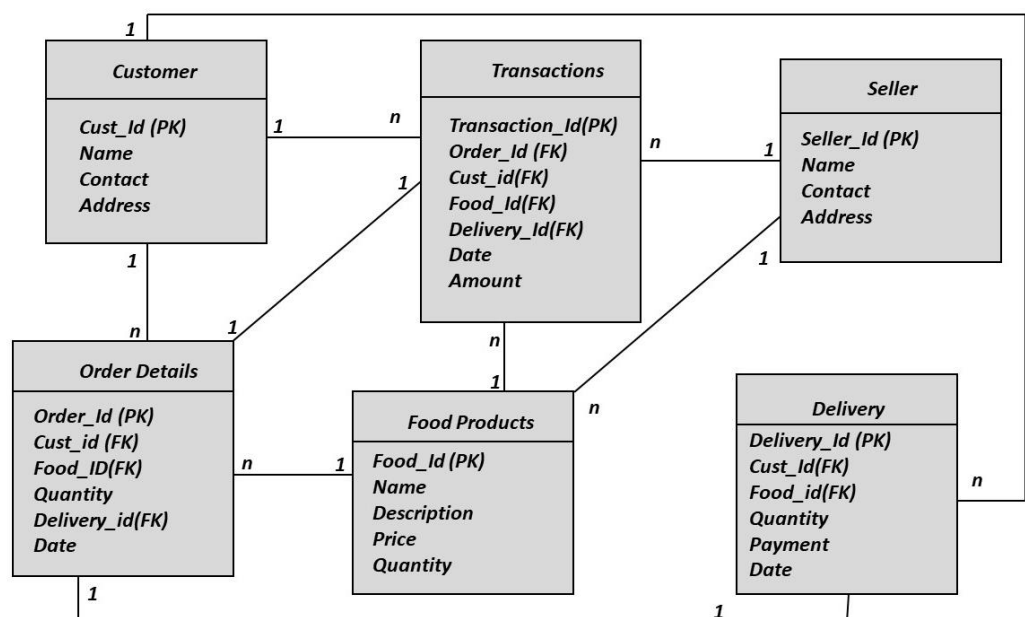
### 3.3.3 Schema Diagram

Figure 3.5 shows the Schema Diagram of the project. A schema diagram is a visual representation of the structure and organization of a database. It provides a graphical view of the relationships between different tables or entities within a database, along with the attributes or fields associated with each table. The purpose of a schema diagram is to illustrate the design of a database, showing how data is stored and how different pieces of information are related to each other.

Here are key components and concepts related to a schema diagram:

- **Tables or Entities:** In a database, data is typically organized into tables or entities. Each table represents a specific type of object or information, such as customers, products, orders, etc.
- **Attributes or Fields:** Tables contain attributes or fields, which represent the properties or characteristics of the objects they store. For example, a "Customer" table might have attributes such as "CustomerID," "FirstName," "LastName," etc.

- **Primary Keys:** A primary key is a unique identifier for each record in a table. It ensures that each record can be uniquely identified and distinguishes one record from another. In a schema diagram, primary keys are often denoted by underlining the attribute or field.
- **Foreign Keys:** Foreign keys are used to establish relationships between tables. A foreign key in one table refers to the primary key in another table. This linkage helps maintain referential integrity and ensures that the data is consistent across related tables.



**Fig. 3.5 Schema Diagram**

Relationships: Lines or connectors between tables represent relationships. These lines indicate how data in one table is related to data in another. Common types of relationships include one-to-one, one-to-many, and many-to-many

### 3.3.4 Sequence Diagram

Figure 3.6 shows the Sequence diagram of the project. A sequence diagram is a type of UML diagram used to visualize the interactions between objects or components in a system over time. It shows the sequence of messages passed between the different objects or components involved in a particular use case

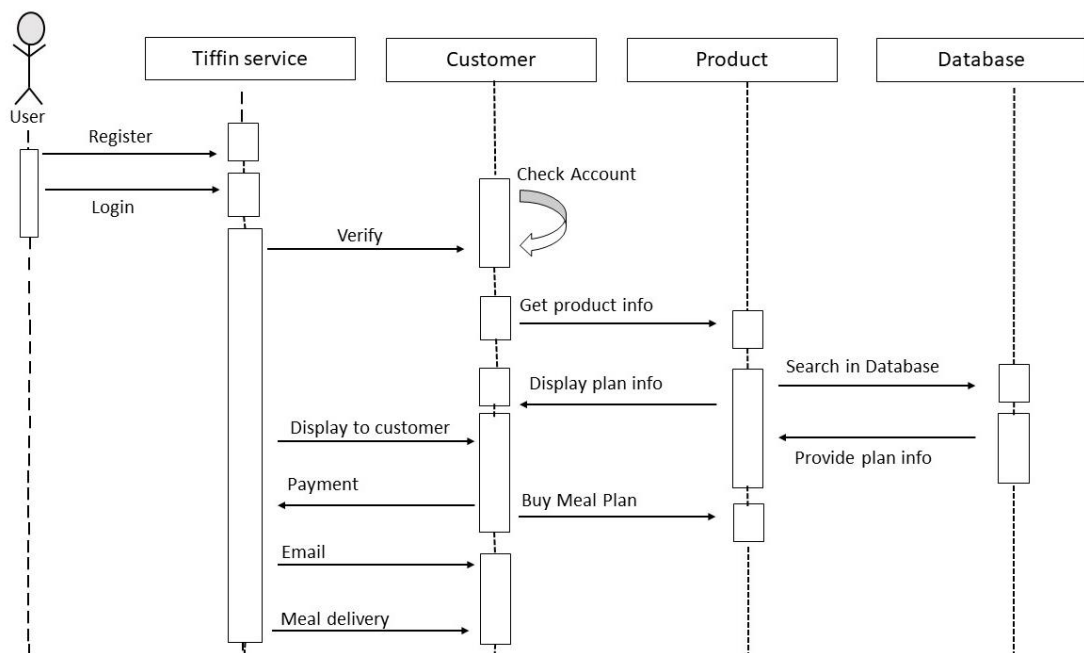
or scenario.

A sequence diagram consists of vertical lines representing the lifelines of the objects or components involved in the interaction, with messages passing horizontally between them. Each message indicates a specific action that is performed by an object or component in response to a previous message.

Sequence diagrams can be used to model a wide variety of interactions within a system, such as the flow of control in a use case scenario, the order in which processor services are executed, or the interaction between a user interface and a backend system.

Sequence diagrams are useful for both high-level design and detailed implementation, as they provide a clear and concise visualization of how different parts of a system interact with each other. They are especially helpful for identifying potential issues or bottlenecks in a system's performance, and can aid in the testing and debugging of complex systems.

The Sequence Diagram for the proposed system is as follows:



**Fig. 3.6 Sequence Diagram**

### 3.4 Design and Test Steps

The design and test steps are crucial components of the software development process. They ensure that the software being developed meets the requirements and

functions as intended. Here are the general steps involved in the design and testing phases:

## **1. Design Phase**

During the design phase, the following steps are typically followed:

- **Requirements gathering:** This is the process of gathering and documenting the requirements of the system. This can be done through interviews, questionnaires, and other techniques.
- **System design:** This involves creating a design specification for the system. This includes creating a high-level architecture of the system, defining the software components, and determining how they will interact with each other.
- **Detailed design:** This involves creating a detailed design of each component in the system. This includes specifying the interfaces, data structures, algorithms, and any other relevant details.

## **2. Test Phase**

During the test phase, the following steps are typically followed:

- **Test planning:** This involves creating a test plan that outlines the testing strategy, test objectives, and test cases to be executed.
- **Test case development:** This involves creating test cases based on the test plan. The test cases should be designed to ensure that all aspects of the software are thoroughly tested.
- **Test execution:** This involves executing the test cases and recording the results. Any defects or issues found during testing are documented and tracked.
- **Defect management:** This involves managing and tracking the defects found during testing. The defects are prioritized, assigned to developers, and tracked until they are resolved.

Overall, the design and testing phases are critical to the success of a software development project. By following these steps, software developers can ensure that the software being developed meets the requirements and functions as intended.

The project design was carried out in three phases which are as follows –

- **Designing the Frontend.**

- Creating the Backend.
- Integrating both Frontend and Backend.

Developing a robust frontend user interface is very important as it is the medium with which the user will interact with its data.

When done integrating the system, the testing procedure starts. System testing is the process of checking whether the developed system is working according to the objective and requirement.

### **Types of testing :**

Testing is an integral part of software development. It helps ensure that the software meets the desired quality and functional requirements. There are various types of testing that can be performed on software, including:

- **Unit Testing :**

Unit testing is a type of testing that focuses on testing individual units or components of the software. It involves writing automated tests for each unit and checking if they work as expected.

- **Integration Testing:**

Integration testing involves testing the interaction between different modules or components of the software. It is performed after unit testing to ensure that the different components of the software work together as expected.

- **System Testing:**

System testing involves testing the entire software system as a whole. It is performed after integration testing to ensure that the entire system meets the functional and non- functional requirements.

- **Acceptance Testing:**

Acceptance testing is performed to ensure that the software meets the requirements of the end-user. It is typically performed by the end-user or a representative of the end- user.

- **Regression Testing:**

Regression testing is performed after making changes to the software to ensure that the existing functionality of the software is not impacted by the changes.

- **Performance Testing:**

Performance testing is performed to ensure that the software performs well under different load conditions. It helps identify the maximum number of users or requests that the software can handle without performance degradation.

- **Security Testing:**

Security testing involves testing the software for vulnerabilities and weaknesses that can be exploited by malicious actors. It helps ensure that the software is secure and protects the user's data and privacy.

Moreover, to ensure the quality and functionality of software applications, the following Testing Procedures are also followed:

**Black-Box Testing:**

Black-box testing is a testing method where the tester has no knowledge of the internal workings of the software being tested. The tester only focuses on the inputs and outputs of the system, testing its functionality without any knowledge of the code behind the software. The purpose of black-box testing is to verify that the software works as expected from the user's perspective.

**Advantages of Black-Box Testing:**

- The tester doesn't require any knowledge of the software's internal workings.
- Tests are performed from the user's perspective, ensuring that the software meets the user's requirements.
- Black-box testing can help find issues with the software that are not apparent from the code.

**Disadvantages of Black-Box Testing:**

- The tester is limited to testing only what the software should do and not what it actually does.
- Black-box testing can be time-consuming and requires a thorough understanding of the software's requirements.



- It may be difficult to isolate the root cause of an issue with the software.

### **White-Box Testing:**

White-box testing is a testing method where the tester has knowledge of the internal workings of the software being tested. The tester has access to the code and can test the software at the code level. The purpose of white-box testing is to verify that the software is performing as expected from the developer's perspective.

### **Advantages of White-Box Testing:**

- The tester has access to the code and can test the software at a granular level.
- White-box testing can help identify coding errors and issues that could lead to vulnerabilities.
- The test coverage can be measured more accurately.

### **Disadvantages of White-Box Testing:**

- The tester needs to have knowledge of the software's internal workings.
- White-box testing can be more time-consuming than black-box testing.
- White-box testing may not be effective in testing the software from the user's perspective.

In conclusion, both black-box and white-box testing methods are valuable and have their advantages and disadvantages. Depending on the software being developed, one or both testing methods can be utilized to ensure the quality and functionality of the software.

## **3.5 Algorithms and Pseudo Code**

An algorithm is a set of instructions designed to solve a specific problem or perform a specific task. It is a step-by-step procedure that describes how to accomplish a task in a finite amount of time. Algorithms can be implemented in a variety of ways, including computer programs, flowcharts, or natural language descriptions.

In computer science, algorithms are used to perform tasks such as searching and sorting data, calculating mathematical functions, and solving complex problems. They are an

essential part of computer programming, as they provide a blueprint for developers to follow when creating software.

A well-designed algorithm is efficient, reliable, and easy to understand. It should be able to handle a wide range of input data, and produce accurate and consistent results every time it is executed.

Furthermore, Pseudocode is a method of writing out a computer program or algorithm in plain language, with some syntax that resembles the structure of a programming language. Pseudocode is not a programming language itself, but rather an informal way of describing the logic of a program before it is written in a specific programming language.

The purpose of pseudocode is to help programmers plan and organize their code before they begin writing it. By breaking down a program into smaller, more manageable steps, pseudocode can help identify potential issues or logic errors in a program before it is implemented in code. Additionally, pseudocode can serve as a form of documentation for a program, making it easier for others to understand and modify the code in the future.

Pseudocode typically uses standard programming concepts such as conditional statements (if/else), loops (for/while), and functions. However, the syntax used in pseudocode is not as strict as that of a programming language, and there are no formal rules for how it should be written. As such, pseudocode can be written in a variety of ways, depending on the preferences of the programmer.

### **3.6 Database Design**

To implement the database design, we have used a NoSQL database such as MongoDB. MongoDB provides a flexible data model that allows for easy storage and retrieval of data. You have also used an Object Data Modeling (ODM) library such as Mongoose to define the schema and perform operations such as CRUD (Create, Read, Update, Delete) on the database.

Designing the database for an e-commerce website involves structuring the data to support the storage and retrieval of information related to products, orders, customers, and other essential aspects of the business.

- **User Schema**

Figure 3.7 shows the user schema. It refers to the essential information that is stored for every user in the database, including the information that they provide at the time of registering or logging into the web application

```
const customerSchema = new mongoose.Schema({
  name: {
    type: String,
    minlength: 3,
    maxlength: 50,
    required: true,
  },
  email: {
    type: String,
    minlength: 3,
    maxlength: 50,
    required: true,
    unique: true,
  },
  password: {
    type: String,
    minlength: 3,
    maxlength: 255,
    required: true,
  },
  address: {
    area: {
      type: String,
      minlength: 5,
      maxlength: 255,
      required: true,
    },
    city: {
      type: String,
      minlength: 3,
      maxlength: 50,
      required: true,
    },
    pincode: {
      type: String,
      length: 6,
      required: true,
    },
  },
});
```

**Fig. 3.7 User Schema**

- **Product Schema**

Figure 3.8 shows the product schema of the website. It contains the information about a particular product that is displayed on the website , added by the vendor as admin. It will contain the info that is visible to the user during decision making on what to choose from the list. It contains the information about the vendor and ratings they have.

```
const tiffinVendorSchema = new mongoose.Schema({
  businessName: {
    type: String,
    minlength: 3,
    maxlength: 50,
    required: true,
  },
  email: {
    type: String,
    minlength: 3,
    maxlength: 50,
    required: true,
    unique: true,
  },
  password: {
    type: String,
    minlength: 3,
    maxlength: 255,
    required: true,
  },
  address: {
    area: {
      type: String,
      minlength: 5,
      maxlength: 255,
      required: true,
    },
    city: {
      type: String,
      minlength: 3,
      maxlength: 50,
      required: true,
    },
    pincode: {
      type: String,
      length: 6,
      required: true,
    },
  },
  phone: {
    type: String,
    length: 10,
    required: true,
  },
  rating: {
    numberOfRatings: { type: Number, required: true },
    currentRating: { type: Number, required: true },
    customerRatings: [
      {
        rating: { type: Number },
        customerId: { type: mongoose.Schema.Types.ObjectId },
        review: {
          title: {
            type: String,
            minlength: 3,
            maxlength: 50,
          },
          text: {
            type: String,
            minlength: 3,
            maxlength: 255,
          },
        },
      },
    ],
  },
}, { timestamps: true },
```

**Fig. 3.8 Product Schema (part 1)**

Figure 3.9 shows the second part of product schema that contains the meal plans as well as the price of the plans the vendor here is offering to the customer.

```
monthRate: {
  oldRate: {
    type: Number,
    min: 100,
    max: 200000,
  },
  discountRate: {
    type: Number,
    min: 100,
    max: 200000,
  },
  minMonthForNewRate: {
    type: Number,
    min: 2,
    max: 12,
  },
},
routine: {
  breakfast: {
    type: String,
    minlength: 3,
    maxlength: 50,
  },
  lunch: {
    type: String,
    minlength: 3,
    maxlength: 50,
  },
  dinner: {
    type: String,
    minlength: 3,
    maxlength: 50,
  },
},
pending: [
  {
    type: mongoose.Schema.Types.ObjectId,
  },
],
hasVeg: {
  type: Boolean,
  default: false,
},
});
```

**Fig. 3.9 Product Schema (part 2)**

- **Subscription Schema**

Figure 3.10 shows the subscription schema of the website. The customer can add subscription for any meal plan provided by any vendor according to their convenience and the vendor then is independent enough to either cancel or approve of the subscription request provided by the customer.

```

const subscriptionSchema = new mongoose.Schema({
  customerId: { type: mongoose.Schema.Types.ObjectId, required: true },
  vendorId: { type: mongoose.Schema.Types.ObjectId, required: true },
  customerName: {
    type: String, minlength: 3, maxlength: 50, required: true,
  },
  vendorName: {
    type: String, minlength: 3, maxlength: 50, required: true,
  },
  monthRateForEachOpted: {
    type: Number, min: 100, max: 200000, required: true,
  },
  opted: {
    breakfast: {
      type: Boolean, default: false,
    },
    lunch: {
      type: Boolean, default: false,
    },
    dinner: {
      type: Boolean, default: false,
    },
  },
  address: {
    area: {
      type: String, minlength: 5, maxlength: 255, required: true,
    },
    city: {
      type: String, minlength: 3, maxlength: 50, required: true,
    },
    pincode: {
      type: String, length: 6, required: true,
    },
  },
  durationDays: {
    type: Number, min: 30, max: 1000, required: true,
  },
  isAccepted: {
    type: Boolean, default: false,
  },
  startDate: {
    type: Date,
  },
  endDate: {
    type: Date,
  },
});

```

Fig. 3.10 Subscription Schema

# CHAPTER 4

## RESULTS / OUTPUTS

### 1. Signup Page

Figure 4.1 shows the customer registration page and Figure 4.2 shows the vendor registration page of the website.

A signup page, also known as a registration page, is a web page that allows users to create an account or sign up for a service, platform, or website. It is a crucial component of user onboarding processes and is designed to collect necessary information from users to create their accounts.

Mom's Raasoi Login Register

Customer Register

name

email

password

area

city

pincode

Register

Not a Customer? Tiffin Vendor

Already Registered? Login

**Fig. 4.1 Customer Signup Page**

Mom's Raasoi Login Register

Tiffin Vendor Register

business name

email

password

area

city

pincode

phone

monthly rate

monthly discounted rate

minimum month for discount

breakfast items

lunch items

dinner items

Veg option present? ☐

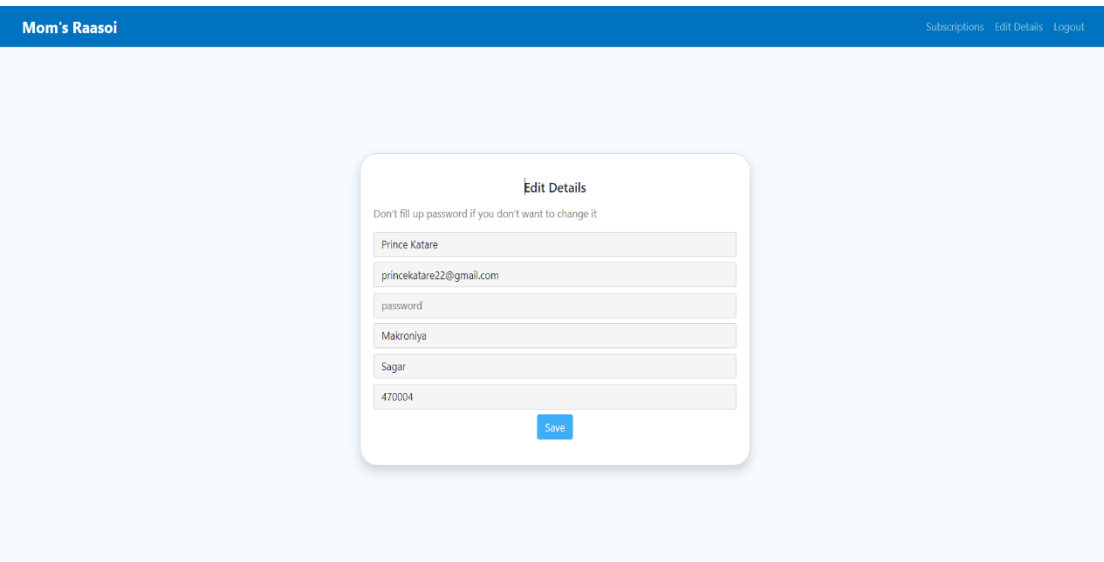
Register

Not a Tiffin Vendor? Customer

Already Registered? Login

**Fig. 4.2 Vendor Signup Page**

Figure 4.3 shows the edit details page. After Signup, the customer as well the vendor can choose to edit their details/ information from the tab provided on the upper right corner of the screen. In this case , they can change their password name or their additional information accordingly.



**Fig. 4.3 Edit Details Page**

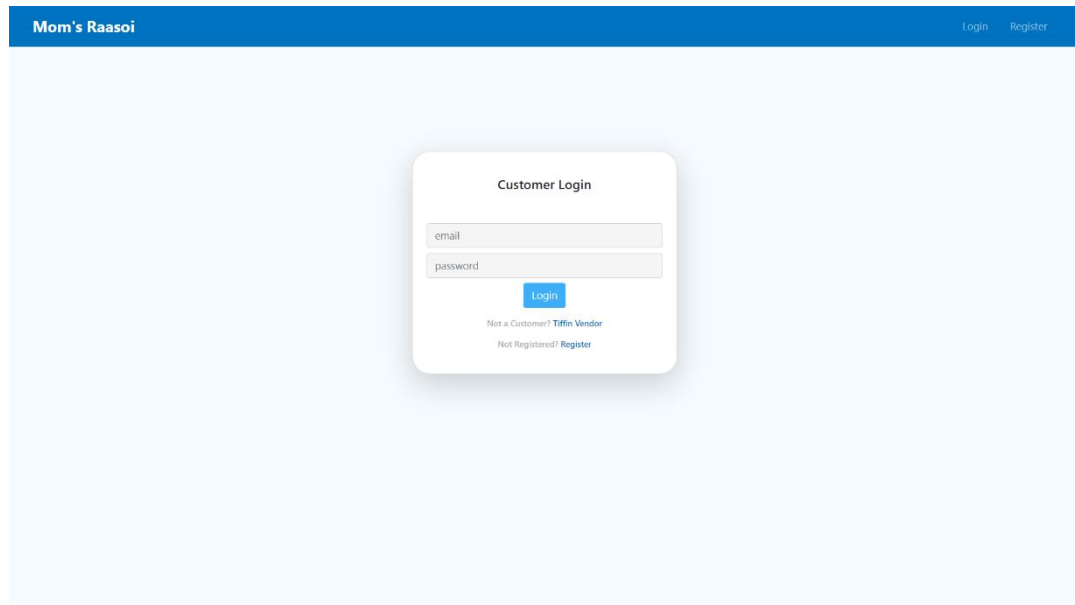
## **2. Login Page**

The figure 4.4 shows the customer login page and the figure 4.5 shows the vendor login page.

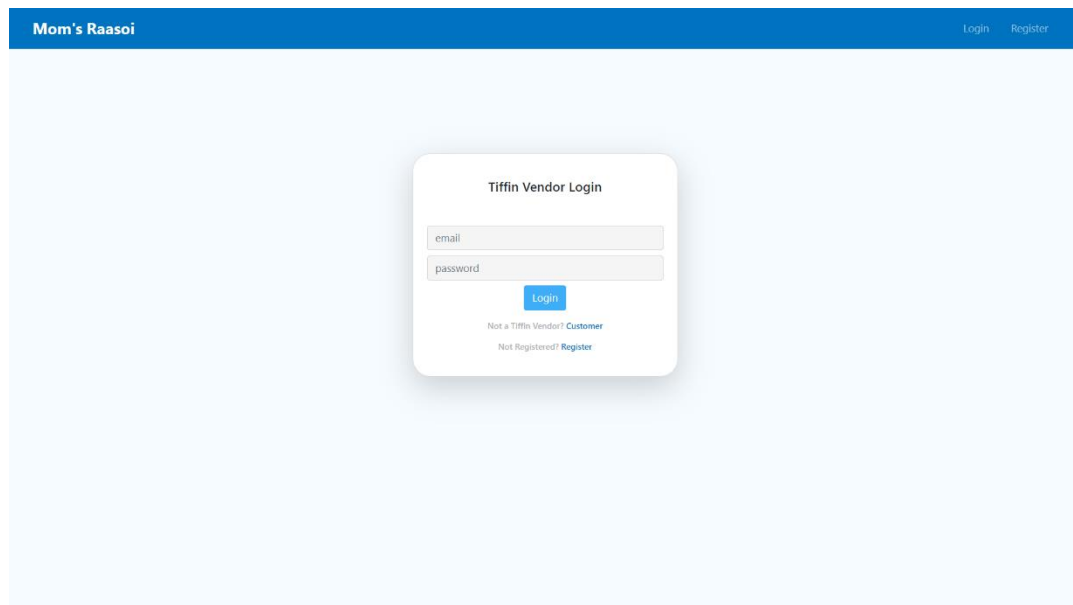
The login page requires users to enter their username and password, which are then verified against the database of authorized users. If the credentials are correct, the user is granted access to the system and directed to their respective dashboard, either the admin dashboard or the staff dashboard.

If the user enters incorrect login credentials, an error message is displayed, indicating that the username or password is incorrect. The user will then be prompted to re-enter their credentials.





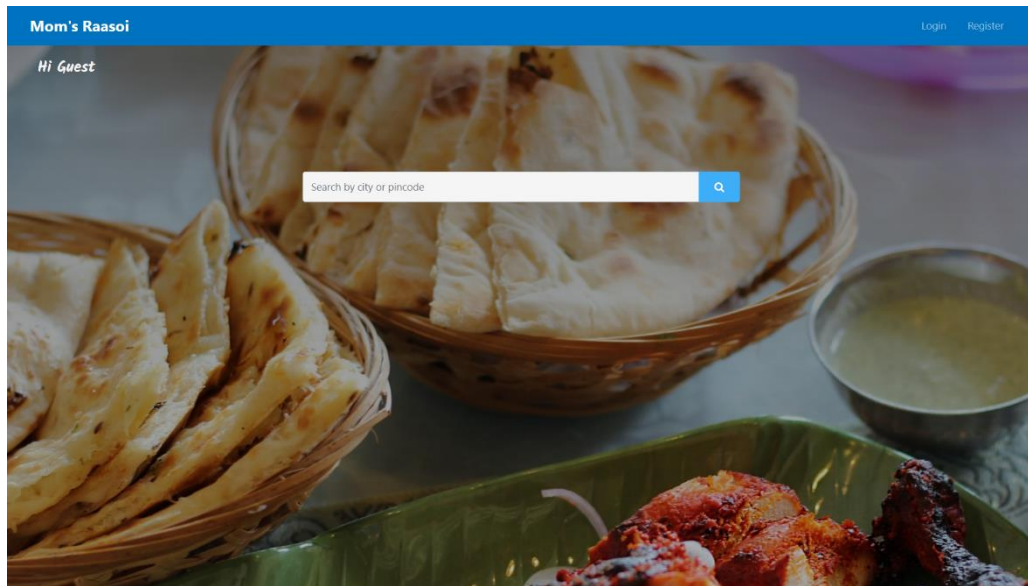
**Fig. 4.4 Customer Login Page**



**Fig. 4.5 Vendor Login Page**

### **3. Home Page**

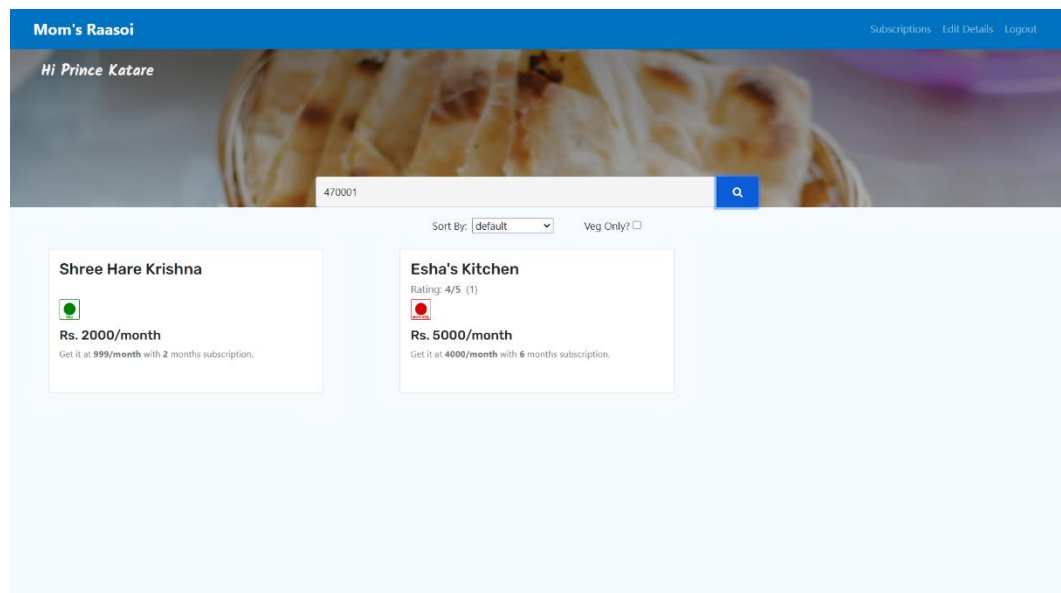
Figure 4.6 shows the landing page of the website. It consists of various navigation options and lists the brief products available to us. Once a customer has registered or logged in to their account, they will land on this page and can then further search for vendors and availability of meals.



**Fig. 4.6 Home Page**

#### **4. Search vendor by Pin-code**

Figure 4.7 shows the search vendor page. This is a feature specially for the customers where the customer can look for possible vendors near him/her by searching through the search bar by desired pin-code. All the vendors in that area will be displayed and the user can also check the veg / non-veg type of food as per requirement and preference.



**Fig. 4.7 List of Available Vendors (By Pin-Code)**

## 5. Add subscription

Figure 4.8 shows the add meal subscription page. The customer can choose to add subscription after setting the desired meal plan for as many days required. Customer can also set the type (veg/non-veg) and time of meal to be delivered (breakfast/lunch/dinner or all of them).

Also, the customer can choose to terminate the subscription anytime they want.

The screenshot displays the 'Add Meal Subscription' interface. At the top, there's a blue header with 'Mom's Raasoi' and navigation links for 'Subscriptions', 'Edit Details', and 'Logout'. Below the header, a filter dropdown is set to 'All subscriptions'. The main content area features a subscription card for 'Esha's Kitchen'. The card lists the following details: 'Meals Opted : breakfast lunch dinner', 'Address: Makroniya,Sagar-470004', 'Status: pending', 'Duration : 1 months', and 'Rs. 15000'. A red 'Cancel' button is located at the bottom right of the card.

**Fig. 4.8 Add Meal Subscription**

## 6. Approve subscription request (by vendor)

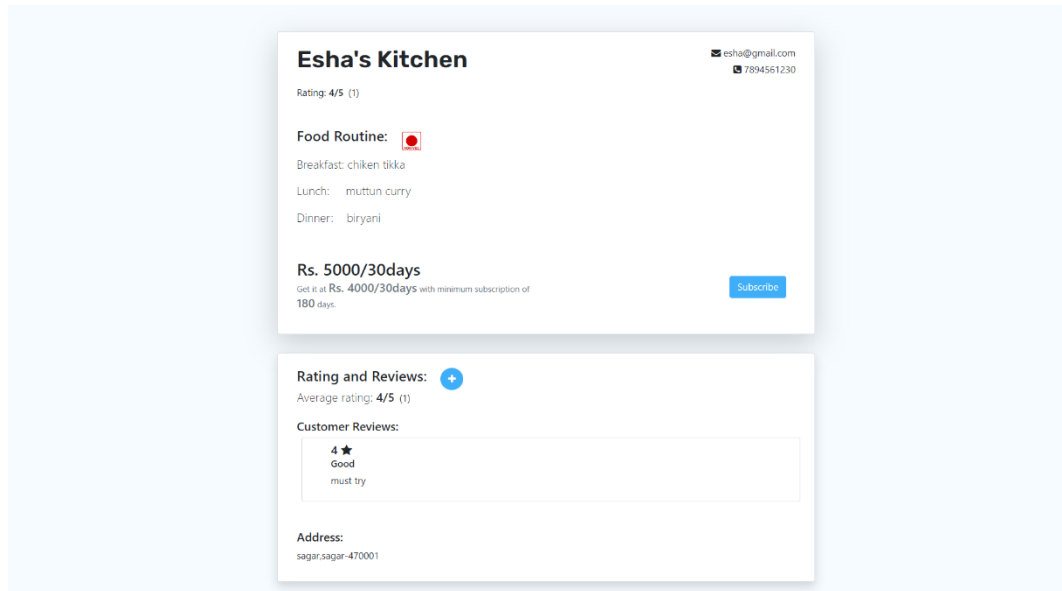
Figure 4.9 shows the approve/ cancel subscription page. Once the subscription is added by the customer, the vendor will receive a request to either accept the subscription or cancel it accordingly. This is a vendor exclusive feature where the vendor can decide to approve or cancel a subscription in accordance to their availability and convenience.

The screenshot shows the 'Approve/cancel subscription request' interface. It has the same blue header as Figure 4.8. The filter dropdown is also set to 'All subscriptions'. The main content area displays a subscription card for 'Prince Katare'. The card details are: 'Meals Opted : breakfast lunch dinner', 'Address: Makroniya,Sagar-470004', 'Status: pending', 'Duration : 1 months', and 'Rs. 15000'. At the bottom right of the card, there are two buttons: a yellow 'Approve' button and a red 'Cancel' button.

**Fig. 4.9 Approve/cancel subscription request**

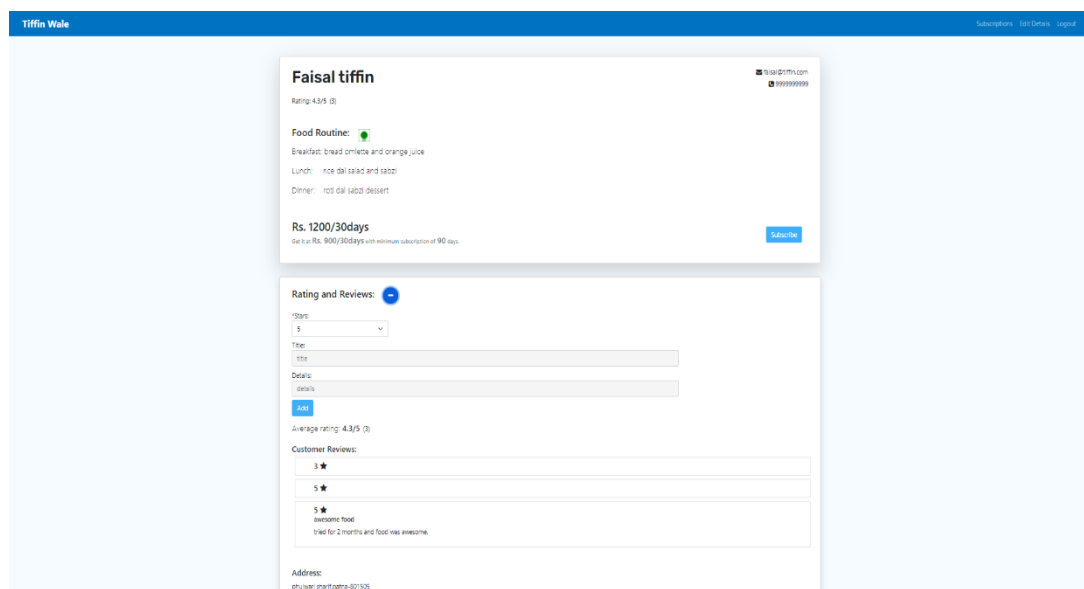
## 7. Add / See Reviews

Figure 4.10 shows the see reviews page. Here, the reviews added previously can be seen on the page of the vendor. The customers after having consumed the product after subscribing to the vendor, can add reviews which can be displayed.



**Fig. 4.10 See Reviews Page**

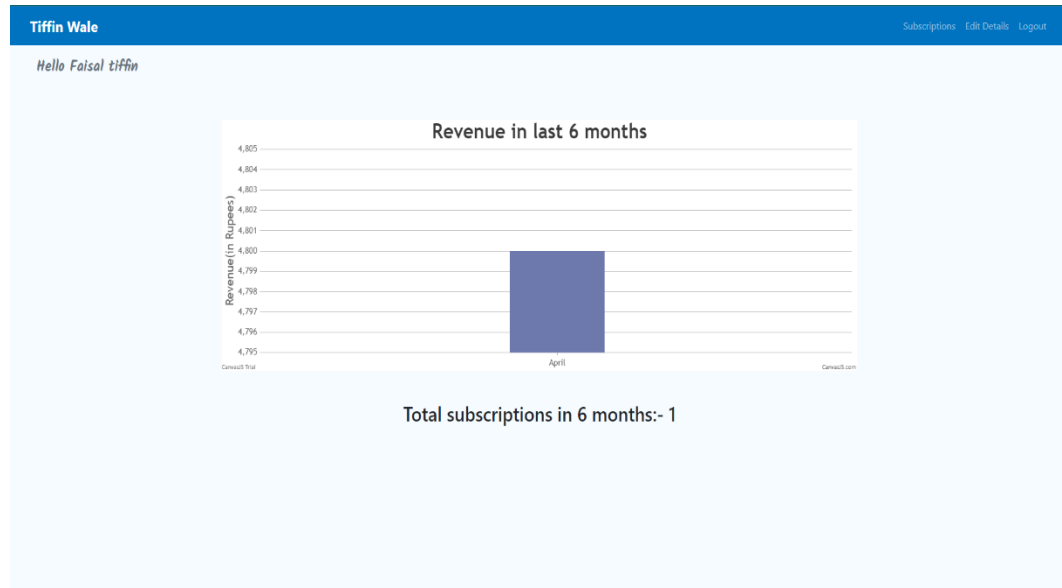
Figure 4.11 shows the add reviews page. After subscribing, the customer can add reviews (comments and ratings) to help others decide which vendor suits them best.



**Fig. 4.11 Add Reviews Page**

## 8. See the Revenue (for Vendor)

Figure 4.12 shows the revenue calculation page. The vendor can assess their revenue at the end of the 6<sup>th</sup> calendar month by checking the stats on the platform. It shows the revenue from subscriptions by consumers as it calculates the amount attained after the 6<sup>th</sup> calendar month.



**Fig. 4.12 Revenue Chart Page**

## **CHAPTER 5**

### **CONCLUSIONS / RECOMMENDATIONS**

In conclusion, the development and implementation of the online tiffin service project have been a significant milestone, marked by substantial achievements and contributions to our business objectives. The project aimed to create a seamless online meal providing experience for our customers, and through meticulous planning and execution, we have successfully achieved many of our goals.

The platform now provides a user-friendly interface, robust product catalog, and secure checkout process, ensuring a positive and efficient experience for our customers. The integration of various payment gateways and responsive design has enhanced accessibility and convenience, contributing to increased user engagement.

The implementation of analytics and tracking tools has enabled us to gather valuable insights into user behavior, purchase patterns, and website performance. These insights will play a crucial role in ongoing optimization efforts and marketing strategies.

#### **Recommendations:**

1. **Mobile Optimization:**

Given the increasing trend in mobile commerce, prioritize further optimization for mobile devices to enhance the user experience on smartphones and tablets.

2. **Personalization and User Recommendations:**

Implement personalized user experiences and product recommendations based on user behavior and preferences. This can be achieved through data-driven algorithms and machine learning.

3. **Customer Feedback and Surveys:**

Implement a system for collecting customer feedback and conducting surveys to understand user satisfaction, identify pain points, and gather suggestions for improvement.

## REFERENCES

- [1] N. I. Naim, “ ReactJS: An Open-Source JavaScript library for front-end development,” Bachelor of Engineering , Dept. Information Technology , Metropolis University of Applied Sciences , 2017 [Accessed : 21-Jan -2024]
  
- [2] H. Shah, “ Node.js Challenges in Implementation ,” Master of Science, Dept. Computer Science , SZABIST , Dubai , 2017 [Accessed : 08-Mar-2024]
  
- [3] Geeks for Geeks (2022, Apr. 19). Express.js Application Complete Reference [Online]. Available : <https://www.geeksforgeeks.org/express-js-application-complete-reference/> [Accessed : 20-Feb-2024]
  
- [4] MongoDB (2018, Apr. 27) NoSQL Databases Pros and Cons. [online] Available: <https://www.mongodb.com/scale/nosql-databases-pros-and-cons> [Accessed : 5-Apr-2024]
  
- [5] WordPress : Reference | REST API Handbook (2013, Mar 13). [online] Available: <https://developer.wordpress.org/rest-api/reference/> [Accessed : 11-Apr-2024]

# APPENDICES

## 7.1 Appendix 1 – ReactJS and Handlebars

### Introduction

ReactJS is a popular JavaScript library used for building user interfaces. It was developed by Facebook and released in 2013 as an open-source project. ReactJS follows a component-based approach to building UIs, where each component represents a small part of the user interface. Components can be reused and combined to build complex UIs.

### Features

Some key features of ReactJS include:

- Virtual DOM: ReactJS uses a virtual DOM to update the actual DOM efficiently.
- JSX: ReactJS allows developers to write HTML-like syntax in JavaScript code using JSX.
- One-way data binding: ReactJS follows a unidirectional data flow, where data flows from parent to child components.
- Server-side rendering: ReactJS can be used for server-side rendering to improve page load times and SEO.

### Advantages

Some advantages of using ReactJS include:

- Fast rendering: ReactJS uses a virtual DOM and efficient diffing algorithms to update the DOM quickly.
- Reusability: ReactJS components can be reused in multiple parts of an application, making development faster and more efficient.
- Community: ReactJS has a large and active community, with many libraries and tools available to make development easier.

### Disadvantages

Some disadvantages of using ReactJS include:

- Steep learning curve: ReactJS can be difficult to learn for developers who are new to JavaScript or component-based UI development.
- Complexity: ReactJS can become complex as applications grow, making it harder to maintain and debug.



- Limited functionality: ReactJS only provides the view layer of an application, so developers need to use other libraries or frameworks to handle state management, routing, and other functionality.

### **Handlebars:**

Handlebars is a popular templating engine for creating dynamic HTML templates. It allows developers to create templates with placeholders for dynamic data, and then use JavaScript code to fill in those placeholders with actual data.

To use Handlebars in a web application, you typically need to include the Handlebars library in your project, along with any other dependencies. Once you've done that, you can create Handlebars templates by writing HTML code with special placeholders, called "handlebars expressions," which look like this: `{{variable}}`. The variable inside the double braces represents the dynamic data that will be filled in at runtime.

Handlebars also supports more advanced features, such as conditionals, partials, and helpers, which can be used to create more complex templates.

## **7.2 Appendix 2 – NodeJS**

Node.js is an open-source, cross-platform runtime environment that enables developers to run JavaScript code outside of a web browser. It's built on the V8 JavaScript engine, the same engine that powers Google Chrome, and provides an event-driven, non-blocking I/O model that makes it ideal for building scalable, high-performance applications.

Node.js has a vast ecosystem of modules and packages available through the npm (Node Package Manager) registry, which makes it easy to extend its functionality and reuse existing code. It's commonly used for building server-side applications, such as web servers, APIs, and microservices, as well as command-line tools and desktop applications.

### **Some of the key features of Node.js include:**

- Asynchronous I/O: Node.js provides an event-driven, non-blocking I/O model that allows multiple I/O operations to be performed

concurrently without blocking the main thread, which makes it ideal for building scalable, high-performance applications.

- CommonJS modules: Node.js uses the CommonJS module system, which provides a simple and standardized way of organizing and sharing code between different files and modules.
- Built-in modules: Node.js provides a set of built-in modules, such as `http`, `fs`, and `path`, that provide core functionality for building web servers, file systems, and other common tasks.
- npm: Node.js comes with npm (Node Package Manager) pre-installed, which provides access to a vast ecosystem of modules and packages that can be easily installed and managed.
- Cross-platform: Node.js is available on a wide range of platforms, including Windows, macOS, Linux, and many others.

Overall, Node.js is a powerful and flexible runtime environment that is well-suited for building a wide range of applications, from simple command-line tools to complex web servers and APIs. Its event-driven, non-blocking I/O model and extensive ecosystem of modules and packages make it a popular choice among developers for building scalable, high-performance applications.

## STUDENT PROFILE



**Name :** Esha Mitra

**Enrolment :** 201B107

**Email :** [esha272103@gmail.com](mailto:esha272103@gmail.com)

**Address :** Jaypee University of Engineering and Technology, Guna

**Contact :** 9305588057



**Name :** Prince Katare

**Enrolment :** 201b197

**Email :** [princekatare22@gmail.com](mailto:princekatare22@gmail.com)

**Address :** Jaypee University of Engineering and Technology , Guna

**Contact :** 7024733708



**Name :** Navanshu

**Enrolment :** 201b164

**Email :** [navanshu191@gmail.com](mailto:navanshu191@gmail.com)

**Address :** Jaypee University of Engineering and Technology , Guna

**Contact :** 7728052967