

Capstone Project – III
Ghar Ka Khana
U. V. Patel College of Engineering



**Ganpat
University**
॥ विद्यया समाजोत्कर्षः ॥

**U.V. Patel
College of
Engineering**



Internal Guide:
Prof. Barkha Mehta

Prepared By:
Mr. Joel Roy (21012021037) (IT)
Mr. Abhay Hingrajiya (22012012024) (CE)
Mr. Adarsh Lodhi (22012012036) (CE)
Mr. Dhruv Bhatt (22012022002) (IT)

B. Tech Semester-VII
(Computer Engineering)
Nov-Dec, 2024

Submitted to,
Department of Computer Engineering
U.V. Patel College of Engineering
Ganpat University, Ganpat Vidyanagar - 384 012



U.V. Patel
College of
Engineering



07/12 /2024

CERTIFICATE

TO WHOM SO EVER IT MAY CONCERN

This is to certify that Mr. Abhay Hingrajiya student of **B.Tech. Semester VII (Computer Engineering)** has completed his full semester Capstone Project-III work titled “Ghar ka khana” satisfactorily in partial fulfillment of the requirement of a Bachelor of Technology degree in Computer Engineering of Ganpat University, Ganpat Vidyanagar, Mehsana in the year 2024-2025.

Project Guide

Prof. Barkha Mehta

Dr. Paresh Solanki,

Head, Computer Engineering



U.V. Patel
College of
Engineering



07/12 /2024

CERTIFICATE

TO WHOM SO EVER IT MAY CONCERN

This is to certify that Mr. Adarsh Lodhi student of **B.Tech. Semester VII (Computer Engineering)** has completed his full semester Capstone Project-III work titled “Ghar ka khana” satisfactorily in partial fulfillment of the requirement of a Bachelor of Technology degree in Computer Engineering of Ganpat University, Ganpat Vidyanagar, Mehsana in the year 2024-2025.

Project Guide

Prof. Barkha Mehta

Dr. Paresh Solanki,

Head, Computer Engineering

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success. We at this moment take this opportunity to thank everyone who has directly or indirectly helped us in preparing our project on “Ghar Ka Khana”.

We are thankful to Prof. Barkha Mehta who gave nice guidance & support to us to fulfill and complete the project within the duration. Thanks to All those valuable suggestions and comments the text has benefited so much. We express our gratitude towards all faculty members who have guided and directed us at every moment to fulfill our mission and produce this work in front of you. We hope you will undoubtedly find the matter interesting and informative as well. At last, thanks to all users for their interest in our work.

ABSTRACT

'Ghar Ka Khana' is a dynamic web application designed to bridge the gap between home cooks and customers seeking authentic homemade meals. This innovative platform allows food providers to easily list and manage their dishes, offering detailed information such as dish name, price, flavor, and availability. Users can explore a diverse range of dishes, view comprehensive details, and place orders seamlessly, enhancing their culinary experience.

The application features a user-centric design with a focus on usability and security. It incorporates secure authentication mechanisms to safeguard user information and ensure a smooth, trustworthy experience for all participants. Additionally, 'Ghar Ka Khana' includes functionalities for managing delivery personnel, enabling them to handle their delivery tasks efficiently.

Built with modern web technologies, the platform uses React for building interactive user interfaces, Redux for state management, and Tailwind CSS for responsive and stylish design. The integration of these technologies provides a robust and scalable solution, enhancing the overall user experience. With an emphasis on simplicity, accessibility, and efficiency, 'Ghar Ka Khana' strives to make homemade food not only accessible but also enjoyable, fostering a vibrant community of food enthusiasts and providers.

The project also focuses on scalability and performance, ensuring that the application can handle increasing numbers of users and transactions smoothly. The commitment to continuous improvement and user feedback drives the development of additional features and enhancements, making 'Ghar Ka Khana' a versatile and future-ready platform for the culinary community.

INDEX

1	Introduction.....	1
1.1	Project Overview.....	1
1.2	Background.....	1
1.3	Purpose.....	2
1.3.1	Problem Statement.....	2
1.3.2	Project Aim.....	3
1.3.3	Project Objective.....	3
1.4	Project Scope.....	3
1.5	Impact, Significance, and Contributions.....	3
1.6	Definition, Acronyms, and Abbreviations.....	3
1.6.1	Definitions.....	3
1.6.2	Acronyms and Abbreviations.....	3
1.7	Technology Stack.....	4
1.8	Developer Hardware configuration.....	4
2	Literature Review.....	6
2.1	Overview of Food Delivery Platforms.....	7
2.2	Literature Survey Analysis Table.....	7
3	Software Requirement Specification.....	8
3.1	Functional Requirement.....	8
3.1.1	Functional Requirement of Module Admin.....	8
3.1.2	Functional Requirement of Module User.....	8
3.2	Non-Functional Requirement.....	8
4	Diagrams.....	9
4.1	Flow Chart (User).....	11
4.2	Flow Chart (Admin).....	12
4.3	Use-Case Diagram.....	13
4.4	E-R diagram.....	14
4.5	Sequence Diagram.....	15
4.6	Activity Diagram.....	17
4.7	Activity Diagram(Admin).....	17
4.8	Activity Diagram(User).....	18
4.9	Class Diagram.....	19
4.10	State Diagram.....	20

5	Implementation Details.....	21
5.1	Data Dictionary.....	21
5.8	Program Code.....	24
6	Testing.....	31
6.1	Unit Testing.....	31
7	Prototypes.....	33
8	Conclusion.....	42
9	Future Work.....	43
10	References.....	44

LIST OF FIGURES

Figure 4.1 Flow Chart (Delivery Boy).....	10
Figure 4.2 Flow Chart (Consumer).....	11
Figure 4.3 Flow Chart (Provider).....	12
Figure 4.4 Use-Case Diagram.....	13
Figure 4.5 E-R Diagram.....	14
Figure 4.6 Sequence Diagram.....	15
Figure 4.7 Activity Diagram (Consumer).....	17
Figure 4.8 Activity Diagram (Provider).....	18
Figure 4.9 Activity Diagram (Delivery Boy).....	19
Figure 4.10 Class Diagram.....	20
Figure 7.1 Sign Up Page.....	33
Figure 7.2 Sign in Page.....	34
Figure 7.3 Provider Home page.....	35
Figure 7.4 Consumer Home Page.....	36
Figure 7.5 Delivery boy.....	37
Figure 7.6 Provider order Page.....	37
Figure 7.7 Provider Confirm order Page.....	38
Figure 7.8 Admin login Page.....	39
Figure 7.9 Food provider Dashboard.....	39
Figure 7.10 Food Consumer Dashboard.....	40
Figure 7.11 Food Delivery Dashboard.....	40
Figure 7.12 Verify User'sPage.....	41
Figure 7.13 Add new admin Page.....	41

LIST OF TABLES

Table 2.1 Literature Survey Analysis Table.....	7
Table 5.1 DeliveryBoyTable.....	21
Table 5.2 DishInfo Table.....	21
Table 5.3 DishStatusTable.....	22
Table 5.4 FoodConsumerTable.....	22
Table 5.5 FoodProviderTable.....	22
Table 5.6 ItemDetailsTable.....	23
Table 5.7 OrderInfoTable.....	23
Table 6.1 Testing Table.....	31

1 INTRODUCTION

1.1 Project Overview

- "Ghar ka Khana" is a web-based platform that connects students and hostelers with local mothers who prepare homemade food. The platform enables users to browse, order, and pay for a variety of homemade dishes. The system is developed using Node.js, React, and MongoDB.

1.2 Background

- With the increasing demand for healthy and home-cooked meals among students and hostelers, "Ghar ka Khana" aims to bridge the gap by providing a convenient way to access homemade food. The platform also offers local mothers an opportunity to share their culinary skills and generate income.

1.3 Purpose

- The purpose of 'Ghar Ka Khana' is to provide a convenient platform for students and hostelers to access affordable, healthy, and home-cooked meals prepared by local mothers. The platform empowers food providers to share their culinary skills and generate income, while promoting a sustainable and community-driven ecosystem for wholesome dining options.

1.3.1 Problem Statement

- Students and hostelers often lack access to healthy, home-cooked meals, leading them to rely on fast food and unhealthy options. Local mothers, on the other hand, may have the ability to cook nutritious meals but lack a platform to reach potential customers.

1.3.2 Project Aim

- The aim of "Ghar ka Khana" is to develop a comprehensive and user-friendly online platform that bridges the gap between students or hostelers and local mothers who specialize in homemade cooking. This platform aspires to offer a convenient solution for students and hostelers who often struggle to find healthy, home-cooked meals, by providing them with easy access to a variety of nutritious dishes made with care and authenticity.
- In addition to catering to the dietary needs of students and hostelers, "Ghar ka Khana" also serves as an empowering tool for local mothers. It creates an opportunity for them to showcase their culinary skills and share their passion for cooking with a broader audience. By monetizing their expertise, these mothers can generate a sustainable income, contributing to their financial independence.

1.3.3 Project Objective

- Develop a user-friendly web interface for students and hostellers to browse and order homemade food.
- Create a backend system that supports user authentication, order management, and payment processing.
- Provide a separate interface for mothers to manage their dish listings and orders.

1.1 Ensure data security and privacy for all users. Project Scope

- Initial functional requirements will be: -
 - Secure registration and manage the profile of Users.
 - Creating a Shopping cart so that users can shop 'n' no. of templates and checkout finally with the entire shopping carts.
 - Adequate payment mechanism and gateway for all popular credit cards, debit cards, and other relevant payment options (Like UPI, and net banking) as available
 - Browsing through the "MMW" to see the readymade templates that are there in each category of product website templates like Coffee Shop websites, Automobile Company's Website, Local Business Websites, etc.

1.2 Impact, Significance, and Contributions.

Impact: The "Ghar ka Khana" project aims to bridge the gap between students/hostelers and local mothers, allowing access to healthy homemade meals. This initiative supports students in maintaining a balanced diet away from home while enabling mothers to generate income from their culinary skills.

Significance: By offering a platform for authentic homemade food, "Ghar ka Khana" promotes healthier eating habits among students and provides a marketplace for mothers to showcase and monetize their cooking talents.

Contributions: This project contributes to the community by fostering connections between students and local families, encouraging a sense of home and support even in distant locations. It also empowers women by providing them with a platform to earn independently.

1.3 Definition, Acronyms and Abbreviations

1.3.1 Definitions

- **User:-** Refers to individuals (students or hostellers) using the "Ghar ka Khana" platform to order homemade.
- **Admin:-** Denotes administrators responsible for managing the platform, overseeing orders, and ensuring smooth operations.
- **Delivery:-** Refers to individuals responsible for picking up meals from the cooks and delivering them to the users.

1.3.2 Acronyms and Abbreviations

- SRS:-Software Requirement Specification
- UI:-User Interface
- API:-Application Programming interface
- HTML:-Hypertext Markup Language
- CSS:-Cascading Style Sheets
- JS:-JavaScript
- Mongo DB - Non-Relational Database
- HTTPS:- Hypertext Transfer Protocol Secure

1.4 Technology Stack

- **React:** React is widely known for its component-based architecture, making it easy to develop reusable UI elements. It uses a virtual DOM for efficient rendering and offers features like JSX for writing HTML in JavaScript and state management through tools like Redux or Context API.
- **Node.js:** Node.js allows developers to use JavaScript for server-side scripting, enabling the creation of lightweight and scalable web applications. It provides a non-blocking, event-driven architecture, making it ideal for handling concurrent requests and building real-time applications.
- **Mongo DB:** MongoDB is a popular NoSQL database designed for handling large volumes of unstructured or semi-structured data
- **Tailwind CSS:** Tailwind CSS is a utility-first CSS framework that streamlines the process of styling web applications. It provides a set of pre-designed utility classes that can be directly applied to HTML elements, enabling rapid prototyping and flexible customization without writing custom CSS code.
- **Jira:** Jira is a versatile project management tool used for planning, tracking, and managing software development projects. It offers features like issue tracking, agile project management, customizable workflows, and integration with various development tools, enhancing team collaboration and productivity.
- **OpenStreet API :** OpenStreet API is a collection of services provided by OpenStreetMap (OSM), a collaborative project that creates a free, editable map of the world. The API allows developers to integrate map data, perform geocoding (converting addresses into coordinates), reverse geocoding, and display location-based information in web or mobile applications. It's commonly used for building custom map interfaces and navigation solutions.
- **ByteCrypt :** ByteCrypt is typically a tool or library focused on encrypting and decrypting data at a byte level. It is used to securely protect sensitive information like passwords, authentication tokens, or personal data in applications, ensuring confidentiality and integrity by encoding data in such a way that only authorized parties can decode it. The exact definition may vary based on context, but the core concept revolves around data encryption.
- **JWT :** JWT is an open standard (RFC 7519) for securely transmitting information between parties as a JSON object. It's widely used in authentication systems to verify the identity of users. The token is signed, ensuring the data's integrity and preventing tampering. JWTs are commonly used for implementing stateless authentication in APIs, allowing users to log in and stay authenticated without storing session data on the server.

- **Express** : Express is a fast, unopinionated, and minimalist web framework for Node.js. It provides a robust set of features for building web and mobile applications, such as routing, middleware support, and HTTP utility methods. Express simplifies the development of APIs and web servers by providing easy-to-use methods for handling requests, responses, and server configurations.
- **Axios** : **Axios** is a popular JavaScript library used for making **HTTP requests** from the browser or Node.js. It simplifies the process of sending asynchronous requests to REST APIs, handling **GET, POST, PUT, DELETE** requests, and more. Axios also offers features like automatic transformation of JSON data, timeout management, and response/error handling, making it easier to interact with APIs and handle asynchronous data in applications.

1.5 Developer Hardware Configuration

- Processor: Minimum Intel i3
- RAM: 4GB
- Graphics: No dedicated graphics required
- Storage: No SSD required
- Operating System: Windows 7,8,10

2 LITERATURE REVIEW

2.1 Overview of Food Delivery Platforms

- Food delivery platforms have transformed the way people access meals, catering to diverse consumer needs and preferences. Key players in this space offer a variety of services, from restaurant takeout to homemade meal options.
- Uber Eats: A major player in the food delivery industry, Uber Eats partners with restaurants to deliver a wide range of meals. It uses a mobile app to manage orders, deliveries, and payments, providing users with real-time tracking and a broad selection of restaurants.
- DoorDash: Known for its extensive network of local restaurants and grocery stores, DoorDash provides delivery services via a user-friendly app. It emphasizes quick delivery times and offers various subscription plans for users and exclusive deals from partner restaurants.
- Grubhub: Grubhub offers a platform for users to order food from local restaurants with features such as reviews, ratings, and detailed restaurant information. It also provides promotional offers and has a strong presence in urban areas.
- Postmates: Postmates distinguishes itself with a broad range of delivery options, including food, groceries, and other essentials. Its app allows users to order from various types of vendors and track their deliveries in real-time.

2.1 Literature Survey Analysis Table

No	Website / Platform	Services Offered	Key Features & Strengths	Target Audience	Notable Features
1.	Zomato	Restaurant discovery and food delivery	Wide range of restaurant options;	Urban consumers, food enthusiast	Restaurant reviews; delivery tracking
2.	Swiggy	Food delivery and restaurant discovery	Quick delivery service; wide restaurant selection	Urban consumers, busy professionals	Express delivery; live order tracking
3.	Big Basket	Grocery and food delivery	Comprehensive grocery selection; home delivery	Families, working professionals	Fresh produce delivery wide product range
4.	Dunzo	Courier and food delivery	Multi-service platform; quick deliveries	Urban dwellers, busy individuals	Fast delivery for groceries, food, and more
5.	Food Panda	-Food delivery and restaurant discovery	Variety of restaurant options; discounts and deals	Food lovers, working professionals	Flexibility and extensions
6	FreshMenu	Gourmet meal delivery	Chef-curated meals; daily changing menu	Foodies, busy professionals	Fresh, gourmet meal options; meal plans

3 SOFTWARE REQUIREMENT SPECIFICATION

3.1 Functional Requirement

3.1.1 *Functional Requirement of Module Admin*

- Login:
Input: Enter admin ID and password.
Process: Validate admin credentials.
Output: Display the administrator dashboard.
- View User Data:
Input: Click on the "View Users" button.
Process: Retrieve and display information of registered users.
Output: Display all user details, including order history and activity.
- View Orders:
Input: Click on the "View Orders" button.
Process: Retrieve and display current and past orders placed by users.
Output: Show order details like user, delivery status, and assigned cook.
- Manage Cooks:
Input: Click on the "Manage Cooks" button.
Process: View, add, or remove cooks from the platform.
Output: Display cook information and their activity status.

3.1.2 *Functional Requirement of Module User*

- Registration:
Input: User provides details and sets user ID and password.
Process: Information is stored in the database.
Output: User profile is created, and the user is redirected to the login page.
- Login:
Input: Enter user ID and password.
Process: Validate the credentials.
Output: User profile page is displayed.
- Forgot Password:
Input: Click on "Forgot Password" and enter a new password.
Process: OTP is sent to the registered email. Validate the OTP.
Output: New password is set in the database, and the user is redirected to the login page.
- View Meals:
Input: Click on the meals section in the navigation bar.
Process: Fetch meals prepared by different cooks.
Output: Display a list of available meals.

- **Place Order:**
Input: Click on the "Order" button and specify meal preferences.
Process: The order is created and stored in the database.
Output: The order is added to the user's profile for tracking.
- **Track Order:**
Input: Navigate to the "Track Order" section.
Process: Fetch order status from the database.
Output: Display real-time order status.
- **View Order History:**
Input: Click on the "Order History" section.
Process: Fetch previous orders from the database.
Output: Display a list of past orders with details.

3.2 Non-Functional Requirement

- **Performance:**
Ensure fast and responsive browsing and order placement.
- **Security:**
Protect users' personal information, payment details, and order history from unauthorized access.
- **Usability:**
Provide an intuitive, user-friendly interface for both students/hostelers and mothers.
- **Reliability:**
Maintain consistent platform functionality without errors or crashes.
- **Scalability:**
Handle increased traffic and orders as the user base grows without performance issues.
- **Compatibility:**
Ensure the platform works seamlessly across different devices (desktop, mobile, tablet) and browsers.
- **Data Backup:**
Regularly back up user data (order history, profile info) to prevent data loss.
- **Regulatory Compliance:**
Follow food safety regulations and data protection laws to ensure compliance and privacy.

4 DIAGRAMS

4.1 Flow chart (Delivery Boy)

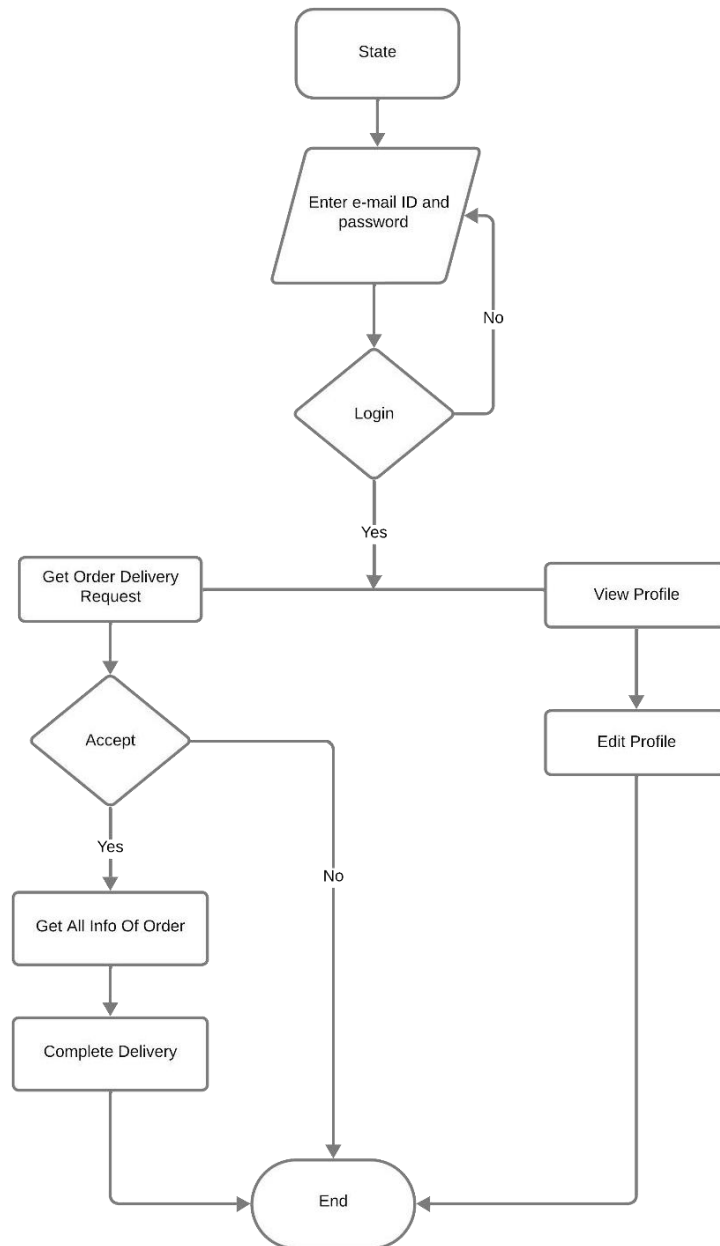
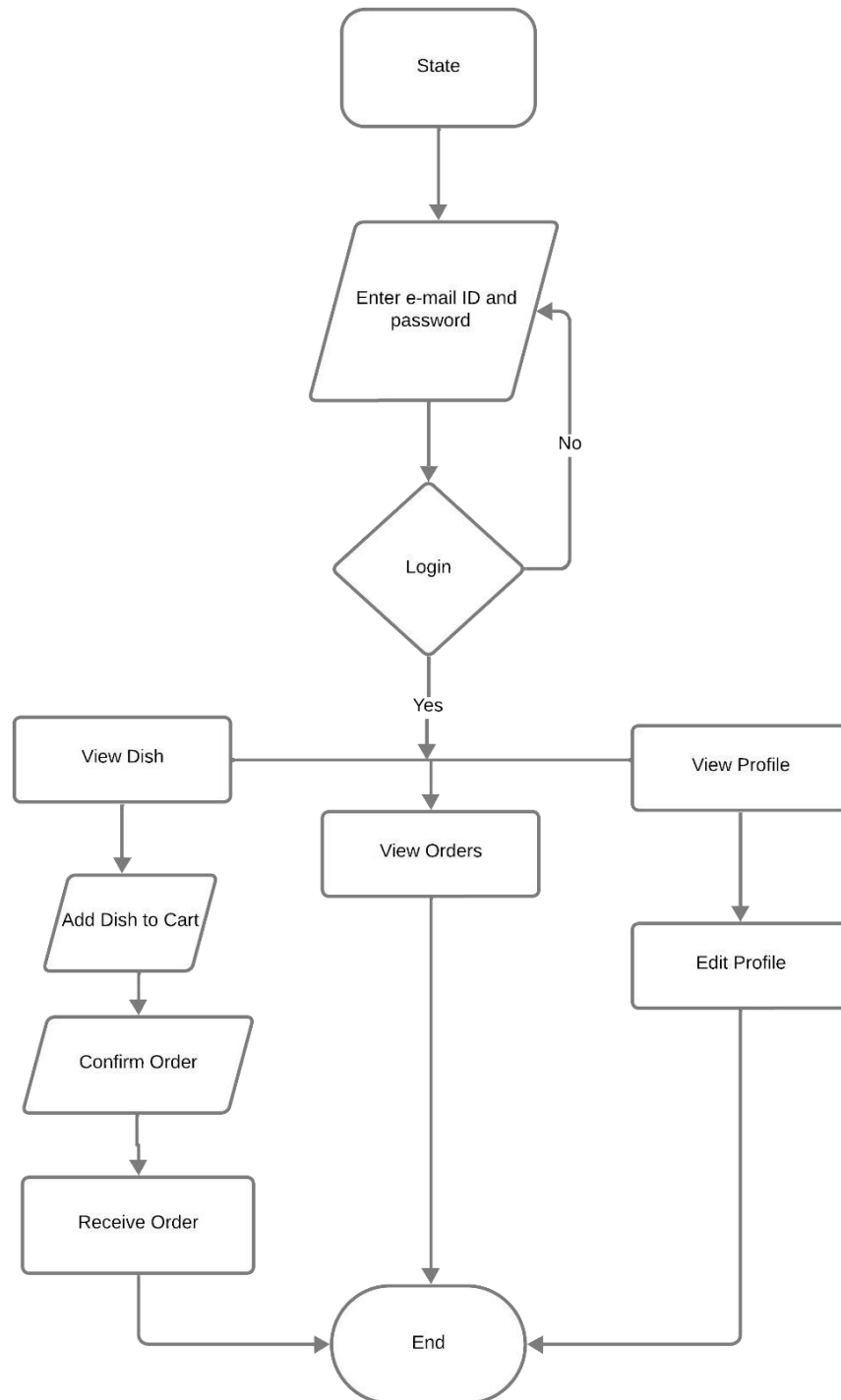


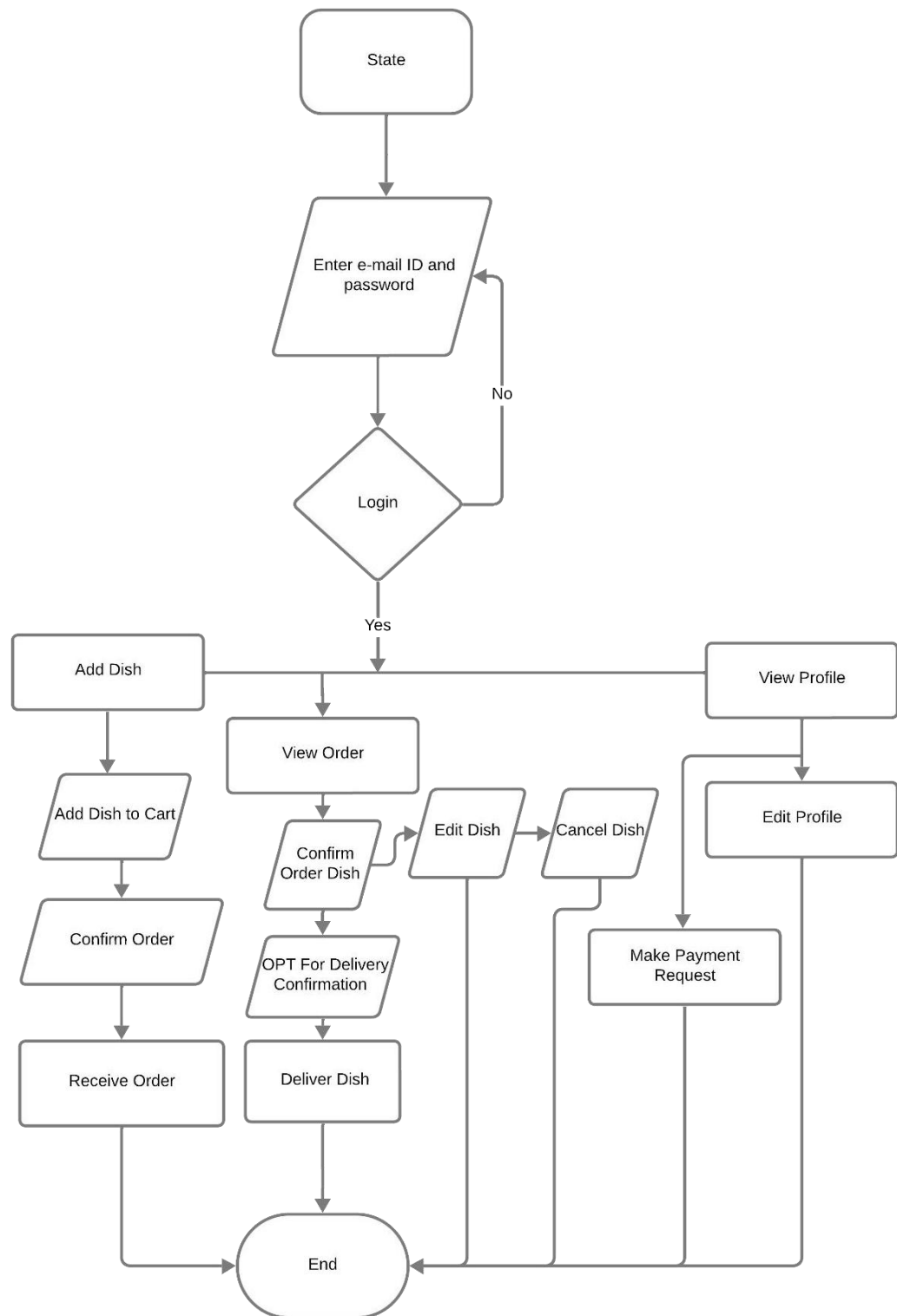
Figure 4.1 Flow Chart (Delivery Boy)

4.2 Flow Chart (Food Consumer)



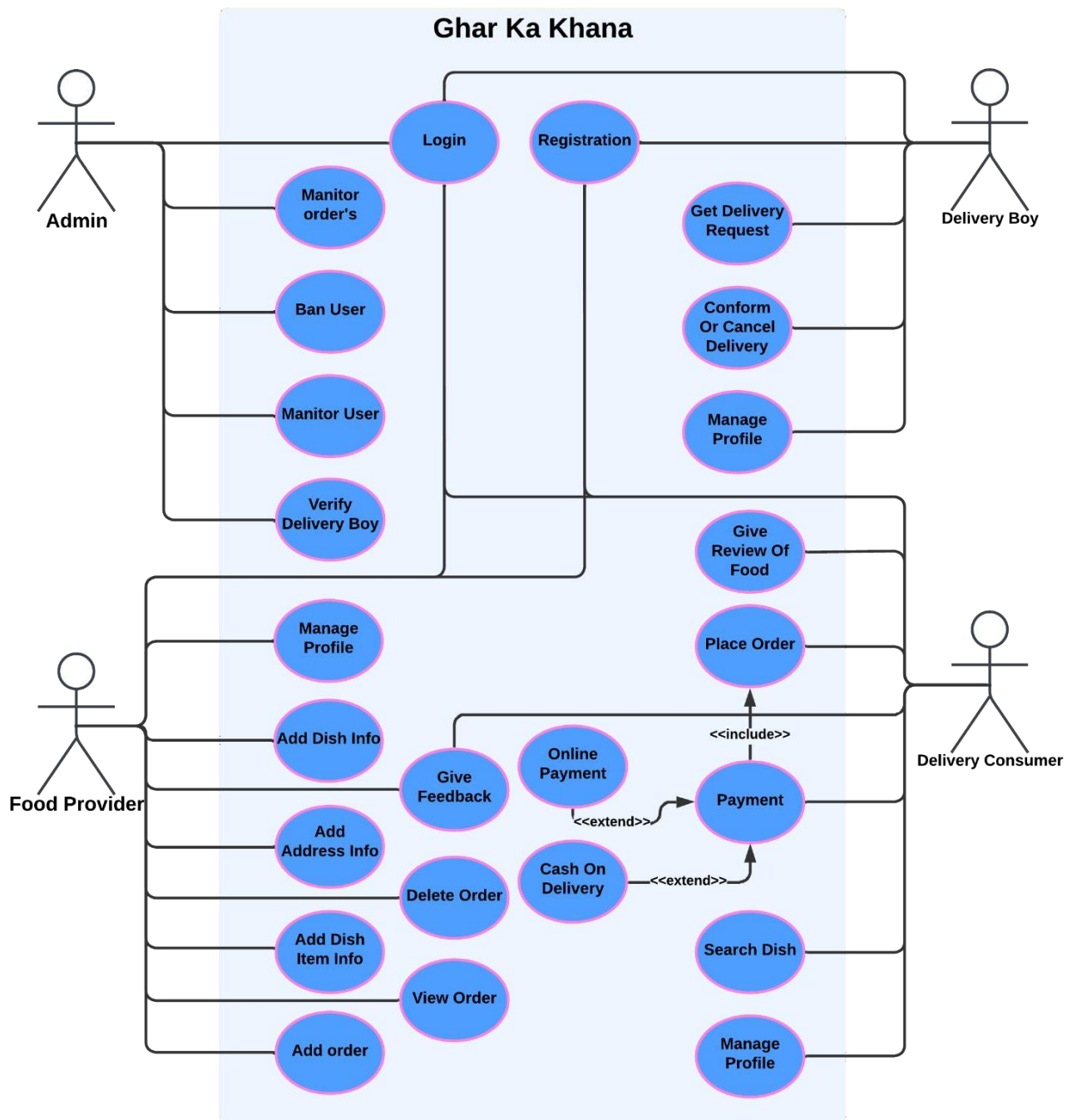
4.2 Flow Chart (Food Consumer)

4.3 Flow Chart (Food Provider)



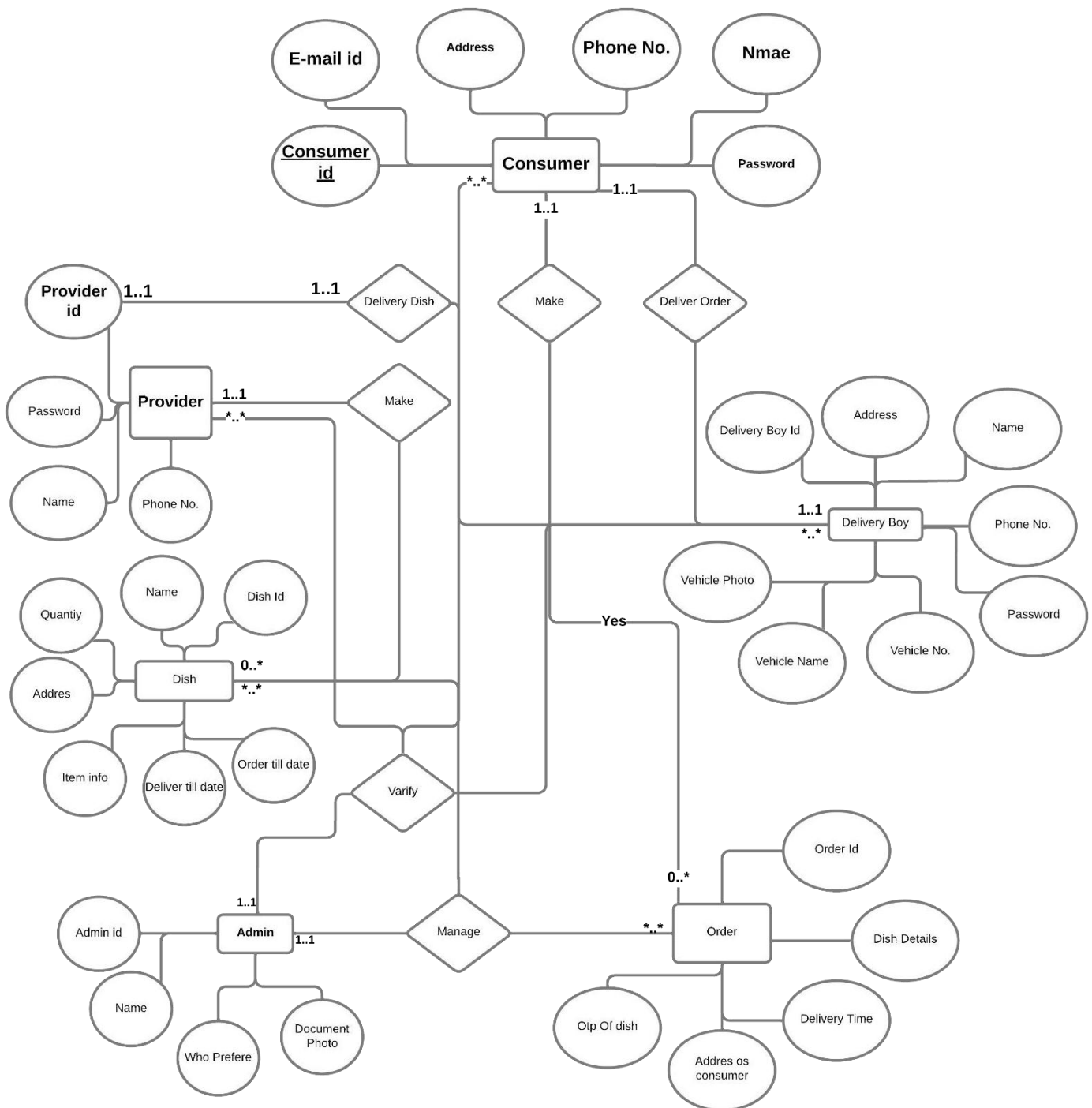
4.3 Flow Chart (Food Provider)

4.4 Use-Case Diagram



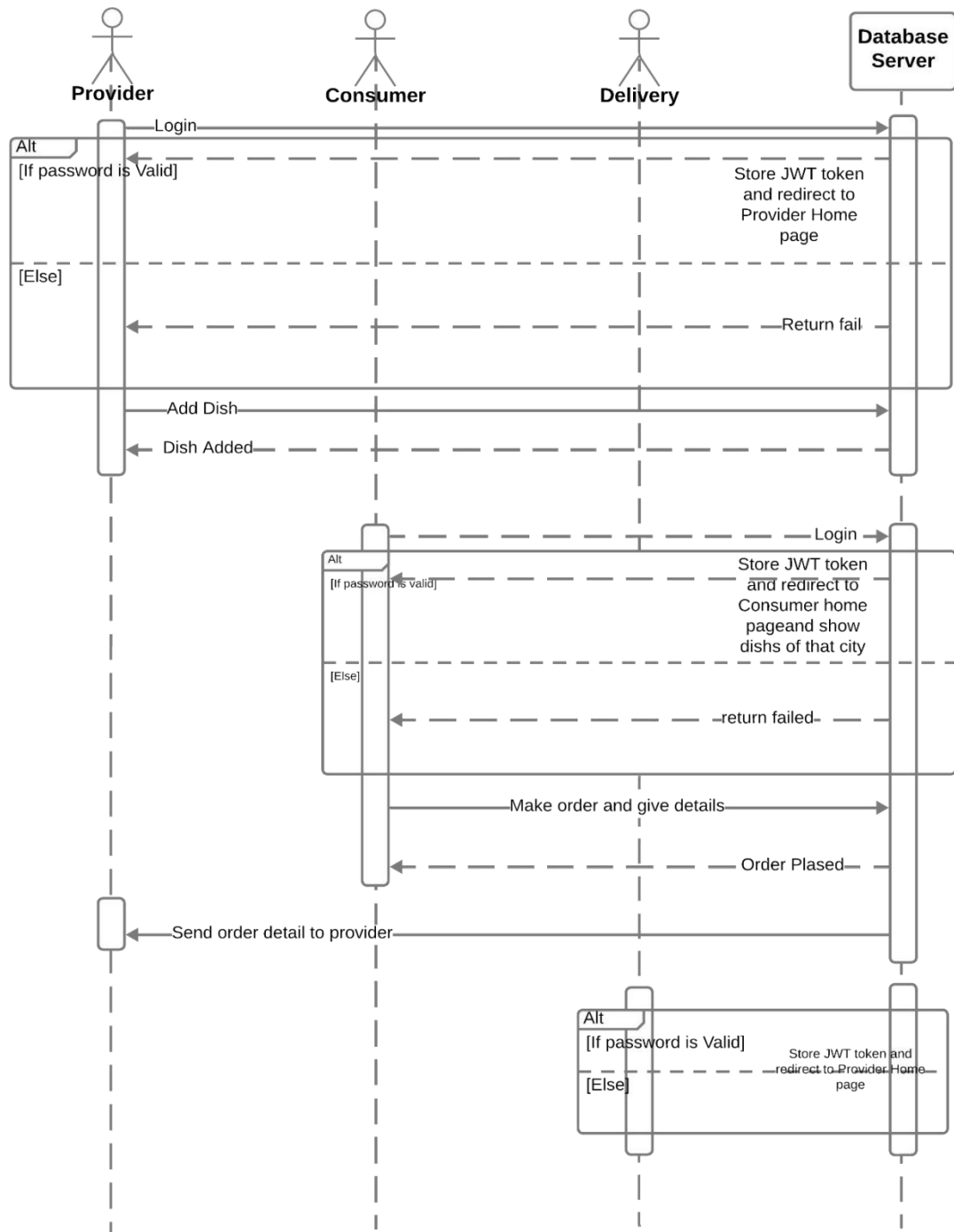
4.4 Use-Case Diagram

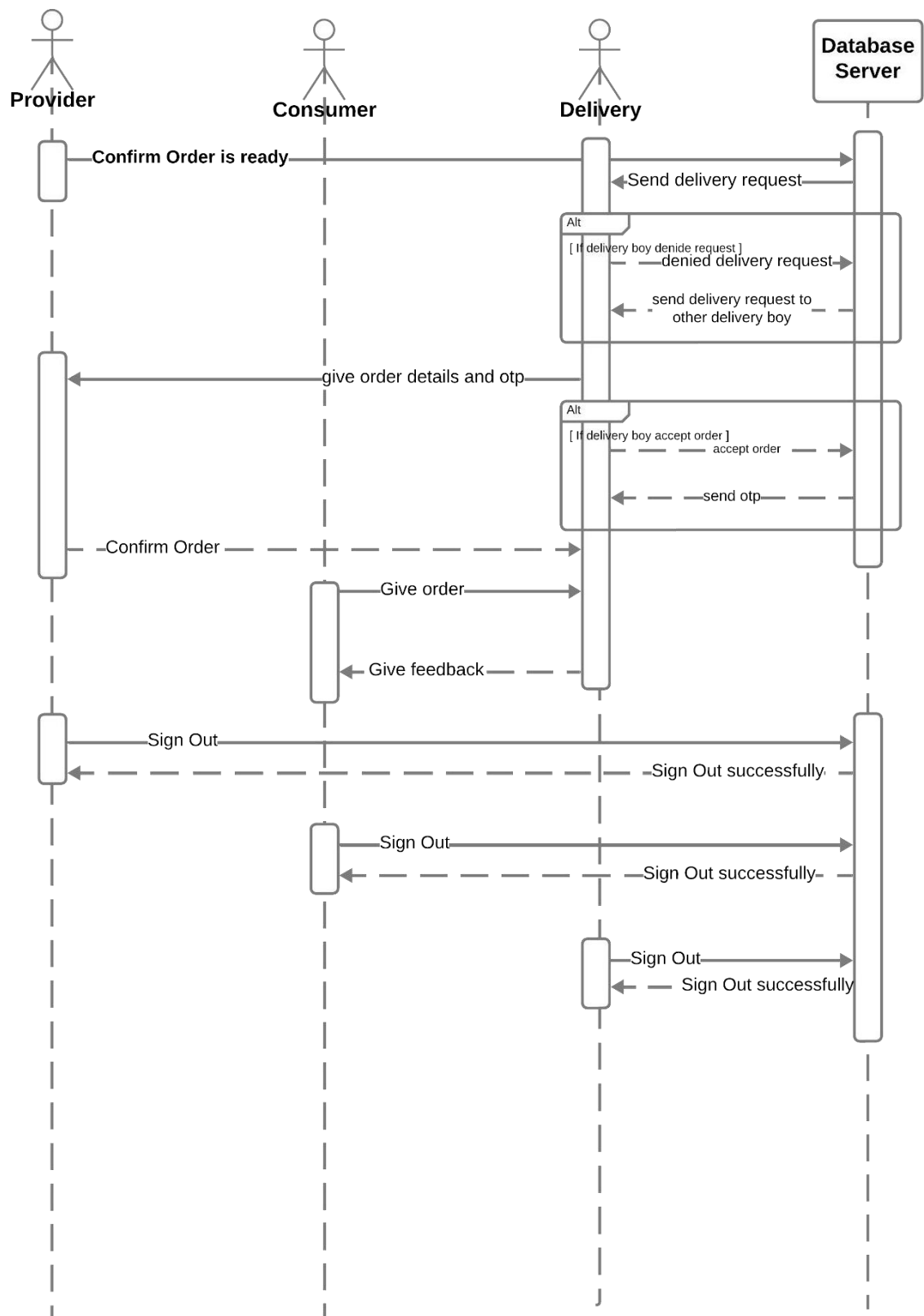
4.5 E-R diagram



4.5 E-R diagram

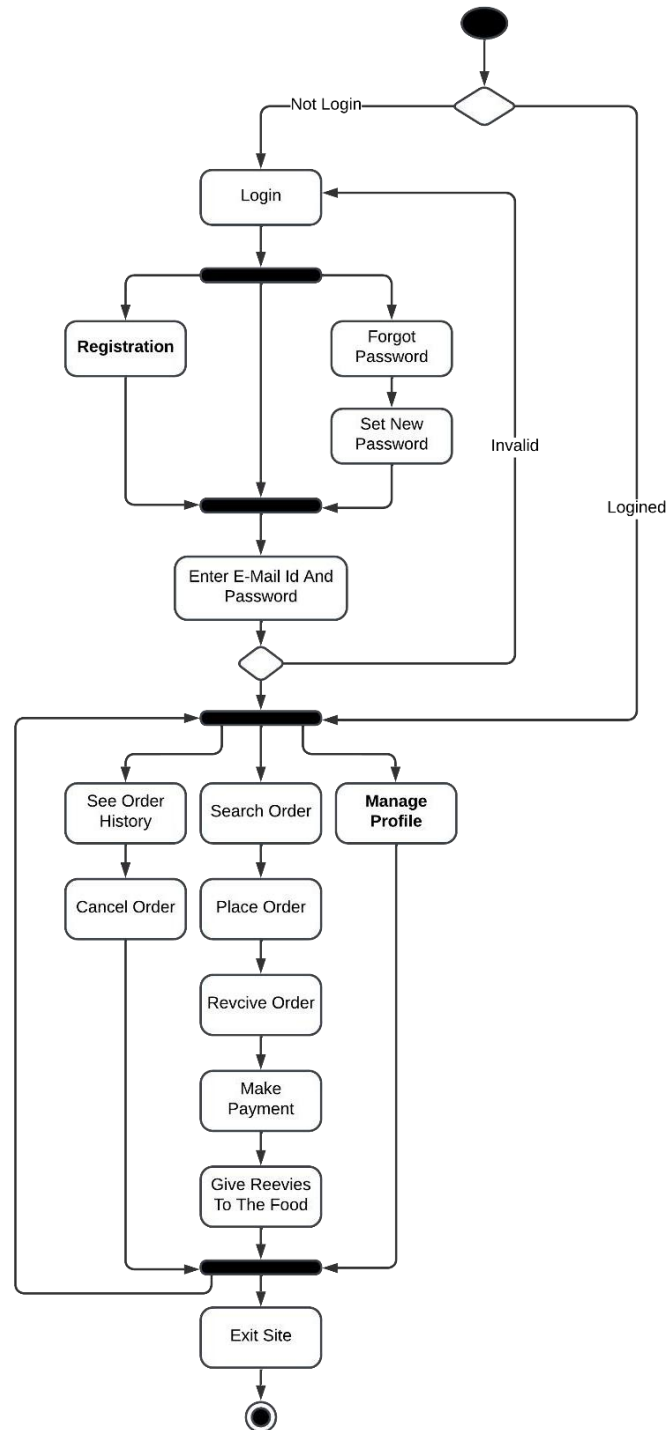
4.6 Sequence Diagram



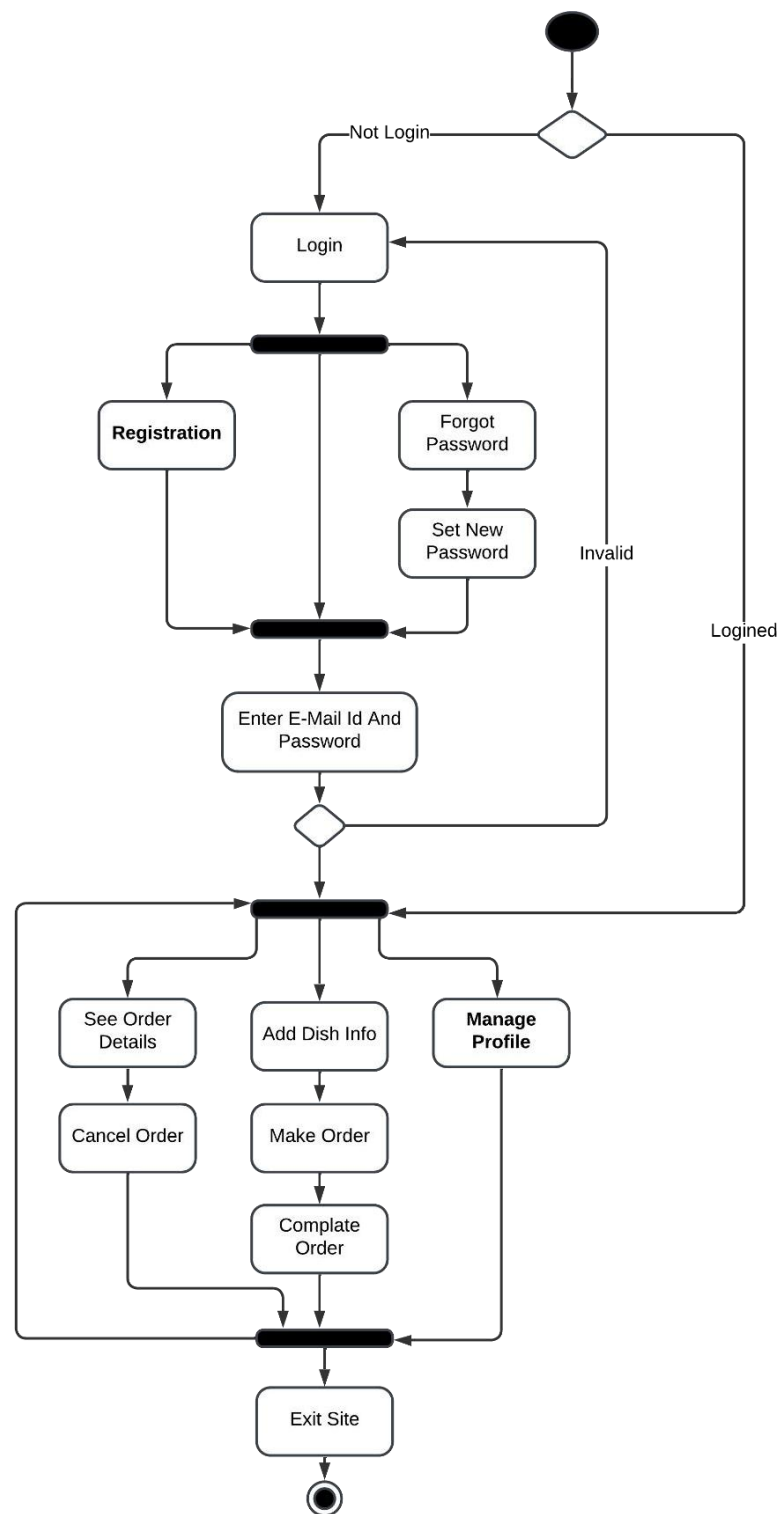


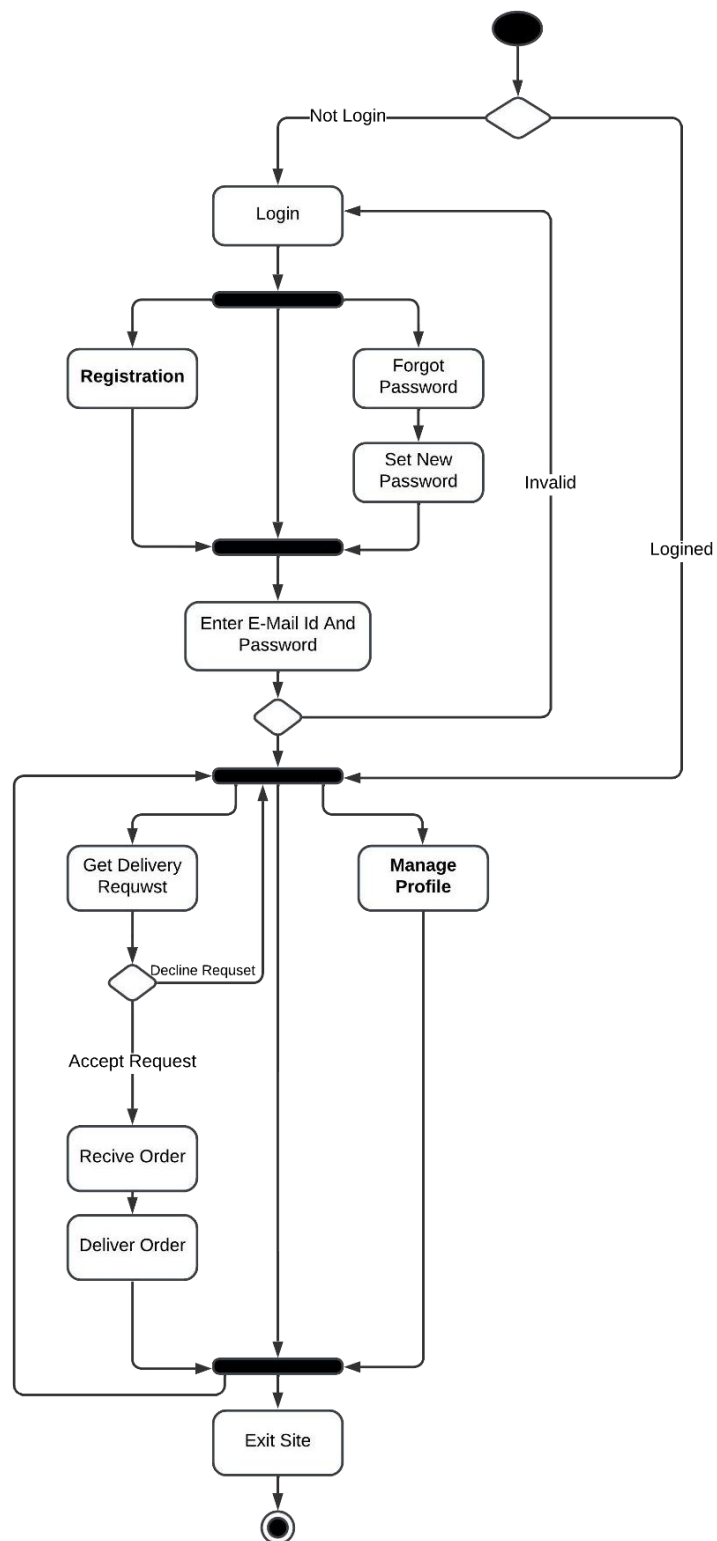
4.6 Sequence Diagram

4.7 Activity Diagram(Food Consumer)

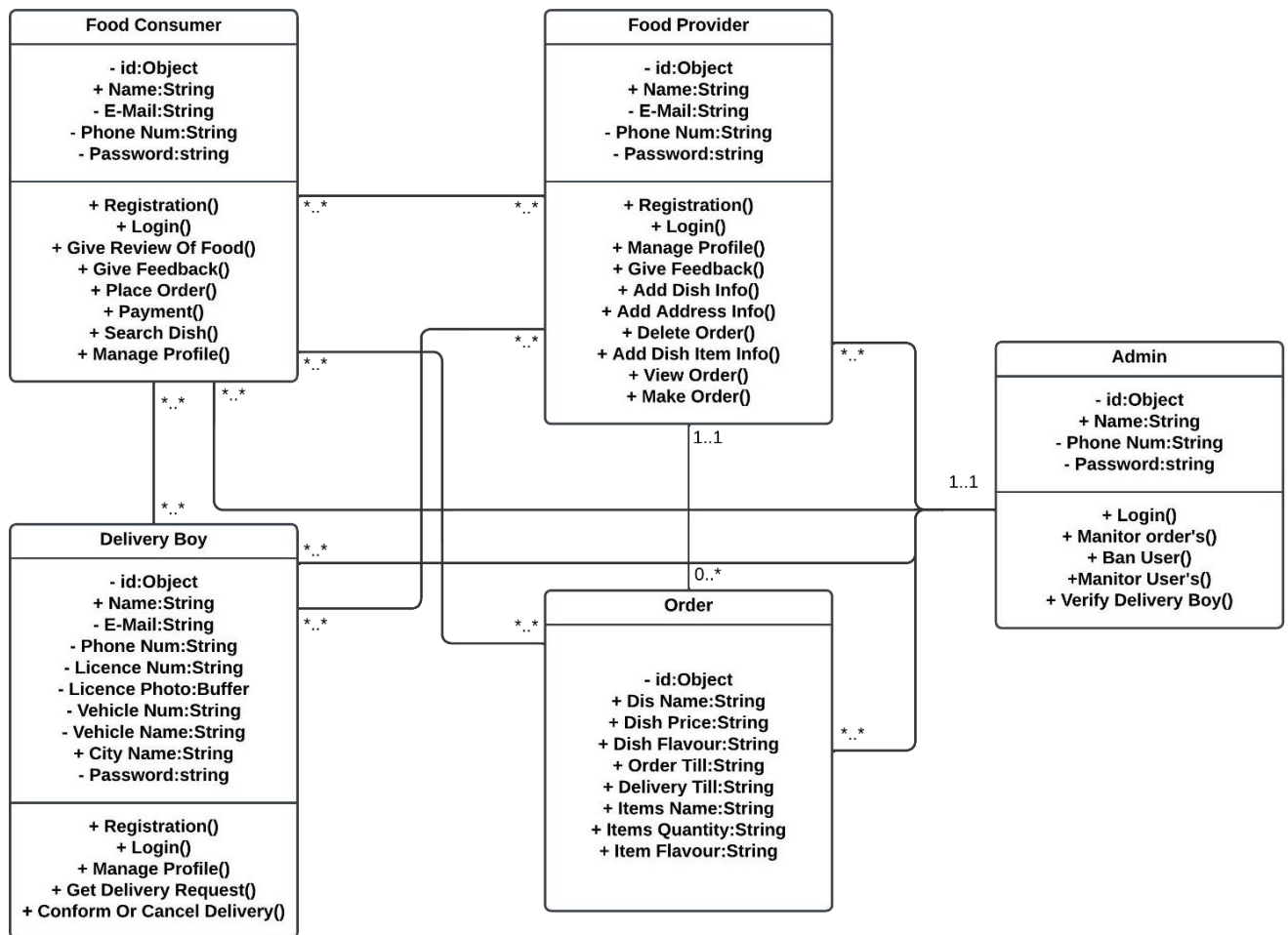


4.7 Activity Diagram(Food Consumer)

4.8 Activity Diagram (Food Provider)**4.8 Activity Diagram (Food Provider)**

4.9 Activity Diagram (Delivery Boy)**4.9 Activity Diagram (Delivery Boy)**

4.10 Class Diagram



4.10 Class Diagram

5 IMPLEMENTATION DETAILS

5.1 data dictionary

5.1 Table: DeliveryBoy

No.	Column Name	Data Type	Description
1	name	character varying	The full name of the delivery boy.
2	email	character varying	The email address of the delivery boy
3	phoneNum	character varying	The 10-digit phone number of the delivery boy
4	licenseNum	character varying	The unique license number of the delivery boy.
5	licensePhoto	bytea	Binary data for the license photo of the delivery boy.
6	vehicleNum	character varying	The unique vehicle number of the delivery boy's vehicle.
7	vehicleName	character varying	The name or type of the delivery boy's vehicle.
8	cityName	character varying	The name of the city where the delivery boy operates.
9	status	boolean	Indicates whether the delivery boy is active (true) or inactive (false).

5.2 Table: DishInfo

No.	Column Name	Data Type	Description
1	providerId	ObjectId	Reference to the FoodProvider collection, indicating the food provider.
2	dishName	character varying	Name of the dish provided by the food provider.
3	address	character varying	Address where the dish is being offered.
4	cityName	character varying	Name of the city where the dish is available.
5	pincode	character varying	Postal code of the dish's location.
6	dishPrice	numeric	Price of the dish in currency units.
7	dishQuantity	numeric	Quantity available for the dish.
8	orderTill	character varying	Last time until orders for the dish can be placed.
9	deliveryTill	character varying	Last time until the dish can be delivered.
10	repeat	array	Days of the week when the dish is available (e.g., ["Monday", "Tuesday"]).

5.3 Table: DishStatus

No.	Column Name	Data Type	Description
1	dishId	ObjectId	Reference to the DishInfo collection, indicating the associated dish.
2	providerId	ObjectId	Reference to the FoodProvider collection, indicating the food provider.
3	availableQuantity	numeric	Quantity of the dish currently available.
4	pendingQuantity	numeric	Quantity of the dish pending in orders.
5	completeQuantity	numeric	Quantity of the dish completed (delivered successfully).
6	cancelQuantity	numeric	Quantity of the dish canceled.
7	readyForDelivery	boolean	Indicates whether the dish is ready for delivery (true or false).

5.4 Table: FoodConsumer

No.	Column Name	Data Type	Description
1	name	character varying	Name of the food consumer.
2	email	character varying	Unique email of the food consumer
3	phoneNumber	character varying	Unique 10-digit phone number of the food consumer.
4	address	array of character varying	List of addresses associated with the food consumer.
5	password	character varying	Password for the food consumer account(length : 6 char)

5.5 Table: FoodProvider

No.	Column Name	Data Type	Description
1	name	character varying	Name of the food provider.
2	email	character varying	Unique email of the food provider
3	phoneNumber	character varying	Unique 10-digit phone number of the food provider.
4	password	character varying	Password for the food provider account (minimum length: 6 char)
5	negativeScore	numeric	Score tracking negative performance or feedback (default: 0).

5.6 Table: ItemDetails

No.	Column Name	Data Type	Description
1	dishId	ObjectId (Reference)	Reference to the associated dish from the DishInfo collection.
2	itemName	character varying	Name of the item; trimmed to remove leading and trailing spaces.
3	itemQuantity	character varying	Quantity of the item (e.g., "2 plates", "500 grams").

5.7 Table: OrderInfo

No.	Column Name	Data Type	Description
1	consumerId	ObjectId (Reference)	Reference to the FoodConsumer who placed the order.
2	deliveryBoyId	ObjectId (Reference)	Reference to the DeliveryBoy assigned to the order.
3	dishInfo	Map (Key: ObjectId, Value: Number)	A map storing dishId as the key and its ordered quantity as the value.
4	paymentMethod	character varying	Payment method used for the order (e.g., "Cash", "Card", etc.).
5	consumerAddress	character varying	Address provided by the consumer for order delivery.
6	status	character varying (Enum)	Current status of the order: pending, confirmed, proceedToDelivery, delivered, or canceled.
7	dishPrice	numeric	Total price of the dishes in the order.
8	gstPrice	numeric	GST (Goods and Services Tax) applied to the order.
9	deliveryPrice	numeric	Delivery charges for the order.
10	totalPrice	numeric	Total cost of the order, including dish price, GST, and delivery charges.
11	expectedDeliveryDate	timestamp	Expected delivery date and time for the order.
12	pickedupDeliveryDate	timestamp	Date and time when the order was picked up for delivery.
13	completeDeliveryDate	timestamp	Date and time when the order was delivered successfully.
14	otp	Map (Key: ObjectId, Value: Number)	A map storing dish-related OTPs for verification during delivery.
15	dishDelivery	Map (Key: ObjectId, Value: Boolean)	A map indicating whether each dish was delivered (true) or not (false).
16	cancelDishes	Array of ObjectId (References)	List of dishIds that were canceled in the order.

5.8 Program Code

Admin Form:

```
import React, { useState } from 'react';
import { motion } from 'framer-motion';
import axios from 'axios'

const AddAdminForm = () => {
  const [formData, setFormData] =
    useState({name: "",
    phoneNumber: "",
    email: "",
    aadhaarPhoto: null,
    aadhaarNumber: "",
    password: "",
    confirmPassword: "",
    city: "",
    });

  const [errors, setErrors] = useState({});

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData((prevState) => ({
      ...prevState,
      [name]: value,
    }));
  };

  const handleFileChange = (e) =>
    {setFormData((prevState) => ({
      ...prevState,
      aadhaarPhoto: e.target.files[0],
    }));
  };

  const validateForm = () =>
    {const validationErrors =
    {}};

  // Name validation
  if (!formData.name) validationErrors.name = 'Name is required';

  // Phone number validation (Assuming Indian phone number format)
  const phoneRegex = /^[6-9]\d{9}$/;
  if (!formData.phoneNumber)
    { validationErrors.phoneNumber = 'Phone number is
    required';
  } else if (!phoneRegex.test(formData.phoneNumber))
    { validationErrors.phoneNumber = 'Enter a valid phone
    number';
```



```
const emailRegex = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;if
(!formData.email) {
  validationErrors.email = 'Email is required';
} else if (!emailRegex.test(formData.email))
{ validationErrors.email = 'Enter a valid email
address';
}

// Aadhaar card number validation (12 digits)
const aadhaarRegex = /^\d{12}$/;
if (!formData.aadhaarNumber) {
  validationErrors.aadhaarNumber = 'Aadhaar number is required';
} else if (!aadhaarRegex.test(formData.aadhaarNumber))
{ validationErrors.aadhaarNumber = 'Enter a valid 12-digit Aadhaar
number';
}

// Aadhaar photo validation
if (!formData.aadhaarPhoto) {
  validationErrors.aadhaarPhoto = 'Aadhaar photo is required';
}

// Password validation
if (!formData.password)
{ validationErrors.password = 'Password is
required';
} else if (formData.password.length < 6) {
  validationErrors.password = 'Password must be at least 6 characters';
}

// Confirm password validation
if (formData.password !== formData.confirmPassword)
{ validationErrors.confirmPassword = 'Passwords do not
match';
}

// City validation
if (!formData.city)
{ validationErrors.city = 'City is
required';
}

setErrors(validationErrors);
return Object.keys(validationErrors).length === 0;
};

const handleSubmit = async (e) =>
{e.preventDefault();

if (validateForm())
{try {
  console.log('Form submitted successfully with:', formData);
```

```
console.log('Admin added successfully:', res.data);
alert('Admin added successfully!');
```

```
// Optionally, reset the form or navigate elsewhere
setFormData({
  name: "",
  phoneNumber: "",
  email: "",
  aadhaarPhoto: null,
  aadhaarNumber: "",
  password: "",
  confirmPassword: "",
  city: "",
});
}
} catch (error) {
  // Handle errors
  console.error('Error adding admin:', error.response?.data || error.message);
  alert(
    `Failed to add admin:
    ${ error.response?.data?.message ||
    error.message
    }`
  );
}
} else {
  // Validation error feedback
  alert('Please correct the errors in the form.');
```

```
}
};

return (
```

```
<motion.div
  className="max-w-lg mx-auto p-6 bg-white rounded-lg shadow-lg mt-10"
  initial={{ opacity: 0 }}
  animate={{ opacity: 1 }}
  transition={{ duration: 0.5 }}
>
  <h2 className="text-3xl font-semibold text-center text-gray-700 mb-6">Add New Admin</h2>
  <form onSubmit={handleSubmit} className="space-y-4">
    { /* Name */ }
    <div>
      <label htmlFor="name" className="block text-sm font-medium text-gray-600">Name</label>
      <input
        type="text"
        name="name"
        value={formData.name}
        onChange={handleChange}
        className="w-full mt-2 p-3 border border-gray-300 rounded-md focus:ring-2 focus:ring-blue-500"
      />
      {errors.name && <p className="text-red-500 text-sm">{errors.name}</p>}
    </div>
```

```

<div>
  <label htmlFor="phoneNumber" className="block text-sm font-medium text-gray-600">Phone
Number</label>
  <input
    type="tel"
    name="phoneNumber"
    value={formData.phoneNumber}
    onChange={handleChange}
    className="w-full mt-2 p-3 border border-gray-300 rounded-md focus:ring-2 focus:ring-blue-500"
  />
  {errors.phoneNumber && <p className="text-red-500 text-sm">{errors.phoneNumber}</p>}
</div>

{/* Email */}
<div>
  <label htmlFor="email" className="block text-sm font-medium text-gray-600">Email</label>
  <input
    type="email"
    name="email"
    value={formData.email}
    onChange={handleChange}
    className="w-full mt-2 p-3 border border-gray-300 rounded-md focus:ring-2 focus:ring-blue-500"
  />
  {errors.email && <p className="text-red-500 text-sm">{errors.email}</p>}
</div>

{/* Aadhaar Photo */}
<div>
  <label htmlFor="aadhaarPhoto" className="block text-sm font-medium text-gray-600">Aadhaar Card
Photo</label>
  <input
    type="file"
    accept="image/*"
    onChange={handleFileChange}
    className="w-full mt-2 p-3 border border-gray-300 rounded-md focus:ring-2 focus:ring-blue-500"
  />
  {errors.aadhaarPhoto && <p className="text-red-500 text-sm">{errors.aadhaarPhoto}</p>}
</div>

{/* Aadhaar Number */}
<div>
  <label htmlFor="aadhaarNumber" className="block text-sm font-medium text-gray-600">Aadhaar
Number</label>
  <input
    type="text"
    name="aadhaarNumber"
    value={formData.aadhaarNumber}
    onChange={handleChange}
    className="w-full mt-2 p-3 border border-gray-300 rounded-md focus:ring-2 focus:ring-blue-500"
  />
  {errors.aadhaarNumber && <p className="text-red-500 text-sm">{errors.aadhaarNumber}</p>}
</div>

```

```

    { /* Password */ }
    <div>
      <label htmlFor="password" className="block text-sm font-medium text-gray-600">Password</label>
      <input
        type="password"
        name="password"
        value={formData.password}
        onChange={handleChange}
        className="w-full mt-2 p-3 border border-gray-300 rounded-md focus:ring-2 focus:ring-blue-500"
      />
      {errors.password && <p className="text-red-500 text-sm">{errors.password}</p>}}
    </div>

    { /* Confirm Password */ }
    <div>
      <label htmlFor="confirmPassword" className="block text-sm font-medium text-gray-600">Confirm
      Password</label>
      <input
        type="password"
        name="confirmPassword"
        value={formData.confirmPassword}
        onChange={handleChange}
        className="w-full mt-2 p-3 border border-gray-300 rounded-md focus:ring-2 focus:ring-blue-500"
      />
      {errors.confirmPassword && <p className="text-red-500 text-sm">{errors.confirmPassword}</p>}}
    </div>

    { /* City */ }
    <div>
      <label htmlFor="city" className="block text-sm font-medium text-gray-600">City</label>
      <input
        type="text"
        name="city"
        value={formData.city}
        onChange={handleChange}
        className="w-full mt-2 p-3 border border-gray-300 rounded-md focus:ring-2 focus:ring-blue-500"
      />
      {errors.city && <p className="text-red-500 text-sm">{errors.city}</p>}}
    </div>

    { /* Warning Message */ }
    <div className="text-red-600 text-sm mt-2">
      <p className="italic">* You are responsible for this admin. You take full guarantee of this admin.</p>
    </div>

    { /* Submit Button */ }
    <motion.button
      type="submit"
      className="w-full py-3 mt-4 bg-blue-500 text-white rounded-md hover:bg-blue-600 focus:outline-none
      focus:ring-2 focus:ring-blue-400"
      whileHover={{ scale: 1.05 }}
      whileTap={{ scale: 0.95 }}
    >

```

```

    Add Admin
  </motion.button>
</form>
</motion.div>
);
};

```

```
export default AddAdminForm;
```

5.2.2 Dashboard

Code :

```

import React, { useState } from 'react';
import AdminProvider from './AdminProvider';
import AdminConsumer from './AdminConsumer';
import AdminDeliveryBoy from './AdminDeliveryBoy';
import AdminVerifyUser from './AdminVerifyUser';
import AddAdminForm from './AddAdminForm';
import { motion, AnimatePresence } from 'framer-motion';

const AdminDashboard = () => {
  const [divUser, setDivUser] = useState('provider');

  // Function to handle setting the active user type
  const handleSetDivUser = (userType) => {
    setDivUser(userType);
  };

  return (
    <div className="flex min-h-screen bg-gray-100">
      {/* Sidebar */}
      <div className="w-64 bg-green-800 text-white p-4">
        <h2 className="text-2xl font-bold mb-6">Admin Dashboard</h2>
        <ul>
          <li
            className={`mb-4 cursor-pointer ${divUser === 'provider' ? 'font-bold underline' : ''}`}
            onClick={() => handleSetDivUser('provider')}
          >
            <a className="hover:underline">Food Providers</a>
          </li>
          <li
            className={`mb-4 cursor-pointer ${divUser === 'consumer' ? 'font-bold underline' : ''}`}
            onClick={() => handleSetDivUser('consumer')}
          >
            <a className="hover:underline">Food Consumer</a>
          </li>
          <li
            className={`mb-4 cursor-pointer ${divUser === 'deliver' ? 'font-bold underline' : ''}`}
            onClick={() => handleSetDivUser('deliver')}
          >

```

```

    <a className="hover:underline">Food Deliver</a>
  </li>
  <li
    className={`mb-4 cursor-pointer ${divUser === 'verify' ? 'font-bold underline' : ''} onClick={()
    => handleSetDivUser('verify')}`
  >
    <a className="hover:underline">Varify User</a>
  </li>
  <li
    className={`mb-4 cursor-pointer ${divUser === 'addAdmin' ? 'font-bold underline' : ''}
    onClick={() => handleSetDivUser('addAdmin')}`
  >
    <a className="hover:underline">Add new Admin</a>
  </li>
</ul>
</div>

```

```

  { /* Main Content */
  { /* <AdminConsumer /> */
  <div className="flex-1 p-8">
    {divUser === 'provider' ? (
      <AdminProvider />
    ) : divUser === 'consumer' ? (
      <AdminConsumer />
    ) : divUser === 'deliver' ? (
      <AdminDeliveryBoy />
    ) : divUser === 'verify' ? (
      <AdminVerifyUser />
    ) : divUser === 'addAdmin' ? (
      <AddAdminForm />
    ) : (
      <AdminConsumer />
    )}
  </div>
</div>
);
};
export default AdminDashboard;

```

6 TESTING

6.1 Unit testing

- In our project, we've embraced unit testing as a core practice in software development. By rigorously testing individual components or functions in isolation, we ensure they perform as expected. Automated unit tests run frequently, catching bugs early and enabling swift fixes. This approach not only enhances code quality but also fosters project stability and facilitates smooth refactoring. Overall, unit testing is integral to our software testing lifecycle, helping us deliver reliable software to users.

ID	Test Scenario	Preconditions	Test Steps	Actual Result	Pass/Fail Criteria
TC_001	Sign in to "Ghar Ka Khana" website and log in successfully.	Sign-in page is accessible.	<ul style="list-style-type: none"> - Go to the Sign-in Page. - Fill all the required details. - Click on the "Sign In" button. 	User should be redirected to the Home page after a successful sign-in. If the user is already signed in, they should be logged in directly without needing to sign in again.	Pass.
TC_002	Browse and order dishes on the Home page (Consumer activity).	The provider offers a list of dishes.	<ul style="list-style-type: none"> - Go to the Home page. - View the available dishes. - Select a dish and place an order. - Wait for the delivery boy to deliver the dish. 	Consumer should receive the ordered dish after placing the order.	Pass.
TC_003	Verify if dishes are visible on the Home page.	User is logged in and on the Home page.	<ul style="list-style-type: none"> - Go to the Home page. - Check if dishes are displayed. - Verify the names, images, and prices of the dishes. 	The Home page should display the available dishes with proper names, images, and prices.	Pass.
TC_004	Check if ordered dishes are confirmed by the provider.	User is logged in and has placed an order.	<ul style="list-style-type: none"> - Place an order. - Check the order status in the "Orders" section. - Verify if the status shows "Confirmed" or "Pending". 	Ordered dishes should have a status of "Confirmed" when the provider accepts the order.	Pass.
TC_005	Verify where the delivery will take place or pick-up location.	User is logged in and placing an order.	<ul style="list-style-type: none"> - Select dishes and proceed to checkout. - Verify the delivery or pick-up location displayed at the checkout page. 	The system should clearly display the delivery address or pick-up location.	Pass.

TC_006	Check the estimated delivery time for an order.	User has placed an order.	<ul style="list-style-type: none"> - Place an order. - Verify the estimated delivery time displayed on the order confirmation page or notification. 	The system should display an accurate estimated delivery time after placing the order.	Pass.
TC_007	Check the order history for past orders.	User is logged in and has previous orders.	<ul style="list-style-type: none"> - Open "Ghar Ka Khana". - Navigate to the "Order History" section. - Locate a past order and view its details. 	Order history should display order details such as food items, delivery time, and price.	Pass.
TC_009	Verify request reaches the delivery boy with an OTP.	Provider has confirmed the order.	<ul style="list-style-type: none"> - Provider confirms the order. - Delivery boy receives the order details. - Verify if the OTP is shared with the delivery boy. 	Delivery boy should receive the order details and OTP for delivery verification.	Pass.
TC_010	Check provider's ability to add dishes to the menu.	Provider is logged in.	<ul style="list-style-type: none"> - Provider logs in. - Navigate to the menu section. - Add a new dish. - Confirm if the dish is successfully added. 	Provider should be able to add dishes to the menu successfully.	Pass.
TC_011	Verify provider can mark the dish as "Ready."	Order is in preparation.	<ul style="list-style-type: none"> - Provider logs in. - Select an order. - Mark the dish as "Ready". - Confirm the status update. 	Provider should be able to mark the dish as "Ready," and the delivery boy should be notified.	Pass.
TC_012	Verify delivery boy collects the dish after it's marked "Ready."	Provider has marked the dish as "Ready."	<ul style="list-style-type: none"> - Delivery boy logs in. - Check the status of the assigned order. - Collect the dish from the provider. - Update status to "Collected." 	Delivery boy should be able to collect the dish and update the status to "Collected."	Pass.
TC_013	Verify "Forget Password" functionality.	User has an account but forgot the password.	<ul style="list-style-type: none"> - Go to the login page. - Click on "Forget Password". - Check for the reset password email. - Reset the password. 	User should receive a password reset email, and the password reset process should complete successfully.	Pass.
TC_014	Provider meal repetition option (Mon to Sun).	Provider is logged in and browsing dishes.	<ul style="list-style-type: none"> - Go to the meal ordering section. - Select which days the provider wants to repeat the meal order - Choose the desired days. 	Provider should be able to select which days they want to repeat their meal order (Monday to Sunday).	Pass.

7. PROTOTYPE:

7.1 Sign Up Page:



Sign up to Ghar Ka Khana

Are you a?

☒ Food Consumer ☐ Food Provider ☐ Delivery

Name

joelroy8140@gmail.com

Phone Number

.....


Confirm Password

Sign Up

Already have an account? [Log in](#)

7.1 Sign Up Page:

7.2 Login Page:



Login

joelroy8140@gmail.com

.....


[Forgot password?](#)

SIGN IN

[Don't have an account? Sign up](#)

7.2 Login Page:

7.3 Provider Home Page:

 Ghar Ka Khana

HomeOrdersProfileAbout UsSign Out

Show Us Your Homemade Magic Today!

Craft your lovingly prepared meals and deliver them to those who appreciate your culinary artistry.

Upload Your Tasty Treats

Dish Name

Enter dish name

Dish Flavor

Select a flavor

Dish Price

Enter price


Order Till

--:--:--

Address

Enter address

Activate Windows
Go to Settings to activate Windows.

 Ghar Ka Khana

HomeOrdersProfileAbout UsSign Out

Order Till

--:--:--

Address

Enter address

☐ Any time

Delivery Till

--:--:--

☐ Any time

Your Address

Get Your Address

City Name

Enter city name

Pincode

Enter pincode

Item Name

Item Quantity


Flavor

+ Add Item

Activate Windows
Go to Settings to activate Windows.

7.3 Provider Home Page

7.4 Consumer Home page:

 Ghar Ka Khana

HomeOrdersProfileAbout UsSign Out

Order Till

--:-- --

☐ Any time

Delivery Till

--:-- --

☐ Any time

Address

Enter address

Your Address

Get Your Address

City Name

Enter city name

Pincode

Enter pincode

Item Name

Item Quantity

Flavor

+ Add Item

Activate Windows
Go to Settings to activate Windows.

7.4 Consumer Home page

7.5 Delivery Home Page:

The screenshot shows the 'Admin Dashboard' on the left sidebar with options: Food Providers, Food Consumer, **Food Deliver**, Verify Users, Add new Admin, and Sign Out. The main content area is titled 'Delivery Boys Overview' and includes four filter boxes: 'Filter by City Name', 'Filter by Delivery Boy Email', 'Filter by Order ID', and 'Filter by Status'. Below the filters, there are two profile cards for 'Abhay Hingrajiya'. The first card shows details: ID: 674594c9716e8d2acc243722, Email: testdeliveryboy@gamil.com, Phone: 0074738478, Vehicle: Honda, Vehicle Number: 1234567898, License Number: 123456789, City: Mehsana, and 'No orders available'. The second card shows: ID: 6747ffbeeceb065a729ee020, Email: testconsumer@gmail.com, Phone: 0787938478. A Windows watermark is visible in the bottom right corner.

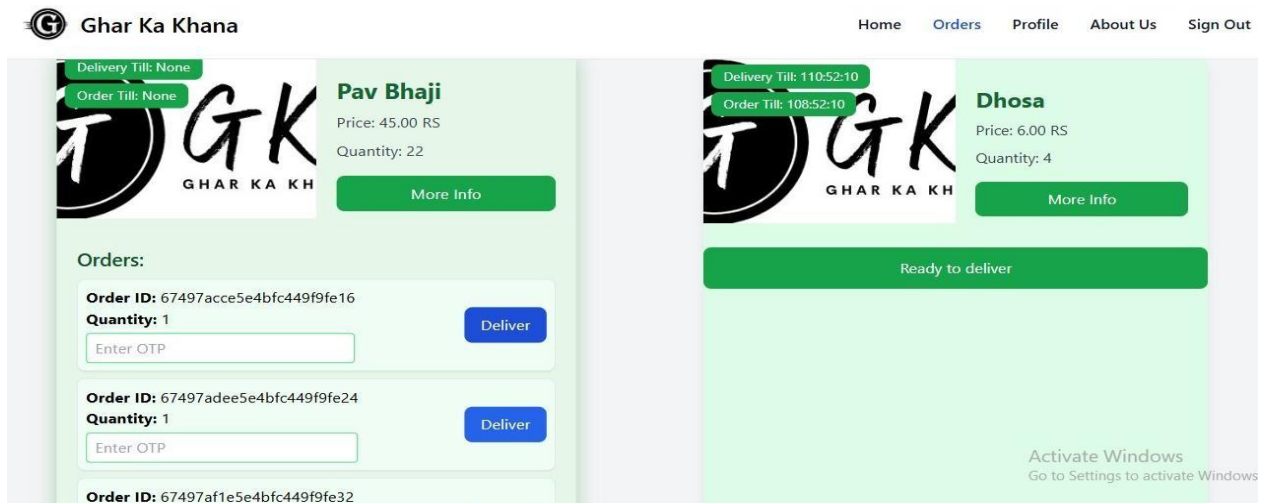
7.5 Delivery Home Page:

7.6 Provider order page

The screenshot shows the 'Ghar Ka Khana' provider order page. The header includes the logo and name 'Ghar Ka Khana' on the left, and navigation links 'Home', 'Orders', 'Profile', 'About Us', and 'Sign Out' on the right. The main content area features four horizontal bars with dropdown arrows: 'Pending Orders' (green), 'Available Orders' (green), 'Complete Orders' (green), and 'Cancel Orders' (red).

7.6 Provider order page

7.7 Provider confirm order page



7.7 Provider confirm order page

7.8 Admin login page

Admin Login

Email

Password

Login

[Forgot your password? Reset it here](#)

7.8 Admin login page

7.9 Food provider Dashboard

Admin Dashboard

- Food Providers
- Food Consumer
- Food Deliver
- Verify Users
- Add new Admin
- Sign Out

Providers Overview

Filter by City Name: Filter by Pincode: Filter by Provider Email: All Statuses ▼

Total Providers: 5Total Dishes: 72

Abhay Hingrajiya

Email: abhyhingrajiya250@gmail.com
Phone: 0734738478
Negative Score: 0

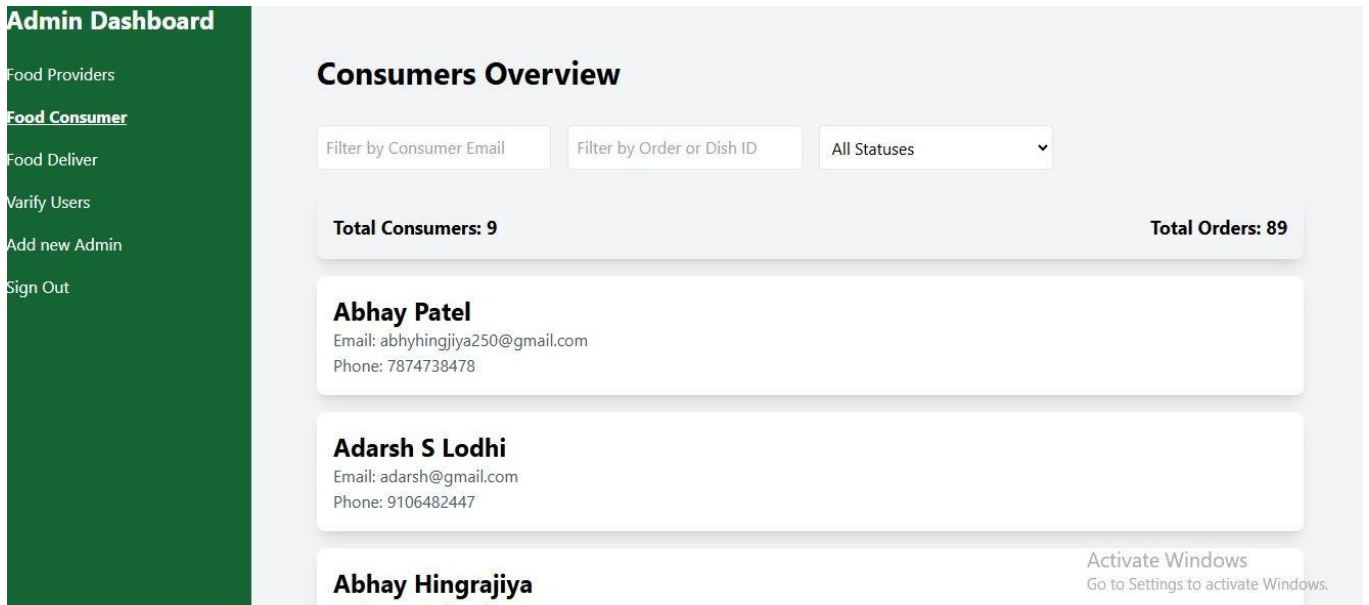
Gujarati Thali

Dish Id: 66e51f078a5e7271da08c5a6
Address: Ganpat University, NH68, Mahesana, Mahesana Taluka, Mahesana, Gujarat, 384001, India
City Name: Mahesana
Pincode: 384001
Dish Price: ₹70
Dish Repeat: Wednesday Friday

Activate Windows
Go to Settings to activate Windows.

7.9 Food provider Dashboard

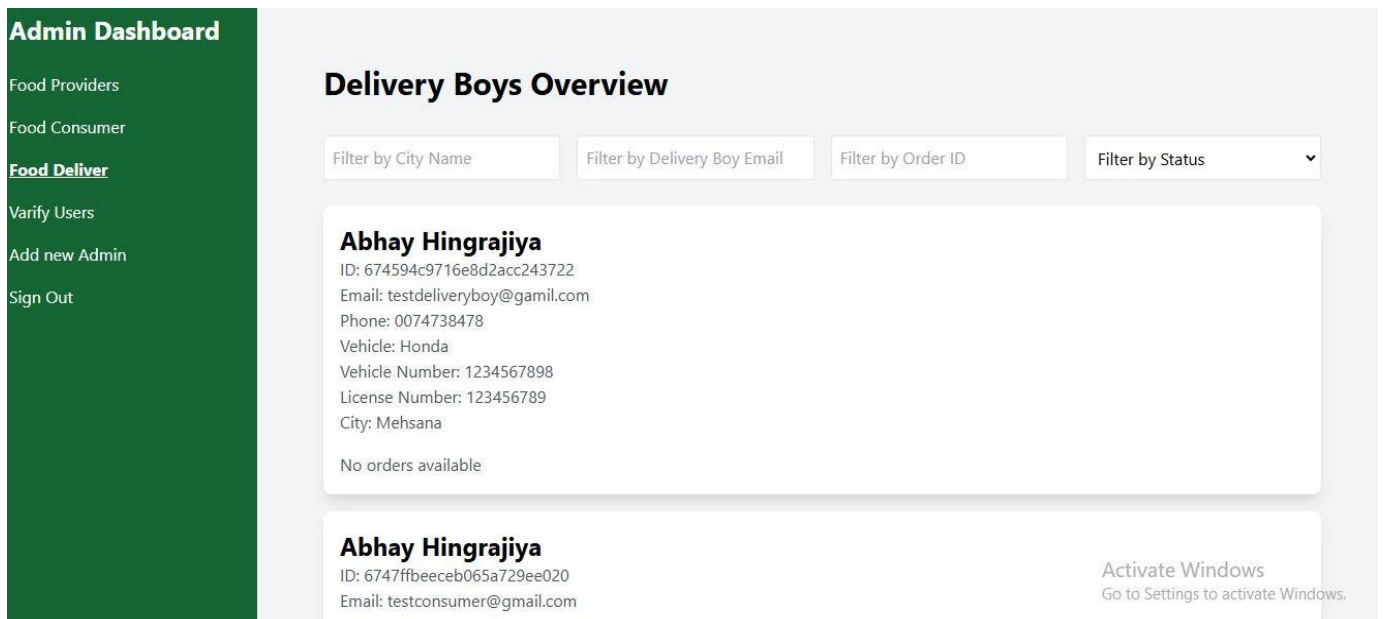
7.10 Food Consumer Dashboard



The screenshot displays the 'Food Consumer Dashboard' interface. On the left is a dark green sidebar with the title 'Admin Dashboard' and a list of menu items: 'Food Providers', 'Food Consumer' (highlighted), 'Food Deliver', 'Verify Users', 'Add new Admin', and 'Sign Out'. The main content area is titled 'Consumers Overview' and features three filter buttons: 'Filter by Consumer Email', 'Filter by Order or Dish ID', and a dropdown menu for 'All Statuses'. Below the filters, two summary boxes show 'Total Consumers: 9' and 'Total Orders: 89'. The dashboard lists three consumer profiles: 'Abhay Patel' (Email: abhyhingjiya250@gmail.com, Phone: 7874738478), 'Adarsh S Lodhi' (Email: adarsh@gmail.com, Phone: 9106482447), and 'Abhay Hingrajiya' (Email: abh...@gmail.com). A Windows activation watermark is visible in the bottom right corner.

7.10 Food Consumer Dashboard

7.11 Food Delivery Dashboard



The screenshot displays the 'Food Delivery Dashboard' interface. The sidebar is identical to the previous dashboard, with 'Food Deliver' highlighted. The main content area is titled 'Delivery Boys Overview' and includes four filter buttons: 'Filter by City Name', 'Filter by Delivery Boy Email', 'Filter by Order ID', and a dropdown menu for 'Filter by Status'. A single delivery boy profile is shown for 'Abhay Hingrajiya' with the following details: ID: 674594c9716e8d2acc243722, Email: testdeliveryboy@gamil.com, Phone: 0074738478, Vehicle: Honda, Vehicle Number: 1234567898, License Number: 123456789, and City: Mehsana. Below the profile, it states 'No orders available'. A Windows activation watermark is present in the bottom right corner.

7.11 Food Delivery Dashboard

7.12 Verify Users

Admin Dashboard


- Food Providers
- Food Consumer
- Food Deliver
- Verify Users**
- Add new Admin
- Sign Out

Pending Verification Requests

Abhay Hingrajiya (Provider)

Email: auyguiya18@gmail.com
Phone: 0787473840

ID: 67507af5c322fedcc1253fcb
Aadhar Card Number: 123456789123
Aadhar Card:



Add a comment (optional)

Verify UserBlock UserSend Comment

Activate Windows
Go to Settings to activate Windows.

7.12 Verify Users

7.13 Add new Admin

Admin Dashboard

- Food Providers
- Food Consumer
- Food Deliver
- Verify Users
- Add new Admin**
- Sign Out

Add New Admin

Name

Phone Number

Email

Aadhaar Card Photo
 No file chosen

Aadhaar Number

Password

Confirm Password

City

Activate Windows
Go to Settings to activate Windows.

7.13 Add new Admin

8 CONCLUSION :

- "Ghar ka Khana" successfully addresses the need for healthy, home-cooked meals among students and hostelers.
- The platform empowers local mothers by providing an opportunity to showcase and monetize their culinary skills.
- It bridges the gap between students/hostelers and local home cooks, offering a convenient and affordable solution for nutritious meals.
- The project includes a user-friendly web interface, ensuring easy browsing, ordering, and payment processing for students and hostelers.
- A separate interface for mothers allows them to manage their dish listings and orders efficiently.
- The platform promotes a sustainable, community-driven ecosystem, benefiting both consumers and food providers.
- "Ghar ka Khana" contributes to the financial independence of local mothers, fostering economic empowerment.
- The initiative aligns with the growing trend of healthy eating and supports a community-based approach to wholesome dining options.

9 FUTURE WORK:

- **Mobile App Integration:** Develop a dedicated mobile application for "Ghar ka Khana," allowing users to browse, order, and track homemade meals seamlessly on their smartphones.
- **Responsive User Interface:** Ensure the platform is fully responsive, adapting smoothly to different screen sizes and devices, improving user experience for mobile and tablet users.
- **Enhanced Order Tracking System:** Implement real-time order tracking for both users and delivery personnel, providing updates on meal preparation, delivery time, and more.
- **Personalized Meal Suggestions:** Introduce a recommendation system that suggests meals to users based on their preferences and previous orders.
- **Expanded Cook Base:** Expand the network of mothers preparing meals, ensuring users have access to a wider variety of homemade food options.
- **Subscription Plans:** Offer meal subscription packages where users can sign up for daily, weekly, or monthly meal plans for a consistent, hassle-free food experience.
- **Nutritional Information:** Provide detailed nutritional information for each meal, allowing users to make informed choices about their food.
- **Enhanced Payment Options:** Integrate additional payment gateways and provide users with more flexibility in making payments, such as digital wallets, UPI, and bank transfers.
- **Map Integration:** Add map integration to show delivery routes, nearby cooks, and estimated arrival times directly on the platform.

10 REFERENCES

- [zomato.com](https://www.zomato.com)
- [swiggy.com](https://www.swiggy.com)
- [bigbasket.com](https://www.bigbasket.com)
- [dunzo.com](https://www.dunzo.com)
- [foodpanda.com](https://www.foodpanda.com)
- [freshmenu.com](https://www.freshmenu.com)
- [masalabox.com](https://www.masalabox.com)
- [box8.in](https://www.box8.in)
- [nosh.co.in](https://www.nosh.co.in)
- [eazydiner.com](https://www.eazydiner.com)

U. V. PATEL COLLEGE OF ENGINEERING



Ganpat University-U. V. Patel College of Engineering (GUNI-UVPC) is situated in Ganpat Vidyanagar campus. It was established in September 1997 with the aim of providing educational opportunities to students from It is one of the constituent colleges of Ganpat University various strata of society. It was armed with the vision of educating and training young talented students of Gujarat in the field of Engineering and Technology so that they could meet the demands of Industries in Gujarat and across the globe. The College is named after Shri Ugarchandbhai Varanasi Bhai Patel, a leading industrialist of Gujarat, for his generous support. It is a self- financed institute approved by All India Council for Technical Education (AICTE), New Delhi and the Commissionerate of Technical Education, Government of Gujarat. The College is spread over 25 acres of land and is a part of Ganpat Vidyanagar Campus. It has six ultra- modern buildings of architectural splendor, class rooms, tutorial rooms, seminar halls, offices, drawing hall, workshop, library, well equipped departmental laboratories, and several computer laboratories with internet connectivity through 1 Gbps Fiber link, satellite link education centre with two-way audio and one-way video link. The superior infrastructure of the Institute is conducive for learning, research, and training. The Institute offers various undergraduate programs, postgraduate programs, and Ph.D. programs. Our dedicated efforts are directed towards leading our student community to the acme of technical excellence so that they can meet the requirements of the industry, the nation and the world at large. We aim to create a generation of students that possess technical expertise and are adept at utilizing the technical 'know-hows' in the service of mankind. We strive towards these Aims and Objectives:

- To offer guidance, motivation, and inspiration to the students for well-rounded development of their personality.
- To impart technical and need-based education by conducting elaborated training programs.
- To shape and Mold the personality of the future generation.
- To construct fertile ground for adapting to dire challenges.