# HOMECHEF WEB APPLICATION

## INSTANT ACCESS TO HOME COOKED MEALS

**A MINI PROJECT REPORT**

***Submitted by***

**S A SUNIL ALDO**

**142224621057**

***In partial fulfillment for the award of the degree of***

**MASTER OF COMPUTER APPLICATIONS**


**SRM VALLIAMMAI ENGINEERING COLLEGE**

**(An Autonomous Institution)**

**SRM NAGAR, KATTANKULATHUR**

**ANNA UNIVERSITY, CHENNAI 600 025**

**MAY, 2025**

## BONAFIDE CERTIFICATE

Certified that this project report **"HOMECHEF WEB APPLICATION - INSTANT ACCESS TO HOME COOKED MEALS"** is the Bonafide work of **"SUNIL ALDO S A (14224621057)"** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Signature of the Project Guide
Mr. K. Maniraj M.Sc., M.Tech., (Ph.D)
Assistant Professor (Sr. Gr),
Department of Computer Applications,
SRM Valliammai Engineering College,
Kattankulathur.

Signature of the Project Coordinator
Dr. K. Ponmozhi, M.Tech., Ph.D.,
Associate Professor,
Department of Computer Applications,
SRM Valliammai Engineering College,
Kattankulathur.

Signature of the HOD
Dr. S. Parthasarathy MCA., M.Tech., MBA., Ph.D.
Professor and Head,
Department of Computer Applications,
SRM Valliammai Engineering College,
Kattankulathur.

**Submitted to the University Examination held on _____ at SRM Valliammai Engineering College, Kattankulathur.**

**INTERNAL EXAMINER** **HOD**

# ACKNOWLEDGEMENT

The present research work has reached its zenith with the constant help and encouragement from many people. Present study is the product of the interaction of a number of minds. Knowingly and unknowingly, many people have contributed to the completion of this study. Therefore, I earnestly thank all those who have helped me in the pursuit of my research

At the outset I express my deep respect and earnest thanks and cavernous sense of gratitude to our esteemed Founder Chairman & Chancellor **Dr. T. R. Paarivendhar**, **Thiru. Ravi Pachamuthu**, Chairman SRM Group & Pro-Chancellor (Admin). **Ms. R. Harini**, Correspondent, SRM VEC and authorities of SRM Valliammai Engineering College for the patronage of our welfare rooted in the academic year.

We express our heartfelt thanks to our Director, **Dr.B.Chidhambara Rajan,M.E., Ph.D.,** and we acknowledge our Principal **Dr. M. Murugan, M.E., Ph.D.,** with respect for his leadership and presence throughout our academic journey. We extend our sincere gratitude to our Vice Principal, **Dr. S. Visalakshi, M.E., Ph.D.,** for her unwavering commitment and dedication that have greatly contributed to the growth of the institution and pursuit department of excellence.

We extend our heartfelt gratitude to our Professor and Head of the Department – Computer Applications, **Dr. S. Parthasarathy, M.C.A., M.Tech., M.Phil., M.B.A., Ph.D.,** for his unstinted support and encouragement to motivate us to push our boundaries and achieve our goals.

We sincerely thank our project coordinators **Dr. V. Santhana Marichamy, M.C.A., M.E., Ph.D.,** and **Mr.M.Nagarajan, MCA.,M.Tech.,(Ph.D)** for their dedicated supervision and valuable guidance throughout the project. Their timely support and insightful feedback were crucial to the successful completion of our work.

We gratefully acknowledge our project guide, **Mr.K.Maniraj, M.Sc., M.Tech (Ph.D)**, for his consistent support and expert advice that helped us overcome challenges and complete our project effectively. We would also like to thank all the teaching and non-teaching staff members of our department for their support during the project work, especially for providing us with the essential facilities.

# ABSTRACT

This project presents the design and development of a web-based food service platform that connects home cooks with students and working professionals residing in PGs and rented accommodations. The platform aims to address the challenges faced by individuals who struggle with daily cooking due to busy schedules, limited access to kitchen resources, and lack of reliable cooking help. Users can book services from cooks who offer in-home cooking, takeaway meals, or event-based catering.

The system supports three main roles: Admin, Cooks, and Users. Users can register, browse available chefs, personalize their meal preferences, and book cooking services as per their needs. Cooks can manage their profiles, update availability, and handle bookings, while the Admin oversees platform operations to ensure efficiency and reliability.

The project also explores the use of personalization techniques and machine learning-based recommendation algorithms to enhance user satisfaction. Additionally, it addresses challenges such as meal planning, ingredient sourcing, and dietary customization.

Experimental results indicate that the platform improves access to healthy, home-cooked meals and provides a practical solution to modern-day cooking constraints.

# TABLE OF CONTENTS

# CHAPTER I

# INTRODUCTION

This chapter deals with the application study of the project domain, software specifications, and a brief overview of the technologies which is used for “HOMECHEF WEB APPLICATION”.

## 1.1 PROBLEM DEFINITION

In today's fast-paced world, many individuals face a lack of time for cooking due to demanding work schedules and busy lifestyles. This often results in the dependence on fast food or processed meals, which are typically unhealthy and lack essential nutrients. The inability to consistently prepare home-cooked meals affects not only physical well-being but also contributes to poor eating habits over time.Adding to this challenge is the difficulty in finding reliable and skilled cooks. Users often struggle to identify trustworthy individuals who can either cook at their residence or offer hygienic takeaway services. The process is often uncertain, time-consuming, and lacks a verified platform for safe hiring.

Consequently, unhealthy eating habits become a norm due to limited access to affordable and nutritious home-style food. Many individuals, especially those living alone or away from family, find it hard to maintain a balanced diet without dependable meal options.

Organizing special events or occasional family gatherings also becomes a hassle without a structured system to hire temporary cooks or chefs. Users typically rely on word-of-mouth references, which are not always reliable or available.

These challenges highlight the need for a centralized, digital solution that connects users with verified home chefs or cooking services. Such a platform would help address time constraints, promote healthy eating, and simplify the process of finding cooking support for daily needs and special occasions.

## 1.2 SYSTEM ENVIRONMENT

The hardware and software environment used for the development of the project is presented in detail.

### HARDWARE REQUIREMENTS

1. Operating system : Windows
2. Processor : Intel Core i3 or higher
3. RAM : 8.00 GB
4. Storage : 256GB SSD

### SOFTWARE REQUIREMENTS

1. Front-end : HTML, CSS, JavaScript
2. Back-end : PHP version 8.0+
3. DataBase : Mysql Version 5.7 or above
4. Web Server & Host : Apache, Xamp Version 8.2.12
5. Code Editor : Visual Studio, Sublime Text

# CHAPTER II

# SYSTEM ANALYSIS

## 2.1 SYSTEM DESCRIPTION

The proposed system is a web-based home chef service platform that addresses the growing need for accessible, nutritious, and home-cooked meals, especially among students and working professionals living in PGs or rented accommodations. It bridges the gap between users and local chefs by offering a seamless interface for food ordering, cook hiring, and event-based meal planning.

Unlike traditional food delivery apps that only offer restaurant options, this platform focuses on personalized, home-style meals and allows users to book cooks for in-home cooking or takeaway services. The system is designed to enhance both user convenience and chef engagement through intuitive interactions and service customization.

**Key Features of the System**

**1. Multi-Role Access:**

- The system supports three roles: **Admin**, **Cook**, and **User**, each with tailored dashboards and access permissions.

**2. Service Booking and Customization:**

- Users can search for nearby chefs, browse meal options, and book services (in-home, takeaway, or event-based).
- Options to personalize meals based on dietary needs or preferences.

**3. Real-Time Availability & Order Management:**

- Cooks can update their service availability and manage incoming orders.
- Admin can monitor activities, resolve disputes, and ensure quality control.

## 2.2 LITERATURE STUDY

Online food delivery systems have transformed the food service industry by introducing convenience, variety, and real-time accessibility. Major platforms like Swiggy and Zomato primarily focus on restaurant-to-consumer models. However, there's a noticeable gap in platforms that cater specifically to **home-style food and personalized cooking services**.

This project aims to fill that niche by enabling users to connect directly with **home chefs**, offering a more affordable, nutritious, and community-driven alternative to fast food. Unlike existing systems, the proposed platform allows **booking cooks for personalized services**, which is especially beneficial for daily meals, dietary restrictions, or special events.

Studies such as Rahman et al. (2020) highlight the exponential growth of the Indian food delivery market, which is projected to reach $16.1 billion by 2023. This growth is driven by changing urban lifestyles, lack of time for cooking, and the convenience of online platforms.

The proposed system goes a step further by combining technology, community engagement, and accessibility. It not only helps users find reliable food services but also provides cooks with employment opportunities and digital exposure, aligning with broader goals like economic empowerment and healthier food habits.

## 2.3 Problem Analyzed

In many local communities, talented home-based chefs struggle to reach potential customers due to the lack of a proper platform. Meanwhile, people often seek affordable, healthy, and homemade food alternatives but face difficulties finding reliable sources. Existing food delivery systems mostly cater to restaurants, leaving out home kitchens that could offer personalized, nutritious meals.

## 2.4 Objective

The objective of the HomeChef project is to create a user-friendly web application that connects home-based chefs with customers looking for homemade meals in their area. The system aims to provide location-based chef listings, an easy booking process, and admin tools for managing users and services—promoting local entrepreneurship and improving access to quality food.

## 2.5 USE CASE MODEL

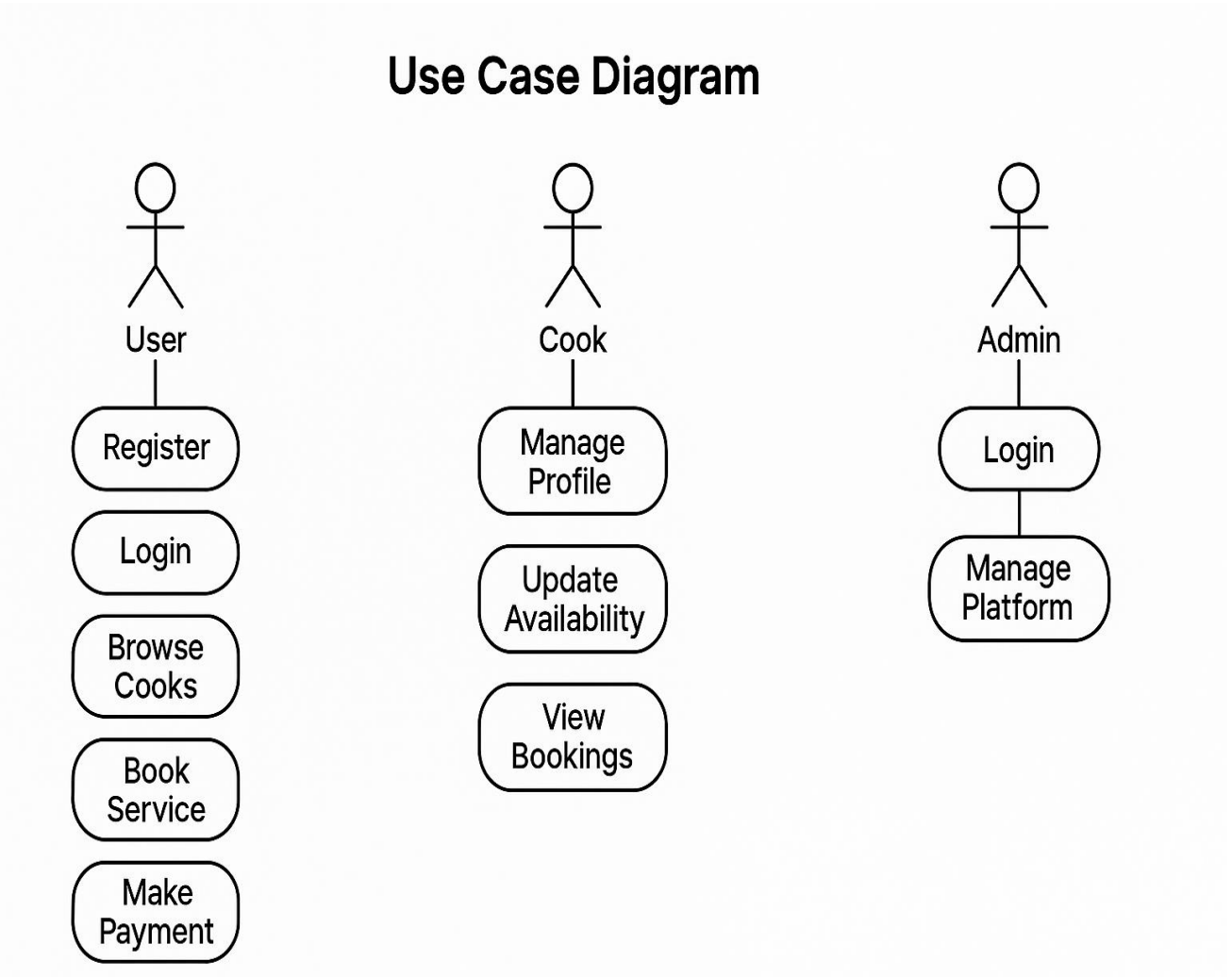



Figure 2.5 (use case model)

## 2.1 SOFTWARE REQUIREMENT SPECIFICATION

### *FUNCTIONAL REQUIREMENTS*

- The system shall allow users (students, professionals) to register and log in with a unique email and password.
- The system shall enable users to browse cook profiles based on location and availability.
- The system shall allow users to book services for in-home cooking, takeaway, or event-based catering.
- The system shall allow cooks to update their availability and manage incoming service requests.
- The system shall provide an admin panel for administrators to manage users, cooks, and bookings**.**

### *NON-FUNCTIONAL REQUIREMENTS*

- The system shall maintain a **res**ponse time of less than 2 seconds for user actions under normal usage.
- The platform shall be compatible with major browsers and accessible on mobile, tablet, and desktop devices.
- The system shall use SSL encryption to ensure secure data transmission and protect user information.
- The system shall ensure 99.9% uptime and perform automatic daily data backups.
- The user interface shall be intuitive and user-friendly, requiring minimal training for first-time users.

# CHAPTER III

# SYSTEM DESIGN

The design model chapter conveys a detailed description and presents a brief overview about the design model of the application

## 3.1 ARCHITECTURAL DIAGRAM

The architecture design model conveys a detailed description and presents a brief overview about the architecture of the application

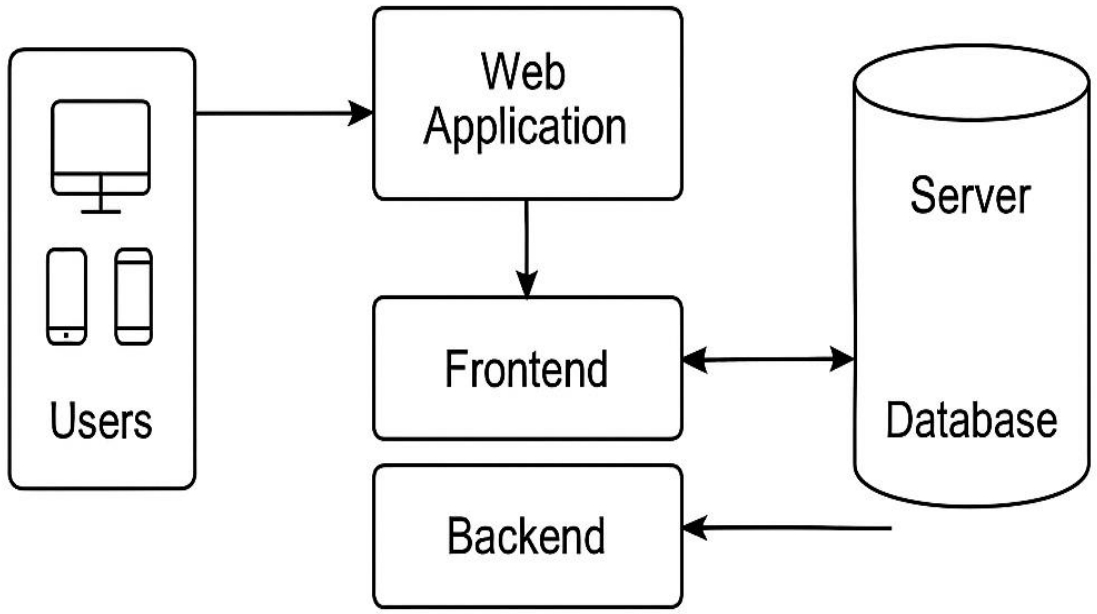



Figure 3.1 (Architectural diagram)

1. **User (Food Seeker / Customer)**

   The primary users of the HomeChef platform include food seekers such as students, working professionals, families, or event organizers. These users interact with the web interface to search, view, and book home chefs. The platform is designed to be user-friendly, enabling easy navigation and input of location and service requirements.

**2. Service Request Input**

Users initiate their search by entering specific inputs such as their location (e.g., "Search chefs near Koramangala"), the type of service they need (e.g., booking for a home visit or ordering a meal), along with preferred date and time. These inputs are captured through intuitive form fields, dropdowns, and filter options on the interface.

**3. Request Processing & Filtering Module**

Once the user submits a request, this module processes the input data to filter available chefs. Filtering is based on parameters like location proximity (using GPS or address-based filtering), the chef's availability calendar, and the type of service offered such as in-home cooking, takeaway, or event catering. This ensures users only see relevant and currently available options.

**4. Location & Service Matching Engine**

This engine uses the user's specified location to match them with chefs operating within a specific service radius. It integrates with Google Maps API to calculate the distance and assess the feasibility of travel for in-home cooking services. Chefs with high ratings and active status are prioritized to enhance the user experience.

**5.Chef Profile and Menu Module**

Once chefs are filtered, users can view detailed chef profiles that include a profile photo, bio, experience, and list of available dishes with prices. Each profile also provides information about service types and available time slots. Users can browse the menu, add dishes to a cart, or directly proceed with a booking.

### 6. Booking & Confirmation System

After selecting a chef and finalizing service details, users can confirm the booking by selecting the desired time, service type (home service or pickup), and location. Upon confirmation, notifications are sent to both the user and the chef. The system updates the booking status in real-time, reflecting stages such as Pending, Confirmed, or Delivered.

### 7. Payment & Transaction Module

The platform supports secure online payments via UPI, debit/credit cards, and digital wallets. This module handles invoice generation, commission distribution between the chef and the admin, and manages refunds or disputes through the admin dashboard to maintain transparency and trust.

### 8. Feedback & Review Loop

After the service is completed, users are encouraged to rate the chef and leave feedback. This review system helps future users make informed decisions and maintains service quality. Chefs can also view customer feedback to improve their offerings and build a trusted reputation on the platform.

### 3.2 STRUCTURAL DESIGN

#### CLASS DIAGRAM

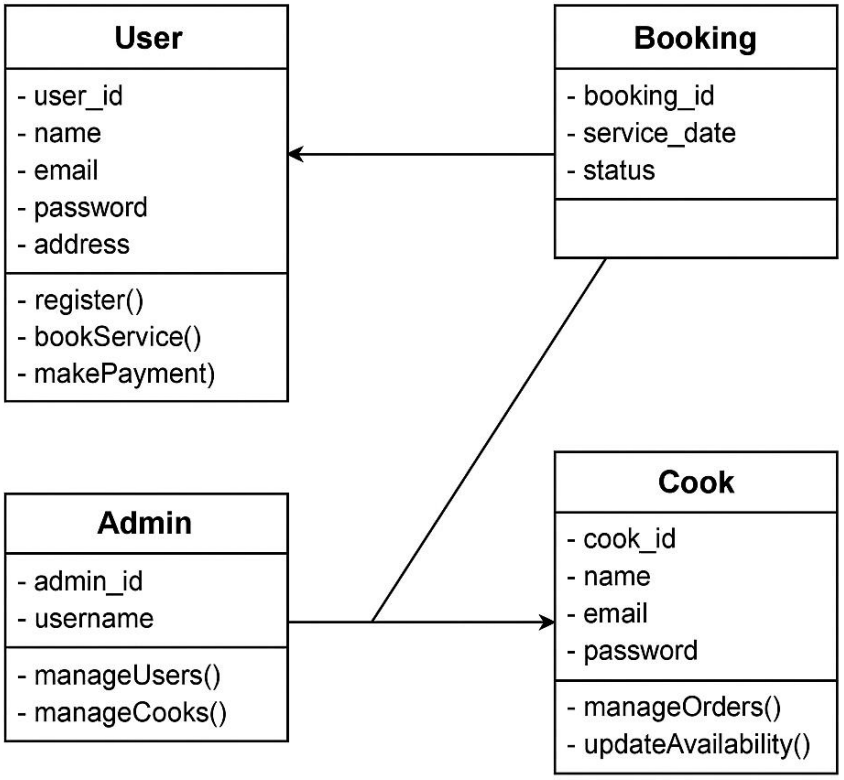



Figure 3.2 (Class diagram)

## Main Activity

**Responsibility:**

Acts as the central hub for user interactions, navigation, and launching core features such as chef search, bookings, orders, and settings.

**Attributes:**

- btnFindChefs, btnMyBookings, btnOrders, btnProfile, btnSettings: Buttons for navigating to various features.
- locationManager: Used for location-based chef discovery.

**Methods:**

- onCreate(): Initializes UI components and services.
- dispatchKeyEvent(): Allows keyboard/voice button inputs for accessible navigation.
- onActivityResult(): Handles results from activities like location selection or chef booking confirmation.

## Find Chefs Activity

**Responsibility:** Handles searching and displaying chefs based on user's location or selected filters.

**Attribute:** locationInput, serviceTypeDropdown, chefListRecyclerView

**Methods:**

- onCreate(): Sets up the UI, location fetching, and chef listing logic.
- fetchChefs(): Queries backend/database for available chefs in the area.
- onChefClick(): Opens detailed chef profile with booking option.

## Chef Profile Activity

**Responsibility:** Displays chef details, available dishes, ratings, and booking options.

**Attributes:**

- chefNameText, menuListView, ratingBar, bookButton
- tts: Reads details aloud on request.

**Methods:**

- onCreate(): Loads chef data and menu from backend.
- bookChef(): Navigates to booking form.

- speakChefInfo(): Uses TTS to read chef bio, availability, and ratings.

## Booking Activity

**Responsibility:** Manages booking a chef for an event, home visit, or food delivery.

**Attributes:**

- datePicker, timeSlotDropdown, addressInput, confirmButton
- tts: Confirms selections and reads back input for visually impaired users.

**Methods:**

- onCreate(): Initializes booking form UI.
- validateBooking(): Checks time, location, availability.
- submitBooking(): Saves booking and updates backend.

## My Bookings Activity

**Responsibility:** Displays current, past, and upcoming chef bookings.

**Attributes:**

- bookingListView, tts: Reads booking details aloud on selection.

**Methods:**

- onCreate(): Loads user bookings from the database.
- onBookingSelect(): Opens detail view or cancel option.
- speakBookingDetails(): Uses TTS for accessibility.

## Database Helper

**Responsibility:** Manages all local SQLite database operations such as saving bookings or offline chef data.

**Methods:**

- onCreate(): Creates tables for chefs, bookings, users.
- onUpgrade(): Upgrades DB schema on version change.
- addBooking(), getChefs(), getBookings(): Query operations.

## Settings Activity

**Responsibility:** Manages app settings like language, accessibility (TTS), and notification preferences.

**Attributes:**

- languageSelector, voiceModeSwitch, tts

**Methods:**

- onCreate(): Loads and saves user preferences.
- handleVoiceSettings(): Enables voice prompts and feedback.
- dispatchKeyEvent(): Enables key or voice control navigation.

### Order History Activity

**Responsibility:**Shows completed food orders and their status.
Attributes:

- orderListView, tts: Reads out order details.

**Methods:**

- onCreate ( ) : Displays orders and invoices.
- speakOrderSummary ( ) : Uses TTS to read the summary.

### Chef Location Service

**Responsibility:** Tracks user location and fetches nearby chefs in real time.

**Methods:**

- getCurrentLocation ( ) : Uses GPS or network for current position.
- getNearbyChefs ( ) : Returns chef list based on radius.

### Notification Receiver

**Responsibility:**

Handles real-time notifications for chef confirmation, booking updates, or order status.

**Methods:**

- onReceive () : Triggers notification display or TTS announcement when new data is received.

## 3.3 BEHAVIOURAL DESIGN

### ACTIVITY DIAGRAM

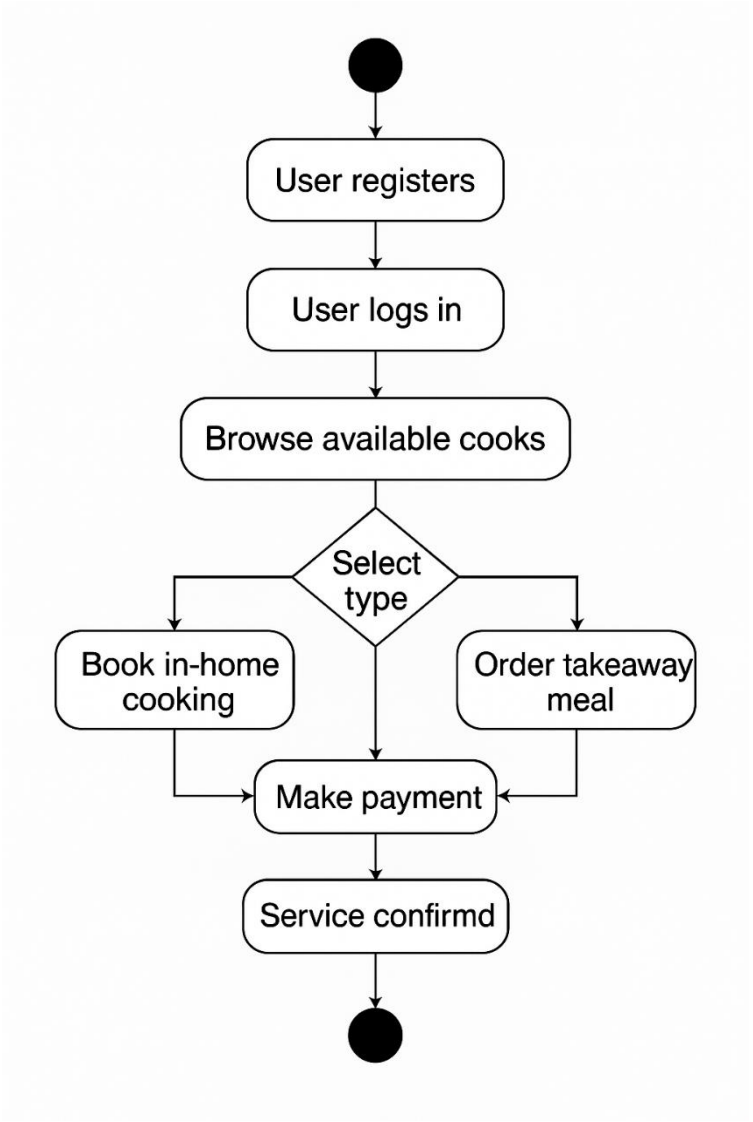



Figure 3.3 (Activity diagram)

When the app is launched, users are taken to a main screen that offers five primary modules designed for smooth interaction and chef discovery:

**Settings**

Allows users to configure preferences such as language, notifications, and location services. Users can enable or disable features using simple touch or voice actions (if supported).

**Chef-Search**

Enables users to find home chefs based on their entered location. The app lists chefs available in the area, showing cuisine type, ratings, and availability. Results can be sorted or filtered for convenience.

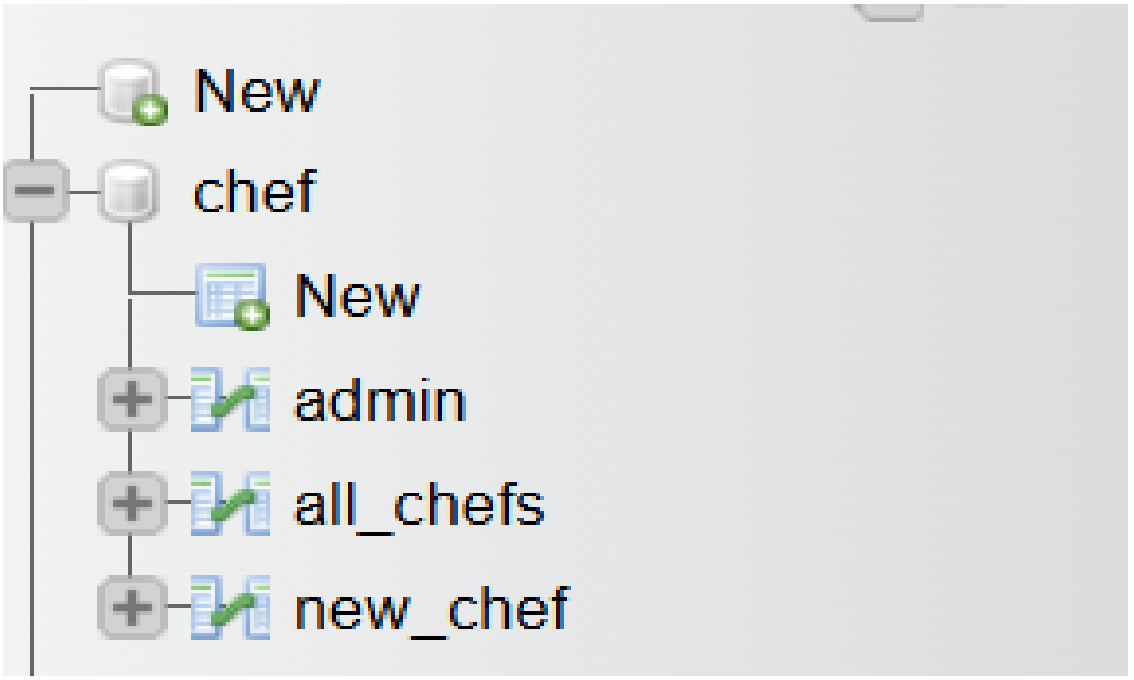
**Bookings**

Users can book a chef for a specific time and date. The interface collects essential information (meal type, timing, special instructions) and confirms the booking with real-time status updates.

**My-Notes**

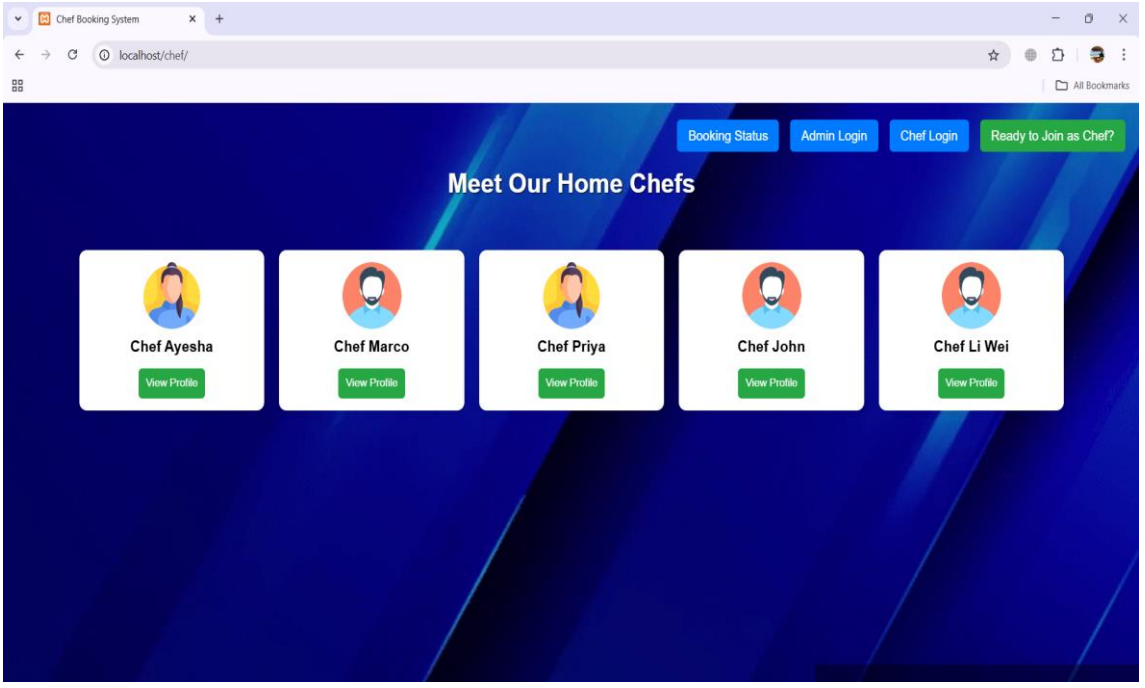
A section for users to save custom food preferences, allergy alerts, or favorite chefs. Notes are stored internally and can be accessed during future bookings to ensure consistency.

## 3.4 TABLE DESIGN

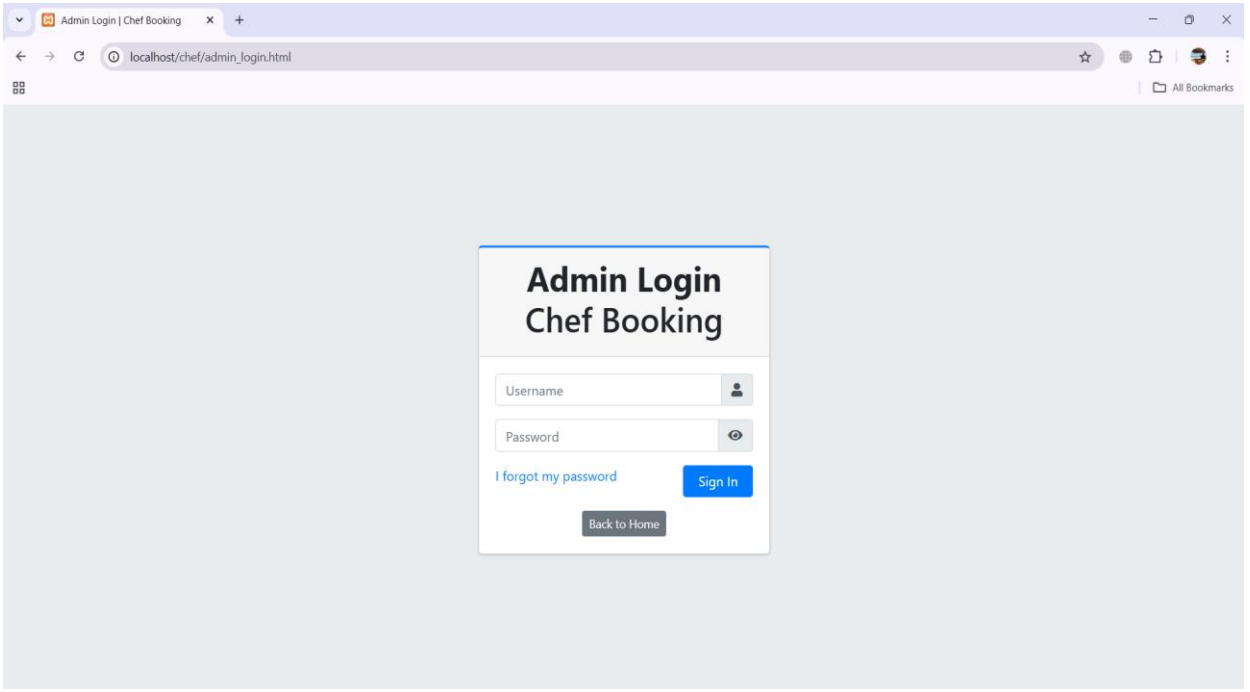



(Screen shows the Table design in database)

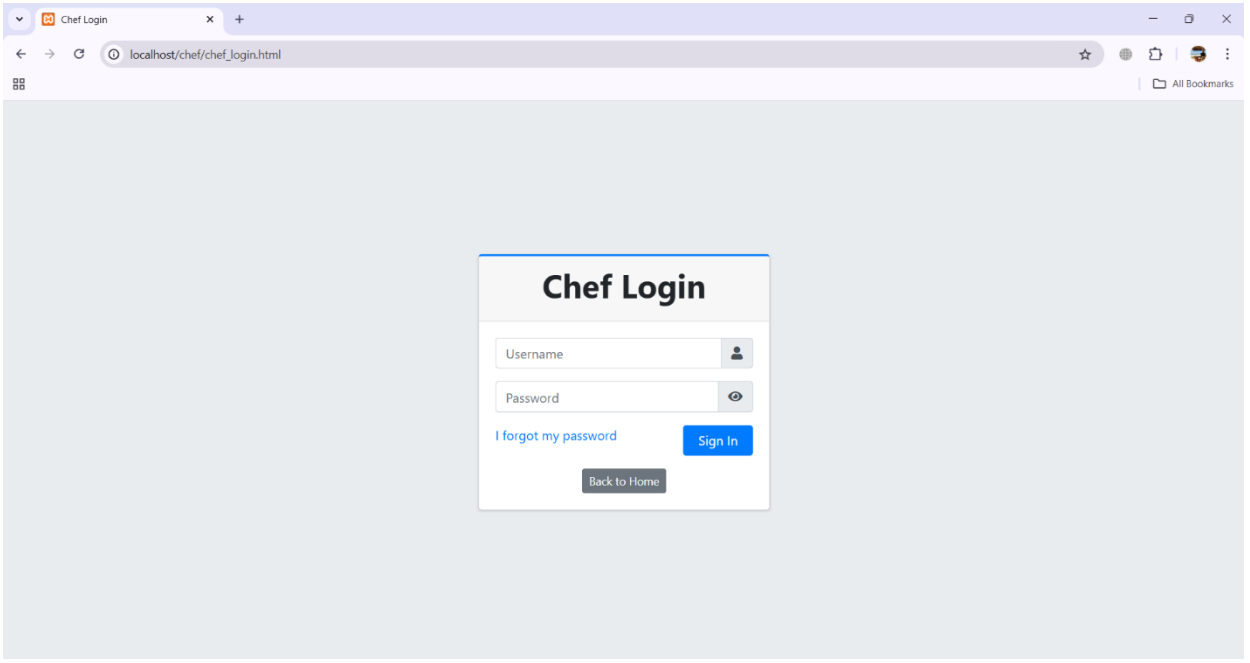
## 3.5 USER INTEFACE DESIGN



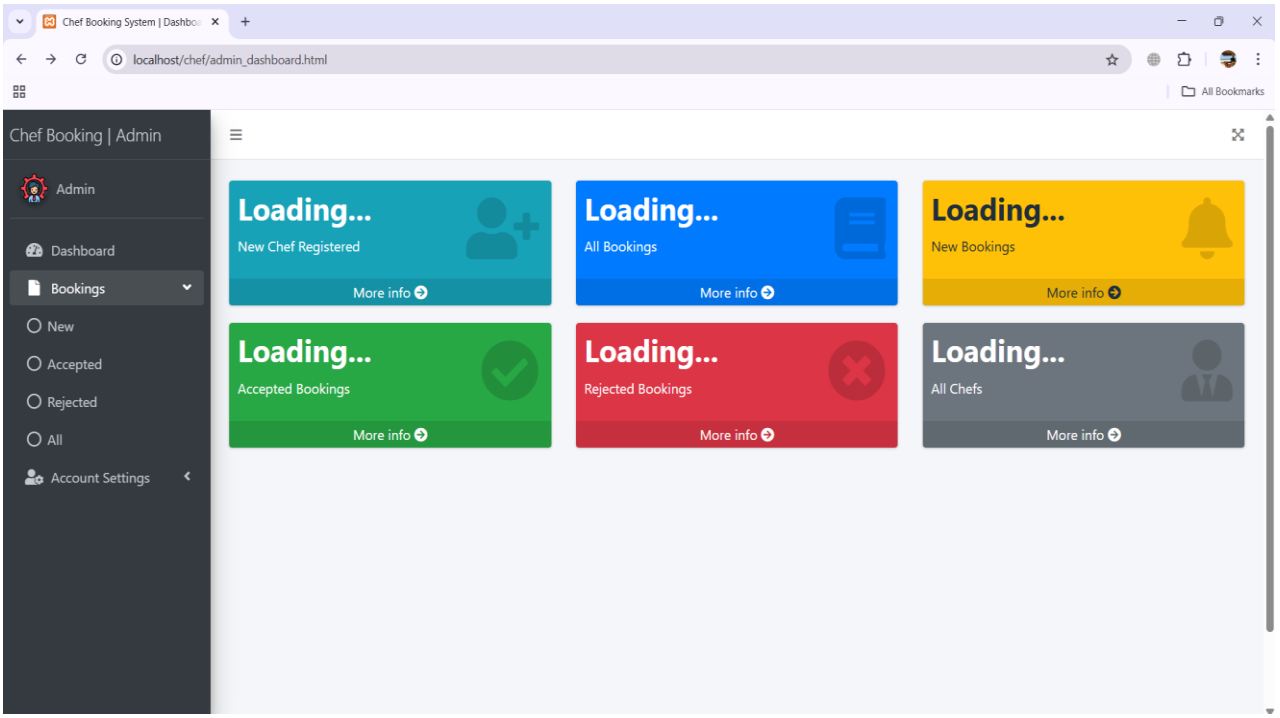

( Screen shows the main menu, where the user can see the chef profiles and choose.)



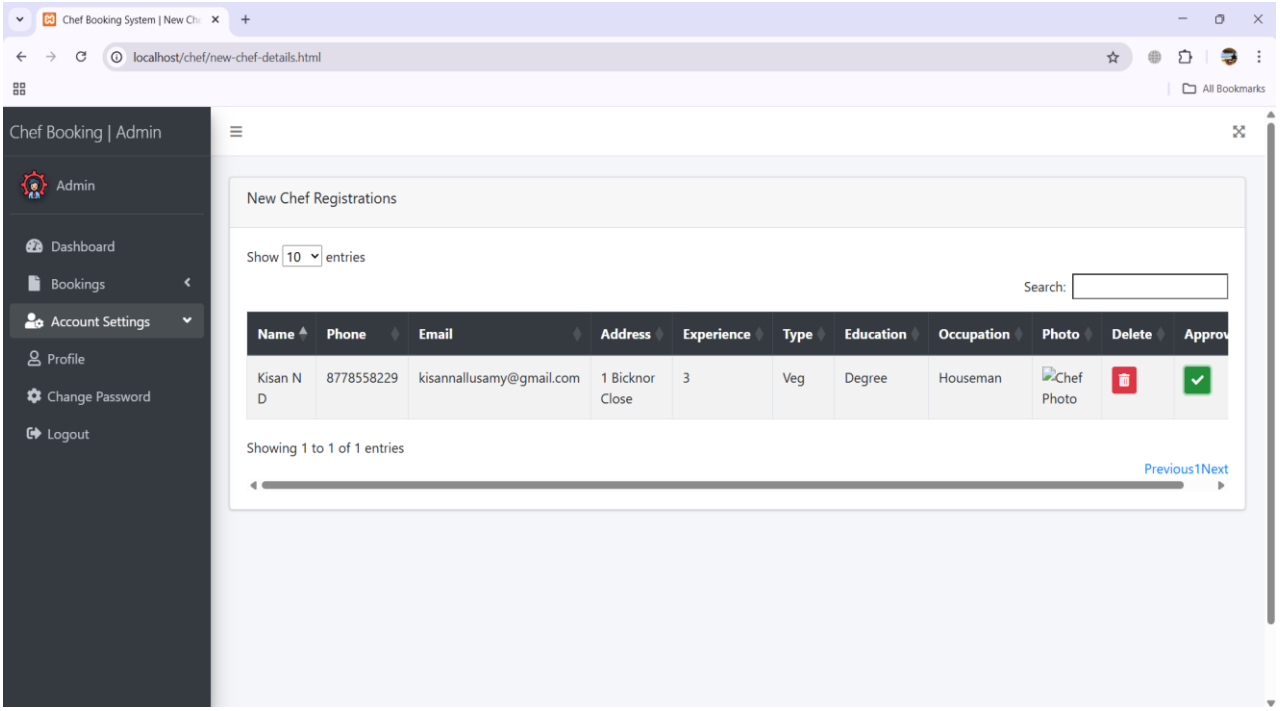

( Screen shows the Admin panel, where the admin can see the chef profiles and Manage it.)



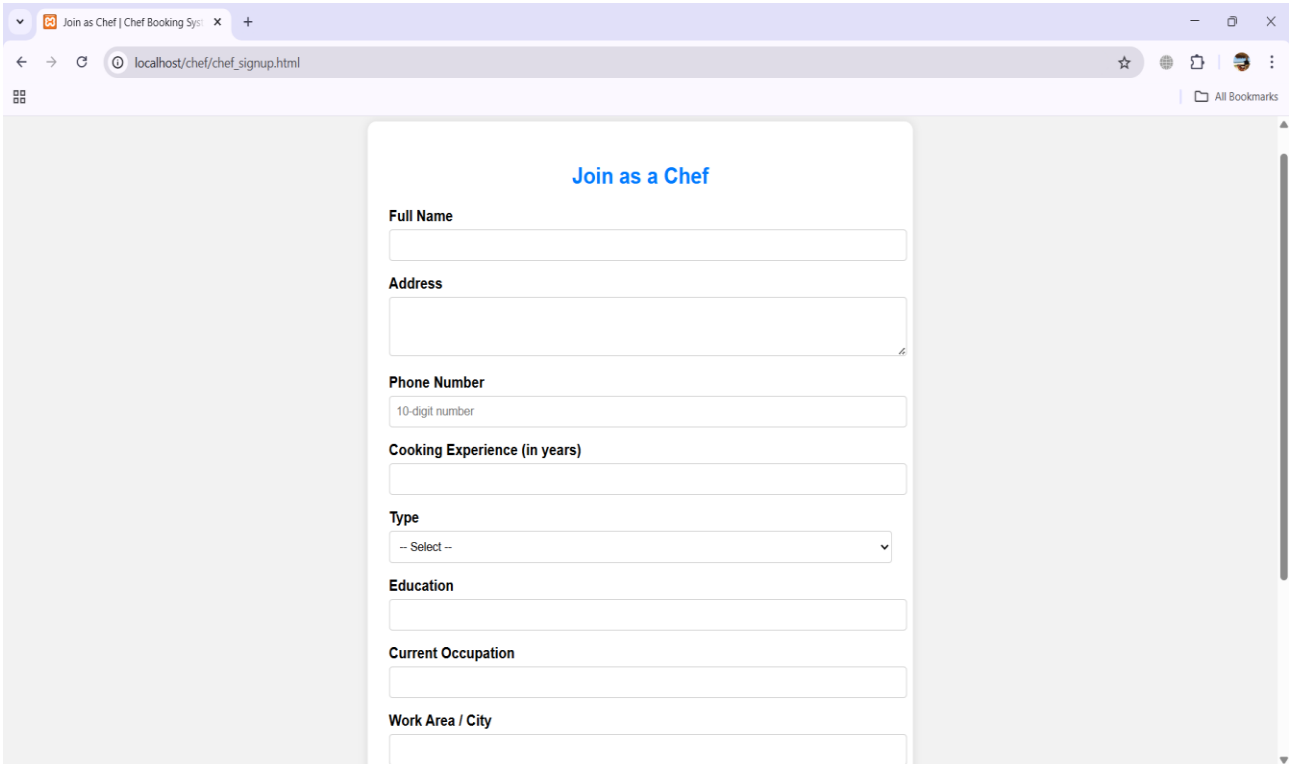

( Screen shows the Chef panel, where the chef can see their orders and Manage it.)



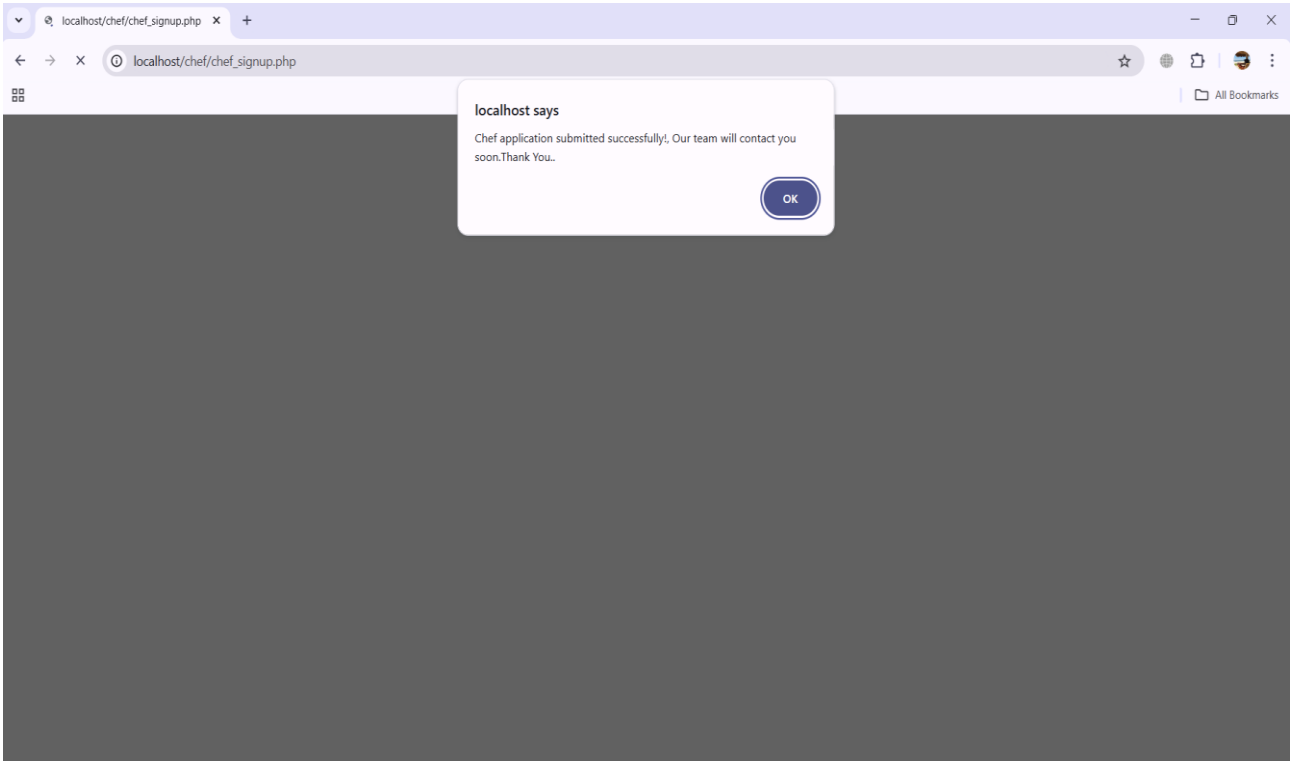

(Screen shows the Admin dashboard, where admin can manage the cooks and users.)




(Screen shows the registration page, where the registered chefs details will be shown to admin.)




(Screen displays the registration page where new chefs can register.)




(Screen displays the message that registration submitted successfully.)

### 3.6 DEPLOYMENT DESIGN

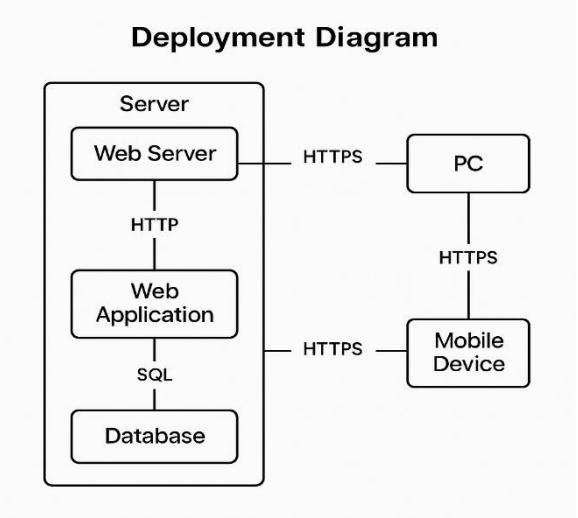



Figure 3.6 (Deployment design)

**Web Browser (Client Interface)**

• Acts as the platform where users (customers, chefs, admins) interact with the system.

• Executes frontend logic and renders the UI built with HTML, CSS, and JavaScript.

• Communicates with backend services via HTTP requests.

**Home Page (Landing Controller)**

• Entry point of the web application.

• Provides access to chef browsing, location-based search, login, and registration.

• Directs users to role-specific dashboards (Customer, Chef, Admin).

**Chef Management Module**

• Allows Admin and Sub-Admins to add, edit, approve, or remove chef profiles.

• Interfaces with the database to fetch and store chef data.

**Booking System**

• Enables customers to book chefs based on availability and location.

• Uses server-side logic to manage booking validation, conflict checking, and confirmation.

• Sends notifications (email/SMS) upon successful bookings.

**Authentication & User Roles**

• Handles login and signup for customers, chefs, admins, and sub-admins.

• Session management via cookies or JWTs for access control.

• Redirects to dashboards based on user role.

**Database (MySQL/PostgreSQL)**

• Stores users, chef profiles, bookings, locations, and feedback.

• Queried and updated by backend services using ORM or raw SQL.

• Supports search and filtering functionality for chef discovery.

**Admin Panel**

• A secure interface for administrators to manage users, chefs, bookings, and reports.

• Accessible only to authenticated admin roles.

• Includes CRUD operations, analytics, and approval workflows.

## 3.7 NAVIGATION DESIGN

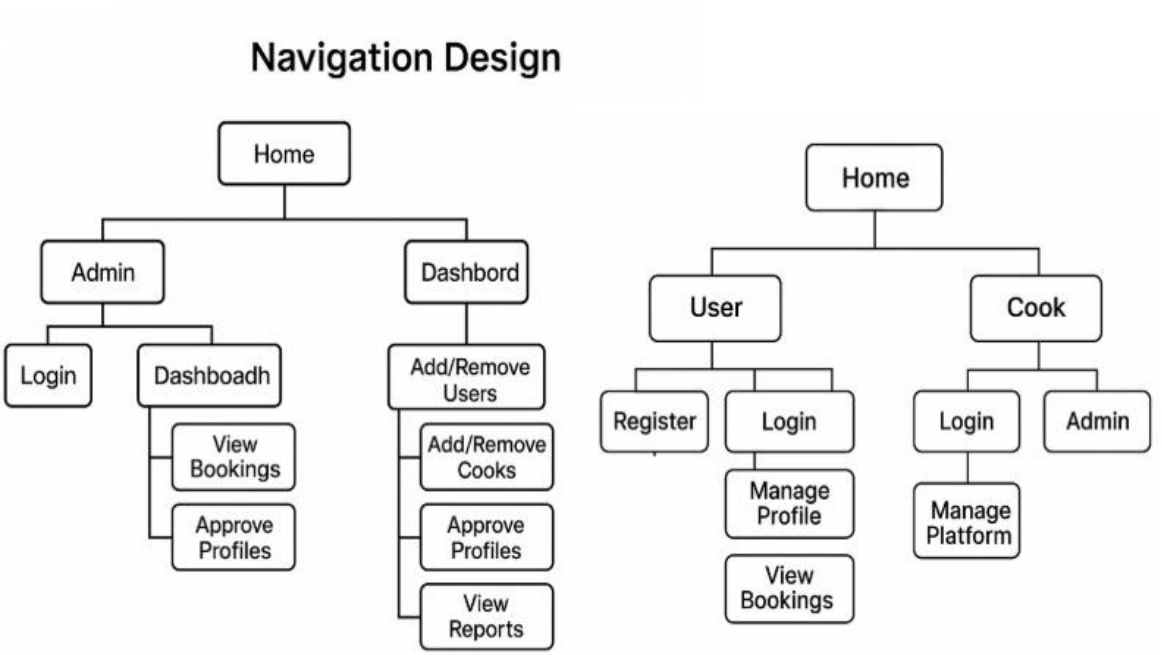



Figure 3.7 (Navigation design)

**User Module**

- **Register/Login:** Allows users to create an account or log in securely.
- **Browse Cooks:** Lets users view available chefs based on filters like cuisine, location, and availability.
- **Book Service:** Enables users to book services for in-home cooking, takeaway, or events.
- **Make Payment:** Provides online payment options (UPI, cards, etc.) to confirm bookings.
- **View Bookings:** Displays upcoming and past bookings for user reference.

**Cook (Chef) Module**

- **Register/Login:** Allows cooks to register and access their account.
- **Update Availability:** Lets cooks specify their working days, time slots, and service types.
- **Manage Orders:** Enables viewing, accepting, or declining incoming service requests.
- **Edit Profile:** Update personal details, specializations, pricing, and service locations.
- **View Feedback:** Lets cooks access reviews and ratings from past users.

**Admin Module**

- **Login:** Secure access to the admin dashboard.
- **Manage Users:** View, block, or delete user accounts as needed.
- **Manage Cooks:** Approve cook registrations, edit profiles, or remove chefs violating guidelines.
- **Manage Bookings:** Monitor and resolve disputes or booking conflicts.
- **Generate Reports:** View analytics on service usage, revenue, and system health.

# CHAPTER IV

# SYSTEM TESTING

This chapter introduces the testing processes involved in the Home Cook Booking Web Platform project. Testing was conducted to validate both the internal logic and external functionality of the system. The goal was to uncover errors and ensure that defined inputs produce the expected results.

## 4.1 TEST CASES AND TEST REPORTS

### Performance Analysis On Various Testing

Software testing was performed during the pre-implementation stage using multiple strategies to validate the system's modules and overall behavior.

### Functional Testing

Functional testing was carried out to verify that all features of the HomeChef platform performed as intended. Core functionalities like chef search, user registration, login, booking services, and payments were tested to ensure proper execution. Each user interaction, such as selecting a chef, choosing a time slot, or completing a payment, was verified to match expected behavior. Special attention was given to business rules—for example, preventing double bookings and ensuring that only available chefs appeared in search results.

### Usability Testing

Once individual modules passed unit testing, interface testing was performed to confirm smooth user interaction. The focus was on verifying dropdowns, buttons, links, and form inputs. For instance, the "Book Service" and "Manage Profile" interfaces were tested to validate data submission, correct redirect after actions, and message alerts. The UI was checked for responsiveness across devices (PC, mobile).

**Compatibility Testing**

To ensure the application performs consistently across various platforms, compatibility testing was done on multiple browsers such as Chrome, Firefox, Safari, and Edge. Additionally, the website was tested on different screen sizes (desktop, tablet, and smartphones) to check responsiveness. All visual elements, input fields, and interactive features were checked to confirm consistent behavior and display across devices and operating systems.

**Performance Testing**

Performance testing was carried out to evaluate the system's behavior under stress. Load tests simulated multiple users accessing the platform at once, especially during peak times like lunch and dinner hours. The responsiveness of key features such as the booking module, chef listings, and payment gateway were measured. The application maintained acceptable speed and stability under concurrent access, with improvements made to optimize page load times and database queries.

**Security Testing**

Security testing focused on protecting user data and ensuring secure transactions. The login system was tested for vulnerabilities such as SQL injection and session hijacking. Payment integration was reviewed to ensure encryption and secure handling of financial data. Password storage, authentication mechanisms, and data validation processes were evaluated to maintain platform integrity and user trust.

# CHAPTER V

# SYSTEM IMPLEMENTATION

**Install XAMPP or WAMP (for local development)**

- XAMPP/WAMP provides an integrated package including Apache server, MySQL, and PHP.
- Recommended PHP version: 8.0+
- MySQL version: 5.7 or above

**Install a Code Editor**

- Preferred: Visual Studio Code
- Alternative options: Sublime Text, PHPStorm

**Database Setup**

- Open phpMyAdmin through your local server (http://localhost/phpmyadmin).
- Create a new database: chef
- Import the provided SQL schema file to initialize tables such as admin, new_chef, all_chefs, etc.

**Project Folder Configuration**

- Place your project folder (e.g., homechef_project) inside the htdocs directory (for XAMPP) or www (for WAMP).
- Ensure file paths and base URLs are correctly configured in configuration files (e.g., config.php).

**Application Metadata**

- Project Name: HomeChef
- Base URL: http://localhost/homechef_project
- Admin Panel Path: http://localhost/homechef_project/admin
- Database Name: chef

**PHP and MySQL Integration**

- Database connection is managed via a centralized file: dbconnect.php
- Prepared statements are used for secure data interaction (preventing SQL injection).

**JavaScript & Libraries**

- Uses jQuery for AJAX requests and form handling.
- Bootstrap 5 is integrated for responsive design and UI components.

**Run the Project**

- Start Apache and MySQL via XAMPP/WAMP Control Panel.
- Access the app in the browser via http://localhost/homechef_project
- Admin login available at http://localhost/homechef_project/admin

# CHAPTER VI

## 6.1 CONCLUSION

The HomeChef web application is a user-friendly platform that connects home-based chefs with local customers seeking fresh, homemade meals. It simplifies the process of discovering chefs, viewing menus, and placing orders through an intuitive interface. The platform not only enhances convenience for customers but also empowers home chefs—especially women and small-scale cooks—by offering them a digital space to showcase their culinary skills and earn income.Through features like location-based chef discovery, real-time booking, chef registration, and order management, HomeChef ensures a seamless and personalized food delivery experience. The inclusion of admin and sub-admin panels strengthens quality control, improves service monitoring, and ensures platform reliability.

In addition to supporting local entrepreneurship, the project promotes healthier eating by offering hygienic, home-cooked meals as an alternative to fast food. It encourages community interaction, reduces food delivery monopolies, and contributes to social and economic inclusion. By bridging the gap between home kitchens and customers, HomeChef supports Sustainable Development Goal 3 by promoting good health, and aligns with SDG 8 by fostering decent work and economic growth.

## 6.2 FUTURE ENHANCEMENT

### Integration with Smart Kitchen Devices

In future versions of the HomeChef platform, integrating with smart kitchen devices can significantly improve efficiency and user experience. By connecting the application with smart ovens, timers, and kitchen sensors, home chefs can automate parts of the cooking process, receive real-time alerts, and better manage their kitchen operations.

For example, the system can send reminders for order preparation time, adjust cooking settings based on the dish selected, or notify the chef when ingredients are running low. This smart integration will help chefs maintain consistent quality, reduce cooking errors, and save time—leading to quicker deliveries and increased customer satisfaction.

# APPENDIX

**HTML FORM:**

```
 <!DOCTYPE html>
<html>
<head>
  <title>Chef Booking System</title>
</head>
<body>
<!-- Background layers -->
<div class="background"></div>
<div class="overlay"></div>
<!-- Top buttons -->
<div class="top-buttons">
  <a href="check-status.php">Booking Status</a>
  <a href="admin_login.html">Admin Login</a>
  <a href="chefs/">Chef Login</a>
  <a href="chef_signup.html" class="join-btn">Ready to Join as Chef?</a>
</div>
<br><br>
<h1>Meet Our Home Chefs</h1>
<!-- Chef Cards -->
<div class="container">
  <div class="card female">
    <div class="avatar"></div>
    <h3>Chef Ayesha</h3>
    <button onclick="alert('Redirect to profile page')">View Profile</button>
  </div>
  <div class="card male">
    <div class="avatar"></div>
    <h3>Chef Marco</h3>
<button onclick="alert('Redirect to profile page')">View Profile</button>
```

```
  </div>
  <div class="card female">
    <div class="avatar"></div>
    <h3>Chef Priya</h3>
    <button onclick="alert('Redirect to profile page')">View Profile</button>
  </div>
  <div class="card male">
    <div class="avatar"></div>
    <h3>Chef John</h3>
    <button onclick="alert('Redirect to profile page')">View Profile</button>
  </div>
  <div class="card male">
    <div class="avatar"></div>
    <h3>Chef Li Wei</h3>
    <button onclick="alert('Redirect to profile page')">View Profile</button>
  </div>
</div>
</body>
</html>
```

**PHP SCRIPT:**

```
<?php
include("config.php"); // make sure this file sets up $con
if ($_SERVER["REQUEST_METHOD"] === "POST") {
    $name = mysqli_real_escape_string($con, $_POST['name']);
    $address = mysqli_real_escape_string($con, $_POST['address']);
    $phone = mysqli_real_escape_string($con, $_POST['phone_number']);
    $experience = (int) $_POST['cooking_experience'];
    $type = mysqli_real_escape_string($con, $_POST['type']);
    $education = mysqli_real_escape_string($con, $_POST['education']);
    $occupation = mysqli_real_escape_string($con, $_POST['occupation']);
    $work_area = mysqli_real_escape_string($con, $_POST['work_area']);
```

```
    $email = mysqli_real_escape_string($con, $_POST['email_id']);
    $photo = $_FILES['photo']['name'];
    $photo_tmp = $_FILES['photo']['tmp_name'];
    $upload_folder = "uploads/";
    if (!is_dir($upload_folder)) {
        mkdir($upload_folder, 0755, true);
    }
    $photo_path = $upload_folder . basename($photo);
    if (move_uploaded_file($photo_tmp, $photo_path)) {
        $query = "INSERT INTO new_chef
            (name, address, phone_number, cooking_experience, type, education, occupation, work_area, email_id, photo)
            VALUES
            ('$name', '$address', '$phone', '$experience', '$type', '$education', '$occupation', '$work_area', '$email', '$photo_path')";
        if (mysqli_query($con, $query)) {
            echo "<script>alert('Chef application submitted successfully!, Our team will contact you soon.Thank You..'); window.location.href='index.html';</script>";
        } else {
            echo "<script>alert('Database error: " . mysqli_error($con) . "');</script>";
        }
    } else {
        echo "<script>alert('Failed to upload photo.');</script>";
    }
} else {
    echo "<script>alert('Invalid request.');</script>";
}
?>
```

**HTML FORM:**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
```

```
  <title>Chef Booking System | Dashboard</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/admin-
lte@3.2/dist/css/adminlte.min.css">
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/bootstrap.min.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">
</head>
<body class="hold-transition sidebar-mini layout-fixed">
<div class="wrapper">
  <div id="navbar-container"></div>
  <div id="sidebar-container"></div>
  <div class="content-wrapper">
    <section class="content pt-4">
      <div class="container-fluid">
        <div class="row">
          <!-- ✅ New Chef Registered -->
          <div class="col-lg-4 col-6">
            <div class="small-box bg-info">
              <div class="inner">
                <h3 id="new-chef-count">Loading...</h3>
                <p>New Chef Registered</p>
                <ul id="new-chef-list" class="mt-2 mb-0 pl-3" style="font-size: 14px;"></ul>
              </div>
              <div class="icon"><i class="fas fa-user-plus"></i></div>
              <a href="new-chef-details.html" class="small-box-footer">More info <i class="fas fa-
arrow-circle-right"></i></a>
            </div>
          </div>
  <!-- All Bookings -->
          <div class="col-lg-4 col-6">
            <div class="small-box bg-primary">
              <div class="inner">
```

```
            <h3 id="all-bookings">Loading...</h3>
            <p>All Bookings</p>
          </div>
          <div class="icon"><i class="fas fa-book"></i></div>
          <a href="all-bookings.php" class="small-box-footer">More info <i class="fas fa-arrow-circle-right"></i></a>
        </div>
      </div>
      <!-- New Bookings -->
      <div class="col-lg-4 col-6">
        <div class="small-box bg-warning">
          <div class="inner">
            <h3 id="new-bookings">Loading...</h3>
            <p>New Bookings</p>
          </div>
          <div class="icon"><i class="fas fa-bell"></i></div>
          <a href="new-chef-details.html" class="small-box-footer">More info <i class="fas fa-arrow-circle-right"></i></a>
        </div>
      </div>
      <!-- Accepted Bookings -->
      <div class="col-lg-4 col-6">
        <div class="small-box bg-success">
          <div class="inner">
            <h3 id="accepted-bookings">Loading...</h3>
            <p>Accepted Bookings</p>
          </div>
          <div class="icon"><i class="fas fa-check-circle"></i></div>
          <a href="accepted-bookings.php" class="small-box-footer">More info <i class="fas fa-arrow-circle-right"></i></a>
        </div>
      </div>

      <!-- Rejected Bookings -->
```

```
        <div class="col-lg-4 col-6">
          <div class="small-box bg-danger">
            <div class="inner">
              <h3 id="rejected-bookings">Loading...</h3>
              <p>Rejected Bookings</p>
            </div>
            <div class="icon"><i class="fas fa-times-circle"></i></div>
            <a href="rejected-bookings.php" class="small-box-footer">More info <i class="fas fa-arrow-circle-right"></i></a>
          </div>
        </div>
        <!-- All Chefs -->
        <div class="col-lg-4 col-6">
          <div class="small-box bg-secondary">
            <div class="inner">
              <h3 id="chef-count">Loading...</h3>
              <p>All Chefs</p>
            </div>
            <div class="icon"><i class="fas fa-user-tie"></i></div>
            <a href="manage-chefs.php" class="small-box-footer">More info <i class="fas fa-arrow-circle-right"></i></a>
          </div>
        </div>
      </div>
    </div>
  </section>
 </div>
 <div id="footer-container"></div>
</div>
<!-- ✅ Scripts -->
<script src="https://cdn.jsdelivr.net/npm/jquery@3.6.4/dist/jquery.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/js/bootstrap.bundle.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/admin-lte@3.2/dist/js/adminlte.min.js"></script>
<!-- ✅ Load navbar/sidebar/footer -->
```

```
<script>
  $('#navbar-container').load('navbar.php');
  $('#sidebar-container').load('sidebar.php', function () {
    setTimeout(() => {
      $('[data-widget="treeview"]').Treeview('init');
    }, 100);
  });
  $('#footer-container').load('footer.php');
</script>
<!-- ✅ Load dashboard data -->
<script>
  fetch('get-dashboard-data.php')
    .then(response => response.json())
    .then(data => {
      document.getElementById('all-bookings').textContent = data.allbookings;
      document.getElementById('new-bookings').textContent = data.newbookings;
      document.getElementById('accepted-bookings').textContent = data.acceptedbookings;
      document.getElementById('rejected-bookings').textContent = data.rejectedbookings;
      document.getElementById('chef-count').textContent = data.chefCount;
      document.getElementById('new-chef-count').textContent = data.newchefcount;
      // Clear the list if it exists
      const list = document.getElementById('new-chef-list');
      if (list) {
        list.innerHTML = '';
      }
    })
    .catch(error => {
      console.error("Error loading dashboard data:", error);
    });
</script>
</body>
</html>
```

**PHP SCRIPT:**

```
<?php
session_start();
include('config.php');
if (!isset($_SESSION['aid']) || strlen($_SESSION['aid']) == 0) {
   header('Location: admin_dashboard.html'); // or your actual login page
   exit;
}
header('Location: admin_dashboard.html');
exit;
?>
```

# REFERENCES


1. Swiggy. (2024). *How Swiggy works*.
2. Zomato. (2024). *Zomato for home chefs*.
3. Sharma, P., & Rani, M. (2023). Connecting local food talent with digital platforms: A case study on home-based food delivery. *International Journal of Modern Computing*, 11(4), 213-220.
4. Kumar, A., & Singh, R. (2022). Role of location-based services in food delivery apps. *Journal of Mobile Applications and Services*, 9(1), 56-68.
5. Bhatia, K., & Kaur, G. (2021). Impact of food tech startups on the gig economy. *Indian Journal of E-Commerce and Digital Innovation*, 6(3), 99-108.
6. Google. (2023). *Firebase for web apps: Realtime database and authentication*.
7. Sharma, N. (2022). User-centric design for food ordering interfaces. *UX Review Journal*, 5(2), 45-53.
8. Kapoor, S., & Jain, V. (2020). Secure and scalable PHP & MySQL based web applications. *International Conference on Web Engineering*, 223-231.
9. Figma. (2024). *Collaborative UI/UX design*.
10. World Economic Forum. (2023). *Digital inclusion and the future of food delivery platforms*.
11. Khan, A., & Patel, S. (2021). Enhancing user trust in food delivery platforms through transparency and hygiene ratings. *Journal of Digital Consumer Behavior*, 8(1), 89-97.
12. W3C. (2023). *Web accessibility guidelines for inclusive interfaces*.
13. Bootstrap. (2024). *Responsive web design made easy*.
14. Mozilla Developer Network. (2024). *HTML, CSS, and JavaScript documentation*.

# SDG CERTIFICATION

**SRM VALLIAMMAI ENGINEERING COLLEGE**
(An Autonomous Institution)
ESTD. 1999 - Accredited by NBA - Approved by AICTE
'A' Grade Accreditation by NAAC
ISO 9001:2015 Certified - Affiliated to Anna University Chennai

## CENTRE OF EXCELLENCE IN SUSTAINABILITY

### SDG CERTIFICATE

This is to certify that the project work titled "**HOMECHEF WEB APPLICATION INSTANT ACESS TO HOME COOKED MEALS**" has been successfully completed by **S A SUNIL ALDO (142224621057)**, of Master of Computer Applications, during the academic year 2024-25. This project aligns with the United Nations Sustainable Development Goals and mapped to the following Sustainable Development Goal(s) (SDGs):

| SDG Number | Name | Brief Justification |
|---|---|---|
| SDG 3 | Good Health and Well-Being | By promoting access to hygienic, home-cooked meals for better health. |


PROJECT SUPERVISOR CES COORDINATOR HOD/MCA