HOMECHEF WEB APPLICATION
INSTANT ACCESS TO HOME COOKED MEALS

Submitted by

S A SUNIL ALDO

142224621057

of

VALLIAMMAI ENGINEERING COLLEGE

A MINI PROJECT REPORT

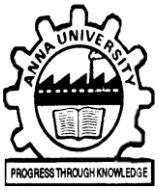
Submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

In partial fulfillment for the award of the degree

of

MASTER OF COMPUTER APPLICATIONS



ANNA UNIVERSITY, CHENNAI

MAY, 2025

BONAFIDE CERTIFICATE

Certified that this mini project report "**HOMECHEF – INSTANT ACCESS TO HOME COOKED MEALS**" is the bonafide work for "**S A SUNIL ALDO [142224621057]**" who carried out the project under my supervision.

SIGNATURE	**SIGNATURE**
Mr.R.Maniraj M.Tech.,Ph.D.	**Dr.S.Parthasarathy M.Tech.,Ph.D.**
ASSISTANT PROFESSOR	**HEAD OF THE DEPARTMENT**
Department of AI & DS,	Department of Computer Application,
SRM Valliammai Engineering College,	SRM Valliammai Engineering College,
Kattankulathur-603 203.	Kattankulathur-603 203.

Submitted to the University Examination held on ..... at Valliammai Engineering College, Kattankualthur.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

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.

ACKNOWLEDGEMENT

First and foremost, we would like to extend our heartfelt respect and gratitude to the Management, Director **Dr. B. Chidhambara Rajan, M.E., Ph.D.,** Principal **Dr. M. Murugan, M.E., Ph.D.,** and Vice Principal **Dr. S. Visalakshi M.E., Ph.D.,** who helped us in our endeavors.

We also extend our heartfelt respect to our beloved Head of the Department **Dr. S. Parthasarathy M.Tech., Ph.D., Professor** for offering his sincere support throughout the final project work.

We thank our Project Coordinator **Dr. K . Ponmozhi, M.Tech., Ph.D., Associate Professor** for her consistent guidance and encouragement throughout the progress of the final project.

We thank our Final Project Guide **Mr. R. Maniraj, M.Tech.,Ph.D Assistant Professor** for her valuable guidance, support, and active interest for the successful implementation of the final project.

We would also thank all the **Teaching and Non-Teaching staff members** of our department for their constant support and encouragement throughout the course of this final project work.

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
3. DataBase : Mysql
4. Server & Host : Apache, Xamp

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.

4. Payment Gateway Integration:

- Secure online payments through platforms like Razorpay or UPI.
- Transaction history is stored for both cooks and users.

5. Location-based Services:

- Integration with Google Maps API for identifying cooks within a specific area or service radius.

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.

2.1 USE CASE MODEL

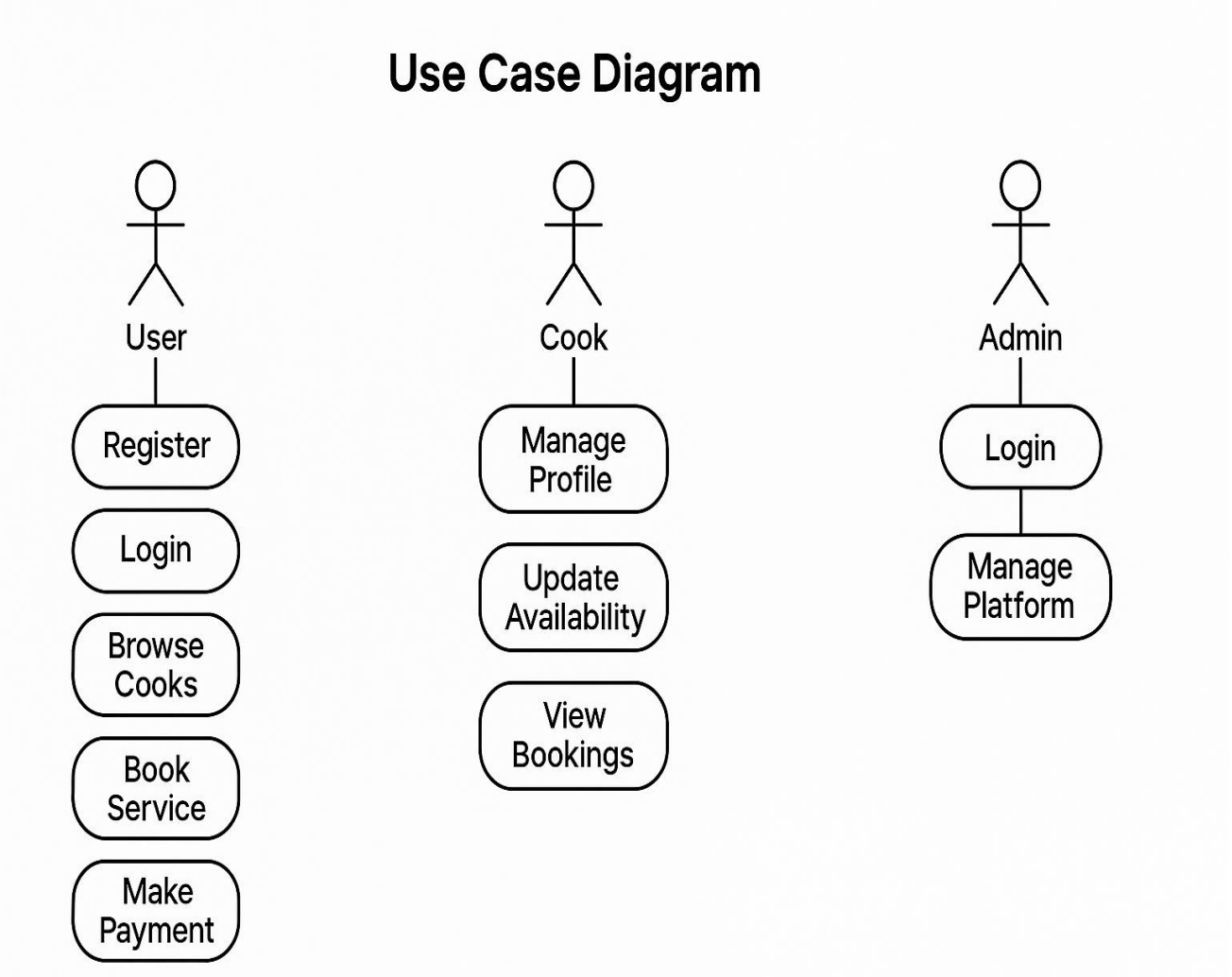


Figure 2.3 (use case model)

2.2 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 **response 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

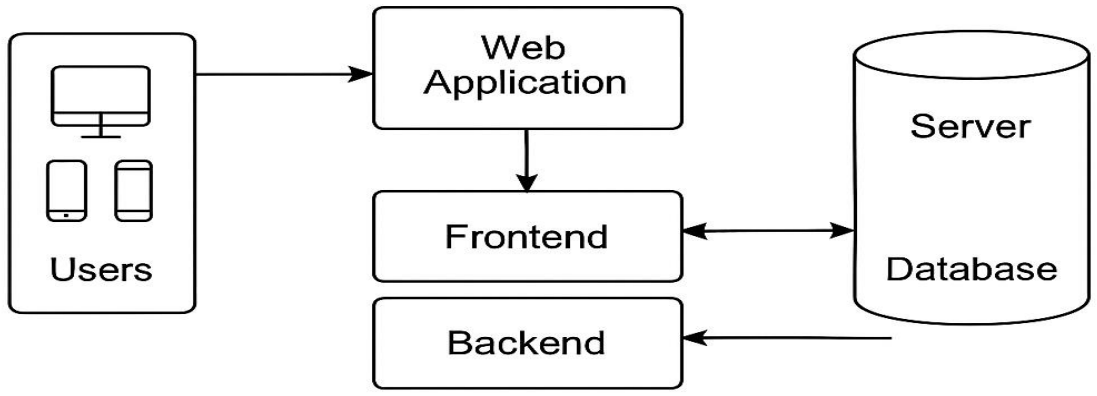


Figure 3.1 (Architectural diagram)

1. User (Food Seeker / Customer)

- The person interacts with the web or mobile interface to search, view, and book a home chef.
- Users include students, working professionals, families, or event organizers.
- Interface is user-friendly, allowing easy input of location and service requirements.

2. Service Request Input

Users interact with the platform by providing inputs such as:

- **Location** (e.g., "Search chefs near Koramangala")
- **Type of service** (e.g., "Book for home visit" or "Order a meal")
- **Date and time preferences**
- The input is taken via form fields, dropdowns, or filters.

3. Request Processing & Filtering Module

- This module processes user inputs.
- Filters available chefs based on:
- Location proximity (using GPS or address-based filtering)
- Availability calendar
- Service type offered (in-home cooking, takeaway, event catering)
- Ensures users only see relevant and available options.

4. Location & Service Matching Engine

- Uses the user's input location to match with chefs within a service radius.
- Integrates with **Google Maps API** to determine:
- Distance between user and chef
- Travel feasibility for in-home services
- Prioritizes chefs who are active and have good ratings.

5. Chef Profile and Menu Module

- Displays selected chef profiles including:
- Profile photo, bio, experience
- Available dishes with prices
- Service options and time slots
- Allows users to add dishes to a cart or proceed with a service booking.

6. Booking & Confirmation System

- Users proceed to confirm:
- Booking time, location, and dishes
- Service type (home service or pickup)
- Upon confirmation, both user and chef receive notifications.
- Booking status updates in real-time (Pending, Confirmed, Delivered).

7. Payment & Transaction Module

- Supports secure online payments through:
- UPI, Debit/Credit Cards, Wallets

  Ensures:

- Invoice generation
- Chef and admin receive their respective commissions
- Refunds/Disputes handled through admin dashboard

8. Feedback & Review Loop

- After service, users can:
- Rate the chef
- Leave feedback
- This data helps other users make informed decisions and improves platform reliability.
- Cooks can also view feedback to enhance their offerings.

3.2 STRUCTURAL DESIGN

CLASS DIAGRAM

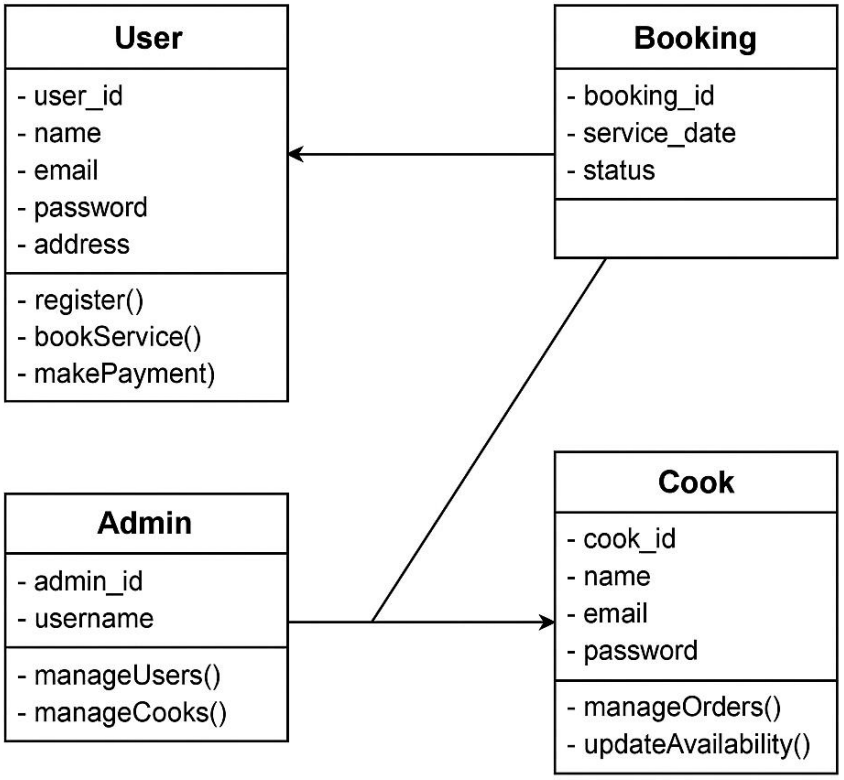


Figure 3.2 (Class diagram)

MainActivity

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.
- `tts`: TextToSpeech instance for accessible audio feedback.
- `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.

FindChefsActivity

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

Attributes:

- `locationInput`, `serviceTypeDropdown`, `chefListRecyclerView`
- `chefAdapter`: Adapter for displaying chefs in a list or grid.
- `tts`: For reading chef names or service info aloud (for accessibility).

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.

ChefProfileActivity

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.

BookingActivity

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.

MyBookingsActivity

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.

DatabaseHelper

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.

SettingsActivity

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.

OrderHistoryActivity

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.

ChefLocationService

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.

NotificationReceiver

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

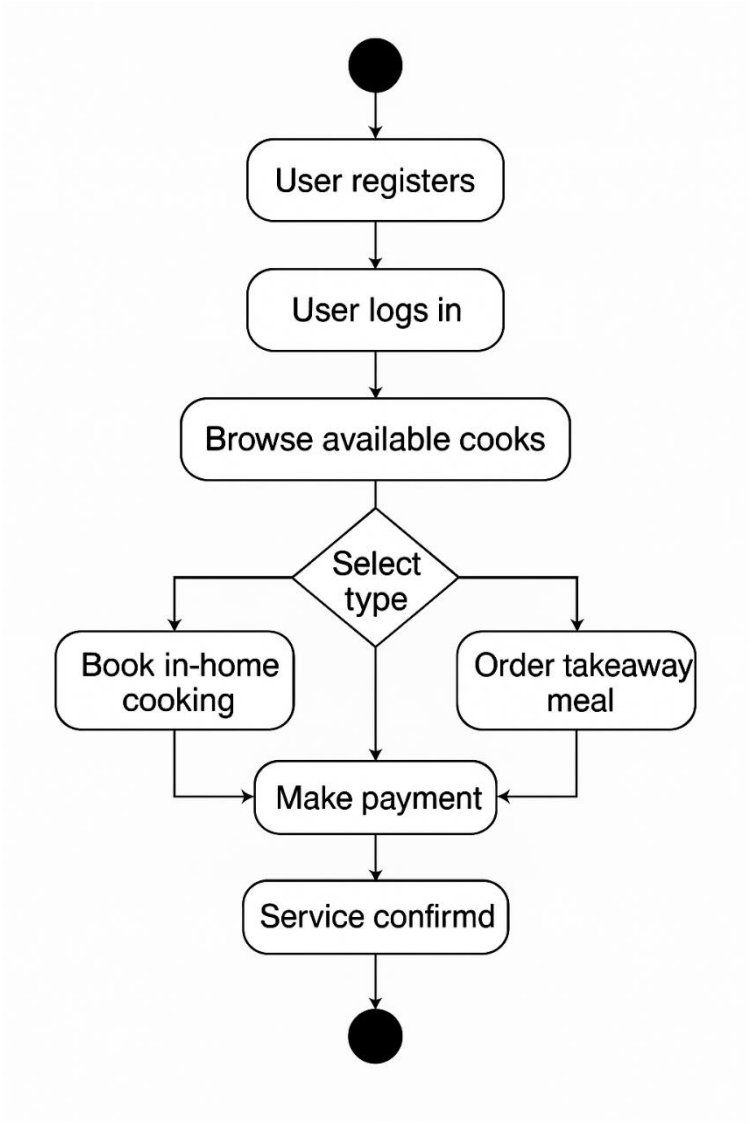


Figure 3.4 (Activity diagram)

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

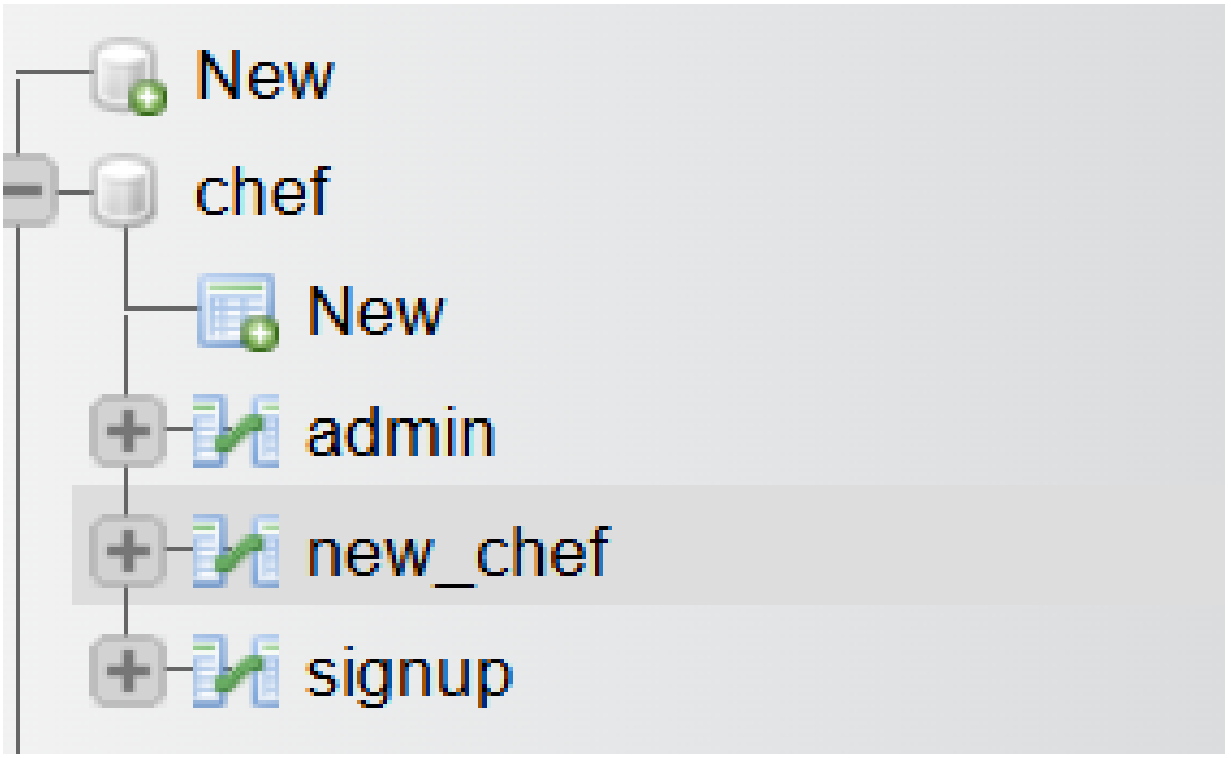
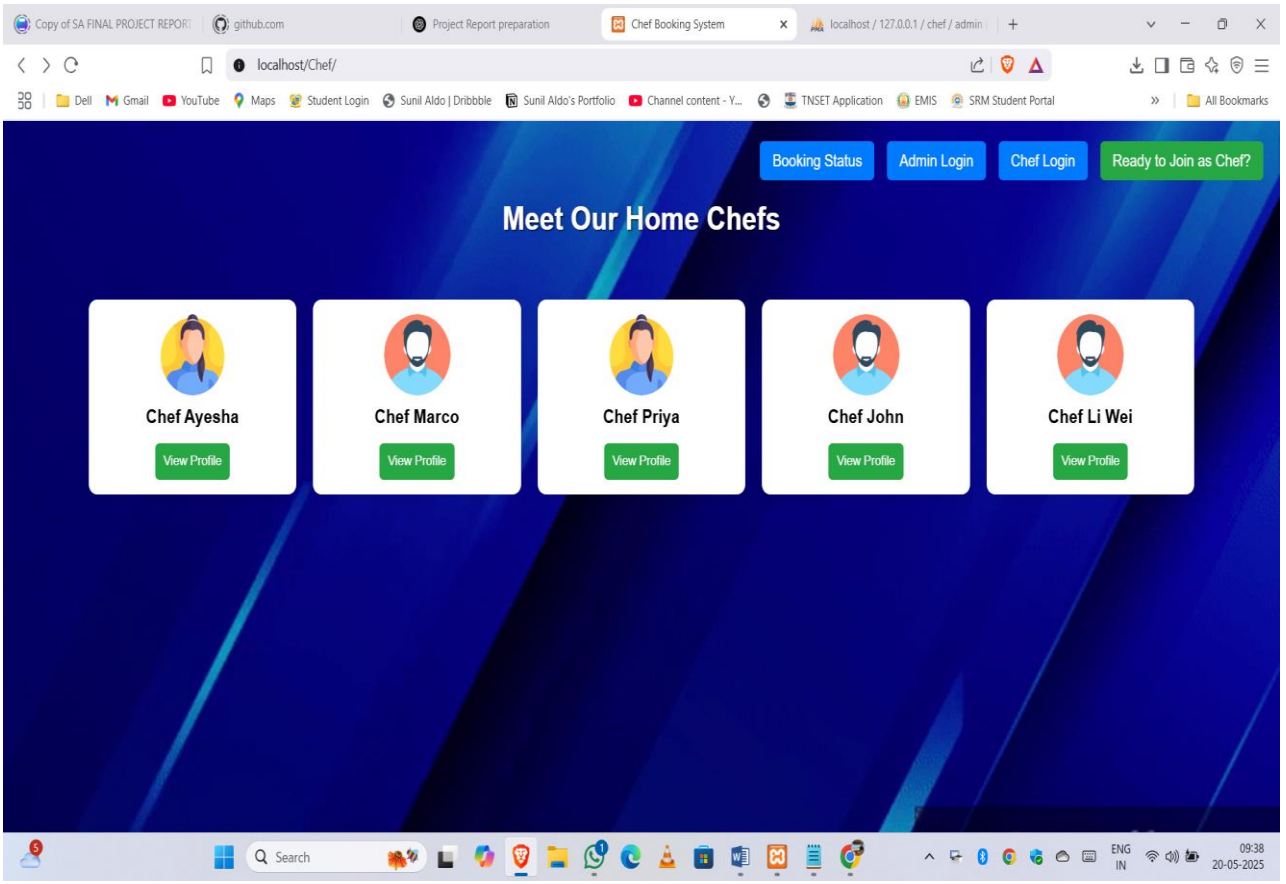
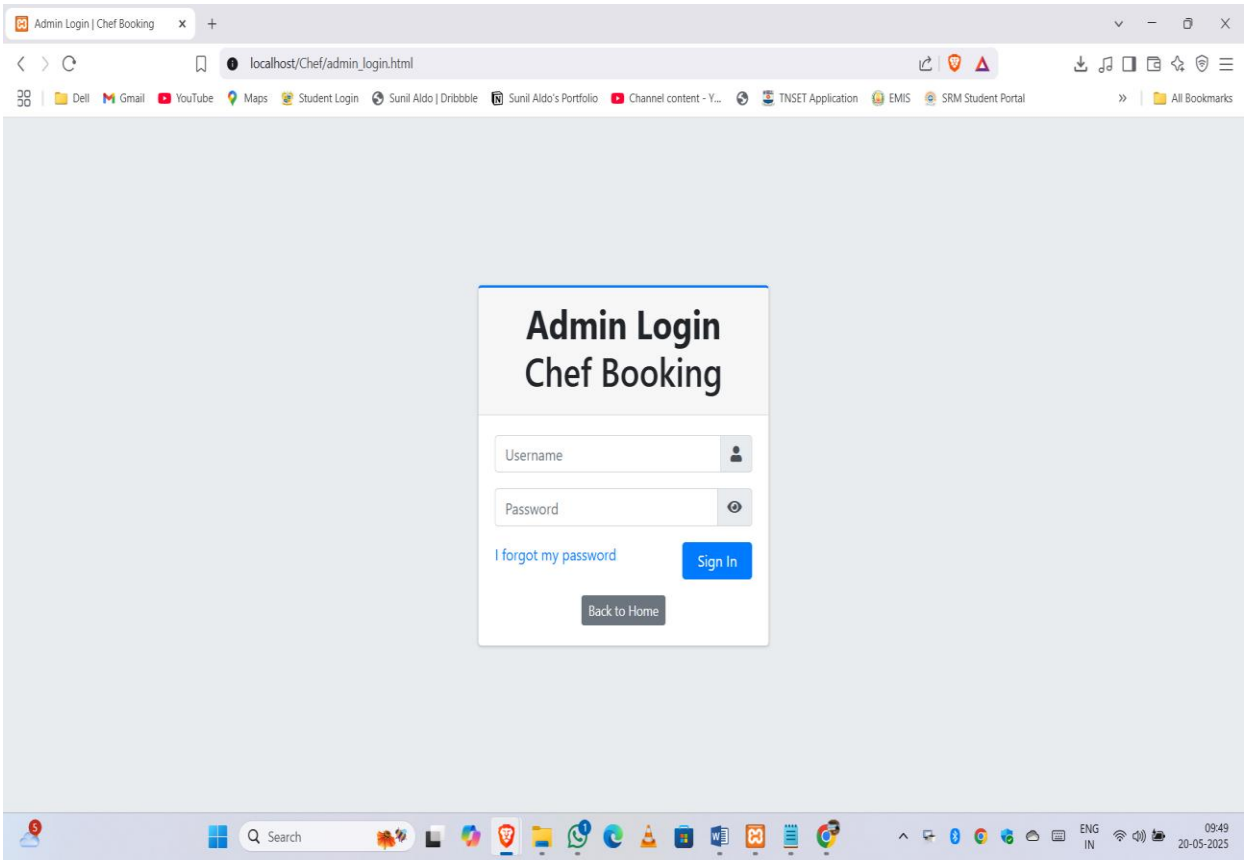


Table 3.1 (Table design)

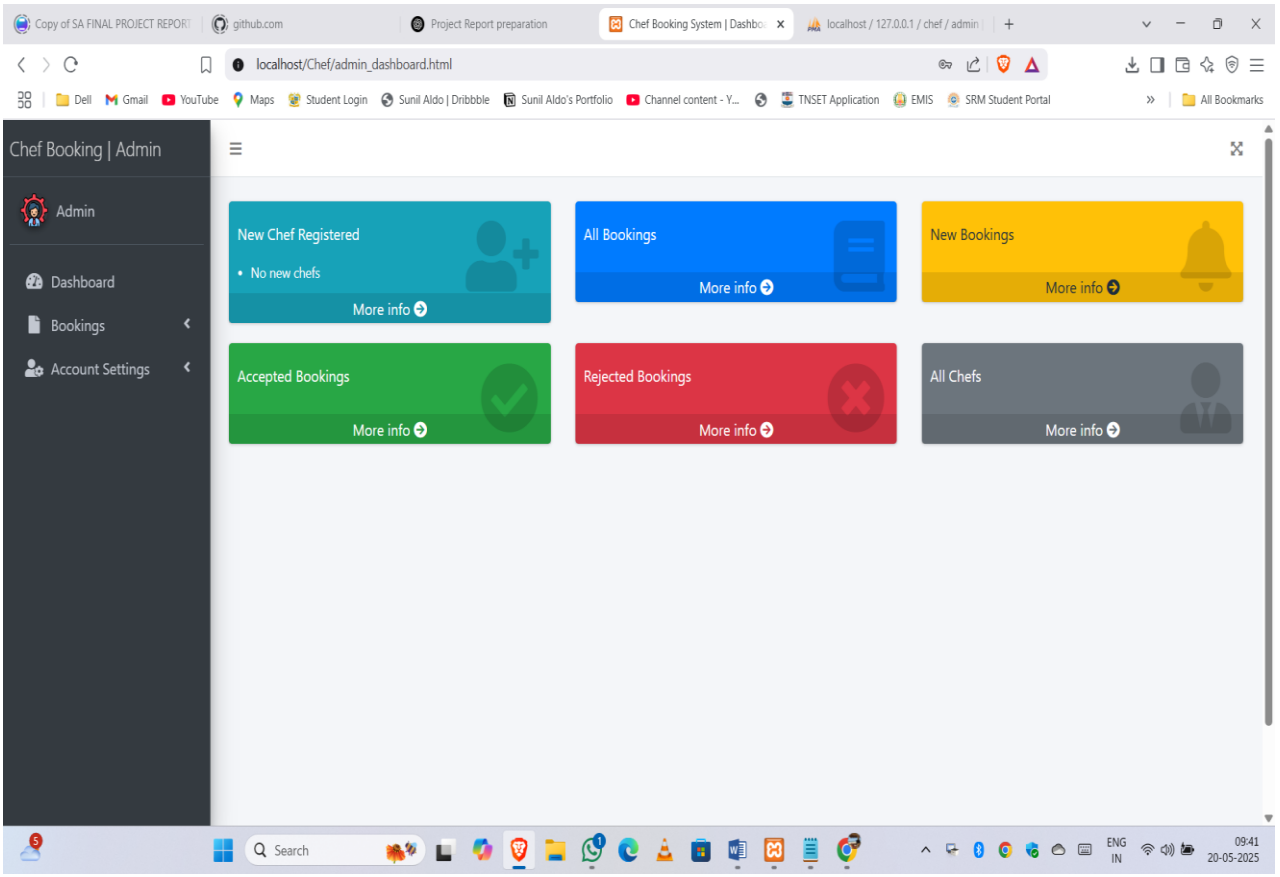
3.5 USER INTEFACE DESIGN



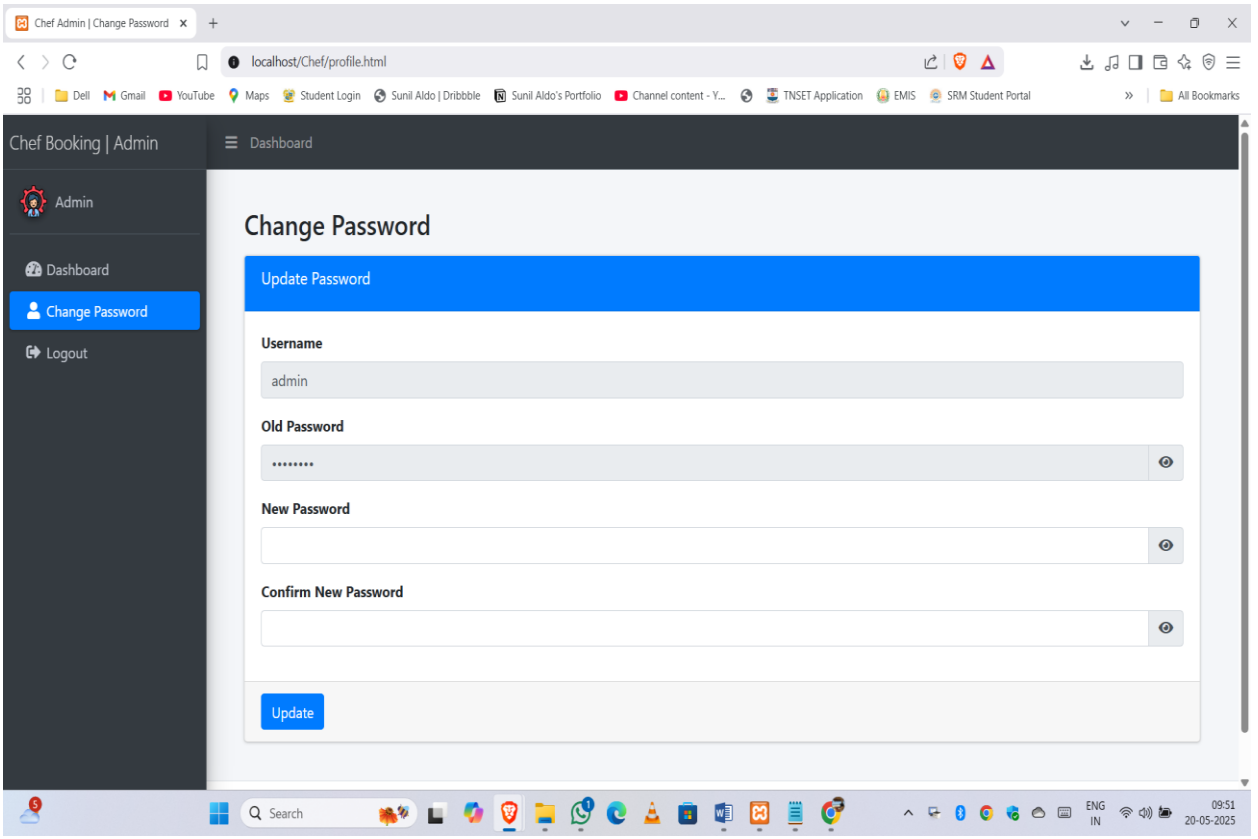
Screen 3.1 (Shows the main menu, where the user can see the chef profiles and choose.)



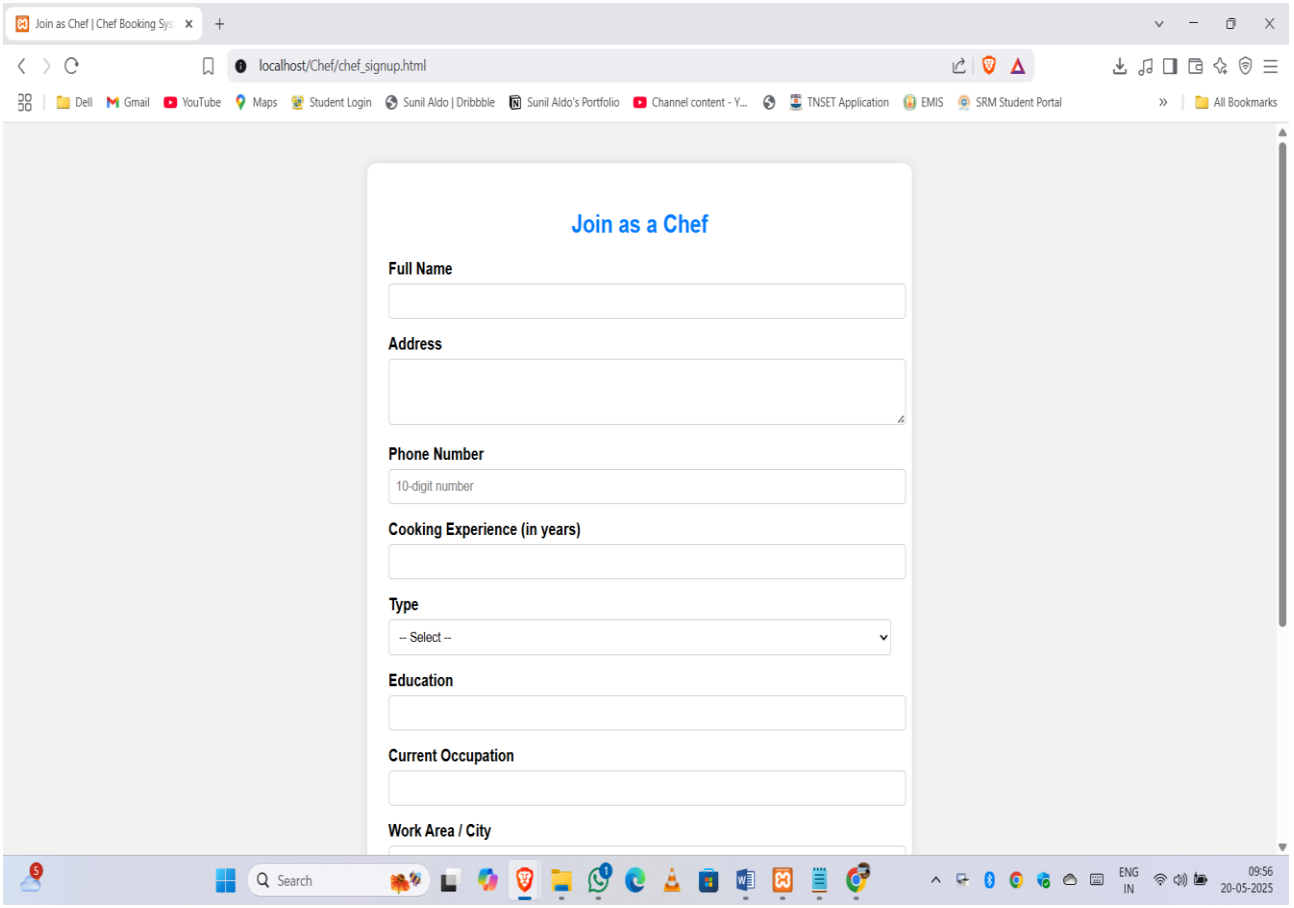
Screen 3.2 (Shows the Admin panel, where the admin can see the chef profiles and Manage it.)



Screen 3.2 (Shows the Admin dashboard, where admin can manage the cooks and users.)



Screen 3.3 (Displays the page which help admin to modify his details.)



Screen 3.3 (Displays the page which help admin to modify his detail.

3.6 DEPLOYMENT DESIGN

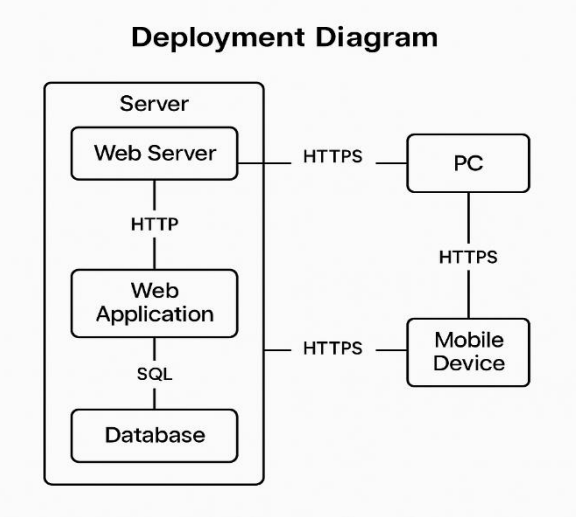


Figure 3.6 (Deployment design)

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.

• Communicates with frontend via secure API endpoints.

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

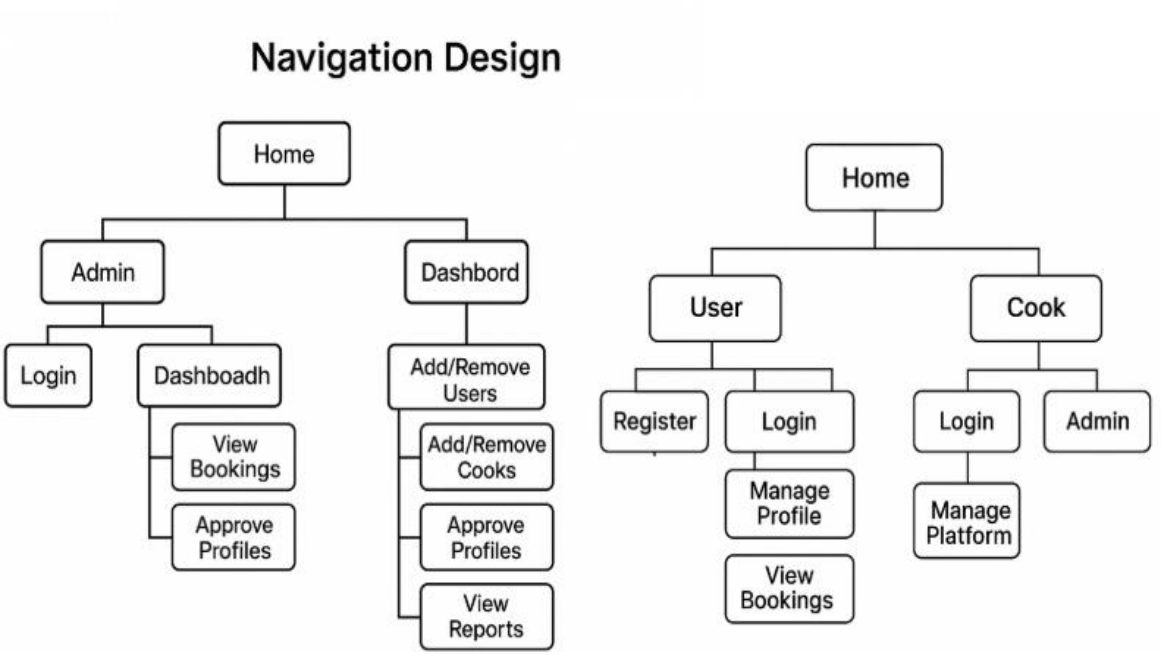


Figure 3.11 (Navigation design)

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.

3.8 CODE DESIGN

```
<!DOCTYPE html>
<html>
<head>
  <title>Chef Booking System</title>
  <style>
    body, html {
      margin: 0;
      padding: 0;
      height: 100%;
      font-family: Arial, sans-serif;
    }

    /* Background with overlay */
    .background {
      position: fixed;
      top: 0;
      left: 0;
      height: 100%;
      width: 100%;
      background-image: url('6.gif');
      background-size: cover;
      background-position: center;
      background-attachment: fixed;
      z-index: -2;
    }

    .overlay {
      position: fixed;
      top: 0;
```

```
  left: 0;
  height: 100%;
  width: 100%;
  background-color: rgba(0, 0, 0, 0.2); /* semi-transparent black */
  z-index: -1;
}

h1 {
  text-align: center;
  padding: 20px;
  color: #fff;
  text-shadow: 1px 1px 2px #000;
}

.container {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  gap: 20px;
  padding: 20px;
}

.card {
  background: #fff;
  border-radius: 10px;
  box-shadow: 0 2px 8px rgba(0,0,0,0.1);
  width: 220px;
  padding: 15px;
  text-align: center;
}

.avatar {
```

```
  width: 80px;
  height: 80px;
  background-size: cover;
  background-position: center;
  border-radius: 50%;
  margin: 0 auto 10px;
}

.male .avatar {
  background-image: url('https://cdn-icons-png.flaticon.com/512/236/236831.png');
}

.female .avatar {
  background-image: url('https://cdn-icons-png.flaticon.com/512/4140/4140047.png');
}

.card h3 {
  margin: 10px 0 5px;
}

.card button {
  background-color: #28a745;
  color: white;
  border: none;
  padding: 10px;
  border-radius: 5px;
  margin-top: 10px;
  cursor: pointer;
}

.card button:hover {
  background-color: #218838;
```

```
}

.top-buttons {
  position: absolute;
  top: 20px;
  right: 20px;
  display: flex;
  gap: 15px;
  flex-wrap: wrap;
}

.top-buttons a {
  padding: 10px 15px;
  background-color: #007bff;
  color: white;
  border-radius: 5px;
  text-decoration: none;
}

.top-buttons a:hover {
  background-color: #0056b3;
}

.top-buttons a.join-btn {
  background-color: #28a745;
}

.top-buttons a.join-btn:hover {
  background-color: #218838;
}

.profile {
```

```
  max-width: 600px;
  background: white;
  margin: 40px auto;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 2px 8px rgba(0,0,0,0.1);
}

.profile h2 {
  text-align: center;
  margin-bottom: 10px;
}

.profile p {
  margin: 6px 0;
}

.book-btn, .back-btn {
  display: block;
  width: 100%;
  padding: 12px;
  color: white;
  border: none;
  border-radius: 5px;
  margin-top: 10px;
  text-align: center;
  text-decoration: none;
}

.book-btn {
  background: #007bff;
}
```

```
    .book-btn:hover {
      background: #0056b3;
    }

    .back-btn {
      background: #6c757d;
    }

    .back-btn:hover {
      background: #5a6268;
    }
  </style>
</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>
```

chef_signup:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Join as Chef | Chef Booking System</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
    body {
      font-family: Arial, sans-serif;
      background: #f2f2f2;
      padding: 30px;
    }
    .form-container {
      background: #fff;
      max-width: 600px;
      margin: auto;
      padding: 25px;
      border-radius: 10px;
      box-shadow: 0 0 10px rgba(0,0,0,0.1);
    }
    h2 {
      text-align: center;
      margin-bottom: 20px;
      color: #007bff;
    }
    .form-group {
      margin-bottom: 15px;
    }
    label {
      display: block;
      font-weight: bold;
      margin-bottom: 5px;
```

```
    }
    input, textarea, select {
      width: 100%;
      padding: 8px;
      border-radius: 4px;
      border: 1px solid #ccc;
    }
    button {
      background-color: #28a745;
      color: white;
      padding: 12px;
      width: 100%;
      border: none;
      border-radius: 5px;
      font-size: 16px;
      cursor: pointer;
      margin-top: 10px;
    }
    button:hover {
      background-color: #218838;
    }
  </style>
</head>
<body>

  <div class="form-container">
    <h2>Join as a Chef</h2>
    <form method="POST" action="chef_signup.php" enctype="multipart/form-data">

      <div class="form-group">
        <label for="name">Full Name</label>
        <input type="text" name="name" id="name" required>
      </div>
```

```
    <div class="form-group">
      <label for="address">Address</label>
      <textarea name="address" id="address" rows="3" required></textarea>
    </div>

    <div class="form-group">
      <label for="phone_number">Phone Number</label>
      <input type="tel" name="phone_number" id="phone_number" required pattern="[0-9]{10}"
placeholder="10-digit number">
    </div>

    <div class="form-group">
      <label for="cooking_experience">Cooking Experience (in years)</label>
      <input type="number" name="cooking_experience" id="cooking_experience" required
min="0">
    </div>

    <div class="form-group">
      <label for="type">Type</label>
      <select name="type" id="type" required>
        <option value="">-- Select --</option>
        <option value="Veg">Veg</option>
        <option value="Non Veg">Non Veg</option>
      </select>
    </div>

    <div class="form-group">
      <label for="education">Education</label>
      <input type="text" name="education" id="education" required>
    </div>

    <div class="form-group">
      <label for="occupation">Current Occupation</label>
      <input type="text" name="occupation" id="occupation">
```

```
    </div>

    <div class="form-group">
      <label for="work_area">Work Area / City</label>
      <input type="text" name="work_area" id="work_area" required>
    </div>

    <div class="form-group">
      <label for="email_id">Email ID</label>
      <input type="email" name="email_id" id="email_id" required>
    </div>

    <div class="form-group">
      <label for="photo">Upload Photo</label>
      <input type="file" name="photo" id="photo" accept="image/*" required>
    </div>

    <button type="submit">Submit Application</button>
  </form>
 </div>
</body>
</html>
```

chef_signup:

```
<?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>";
}
?>
```

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.

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.

Unit Testing

Each module—User, Cook, and Admin—was tested independently to ensure core functionality. For example, the **User module** was tested to verify registration, login, and booking actions worked as expected. The **Cook module** was tested to ensure proper profile updates and availability status. The **Admin module** was checked for platform oversight functionalities like user/cook approval and booking monitoring. Any issues in form validation, business logic, or page navigation were detected and resolved during this stage.

Interface 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).

Black Box Testing

Black box testing was done from the user's perspective without accessing internal code. Functionalities like registration, login, searching for cooks, booking services, and making payments were executed, and the output was checked against expected results. Invalid data (e.g., wrong email format or blank fields) were entered to test system responses, ensuring the display of appropriate error or success messages.

Integrated Testing

All modules—User, Cook, and Admin—were integrated and tested together. A complete workflow was simulated, such as a user registering, booking a cook, and the cook responding to the booking. Integration between modules (e.g., booking reflected correctly in both user and cook dashboards) was validated. The payment system, admin approvals, and notification feedback were also checked for consistency and reliability.

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: `homechef_db`
- Import the provided SQL schema file to initialize tables such as `users`, `chefs`, `orders`, 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: `homechef_db`

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

CONCLUSION

The HomeChef web application is a powerful platform designed to connect home-based chefs with customers seeking fresh, homemade meals in their locality. By offering a seamless interface for customers to discover chefs, browse menus, and place orders, along with robust admin tools to manage users and listings, the platform brings efficiency and personalization to food delivery services. With features like chef registration, location-based filtering, and real-time booking management, HomeChef promotes local talent, encourages entrepreneurship, and delivers a unique culinary experience. This project bridges the gap between home kitchens and hungry customers, redefining how communities access quality homemade food online.

REFERENCE

1. Swiggy. (2024). *How Swiggy works*. Retrieved from https://www.swiggy.com
2. Zomato. (2024). *Zomato for home chefs*. Retrieved from https://www.zomato.com
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*. Retrieved from https://firebase.google.com
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*. Retrieved from https://www.figma.com
10. World Economic Forum. (2023). *Digital inclusion and the future of food delivery platforms*. Retrieved from https://www.weforum.org
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*. Retrieved from https://www.w3.org/WAI/
13. Bootstrap. (2024). *Responsive web design made easy*. Retrieved from https://getbootstrap.com
14. Mozilla Developer Network. (2024). *HTML, CSS, and JavaScript documentation*. Retrieved from https://developer.mozilla.org

APPENDICES

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	Goof Health and Well-Being	By promoting access to hygienic, home-cooked meals for better health.

CES COORDINATOR PROJECT COORDINATOR HOD/CSE