M.KUMARASAMY
COLLEGE OF ENGINEERING
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 & ISO 14001:2015 Certified Institution
Thalavapalayam, Karur - 639 113.

A Project Report
on

QUICK HIRE PLATFORM

Submitted in partial fulfillment of requirements for the award of the course of

MCB1733-QUICK HIRE PLATFORM

Under the guidance of

MRS. S. VANISRI, MCA

Assistant Professor/MCA

Submitted By

1. **LOGESHWARAN R (927624MCA024)**
2. **LOGESHWARAN B (927624MCA023)**
3. **KAUSHIK JAYAVEL P (927624MCA021)**
4. **NANDHAKUMAR S (927624MCA027)**
5. **PERIYASAMY S (927624MCA030)**
6. **SANTHOSH K (927624MCA036)**

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS
M.KUMARASAMY COLLEGE OF ENGINEERING
(Autonomous)

KARUR – 639 113

MAY 2025

M.KUMARASAMY COLLEGE OF ENGINEERING
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 & ISO 14001:2015 Certified Institution
Thalavapalayam, Karur - 639 113.

M. KUMARASAMY COLLEGE OF ENGINEERING

(Autonomous Institution affiliated to Anna University, Chennai)
KARUR – 639 113

BONAFIDE CERTIFICATE

Certified that this project report on **"QUICK HIRE PLATFORM "**is the Bonafide work of **LOGESHWARAN R(927624MCA024), LOGEESHWARAN B(927624MCA023), KAUSHIK JAYAVEL P (927624MCA021),NANDHAKUMAR S(927624MCA027),PERIYASAMY S(927624MCA030),SANTHOSH K(927624MCA036)** carried out the project work during the academic year 2024 - 2025 under my supervision.

Signature	Signature
Mrs. S. VANISRI, MCA	**Dr.S.Vanithamani,MCA,M.Phil,Ph.D**
INTERNAL GUIDE,	**HEAD OF THE DEPARTMENT,**
Department of Master of Computer Applications	Department of Master of Computer Applications
M. Kumarasamy College of, Engineering,	M. Kumarasamy College of Engineering,
Thalavapalayam, Karur -639 113.	Thalavapalayam, Karur -639 113.

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

VISION

To meet the technology and evolve innovative applications according to software industry and to promote technological advancement through knowledge dissemination

MISSION

M1: To achieve excellence in the field of computer applications.

M2: To create quality professionals to meet the emerging industrial needs**.**

M3: To inculcate ethical and professional standards among our students by providing quality education.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1: Ability to face the changing trends and career opportunities in computer application
PEO2: Exhibit their expertise using sound problem- solving skills through design and development of computer applications
PEO3: Successful professional career with integrity and societal commitments

PROGRAMME OUTCOMES

PO1: Computational Knowledge: Apply knowledge of computing fundamentals, computing specialization, mathematics, and domain knowledge appropriate for the computing specialization to the abstraction and conceptualization of computing models from defined problems and requirements
PO2: Problem Analysis: Identify, formulate, research literature, and solve complex computing problems reaching substantiated conclusions using fundamental principles of mathematics, computing sciences, and relevant domain disciplines.
PO3: Design /Development of Solutions: Design and evaluate solutions for complex computing problems, and design and evaluate systems, components, or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal, and environmental considerations.

PO4: Conduct Investigations of Complex Computing Problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5: Modern Tool Usage: Create, select, adapt and apply appropriate techniques, resources, and modern computing tools to complex computing activities, with an understanding of the limitations.
PO6: Professional Ethics: Understand and commit to professional ethics and cyber regulations, responsibilities, and norms of professional computing practices
PO7: Life-long Learning: Engage in lifelong learning independently for continual development to improve knowledge and competence as a computing professional
PO8: Project management and finance: Demonstrate knowledge and understanding of management principles and apply these to multidisciplinary software development as a team member and manage projects
PO9: Communication Efficacy: Understand and communicate effectively with the computing community and with society at large, regarding complex computing systems activities confidently and effectively by writing effective reports and design documentations by adhering to appropriate standards, make effective presentations and give / receive clear instructions

PO10: Societal and Environmental Concern: Understand responsibilities and consequences based on societal, environmental, health, safety, legal and cultural issues within local and global contexts relevant to professional computing practices
PO11: Individual and Team Work: Function effectively as an individual, as a member or leader in diverse teams in multidisciplinary environments
PO12: Innovation and Entrepreneurship: Identify a timely opportunity for entrepreneurship and use innovation to pursue and create value addition for the betterment of the individual and society at large.

PROGRAMME SPECIFIC OUTCOMES (PSO)

PSO1: Apply the theoretical and practical knowledge of computer science in formulating, modelling and developing solutions to the real-world problems.
PSO2: Analyze Design and implement the application software systems that meet the automation requirement of society and industry.
PSO3: Ability to apply knowledge of layered network Models, their protocols and technologies in building network and Internet based applications.
PSO4: Design, test, develop and maintain desktop, web, mobile and cross platform software applications using modern tools and technologies.

ABSTRACT

Quick Hire is a full-stack web application designed to streamline the process of hiring and offering services in a secure and user-friendly environment. The platform enables seamless interaction between service seekers and service providers through a role-based authentication system that categorizes users into Admin, User, and Member roles.Key features of the application include session-based login/logout, dynamic homepage personalization, secure signup and login functionality, and role-based access control, ensuring that each user accesses only the relevant features. Users can book services, submit complaints, and write reviews, while service providers (Members) can add and manage their services. Admins have full access to all pages and administrative controls.

The platform integrates Twilio API to send SMS confirmations after bookings, enhancing user communication. Additionally, features like data filtering by date/service, Excel/PDF export options, and visibility restrictions ensure better data management and administrative oversight.

Quick Hire emphasizes simplicity, security, and scalability, making it a practical solution for efficient online service hiring and management.

M.KUMARASAMY
COLLEGE OF ENGINEERING
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 & ISO 14001:2015 Certified Institution
Thalavapalayam, Karur - 639 113.

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	POs MAPPED	PSOs MAPPED
Quick Hire is a full-stack web application designed to streamline the process of hiring and offering services in a secure and user-friendly environment. The platform enables seamless interaction between service seekers and service providers through a role-based authentication system that categorizes users into Admin, User, and Member roles. Key features of the application include session-based login/logout, dynamic homepage personalization, secure signup and login functionality, and role-based access control, ensuring that each user accesses only the relevant features. Users can book services, submit complaints, and write reviews, while service providers (Members) can add and manage their services. Admins have full access to all pages and administrative controls. The platform integrates Twilio API to send SMS confirmations after bookings, enhancing user communication. Additionally, features like data filtering by date/service, Excel/PDF export options, and visibility restrictions ensure better data management and administrative oversight. Quick Hire emphasizes simplicity, security, and scalability, making it a practical solution for efficient online service hiring and management.	PO1, PO2, PO3, PO4, PO5, PO8, PO10, PO11, PO12,	PSO1, PSO2, PSO3, PSO4

Note: 1- Low, 2-Medium, 3- High

M.KUMARASAMY COLLEGE OF ENGINEERING
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 & ISO 14001:2015 Certified Institution
Thalavapalayam, Karur - 639 113.

INTERNAL GUIDE **HEAD OF THE DEPARTMENT**

TABLE OF CONTENTS

CH No.	TITLE	PAGE No.
1	**Introduction**	
	1.1Introduction	10
	1.2 Objectives	10
	1.3 Existing System	11
	1.4 Proposed System	12
	1.5 System Requirements	13
2	**Project Methodology**	
	2.1 MongoDB Data Management	14
	2.2 Express.js and MongoDB Integration	14
	2.3 Authentication and Security	14
	2.4 Fast2SMS Integration	15
	2.5 Booking Workflow	15
	2.6 Block Diagram	16
3	**Module Description**	
	3.1 Authentication Module	17
	3.2 Admin Module	17
	3.3 User Module	17
	3.4 Member Module	18

	3.5 Booking Module	18
	3.6 Compliant & Review Module	18
	3.7 Search and Filter Module	19
	3.8 Export & Reporting Module	19
4	**Sample Source Code**	20
5	**Screenshots**	31
6	**Conclusion**	42
7	**Future Enhancements**	43
8	**References**	44

CHAPTER 1
INTRODUCTION

1.1 Introduction

In today's fast-paced digital world, the demand for quick and reliable service hiring platforms is growing rapidly. Traditional methods of finding and hiring services are often time-consuming, inefficient, and lack transparency. **Quick Hire** is a modern, web-based solution designed to bridge the gap between service providers and service seekers by offering a centralized, user-friendly platform.

Built using the MERN (MongoDB, Express.js, Node.js, and vanilla JavaScript) stack, Quick Hire offers a secure and role-based environment where users can book services, and service providers can showcase and manage their offerings. The application supports session-based authentication, ensuring that user data and interactions remain secure throughout their session. With features such as complaint and review systems, admin controls, booking management, and real-time SMS notifications using Fast2sms, Quick Hire delivers a complete service marketplace experience.

1.2 Objectives

The primary objective of the **Quick Hire** project is to streamline and automate the process of service hiring and management through a secure, role-based web application. It enables users to book services, providers to manage listings, and administrators to oversee the entire system with ease. The platform ensures session-based authentication, real-time SMS confirmations via Twilio, and structured user roles to enhance security and user experience. It aims to improve service accessibility, ensure data accuracy, and simplify interaction between users and service providers. With features like complaint handling, review management, and data export capabilities, Quick Hire delivers an efficient, transparent, and user-friendly environment for managing service-related activities-inreal-time.

1.3 Existing System

Currently, many service hiring processes are handled through informal methods such as phone calls, word-of-mouth referrals, or basic directory listings. In some cases, outdated web applications or social media platforms are used, which lack integration, security, and scalability. These approaches often require manual coordination between service providers and users, resulting in communication gaps and inefficiencies. Additionally, most systems lack role-based access, real-time updates, and automated confirmation mechanisms, making the process tedious and unreliable.

Disadvantages:

1. **No Role-Based Access Control:** Traditional systems often do not differentiate between users, providers, and admins, leading to security and permission issues.
2. **Lack of Real-Time Interaction:** Bookings and service updates are not synchronized in real-time, causing confusion and delays.
3. **Manual Communication:** Confirmation of bookings and updates are often handled manually, increasing the risk of missed or delayed information.
4. **Poor Data Management:** User and service data are stored in decentralized formats, making it hard to retrieve or update efficiently.
5. **No Automated Notifications:** Existing systems lack SMS/email integration for alerts and confirmations.
6. **Limited Administrative Control:** Admins have minimal access to analytics, user tracking, or control over complaints and reviews.

1.4 Proposed System

The Quick Hire Project introduces a role-based service hiring platform using Express.js, MongoDB, and Vanilla JavaScript, streamlining the connection between service providers and customers. The system is divided into three main portals: Admin Portal, User Portal, and Member Portal. It enables dynamic session-based login, secure data handling, and real-time interaction. MongoDB's flexible schema design allows scalable storage of service listings, bookings, complaints, and user data, while Express ensures smooth backend logic and route protection. The system enhances usability by offering personalized dashboards, secure login/logout functionality, and efficient service management.

Advantages

1. **Role-Based Access Control**: Each user role—Admin, User, or Member—gets a tailored interface with dedicated permissions, ensuring secure and organized functionality.
2. **Session-Based Authentication**: Secure login mechanisms maintain session state, preventing unauthorized access and improving user experience.
3. **Dynamic Service Management**: Members can add services, users can book services, and admins can manage all activities centrally.
4. **Centralized Booking Dashboard**: Admins can view all bookings, apply filters, and export data to Excel/PDF for analysis or records.
5. **Real-Time Notifications**: Users receive instant SMS confirmations (via Twilio) after successful bookings, enhancing communication.
6. **Complaint & Review Modules**: Integrated feedback and complaint system allows users to raise concerns and provide reviews, promoting transparency and service quality.

1.5 System Requirements

Hardware Specifications:

- **Processor**: Intel Core i3 or equivalent, with a speed of 2.0 GHz or higher
- **RAM**: 4 GB minimum (8 GB recommended for development)
- **Hard Disk**: 100 GB or more (SSD preferred for faster performance)
- **Monitor**: Standard color monitor (1024x768 resolution or higher)
- **Keyboard**: Standard USB or wireless keyboard
- **Mouse**: Standard USB or wireless mouse
- **Motherboard**: Intel-compatible motherboard
- **Network**: Internet connection for accessing Firebase, npm packages, and SMS API

Software Specifications :

- **Operating System**: Windows 10 / Linux / macOS
- **Backend Platform**: Node.js with Express.js
- **Frontend Technologies**: HTML, CSS, JavaScript (Vanilla JS)
- **Database**: MongoDB (NoSQL)
- **Runtime Environment**: Node.js (v16 or above)
- **Code Editor**: Visual Studio Code or any modern IDE
- **Package Manager**: npm (Node Package Manager)
- **Version Control**: Git
- **API Services**: Twilio /Fast2SMS (for SMS notifications)
- **Browser**: Google Chrome / Mozilla Firefox (latest version)

CHAPTER 2
PROJECT METHODOLOGY

2.1 MongoDB Data Management

For the Quick Hire project, MongoDB—a NoSQL database—is utilized to efficiently manage dynamic user, service, booking, complaint, and review data. MongoDB's flexible schema and document-based structure allow the application to store a wide variety of data types, supporting rapid development and scalability. Data such as user profiles, booking records, service details, and feedback are stored in structured collections. This setup allows quick read/write operations, which ensures a responsive experience for users and service providers. With built-in indexing and query optimization, MongoDB facilitates fast data retrieval, making it ideal for real-time service platforms like Quick Hire.

2.2 Express.js and MongoDB Integration

The backend of the Quick Hire platform is built using Express.js, which seamlessly integrates with MongoDB via the Mongoose ODM (Object Data Modeling) library. This integration allows efficient communication between the server and database. All data operations—such as user authentication, service bookings, and complaint logging—are handled through Express route handlers that interact with MongoDB. This setup supports modular, scalable backend development and enables quick deployment of new features. The integration ensures real-time updates, allowing Admins and Users to manage bookings and services effectively.

2.3 Authentication and Security

Quick Hire implements session-based authentication using the express-session middleware, ensuring that only authenticated users can access protected pages. User roles—Admin, User, and Member—are managed with strict access control to prevent unauthorized actions. Passwords are securely hashed before being stored in MongoDB, enhancing security. The application also integrates Fast2SMS API to send booking confirmations via SMS, ensuring both convenience and verification. All sensitive operations are protected by middleware functions that validate sessions and user roles, creating a secure and trustworthy environment for all platform users.

2.4 Fast2SMS Integration

Quick Hire integrates the Fast2SMS API to enhance user communication through instant SMS notifications. When a user successfully books a service, the backend triggers a Fast2SMS API call that sends a confirmation message to the user's registered mobile number. This real-time alert system increases user trust and ensures transparency in the booking process. The Fast2SMS integration is handled through HTTP POST requests from the Node.js backend, utilizing secure API keys to authenticate requests. This feature not only improves the user experience but also serves as an additional layer of verification for both users and service providers. It can be further expanded to include SMS alerts for remind

2.5 Booking Workflow

The booking workflow in Quick Hire is designed to be intuitive, efficient, and secure, catering to both users and service providers. The process begins with an authenticated user browsing available services and selecting the desired one. The workflow includes form submission with relevant details such as service type, preferred date/time, and contact information.

Once submitted, the backend validates the input, stores the booking data in MongoDB, and assigns it to the selected service provider. A confirmation message is immediately sent to the user via the integrated Fast2SMS API.

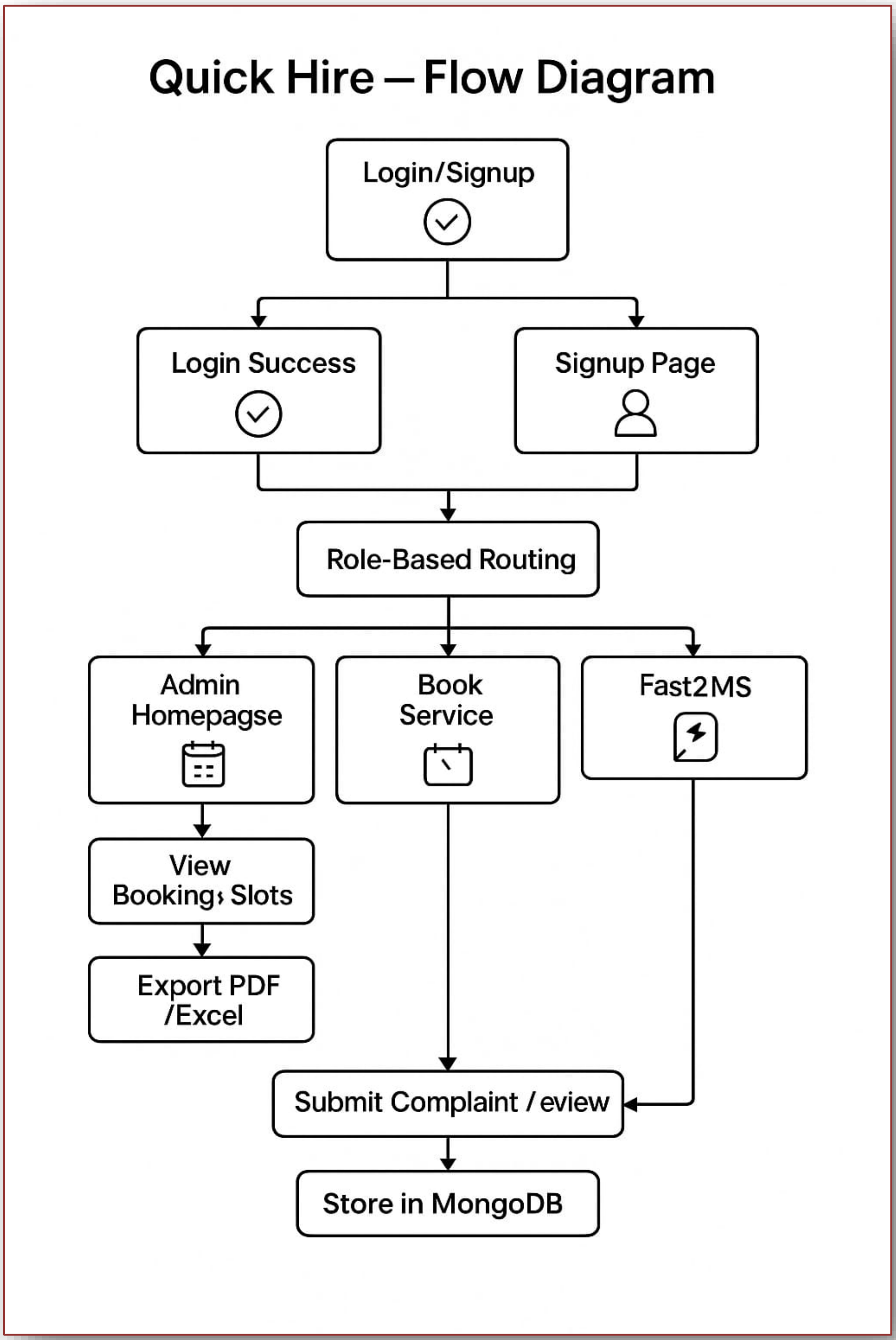
The system ensures that duplicate or conflicting bookings are avoided through backend validation. Admins and service providers can view all bookings through a role-specific dashboard, enabling efficient service management and follow-up.

Workflow Steps:

1. **Login/Signup** – User logs in to access the booking functionality.
2. **Service Selection** – User browses and selects a service.
3. **Form Submission** – User fills out the booking form with service preferences.
4. **Data Validation** – Backend validates the input and checks for conflicts.
5. **Database Update** – Booking details are stored in MongoDB.
6. **SMS Notification** – Confirmation SMS is sent using Fast2SMS.
7. **Admin/Provider Access** – Admins and relevant members can view and manage bookings.

This streamlined process ensures that all stakeholders—users, service providers, and admins—can operate within a unified, responsive, and automated service booking system.

2.6 Block Diagram



CHAPTER 3
MODULES

3.1 AUTHENTICATION MODEL:

- User login and signup with session-based authentication.
- Role-based login (Admin, User, Member).
- Redirection to appropriate dashboards based on roles.
- Secure logout and session timeout handling.

3.2 ADMIN MODULE:

- Dashboard for viewing all users, members, and services.
- Manage (add/edit/delete) service categories.
- View and export booking data (Excel/PDF).
- Filter bookings by date, service, or user.
- View and manage complaints and reviews.
- Access control and monitoring of user activities.

3.3 USER MODULE:

- View available services posted by members.
- Book a service with confirmation via Twilio SMS.
- View booking history and status.
- Submit reviews and complaints.
- Update profile and contact information.

3.4 Member Module:

- Add new services with details (title, category, availability, etc.).
- View and manage services posted by the member.
- View booking requests related to their services.
- Respond to user feedback (optional).
- Edit or delete posted services.

3.5 Booking Module:

- Handles service booking by users.
- Stores booking details (service, user, time, status).
- Sends confirmation messages via Twilio.
- Admin can monitor all bookings.
- Members can view bookings related to their services.

3.6 Complaint & Review Module:

- Users can submit complaints regarding services or members.
- Users can write reviews and rate services.
- Admin can view and resolve complaints.
- Members can view reviews for their services.

3.7 Search and Filter Module:

- Allows users/admins to search services or bookings by keyword, date, category, or member.
- Filters help in managing large datasets efficiently.

3.8 Export & Reporting Module:

- Admin can export booking details to Excel/PDF.
- Generates summary reports for review and backup...

CHAPTER 4

Sample Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Quick Hire - Home</title>
  <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap"
rel="stylesheet">
  <style>
    body {
       margin: 0;
       background: url('./thumbnail_image.png') no-repeat center center fixed;
      background-size: cover;
      overflow: auto;
  font-family: 'Poppins', sans-serif;
  transition: background 0.3s, color 0.3s;
}


    header {
      display: flex;
      justify-content: space-between;
      align-items: center;
      padding: 20px 50px;
      position: absolute;
      top: 0;
      width: 100%;
      z-index: 10;
    }

    .logo h1 {
      margin: 0;
      color: white;
      font-size: 28px;
    }

    nav ul {
      list-style: none;
      display: flex;
      gap: 20px;
      margin: 0;
      padding: 0;
    }
    nav ul li a,
nav ul li button {
  color: white;
  text-decoration: none;
  font-weight: 500;
  background: none;
  border: none;
  cursor: pointer;
  padding: 8px 12px;
  border-radius: 4px;
  transition: background-color 0.3s ease, transform 0.15s ease;
  -webkit-tap-highlight-color: transparent; /* Removes blue tap highlight on mobile */
}

nav ul li a:hover,
nav ul li button:hover {
  background-color: rgba(255, 255, 255, 0.2); /* Soft glow */
}

nav ul li a:active,
nav ul li button:active {
  transform: scale(0.96); /* Touch/click feedback */
  background-color: rgba(255, 255, 255, 0.3);
}

    .hero {
      height: 100vh;
      display: flex;
      align-items: center;
      justify-content: center;
      text-align: center;
      color: white;
      padding: 0 20px;
    }

    .hero-content {
      max-width: 600px;
    }

    .hero h2 {
      font-size: 48px;
      margin-bottom: 20px;
    }

    .hero p {
      font-size: 20px;
      margin-bottom: 30px;
    }
    #welcomeMsg {
  color: white;
  font-weight: 600;
    }

    .btn {
  display: inline-block;
  background-color: #ff6b6b;
  color: white;
  padding: 12px 25px;
  margin: 0 10px;
  border-radius: 5px;
  text-decoration: none;
  font-weight: 600;
  transition: background-color 0.3s ease, transform 0.2s ease, box-shadow 0.3s ease;
  cursor: pointer; /* Make sure it's clear it's clickable */
}

.btn:hover {
  background-color: #ff4b4b;
  transform: scale(1.05); /* Slightly grow the button */
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2); /* Add a subtle shadow for depth */
}

.btn:active {
  transform: scale(0.98); /* Shrink slightly for click effect */
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.15); /* Reduce shadow on click */
}

    .theme-toggle {
  position: fixed;
  bottom: 20px;
  right: 20px;
  color: rgb(235, 240, 238);
  z-index: 20;
        }
    .dark-theme {
      background-color: #121212;
      color: #f0f0f0;
    }

    .dark-theme .hero {
      background-blend-mode: multiply;
      background-color: rgba(0, 0, 0, 0.6);
    }

    .dark-theme nav ul li a,
    .dark-theme .logo h1,
    .dark-theme .btn {
      color: #f0f0f0;
    }
    .logout-button {
  position: fixed;
  bottom: 20px;
  left: 20px;
  background-color: #f44336; /* Red */
  color: white;
  border: none;
  border-radius: 6px;
  padding: 10px 16px;
  font-size: 14px;
  font-weight: 600;
  cursor: pointer;
  box-shadow: 0 4px 10px rgba(0,0,0,0.15);
  transition: background-color 0.3s ease, transform 0.2s ease;
  z-index: 1000;
}
```

```
.logout-button:hover {
  background-color: #d32f2f;
  transform: scale(1.05);
}

.logout-button:active {
  transform: scale(0.98);
}
footer ul {
  display: flex;
  justify-content: center;
  list-style: none;
  padding: 0;
  margin: 0;
}

footer ul li {
  margin: 0 15px;
}

footer ul li a {
  color: white;
  text-decoration: none;
}

footer ul li a:hover {
  text-decoration: underline;
  color: #000;
}
  </style>
</head>
<body>
  <header>
    <div class="logo"><h1>Quick Hire</h1></div>
    <nav>
      <ul>
        <li><a href="/home">Home</a></li>
        <li><a href="services.html">Services</a></li>
        <li><a href="complaints.html">Complaints</a></li>
        <li><a href="contact.html">Contact Us</a></li>
        <li><a href="reviews.html">Reviews</a></li>
        <li><a href="admin.html">Admin Home</a></li>
        <li id="welcomeMsg" style="display: none;"></li>
        <li id="authLink"><a href="auth.html">Login/Signup</a></li>
        <li id="logoutLink" style="display: none;">
          <button class="logout-button" onclick="logout()">Logout</button>
        </li>
      </ul>
    </nav>
  </header>

  <section class="hero">
    <div class="hero-content">
      <h2>Your Professional Services, Delivered to Your Doorstep</h2>
      <p>Quick, reliable, and affordable services at the tap of a button.</p>
      <a href="services.html" class="btn">Explore Services</a>
```

```
   <a href="booking.html" class="btn">Book Now</a>
  </div>
 </div>
 </section>
 <footer>
  <ul>
   <li><a href="contact.html">About Us</a></li>
   <li><a href="terms.html">Terms & Conditions</a></li>
   <li><a href="unauthorized.html">Privacy Policy</a></li>
  </ul>
 </footer>
 <div style="min-width: 1200px;">
  <div class="theme-toggle">
   <input type="checkbox" id="toggle-theme">
   <label for="toggle-theme"></label>
   <span>🌞 / 🌙</span>
  </div>
 <script>
  window.onload = () => {
   fetch('/api/session')
    .then(res => res.json())
    .then(data => {
     if (data.loggedIn) {
      document.getElementById('authLink').style.display = 'none';
      document.getElementById('logoutLink').style.display = 'inline-block';
      const welcome = document.getElementById('welcomeMsg');
      welcome.style.display = 'inline-block';
      welcome.innerText = `Welcome, ${data.user.username}`;
     } else {
      window.location.href = 'auth.html';
     }
    });
  };

  function logout() {
   fetch('/api/logout').then(() => window.location.href = 'auth.html');
  }
// themes
  const toggleSwitch = document.getElementById('toggle-theme');
  const body = document.body;

  toggleSwitch.addEventListener('change', () => {
   body.classList.toggle('dark-theme');
   localStorage.setItem('theme', body.classList.contains('dark-theme') ? 'dark' : 'light');
  });

  window.addEventListener('DOMContentLoaded', () => {
   const savedTheme = localStorage.getItem('theme');
   if (savedTheme === 'dark') {
    body.classList.add('dark-theme');
    toggleSwitch.checked = true;
   }
  });     </script>
</body></html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Quick Hire - Authentication</title>
  <style>
    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      background: url('notebook.jpg') no-repeat center center/cover;
      display: flex;
      align-items: center;
      justify-content: center;
      height: 100vh;
      margin: 0;
    }

    .container {
      background: #978383;
      padding: 30px;
      border-radius: 16px;
      box-shadow: 0 8px 24px rgba(0, 0, 0, 0.1);
      width: 360px;
      transition: 0.3s ease-in-out;
    }

    .tab-buttons {
      display: flex;
      justify-content: space-between;
      margin-bottom: 25px;
    }

    .tab-buttons button {
      width: 32%;
      padding: 10px;
      border: none;
      border-radius: 8px;
      background: #e0e7ff;
      color: #374151;
      font-weight: bold;
      cursor: pointer;
      transition: background 0.3s, color 0.3s;
    }

    .tab-buttons button.active {
      background: #4f46e5;
      color: #ffffff;
    }

    .form-group {
      margin-bottom: 18px;
    }

    input[type="text"],
    input[type="password"],
    input[type="email"],
    select {
      width: 95%;
      padding: 10px;
      border: 1px solid #cbd5e1;
      border-radius: 8px;
      background: #f9fafb;
      transition: 0.2s border-color;
    }

    input:focus,
    select:focus {
      border-color: #6366f1;
      outline: none;
    }

    button.submit-btn {
      width: 100%;
      padding: 12px;
      background: #4f46e5;
      border: none;
      color: white;
      border-radius: 8px;
      cursor: pointer;
      font-weight: bold;
      transition: background 0.3s;
    }

    button.submit-btn:hover {
      background: #4338ca;
    }

    .form {
      display: none;
    }

    .form.active {
      display: block;
    }
```

```
  </style>
</head>
<body>

  <div class="container">
    <div class="tab-buttons">
      <button id="loginTab" class="active"
onclick="showForm('login')">Login</button>
      <button id="signupTab" onclick="showForm('signup')">Sign Up</button>
      <button id="adminSignupTab" onclick="showForm('adminSignup')">Admin Sign
Up</button>
    </div>

    <!-- Login Form -->
    <div id="loginForm" class="form active">
      <div class="form-group">
        <input type="text" id="loginUsername" placeholder="Username"
autocomplete="username" required />
      </div>
      <div class="form-group">
        <input type="password" id="loginPassword" placeholder="Password"
autocomplete="current-password" required />
      </div>
      <button class="submit-btn" onclick="login()">Login</button>
    </div>

    <!-- User/Member Sign-Up Form -->
    <div id="signupForm" class="form">
      <div class="form-group">
        <input type="text" id="signupUsername" placeholder="Username" required />
      </div>
      <div class="form-group">
        <input type="email" id="signupEmail" placeholder="Email" required />
      </div>
      <div class="form-group">
        <input type="password" id="signupPassword" placeholder="Password" required
/>
      </div>
      <div class="form-group">
        <select id="signupRole" required>
          <option value="user">User</option>
          <option value="member">Member</option>
        </select>
      </div>
      <button class="submit-btn" onclick="signUp()">Sign Up</button>
    </div>
    <!-- Admin Sign-Up Form -->
    <div id="adminSignupForm" class="form">
      <div class="form-group">
        <input type="text" id="adminUsername" placeholder="Username" required />
      </div>
      <div class="form-group">
        <input type="email" id="adminEmail" placeholder="Email" required />
      </div>
      <div class="form-group">
        <input type="password" id="adminPassword" placeholder="Password" required
/>
      </div>
      <div class="form-group">
        <input type="text" id="adminCode" placeholder="Secret Admin Code" required
/>
      </div>
      <button class="submit-btn" onclick="adminSignUp()">Register as Admin</button>
    </div>
  </div>

  <script>
    // Function to toggle between login, signup, and admin signup forms
    function showForm(type) {
      const forms = ['loginForm', 'signupForm', 'adminSignupForm'];
      const tabs = ['loginTab', 'signupTab', 'adminSignupTab'];

      forms.forEach(form =>
document.getElementById(form).classList.remove('active'));
      tabs.forEach(tab => document.getElementById(tab).classList.remove('active'));

      document.getElementById(`${type}Form`).classList.add('active');
      document.getElementById(`${type}Tab`).classList.add('active');
    }

    // Function to handle user login
    async function login() {
      const username = document.getElementById('loginUsername').value;
      const password = document.getElementById('loginPassword').value;

      try {
        const response = await fetch('/api/login', {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify({ username, password })
        });

        const data = await response.json();
        if (data.success) {
          alert("✅ Login successful!");
          switch (data.role) {
            case 'admin':
              window.location.href = '/admin.html';
              break;
            case 'member':
              window.location.href = '/member.html';
              break;
            case 'user':
              window.location.href = '/user.html';
              break;
            default:
              window.location.href = '/';
          }
        } else {
          alert("❌ " + data.message);
        }
      } catch (error) {
        console.error('Login error:', error);
        alert("❌ An error occurred during login.");
      }
    }

    // Function to handle user signup
    async function signUp() {
      const username = document.getElementById('signupUsername').value;
      const email = document.getElementById('signupEmail').value;
      const password = document.getElementById('signupPassword').value;
      const role = document.getElementById('signupRole').value;

      try {
        const response = await fetch('/api/signup', {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify({ username, email, password, role })
        });

        const data = await response.json();

        if (data.success) {
          alert("✅ Signup successful! Please login.");
          showForm('login');
        } else {
          alert("❌ " + data.message);
        }
```

```
   } catch (error) {
    console.error('Signup error:', error);
    alert("❌ An error occurred during signup.");
   }
  }

  // Function to handle admin signup
  async function adminSignUp() {
   const username = document.getElementById('adminUsername').value;
   const email = document.getElementById('adminEmail').value;
   const password = document.getElementById('adminPassword').value;
   const adminCode = document.getElementById('adminCode').value;

   try {
    const response = await fetch('/api/admin-register', {
     method: 'POST',
     headers: { 'Content-Type': 'application/json' },
     body: JSON.stringify({ username, email, password, adminCode })
    });

    const data = await response.json();

    if (data.success) {
     alert('✅ Admin registered successfully!');
     showForm('login');
    } else {
     alert('❌ ' + data.message);
    }
   } catch (error) {
    console.error('Admin registration error:', error);
    alert('❌ An error occurred during admin registration.');
   }
  }
 </script>
</body>
</html>
```

CHAPTER 5
SCREENSHOTS

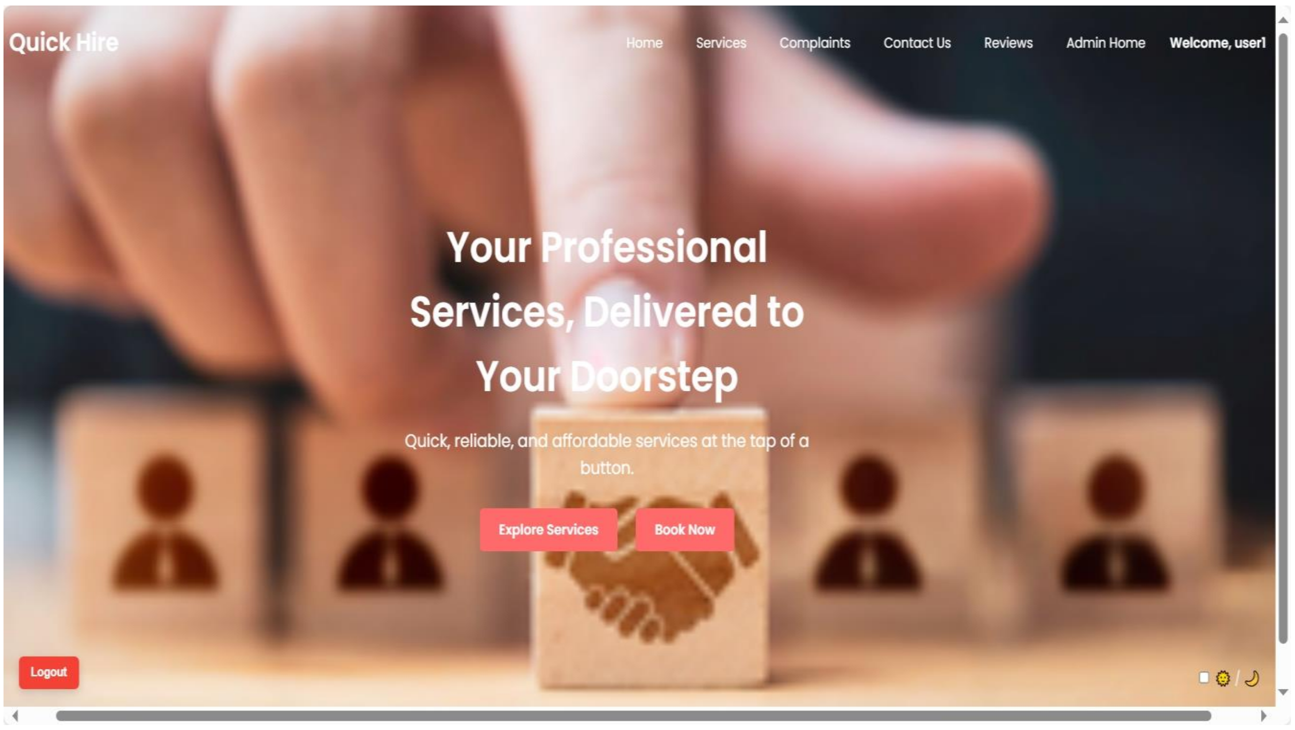

Quick Hire
Home
Services
Complaints
Contact Us
Reviews
Admin Home
Welcome, user1
Your Professional Services, Delivered to Your Doorstep
Quick, reliable, and affordable services at the tap of a button.
Explore Services
Book Now
Logout

Figure 5.1 Home Page

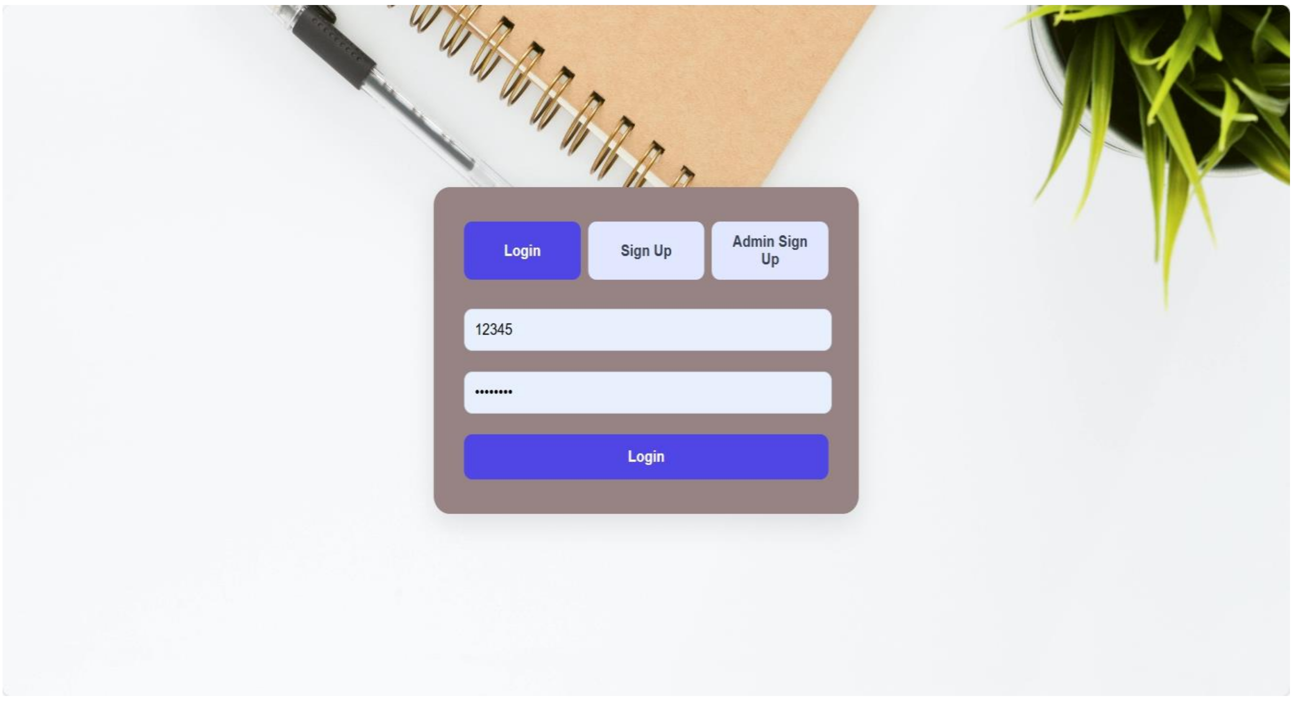


Figure 5.2 Login Page

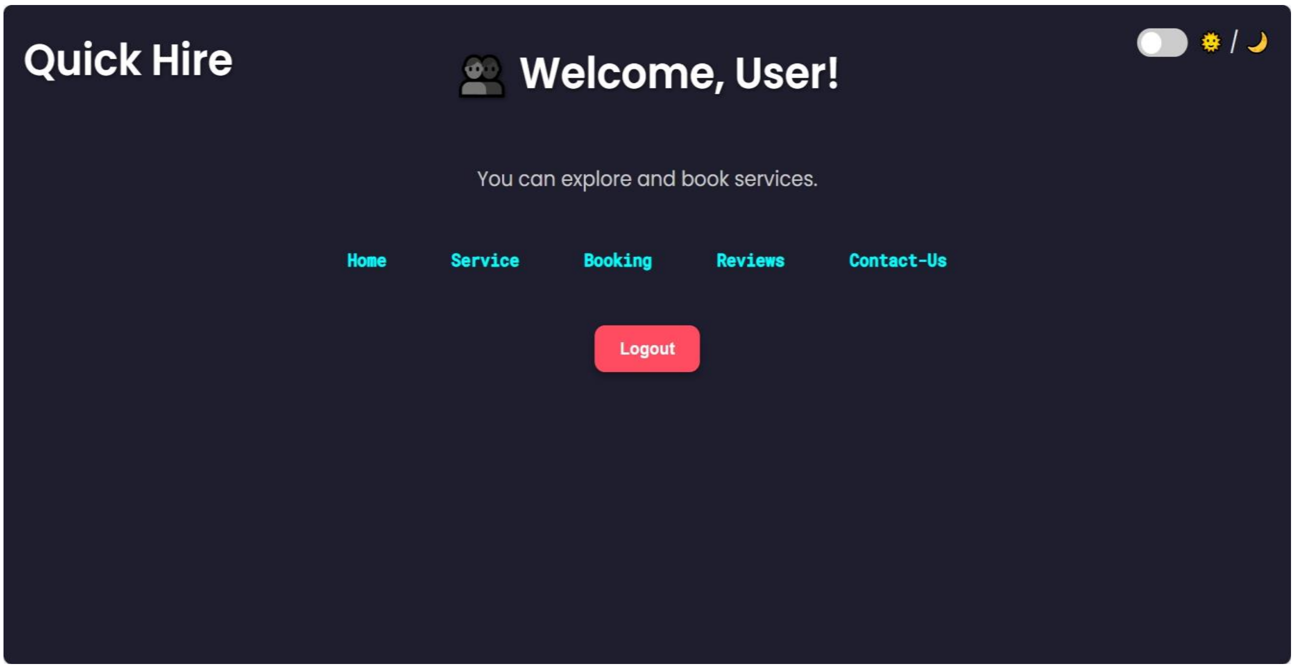
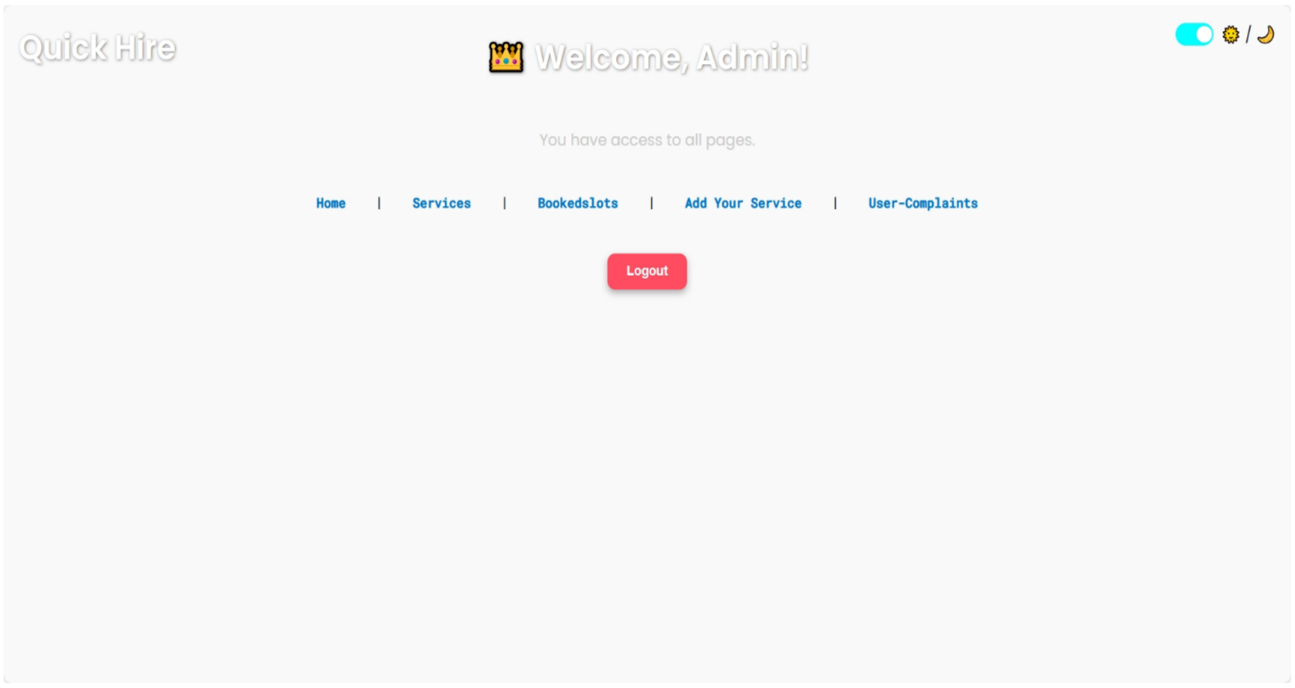
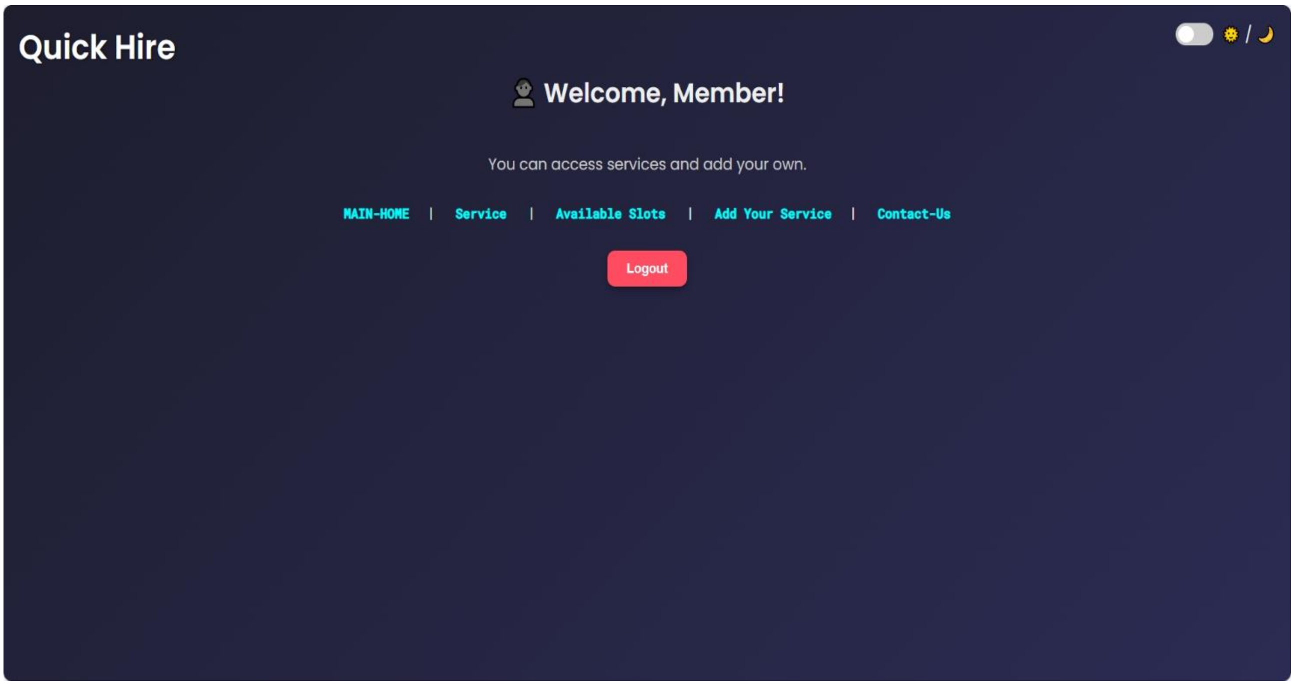


Figure 5.3 User Page


Quick Hire
👑 Welcome, Admin!
☀️ / 🌙
Home | Services | Bookedslots | Add Your Service | User-Complaints
Logout

You have access to all pages.

Figure 5.4 Admin Page

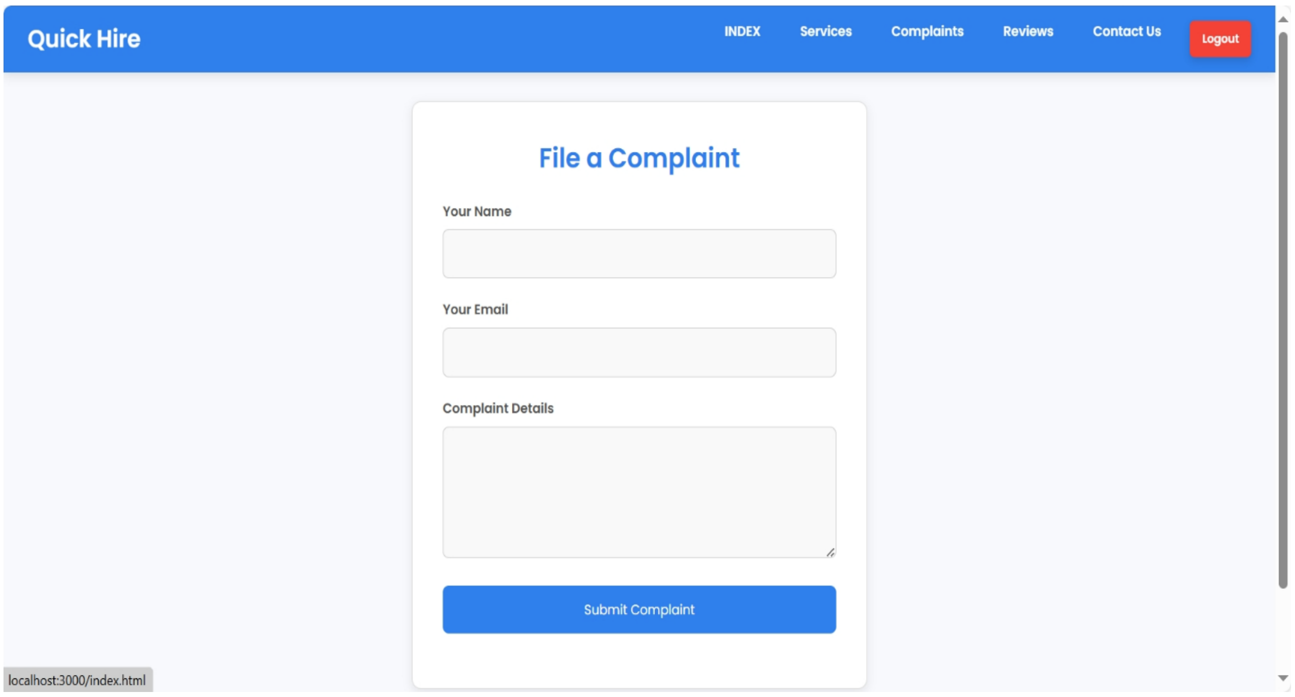

Quick Hire

Welcome, Member!

You can access services and add your own.

MAIN-HOME | Service | Available Slots | Add Your Service | Contact-Us
Logout

Figure 5.5 Member Page



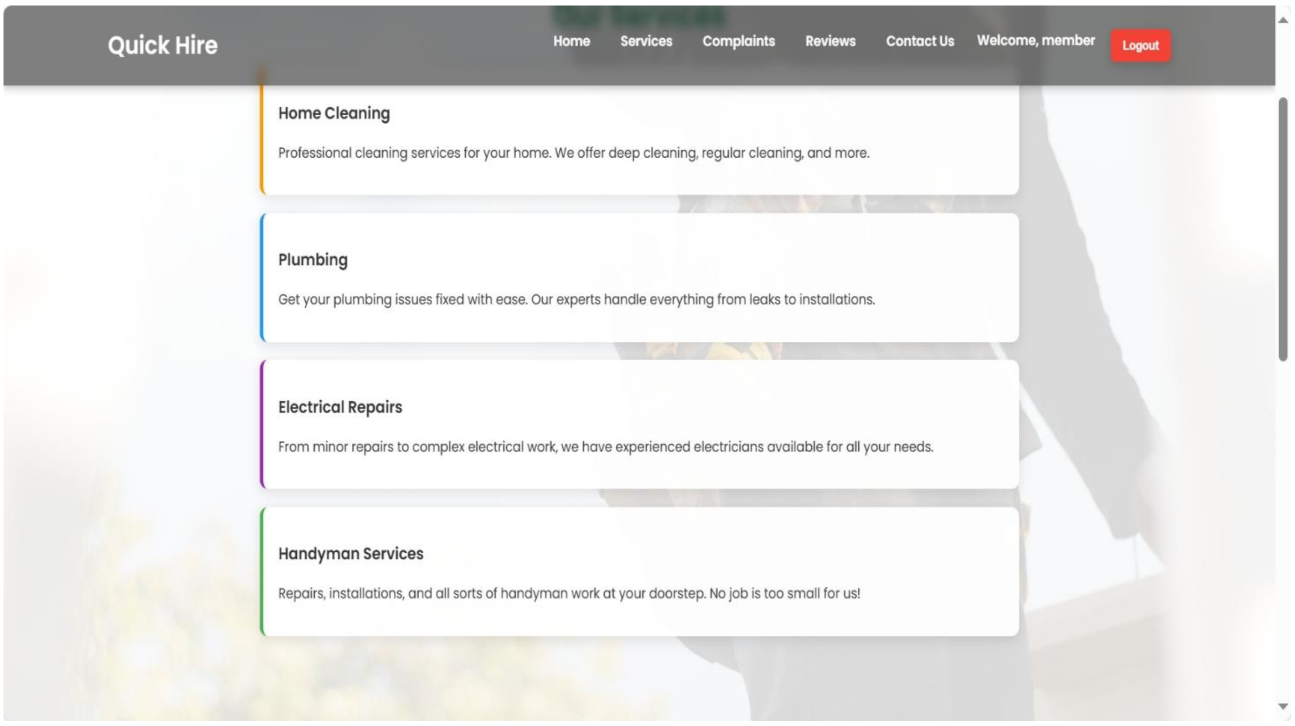
File a Complaint

Your Name

Your Email

Complaint Details

Figure 5.6 Complaint Page



Home Cleaning

Professional cleaning services for your home. We offer deep cleaning, regular cleaning, and more.

Plumbing

Get your plumbing issues fixed with ease. Our experts handle everything from leaks to installations.

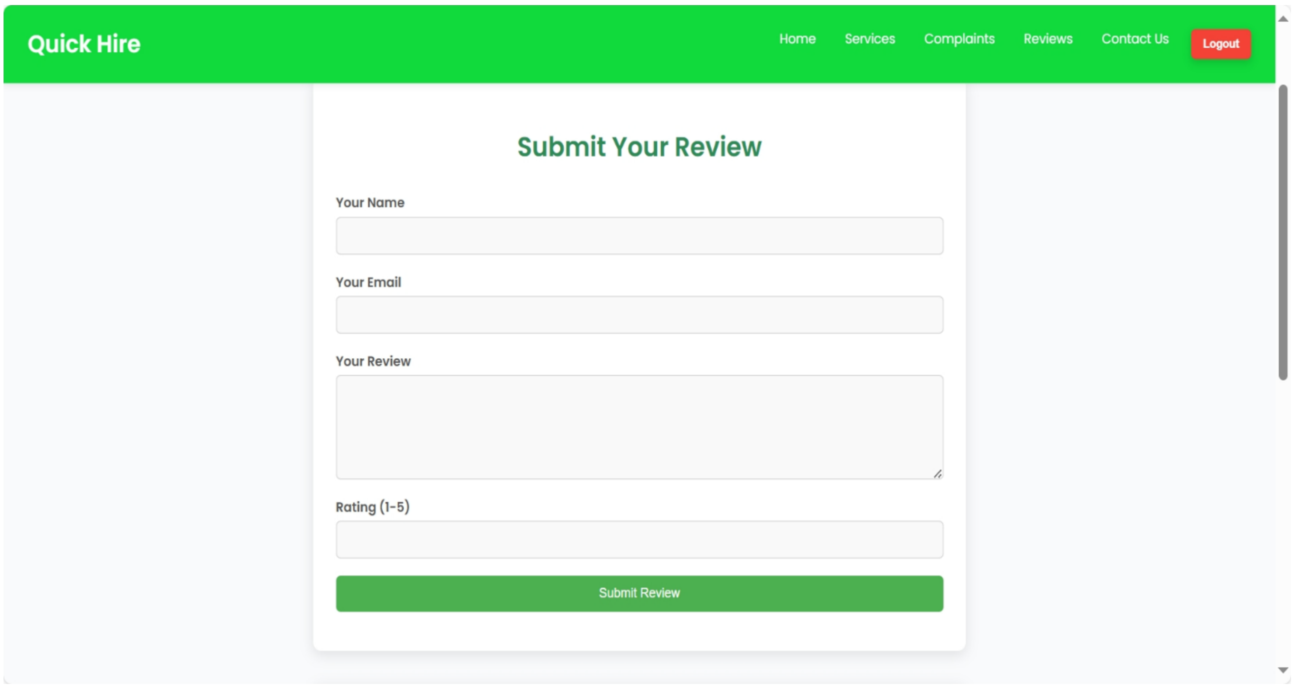
Electrical Repairs

From minor repairs to complex electrical work, we have experienced electricians available for all your needs.

Handyman Services

Repairs, installations, and all sorts of handyman work at your doorstep. No job is too small for us!

Figure 5.7 Service Page



Submit Your Review

Your Name	
Your Email	
Your Review	
Rating (1-5)	

Figure 5.8 Review Page

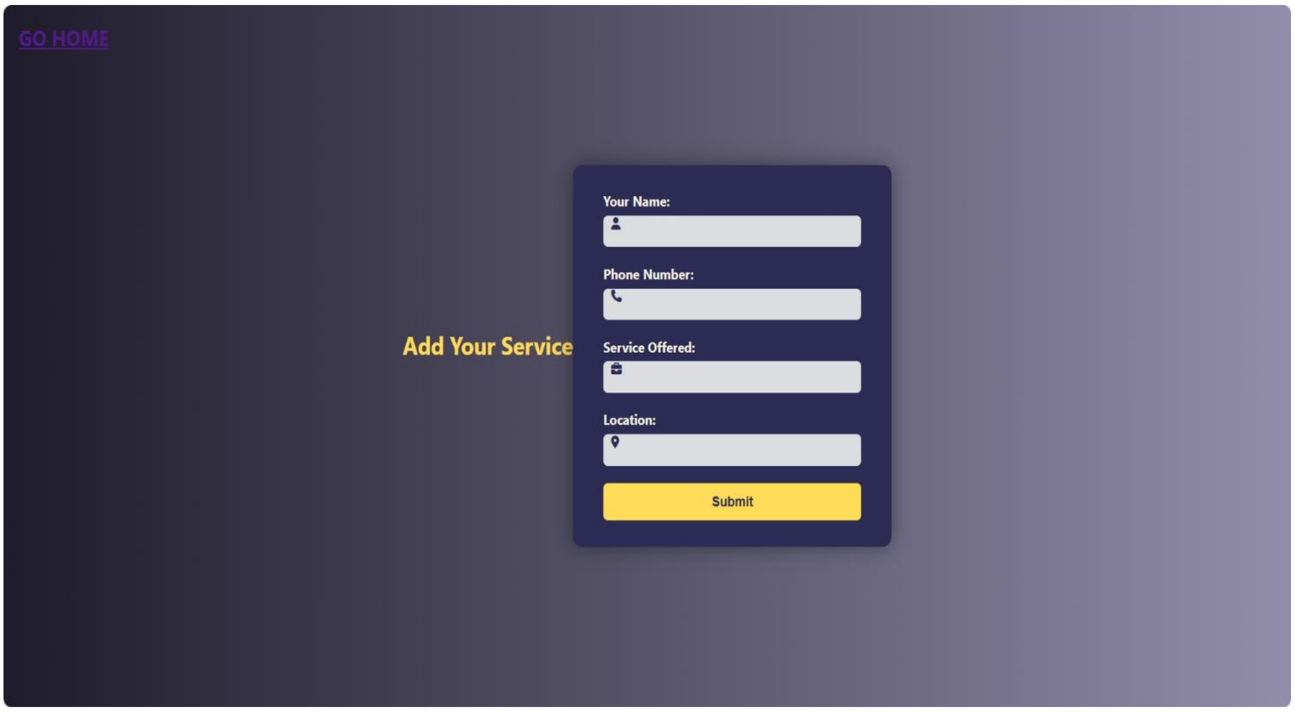
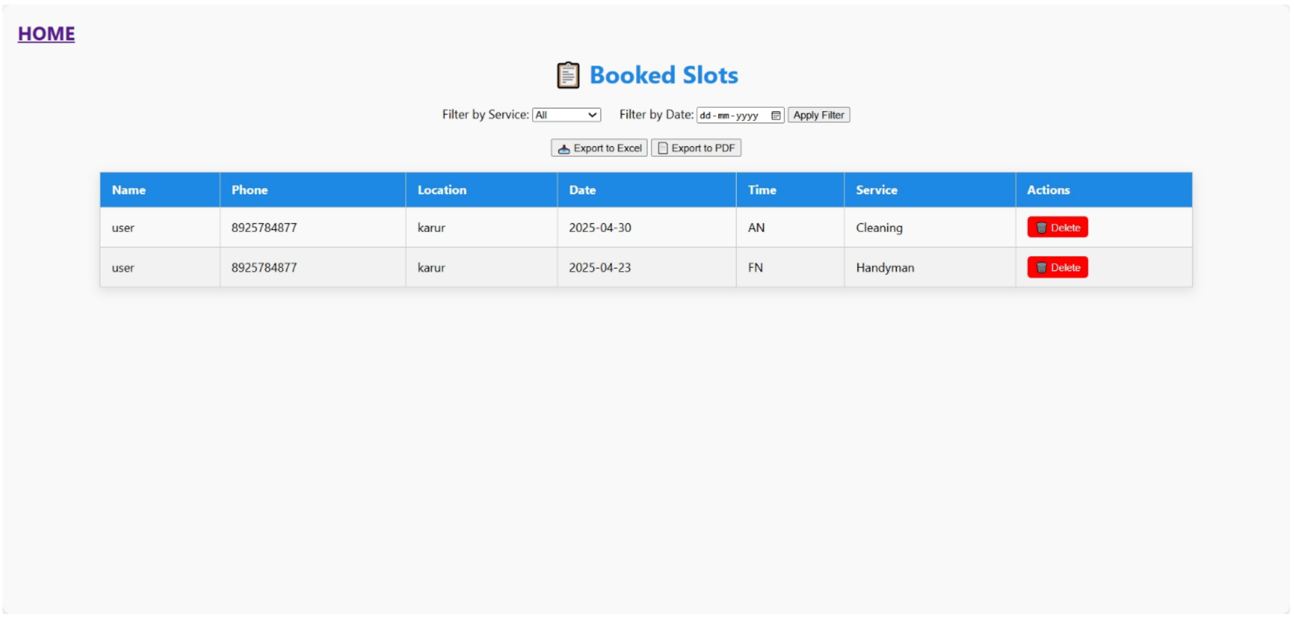
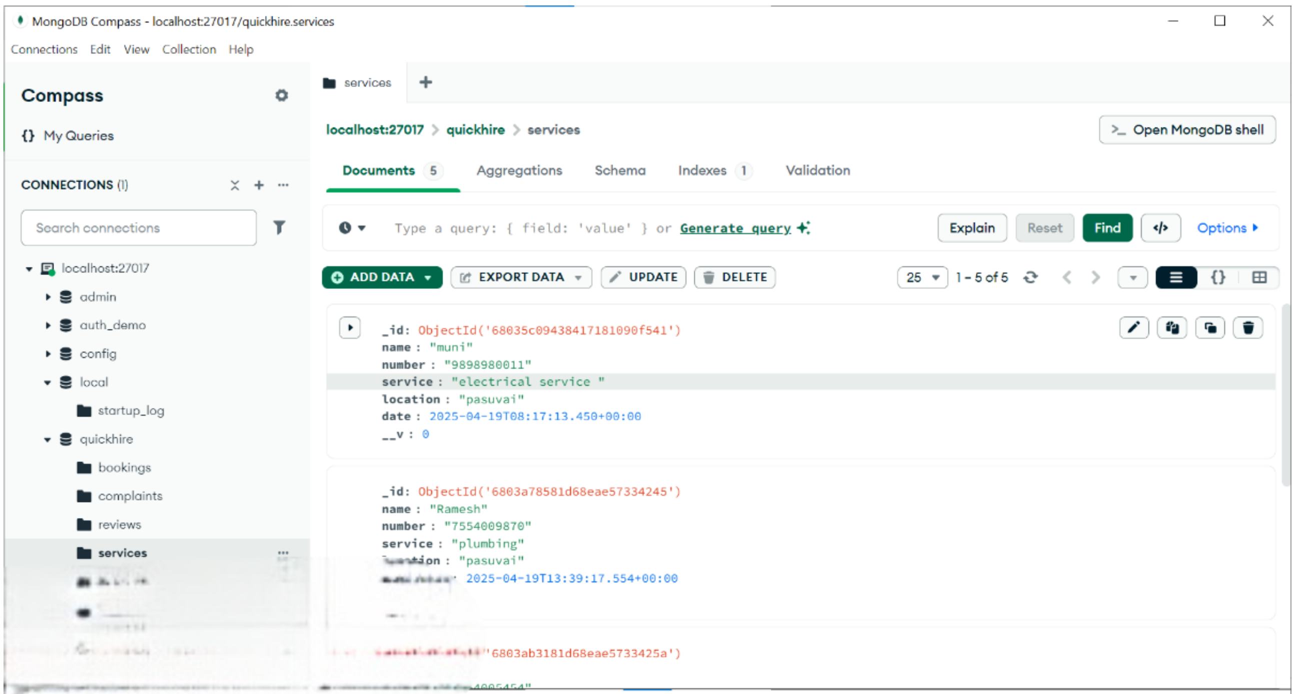


Figure 5.9 Add Service Page



Name	Phone	Location	Date	Time	Service	Actions
user	8925784877	karur	2025-04-30	AN	Cleaning	Delete
user	8925784877	karur	2025-04-23	FN	Handyman	Delete

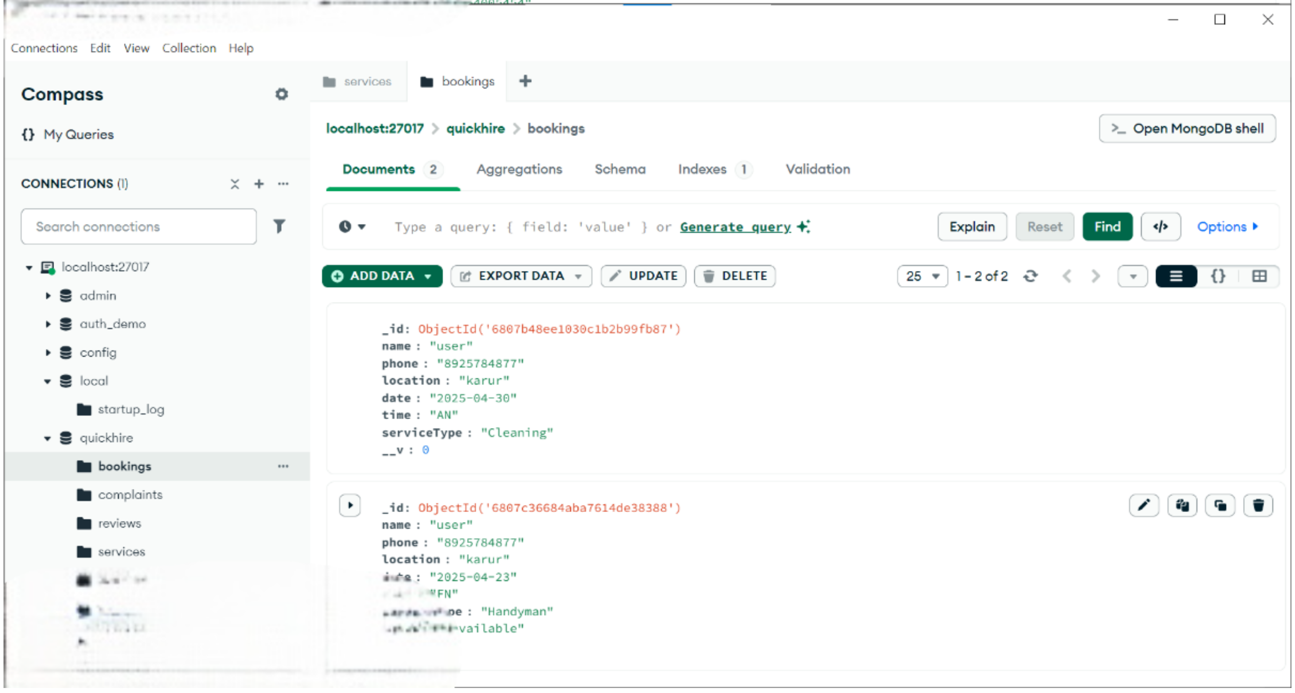
Figure 5.10 Slots Page



```
_id: ObjectId('68035c09438417181090f541')
name : "muni"
number : "9898980011"
service : "electrical service "
location : "pasuvai"
date : 2025-04-19T08:17:13.450+00:00
__v : 0

_id: ObjectId('6803a78581d68eae57334245')
name : "Ramesh"
number : "7554009870"
service : "plumbing"
[illegible] : "pasuvai"
[illegible] 2025-04-19T13:39:17.554+00:00

[illegible]'6803ab3181d68eae5733425a')
[illegible]4005454"
```



```
_id: ObjectId('6807b48ee1030c1b2b99fb87')
name : "user"
phone : "8925784877"
location : "karur"
date : "2025-04-30"
time : "AN"
serviceType : "Cleaning"
__v : 0

_id: ObjectId('6807c36684aba7614de38388')
name : "user"
phone : "8925784877"
location : "karur"
[illegible] : "2025-04-23"
[illegible]"FN"
[illegible]pe : "Handyman"
[illegible]vailable"
```

Figure 5.11 DATA BASE

CHAPTER 5

CONCLUSION

The Quick Hire web application successfully streamlines the service hiring process by providing a secure, user-friendly platform that connects service providers with customers. Through the implementation of session-based authentication and role-based access control, the system ensures a personalized and secure experience for Admins, Members, and Users. Key features such as real-time booking, complaint and review management, and SMS confirmations enhance usability and user engagement.

By integrating technologies like Express.js, MongoDB, and vanilla JavaScript, the application delivers a robust backend with dynamic frontend behavior. Additional functionalities like search, filters, and data export options contribute to the platform's practical value, especially for Admin-level reporting.

Overall, Quick Hire stands as a scalable and efficient solution for digital service management and booking, with the potential for further enhancements such as payment integration, analytics, and mobile responsiveness.

Future Enhancements

1. **Payment Gateway Integration**
    - Enable secure online payments for service bookings through integration with popular payment gateways like Razorpay, Stripe, or PayPal.
2. **Mobile Application Development**
    - Develop Android and iOS mobile apps using React Native or Flutter for better accessibility and user engagement on the go.
3. **Advanced Analytics Dashboard (Admin Panel)**
    - Introduce dashboards with charts and reports to help admins analyze service trends, user activity, and revenue statistics.
4. **Email Notification System**
    - Alongside SMS, integrate email notifications for confirmations, reminders, and promotional messages.
5. **Real-Time Chat Feature**
    - Implement a real-time chat system between users and service providers to improve communication and coordination.
6. **Service Rating and Feedback System**
    - Enhance the review system by adding rating filters, sentiment analysis, and verified feedback tags.
7. **Geo-location and Map Integration**
    - Allow users to find services based on their current location using Google Maps API for improved service relevance.
8. **Multi-Language Support**
    - Add support for regional languages to make the platform more inclusive and accessible.
9. **Subscription Plans for Service Providers**
    - Introduce premium listing options or subscription plans to help service providers promote their offerings.
10. **AI-based Recommendation System**
    - Suggest relevant services to users based on their booking history and preferences using machine learning algorithms.

REFERENCES

- Express.js Documentation
  https://expressjs.com/
- MongoDB Documentation
  https://www.mongodb.com/docs/
- Fast2SMS – Bulk SMS API for Developers
  https://www.fast2sms.com/
- Digital Platforms for Local Services: Challenges & Opportunities
  http://mckinsey.com/industries/technology-media-and-t
- Node.js Documentation
  https://nodejs.org/en/docs/