**MULTISERVICEPROVIDER**

**SERVICE BOOKING APPLICATION**

*Main Project Report*

*Submitted by*

**ABHINAND K S**

**Reg. No.: AJC22MCA-2003**

*In Partial Fulfillment for the Award of the Degree of*

**MASTER OF COMPUTER APPLICATIONS**

**(MCA TWO YEAR)**

[Accredited by NBA]

**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**



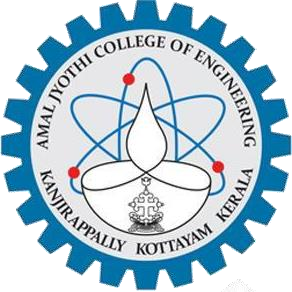
**AMAL JYOTHI COLLEGE OF ENGINEERING**

**KANJIRAPPALLY**

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE Accredited by NAAC. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

# 2023-2024

**DEPARTMENT OF COMPUTER APPLICATIONS AMAL JYOTHI COLLEGE OF ENGINEERING KANJIRAPPALLY**



**CERTIFICATE**

This is to certify that the Project report, “MULTISERVICEPROVIDER**”** is the bonafide work of **ABHINAND K S (Regno: AJC22MCA-2003)** in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

**Ms. Sona Maria Sebastian** **Ms. Meera Rose Mathew**

**Internal Guide Coordinator**

**Rev. Fr. Dr. Rubin Thottupurathu Jose**

**Head of the Department**

## DECLARATION

I hereby declare that the project report **“MULTISERVICEPROVIDER”** is a bona-fide work done at Amal Jyothi College of Engineering, towards the partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2023-2024.

**Date: ABHINAND K S**

**KANJIRAPPALLY Reg: AJC22MCA-2003**

**ACKNOWLEDGEMENT**

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our Manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department

**Rev. Fr. Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the Project Coordinator **Ms. Meera Rose Mathew** for her valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my Guide **Ms. Sona Maria Sebastian** for her inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

**ABHINAND K S**

# ABSTRACT

A “MULTISERVICEPROVIDER” system is a web application that facilitates the seamless booking and delivery of various services such as cleaning, plumbing, laundry, and more. Clients can use this system to request specific services based on their needs, and service providers can then accept these requests and assign appropriate workers to perform the requested tasks. Key features include detailed service listings, flexible booking and scheduling options, worker assignment management, notifications, transparent rating and feedback mechanisms, payment processing, and reputation building through reviews. By seamlessly connecting clients with skilled professionals, the system enhances convenience for clients and expands business opportunities for service providers, fostering efficient and reliable service delivery across a range of industries.

The platform offers users the convenience to reschedule bookings and request new services. Leveraging machine learning, on-demand services are prominently displayed at top the service list. An chatbot addresses user inquiries, ensuring responsive communication. Geo-location guides users to local service providers, enhancing accessibility. Additionally, users can easily track the status of their service bookings, fostering transparency and trust. These features collectively emphasize the platform's commitment to user-centricity and efficient service delivery.

The platform offers service providers a streamlined registration process for new services and a convenient availability calendar. Providers can choose flexible payment options, including fixed rates per service, or commissions. A multi-branch support system facilitates efficient management of different outlets, accompanied by detailed analytics for each branch. Service providers gain insights through work analysis, handle new service requests and approvals, and respond promptly to user feedback. These features collectively create a dynamic and user-friendly environment for efficient service management.

The platform ensures efficient communication for workers by providing real-time updates on appointment changes. It offers performance analytics to aid continuous improvement and allows workers to access detailed task instructions. The platform facilitates task management through a leave application feature and enables workers to apply for new service requests, including qualifications. These features empower workers with the tools and information necessary for effective service delivery and career advancement.

In addition to service providers, each branch within the provider organization is managed by a branch manager responsible for overseeing the workers under the branch and managing booking schedules, salary distribution, and other branch-specific operations. The branch manager utilizes the platform's tools for workforce management, ensuring smooth coordination among workers and timely service delivery within their respective branches. This hierarchical structure further enhances the efficiency and accountability of service provision, contributing to the overall effectiveness of the multiple provider system.

The admin interface ensures a secure and efficient system by verifying new service registrations from providers. Leveraging machine learning, the platform analyzes service provider performance through user feedback, facilitating data-driven decision-making. Admins effectively manage services, overseeing the seamless execution of tasks and ensuring a streamlined service management process. This comprehensive approach empowers administrators to maintain the integrity of the platform, enhance user satisfaction, and uphold high-quality service standards.

**CONTENT**

|  |  |  |
| --- | --- | --- |
| **SL. NO** | **TOPIC** | **PAGE NO** |
| **1** | **INTRODUCTION** | **1** |
| **1.1** | **PROJECT OVERVIEW** | **2** |
| **1.2** | **PROJECT SPECIFICATION** | **2** |
| **2** | **SYSTEM STUDY** | **4** |
| **2.1** | **INTRODUCTION** | **5** |
| **2.2** | **EXISTING SYSTEM** | **5** |
| **2.3** | **DRAWBACKS OF EXISTING SYSTEM** | **6** |
| **2.4** | **PROPOSED SYSTEM** | **7** |
| **2.5** | **ADVANTAGES OF PROPOSED SYSTEM** | **8** |
| **3** | **REQUIREMENT ANALYSIS** | **9** |
| **3.1** | **FEASIBILITY STUDY** | **10** |
| **3.1.1** | **ECONOMICAL FEASIBILITY** | **10** |
| **3.1.2** | **TECHNICAL FEASIBILITY** | **11** |
| **3.1.3** | **BEHAVIORAL FEASIBILITY** | **11** |
| **3.1.4** | **FEASIBILITY STUDY QUESTIONNAIRE** | **12** |
| **3.2** | **SYSTEM SPECIFICATION** | **14** |
| **3.2.1** | **HARDWARE SPECIFICATION** | **14** |
| **3.2.2** | **SOFTWARE SPECIFICATION** | **14** |
| **3.3** | **SOFTWARE DESCRIPTION** | **15** |
| **3.3.1** | **DJANGO** | **15** |
| **3.3.2** | **SQLite** | **15** |
| **3.3.2** | **HTML** | **16** |
| **4** | **SYSTEM DESIGN** | **17** |
| **4.1** | **INTRODUCTION** | **18** |
| **4.2** | **UML DIAGRAM** | **18** |
| **4.2.1** | **USE CASE DIAGRAM** | **19** |
| **4.2.2** | **SEQUENCE DIAGRAM** | **21** |
| **4.2.3** | **STATE CHART DIAGRAM** | **23** |
| **4.2.4** | **ACTIVITY DIAGRAM** | **24** |
| **4.2.5** | **CLASS DIAGRAM** | **26** |
| **4.2.6** | **OBJECT DIAGRAM** | **28** |

|  |  |  |
| --- | --- | --- |
| **4.2.7** | **COMPONENT DIAGRAM** | **28** |
| **4.2.8** | **DEPLOYMENT DIAGRAM** | **29** |
| **4.3** | **USER INTERFACE DESIGN USING FIGMA** | **30** |
| **4.4** | **DATABASE DESIGN** | **32** |
| **5** | **SYSTEM TESTING** | **40** |
| **5.1** | **INTRODUCTION** | **41** |
| **5.2** | **TEST PLAN** | **41** |
| **5.2.1** | **UNIT TESTING** | **42** |
| **5.2.2** | **INTEGRATION TESTING** | **42** |
| **5.2.3** | **VALIDATION TESTING** | **43** |
| **5.2.4** | **USER ACCEPTANCE TESTING** | **43** |
| **5.2.5** | **AUTOMATION TESTING** | **43** |
| **5.2.6** | **SELENIUM TESTING** | **44** |
| **6** | **IMPLEMENTATION** | **60** |
| **6.1** | **INTRODUCTION** | **61** |
| **6.2** | **IMPLEMENTATION PROCEDURE** | **61** |
| **6.2.1** | **USER TRAINING** | **61** |
| **6.2.2** | **TRAINING ON APPLICATION SOFTWARE** | **61** |
| **6.2.3** | **SYSTEM MAINTENANCE** | **62** |
| **7** | **CONCLUSION & FUTURE SCOPE** | **63** |
| **7.1** | **CONCLUSION** | **64** |
| **7.2** | **FUTURE SCOPE** | **64** |
| **8** | **BIBLIOGRAPHY** | **65** |
| **9** | **APPENDIX** | **67** |
| **9.1** | **SAMPLE CODE** | **68** |
| **9.2** | **SCREEN SHOTS** | **96** |

## List of Abbreviation

IDE Integrated Development Environment

HTML Hyper Text Markup Language.

CSS Cascading Style Sheet

SQLite Relational Database Management System

UML Unified Modeling Language

AJAX Asynchronous JavaScript and XML

JS Java Script

PK Primary Key

FK Foreign Key

# CHAPTER 1 INTRODUCTION

### PROJECT OVERVIEW

A “MULTISERVICEPROVIDER” system is a web application that facilitates the seamless booking and delivery of various services such as cleaning, plumbing, laundry and more. Clients can use this system to request specific services based on their needs and service providers can then accept these requests and assign appropriate workers to perform the requested tasks. Key features include detailed service listings, flexible booking and scheduling options, worker assignment management

,notifications transparent rating and feedback mechanisms, payment processing, and reputation building through reviews. By seamlessly connecting clients with skilled professionals, the system enhances convenience for clients and expands business opportunities for service providers, fostering efficient and reliable service delivery across a range of industries.

### PROJECT SPECIFICATION

The “MULTISERVICEPROVIDER” system is an advanced web application designed to streamline the process of accessing and delivering diverse services. Acting as a centralized platform, it efficiently connects clients in search of various services, ranging from cleaning and plumbing to laundry and beyond, with skilled and reliable service providers. Its user-friendly interface simplifies the service request process for clients, enabling them to effortlessly specify their service requirements.

### User

* User can reschedule the booking
* User can request for a new service
* Users view the on demand services on the top of the service list(ML)
* Chatbot for user enquiries
* List Provider based on geolocation
* Track servic booking

### Service Providers

* Service Provider can register for new services
* Decide on the payment for branch managers. It could be fixed rates per service, commissions based
* multi-branch support system allowing them to manage different branches or outlets efficiently.
* Get detailed analytics of each branch
* Work analysis
* View new service request and approval

### Branch Manager

* Service booking calender
* Respond to user feedback
* Add new workers
* Give salary to the workers . It could be, fixed rates per service, commissions based
* View work report and approval
* Leave management of workers

### Worker

* Receive real-time update about the appointment cancellation or rescheduling
* Provide workers with analytics and insights into their performance
* Allow workers to view detailed task notes and instructions provided by users or service providers to ensure they understand the specific requirements of the service.
* Leave application
* Apply for new service request with qualification

**Admin**

* Verify the new service registration by service providers
* Analyse the performance of service providers using feedback from users(ML)
* Manage user requested services
* Service management

### Technologies Used:

Frontend: HTML, CSS, JavaScript, AJAX. Backend: Django.

Database: SQLite.

Payment Gateway Integration: Razor pay.

# CHAPTER 2 SYSTEM STUDY

### INTRODUCTION

In an era defined by the dynamic evolution of service delivery and the increasing reliance on digital platforms, the “MULTISERVICEPROVIDER” system emerges as a pivotal solution catering to the contemporary landscape of service provision. Designed as a robust and versatile web application, the Multiservice Provider System represents a paradigm shift in facilitating seamless connections between service seekers and providers across a spectrum of industries. This innovative system transcends the limitations of traditional service models by offering a comprehensive suite of functionalities that streamline operations for administrators and extend unparalleled convenience and choice to clients. By fostering efficient interactions among Clients, Service Providers, Workers, and Administrators, this platform redefines service accessibility and delivery, ushering in an era of efficient and reliable service provision across diverse sectors.

### EXISTING SYSTEM

The existing system operates solely within a limited scope, enabling users to book a specific service without comprehensive options for broader service engagement. This limitation poses several disadvantages, restricting user choice and versatility. Users are confined to selecting from a narrow range of services, hindering their ability to explore and access a more extensive array of available offerings. This lack of variety may lead to reduced user satisfaction and engagement due to the system's inability to cater to diverse needs. Additionally, the constrained service options limit the platform's potential to attract a wider user base, limiting its overall growth and impact within the market. Consequently, the platform's effectiveness is curtailed by its restricted service scope, limiting user choice and impeding the platform's potential for increased user engagement and market expansion.

### NATURAL SYSTEM STUDIED

The studied natural system is a platform that offers users the capability to book a singular, specific service within a restricted service scope. This system confines users to select only one particular service from a limited range of available offerings. Users encounter limitations when attempting to explore or access a broader spectrum of services, as the platform restricts their choices to a singular option. This restricted service model presents notable disadvantages, hindering user flexibility, choice, and overall satisfaction. The inherent limitations of this system result in a reduced capacity to cater to diverse user needs, potentially limiting user engagement and platform growth within the market.

### DESIGNED SYSTEM STUDIED

The meticulously crafted “MULTISERVICEPROVIDER” System is an innovative platform meticulously designed with a focus on user-centric features, catering to both administrators and customers. Admins are empowered with efficient service management tools, enabling seamless additions, updates, and removals from the service inventory. The platform boasts an intuitive service categorization system, simplifying user navigation and discovery. In-depth insights into customer interactions and paid memberships equip admins with valuable data for informed decision-making and refined strategies. Robust security measures ensure secure access through a fortified login mechanism, safeguarding sensitive data and user information.

Customers, on the other hand, are offered a seamless and engaging experience. Membership registration is effortless, ensuring user engagement from the outset. Secure payment processing guarantees a trustworthy transaction environment, while the option for service rentals adds flexibility to user choices. The platform enriches the user journey with personalized wishlists, empowering users to curate and save their preferred service selections. A robust search functionality further enhances user convenience, allowing for precise service discovery. The streamlined service request process, akin to an optimized shopping cart, simplifies the checkout procedure, ensuring a hassle-free user experience.

This system is meticulously poised to redefine service interactions, fostering increased user engagement, satisfaction, and loyalty. With its user-centric design and array of features, it aims to establish itself as a comprehensive and efficient platform meeting the diverse needs and preferences of both service providers and seekers within a multifaceted service landscape.

### DRAWBACKS OF EXISTING SYSTEM

* Limited Service Range: The existing system might restrict users to a narrow range of service options, limiting their choices and reducing the platform's attractiveness for users with diverse needs.
* Insufficient Service Information: Lack of comprehensive details or descriptions for services offered might hinder users from making informed decisions, impacting user satisfaction and trust in the platform.
* Inefficient Task Assignment: The system may lack efficient tools for service providers to assign tasks to workers based on their expertise and performance, leading to suboptimal task distribution and potentially affecting service quality.
* Inadequate User Engagement Features: Limited personalized recommendations or user engagement strategies might result in reduced user interaction and lower retention rates, impacting the platform's growth potential.
* Security Vulnerabilities: Potential security gaps or vulnerabilities within the system may compromise user data integrity, leading to concerns about confidentiality and trust among users.
* Complex Interface or Navigation: An interface that is challenging to navigate or a complex user journey in service requests may deter users, impacting the overall user experience and satisfaction.
* Scalability Challenges: As the service offerings grow, the system might face scalability issues, leading to performance degradation and slower response times, affecting user experience and satisfaction.

### PROPOSED SYSTEM

“MULTISERVICEPROVIDER” stands as an innovative and comprehensive platform redefining the landscape of service provisioning and utilization. Offering an extensive spectrum of services encompassing cleaning, maintenance, technical support, and a myriad of essential services, “MULTISERVICEPROVIDER” caters comprehensively to the diverse needs of users. The platform prides itself on a user-centric interface, ensuring effortless navigation, service selection, and management.

This advanced system places emphasis on personalized service recommendations, ensuring customized experiences for each user. Equipped with sophisticated task allocation tools, “MULTISERVICEPROVIDER” empowers service providers to efficiently assign tasks to skilled workers based on their expertise, optimizing service quality and delivery.

Security is paramount for “MULTISERVICEPROVIDER”, implementing robust authentication protocols and encrypted payment gateways to safeguard user data and foster trust within the user community. Additionally, a robust feedback mechanism encourages user engagement, allowing for ratings, reviews, and personalized service preferences.

With a steadfast commitment to enhancing convenience, reliability, and user satisfaction, “MULTISERVICEPROVIDER” endeavors to revolutionize the multiservice provider domain. Its core objective is to simplify service accessibility and delivery, adeptly catering to the multifaceted needs of both service providers and seekers across diverse industries and user demographics.

### ADVANTAGES OF PROPOSED SYSTEM

* Diverse Service Options: Provides a wide array of services catering to various user needs.
* Efficient Search and Selection: Enables users to find and choose service providers based on ratings and reviews, ensuring quality service selection.
* Efficient Scheduling: Facilitates convenient scheduling of services for users.
* Real-time Service Tracking: Allows users to track service status, ensuring transparency and reliability.
* Feedback Mechanism: Encourages users to leave reviews and ratings, enhancing transparency and service accountability.
* User-Friendly Interface: Provides a seamless and intuitive user experience, leading to higher engagement and satisfaction.
* Efficient Task Allocation: Empowers service providers to assign tasks to workers based on their skills and reports, ensuring optimized service delivery.
* Detailed Service History: Enables service providers to access and manage service history efficiently.
* Comprehensive Management: Allows admins to oversee and manage users, service providers, and workers on a single platform.
* Work Report Approval: Enables admins to approve work reports, ensuring accuracy and accountability.
* Cancellation Facility: Provides users the flexibility to cancel bookings when needed.
* Multiple Worker Assignment: Allows service providers to assign multiple workers based on reports, ensuring efficient service completion.

# CHAPTER 3 REQUIREMENT ANALYSIS

## FEASIBILITY STUDY

Feasibility is the degree to which a project can be carried out successfully. A feasibility study is conducted to assess the solution's viability, which establishes whether it is viable and implementable in the program. The feasibility study considers details like the availability of resources, software development costs, the advantages of the software to the business once it is built, and the costs associated with maintaining it. The outcome of the feasibility study should be a report recommending whether the requirements engineering, and system development process should be continued. A system is of no real value to a corporation if it does not serve its goals. Even though this may seem obvious, many organizations create systems that do not support their goals, either because they lack a clear statement of these goals, because they fail to specify the system's business requirements, or because other organizational or political factors have an impact on the procurement of the system.

### Economical Feasibility

The economic feasibility of “MULTISERVICEPROVIDER” Project involves a comprehensive evaluation of financial aspects to gauge its viability and profitability. Initial development costs encompass software creation, platform design, database setup, and system testing. Ongoing operational expenses consist of server maintenance, updates, customer support, marketing, and staffing needs. Revenue generation hinges on projected service fees from users or service providers, aligned with anticipated service uptake and market demand. Calculating the Return on Investment (ROI) and determining the Break-Even Point are critical metrics to ascertain profitability. Assessing profit margins per service transaction involves deducting direct costs (e.g., payment processing fees, service provider charges) from service fees. Market research delves into demand assessment, competition analysis, target demographic identification, and estimations of potential market growth. The project's sustainability is gauged by its adaptability to market changes, scalability, and capacity to stay competitive amidst industry fluctuations. This comprehensive economic analysis aims to provide insights into the project's financial potential for long-term success and viability.

### Technical Feasibility

The technical feasibility assessment for the “MULTISERVICEPROVIDER” Project involves a thorough evaluation of essential technological aspects pivotal for its successful implementation. Central to this evaluation is ensuring the availability or acquisition of an appropriate technology infrastructure, encompassing robust web hosting capabilities, scalable databases, and adequate server capacity to sustain platform operations. The project's success heavily relies on the presence of a skilled and proficient software development team capable of effectively constructing and maintaining the multiservice provider platform. Seamless integration with third-party services, such as payment gateways and mapping services, stands as a critical consideration. Moreover, the platform's architecture must exhibit scalability to accommodate an increasing influx of traffic and data as the project expands. Robust security measures to safeguard user data and transactions are imperative, along with ensuring the platform's compatibility across various devices and optimal performance. Compliance with data protection and privacy regulations and a comprehensive assessment of financial resources required for development, hosting, maintenance, and support are also essential aspects within the technical feasibility analysis. This thorough evaluation aims to identify potential technological challenges and facilitate informed decision-making to establish a successful multiservice provider platform.

### Behavioral Feasibility

The behavioral feasibility assessment for the “MULTISERVICEPROVIDER” Project is a critical evaluation encompassing several key aspects crucial for its seamless integration into the organization's existing operations. This evaluation commences with a thorough analysis of resource availability, assessing the presence of requisite human capital and technological infrastructure necessary for successful implementation within the project's designated timeframe and budget constraints. Compatibility with current operational processes stands as a pivotal consideration, warranting an evaluation of the platform's ability to integrate harmoniously with existing systems like service management and task allocation.

Change management strategies are imperative to address potential organizational resistance stemming from operational alterations induced by the platform's implementation. Ensuring third- party support and compliance with legal and regulatory obligations are integral components, emphasizing the readiness of external services and adherence to pertinent laws regarding service provisions, data protection, and user privacy.

Gathering user feedback remains crucial to gauge user acceptance levels, identify areas for

improvement, and align platform features with user preferences. Lastly, analyzing operational costs in alignment with the organization's budgetary framework is essential to ensure financial viability. This comprehensive behavioral feasibility assessment aims to ensure the Multiservice Provider Project's successful integration, fostering effective adoption within the organization's operational landscape.

### Feasibility Study Questionnaire

**Project Overview**

The “MULTISERVICEPROVIDER” Platform is a comprehensive web application aiming to connect users with a wide range of services within their locality. It offers a centralized platform where service seekers can discover and engage with various service providers. The platform facilitates efficient service bookings, provider showcasing, and user engagement, intending to elevate the accessibility and convenience of accessing diverse services.

**To what extent is the system proposed for?**

The platform primarily targets service seekers and providers within urban and suburban areas, catering to their diverse service needs and aspirations for efficient service delivery and discovery.

**Specify the Viewers/Public involved in the System**

Service seekers, service providers, workers employed by providers the core audience of the “MULTISERVICEPROVIDER” platform

**List the Modules included in your System**

The system encompasses modules including User Management, Service Provider Profiles, Service Booking and Scheduling, Provider-User Interaction, Worker Management, Search and Filtering,Notification System, and Database Management.

**Identify the users in your project**

* Service Seekers/Users
* Service Providers/Providers
* Workers (Under Service Providers)
* Administrators

**Who owns the system?**

Administrator

**System's Relation to Firm/Industry/Organization**

The “MULTISERVICEPROVIDER” platform pertains to the service industry, focusing on connecting service seekers with diverse service providers and their associated workers within the local landscape.

**Details of person that you have contacted for data collection?**

Philip James, the Administrator responsible for managing koloapp.in a platform for discovering services, provided valuable insights and details relevant to “MULTISERVICEPROVIDER” platform during our data collection phase.

**Questionnaire to collect details about the project? (min 10 questions, include descriptive answers, attach additional docs (e.g. Bill receipts, certificate models), if any?)**

**Q1:In your experience, what are the biggest pain points when seeking or providing services in the current landscape?**

Answer: For service seekers, it's often the struggle to find reliable, vetted service providers promptly. Service providers face challenges in managing appointments efficiently and establishing trust with potential customers.

**Q2: How crucial is ease of use and accessibility in a platform for service seeking or provision?** Answer: Accessibility and ease of use are paramount. A platform that's intuitive and accessible on various devices ensures a seamless experience for users and providers, encouraging frequent use.

**Q3: Can you elaborate on the challenges faced with traditional booking or service provision methods, such as manual scheduling or word-of-mouth referrals?**

Answer: Manual scheduling can be time-consuming for both parties, leading to missed opportunities or appointments. Word-of-mouth referrals, while valuable, limit the scope of service visibility.

**Q4: What functionalities do you consider essential for efficient service booking and management from a service provider's perspective?**

Answer: Features like real-time availability updates, clear service descriptions, easy appointment scheduling, and user-friendly interfaces significantly enhance service provider management.

**Q5: How important is transparency and credibility in the selection of service providers or when offering services?**

Answer: Transparency and credibility are paramount. Verified service providers and user reviews help in making informed decisions, ensuring trust between parties.

**Q6: From a user's standpoint, what features would streamline the process of finding and booking services?**

Answer: A feature-rich platform offering service categorization, top-rated services displayed prominently, and a straightforward booking process would greatly enhance user experience.

**Q7: Could you describe any instances where a collaborative platform for sharing service experiences and insights would have been beneficial?**

Answer: Such a platform fosters a sense of community and trust. Users can share positive experiences and insights, assisting others in making informed choices.

**Q8: What functionalities or tools would facilitate efficient communication between service providers and users during the service request process?**

Answer: Integrated messaging systems or chat features allowing direct communication, automated appointment reminders, and clear, concise service descriptions contribute to efficient communication.

**Q9: How significant is the ability to track and manage service history or past interactions for service providers or users?**

Answer: Service history tracking ensures better service delivery by providers and helps users easily access past services for reference or rebooking.

**Q10: What level of personalization or customization would enhance the user experience when seeking or providing services?**

Answer: Customizable preferences for service recommendations, personalized user profiles, and tailored service suggestions based on historical usage patterns could greatly enrich the experience.

## SYSTEM SPECIFICATION

* + 1. **Hardware Specification**

Processor Intel Pentium and above

RAM 4GB and above

Hard disk 256 HDD and above

### Software Specification

Front End HTML5, CSS, Bootstrap

Back End Django, SQLite

Database SQLite

Client on PC Windows 7 and above.

Technologies used Django, HTML5, AJAX, Bootstrap, JS, jQuery

## SOFTWARE DESCRIPTION

### DJANGO

Django, a leading open-source web framework, epitomizes the pinnacle of modern web development with its efficiency and versatility. Built on Python, Django follows the Model-View- Controller architecture, prioritizing simplicity and flexibility. Noteworthy features include its Object-Relational Mapping system, streamlined URL routing, and a user-friendly template engine. The built-in admin interface simplifies data management, while robust security measures and middleware support enhance application integrity. Django scales effortlessly, accommodating growing datasets and traffic demands. Its REST Framework extension further extends its utility to API development. Supported by an active community and extensive documentation, Django stands as a powerful toolkit, seamlessly shaping the development of dynamic and secure web applications for diverse purposes, from content management systems to Restful API. In essence, Django empowers developers with a comprehensive and adaptable framework for crafting sophisticated and salable web solutions**.**

### SQLite

SQLite, a lightweight and server less relational database management system, is celebrated for its simplicity, portability, and efficiency. Operating as a self-contained, single-file database, SQLite requires minimal setup and eliminates the need for a separate server process. This design makes it a preferred choice for embedded systems, mobile applications, and small to medium-scale projects. With zero configuration and a server less architecture, SQLite streamlines the database management process. It supports ACID transactions, ensuring data integrity and reliability, even in the face of system interruptions. The dynamic typing feature allows flexible data storage, accommodating various data types within the same column.

Noteworthy is SQLite's cross-platform compatibility, making it versatile across different operating systems and environments. Its wide adoption is evident in applications ranging from mobile apps to web browsers, where it is utilized for local data storage. SQLite stands out as a go-to solution for projects requiring a lightweight, self-contained, and easily deployable database system, embodying simplicity without compromising on functionality and reliability.

### HTML

HTML (Hypertext Markup Language) serves as the backbone of web development, providing the fundamental structure and elements for creating web pages. As a core technology, HTML facilitates content presentation, defining the layout, and structuring data within web applications. Its simplicity lies in the markup tags that encapsulate content, guiding browsers on how to display information. HTML5, the latest version, introduces advanced features, such as native support for multimedia elements (audio, video), enhanced form controls, and canvas for graphics rendering. This iteration fosters the development of interactive and engaging web experiences, empowering developers to craft dynamic content and responsive layouts that adapt seamlessly across devices. HTML's versatility, coupled with its interoperability across browsers and platforms, solidifies its position as an indispensable language in web development, serving as the foundation upon which modern web applications are built.

This addition elaborates on HTML's role in web development, emphasizing its pivotal role in structuring content and enabling the creation of engaging and interactive web experiences across different platforms and devices.

# CHAPTER 4 SYSTEM DESIGN

## INTRODUCTION

Any designed system or product's development process begins with the design phase. An efficient system depends on a well-executed design, which is a creative process. It entails utilizing a variety of techniques and concepts to completely specify a procedure or system for it to be implemented. Regardless of the development paradigm used in software engineering, the design phase is essential. It seeks to produce the architectural detail needed to design a system or product and acts as the process's technical backbone.

To maximize every aspect of efficiency, performance, and accuracy, this program underwent a comprehensive design phase. A user-oriented document is converted into a document for programmers or database workers throughout the design phase.

## UML DIAGRAM

Software engineering uses the Unified Modeling Language (UML), a standardized visual language, to model, develop, and document software systems. UML diagrams provide a concise and organized approach to represent different facets of a software system, serving as a common communication tool for developers, stakeholders, and designers. There are many different varieties of UML diagrams, including class diagrams, sequence diagrams, use case diagrams, and more. Each is designed to communicate a particular piece of knowledge about the design, operation, and interactions of the system. UML diagrams are essential to the software development process because they help with the visualization, analysis, and design of complex systems, which in turn makes the process more effective and efficient.

* + - Use case diagram
    - Sequence diagram
    - State diagram
    - Activity diagram
    - Class diagram
    - Object diagram
    - Component diagram
    - Deployment diagram

## USE CASE DIAGRAM

A use case diagram is a visual representation of the interactions between system components. An approach for identifying, outlining, and organizing system requirements is called a use case. The word "system" here refers to a thing that is being created or run, like a website for mail- order product sales and services. UML (Unified Modeling Language), a standard language for the modelling of real-world objects and systems, uses use case diagrams. The planning of general requirements, the validation of a hardware design, the testing and debugging of a software product in development, the creation of an online help reference, or the completion of a job focused on customer support are all examples of system objectives. For instance, use cases in a product sales context can involve customer service, item ordering, catalogue updating, and payment processing. There are four elements in a use case diagram.

* + The boundary, which defines the system of interest in relation to the world around it.
  + The actors, usually individuals involved with the system defined according to their roles.
  + The use cases, which are the specific roles are played by the actors within and around the system.
  + The relationships between and among the actors and the use cases. Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we must use the following guidelines to draw an efficient use case diagram.
  + The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.
  + Give a suitable name for actors.
  + Show relationships and dependencies clearly in the diagram.
  + Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
  + Use notes whenever required to clarify some important points.

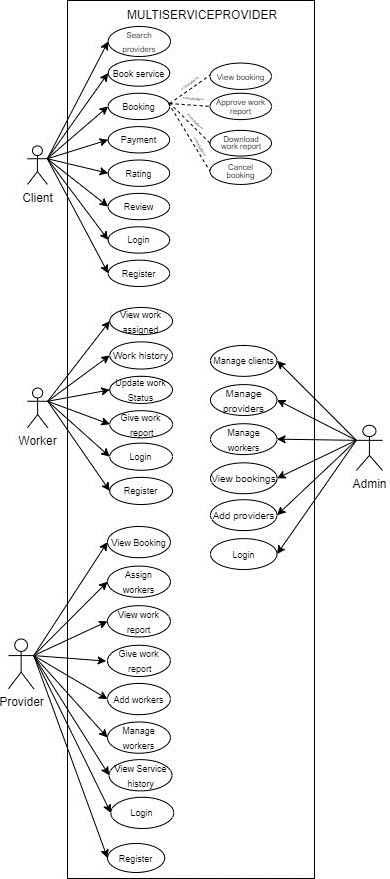


Figure 1: Use Case diagram

## SEQUENCE DIAGRAM

A sort of Unified Modeling Language (UML) diagram called a sequence diagram is used in software engineering to depict the communications and interactions that take place across time between various system components or objects. It gives a live view of the interactions between objects as they work together to complete a certain job or situation.

Objects are shown as lifelines in a sequence diagram, and vertical lines indicate their existence over time. The flow of calls or messages between objects is shown by arrows and lines between lifelines. These diagrams are very helpful for demonstrating the interactions within a particular use case, assisting in determining the sequence of operations and the functions that each object performs.

Sequence diagrams are essential for understanding the behavior of a system and for ensuring that the software functions as intended, as they offer a clear and detailed depiction of how different components or objects collaborate to achieve a particular functionality.

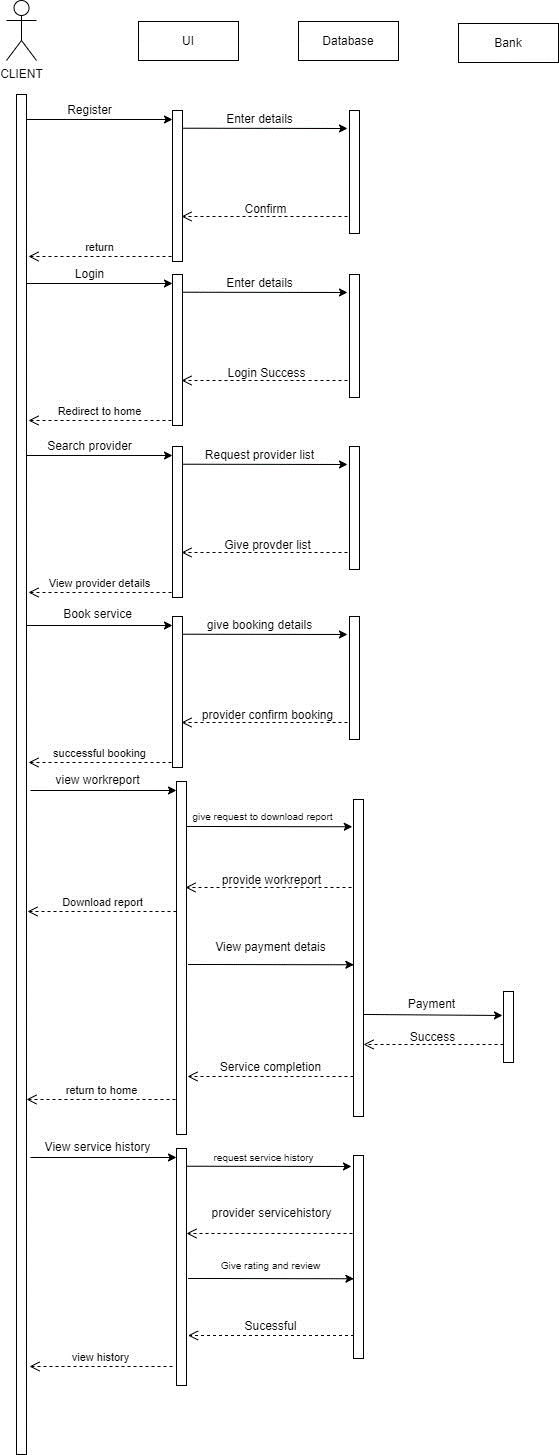


Figure 2: Sequence diagram

## State Chart Diagram

A state diagram, also known as a state machine diagram or state chart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modeling Language (UML). In this context, a state defines a stage in the evolution or behavior of an object, which is a specific entity in a program or the unit of code representing that entity. A state diagram resembles a flowchart in nature; however, a flowchart shows the processes within a system that alters the state of an object rather than the actual state changes themselves. The first step to creating a state chart diagram is identifying the initial and final states of a system. Then, all the possible existing states are placed in relation to the beginning and the end. Lastly, all of the events that trigger state changes are labeled as transition elements.

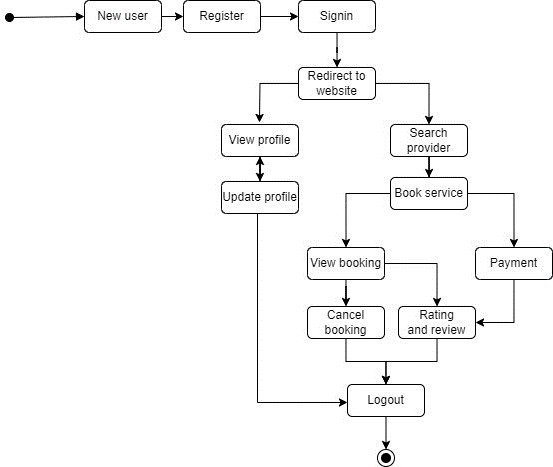


Figure 3: State Chart diagram

## Activity Diagram

An activity diagram is essentially a flowchart that shows how one activity leads to another. The action might be referred to as a system operation. One operation leads to the next in the control flow. This flow may be parallel, contemporaneous, or branched. Activity diagrams use many features, such as fork, join, etc., to cope with all types of flow control. An activity diagram is a behavioral diagram i.e., it depicts the behavior of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

Activity diagram is another important behavioral diagram in UML diagram to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity.

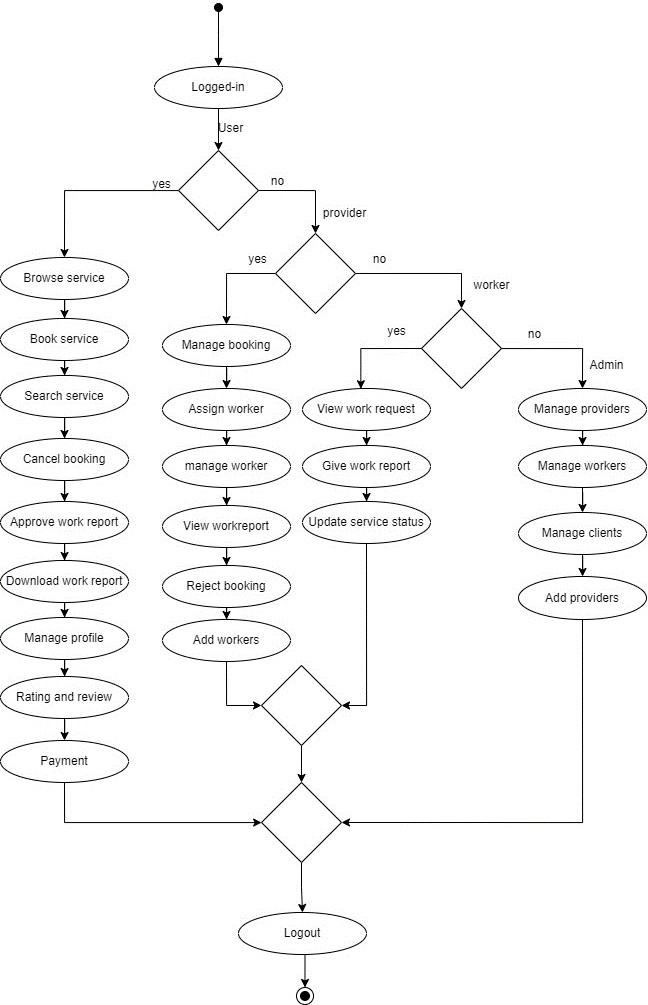


Figure 4: Activity diagram

## Class Diagram

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling, translating the models into programming code. Class diagrams can also be used for data modeling. Class diagrams are the blueprints of your system or subsystem. You can use class diagrams to model the objects that make up the system, to display the relationships between the objects, and to describe what those objects do and the services that they provide. Class diagrams are useful in many stages of system design. In the analysis stage, a class diagram can help you to understand the requirements of your problem domain and to identify its components. In an object-oriented software project, the class diagrams that you create during the early stages of the project contain classes that often translate into actual software classes and objects when you write code. Later, you can refine your earlier analysis and conceptual models into class diagrams that show the specific parts of your system, user interfaces, logical implementations, and so on. Your class diagrams then become a snapshot that describes exactly how your system works, the relationships between system components at many levels, and how you plan to implement those components.

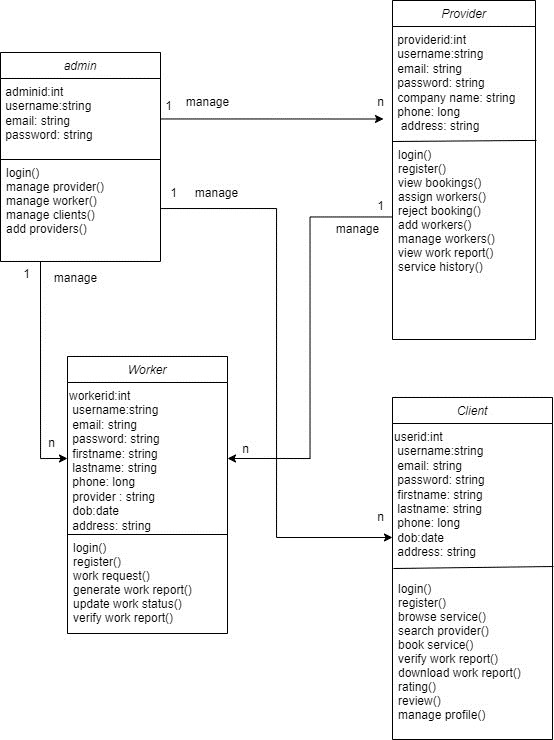


Figure 5: Class diagram

## Object Diagram

Since class diagrams are the source of object diagrams, class diagrams are a prerequisite for object diagrams. An instance of a class diagram is represented by an object diagram. Class and object diagrams both use the same fundamental ideas. The static view of a system is also represented by object diagrams, but this static view represents a momentary snapshot of the system. To represent a group of items and their connections as an instance, object diagrams are employed

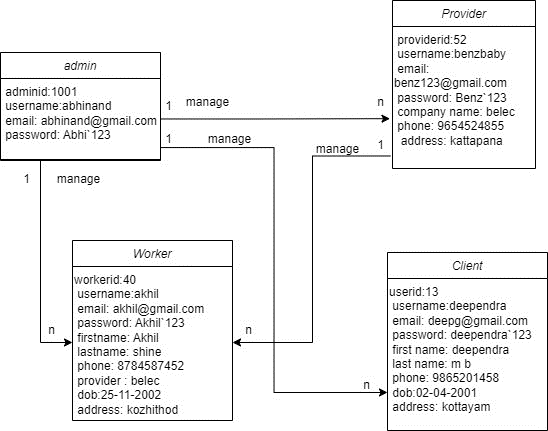


Figure 6: Object diagram

## Component Diagram

UML diagrams are used to represent and describe the behavior of object-oriented systems, assisting in visualization, explanation, and thorough documentation. Particularly in class diagrams, where they record the static behavior of a system, these diagrams play a vital role. On the other hand, graphics can impair the efficiency of real multitasking systems. To maintain coordination, each part within the broader process has a distinct duty and only communicates with other essential components.

A component diagram depicts how components are wired together to form larger components or software systems. They are used to illustrate the structure of arbitrarily complex systems.

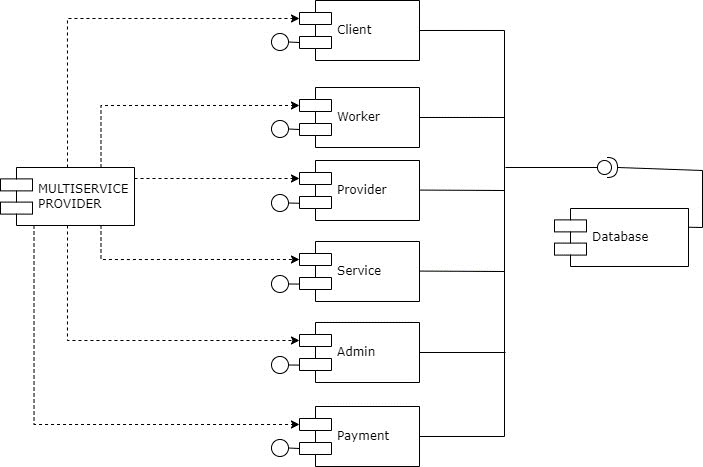


Figure 7: Component diagram

## Deployment Diagram

The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships. It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths. In contrast to other UML diagram types, which primarily depict the logical components of a system. The deployment diagram does not focus on the logical components of the system, but it put its attention on the hardware topology.

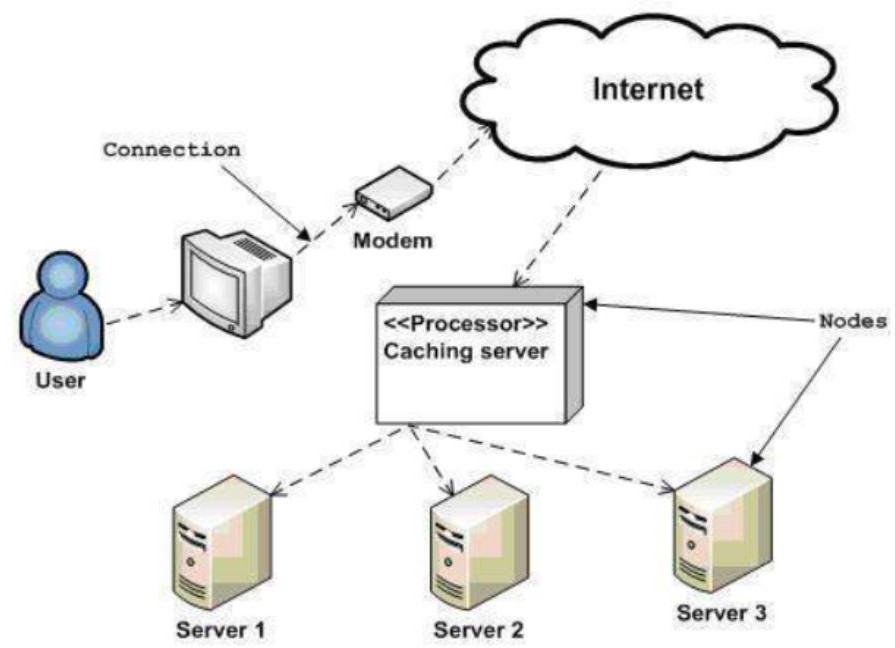
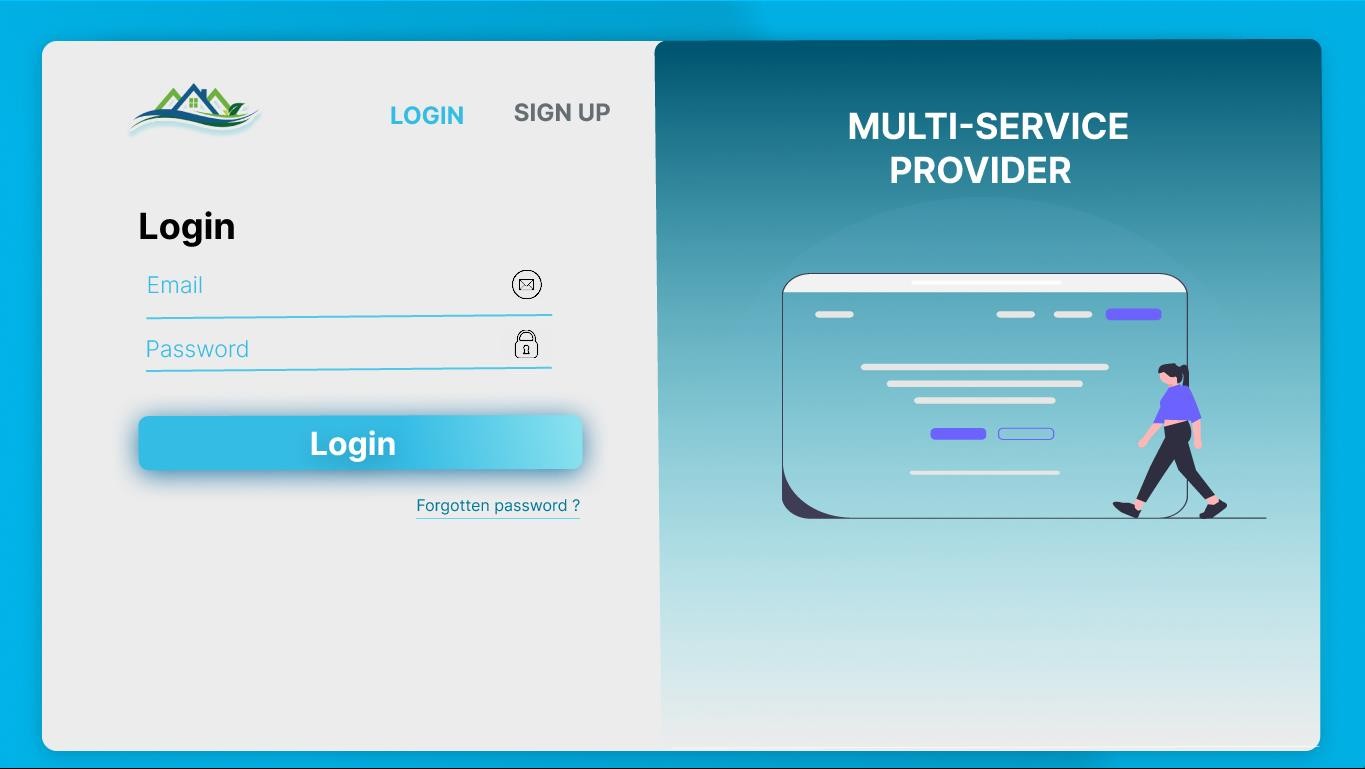


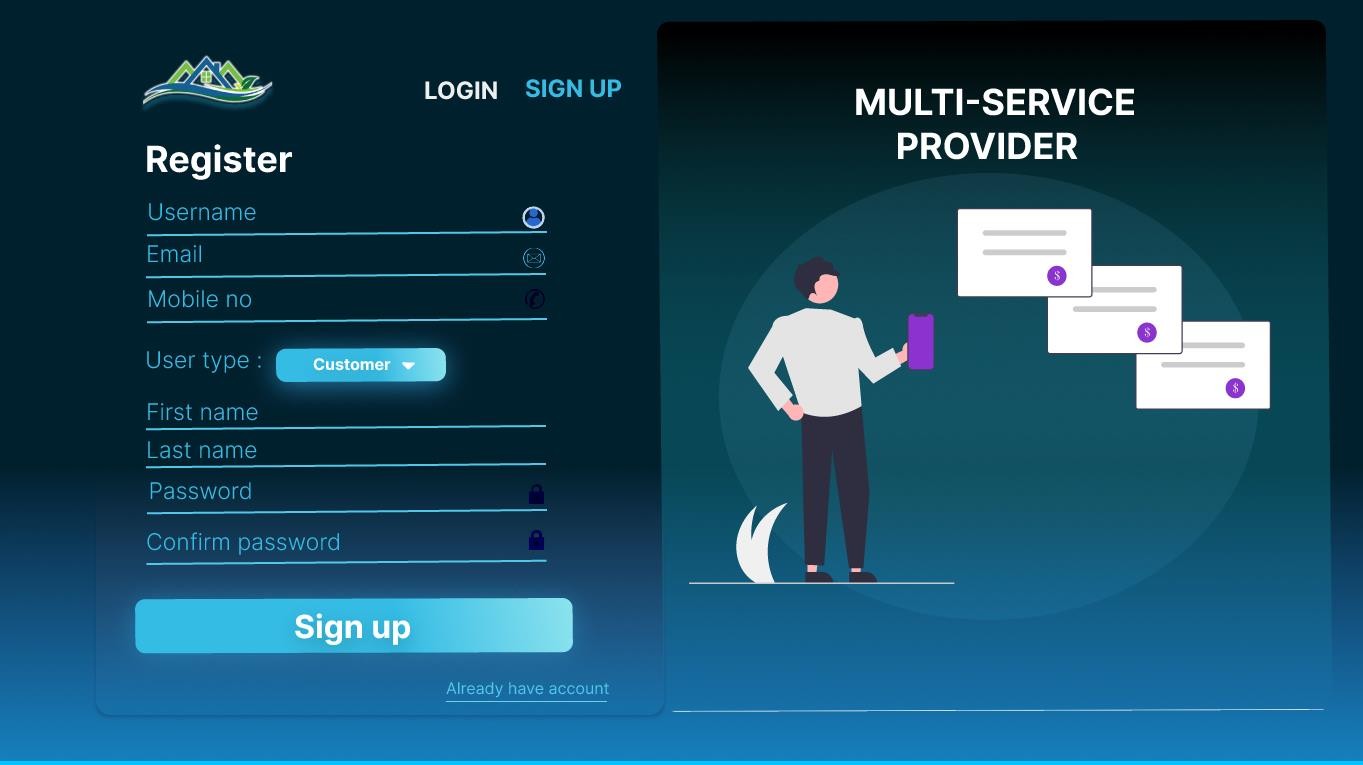
Figure 8: Deployment diagram

## USER INTERFACE DESIGN USING FIGMA

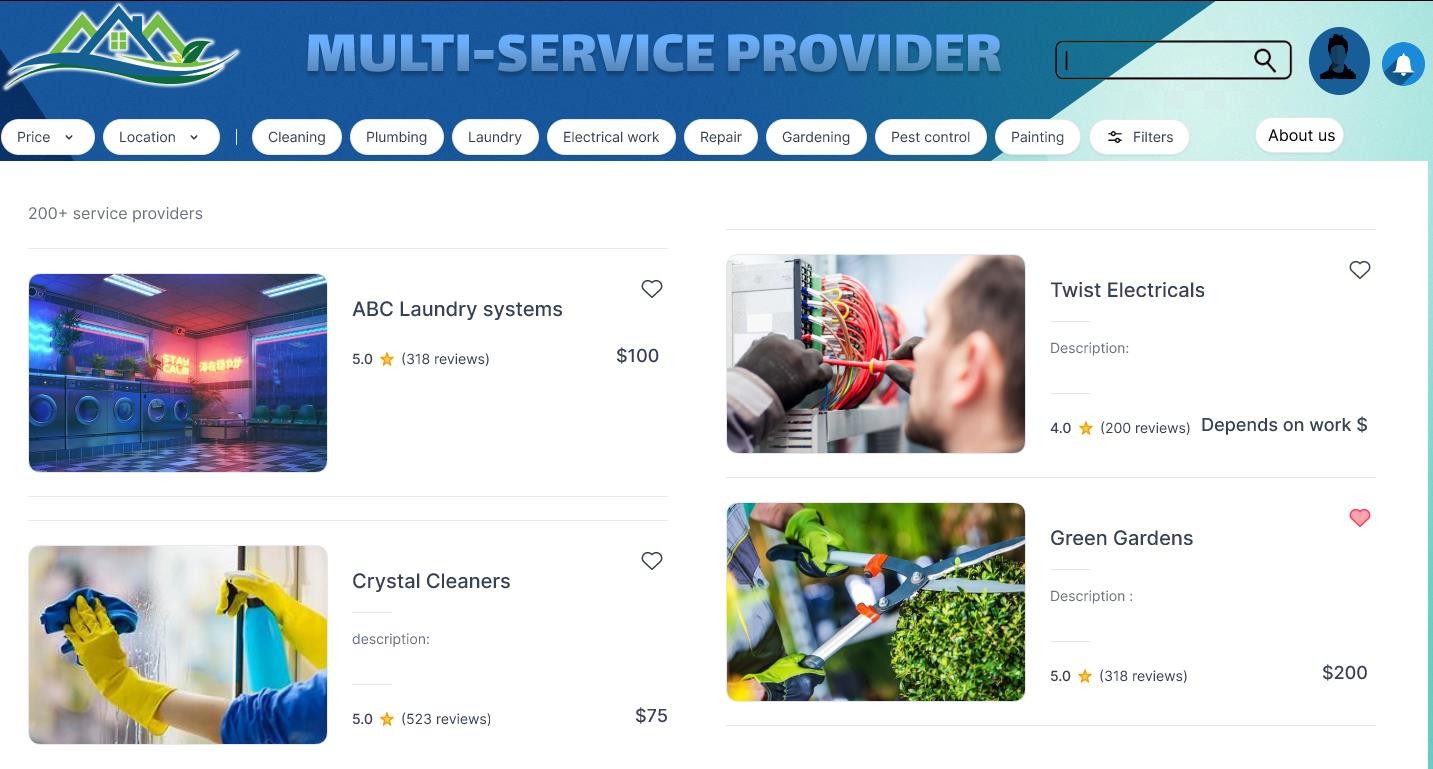
### Login page:



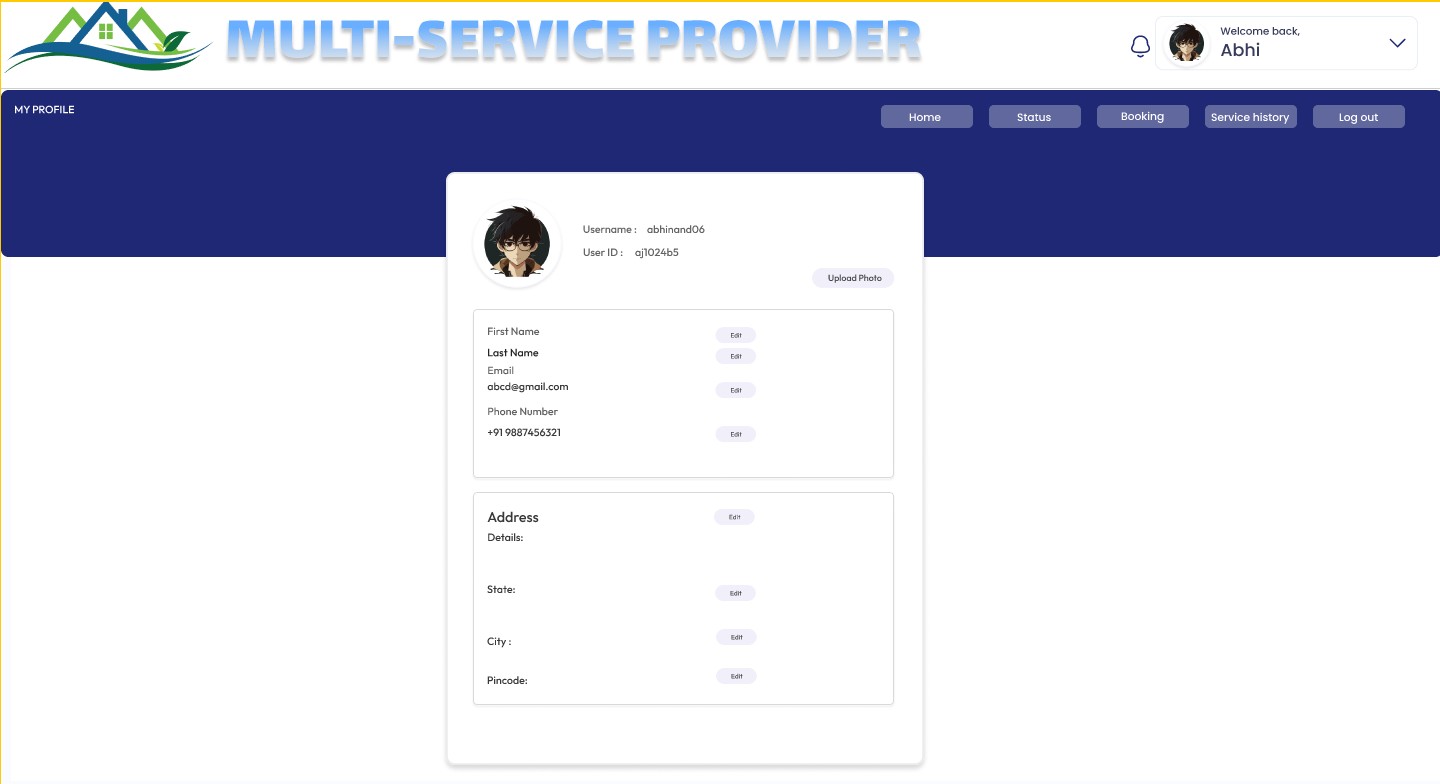
**Registration form :**



### Home page:



**Profile page :**



## DATABASE DESIGN

A database is a collection of data that has been organized to make it simple to manage and update. Information security could be one of the main aims of any database. The database design process is divided into two steps. The goal of the first step is to gather user requirements to create a database that as clearly as possible satisfies user needs. It is known as information- level design and is carried out without the aid of any DBMS. An information-level design is changed to a specific DBMS design that will be used to develop the system in the following stage. The physical-level design phase is where the characteristics of the specific DBMS are considered.

### Relational Database Management System (RDBMS)

A Relational Database Management System (RDBMS) is a critical software application for organizing and managing data in a structured manner. It stores data in tables with rows and columns, where each row represents a record, and each column is a specific attribute or field. RDBMS systems like MySQL, Oracle, or Microsoft SQL Server ensure data integrity, enforce relationships between tables, and allow for efficient data retrieval and manipulation through Structured Query Language (SQL). These systems are widely used in various applications, such as e-commerce websites, financial systems, and inventory management, due to their robustness, scalability, and ability to handle complex data structures, making them essential for modern data-driven environments.

A Relational Database Management System (RDBMS) is a cornerstone of modern data management. It structures data into tables, where each row represents a unique entry, and each column corresponds to a specific attribute, ensuring a logical and organized storage system. RDBMS systems employ complex algorithms for data retrieval and manipulation, guaranteeing data consistency and adherence to predefined relationships between tables. Popular RDBMS software, including PostgreSQL, MySQL, and Microsoft SQL Server, provides robust features for transactions, data security, and scalability. These systems are integral to a myriad of applications, from healthcare records and customer databases to inventory control and financial platforms, supporting the efficient storage, retrieval, and analysis of vast datasets, making them essential tools in today's data-driven world.

### Normalization

Normalization is a database design technique used in relational database management systems (RDBMS) to eliminate data redundancy and improve data integrity. It involves organizing data in a way that reduces data duplication and ensures that relationships between tables are defined and maintained.

The primary goals of normalization are:

Minimizing Data Redundancy: By breaking down data into separate tables and ensuring that each piece of data is stored only once, normalization helps reduce the chances of data inconsistencies or errors.

Ensuring Data Integrity: Normalization enforces the rules of referential integrity, which means that relationships between tables are well-defined and maintained. This ensures that data remains accurate and consistent.

Normalization typically involves dividing a database into multiple related tables and using primary keys and foreign keys to establish relationships between these tables. There are several normal forms, from First Normal Form (1NF) to Fifth Normal Form (5NF), each with specific rules and requirements. The level of normalization achieved depends on the specific needs of the database and the trade-off between data redundancy and query performance.

By applying normalization principles, designers can create efficient and reliable database structures that support data integrity and ease data maintenance and manipulation.

There are several normal forms (NF) that define specific rules and requirements for achieving progressively higher levels of normalization. Here are the most common normal forms, from First Normal Form (1NF) to Fifth Normal Form (5NF):

First Normal Form (1NF):

* Each table has a primary key.
* All columns contain atomic (indivisible) values.
* There are no repeating groups or arrays in columns. Second Normal Form (2NF):
* The table is in 1NF.
* All non-key attributes are fully functionally dependent on the entire primary key. In other words, all non-key attributes must be dependent on the entire primary key, not just part of it.

Third Normal Form (3NF):

* The table is in 2NF.
* There is no transitive dependency, meaning that non-key attributes are not dependent on other non-key attributes.

Boyce-Codd Normal Form (BCNF):

* A stricter version of 3NF.
* It enforces that every non-trivial functional dependency involves a super key. Fourth Normal Form (4NF):
* Addresses multi-valued dependencies.
* Eliminates any multi-valued dependencies within the data. Fifth Normal Form (5NF):
* Also known as Project-Join Normal Form (PJ/NF).
* Handles cases where data can be derived by joining multiple tables.

### Sanitization

In Python Django, sanitization refers to the process of cleaning and validating data to ensure that it is safe and free from malicious content before it is used or stored in a database. Sanitization is a crucial security practice to prevent various forms of attacks, such as cross-site scripting (XSS) and SQL injection, which can compromise the security and integrity of a web application.

### Indexing

Indexing in Django is a fundamental database optimization technique. It involves creating data structures, or indexes, on specific fields to expedite data retrieval. These indexes act as signposts that enable the database to swiftly locate and fetch relevant data, especially in tables with substantial amounts of information. Django offers automatic index creation for primary keys, unique fields, and foreign keys. Additionally, developers can define custom indexes for fields frequently used in filtering, sorting, or searching. The choice of proper indexing plays a pivotal role in improving query performance, resulting in more responsive and scalable web applications. It's essential to consider application-specific query patterns and create indexes accordingly, as well as to understand the capabilities and limitations of the chosen database backend. Effective indexing is a cornerstone of efficient database operations in Django, contributing to enhanced application performance.

### TABLE DESIGN

1. **Table : user**

Primary key: **User\_Id**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NO | FIELD NAME | DATA TYPE  (size) | KEY CONSTRAINT | DESCRIPTION OF THE FIELD |
| 1. | User\_Id | Int(10) | Primary key, Autoincrement | User id |
| 2. | Username | Varchar(20) | Not Null,Unique | User name |
| 4. | Email | Varchar(30) | Not Null,Unique | Email |
| 5. | Password | Varchar(20) | Not Null | Password |
| 6. | ROLE | Varchar(15) | Not Null | Role of the user |
| 7. | Status | Varchar(15) | Not Null | For activating and deactivating user |

1. **Table : client**

Foreign key: **User\_Id** references table **user**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NO | FIELD NAME | DATA TYPE | KEY CONSTRAINT | DESCRIPTION OF THE FIELD |
| 1. | User\_Id | Int(10) | Foreign key | User Id |
| 2. | First name | Varchar(20) | Not Null | First name of client |
| 3. | Last name | Varchar(20) | Not Null | Last name of client |
| 4. | DOB | Date | Not Null | Date of birth |
| 5. | Phone no | Int(10) | Not Null Unique | Phone number |
| 6. | State | Varchar(20) | Not Null | State |
| 7. | District | Varchar(20) | Not Null | District |

1. **Table : provider**

Primary key: **Provider\_id**

Foreign key: **User\_Id** references table **user**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NO | FIELD NAME | DATA TYPE | KEY CONSTRAINT | DESCRIPTION OF THE FIELD |
| 1. | Provider\_Id | Int(10) | Primary key, Auto increment | Provider Id |
| 2. | User\_id | Int(10) | Foreign key | User id of provider |
| 3. | Provider name | Varchar(20) | Not Null,Unique | Provider name |
| 4. | Servicet type | Varchar(20) | Not Null | Type of service providing |
| 5. | Date of creation | Date | Not Null | Date company started |
| 6. | Phone no | Int(10) | Not Null,Unique | Phone number |
| 7. | State | Varchar(20) | Not Null | State |
| 8. | District | Varchar(20) | Not Null | District |
| 9. | Email | Varchar(30) | Not Null,Unique | Email |

1. **Table : worker**

Primary key: **Worker\_id**

Foreign key: **User\_Id** references table **user**

Foreign key: **Provider\_id** references table **provider**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NO | FIELD NAME | DATA TYPE | KEY CONSTRAINT | DESCRIPTION OF THE FIELD |
| 1. | Worker\_id | Int(10) | Primary key, Auto increment | Worker id |
| 2. | User\_Id | Int(10) | Foreign key | User Id of worker |
| 3. | Provider\_id | Int(10) | Foreign key | Provider id which the worker works for |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4. | DOB | Date | Not Null | Date of birth |
| 5. | Date of joining | Date | Not Null | Date of joining company |
| 6. | Phone no | Int(10) | Not Null,Unique | Phone number |
| 7. | State | Varchar(20) | Not Null | State |
| 8. | District | Varchar(20) | Not Null | District |

1. **Table : booking**

Primary key: **Booking\_id**

Foreign key: **User\_Id** references table **user**

Foreign key: **Provider\_id** references table **provider**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NO | FIELD NAME | DATA TYPE | KEY CONSTRAINT | DESCRIPTION OF THE FIELD |
| 1. | Booking\_id | Int(10) | Primary key, Auto increment | Booking id |
| 2. | User\_id | Int(10) | Foreign key | Clients User id |
| 3. | Provider\_Id | Int(10) | Foreign key | Provider id which the client booked |
| 4. | Booking\_date | Date | Not Null | Booking date |
| 5. | Status | Varchar(10) | Not Null | Booking status |

1. **Table : service**

Primary key: **Service\_id**

Foreign key: Worker**\_id** references table **worker** Foreign key: **User\_Id** references table **user**

Foreign key: **Booking\_id** references table **booking**

Foreign key: **Provider\_id** references table **provider**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NO | FIELD NAME | DATA TYPE | KEY CONSTRAINT | DESCRIPTION OF THE FIELD |
| 1. | Service\_id | Int(10) | Primary key, Auto increment | Service id |
| 2. | Worker\_id | Int(10) | Foreign key | Worker id of worker assigned for service |
| 3. | User\_Id | Int(10) | Foreign key | User Id of client |
| 4. | Provider\_id | Int(10) | Foreign key | Provider id which the client booked |
| 5. | Booking\_id | Int(10) | Foreign key | Booking id |
| 6. | Date | Date | Not Null | Date of booking |
| 7. | District | Varchar(20) | Not Null | District |
| 8. | Status | Varchar(20) | Not Null | Status of the service |

1. **Table : payment**

Primary key: **Payment\_id**

Foreign key: **Service\_id** references table **service**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NO | FIELD NAME | DATA TYPE | KEY CONSTRAINT | DESCRIPTION OF THE FIELD |
| 1. | Payment\_Id | Int(10) | Primary key, Auto increment | Payment Id |
| 2. | Service\_id | Int(10) | Foreign key | Service Id |
| 3. | Payment\_date | Date | Not Null | Date of payment |
| 4. | Status | Varchar(10) | Not Null | Payment Status |

1. **Table : feedback**

Primary key: **Rating\_id**

Foreign key: **Service\_id** references table  **service**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NO | FIELD NAME | DATA TYPE | KEY CONSTRAINT | DESCRIPTION OF THE FIELD |
| 1. | Rating\_id | Int(10) | Primary key, Auto increment | Rating Id |
| 2. | Service\_id | Int(10) | Foreign key | Service Id |
| 3. | Rating | Int(5) | Not Null | Rating of the service |
| 4. | Review | Varchar(300) | Not Null | Review of the service |

# `

**CHAPTER 5 SYSTEM TESTING**

## INTRODUCTION

Software testing is a way to check if the computer program works like it's supposed to. We use testing to make sure the software does what it is supposed to do. Validation means checking or testing things like software to make sure they meet the requirements and standards they are supposed to follow. Software testing is a way to check if a program works well. It goes along with other methods like checking and walking through the program. Validation means making sure that what the user wanted is what they got. There are several rules that can serve as testing objectives.

They are:

Testing is a process of executing a program with the intent of finding an error.

* + A good test case is one that has high possibility of finding an undiscovered error.
  + A successful test is one that uncovers an undiscovered error.

If a test works well and follows its goals, it can find mistakes in the software. The test showed that the computer program is working like it's supposed to and is doing well.

There are three ways to test program.

* + For correctness
  + For implementation efficiency
  + For computational complexity

## TEST PLAN

A test plan suggests several required steps that need be taken to complete various testing methodologies. The activity that is to be taken is outlined in the test plan. A computer program, its documentation, and associated data structures are all created by software developers. It is always the responsibility of the software developers to test each of the program's separate components to make sure it fulfills the purpose for which it was intended. To solve the inherent issues with allowing the builder evaluate what they have developed, there is an independent test group (ITG). Testing's precise goals should be laid forth in quantifiable language. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test.

The levels of testing include:

* + Unit testing
  + Integration Testing
  + Data validation Testing & Output Testing

### Unit Testing

Unit testing concentrates verification efforts on the software component or module, which is the smallest unit of software design. The component level design description is used as a guide when testing crucial control paths to find faults inside the module's perimeter. The level of test complexity and the untested area determined for unit testing. Unit testing is white-box focused, and numerous components may be tested simultaneously. To guarantee that data enters and exits the software unit under test properly, the modular interface is tested. To make sure that data temporarily stored retains its integrity during each step of an algorithm's execution, the local data structure is inspected. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested.

Before starting any other test, tests of data flow over a module interface are necessary. All other tests are irrelevant if data cannot enter and depart the system properly. An important duty during the unit test is the selective examination of execution pathways. Error circumstances must be foreseen in good design, and error handling paths must be put up to cleanly reroute or halt work when an error does arise. The final step of unit testing is boundary testing. Software frequently fails at its limits.

In the “MULTISERVICEPROVIDER” System, unit testing was carried out by treating each module as a distinct entity and subjecting them to a variety of test inputs. The internal logic of the modules had some issues, which were fixed. Each module is tested and run separately after coding. All unused code was eliminated, and it was confirmed that every module was functional and produced the desired outcome

### Integration Testing

Integration testing is a methodical approach for creating the program's structure while also carrying out tests to find interface issues. The goal is to construct a program structure that has been determined by design using unit tested components. The program is tested. Correction is challenging since the size of the overall program makes it challenging to isolate the causes. As soon as these mistakes are fixed, new ones arise, and the process repeats itself in an apparently unending cycle. All the modules were integrated after unit testing was completed in the system to check for an interface inconsistency. A distinctive program structure also developed when discrepancies in program structures.

### Validation Testing or System Testing

The testing process comes to an end here. This involved testing the entire system in its entirety, including all forms, code, modules, and class modules. Popular names for this type of testing include system tests and black box testing. The functional requirements of the software are the main emphasis of the black box testing approach. That example, using Black Box testing, a software engineer can create sets of input conditions that will fully test every program requirement. The following sorts of problems are targeted by black box testing: erroneous or missing functions, interface errors, data structure or external data access errors, performance errors, initialization errors, and termination errors.

### Output Testing or User Acceptance Testing

The system considered is tested for user acceptance; here it should satisfy the firm’s need. The software should keep in touch with perspective system; user at the time of developing and making changes whenever required.

This done with respect to the following points:

* + Input Screen Designs,
  + Output Screen Designs,

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future.

### Automation Testing

Automation Testing is a software testing technique that performs using special automated testing software tools to execute a test case suite. Essentially, it’s a test to double-check that the equipment or software does exactly what it was designed to do. It tests for bugs, defects, and any other issues that can arise with product development. Although some types of testing, such as regression or functional testing can be done manually, there are greater benefits of doing it automatically. Automation testing can be run at any time of the day. It uses scripted sequences to examine the software. It then reports on what has been found, and this information can be compared with earlier test runs. Automation developers generally write in the following programming languages: C#, JavaScript, and Ruby.

### Selenium Testing

Selenium is an open-source automated testing framework used to verify web applications across different browsers and platforms. Selenium allows for the creation of test scripts in various programming languages such as Java, C#, and Python. Jason Huggins, an engineer at Thought Works, developed Selenium in 2004 while working on a web application that required frequent testing. He created a JavaScript program called "JavaScriptTestRunner" to automate browser actions and improve testing efficiency. Selenium has since evolved and continues to be developed by a team of contributors. In addition to Selenium, another popular tool used for automated testing is Cucumber. Cucumber is an open-source software testing framework that supports behavior- driven development (BDD). It allows for the creation of executable specifications in a human- readable format called Gherkin. One of the advantages of using Cucumber is its ability to bridge the gap between business stakeholders and technical teams. By using a common language, Cucumber facilitates effective communication and collaboration during the testing process. It promotes a shared understanding of the requirements and helps ensure that the developed software meets the intended business goals. Cucumber can be integrated with Selenium to combine the benefits of both tools. Selenium is used for interacting with web browsers and automating browser actions, while Cucumber provides a structured framework for organizing and executing tests. This combination allows for the creation of end-to-end tests that verify the behavior of web applications across different browsers and platforms, using a business-readable and maintainable format.

### Example:

**Test Case 1 Code**

package definition;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver; import org.openqa.selenium.WebElement;

import org.openqa.selenium.By; import io.cucumber.java.en.Given; import io.cucumber.java.en.When; import io.cucumber.java.en.Then;

public class Loginsteps {

WebDriver driver;

@Given("browser is open") public void browserIsOpen() {

System.setProperty("webdriver.gecko.driver", "C:\\Users\\abhin\\OneDrive\\Desktop\\geckodriver.exe");

driver = new FirefoxDriver(); driver.manage().window().maximize();

}

@Given("user is on login page") public void userIsOnLoginPage() {

System.setProperty("webdriver.gecko.driver", "C:\\Users\\abhin\\OneDrive\\Desktop\\geckodriver.exe");

driver = new FirefoxDriver(); driver.manage().window().maximize();

String loginPageURL = "http://127.0.0.1:8000/signin/"; driver.get(loginPageURL);

}

@When("user enters username and password") public void userEntersUsernameAndPassword() {

// Locate the username and password fields and input data

WebElement usernameField = driver.findElement(By.id("username")); // Replace "username" with the actual ID of the username field

WebElement passwordField = driver.findElement(By.id("password")); // Replace "password" with the actual ID of the password field

usernameField.sendKeys("akash"); // Replace "your\_username" with the actual username passwordField.sendKeys("Akash`123"); // Replace "your\_password" with the actual password

}

@When("User clicks on login")

public void userClicksOnLogin() {

// Locate and click the login button

WebElement loginButton = driver.findElement(By.id("submit")); // Replace "login-button" with the actual ID of the login button

loginButton.click();

}

@Then("user is navigated to the home page") public void userIsNavigatedToHomePage() {

// Verify navigation to the home page by checking the URL, title, or elements on the home

page

String homePageURL = driver.getCurrentUrl();

String expectedHomePageURL = "http://127.0.0.1:8000/bookinghistory/userpage"; // Replace

with the expected URL of the home page

if (homePageURL.equals(expectedHomePageURL)) {

// Navigation to the home page is successful System.out.println("Successfully navigated to the home page");

} else {

// Navigation to the home page failed System.out.println("Failed to navigate to the home page");

}

// Remove the 'PendingException' or handle the logic here based on the verification

// throw new io.cucumber.java.PendingException();

}

### Screenshot

**Test Report**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case 1** | | | | | |
| **Project Name: MULTISERVICEPROVIDER** | | | | | |
| **Login Test Case** | | | | | |
| **Test Case ID:** Test\_1 | | | **Test Designed By:** Abhinand K S | | |
| **Test Priority (Low/Medium/High):** High | | | **Test Designed Date:** 30-12-2023 | | |
| **Module Name**: Login page | | | **Test Executed By :** Ms. Sona Maria Sebastian | | |
| **Test Title :** User Login | | | **Test Execution Date:** 02-12-2023 | | |
| **Description:** Verify login  with valid username and password | | |  | | |
| **Pre-Condition :**User has valid username and password | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/ Fail)** |
| 1 | Navigation to Login  Page |  | Login Page should be displayed | Login Page displayed | Pass |
| 2 | Provide valid Username | Username: akash | User should be | User is logged in and | Pass |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | Provide Valid Password | Password: Akash`123 | able to login | navigated to corresponding Home Page |  |
| 4 | Click on Login  button |  |
| **Post-Condition:** User is validated with database and successfully login into account. The Account session details are logged in database | | | | | |

### Test Case 2 Code

package definition;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver; import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.support.ui.ExpectedConditions; import org.openqa.selenium.support.ui.WebDriverWait;

import java.time.Duration;

import java.util.concurrent.TimeoutException; import org.junit.Assert;

import io.cucumber.java.en.Given; import io.cucumber.java.en.When; import io.cucumber.java.en.Then; import io.cucumber.java.en.And;

public class updateprofile {

WebDriver driver;

WebDriverWait wait;

@Given("the browser is open") public void browserIsOpen() {

System.setProperty("webdriver.gecko.driver", "C:\\Users\\abhin\\OneDrive\\Desktop\\geckodriver.exe");

driver = new FirefoxDriver(); driver.manage().window().maximize();

wait = new WebDriverWait(driver, Duration.ofSeconds(10));

}

@Given("the user is on the login page")

public void userIsOnLoginPage() {

String loginPageURL = "http://127.0.0.1:8000/signin/"; driver.get(loginPageURL);

}

@When("the user enters their username and password") public void userEntersUsernameAndPassword() {

WebElement usernameField = driver.findElement(By.id("username")); WebElement passwordField = driver.findElement(By.id("password"));

usernameField.sendKeys("akash"); // Replace with the correct username passwordField.sendKeys("Akash`123"); // Replace with the correct password

}

@When("the user clicks on the login button") public void userClicksOnLogin() {

WebElement loginButton = driver.findElement(By.id("submit")); loginButton.click();

}

@Then("the user is navigated to the home page") public void userIsNavigatedToHomePage() {

String homePageURL = driver.getCurrentUrl();

String expectedHomePageURL = "http://127.0.0.1:8000/bookinghistory/userpage";

Assert.assertEquals(expectedHomePageURL, homePageURL);

}

@When("the user navigates to the profile update page") public void theUserNavigatesToProfileUpdatePage() {

WebElement updateProfileLink = driver.findElement(By.id("updateProfileLink")); updateProfileLink.click();

}

@Then("the user should be on the profile update page") public void theUserShouldBeOnProfileUpdatePage() {

String expectedUrl = "http://127.0.0.1:8000/update\_profile/"; String currentUrl = driver.getCurrentUrl();

Assert.assertEquals(expectedUrl, currentUrl);

}

@And("the user updates the profile with the last name {string}")

public void theUserUpdatesProfileWithLastName(String newLastName) { WebElement lastNameField = driver.findElement(By.id("last\_name")); lastNameField.clear();

lastNameField.sendKeys(newLastName);

}

@And("the user updates the profile with the phone number {string}")

public void theUserUpdatesProfileWithPhoneNumber(String newPhoneNumber) {

WebElement phoneField = driver.findElement(By.id("phone")); phoneField.clear();

phoneField.sendKeys(newPhoneNumber);

}

// Existing steps...

@And("the user submits the updated profile") public void theUserSubmitsUpdatedProfile() {

WebElement updateButton = driver.findElement(By.xpath("//button[contains(text(),'Update Profile')]"));

updateButton.click();

WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(15)); // Adjust the duration as needed

try {

WebElement successMessage = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("success-message")));

Assert.assertTrue("Profile updated successfully message not displayed", successMessage.isDisplayed());

} catch (org.openqa.selenium.TimeoutException e) {

// Handle timeout exception if the success message doesn't appear within the specified time System.out.println("Timeout waiting for success message");

e.printStackTrace();

// You might want to handle this situation based on your test case requirements

}

}

@Then("the user should see the profile updated successfully") public void userShouldSeeProfileUpdatedSuccessfully() {

WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(15)); // Adjust the duration as needed

try {

WebElement successMessage = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("success-message")));

Assert.assertTrue("Profile updated successfully message not displayed", successMessage.isDisplayed());

} catch (org.openqa.selenium.TimeoutException e) {

// Handle timeout exception if the success message doesn't appear within the specified time System.out.println("Timeout waiting for success message");

e.printStackTrace();

// You might want to handle this situation based on your test case requirements

}

}

}

### Screenshot

**Test Report**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case 2** | | | | | |
| **Project Name: MULTISERVICEPROVIDER** | | | | | |
| **Update Profile Test Case** | | | | | |
| **Test Case ID:** Test\_2 | | | **Test Designed By:** Abhinand K S | | |
| **Test Priority (Low/Medium/High):** High | | | **Test Designed Date:** 02-12-2023 | | |
| **Module Name**: Update profile | | | **Test Executed By :** Ms. Sona Maria Sebastian | | |
| **Test Title :** Update profile | | | **Test Execution Date:** 03-12-2023 | | |
| **Description:** User updated user profile deatails | | |  | | |
| **Pre-Condition :**User has valid username and password | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected**  **Result** | **Actual**  **Result** | **Status(Pass/**  **Fail)** |
| 1 | Navigation  to Login Page |  | Login Page should be displayed | Login Page displayed | Pass |
| 2 | Provide valid Username | Username: akash | User should be | User is logged in and |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | Provide Valid Password | Password: Akash`123 | able to login  Profile page should be displayed | navigated to corresponding Home Page Profile page should be displayed | Pass  Pass |
| 4 | Navigation to Update profile page |  |
| 5 | Provide the update deatails | Lastname: Thomas Phone:  9876787678 |  |  |  |
| 6 | User clicks on update |  | Update successful message should  be displayed | Update successful message is  displayed | Pass |
| **Post-Condition:** The user's profile data has been successfully updated and that the system acknowledges the update with a relevant message. It also maintains the user's login session, allowing them to continue using the system with the updated details. | | | | | |

### Test Case 3 Code

package definition;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver; import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.support.ui.ExpectedConditions; import org.openqa.selenium.support.ui.WebDriverWait;

import java.time.Duration; import org.junit.Assert;

import io.cucumber.java.en.Given; import io.cucumber.java.en.When; import io.cucumber.java.en.Then; import io.cucumber.java.en.And;

public class search {

WebDriver driver;

WebDriverWait wait;

@Given("the browser is open") public void browserIsOpen() {

System.setProperty("webdriver.gecko.driver", "C:\\Users\\abhin\\OneDrive\\Desktop\\geckodriver.exe");

driver = new FirefoxDriver(); driver.manage().window().maximize();

wait = new WebDriverWait(driver, Duration.ofSeconds(10));

}

@And("the user is on the login page") public void userIsOnLoginPage() {

String loginPageURL = "http://127.0.0.1:8000/signin/"; driver.get(loginPageURL);

}

@When("the user enters their username and password") public void userEntersUsernameAndPassword() {

WebElement usernameField = driver.findElement(By.id("username")); WebElement passwordField = driver.findElement(By.id("password"));

usernameField.sendKeys("akash"); // Replace with the correct username passwordField.sendKeys("Akash`123"); // Replace with the correct password

}

@And("the user clicks on the login button") public void userClicksOnLogin() {

WebElement loginButton = driver.findElement(By.id("submit")); loginButton.click();

}

@Then("the user is navigated to the home page") public void userIsNavigatedToHomePage() {

String homePageURL = driver.getCurrentUrl();

String expectedHomePageURL = "http://127.0.0.1:8000/bookinghistory/userpage";

Assert.assertEquals(expectedHomePageURL, homePageURL);

}

@When("the user clicks on the search icon") public void userClicksOnSearchIcon() {

WebElement searchIcon = driver.findElement(By.id("search-icon")); // Replace with the correct locator for the search icon

searchIcon.click();

}

@Then("the search box appears") public void searchBoxAppears() {

WebElement searchBox = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("search-bar"))); // Replace with the correct locator for the search box

Assert.assertTrue("Search box is not visible", searchBox.isDisplayed());

}

@When("the user enters {string} in the search box")

public void userEntersTextInSearchBox(String searchText) {

WebElement searchBox = driver.findElement(By.id("search-bar")); // Replace with the

correct locator for the search box searchBox.sendKeys(searchText);

}

@And("the user clicks on the search button") public void userClicksOnSearchButton() {

WebElement searchButton = driver.findElement(By.id("search-submit")); // Replace with the correct locator for the search button

searchButton.click();

}

@Then("the user should be redirected to the search results page") public void userRedirectedToSearchResultsPage() {

String expectedUrl = "http://127.0.0.1:8000/search\_providers/?query=benzelectricals"; String currentUrl = driver.getCurrentUrl();

if (currentUrl.contains(expectedUrl)) {

System.out.println("User is redirected to the search results page.");

} else {

throw new AssertionError("User is not redirected to the expected search results page.");

}

}

}

### Screenshot

**Test Report**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case 3** | | | | | |
| **Project Name: MULTISERVICEPROVIDER** | | | | | |
| **Search Test Case** | | | | | |
| **Test Case ID:** Test\_3 | | | **Test Designed By:** Abhinand K S | | |
| **Test Priority (Low/Medium/High):** High | | | **Test Designed Date:** 02-12-2023 | | |
| **Module Name**: Search | | | **Test Executed By :** Ms. Sona Maria Sebastian | | |
| **Test Title :** Search provider | | | **Test Execution Date:** 03-12-2023 | | |
| **Description:** User updated user profile deatails | | |  | | |
| **Pre-Condition :**User has valid username and password | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | **Actual Result** | **Status(Pass/ Fail)** |
| 1 | Navigation to Login  Page |  | Login Page should be displayed | Login Page displayed | Pass |
| 2 | Provide valid Username | Username: akash | User should be able to login | User is logged in and navigated to corresponding Home Page | Pass |
| 3 | Provide Valid Password | Password: Akash`123 |
| 4. | User clicks on search icon |  | Search bar should be displayed | Search bar is displayed |  |
| 5 | Provide the search deatails | Search:benze lectrical |  |  |  |
| 6 | User clicks on Submit |  | Search result should be  displayed | Search result should is  displayed | Pass |
| **Post-Condition:** The search functionality successfully displays relevant results based on the provided search details while ensuring the user's continued logged-in session for further interaction with the system. | | | | | |

### Test Case 4 Code

package definition;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver; import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.support.ui.ExpectedConditions; import org.openqa.selenium.support.ui.WebDriverWait;

import java.time.Duration; import org.junit.Assert;

import io.cucumber.java.en.Given; import io.cucumber.java.en.When; import io.cucumber.java.en.Then; import io.cucumber.java.en.And;

public class pdfdownload {

WebDriver driver;

WebDriverWait wait;

@Given("the browser is open") public void browserIsOpen() {

System.setProperty("webdriver.gecko.driver", "C:\\Users\\abhin\\OneDrive\\Desktop\\geckodriver.exe");

driver = new FirefoxDriver(); driver.manage().window().maximize();

wait = new WebDriverWait(driver, Duration.ofSeconds(10));

}

@Given("the user is on the login page") public void userIsOnLoginPage() {

String loginPageURL = "http://127.0.0.1:8000/signin/"; driver.get(loginPageURL);

}

@When("the user enters their username and password") public void userEntersUsernameAndPassword() {

WebElement usernameField = driver.findElement(By.id("username")); WebElement passwordField = driver.findElement(By.id("password"));

usernameField.sendKeys("akash"); // Replace with the correct username passwordField.sendKeys("Akash`123"); // Replace with the correct password

}

@When("the user clicks on the login button") public void userClicksOnLogin() {

WebElement loginButton = driver.findElement(By.id("submit")); loginButton.click();

}

@Then("the user is navigated to the dashboard page") public void userIsNavigatedToDashboardPage() {

wait.until(ExpectedConditions.urlToBe("http://127.0.0.1:8000/bookinghistory/userpage")); String currentUrl = driver.getCurrentUrl(); Assert.assertEquals("http://127.0.0.1:8000/bookinghistory/userpage", currentUrl);

}

@When("the user clicks on the work report") public void userClicksOnWorkReport() {

WebElement workReportLink = driver.findElement(By.id("work-report-link")); workReportLink.click();

}

@Then("the user should be redirected to the work report page") public void userRedirectedToWorkReportPage() {

wait.until(ExpectedConditions.urlContains("http://127.0.0.1:8000/client\_work\_reports/7/")); String currentUrl = driver.getCurrentUrl();

Assert.assertTrue("User not redirected to the work report page", currentUrl.contains("http://127.0.0.1:8000/client\_work\_reports/7/"));

}

@And("the user clicks on download now") public void userClicksOnDownloadNow() {

WebElement downloadButton = driver.findElement(By.id("download-pdf-link")); downloadButton.click();

}

@Then("the PDF report should be downloaded successfully") public void pdfReportDownloadedSuccessfully() {

// Add code here to verify if the PDF is downloaded successfully

// This might involve checking the file system or a designated download location

// For example:

// Assert.assertTrue("PDF file not downloaded", isFileDownloaded("report.pdf"));

}

// Additional methods to check if the file is downloaded

// You might need to implement this based on your specific setup

// private boolean isFileDownloaded(String fileName) {

// File file = new File("path/to/download/directory/" + fileName);

// return file.exists();

// }

}

### Screenshot

**Test Report**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case 4** | | | | | |
| **Project Name: MULTISERVICEPROVIDER** | | | | | |
| **Download PDF Test Case** | | | | | |
| **Test Case ID:** Test\_4 | | | **Test Designed By:** Abhinand K S | | |
| **Test Priority (Low/Medium/High):** High | | | **Test Designed Date:** 02-12-2023 | | |
| **Module Name**: Download | | | **Test Executed By :** Ms. Sona Maria Sebastian | | |
| **Test Title :** Download workreport | | | **Test Execution Date:** 03-12-2023 | | |
| **Description:** User downloads the workreport given by worker | | |  | | |
| **Pre-Condition :**User has valid username and password | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected**  **Result** | **Actual**  **Result** | **Status(Pass/**  **Fail)** |
| 1 | Navigation  to Login Page |  | Login Page should be displayed | Login Page displayed | Pass |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2 | Provide valid Username | Username: akash | User should be able to login | User is logged in and navigated to corresponding Home Page | Pass |
| 3 | Provide Valid Password | Password: Akash`123 |
| 4. | User clicks on bookings |  | Client bookings should be displayed | Client bookings will be displayed |  |
| 5 | User clicks on the download button |  | Workreport file  should be downloaded | Workreport file  should isdownloaded | Pass |
| **Post-Condition:** the user effectively retrieves and downloads the work report file related to client bookings while maintaining an uninterrupted login session within the system. | | | | | |

# CHAPTER 6 IMPLEMENTATION

## INTRODUCTION

Implementation is the crucial phase during which the planned system materializes into a tangible, operational entity. Building user trust and confidence in the newly developed system is vital for its successful integration. The focus lies on user training and the creation of supportive resources. Conversion typically occurs during or after user training. In essence, implementation signifies the transition from a conceptual system design to its practical, functional existence. It involves the process of replacing or modifying existing systems to ensure a seamless shift towards the new system. Thorough planning, comprehensive system analysis, and the formulation of effective transition strategies are integral to the implementation process.

## IMPLEMENTATION PROCEDURES

During the software implementation process, the software is installed in its intended environment and functions as expected. In certain organizations, an individual not directly involved in software usage oversees and greenlights the project's development. Initial hesitations may arise, and it is crucial to address any uncertainties before they escalate into formidable resistance. It is essential to demonstrate to users why the new system is superior to its predecessor and to instill a sense of trust in the software. Users should receive comprehensive training to ensure their comfort and confidence with the application. To evaluate the outcome, one must ensure that the server program is operational on the server before assessing the system's functionality.

### User Training

User training is essential for imparting the necessary skills to individuals in utilizing and adapting to the system. It is imperative that users feel at ease and confident in their ability to navigate and operate the computer system effectively. Particularly when faced with complex functionalities, the significance of comprehensive user training becomes even more pronounced. This training equips users with the ability to input information, manage errors, query the database, and utilize various tools for generating reports and accomplishing other critical tasks.

### Training on the Application Software

Training on the application software involves an in-depth understanding of how to operate a new program after familiarizing oneself with the fundamentals of computer usage. This training aims to provide comprehensive guidance on navigating the new system, encompassing the utilization of various screens, accessing support resources, troubleshooting mistakes, and

rectifying errors. Its primary objective is to equip users with the requisite knowledge and skills

to effectively engage with the system, tailored to the specific needs and roles of different user groups.

### System Maintenance

System maintenance is a complex challenge within the sphere of system development. It encompasses vital tasks and ensures the effective functioning of software during the maintenance phase of the software life cycle. Once a system is operational, it requires ongoing attention to ensure its continued smooth operation. Prioritizing software maintenance throughout the development process is essential to enable the system to adapt to environmental changes. The practice of software maintenance extends beyond mere error detection within the code.

# CHAPTER 7 CONCLUSION AND FUTURE SCOPE

## CONCLUSION

The “MULTISERVICEPROVIDER” system represents a pivotal solution in revolutionizing service accessibility and delivery across diverse industries. Designed with a robust architecture and an intuitive interface, this system epitomizes efficiency in connecting clients with a wide array of services seamlessly. Its implementation signifies a leap forward from traditional service procurement methods, enhancing the simplicity and efficacy of managing service requests. The system empowers users to effortlessly register, access detailed service information, and efficiently manage bookings, while offering administrators streamlined control over user management and access privileges. Rigorous real-world testing has validated the system's alignment with specified project requirements. Continuous improvements and future enhancements are underway, ensuring the system remains adaptable and responsive to the ever- evolving landscape of service delivery.

## FUTURE SCOPE

In the foreseeable future of our project, several innovative enhancements are on the horizon to revolutionize the user, service provider, worker, and admin experiences. Users will witness a refined system, now equipped with advanced recommendation algorithms driven by machine learning, ensuring that on-demand services cater precisely to individual preferences. A chatbot powered by AI will serve as a personalized assistant, offering seamless assistance and information. Service providers will enjoy expanded functionalities, including multi-branch support, enhanced service registration processes, and nuanced payment options for workers, whether it's hourly wages, fixed rates, or commission-based. Real-time updates for workers regarding appointment changes, insightful analytics, and detailed task instructions will optimize their efficiency and performance. Admin tools will encompass sophisticated analytics to gauge service provider performance through user feedback and provide valuable insights for improvement. Additionally, geolocation services and robust notification systems will augment the entire ecosystem, heralding an era of elevated user engagement and efficiency in service delivery. These imminent advancements promise a future where our system becomes even more intuitive, personalized, and empowering for all stakeholders involved..

# CHAPTER 8 BIBLIOGRAPHY

### REFERENCES:

* Gary B. Shelly, Harry J. Rosenblatt, "System Analysis and Design", 2009 - Inspiring the comprehensive approach and systematic analysis employed in Urban Scout's design process.
* Andrew Hunt, David Thomas, "The Pragmatic Programmer: From Journeyman to Master", Pearson India, 1st Edition (2008) - Contributing to the emphasis on practical, efficient, and effective programming practices within Urban Scout's development.
* Ken Schwaber, Mike Beedle, "Agile Software Development with Scrum", Pearson (2008) - Influencing the adoption of agile software development methodologies, enabling Urban Scout to iterate and respond to changing requirements efficiently.
* Lisa Crispin, Janet Gregory, "Agile Testing: A Practical Guide for Testers and Agile Teams, Addison Wesley Professional, 1st Edition (2008) - Guiding the integration of agile testing

practices, ensuring continuous testing and quality assurance throughout Urban Scout's development lifecycle.

### WEBSITES:

* + [www.thumbtack.com](http://www.thumbtack.com/)
  + [www.taskrabbit.com](http://www.taskrabbit.com/)
  + [www.fiverr.com](http://www.fiverr.com/)
  + [www.angieslist.com](http://www.angieslist.com/)
  + www.koloapp.io

# CHAPTER 9 APPENDIX

## 9.1 Sample Code

### Clientbooking

{% load static %}

<!doctype html>

<html class="no-js" lang="en">

<!-- Mirrored from htmldemo.net/naturecircle/naturecircle/ by HTTrack Website Copier/3.x [XR&CO'2014], Sun, 01 Oct 2023 11:40:36 GMT -->

<head>

<meta charset="utf-8">

<meta http-equiv="x-ua-compatible" content="ie=edge">

<title>Multiservice provider</title>

<meta name="description" content="">

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Favicon -->

<link rel="shortcut icon" type="image/x-icon" href="{% static 'userassets/img/icon.png' %}">

<!-- All css here -->

<link rel="stylesheet" href="{% static 'userassets/css/bootstrap.min.css' %}">

<link rel="stylesheet" href="{% static 'userassets/css/font-awesome.min.css' %}">

<link rel="stylesheet" href="{% static 'userassets/css/ie7.css' %}">

<link rel="stylesheet" href="{% static 'userassets/css/plugins.css' %}">

<link rel="stylesheet" href="{% static 'userassets/css/style.css' %}">

<script src="{% static 'userassets/js/vendor/modernizr-3.5.0.min.js' %}"></script>

</head>

<style>

.project {

margin: 15px 0;

}

.no-gutter .project { margin: 0 !important;

padding: 0 !important;

}

.has-spacer { margin-left: 30px;

margin-right: 30px; margin-bottom: 30px;

}

.has-spacer-extra-space { margin-left: 30px; margin-right: 30px; margin-bottom: 30px;

}

.has-side-spacer { margin-left: 30px; margin-right: 30px;

}

.project-title {

font-size: 1.25rem;

}

.project-skill { font-size: 0.9rem; font-weight: 400;

letter-spacing: 0.06rem;

}

.project-info-box { margin: 15px 0; background-color: #fff; padding: 30px 40px; border-radius: 5px;

}

.project-info-box p { margin-bottom: 15px; padding-bottom: 15px;

border-bottom: 1px solid #d5dadb;

}

.project-info-box p:last-child { margin-bottom: 0;

padding-bottom: 0; border-bottom: none;

}

img {

width: 100%;

max-width: 100%; height: auto;

-webkit-backface-visibility: hidden;

}

.rounded {

border-radius: 5px !important;

}

.btn-xs.btn-icon { width: 34px; height: 34px;

max-width: 34px !important; max-height: 34px !important; font-size: 10px;

line-height: 34px;

}

/\* facebook button \*/

.btn-facebook, .btn-facebook:active, .btn-facebook:focus { color: #fff !important;

background: #4e68a1; border: 2px solid #4e68a1;

}

.btn-facebook:hover { color: #fff !important; background: #3b4f7a; border: 2px solid #3b4f7a;

}

.booking-form-box label,

.booking-form-box input,

.booking-form-box textarea { display: block;

margin-bottom: 10px; width: 600px;

}

.booking-form-box textarea { resize: vertical;

}

.booking-form-box button { background-color: #007bff;

color: white; border: none;

padding: 10px 20px; border-radius: 5px; cursor: pointer; transition-duration: 0.4s;

text-align: center; /\* Align the button text in the center \*/ display: block; /\* Make the button a block element \*/ margin: 0 auto; /\* Center the button horizontally \*/

}

.booking-form-box button:hover { background-color: #0056b3;

}

</style>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css" integrity="sha512-...your-sha512-hash..." crossorigin="anonymous" referrerpolicy="no-referrer" />

<body>

<!-- Header Area Start -->

<header class="header-area header-sticky">

<div class="header-container">

<div class="row">

<div class="col-lg-5 display-none-md display-none-xs">

<div class="ht-main-menu">

<nav>

<ul>

<li class="active"><a href="userpage.html">HOME </a>

</li>

<li><a href="">Services <i class="fa fa-angle-down"></i></a>

<ul class="ht-mega-menu">

<li>

<ul>

<li><a href="">House cleaning</a></li>

<li><a href="">Laundry</a></li>

<li><a href="">Painting</a></li>

<li><a href="">Gardening</a></li>

</ul>

</li>

<li>

<ul>

<li><a href="">Pest control</a></li>

<li><a href="">Plumbing</a></li>

<li><a href="">Repair</a></li>

<li><a href="">Electrical</li>

</ul>

</li>

<li>

</ul>

</li>

<li><a href="#bookings">Bookings</a></li>

<li><a href="#userprofile">Status</a></li>

<li><a href="{% url 'user\_logout' %}">Logout</a></li>

</ul>

</nav>

</div>

</div>

<div class="col-lg-2 col-sm-4">

<div class="logo text-center">

<a href="" class="logo-text">MULTISERVICE <span class="provider-text" style="padding-top: 30px;">PROVIDER</span></a>

</div>

</div>

<div class="col-lg-5 col-sm-8">

<div class="header-content d-flex justify-content-end">

<div class="settings-wrapper">

<a href="#"><i class="fas fa-bell"></i></a>

</div>

<div class="cart-wrapper">

<a href="#"><i class="fas fa-user"></i></a>

<span>{{request.user}}</span>

</div>

</div>

</div>

</div>

</div>

<!-- Header Area End -->

<!-- Mobile Menu Area Start -->

<div class="mobile-menu-area">

<div class="mobile-menu container">

<nav id="mobile-menu-active">

<ul class="menu-overflow">

<li><a href="userpage">HOME</a>

</li>

<li><a href="#">Services <i class="fa fa-angle-down"></i></a>

</li>

<li><a href="#bookings">Bookings</a></li>

<li><a href="#userprofile">User Profile</a></li>

<li><a href="#notifications">Notifications</a></li>

</ul>

</nav>

</div>

</div>

<!-- Mobile Menu Area End -->

</header>

<!-- Header Area End -->

<!-- Hero Area Start -->

<div class="ht-hero-section fix">

<div class="ht-hero-slider">

<!-- Single Slide Start -->

<div class="ht-single-slide" style="background-image: url('{% static 'userassets/img/bg/mainhead.jpg' %}')">

<div class="container">

<div class="content-section">

<!-- Your content goes here -->

<div class="container">

<!-- Your content elements and sections go here -->

<br> <br> <br> <br> <br> <br>

<div class="container">

validateForm()">

}}">

}}">

client\_district }}">

required>

<div class="booking-form-box">

<form method="post" action="{% url 'create\_booking' %}" onsubmit="return

{% csrf\_token %}

<!-- Hidden input fields for providerid and userid -->

<input type="hidden" id="providerid" name="providerid" value="{{ provider\_id

<input type="hidden" id="userid" name="userid" value="{{ client\_id }}">

<input type="hidden" id="clientphone" name="clientphone" value="{{ client\_phone

<input type="hidden" id="clientdistrict" name="clientdistrict" value="{{

<input type="hidden" id="name" name="name" value="{{ name }}">

<!-- Display providername and servicetype -->

<p>Provider: {{ providername }}</p>

<p>Service Type: {{ servicetype }}</p>

<!-- Fetch client instance and display the client's name -->

{% with client=request.user.client %}

<p>Your Name: {{ client.first\_name }} {{ client.last\_name }}</p>

{% endwith %}

<!-- Hidden input fields for providerid and userid -->

<!-- Other form elements here -->

<div class="form-group">

<label for="service\_date">Preferred Service Date:</label>

<input type="date" id="service\_date" name="service\_date" class="form-control"

</div>

<div class="form-group">

<label for="service\_time">Preferred Service Time:</label>

<input type="time" id="service\_time" name="service\_time" class="form- control" step="3600" min="08:00" max="18:00" required>

</div>

<button type="submit" class="btn btn-primary">Submit</button>

</form>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<!-- Single Slide End -->

<!-- Hero Area End -->

<!-- Food Categry Area Start -->

<!-- All js here -->

<script>

function validateForm() {

var serviceDate = document.getElementById("service\_date").value; var serviceTime = document.getElementById("service\_time").value;

// Validate service date (should not be empty and within a month from the current date) var currentDate = new Date();

var selectedDate = new Date(serviceDate); var oneMonthLater = new Date();

oneMonthLater.setMonth(currentDate.getMonth() + 1);

if (serviceDate.trim() === "" || selectedDate > oneMonthLater) { alert("Please select a valid service date within the next month."); return false;

}

// Extract hours and minutes from the serviceTime input var hours = parseInt(serviceTime.substring(0, 2));

// Validate service time (should not be empty and between 8am and 6pm) if (serviceTime.trim() === "" || hours < 8 || hours >= 18) {

alert("Please select a valid service time between 8am and 6pm."); return false;

}

return true;

}

</script>

<script src="{% static 'userassets/js/vendor/jquery-3.6.0.min.js' %}"></script>

<script src='{% static "userassets/js/bootstrap.min.js" %}'></script>

<script src="{% static 'userassets/js/plugins.js' %}"></script>

<script src="{% static 'userassets/js/ajax-mail.js' %}"></script>

<script src="{% static "userassets/js/main.js" %}"></script>

</body>

</html>

### Workreport and payment

{% load static %}

<!doctype html>

<html class="no-js" lang="en">

<!-- Mirrored from htmldemo.net/naturecircle/naturecircle/ by HTTrack Website Copier/3.x [XR&CO'2014], Sun, 01 Oct 2023 11:40:36 GMT -->

<head>

<meta charset="utf-8">

<meta http-equiv="x-ua-compatible" content="ie=edge">

<title>Multiservice provider</title>

<meta name="description" content="">

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Favicon -->

<link rel="shortcut icon" type="image/x-icon" href="{% static 'userassets/img/icon.png'

%}">

<!-- All css here -->

<link rel="stylesheet" href="{% static 'userassets/css/bootstrap.min.css' %}">

<link rel="stylesheet" href="{% static 'userassets/css/font-awesome.min.css' %}">

<link rel="stylesheet" href="{% static 'userassets/css/ie7.css' %}">

<link rel="stylesheet" href="{% static 'userassets/css/plugins.css' %}">

<link rel="stylesheet" href="{% static 'userassets/css/style.css' %}">

<script src="{% static 'userassets/js/vendor/modernizr-3.5.0.min.js' %}"></script>

</head>

<style>

.project {

margin: 15px 0;

}

.no-gutter .project { margin: 0 !important;

padding: 0 !important;

}

p {

font-family: "Barlow", sans-serif !important; font-weight: 300;

font-size: 1rem; color: #686c6d;

letter-spacing: 0.03rem; margin-bottom: 10px;

}

b, strong {

font-weight: 700 !important;

}

.border-secondary {

border-color: #d5dadb !important;

}

.opc-25 {

opacity: 0.25 !important;

}

.mb-30 {

margin-bottom: 30px !important;

}

.mt-30 {

margin-top: 30px !important;

}

.spacer-line-full-width, .spacer-line-fw { width: 100%;

line-height: 0;

border-bottom: 2px solid #f7f7f7;

}

/\* Apply standard font and add space \*/ a.logo-text {

font-family: Arial, sans-serif; /\* Use Arial or any standard sans-serif font \*/ font-size: 30px; /\* Set the font size \*/

font-weight: bold; /\* Make the text bold \*/

text-decoration: none; /\* Remove underline from the link \*/ color: white; /\* Set a standard text color \*/

text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3); /\* Apply text shadow for a cool effect \*/ margin-bottom: 35px;

}

/\* Style the provider text separately \*/ span.provider-text {

padding-top: 30px !important;

}

@font-face {

font-family: 'CustomFont'; src: url('path/to/font.woff');

}

.icon-Notification:before { font-family: 'CustomFont';

content: "\e123"; /\* Replace with the correct Unicode character for your notification icon

\*/

}

.provider {

border: 1px solid #ccc; padding: 10px;

margin-top: 10px; margin-bottom: 10px; border-radius: 5px;

background-color: #f9f9f9;

}

.provider strong { display: block; margin-bottom: 5px;

}

.provider-card {

background-color: rgba(255, 255, 255, 0.8); /\* Set background color with opacity \*/ border-radius: 15px; /\* Add curved borders \*/

margin: 10px; padding: 15px;

}

.report-list {

list-style: none; padding: 0;

}

.report-item {

margin-bottom: 20px; background-color: #fff; border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); padding: 20px;

}

.report-item ul { padding: 0;

margin: 0;

}

.report-item li {

margin-bottom: 10px;

}

.report-item a {

text-decoration: none; color: #007bff;

}

</style>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font- awesome/6.0.0/css/all.min.css" integrity="sha512-...your-sha512-hash..." crossorigin="anonymous" referrerpolicy="no-referrer" />

<body>

<!-- Header Area Start -->

<header class="header-area header-sticky">

<div class="header-container">

<div class="row">

<div class="col-lg-5 display-none-md display-none-xs">

<div class="ht-main-menu">

<nav>

<ul>

<li class="active"><a href="{% url 'userpage' %}">HOME </a></li>

</li>

<li><a href="">Services <i class="fa fa-angle-down"></i></a>

<ul class="ht-mega-menu">

<li>

<ul>

<li><a href="{% url 'service\_providers\_by\_category' category='Cleaning' %}">Cleaning</a></li>

<li><a href="{% url 'service\_providers\_by\_category' category='Laundry' %}">Laundry</a></li>

<li><a href="{% url 'service\_providers\_by\_category' category='Electrical' %}">Electrical</li>

<li><a href="{% url 'service\_providers\_by\_category' category='Plumbing' %}">Plumbing</a></li>

<li><a href="{% url 'service\_providers\_by\_category' category='Pestcontrol' %}">Pest control</a></li>

<li><a href="{% url 'service\_providers\_by\_category' category='Repair' %}">Repair</a></li>

endcomment %}

</ul>

</li>

</ul>

</li>

{% comment %} <li><a href="#bookings">Bookings</a></li> {%

<li><a href="{% url 'user\_logout' %}">Logout</a></li>

</ul>

</nav>

</div>

</div>

<div class="col-lg-2 col-sm-4">

<div class="logo text-center">

<a href="" class="logo-text">MULTISERVICE <span class="provider-text" style="padding-top: 30px;">PROVIDER</span></a>

</div>

</div>

<div class="col-lg-5 col-sm-8">

<div class="header-content d-flex justify-content-end">

<div class="search-wrapper">

<a href="#"><span class="icon icon-Search"></span></a>

<form action="#" class="search-form">

<input type="text" placeholder="Search entire store here ...">

<button type="button">Search</button>

</form>

</div>

<div class="settings-wrapper">

<a href="#"><i class="fas fa-bell"></i></a>

</div>

<div class="cart-wrapper">

<a href="#"><i class="fas fa-user"></i></a>

<span>{{request.user}}</span>

</div>

</div>

</div>

</div>

</div>

<!-- Header Area End -->

<!-- Mobile Menu Area Start -->

<div class="mobile-menu-area">

<div class="mobile-menu container">

<nav id="mobile-menu-active">

<ul class="menu-overflow">

<li><a href="userpage">HOME</a>

</li>

<li><a href="#">Services <i class="fa fa-angle-down"></i></a>

</li>

<li><a href="#bookings">Bookings</a></li>

<li><a href="#userprofile">User Profile</a></li>

<li><a href="#notifications">Notifications</a></li>

</ul>

</nav>

</div>

</div>

<!-- Mobile Menu Area End -->

</header>

<!-- Header Area End -->

<!-- Hero Area Start -->

<div >

<div >

<!-- Single Slide Start -->

<div style="background-image: url('{% static 'userassets/img/bg/mainhead.jpg' %}')">

<div class="container">

<div class="content-section">

<!-- Your content goes here -->

<div class="container">

<!-- Your content elements and sections go here -->

<br> <br> <br> <br> <br> <br>

<div class="container">

<h1>Your Work Report status</h1>

<ul class="report-list">

{% for report in client\_reports %}

<li class="report-item">

<ul>

<li>Worker Name: {{ report.user\_id.first\_name }} {{ report.user\_id.last\_name }}</li>

<li>Duration of Work: {{ report.duration\_of\_work }}</li>

<li>Requirements: {{ report.requirements }}</li>

<li>Cost: {{ report.cost }}</li>

<li>Workers needed: {{ report.num\_workers\_needed }}</li>

<li>payment status: {{ report.serviceid.paymentstatus }}</li>

<li>

<a href="{% url 'download\_work\_report' report\_id=report.reportid

%}" id="download-pdf-link">Download PDF</a>

report.reportid }}"> report.serviceid.serviceid }}">

{% if report.serviceid.status == 'reportgiven' %}

<li>

<form method="post" action="{% url 'approve\_report' %}">

{% csrf\_token %}

<input type="hidden" name="report\_id" value="{{

<input type="hidden" name="service\_id" value="{{

<button type="submit">Approve Report</button>

</form>

</li>

report.reportid }}">

{% endif %}

{% if report.serviceid.status == 'reportgiven' %}

<li>

<form method="post" action="{% url 'cancel\_service' %}">

{% csrf\_token %}

<input type="hidden" name="report\_id" value="{{

<input type="hidden" name="service\_id" value="{{

report.serviceid.serviceid }}">

<button type="submit">Cancel Service</button>

</form>

</li>

{% endif %}

{% if report.serviceid.paymentstatus == 'requested' %}

<li>

<form action="{% url 'payment\_success' %}" method="POST">

{% csrf\_token %}

<input type="hidden" name="service\_id" value="{{

report.serviceid.serviceid }}">

provider"

background-design\_1223-72.jpg"

<script src="https://checkout.razorpay.com/v1/checkout.js" data-key="rzp\_test\_bWe8tmmP1WXOe7"

data-amount="{{ report.cost }}" data-currency="INR"

data-order\_id="{{ payment.id }}" data-buttontext="Pay with Razorpay" data-name="Multiservice provider"

data-description="An efficient and reliable service

data-image="https://img.freepik.com/free-vector/payment- data-prefill.name="Abhinand K S"

data-prefill.email=["abhinandks2024a@mc](mailto:abhinandks2024a@mca.ajce.in)a[.ajce.in"](mailto:abhinandks2024a@mca.ajce.in) data-theme.color="#F37254">

</script>

</form>

</li>

{% endif %}

</ul>

</li>

{% empty %}

<li>No work reports available for this client.</li>

{% endfor %}

</ul>

</div>

</div>

</div>

</div>

</div>

</div>

<!-- Single Slide End -->

<!-- Hero Area End -->

<!-- Food Categry Area Start -->

<!-- All js here -->

<script>const costElements = document.querySelectorAll('.report-item li:nth-child(4)');

// Loop through each element and update the displayed cost costElements.forEach((element) => {

const originalCost = parseFloat(element.textContent.split(': ')[1]); const dividedCost = originalCost / 100;

element.textContent = `Cost: ${dividedCost}`;

});</script>

<script src="{% static 'userassets/js/vendor/jquery-3.6.0.min.js' %}"></script>

<script src='{% static "userassets/js/bootstrap.min.js" %}'></script>

<script src="{% static 'userassets/js/plugins.js' %}"></script>

<script src="{% static 'userassets/js/ajax-mail.js' %}"></script>

<script src="{% static "userassets/js/main.js" %}"></script>

</body>

</html>

**Search.html**

{% load static %}

<!doctype html>

<html class="no-js" lang="en">

<!-- Mirrored from htmldemo.net/naturecircle/naturecircle/ by HTTrack Website Copier/3.x [XR&CO'2014], Sun, 01 Oct 2023 11:40:36 GMT -->

<head>

        <meta charset="utf-8">

        <meta http-equiv="x-ua-compatible" content="ie=edge">

        <title>Multiservice provider</title>

        <meta name="description" content="">

        <meta name="viewport" content="width=device-width, initial-scale=1">

        <!-- Favicon -->

        <link rel="shortcut icon" type="image/x-icon" href="{% static 'userassets/img/icon.png' %}">

        <!-- All css here -->

        <link rel="stylesheet" href="{% static 'userassets/css/bootstrap.min.css' %}">

        <link rel="stylesheet" href="{% static 'userassets/css/font-awesome.min.css' %}">

        <link rel="stylesheet" href="{% static 'userassets/css/ie7.css' %}">

        <link rel="stylesheet" href="{% static 'userassets/css/plugins.css' %}">

        <link rel="stylesheet" href="{% static 'userassets/css/style.css' %}">

        <script src="{% static 'userassets/js/vendor/modernizr-3.5.0.min.js' %}"></script>

    </head>

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css" integrity="sha512-...your-sha512-hash..." crossorigin="anonymous" referrerpolicy="no-referrer" />

    <body>

        <!-- Header Area Start -->

        <header class="header-area header-sticky">

            <div class="header-container">

                <div class="row">

                    <div class="col-lg-5 display-none-md display-none-xs">

                        <div class="ht-main-menu">

                            <nav>

                                <ul>

                                    <li class="active"><a href="userpage.html">HOME </a>

                                    </li>

                                    <li><a href="">Services <i class="fa fa-angle-down"></i></a>

                                        <ul class="ht-mega-menu">

                                            <li>

                                                <ul>

                                                    <li><a href="">House cleaning</a></li>

                                                    <li><a href="">Laundry</a></li>

                                                    <li><a href="">Painting</a></li>

                                                    <li><a href="">Gardening</a></li>

                                                </ul>

                                            </li>

                                            <li>

                                                <ul>

                                                    <li><a href="">Pest control</a></li>

                                                    <li><a href="">Plumbing</a></li>

                                                    <li><a href="">Repair</a></li>

                                                    <li><a href="">Electrical</li>

                                                </ul>

                                            </li>

                                            <li>

                                        </ul>

                                    </li>

                                    <li><a href="#bookings">Bookings</a></li>

                                    <li><a href="#userprofile">Status</a></li>

                                    <li><a href="{% url 'user\_logout' %}">Logout</a></li>

                                </ul>

                            </nav>

                        </div>

                    </div>

                    <div class="col-lg-2 col-sm-4">

                        <div class="logo text-center">

                            <a href="" class="logo-text">MULTISERVICE <span class="provider-text" style="padding-top: 30px;">PROVIDER</span></a>

                        </div>

                    </div>

                    <div class="col-lg-5 col-sm-8">

                        <div class="header-content d-flex justify-content-end">

                            <div class="search-wrapper">

                                <a href="#"><span class="fas fa-search"></span></a>

                                <form action="{% url 'search\_providers' %}?user\_id={{ user.id }}" class="search-form" method="GET" onsubmit="return validateForm()">

                                  <input type="text" placeholder="Search provider here ..." name="query">

                                  <button type="submit">Search</button>

                              </form>

                            </div>

                            <div class="settings-wrapper">

                                <a href="#"><i class="fas fa-bell"></i></a>

                            </div>

                            <div class="cart-wrapper">

                                <a href="#"><i class="fas fa-user"></i></a>

                                <span>{{request.user}}</span>

                            </div>

                        </div>

                    </div>

                </div>

            </div>

            <!-- Header Area End -->

            <!-- Mobile Menu Area Start -->

            <div class="mobile-menu-area">

                <div class="mobile-menu container">

                    <nav id="mobile-menu-active">

                        <ul class="menu-overflow">

                            <li><a href="userpage">HOME</a>

                            </li>

                            <li><a href="#">Services <i class="fa fa-angle-down"></i></a>

                            </li>

                            <li><a href="#bookings">Bookings</a></li>

                            <li><a href="#userprofile">User Profile</a></li>

                            <li><a href="#notifications">Notifications</a></li>

                        </ul>

                    </nav>

                </div>

            </div>

            <!-- Mobile Menu Area End -->

        </header>

        <!-- Header Area End -->

        <!-- Hero Area Start -->

        <div class="ht-hero-section fix">

            <div class="ht-hero-slider">

                <!-- Single Slide Start -->

                <div class="ht-single-slide" style="background-image: url('{% static 'userassets/img/bg/mainhead.jpg' %}')">

                    <div class="container">

                      <div class="content-section">

                       <!-- Your content goes here -->

                        <div class="container">

                            <!-- Your content elements and sections go here -->

                            <br> <br> <br> <br> <br> <br>

                            <div class="container">

                              <ul>

                                {% if query %}

                                <h2>Search Results for "{{ query }}"</h2>

                                <div class="search-results-container">

                                {% if providers %}

                                    <ul>

                                        {% for provider in providers %}

                                            <li>

                                                Provider Name: {{ provider.providername }}<br>

                                                Owner Name: {{ provider.ownername }}<br>

                                                Contact Number: {{ provider.contact\_number }}<br>

                                                State: {{ provider.state }}<br>

                                                Service Type: {{ provider.service\_type }}<br>

                                                <a href="{% url 'render\_booking\_form' provider.user.userid %}">Book</a>

                                            </li>

                                        {% endfor %}

                                    </ul>

                                {% else %}

                                    <p>No providers found.</p>

                                {% endif %}

                            {% else %}

                                <p>Please enter a provider name to search.</p>

                            {% endif %}

                              </ul>

                            </div>

                          </div>

                        </div>

                    </div>

                    </div>

                </div>

            </div>

        </div>

        <!-- All js here -->

        <script src="{% static 'userassets/js/vendor/jquery-3.6.0.min.js' %}"></script>

        <script src='{% static "userassets/js/bootstrap.min.js" %}'></script>

        <script src="{% static 'userassets/js/plugins.js' %}"></script>

        <script src="{% static 'userassets/js/ajax-mail.js' %}"></script>

        <script src="{% static "userassets/js/main.js" %}"></script>

    </body>

</html>

### View.py

from django.shortcuts import render, redirect from django.contrib import messages

from django.contrib.auth import authenticate, login, get\_user\_model from django.contrib.auth.decorators import login\_required

from .models import MyUser,Client,Worker,ClientBooking,Service,WorkerReport from django.core.exceptions import ValidationError

from .views import \*

from django.views.decorators.cache import cache\_control from django.contrib.auth.views import

PasswordResetView,PasswordResetConfirmView,PasswordResetDoneView,PasswordResetCompleteView from django.urls import reverse\_lazy

from django.contrib.auth.hashers import make\_password from social\_django.models import UserSocialAuth

from django.contrib.auth.hashers import check\_password class CustomPasswordResetView(PasswordResetView):

template\_name = 'password\_reset\_form.html' # Your template for the password reset form email\_template\_name = 'password\_reset\_email.html' # Your email template for the password reset

email

success\_url = reverse\_lazy('resetpassworddone') # URL to redirect after successful form submission class CustomPasswordResetConfirmView(PasswordResetConfirmView):

template\_name = 'password\_reset\_confirm.html' # Your template for password reset confirmation form success\_url = reverse\_lazy('passwordresetcomplete') # URL to redirect after successful password reset

class CustomPasswordResetDoneView(PasswordResetDoneView):

template\_name = 'password\_reset\_done.html' # Your template for password reset done page class CustomPasswordResetCompleteView(PasswordResetCompleteView):

template\_name = 'password\_reset\_complete.html' # Your template for password reset complete page # Signin View

from django.contrib.auth import authenticate, login from .models import Client, ServiceProvider, Worker

def signin(request): request.session.flush()

if request.method == 'POST':

username = request.POST['username'] password = request.POST['password']

myuser = authenticate(request, username=username, password=password) if myuser is not None:

request.session['username'] = username login(request, myuser)

# Redirect to the corresponding user page and pass the user\_instance as context if myuser.role == "client":

return redirect('userpage') elif myuser.role == "provider":

return redirect('providerpage') elif myuser.role == "worker":

return redirect('workerpage') elif myuser.role == "admin":

return redirect('custom\_admin\_page')

else:

messages.error(request, "Incorrect Login. Please check your credentials.") return redirect('signin')

return render(request, 'signin.html')

# Register View

def register(request):

user = None # Initialize user variable outside the if block if request.method == 'POST':

# Get form data

firstname = request.POST.get('firstname') lastname = request.POST.get('lastname') username = request.POST.get('username', '') phoneno = request.POST.get('mobile\_no') state = request.POST.get('state')

dob = request.POST.get('dob') district = request.POST.get('district') email = request.POST.get('email')

password = request.POST.get('password') confirmpassword = request.POST.get('confirm\_password') role = "client" # Get selected role from the dropdown

# Check for empty fields

if not firstname or not lastname or not username or not phoneno or not state or not dob or not district or not email or not password or not confirmpassword or not role:

messages.error(request, 'All fields are required.') return render(request, 'signup.html')

# Check password match

if password != confirmpassword:

messages.error(request, 'Password does not match the confirm password.') return render(request, 'signup.html')

try:

if MyUser.objects.filter(username=username).exists(): messages.error(request, 'Username already taken.')

elif MyUser.objects.filter(email=email).exists(): messages.error(request, 'Email already taken.')

elif Client.objects.filter(phone=phoneno).exists(): messages.error(request, 'Phone number already taken.')

else:

# Create MyUser instance

hashed\_password = make\_password(password) User = get\_user\_model()

user = User.objects.create( username=username, email=email, password=hashed\_password, role=role

)

# Create Client instance associated with the created MyUser instance client = Client.objects.create(

first\_name=firstname, last\_name=lastname, username=username, email=email, dob=dob,

password=hashed\_password, phone=phoneno, district=district,

state=state, role=role, user=user

)

messages.success(request, 'Account successfully registered.')

return redirect('signin') # Redirect to signin page after successful registration

except ValidationError as e: messages.error(request, f'Error: {e}')

return render(request, 'signup.html') def user\_logout(request):

try:

del request.session['username'] request.session.flush()

except KeyError: pass

return redirect('signin')

def worker\_logout(request): try:

del request.session['username'] request.session.flush()

except KeyError: pass

return redirect('signin')

def provider\_logout(request): try:

del request.session['username'] request.session.flush()

except KeyError: pass

return redirect('signin')

def admin\_logout(request): try:

del request.session['username'] except KeyError:

pass

return redirect('index') def index(request):

request.session.flush()

return render(request, "index.html") def services(request):

request.session.flush()

return render(request, "servicelist.html") @login\_required

def userpage(request):

if 'username' in request.session:

username = request.session['username']

try:

user = MyUser.objects.get(username=username) if user.role == 'client':

# Check if the user exists in the Client model try:

client = Client.objects.get(user=user) user\_id = user.userid # Get the user ID

return render(request, 'userpage.html', {'user\_id': user\_id}) except Client.DoesNotExist:

messages.warning(request, "User profile not found. Please update your profile.") # Pass username and email to google\_profile\_update view

email = user.email

return redirect('google\_profile\_update', username=username, email=email,user=user)

else:

messages.error(request, "You don't have permission to access this page.") return redirect('signin')

except MyUser.DoesNotExist: messages.error(request, "User does not exist.")

return redirect('signin') # Redirect to signin page if user does not exist

else:

return render(request, 'userpage.html')

@login\_required

@cache\_control(no\_cache=True, must\_revalidate=True, no\_store=True, max\_age=0) def workerpage(request):

if 'username' in request.session:

username = request.session['username'] try:

user = MyUser.objects.get(username=username) if user.role == 'worker':

# Retrieve the Worker object associated with the logged-in user worker = Worker.objects.get(user=user)

return render(request, 'workerpage.html', {'worker': worker}) else:

messages.error(request, "You don't have permission to access this page.") return redirect('signin')

except MyUser.DoesNotExist: messages.error(request, "User does not exist.")

else:

messages.error(request, "Login failed. Please check your credentials.") return redirect('signin')

@login\_required

@cache\_control(no\_cache=True, must\_revalidate=True, no\_store=True, max\_age=0) def providerpage(request):

if 'username' in request.session:

username = request.session['username'] try:

user = MyUser.objects.get(username=username) if user.role == 'provider':

context = { 'user': user,

'provider\_id': user.userid # Pass the provider's ID to the template context

}

return render(request, 'providerpage.html', context) else:

messages.error(request, "You don't have permission to access this page.") except MyUser.DoesNotExist:

messages.error(request, "User does not exist.")

else:

messages.error(request, "Login failed. Please check your credentials.")

return redirect('signin') # Redirect should be here if the user is not logged in or doesn't have

from django.core.mail import send\_mail from django.contrib import messages

from django.shortcuts import render, redirect from django.shortcuts import get\_object\_or\_404 from .models import MyUser

from django.template.loader import render\_to\_string

from django.contrib.auth.decorators import login\_required def deactivate\_user(request, userid):

user = get\_object\_or\_404(MyUser, userid=userid) if user.is\_active:

user.is\_active = False user.save()

# Send deactivation email subject = 'Account Deactivation'

message = 'Your account has been deactivated by the admin.'

from\_email = 'abhinandks2024a@mca.ajce.in' # Replace with your email recipient\_list = [user.email]

html\_message = render\_to\_string('deactivation\_email.html', {'user': user}) send\_mail(subject, message, from\_email, recipient\_list, html\_message=html\_message)

return redirect('custom\_admin\_page')

def activate\_user(request, userid):

user = get\_object\_or\_404(MyUser, userid=userid) if not user.is\_active:

user.is\_active = True user.save()

subject = 'Account activated'

message = 'Your account has been activated.'

from\_email = 'abhinandks2024a@mca.ajce.in' # Replace with your email recipient\_list = [user.email]

html\_message = render\_to\_string('activation\_email.html', {'user': user}) send\_mail(subject, message, from\_email, recipient\_list, html\_message=html\_message)

return redirect('custom\_admin\_page') from django.core.mail import EmailMessage

from django.template.loader import render\_to\_string from django.conf import settings

from .forms import ClientProfileForm

from django.http import HttpResponseForbidden @login\_required

def provider\_registration(request):

if request.user.role == 'admin': # Assuming 'role' is the attribute in your User model indicating the user's role

if request.method == 'POST':

provider\_name = request.POST.get('providername')

provider\_email = request.POST.get('email') # Validate the input fields here if necessary

# Replace 'YOUR\_BASE\_URL' with the actual base URL of your website base\_url = 'http://127.0.0.1:8000/provider\_reg'

# Create a registration link

registration\_path = "register" # Relative path for registration registration\_link = f"{base\_url}/{register}"

# Render HTML content for the email

html\_message = render\_to\_string('provider\_registration\_email.html', { 'provider\_name': provider\_name,

'registration\_link': registration\_link

})

# Send HTML email to the provider's email subject = 'Provider Registration Link'

plain\_message = f"Click the following link to complete your registration: {registration\_link}" from\_email = settings.DEFAULT\_FROM\_EMAIL

email = EmailMessage(subject, plain\_message, from\_email, [provider\_email]) email.content\_subtype = "html"

email.send(fail\_silently=False)

# Redirect to a success page or display a success message return render(request, 'provider\_registration\_success.html')

else:

messages.error(request, "Login failed. Please check your credentials.")

return render(request, 'provider\_registration\_form.html') else:

return render(request, 'signin.html')

def provider\_reg(request): request.session.flush()

return render(request, "provider\_registration.html") from django.urls import reverse

def worker\_registration(request): if request.method == 'POST':

# Get data from the form

worker\_name = request.POST.get('workerrname') worker\_email = request.POST.get('email') base\_url = 'http://127.0.0.1:8000/worker\_reg'

# Validate the input fields here if necessary

# Generate registration link using reverse() function registration\_link = f"{base\_url}"

# Render HTML content for the email

html\_message = render\_to\_string('worker\_registration\_email.html', { 'worker\_name': worker\_name,

'registration\_link': registration\_link,

})

# Send HTML email to the worker's email subject = 'Worker Registration Link'

from\_email = settings.DEFAULT\_FROM\_EMAIL

email = EmailMessage(subject, html\_message, from\_email, [worker\_email]) email.content\_subtype = "html"

email.send(fail\_silently=False)

# Redirect to a success page or display a success message return render(request, 'worker\_registration\_success.html')

return render(request, 'worker\_registration\_form.html') def worker\_reg(request):

request.session.flush()

return render(request, "worker\_registration.html")

def providerlist(request): request.session.flush()

return render(request, "providerlist.html")

def signup\_redirect(request):

messages.error(request, "Something wrong here, it may be that you already have account!") return redirect("signin")

@login\_required

def book\_service(request):

return render(request, "book\_service.html")

from django.shortcuts import render, redirect from .models import ServiceProvider

from django.contrib import messages

from django.contrib.auth.hashers import make\_password def providerregister(request):

if request.method == 'POST': # Get form data

providername = request.POST.get('providername') ownername = request.POST.get('ownername') username = request.POST.get('username') password = request.POST.get('password')

state = request.POST.get('state') district = request.POST.get('district')

contact\_number = request.POST.get('contact\_no') email = request.POST.get('email')

service\_type = request.POST.get('service\_type') role = "provider"

# Check for empty fields

if not providername or not ownername or not username or not password or not state or not district or not contact\_number or not email or not service\_type:

messages.error(request, 'All fields are required.') return render(request, 'provider\_registration.html')

try:

if not email:

raise ValidationError("Email is required.") # Create MyUser instance

User = get\_user\_model()

user = User.objects.create\_user( username=username, email=email,

password=password, # Store the password (it will be hashed internally) role=role,

is\_active=False

)

# Create ServiceProvider instance associated with the created MyUser instance service\_provider = ServiceProvider.objects.create(

user=user, # Assign the MyUser instance to the user field of ServiceProvider providername=providername,

ownername=ownername,

password=make\_password(password), # Store the hashed password state=state,

username=username, email=email, district=district,

contact\_number=contact\_number, service\_type=service\_type, role=role,

)

messages.success(request, 'Registration request sent to admin for approval.') return redirect('signin') # Redirect to signin page after successful registration

except ValidationError as e: messages.error(request, f'Error: {e}')

return render(request, 'provider\_registration.html') def workerregister(request):

if request.method == 'POST': # Get form data

firstname = request.POST.get('firstname') lastname = request.POST.get('lastname') username = request.POST.get('username', '') phoneno = request.POST.get('mobile\_no') state = request.POST.get('state')

dob = request.POST.get('dob') district = request.POST.get('district') email = request.POST.get('email')

password = request.POST.get('password') confirmpassword = request.POST.get('confirm\_password') role = "worker" # Get selected role from the dropdown

providername = request.POST.get('providername') # Get providername from the form # Check for empty fields

if not firstname or not lastname or not username or not phoneno or not state or not dob or not district or not email or not password or not confirmpassword or not role or not providername:

messages.error(request, 'All fields are required.') return render(request, 'signup.html')

# Check password match

if password != confirmpassword:

messages.error(request, 'Password does not match the confirm password.') return render(request, 'signup.html')

# Get provider based on providername try:

provider = ServiceProvider.objects.get(providername=providername) User = get\_user\_model()

user = User.objects.create\_user( username=username, email=email,

password=password, # Store the password (it will be hashed internally) role=role,

is\_active=False

)

# Create Worker instance associated with the created MyUser instance and provider worker = Worker.objects.create(

user=user, first\_name=firstname, last\_name=lastname, dob=dob, email=email, phone=phoneno, district=district, state=state, role="worker", status="available",

provider=provider.user\_id # Assign the provider to the worker

)

messages.success(request, 'Registration request sent to provider for approval.') return redirect('signin') # Redirect to signin page after successful registration

except ServiceProvider.DoesNotExist: messages.error(request, 'Invalid provider name.') return render(request, 'signup.html')

# Create MyUser instance return render(request, 'signup.html')

@login\_required

def service\_providers\_by\_category(request, category):

providers = ServiceProvider.objects.filter(user is\_active=True, service\_type=category) context = {'providers': providers, 'category': category}

return render(request, 'providerlist.html', context)

#update profile

from django.contrib.auth.decorators import login\_required from django.shortcuts import render, redirect

from django.contrib import messages

from django.contrib.auth.hashers import make\_password from django.shortcuts import get\_object\_or\_404 @login\_required

def update\_profile(request):

client = Client.objects.get(user=request.user) if request.method == 'POST':

# Get data from the POST request

here

username = request.POST.get('username') # Assuming username is unique

email = request.POST.get('email') # Note: Email field is read-only in the form, no need to process it

first\_name = request.POST.get('first\_name') last\_name = request.POST.get('last\_name') phone = request.POST.get('phone')

try:

# Retrieve the Client object based on the username client = Client.objects.get(username=username)

# Update Client model fields client.first\_name = first\_name client.last\_name = last\_name client.phone = phone

client = client # Get the related User object client.username = username

client.email = email client.save()

# Update User model fields (assuming User is your custom User model) user = client.user # Get the related User object

user.username = username

user.email = email # Update email field if needed user.save()

messages.success(request, 'Profile updated successfully')

return redirect('update\_profile') # Replace 'login' with your desired redirect URL

except Client.DoesNotExist: messages.error(request, 'Client not found')

# Handle the error, redirect to an error page or show an error message as needed

return render(request, 'update\_profile.html',{'client': client}) @login\_required

def profile\_view(request):

client = Client.objects.get(user=request.user) # Assuming you have a Client model related to the User model

return render(request, 'profile\_view.html', {'client': client}) from django.contrib.auth.hashers import make\_password

def google\_authenticate(request): try:

user\_social = UserSocialAuth.objects.get(provider='google-oauth2', user=request.user) user = user\_social.user

except UserSocialAuth.DoesNotExist: user = request.user

user.role = 'client' user.save()

return redirect('userpage')

from django.shortcuts import get\_object\_or\_404 from .models import ServiceProvider

@login\_required

def admin\_requests(request):

provider\_requests = ServiceProvider.objects.filter(user is\_active=False)

return render(request, 'providerrequest.html', {'provider\_requests': provider\_requests}) @login\_required

def activate\_provider(request, user\_id):

# Retrieve the ServiceProvider instance associated with the request ID

provider\_request = get\_object\_or\_404(ServiceProvider, user\_id=user\_id, user is\_active=False)

# Set is\_active to True and save the user provider\_request.user.is\_active = True provider\_request.user.save()

subject = 'Account approved'

message = 'Your account has been approved .'

from\_email = 'abhinandks2024a@mca.ajce.in' # Replace with your email recipient\_list = [provider\_request.user.email]

html\_message = render\_to\_string('activation\_email.html', {'user': provider\_request.user}) send\_mail(subject, message, from\_email, recipient\_list, html\_message=html\_message) return redirect('admin\_requests')

# Return a success response (you can customize the response as needed) from django.shortcuts import render, get\_object\_or\_404

from .models import ServiceProvider @login\_required

def render\_booking\_form(request, userid=None): # Get the current logged-in user

current\_user = request.user

# Fetch client ID based on user role client\_id = None

client\_name= None client\_phone = None client\_district = None

if current\_user.role == 'client': client\_id = current\_user.userid

client\_phone = current\_user.client.phone client\_district = current\_user.client.district client\_name = current\_user.client.username

# Process the provider ID from the URL parameters if userid is not None:

try:

provider\_id = int(userid)

# Fetch provider information from the database

provider = get\_object\_or\_404(ServiceProvider, user\_id=provider\_id)

# Include providername and servicetype in the context dictionary context = {

'client\_id': client\_id, 'client\_phone': client\_phone, 'provider\_id': provider\_id, 'client\_district': client\_district,

'providername': provider.providername, 'servicetype': provider.service\_type, 'name':client\_name

}

return render(request, 'book\_service.html', context)

except ValueError:

# Handle the case where provider\_id is not a valid integer

# You can raise an error, redirect, or handle it as per your requirement pass

return render(request, 'book\_service.html', {'client\_id': client\_id, 'client\_phone': client\_phone,'name': client\_name, 'provider\_id': provider\_id, 'client\_district': client\_district})

@login\_required

def create\_booking(request):

if request.method == 'POST':

name = request.POST.get('name')

service\_date = request.POST.get('service\_date') service\_time = request.POST.get('service\_time')

provider\_id = request.POST.get('providerid') # Get providerid from the form data client\_id = request.POST.get('userid')

client\_phone = request.POST.get('clientphone')

client\_district = request.POST.get('clientdistrict') # Get userid from the form data new\_booking = ClientBooking(

name=name, phone=client\_phone, district=client\_district, date=service\_date, time=service\_time,

providerid\_id=provider\_id, # Save providerid in the ClientBooking object clientid\_id=client\_id,

status="pending" # Save userid in the ClientBooking object

)

new\_booking.save()

messages.success(request, 'Appointment booked successfully.') return redirect('userpage')

else:

# Handle GET request (if needed) # ...

return render(request, 'signup') from django.db.models import Q

@login\_required

def search\_providers(request):

query = request.GET.get('query', '') # Get the query parameter from the request, default to an empty string if not present

client\_id = request.session.get('client\_id') # Assuming client\_id is stored in the session # Use Q objects to perform case-insensitive similar name search

providers = ServiceProvider.objects.filter(Q(providername iexact=query) |

Q(providername icontains=query))

context = {

'query': query, # Pass the query back to the template to display in the search results 'providers': providers,

'client\_id': client\_id, # Pass the client ID in the context

}

return render(request, 'search\_results.html', context) @login\_required

def provider\_bookings(request):

provider\_id = request.user.userid

bookings = ClientBooking.objects.filter(providerid=provider\_id).exclude(status='completed')

context = {

'bookings': bookings

}

return render(request, 'provider\_bookings.html', context) @login\_required

def bookinghistory(request):

# Get the logged-in provider's user ID provider\_id = request.user.userid

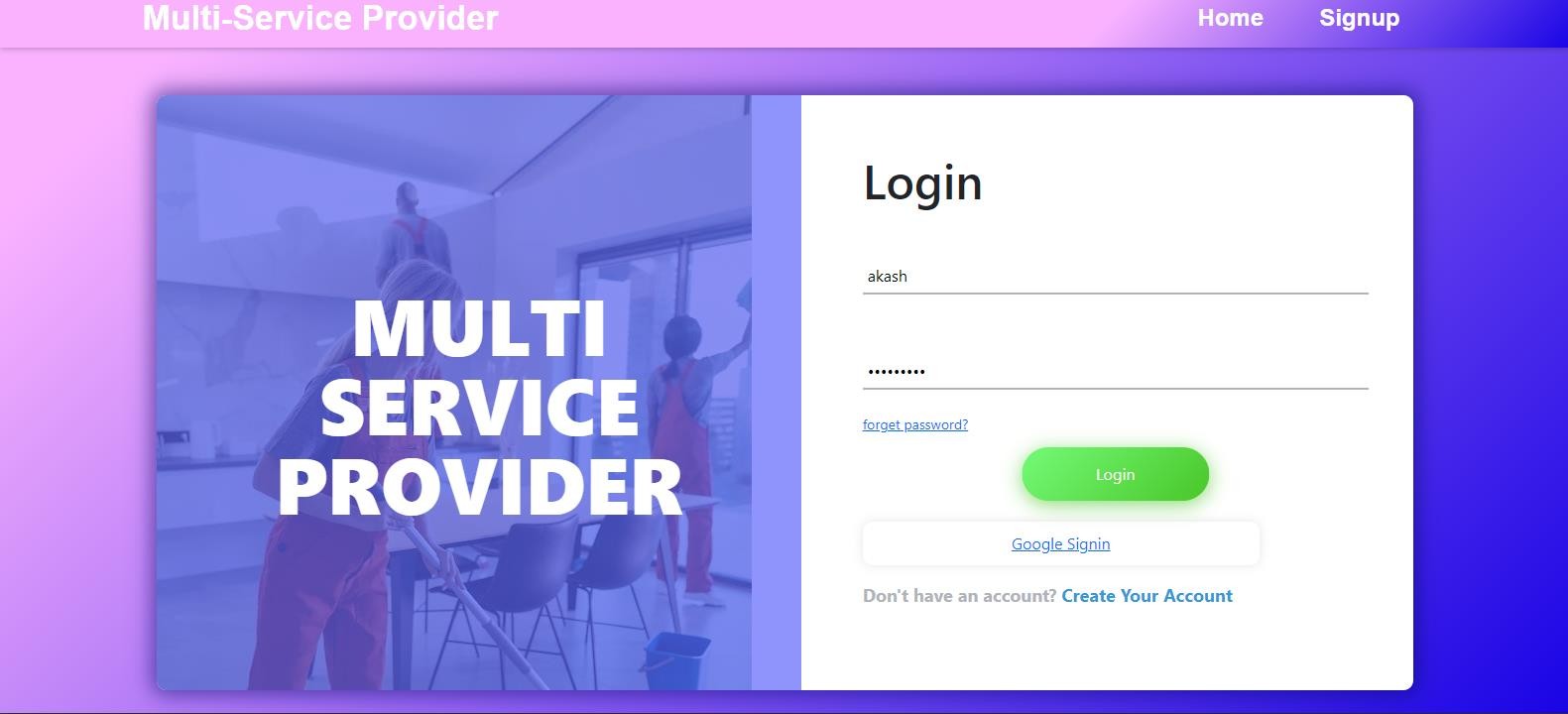
# Filter bookings based on the provider's providerid and status is 'pending' bookings = Service.objects.filter(providerid\_id=provider\_id, status='Completed') context = {

'bookings': bookings

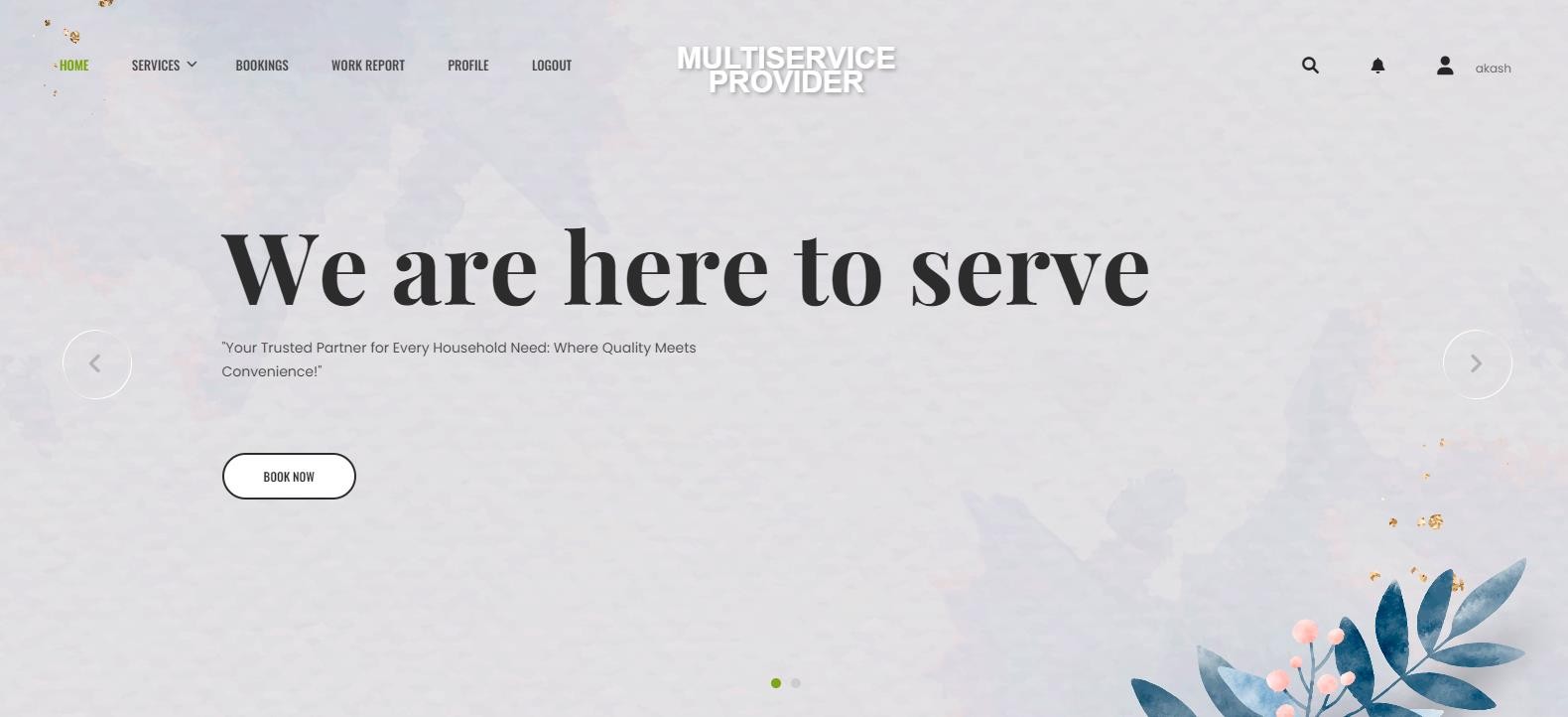
}

return render(request, 'bookinghistory.html', context)

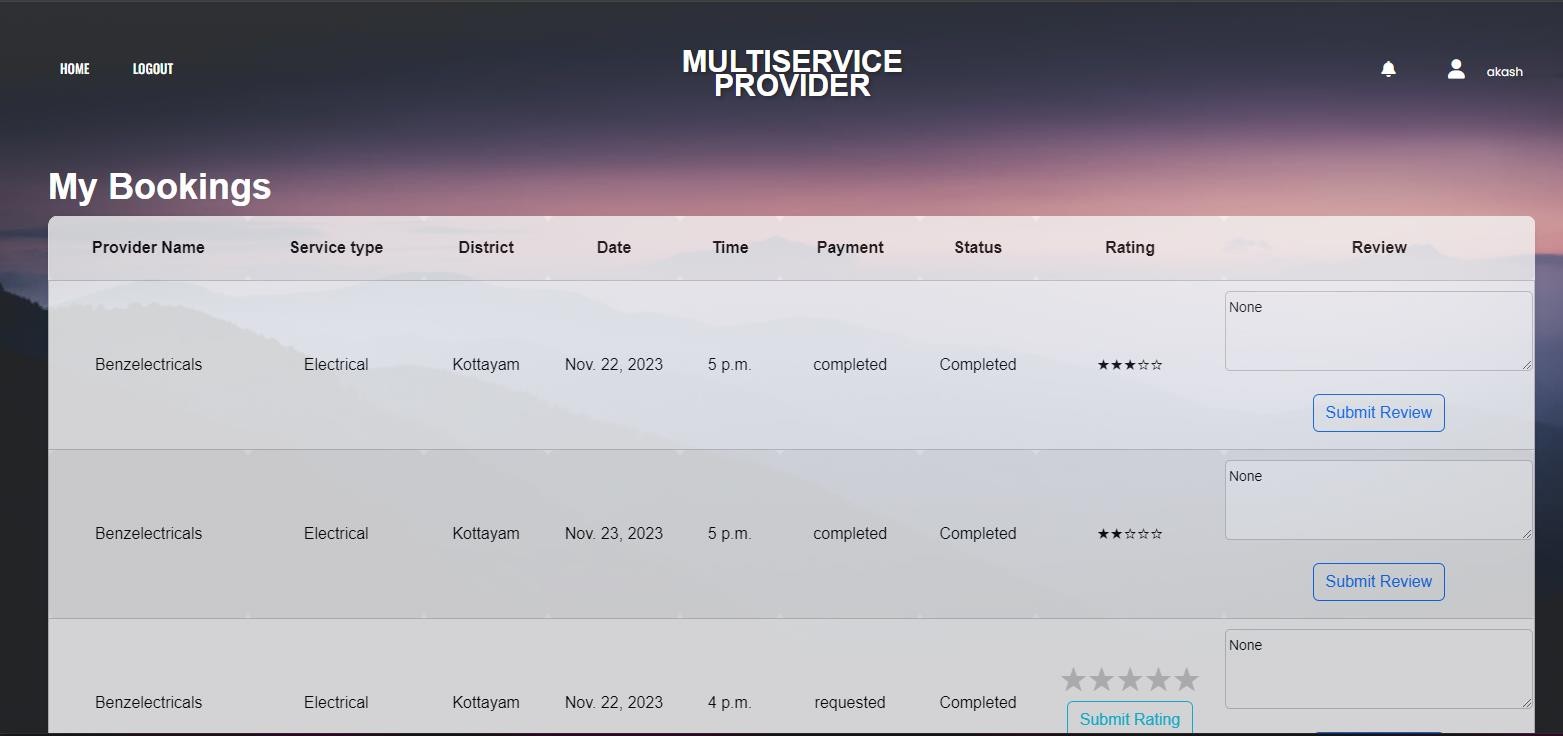
## 9.1 Screen Shots

**Login page**

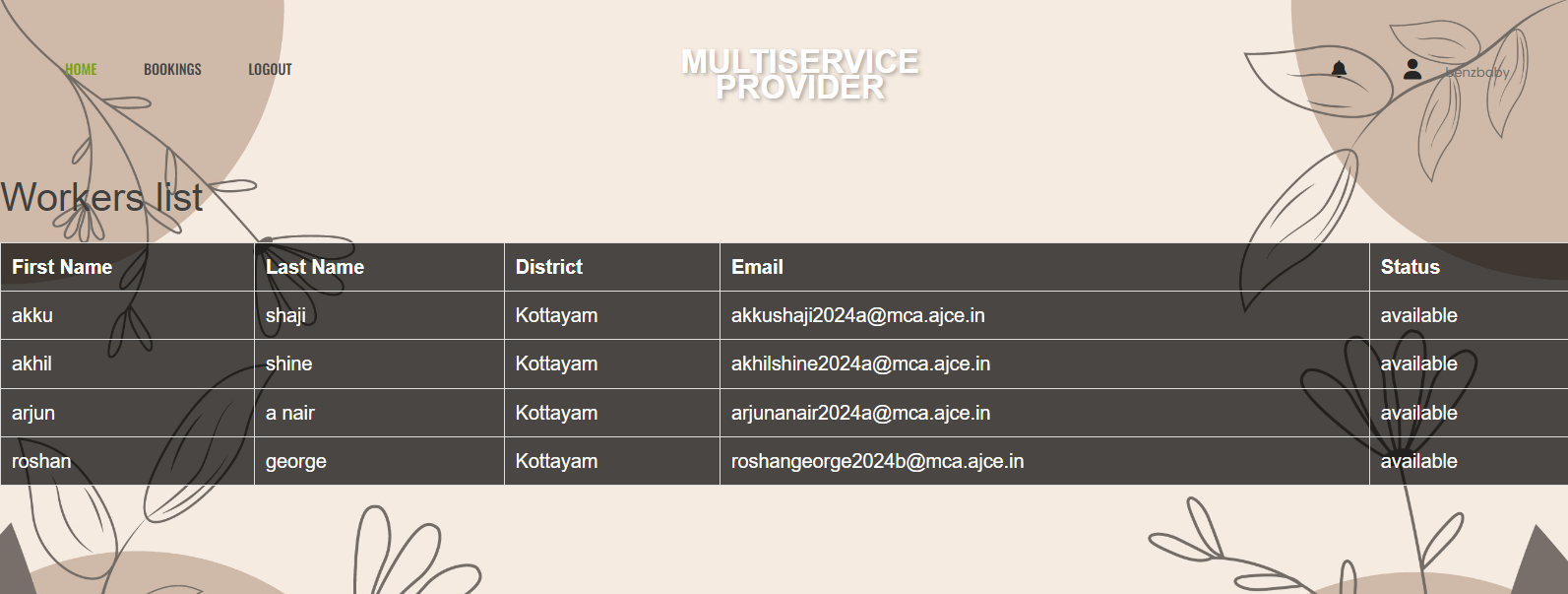
## Client page



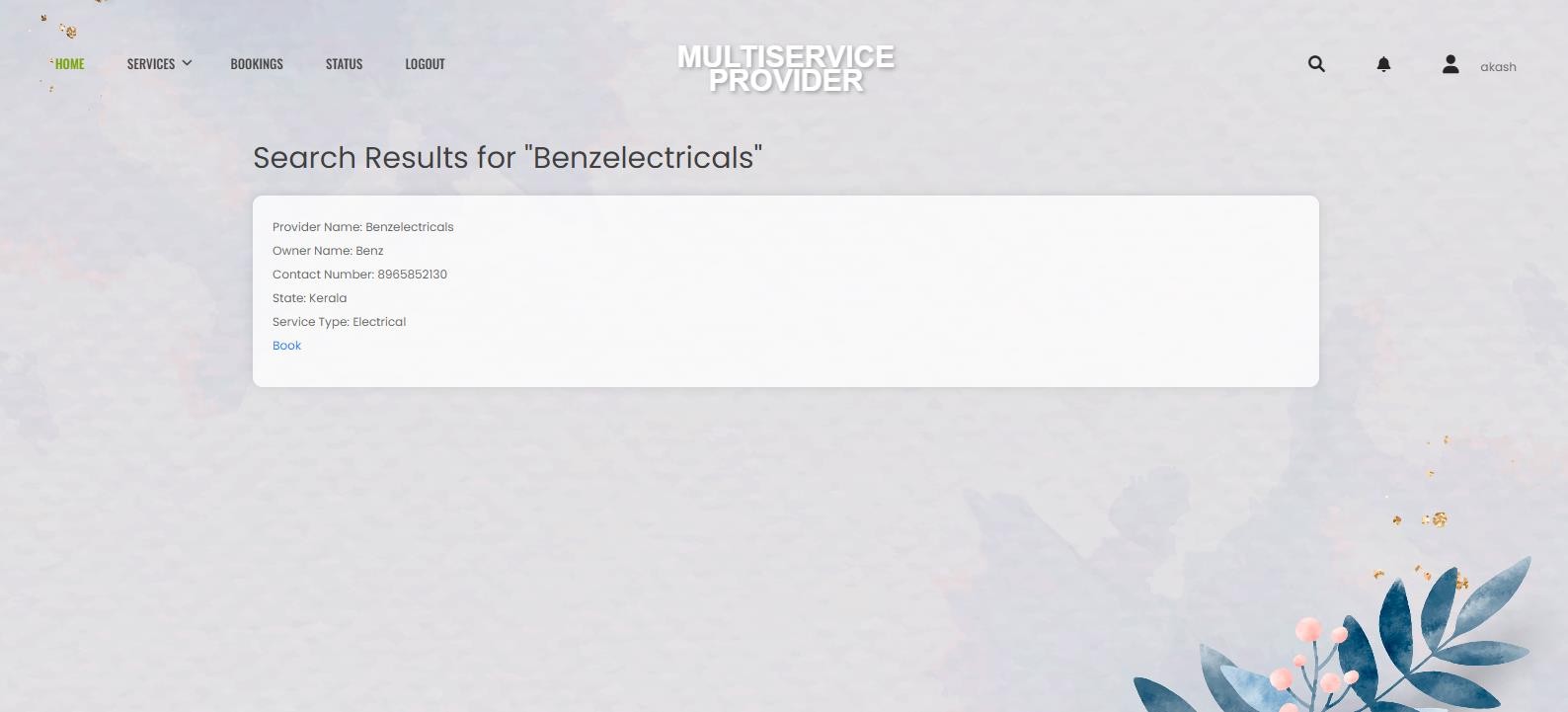
**Client bookings**



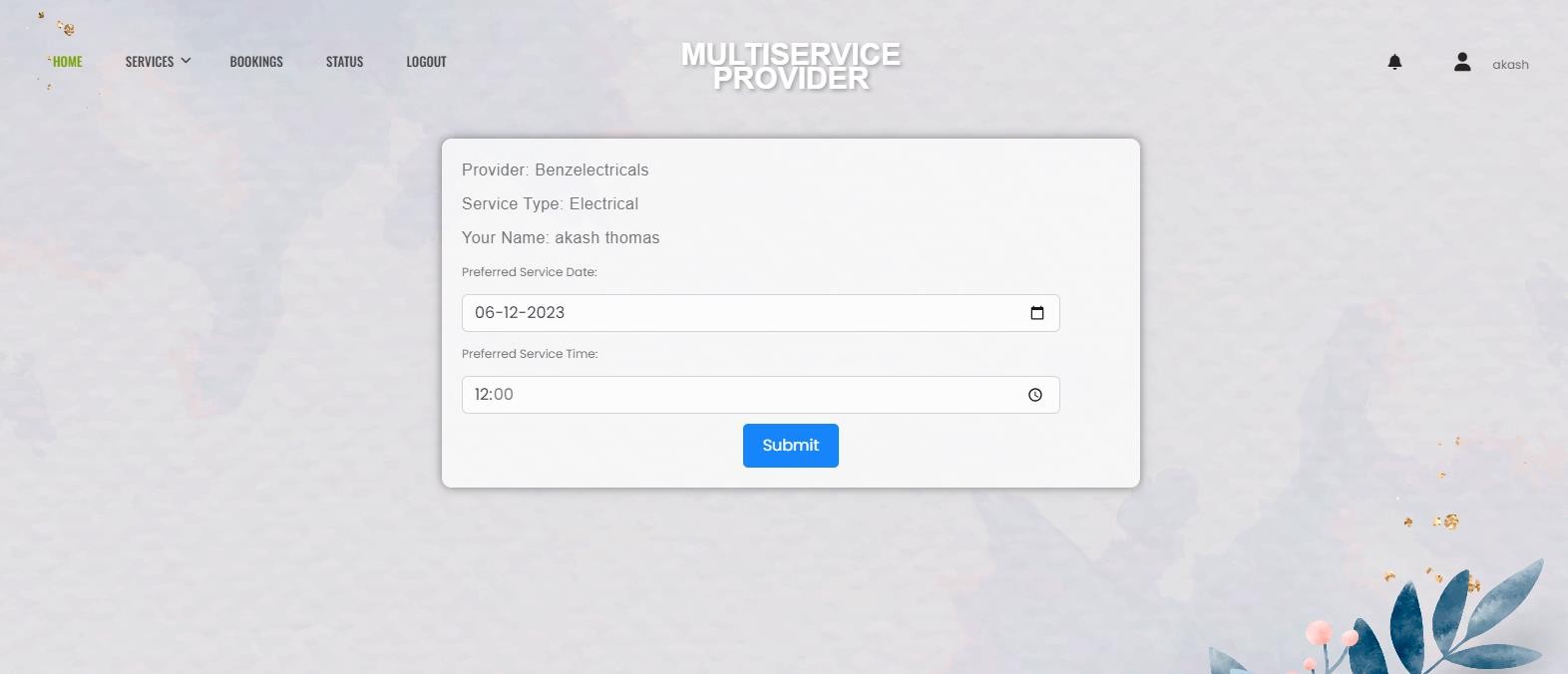
**Worker List**

****

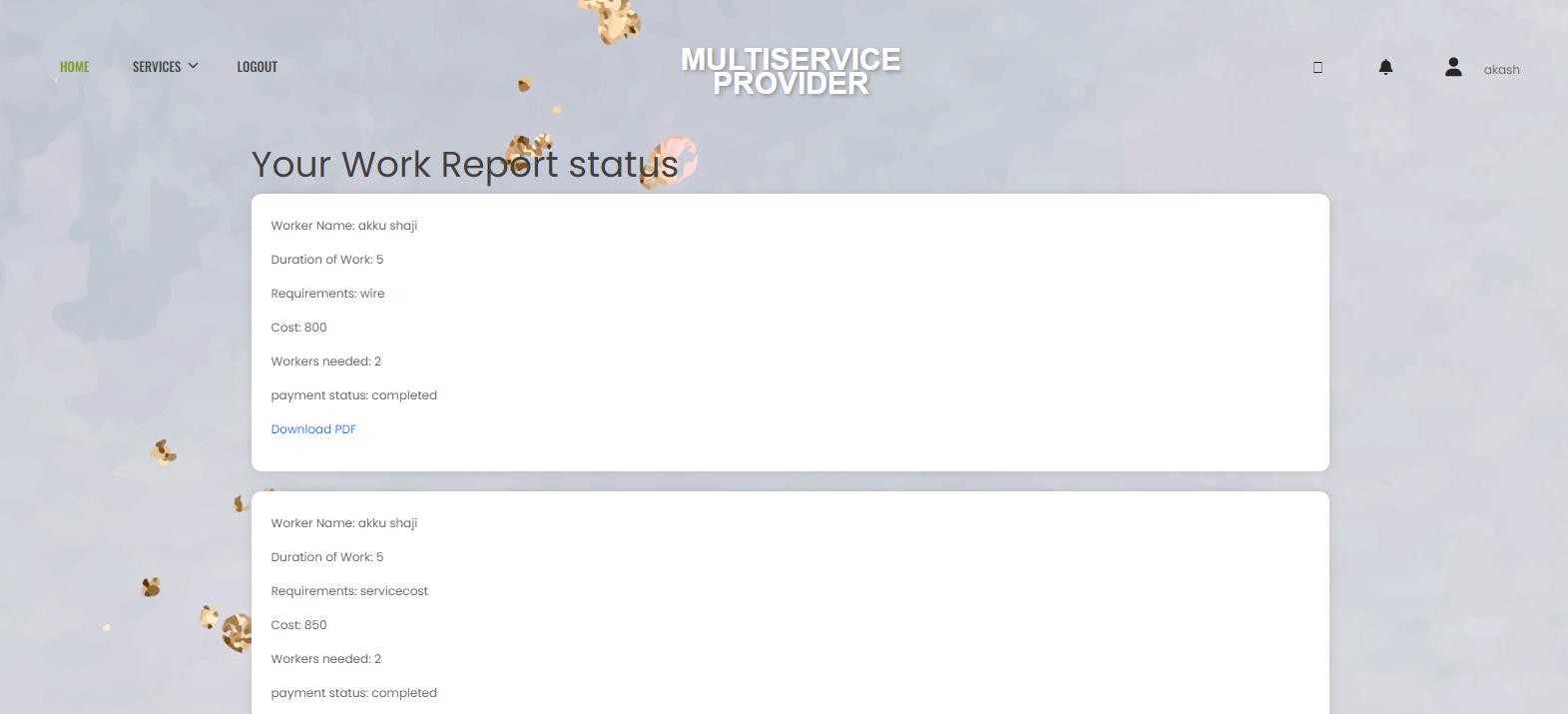
## Provider Search



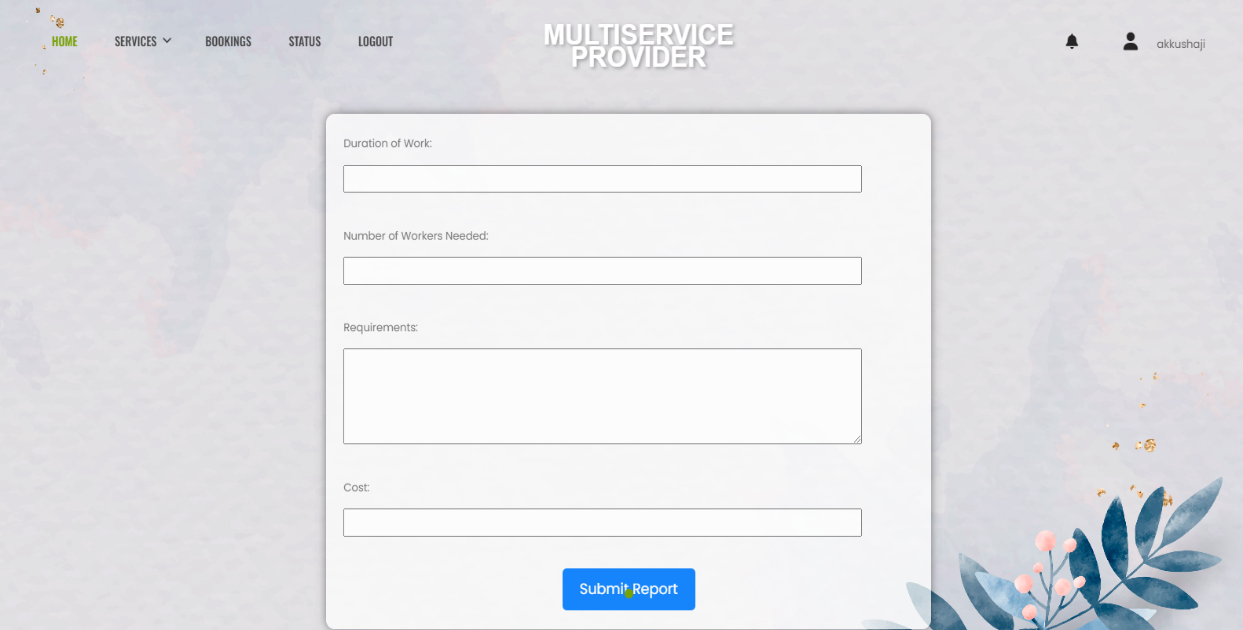
**Book service**



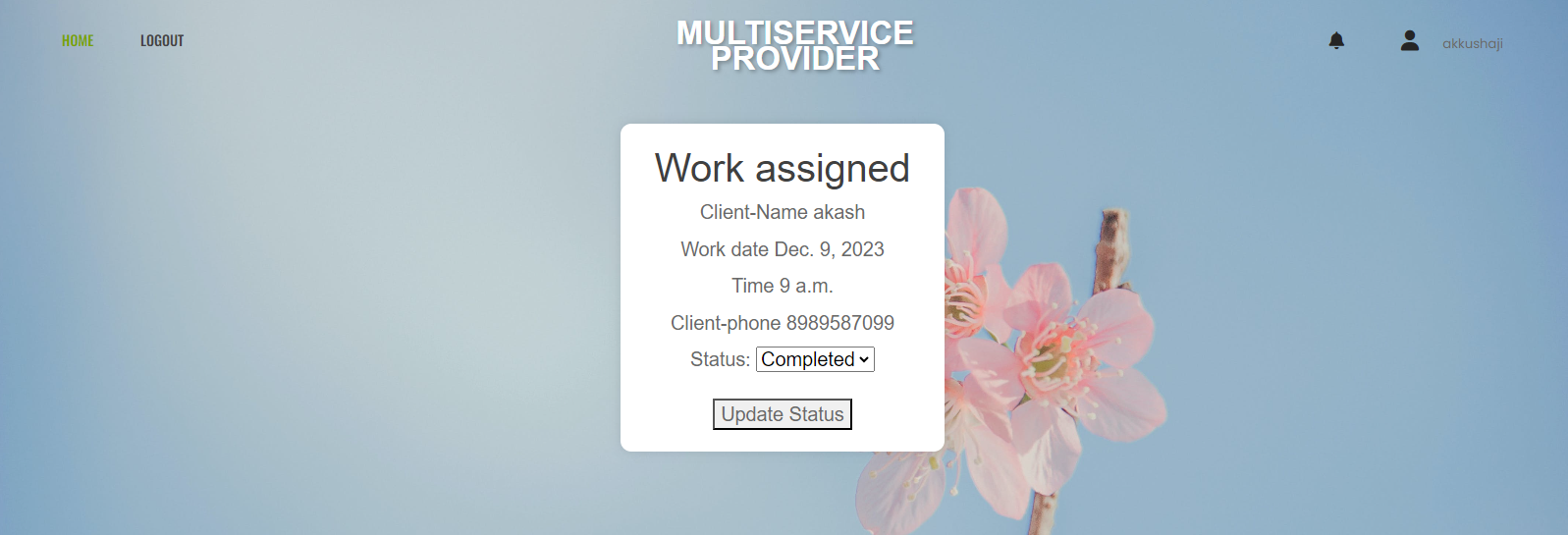
### Work report



**Work Report Form**

****

**Work Status Update**

****