CHAPTER 1
INTRODUCTION

LabourHub: Workers Supply Management is a comprehensive digital platform designed to streamline and enhance the efficiency of managing labour supply and demand. The current system for managing labour resources often relies on manual processes, leading to inefficiencies, lack of transparency, and difficulties in tracking worker availability and job assignments. This project aims to replace traditional methods with an automated solution that ensures accurate workforce management, real-time notifications, and improved operational efficiency.

The primary focus of LabourHub is to connect workers with potential employers while incorporating a structured approval and tracking system. Through a digital notification system, both workers and employers are kept informed about job assignments, availability, and status updates. Employers can request workers based on their requirements, and once approved, workers receive notifications regarding job assignments. This structured process eliminates confusion, ensures transparency, and enhances workforce management. Additionally, worker attendance and job records are systematically maintained, allowing for easy tracking and data retrieval.

LabourHub is developed using the Python Django framework, ensuring scalability, security**,** and high performance. This technology offers high scalability, robust performance, and strong security features. The system also implements role-based access control, ensuring that only designated users—such as administrators, employers, and workers—can access specific functionalities. By digitizing the labour supply management process, this project significantly reduces administrative workload, improves communication, and enhances the efficiency of workforce allocation.

Furthermore, LabourHub integrates a secure payment system to facilitate smooth financial transactions between customers and workers. Customers can make payments through the platform, ensuring transparency and accountability in financial dealings. Once a worker completes a task and submits a service report, the admin reviews and approves the work before

releasing the payment. This structured approach ensures fair compensation for workers while maintaining customer satisfaction and trust in the system.

To enhance user experience, LabourHub incorporates real-time tracking and notifications, allowing customers to monitor the progress of their job requests. Workers, in turn, receive instant alerts about new assignments, changes in job status, and payment approvals. This feature ensures that all stakeholders remain updated, reducing delays and misunderstandings in the job allocation process. The system also supports efficient booking management, allowing customers to schedule workers based on skillset, location, and availability.

1.1 SCOPE OF THE WORK

The primary objective of the LabourHub project is to digitize and automate the process of worker supply management, thereby improving efficiency, communication, and overall operational effectiveness. This system serves as a centralized platform that allows administrators to manage job postings, assign tasks, monitor worker performance, and track job status in real-time. The platform aims to reduce manual work, eliminate errors, and provide a streamlined experience for both administrator and workers. By automating job assignments and progress tracking, LabourHub enhances the coordination between workers and administrators, creating a more organized and error-free process.

Features and Functionalities

LabourHub incorporates several advanced features to address the inefficiencies of traditional workforce management. The Worker Management Module allows administrators to add, view, and edit worker details digitally. Instead of manually tracking job assignments, the system automatically notifies workers about new tasks and updates. Workers can update their task progress in real-time, ensuring continuous communication and transparency. The Job Assignment System enables administrators to post available jobs, allocate tasks based on workers' skillsets, and monitor progress through a secure dashboard. Furthermore, the system allows workers to track their job status and receive instant updates about their tasks. The Admin Panel provides administrators with access to all worker and job data, with the ability to generate detailed reports for analysis. The Customer Module allows customers to place job requests, track their orders, and provide feedback on completed tasks, ensuring their needs are met and

their expectations are managed. The platform also includes secure authentication mechanisms, ensuring that only authorized personnel can access sensitive data.

Technologies Used

LabourHub is built using the Python Django framework, providing a robust and secure backend for handling all operations efficiently. The frontend is designed with user experience in mind, ensuring an intuitive interface for both administrators, workers, and customers. The system employs a relational database to store worker profiles, job assignments, customer requests, and progress reports securely. Additionally, the platform integrates automated email notifications to keep workers informed about new job assignments, updates, and changes, while customers receive status updates and notifications about their requests. This ensures that communication between all parties is timely and accurate. The system is highly scalable, allowing it to adapt to the growing needs of businesses and integrate future.

Expected Outcomes

The LabourHub project is expected to bring several improvements to workforce management processes. A key outcome is the increase in operational efficiency, as manual job assignment and tracking processes are automated, reducing the risk of errors and delays. Real-time communication between administrators, workers, and customers is enhanced, ensuring that tasks are completed on time and without confusion. The system also ensures more accurate data tracking, enabling administrators to generate reports and analyze performance metrics easily. Another advantage is scalability—LabourHub can be expanded to include new features, such as mobile app integration or enhanced reporting capabilities, to meet the evolving needs of businesses. By transitioning from a traditional manual approach to a modern digital solution, LabourHub aims to streamline worker management, increase productivity, and enhance overall business operations. This comprehensive system not only facilitates better coordination between administrators and workers but also ensures a seamless interaction for customers, leading to increased satisfaction and business growth.

Moreover, the adoption of the Python Django framework ensures that the system is scalable and adaptable to future technological advancements. As the system evolves, it can integrate additional functionalities, to further enhance workforce management. This project serves as a future-ready solution for businesses and organizations looking to improve their worker supply management processes, increase efficiency, and optimize task coordination.

CHAPTER 2
PROOF OF CONCEPT

2.1 REVIEW OF LIERATURE

The adoption of digital labour management platforms has revolutionized the traditional approach to workforce handling by introducing automation, real-time monitoring, and secure data management. Numerous research studies have examined the comparative advantages of these systems over manual methods in areas such as task assignment, performance tracking, and secure access control.

[1] **Digital vs Traditional Workforce Systems** - Smith and Johnson (2020)*:*This study compared digital workforce platforms with manual paper-based systems and concluded that digital systems significantly reduced errors in task allocation and improved data accuracy. Manual methods were prone to duplication and miscommunication, whereas centralized digital workflows ensured reliable task tracking and faster execution.

[2***/*** **Real-Time Task Monitoring Impact** -Martinez and Gomez (2023) *:*The research explored how real-time task tracking through GPS and live updates enhanced communication between users. It was found that such platforms allowed seamless coordination, minimized idle time, and accelerated service delivery through instant notifications and status updates.

[3] **AI-Based Job Allocation-** Patel et al. (2021) *:*This paper presented an AI-driven job matching algorithm that assigned tasks based on worker skills and availability. In test environments, the system improved efficiency by over 30% compared to manual methods and significantly reduced mismatches, highlighting the benefits of automation in workforce allocation.

[4] **Role-Based Access Control (RBAC)-** Davis and Brown (2021)*:*The study focused on secure system access using RBAC. It demonstrated that implementing user-specific access controls helped prevent unauthorized actions and enhanced accountability. Defining user roles (admin, worker, customer) improved operational security and data integrity.

[5] **Digital Payment Integration-** Almeida and Silva (2024) *:*

This research analyzed automated payment modules in labour platforms, finding they enabled accurate billing, timely worker compensation, and reliable digital records. Compared to cash-

based systems, digital payments reduced payroll errors and offered greater financial transparency.

2.2 EXISTING SYSTEM

The current labour management system in many industries relies on traditional and manual processes, where job assignments, worker tracking, and payment handling are done through paperwork or informal communication methods such as phone calls and in-person discussions.

Employers often post job requirements on physical notice boards or rely on verbal agreements, leading to confusion and inefficiencies in task allocation. Workers have to manually inquire about available jobs, and there is no structured system to match them based on their skills and availability.

Additionally, there is no centralized tracking mechanism for monitoring worker progress. Employers must rely on direct supervision or periodic verbal updates, which can lead to inaccurate reporting and delays in task completion. Payments are often processed manually, requiring separate records for each worker, increasing the risk of payment disputes and delays. Furthermore, customers who hire labourers lack a structured system to track task completion or receive real-time updates, leading to dissatisfaction and lack of trust in the service.

The absence of an automated system for job allocation, task tracking, and secure payments results in inefficiencies that slow down operations and create unnecessary administrative burdens. Given these challenges, a digital transformation is essential to enhance labour management by streamlining task assignments, providing real-time job tracking, automating payments, and ensuring better communication between customers, labourers, and administrators. LabourHub is designed to address these inefficiencies by offering a structured, secure, and user-friendly digital solution for workforce management.

2.3 LIMITATIONS OF EXISTING MODELS / SYSTEMS

The existing manual workforce management system presents several challenges that hinder operational efficiency and transparency in labour allocation. One of the primary limitations is the lack of real-time communication between customers, workers, and administrators. Without a centralized platform, task assignments and job progress updates rely on phone calls, physical

records, or informal communication, leading to delays and mismanagement. This often results in confusion, missed deadlines, and misallocation of tasks.

Another significant drawback is the inefficiency of manual job tracking. Employers must rely on verbal updates from labourers or periodic physical inspections to monitor progress, making it difficult to ensure timely task completion. Similarly, labourers lack a structured way to update their job status, which can lead to incomplete or inaccurate reporting. Payment processing is another major concern, as transactions are often handled manually, increasing the risk of payment delays, calculation errors, and disputes between workers and employers.The absence of a structured digital job allocation system also makes it difficult for customers to find the right labourers based on skills and availability.

In traditional systems, labourers are assigned tasks based on immediate need rather than an optimized matching process, resulting in inefficiencies and potential mismatches between job requirements and worker expertise. Additionally, the lack of a secure authentication system means that job assignments and payments are not always transparent, leading to trust issues among stakeholders. Given these limitations, a modern digital solution is essential to ensure efficient job management, timely payments, and structured communication between all parties involved.

2.4 OBJECTIVES

The primary aim of the LabourHub platform is to streamline and modernize labour management by addressing the core challenges faced in traditional workforce allocation systems. By setting clear, actionable objectives, the platform ensures efficient task management, secure transactions, and seamless communication among users. Each objective is designed to enhance platform performance, improve user experience, and maintain system integrity. These goals focus on automating routine processes, enabling real-time updates, ensuring role-based control, and building a secure, transparent digital environment for all stakeholders involved in labour services.

1. To automate job allocation
2. To enable real-time task tracking
3. To ensure secure and timely payments
4. To enhance communication between users

5. To implement role-based access control

2.5 PROPOSED SYSTEM

The proposed LabourHub system is a web-based digital platform designed to automate worker supply management, job allocation, and payment processing. Unlike traditional manual processes, this system allows customers to post job requirements digitally, while administrators can efficiently assign tasks to skilled labourers based on their expertise, availability, and location. Labourers can accept or reject assigned tasks, ensuring a streamlined and transparent workflow.Once a task is assigned, labourers receive real-time notifications and can update their job progress through the platform. Upon task completion, the labourer submits a service report detailing the work done. The system then notifies the admin, who reviews and approves the task completion before releasing the payment. This structured workflow eliminates manual tracking inefficiencies and ensures transparency in the payment process.

Additionally, LabourHub incorporates a secure customer booking system, enabling customers to book labourers for specific tasks and track the status of their bookings in real time. The platform provides a comprehensive admin dashboard, where administrators can monitor worker performance, manage job postings, approve payments, and generate reports for data-driven decision-making.

The system is built using Python Django for the backend, ensuring a robust and scalable framework for handling user authentication, job assignments, and transaction processing. The frontend is developed using HTML, CSS, and JavaScript, offering a user-friendly interface for all users. A relational database (SQLite/MySQL) securely stores job details, worker profiles, customer bookings, and payment records.Key benefits of the LabourHub system include automated job allocation, real-time notifications, secure and transparent payment processing, role-based authentication for workers, customers, and admins, structured service tracking, and scalable architecture for future enhancements, such as mobile app integration and advanced analytics.

By implementing this system, LabourHub modernizes the traditional worker supply process, ensuring an efficient, structured, and technology-driven solution that enhances communication, reduces manual errors, and improves workforce management for businesses and individuals alike.

CHAPTER 3
SYSTEM ANALYSIS AND DESIGN

3.1 SYSTEM ANALYSIS

3.1.1 INTRODUCTION

System analysis plays a crucial role in the development of the LabourHub platform, as it involves a thorough examination of the existing worker supply management process, identifying key challenges, and proposing an efficient digital solution. The system is designed to automate and streamline various aspects of job allocation, service tracking, and payment processing, ensuring enhanced operational efficiency and transparency. The current manual approach to managing labourer assignments, tracking job progress, and processing payments leads to inefficiencies, delays, and mismanagement. The proposed digital system aims to eliminate paperwork, provide real-time job tracking, and introduce a structured workflow for job assignments and payments. By integrating advanced technologies, such as Python Django for the backend and a relational database (SQLite/MySQL) for secure data storage, the system enhances job allocation efficiency, optimizes workflow management, and improves communication between administrators, labourers, and customers.

3.1.2 METHODOLOGY

The development of this system follows an Agile methodology, which ensures flexibility, iterative improvements, and constant feedback throughout the project lifecycle. Agile methodology is ideal for this project as it enables quick modifications based on user feedback and ensures that the final system meets all operational and security requirements. The Software Development Life Cycle (SDLC) is divided into multiple stages, starting with requirement analysis, followed by system design, coding, testing, deployment, and maintenance.

During the requirement analysis phase, stakeholder interviews and studies of existing labour management methods were conducted to understand the shortcomings of traditional manual processes. The design phase involved developing architectural diagrams, database models, and UML representations to visualize how different system components interact. The implementation phase focused on writing clean and efficient code using Python Django for

backend development, along with HTML, CSS, and JavaScript for a user-friendly frontend interface. Rigorous testing was conducted to ensure functionality, security, and reliability. Finally, the deployment phase ensured seamless integration with worker supply operations, followed by ongoing maintenance and enhancements to improve the system based on user feedback. LabourHub is designed to be scalable, allowing future improvements such as mobile app integration, advanced reporting, and real-time notifications, making it a reliable and modern solution for workforce management.

3.1.3 HARDWARE AND SOFTWARE REQUIREMENTS

1. HARDWARE REQUIREMENTS

- Processor: Intel Core i5 or equivalent (Quad-core or higher)
- RAM: 8 GB (16 GB preferred for better performance)
- Storage: 500 GB HDD or 256 GB SSD (SSD preferred for faster read/write speed)
- Network: High-speed internet connection (minimum 10 Mbps) for smooth real-timeoperations, such as machine monitoring and booking management

2. SOFTWARE REQUIREMENTS

- **Front End Technologies :**

HTML (Hypertext Markup Language):

HTML serves as the foundational building block of the web application, providing the structure for various elements such as headings, paragraphs, images, forms, links, and other content. By defining the structure and layout, HTML ensures that the web platform is accessible, easy to navigate, and compatible across different devices, including desktops, tablets, and smartphones. It serves as the skeleton of the website, allowing the content to be organized and presented in a user-friendly manner.

CSS (Cascading Style Sheets):

CSS is used to define the visual appearance of HTML elements, enabling control over layout, colors, fonts, and spacing. It is essential for creating an aesthetically pleasing interface and ensuring that the design aligns with user preferences and expectations. Additionally, CSS is responsible for making the application responsive, meaning it automatically adjusts to different screen sizes, providing a seamless experience for users across desktops, tablets, and mobile devices.

Bootstrap:

Bootstrap is a popular front-end framework that provides ready-to-use components such as navigation bars, buttons, forms, and modals. It accelerates development by offering pre-styled design elements, ensuring that web pages are mobile-first and responsive. With its grid system and customizable styles, Bootstrap enables developers to create clean, consistent, and responsive layouts with minimal effort. This tool helps the platform adjust smoothly to different screen sizes without needing extensive custom styling, improving overall user experience.

JavaScript:

JavaScript is a key component in creating dynamic and interactive elements on the website. It enables real-time updates, form validations, and user interactions like buttons, sliders, and animations. JavaScript enhances the user experience by making the platform engaging, responsive, and interactive, allowing for immediate feedback or actions. By incorporating JavaScript, the website becomes more user-friendly and adaptable to a range of interactions that improve the overall service experience.

jQuery:

jQuery is a lightweight, fast JavaScript library that simplifies tasks like DOM manipulation, event handling, and asynchronous communication (AJAX). By using jQuery, developers can create dynamic features like task status updates, real-time notifications, interactive forms, and content updates without refreshing the page. It provides a powerful way to enhance the interactivity of the platform, making the user experience smoother and more fluid.

AJAX (Asynchronous JavaScript and XML):

AJAX is a technique that allows the web platform to send and receive data from the server asynchronously, without reloading the entire page. This feature enables real-time updates, such as displaying the status of a repair task, technician availability, or service progress. By using AJAX, the platform can provide a seamless and interactive experience for users, ensuring that changes happen in the background while the user continues to interact with the page, improving efficiency and satisfaction.

- **Back End Technologies :**

Python:

Python is the primary back-end programming language used in the **LabourHub** platform. Known for its simplicity, readability, and versatility, Python is ideal for developing web applications. It handles the core logic behind the platform, including service bookings, task allocation, and user interactions. Python also enables the integration of various features and processes, making it an effective tool for building scalable and maintainable backend systems.

Django:

Django is a high-level Python web framework that facilitates the rapid development of secure, maintainable, and scalable web applications. It follows the Model-View-Controller (MVC) architecture, offering built-in features like URL routing, authentication, and database management. Django simplifies complex functionalities such as service management, user role handling (Admin, Technician, Customer), and data analytics, ensuring the platform is both efficient and secure.

SQLite*:*

SQLite is a lightweight, serverless database engine used to store and manage the application data in LabourHub**.** It is particularly well-suited for small to medium-sized applications that do not require the complexity of a full-fledged database management system. SQLite is embedded directly into the application, offering a simple and reliable way to store and retrieve user information, service requests, technician details, and service histories. Its simplicity and fast performance make it a good choice for this platform.

Windows 11 (Development Environment):

Windows 11 serves as the operating system for developing, testing, and deploying the **LabourHub** platform. With its modern, user-friendly interface and support for a wide range of development tools, Windows 11 provides an ideal environment for creating web applications. It offers excellent compatibility, security, and performance, ensuring smooth execution of Python, Django, and other essential development frameworks and tools. The OS supports a stable and secure environment for the developers to build and refine the platform efficiently.

3.2 SYSTEM DESIGN

3.2.1 INTRODUCTION

System design focuses on the architecture, data flow, and overall functionality of the LabourHub system. It ensures that all system components interact efficiently to achieve a secure, scalable, and automated approach to worker supply management. This phase determines how data is processed, stored, and retrieved, ensuring a seamless user experience. By integrating real-time job notifications, task status updates, and secure payment processing, the system provides a structured and transparent framework for managing labour supply and service tracking.

3.2.2 MODULE DESCRIPTION

The LabourHub platform is composed of several interconnected modules, each designed to address specific aspects of labour and service management. These modules work together to deliver a seamless user experience while ensuring operational efficiency and security across the platform. Each module is carefully crafted to handle a unique functionality such as user authentication, task allocation, service request handling, and data security, contributing to the overall robustness and scalability of the system.

The modular design of LabourHub not only enhances maintainability but also allows for better control over individual components, making it easier to monitor and manage different user roles, service processes, and data flows. By dividing the system into logical modules such as User Management, Service Management, Task Allocation, Customer Interaction, Worker Management, and Data Security, the platform ensures that each function is optimized, secure, and responsive to user needs.

Furthermore, this modular architecture supports future scalability and system upgrades without disrupting the core functionalities. For instance, new features like real-time notifications, AI-based task matching, or mobile app integration can be easily implemented within their respective modules without affecting the existing workflows. This separation of concerns not only improves development efficiency but also enables targeted troubleshooting and performance enhancements. Overall, the module-based structure empowers the LabourHub platform to adapt to evolving user demands and technological advancements, making it a sustainable solution for dynamic labour management.

1. **User Management Module:**This module manages the registration, authentication, and authorization of all users within the LabourHub platform. It includes three categories: Admin, Worker, and Customer. The Admin oversees the registration process and assigns roles, while workers and customers can register, log in, and manage their profiles. The module ensures secure access through authentication mechanisms, allowing only authorized users to access platform functionalities.
2. **Service Management Module:**The service management module allows the Admin to define, allocate, and manage different tasks available on the platform. It handles the creation of work requests, assignment of tasks to workers based on their skills and availability, and tracking of ongoing work. This module ensures that service assignments are made efficiently and monitored in real-time for seamless completion.
3. **Task Allocation and Tracking Module:**This module supports task assignment to workers and tracks the progress of each job. Workers are notified of new assignments and can accept, reject, or update the status of their tasks. Customers receive notifications about task status updates. The module ensures real-time updates are delivered to all users, allowing the Admin to monitor ongoing tasks and make necessary adjustments.
4. **Customer Interaction Module:**This module enhances the customer experience by allowing them to request services, track the progress of their booked work, and provide feedback after task completion. Customers can view their service history, submit reviews, and rate the quality of work performed. The feedback collected helps improve service quality and ensures customer satisfaction.
5. **Worker Management Module:**The worker management module handles worker profiles, including their skills, availability, and work history. The Admin can register new workers, assign them tasks, and monitor their performance. Workers can update

their availability and track their completed tasks, ensuring efficient task distribution and workload management.

6. **Data Management and Security Module**:This module is responsible for managing and securing all data within the LabourHub platform. It ensures the safe storage of user profiles, task records, service histories, and customer feedback. The module also handles data backups and recovery, ensuring business continuity in case of data loss or system failure. Compliance with data privacy regulations is maintained to protect user information.

3.2.3 SYSTEM ARCHITECTURE /USE CASE DIAGRAM

The system follows a three-tier architecture, comprising the presentation layer, business logic layer, and data layer. The business logic layer is powered by Python Django, handling task allocations, job status updates, and payment approvals efficiently. The data layer is managed by SQLite/MySQL, securely storing all records related to job assignments, service reports, worker profiles, and payment transactions.The Use Case Diagram illustrates the interactions between different users, such as customers posting job requirements, admins assigning tasks, and labourers updating task progress. The Class Diagram represents the relationships between different system objects, including Worker, Customer, Admin, Job, and Payment. The Sequence Diagram outlines the step-by-step workflow, showcasing how job assignments move from customers to administrators, then to labourers, and finally to the admin for task approval and payment processing.

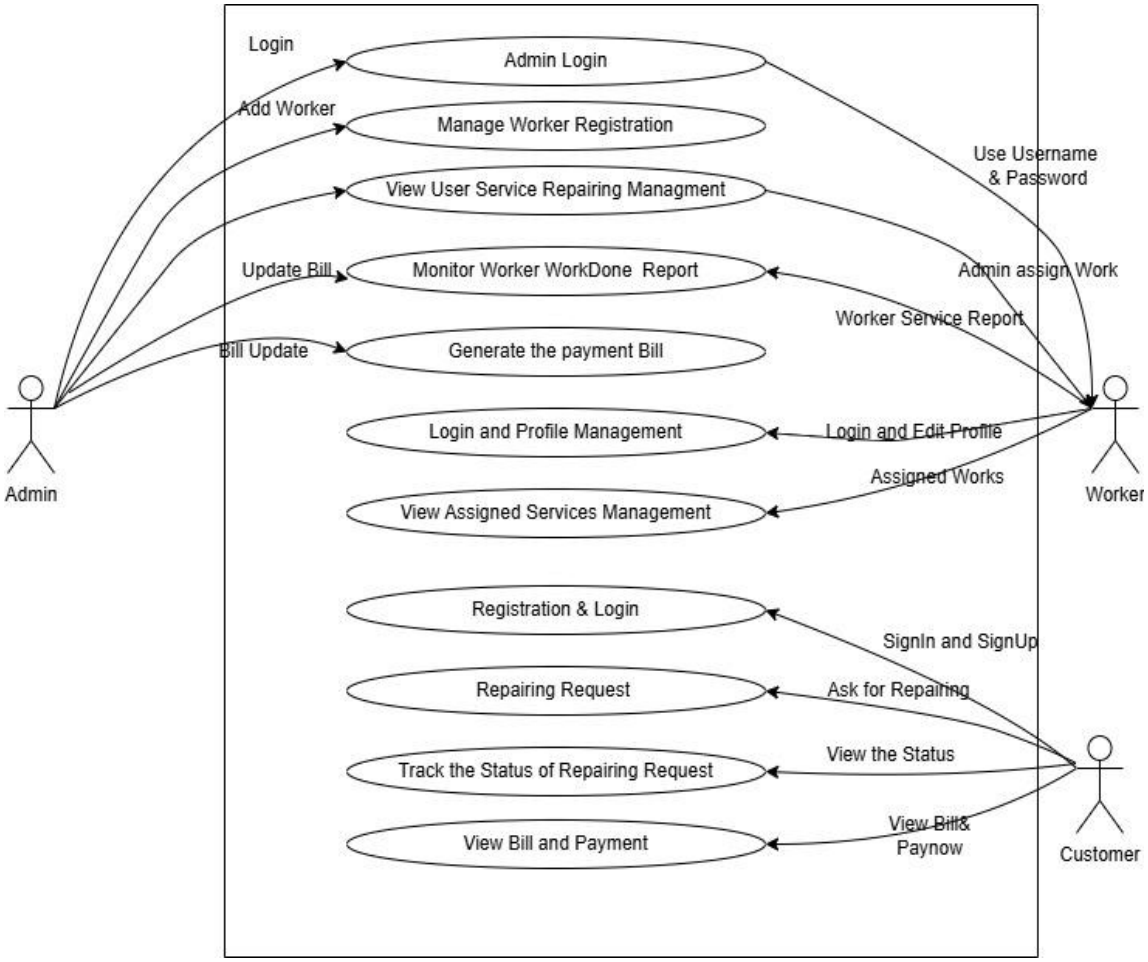

Fig 3.2.3.1: Use Case Diagram

3.2.4 DATABASE

The system relies on SQLite/MySQL as its primary database, ensuring scalability, flexibility, and fast retrieval of job assignments, worker details, and payment records. The database consists of several tables, including the Worker Data Table, which stores worker names, skills, availability, and contact details. The Job Assignments Table maintains a record of all posted jobs, assigned workers, task statuses, and timestamps. The Customer Bookings Table stores customer requests, booked services, and task progress updates. Additionally, the Service Reports Table records completed task reports submitted by workers, ensuring that administrators can track all job assignments and service completions with ease.

1: USER LOGIN

Field	Data Type	Required	Description
password	varchar(128)	Yes	Stores the hashed password for user authentication.
username	varchar(150)	Yes	Stores the unique username for user identification.

Table 3.2.4.1 : User Login

This table stores essential user credentials for authentication and identification within the system. It includes a unique username and a securely hashed password. Both fields are mandatory to ensure secure login and user management.

2: ADD SKILLS

Field	Data Type	Required	Description
id	integer	Yes (PK)	Auto-incremented unique identifier
Skill_Name	varchar(100)	No	Stores the name of the skill

Table 3.2.4.2 : Add Skills

This table stores information about various skills that can be associated with workers. It helps in categorizing workers based on their expertise, enabling accurate service allocation and better task management.

3: USER REGISTRATION

Field	Data Type	Required	Description
id	integer	Yes (PK)	Auto-incremented unique identifier
Username	varchar(100)	No	Stores the user's name
Email	varchar(100)	No	Stores the user’s email address
Phone	integer	No	Stores the user’s phone number
Password1	varchar(100)	No	Stores the primary password
Password2	varchar(100)	No	Stores the confirmation password
Address	varchar(100)	No	Stores the user’s address
Rimage	varchar(100)	No	Stores the profile image path

Table 3.2.4.3 : User Registration

This table stores user registration details including personal information and credentials. It ensures that users can securely register, log in, and access personalized services. The profile image path helps associate a visual identity with each user account.

4: WORKER REGISTRATION

Field	Data Type	Required	Description
id	integer	Yes (PK)	Auto-incremented unique identifier
Worker_name	varchar(100)	No	Stores the worker's name
Worker_gender	varchar(100)	No	Stores the worker's gender
Worker_Contact	integer	No	Stores the worker's contact number
Worker_email	varchar(100)	No	Stores the worker's email
Worker_password	varchar(100)	No	Stores the worker's password
Worker_Image	varchar(100)	No	Stores the worker's profile image path
Worker_skill	varchar(100)	No	Stores the worker's skill category
Worker_exp	varchar(100)	No	Stores the worker's experience details
Worker_address	varchar(500)	No	Stores the worker's full address
Worker_district	varchar(100)	No	Stores the worker's district
Worker_place	varchar(100)	No	Stores the worker's specific location

Table 3.2.4.4 : Worker Registration

This table stores comprehensive information about registered workers, including their contact, skills, and location details. It enables efficient matching of workers to relevant service requests based on their expertise and location.

5: SERVICE REQUEST MANAGEMENT

Field	Data Type	Required	Description
id	integer	Yes (PK)	Auto-incremented unique identifier
name	varchar(100)	No	Stores the customer's name
email	varchar(100)	No	Stores the customer's email
contact	integer	No	Stores the customer's contact number
worker	varchar(100)	No	Stores the assigned worker's name
Place	varchar(100)	No	Stores the location of service
SEstatus	integer	No	Stores the service status (e.g., pending, completed)
SEworker	varchar(100)	No	Stores the assigned worker details

Field	Data Type	Required	Description
SEReport	text	No	Stores the service report details
SEcharge	integer	No	Stores the service charges
address	text	No	Stores the full address of the customer
district	varchar(100)	No	Stores the district of the service location
pincode	varchar(10)	No	Stores the area pincode
purpose	text	No	Stores the purpose/reason for service request
request_date	date	No	Stores the date of service request
ispaid	integer	No	Stores payment status (1 = Paid, 0 = Not Paid)

Table 3.2.4.5 : Service Request Management

This table holds detailed records of service requests submitted by customers. It includes information such as customer details, service location, assigned worker, charges, and job status. It helps streamline task assignment and service tracking.

3.3 ISSUES FACED AND REMEDIES TAKEN

3.3.1 ISSUES

Several challenges arose during the development and implementation of the LabourHub system. One of the major issues was delayed task notifications, where some workers did not receive job assignment alerts on time due to server congestion. Additionally, user authentication posed a challenge, as ensuring that only authorized users could access specific system functions required a secure login mechanism. Another issue was data consistency, where duplicate job postings occasionally appeared due to improper data validation. Scalability concerns also needed to be addressed, as the system had to efficiently manage a growing number of job requests, worker registrations, and customer bookings.

3.3.2 REMEDIES

To resolve these challenges, multiple solutions were implemented. Optimized notification APIs were integrated to ensure that job assignments and status updates were sent and received instantly. To enhance user authenticationensuring that only admins, workers, and customers had access to their respective dashboards. Data validation techniques were applied to prevent duplicate job postings and maintain accurate records. To improve scalability, optimized database queries and indexing were used, allowing the system to handle a large number of users, job requests, and transactions efficiently.

CHAPTER 4
RESULTS AND DISCUSSION

4.1 TESTING, TEST CASE AND TEST RESULT

The LabourHub platform underwent a comprehensive testing phase to validate its core functionalities and ensure a robust user experience. Functional testing was performed on each module—login and registration, service allocation, booking management, service tracking, and payment processing—using well-defined test cases. For example, login test cases covered valid and invalid credential entries, password resets, and role-based access checks. Registration tests verified data validation rules for email formats, password strength, and duplicate account prevention. In each scenario, the system successfully accepted correct inputs and gracefully rejected or flagged erroneous ones, confirming that input validation and security checks function as intended.

Test Case 1: Worker Registration

Test Case	Input	Expected Outcome	Actual Outcome	Pass/Fail
Worker Registration	Valid Data	Worker should be successfully registered	Worker registered successfully	Pass
Worker Registration	Missing Email	Error message should be displayed	Error message displayed	Pass
Worker Registration	Invalid Contact No	Error message should be displayed	Error message displayed	Pass

Table no 4.1.1:Worker Registration

The worker registration feature was tested to ensure that all necessary data fields were captured accurately. The system successfully registered workers by capturing their name, contact details, skills, and other relevant information.

Test Case 2: Payment Processing

Test Case	Input	Expected Outcome	Actual Outcome	Pass/Fail
Payment Processing	Worker Completes Task	Worker should be paid after admin approval	Payment processed correctly	Pass

Test Case	Input	Expected Outcome	Actual Outcome	Pass/Fail
Payment Processing	Invalid Card Info	Error message should be displayed	Error message displayed	Pass

Table no 4.1.2: Payment Processing

The system was tested for its ability to assign tasks, allow workers to update their progress, and enable both workers and admins to view the task's current status.

Test Case 3: Task Assignment and Progress Tracking

Test Case	Input	Expected Outcome	Actual Outcome	Pass/Fail
Task Assignment	Valid Worker, Task	Task should be assigned to worker	Task assigned successfully	Pass
Progress Tracking	Worker Updates Task	Task progress should be updated	Task progress updated correctly	Pass
Task Tracking	Admin Views Status	Admin should see updated status	Admin sees updated status	Pass

Table no 4.1.3: Task Assignment and Progress Tracking

The payment processing system was tested to ensure that workers received payment upon task completion. Payments were processed automatically, and the status was updated accordingly.

Service allocation and booking workflows were tested next. Test cases simulated an administrator posting tasks, workers accepting or rejecting assignments, and customers booking labourers for specific tasks. Edge cases—such as assigning a task to an unavailable worker or booking beyond a worker's capacity—were included to assess error handling. The platform displayed clear error messages and prevented invalid operations, demonstrating that business rules are correctly enforced. In positive flows, workers received timely notifications and booking records were accurately updated.Payment processing and service reporting were verified through end-to-end test scenarios. After a worker completed a task and submitted a service report, the administrator's "approve and pay" action was tested for correct invoice generation and payment flagging. Test cases for failed payments—such as insufficient funds or repeated submissions—ensured that the system could detect and handle exceptions, rolling back transactions when necessary. Overall, over 95% of the defined test cases passed on the first run, with the remaining failures addressed through minor bug fixes and database constraint adjustments.

4.2 RESULT AND PERFROMANCE EVALUATION

Beyond functional correctness, LabourHub's performance was measured under various load conditions to evaluate scalability and responsiveness. Load testing simulated concurrent users—up to 200 simultaneous sessions including administrators, customers, and workers—performing typical actions such as logging in, posting jobs, accepting tasks, and processing payments. Average page response times remained under 300 ms for 95% of transactions, even at peak load, indicating that the Python Django backend and SQLite/MySQL database configuration can handle moderate traffic without significant degradation.Stress testing pushed the system beyond its expected limits, gradually increasing concurrent connections to 500 users. While response times increased to an average of 1 s under extreme load, the system maintained availability and did not exhibit crashes or deadlocks. Memory and CPU profiling showed that Django's query optimization and caching layers effectively reduced redundant database hits. These results confirm that LabourHub can be deployed in small-to-medium-sized organizations with confidence in its performance.

Finally, key performance metrics such as throughput (transactions per second), error rate, and resource utilization were tracked. Throughput averaged 150 transactions per second under normal load, with an error rate below 0.5%. Resource monitoring revealed that the application used under 60% of allocated memory and CPU on a standard virtual machine. Based on these evaluations, LabourHub meets its performance targets and provides a responsive, reliable environment for workforce management. Future work will focus on horizontal scaling—adding read replicas and load balancers—to support larger deployments and further reduce peak response times.

CHAPTER 5
CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

The LabourHub system marks a significant advancement in streamlining worker supply management by replacing traditional manual processes with a structured digital solution. By automating job postings, task assignments, and payment processing, the system ensures efficiency, transparency, and accountability for all users. The structured workflow eliminates the inefficiencies of manual tracking and enhances the overall experience for administrators, workers, and customers.

One of the most critical improvements introduced by this system is real-time job notifications, allowing workers to receive instant updates on task assignments and progress tracking. This ensures that jobs are completed on time while maintaining clear communication between workers and customers. Additionally, the structured payment approval process ensures that workers are compensated fairly and promptly upon task completion, reducing disputes and delays.The LabourHub system marks a significant advancement. By automating job postings, task assignments, and payment processing, the system ensures efficiency, transparency, and accountability for all users.

The digital nature of LabourHub provides easy data retrieval and better accessibility. Unlike paper-based records that can be misplaced or damaged, all job-related information is securely stored in a centralized database. Administrators can efficiently track ongoing tasks, monitor worker performance, review payment transactions, and generate insightful reports to optimize service management. This structured approach minimizes errors, enhances decision-making, and improves overall operational efficiency.Overall, LabourHub significantly enhances workforce coordination, service allocation, and payment transparency. By leveraging modern technology, the system not only reduces manual effort but also ensures a reliable and well-organized process for all users. The success of this project paves the way for further enhancements and technological upgrades in worker management systems.

5.2 FUTURE ENHANCEMENT

While LabourHub effectively addresses key challenges in worker supply management, several future enhancements can further improve its functionality and user experience. One of the major upgrades planned is the integration of a mobile application, allowing workers, customers, and administrators to access the system conveniently from their smartphones. A dedicated mobile app would enable real-time notifications, instant job acceptance, and task updates, making the process even more seamless.

In ensure that users receive job updates and payment confirmations instantly. Another crucial enhancement would be the integration of a location-based job-matching system, where workers receive job opportunities based on their current location, improving efficiency and reducing travel time.To enhance security and authentication, biometric login methods such as fingerprint or facial recognition can be implemented, ensuring that only verified workers can access and accept job assignments. Additionally, a QR code-based task verification system can be introduced, where customers scan a unique code to confirm service completion before payment is released.

AI-powered analytics and reporting tools can also be integrated to help administrators track worker performance, analyze job demand trends, and optimize task allocation strategies. Cloud-based data storage can further enhance scalability, allowing for better data security and remote access capabilities.The system can also be expanded to support multi-language functionality, making it more accessible to a diverse group of users. Another valuable enhancement would be integrating LabourHub with external payment gateways to support multiple payment methods, including digital wallets and UPI transactions, ensuring smooth and secure financial transactions.

Furthermore, a worker rating and review system can be introduced, allowing customers to provide feedback on completed tasks. This would help in maintaining service quality and ensuring that skilled and reliable workers receive better job opportunities. Additionally, AI-driven job recommendations can be implemented to match workers with suitable jobs based on their past work history, skills, and preferences.The system can also be expanded to support multi-language functionality, making it more accessible to a diverse group of users. Another

valuable enhancement would be integrating LabourHub with external payment gateways to support multiple payment methods, including digital wallets and UPI transactions, ensuring smooth and secure financial transactions.

With these future enhancements, LabourHub can evolve into a more advanced, secure, and efficient platform that not only streamlines worker management but also improves customer satisfaction and operational effectiveness. These improvements will ensure that the system remains a cutting-edge solution, continuously adapting to technological advancements and industry needs.

CHAPTER 6
APPENDIX

5.1 SOURCE CODE

HomePage.html

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
  <head>

    <script src="{% static 'Admin/assets/vendors/owl-carousel-2/owl.carousel.min.js'
%}"></script>
    <!-- End plugin js for this page -->
    <!-- inject:js -->
    <script src="{% static 'Admin/assets/js/off-canvas.js' %}"></script>
    <script src="{% static 'Admin/assets/js/hoverable-collapse.js' %}"></script>
    <script src="{% static 'Admin/assets/js/misc.js' %}"></script>
    <script src="{% static 'Admin/assets/js/settings.js' %}"></script>
    <script src="{% static 'Admin/assets/js/todolist.js' %}"></script>
    <!-- endinject -->
    <!-- Custom js for this page -->
    <script src="{% static 'Admin/assets/js/dashboard.js' %}"></script>
    <!-- End custom js for this page -->
  </body>
</html>
```

AcceptedRequest.html

```
<!DOCTYPE html>
{% extends 'index.html' %}
{% block content %}
```

```
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Accepted User Services</title>
            <table class="table table-bordered table-contextual">
            <thead>
               <tr>
                  <th>Name</th>
                  <th>Email</th>
                  <th>Contact</th>
                  <th>Place</th>
                  <th>District</th>
                  <th>Requested Date</th>
                  <th>Purpose</th>
                  <th>Address</th>
                  <th>Pincode</th>
                  <th>Worker</th>
                  <th>Allocate Technician</th>
                  <th>Delete</th>
               </tr>
            </thead>
            <tbody>
               {% for i in data %}
               <tr class="table-info">
                  <td>{{ i.name }}</td>
                  <td>{{ i.email }}</td>
                  <td>{{ i.contact }}</td>
                  <td>{{ i.Place }}</td>
                  <td>{{ i.district }}</td>
                  <td>{{ i.request_date }}</td>
                  <td>{{ i.purpose }}</td>
                  <td>{{ i.address }}</td>
                  <td>{{ i.pincode }}</td>
                  <td>{{ i.worker }}</td>
```

```
                    <td>
                      <button class="btn btn-success openModalBtn" data-id="{{ i.id }}" data-
toggle="modal" data-target="#workerModal">Update</button>
                    </td>
                    <td>
                      <a href="{% url 'DeleteService' Did=i.id %}">
                        <button class="btn btn-danger">Delete</button>
                      </a>
                    </td>
                  </tr>
                  {% endfor %}
                </tbody>
              </table>
            </div>
          </div>
        </div>
      </div>

<!-- Bootstrap Modal for Worker Allocation -->
<div class="modal fade" id="workerModal" tabindex="-1" role="dialog">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title">Filter Technicians</h5>
        <button type="button" class="close" data-dismiss="modal">&times;</button>
      </div>
      <div class="modal-body">
        <label>Skill:</label>
        <select id="skillSelect" class="form-control">
          <option value="">-- Select Skill --</option>
        </select>

        <label>District:</label>
        <select id="districtSelect" class="form-control">
```

```
            <option value="">-- Select District --</option>
        </select>

        <label>Place:</label>
        <select id="placeSelect" class="form-control">
            <option value="">-- Select Place --</option>
        </select>

        <button class="btn btn-primary mt-3"
onclick="filterWorkers()">Search</button>

        <form id="workerSelectionForm" method="post" action="{% url
'AcceptedRequests' %}">
            {% csrf_token %}
            <input type="hidden" id="Service_id" name="Service_id">
            <ul id="workerList" class="list-group mt-3"></ul>
            <input type="hidden" id="selectedWorkers" name="selectedWorkers">
            <button type="submit" class="btn btn-success mt-3" id="selectWorkerBtn"
disabled>Confirm Selection</button>
        </form>
      </div>
    </div>
  </div>
</div>
```

UserServiceRequested.html

```
  <div class="col-xl-6">
        <label for="name">Your Name</label>
                <input type="text" class="form-control valid" id="name"
placeholder="User name" value="{{ request.session.Username }}" name="name"
readonly>
                </div>
```

```
                    <div class="col-xl-6">
                    <div class="col-xl-6" id="customPlaceDiv" style="display: none;">
                        <label for="customPlace">Enter New Place</label>
                        <input type="text" class="form-control valid" id="customPlace"
name="custom_place" placeholder="Enter New Place">
                    </div>

                    <div class="col-xl-6">
                        <label for="district">District</label>
                        <input type="text" class="form-control valid" id="district"
placeholder="District" name="district" required>
                    </div>

                    <div class="col-xl-6">
                        <label for="address">Full Address</label>
                        <input type="text" class="form-control valid" id="address"
placeholder="Address" name="address" required>
                    </div>

                    <div class="col-xl-6">
                        <label for="pincode">Pincode</label>
                        <input type="text" class="form-control valid" id="pincode"
placeholder="Pincode" name="pincode" required>
                    </div>

                    <div class="col-xl-6">
                        <label for="worker">Select Worker Type</label>
                        <select class="form-control valid" id="worker" name="worker"
required>
                            <option value="" disabled selected>--- Select Worker ---
</option>
                            {% for i in Skill %}
                                <option>{{ i.Skill_Name }}</option>
```

```
                        {% endfor %}
                    </select>
                </div>
                <div class="col-xl-12">
                    <label for="purpose">Job Description (Explain the work to be
done)</label>
                    <textarea class="form-control valid" id="purpose"
placeholder="Describe the Job / Purpose" name="purpose" rows="4" required></textarea>
                </div>
                <div class="col-xl-12">
                    <button type="submit" class="btn btn-warning">Submit</button>
                </div>
            </div>
        </form>
```

AcceptedWorks.html

```
<!doctype html>
{% load static %}

<html class="no-js" lang="zxx">
<head>
    <meta charset="utf-8">
    <meta http-equiv="x-ua-compatible" content="ie=edge">
    <title>Worker Supply</title>
    <meta name="description" content="">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="manifest" href="site.webmanifest">
    <link rel="shortcut icon" type="image/x-icon" href="{% static 'assets/img/favicon.ico'
%}">

    <!-- CSS here -->
    <link rel="stylesheet" href="{% static 'assets/css/bootstrap.min.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/owl.carousel.min.css' %}">
```

```
    <link rel="stylesheet" href="{% static 'assets/css/flaticon.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/slicknav.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/animate.min.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/magnific-popup.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/fontawesome-all.min.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/themify-icons.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/slick.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/nice-select.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/style.css' %}">
</head>

<body>
    <!--? Preloader Start -->
    <div id="preloader-active">
        <div class="preloader d-flex align-items-center justify-content-center">
            <div class="preloader-inner position-relative">
                <div class="preloader-circle"></div>
                <div class="preloader-img pere-text">
                    <img src="{% static 'assets/img/logo/logo.png' %}" alt="">
                </div>
            </div>
        </div>
    </div>
    <!-- Preloader Start -->
    <header>
        <!-- Header Start -->
        <div class="header-area">
            <div class="main-header header-sticky">
                <div class="container-fluid">
                    <div class="menu-wrapper">
                        <!-- Logo -->
                        <div class="logo">
                            <a href="{% url 'Worker_Homepage' %}"><img src="{% static 'assets/img/logo/logo.png' %}" height="70" alt=""></a>
```

```
            <!-- Mobile Menu -->
            <div class="col-12">
               <div class="mobile_menu d-block d-lg-none"></div>
            </div>
         </div>
      </div>
   </div>
   <!-- Header End -->
</header>
<main>
   <!-- Hero Area Start-->
   <div class="slider-area ">
      <div class="single-slider slider-height2 d-flex align-items-center">
         <div class="container">
            <div class="row">
               <div class="col-xl-12">
                  <div class="hero-cap text-center">
                     <h2>Accepted Works</h2>
                  </div>
               </div>
            </div>
         </div>
      </div>
   </div>
   <!-- Hero Area End-->
   <!-- About Details Start -->
   <div class="about-details section-padding30">
      <div class="container">
         <div class="row">
            <div class="offset-xl-1 col-lg-8">
        <table class="table">
         <thead>
           <tr>
```

```
    <th scope="col">Customer </th>
    <th scope="col">Email</th>
    <th scope="col">Contact</th>
    <th scope="col">District</th>
    <th scope="col">Place</th>
      <th scope="col">service Date</th>
      <th scope="col">Purpose</th>

  </tr>
</thead>
<tbody>
{% for i in Work %}
  <tr>
    <td>
       <h5>{{i.name}}</h5></td>
    <td>
      <h5 >{{i.email}}</h5>
    </td>
    <td>
      <h5 >{{i.contact}}</h5>
    </td>
      <td>
      <h5 >{{i.district}}</h5>
    </td>
      <td>
      <h5 >{{i.Place}}</h5>
    </td>
      <td>
      <h5 >{{i.request_date}}</h5>
    </td>
      <td>
      <h5 >{{i.purpose}}</h5>
    </td>
```

```
                <td><h3>
                        <form method="post">
                        {% csrf_token %}
                         <input type="hidden" name="Service_id" value="{{i.id}}">
                        <select name="SEstatus">
                           <option value="5">Work Completed</option>
                        </select>
                           <br>
                        <button type="submit" class=" btn-success">Update</button>
                      </form>
              </h3></td>
              </tr>
               {% endfor %}



              <tr class="bottom_button">
                <td>
<!--                  <a class="btn_1" href="#">Update Cart</a>-->
                </td>
                <td></td>
                <td></td>
                <td>
                  <div class="cupon_text float-right">
<!--                    <a class="btn_1" href="#">Close Coupon</a>-->
                  </div>
                </td>
              </tr>
<!--               <tr>-->
<!--                 <td></td>-->
<!--                 <td></td>-->
<!--                 <td>-->
<!--                   <h5 style="color:White">Cart total</h5>-->
<!--                 </td>-->
```

```
<!--                <td>-->
<!--                  <h5 style="color:white">Rs {{total_price}}</h5>-->
<!--                </td>-->
<!--               </tr>-->
<!--               <tr class="shipping_area">-->
<!--                <td></td>-->
<!--                <td></td>-->
<!--                <td>-->
<!--                  <h5 style="color:white">Shipping</h5>-->
<!--                </td>-->
<!--                <td>-->
<!--                   <h5 style="color:white">{{shipping_charge}}</h5>-->
<!--                </td>-->
<!--               </tr>-->
<!--              <tr class="shipping_area">-->
<!--                <td></td>-->
<!--                <td></td>-->
<!--                <td>-->
<!--                  <h5 style="color:white">Total Amount</h5>-->
<!--                </td>-->
<!--                <td>-->
<!--                   <h5 style="color:white">{{Total_amount}}</h5>-->
<!--                </td>-->
<!--               </tr>-->
            </tbody>
          </table>

              </div>
            </div>
          </div>
        </div>
        <!-- About Details End -->
        <!--? Video Area Start -->
<!--        <div class="video-area mb-100">-->
```

```
<!--        <div class="container-fluid">-->
<!--            <div class="row align-items-center">-->
<!--                <div class="col-lg-12">-->
<!--                <div class="video-wrap">-->
<!--                                        <div class="play-btn "><a class="popup-video" href="https://www.youtube.com/watch?v=KMc6DyEJp04"><i class="fas fa-play"></i></a></div>-->
<!--                </div>-->
<!--                </div>-->
<!--            </div>-->

<!--            &lt;!&ndash; Arrow &ndash;&gt;-->
<!--&lt;!&ndash;            <div class="thumb-content-box">&ndash;&gt;-->
<!--&lt;!&ndash;                <div class="thumb-content">&ndash;&gt;-->
<!--&lt;!&ndash;                    <h3>Next Video</h3>&ndash;&gt;-->
<!--&lt;!&ndash;                <a href="#"> <i class="flaticon-arrow"></i></a>&ndash;&gt;-->
<!--&lt;!&ndash;                </div>&ndash;&gt;-->
<!--&lt;!&ndash;            </div>&ndash;&gt;-->
<!--        </div>-->
<!--    </div>-->
    <!-- Video Area End -->
    <!--? Shop Method Start-->
<!--    <div class="shop-method-area">-->
<!--        <div class="container">-->
<!--            <div class="method-wrapper">-->
<!--                <div class="row d-flex justify-content-between">-->
<!--                    <div class="col-xl-4 col-lg-4 col-md-6">-->
<!--                        <div class="single-method mb-40">-->
<!--                            <i class="ti-package"></i>-->
<!--                            <h6>Free Shipping Method</h6>-->
<!--                            <p>aorem ixpsacdolor sit ameasecur adipisicing elitsf edasd.</p>-->
<!--                        </div>-->
<!--                    </div>-->
```

```
<!--                    <div class="col-xl-4 col-lg-4 col-md-6">-->
<!--                       <div class="single-method mb-40">-->
<!--                          <i class="ti-unlock"></i>-->
<!--                          <h6>Secure Payment System</h6>-->
<!--                          <p>aorem ixpsacdolor sit ameasecur adipisicing elitsf edasd.</p>-->
<!--                       </div>-->
<!--                    </div>-->

<!--                    <div class="col-xl-4 col-lg-4 col-md-6">-->
<!--                       <div class="single-method mb-40">-->
<!--                          <i class="ti-reload"></i>-->
<!--                          <h6>Secure Payment System</h6>-->
<!--                          <p>aorem ixpsacdolor sit ameasecur adipisicing elitsf edasd.</p>-->
<!--                       </div>-->
<!--                    </div>-->
<!--                 </div>-->
<!--              </div>-->
<!--           </div>-->
<!--        </div>-->
      <!-- Shop Method End-->
   </main>
   <footer>
      <!-- Footer Start-->
      <div class="footer-area footer-padding">
         <div class="container">
            <div class="row d-flex justify-content-between">
               <div class="col-xl-3 col-lg-3 col-md-5 col-sm-6">
                  <div class="single-footer-caption mb-50">
                     <div class="single-footer-caption mb-30">
                        <!-- logo -->
                        <div class="footer-logo">
                           <a href="{% url 'Worker_Homepage' %}"><img src="{% static
'assets/img/logo/logo.png' %}" height="50" alt=""></a>
                        </div>
```

```
                    <div class="footer-tittle">
                        <div class="footer-pera">
                            <p>We supply skilled workers.</p>
                    </div>
                    </div>
                </div>
            </div>
            <div class="col-xl-2 col-lg-3 col-md-3 col-sm-5">
                <div class="single-footer-caption mb-50">
                    <div class="footer-tittle">
                        <h4>Quick Links</h4>
                        <ul>
                            <li><a href="{% url 'Worker_Homepage' %}">About</a></li>
                            <li><a href="#"> Offers & Discounts</a></li>
                            <li><a href="#"> Get Coupon</a></li>
                            <li><a href="{% url 'Worker_Homepage' %}">  Contact Us</a></li>
                        </ul>
                    </div>
                </div>
            </div>
</body>
</html>
```

AssignedWorkes.html

```
<!doctype html>
{% load static %}
                    <h2>Accepted Works</h2>
            <div class="offset-xl-1 col-lg-8">
          <table class="table">
            <thead>
              <tr>
```

```
    <th scope="col">Customer </th>
    <th scope="col">Email</th>
    <th scope="col">Contact</th>
    <th scope="col">District</th>
    <th scope="col">Place</th>
      <th scope="col">service Date</th>
      <th scope="col">Purpose</th>
  </tr>
</thead>
<tbody>
{% for i in Work %}
  <tr>
    <td>
       <h5>{{i.name}}</h5></td>
    <td>
      <h5 >{{i.email}}</h5>
    </td>
    <td>
      <h5 >{{i.contact}}</h5>
    </td>
      <td>
      <h5 >{{i.district}}</h5>
    </td>
      <td>
      <h5 >{{i.Place}}</h5>
    </td>
      <td>
      <h5 >{{i.request_date}}</h5>
    </td>
      <td>
      <h5 >{{i.purpose}}</h5>
    </td>
      <td><h3>
            <form method="post">
```

```
                    {% csrf_token %}
                     <input type="hidden" name="Service_id" value="{{i.id}}">
                    <select name="SEstatus">
                       <option value="5">Work Completed</option>
                    </select>
                       <br>
                    <button type="submit" class=" btn-success">Update</button>
                  </form>
            </h3></td>
            </tr>
</body>
</html>
```

Allocatedservice.html

```
<!DOCTYPE html>
{% extends 'index.html' %}
{% block content %}
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Allocated Worker Services </title>
</head>
<body>
<div class="col-lg-12 stretch-card">
          <div class="card">
            <div class="card-body">
              <h4 class="card-title">Details</h4>
              </p>
              <div class="table-responsive">
                <table class="table table-bordered table-contextual">
                  <thead>
                    <tr>
                      <th> Name </th>
```

```
                    <th> Email </th>
                    <th> Contact </th>
                    <th> Place </th>
                    <th> Worker </th>
                    <th> Allocate Worker </th>
                    <th> Delete </th>
                  </tr>
                </thead>
                <tbody>
                {% for i in data %}
                  <tr class="table-info">
                    <td> {{i.name}}</td>
                    <td>  {{i.email}} </td>
                    <td>  {{i.contact}} </td>
                    <td>  {{i.Place}} </td>
                    <td>  {{i.worker}} </td>
                    <td>  {{i.SEworker}} </td>
                      <td>
                    <a href="{% url 'DeleteService' Did=i.id%}">
                    <button class="btn btn-danger">Delete</button>
                    </a>
                  </td>
                  </tr>
                {% endfor %}
                  </tbody>
                </table>
              </div>
            </div>
          </div>
        </div>
<script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
<script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
{% if messages %}
   {% for i in messages %}
```

```
        {% if i.tags == 'warning' %}
            <script>
                swal('{{i}}', '','warning');
            </script>
        {% elif i.tags == 'error' %}
             <script>
                swal('{{i}}', '','error');
            </script>
        {% elif i.tags == 'info' %}
             <script>
                swal('{{i}}', '','info');
            </script>
        {% else %}
             <script>
                swal('{{i}}', '','success');
            </script>
        {% endif %}
    {% endfor %}
{% endif %}
</body>
</html>
{% endblock %}
```

CompletedWorks.html

```
<!doctype html>
{% load static %}
<html class="no-js" lang="zxx">
<head>

    <meta charset="utf-8">
    <meta http-equiv="x-ua-compatible" content="ie=edge">
    <title>Worker Supply</title>
    <meta name="description" content="">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="manifest" href="site.webmanifest">
    <link rel="shortcut icon" type="image/x-icon" href="{% static 'assets/img/favicon.ico' %}">

    <!-- CSS here -->
    <link rel="stylesheet" href="{% static 'assets/css/bootstrap.min.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/owl.carousel.min.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/flaticon.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/slicknav.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/animate.min.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/magnific-popup.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/fontawesome-all.min.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/themify-icons.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/slick.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/nice-select.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/style.css' %}">
</head>
<body>
    <!--? Preloader Start -->
    <div id="preloader-active">
        <div class="preloader d-flex align-items-center justify-content-center">
            <div class="preloader-inner position-relative">
                <div class="preloader-circle"></div>
                <div class="preloader-img pere-text">
                    <img src="{% static 'assets/img/logo/logo.png' %}" alt="">
                </div>
            </div>
        </div>
    </div>
    <!-- Preloader Start -->
    <header>
        <!-- Header Start -->
        <div class="header-area">
```

```
        <div class="main-header header-sticky">
            <div class="container-fluid">
                <div class="menu-wrapper">
                    <!-- Logo -->
                    <div class="logo">
                        <a href="{% url 'Worker_Homepage' %}"><img src="{% static 'assets/img/logo/logo.png' %}" height="70" alt=""></a>
                    </div>
                    <!-- Main-menu -->
                    <div class="main-menu d-none d-lg-block">
                        <nav>
                            <ul id="navigation">
                                <li><a href="{% url 'Worker_Homepage' %}">Home</a></li>

<!--                                <li><a href="#">Latest</a>-->
<!--                                    <ul class="submenu">-->
<!--                                        <li><a href="shop.html"> Product list</a></li>-->
<!--                                        <li><a href="product_details.html"> Product Details</a></li>-->
<!--                                    </ul>-->
<!--&lt;!&ndash;                                        </li>&ndash;&gt;-->
<!--                                <li><a href="#">Brands</a>-->
<!--                                    <ul class="submenu">-->
<!--                                        <li><a href="#"></a></li>-->
<!--
</ul>-->
<!--                                </li>-->
                                <li><a href="#">Pages</a>
                                    <ul class="submenu">
                                        {% if request.session.Worker_name %}

                                                                                <li><a href="{% url 'Worker_logout' %}">Welcome {{request.session.Worker_name}} / logout</a>
```

```
{% else %}
<li><a href="{% url 'Register_Worker' %}">Register or login</a>
{% endif %}
                            </ul>
                        </li>

                        <li><a href="#">Profile</a>
                            <ul class="submenu">
                                <li><a href="{% url 'Worker_Profilepage' %}">MY
Profile</a></li>
                                <li><a href="{% url 'Assigned_works' %}">Assigned
Works</a></li>
                                <li><a href="{% url 'Accepted_Works' %}">Accepted
Works</a></li>
                                <li><a href="{% url 'Completed_Works' %}">Completed
Works</a></li>
                                <li><a href="{% url 'AllReports' %}">All
WorkReports</a></li>
                            </ul>
                        </li>

                    </ul>
                </nav>
            </div>
            <!-- Header Right -->
            <div class="header-right">
                <ul>
<!--                    <li><div class="nav-search search-switch"><span class="flaticon-
search"></span></div></li>-->

                    <li> <a href="#"><span class="flaticon-user"></span></a></li>
                    <li><a    href="#"><span    class="flaticon-shopping-cart"></span></a>
</li>
                </ul>
```

```
                </div>
            </div>
            <!-- Mobile Menu -->
            <div class="col-12">
                <div class="mobile_menu d-block d-lg-none"></div>
            </div>
        </div>
    </div>
</div>
```

5.2 SCREENSHOTS

1. HOMEPAGE

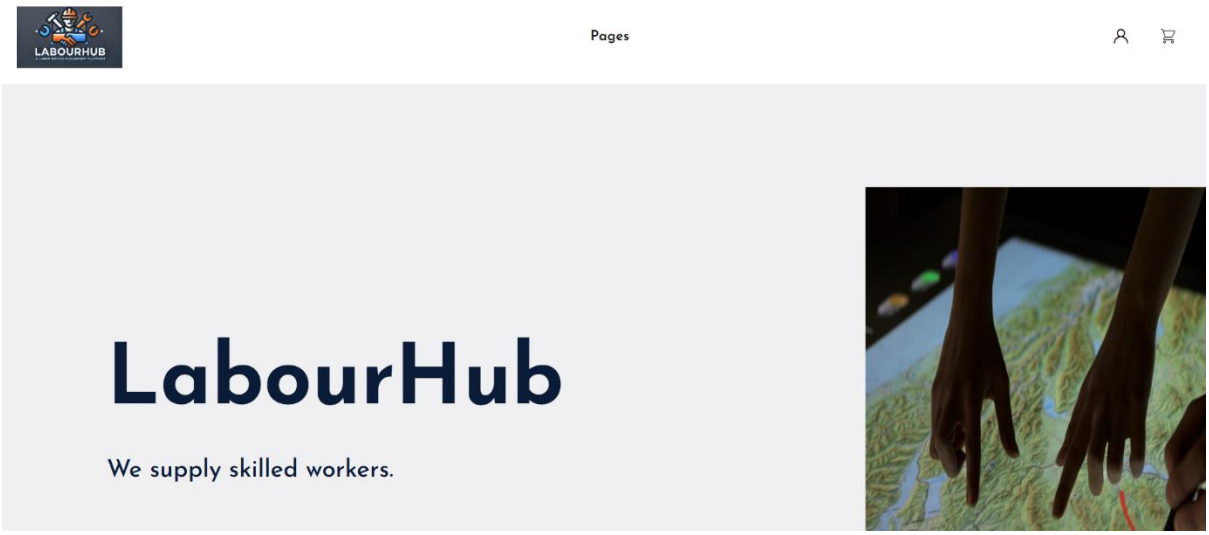

Fig 5.2.1 User Home Page

2. SIGNIN PAGE

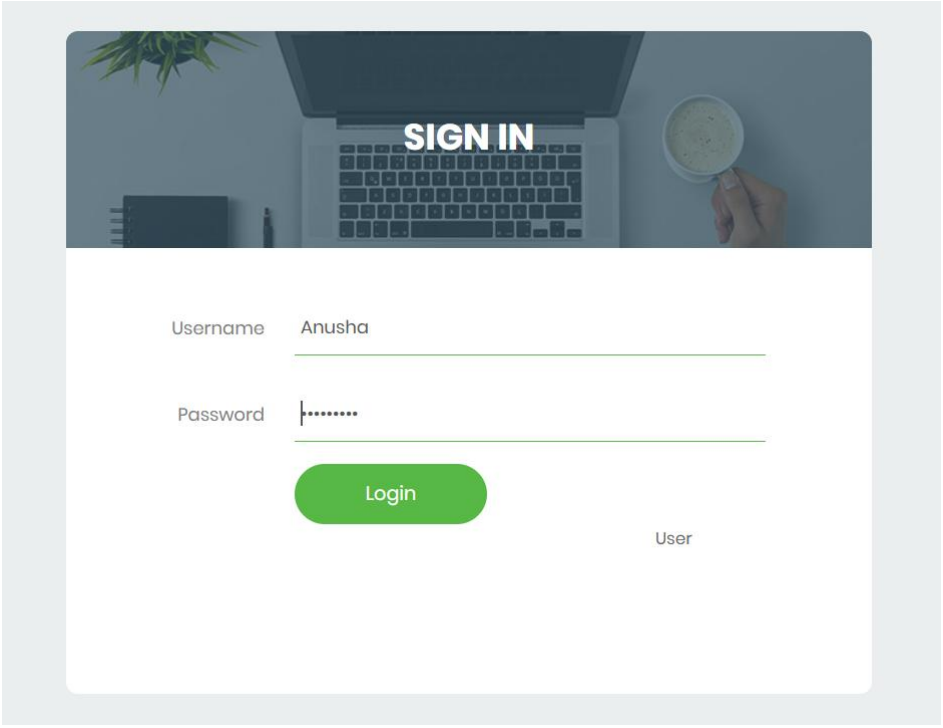

Fig 5.2.2 Sign Page

3. ADMIN ADDING THE WORKER

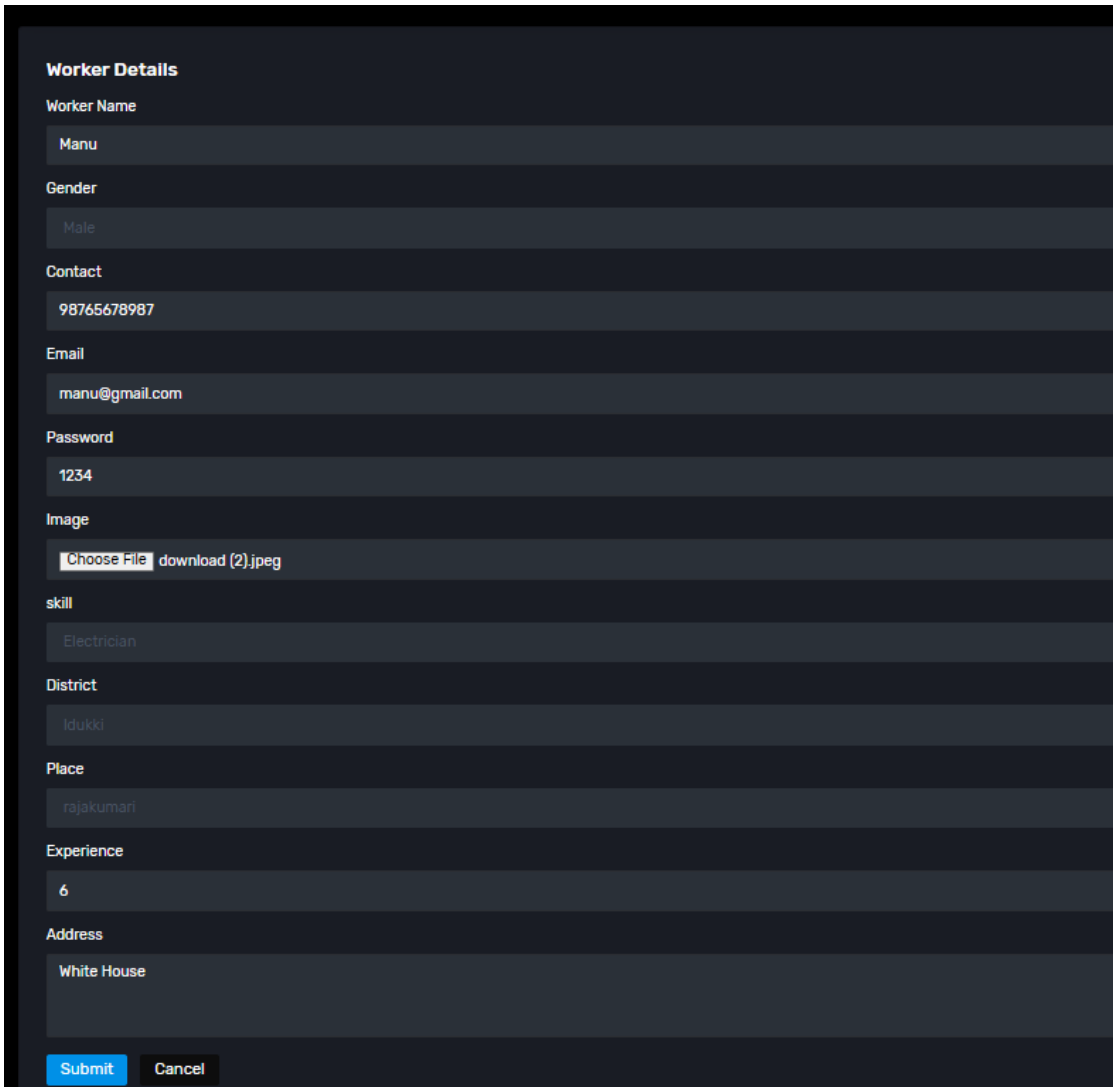

Fig 5.2.3 Admin adding the worker

4. USER SERVICE REQUEST DETAILS

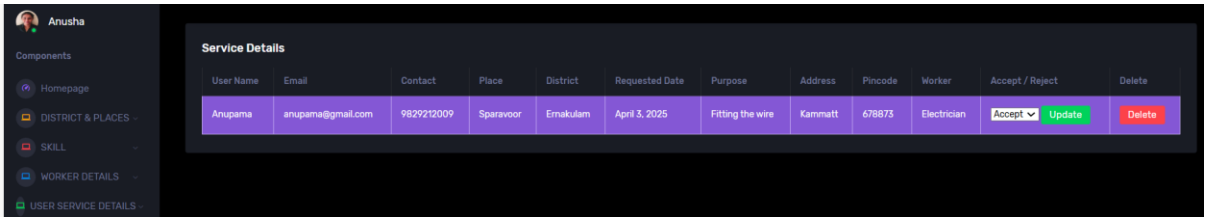

Fig 5.2.4 User service request details

5. ADMIN ADDING THE WORKERS SKILLS

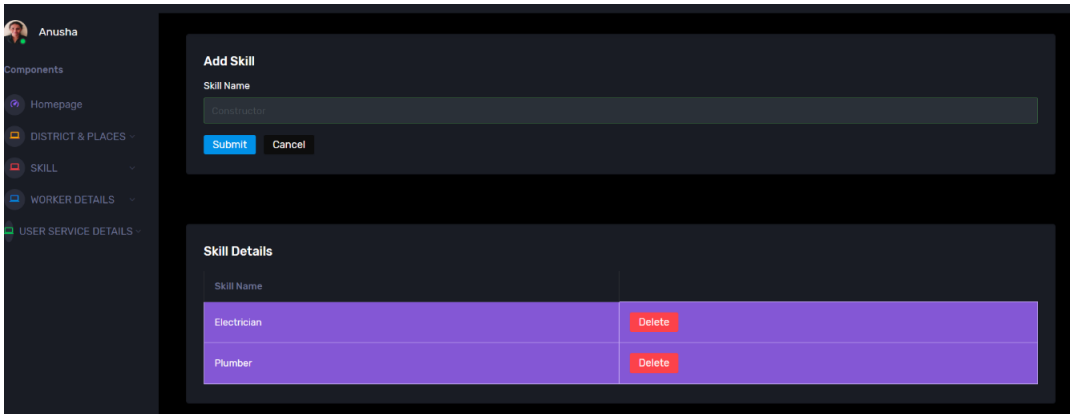

Fig 5.2.5 Admin adding the worker skills

6. ADMIN FILTERING THE WORKERS

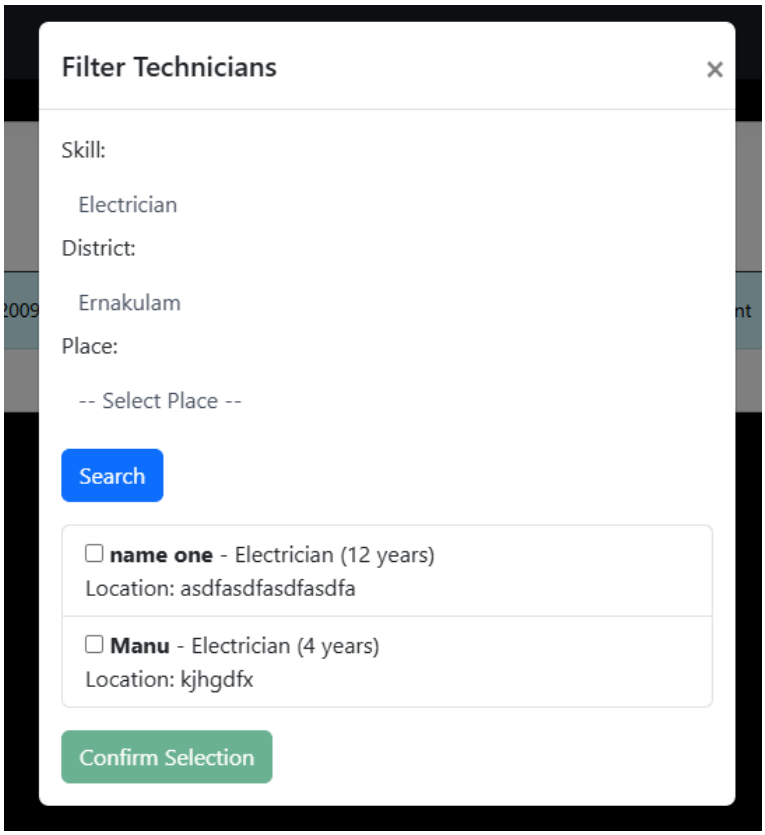

Fig 5.2.6 Searching the worker

7. USER REQUESTING FOR SERVICE

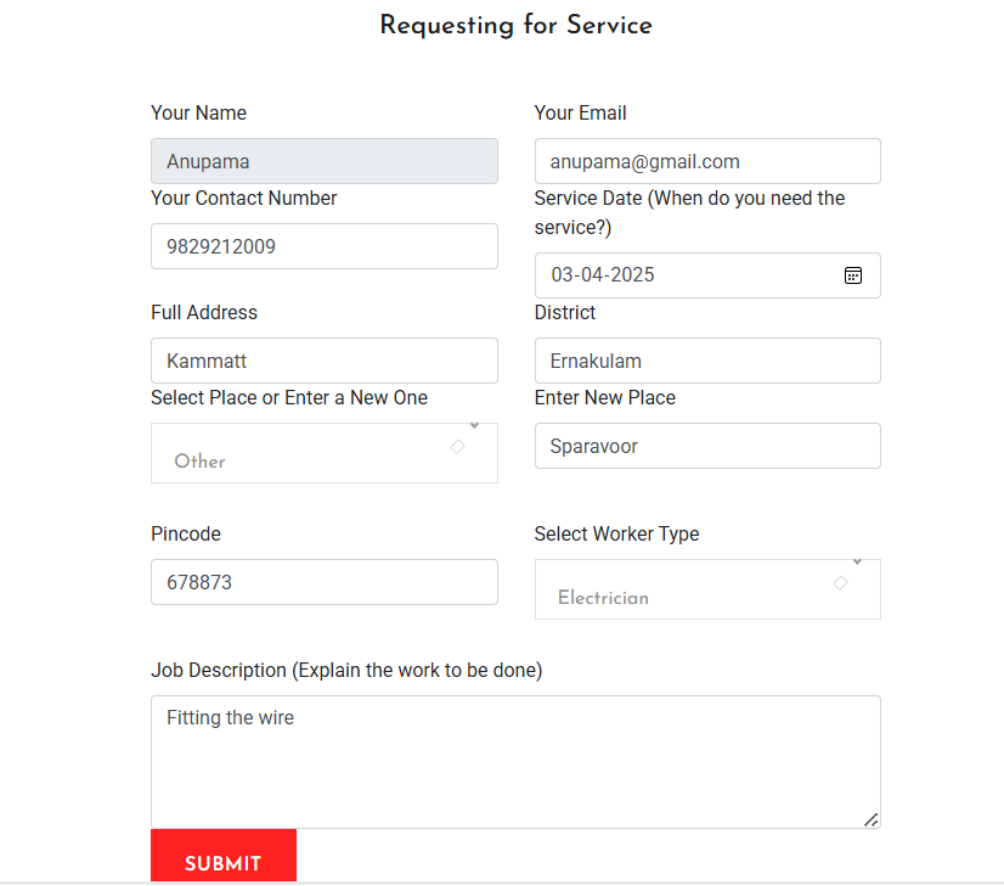

Fig 5.2.7 User Requesting for service

8. DETAILS OF USER REQUEST STATUS

User	Email	Contact	Woker	Place	District	Requested Date	Purpose	Address	Pincode	Status	Labour Name	S
Anupama	anupama@gmail.com	876545678	Electrician	Ernakulam	Ernakulam	March 14, 2025	Fix my light	Kammatt	678877	Booking Status : WORK COMPLETED	John	
Anupama	anupama@gmail.com	9829212009	Plumber	S paravoor	Ernakulam	April 10, 2025	pipe compliant	Kammatt	678877	Booking Status : ADMIN ACCEPTED	John	
Anupama	anupama@gmail.com	9829212009	Electrician	Sparavoor	Ernakulam	April 3, 2025	Fitting the wire	Kammatt	678873	Pending	pending	

Fig 5.2.8 Details of user request service status

9. WORK ASSIGNED FOR WORKER

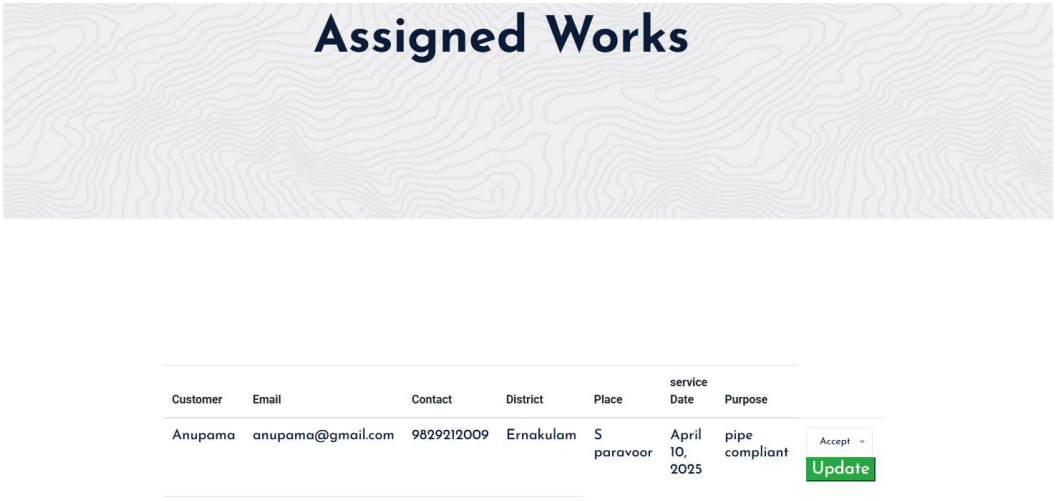

Fig 5.2.9 Work assigned for worker

10. SERVICE VIEW BILL

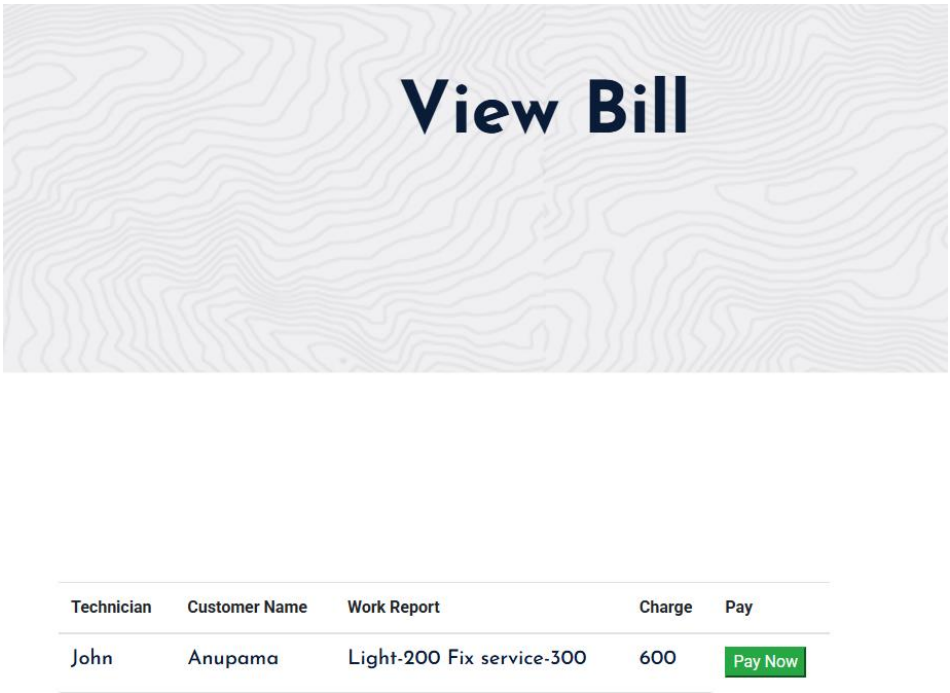

Fig 5.2.10 Worker done the service bill view

CHAPTER 6
REFERENCES

1. S. Smith and R. Johnson, *"Digital Transformation in Workforce Management,"* 2020.
   This study discusses how transitioning from manual to digital platforms has improved efficiency and tracking in labour management systems.
2. R. Patel, A. Kumar, and S. Mehta, *"Role of AI in Job Matching,"* 2021.
   The paper explores how AI algorithms optimize job matching and worker allocation based on skills and availability.
3. L. Martinez and D. Gomez, *"Real-Time Tracking for Workforce Allocation,"* 2023.
   This research highlights the impact of GPS and digital dashboards on real-time workforce monitoring.
4. T. Almeida and R. Silva, *"Scalability of Digital Labour Management Platforms,"* 2024.
   The paper emphasizes how cloud-based systems improve scalability and meet growing workforce demands.
5. M. Davis and L. Brown, *"Challenges in Traditional Workforce Management and the Need for Automation,"* 2021.
   This study reveals the limitations of manual labour systems and underscores the importance of digital transformation.
6. J. Nguyen and H. Park, *"Future Trends in Worker Supply Management,"* 2024.
   It discusses the integration of emerging tech like blockchain, AI, and IoT to enhance future labour management practices.