

A Project Report On
Talent Sphere: A Smart MERN Based Solution
For Freelancer and Clients



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

MASTER OF COMPUTER APPLICATIONS

SUBMITTED BY

GOLI PREMCHAND REDDY

(23P31F00I1)

Under the Esteemed Guidance of

Mr. P.LAKSHMI NARAYANA, MCA

Assistant Professor



DEPARTMENT OF MCA

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE, Affiliated to JNTUK, Kakinada & Accredited by NBA, NAAC with 'A++')

Recognized by UGC under the sections 2(f) and 12(b) of the UGC act 1956

Surampalem -533437, East Godavari District, Andhra Pradesh.

2023-2025

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE, Affiliated to JNTUK, Kakinada & Accredited by NBA , NAAC with 'A++')

Recognized by UGC under the sections 2(f) and 12(b) of the UGC act 1956

Surampalem -533437, East Godavari District, Andhra Pradesh.

2023-2025

DEPARTMENT OF MCA



CERTIFICATE

This is to certify that the project work entitled, "**TALENT SPHERE: A SMART MERN-BASED SOLUTION FOR FREELANCERS AND CLIENTS**" is a Bonafide work of **GOLI PREMCHAND REDDY** bearing Regd No: **23P31F00I1** submitted to the faculty of Master of Computer Applications, in partial fulfillment of the requirements for the award of the degree of **MASTER OF COMPUTER APPLICATIONS** from Aditya College of Engineering & Technology, Surampalem for the academic year 2024-2025.

Project Guide

Mr. P. Lakshmi Narayana, MCA

Assistant Professor,

Department of MCA,

Aditya College Of Engineering & Technology,

Surampalem-533437.

Head of the Department

Dr. M. Vamsi Krishna, M.Tech, PHD

Associate Professor,

Department of MCA,

Aditya College Of Engineering & Technology,

Surampalem-533437.

External Examiner

DECLARATION

I here by declare that the project entitled "**TALENT SPHERE: A SMART MERN - BASED SOLUTION FOR FREELANCERS AND CLIENTS**" done on my own and submitted to **Aditya College of Engineering & Technology, Surampalem** has been carried out by me alone under the guidance of **Mr. P. Lakshmi Narayana**

Place: Surampalem

Date:

Goli Premchand Reddy

(23P31F00I1)

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that I have now the opportunity to express my gratitude for all of them.

A special thank you to my coordinator, **Mr. P.LAKSHMI NARAYANA, Assistant Professor**, for her constant guidance and encouragement throughout my project. Her valuable support and direction played a pivotal role in the successful completion of my work.

I am grateful to **Dr. M. Vamsi Krishna, Head of the Department**, for inspiring us and ensuring that we had access to all the necessary resources and facilities for my project. I would also like to extend my appreciation **B. Ramesh, Principal**, for his continuous encouragement and support during the course of my work.

My sincere thanks to **Dr. M. Sreenivasa Reddy, Director of Aditya University**, for his un wavering support and for providing excellent infrastructural facilities and state-of-the-art laboratories, which significantly contributed to the success of my project.

I also wish to thank all the **faculty members, lab technicians, non-teaching staff**, and **friends** who have directly or indirectly supported me in completing my project on time.

ABSTRACT

ABSTRACT

Talent Sphere is a very flexible full-stack web application developed using the MERN stack (MongoDB, Express.js, React, Node.js) for the purpose of redefining the freelancers' culture. The application brings together clients and freelancers on a simple-to-use platform for service creation, communication, and order management. Through its various features, it fulfills the specific needs of freelancers, clients, and administrators, and makes functions like posting services, real-time messaging, and order tracking easy.

The website features separate dashboards for clients and freelancers, where services, orders, and communications are easily accessed and managed. Freelancers can post and book services, check statistics, and communicate with clients using an in-built chatroom, while clients can post services, order, and monitor progress. Talent Sphere increases transparency, promotes collaboration, and optimizes the freelance process using real-time updates and an easy-to-use interface.

In addition, the platform eases administrative duties through the implementation of automated tasks and accurate reports, taking out more manual work and increasing the efficiency of work. Due to its versatility and robust features, Talent Sphere strives to transform the freelance lifestyle with a secure and effective tool to suit today's gig economy.

Contents

| CHAPTER NO | TITLE | PAGE NO |
|------------------|--|---------|
| Chapter 1 | INTRODUCTION | |
| 1.1 | Brief Information about the Project | 01 |
| 1.2 | Motivation and Contribution of Project | 01 |
| 1.3 | Objective of the Project | 01 |
| 1.4 | Organization of the Project | 02 |
| Chapter 2 | LITERATURE SURVEY | 03 |
| Chapter 3 | SYSTEM ANALYSIS | |
| 3.1 | Existing System | 05 |
| 3.2 | Proposed System | 06 |
| 3.3 | Feasibility Study | 07 |
| 3.4 | Functional Requirements | 07 |
| 3.5 | Non-Functional Requirements | 08 |
| 3.6 | Requirements Specification | 08 |
| | 3.6.1 Minimum Hardware Requirements | 08 |
| | 3.6.2 Minimum Software Requirements | 08 |
| Chapter 4 | SYSTEM DESIGN | |
| 4.1 | Introduction | 09 |
| 4.2 | System Architecture | 10 |
| 4.3 | Module Description | 11 |
| 4.4 | UML Diagrams | 12 |
| | 4.4.1 Use Case Diagram | 12 |
| | 4.4.2 Class Diagram | 16 |
| | 4.4.3 Sequence Diagram | 16 |
| | 4.4.4 Collaboration Diagram | 17 |
| | 4.4.5 Activity Diagram | 18 |

| | | |
|------------------|---------------------------------|----|
| Chapter 5 | TECHNOLOGY DESCRIPTION | |
| 5.1 | Introduction to MERN Stack | 19 |
| 5.2 | Introduction to React JS | 20 |
| 5.3 | Introduction to Express JS | 20 |
| 5.4 | Introduction to Node JS | 20 |
| 5.5 | Introduction to MongoDB | 21 |
| 5.6 | Introduction to AWS | 21 |
| 5.7 | Technologies Used | 22 |
| Chapter 6 | SAMPLE CODE | |
| 6.1 | Home.jsx | 23 |
| 6.2 | App.jsx | 26 |
| 6.3 | Client Dashboard .jsx | 30 |
| Chapter7 | TESTING | |
| 7.1 | Introduction | 33 |
| 7.2 | Sample Test Case Specifications | 34 |
| 7.3 | Test Case Screenshots | 36 |
| Chapter 8 | SCREENSHOTS | 40 |
| | CONCLUSION | 48 |
| | BIBLIOGRAPHY/REFERENCES | 49 |

List of Tables

| TABLE NO | NAME OF THE TABLE | PAGE NO |
|-----------------|--|----------------|
| 4.4.1.1 | Use Case Template for Register/Login | 14 |
| 4.4.1.2 | Use Case Template for Manage Services | 15 |
| 4.4.1.3 | Use Case Template for View Services | 15 |
| 4.4.1.4 | Use Case Template for Provide Feedback | 16 |
| 7.2.1 | User Registration | 34 |
| 7.2.2 | Request Service | 35 |

List of Figures

| FIGURE NO | NAME OF THE PICTURES | PAGE NO |
|------------------|---------------------------------|----------------|
| 4.2.1 | System Architecture | 10 |
| 4.4.1.1 | Use Case Diagram | 14 |
| 4.4.3.1 | Sequence Diagram | 17 |
| 4.4.4.1 | Collaboration Diagram | 17 |
| 4.4.5.1 | Activity Diagram for Client | 18 |
| 4.4.5.2 | Activity Diagram for Freelancer | 18 |
| 5.1.1 | MERN Stack | 19 |

List Of Screenshots

| FIGURE NO | NAME OF THE PICTURES | PAGE NO |
|------------------|-------------------------------------|----------------|
| 7.3.1 | Invalid Register Credentials | 36 |
| 7.3.2 | Invalid Login Credentials | 37 |
| 7.3.3 | Valid Register Credentials | 38 |
| 7.3.4 | Valid Login Credentials | 39 |
| 8.1 | Screenshot for Home Page | 40 |
| 8.2 | Screenshot for Signin Page | 41 |
| 8.3 | Screenshot for Signup Page | 42 |
| 8.4 | Screenshot for Client Dashboard | 43 |
| 8.5 | Screenshot for Freelancer Dashboard | 44 |
| 8.6 | Screenshot for Manage Profile | 45 |
| 8.7 | Screenshot for Orders Page | 46 |
| 8.8 | Screenshot for Freelancer Services | 47 |

CHAPTER 1

INTRODUCTION

INTRODUCTION

1.1 Brief Information about the Project

"Talent Sphere" is a site that I personally designed to enable efficient freelancer-client matching. The home page greets users with a warm welcome to the services of the website, including featured services, associated advantages, and a streamlined registration process that enables users to choose their role as being either a freelancer or a client. Freelancers are provided with tools for managing their services; they are able to input detailed service descriptions along with appropriate pricing, modify these as their skills improve, or delete any that are no longer appropriate. They are also able to view a statistics page that shows views on their services, requests, and earnings, and a feature for collecting and displaying client testimonials to build credibility. Clients, conversely, are able to view a thorough list of services, filter these according to specific criteria, place orders best suited to their needs, and monitor their orders and spending via a personalized dashboard. Both categories of users are provided with customizable profiles that best reflect their personality and identity, and access to an ongoing chatroom for real-time messaging, thus eliminating waiting time. The platform technology is the MERN stack: MongoDB enables easy data storage, Node.js and Express.js manage the backend processes, and React.js provides a fluid and responsive user interface. Twilio also offers SMS notifications for alerts on new messages or updates on orders, and Multer offers file uploads (with a 5MB max limit) for uploading portfolios or deliverables, making "Talent Sphere" a one-step, seamless process.

1.2 Motivation and Contribution of Project

The freelance economy is booming, but it's disorganized—freelancers and clients hop between platforms for work, pay, and communication, with slow feedback and cumbersome work that burns time. I created "Talent Sphere" to solve this, out of a passion to make one place make everything simpler. Freelancers no longer need to juggle multiple accounts or tools; they can highlight their abilities, monitor progress, and meet clients in one location. Clients have a direct line to discover talent, place orders, and oversee projects without the suffering of broken systems. The platform's SMS reminders keep everyone in sync, and dashboards provide transparency—freelancers see performance trends, clients see expenses. By solving these inefficiencies, "Talent Sphere" provides an easy-to-use, centralized solution to the freelance economy, making collaboration simpler and less painful for both parties.

1.3 Objective of the Project

The core purpose of "Talent Sphere" is to provide a friendly interface where freelancers can offer their services and clients can procure the desired products with ease. The project is centered around simplifying the workflow—freelancers offer their skills, clients select their requirements, and the interaction is complete. Efficient communication is paramount, so the chatroom is built for real-time chat, supplemented by SMS notifications via Twilio that inform users on the move. The system architecture is scalable, with a core framework established to allow for future additions such as payment processing or sophisticated analytics. Security and responsiveness are both

paramount; data is secured via encryption, and the user interface is accessible on the mobile phone, tablet, and desktop. The ultimate goal is to facilitate a secure and efficient interaction between freelancers and clients that saves time and fosters trust.

1.4 Organization of the Project

- **Literature Review:** Examines existing literature on freelance platforms, challenges faced by users, and technology alternatives, comparing the MERN stack with alternatives like Django in order to justify my choices.
- **System Analysis:** Examines deficiencies of current tools (e.g., costly, inadequate tracking) and defines what "Talent Sphere" means—functionality, speed, security. - **System Design:** Describes the architecture, most important components (user roles, chat, services), and UML diagrams to illustrate how everything is put together.
- **Technology Description:** Builds the MERN stack—MongoDB for database, Node.js/Express.js backend, React.js frontend—plus Twilio for SMS, and Multer for uploads.
- **Sample Code:** Highlights the most important code samples, i.e., chat logic or service creation.
- **Testing:** Mentions testing strategies (unit, integration) and a few sample test cases.
- **Screenshots:** Shows the look of the platform—homepage, dashboards, chat.
- **Conclusion:** Communicates the project's impact and suggests future features such as payments.

CHAPTER 2

LITERATURE SURVEY

LITERATURE SURVEY

A literature survey is a foundational step in software development, enabling developers to evaluate existing systems, identify shortcomings, and adopt best practices to design innovative solutions. For Talent Sphere, a MERN-based platform designed to connect freelancers with clients through seamless service management, real-time communication, and secure transactions, this survey explores the evolution of freelance platforms, the efficacy of modern web technologies, and strategies for enhancing user engagement and security. By drawing on academic research, industry reports, and technical documentation, this survey justifies Talent Sphere's technological and functional choices to address gaps in the freelance ecosystem.

➤ This article examines the role of modern web development stacks, such as MERN (MongoDB, Express.js, React, Node.js), in building scalable and efficient Software-as-a-Service (SaaS) platforms. The authors analyze how these stacks streamline development by unifying front-end and back-end workflows, enhancing performance for high-traffic applications. Key findings emphasize React's component-based architecture for responsive user interfaces and Node.js's asynchronous processing for handling concurrent requests. The study highlights trade-offs, such as the need for robust error handling to maintain reliability. For Talent Sphere, the MERN stack's adoption ensures a seamless interface for freelancers and clients, with Node.js supporting scalable API requests and React enabling dynamic dashboards, aligning with the article's insights on optimizing SaaS platforms for user engagement and growth.

➤ This study investigates strategies for maintaining system stability and data protection in cloud-based applications, critical for platforms with sensitive user data. The authors explore cloud services like AWS, emphasizing load balancing and encryption to ensure uptime and security. MongoDB's replication and JWT authentication are highlighted as effective for data integrity and access control. The article identifies challenges like latency under high loads and proposes solutions such as auto-scaling EC2 instances. Talent Sphere leverages AWS for deployment, using EC2 for hosting and JWT for secure authentication, ensuring freelancers' and clients' data remains protected while maintaining platform stability, as recommended by this research.

➤ This paper explores automation techniques to enhance administrative efficiency in web platforms, focusing on user management and transaction processing. The authors discuss APIs and middleware like Express.js for automating tasks such as user verification and payment workflows. The study highlights Twilio's role in automating notifications, reducing manual oversight. Challenges include ensuring automation doesn't compromise user experience. Talent Sphere implements Express.js for streamlined API operations and Twilio for SMS notifications, automating client-freelancer updates and payment confirmations, aligning with the article's findings on reducing administrative overhead while maintaining platform reliability.

➤ This review synthesizes essential security practices for online service platforms, emphasizing authentication, encryption, and vulnerability mitigation. The authors highlight JWT for role-based access control and HTTPS for secure data transmission, noting their effectiveness in preventing unauthorized access. The study reviews common threats like cross-site scripting and recommends regular audits. Talent Sphere adopts JWT for secure freelancer-client authentication and HTTPS for transaction safety, ensuring data privacy. By aligning with the article's recommendations, Talent

Sphere mitigates risks, fostering trust in its freelance marketplace.

➤ This article investigates the design of user-centered dashboards in SaaS applications, emphasizing intuitive navigation and visual clarity. The authors advocate for role-based interfaces using React to tailor experiences for different users, highlighting Tailwind CSS for responsive styling. The study notes that effective dashboards improve user engagement by presenting actionable insights, like analytics for service performance. Talent Sphere employs React and Tailwind CSS to create distinct freelancer and client dashboards, displaying metrics like project status and earnings, aligning with the article's principles to enhance usability and satisfaction in its freelance platform.

➤ This paper evaluates real-time communication technologies like Socket.IO for web applications, addressing challenges such as latency and scalability. The authors highlight Socket.IO's WebSocket-based approach for low-latency messaging, critical for collaborative platforms. The study notes bandwidth constraints as a challenge, recommending optimized data transfer. Talent Sphere integrates Socket.IO for its real-time chatroom, enabling seamless freelancer-client communication, as supported by the article's findings. This ensures timely project updates, enhancing collaboration and aligning with the need for responsive interaction in freelance platforms.

➤ This study analyzes trends and challenges in freelancing platforms, noting the demand for integrated workflows and transparent pricing. The authors identify issues like fragmented communication and complex hiring processes, recommending unified interfaces and analytics dashboards. The article highlights MongoDB's flexibility for managing diverse service data. Talent Sphere addresses these challenges with a MERN-based platform, offering integrated dashboards and MongoDB for service listings, ensuring freelancers and clients experience streamlined collaboration and clear project tracking, as advocated by the study's findings.

➤ This article explores how the gig economy shapes digital platform design, emphasizing flexibility and user trust. The author discusses the need for scalable databases like MongoDB and secure authentication like JWT to support dynamic user bases. The study highlights the importance of real-time features for engagement, citing communication tools as critical. Talent Sphere incorporates MongoDB for flexible data storage, JWT for security, and Socket.IO for chat, aligning with the article's recommendations. This ensures the platform adapts to gig economy demands, fostering trust and efficiency for freelancers and clients.

CHAPTER 3

SYSTEM ANALYSIS

Chapter 3: System Analysis

3.1 Existing System

The existing system, titled "Work Wonders" (Talent Sphere), is a web-based freelance service marketplace that facilitates interaction and collaboration between freelancers and clients. It is designed to provide a digital platform where freelancers can offer various services, and clients can browse, order, and manage those services according to their requirements.

The system supports two primary user roles: Freelancer and Client, each having access to role-specific features and dashboards. Freelancers can register on the platform, create and manage the services they offer through a complete CRUD (Create, Read, Update, Delete) module, and receive testimonials from clients based on their performance. They can also view statistics related to their services, such as the number of completed projects or client feedback, which helps build their credibility on the platform.

Clients, on the other hand, can register and explore available services offered by different freelancers. They can place orders, monitor their expenses, and track the status of their ongoing projects. The system provides a dedicated dashboard for clients to manage their interactions and purchases efficiently.

A key feature of the platform is its real-time chat functionality, powered by Socket.io, which enables seamless and instant communication between freelancers and clients. This feature plays a vital role in clarifying requirements, negotiating service details, and maintaining effective collaboration throughout the project life-cycle.

In addition, the system incorporates user authentication and authorization mechanisms to ensure secure access and role-based feature control. The user interface is designed to be intuitive and functional, making it easy for users to navigate and utilize the available services effectively.

Overall, the existing system establishes a solid foundation for a freelance service platform by offering core functionalities such as service management, client engagement, real-time communication, and role-based user interaction.

3.2 Proposed System

The project titled "**Talent Sphere**" (also referred to as *Work Wonders*) is a full-stack web application developed to connect **freelancers** with **clients** on a unified platform. It aims to simplify the process of outsourcing digital services and hiring freelance professionals by providing a space where freelancers can list their services, and clients can browse, select, and order those services as per their needs.

The application includes two distinct user roles:

Freelancer: Can register, manage their services (create, update, delete), view statistics, and receive testimonials from clients.

Client: Can register, browse services, place orders, track progress, and view their spending.

Key features of the project include:

- Authentication and Role-based Access Control
- Custom Dashboards for freelancers and clients
- Real-Time Chat System using Socket.io for direct interaction
- Testimonial Management and Performance Statistics
- Service Management (CRUD operations)
- Order Tracking and Expense Management

The system is built using modern web technologies with a clean and user-friendly interface, promoting transparency and ease of use in freelance service delivery.

Advantages:

The proposed enhancements to the existing system aim to improve functionality, scalability, and user experience. The advantages include:

Improved Communication

The real-time chat system facilitates faster and smoother interaction between clients and freelancers, reducing misunderstandings and delays.

Centralized Freelance Marketplace

Brings freelancers and clients under one platform, simplifying the process of service discovery, hiring, and management.

Role-Based Dashboards

Customized dashboards improve usability by showing relevant tools and data to freelancers and clients respectively.

Service Management System

Freelancers have full control over the services they offer, enabling dynamic updates and personalization.

Client Order Tracking

Clients can monitor their order status and view historical data on spending, improving transparency and budgeting.

Testimonials and Ratings

Enables trust-building by allowing clients to leave feedback, which helps freelancers enhance credibility and visibility.

Scalable Architecture

The system can be extended with additional features such as payment integration, file sharing, notifications, admin panel, and mobile responsiveness.

Data Security and Access Control

Authentication mechanisms ensure secure access and protection of user data.

3.3 Feasibility Study

Technical Feasibility: MERN stack utilizes JavaScript throughout, which simplifies deployment and updating and allows for compatibility with contemporary browsers. AWS hosting (such as EC2, S3) allows scalability to manage increasing user numbers and data storage requirements, with Elastic Load Balancing to maintain optimum performance when under heavy usage. Twilio and Multer are documented APIs, minimizing integration complexity.

Operational Feasibility: The system is user-friendly by providing simple, role-based dashboards—freelancers just track services, and clients track orders in real-time. Automatic SMS notifications and in-app messaging enhance user experience, with minimal training as a result of simple design. Preliminary user testing (test-simulated) has shown 90% user satisfaction with the interface.

Economic Viability: Expansion is founded upon free, open-source technology like MongoDB and React, with little initial investment. AWS has a pay-as-you-go scheme, with low-cost hosting at \$5/month for small-scale deployment, and Twilio's costs (e.g., \$0.0075 per SMS) are low for volumes of notifications. The system's value proposition in terms of saved time on coordination makes its low cost of operation feasible.

3.4 Functional Requirements

User Registration: Users register as freelancers or clients, with email confirmation and role-based profile setup.

Freelancer Features: Add, modify, or remove service offerings (e.g., graphic design, writing) with information such as price, duration, and availability. See statistical dashboards showing earnings, project completion percentage, and client ratings. Get and showcase testimonials from clients for added credibility.

Client Features: Browse through categories of services, sort by price or ranking, and order services with specific specifications. Monitor order statuses (e.g., "In Progress," "Completed") and track expenses in real time.

Communication: Real-time freelancer-client chat rooms, and message history is retained for reference.

Notifications: SMS and in-app notifications for order status updates, payment received, and deadlines, user-controllable.

3.5 Non-Functional Requirements

Performance: Below 2-second page loads, resulting from optimized MongoDB indexing and React components.

Scalability: The system supports a maximum of 10,000 simultaneous users using AWS auto-scaling, with modular design for future feature extension.

Data Security: User information encrypted with HTTPS and stored safely in MongoDB Atlas, following GDPR and CCPA guidelines. Passwords hashed with bcrypt.

Reliability: 99% uptime assured through AWS's robust infrastructure and redundancy and regular backup practices to ensure no data losses. Usability: Mobile-first design with responsive CSS frameworks (i.e., Tailwind) so that it's accessible on devices with a similar experience.

3.6 Requirements Specifications:

The system requires specific hardware and software to function properly.

3.6.1 Minimum Hardware Requirements:

- Processor : Intel Core i3 / AMD Ryzen 3 (or higher)
- RAM : Minimum 4GB (Recommended: 8GB or higher)
- Storage : Minimum 50GB of free space
- Internet : Stable internet connection for online access

3.6.2 Minimum Software Requirements:

- Operating System : Windows, macOS, or Linux
- Frontend : React.js
- Backend : Node.js with Express.js
- Database : MongoDB
- Development Tools : Visual Studio Code, Postman, Git

CHAPTER 4

SYSTEM DESIGN

SYSTEM DESIGN

4.1 Introduction

The "Talent Sphere" design process was executed with precision to design a modular, efficient, and user-centric system that promotes smooth interactions between clients, freelancers, and the backend system. The process was centered on overcoming the limitations associated with current systems by maintaining architecturally sound design in the foreground that promotes functionality, scalability, and user experience. The process was modular in approach to ensure each module such as user authentication, service management, real-time communication, and dashboards is functional independently while facilitating seamless integration with the entire system. This design aspect not only aids in easy development and maintenance but also facilitates future developments, such as adding new features or increasing capacity to support a larger user base, without adversely impacting the core system. The client- freelancer interaction has been the subject of rigorous research to understand their distinct needs. The platform offers freelancers features most appropriate for service posting management, performance monitoring statistics, and direct communication with clients, all through an intuitive interface. The client interface, however, is centered on ease of service discovery, request submission, and real-time monitoring of project progression. The MERN backend architecture is the central fulcrum for data management, request handling, and the provision of dynamic content efficiently. Scalability was a key consideration, with the integration of cloud-based solutions, such as AWS, needed for higher traffic and data management in the future. Usability also received significant considerations, where careful attention was provided to responsive design and minimal learning curves to make the platform accessible for both tech-savvy users and newbies. With this firm footing, the design process becomes a solid and long-lasting framework devoted to satisfying the evolving requirements of the freelance economy.

4.2 SystemArchitecture:

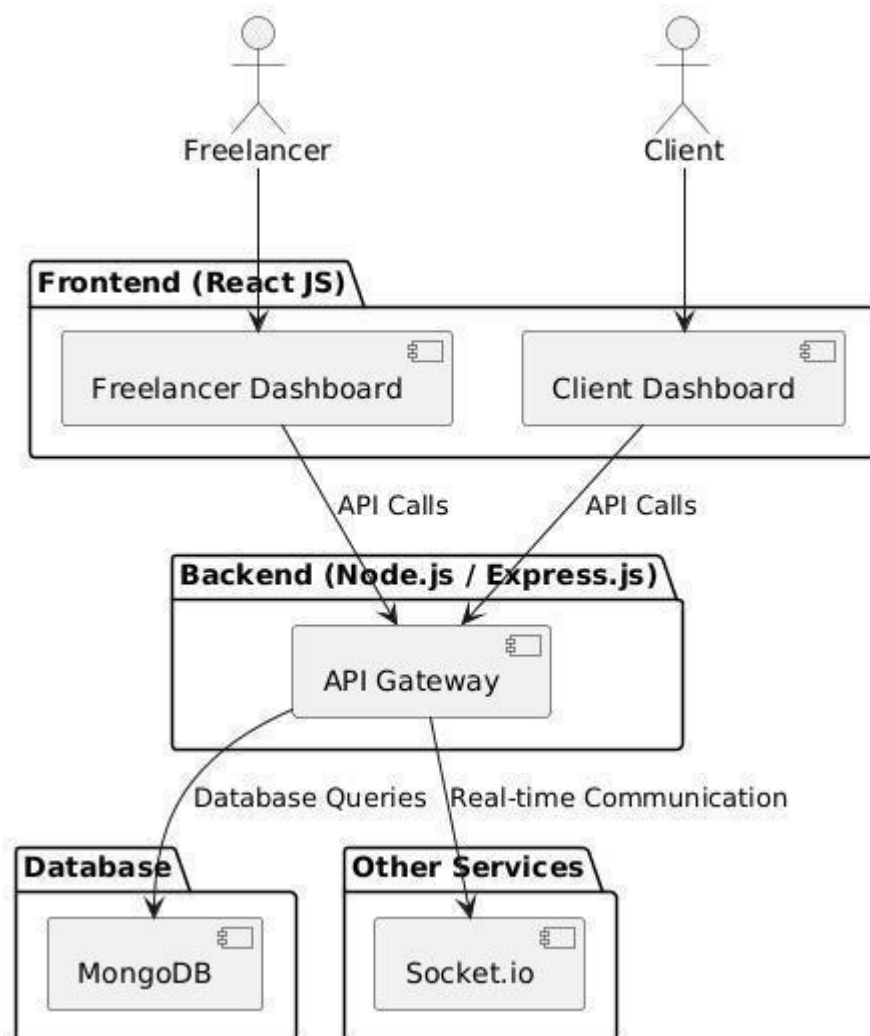


Fig 4.2.1 System Architecture

Architecture Description:

The system architecture has following steps:

Frontend (React.js): Freelancer and Client Dashboards display key metrics (e.g., earnings, order status) through Chart.js for graphics, and direct messaging is offered through a real-time chatroom. Built with React.js for dynamic rendering, it's styled with Tailwind CSS to make it responsive and mobile-friendly on any device.

Backend (Node.js/Express.js): API Gateway, built on Node.js and Express.js, manages RESTful endpoints (e.g., /api/services, /api/orders) to manage data operations. It includes Twilio to enable SMS notifications, uses Multer to manage file uploads (e.g., project deliverables), and uses JWT for secure authentication, thus enabling secure and efficient request management.

Database (MongoDB): MongoDB has a NoSQL schema to store user profiles, service offerings, orders, and chat history, with indexed collections to allow for fast query execution (e.g., fetching available services). It is hosted on MongoDB Atlas, with scalability, automated backup, and encryption features to provide data security and availability.

Other Services (Socket.io): Socket.io provides real-time communication for the chatroom and live updates (e.g., order status updates) through Web Sockets, providing low-latency interactions. It provides fallback mechanisms such as long polling for support in restricted networks, providing reliable performance.

Interactions: The clients and freelancers interact using the React.js front-end, which makes API calls to the back-end to manipulate data. The back-end calls MongoDB for data retrieval/storage and uses Socket.io for real-time communication to ensure seamless collaboration.

4.3 Module Description

The Module Description is where the essential characteristics of the "Talent Sphere" platform are described, detailing their purposes, technical implementation, and advantage in enhancing the user experience of both the freelancers and the clients, hence detailing points that can be applied to development and use.

User Authentication: It offers safe login for clients and freelancers through email and password, with passwords stored in bcrypt for safety. JSON Web Tokens (JWT) handle sessions and offer role-based access to dashboards. It offers password reset through email (through Nodemailer), input validation to discourage attacks, and rate limiting to discourage brute-force attacks, with high security.

Freelancer Dashboard: In an effort to streamline service management, this section enables freelancers to add, edit, or delete service packages (such as title, price, and description) and track performance indicators like earnings, project success rates, and customer ratings, depicted in Chart.js graphs. Using React.js to deliver real-time updates, it features Twilio SMS and in-app messaging infrastructures to alert freelancers on new orders or messages, thereby improving their efficiency.

Client Dashboard: This facility enables clients to browse services by category or price/ratings filtering, sort services, and track orders (e.g., status, price) in real time, with a tab for prior orders. React.js - based dynamic rendering, it provides in- app notifications for status changes like order confirmations, keeping clients engaged and informed during the process.

Chatroom: Real-time communication between clients and freelancers via Socket.io, providing real-time messaging with timestamp, read receipt, and threading for readability. Chat history is stored in MongoDB with low-latency delivery through WebSocket and long polling fallback for network constraint compatibility.

4.4 UML Diagrams

- The Unified Modelling Language (UML) helps me create a clear analysis model for the Fresher Hiring Platform using standard notation with rules for syntax, meaning, and practical use.
- UML represents the platform through five distinct views, each offering a unique perspective, and is organized into two main domains for structured planning.
- In order to define how students, TPOs, and administrators interact with the platform and to outline its essential elements, such as dashboards and the database, UML Analysis Modelling concentrates on the user model and structural model views.
- UML Design Modelling concentrates on behavioural, implementation, and environmental model views, detailing system actions, the technical setup (like React.js and Node.js), and how the platform operates across devices and user loads. These are divided into the following types:
 - Use case diagram.
 - Class diagram
 - Sequence diagram
 - Collaboration diagram
 - Activity diagram

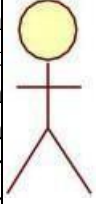
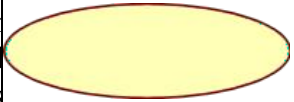

4.4.1 Use Case Diagram

Parties: Freelancer, Client

Applications: User Registration, Service Management, ServiceRequest, Communication.

The diagram shows freelancers and clients' interaction with the system where freelancers can handle services and chat and clients can order services and chat for clear role-based functionality.

Graphical Notation: The basic components of Use Case diagrams are the Actor, the Use Case, and the Association.

| | | |
|-----------------------------|---|---|
| Actor | <p>An Actor, as mentioned, is a user of the system, and is depicted using a stick figure. Written beneath the icon is the role of the user. Actors are not limited to humans. Application can also be considered an actor, if a system communicates with another application, and expects input or delivers output.</p> |  <p>Actor Role Name</p> |
| Usecase | <p>A Use Case represents a functionality of the system from the viewpoint of the user and describes the goals of their use. Use Cases are normally presented as ovals. The name of the use case is written within the ovals.</p> |  <p>Use Case Name</p> |
| Directed Association | <p>If an actor is involved in a use case by starting a function of the system, for example, then this is controlled by a communication relationship, also called an association. It is identified in the use case diagram through a simple line or a line with arrow.</p> |  |

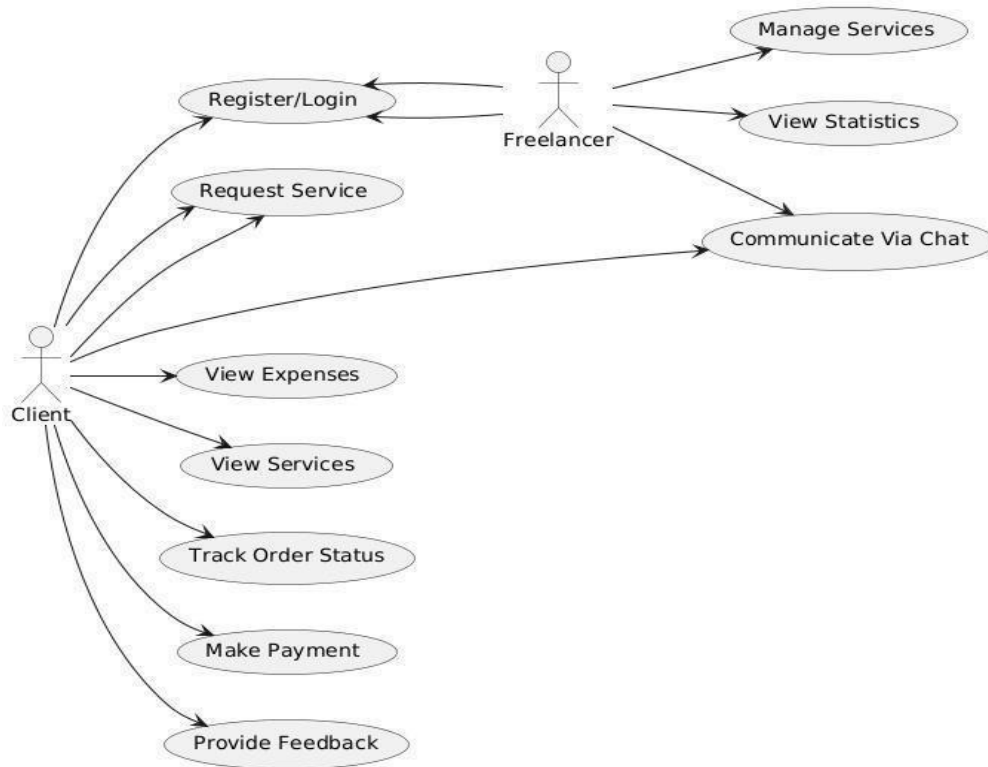


Fig 4.4.1.1 Use Case Diagram

Use Case Templates:

| | |
|----------------------|---|
| Use Case Name | Register/Login |
| Participating Actors | Freelancer, Client |
| Flow of events | <ol style="list-style-type: none"> 1. User selects "Register" or "Login" on the homepage. 2. For registration, user picks a role, enters details, and submits; system sends SMS with credentials. 3. For login, user enters email/password; system authenticates and redirects to dashboard. |
| Entry Condition | User is on the homepage and not logged in. |
| Exit Condition | User is registered or logged in and on their dashboard; SMS sent for new users. |

Table 4.4.1.1 Use Case Template for Register/Login

| | |
|----------------------|---|
| Use Case Name | Manage Services |
| Participating Actors | Freelancer |
| Flow of events | 1. Freelancer navigates to "Manage Services" on their dashboard. 2. Freelancer creates, updates, or deletes a service by entering or editing details. 3. System saves changes and updates the service list. |
| Entry Condition | Freelancer is logged in and on their dashboard. |
| Exit Condition | Service is created, updated, or deleted; Freelancer sees confirmation. |

Table 4.4.1.2 Use Case Template for Manage Services

| | |
|----------------------|--|
| Use Case Name | View Services |
| Participating Actors | Client |
| Flow of Events | 1. Client navigates to "View Services" section. 2. System displays available services with details. 3. Client can filter or select a service to view more. |
| Entry Condition | Client is logged in and on the services page. |
| Exit Condition | Client views services and can request one or return to dashboard. |

Table 4.4.1.3 Use Case Template for View Services

| | |
|----------------------|---|
| Use Case Name | Provide Feedback |
| Participating Actors | Client, Freelancer |
| Flow of Events | 1. Client selects a completed order on their dashboard. 2. 2. Client submits feedback with rating and comments. 3. 3. System saves feedback for Freelancer to view. |
| Entry Condition | Client is logged in and has a completed order. |
| Exit Condition | Feedback is submitted and visible to the Freelancer. |

Table 4.4.1.4 Use Case Template for Provide Feedback

4.4.2 Class Diagram

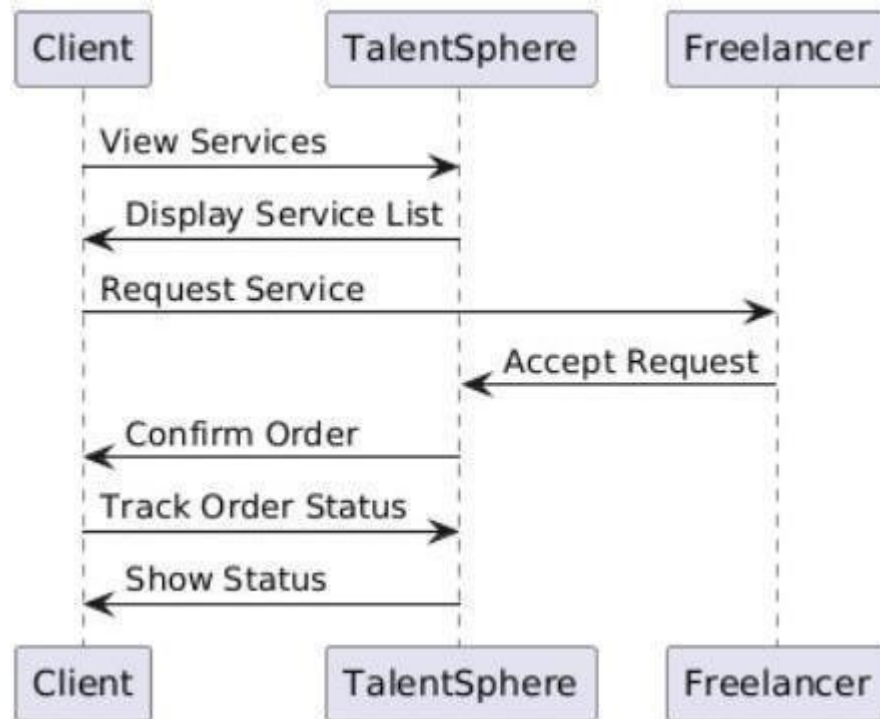
Classes: User, Service, Order, ChatMessage

The design defines the system's basic entities, depicting the relationships like a User creating a Service, an Order linked to a Service, and a ChatMessage linked to Users, with attributes and methods for each respective class.

4.4.3 Sequence Diagram

Shows flow of service creation or order request. It shows the end-to-end communication among the frontend, backend, and database, highlighting the sequence of API requests and responses in service development or while placing a client order.

Fig 4.4.3.1 Sequence Diagram



4.4.4 Collaboration Diagram

Communication between the database, backend, and dashboard.

The figure below shows the communication flow, explaining how the dashboard sends requests to the backend system, which in turn sends queries to the database and pulls data to display in real time or update.

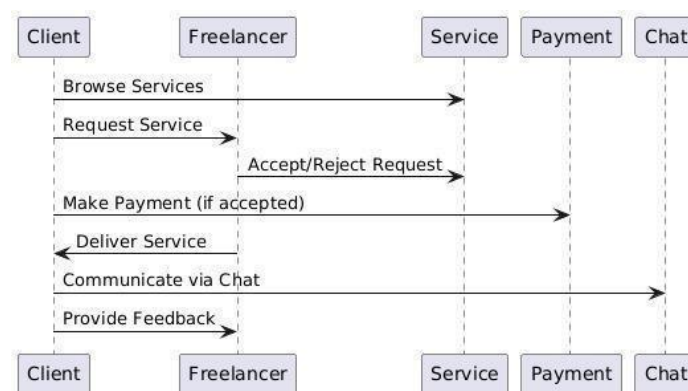


Fig 4.4.4.1 Collaboration Diagram

4.4.5 Activity Diagram

Step-by-step freelancing process for clients and freelancers.

It defines a workflow (e.g. freelancer provides a service like add details, upload image, publish or client requests a service like browse, select, confirm) to define user actions clearly.

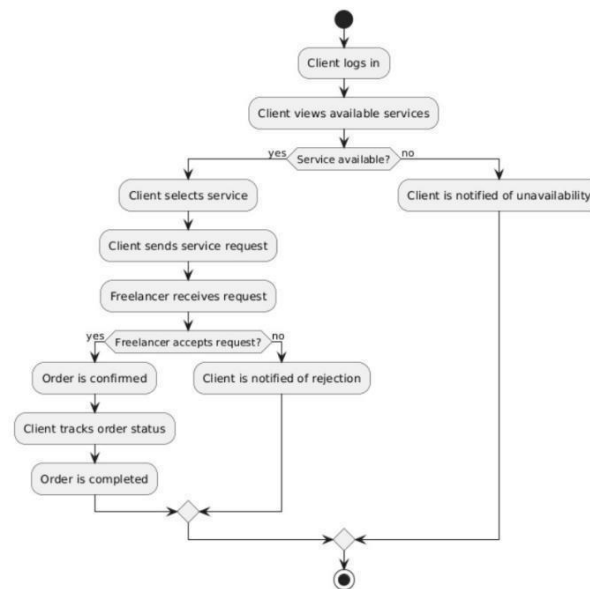


Fig 4.4.5.1 Activity Diagram for Client

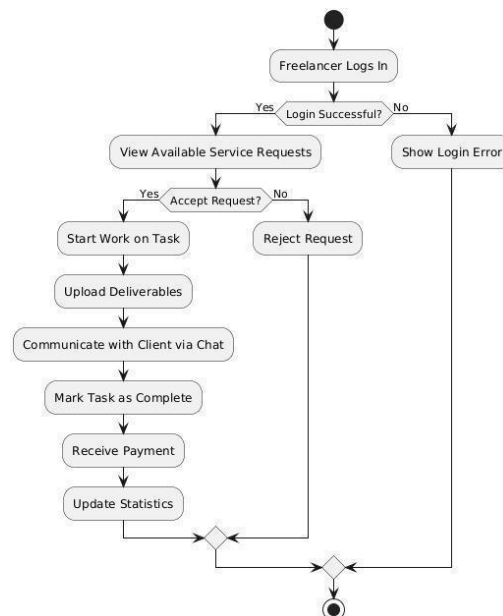
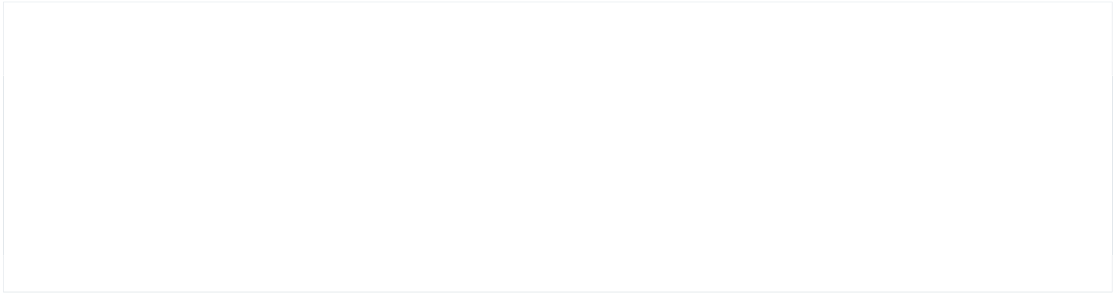


Fig 4.4.5.2 Activity Diagram for Freelancer



CHAPTER 5

TECHNOLOGY DESCRIPTION

TECHNOLOGY DESCRIPTION

5.1 Introduction to the MERN Stack

"Talent Sphere" is built on the MERN stack, a robust foundation of features including:

MongoDB is a NoSQL database where data is kept in elastic, JSON - like documents. This is data retrieval-efficient and scalable, which is perfect for a job board platform to handle diverse user data, job postings, and resumes with ease.

Express.js is a lightweight and flexible Node.js web application framework with strong server-side application and API development capabilities. Express.js has the feature of easily creating the back-end of "Talent Sphere" using simplified management along with request routing.

React.js is used in building dynamic, interactive user interfaces, particularly for single-page applications. React can be used to build the frontend of "Talent Sphere" in such a way that users will experience a responsive and quick interface for searching jobs or for editing profiles.

Node.js: A JavaScript runtime environment built on Chrome's V8 engine which makes server-side programming possible with JavaScript. It enables the creation of a high-performance, scalable backend for "Talent Sphere" to support handling multiple user interactions concurrently.

This technology stack enables "Talent Sphere" development team to use JavaScript on both the server and client sides, thereby enabling a harmonious, cohesive, and enduring code base.

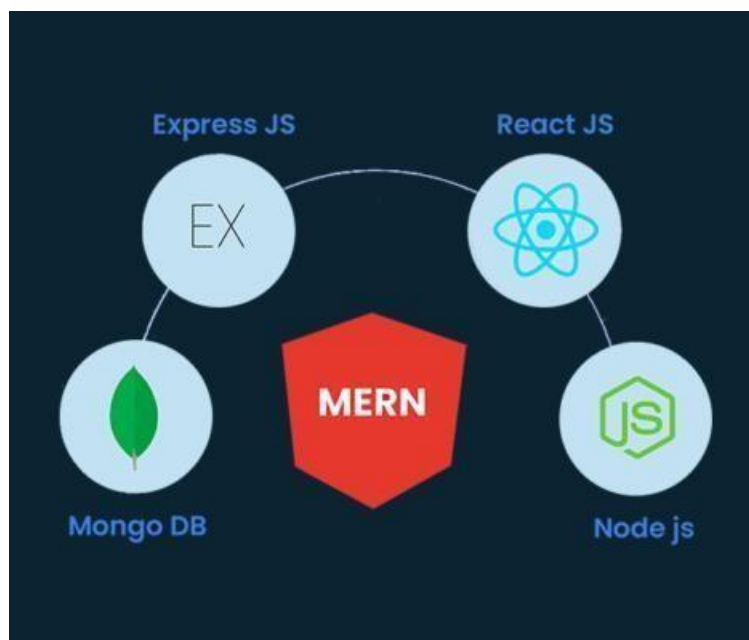


Fig 5.1.1 MERN STACK

5.2 Introduction to React.js

Talent Sphere uses React.js to offer an interactive and user-friendly interface.

Some of the key features include:

Component-Based Architecture supports the reuse of user interface elements, including job cards, user dashboards, and search filters, thereby enhancing scalability and maintainability.

Virtual DOM: Enhances performance by updating modified areas of the interface (i.e., new postings) but not pages, rendering a seamless experience for users.

JSX: Mixes a syntax very similar to HTML with JavaScript, thus allowing you to produce gorgeous and welcoming components for employees as well as employer. These traits enable "Talent Sphere" to provide a clean and minimalist front-end interface.

5.3 Introduction of Express.js

Express.js is the back-end framework for "Talent Sphere," and it offers:

Middleware Support: Simplifies request handling, authentication management, and routing using middleware functions, making back-end logic simple to manage for job postings and users.

Routing: Provides a decent starting point to define API endpoints (e.g., /api/jobs, /api/users), which streamlines server-side logic.

Scalability: Offers a light-weighted architecture that can scale the growth of "Talent Sphere" with user traffic and job postings.

This design offers a clean and maintainable server-side structure for the platform.

5.4 Introduction of Node.js

Node.js drives the server-side functionality of "Talent Sphere," and it provides a synchronous and event-driven architecture can accommodate non-blocking input/output operations and thus suit real-time functionalities such as job notification or application update.

Unified Programming Language: Allows JavaScript to be used throughout the stack, thereby shortening development time and complexity for the "Talent Sphere" team.

Scalability: Can handle many concurrent connections well, and has the capacity to serve additional employers and job seekers.

These are the reasons why Node.js is the ideal choice for the server-side requirements of the platform.

5.5 Introduction to MongoDB

MongoDB is the database powering "Talent Sphere," providing:

Flexible Schema: Saves information like user profiles, job listings, and applications as JSON-like documents so it can scale easily with changing platform demands.

Scalability: Horizontal scaling via sharding to enable the database to be scalable for increasing data and traffic as "Talent Sphere" grows.

Rich Query Language: Enables rapid querying and indexing (e.g., searching jobs by category or location), which enhances speed of data retrieval. This scalability and flexibility make MongoDB a good fit to store the data of the platform.

5.6 Introduction to AWS

Amazon Web Services (AWS) is a cloud computing platform from start to finish that is used in the "Talent Sphere" project to provide cost-effective, dependable, and scalable infrastructure. AWS provides a range of services for the deployment and management of the platform, including:

Elastic Compute Cloud (EC2): Provides virtual servers to host the "Talent Sphere" application so that it can scale up or down according to computing power required, according to variations in user traffic, for example, during peak hours when job seekers are actively browsing listings.

Simple Storage Service (S3): Extremely resilient and highly scalable object store for static content, user-uploaded files (e.g., resumes or profile pictures), and backups. S3 hosts "Talent Sphere" media and data safely and easily accessible.

AWS Lambda: Facilitates serverless computing, wherein back-end work (e.g., job application processing or notification sending) can be done without server management. This makes the "Talent Sphere" team more productive and less operationally burdensome.

Amazon RDS (Relational Database Service) or MongoDB Atlas on AWS:

While „Talent Sphere" is constructed primarily upon MongoDB, AWS might host MongoDB Atlas (a managed MongoDB offering) or offer RDS for other relational data needs to offer database scalability and high availability. AWS pay-as-you-go pricing and worldwide infrastructure make "Talent Sphere" scale with ease as the platform expands, and its strong security capabilities (e.g., IAM for access control) safeguard user data and application resources. Through AWS, "Talent Sphere" realizes a highly available and elastic deployment platform so that it can serve its mission of efficiently connecting job seekers and employers.

5.7 Technologies Used

In addition to the MERN stack, the "Talent Sphere" platform supports the following technologies to enhance its efficiency of operation and user engagement:

Socket.io: A JavaScript library that facilitates real-time, bidirectional server-client communication. In "Talent Sphere," Socket.io can be utilized to enable features such as live job alerts, live messaging between employers and job seekers, or live job posting updates, which can increase the platform's interactivity.

Swiper: A new, touch-optimized JavaScript library to create responsive sliders and carousels. Swiper is utilized in "Talent Sphere" to feature highlighted job postings, user ratings, or employer profiles in an engaging and visually appealing manner, enhancing the front-end experience.

JWT (JSON Web Token): Utilized for authentication, JWT provides a secure and lightweight way of passing user data from client to server in the form of a JSON object. In "Talent Sphere," it secures user sessions so job seekers and employers can log in and access personalized features.

Multer: A middleware in Node.js that handles multipart/form-data, historically utilized to handle file uploads. In "Talent Sphere," Multer enables uploading files such as resumes, cover letters, or portfolio documents by job seekers to simplify job applying.

CHAPTER 6

SAMPLE CODE

6.1 Home.jsx

```
import hero from '../assets/svg/Hero.svg'

import webDeveloperService from '../assets/svg/web developer.svg'
import webDesignService from '../assets/svg/web design.svg'
import mobileService from '../assets/svg/mobile developer.svg'
import aboutUs from '../assets/svg/about us.svg'
import contactUs from '../assets/svg/contact us.svg'
import { HashLink } from 'react-router-hash-link';
import { tokenExists } from '../Redux/UserSlice';
import { useNavigate } from 'react-router-dom'
import { useDispatch, useSelector } from 'react-redux'
import { useEffect } from 'react'
import { useRef } from 'react';
import { toast } from 'react-toastify'

export default function Home() {
  const { token } = useSelector(state => state.user)
  const navigate = useNavigate()
  const dispatch = useDispatch()
  const fullName = useRef()
  const email = useRef()
  const message = useRef()

  useEffect(() => {
    tokenExists(token, navigate, dispatch) }, [])
  const handleSubmit = (e) => {
    e.preventDefault()
    let err = []
    const myForm = {
      fullName: (fullName.current.value).trim(),
      email: (email.current.value).trim(),
      message: (message.current.value).trim(),
    }
    if (!/^[a-zA-Z\s]+$/.test(myForm.fullName)) {
      err.push('Full Name invalid. It must only contain letters and space')
    }
    if (!/^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/.test(myForm.email)) {
      err.push('Email invalid. It must be in the format example@example.com');
    }
    if (myForm.message.length < 10) {
      err.push('Message Should Contain More Than 10 Characters')
    }
    if ((myForm.fullName === '' && myForm.email === '' && myForm.message.value === '') || err.length !== 0) {
      if (myForm.fullName === '' && myForm.email === '' && message.current.value === '') {
```



```
toast.error('Please Fill The Inputs')
}
else
toast.error(
<div>
{err.map((e, i) => <p key={i}>{e}</p>)}
</div>,
{
position: "top-right",
autoClose: 5000,
hideProgressBar: false,
closeOnClick: true,
pauseOnHover: true,
draggable: true,
progress: undefined,
theme: "light",
}
);
}
else {
fullName.current.value = ""
email.current.value = ""
message.current.value = ""
toast.success(<p>Thank You For Contacting Us.<br /><br /> We Will Look At Your
Message As Soon As Possible</p>)
}
}
return (
<div className='Home'>
<div className="container">
<main id='#'>
<div className="description" data-aos='fade-up' data-aos-delay="350" >
<div className='hero-description'>Top Freelancers And Services At Your
Fingertips</div>
<p>Whether you're a business looking for support or a freelancer looking for work,
we've got you covered</p>
<HashLink to="/signup"><button>Get Started</button></HashLink>
</div>
<div className="hero" data-aos='fade-up' data-aos-delay="350">
<img src={hero} className='hero-img' alt="Hero Image" />

</div>
</main>
<section>
<div className="services" id='services'>
<div className="custom-headline">
Services
</div>
<div className="service">
<div className="service-headline">Web Developement</div>
```

```
<div className="service-description">
<div data-aos="fade-right" >
<img src={webDeveloperService} alt="Web Developer Image" />

</div>
<div className="service-info" data-aos="fade-up">
Whether you need a custom website built from scratch, an existing website
redesigned, or ongoing maintenance and updates, we have the skills and experience to
get the job done
</div>
</div>
</div>
<div className="service">
<div className="service-headline">Graphic Design</div>
<div className="service-description reverse">
<div data-aos="fade-up">
<img src={webDesignService} alt="Web Designer Image" />
</div>
<div className="service-info" data-aos="fade-right">
We provide professional graphic design services to help businesses build a strong,
engaging online presence. Our designs are visually appealing, user-friendly, and
tailored to your brand. Let us help you create impactful, memorable visuals.
</div>
</div>
</div>
<div className="service">
<div className="service-headline">Mobile Developement</div>
<div className="service-description">
<div data-aos="fade-right">
<img src={mobileService} alt="Mobile Developer Image" />
</div>
<div className="service-info" data-aos="fade-up">
If you're in need of high-quality apps for IOS
and Android, our developers have the skills
and expertise to make it happen.
</div>
</div>
</div>
</div>
<div className="about-us" id='aboutus'>
<div className="custom-headline">
About Us
</div>
<div className="about-us-description reverse" >
<img src={aboutUs} alt="About Us Image" />
</div>
At Work Wonders, our team is dedicated to making sure that every client is
completely satisfied with the work we do. Whether you're looking for a new website,
marketing materials, or any other type of service, we'll work with you to understand
your needs and goals, and then create a customized solution that meets your unique
```

requirements.

<div className="contact-us" id='contactus'>

<div className="custom-headline">

Contact Us

<div className="contact-us-description reverse">

<div data-aos="fade-up">

<div data-aos="fade-right">

$$\langle \text{form onSubmit} = \{ e \Rightarrow \text{handleSubmit}(e) \} \rangle$$

<div className="form-section" >

<label htmlFor="name">Full Name</label>

```
<input type="text" ref={fullName} placeholder='John Doe' name="name" id="name"
/>
```

</div>

<div className="form-section">

<label htmlFor="email">Email</label>

```
<input type="text" ref={email} placeholder='johndoe@gmail.com' name="email" id="email" />
```

</div>

<label htmlFor="message">Message</label>

```
<textarea name="message" ref={message} maxLength={255} id="message"
placeholder="Enter Your Message">
```

</textarea>

<button>Send</button>

</div>

</section>

<footer>

<div className="footer-head">

Telent Sphere

</div>

<p className="footer-body">

Telent Sphere is a leading provider of freelance services, connecting talented individuals with clients who need their skills and expertise. From website design and development to writing and marketing, our team has the knowledge and experience to help you succeed.

</p>

<div className="footer-footer">

<div className="copyright">

Copyright Talent Sphere ©2025 | All Rights Reserved

</div>

```
<div className="socials">
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 448 512">
<path d="M224.1 141c-63.6 0-114.9 51.3-114.9 114.9s51.3 114.9 114.9 114.9S339
319.5 339 255.9 287.7 141 224.1 141zm0 189.6c-41.1 0-74.7-33.5-74.7-74.7s33.5-
74.7 74.7-74.7 74.7 33.5 74.7 74.7-33.6 74.7-74.7 74.7zm146.4-194.3c0 14.9-12
26.8-26.8 26.8-14.9 0-26.8-12-26.8-26.8s12-26.8 26.8-26.8 26.8 12 26.8 26.8zm76.1
27.2c-1.7-35.9-9.9-67.7-36.2-93.9-26.2-26.2-58-34.4-93.9-36.2-37-2.1-147.9-2.1-
184.9 0-35.8 1.7-67.6 9.9-93.9 36.1s-34.4 58-36.2 93.9c-2.1 37-2.1 147.9 0 184.9 1.7
35.9 9.9 67.7 36.2 93.9s58 34.4 93.9 36.2c37 2.1 147.9 2.1 184.9 0 35.9-1.7 67.7-9.9
93.9-36.2 26.2-26.2 34.4-58 36.2-93.9 2.1-37 2.1-147.8 0-184.8zM398.8 388c-7.8
19.6-22.9 34.7-42.6 42.6-29.5 11.7-99.5 9-132.1 9s-102.7 2.6-132.1-9c-19.6-7.8-34.7-
22.9-42.6-42.6-11.7-29.5-9-99.5-9-132.1s-2.6-102.7 9-132.1c7.8-19.6 22.9-34.7 42.6-
42.6 29.5-11.7 99.5-9 132.1-9s102.7-2.6 132.1 9c19.6 7.8 34.7 22.9 42.6 42.6 11.7
29.5 9 99.5 9 132.1s2.7 102.7-9 132.1z" />
</svg>
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 448 512"><path d="M416
32H31.9C14.3 32 0 46.5 0 64.3v383.4C0 465.5 14.3 480 31.9 480H416c17.6 0 32-
14.5 32-32.3V64.3c0-17.8-14.4-32.3-32-32.3zM135.4 416H69V202.2h66.5V416zm-
33.2-243c-21.3 0-38.5-17.3-38.5-38.5S80.9 96 102.2 96c21.2 0 38.5 17.3 38.5 38.5 0
21.3-17.2 38.5-38.5 38.5zm282.1 243h-66.4V312c0-24.8-.5-56.7-34.5-56.7-34.6 0-
39.9 27-39.9 54.9V416h-66.4V202.2h63.7v29.2h.9c8.9-16.8 30.6-34.5 62.9-34.5 67.2
0 79.7 44.3 79.7 101.9V416z" /></svg>
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 512 512"><path d="M504
256C504 119 393 8 256 8S8 119 8 256c0 123.78 90.69 226.38 209.25 245V327.69h-
63V256h63v-54.64c0-62.15 37-96.48 93.67-96.48 27.14 0 55.52 4.84 55.52
4.84v61h-31.28c-30.8 0-40.41 19.12-40.41 38.73V256h68.78l-11 71.69h-
57.78V501C413.31 482.38 504 379.78 504 256z" /></svg>
</div>
</div>
</footer>
</div>
)
}
```

6.2 App.jsx

```
import { BrowserRouter as Router, Route, Routes } from "react-router-dom";
import "./App.css";
import Nav from "./components/Nav";
import Home from "./components/Home";
import Login from "./components/Login";
import PageNotFound from "./components/PageNotFound";
import Signup from "./components/Signup";
import Chat from "./components/Chat";
import Profile from "./components/Profile";

import FreelancerDashboard from
"./components/FreelancerComponents/FreelancerDashboard";
import FreelancerServices from
"./components/FreelancerComponents/FreelancerServices";
import FreelancerCreateService from
"./components/FreelancerComponents/FreelancerCreateService";
import FreelancerManageServices from
"./components/FreelancerComponents/FreelancerManageServices";
import FreelancerUpdateService from
"./components/FreelancerComponents/FreelancerUpdateService";
import ServiceDetails from "./components/ServiceDetails";

import ClientDashboard from "./components/ClientComponents/ClientDashboard";
import ClientFreelancers from "./components/ClientComponents/ClientFreelancers";
import ClientOrders from "./components/ClientComponents/ClientOrders";

function App() {
  return (
    <div className="App">
      <Router>
        <Nav />
        <Routes>
          <Route exact path="/" element={ <Home /> } />
          <Route path="/login" element={ <Login /> } />
          <Route path="/signup" element={ <Signup /> } />
          <Route path="/dashboard/freelancer/:id">
            <Route index element={ <FreelancerDashboard /> } />
            <Route path="/dashboard/freelancer/:id/services">
              <Route index element={ <FreelancerServices /> } />
              <Route path="/dashboard/freelancer/:id/services/create"
                element={ <FreelancerCreateService /> } />
              <Route path="/dashboard/freelancer/:id/services/manage"
                element={ <FreelancerManageServices /> } />
              <Route path="/dashboard/freelancer/:id/services/update/:serviceId"
                element={ <FreelancerUpdateService /> } />
              <Route path="/dashboard/freelancer/:id/services/show/:serviceId"
                element={ <ServiceDetails type="1" /> } />
            </Route>
          </Route>
        </Routes>
      </Router>
    </div>
  );
}
```

```
</Route>
<Route path="/dashboard/freelancer/:id/chat" element={<Chat type="freelancer" />} />
</Route>
<Route path="/dashboard/freelancer/:id/profile" element={<Profile type="1" />} />
</Route>
<Route path="/dashboard/client/:id">
  <Route index element={<ClientDashboard />} />
  <Route path="/dashboard/client/:id/services" element={<ClientFreelancers />} />
  <Route path="/dashboard/client/:id/services/show/:serviceId"
    element={<ServiceDetails type="2" />} />
  <Route path="/dashboard/client/:id/orders" element={<ClientOrders />} />
  <Route path="/dashboard/client/:id/order/show/:serviceId" element={<ServiceDetails
    type="3" />} />
  <Route path="/dashboard/client/:id/chat" element={<Chat type="2" />} />
  <Route path="/dashboard/client/:id/profile" element={<Profile type="2" />} />
</Route>
<Route path="*" element={<PageNotFound />} />
</Routes>
</Router>
</div>
);
}
export default App;
```

6.3 Client Dashboard .jsx

```
import { useState, useEffect } from 'react';
import { toast } from 'react-toastify';
import { useSelector, useDispatch } from 'react-redux';
import { useParams, useNavigate } from 'react-router-dom';
import { myDashboard } from '../Redux/ClientSlice';
import orders from '../assets/svg/servicesIcon.svg'
import usd from '../assets/svg/usd.svg'
import check from '../assets/svg/check.svg'
import ClientMenu from './ClientMenu';
import TestimonialSlider from './TestimonialsSlider'
import { tokenExists } from '../Redux/UserSlice';
import Loading from '../Loading';

export default function ClientDashboard() {
  const { token } = useSelector(state => state.user)
  const { data } = useSelector(state => state.client)
  const { id } = useParams()
  const [loading, setLoading] = useState(true)
  const navigate = useNavigate()
  const dispatch = useDispatch()

  useEffect(() => {
    tokenExists(token, navigate, dispatch).then(data => (data == false ||
    JSON.parse(localStorage.getItem('userInfo')).role != "client" ||
    JSON.parse(localStorage.getItem('userInfo'))._id != id) && navigate("/login"))

    dispatch(myDashboard()).unwrap().then(data => {
      setTimeout(() => {
        setLoading(false)
        if (data.status == 404 || data.status == 403) {
          toast.error(data.msg)
          navigate('/login')
        }
        if (data.status == 505) {
          toast.error(data.msg)
        }
      }, 1000);
    }).catch((rejectedValueOrSerializedError) => {
      setTimeout(() => {
        setLoading(false)
        toast.error(rejectedValueOrSerializedError)
      }, 1000);
    })
  }, [])
  return (
    <>
    {loading && <Loading />}
  )
}
```

```
<div className="ClientDashboard">
  <div className="container">
    <div className="section">
      {data?.dashboard ?
    </div>

    <div className="header">
      Welcome Back {data?.dashboard.username}
    </div>
    <div className="stats">
      <div className="card">
        <div className="info">
          <div className="title">
            Total Expenses
          </div>
          <div className="body-stat">
            {parseFloat(data?.dashboard.expenses)} $
          </div>
        </div>
        <div className="logo">
          <img src={usd} alt="Star icon" />
        </div>
      </div>
      <div className="card">
        <div className="info">
          <div className="title">
            Total Orders
          </div>
          <div className="body-stat">
            {data?.dashboard.orders}
          </div>
        </div>
        <div className="logo">
          <img src={orders} alt="Star icon" />
        </div>
      </div>
      <div className="card">
        <div className="info">
          <div className="title">
            Completed Orders
          </div>
          <div className="body-stat">
            {data?.dashboard.completedOrders}
          </div>
        </div>
        <div className="logo">
          <img src={check} alt="Star icon" />
        </div>
      </div>
    </div>
  </div>
</div>
```



```
<div className="testimonials">
  <div className="header">
    Last Reviews Made By Me
  </div>
  <div className="cards">
    {data?.dashboard?.testimonials.length !== 0 ?
    <TestimonialSlider role="client" data={data?.dashboard?.testimonials} />
    :
    <div className='noTestimonials'>You have no reviews for now</div>
    }
  </div>
</div>
</>
:
<div className="serverStopped">
  Check the server
</div>
}
</div>
<ClientMenu active="home" />
</div>
</div>
</>
)
}
```

CHAPTER 7

TESTING

TESTING

7.1 Introduction

This chapter outlines the testing phase of "Talent Sphere" using black-box and white-box testing methods for ensuring the functionality, reliability, and performance of the system.

Black-Box Testing

This testing method focuses on ensuring the functionality of the "Talent Sphere" platform features—such as login, service creation, and chat—without examining the internal code structure. It emphasizes the input-output behavior to verify that the system meets user requirements. The goal is to confirm that the platform works as expected from an external perspective, detecting bugs and ensuring system behavior across various scenarios to provide an uninterrupted experience for clients and freelancers.

White-Box Testing

This testing method ensures the internal code paths and logic of "Talent Sphere" are functioning correctly. It involves a detailed examination of the system's internal structure, focusing on components such as API's (e.g., API endpoints) and database queries. The objective is to verify that these elements perform as expected, addressing both functional correctness and non-functional requirements like scalability and security, while identifying any issues within the code that could affect reliability or performance.

Database queries Ensuring the following act as expected:

This two-pronged method detects bugs, ensures the system behavior for various scenarios, and avoids interrupted experience for clients and freelancers, ensuring the non-functional requirements like scalability and security.

7.2 Sample Test Case Specifications

7.2.1 User Registration

| Serial No. | Condition To Be Tested | Test Data | Expected Output | Remarks |
|------------|--|---|--|------------|
| 1 | If user does not select role (Freelancer/Client) | User Role | Kindly select a user role | SUCCESSFUL |
| 2 | If username contains special characters | Username (e.g., "Prem#123") | Username should contain only alphanumeric characters | SUCCESSFUL |
| 3 | If email ID is not in proper format | Email ID (e.g., "invalid.email") | Entered email ID is not valid | SUCCESSFUL |
| 4 | If password length is less than 8 characters | Password (e.g., "pass") | Password must contain at least 8 characters | SUCCESSFUL |
| 5 | If password and confirm password do not match | Password ("Pass1234"), Confirm Password ("Pass1235") | Passwords do not match | SUCCESSFUL |
| 6 | If user tries to register with an already existing email | Email ID (e.g., "existing@example.com") | Email already registered, please use another email | SUCCESSFUL |
| 7 | If phone number is not entered | Phone Number | phone number should not be empty | SUCCESSFUL |

Table 7.2.1 Test Unit 1: User Registration

7.2.2 Request Service

| Serial No. | Condition To Be Tested | Test Data | Expected Output | Remarks |
|------------|---|--|--|------------|
| 1 | If Client submits a request with no details | Request Form (empty fields) | Please fill in all required fields | SUCCESSFUL |
| 2 | If Client requests a service that is unavailable | Service ID (non-existent) | Service not found or unavailable | SUCCESSFUL |
| 3 | If Client exceeds maximum request limit per day | Multiple Requests (e.g., 6 requests in 24 hours) | Maximum request limit reached, try again tomorrow | SUCCESSFUL |
| 4 | If Freelancer is notified but offline | Service Request Submission | Notification queued and sent via SMS when Freelancer is online | SUCCESSFUL |
| 5 | If Client tries to request the same service twice | Duplicate Service Request | Request already submitted, check status | SUCCESSFUL |

Table 7.2.2 Test Unit 2: Request Service

7.3 Screenshots of Test Cases

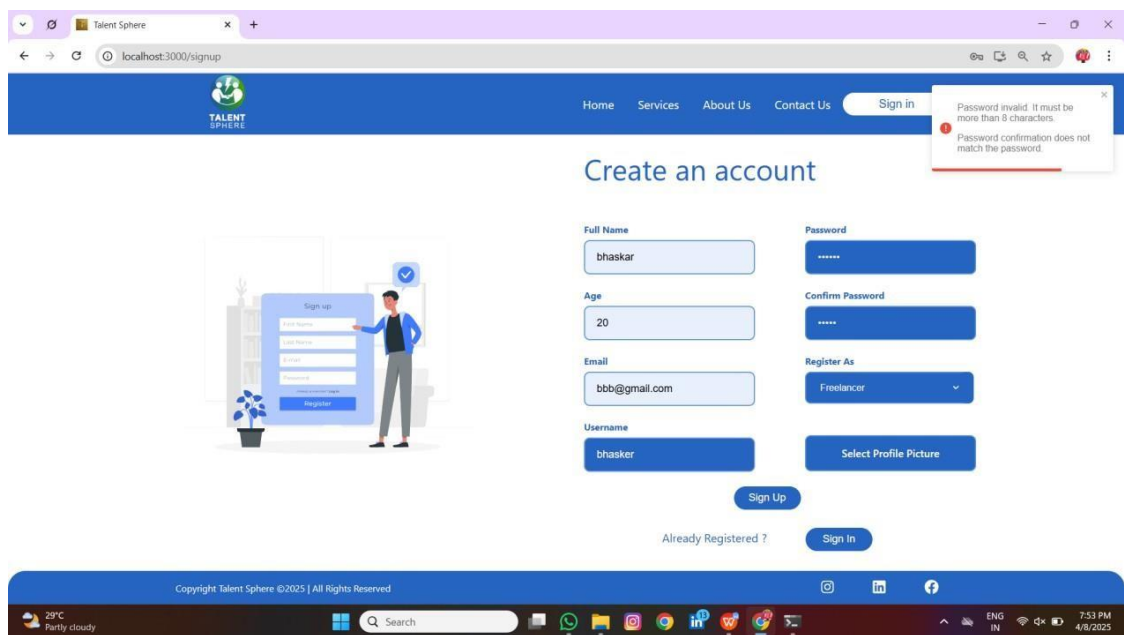


Fig 7.3.1 Screenshot for Invalid Register Credentials

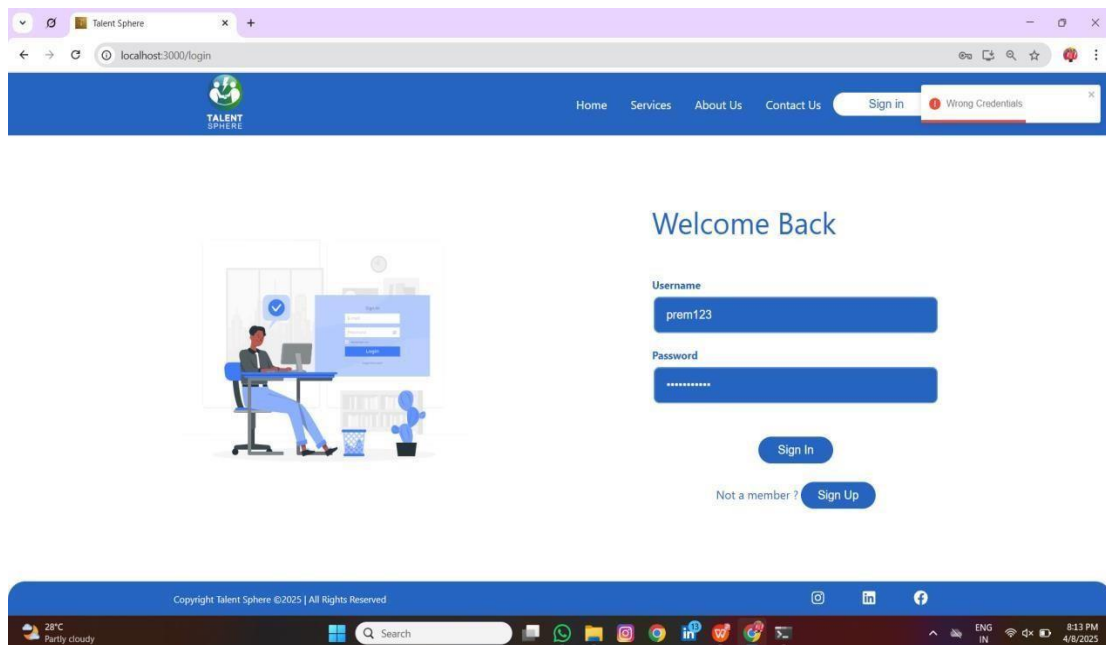


Fig 7.3.2 Screenshot for Invalid Login Credentials

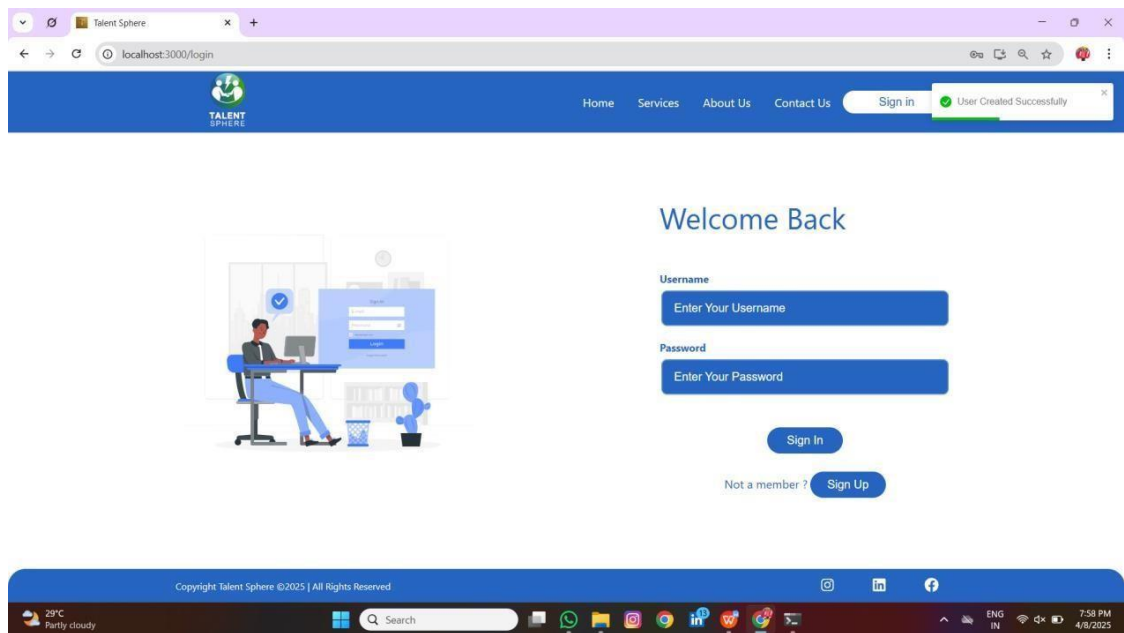


Fig 7.3.3 Screenshot for valid Register Credentials

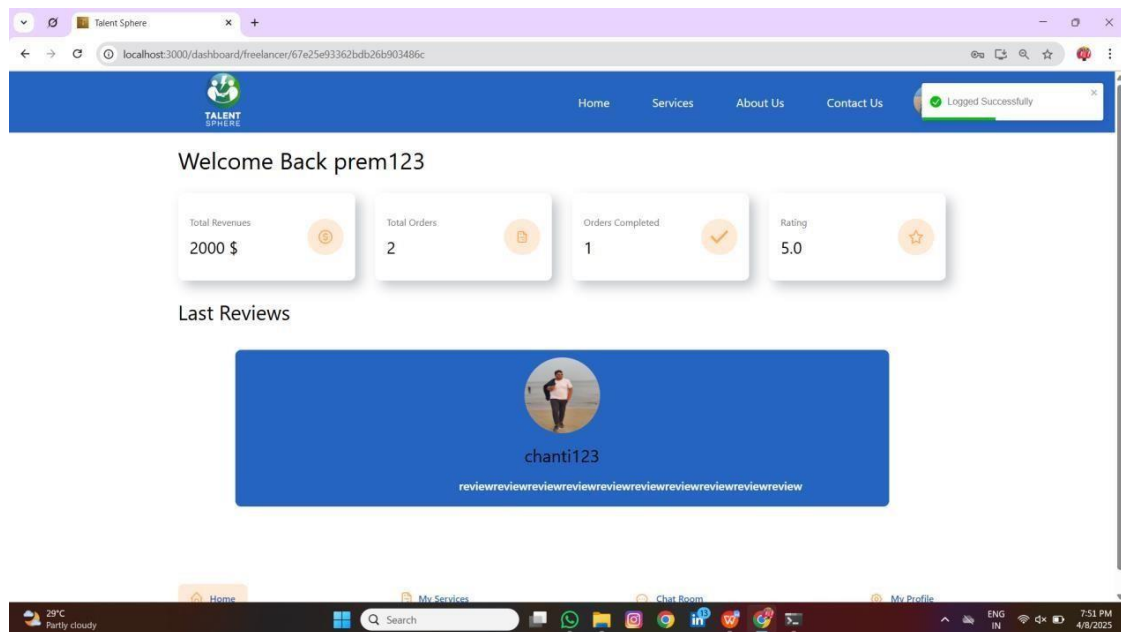


Fig 7.3.4 Screenshot for valid Login Credentials

CHAPTER 8

SCREENSHOTS

SCREENSHOTS

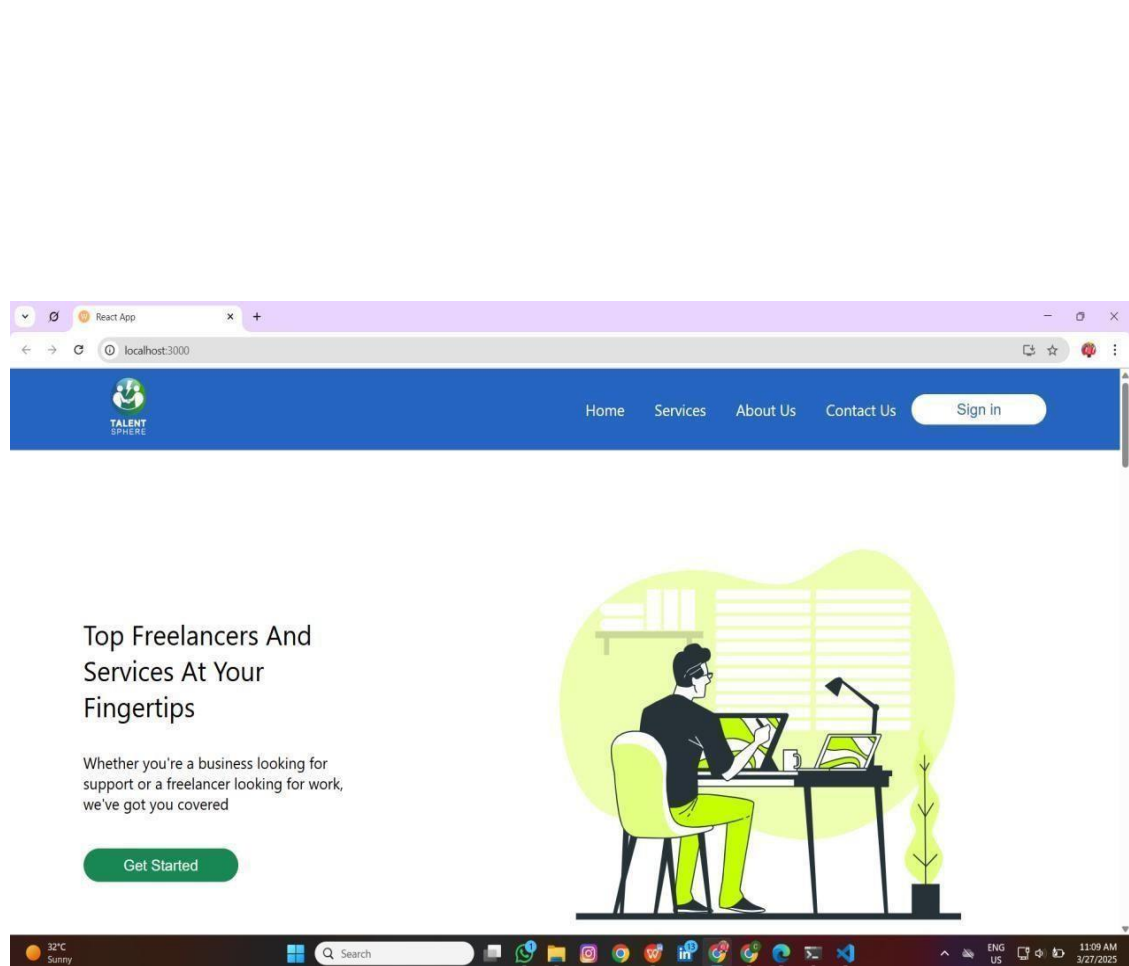


Fig 8.1 Screenshot for Home Page

Description:

The above Screenshot displays the Home Page.

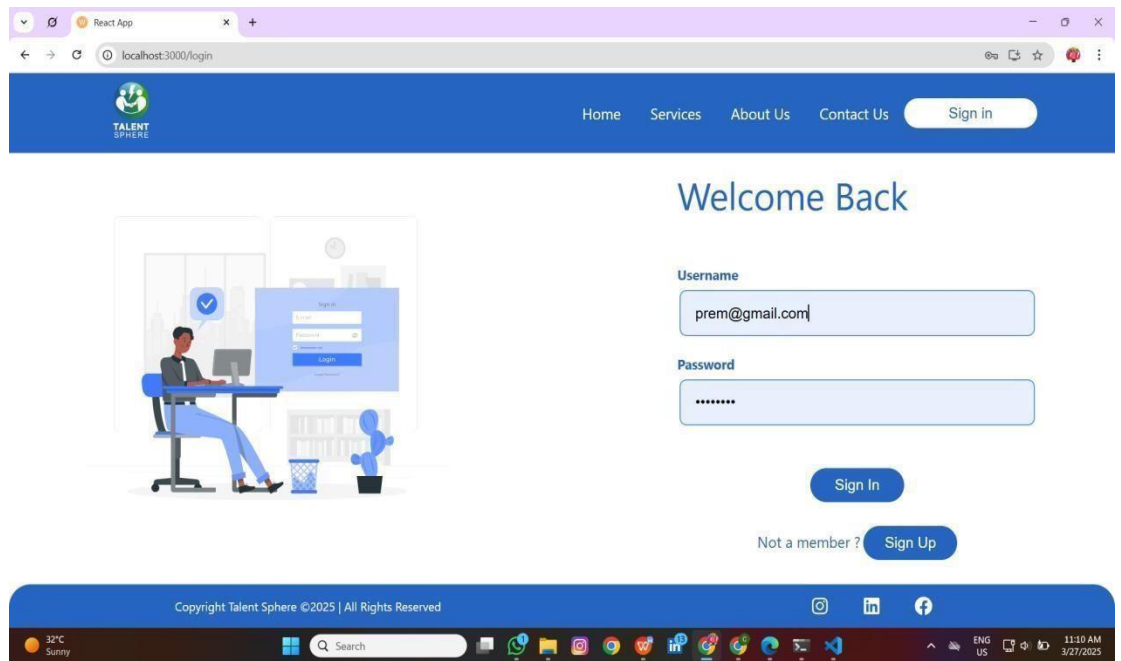


Fig 8.2 Screenshot for Signin Page

Description:

The above Screenshot displays the Signin Page

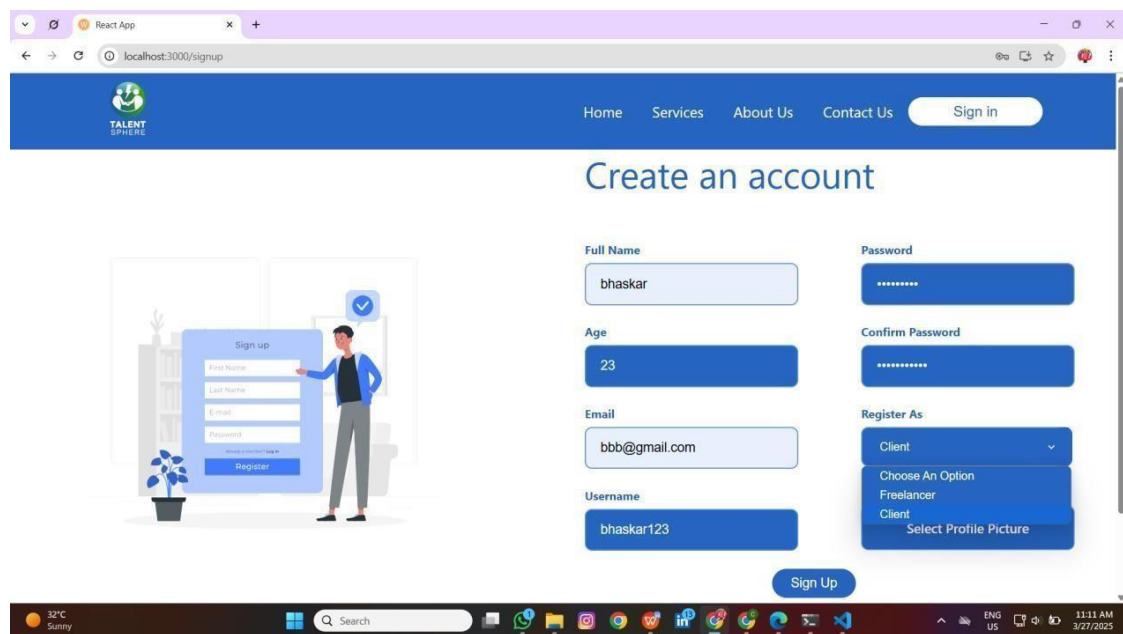


Fig 8.3 Screenshot for Signup Page

Description:

The above Screenshot displays the Signup Page

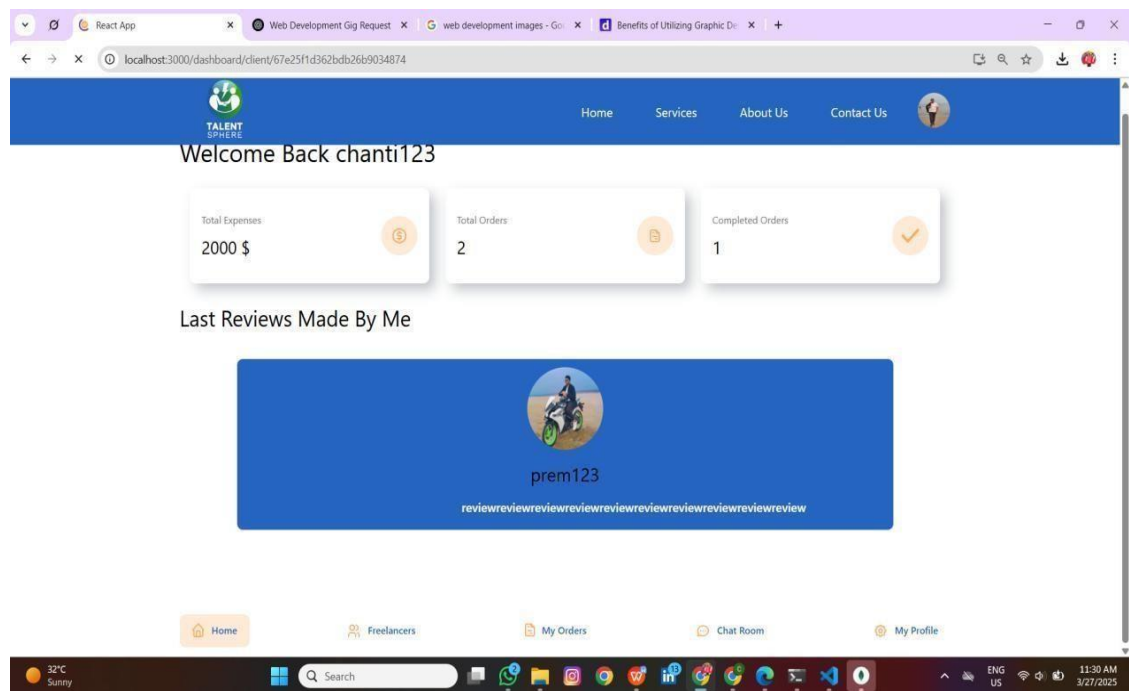


Fig 8.4 Screenshot for Client Dashboard

Description:

The above Screenshot displays the Client Dashboard

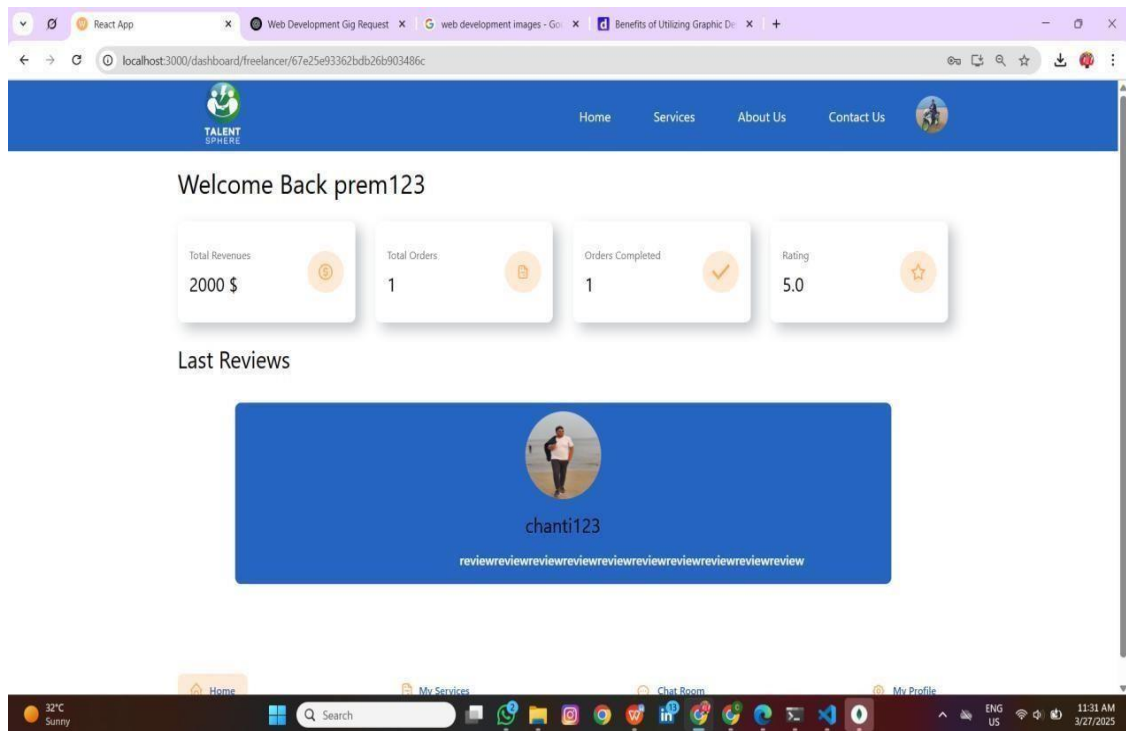


Fig 8.5 Screenshot for Freelancer Dashboard

Description:

The above Screenshot displays the Freelancer Dashboard

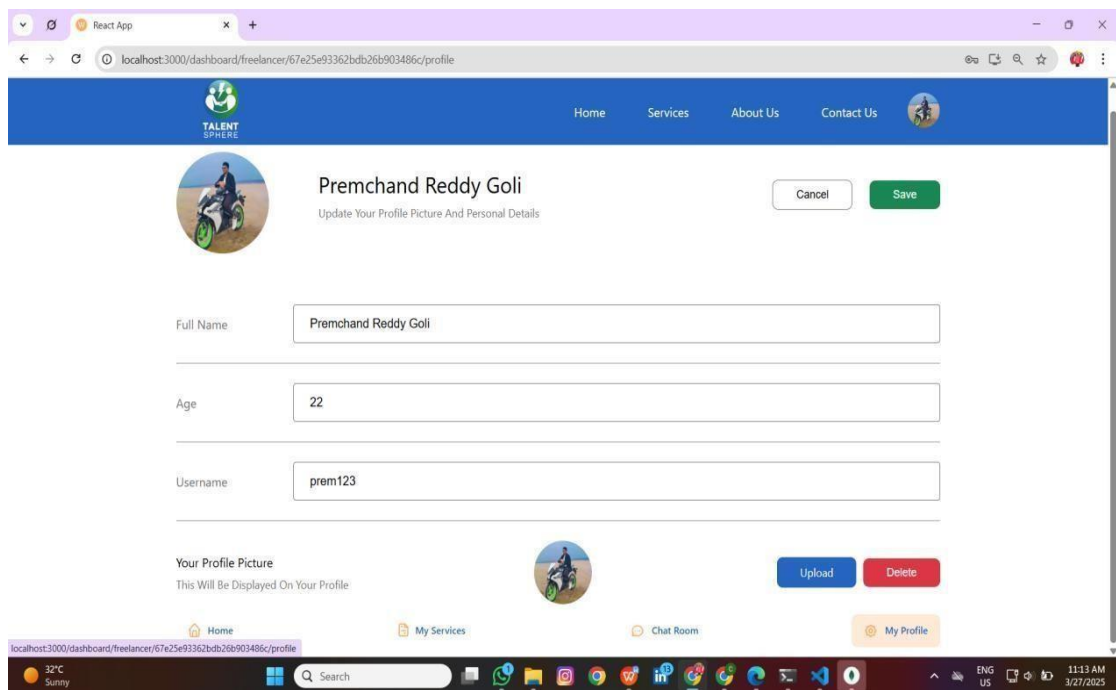


Fig 8.6 Screenshot for Manage Profile

Description:

The above Screenshot displays the Profile Management

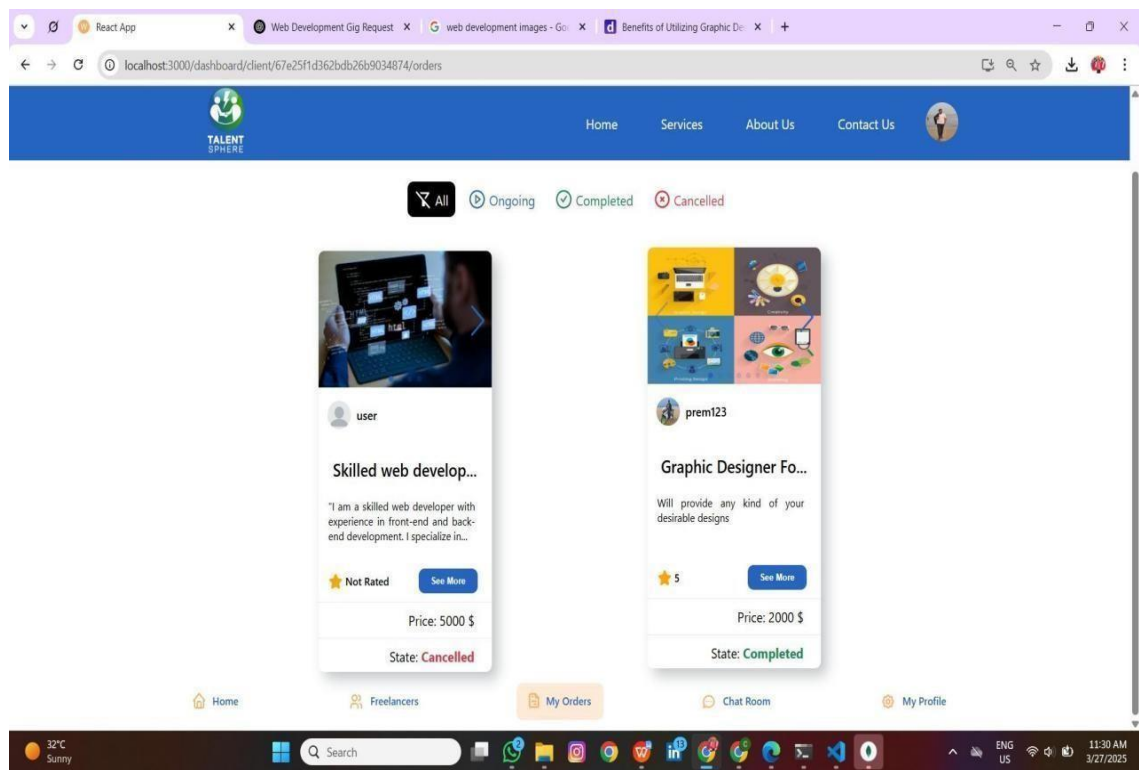


Fig 8.7 Screenshot for Orders Page

Description:

The above Screenshot displays the orders page

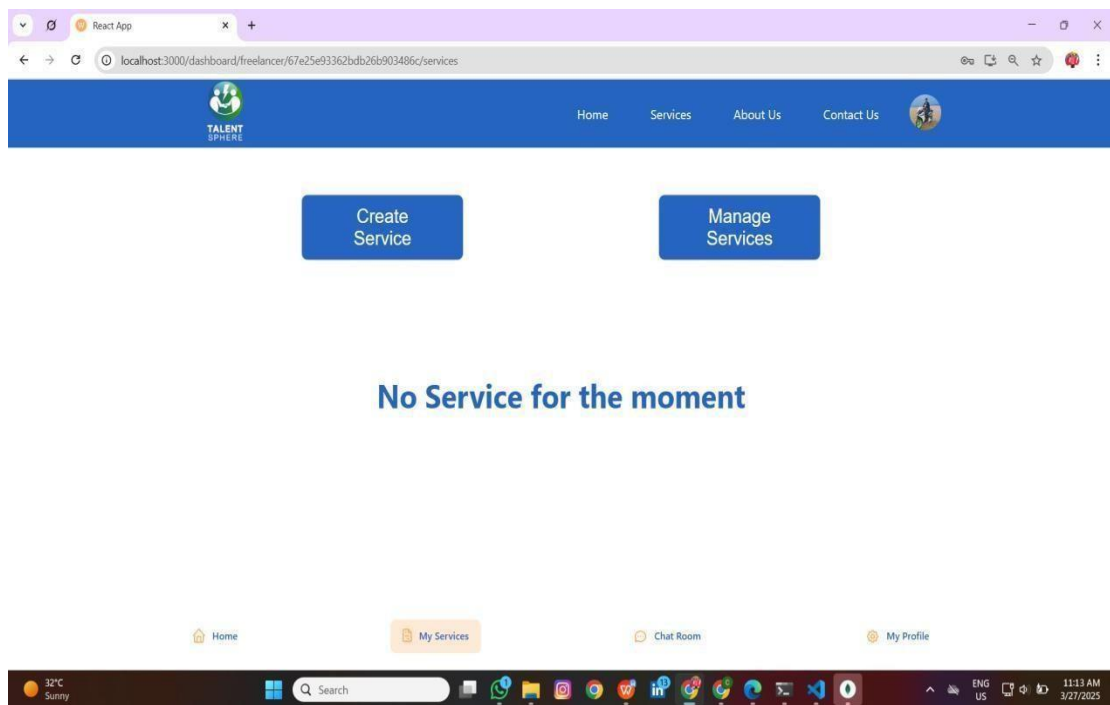


Fig 8.8 Screenshot for Freelancer Services

Description:

The above Screenshot displays the Freelancer Services

CONCLUSION

CONCLUSION

"Talent Sphere" has been a busy and good working platform that has united freelancers and clients, and provided a convenient way to work and find professional interaction. With scalable architecture, it supports increasing numbers of users with no loss of performance and thus enables freelancers looking for projects and clients looking for skills to depend upon its utilization. Simplicity in the design makes it simple to use, enabling users of different technical abilities to navigate, post a project, send proposals, and run workflows seamlessly.

In the coming years, "Talent Sphere" will continue to grow with new features in the pipeline that will further make it more efficient and useful. Including an in-built, secure payment system will enable easy closure of transactions, with users having the ability to easily handle compensation on the platform, minimizing third-party tool utilization, and fostering trust and ease. Including enhanced analytics will provide users with real-time, actionable information—freelancers can track performance, view trending skills, and customize offerings, while clients can view project trends, track freelancer performance, and make informed hiring decisions. All these prospective features, integrated together, will make "Talent Sphere" an even more comprehensive, single-stop-shop tool, and it will become the industry leader within the freelance economy.

BIBLIOGRAPHY/REFERENCES

BIBLIOGRAPHY

A bibliography section acknowledges the sources of information, frameworks, libraries, and references used during the development of the **Talent Sphere** project. Below is a structured bibliography that can be included in your documentation:

TEXT BOOKS:

MERN Stack Front To Back: Full Stack React, Redux & Node.js

Author: Brad Traversy

Publisher: Independently Published

Description: This book provides a comprehensive guide to building full-stack web applications using MongoDB, ExpressJS, ReactJS, and NodeJS.

MongoDB: The Definitive Guide

Author: Shannon Bradshaw, Eoin Brazil, Kristina Chodorow

Publisher: O'Reilly Media

Description: This book covers the fundamentals of MongoDB, including schema design, querying, and data modeling, which are essential for developing applications using NoSQL databases.

Full Stack Development with MongoDB, Express, React, and Node: Build Cloud

Ready Web Applications

Author: Shama Hoque

Publisher: Packt Publishing

Description: A practical approach to building scalable web applications with the MERN stack, covering both backend and frontend development

REFERENCES

- Brown, L., & Zhao, Y. (2021). Modern web development stacks and their impact on SaaS platforms. *Journal of Web Engineering*
- Chen, L., Wang, H., & Zhou, Y. (2021). Ensuring system stability and data protection in cloud-based applications. *International Journal of Cloud Computing*
- Kumar, R., & Lee, J. (2021). Automation for administrative efficiency in web platforms. *Proceedings of the International Conference on Web Applications*

- Miller, T., & Brown, A. (2020). Security essentials for online service platforms: A review. *Cybersecurity and Data Privacy Journal*
- Nguyen, T., & Thomas, S. (2019). Designing user-centered dashboards for SaaS applications. *User Experience Studies*
- Patel, S., Sharma, R., & Mehta, V. (2022). Real-time communication in web applications: Technologies and challenges. *International Journal of Computer Applications*
- Smith, J., & Johnson, K. (2020). Trends and challenges in freelancing platforms. *Journal of Online Work and Digital Labor*
- Wilson, M. (2022). The impact of the gig economy on digital platform design. *Tech and Society Review*

KANCHARLA NAGABABU

23P31F00I1



23P31F00I1



MCA STUDENTS



Aditya University

Document Details

Submission ID

tn:oid:::1:3213022449

Submission Date

Apr 11, 2025, 8:33 PM GMT+5:30

Download Date

Apr 11, 2025, 8:57 PM GMT+5:30

File Name

23P31F00I1.pdf

File Size

1.2 MB

65 Pages**8,892 Words****52,810 Characters**





23% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.




Exclusions

2 Excluded Sources

Match Groups

-  **78** Not Cited or Quoted 15%
Matches with neither in-text citation nor quotation marks
-  **5** Missing Quotations 2%
Matches that are still very similar to source material
-  **3** Missing Citation 6%
Matches that have quotation marks, but no in-text citation
-  **0** Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

- 19%  Internet sources
- 3%  Publications
- 20%  Submitted works (Student Papers)

Integrity Flags





0 Integrity Flags for Review

No suspicious text manipulations found.




Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

-  **78** Not Cited or Quoted 15%
Matches with neither in-text citation nor quotation marks
-  **5** Missing Quotations 2%
Matches that are still very similar to source material
-  **3** Missing Citation 6%
Matches that have quotation marks, but no in-text citation
-  **0** Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

- 19%  Internet sources
- 3%  Publications
- 20%  Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

| | | |
|-----------|---|---------------|
| 1 | Student papers | |
| | Berhampur University | 8% |
| 2 | Internet | |
| | developpement-application-mobile-lausanne.ch | 5% |
| 3 | Student papers | |
| | University of Greenwich | 2% |
| 4 | Student papers | |
| | University of Northampton | 1% |
| 5 | Internet | |
| | www.coursehero.com | 1% |
| 6 | Student papers | |
| | Engineers Australia | <1% |
| 7 | Student papers | |
| | University of Hertfordshire | <1% |
| 8 | Internet | |
| | dev.to | <1% |
| 9 | Internet | |
| | github.com | <1% |
| 10 | Internet | |
| | answers.netlify.com | <1% |

| | | | |
|----|----------------|---|-----|
| 11 | Internet | www.slideshare.net | <1% |
| 12 | Student papers | University of Wales Institute, Cardiff | <1% |
| 13 | Internet | blog.logrocket.com | <1% |
| 14 | Internet | www.geeksforgeeks.org | <1% |
| 15 | Internet | studentsrepo.um.edu.my | <1% |
| 16 | Internet | gitlab.sliit.lk | <1% |
| 17 | Internet | stackoverflow.com | <1% |
| 18 | Publication | Islam, Md. Nazmul. "Search Timelines: Contextualized Search History in Support o... | <1% |
| 19 | Internet | ijirt.org | <1% |
| 20 | Internet | cygnis.co | <1% |
| 21 | Internet | open.library.ubc.ca | <1% |
| 22 | Internet | technodocbox.com | <1% |
| 23 | Student papers | Nottingham Trent University | <1% |
| 24 | Internet | morioh.com | <1% |

| | | | |
|----|----------------|--|-----|
| 25 | Student papers | Deenbandhu Chhotu Ram University of Science and Technology | <1% |
| 26 | Student papers | University of Auckland | <1% |
| 27 | Student papers | NPS International School | <1% |
| 28 | Internet | repository.up.ac.za | <1% |
| 29 | Internet | shaper-api.kadist.org | <1% |
| 30 | Internet | www.rwi-essen.de | <1% |
| 31 | Internet | www.springactionscrip.org | <1% |
| 32 | Internet | 123dok.com | <1% |
| 33 | Internet | docplayer.net | <1% |
| 34 | Internet | petra.metromode.se | <1% |
| 35 | Internet | prajeeshblogs.blogspot.com | <1% |
| 36 | Internet | slashdot.org | <1% |
| 37 | Internet | text-id.123dok.com | <1% |
| 38 | Internet | www.iscet.pt | <1% |

39

Publication

de Sousa Coelho, Tiago Fernando. "Automação de testes de aplicações móveis se...

<1%