Project Report Format

1. INTRODUCTION

1.1 Project Overview

1.2 Purpose

2. IDEATION PHASE

2.1 Problem Statement

2.2 Empathy Map Canvas

2.3 Brainstorming

3. REQUIREMENT ANALYSIS

3.1 Customer Journey map

3.2 Solution Requirement

3.3 Data Flow Diagram

3.4 Technology Stack

4. PROJECT DESIGN

4.1 Problem Solution Fit

4.2 Proposed Solution

4.3 Solution Architecture

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

7. RESULTS

7.1 Output Screenshots

8. ADVANTAGES & DISADVANTAGES

9. CONCLUSION

10. FUTURE SCOPE

11. APPENDIX

Source Code(if any)

Dataset Link

GitHub & Project Demo Link

1.INTRODUCTION

1.1 Project Overview:

SB Works is a web-based freelancing platform designed to connect clients and freelancers under administrative supervision. The primary objective of this project is to create a structured environment where clients can post projects, freelancers can apply and submit work, and administrators can monitor system activities and manage users.
The application supports three major roles: Admin, Client, and Freelancer. Each role has dedicated dashboards and functionalities. Clients are allowed to create and manage projects. Freelancers can browse available projects, apply to them, and submit completed work. Administrators have system-level visibility and can monitor users, projects, and applications.
The system also integrates real-time communication functionality using socket technology, allowing direct interaction between users.
The core features of the system include user registration and login with OTP verification, role-based dashboards, project creation and management, freelancer applications, work submission, review mechanisms, administrative monitoring, and real-time chat functionality.

1.2 Purpose:

The primary purpose of this project is to design and develop a full-stack web-based Freelance Management System that connects administrators and freelancers through a structured digital platform.The system aims to streamline project posting, application handling, and user management while ensuring secure authentication, role-based access control, and efficient data management.It also serves as a practical implementation of modern web development technologies including React, Node.js, Express.js, and MongoDB in a real-world application scenario

Key objectives of the system include:

- To build a complete MERN stack application integrating frontend, backend, and database components into a cohesive system.
- To ensure secure user login using JWT-based authentication and role-based access control for protected resources.
- To allow administrators to manage users and projects, and enable freelancers to apply, track, and manage their project activities.
- To design an intuitive, responsive UI that enhances usability across desktop and mobile devices.
- To securely store user credentials using password hashing and implement proper validation and error handling mechanisms.
- To follow a modular and scalable backend structure that supports future enhancements and potential real-world deployment.
- To implement complete Create, Read, Update, and Delete functionality for users, projects, and applications.
- To create a system that can later incorporate advanced features such as payment integration, real-time notifications, and enhanced analytics.

2.IDEATION PHASE

2.1 Problem Statement:

Managing freelance projects and user interactions manually or through unstructured platforms often leads to inefficiencies, miscommunication, and lack of transparency. Many existing systems do not provide secure role-based access, streamlined project management, or centralized data handling for administrators and freelancers. This creates challenges in tracking project progress, handling applications, and ensuring data security. Additionally, the absence of structured authentication mechanisms increases the risk of unauthorized access. Therefore, there is a need for a secure, scalable, and user-friendly web-based Freelance Management System that centralizes project operations, improves communication, ensures data integrity, and provides efficient role-based access control for all users.

Problem Statement -1:

I am	A business owner/ client
I'm trying to	find skilled freelancers to complete my projects efficiently and within budget
But	I cannot easily identify reliable freelancers with verified skills and past work in one trusted platform
Because	many freelancers lack structured profiles, transparent reviews, and direct communication channels
Which makes me feel	uncertain about hiring decisions and worried about project quality

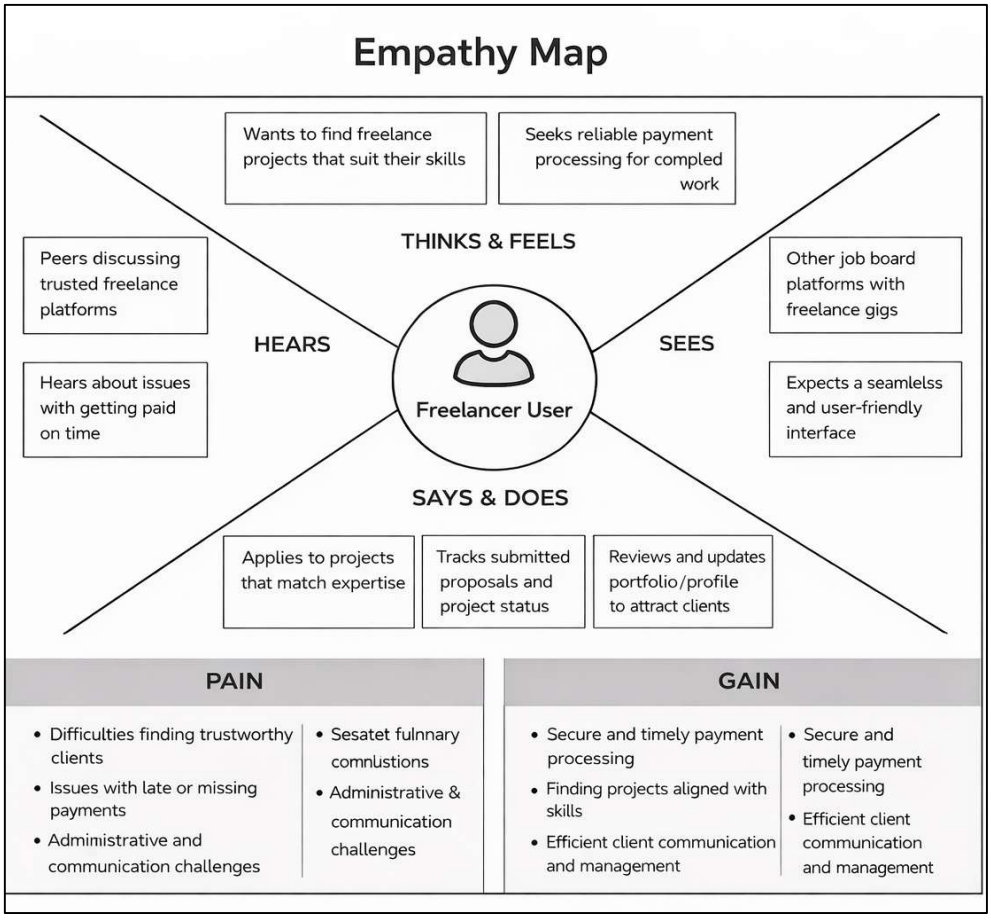
Problem Statement -2:

I am	a freelancer
I'm trying to	find suitable freelance projects and showcase my skills to potential clients
But	I cannot easily discover relevant projects or connect with genuine clients in a secure platform
Because	many platforms are overcrowded and do not provide fair visibility or smooth proposal and communication systems
Which makes me feel	overlooked, frustrated, and limited in growing my freelance career

Problem Statement (PS)	I am	I'm trying to	But	Because	Which makes me feel
PS-1	A client	Hire skilled freelancers for my projects within budget and timeline	It is hard to identify reliable freelancers and compare their skills and experience	Freelancer profiles, reviews, and communication are scattered or unclear across platforms	Uncertain and worried about project success
PS-2	A freelancer	Find suitable projects and showcase my skills to genuine clients	It is difficult to discover relevant projects and get selected fairly	Platforms are overcrowded and lack transparent bidding and direct communication systems	Overlooked and frustrated

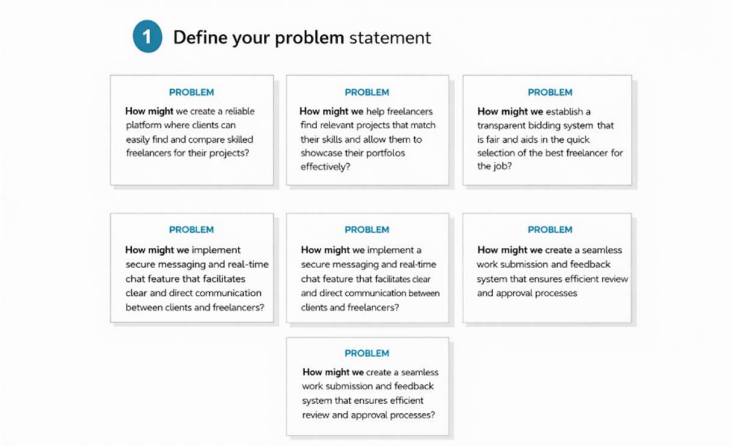
2.2 Empathy Map:

User: - Charan (A working professional as a freelancer.)

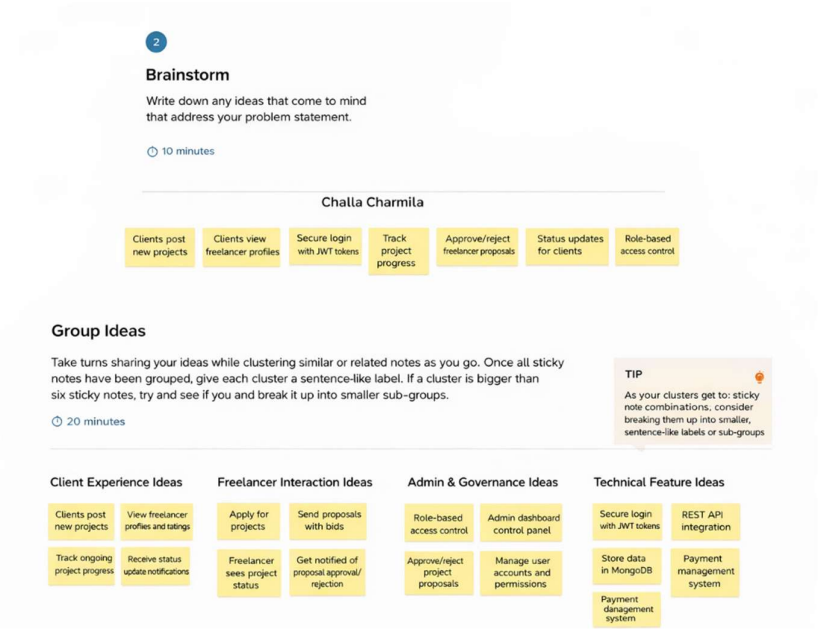


2.2 Brainstorm & Idea Prioritization:-

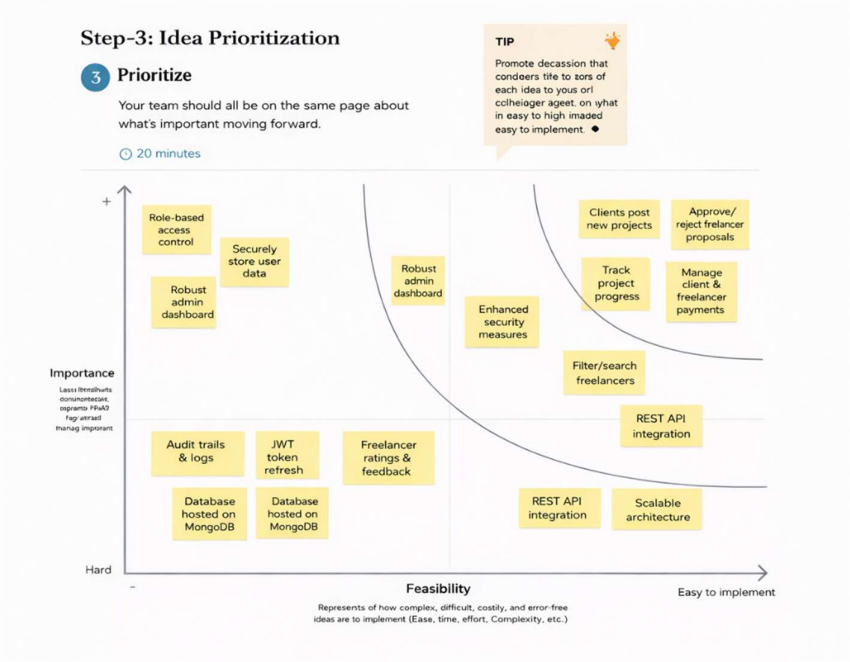
Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping



Step-3: Idea Prioritization



3. REQUIREMENT ANALYSIS

3.1 Journey map: -

3.2 Solution Requirement

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through form Registration via email Role selection (Client/ Freelancer)
FR-2	User Authentication	Login using Email & Password Logout Password reset
FR-3	Freelancer Profile Management	Create freelancer profile Add skills & bio Upload portfolio
FR-4	Project Posting	Client creates project Add description, budget, deadline Edit/ delete project

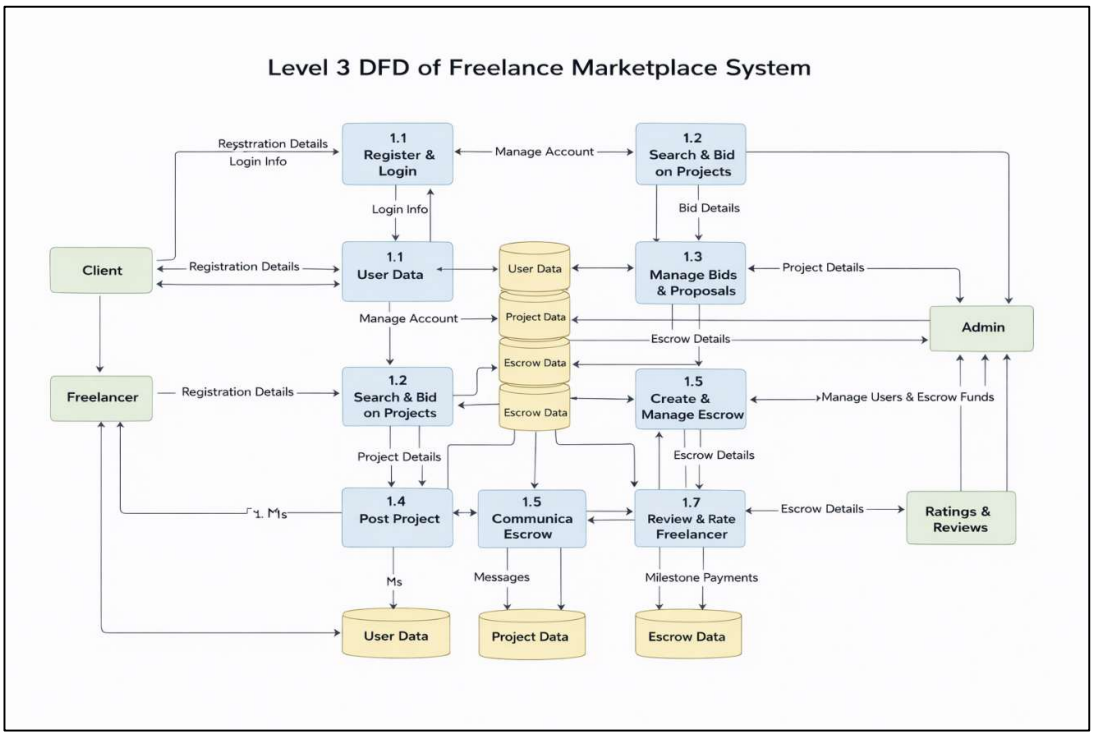
FR-5	Product Browsing & Search	View all projects Filter by skills/ budget View project details
FR-6	Bidding/ Applications	Freelancer applies to project Submit proposal & bid View application status
FR-7	Freelancer Selection	Client reviews applications Accept/ reject freelancer
FR-8	Chat & Communication	Real-time chat client and freelancer interaction Message notifications
FR-9	Project Submission	Freelancer submits work Client reviews submission
FR-10	Reviews & Ratings	Client rates freelancer View feedback
FR- 11	Admin Management	View users & projects Monitor platform activity

Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The platform should provide an intuitive and user-friendly interface for clients and freelancers with easy navigation and clear workflows.
NFR-2	**Security**	User data, passwords, and communications must be securely stored and transmitted using encryption and authentication mechanisms.
NFR-3	**Reliability**	The system should ensure consistent operation with minimal failures and accurate handling of projects, applications, and messages.
NFR-4	**Performance**	The platform should load pages and respond to user actions quickly, even with multiple concurrent users.
NFR-5	**Availability**	The system should be accessible 24/7 with minimal downtime for users across different locations
NFR-6	**Scalability**	The architecture should support growth in users, projects, and data without performance degradation.

3.3 Data Flow Diagram: -



3.4 Technology Stack: -

Technical Architecture: -

The Freelance finding application follows a 3-Tier Client–Server Architecture:

- **Presentation Layer (Frontend):** The frontend is developed using React.js to provide a responsive and user-friendly interface for Clients and Freelancers. It handles user interactions, dashboards, forms, and communicates securely with backend APIs.
- **Application Layer (Backend):** The backend is built with Node.js and Express.js to manage business logic, authentication, and project workflows. It processes API requests, handles role-based access control, and ensures secure data transactions.
- **Data Layer (Database):** MongoDB is used to store user profiles, projects, bids, payments, and reviews in structured collections. It ensures data integrity, scalability, and secure storage with proper indexing and validation.

The frontend uses React to enable dynamic rendering, improved performance, seamless navigation, state management, user interactions, secure communication with backend APIs.

Table-1: Components & Technologies:

S. No	Component	Description	Technology
1.	User Interface	Web interface for clients and freelancers to register, browse projects, apply, chat, and manage work	HTML, CSS, JavaScript, React.js, Bootstrap

2.	Application Logic-1	User authentication, registration, role management, profile handling	Node.js, Express.js
3.	Application Logic-2	Project management, bidding system, freelancer selection	Express.js, Node.js
4.	Application Logic-3	Real-time chat and notifications between client and freelancer	Socket.IO
5.	Database	Stores users, freelancers, projects, applications, chats, reviews	MongoDB, Mongoose
6.	Cloud Database	Database hosting in cloud	MongoDB Atlas
7.	File Storage	Storage for portfolio files, project attachments, submission files	Cloudinary / Local Storage
8.	External API-1	Email notifications for registration, project updates, and messages	Nodemailer/ SMTP
9.	External API-2	Authentication and secure password handling	JWT/ bcrypt
10.	Infrastructure (Server / Cloud)	Backend server deployment and hosting of full-stack MERN application	Node.js Server, Render / AWS / Vercel

Table-2: Application Characteristics:

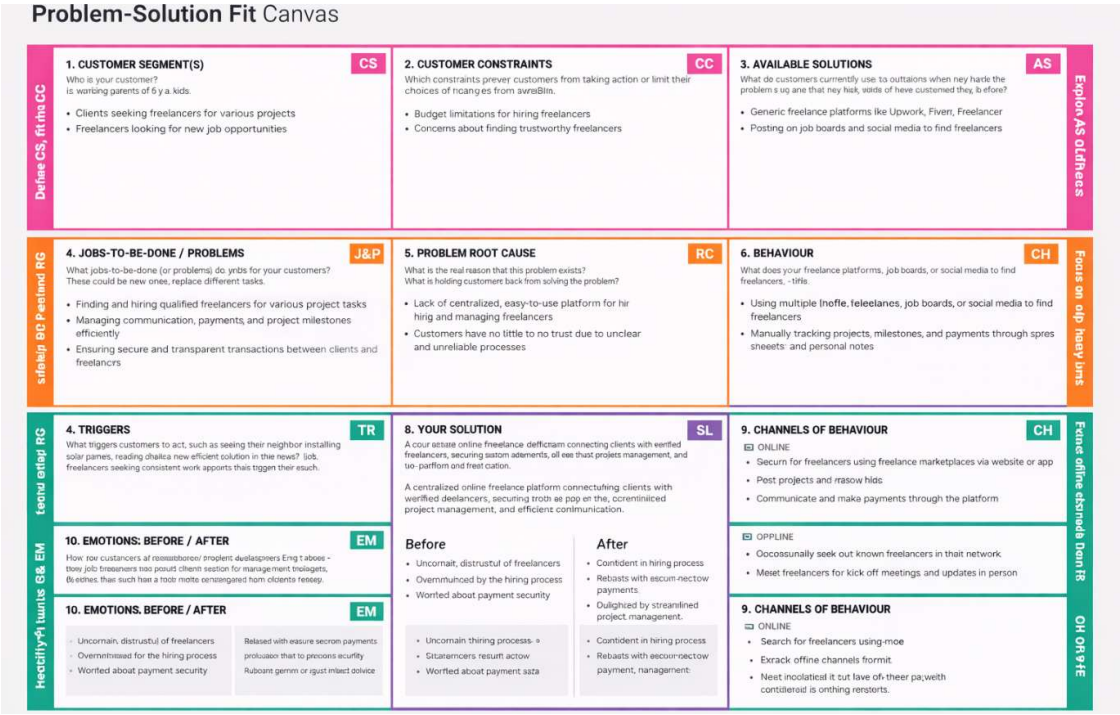
S. No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Frameworks used to develop frontend and backend	React.js, Node.js, Express.js, Bootstrap
2.	Security Implementations	Secure authentication, password encryption, protected APIs	JWT Authentication, bcrypt hashing, HTTPS
3.	Scalable Architecture	MERN 3-tier architecture separating UI, server, and database	React + Node/Express + MongoDB
4.	Availability	Cloud hosting ensures 24/7 platform access	Render / Vercel / MongoDB Atlas
5.	Performance	Efficient API communication and optimized frontend rendering	Axios, REST APIs, React Virtual DOM

4. PROJECT DESIGN

4.1 Problem Solution Fit : -

The project addresses the inefficiencies and lack of transparency in managing freelance projects through unstructured or manual systems. Clients often struggle to find suitable freelancers, while freelancers face difficulty in discovering reliable projects and secure payment mechanisms.

The proposed solution directly fits these problems by providing a centralized, role-based web platform that streamlines project posting, bidding, communication, and management. By integrating secure authentication, structured workflows, and real-time status tracking, the system ensures clarity, accountability, and efficiency for both clients and freelancers, effectively bridging the gap between demand and skilled talent.



4.2 Proposed Solution : -

S.No	Parameter	Description
1	**Problem Statement (Problem to be solved)**	Clients face difficulty in finding reliable and skilled freelancers, while freelancers struggle to discover relevant projects and showcase their capabilities in a trusted and transparent environment.

2	**Idea / Solution Description**	SB Works (FreelanceFinder) is a MERN-based online freelancing platform that connects clients and freelancers in a unified system. Clients can post projects, review freelancer profiles, and hire suitable candidates, while freelancers can browse projects, submit proposals, communicate via real-time chat, and deliver work through the platform.
3	**Novelty / Uniqueness**	The platform integrates project discovery, freelancer verification, bidding, communication, and submission in a single streamlined workflow. It emphasizes transparency through structured profiles, project tracking, and direct client-freelancer interaction within one system.
4	**Social Impact / Customer Satisfaction**	The solution empowers freelancers by providing equal access to opportunities and helps clients efficiently hire skilled professionals. It promotes digital employment, remote collaboration, and economic growth while improving hiring trust and satisfaction.
5	**Business Model (Revenue Model)**	The platform can generate revenue through service fees or commission on successful projects, premium freelancer memberships for enhanced visibility, and featured project listings for clients.
6	**Scalability of the Solution**	Built on MERN stack with cloud-hosted database and modular architecture, the platform can scale to support increasing users, projects, and real-time interactions without performance degradation.

4.3 Solution Architecture : -

The solution architecture follows a layered full-stack design to ensure scalability, security, and maintainability of the Freelance Management System. It is structured into three main layers: Presentation Layer, Application Layer, and Data Layer.

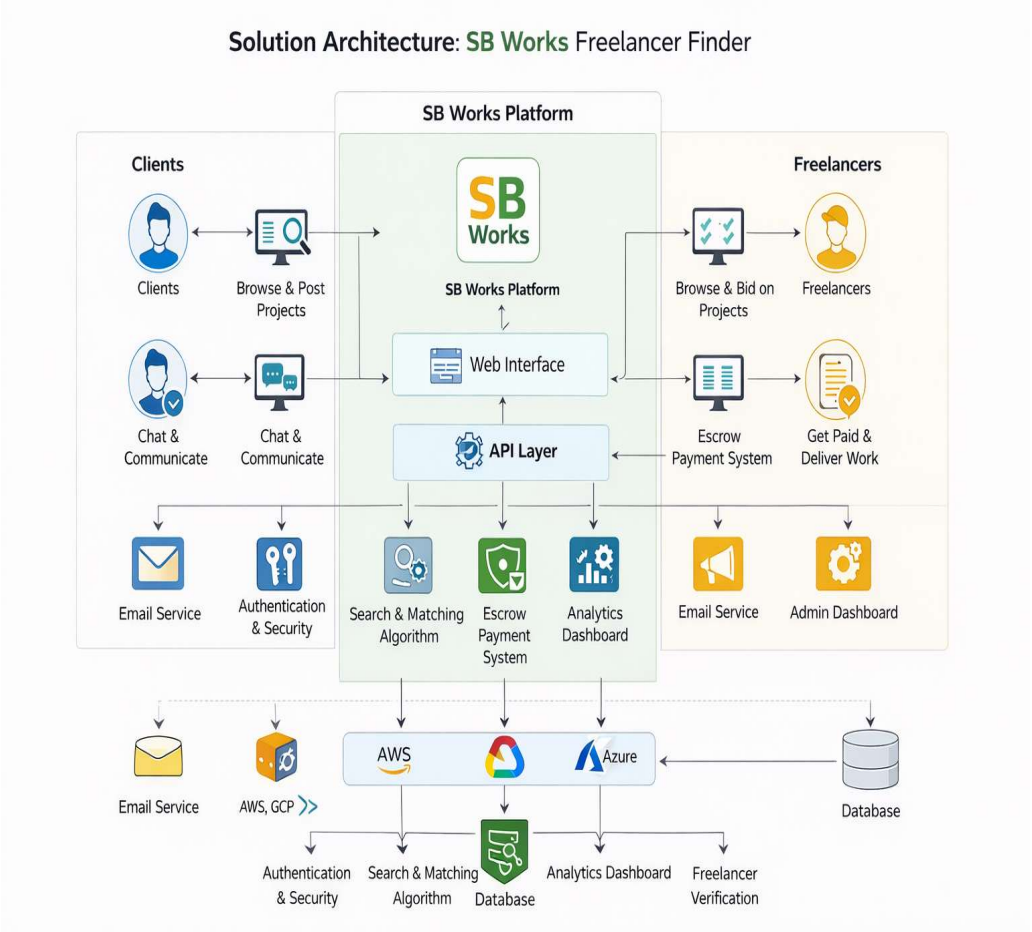
The Presentation Layer (Frontend) is developed using React and follows a Single Page Application architecture. It provides separate dashboards for Clients, Freelancers, and Admin, enabling dynamic content rendering and smooth navigation. This layer handles user interactions, form validations, state management, and secure communication with backend APIs.

The Application Layer (Backend) is built using Node.js and Express.js. It manages business logic, authentication using JWT, role-based authorization, project workflows, and RESTful API endpoints. Middleware ensures request validation, error handling, and secure access control.

The Data Layer (Database) uses MongoDB to store users, projects, bids, payments, and reviews. Proper indexing, schema validation, and secure storage mechanisms ensure data integrity, consistency, and efficient retrieval across the system.

Secure JWT-based authentication ensures protected access, while modular architecture enables scalability, maintainability, and future enhancements.

.



5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning : -

Product Backlog & Sprint Planning (Freelance Finder)

Sprint	**Functional Requirement (Epic)**	**User Story No**	**User Story / Task**	**Story Points**	**Priority**
Sprint-1	Registration	USN-1	As a user, I can register using email and password	2	High
Sprint-1	Registration	USN-2	As a user, I can select role (client/freelancer) during registration	2	High
Sprint-1	Login	USN-3	As a user, I can log in using email & password	1	High
Sprint-1	Profile	USN-4	As a freelancer, I can create and update my profile	3	High
Sprint-1 Total				8	
Sprint-2	**Functional Requirement**	**User Story No**	**User Story**	**Story Points**	**Priority**
Sprint-2	Project Posting	USN-5	As a client, I can post a new project with details	3	High
Sprint-2	Project Browsing	USN-6	As a freelancer, I can browse available projects	2	High
Sprint-2	Bidding	USN-7	As a freelancer, I can apply to a project with proposal	3	High
Sprint-2 Total				**8**	
Sprint-3	**Functional Requirement**	**User Story No**	**User Story**	**Story Points**	**Priority**
Sprint-3	Freelancer Selection	USN-8	As a client, I can review applications and select freelancer	3	High

Sprint-3	Chat	USN-9	As a client/freelancer, I can send messages in chat	5	High
Sprint-3 Total				**8**	
Sprint-4	**Functional Requirement**	**User Story No**	**User Story**	**Story Points**	**Priority**
Sprint-4	Submission	USN-10	As a freelancer, I can submit completed work	2	Medium
Sprint-4	Review	USN-11	As a client, I can review and approve submission	2	Medium
Sprint-4	Admin	USN-12	As an admin, I can view users and project	4	Medium
Sprint-4 Total				**9**	

Sprint Tracker (Velocity Table)

Sprint	**Total Story Points**	**Duration**	**Start**	**End (Planned)**	**Completed**	**Actual**
Sprint-1	8	6 days	Day 1	Day 6	8	Day 6
Sprint-2	8	6 days	Day 7	Day 12	8	Day 12
Sprint-3	8	6 days	Day 13	Day 18	8	Day 18
Sprint-4	9	6 days	Day 19	Day 24	9	Day 24

Velocity Calculation

From sprint table:

- Sprint-1 = 8 points
- Sprint-2 = 8 points
- Sprint-3 = 8 points
- Sprint-4 = 9 points

Total completed story points = 33

Number of sprints = 4

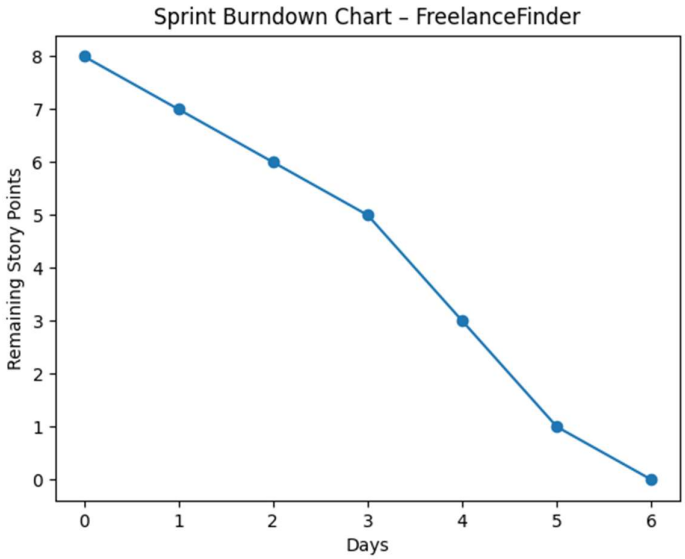
Average Velocity = 33 / 4 = 8.25 ≈ 8 points per sprint

If sprint duration = 6 days:

Velocity per day = 8 / 6 ≈ 1.33 points/day

The team completed a total of 33 story points across 4 sprints. The average team velocity is approximately 8 story points per sprint. With a sprint duration of 6 days, the average velocity is 1.33 story points per day.

Burndown Chart :



6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Functional Testing

Test Scenarios & Results

Test Case ID	Scenario (What to test)	Test Steps (How to test)	Expected Result	Actual Result	Pass/Fail
FT-01	User Registration Validation	Enter valid and invalid email/password during registration	Valid inputs accepted, errors for invalid inputs	System validates inputs correctly	Pass
FT-02	User Login Authentication	Enter correct & incorrect login credentials	Login success for valid; error for invalid	Authentication works correctly	Pass
FT-03	Project Posting	Client fills project form and submits	Project created and visible in listings	Project created successfully	Pass
FT-04	Project Browsing	Freelancer views available projects	Projects list loads correctly	Projects displayed correctly	Pass
FT-05	Bid Submission	Freelancer submits proposal on project	Freelancer submits proposal on project	Bid saved successfully	Pass
FT-06	Freelancer Selection	Client selects freelancer from applications	Freelancer assigned to project	Selection works correctly	Pass
FT-07	Chat Functionality	Send messages between client and freelancer	Messages delivered in real-time	Chat works correctly	Pass
FT-08	Project Submission	Freelancer uploads submission	Submission stored and visible to client	Submission successful	Pass
FT-09	Review & Rating	Client rates freelancer after completion	Client rates freelancer after completion	Rating stored correctly	Pass

Performance Testing

Test Case ID	Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
PT-01	Page Load Response Time	Open project list/dashboard	Page loads under 3 seconds	Loads within limit	Pass
PT-02	Concurrent Users	Multiple users browse/apply simultaneously	System handles without crash	Stable performance observed	Pass
PT-03	Chat Performance	Send multiple rapid messages	Messages delivered instantly	No delay observed	Pass

7. RESULTS

7.1 Output Screenshots

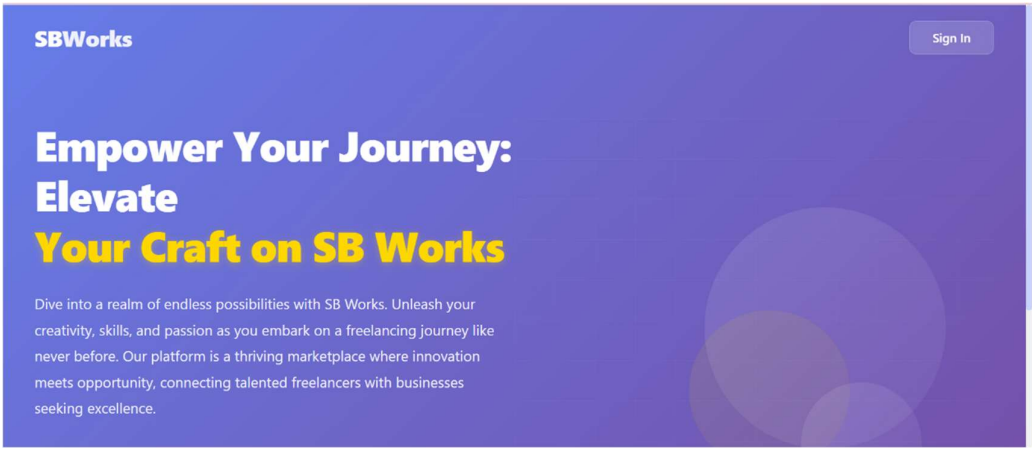


Fig : Public Home Page

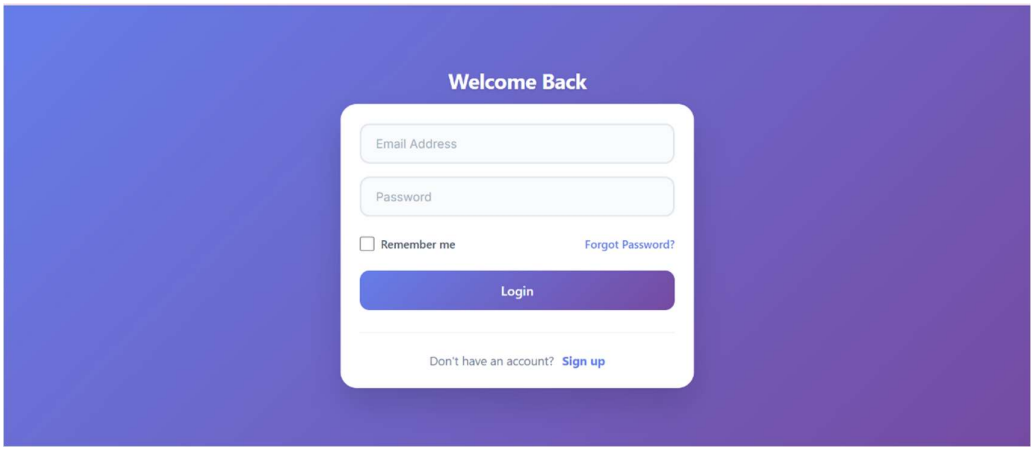


Fig : Login page

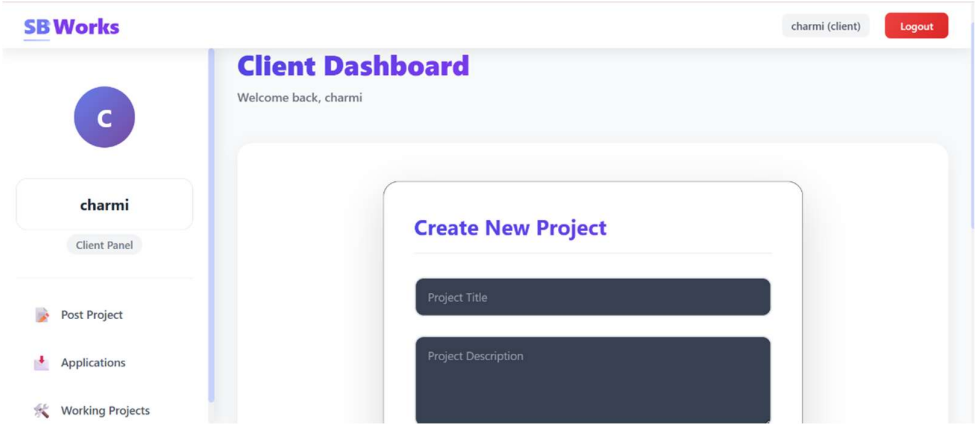


Fig : Client Dashboard

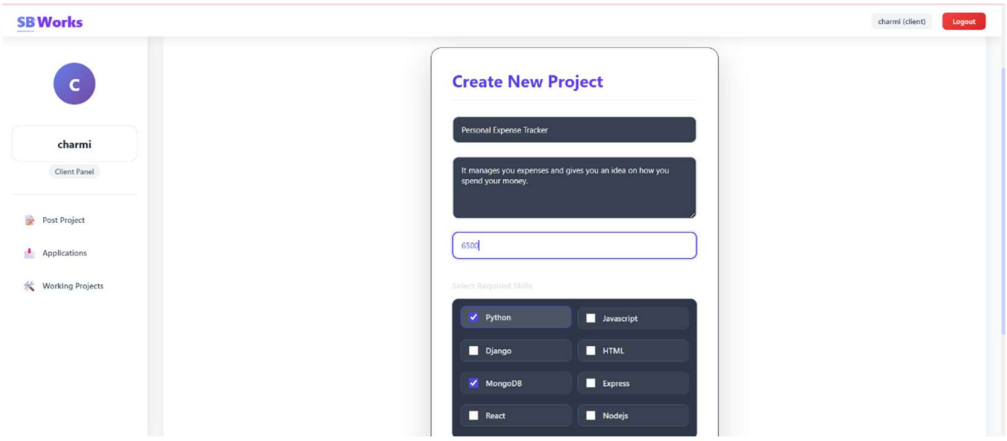


Fig : Uploading of the project in the client

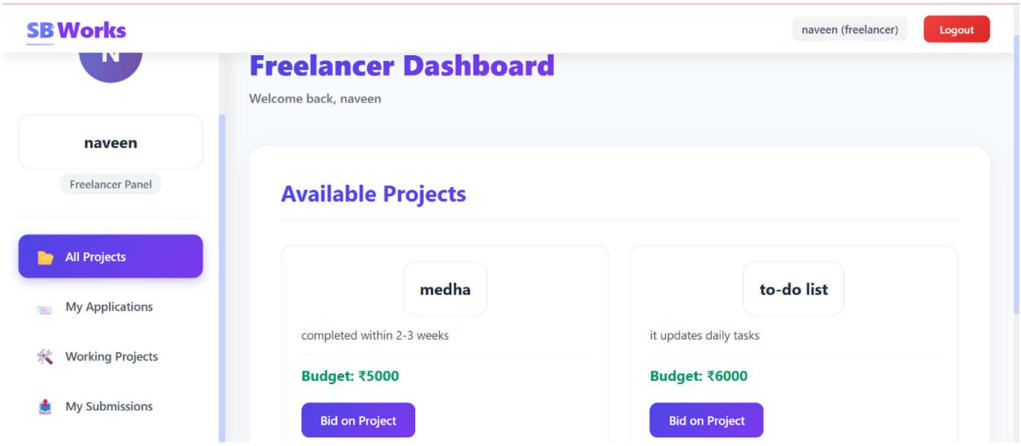


Fig : Freelancers shown all the porjects from clients

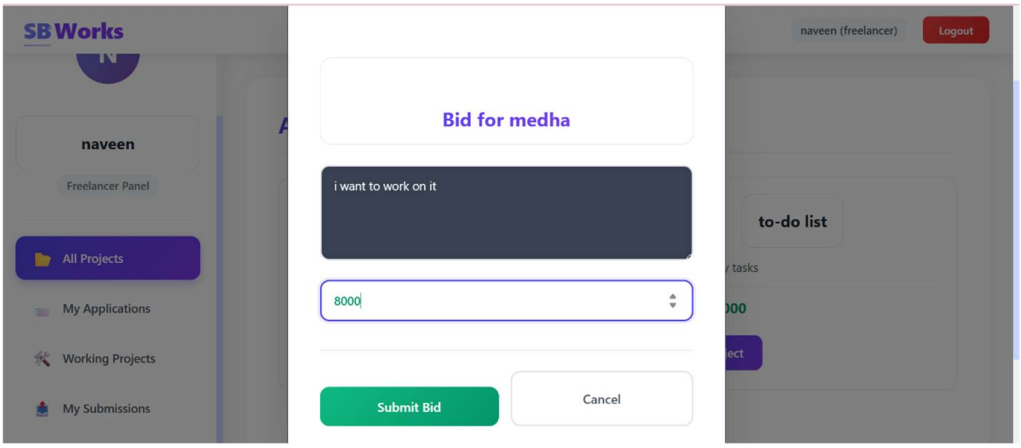


Fig : Freelancer bidding on the project

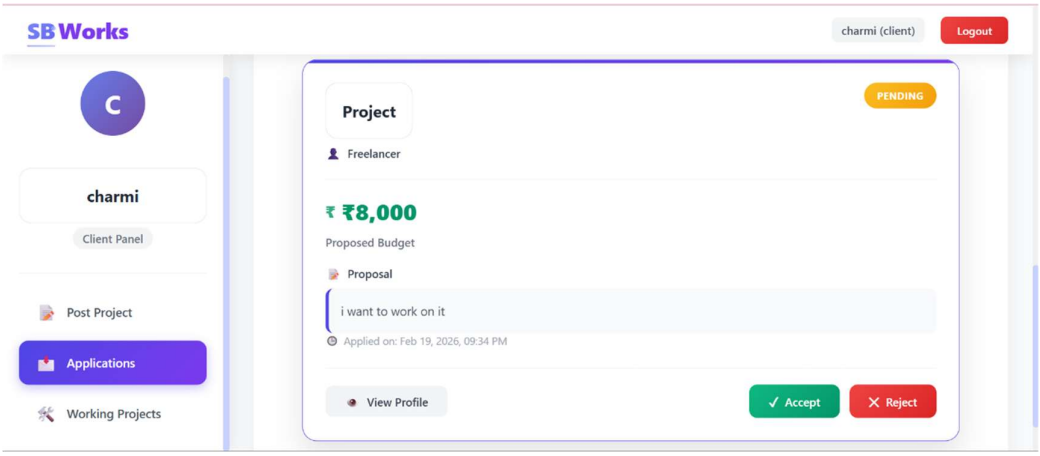


Fig : Client getting the bid from the freelance

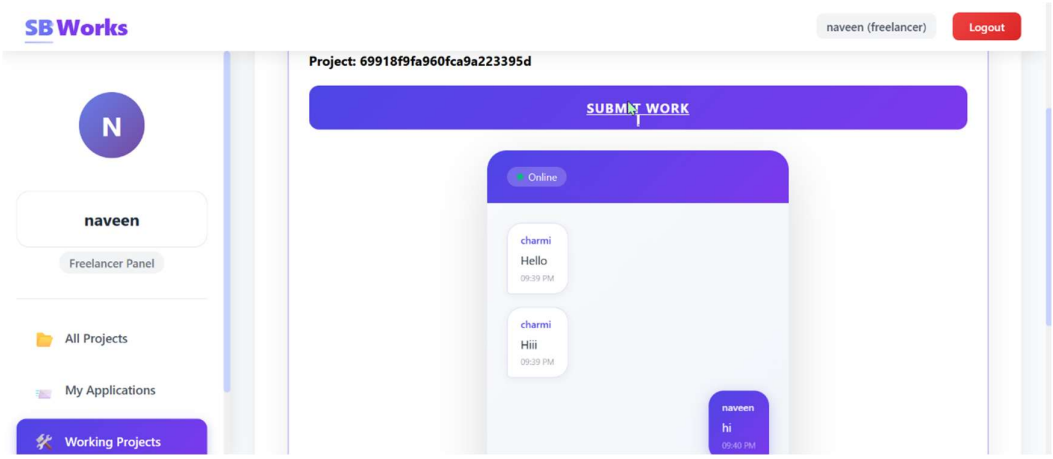


Fig : Conversation build up between the client and freelancer

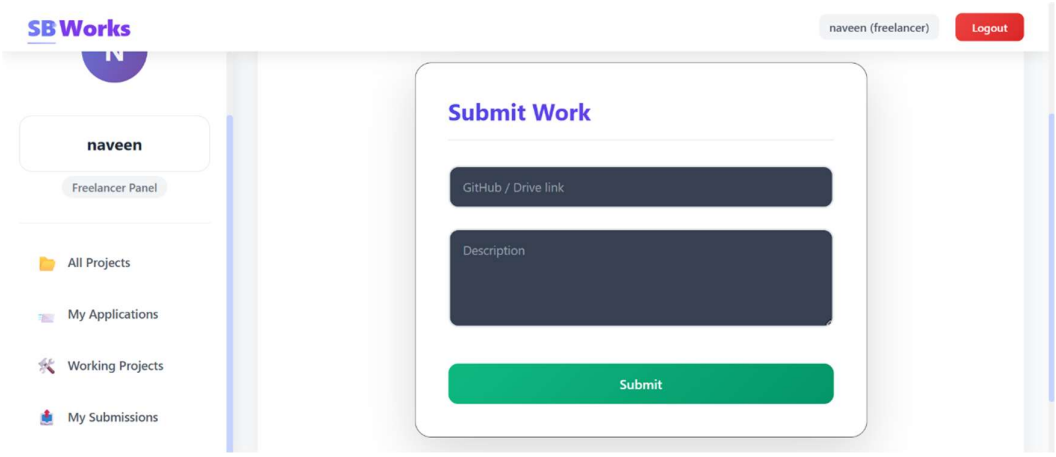


Fig : Freelancer submitting the work

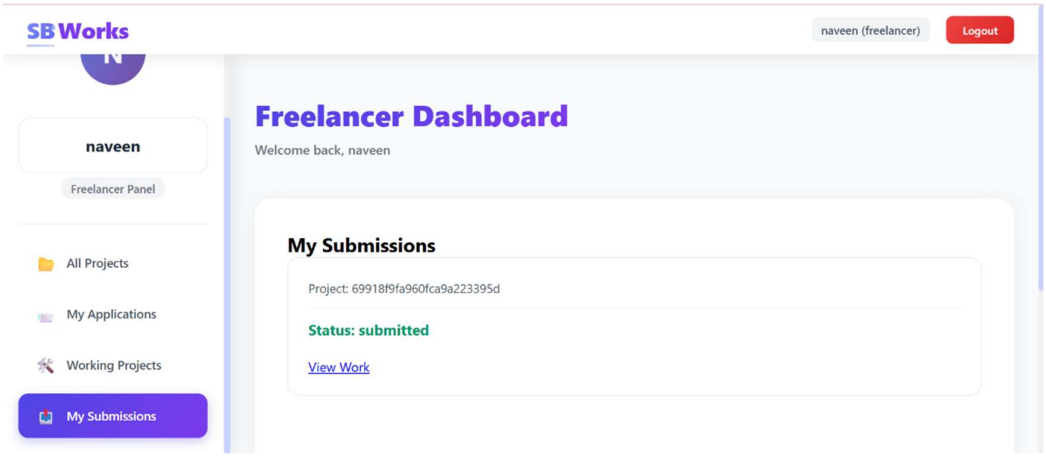


Fig : Freelancer submitting the project

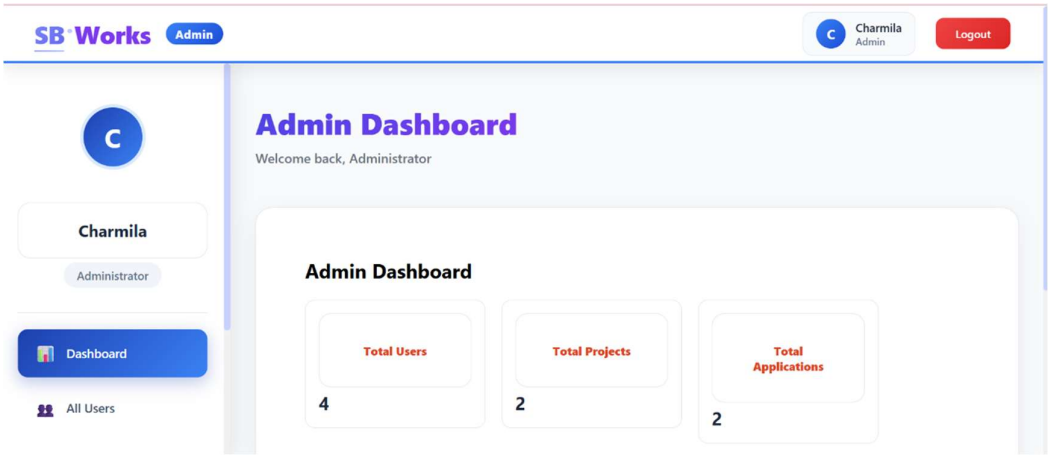


Fig : Admin dashboard

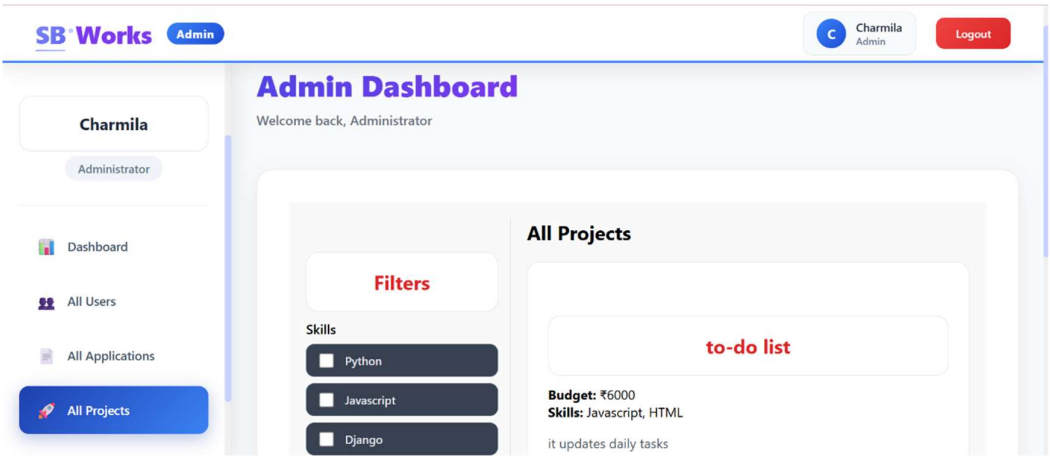


Fig : All the projects viewed by the admin

8. ADVANTAGES & DISADVANTAGES

Advantages of the Project

1. **Full-Stack Implementation**
   The project demonstrates complete frontend and backend integration using the MERN stack, providing practical experience in building and deploying real-world web applications.
2. **Role-Based Access Control**
   Separate roles for Client, Freelancer, and Admin ensure secure access control and proper authorization for sensitive operations.
3. **Scalable Architecture**
   The modular backend structure and layered architecture make the system easy to maintain, update, and extend with new features.
4. **Secure Authentication**
   JWT-based authentication and password hashing enhance data protection and prevent unauthorized access.
5. **Responsive User Interface**
   The React-based SPA provides dynamic rendering, smooth navigation, and improved user experience across devices.
6. **Efficient Project Management**
   The system centralizes project posting, bidding, and tracking, improving workflow transparency and efficiency.
7. **Database Flexibility**
   MongoDB's NoSQL structure allows flexible data modeling and easy scaling as the number of users and projects grows.
8. **Clear Separation of Concerns**
   Frontend, backend, and database layers are logically separated, improving maintainability and code readability.

Disadvantages of the Project

1. **Limited Real-Time Features**
   The system does not currently support real-time notifications or live messaging between users.
2. **No Integrated Payment Gateway**
   Payment processing is not fully automated with a secure third-party payment integration.
3. **Basic Role Management**
   Only predefined roles are supported, without granular permission customization.

4. **Token Expiration Handling**
   The absence of refresh token mechanisms may interrupt user sessions after token expiry.
5. **Limited Advanced Search**
   The platform lacks advanced filtering and recommendation algorithms for better project–freelancer matching.
6. **Initial Learning Curve**
   Understanding and managing the MERN stack may be challenging for beginners.
7. **Scalability Constraints**
   Without advanced caching and load balancing, performance may reduce under very high traffic conditions.

9. CONCLUSION

The Freelance Management System project successfully demonstrates the design and development of a secure, scalable, and user-centric full-stack web application using the MERN stack (MongoDB, Express.js, React, and Node.js). The primary objective of the project was to create a centralized digital platform that efficiently connects clients and freelancers while ensuring transparency, structured workflows, and secure data management. Through systematic planning, design, and implementation, the project achieves its goal of streamlining freelance project operations in a practical and technically sound manner.

The frontend, built with React, provides a responsive and dynamic Single Page Application that enhances user experience through smooth navigation and real-time interface updates. Separate dashboards for clients, freelancers, and administrators ensure clarity of responsibilities and efficient task management. The backend, developed using Node.js and Express.js, effectively handles business logic, API communication, authentication, and authorization. JWT-based authentication and password hashing mechanisms strengthen system security and protect sensitive user data.

The use of MongoDB as the database enables flexible data modeling and scalable storage for users, projects, bids, and related records. Proper validation and structured schema design ensure data integrity and reliability. The modular architecture followed in the backend promotes maintainability, making it easier to enhance features or fix issues in the future. By separating concerns into presentation, application, and data layers, the system maintains a clean and organized structure.

From a functional perspective, the platform provides essential features such as user registration, secure login, project posting, application management, role-based access control, and administrative monitoring. These features collectively create a controlled and efficient environment for freelance collaboration. The project also highlights the importance of structured testing, documentation, and deployment planning in building reliable software systems.

While the current implementation meets core requirements, certain limitations such as the absence of real-time notifications, advanced recommendation systems, and integrated payment gateways present opportunities for further development. These enhancements can significantly improve user engagement and system scalability. The project therefore not only serves as a working application but also as a foundation for future innovation and expansion.

Overall, the Freelance Management System reflects a comprehensive understanding of modern web development principles, secure authentication practices, database management, and layered architecture design. It demonstrates the practical application of theoretical knowledge into a real-world solution. The system is capable of handling essential freelance workflow processes efficiently and securely, making it a strong base for further enhancements and potential commercial deployment.

10. FUTURE SCOPE

The Freelance Finding project has strong potential for expansion in terms of scalability, security, and feature enhancement. With additional integrations and advanced functionalities, the system can evolve into a complete enterprise-level freelance platform. Future improvements can significantly enhance user experience, automation, and overall system performance.

1. Real-Time Communication System

Integrate live chat and instant notifications using WebSocket technology for seamless interaction.This will enable faster decision-making between clients and freelancers. It will significantly enhance user engagement and collaboration efficiency.

2. Advanced Payment Gateway Integration

Implement secure payment gateways with escrow and milestone-based release mechanisms. This will ensure secure transactions and build trust between users. Automated payment tracking will improve financial transparency and accountability.

3. AI-Based Recommendation System

Develop intelligent matching algorithms to recommend projects to freelancers and suitable freelancers to clients. Machine learning models can analyze skills, ratings, and past activity. This will improve project success rates and reduce search time.

4. Mobile Application Development

Create a mobile application using cross-platform frameworks for wider accessibility. Push notifications can improve user responsiveness and engagement. This will expand the platform's reach and usability.

5. Advanced Analytics Dashboard

Introduce visual reports and performance analytics for users and administrators. Data insights can help track project progress, earnings, and productivity. This will support informed decision-making and platform optimization.

6. Custom Role and Permission Management

Allow dynamic creation of roles with granular permission settings. This will make the system adaptable for larger organizations. It will enhance flexibility and enterprise-level usability.

7. Cloud Scalability and DevOps Enhancement

Implement containerization and automated CI/CD pipelines for smoother deployment. Load balancing and caching mechanisms can improve performance under heavy traffic. This will ensure high availability, scalability, and reliability in production environments.

11. APPENDIX

My Project Source code Files are available at :

https://drive.google.com/drive/folders/1kcJ-GexOT2mXwTMGL0MzXz7VeimVtBjo?usp=sharing

My project Demo Video link is available at :

https://drive.google.com/file/d/1Z43JVtqQgs9Gz9h719uokWyDsm-v6V2z/view?usp=sharing

Github Resopitory Link :

https://github.com/ansar-ali11/SmartBridge