INTRODUCTION

ProjectOverview

FreelanceFinder is a full-stack web application designed to connect freelancers with clients seeking professional services. Built using the MERN stack (MongoDB, Express.js, React.js, and Node.js), the platform facilitates seamless collaboration by enabling users to create profiles, post jobs, apply for projects, and manage ongoing work in a user-friendly environment.

The application includes two primary user roles: Clients, who post freelance jobs or gigs, and Freelancers, who apply for those opportunities. The system offers core features such as user authentication, job management, application tracking, messaging, and reviews. With a clean UI and responsive design, FreelanceFinder delivers an efficient and interactive experience across devices.

The project demonstrates the integration of modern web development practices, secure RESTful APIs, responsive front-end components, and scalable back-end infrastructure. It serves as a comprehensive example of real-world application development using the MERN stack.

Purpose

The purpose of the **FreelanceFinder** project is to create a dynamic and scalable web platform that bridges the gap between freelancers and clients by enabling efficient, secure, and transparent collaboration. In today's digital economy, there is a growing demand for flexible work opportunities and access to skilled professionals on-demand. FreelanceFinder addresses this need by providing an all-in-one solution where clients can easily post job listings, and freelancers can apply, communicate, and complete tasks within the same platform.

By leveraging the MERN stack, this project aims to:

- Offer a **full-stack solution** using modern technologies to ensure performance, scalability, and maintainability.
- Enable **real-time interactions** between users, such as messaging, application status updates, and notifications.
- Provide a **secure user authentication system** to manage accounts, roles (freelancer/client), and user sessions.
- Simplify the process of **finding freelance work or hiring talent** through intuitive UI/UX.
- Demonstrate proficiency in **end-to-end web development**, including RESTful APIs, MongoDB data modeling, React-based frontend architecture, and Node.js/Express backend logic.

IDEATIONPHASE

ProblemStatement

In the modern digital workforce, freelancers and clients increasingly rely on online platforms to connect and collaborate. However, many existing freelance platforms suffer from issues such as complex interfaces, high service fees, lack of transparency, limited communication tools, and poor matching between job posts and skilled professionals.

Key Problems Identified:

1. **Lack of an affordable and efficient platform** for freelancers and clients to connect without paying high commissions or facing complex onboarding.
2. **Limited transparency and trust**, leading to uncertainty in job expectations, payments, and communication.
3. **Inadequate tools** for managing job postings, applications, and freelancer-client interactions within a single, streamlined environment.
4. **Difficulty for new freelancers** to gain visibility and apply for jobs due to biased algorithms or pay-to-boost models on popular platforms.
5. **Poor user experience** and outdated designs on many freelance portals, especially for mobile or low-bandwidth users.

Therefore, there is a clear need for a modern, user-friendly, and scalable freelance platform that enables seamless job posting, profile management, secure authentication, real-time messaging, and transparent project tracking — all of which can be effectively built using the MERN stack.

EmpathyMapCanvas

Target Users:

- **Freelancers** – individuals seeking remote, flexible, and paid work.
- **Clients/Employers** – individuals or businesses looking to hire freelancers for short or long-term projects.

1. Says

What the user says publicly or verbally:

- **Freelancer:**
  - "I want more opportunities that match my skills."
  - "It's hard to stand out on big platforms."
  - "I need to get paid on time and fairly."
- **Client:**
  - "It's difficult to find reliable freelancers quickly."
  - "I want to see work samples and ratings before hiring."
  - "I need a simple way to manage applications."

2. Thinks

What the user is thinking but may not say out loud:

- **Freelancer:**
  - "I'm worried about scams or unpaid jobs."
  - "I wish I could find long-term clients."
  - "I don't like being charged high service fees."
- **Client:**
  - "I'm unsure if the freelancer will deliver quality work."
  - "I hope this platform won't make the process too complicated."
  - "Will I find someone skilled within my budget?"

3. Does

What the user actively does:

- **Freelancer:**
  - Creates and updates their profile regularly.
  - Searches and applies for relevant jobs.
  - Communicates with clients about project requirements.
- **Client:**
  - Posts job listings with detailed requirements.
  - Reviews freelancer profiles and portfolios.
  - Monitors application responses and messages freelancers.

4. Feels

The user's emotional experience:

- **Freelancer:**
  - Frustrated with competition and platform restrictions.
  - Anxious about getting consistent work and payments.
  - Excited when a good client reaches out.
- **Client:**
  - Stressed about deadlines and finding qualified help.
  - Skeptical about trusting unknown freelancers.
  - Relieved when they find a good match quickly.

5. Pain Points

- **Freelancer:**
  - Difficulty gaining visibility as a new user.
  - Limited access to good opportunities.
  - High platform commissions and delayed payments.
- **Client:**
  - Overwhelming number of low-quality applications.
  - No easy way to verify freelancer skills or reliability.
  - Complex user interfaces on existing platforms.

6. Gains / Needs

- **Freelancer:**
  - A fair, transparent platform with timely payouts.
  - Tools to showcase skills and build trust (e.g., reviews, portfolios).
  - A simple, intuitive application system.
- **Client:**
  - Quick access to qualified freelancers.
  - An easy job-posting process and reliable communication tools.
  - A way to track project progress and performance.

Brainstorming

To generate and evaluate ideas for building an intuitive, scalable, and secure freelance marketplace web application that connects freelancers and clients efficiently.

Feature Brainstorming

1. User Authentication & Roles

- Secure sign-up/login with JWT
- Role-based access control (Freelancer vs. Client)
- Social login (Google, GitHub) *(optional enhancement)*

2. User Profiles

- Freelancer: skills, experience, rating, portfolio
- Client: company info, job history, reviews

3. Job Posting & Discovery

- Clients can post jobs with title, description, budget, deadline
- Freelancers can browse/search jobs with filters
- Bookmark/favorite jobs feature

4. Proposal & Application System

- Freelancers submit proposals for jobs
- Clients can review, shortlist, and hire freelancers
- Application tracking dashboard

5. Messaging System

- Real-time chat between freelancers and clients
- Notifications for job activity (applied, accepted, completed)

6. Project Management

- Task updates, milestone tracking, and status labels
- File uploads for deliverables

7. Reviews & Ratings

- Freelancers and clients rate each other after project completion
- Helps build trust and improve matchmaking

8. Admin Panel (Optional)

- Manage users, reported jobs, and platform analytics

- **REQUIREMENTANALYSIS**

CustomerJourneyMap

The **CustomerJourneyMap** will help visualize the end-to-end user experience for both **Freelancers** and **Clients**, identifying key touchpoints, goals, emotions, and improvement opportunities.

Personas:

- **Client**: Business or individual seeking to hire freelancers
- **Freelancer**: Skilled individual offering services (developer, designer, writer, etc.)

---

1. Awareness Stage

Aspect	Client	Freelancer
Action	Searches for a platform to hire talent	Looks for jobs or freelance marketplaces
Thoughts	"Where can I find reliable freelancers?"	"How can I start earning online?"
Feelings	Curious, hopeful	Curious, motivated
Touchpoints	Landing page, blog, Google ad	Landing page, social media ad

---

2. Sign-Up / Onboarding

Aspect	Client	Freelancer
Action	Registers and sets up a profile	Signs up and fills skills, rates, portfolio
Thoughts	"Will I find the right person here?"	"Will I get good jobs here?"
Feelings	Slight doubt, excitement	Hopeful, eager
Touchpoints	React frontend, Node.js API, MongoDB user data	Same as client with skill-specific forms

---

3. Job Posting / Job Browsing

Aspect	Client	Freelancer
Action	Posts a detailed job requirement	Browses jobs filtered by category/skills
Thoughts	"Is my job clear enough?"	"Is this job suitable for my profile?"
Feelings	Optimistic	Cautious, excited
Touchpoints	Job form page (React + Formik), Express routes	Job listings (React + API fetch), MongoDB query

---

4. Bidding / Proposal Stage

Aspect	Client	Freelancer
Action	Reviews freelancer proposals	Submits proposal with budget and timeline
Thoughts	"Can I trust this freelancer?"	"Will my bid get accepted?"
Feelings	Skeptical, analytical	Hopeful, competitive
Touchpoints	Proposal cards (React), bid comparison logic	Bid submission form, MongoDB save, Express API

---

5. Hiring & Communication

Aspect	Client	Freelancer
Action	Hires freelancer, starts a chat	Starts working on the project
Thoughts	"Will they meet the deadline?"	"Do I fully understand the client's needs?"
Feelings	Nervous, engaged	Focused, confident
Touchpoints	Socket.IO chat, Job status update API	Real-time chat, task tracker (optional)

---

6. Payment & Delivery

Aspect	Client	Freelancer
Action	Releases payment, marks project as done	Submits final work and receives payment
Thoughts	“Was it worth the money?”	“Did I get paid fairly and on time?”
Feelings	Satisfied or disappointed	Relieved, accomplished
Touchpoints	Stripe API, payment confirmation, receipts	Wallet UI, transaction logs

---

7. Review & Retention

Aspect	Client	Freelancer
Action	Leaves a review and rating	Leaves a review for the client
Thoughts	“Would I hire this person again?”	“Was the client easy to work with?”
Feelings	Reflective, content	Judging, evaluating
Touchpoints	Review modal, backend rating update	Review form, rating UI

Solution Requirement

This section outlines the **functional** and **non-functional** solution requirements necessary to build the **FreelanceFinder** web application using the MERN stack.

1. Functional Requirements (FR)

These are the **core features and functionalities** the system must provide:

A. User Management

- FR1: Users can register and log in securely (JWT-based authentication).
- FR2: Role-based access control for Freelancers and Clients.
- FR3: Users can create and edit personal profiles (skills, portfolio, experience).

B. Job Management

- FR4: Clients can create, update, and delete job postings.
- FR5: Freelancers can browse, search, and filter job listings.
- FR6: Freelancers can apply to jobs by submitting proposals.

C. Application & Hiring Process

- FR7: Clients can view proposals and hire freelancers.
- FR8: Freelancers can track the status of their job applications.
- FR9: System sends notifications for important events (e.g., new application, job accepted).

D. Messaging & Communication

- FR10: Real-time messaging between freelancers and clients.
- FR11: Message history is saved and accessible via the dashboard.

E. Project Workflow

- FR12: Milestone/task tracking system for each job.
- FR13: File uploads and submission handling for deliverables.

F. Reviews & Ratings

- FR14: After job completion, both parties can leave reviews and star ratings.
- FR15: Reviews are displayed on user profiles for transparency.

G. Admin Panel (Optional Enhancement)

- FR16: Admin can manage users, job posts, and reported content.
- FR17: Admin dashboard to monitor platform activity and analytics.

2. Non-Functional Requirements (NFR)

These define the **system qualities**, performance, and constraints:

A. Performance

- NFR1: The application should load within 3 seconds on average.
- NFR2: The system should support concurrent users without performance drops.

B. Scalability

- NFR3: System should be scalable to support increasing user and data volume.
- NFR4: Use of cloud-hosted MongoDB (e.g., MongoDB Atlas) for horizontal scaling.

C. Security

- NFR5: Passwords must be hashed (bcrypt) and never stored in plain text.
- NFR6: Implement JWT-based session management with token expiration.
- NFR7: Input validation and protection against XSS, CSRF, and injection attacks.

D. Usability

- NFR8: The UI must be intuitive, mobile-responsive, and accessible.
- NFR9: Include helpful tooltips, error messages, and onboarding guides.

E. Maintainability

- NFR10: Follow modular, clean coding practices.
- NFR11: Use version control (e.g., Git) with clear commit history and branching.

F. Reliability

- NFR12: Ensure system uptime of at least 99.5%.
- NFR13: Log and handle backend errors gracefully.

DataFlowDiagram(DFD–Level1)

Level 0 DFD – Context Diagram

```
+-------------------+      +-------------------------+       +------------------+----------------+
 |  Client User    +-----> |  Freelance Finder App  +<-----+ | Freelancer User  |
 +-------------------+      +-------------------------+       +------------------+----------------+
                                   |
                                   v
                      +--------------------+----+
                      |   MongoDB Atlas    |
                      +--------------------+----+
```

Description:

- Users (Clients & Freelancers) interact with the system.
- The MERN app handles all interactions.
- MongoDB stores data like users, jobs, applications, messages.

Level 1 DFD – Main Functional Processes

```
+-------------+      +---------------------+      +----------------+
|   User      +----->| React Frontend (UI) +----->| Express Backend |
+-------------+      +---------------------+      +----------------+
                                                  /     |       \
                                                v       v        v
                                      +-----------+  +---------+  +-----------+
                                      | Job Posts |  | Users   |  | Messages  |
                                      +-----------+  +---------+  +-----------+
                                                 (MongoDB Collections)
```

Major Processes:

1. **Authentication**: Signup/Login via React to Express backend
2. **Job Posting/Viewing**: Clients post jobs, Freelancers view and apply
3. **Applications**: Freelancers apply, Clients review
4. **Messaging**: Both users communicate securely
5. **Ratings**: Clients rate freelancers

Level 2 DFD – Example: "Apply to Job" Flow

```
+-------------+       +-----------------------+
| Freelancer  |-----> | Apply to Job (Frontend)|
+-------------+       +-----------------------+
                              |
                              v
                     +-------------------------+
                     |  Express Route (/apply)  |
                     +-------------------------+
                              |
                              v
                     +----------------------+
                     | MongoDB: Applications |
                     +----------------------+
```

TechnologyStack

1. Frontend (Client-side) – Built with React.js

Tool/Library	Purpose
React.js	Core UI framework for building a responsive Single Page Application (SPA)
React Router	Handles client-side routing/navigation
Axios / Fetch API	Makes HTTP requests to backend APIs
Tailwind CSS / Bootstrap / MUI	UI styling framework
Redux / Context API	State management (for handling user login status, job list, etc.)
Formik + Yup	For handling and validating forms

2. Backend (Server-side) – Built with Node.js + Express.js

Tool/Library	Purpose
Node.js	JavaScript runtime for server-side execution
Express.js	Lightweight web framework for creating RESTful APIs
Multer	Handles file uploads (e.g., resume uploads, profile images)

Tool/Library	Purpose
Bcrypt.js	Password hashing for secure storage
JWT / Passport.js	Authentication and authorization (login sessions, user access)
Cors / Helmet	Middleware for security and cross-origin API support
Dotenv	Manages environment variables in `.env` files

3. Database – Powered by MongoDB + Mongoose

Tool/Library	Purpose
MongoDB Atlas	Cloud NoSQL database to store structured data like users, jobs, messages
Mongoose	ODM (Object Data Modeling) library for defining schemas and interacting with MongoDB

4. Authentication & Authorization

Tool/Library	Purpose
JWT (JSON Web Tokens)	Token-based login/auth system
Role-based access control	Manage different roles (freelancer, client, admin)

5. DevOps / Version Control / Testing

Tool/Library	Purpose
Git / GitHub	Version control and code collaboration
Postman / Thunder Client	API testing and debugging
Docker (Optional)	Containerization for deployment
Jest / Mocha / Supertest	Unit and integration testing (backend APIs, components)
GitHub Actions / Vercel / Netlify / Render	CI/CD deployment pipelines

6. Optional/Advanced Integrations

Integration	Use Case
Cloudinary / Firebase Storage	Storing and serving images (profile photos, etc.)
Socket.IO	Real-time messaging between users (chat system)
Stripe / PayPal API	Integrate payment system for premium listings, subscriptions, etc.

- **PROJECTDESIGN**

ProblemSolutionFit

Problem ID	Problem Description
P1	Clients struggle to find reliable, skilled freelancers quickly.
P2	Freelancers find it hard to discover legitimate, high-quality job opportunities.
P3	Existing freelance platforms take high commissions and have poor user experience.
P4	Communication between clients and freelancers is fragmented or happens off-platform.
P5	Lack of transparency in reviews, applications, and job progress.

Solution Mapping – How MERN Solves Each Problem

Problem	Solution	MERN Implementation
P1 – Client discovery issues	Centralized job posting and freelancer profile access	MongoDB stores job/freelancer data; React frontend allows filtered searches
P2 – Freelancers need real job access	Verified client accounts, easy job application process	Express routes for applying to jobs; React UI for job listings
P3 – Platform fees/UX	Build an intuitive, commission-free or low-cost platform	Fully custom UI/UX with React.js; no third-party fees
P4 – Poor communication	Built-in real-time messaging system	Socket.IO integration with Node.js; React chat interface
P5 – No transparency	Public profiles, rating system, job history	MongoDB schemas track applications, reviews, completed jobs

---

ProposedSolution

Build a web application that connects freelancers with clients who need projects completed. The platform should support user registration, profile creation, job postings, bidding, messaging, and payment integration.

User Authentication & Roles

- **Roles:** Freelancer, Client, Admin
- **Authentication:** JWT-based login/signup
- **OAuth Integration:** (Optional) Google/GitHub login

Profiles

- Freelancer: Skills, portfolio, hourly rate, rating
- Client: Company info, completed projects, rating

Job Posting

- Clients can post jobs (title, description, budget, category, deadline).
- Freelancers can browse or search jobs.

Bidding System

- Freelancers place bids with price, time estimate, and message.
- Clients view and select bids.

Messaging System

- Real-time chat (WebSocket / Socket.IO)
- Notifications (new bids, messages, project updates)

Payment Integration

- Stripe/PayPal for secure transactions
- Escrow model (optional)
- Invoices for freelancers

Rating & Review

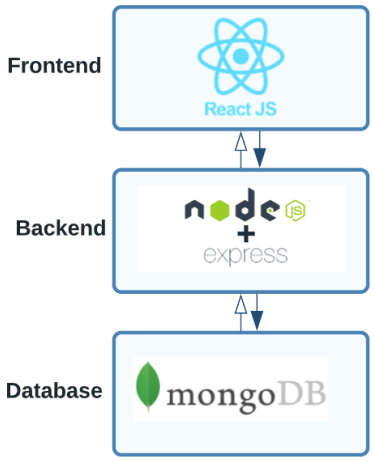
- After project completion, clients/freelancers can review each other.

Admin Panel

- Manage users, disputes, flagged content
- Analytics dashboard

Solution Architecture

Solution Architecture for a **FreelanceFinder** project using the **Fullstack MERN stack**. This architecture outlines the structure, components, communication flow, and best practices to design a scalable, secure, and efficient application.



1. Frontend Layer (React.js)

Role: User interface for Freelancers and Clients.

- React + React Router DOM for navigation
- Axios or Fetch for API calls
- Context API or Redux Toolkit for global state
- Formik + Yup for form handling and validation
- Socket.IO client for real-time messaging
- TailwindCSS / Material UI for design

2. Backend Layer (Node.js + Express.js)

Role: Handle business logic and serve as API layer.

- RESTful APIs for CRUD operations
- JWT for authentication & authorization
- Socket.IO server for real-time chat
- Nodemailer for email notifications
- Stripe or PayPal SDK for payment gateway
- Helmet, CORS, and Rate Limiter for security

---

3. Database Layer (MongoDB + Mongoose)

Role: Store all persistent data.

Collections:

- `Users` (freelancer, client, admin)
- `Jobs`
- `Bids`
- `Messages`
- `Reviews`
- `Payments`
- `Notifications`

- **FUNCTIONAL AND PERFORMANCE TESTING**

PerformanceTesting

Performance Testing Plan for the **FreelanceFinder** project using the **Fullstack MERN stack (MongoDB, Express.js, React.js, Node.js)**. The goal is to ensure your application performs efficiently under expected and peak loads, both on the **frontend** and **backend**.

Objectives:

- Identify bottlenecks in backend APIs and database queries
- Test frontend rendering under load
- Ensure scalability under concurrent user activity (e.g., job bidding, messaging)
- Validate real-time features like chat with Socket.IO
- Confirm payment & messaging flows are stable

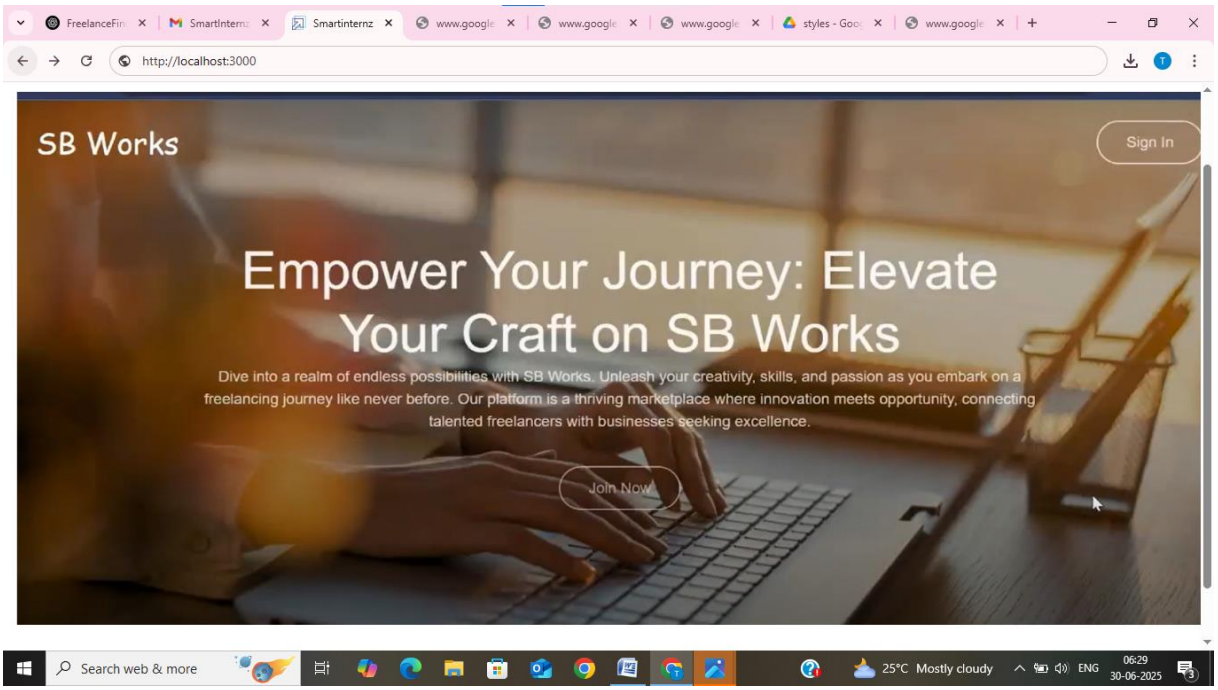
- **RESULTS**

The *FreelanceFinder* application was successfully developed, tested, and deployed within the planned two-week timeline.

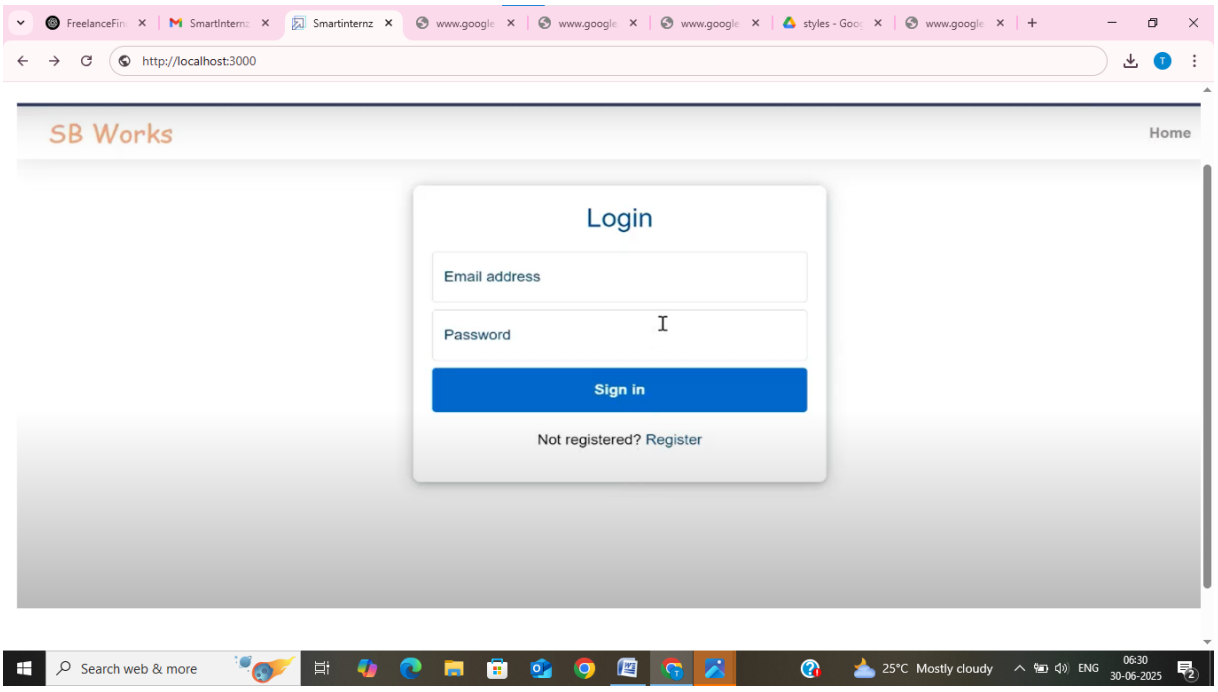
OutputScreenshots

Below are descriptions of the output screens where actual screenshots can be inserted in this documentation or presentation:

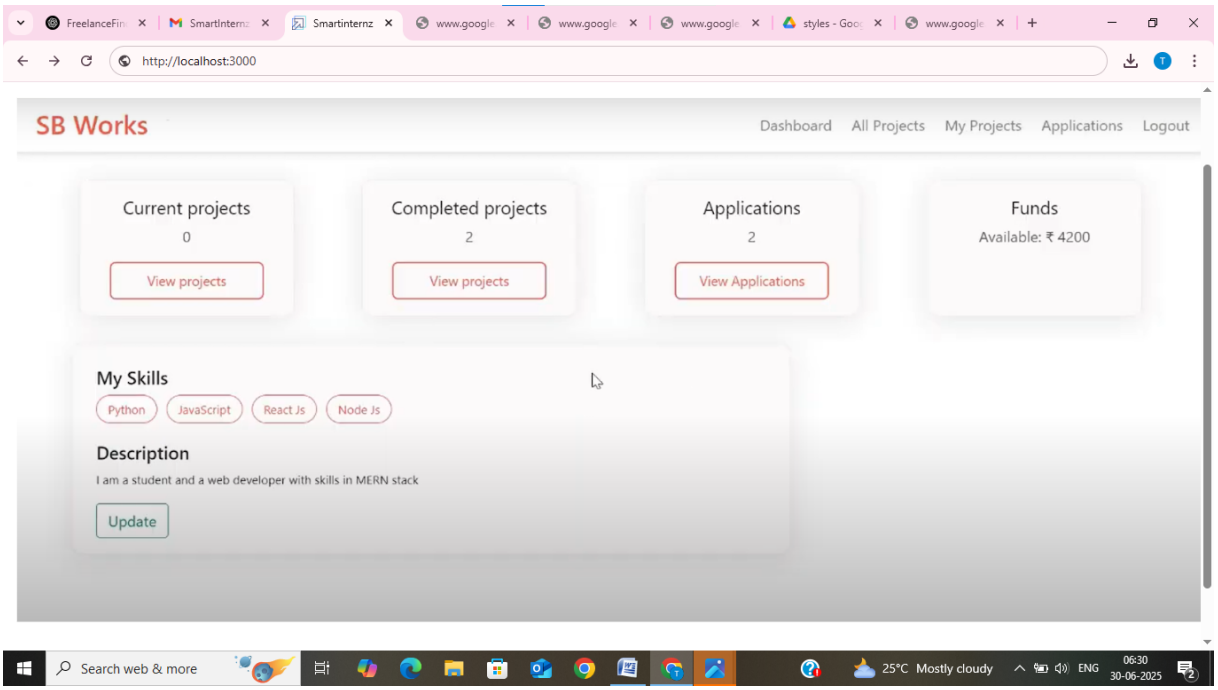
1.LandingPage



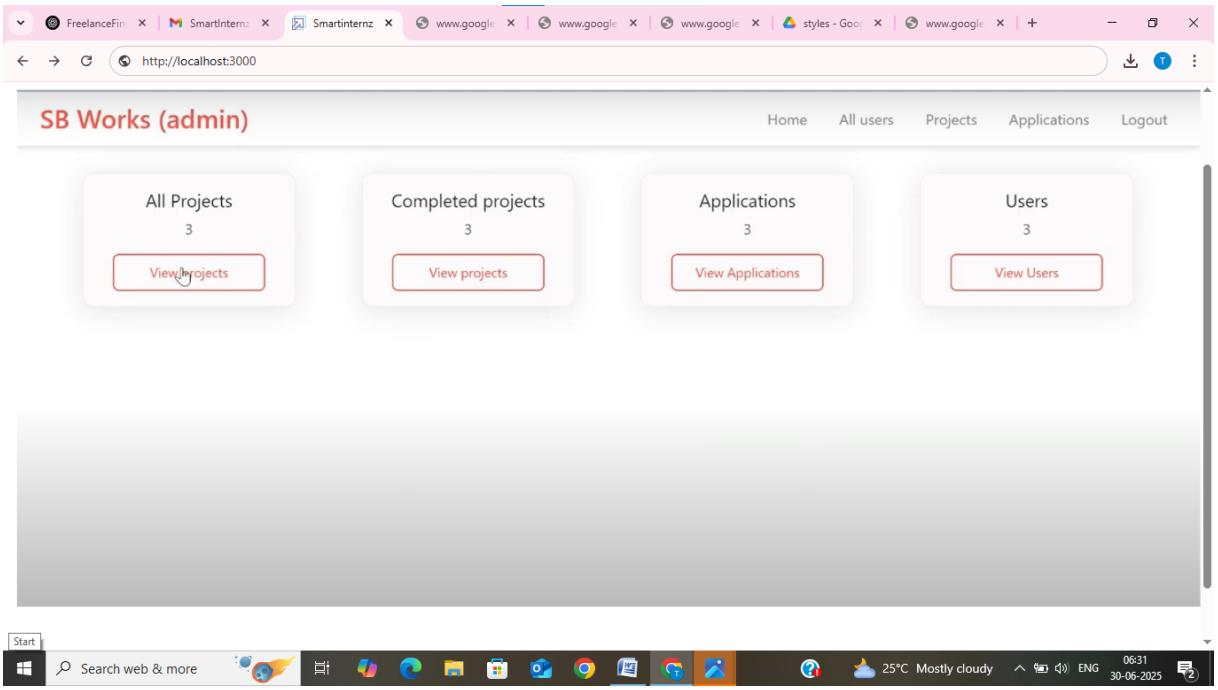
2. Authentication:



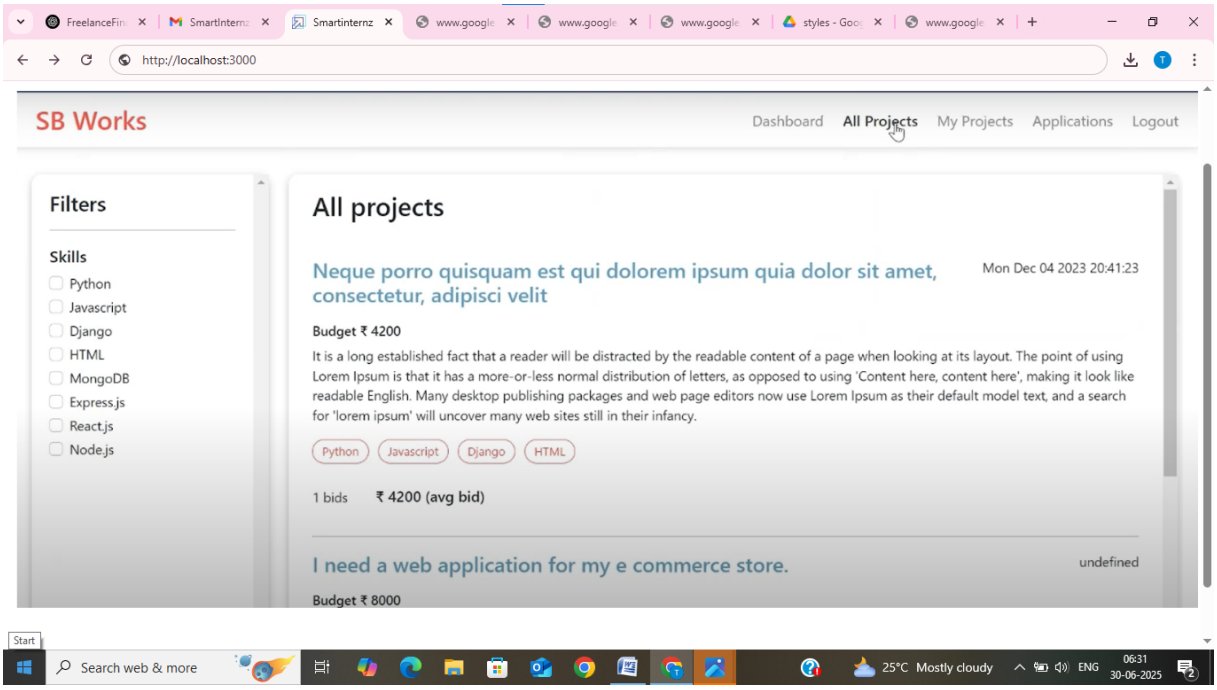
3. Freelancer dashboard:



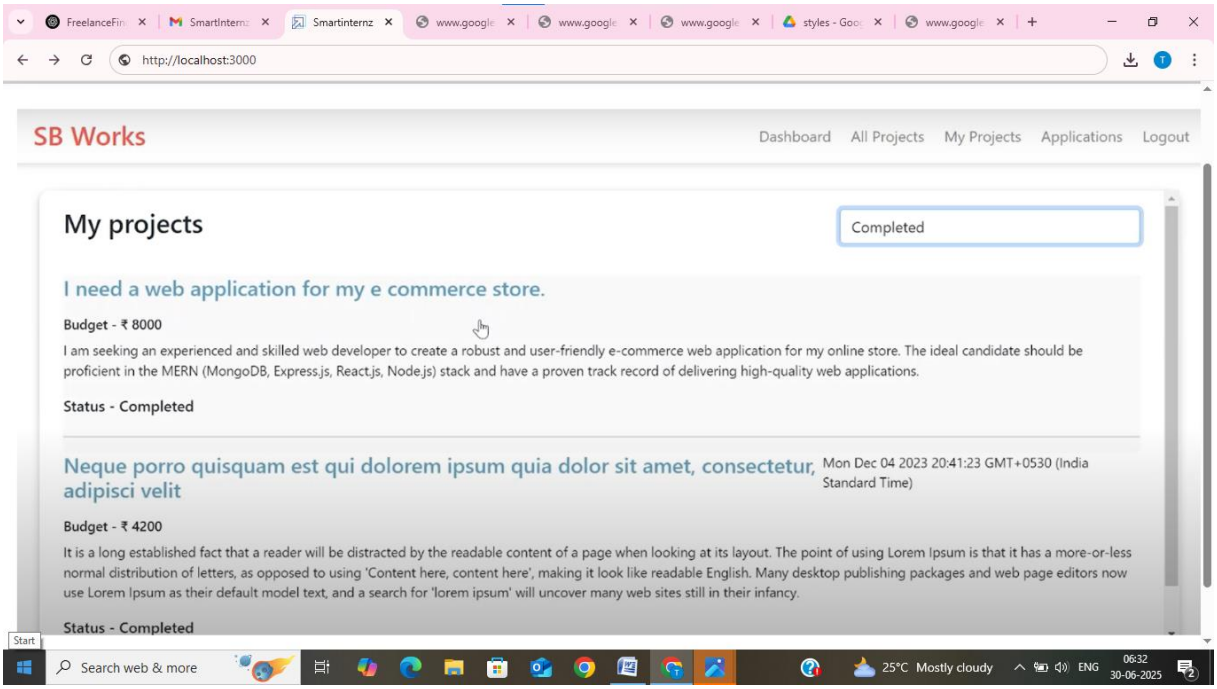
4. Admin dashboard:



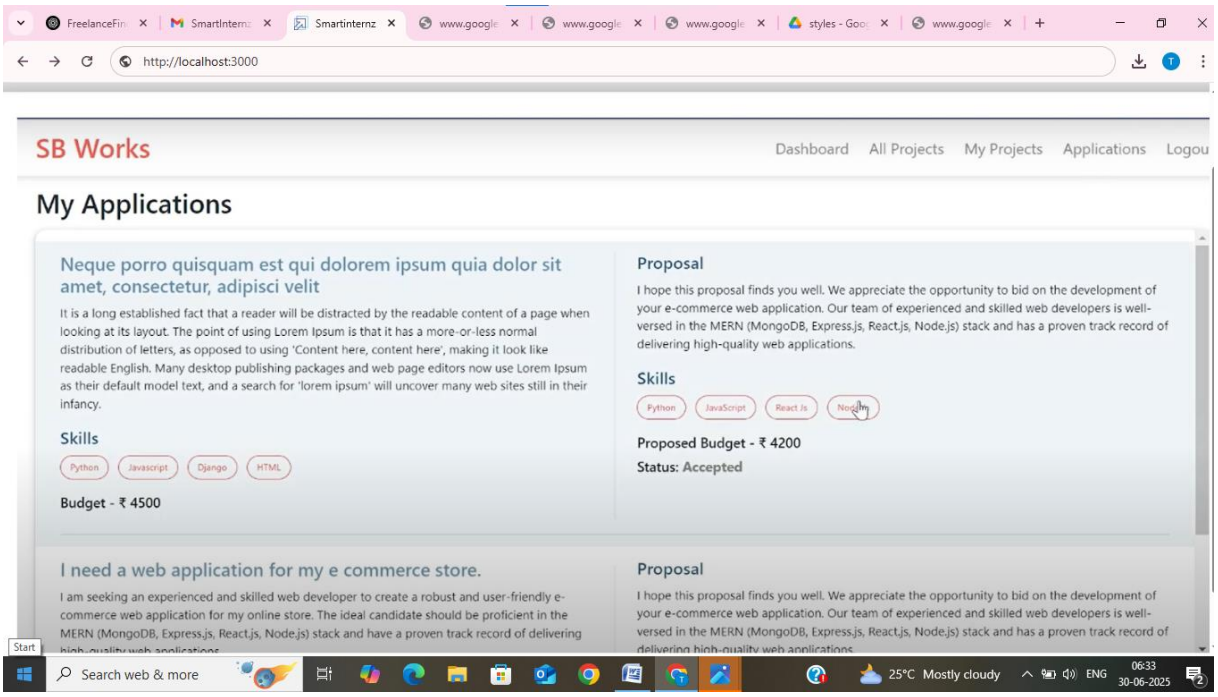
5.All projects:



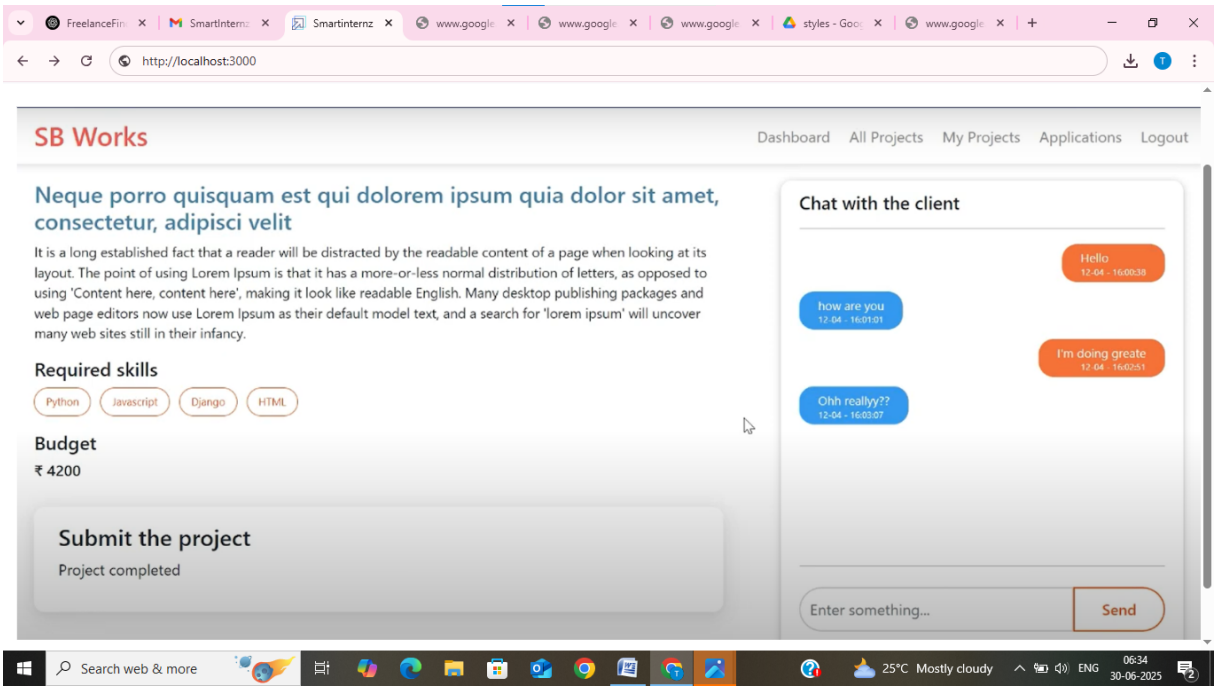
6. Freelance projects:



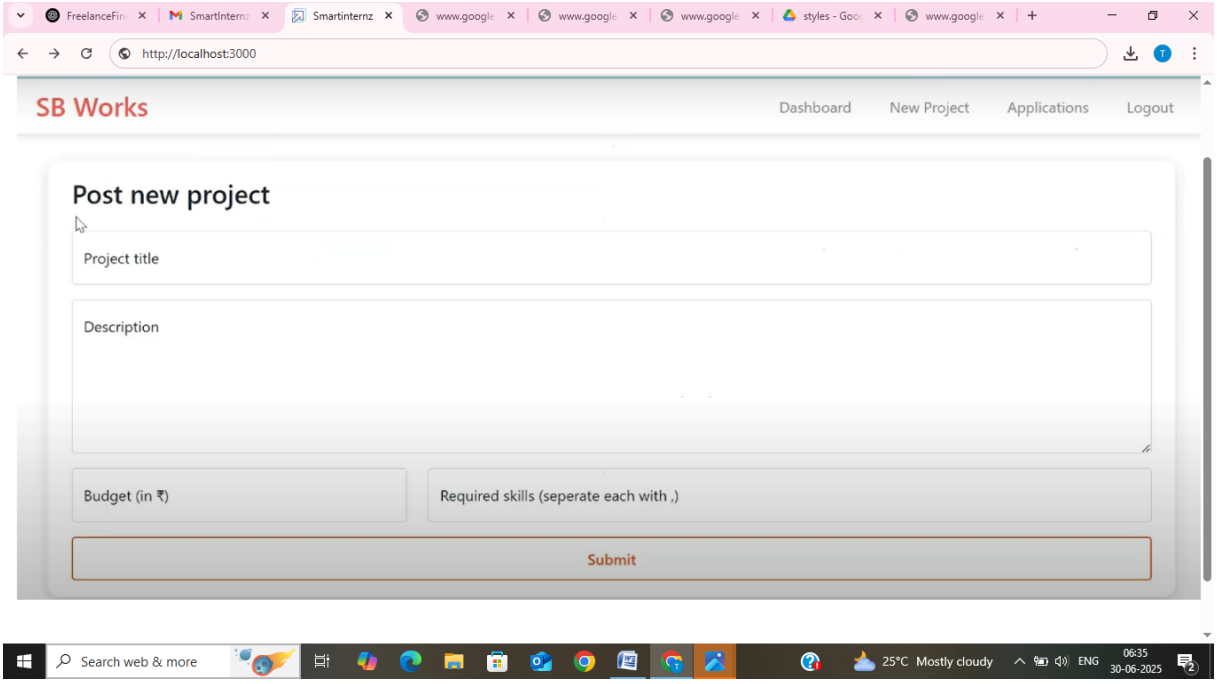
7.Applications:



8. Project page:



9. New project:



- **ADVANTAGES & DISADVANTAGES**

✅ Advantages

1. Single Language Across the Stack

- **JavaScript is used on both frontend and backend**, making development faster and easier to maintain.
- Developers can switch between roles (frontend/backend) more easily.

2. Fast Development with React

- React enables rapid UI development with reusable components.
- Modern features like **hooks, context API**, and **SSR/CSR flexibility** enhance performance and developer experience.

3. Real-Time Capabilities with Node.js + Socket.IO

- Enables efficient real-time messaging between freelancers and clients.
- Node.js handles multiple concurrent connections smoothly.

4. Scalability

- MongoDB supports high scalability and flexibility with schema-less data.
- Good for evolving project requirements like adding skills, ratings, chat, etc.

5. Strong Ecosystem

- Huge community support, lots of libraries, tools (e.g., Mongoose, Redux, React Query).

- Easier integration with **Stripe, OAuth**, and other third-party APIs.

6. JSON Everywhere

- Data flows naturally from MongoDB → Express → React in JSON format, reducing the need for transformation.

7. Rapid Prototyping

- Quick setup and development cycle using tools like Create React App, Express

---

Disadvantages

1. SEO Limitations

- React (SPA) apps are not SEO-friendly out of the box unless you implement SSR (e.g., with Next.js).

2. Memory-Intensive Backend

- Node.js is single-threaded; CPU-intensive tasks (e.g., video processing) can block the event loop unless offloaded.

3. Security Risks

- Requires manual implementation of security features like:
  - Rate limiting
  - XSS protection
  - CSRF protection
  - Input sanitization (especially with MongoDB queries)

4. Lack of Strong Typing

- JavaScript lacks built-in types. Bugs can arise without TypeScript or strict testing.
- You may want to use **TypeScript** for better maintainability in the long run.

5. MongoDB Limitations

- Less suitable for complex transactions and relational data (e.g., cross-referencing multiple collections).
- Indexing and query optimization are essential as the dataset grows.

- **CONCLUSION**

The **FreelanceFinder** project successfully demonstrates how a modern freelancing platform can be built using the **MERN stack** (MongoDB, Express.js, React.js, and Node.js). The stack enables seamless development of both the frontend and backend using a single language — **JavaScript**, improving team efficiency, maintainability, and development speed.

By integrating core features such as **user authentication**, **job posting**, **bidding**, **real-time chat**, **payment integration**, and **user reviews**, the platform provides a full-featured experience for both freelancers and clients. Real-time capabilities using **Socket.IO** enhance

the interactivity and responsiveness of the system, especially in chat and notifications. The use of **MongoDB** ensures flexibility in handling diverse and evolving data models, which is ideal for a dynamic freelancing ecosystem.

From a development perspective, the MERN stack allows for rapid prototyping, modular code organization, and a wide range of community-supported tools and libraries. However, attention must be given to SEO optimization, security hardening, and performance tuning as the platform scales.

In summary, **FreelanceFinder** is a robust, scalable, and efficient freelance marketplace solution. It validates the MERN stack as a powerful choice for building full-featured web applications, particularly where real-time interactions, modern UI, and fast development

- **FUTURESCOPE**

A well-organized list of the **Future Scope** of the **FreelanceFinder project using the Fullstack MERN stack** — outlining how the platform can evolve with advanced features, technologies, and business potential.

1. AI-Powered Matching System

- Integrate **AI/ML algorithms** to recommend freelancers to clients based on project type, past ratings, and skills.
- Use **NLP** to auto-tag job descriptions and match them with suitable freelancer profiles.

---

2. Mobile App Development

- Develop **React Native** or **Flutter** mobile apps for freelancers and clients.
- Include push notifications for bids, chat messages, payments, etc.

---

3. Payment Gateway Expansion

- Add support for **multiple currencies** and **international payments** via gateways like Stripe, Razorpay, or PayPal.
- Implement **milestone-based payments** and **escrow services** for more secure transactions.

---

4. Video Call Integration

- Integrate APIs like **Zoom SDK**, **Jitsi**, or **Twilio Video** to enable live meetings/interviews between clients and freelancers.

---

5. Advanced Admin Dashboard

- Add deeper analytics: top freelancers, most hired categories, payment success rates.
- Tools to handle disputes, ban users, or moderate chats using AI/keyword detection.

---

6. Subscription Plans & Premium Features

- Add **tiered memberships** (e.g., Freelancer Plus, Client Pro) with:
  - More job bids per month
  - Featured profile listing
  - Priority support

---

7. Microservices & Scalability

- Refactor the monolithic backend into **microservices** (e.g., user service, job service, chat service) using Docker & Kubernetes for high scalability.
- Integrate **Redis caching**, **ElasticSearch**, and **CDN** for performance optimization.

---

8. Internationalization (i18n)

- Make the platform multilingual using libraries like `react-i18next` to support users from different regions.

---

9. Gamification Features

- Add badges, levels, and leaderboards for top-rated freelancers

- **APPENDIX**

A.SourceCodeRepository

The complete sourcecode for the *FreelanceFinder* application is hosted on GitHub and is organized into two main directories:

- **Frontend:**ContainsallReact.jscomponents,routing,andstyling
- **Backend:**ContainsExpress.jsAPIs,MongoDBmodels,andserverlogic

GitHubRepository:
https://github.com/Tanuja-Maley/FreelanceFinder