FreelanceFinder: Discovering Opportunities, Unlocking Potential

1. Introduction

Project Title: FreelanceFinder: Discovering Opportunities, Unlocking Potential

Team Members:

Team ID: LTVIP2025TMID43777

Team Size: 4

Team Leader: Domala Sri Varalakshmi Kranthi

Team Member: D Chandra Mouli

Team Member: D Durga Santhosh

Team Member: Mamidi Veera Venkata Surya Durga Seeta

2. Project Overview

Purpose:

FreelanceFinder is a web platform designed to connect freelancers with clients seeking project-based talent. Built using the MERN stack (MongoDB, Express.js, React.js, Node.js) and core web technologies like HTML, CSS, and JavaScript, it provides a secure, scalable, and intuitive interface for posting, bidding on, and managing freelance jobs.

It offers functionality for user registration, job posting, real-time applications, and project submissions. Admin tools help oversee platform activity. The project replicates key realworld workflows such as freelancer-client interactions, job bidding, and task delivery.

In the current gig economy, there's a growing need for efficient platforms that connect businesses with freelance professionals. FreelanceFinder addresses this demand by offering a secure, modern, and scalable web-based platform that bridges the communication gap between clients and freelancers.

Clients can not only post their requirements but also evaluate freelancers based on profile ratings, skills, and previous work history. Freelancers get exposure to a wide range of projects and can apply based on their area of expertise, expected compensation, and availability.

This platform also promotes transparency and accountability by integrating submission tracking and approval workflows, thereby enabling better project lifecycle management.

Features:

- - Secure User Registration and Login
- - Role-based Dashboards for Freelancers and Clients
- - Project Posting, Bidding, and Application Workflow
- - Freelancer Profile Management
- - Project Submission and Review System
- - Admin Panel for Oversight
- - Real-time Messaging and Notifications
- - Fully Responsive Interface

3. Architecture

Frontend:

The frontend of FreelancerFinder is developed using React.js, one of the most popular JavaScript libraries for building interactive user interfaces. React provides a componentbased architecture that ensures reusability and maintainability of code. This structure helps to scale the frontend easily as the application grows. Routing in the application is handled using React Router DOM, allowing for seamless navigation between different views such as the home page, product details, cart, login/register, admin dashboard, etc. Each page is mapped to a specific route that renders the appropriate React component, and route protection is enforced through custom logic that restricts access to admin or authenticated users. To manage the application's global state, the React Context API is used in combination with custom hooks. This setup handles authentication state, cart data, and user details across different components, removing the need for excessive prop drilling. The state management flow ensures that changes in user or cart status are reflected in real time across all dependent UI components. For communication between the frontend and backend, Axios is used as the HTTP client library. Axios is configured with base URLs and headers using interceptors for clean and centralized API handling. Each API request from the frontend sends data to the appropriate backend route and handles the response or error accordingly, providing feedback to the user. The component hierarchy follows a clean structure with separation between pages, components, and utilities. Pages represent full views, like "Home" or "Product", while components represent reusable UI blocks like "ProductCard", "Navbar", or "Footer". Utility files include API handlers and context providers. Styling is done using a combination of CSS Modules and custom stylesheets, ensuring each component has scoped styling and consistency across the application. The frontend is fully responsive, utilizing Flexbox and Grid for layout and media queries to adjust elements on smaller screen devices. This makes the UI suitable for mobile, tablet, and desktop views. Accessibility considerations such as ARIA roles, semantic HTML, and keyboard navigation are partially implemented to improve user experience.

Backend:

The backend of FreelancerFinder is built using Node.js with the Express.js framework. It provides the logic and data management necessary to support the frontend UI and performs all the operations related to authentication, data validation, user management, product CRUD operations, and order processing. The Express application is structured in a modular format with folders such as routes, controllers, models, middleware, and config. This structure enables separation of concerns and better organization of backend functionality. • Routes define the HTTP endpoints that the frontend interacts with, such as /api/users, /api/products, and /api/orders. • Controllers handle the logic of what happens when a particular endpoint is called, such as creating a user, fetching product data, or processing an order. • Models define the structure of the data using Mongoose schemas. • Middleware is used for tasks like authenticating JWT tokens, checking admin access, logging, and handling errors. Authentication in the backend is handled using JWT (JSON Web Tokens). When a user logs in successfully, the server issues a signed token which the client stores in HTTP-only cookies for security. Each protected route in the backend checks the validity of this token to allow or deny access to private resources. Admin-specific routes have an additional middleware layer that checks the user's role. Error handling is implemented using centralized middleware that captures synchronous and asynchronous errors and sends appropriate responses back to the client in a structured format. Request data is validated using both manual checks and Mongoose validation to ensure reliability and integrity of stored data. The backend server is configured to run on port 5000 during development and uses the dotenv package to securely load environment variables, including database URIs, secret keys, and port settings.

Database:

The database layer uses MongoDB, a flexible NoSQL document database, to store and manage all application data including users, products, orders, and cart information. MongoDB is integrated with the backend using Mongoose, an ODM (Object Document Mapper) that provides a schema-based solution to model the application data. The User schema stores details such as username, email, password (hashed), role (user/admin), and timestamps. The Product schema includes name, image URL, price, category, stock quantity, and description. The Order schema keeps track of user ID, ordered products, payment status, shipping details, and order history. Mongoose also supports schema validation and pre/post hooks, which are used to hash user passwords before saving them to the database and to perform cleanup operations on cascading deletions. Relations between users and orders are modeled using references, enabling efficient data retrieval via population queries. MongoDB can be hosted either locally for development or on MongoDB Atlas for production. The connection configuration is maintained in a dedicated file and imported into the main server file to establish a persistent and secure link to the database. The structure and indexing of collections are optimized for read and write operations to ensure smooth performance under high load. The database supports dynamic data updates, which means that when a user adds a product to the cart or places an order, the corresponding documents are updated in real time. All critical operations are secured and validated before being committed to the database.

4. Setup Instructions

Prerequisites:

- - Node.js
- - MongoDB
- - Git

Installation:

1. Clone the repository:

```
git clone https://github.com/yourusername/freelancefinder.git
```

2. Navigate to client and install dependencies:

```
cd client && npm install && npm install vite --save-dev
```

3. Navigate to server and install dependencies:

```
cd ../server && npm install
```

4. Set up .env files with MongoDB URI and JWT secret keys.

5. Folder Structure

Client:

- - /src/components: Navbar, JobCard, Footer
- - /src/pages: Login, Register, Dashboard

- - /src/context: Auth and App context
- - /src/api: Axios configuration and services
- - /src/App.js: Main routes and layout

Server:

- - /routes: User, Project, Application, Chat
- - /controllers: Logic handlers for APIs
- - /models: User, Project, Application schemas
- - /middleware: JWT Auth and error handler - /config: MongoDB connection

6. Running the Application

Frontend:

cd client && npm run dev

Backend:

cd server && npm run dev

7. API Documentation

User APIs

- - POST /api/register: Register a new user
- - POST /api/login: Authenticate user
- - GET /api/fetch-users: Get all users (admin)

Freelancer APIs

- - GET /api/fetch-freelancer: Get freelancer profile
- - PUT /api/update-freelancer: Update profile

Project APIs

- - GET /api/fetch-project/:id: Get a project
- - GET /api/fetch-projects: Get all projects
- - POST /api/new-project: Post a new project

Application/Bid APIs

- - POST /api/make-bid: Submit a bid
- - GET /api/fetch-applications: View applications
- - POST /api/approve-application: Approve bid
- - POST /api/reject-application: Reject bid

Project Submission APIs

- - POST /api/submit-project: Submit final work
- - POST /api/approve-submission: Client approves
- - POST /api/reject-submission: Client rejects

Chat APIs

- - GET /api/fetch-chats: Retrieve messages

8. Authentication

- - JWT-based authentication with HTTP-only cookies.
- - Middleware protects private routes.
- - Admin and user role-based access control.

9. User Interface

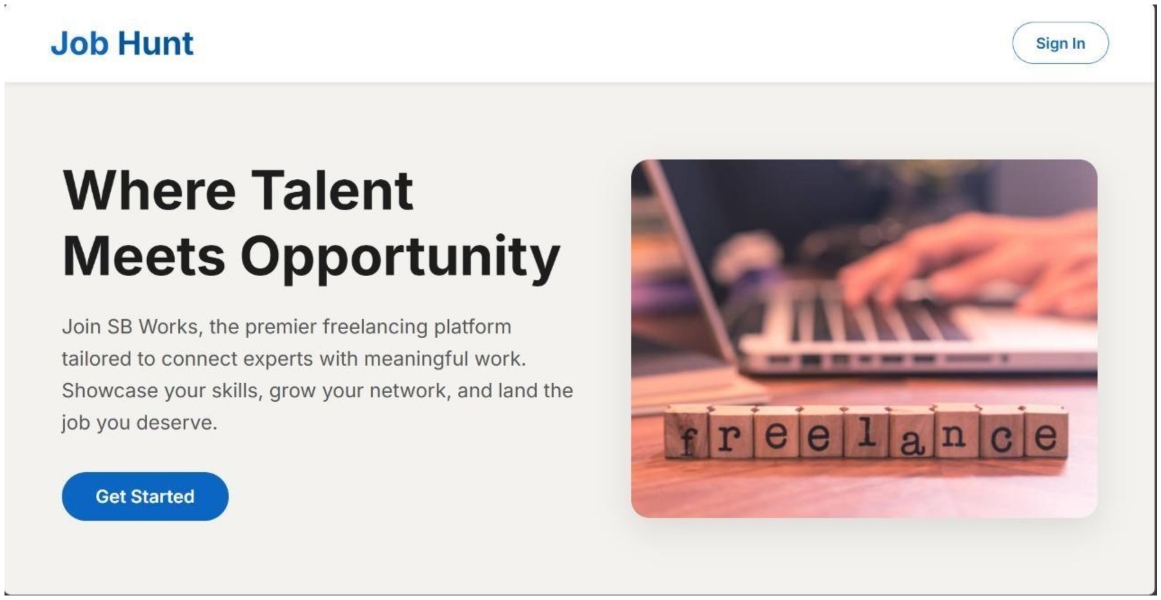
- - Fully responsive layout using Flexbox & Grid.
- - Modern UI with clean components.
- - Role-specific dashboards.
- - Real-time chat UI.

10. Testing

- - **Manual Testing**: Manual testing was conducted thoroughly across all critical functionalities of the application. These include user registration and login, product browsing, cart operations, checkout, order placement, and admin functionalities like product management. Each flow was tested for both valid and invalid input data to ensure the UI handles errors gracefully and the backend responds appropriately. - API Testing: Verified endpoints via Postman.
- - Validation Tests: All inputs tested for format and constraint.
- - Authorization Testing: Role and token validations tested.

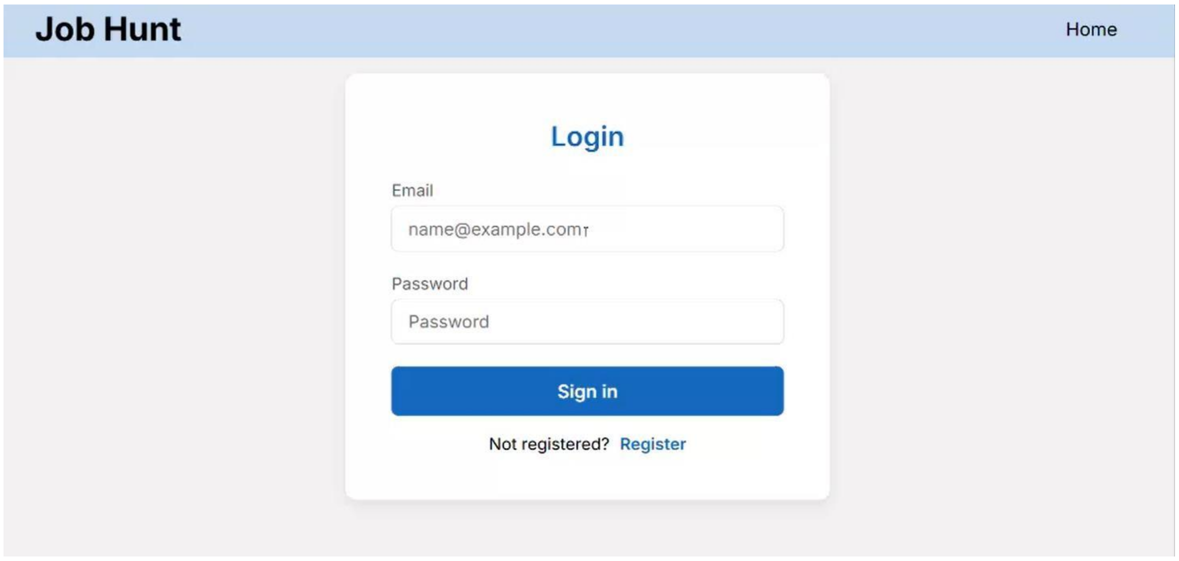
  - Responsiveness: Manual testing was conducted across various screen sizes and devices using browser dev tools to ensure that the UI maintains consistency and functionality across resolutions.

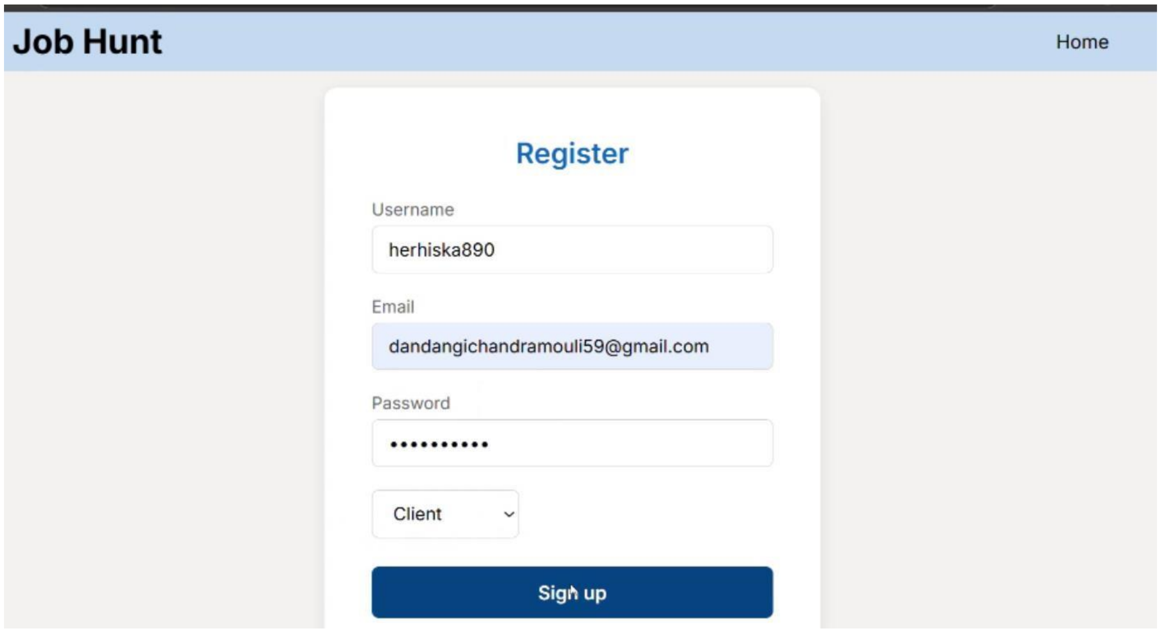
11. Screenshots or Demo



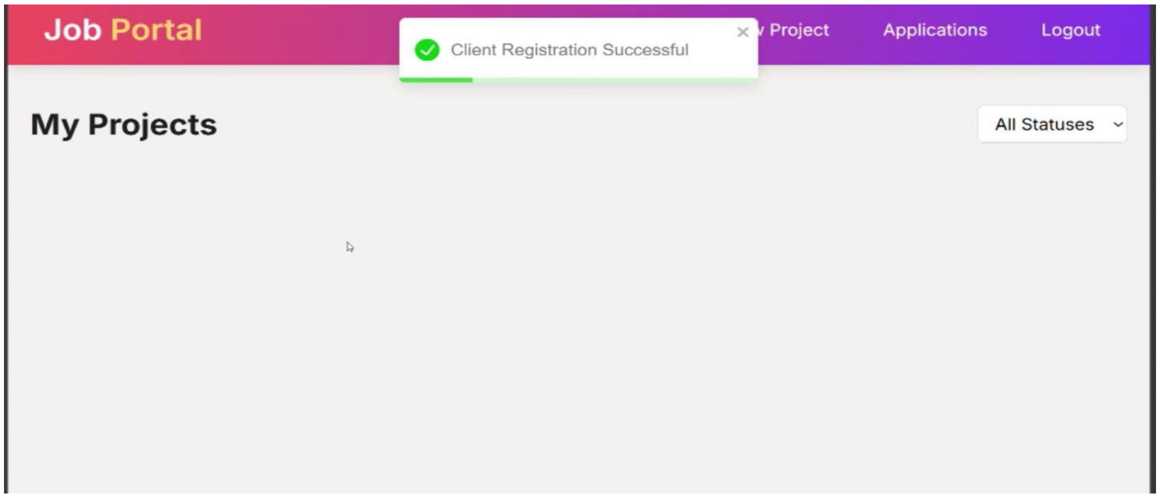
Where Talent Meets Opportunity

Join SB Works, the premier freelancing platform tailored to connect experts with meaningful work. Showcase your skills, grow your network, and land the job you deserve.





Username: herhiska890
Email: dandangichandramouli59@gmail.com
Password: ••••••••••
Client

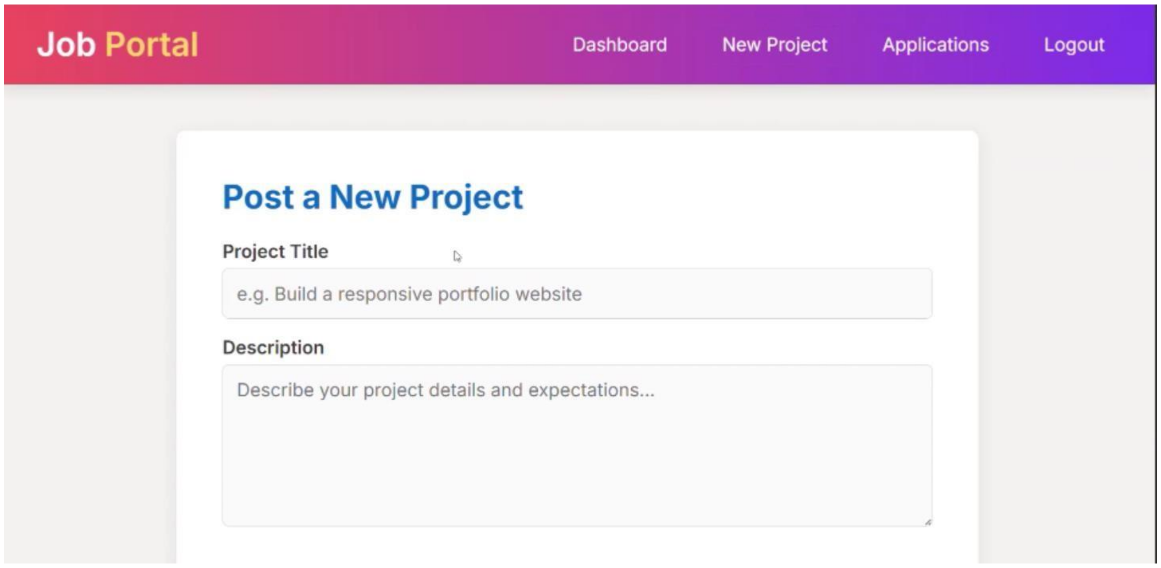

Job Portal

Client Registration Successful

Project Applications Logout

My Projects

All Statuses


Job Portal

Dashboard New Project Applications Logout

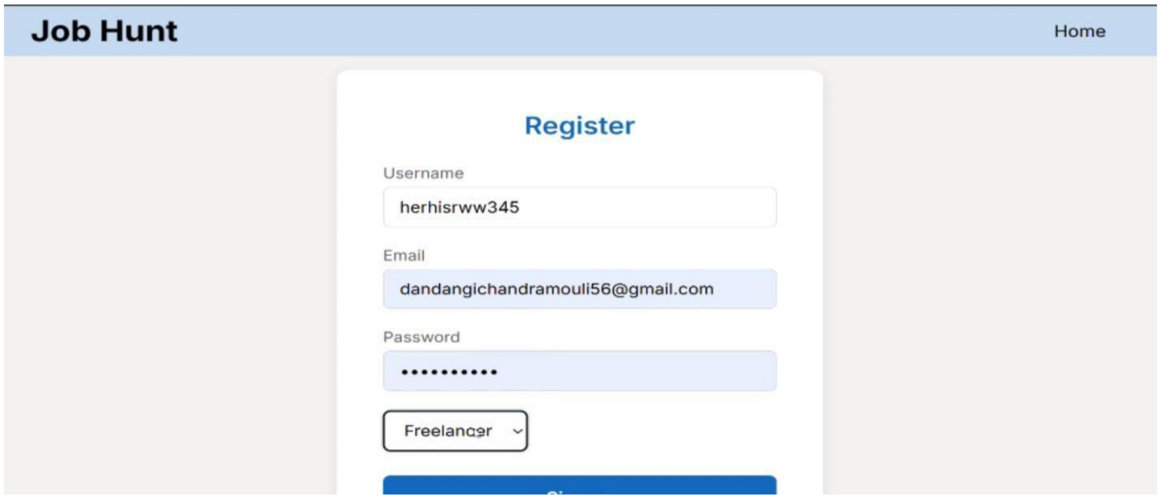
Post a New Project

Project Title

e.g. Build a responsive portfolio website

Description

Describe your project details and expectations...


Job Hunt

Home

Register

Username

herhisrww345

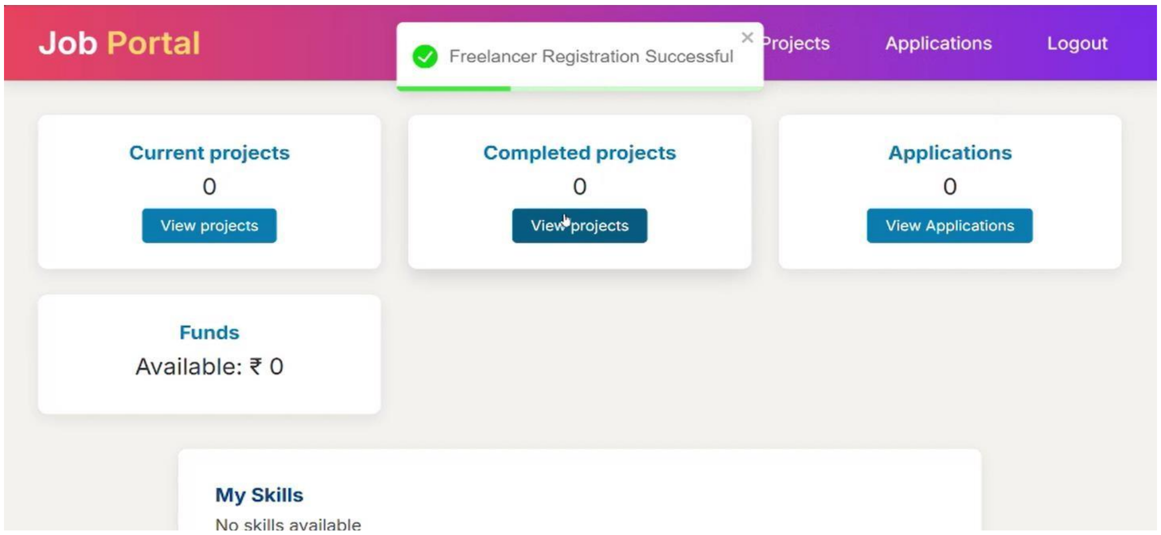
Email

dandangichandramouli56@gmail.com

Password

•••••••••••

Freelancer



12. Known Issues

- Some responsiveness issues may occur on older devices
- Lack of full test coverage for edge cases
- Product filtering may lag with very large data sets.

13. Future Enhancements

- - AI-Based Skill Match Suggestions
- - Payment Gateway Integration
- - Review and Rating System
- - Notification Enhancements
- - Dark Mode and Theme Switching