Freelancing Application MERN

TalentTrove

**INTRODUCTION**

TalentTrove is more than just a freelance marketplace—it's a dynamic digital workspace designed to connect visionary project owners with top-tier independent professionals. Whether you're looking to bring your project to life or showcase your expertise, our intuitive, user-friendly dashboard makes the entire process seamless**.**

Team member : Aman Kumar

**Project Overview**

**Purpose**

TalentTrove is a freelance marketplace designed to connect professionals with potential clients. It provides a seamless platform for freelancers to showcase their skills, bid on projects, and collaborate efficiently with clients. The goal is to create a secure, user-friendly environment that simplifies project management, enhances communication, and ensures smooth transactions.

Features

* Intuitive Dashboard – Manage projects, proposals, and communications in one place.
* Secure Bidding System – Submit proposals with integrated portfolio attachments.
* Real-Time Chat – Communicate with clients for seamless collaboration**.**
* Portfolio Management – Showcase past projects and achievements.
* Authentication & Security – Secure login with JWT and OAuth authentication.
* Efficient Project Delivery – Submit work, track progress, and receive feedback.
* Optimized Database – Uses MongoDB for efficient data storage and retrieval.
* Responsive UI – Built with React.js and Tailwind CSS for an enhanced user experience.

**Scenario-Based Case Study: Emma’s Journey on TalentTrove**

**A Fresh Start in Freelancing**  
Emma, a recent graphic design graduate, is eager to showcase her creativity and build a strong freelance portfolio. While exploring online freelancing opportunities, she discovers TalentTrove, an innovative freelance marketplace designed for professionals like her.

**Finding the Perfect Project**  
Impressed by TalentTrove’s intuitive, user-friendly dashboard, Emma browses various project categories. She comes across an exciting opportunity—a local bakery, "Sugar Rush," is looking for a logo redesign. The detailed project description highlights the bakery’s brand identity and target audience, giving Emma a clear direction for her creative approach.

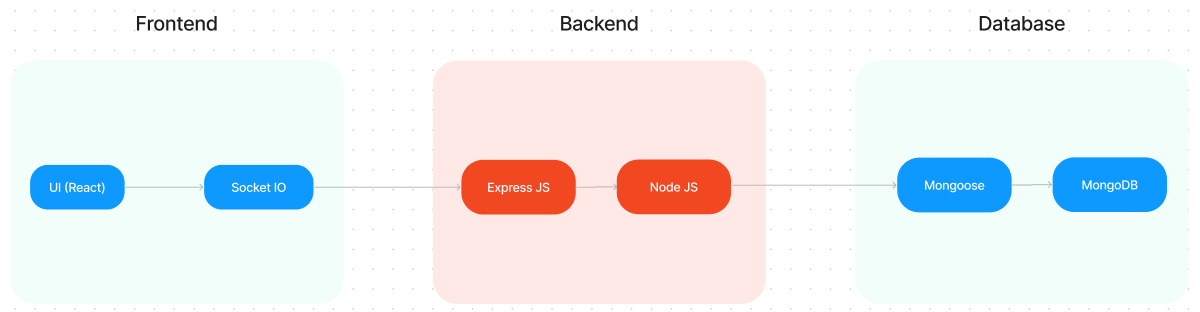
**Bidding with Confidence**  
Confident in her design skills, Emma thoroughly reviews the bakery’s existing branding materials through TalentTrove’s integrated portfolio system. She submits a compelling proposal, showcasing her relevant experience and attaching sample designs stored securely on the platform.

**Seamless Communication & Collaboration**  
Impressed by Emma’s portfolio and competitive pricing, "Sugar Rush" selects her proposal. Thanks to TalentTrove’s interactive chat system, Emma and the client discuss project specifics, exchange ideas, and refine the design direction—all within the platform.

**Delivery & Feedback**  
Once Emma finalizes the logo, she submits it directly through TalentTrove. The bakery reviews her work, provides feedback, and requests minor revisions if needed. The platform ensures a smooth collaboration process, allowing both parties to work towards the perfect final design.

**Building a Thriving Career**  
With "Sugar Rush" thrilled by the final design, Emma receives a glowing review, boosting her reputation on TalentTrove. Encouraged by this success, she continues bidding on new projects, steadily growing her portfolio and gaining more clients. With each successful project, Emma moves closer to building a thriving freelance career on TalentTrove—a platform where skills shine and opportunities flourish.

**TECHNICAL ARCHITECTURE**



The technical architecture of **TalentTrove**, an innovative freelance marketplace, follows a **client-server model**, where the frontend serves as the client and the backend acts as the server.

**Frontend Architecture**

The **frontend** is responsible for delivering an intuitive **user-centric dashboard** that ensures a seamless user experience. It integrates the **Axios** library to facilitate smooth communication with the backend through **RESTful APIs**, enabling real-time interactions.

To enhance UI/UX, the frontend leverages **Tailwind CSS** for modern styling and **Material UI** for interactive components, creating a visually appealing and responsive interface.

**Backend Architecture**

On the **backend**, **TalentTrove** utilizes the **Express.js** framework to manage server-side logic and real-time interactions. **Express.js** provides a robust and scalable foundation for handling client requests, processing responses, and ensuring smooth data flow between users and the platform.

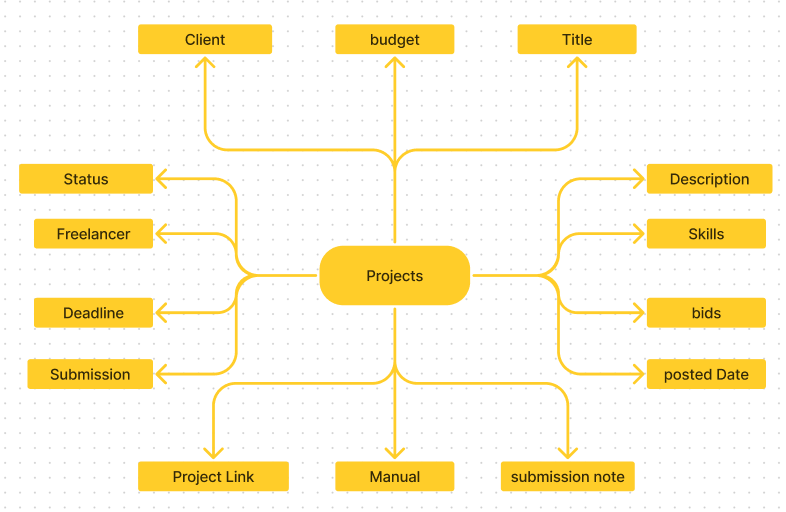
With this efficient **client-server** setup, **TalentTrove** delivers a secure, fast, and engaging freelancing experience for both project owners and independent professionals

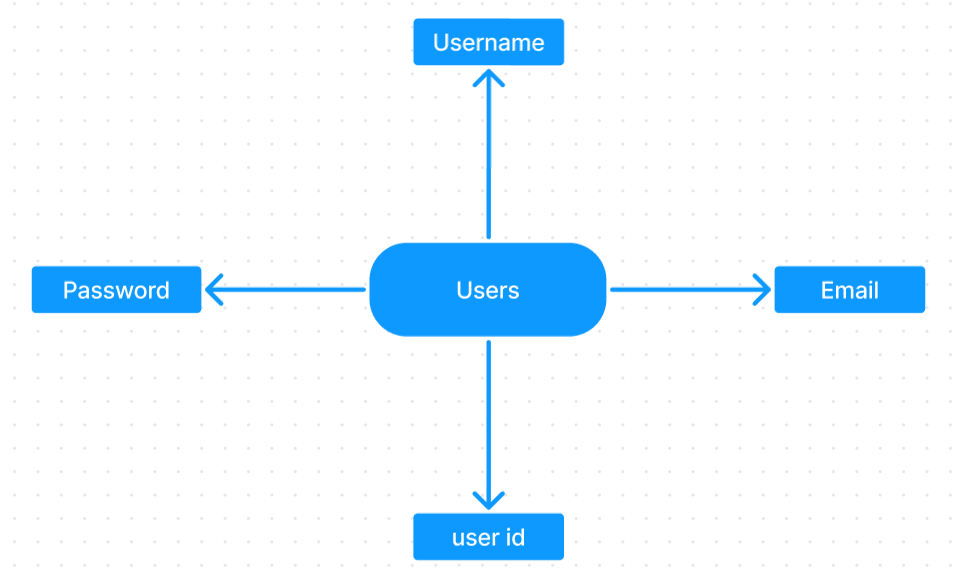
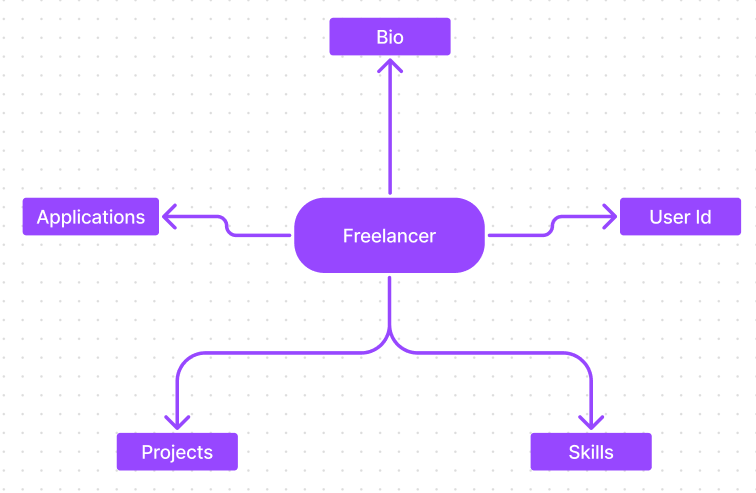
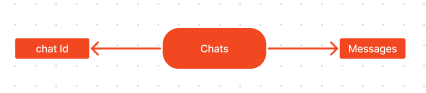
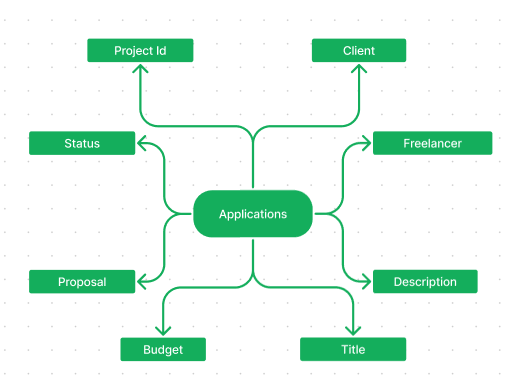
For efficient data storage and retrieval, **TalentTrove**, an innovative freelance marketplace, relies on **MongoDB**. This NoSQL database provides a **scalable and high-performance** solution for managing various types of data, including **user profiles, project details, bids, transactions, and communication logs**. MongoDB’s flexible schema structure ensures quick and reliable access to data, enabling a seamless freelancing experience.

The integration of **MongoDB, Express.js, and a dynamic frontend** forms the backbone of **TalentTrove’s** robust technical architecture. Together, these components enable:

* **Real-time communication** between clients and freelancers.
* **Efficient data exchange** for project bidding, submissions, and payments.
* **Seamless integration** of user profiles, project listings, and transactions.

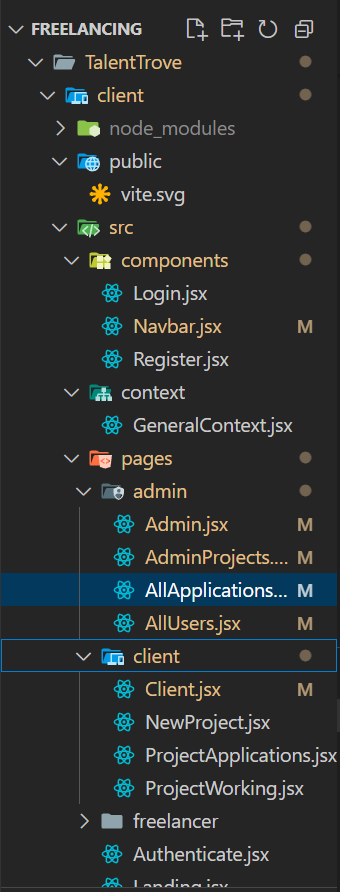
This architecture ensures a **smooth, secure, and immersive** experience for users, making **TalentTrove** a go-to platform for freelancers and project owners alike.

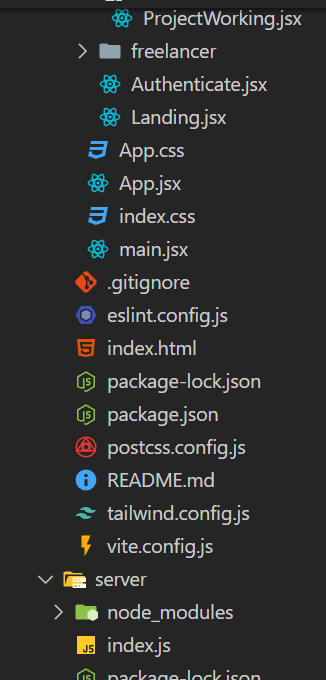
**ER DIAGRAM **

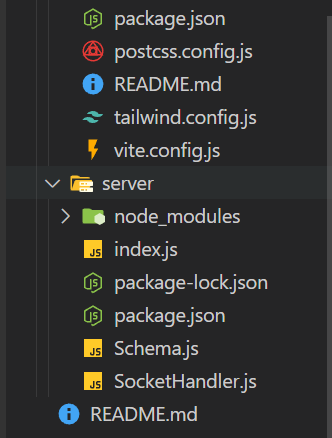
****

**TalentTrove**, an innovative freelance marketplace, seamlessly connects project owners with skilled independent professionals through a user-friendly platform. Clients can easily post tasks and assignments with detailed requirements, browse freelancer profiles, and select the perfect match for their projects. On the other hand, freelancers can submit compelling proposals, collaborate with project owners through a secure chat system, and deliver work efficiently while ensuring secure and timely payments. A dedicated platform management team ensures quality, security, and smooth communication, making **TalentTrove** the ultimate freelancing hub. Join **TalentTrove** today and experience a smarter, more efficient way to freelance**.**

**PROJECT STRUCTURE**

****

****

****

**TalentTrove**, an innovative freelance marketplace, leverages **React.js** to build a seamless, user-centric dashboard. The client-side is designed with reusable components for profiles, tasks, assignments, and chat, assembled into intuitive pages such as project browsing and freelancer profiles. Shared data, including user information and search filters, is efficiently managed using **React Context** for a smooth experience.

On the **server-side**, **Node.js** processes API requests related to user management, project actions, and real-time communication. Mongoose models ensure structured and efficient interaction with the **MongoDB** database, maintaining data integrity and scalability. This well-integrated architecture ensures a fast, secure, and immersive freelancingexperience on **TalentTrove.**

**PRE-REQUISTIC:**

Here are the key prerequisites for developing a full-stack application using Express Js, MongoDB, React.js:

**Node.js and npm:** Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server-side. It provides a scalable and efficient platform for building network applications. Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side

**Express.js:**Express.js is a fast and minimalist web application framework for Node.js. It simplifies the process of creating robust APIs and web applications, offering features like routing, middleware support, and modular architecture. Install Express.js, a web application framework for Node.js, which handles server-side routing, middleware, and API development.

**MongoDB:** MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-like format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data. Set up a MongoDB database to store your application's data.

**React.js:** React.js is a popular JavaScript library for building user user-centric dashboards. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications. Install React.js, a JavaScript library for building user user-centric dashboards.

**HTML, CSS, and JavaScript:**Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

**Database Connectivity:** Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Express Js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations

**Front-end Framework:** Utilize React Js to build the user-facing part of the application, including entering booking room, status of the booking, and user user-centric dashboards for the platform management team dashboard. For making better UI we have also used some libraries like material UI and bootstrap.

**Version Control:** Use Git for version control, enabling partnership and teamwork and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

**Development Environment:** Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

•Install the required dependencies by running the following commands:

cd client

npm install

../cd server

npm install

**Start the Development Server:**

• To start the development server, execute the following command:

npm start

• The SB Works, an innovative freelance marketplace app will be accessible at

You have successfully installed and set up the SB Works, an innovative freelance marketplace application on your local machine. You can now proceed with further customization, development, and testing as needed.

**API Documentation**

**Authentication API**

**Register a User**

**Endpoint:**/register  
**Method:**POST  
**Request Body:**

{  
  "username": "JohnDoe",  
  "email": "johndoe@example.com",  
  "password": "securepassword",  
  "usertype": "freelancer"  
}

**Response:**

{  
  "\_id": "user12345",  
  "username": "JohnDoe",  
  "email": "johndoe@example.com",  
  "usertype": "freelancer"  
}

**Freelancer API**

**Fetch Freelancer Details**

**Endpoint:**/fetch-freelancer/:id  
**Method:**GET  
**Response:**

{  
  "userId": "user12345",  
  "skills": ["JavaScript", "React"],  
  "description": "Full-stack developer"  
}

**Update Freelancer Profile**

**Endpoint:**/update-freelancer  
**Method:**POST  
**Request Body:**

{  
  "freelancerId": "freelancer123",  
  "updateSkills": ["Node.js", "MongoDB"],  
  "description": "MERN Stack Developer"  
}

**User Login**

**Endpoint:**/login  
**Method:**POST  
**Request Body:**

{  
  "email": "johndoe@example.com",  
  "password": "securepassword"  
}

**Response:**

{  
  "\_id": "user12345",  
  "username": "JohnDoe",  
  "email": "johndoe@example.com",  
  "usertype": "freelancer"  
}

**Authentication & Authorization in the Project**

This project uses **email and password-based authentication** with **bcrypt** for password hashing. The authentication process includes **user registration and login**, but it does not seem to implement token-based authentication (e.g., JWT) or session management.

**User Registration (/register)**

* The user provides a **username, email, password, and user type** (either client or freelancer).
* The password is **hashed** using bcrypt before storing it in the database.
* If the user is a **freelancer**, an additional Freelancer profile is created.

**User Login (/login)**

* The user submits **email and password**.
* The system checks if the email exists in the database.
* The password is **compared** with the hashed password using bcrypt.compare().
* If credentials are valid, the user is logged in.

**Application flow:**

**Freelancer Responsibilities:**

• Project Submission: Freelancers are responsible for submitting completed and high-quality work for the assigned tasks and assignments through the platform.

• Compliance: Ensure that the submitted work adheres to client requirements, industry standards, and any specific guidelines outlined by the platform.

• Effective Communication: Actively engage in interactive dialogue system with project owners, promptly responding to messages, asking clarifying questions, and providing updates on the project progress.

• Time Management: Manage time effectively to meet project deadlines and deliver work in a timely manner.

• Professionalism: Conduct oneself professionally by maintaining a respectful and cooperative attitude with project owners and fellow independent professionals.

• Quality Assurance: Deliver work that is accurate, well-executed, and free from errors to maintain client satisfaction.

**Client Responsibilities:**

• Clear Project Description: Provide a detailed and comprehensive project description, including deliverables, desired outcomes, and any specific requirements.

• Timely Communication: Respond promptly to freelancer inquiries, providing necessary information and feedback in a timely manner.

• Payment Obligations: Fulfill the agreed-upon payment terms promptly and fairly upon satisfactory completion of the project.

• Feedback and Evaluation: Provide constructive feedback and evaluate the freelancer's performance, helping them improve and providing valuable insights.

**Admin Responsibilities:**

Data Oversight: As a platform management team, one of your key responsibilities is to monitor and ensure the integrity and data integrity and safe transactions of all data on the platform

Policy Enforcement: Admins play a crucial role in enforcing platform policies, guidelines, and ethical standards.

Conflict Resolution: In the event of disputes or issues within the community, it is the platform management team's responsibility to address them promptly and impartially

User Support and Communication: Admins should provide support and guidance to users on the platform

Platform Maintenance and Improvement: Admins are responsible for the overall maintenance and improvement of the research platform.

**Project Flow:**

**Milestone 1: Project setup and configuration.**

Folder setup:

Now, firstly create the folders for frontend and backend to write the respective code and install the essential libraries.

Client folders.

Server folders

Installation of required tools:

1. Open the frontend folder to install necessary tools

For frontend, we use:

React

Tailwind CSS

Material UI

Axios

react-bootstrap

2. Open the backend folder to install necessary tools

For backend, we use:

Express Js

Node JS

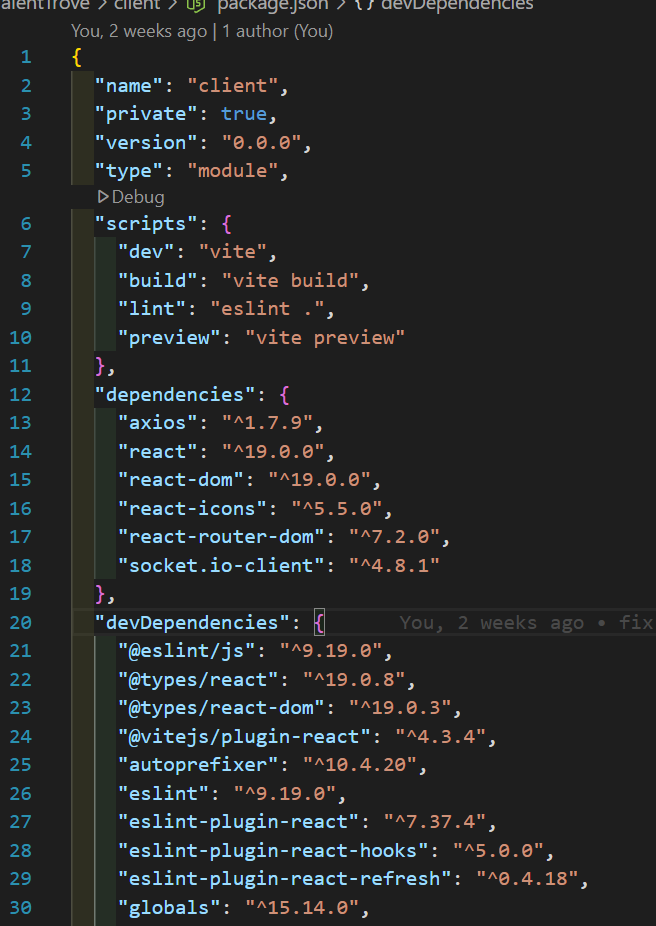
MongoDB

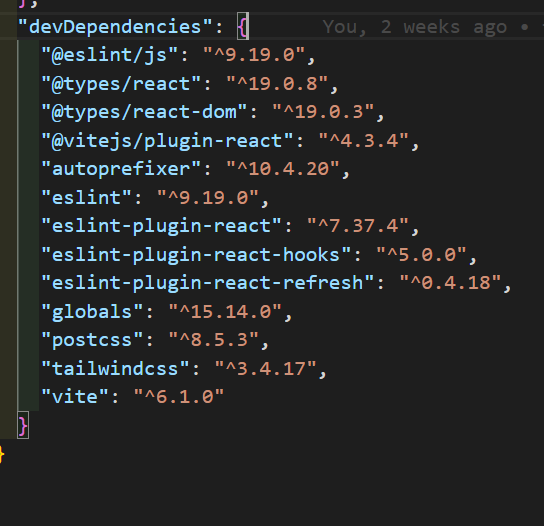
Mongoose

Cors

Bcrypt

After the installation of all the libraries, the package.json files for the frontend looks like the one mentioned below.





After the installation of all the libraries, the package.json files for the backend looks like the one mentioned below.



**Milestone 2: Backend Development**

**1. Project Setup:** Create a project directory and initialize it using npm init. Install required dependencies like Express.js, Mongoose, body-parser, and cors.

**2. Database Configuration:** Set up a MongoDB database (locally or using a cloud service like MongoDB Atlas).

Create collections for:

Users (storing user information, account type)

Projects (project details, budget, skills required)

Applications (freelancer proposals, rate, portfolio link)

Chat (interactive dialogue system history for each project)

Freelancer (extended user details with skills, experience, ratings)

**3. Express.js Server:** Create an Express.js server to handle HTTP requests and API endpoints. Configure body-parser to parse request bodies and cors for cross-origin requests.

**4. API Routes:** Define separate route files for user management, project listing, application handling, chat functionality, and freelancer profiles.

Implement route handlers using Express.js to interact with the database:

User routes: registration, login, profile management.

Project routes: project creation, listing, details retrieval.

Application routes: submit proposals, view applications.

Chat routes: send and receive messages within tasks and assignments.

Freelancer routes: view and update profiles, showcase skills.

**5. Data Models:** Define Mongoose schemas for each data entity:

User schema

Project schema

Application schema

Chat schema

Freelancer schema (extends User schema with skills, experience)

Create Mongoose models to interact with the MongoDB database.

Implement CRUD operations for each model to manage data.

**6. User Authentication:** Implement user authentication using JWT or session-based methods.Create routes and middleware for user registration, login, and logout.Use authentication middleware to protect routes requiring user authorization (e.g., applying for tasks and assignments).

**7. Project Management:** Allow project owners to post tasks and assignments with details and budget.Enable independent professionals to browse tasks and assignments, search by skills, and submit proposals.Implement a system for project owners to review applications and choose independent professionals.

**8. Secure Communication & Collaboration:** Integrate a secure chat system within tasks and assignments for interactive dialogue system between project owners and independent professionals.Allow file attachments and feedback exchange to facilitate partnership and teamwork.

**9. Admin Panel:** Implement a platform management team panel with functionalities like:

Managing users

Monitoring project updates and applications

Accessing transaction history

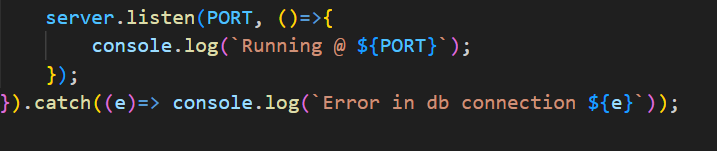
**Milestone 3: Database development**

Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas.

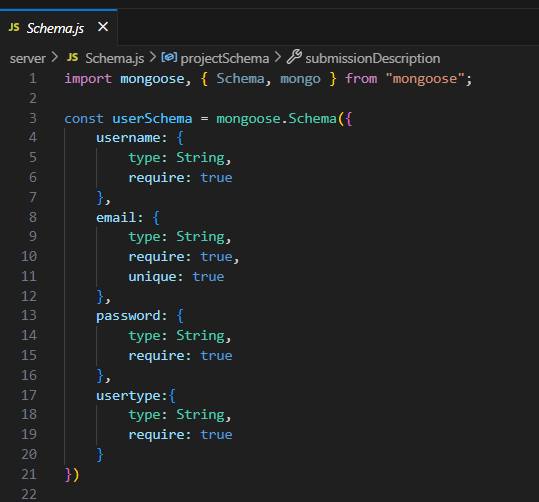
Create a database and define the necessary collections for users, freelancer, tasks and assignments, chats, and applications.

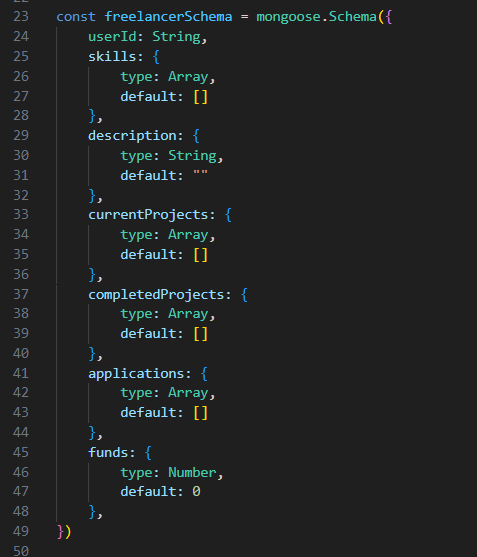
Connect the database to the server with the code provided below.

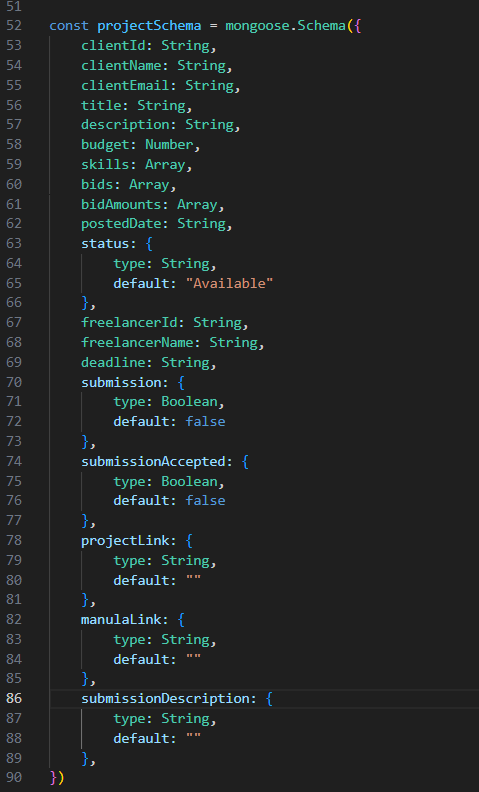
****

****

The Schemas for the database are given below.

****

****

****

****

**Milestone 4: Frontend development**

**1. Setting the Stage:** The TalentTrove, an innovative freelance marketplace frontend thrives on React.js. To get started, we'll:

Create the initial React application structure.

Install essential libraries for enhanced functionality.

Organize project files for a smooth development experience.

This solid foundation ensures an efficient workflow as we bring the TalentTrove, an innovative freelance marketplace user-centric dashboard to life.

**2. Crafting the User Experience:** Next, we'll focus on the user user-centric dashboard (UI). This involves:

Designing reusable UI components like buttons, forms, and project cards.

Defining the layout and styling for a visually appealing and consistent user-centric dashboard.

Implementing navigation elements for intuitive movement between features.

These steps will create a user-friendly experience for both independent professionals and project owners.

**3. Bridging the Gap:** The final stage connects the visual user-centric dashboard with the backend data. We'll:

Integrate the frontend with TalentTrove, an innovative freelance marketplace API endpoints.

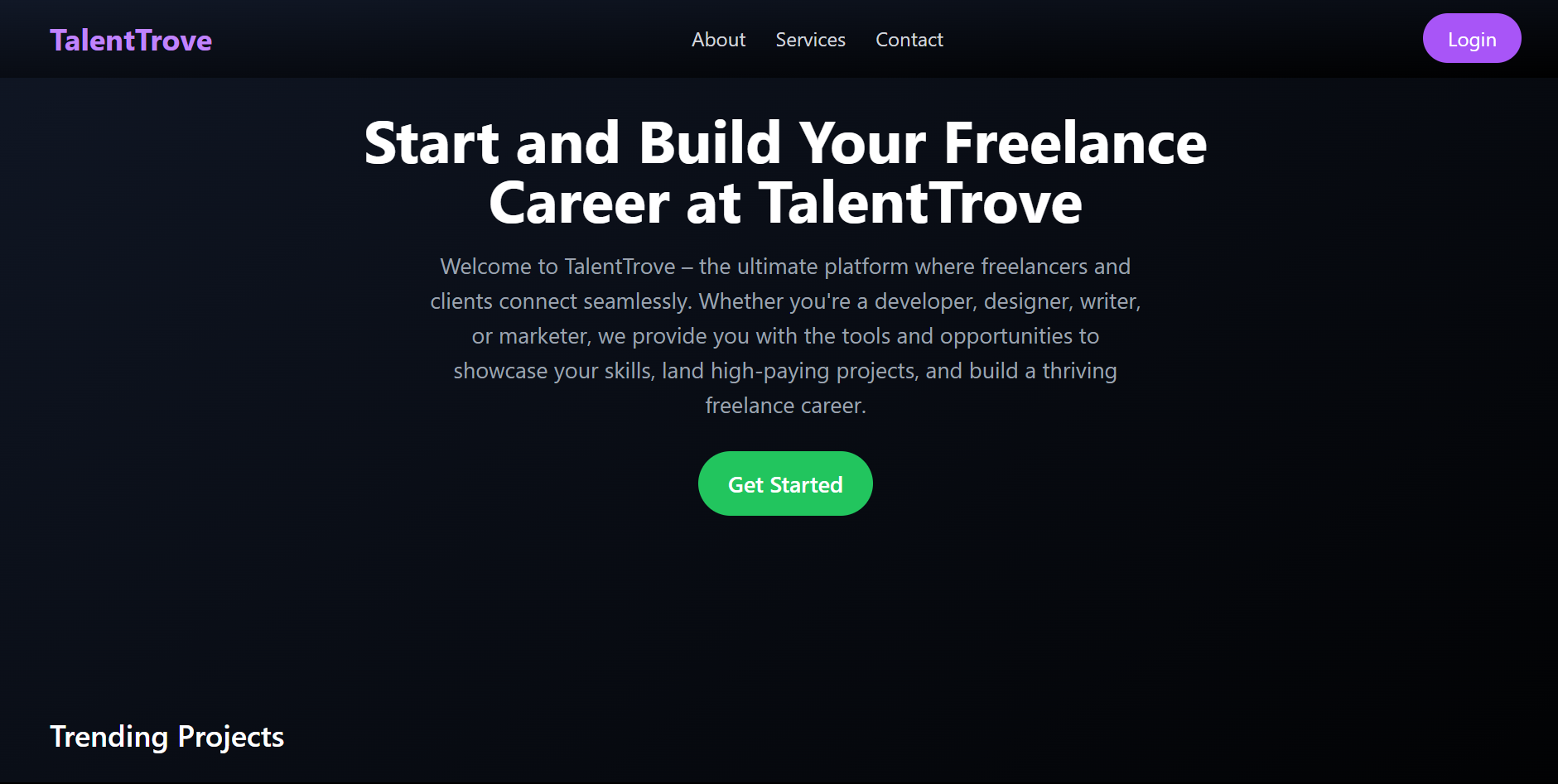
Implement data binding to ensure dynamic updates between user interactions and the displayed information.

This completes the frontend development, bringing the TalentTrove, an innovative freelance marketplace platform to life for users.

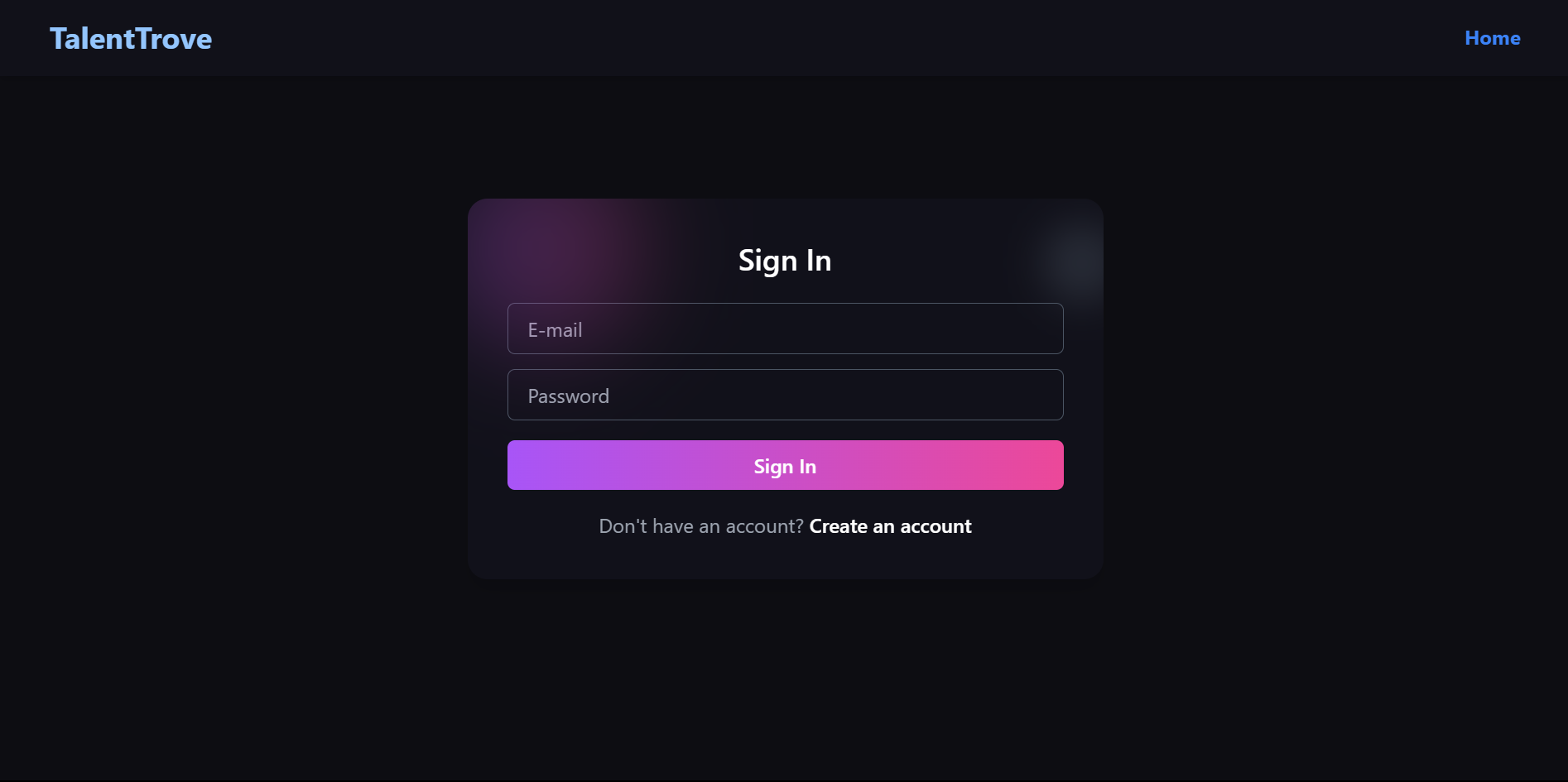
**User Interface**

On completing the development part, we then run the application one last time to verify all the functionalities and look for any bugs in it. The user user-centric dashboard of the application looks a bit like the images provided below.

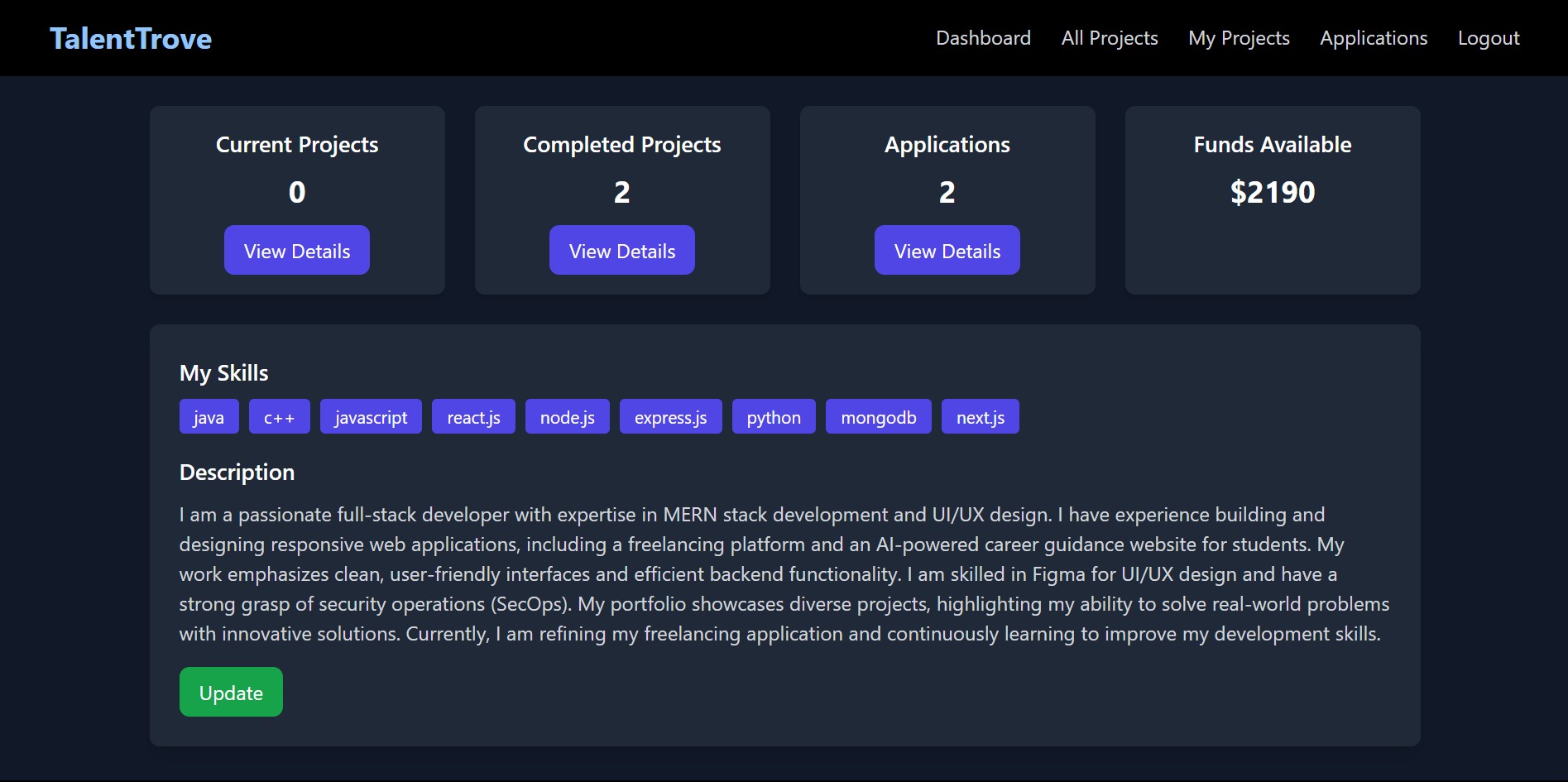
**Landing page:**



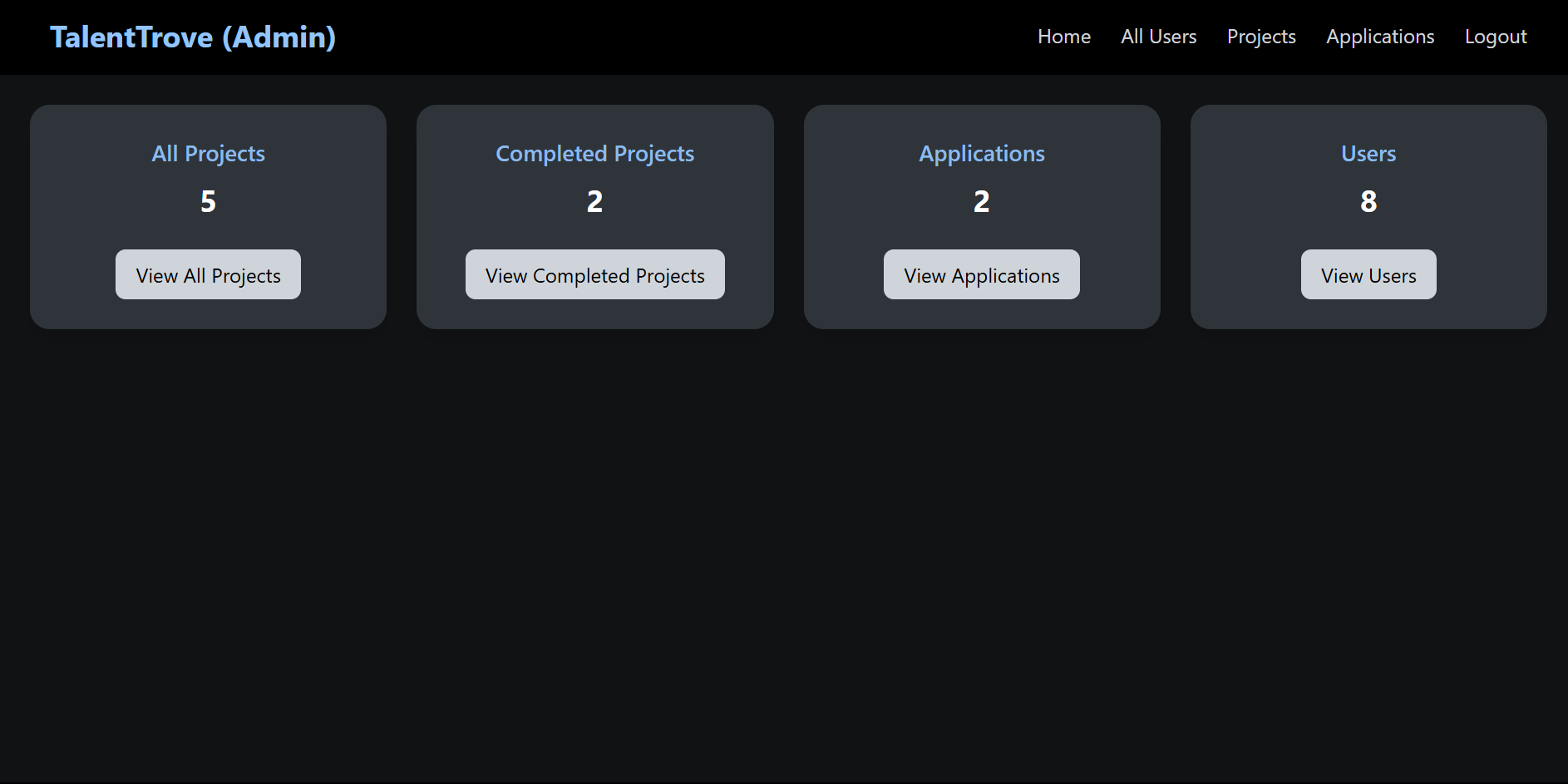
**Authentication:**



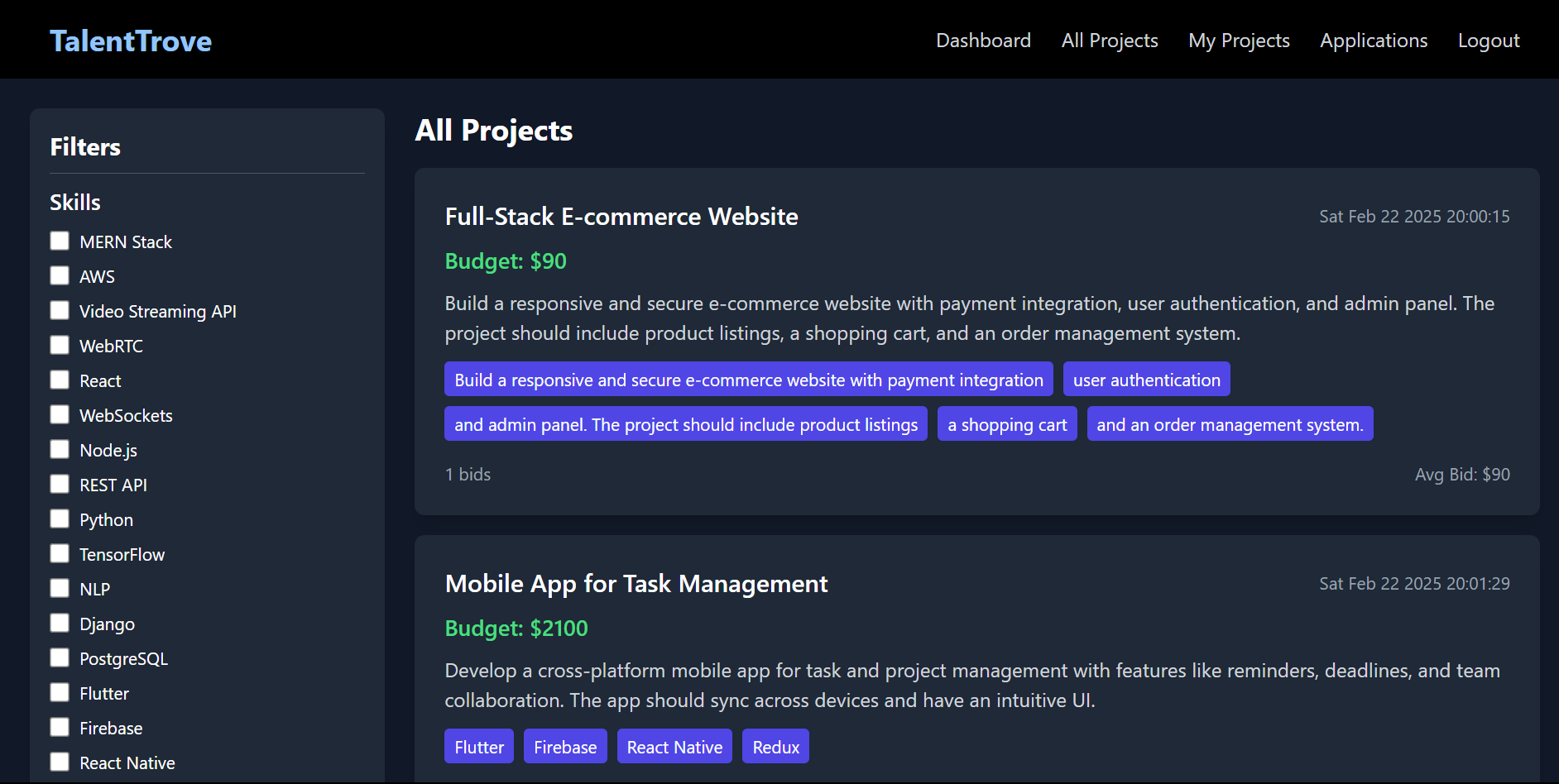
**Freelancer dashboard:**



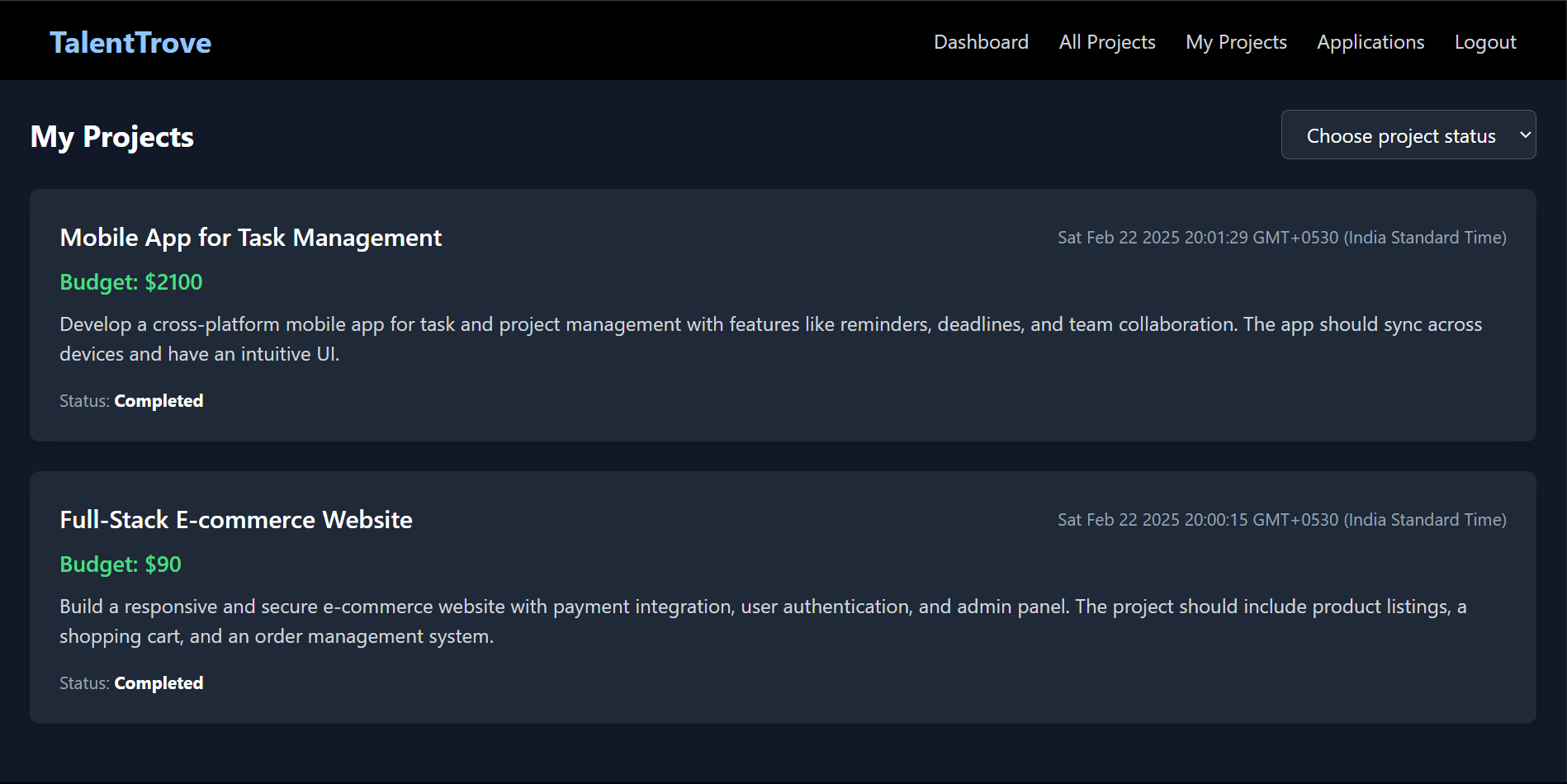
**Admin dashboard:**



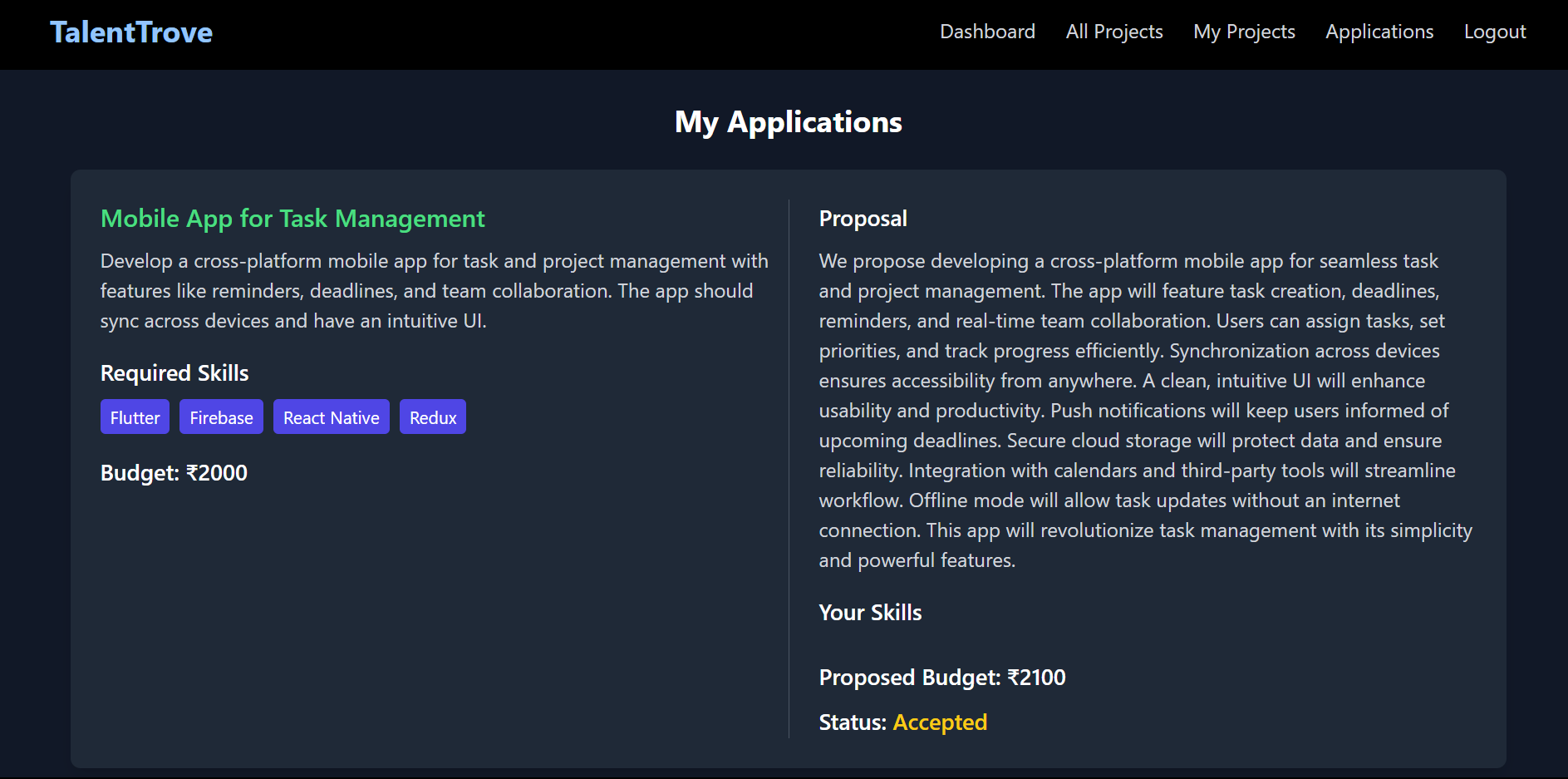
**All Projects:**



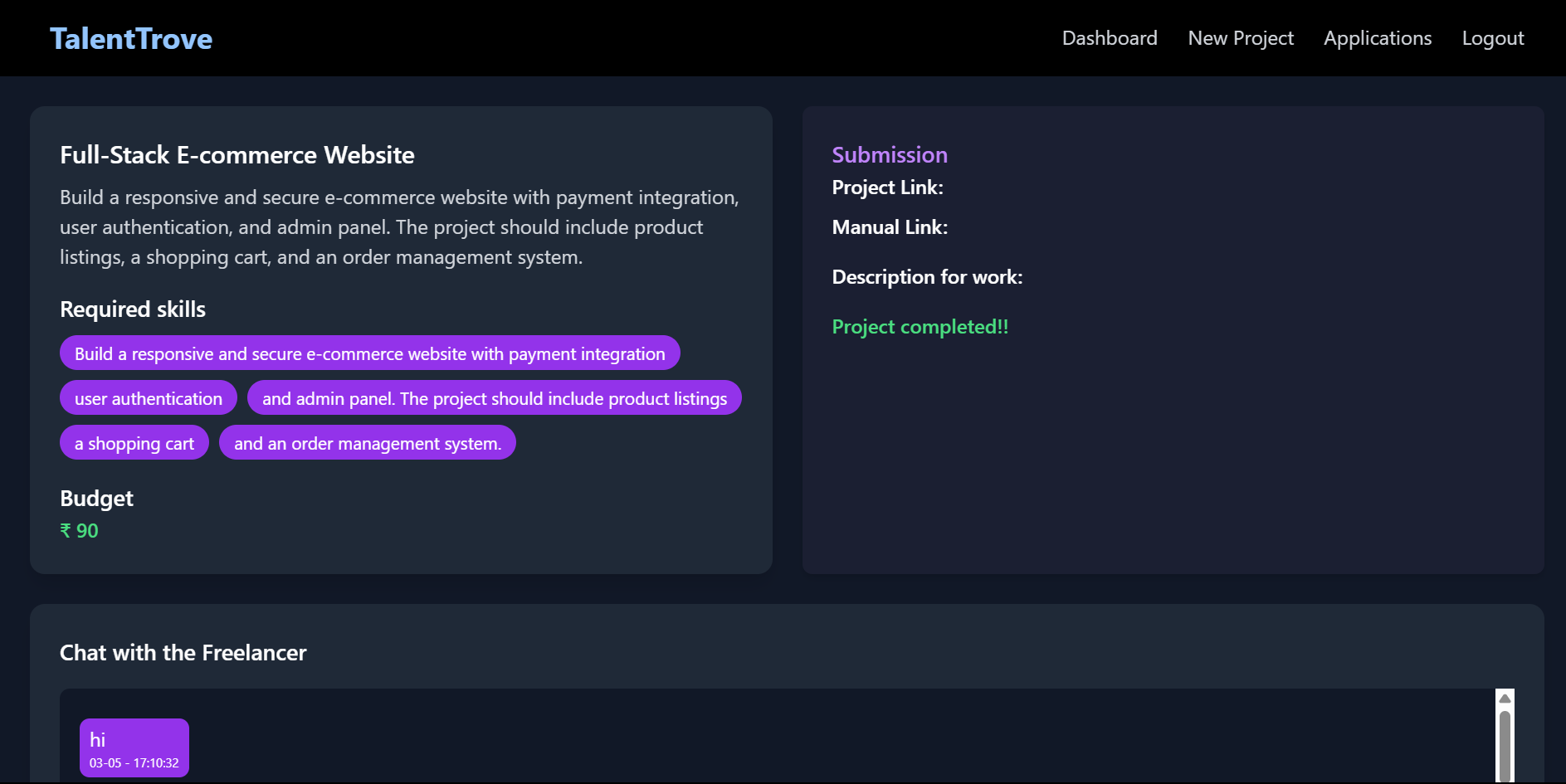
**Freelance Projects:**



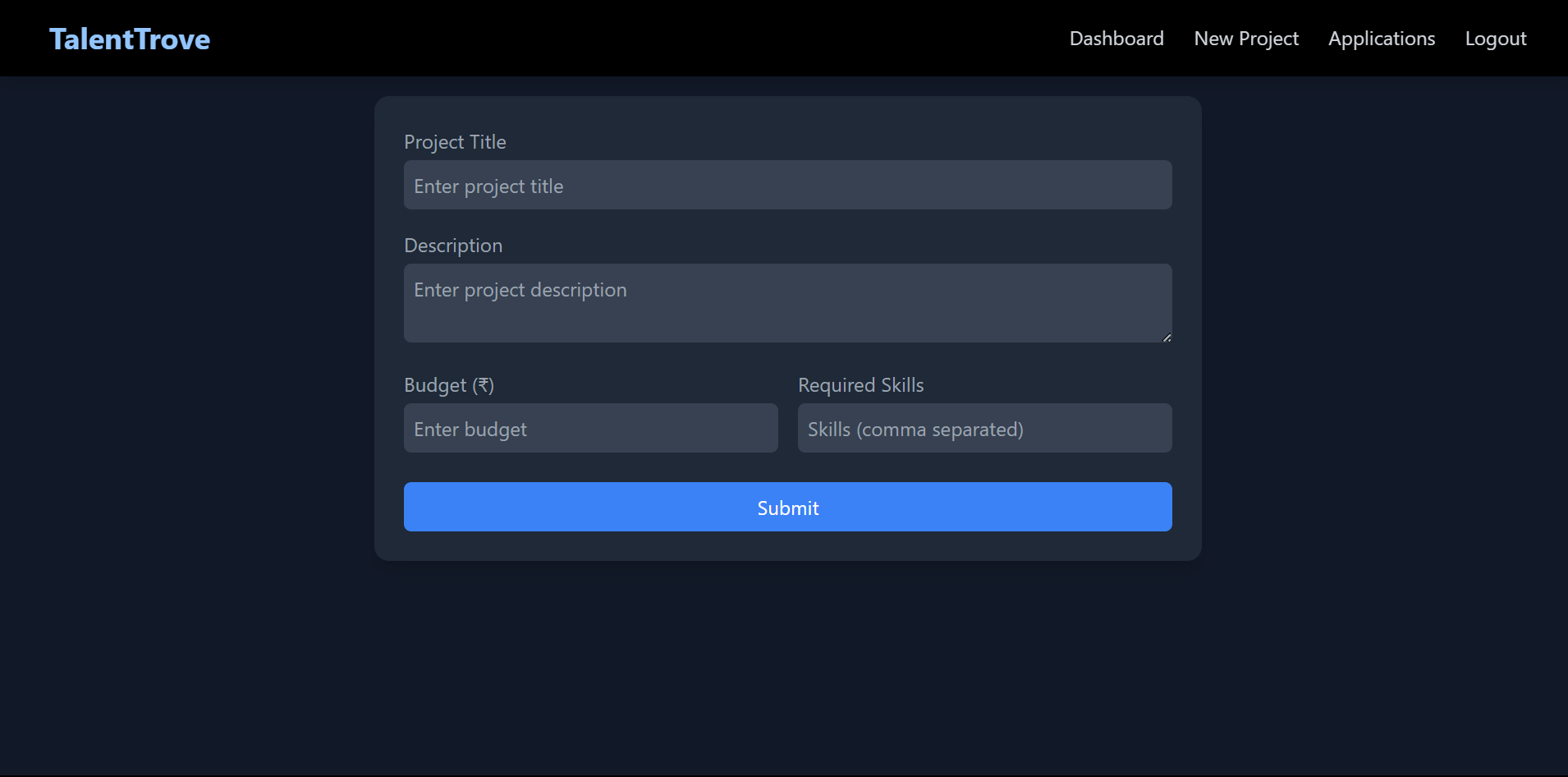
**Applications:**



**Project page:**



**New project:**



**Testing**

**1. Testing Strategy**

The testing approach for this project can be divided into three main types:

**Unit Testing** – Tests individual functions such as authentication (/register, /login) and CRUD     operations for projects and users.  
**Integration Testing** – Ensures that different modules (database, authentication, API endpoints)    work together as expected.  
**End-to-End (E2E) Testing** – Simulates real-world user interactions with the system, including login, project posting, bidding, and approvals.

**2. Tools Used**

* **Jest** – For unit and integration testing of backend functions.
* **Supertest** – For API testing, verifying that endpoints return expected responses.
* **Postman** – Used for manual testing of API requests and responses.
* **MongoDB In-Memory Server** – For running tests without modifying the actual database.
* **Socket.io Testing** – Verifies real-time communication in chat and notifications.

**3. Example Test Cases**

**User Registration & Login Tests** – Validate email format, password hashing, and incorrect credentials handling.  
  **Project CRUD Tests** – Ensure projects are created, retrieved, updated, and deleted properly.  
  **Bidding & Approval Workflow Tests** – Verify freelancer applications, project assignments, and submission approvals.  
  **Chat System Tests** – Check if messages are sent and received in real-time.

**Future Enhancements**

Here are some potential improvements and additional features for the project:

**1. Advanced Authentication & Security**

* Implement **JWT-based authentication** for better security and scalability.
* Enable **OAuth (Google, GitHub) login** for easier user onboarding.
* Add **role-based access control (RBAC)** to restrict user actions based on roles (Client, Freelancer, Admin).

**2. Enhanced User Experience (UI/UX)**

* Improve the **dashboard UI** with better analytics and real-time updates.
* Add **dark mode support** for user preference.
* Implement a **mobile-friendly responsive design**.

**3. AI-Powered Features**

* Integrate an **AI-powered job recommendation system** for freelancers based on skills.
* Implement **chatbots** for automated support and FAQs.
* Use **AI-driven proposal analysis** to help clients select the best freelancer.

**4. Payment & Escrow System**

* Implement **Stripe/PayPal integration** for secure payments.
* Introduce an **escrow system** to hold funds until project completion.

**5. Project Management & Tracking**

* Add **Kanban boards** for tracking project progress.
* Implement **milestone-based payments** for better project security.
* Allow **file sharing and document collaboration** within the platform.

**6. Performance & Scalability**

* Optimize **MongoDB queries** for faster data retrieval.
* Introduce **microservices architecture** to scale the backend.
* Implement **load balancing** for better traffic handling.