

FREELANCING APPLICATION USING MERN STACK

Project Overview

Lancers is a next-generation freelancing platform designed to revolutionize how clients connect with skilled freelancers. The platform enables clients to post a wide range of projects, from creative tasks to technical assignments, while empowering freelancers to bid on these projects based on their expertise. With a user-friendly interface and integrated tools for seamless communication and collaboration, Lancers ensures a transparent and efficient freelancing experience. The platform emphasizes reliability, quality, and secure interactions between clients and freelancers.

Use Cases

1. Clients Finding Freelancers

- Clients can post detailed project descriptions and review freelancer profiles, portfolios, and past work to select the best-suited freelancer for their needs.

2. Freelancers Bidding on Projects

- Freelancers browse through project listings, submit tailored proposals, and attach portfolio samples to showcase their suitability for specific tasks.

3. Project Collaboration and Management

- Integrated communication tools enable clients and freelancers to discuss project specifics, exchange feedback, and refine deliverables efficiently.

4. Project Submission and Feedback

- Freelancers submit completed projects directly through the platform, and clients review the work, provide feedback, and request revisions if necessary.

5. Building Reputations

- Successful projects and positive reviews enhance freelancer profiles, making it easier for them to secure future opportunities.

6. Secure Transactions

- Lancers ensures secure and transparent financial transactions between clients and freelancers.

Features

1. For Clients

- **Project Posting:** Easy-to-use interface for posting projects with detailed requirements.
- **Freelancer Selection:** Review freelancer profiles, portfolios, and client testimonials.
- **Integrated Communication:** Chat system for real-time collaboration and clarification.
- **Feedback System:** Provide reviews and feedback for completed projects.

2. For Freelancers

- **Project Search:** Browse diverse project categories tailored to different skill sets.
- **Proposal Submission:** Submit personalized proposals with attached portfolio samples.
- **Portfolio Management:** Securely store and showcase portfolio items within the platform.
- **Profile Building:** Gain traction and visibility through positive client reviews.

3. Platform Administration

- **Secure Transactions:** Monitor and facilitate secure financial exchanges.
- **Quality Assurance:** Vet freelancers to ensure reliability and maintain platform integrity.
- **Dispute Resolution:** Assist in resolving conflicts between clients and freelancers.

4. Technical Excellence

- **Frontend:** Intuitive and responsive interface leveraging Bootstrap and Material UI.
- **Backend:** Robust server-side management with Express.js for efficient request handling.
- **Database:** Scalable and fast data storage with MongoDB.
- **Real-Time Communication:** Streamlined interactions through RESTful APIs and Axios library integration.

Lancers aims to foster a collaborative freelancing ecosystem where efficiency, transparency, and quality drive the success of both clients and freelancers.

ARCHITECTURE

FRONT-END:

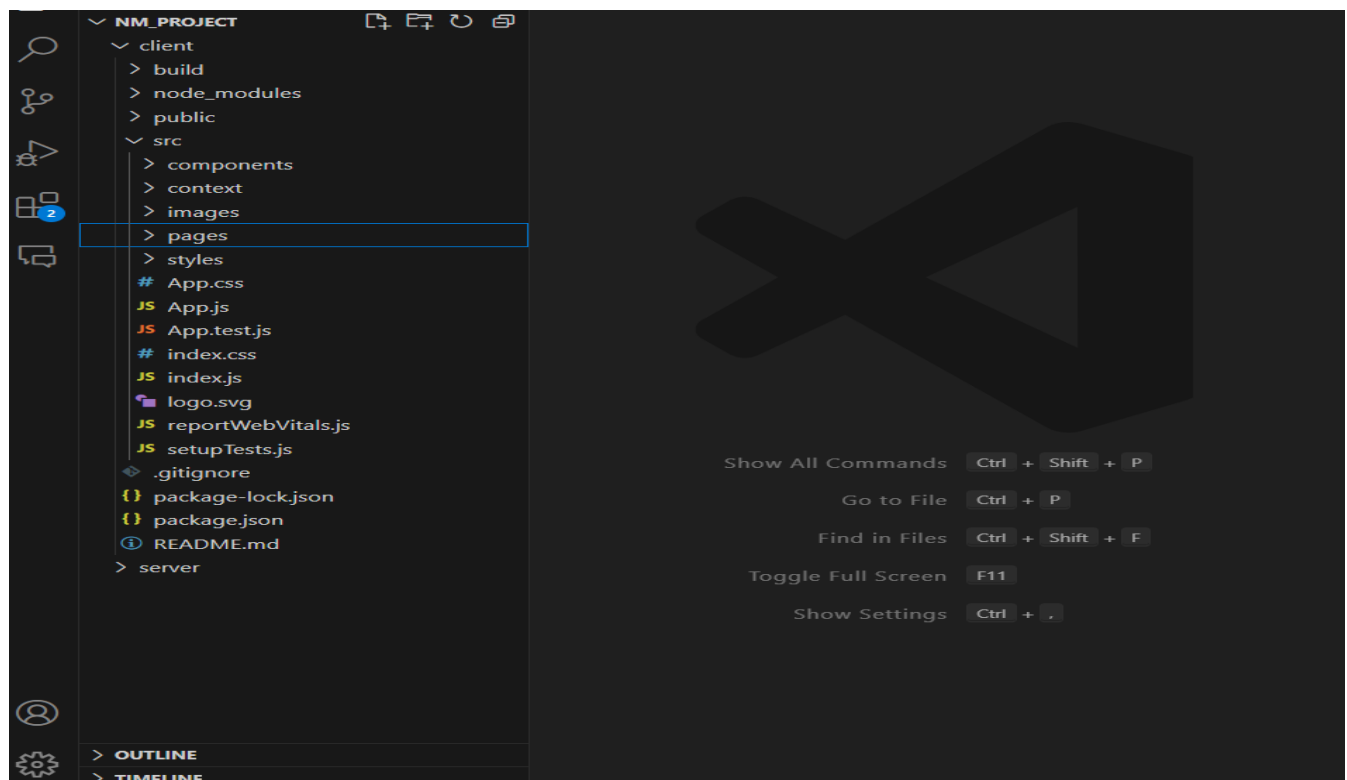
The frontend is responsible for the user-facing interface, allowing users to interact with the platform.

- **Frameworks & Libraries:**

- **React.js:** Used for building the user interface and managing state.
- **Bootstrap & Material UI:** Enhance the interface with responsive and visually appealing design elements.
- **Axios Library:** Facilitates communication with the backend via RESTful APIs.

- **Features:**

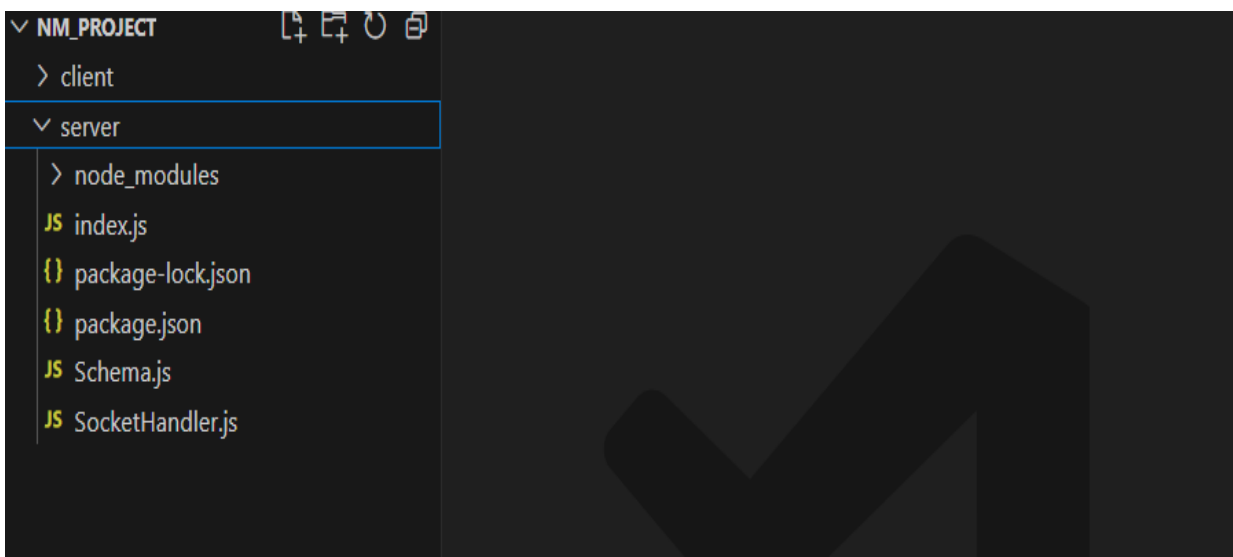
- User dashboards for clients and freelancers.
- Forms for project posting, bidding, and proposal submissions.
- Real-time chat system for communication between users.
- Notification system for project updates and alerts.



BACK-END:

The backend manages the server-side logic, API handling, and business rules of the platform.

- **Framework:**
 - **Express.js:** Handles HTTP requests, API routing, and user authentication.
- **Features:**
 - **User Management:** Authentication, authorization, and role-based access control (clients, freelancers, admins).
 - **Project Management:** CRUD operations for projects, proposals, and user profiles.
 - **Messaging System:** Real-time communication powered by WebSocket or similar technologies.
 - **Transaction Processing:** Secure handling of payments and invoices.



DATABASE:

The database stores and retrieves all platform data, ensuring scalability and reliability.

- **Database Type:**
 - **MongoDB:** A NoSQL database, chosen for its flexibility and ability to handle large datasets efficiently.
- **Stored Data:**
 - User profiles, portfolios, and credentials.
 - Project details, proposals, and bid histories.
 - Messages and collaboration records.
 - Transaction logs and payment histories.

CONNECTIONS (1)		Sort by Collection Name		View	
Search connections					
▼ NM_Project					
▼ Freelancing					
applications					
chats					
freelancers					
projects					
users					
▶ admin					
▶ config					
▶ local					

applications				
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	2	619.00 B	1	36.86 kB

chats				
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	3	582.00 B	1	36.86 kB

freelancers				
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	6	210.00 B	1	36.86 kB

projects				
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	2	725.00 B	1	36.86 kB

users				
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	6	180.00 B	2	73.73 kB

Node.js

- **Runtime Environment:** Node.js is used for the backend runtime, enabling the server-side logic to be written in JavaScript. It is efficient and well-suited for handling numerous concurrent requests, which is critical in a real-time system.

Express.js

- **Web Framework:** Express.js is a lightweight, flexible framework built on top of Node.js that simplifies routing, middleware integration, and request handling. It provides a set of tools to build RESTful APIs and manage HTTP requests with minimal boilerplate code.

MongoDB

- **NoSQL Database:** MongoDB is a NoSQL database, which allows the storage of structured and unstructured data (e.g., user details, appointment records, doctor profiles). Its flexibility and scalability make it ideal for applications that need to handle large volumes of data with varying structures.

DATABASE:

The freelancing application database is structured with multiple collections designed to manage users, freelancers, projects, applications, and chat interactions. Below is a detailed description of each collection, along with their fields and data types:

1. Users Collection

Stores basic information about all users, including clients and freelancers.

- **Collection Name:** users
- **Fields:**
 - username: String - Name of the user. (Required)
 - email: String - Email of the user. Must be unique. (Required)
 - password: String - Hashed password of the user. (Required)
 - usertype: String - Indicates if the user is a client or freelancer. (Required)

2. Freelancers Collection

Stores detailed information about freelancers, including their skills, projects, and financial details.

- **Collection Name:** freelancer
- **Fields:**
 - userId: String - Reference to the associated user in the users collection.
 - skills: Array - List of skills the freelancer possesses. Defaults to an empty array.
 - description: String - A brief description of the freelancer. Defaults to an empty string.
 - currentProjects: Array - List of projects the freelancer is currently working on. Defaults to an empty array.
 - completedProjects: Array - List of projects the freelancer has completed. Defaults to an empty array.
 - applications: Array - List of applications submitted by the freelancer. Defaults to an empty array.
 - funds: Number - Current funds earned by the freelancer. Defaults to 0.

3. Projects Collection

Contains all projects posted by clients, including details like budget, required skills, and current status.

- **Collection Name:** projects
- **Fields:**
 - clientId: String - ID of the client who posted the project.
 - clientName: String - Name of the client who posted the project.
 - clientEmail: String - Email of the client.
 - title: String - Title of the project.
 - description: String - Description of the project.
 - budget: Number - Budget allocated for the project.
 - skills: Array - List of required skills for the project.
 - bids: Array - List of freelancer IDs who have bid on the project.
 - bidAmounts: Array - List of bid amounts corresponding to the freelancers in bids.
 - postedDate: String - Date when the project was posted.
 - status: String - Current status of the project. Defaults to "Available".
 - freelancerId: String - ID of the freelancer assigned to the project.
 - freelancerName: String - Name of the assigned freelancer.
 - deadline: String - Deadline for project completion.
 - submission: Boolean - Indicates if the project has been submitted. Defaults to false.
 - submissionAccepted: Boolean - Indicates if the submission was accepted. Defaults to false.
 - projectLink: String - Link to the submitted project deliverable. Defaults to an empty string.
 - manulaLink: String - Placeholder for an additional link. Defaults to an empty string.
 - submissionDescription: String - Description of the project submission. Defaults to an empty string.

4. Applications Collection

Manages applications submitted by freelancers for projects.

- **Collection Name:** applications
- **Fields:**
 - projectId: String - ID of the associated project.
 - clientId: String - ID of the client who owns the project.
 - clientName: String - Name of the client who owns the project.
 - clientEmail: String - Email of the client.
 - freelancerId: String - ID of the freelancer applying for the project.
 - freelancerName: String - Name of the applying freelancer.
 - freelancerEmail: String - Email of the freelancer.
 - freelancerSkills: Array - List of skills of the applying freelancer.
 - title: String - Title of the associated project.
 - description: String - Description of the associated project.
 - budget: Number - Budget of the project.
 - requiredSkills: Array - List of skills required for the project.
 - proposal: String - Proposal submitted by the freelancer.
 - bidAmount: Number - Bid amount proposed by the freelancer.
 - estimatedTime: Number - Estimated time for project completion in days.
 - status: String - Status of the application. Defaults to "Pending".

5. Chats Collection

Manages chat conversations between clients and freelancers.

- **Collection Name:** chats
- **Fields:**
 - _id: String - Unique identifier for the chat (e.g., a combination of client and freelancer IDs). (Required)
 - messages: Array - List of messages exchanged in the chat.

Setup Instructions & Running the Application

Prerequisites

1. Node.js and npm

- **Purpose:** Node.js allows you to run JavaScript on the server-side, and npm (Node Package Manager) helps manage dependencies.
- **Download:** [Node.js Official Website](#)
- **Installation Instructions:** [Install Node.js and npm](#)

2. Express.js

- **Purpose:** Express.js simplifies server-side routing, middleware handling, and API development.
- **Installation Command:**

```
npm install express
```

3. MongoDB

- **Purpose:** MongoDB is a NoSQL database for storing application data.
- **Download:** [MongoDB Community Server](#)
- **Installation Instructions:** [Install MongoDB](#)

4. React.js

- **Purpose:** React.js is used for creating dynamic and reusable front-end user interfaces.
- **Installation Guide:** [Create a React App](#)

5. Version Control: Git

- **Purpose:** Git helps with version control and collaboration.
- **Download Git:** [Git Download](#)

6. Development Environment

- Recommended IDE: [Visual Studio Code](#)

Installation Steps

Backend Setup (Node.js and Express)

- **Navigate to the Backend Directory**

If the project is separated into client(Frontend) and server(backend) directories:

“cd server”

- **Install Backend Dependencies**

Install the required Node.js packages using npm:

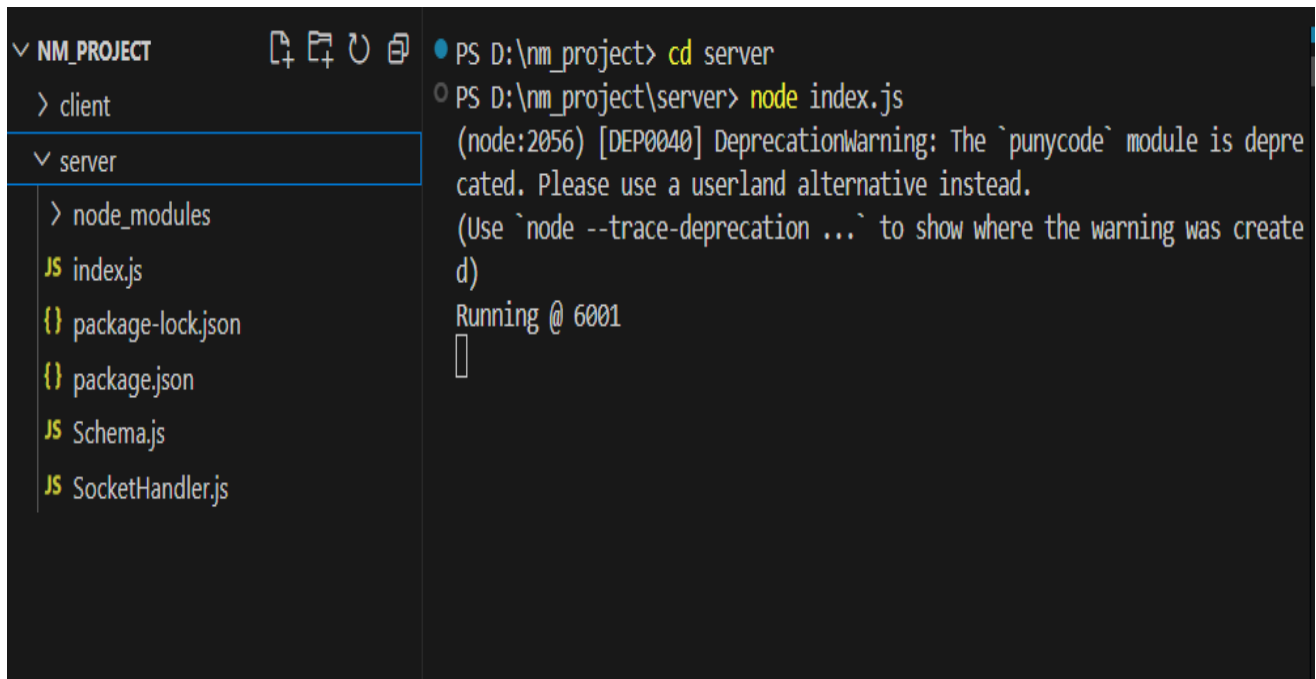
“npm install”

- **Start the Backend Server**

Start the backend server using the following command:

“node index.js”

The server should now be running @6001.



```
PS D:\nm_project> cd server
PS D:\nm_project\server> node index.js
(node:2056) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
Running @ 6001
█
```

Frontend Setup (React.js)

- **Navigate to the Frontend Directory**

Move to the frontend folder:

```
"cd client"
```

- **Install Frontend Dependencies**

Install React.js dependencies using npm:

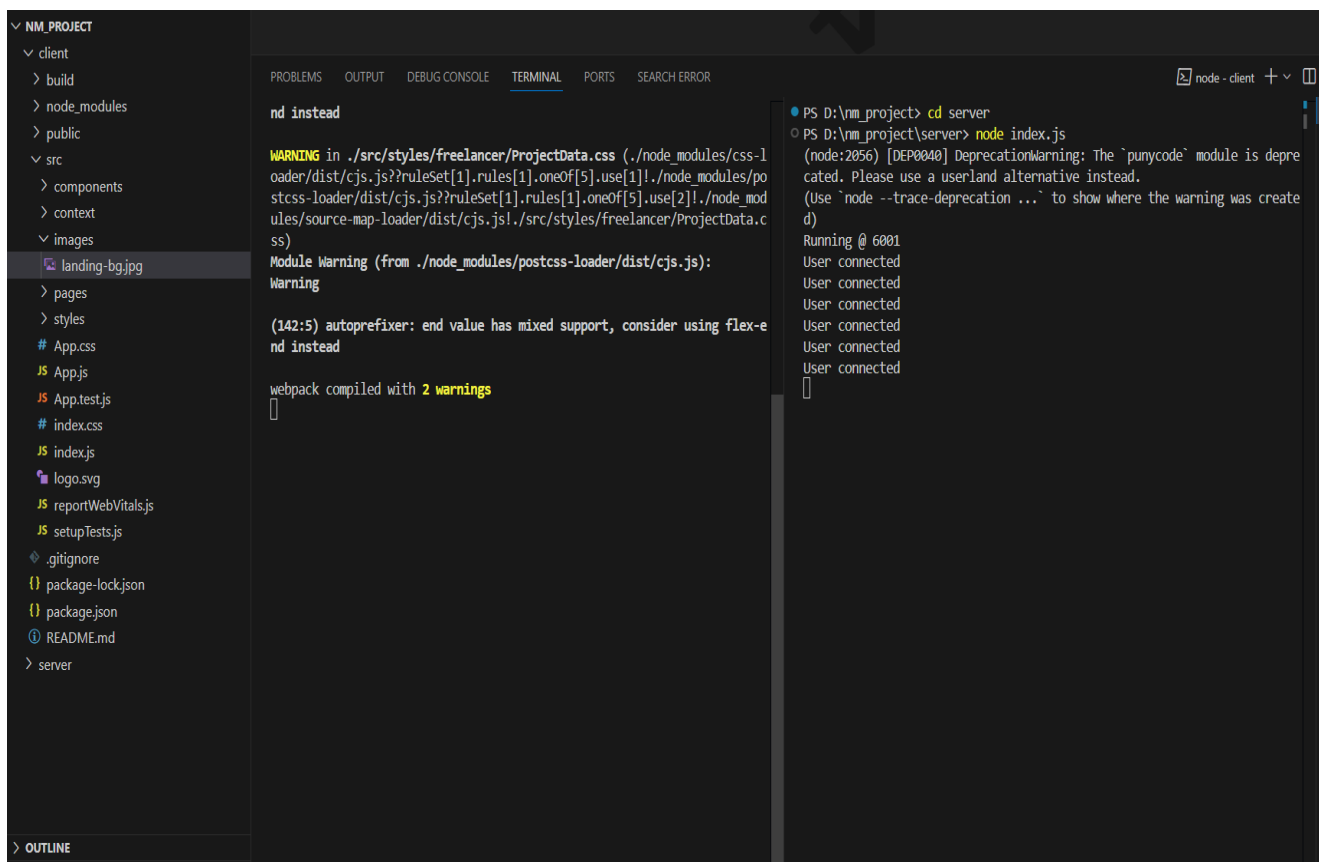
```
"npm install"
```

- **Start the Frontend Development Server**

Run the React development server using:

```
"npm start"
```

The frontend should now be running on <http://localhost:3000>



API DOCUMENTATION

Base URL:

http://localhost:5000/api (Example)

Endpoints

1. User Management

1.1 Create a New User

POST /users/register

Description: Registers a new user on the platform.

Request Body:

```
{
  "username": "string",
  "email": "string",
  "password": "string",
  "usertype": "string" // 'client' or 'freelancer'
}
```

Response:

- **201 Created:** User successfully registered.
- **400 Bad Request:** Validation error or email already exists.

1.2 User Login

POST /users/login

Description: Authenticates a user and generates a session token.

Request Body:

```
{
  "email": "string",
  "password": "string"
}
```

Response:

- **200 OK:** Returns user details and token.
- **401 Unauthorized:** Invalid credentials.

1.3 Get User Profile

GET /users/:id

Description: Fetches the details of a specific user by their ID.

Response:

```
{
  "id": "string",
  "username": "string",
  "email": "string",
  "usertype": "string"
}
```

2. Freelancer Management

2.1 Get Freelancer Details

GET /freelancers/:userId

Description: Fetch freelancer profile by user ID.

Response:

```
{
  "userId": "string",
  "skills": ["string"],
  "description": "string",
  "currentProjects": ["string"],
  "completedProjects": ["string"],
  "applications": ["string"],
  "funds": "number"
}
```

2.2 Update Freelancer Profile

PUT /freelancers/:userId

Description: Update freelancer's profile (e.g., skills, description).

Request Body:

```
{
  "skills": ["string"],
  "description": "string"
}
```

Response:

- **200 OK:** Profile updated successfully.
- **404 Not Found:** Freelancer profile not found.

3. Project Management

3.1 Create a New Project

POST /projects

Description: Allows a client to create a new project.

Request Body:

```
{
  "clientId": "string",
  "clientName": "string",
  "clientEmail": "string",
  "title": "string",
  "description": "string",
  "budget": "number",
  "skills": ["string"],
  "deadline": "string" // e.g., "2024-12-31"
}
```

Response:

- **201 Created:** Project successfully created.
- **400 Bad Request:** Missing or invalid fields.

3.2 Get All Projects

GET /projects

Description: Retrieve a list of all projects.

Response:

```
[
  {
    "id": "string",
    "clientId": "string",
    "title": "string",
    "description": "string",
    "budget": "number",
    "status": "string" // 'Available' or 'Assigned'
  }
]
```

3.3 Update Project Status

PUT /projects/:projectId/status

Description: Update the status of a project (e.g., Available, Assigned, Completed).

Request Body:

```
{
  "status": "string"
}
```

Response:

- **200 OK:** Status updated successfully.
- **404 Not Found:** Project not found.

4. Application Management

4.1 Submit an Application

POST /applications

Description: Allows a freelancer to apply for a project.

Request Body:

```
{
  "projectId": "string",
  "freelancerId": "string",
  "proposal": "string",
  "bidAmount": "number",
  "estimatedTime": "number" // in days
}
```

Response:

- **201 Created:** Application successfully submitted.
- **400 Bad Request:** Missing or invalid fields.

4.2 Get Applications for a Project

GET /applications/:projectId

Description: Retrieve all applications for a specific project.

Response:

```
[
  {
    "id": "string",
    "freelancerName": "string",
    "proposal": "string",
    "bidAmount": "number",
    "status": "string" // e.g., 'Pending', 'Accepted', 'Rejected'
  }
]
```


5. Chat Management

5.1 Get Chat History

GET /chats/:id

Description: Fetch chat history between a client and freelancer for a specific project.

Response:

```
{
  "_id": "string",
  "messages": [
    {
      "sender": "string",
      "message": "string",
      "timestamp": "string"
    }
  ]
}
```

5.2 Send a Message

POST /chats/:id

Description: Send a new message in a chat thread.

Request Body:

```
{
  "sender": "string",
  "message": "string"
}
```

Response:

- **200 OK:** Message sent successfully.

6. Admin Management

6.1 Get All Users

GET /admin/users

Description: Admin retrieves a list of all users.

Response:

```
[
  {
    "id": "string",
    "username": "string",
    "email": "string",
    "usertype": "string"
  }
]
```

6.2 Get All Projects

GET /admin/projects

Description: Admin retrieves a list of all projects.

Response: Similar to /projects.

6.3 Resolve Disputes

POST /admin/disputes/:projectId

Description: Admin resolves disputes related to a project.

Request Body:

```
{
  "resolution": "string"
}
```

Response:

- **200 OK:** Dispute resolved successfully.

USER INTERFACE

The Lancers platform will have distinct interfaces tailored to three key user roles: **Freelancer**, **Client**, and **Admin**. Each interface will provide an intuitive and responsive design, ensuring a smooth experience across devices. Below is a detailed overview of the UI:

1. Freelancer Interface

Dashboard

- **Overview Section:** Displays freelancer's profile summary, including:
 - Current funds.
 - Active projects.
 - Notifications for deadlines or messages.
- **Statistics:** Visual representation of:
 - Number of completed projects.
 - Earnings breakdown.
 - Skill demand trends (optional).

Profile Management

- **Edit Profile:** Update skills, description, and portfolio links.
- **View Profile:** Publicly visible profile that clients can view.
- **Manage Funds:** Section to request payouts or review earnings history.

Project Listings

- **Available Projects:** A searchable and filterable list of projects based on:
 - Budget range.
 - Skills required.
 - Deadline.
- **Apply for Projects:** Detailed project page with a form for submitting a bid and proposal.

Active Projects

- **Current Assignments:** List of ongoing projects with details like:
 - Deadline.
 - Communication with clients.
 - Submission status.
- **Submit Work:** Upload completed work with optional description and files.

Messages

- **Chat System:** Real-time chat feature to communicate with clients.
- **Notifications:** Alert system for new messages or updates.

2. Client Interface

Dashboard

- **Overview Section:** Displays project statistics, including:
 - Ongoing projects.

- Pending applications.
- Budget summary.

Profile Management

- **Edit Profile:** Update personal or company details.
- **View Profile:** Public-facing client profile showcasing posted projects.

Post New Project

- **Project Creation Form:** Inputs for title, description, skills required, budget, and deadline.
- **Draft Saving:** Option to save an incomplete project as a draft.

Project Management

- **My Projects:** A categorized list:
 - **Active Projects:** Ongoing with freelancers assigned.
 - **Pending Applications:** Projects awaiting freelancer selection.
 - **Completed Projects:** History of finished projects.
- **Review Applications:** View freelancer bids and proposals, including:
 - Freelancer profile preview.
 - Accept or reject options for applications.
- **Track Progress:** Updates from freelancers, including:
 - Status updates.
 - Deliverables.

Payment Management

- **Payment Portal:** To release payments for completed work.
- **Transaction History:** Overview of past payments.

Messages

- **Chat System:** Real-time chat with freelancers.

3. Admin Interface

Dashboard

- **Platform Overview:** Summary of key metrics:
 - Number of active users (freelancers and clients).
 - Ongoing and completed projects.
 - Dispute cases.
- **Search & Filter Tools:** To find users, projects, or disputes.

User Management

- **View Users:** Lists all users with search and filter capabilities.
- **Manage User Accounts:**
 - Suspend or activate accounts.
 - Resolve profile-related disputes.
 - Edit user roles (e.g., upgrading a freelancer to client).

Project Management

- **View All Projects:** Detailed list of every project, categorized by:

- Status (Available, Assigned, Completed).
- Budget range.
- **Monitor Progress:** View activity logs for active projects.
- **Resolve Issues:** Address flagged or disputed projects.

Dispute Resolution

- **Dispute Dashboard:** List of disputes raised by clients or freelancers.
- **Resolution Workflow:** Tools to review submissions, chat logs, and other evidence to make decisions.

Platform Settings

- **Policy Management:** Update terms, guidelines, and rules.
- **Announcements:** Publish news or platform-wide notifications.

4. Shared Features Across All Interfaces

Header

- **Logo:** Links back to the dashboard.
- **Navigation Bar:** Links to key sections like dashboard, profile, projects, and messages.
- **Search Bar:** Universal search for projects, users, or chats.
- **Notifications Icon:** Alerts for new messages, updates, or deadlines.

Footer

- Links to:
 - Privacy Policy.
 - Terms and Conditions.
 - Support or Contact Us.

Responsive Design

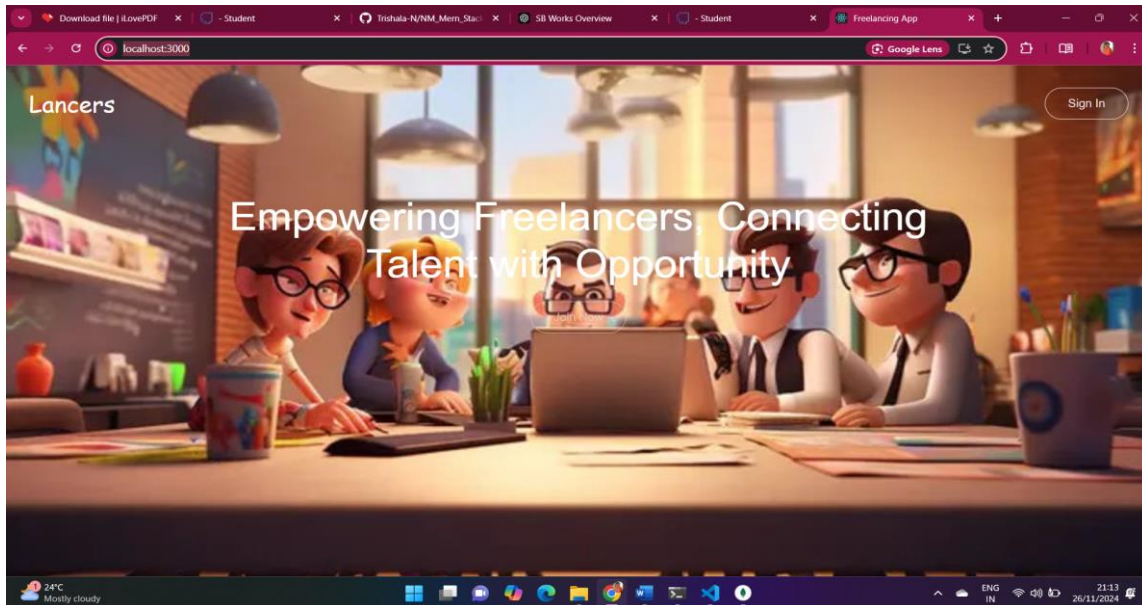
- **Desktop View:** Optimized for large screens with multi-column layouts.
- **Mobile View:** Compact, scrollable design with collapsible menus for easy navigation.

Additional Features

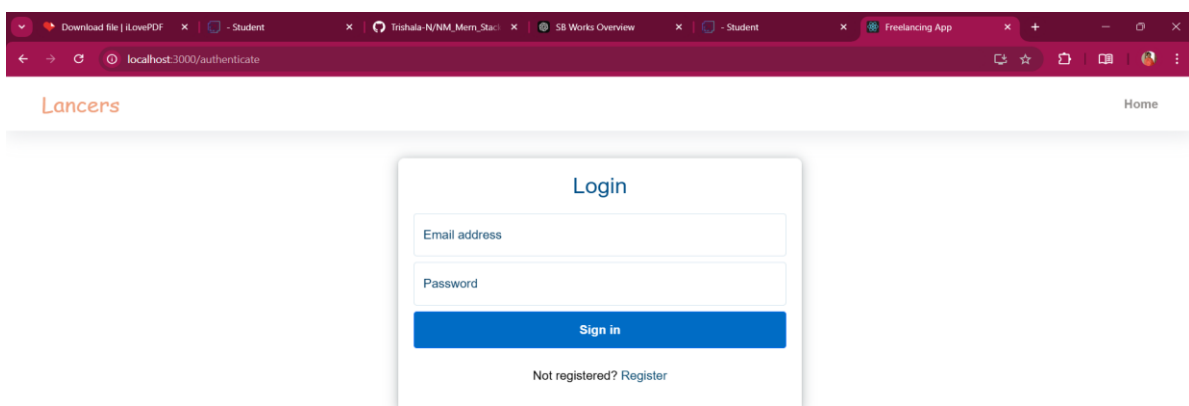
- **Authentication & Security**
 - Role-based access (Freelancer, Client, Admin).
 - Email-based login and password reset.
 - Two-factor authentication (optional).
- **UI Styling**
 - **Modern Design:** Material-UI or Bootstrap for a professional look.
 - **Theme Support:** Light and dark modes for better user experience.

PROJECT IMPLEMENTATION:

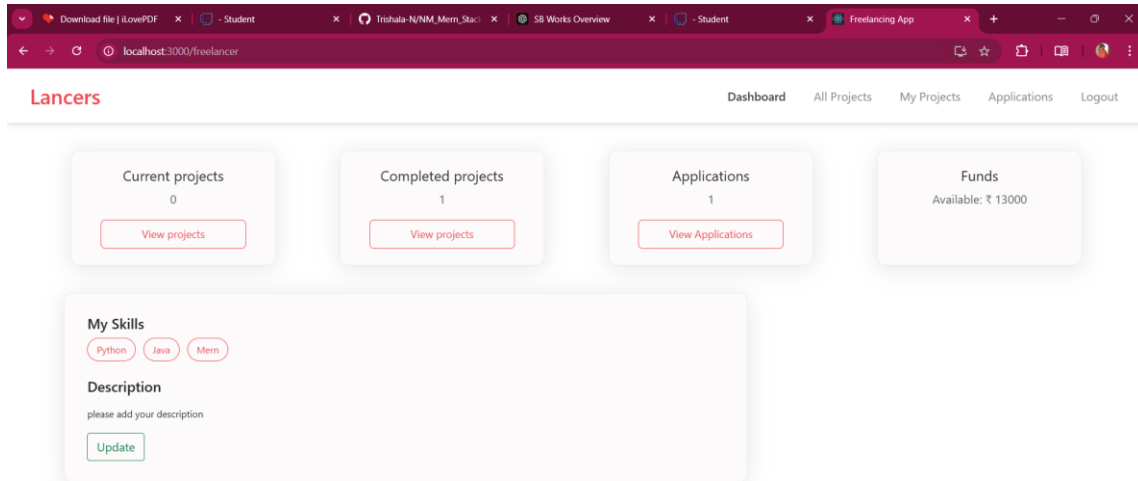
- **Landing page:**



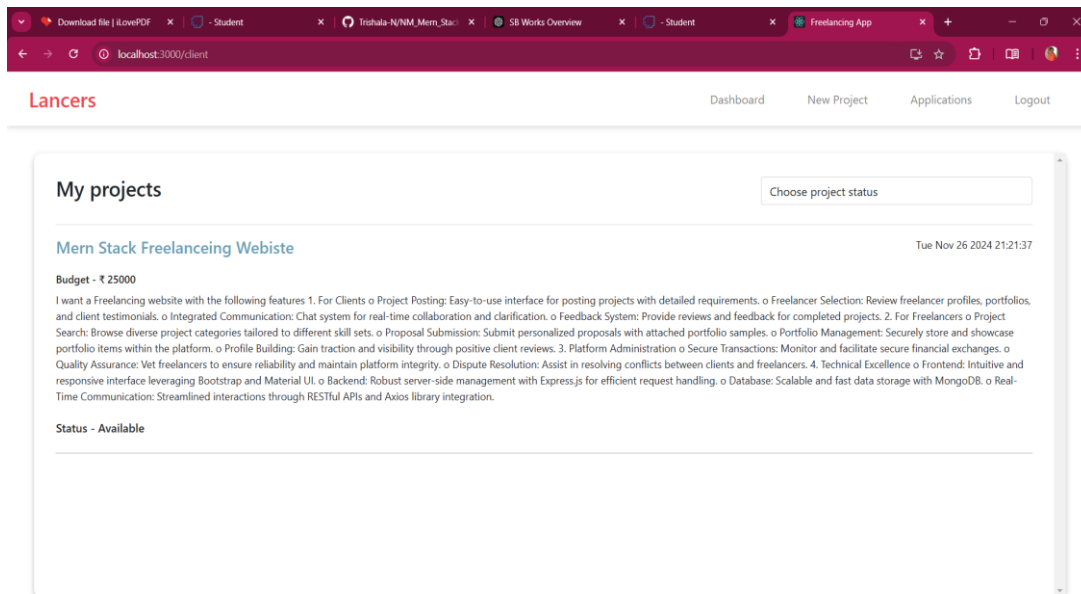
- **Authentication:**



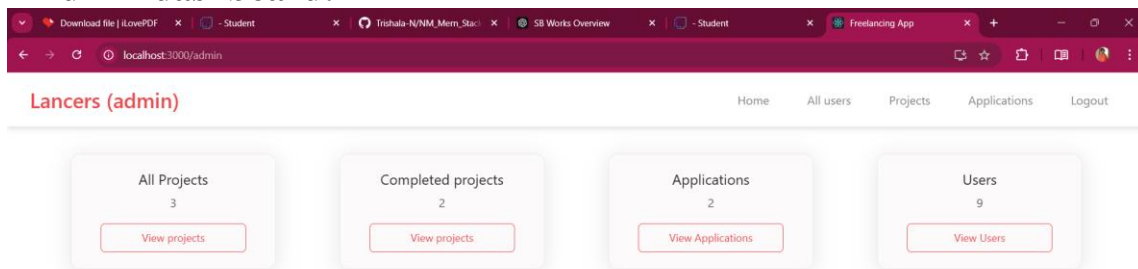
- **Freelancer dashboard:**



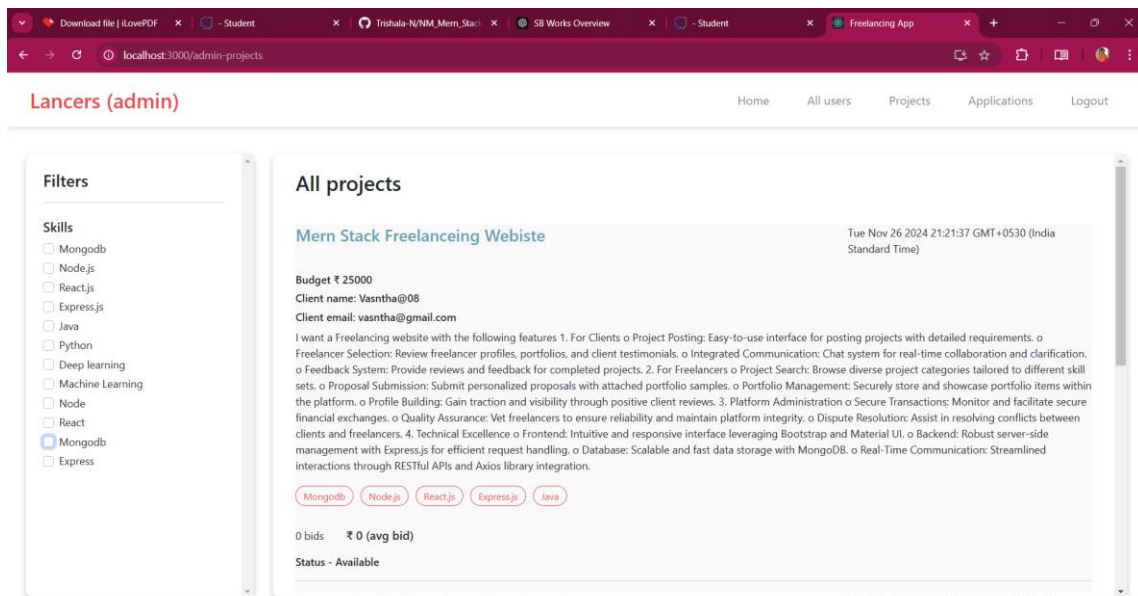
- **Client dashboard:**



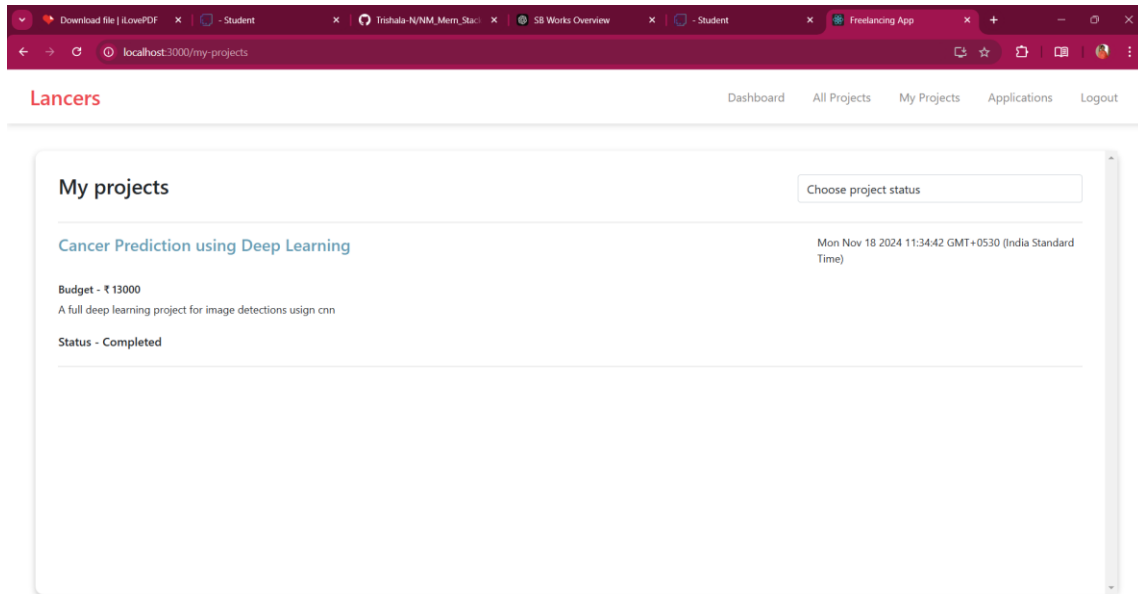
- **Admin dashboard:**



- **All projects:**



- **Freelance projects:**



- **Applications:**

