

NAAN MUDHALVAN PROJECT REPORT

TITLE :FREELANCING APPLICATION - MERN

COLLEGE: AALIM MUHAMMED SALEGH COLLEGE OF ENGINEERING

MEMBERS:

- 1.CHOCKALINGAM S RM (110121104022)
- 2.IKRAM MUSTAFA HUSSAIN A (110121104027)
- 3.KAMALESHWARAN S (110121104032)
- 4.MOHAMED ABDULLAH F (110121104039)

Freelancing Application MERN

INTRODUCTION:

IKAC Works is a freelancing platform designed to connect clients with skilled freelancers. It provides an intuitive interface for project posting, bidding, and seamless collaboration. With a dedicated admin team focused on ensuring security and effective communication, IKAC Works strives to be the preferred platform for both clients and freelancers.

DESCRIPTION:

Welcome to **IKAC Works**, a revolutionary freelancing platform that redefines how clients connect with skilled freelancers. Our intuitive interface allows clients to post diverse projects—from creative endeavors to technical tasks—while freelancers can seamlessly bid on these opportunities based on their expertise and skills.

At **IKAC Works**, we prioritize efficiency and transparency in the freelancing process. Clients can review freelancer profiles, assess past work, and select the ideal candidate for their projects. Once a freelancer is chosen, clients can easily communicate and collaborate within the platform, making the workflow smooth and efficient.

Our dedicated admin team ensures the integrity and security of every transaction. With stringent oversight, we guarantee the reliability and quality of freelancers on our platform. The admin team upholds the platform's integrity and facilitates smooth communication between clients and freelancers, fostering positive and productive working relationships.

Freelancers on **IKAC Works** benefit from a straightforward project submission process. After completing an assigned project, freelancers can submit their work directly through the platform, providing clients a hassle-free experience. Clients can review the work and provide feedback, encouraging a collaborative environment focused on excellence.

Stay informed about the latest projects and industry trends with real-time updates and notifications. **IKAC Works** aims to be the go-to platform for clients seeking dependable freelancers and freelancers looking for exciting opportunities to showcase their skills.

Join **IKAC Works** today and experience a new era of freelancing, where projects are managed efficiently, skills are recognized, and collaborations thrive in a secure and dynamic environment.

Scenario-based case-study:

Sarah, a recent graduate with a degree in graphic design, is eager to showcase her skills and build a strong freelance portfolio. She stumbles upon **IKAC Works** while searching for online freelancing opportunities.

Finding the Perfect Project: Impressed by the user-friendly interface, Sarah browses through various project categories. She discovers a project posted by a local bakery, "Sugar Rush," seeking a logo redesign. The project description details the bakery's brand identity and target audience, giving Sarah a clear understanding of the client's needs.

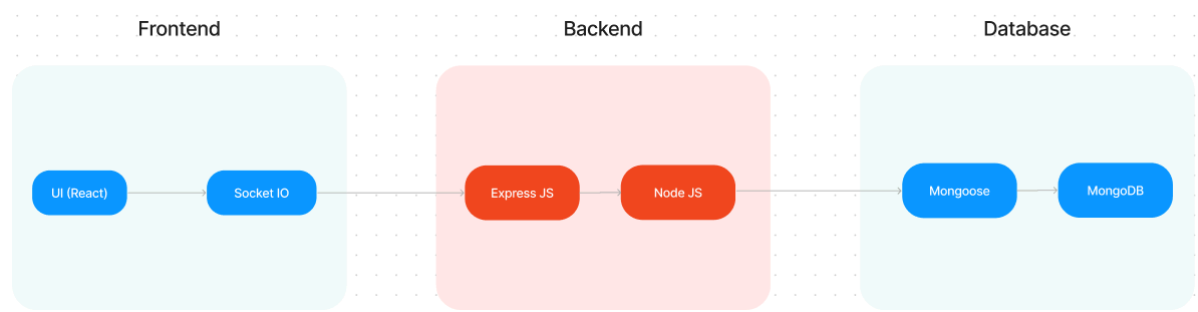
Bidding with Confidence: Confident in her design skills, Sarah dives into the project details. **IKAC Works** allows her to review the bakery's previous marketing materials, further solidifying her design approach. She submits a compelling proposal highlighting her relevant experience and attaches a few samples from her portfolio, securely stored within the platform.

Communication & Collaboration: "Sugar Rush" selects Sarah's proposal based on her impressive portfolio and competitive pricing. **IKAC Works** facilitates seamless communication between Sarah and the bakery, allowing them to discuss project specifics and refine the design direction through an integrated chat system.

Delivery & Feedback: Once finalized, Sarah submits her logo design through the **IKAC Works** platform. "Sugar Rush" can review the design, provide feedback, and request minor revisions if needed. **IKAC Works** fosters a collaborative environment where both parties can work together to achieve the desired outcome.

Building a Thriving Career: Following successful project completion and a glowing review from "Sugar Rush," Sarah's profile on **IKAC Works** gains traction. The positive experience encourages her to actively seek new projects on the platform. With a growing portfolio and strong client testimonials, Sarah is well on her way to establishing a thriving freelance career on **IKAC Works**.

TECHNICAL ARCHITECTURE:



The technical architecture of **IKAC Works** follows a client-server model, where the frontend serves as the client and the backend acts as the server. The frontend includes the user interface, and presentation, and integrates the Axios library to facilitate seamless communication with the backend through RESTful APIs.

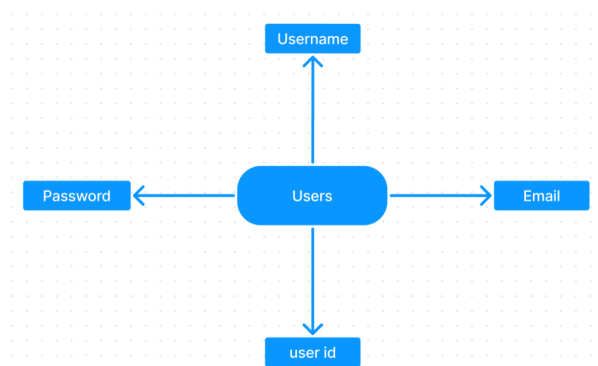
To enhance user experience, the frontend leverages the Bootstrap and Material UI libraries, creating a visually appealing and responsive interface for users.

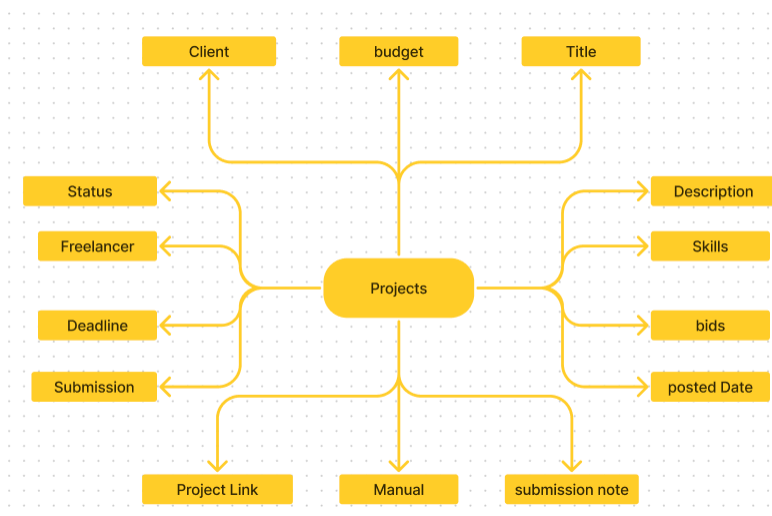
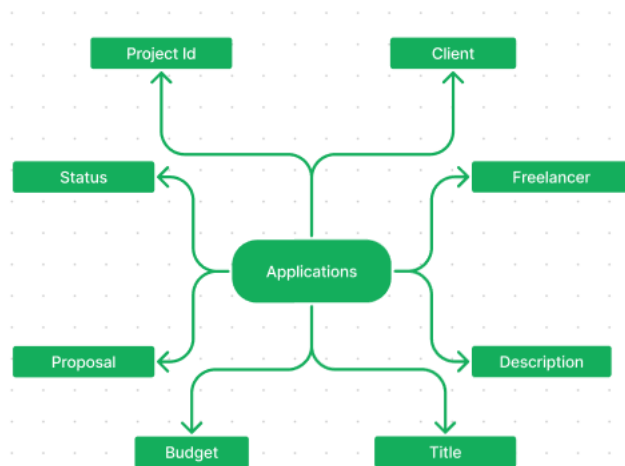
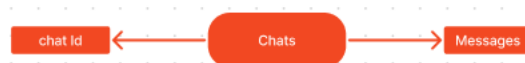
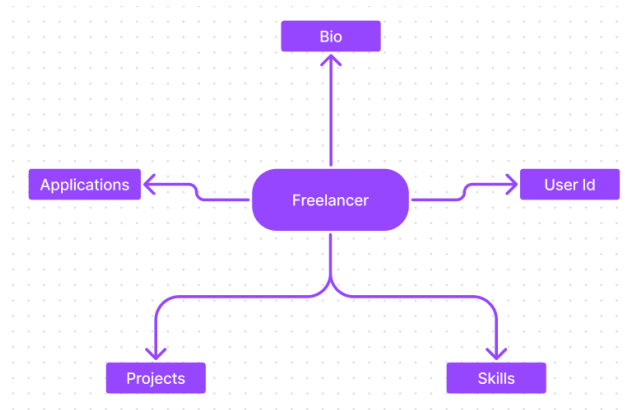
On the backend, we utilize the Express.js framework to manage server-side logic and communication. Express.js provides a robust foundation for efficiently handling requests and responses.

For data storage and retrieval, **IKAC Works** relies on MongoDB. MongoDB offers a scalable and efficient solution for storing various data, including user profiles, project details, and communication records. This ensures quick and reliable access to the information required to support a seamless user experience.

Together, the frontend and backend components, complemented by Express.js and MongoDB, form a comprehensive technical architecture for **IKAC Works**. This architecture supports real-time communication, efficient data exchange, and seamless integration, ensuring a smooth and immersive experience for users on the platform.

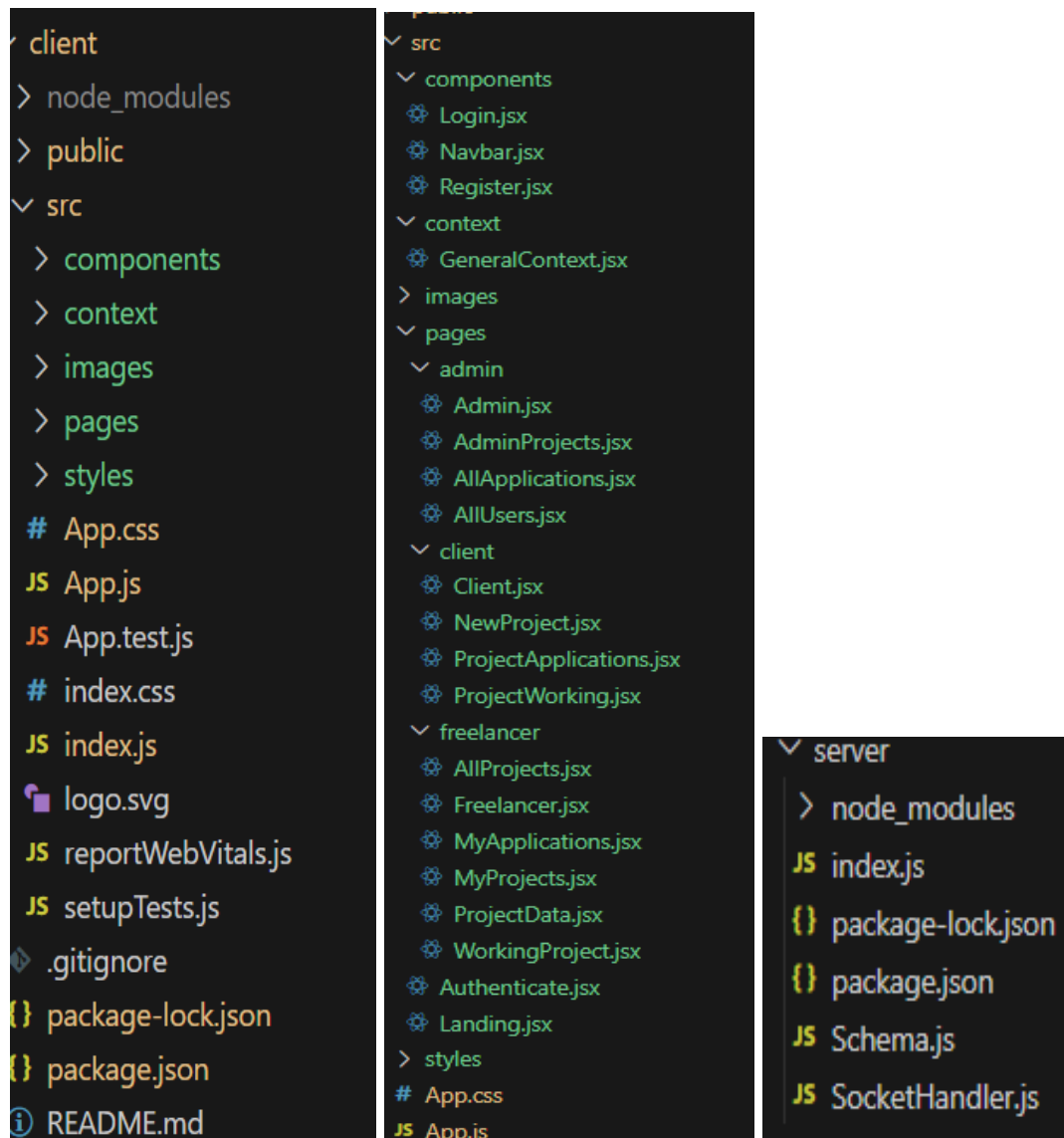
ER DIAGRAM:





IKAC Works connects clients with skilled freelancers through a user-friendly platform. Clients can post projects with detailed descriptions and browse freelancer profiles to find the perfect match. Freelancers can submit proposals, collaborate with clients through a secure chat feature, and submit work for review and payment in a safe environment. An admin team ensures quality control and smooth communication, making IKAC Works a top choice for both clients and freelancers.

PROJECT STRUCTURE:



IKAC Works leverages React.js for the user interface. The client-side code consists of reusable components for profiles, projects, and chat, organized into pages like project browsing and freelancer profiles. Shared data, such as user information or search filters, is managed with React

Context. On the server side, Node.js handles API requests for user management, project actions, and communication. Mongoose models ensure structured interaction with the MongoDB database. This breakdown provides a foundational understanding of IKAC Works' architecture.

PRE-REQUISITES:

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

Download: <https://nodejs.org/en/download/>

Installation instructions: <https://nodejs.org/en/download/package-manager/>

✓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.

Installation: Open your command prompt or terminal and run the following command:

npm install express

✓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.

Download: <https://www.mongodb.com/try/download/community>

Installation instructions: <https://docs.mongodb.com/manual/installation/>

✓React.js:

React.js is a popular JavaScript library for building user interfaces. 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 interfaces.

Follow the installation guide: <https://reactjs.org/docs/create-a-new-react-app.html>

✓**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 interfaces for the admin dashboard. For making better UI we have also used some libraries like material UI and bootstrap.

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

Git: Download and installation instructions can be found at: <https://git-scm.com/downloads>

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

- Visual Studio Code: Download from <https://code.visualstudio.com/download>

Install Dependencies:

- Navigate into the cloned repository directory:
`cd freelancer-app-MERN`

- 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 IKAC Works app will be accessible at <http://localhost:3000>

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

Application flow:

Freelancer Responsibilities:

- **Project Submission:** Freelancers are responsible for submitting completed and high-quality work for the assigned projects 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 communication with clients, 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 clients and fellow freelancers.
- **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 an admin, one of your key responsibilities is to monitor and ensure the integrity and security 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 admin'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 the necessary tools

For frontend, we use:

- React
- Bootstrap
- Material UI
- Axios
- react-bootstrap

2. Open the backend folder to install the 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.

```
package.json M X
client > package.json > ...
3  "version": "0.1.0",
4  "private": true,
5  "dependencies": {
6    "@testing-library/jest-dom": "^5.17.0",
7    "@testing-library/react": "^13.4.0",
8    "@testing-library/user-event": "^13.5.0",
9    "axios": "^1.5.1",
10   "react": "^18.2.0",
11   "react-dom": "^18.2.0",
12   "react-icons": "^4.11.0",
13   "react-router-dom": "^6.19.0",
14   "react-scripts": "5.0.1",
15   "socket.io-client": "^4.7.2",
16   "uuid": "^9.0.1",
17   "web-vitals": "^2.1.4"
18 },
19 "scripts": {
20   "start": "react-scripts start",
21   "build": "react-scripts build",
22   "test": "react-scripts test",
23   "eject": "react-scripts eject"
24 },
25 "eslintConfig": {
26   "extends": [
27     "react-app",
28     "react-app/jest"
29   ]
30 },
31 "browserslist": {
32   "production": [
33     ">0.2%",
34     "not dead",
35     "not op_mini all"
36   ],
37   "development": [
38     "last 1 chrome version",
39     "last 1 firefox version",
40     "last 1 safari version"
41   ]
42 }
43 }
44 }
```

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

```
package.json X
server > package.json > ...
1  {
2    "name": "server",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "type": "module",
7    "scripts": {
8      "test": "echo \"Error: no test specified\" && exit 1"
9    },
10   "keywords": [],
11   "author": "",
12   "license": "ISC",
13   "dependencies": {
14     "bcrypt": "^5.1.1",
15     "body-parser": "^1.20.2",
16     "cors": "^2.8.5",
17     "express": "^4.18.2",
18     "http": "^0.0.1-security",
19     "mongoose": "^7.6.1",
20     "socket.io": "^4.7.2",
21     "uuid": "^9.0.1"
22   }
23 }
```

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 (communication 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 projects.
 - 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 projects).

7. Project Management:

- Allow clients to post projects with details and budget.
- Enable freelancers to browse projects, search by skills, and submit proposals.
- Implement a system for clients to review applications and choose freelancers.

8. Secure Communication & Collaboration:

- Integrate a secure chat system within projects for communication between clients and freelancers.
- Allow file attachments and feedback exchange to facilitate collaboration.

9. Admin Panel (Optional):

- Implement an admin panel with functionalities like:
- Managing users
- Monitoring project updates and applications
- Accessing transaction history

Milestone 3: Database development

- o Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas.
- o Create a database and define the necessary collections for users, freelancer, projects, chats, and applications.
- o Connect the database to the server with the code provided below.

```

const server = http.createServer(app);

const io = new Server(server, {
  cors: {
    origin: '*',
    methods: ['GET', 'POST', 'PUT', 'DELETE']
  }
});

io.on("connection", (socket) =>{
  console.log("User connected");

  SocketHandler(socket);
})

const PORT = 6001;

mongoose.connect('mongodb://localhost:27017/Freelancing',{
  useNewUrlParser: true,
  useUnifiedTopology: true
}).then(()=>{

  server.listen(PORT, ()=>{
    console.log(`Running @ ${PORT}`);
  });
}).catch((e)=> console.log(`Error in db connection ${e}`));

```

The Schemas for the database are given below:

```

JS Schemas X
server > JS Schemas > [0] projectSchema > submissionDescription
1 import mongoose, { Schema, mongo } from "mongoose";
2
3 const userSchema = mongoose.Schema({
4   username: {
5     type: String,
6     require: true
7   },
8   email: {
9     type: String,
10    require: true,
11    unique: true
12  },
13  password: {
14    type: String,
15    require: true
16  },
17  usertype:{
18    type: String,
19    require: true
20  }
21 })
22

```

```

23 const freelancerSchema = mongoose.Schema({
24   userId: String,
25   skills: {
26     type: Array,
27     default: []
28   },
29   description: {
30     type: String,
31     default: ""
32   },
33   currentProjects: {
34     type: Array,
35     default: []
36   },
37   completedProjects: {
38     type: Array,
39     default: []
40   },
41   applications: {
42     type: Array,
43     default: []
44   },
45   funds: {
46     type: Number,
47     default: 0
48   },
49 })
50

```

```

51
52   const projectSchema = mongoose.Schema({
53     clientId: String,
54     clientName: String,
55     clientEmail: String,
56     title: String,
57     description: String,
58     budget: Number,
59     skills: Array,
60     bids: Array,
61     bidAmounts: Array,
62     postedDate: String,
63     status: {
64       type: String,
65       default: "Available"
66     },
67     freelancerId: String,
68     freelancerName: String,
69     deadline: String,
70     submission: {
71       type: Boolean,
72       default: false
73     },
74     submissionAccepted: {
75       type: Boolean,
76       default: false
77     },
78     projectLink: {
79       type: String,
80       default: ""
81     },
82     manulaLink: {
83       type: String,
84       default: ""
85     },
86     submissionDescription: {
87       type: String,
88       default: ""
89     },
90   })

```

```

92
93   const applicationSchema = mongoose.Schema({
94
95     projectId: String,
96     clientId: String,
97     clientName: String,
98     clientEmail: String,
99     freelancerId: String,
100    freelancerName: String,
101    freelancerEmail: String,
102    freelancerSkills: Array,
103    title: String,
104    description: String,
105    budget: Number,
106    requiredSkills: Array,
107    proposal: String,
108    bidAmount: Number,
109    estimatedTime: Number,
110    status: {
111      type: String,
112      default: "Pending"
113    }
114  })
115
116   const chatSchema = mongoose.Schema({
117     _id: {
118       type: String,
119       require: true
120     },
121     messages: {
122       type: Array
123     }
124   })
125
126   export const User = mongoose.model('users', userSchema);
127   export const Freelancer = mongoose.model('freelancer', freelancerSchema);
128   export const Project = mongoose.model('projects', projectSchema);
129   export const Application = mongoose.model('applications', applicationSchema);
130   export const Chat = mongoose.model('chats', chatSchema);

```

Milestone 4: Frontend development

1. Setting the Stage:

The IKAC Works 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 IKAC Works interface to life.

2. Crafting the User Experience:

Next, we'll focus on the user interface (UI). This involves:

- Designing reusable UI components like buttons, forms, and project cards.
- Defining the layout and styling for a visually appealing and consistent interface.
- Implementing navigation elements for intuitive movement between features.
- These steps will create a user-friendly experience for both freelancers and clients.

3. Bridging the Gap:

The final stage connects the visual interface with the backend data. We'll:

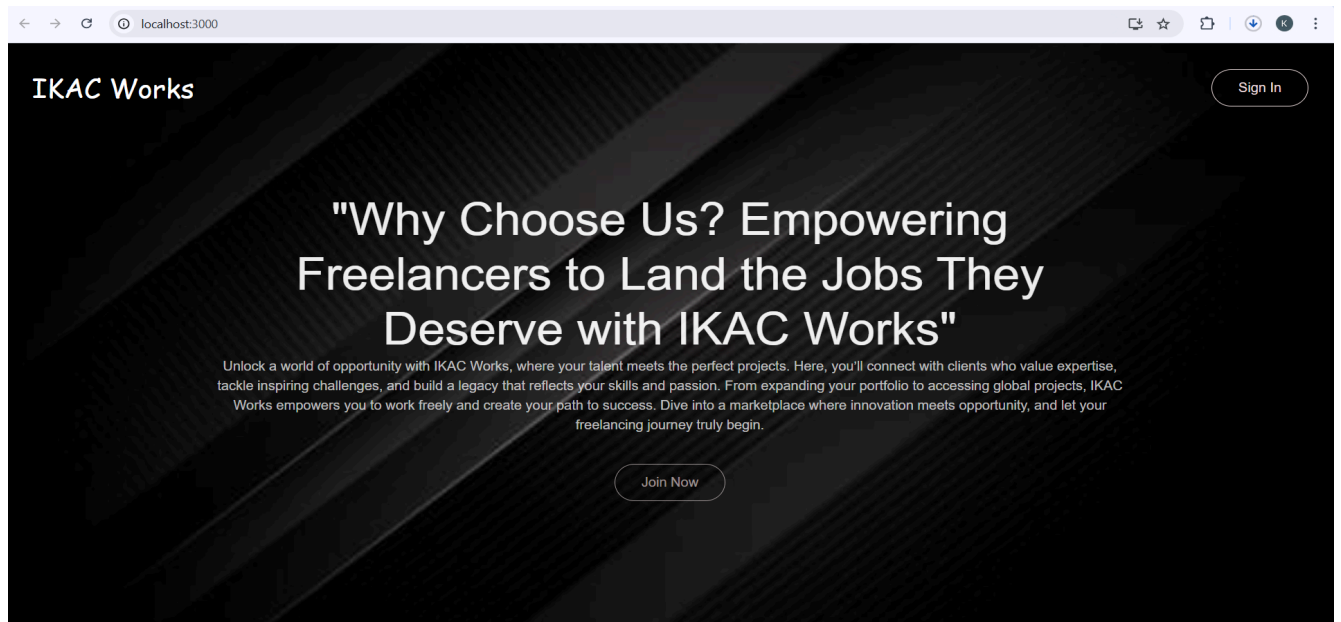
- Integrate the frontend with IKAC Works' API endpoints.
- Implement data binding to ensure dynamic updates between user interactions and the displayed information.

This completes the front-end development, bringing the IKAC Works platform to life for users.

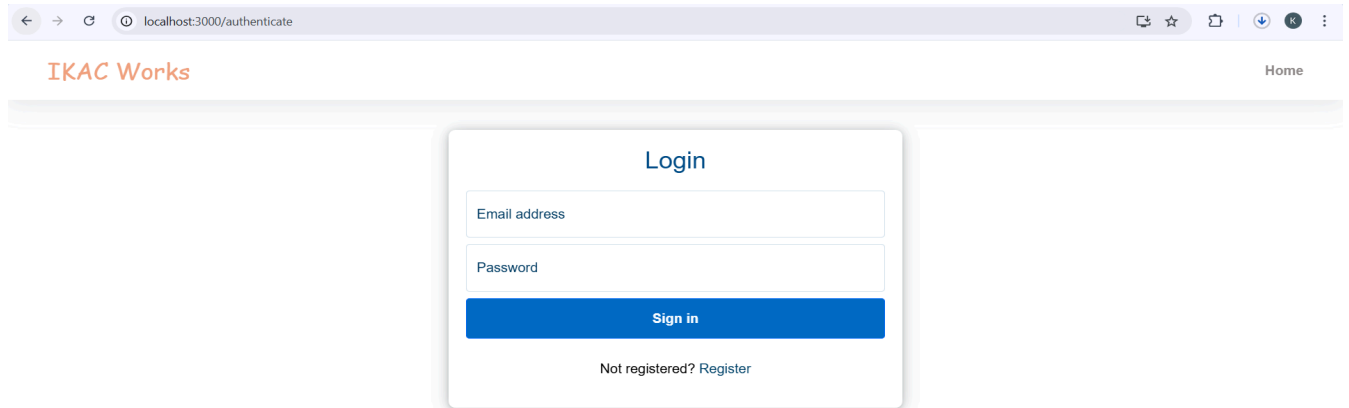
Milestone 5: Project Implementation

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 interface of the application looks a bit like the images provided below.

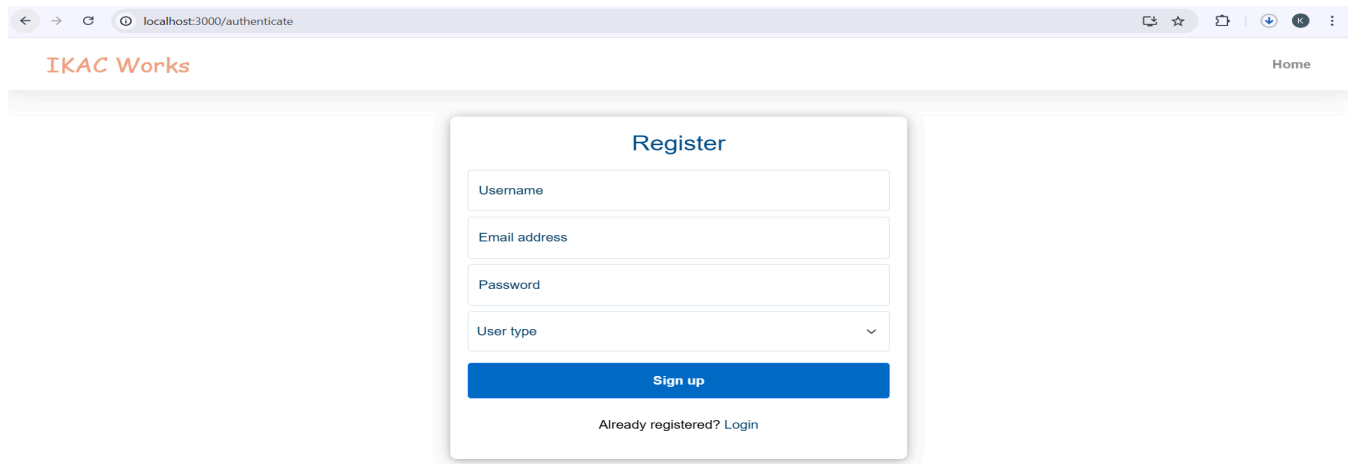
Landing page:



Authentication:

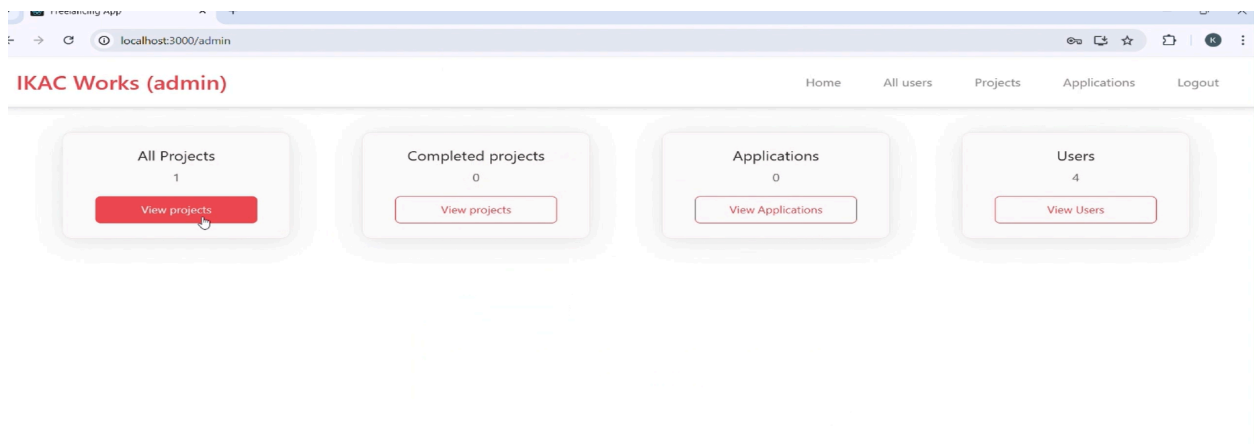


Register Page:



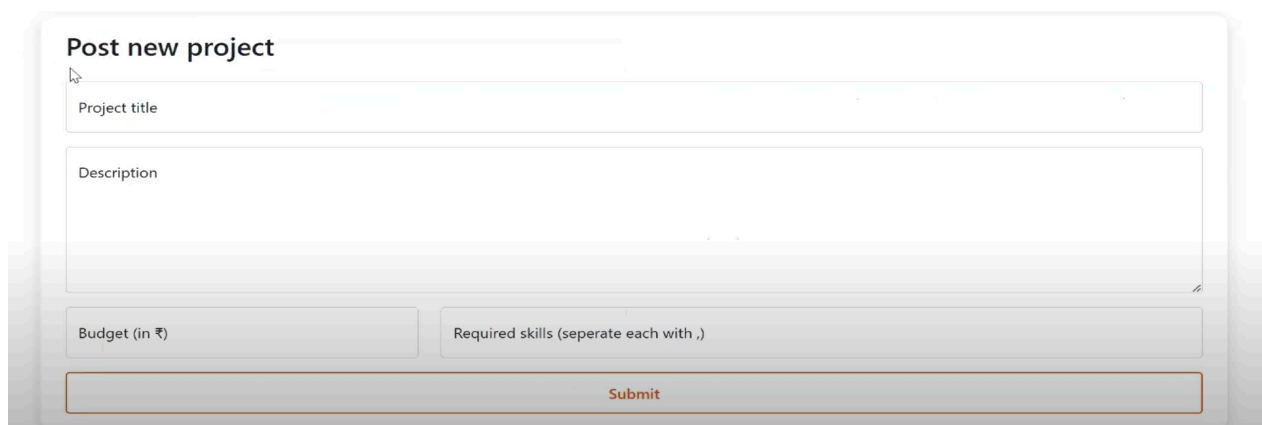
The screenshot shows a web browser at localhost:3000/authenticate. The page has a header with "IKAC Works" on the left and "Home" on the right. The main content is a "Register" form with the following fields: Username, Email address, Password, and User type (a dropdown menu). Below these fields is a blue "Sign up" button. At the bottom of the form, there is a link that says "Already registered? Login".

Admin Dashboard:



The screenshot shows an admin dashboard in a web browser at localhost:3000/admin. The header includes "IKAC Works (admin)" on the left and navigation links "Home", "All users", "Projects", "Applications", and "Logout" on the right. The main area contains four summary cards: "All Projects" with a count of 1 and a red "View projects" button; "Completed projects" with a count of 0 and a "View projects" button; "Applications" with a count of 0 and a "View Applications" button; and "Users" with a count of 4 and a "View Users" button.

New Project:



The screenshot shows a form titled "Post new project". It contains the following fields: "Project title" (a text input), "Description" (a large text area), "Budget (in ₹)" (a text input), and "Required skills (separate each with ,)" (a text input). At the bottom of the form is a large orange "Submit" button.

All Project:

Freelancing App

localhost:3000/admin-projects

IKAC Works (admin)

HomeAll usersProjectsApplicationsLogout

Filters

Skills

☐ java

All projects

java

Mon Oct 28 2024 10:04:41 GMT+0530 (India Standard Time)

Budget ₹ 200000

Client name: mdxabu

Client email: abu@gmail.com

java

0 bids ₹ 0 (avg bid)

Status - Available

All Users:

Freelancing App

localhost:3000/all-users

IKAC Works (admin)

HomeAll usersProjectsApplicationsLogout

All Users

User Id	Username	Email	User Role
6715dab609b197328a8d0626	kamaleshwaran11	101@gmail.com	freelancer
671f13d373b10cbce7eda125	sk	admin@gmail.com	admin
671f140b73b10cbce7eda135	mdxabu	abu@gmail.com	client
671f14f273b10cbce7eda13d	chockalingam	chocks@gmail.com	freelancer