| Class Name: Project (from project.js) | |
| --- | --- |
| Parent Class: mongoose.Model  Subclass: None | |
| Responsibilities   * Define the schema for a project.<br> - Ensure adherence to the schema.<br> - Facilitate CRUD operations in MongoDB. | Collaborators:   * mongoose.Schema<br> - mongoose.model<br> - User model (referenced by the user field) |

| Class Name: App(from server.js) |
| --- |
| Parent Class: Express.Application  Subclass: None explicitly defined |
| Responsibilities   * Set up an Express application * Connect to MongoDB using Mongoose. * Configure middleware for parsing JSON, handling CORS, and file uploads. * Define routes for user registration, login, profile management, and project management. * Implement JWT-based authentication for protected routes. * Handle errors and send appropriate responses. |
| Collaborator: NextResponse from 'next/server'   * express: Web framework for creating the server and handling routing. * mongoose: ODM (Object Data Modeling) library for MongoDB to define schemas and interact with the database. * bcryptjs: Library for hashing passwords. * jsonwebtoken: Library for creating and verifying JWT tokens. * cors: Middleware to enable Cross-Origin Resource Sharing. * multer: Middleware for handling file uploads. * path: Node.js module for working with file and directory paths. * User: Mongoose model representing users in the database. * Project: Mongoose model representing projects in the database. |

| Class Name: User |
| --- |
| Parent Class: mongoose.Model  Subclass: None |
| Responsibilities   * Define the schema for a user in the MongoDB database. * Ensure that each user document adheres to the schema, including required fields and their types. * Facilitate interaction with the MongoDB database through Mongoose for CRUD (Create, Read, Update, Delete) operations. |
| Collaborator: NextResponse from 'next/server'   * mongoose.Schema: Used to define the structure of the User documents. * mongoose.model: Used to compile the schema into a model. |

| Class Name: CurrentProjects(from CurrentProjects.js) |  |
| --- | --- |
| Parent Class: None  Subclass: None |  |
| Responsibilities   * Display Current Projects of user * Enhance UI purposes | Collaborators:   * None (MongoDB in the future) |

| Class Name: EditProfileForm | |
| --- | --- |
| Parent Class: React.Component  Subclass: None | |
| Responsibilities   * Render a form to allow users to edit their profile information. * Handle form state using React hooks. * Fetch the current profile data from the server and populate the form fields. * Submit the updated profile data to the server. * Navigate between different views based on user actions. | Collaborators:   * React: Library for building user interfaces and handling component state. * useState and useEffect: React hooks for managing state and side effects. * useNavigate: Hook from react-router-dom for navigating programmatically. * fetch: Function for making HTTP requests to the server. * localStorage: Web API for storing and retrieving authentication tokens. |

| Class Name: ProfileView |
| --- |
| Parent Class: React.Component  Subclass: None |
| Responsibilities:   * Render the user's profile information. * Fetch the profile data from the server. * Display the fetched profile data. * Provide navigation options to edit the profile or return to the dashboard. |
| Collaborators:   * useState * useRouter * setIsRegister |
|  |

| Class Name: AddProject |  |
| --- | --- |
| Parent Class: React.Component  Subclass: None |  |
| Responsibilities   * Render the form for adding a new project. * Manage form state and handle form submission. * Send a POST request to add the new project. * Navigate to the project list on successful submission. * Provide a cancel button to navigate back to the project list without submitting. | Collaborators:   * React: Library for building user interfaces and handling component state. * useState: React hook for managing state. * useNavigate: Hook from react-router-dom for navigating programmatically. * fetch: Function for making HTTP requests to the server. * localStorage: Web API for storing and retrieving authentication tokens. |

| Class Name: ProjectList | |
| --- | --- |
| Parent Class: React.Component  Subclass: None | |
| Responsibilities   * Fetch and display a list of projects from the server. * Provide UI elements to navigate to add new projects and back to the dashboard. * Display each project with its details and a delete button. * Handle deletion of projects by sending a DELETE request to the server and updating the local state. | Collaborators:   * React: Library for building user interfaces and handling component state. * useState: React hook for managing state. * useEffect: React hook for performing side effects in function components. * useNavigate: Hook from react-router-dom for navigating programmatically. * fetch: Function for making HTTP requests to the server. * localStorage: Web API for storing and retrieving authentication tokens.ter |

| Class Name: Register | |
| --- | --- |
| Parent Class: React.Component  Subclass: None | |
| Responsibilities   * Fetch and display a list of projects from the server. (This is not explicitly implemented in the provided code but can be added separately) * Provide UI elements to navigate to add new projects and back to the dashboard. * Display each project with its details and a delete button. (Not implemented in this code) * Handle deletion of projects by sending a DELETE request to the server and updating the local state. (Not implemented in this code) | Collaborators:   * React: Library for building user interfaces and handling component state. * useState: React hook for managing state. * useEffect: React hook for performing side effects in function components. * useNavigate: Hook from react-router-dom for navigating programmatically. * axios: Library for making HTTP requests to the server. * localStorage: Web API for storing and retrieving authentication tokens. |

| Class Name: Dashboard | |
| --- | --- |
| Parent Class: N/A  Subclass: N/A | |
| Responsibilities   * Manages state for collapsing/expanding side panel (isCollapsed state) * Fetches profile data from an API endpoint (fetchProfileData function) * Handles navigation using useNavigate from 'react-router-dom' * Defines event handlers for various UI interactions (handleProfileView, handleConnect, handleProjects, handleLogout, toggleCollapse) * Renders the main dashboard layout with a side panel and main content area * Displays user profile data if available. | Collaborators:   * React (useState, useEffect) * react-router-dom (useNavigate) |

| Class Name: Index.js | |
| --- | --- |
| Parent Class: N/A  Subclass: N/A | |
| Responsibilities   * Imports necessary modules and components (React, ReactDOM, App, reportWebVitals) * Renders the main application component (<App />) within a React.StrictMode wrapper using ReactDOM.createRoot * Includes a call to reportWebVitals() for measuring performance in the app | Collaborators:   * React (React, ReactDOM) * './App' (Assuming this is the main application component) * './reportWebVitals' (Used for performance measurement) |

| Class Name: reportWebVitals.js | |
| --- | --- |
| Parent Class: N/A  Subclass: N/A | |
| Responsibilities   * Defines a function reportWebVitals that takes onPerfEntry as a parameter * Checks if onPerfEntry is a function and then imports the 'web-vitals' module asynchronously * Once the 'web-vitals' module is loaded, it invokes functions (getCLS, getFID, getFCP, getLCP, getTTFB) provided by the module and passes onPerfEntry to each of them | Collaborators:   * 'web-vitals' (module imported dynamically) * onPerfEntry (a function passed as a parameter to reportWebVitals) |

| Class Name: setupTests.js | |
| --- | --- |
| Parent Class: N/A  Subclass: N/A | |
| Responsibilities   * Comments explaining the purpose of the import and its usage. * Imports '@testing-library/jest-dom' which adds custom Jest matchers for asserting on DOM nodes. * Enhances Jest's capabilities to assert on DOM elements using methods like toHaveTextContent provided by '@testing-library/jest-dom'. | Collaborators:   * '@testing-library/jest-dom' (library providing custom Jest matchers for DOM assertions) |

### **Description of System Interaction with the Environment**

#### **Dependencies and Assumptions:**

1. **Operating System**:
   * The system is designed to be OS-independent, meaning it can run on Windows, macOS, and Linux. However, development and testing are typically done on a Unix-like environment (Linux or macOS).
2. **Programming Languages**:
   * **JavaScript**: For both client-side (React.js) and server-side (Node.js) development.
   * **HTML/CSS**: For client-side rendering and styling.
3. **Frameworks and Libraries**:
   * **Frontend**:
     + **React.js**: For building the user interface.
     + **React Router**: For handling routing in the single-page application.
     + **Axios**: For making HTTP requests from the client to the server.
   * **Backend**:
     + **Express.js**: For building the server-side RESTful APIs.
     + **Mongoose**: For interacting with MongoDB.
     + **Bcrypt.js**: For password hashing.
     + **Jsonwebtoken**: For user authentication.
     + **Cors**: For enabling Cross-Origin Resource Sharing.
   * **Development Tools**:
     + **Nodemon**: For automatic server restarts during development.
     + **Concurrently**: For running both the client and server concurrently during development.
4. **Database**:
   * **MongoDB**: Used to store user data, including emails, hashed passwords, and profile information.
5. **Network Configuration**:
   * The system requires a stable internet connection for accessing the MongoDB Atlas database.
   * Proper network configurations and permissions are needed, including IP whitelisting for MongoDB Atlas.

#### **Architecture Diagram**

+-------------------------------------------------------------+

| Frontend |

| +---------------------+ +-------------------------+ |

| | Register Component | | Login Component | |

| | | | | |

| | | | | |

| +---------+-----------+ +------------+------------+ |

| | | |

| | HTTP Request (Axios) | |

| v v |

| +---------------------+ +-------------------------+ |

| | Profile Component | | Local Storage | |

| | | | (JWT Token) | |

| | | | | |

| +---------------------+ +-------------------------+ |

+-------------------------------------------------------------+

^ ^

| |

| |

v v

+-------------------------------------------------------------+

| Backend |

| +---------------------+ +-------------------------+ |

| | Express.js | | MongoDB | |

| | | | | |

| | | | User Collection | |

| +---------+-----------+ +------------+------------+ |

| | | |

| | Route Handling | |

| v v |

| +---------------------+ +-------------------------+ |

| | Auth Routes | | | |

| | (Register, Login) | | | |

| +---------+-----------+ +-------------------------+ |

| | |

| | Profile Routes |

| v |

| +---------------------+ |

| | Profile Controller | |

| +---------------------+ |

| |

| +---------------------+ |

| | Middleware (JWT Auth)| |

| +---------------------+ |

+-------------------------------------------------------------+