**Key Features to Prioritize**

1. **Frontend (React)**
   * Basic code editor integration.
   * User-friendly interface with essential features.
   * Real-time code editing for single-user sessions.
2. **Backend (Node.js)**
   * Basic API for code compilation and execution.
   * User authentication and project management.
3. **Database (MongoDB)**
   * Store user profiles, code snippets, and project data.
4. **Containerization (Docker)**
   * Simplified Docker setup for consistent development environments.
5. **Deployment**
   * Initial deployment on a cloud platform for user access.

**Revised Timeline and Milestones**

**Week 1-2: Initial Setup and Basic Functionality**

* **Project Setup**
  + Set up repositories (GitHub/GitLab).
  + Create initial Dockerfile and Docker Compose setup.
* **Frontend Development**
  + Initialize React project.
  + Integrate a code editor (e.g., Monaco Editor or CodeMirror).
  + Design basic UI (home page, code editor page).
* **Backend Development**
  + Initialize Node.js project with Express.
  + Develop basic API for code submission and execution.
  + Set up MongoDB and design basic schema for users and projects.

**Week 3-4: Core Features and Integration**

* **Frontend Enhancement**
  + Add user authentication (sign up, login).
  + Implement code editor features (syntax highlighting, basic real-time editing).
* **Backend Enhancement**
  + Implement user authentication and project management APIs.
  + Integrate language compilers/interpreters for JavaScript, Python, Java, and C/C++.
* **Database Integration**
  + Complete CRUD operations for user data, code snippets, and projects.

**Week 5-6: Real-Time Collaboration and Security**

* **Real-Time Features**
  + Implement WebSocket or similar for real-time code updates.
* **Security**
  + Implement basic security measures (input validation, secure storage).

**Week 7-8: Testing, Optimization, and Deployment**

* **Testing**
  + Write and run unit and integration tests for frontend and backend.
* **Optimization**
  + Optimize performance and scalability.
  + Ensure Docker containers work seamlessly.
* **Deployment**
  + Set up CI/CD pipeline for automated testing and deployment.
  + Deploy the application to a cloud platform (AWS, Azure, Google Cloud).
* **User Onboarding**
  + Create basic documentation for users.
  + Onboard initial users and gather feedback.

06/26/2024 (**ADDITIONAL FEATURES** to be implemented before deployment)

1. When I am signed in, it should replace the top two buttons (Sign In and Sign Up) and replace them with my first name with a profile icon and when I click on that icon it should show a big and nicely integrated log-out button. After I click that log out button, it should log me out of the application and it should re direct me to the home page that has the Sign in and sign up button. [COMPLETED]
2. When I sign up successfully, it sends me to home page as of now, but I want it to send me to sign in page. [COMPLETED]
3. If I am not signed in, it should not allow me see my projects. [COMPLETED]
4. If I am not signed in, it should not allow me to save the project and it should redirect me to “Sign In” page. Be very careful here because if I successfully sign in AFTER it redirects me, my contents in the code editor must not be lost. [COMPLETED]
5. If I am signed in, it should display list of projects associated with current signed in user and that projects must be clickable. When the project is clicked, it should be opened in current code editor. [COMPLETED]
6. Whenever I try to save a project with the name that already exists, it should show a message “File already exists, do you want to replace?” if yes is clicked then replace (update), else don’t replace. [COMPLETED]
7. Design the save project button and project name nicely.

[COMPLETED]

1. Design the My Projects page and design the whole UI/UX.

[COMPLETED]

1. Finalize the project.

[COMPLETED]

1. **Timeline Review**
2. **Documentation**
3. **Deploy**
4. **Git hub upload**
5. **Post on linked In**