Action Plan

Week 3

Goals:
Establish the foundation of the app and ensure clear project organization. Develop static versions of core UI components. Define detailed requirements and the app's feature scope.
Tasks:
Set up the React + TypeScript environment, including tools like React Router and Tailwind CSS.
Design a folder structure for components, pages, utilities, and assets.
Build static versions of key UI components (Dashboard, Task Board, Messaging).
Create wireframes and a detailed functional specification for the app.
Set up a GitHub repository with an organized README file and initial commits.
Extra Resources:
https://tailwindcss.com/docs/installation
https://reactrouter.com/web/guides/quick-start

Week 6

Goals:
Implement backend support for task management and user authentication. Begin integrating backend functionality into the frontend.
Tasks
Set up Firebase or Node.js backend with a database for task storage.
Implement CRUD functionality for tasks (create, update, delete, mark complete).
Build reusable task components connected to backend data.
Implement user authentication using Firebase Authentication or another service.
Test the task management flow with authenticated users
Extra Resources:
https://firebase.google.com/docs/auth
https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

Week 9

Goals: Implement visual progress tracking features to improve task overview. Enhance the app's interactivity with charts and drag-and-drop functionality.
Tasks:
Build a Kanban-style task board using a library like `react-beautiful-dnd`.
Create visual progress tracking components, such as charts using Chart.js or D3.js.
Add functionality to filter tasks by status, priority, or team member.
Enhance the UI for task board interactions, including drag-and-drop for task management.
Test the visual components for clarity and responsiveness.
Extra Resources:
https://github.com/atlassian/react-beautiful-dnd

https://www.w3schools.com/js/js_graphics_chartjs.asp

Week 12

Goals: Add advanced user management features and permission control. Integrate notifications and alerts for time-sensitive task updates. Begin making the app mobile-responsive.
Tasks:
Implement role-based access control, creating roles of user, admin, etc.
Set up time-sensitive notifications using Firebase Cloud Messaging.
Test user roles and permissions to ensure proper access control.
Implement responsive design principles using Tailwind CSS
Extra Resources:
https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design

Week 15

Goals: Enhance security features such as data privacy and data protection. Begin app optimization for better performance using API calls and front-end rendering
Tasks:
Implement encryption for sensitive user data.
Create secure HTTP headers and implement two-factor authentication to improve security.
Add role-based permission for data access so only authorized users can access certain data.
Optimize backend API calls with memory caching.
Work on minimizing the frontend bundle size by using code-splitting
Extra Resources:
https://owasp.org/www-project-top-ten/
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

Week 18

Goals: Clean up the app by doing user acceptance testing and gathering feedback. Ensure overall app quality on web and mobile for peak performance.
Tasks:
Conduct user acceptance testing with a group of beta users and refine the UI/UX based on feedback
Improve loading speed and server response time.
Set up automated testing such as unit tests to ensure the app’s functionality.
Prepare deployment pipelines for both the backend and frontend GitHub Actions or CircleCI.
Begin final preparations including writing user documentation and setting up app hosting, using Netlify for frontend and Firebase for backend.
Extra Resources:
https://jestjs.io/
https://circleci.com/docs/