Project Code with Short Explanations

3. vite.config.js

This code sets up a Vite configuration for a project, integrating React and Tailwind CSS as plugins.

5. backend/config/db.js

This code defines an asynchronous function `connectDB` that connects to a MongoDB database using Mongoose, logging the success or failure to the console. It exports `connectDB` as the default export.

2. index.html

This HTML document sets up a basic structure for a Vite + React application, including essential meta information and a script entry point. It prepares the environment to render the React application inside a `div` with the id `root`.

4. backend/index.js

This code sets up an Express server that connects to a MongoDB database, logs incoming requests, and serves authentication and plan-related routes. It listens on port 5000 and uses middleware for JSON parsing, CORS, and environment variable configuration.

1. eslint.config.js

This code sets up ESLint configuration for a project, applying recommended settings for JavaScript, React hooks, and Vite's React refresh, while ignoring the 'dist' directory and customizing rules for unused variables. It targets JavaScript and JSX files, using the latest ECMAScript features and browser global variables.

6. backend/models/User.js

This code defines a Mongoose schema for a User model with fields for name, email, and password, all of which are required, and the email must be unique. It then exports the User model.

7. backend/routes/auth.js

This code sets up Express routes for user registration and login, handling data validation, user existence checks, and response statuses. It also logs relevant information and errors during the process.

10. src/App.jsx

This code sets up a React application with routing using `react-router-dom`, defining routes for authentication, user details form, and a dashboard.

9. src/App.css

This CSS code styles a webpage layout with a centered container, logo animations, and styling for cards and links. It includes hover effects for the logo and a spinning animation for one of the logos, with adjustments for users who prefer reduced motion.

8. backend/routes/plan.js

This code sets up an Express router that, when POSTed to `/generate-plan`, uses Google Generative AI to create a personalized fitness and diet plan for an Indian user based on provided details. The response is formatted in Markdown and includes a workout routine, meal plan, and lifestyle tips.

12. src/main.jsx

This code sets up a React application, rendering the `App` component within React's StrictMode to the DOM element with the ID 'root'. It also imports a CSS file, likely configured for styling with Tailwind.

11. src/index.css

This code sets up global styles for a web page, including typography, colors, and button styles, with specific adjustments for dark and light color schemes. It also includes Tailwind CSS for utility-first styling.

13. src/components/AuthForm.jsx

This component renders a form for user authentication, allowing users to either sign up or log in, and handles form submission by communicating with a backend API. It also toggles between sign-up and login modes based on user interaction.

15. src/pages/Dashboard.jsx

This React component renders a dashboard displaying a personalized fitness plan, including workout routines, meal plans, and lifestyle tips, with navigation and sidebar functionality. It parses and displays sections of a markdown plan and allows users to navigate through different workout days.

14. src/components/Sidebar.jsx

The code defines a `Sidebar` component for a React application, featuring navigation links for a dashboard, meal plan, workout plan, and progress tracking, with a mobile-friendly toggling mechanism. It uses mock icons and navigation links, and includes an overlay for mobile usability, closing the sidebar on link click or overlay tap.

16. src/pages/UserDetailsForm.jsx

This React component renders a form for users to input personal details and generate a personalized plan, which can be previewed, finalized, or edited. It uses state management, form handling, API calls, and conditional rendering to achieve this functionality.