Web Application Productivity System For Students with ADHD: Project To-Do List

---

1. Initial Setup & Configuration:

- ☐ Set up the development environment:
    - ☐ Install Node.js and npm.
    - ☐ Initialize a new project using `npm init`.
- ☐ Set up a Git repository for version control:
    - ☐ Initialize a new repository.
    - ☐ Create a `.gitignore` file for Node.js.
    - ☐ Commit the initial project structure.

2. Backend Development:

- **2.1. Setting Up Express Server:**
    - ☐ Install Express using `npm install express`.
    - ☐ Create an `index.js` (or `server.js`) file.
    - ☐ Set up a basic Express server.
    - ☐ Test the server on a local host.
- **2.2. Database Configuration:**
    - ☐ Install MongoDB.
    - ☐ Set up a MongoDB connection using Mongoose.
    - ☐ Create models for Users, Courses, Resources, Assignments, and Tests.
- **2.3. API Endpoints Creation:**
    - ☐ Create CRUD (Create, Read, Update, Delete) endpoints for Users.
    - ☐ Create CRUD endpoints for Courses.
    - ☐ Create CRUD endpoints for Resources.
    - ☐ Create CRUD endpoints for Assignments & Tests.
- **2.4. User Authentication:**
    - ☐ Install packages for authentication (e.g., `passport`, `bcrypt`).
    - ☐ Set up user registration and login routes.
    - ☐ Implement password hashing.
    - ☐ Implement JWT for token-based authentication.

3. Frontend Development (React.js):

- **3.1. React Setup:**
    - ☐ Use Create React App to initialize the frontend.
    - ☐ Set up routing using `react-router-dom`.
- **3.2. Components Creation:**
    - ☐ Create a layout component (header, footer).
    - ☐ Develop a login/register component.

- ☐ Design course addition and display component.
- ☐ Craft assignment and test tracking components.
- ☐ Formulate components to add, view, edit, and delete resources.
- **3.3. State Management:**
  - ☐ Use React's Context API or Redux for state management.
  - ☐ Set up state and reducers/actions for Users, Courses, Resources, Assignments, and Tests.
- **3.4. Connect Frontend with Backend:**
  - ☐ Use Axios to send and receive HTTP requests to/from the backend.
  - ☐ Connect registration and login components to backend authentication routes.
  - ☐ Link course, resource, assignment, and test components to their respective CRUD endpoints.

4. **Styling and User Interface:**

- ☐ Design a responsive and intuitive layout.
- ☐ Use CSS frameworks/libraries like Bootstrap or TailwindCSS for styling.
- ☐ Implement animations or transitions for better user experience (optional).

5. **Testing:**

- ☐ Write unit tests for backend routes using tools like `jest` or `mocha`.
- ☐ Create frontend component tests using `jest` and `react-testing-library`.
- ☐ Conduct end-to-end testing.

6. **Deployment:**

- ☐ Prepare the app for production (e.g., set NODE_ENV to 'production').
- ☐ Deploy the backend on a platform like Heroku.
- ☐ Build the React app for production and deploy on platforms like Vercel, Netlify, or directly on Heroku.

7. **Post Deployment & Maintenance:**

- ☐ Monitor server logs and application behavior.
- ☐ Address any bugs or issues that arise.
- ☐ Periodically update all dependencies.
- ☐ Seek feedback from users and make iterative improvements.

While the above list covers the main steps involved, the complexity and scope of your project may require adjustments. Remember, developing a full-fledged web application requires frequent reviews, testing, and iterations. It's essential to stay agile, embrace feedback, and be prepared for unexpected challenges.