Test Documentation

1. What Is Being Tested

This test suite covers both UI and API components of a basic full-stack web application that includes authentication and CRUD functionality for a set of items (e.g., a todo list / inventory list). The application consists of a React frontend and a Node.js + Express backend, with MongoDB as the data store.

Frontend/UI Testing (via Cypress):

- Login functionality using valid and invalid credentials
- Creating a new item
- Editing an existing item
- Deleting an item
- Verifying the presence of updated data after actions

Backend/API Testing (via Postman):

- POST /login for authentication and token issuance
- GET /items to retrieve all items
- POST /items to create new items
- PUT /items/:id to update existing items
- DELETE /items/:id to remove items

Each endpoint has been tested for both positive and negative flows, meaning tests were written for expected behavior as well as for edge cases like invalid credentials, malformed requests, and incorrect IDs**.**

2. Test Coverage Areas

UI Test Coverage:

- Form input behavior (login and item creation)
- Button functionality and visibility
- DOM content changes after actions (e.g., item added/edited/deleted)
- User feedback or redirect flows
- Full item lifecycle simulation through interface interaction

API Test Coverage:

- Auth validation and token handling
- CRUD operations on the /items resource
- Server response codes and error handling
- Authorization headers and request validation
- MongoDB integration at a behavioral level (e.g., data created/deleted)

3. Tools Used & Reasoning

- **Cypress** (UI Automation): Chosen for its ease of setup with React, real-time interactive GUI, and tight integration with the DOM. It was used to simulate user behavior and validate end-to-end flows.
- **Postman** (API Automation): Chosen for its mature interface, easy scripting, and widespread use in API testing. Tests include pre-request scripts and assertions using Postman's testing framework.
- **MongoDB** (via Mongoose): Used as a lightweight NoSQL DB for storing items and user data. Not directly tested, but all backend routes interact with it.
- **Node.js + Express**: The backend was built with this stack and all route-level behavior was tested accordingly.

4. How to Run the Tests

To run the UI tests (Cypress):

1. Start the backend (npm run dev in **/server**)
2. Start the frontend (npm run dev in **/client**)
3. In a separate terminal, run: npx cypress open and click the spec file to execute

To run the API tests (Postman):

1. Start the backend (npm run dev in **/server**)
2. Open the Postman file provided*****
3. Make sure localhost:3001 is running
4. Click "Run Collection" and Postman will handle the rest

Note*: Authentication token is handled in test flows using Postman's scripting capabilities to carry tokens between requests, after introducing it once. It does have a time limit of 1 hour and it is recommended you run POST/login to get the updated token for the rest of the collection.

5. Assumptions and Limitations

- It is assumed that the user running the tests already has **Node**, **npm**, and **MongoDB** installed and configured locally.
- No advanced UI styling or animations were implemented. UI tests operate on raw buttons and forms as-is.
- The login system is hardcoded (**admin/admin**) and token-based auth is assumed to be consistent across sessions.
- Error messages and validation on the frontend are minimal as focus was kept on functional coverage, not UX polish.

- There is no separate test database. All tests operate on live MongoDB collections.
- There is only one possible user and re-logging in requires exit of page and re-entry as no log out button is yet implemented