Test Plan – Todo App (React + Node.js)

What is Being Tested

This project consists of a simple Todo web application with:

- A React frontend (built with Vite)
- A Node.js + Express backend API

I test both UI interactions and API endpoints, focusing on:

- Functional flows (login, create/edit/delete todos)
- Validation of successful and failed actions
- Data consistency between actions and visual/API results

Test Coverage Areas

1. UI Automation (Playwright)
    - Login with valid & invalid credentials
    - Create a new todo item
    - Edit an existing item
    - Delete an item
    - Assertions after each action:
        - New item appears in list
        - Edited item shows updated title
        - Deleted item no longer appears
2. API Testing (Supertest + Jest)
    - POST /login – success + failure
    - GET /items – retrieve todos
    - POST /items – create item
    - PUT /items/:id – update existing
    - DELETE /items/:id – delete existing
    - Negative tests: invalid item IDs, bad payloads
3. Code Coverage (UI + API)
    - API: Full coverage with Istanbul via Jest

Tools Used & Why

Tool	*Purpose*	*Why it was chosen*
Playwright	UI functional testing	Fast, powerful, handles modern React well
Supertest	API testing for Express apps	Native integration with Node.js apps
Jest	Test runner for backend	Mature, fast, supports coverage
GitHub Actions	CI pipeline	Automates testing on push/PR
Vite	Frontend build tool	Modern, fast dev server and plugin support

How to Run the Tests

Locally

1. Backend API Tests

```
cd backend
npm install
npm test                    # for tests
npm test -- --coverage      # with code coverage
```

2. Frontend UI Tests

```
cd frontend
npm install
npx playwright install
npx playwright test         # run UI tests
```

In GitHub Actions (CI)

Tests run automatically on:

- Every push to main
- Every Pull Request targeting main

It runs:

- Backend API tests with Jest
- Frontend UI tests with Playwright
- Uploads failed test artifacts (screenshots, videos)

Assumptions & Limitations

- App uses in-memory data (todos reset when server restarts)
- No real authentication/session management (login is mocked for demo)
- No database or persistent layer → limited data state across test runs
- Tests do not cover edge cases like:
    - XSS or security attacks
    - Very large datasets
    - Multiple simultaneous users
- Code coverage only reflects executed frontend code, not full UX