Test Plan & Test Strategy Document

# 1. Test Plan

## 1.1 What is being tested?

This document outlines the testing strategy for a full-stack Todo application consisting of a React frontend and a Node.js/Express backend. The key areas being tested include:  
- Authentication: Valid and invalid login scenarios  
- Todo Operations: Fetching, adding, editing, and deleting todos  
- UI Testing using Selenium  
- API Testing using Mocha, Chai, and Supertest

## 1.2 Test Coverage

|  |  |  |
| --- | --- | --- |
| Module | Coverage | Type |
| Login | Valid & invalid login | UI + API |
| Todo List | Get todos | UI + API |
| Add Todo | Create todo from UI & backend | UI + API |
| Edit Todo | Update a todo item | UI + API |
| Delete Todo | Delete todo item | UI + API |
| Error Handling | Invalid credentials, missing fields | UI + API |

# 2. Test Strategy

## 2.1 Tools Used and Why

|  |  |  |
| --- | --- | --- |
| Tool | Purpose | Reason |
| Mocha | Test runner for JS | Simple, widely used, supports async tests |
| Chai | Assertions | Human-readable assertions |
| Supertest | HTTP testing for Express APIs | Easily test routes with assertions |
| Selenium WebDriver | End-to-end browser automation | Simulates real user interactions |
| Postman | Manual API testing | Easy manual inspection & exploration |
| GitHub Actions | CI/CD integration | Automate test execution |

## 2.2 How to Run the Tests

Backend API Tests:  
1. Make sure backend is NOT running (Supertest will use the app instance).  
2. Open terminal in /backend folder. Use command cd backend press enter  
Use: node app.js  
3. Run: npm install  
4. Run: npx mocha api.test.js

UI Tests (Selenium):  
1. Start your frontend app (e.g., npm start).  
2. Run Selenium tests from /test folder: npx mocha tests/login.test.js

## 2.3 Assumptions and Limitations

Assumptions:  
- Backend runs at http://localhost:5000  
- Frontend is accessible via http://localhost:3000  
- Dummy user exists (test@example.com / password123)  
- Todos are stored in-memory

Limitations:  
- No persistent database integration  
- Tests depend on static selectors and response formats  
- No performance or load testing included  
- Authentication is mock-based, not token-based

**2.4 Use postman for manual API testing.**

Below are the test scenarios:  
Before testing make sure backend is running. Use below command to run backend.  
node app.js  
You will see: Backend running on <http://localhost:5000>  
  
1 – Login Endpoint  
POST = <http://localhost:5000/api/login>  
Go to body (JSON) and paste.  
{

"email": "test@example.com",

"password": "password123"

}

2 – GET Todos.  
GET = <http://localhost:5000/api/todos>

[

{

"id": 1,

"text": "Learn Selenium"

}

]

3- Add Todo

POST = <http://localhost:5000/api/todos>  
{

"text": "Write API test cases"

}

4 – Edit Todo

PUT = <http://localhost:5000/api/todos/:id>  
here replace id with ID  
  
{

"text": "Updated Todo Text"

}

5- Delete Todo  
DELETE = <http://localhost:5000/api/todos/:id>