# **Test Plan / Test Strategy Document**

#### 1. Overview

This document outlines the testing approach for the Automation -project- app, a simple Todo application that includes:

- A React frontend for user interaction
- A Node.js backend API for data operations

Testing ensures that core workflows like login and item management work as expected through automated UI and API tests.

### 2. What Is Being Tested

**Functional Testing:** 

- Login with valid credentials
- Reject login with invalid credentials
- Add, edit, delete todo items
- Ensure list updates accordingly

#### API Validation:

- Verify status codes and JSON responses
- Handle invalid data gracefully

### 3. Test Coverage Areas

Login: Valid and invalid login flows

Item Creation: Adding new tasks

Item Editing: Modifying existing tasks

Item Deletion: Removing tasks

API Response Verification: Status, errors, and data formats

## 4. Tools Used and Why

React - UI development (component-based and maintainable)

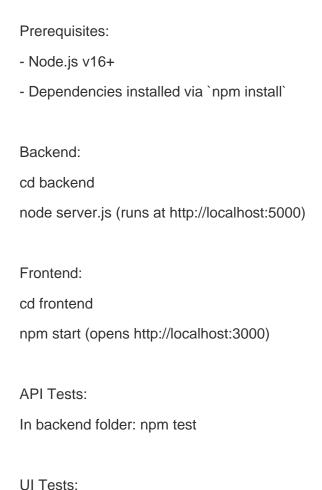
Node.js + Express - API backend (simple REST server)

## **Test Plan / Test Strategy Document**

Cypress - UI test automation (fast and reliable)

Jest + Supertest - API testing (easy HTTP request testing)

#### 5. How to Run the Tests



## 6. Assumptions and Limitations

In frontend folder: npx cypress open

- Backend uses in-memory data (not persistent)
- Hardcoded login: username='admin', password='123'
- Localhost-only testing (no deployment environment assumed)
- No database integration or user roles
- No performance, security, or accessibility testing yet
- UI tests run only on Cypress's built-in browser