**Test Strategy Document – Full Stack Test Automation**

**1. Objective**

This document outlines the test strategy for automating both the **frontend (React-based UI)** and **backend (Node.js API)** of the application. The goal is to ensure functional correctness, reliability, and consistency across UI and API layers.

**2. Scope of Testing**

**What is Tested**

* UI workflows: Registration, login, navigation, and actions via Selenium (Click, send keys, get)
* Backend API endpoints: Authentication and CRUD operations using Postman/Newman.

**What is Not Tested**

* Non-functional testing -Load/Stress/Performance
* End-to-end integration across 3rd party systems.
* Database or server-side logic not exposed via APIs/UI.

**3. Test Coverage**

**UI Tests (Selenium - Java)**

|  |  |
| --- | --- |
| **Module** | **Test Scenarios** |
| Login/Register | Valid/invalid credentials, form validations |
| Navigation | Clicking links, validating redirected pages |
| Actions | Button clicks, input forms, image/logo interactions |

**API Tests (Postman/Newman)**

|  |  |  |
| --- | --- | --- |
| **Endpoint** | **Positive Cases** | **Negative Cases** |
| POST /login | Valid credentials → 200 OK | Invalid → 401 Unauthorized |
| GET /items | Retrieve list → 200 OK | Unauthorized → 403 |
| POST /items | Valid item → 201 Created | Missing fields → 400 Bad Request |
| PUT /items/:id | Valid update → 200 OK | Invalid ID → 404 Not Found |
| DELETE /items/:id | Valid deletion → 204 No Content | Unauthorized → 403 / Invalid ID → 404 |

**4. Tools Used**

|  |  |
| --- | --- |
| Tool | Purpose |
| Selenium | Automate browser-based UI interactions in React application. |
| TestNG | Test framework for structuring Selenium test cases. |
| Postman | Build, run, and debug API test cases manually. |
| Newman | Execute Postman collections via CLI and CI/CD pipelines. |
| Maven | Manage dependencies and test execution lifecycle. |

**5. How to Run the Tests**

**UI Tests (Selenium):**

1. Clone the project and import into your IDE
2. Ensure ChromeDriver is set up and linked in the project.
3. Use Maven to build and run

**API Tests (Postman/Newman):**

1. Import collection into Postman and test manually (for development).
2. For CLI/CI use:

newman run collection.json -e environment.json --reporters cli,html

**6. Assumptions & Limitations**

* Environment URLs and credentials are pre-configured.
* IDs used in update/delete API tests are either hardcoded or generated dynamically.
* Frontend is accessible in a stable environment during test execution.
* Browser compatibility testing is limited to Chrome.
* No mocking/stubbing tools used; tests run against live systems.