■ Test Plan – QA Assessment

1. Project Description

This QA project is based on the cloned GitHub repository: Full-Stack FastAPI Template. The application includes: - Backend (Dockerized API + Swagger) - Frontend (React) - Database (Postgres) I selected this project because it provides a real-world full-stack environment with authentication, CRUD operations, and Docker-based deployment, which are ideal for QA testing.

2. Test Scope & Objectives

The goal is to validate critical user flows, ensure correct error handling, and verify integration between frontend and backend. Focus is on registration, authentication, and password recovery, as these are business-critical features.

3. Test Approach

- Manual Testing → Smoke testing of main features before automation - Automated Testing → Using Cypress (TypeScript) and Swagger for API testing

4. Test Environment

The environment is based on Dockerized backend (FastAPI), frontend (React), and Postgres DB. Detailed setup instructions are documented in the project README file.

5. Critical Test Cases

1. User Registration with valid details
2. User Registration with empty fields
3. User Registration with invalid email
4. User Registration with mismatched passwords
5. User Registration with password < 8 chars
6. User Login with valid credentials
7. User Login with empty fields
8. Password Recovery with existing email
9. Password Recovery with non-existing email
10. Password Recovery with empty email field

6. Risk Assessment & Prioritization

Tests were prioritized based on business impact and risk level: - HIGH Priority → User registration, login, password recovery (core business flows) - MEDIUM Priority → Validation errors (invalid email, mismatched passwords, empty fields) - LOW Priority → UI-only validations (button states, messages visibility) Risks include: - Users unable to register/login → Business critical (High risk) - Wrong error messages → User confusion (Medium risk) - Multiple requests from spamming submit button → Performance impact (Medium risk)

7. Defect Reporting Procedure

All identified defects are reported and tracked in Jira. Each bug report includes steps to reproduce, expected vs. actual results, severity, priority, suggested fix, and assignment to the responsible team. Jira Project Link: Bug Tracking Board