Test Plan for Notes API

Author: Svitlana Koshchii
Created On: 2025-01-31
Project: API Testing using Postman & Newman

Table of Contents

1. Introduction

This document outlines the test plan for the Notes API, tested using the Postman collection "NotesAPI.postman_collection.json" and the environment file "NotesAPIenvironment.postman_environment.json".

1.1 Objective

The goal is to validate the functionality, reliability, performance, and security of the Notes API by executing predefined test cases in Postman.

1.2 Revision History

Version	Date	Author	Modification
0.1	2025-01-31	Svitlana Koshchii	Test plan document created

2. Scope

The tests will cover the following areas:

- API availability (Health Check)
- User Management (Registration, Login, Logout, Profile Management, Deletion)
- Notes Management (CRUD operations on notes) - **TO DO**
- Response validation (status codes, headers, JSON structure)
- Performance (response time thresholds)
- Security checks (authentication, authorization, input validation, XSS)

2.1 Test Environment

- Base URL: {{baseURL}} (configured in the Postman environment file)
- Tools: Postman, Newman, GitHub Actions (for CI/CD execution)
- Test Execution: Local Postman runner and GitHub Actions workflows

2.2 Supported Methods

The Notes API supports the following HTTP methods:

- **GET**: Retrieve resources such as user details and notes.

- **POST**: Create new resources, including user accounts and notes.
- **PUT**: Update existing resources like user profiles and notes.
- **DELETE**: Remove user accounts and notes.

2.3 Authentication

- The Notes API requires authentication for protected endpoints.
- Authentication is handled via a token-based system.
- Upon successful login, a token is returned in the response.
- The token must be included in the x-auth-token header for subsequent requests.
- Token expiration is managed by the API; expired tokens must be refreshed by re-authenticating.
- Unauthorized access attempts will return a 401 Unauthorized response.

2.4 Test Cases Overview

TC scenarios described in the table below

TC ID	Scenario	State	Expected Result
TC01	Health Check	Positive	API returns status 200, response contains "Notes API is Running"
TC02, TC03	Create User - Valid Data	Positive	Status 201, user details returned
TC04	Create User - Empty name	Negative	Status 400, validation error
TC05	Create User - Skipped name	Negative	Status 400, validation error
TC06	Create User - Skipped name	Negative	Status 400, validation error
TC06	Create User - Name length < 4	Negative	Status 400, validation error
TC07	Create User - Name length > 30	Negative	Status 400, validation error
TC08	Create User - Empty email	Negative	Status 400, validation error

TC ID	Scenario	State	Expected Result
TC09	Create User - Skipped email	Negative	Status 400, validation error
TC10	Create User - Already Existing email	Negative	Status 400, validation error
TC11	Create User - Email length > 500	Negative	Status 400, validation error
TC12	Create User - Email without mailbox name	Negative	Status 400, validation error
TC13	Create User - Email without domain	Negative	Status 400, validation error
TC14	Create User - Email without second level domain	Negative	Status 400, validation error
TC15	Create User - Email without top level domain	Negative	Status 400, validation error
TC16	Create User - Empty password	Negative	Status 400, validation error
TC17	Create User - Skipped password	Negative	Status 400, validation error
TC18	Create User - Password length < 6	Negative	Status 400, validation error
TC19	Create User - Password length > 30	Negative	Status 400, validation error
TC20	Create User - Request with empty body	Negative	Status 400, validation error
TC21	Create User - With name, empty email	Negative	Status 400, validation error
TC22	Create User - With name and email, empty password	Negative	Status 400, validation error
TC23	Login - Valid Credentials	Positive	Status 200, error message
TC24	Login - Without email	Negative	Status 401, error message

TC ID	Scenario	State	Expected Result
TC25	Login - Without password	Negative	Status 401, error message
TC26	Login - Invalid Credentials	Negative	Status 401, error message
TC27	Delete User Account	Positive	Status 200, account removed
TC28	Delete User Account which doesn't exist	Negative	Status 401, validation error
TC29	Delete User without token	Negative	Status 401, validation error

3. Test Execution Strategy

- Functional tests will be executed manually in Postman and automated in GitHub Actions using Newman.
- Performance tests will monitor response times (e.g., API should respond in <500ms for health checks).
- Security tests will include authentication checks and payload validation for potential XSS (TO DO - add XSS TCs).
- Regression testing will be done upon API changes to verify no functionality is broken.

4. Test Data Management

- Test users and notes will be created dynamically.
- Cleanup scripts will run to remove test data after execution.

5. Reporting & Logs

- Test execution results will be captured in Postman reports.
- GitHub Actions workflow will generate an HTML report as an artifact.
- Logs will be reviewed for API errors or failures.

6. Deliverables

The following artifacts will be delivered as part of the testing process:

- Test Plan document (this document)

- Postman Collection (NotesAPI.postman_collection.json)
- Postman Environment (NotesAPIenvironment.postman_environment.json)
- Test Execution Reports (generated from Postman/Newman runs)
- Logs and error reports for failed tests
- HTML test reports from GitHub Actions artifacts
- Summary of defects and issues found

7. Exit Criteria

Testing is considered successful when:

- All critical test cases pass without major defects.
- API meets functional and performance expectations.
- No high-severity security vulnerabilities are detected.

8. Risks & Mitigations

- **Risk**: API downtime during testing.
  - *Mitigation*: Schedule tests during non-peak hours, retry failed requests.
- **Risk**: Authentication failures due to expired tokens.
  - *Mitigation*: Refresh tokens before test execution.

9. References

- API Documentation: Notes API Docs
- Postman Collection: NotesAPI.postman_collection.json
- Postman Environment: NotesAPIenvironment.postman_environment.json

10. Glossary

- **API (Application Programming Interface)**: A set of rules that allow different software entities to communicate.
- **CRUD (Create, Read, Update, Delete)**: Basic operations for managing resources in an application.
- **CI/CD (Continuous Integration/Continuous Deployment)**: A set of practices for automating testing and deployment.
- **TC(Test Case):** A single step or a sequence of steps to test the correct behavior/functionality and features of an application

- **XSS (Cross-Site Scripting)**: A security vulnerability where malicious scripts are injected into web applications.
- **JWT (JSON Web Token)**: A compact, URL-safe means of representing claims between parties, often used for authentication.
- **Postman**: An API development tool used for testing and documentation.
- **Newman**: A command-line tool to run Postman collections in an automated environment.