BLOG POST APPLICATION - MASTER TEST PLAN

Software Quality Engineering

(SE-G)

Submitted by

Amna Shahid (21I-1148)
Fatima Qurban (21I-1195)
Ushna Nadeem (21I-1225)

Submitted to

Ma’am Saba Kanwal

Table of Contents

Test Plan Identifier

*Test plan identifier for this "Blog Post Application" is **BPA_01.1**.*

This test plan outlines the testing strategy for the Blog Post Application. It incorporates both manual and automated testing techniques and addresses black-box and white-box testing as part of the overall quality assurance effort.

References

- Requirements specifications which were given before the execution of the project.
- Blogging Platform API Documentation.

Introduction

This is the Test Plan for the ***Blog Post Application*** project. The plan consists of the items, directly or indirectly, relevant to the Blog Post Management process which would include User management and Blog management. The objective of this document is to make sure that Blog Post Application is working according to the defined functionality. We will be doing Unit Testing, System Testing and Acceptance Testing. Each type of testing would be discussed in detail in their respective sections.

The completion time for this project is 2 weeks. No critical delays and third party verifications are anticipated to have a direct impact on the test plan. All the details about the plan and each specific step taken will be discussed further in the plan.

Test Items

The following is a list of the items to be tested:

- **User Authentication**
  - Login and logout functionality.
  - Session management and expiration.
- **Blog CRUD Operations**
  - Create, Edit, Update, and Delete operations for blogs.
  - Input validation and handling of edge cases.
- **Business Logic**
  - Validation workflows (e.g., content length, prohibited characters).
  - Filtering by tags or search keywords.

- **Reports/Outputs**
  - Blog listing and pagination.
  - Search and filter results accuracy.
- **API Endpoints**
  - Endpoints for Create, Read, Update, and Delete.
  - Response validation and performance testing.
- **Database Connectivity and Integrity:**
  - Proper execution of database transactions.
  - Data consistency and rollback scenarios.

Software Risk Issues

- Database failures during CRUD operations.
- Potential security vulnerabilities in user input fields (SQL injection, XSS).
- Poor performance with large datasets.

Features to be Tested

- Users can add a blog with title, body, and optional tags.
- Blogs display correctly on the homepage and detail page.
- Users can edit their blog entries.
- Users can delete blog entries.
- Integration with database (CRUD operations tested thoroughly).
- Validation on user input (e.g., mandatory fields).
- Responsiveness of the application on mobile and desktop.

Features not to be Tested

The features which are not being tested here are:

- Advanced styling or non-critical CSS issues.
- UI/UX designs or aesthetics.

Approach

The **Blog Post Application** is a web-based software project designed to enable users to create, edit, view, delete, and manage blog posts efficiently. It provides essential content management features while integrating multiple functionalities like user authentication, CRUD operations, and content filtering. The application also incorporates business logic to ensure data integrity and supports generating meaningful reports or outputs.

Testing Levels

Unit Testing

Objective: Validate individual components and modules of the application to ensure they work as intended.
Scope:

- Backend logic: Validation rules, data processing, and business logic workflows.
- API endpoints: CRUD operations to verify proper data handling.
- Utility functions: String formatting, date parsing, and other helper functions.

System Testing

Objective: Test the integrated application to ensure end-to-end functionality works as expected.
Scope:

- Integration of frontend, backend, and database.
- User workflows, such as logging in, creating blogs, and viewing filtered results.

Acceptance Testing

Objective: Validate that the application meets user requirements and is ready for deployment.
Scope:

- Real-world user scenarios based on specified requirements.
- Usability, accessibility, and performance from an end-user perspective.

Test Tools

Following tools will be used:

- **Postman**: for testing the API Routes and integration with databases.
- **Cypress**: for testing the web application (Blog Post Application).
- **Istanbul**: for checking the coverages of statements etc from the React code.
- **Jira:** for defect tracking in the test cases.

Meetings

The test team will have one meeting before the start of the testing process and discuss all the requirements. After the testing phase a final meeting will be held to check whether the work done and tasks achieved are up to the mark or do they need any amendments.

Measures and Metrics

Some criterias for testing the application
Code Coverage (for White-Box Testing):

- **Types:**
  - **Statement Coverage**: (Number of Executed Statements / Total Number of Statements) * 100
  - **Decision Coverage:** (Number of Executed Decision Paths / Total Decision Paths) * 100

Automation Execution Time Savings:

- Compare the time taken for manual vs. automated execution of test cases.
- Its purpose is to quantify the efficiency gains from automation.

Item Pass/Fail Criteria

The basic criteria for the items to pass or fail is:

- **Functional:** All features must meet functional requirements with no critical or major defects.
- **Performance:** Response times under a specified time while testing, for API calls and page loads.
- **Security:** No high-severity vulnerabilities detected during testing.
- **Usability:** All critical workflows validated and user-friendly.

Suspension Criteria and Resumption Requirements

Suspension:

Testing will stop if

- A critical feature fails repeatedly.

- The testing environment becomes unavailable.

Resumption:

The will only resume after:

- Defects are fixed and verified.
- The environment is restored.

Test Deliverables

Following are the deliverables of the tests being executed:

- Test cases and scripts.
- Test execution reports.
- Defect reports.
- Final static analysis report.
- Automation testing scripts.

Remaining Test Tasks

TASK	Assigned To	Status
Finalize test cases	Test Manager, Testers	In Progress
Execute all planned tests	Testers	Not Started
Log and resolve defects	Testers, Dev	Ongoing
Retest fixed defects	Testers	Not Started
Prepare and deliver test summary	Test Manager, Testers	Not Started

Environmental Needs

Test environments:

- Local machine for development testing.

- Staging server mimicking production.

Database: Access to the application's test database, which is MongoDB.
Tools: Browser compatibility matrix for UI tests.

Staffing and Training Needs

The staff must have the basic knowledge of the tools for testing like Postman, Cypress, Istanbul, Jira. If they are not that proficient, they must be trained first on project-specific tools and workflows.

Responsibilities

Following roles are there for distribution of responsibilities:

- **Test Manager:** Approve test plans and supervise testing.
- **Developers:** Resolve defects promptly.
- **Testers:** Execute test cases and log defects.

Schedule

This is the planned schedule for this testing phase:
Test Planning: 1 day.
Unit Testing: 2 days.
Integration Testing: 1 day.
System Testing: 1 day.
Acceptance Testing: 1 day.

Planning Risks and Contingencies

These are the basic risks for which we must be prepared:

- **Risk**: Delays in environment setup.
    - **Solution**: Early setup and validation of environments.
- **Risk**: Limited availability of testers.
    - **Solution**: Utilize developer support for basic test execution.

Approvals

Before starting the testing phase, approval from the Test Lead, Project Manager and the Development Manager are required.

Glossary

- **CRUD**: Create, Read, Update, Delete
- **SQL**: Structured Query Language
- **API**: Application Programming Interface
- **UI**: User Interface