Test Plan Template

Title: "In our software testing and quality assurance mini project, we developed a dynamic website providing COVID-19 information, utilizing HTML, CSS, JavaScript, and PHP. We integrated a MySQL database to store user account, comment, and registration form details, and implemented regular expression test cases for testing purposes."

Prepared by:
Prasanna Munde
Shreyas Patil
Pratik Pawar

TABLE OF CONTENTS

1.0 INTRODUCTION

In the dynamic world of software development, we emphasize the importance of quality and reliability. Software Testing and Quality Assurance (QA) play vital roles in this process. Our mini project aims to create a dynamic website providing real-time COVID-19 information using HTML, CSS, JavaScript, PHP, and a MySQL database. Additionally, we'll implement regular expression test cases to ensure functionality and security.

2.0 OBJECTIVES AND TASKS

2.1 Objectives of Test Plan

- Functional Validation: Ensure the website accurately and effectively delivers real-time COVID-19 information..
- User Experience Testing: Evaluate the user interface for uniformity, responsiveness, and compatibility on various devices and browsers.
- Regular Expression Test Cases: Develop and run test cases employing regular expressions to confirm the validity of user inputs for registration and comments.

2.2 Tasks of Test Plan

- Develop the test strategy and define objectives and scope.
- Set up the test environment to mimic the production environment.
- Design test cases for functional, security, and usability testing.
- Execute test cases, including regular expression tests for user input.
- Document and report defects, conduct user acceptance testing, and evaluate performance and scalability.

Sr. No.	#Business Requirements (BR)	#Functional Requirements (FR)	#Test Scenarios (TS)	#Testing Approaches /Strategies (TA)
01	Ensure the website delivers accurate and timely COVID-19 information.	Display the most recent COVID-19 statistics on the homepage.	Validate the currency of displayed COVID-19 data by cross-referencing it with a trusted external source.	Testing for Data Validity, Comparison with External Data
02	Facilitate seamless user account creation, login, and access to personalized features.	Implement user registration and login functionality. Test user registration and login procedures, including credential validation, password resets, and account lockout handling.	Authentication Testing, User Registration Testing.	Authentication Testing, User Registration Testing
03	Verify the accuracy of user data storage in the MySQL database through the registration form.	Capture and save user registration data in the database. Verify that user information, such as username, email, and user type, is stored accurately in the database.	Confirm that user information, including username, email, and user type, is accurately stored in the database	Database Integration Testing, Data Validation Testing

3.0 SCOPE

General

This test plan encompasses various testing aspects, including functional testing to validate essential features, security assessments to safeguard against vulnerabilities, user experience testing, regular expression testing for data validation, scalability and load testing, and thorough documentation. It also encompasses user acceptance testing to guarantee a user-friendly experience. The plan emphasizes the website's reliability, security, and quality, with a strong focus on data integrity and user satisfaction. It's important to note that this test plan excludes testing specific to hardware components.

4.0 TESTING STRATEGY

The testing strategy for the Covid website Application aims to ensure comprehensive coverage of functional, security, compatibility, performance, and usability aspects. This will be achieved through a systematic and well-structured approach that includes the identification of feature groups, definition of testing tasks, and the use of appropriate techniques and tools.

Feature Groups and Testing Approaches:

1. Functional Testing
    - Feature Group: This pertains to User Registration, Dashboard, and Admin functionality.
    - Approach: We will create comprehensive test cases for each of these functions, execute them to ensure proper operation, and employ test automation tools for repetitive and regression testing.

2. Unit Testing:
    - Feature Group: It focuses on Functional Testing.
    - ·Approach: The approach involves conducting Functional Testing on each module, which means we test individual components or modules of the software in isolation.

3. Compatibility Testing
    - Feature Group: This encompasses Cross-Browser and Device Compatibility.
    - Approach : The approach taken involves assessing the software's compatibility across different browsers and devices to ensure a seamless user experience.

4.1 Functional Testing

Functional testing is a critical step in the software testing process, which aims to confirm that a system or application behaves in alignment with its defined requirements. This process thoroughly examines the different functions and capabilities of the software to guarantee their correct operation, free from glitches or unforeseen actions. The primary objective of functional testing is to ensure that the software aligns with the functional requirements as specified in its design documentation.

Participants:

1. Tester - Shreyas
2. Developer - Prasanna

3. Business Analysts - Pratik
4. Product Owners:
5. Quality Assurance (QA) Team

Methodology:
The process of functional testing is characterized by a methodical and organized approach, aiming for a comprehensive validation of software functionality. It commonly incorporates a series of specific steps to achieve this goal.
1. Requirements Analysis
2. Test Planning
3. Test Case Design
4. Test Execution
1. Defect Reporting
2. Regression Testing
3. Test Closure
4. Collaboration with Development:

TEST CASE ID	TEST UNIT/CLASS	TEST CASE	TEST STEPS	EXPECTED RESULT	ACTUAL RESULT	STATUS (PASS/FAIL)

TC001	COVID-19 Data Display	Verify COVID-19 Data Accuracy	1. Visit the website. 2. Compare data with an external source.	COVID-19 data on the website matches the external source.	COVID-19 data is accurate.	PASS
TC002	User Registration and Login	Test User Registration	1. Click on the "Register" button. 2. Fill out the registration form with valid information. 3. Click "Submit." 4. Check for successful registration.	User is successfully registered.	User is registered as expected.	PASS
TC003	User Registration and Login	Test User Login	1. Click on the "Log In" button. 2. Enter valid login credentials (email and password). 3. Click "Log In." 4. Check for successful login.	User is successfully logged in and redirected to the dashboard.	User is logged in as expected.	PASS
TC004	Registration Form Validation	Validate Registration Form	1. Fill out the registration form with invalid or missing information. 2. Click "Submit." 3. Check for error messages.	Appropriate error messages are displayed, and registration is not processed.	Registration form validation is effective.	PASS
TC005	Regular Expression Test Cases	Email Address Validation	1. Enter email addresses with varying formats (valid and invalid). 2. Submit each email for registration.	Valid emails are accepted, and invalid ones trigger appropriate error messages.	Email address validation is effective.	PASS

4.2 Unit Testing

Unit testing is a software testing approach focused on examining isolated individual units or components within a software application to validate their correct behavior. This means assessing the smallest testable sections of the software, often individual functions or methods, to confirm that each unit of code operates as anticipated.

Participants:

1. Testers:
2. Developers:
3. Business Analysts:
4. Product Owners:
5. Quality Assurance (QA) Team

Methodology:

1. **Understand Requirements:**
   - Start by gaining a thorough understanding of the functional and non-functional requirements for the Covid Website. This knowledge is essential for crafting appropriate test cases.
2. **Identify Testable Units**:
   Divide the website into manageable testable units, such as the Dashboard and New User Registration.
3. **Write Test Cases:**
   Create comprehensive test cases for each identified unit, encompassing a range of scenarios, including typical use cases, edge cases, and potential error conditions.
4. **Set Up Test Environment:**
   Establish a dedicated testing environment free from previous test data, ensuring a clean slate for the current testing efforts.
5. **Use Testing Frameworks:**
   Choose an appropriate testing framework corresponding to your programming language (e.g., JUnit for Java, PyTest for Python) to effectively structure and execute your tests.
6. **Arrange-Act-Assert (AAA) Pattern:**
   Organize each test case following the AAA pattern, which involves setting up preconditions, performing actions on the functionality, and confirming the expected outcomes.

4.3 Compatibility Testing

Compatibility testing is a software testing category that focuses on validating the proper operation of the application in diverse settings, devices, web browsers, and operating systems. The central objective is to confirm that the To-do list Application is accessible and maintains consistent performance across various configurations, ensuring a smooth and uniform user experience.

Participants:

1. Testers: Responsible for executing compatibility tests on various environments.

2. Developers: Collaborate with testers to address any compatibility issues identified during testing.
3. IT Administrators: Provide access to different environments and assist in configuring test environments.
4. Product Managers: Offer insights into the target audience and their typical configurations.

Methodology:

1. Environment Identification:
    - Identify the target environments for compatibility testing, including different browsers ,operating systems, and device.
    - Account for differences in screen resolutions, browser versions, and device types.
2. Test Environment Setup:
    - Establish test environments that closely replicate the anticipated user environments.
    - Ensure access to different operating systems, browsers, and devices for testing.
3. Cross-Browser Testing:
    - Test the application across major web browsers such as Chrome, Firefox, Safari, and Edge.
    - Verify that all features work consistently across different browsers.
4. Device Compatibility Testing:
    - Test the application on various devices, including desktops, laptops, tablets, and mobiles.
    - Consider different screen sizes, resolutions, and touch interactions.
5. Operating System Compatibility Testing:
    - Test the application on different operating systems such as Windows, macOS, and Linux.

4.4 Integration Testing

Integration testing is a software testing approach that involves the amalgamation of individual components or modules within a software system for collective testing to confirm their harmonious interaction. The primary aim of integration testing is to identify and resolve any potential problems that may arise from the interplay of different software components..

Participants:

1. Performance Testers: Specialized testers responsible for designing and executing performance and stress tests.
2. Developers: Collaborate with performance testers to optimize code and address performance bottlenecks.
3. IT Administrators: Provide access to server resources, monitor system metrics, and assist in setting up test environments.
4. Product Managers: Define performance criteria based on user expectations and business requirements.

Methodology:

1. Understand System Architecture:
   Attain a thorough grasp of the system architecture, including the interrelationships and interdependencies among various components. This understanding forms the foundation for the design of effective integration tests.

2. Identify Integration Points:
   Identify the specific junctures where various components interact. These integration points could encompass APIs, databases, messaging systems, or any communication channels connecting modules.
3. Define Integration Test Scope:
   Outline the extent of integration testing, including the precise components or modules that require integration and collective testing. Prioritize integration scenarios based on the significance of their functionalities.
4. Create Integration Test Plan:
   - Develop a detailed integration test plan outlining the objectives, test scenarios, test cases, and success criteria. Define the testing strategy, including whether to use a top-down, bottom-up, or incremental approach.
5. Identify Test Data:
   - Prepare test data that represents realistic scenarios for integration testing. Ensure that the data covers a range of inputs and conditions to validate different paths through the integrated components.

5.0 ENVIRONMENT REQUIREMENTS

5.1 Software Requirements:

- Web Browsers Chrome, Firefox, Safari, Edge, and other browsers relevant to the application's target audience.
- Operating Systems: Windows 11
- Database: MySQL database system used in the production environment.
- Server Software Xampp.

- Development Tools IDEs, version control systems (e.g., Git), and debugging tools.
- Testing Tools Selenium WebDriver for automated testing and security testing tools.

5.2 Test Environment Configuration:

- Configuration Files: Maintain configurations files for different environments (e.g., development, testing, and production).
- Environment Variables: Use environment variables to manage dynamic configurations.

Mobile Device Emulators/Simulators:

- Mobile Testing Tools: Emulators or simulators for testing on various mobile devices and platforms.

5.2 Hardware requirements

- Operating system: Windows 11
- System: Intel core i5
- Hard Disk: 1 TB
- Ram: 8 GB

6.0 TEST SCHEDULE

Incorporate the test milestones outlined within the Software Project Schedule, alongside all item transmittal events. Clearly outline any supplementary test milestones required. Provide time estimates for each testing activity, and establish a schedule for each testing task and milestone. For each testing resource, including facilities, tools, and staff, specify their designated periods of utilization.

Task Name	Start Date	Finish Date	Effort Estimation

Test Planning	18/09/23	18/09/23	1 hours
Review Requirements documents	20/09/23	20/09/23	1 hours
Create initial test estimates	21/09/23	21/09/23	0.5 hours
Staff and train new test resources	25/09/23	25/09/23	1 hours
First deploy to QA test environment	29/09/23	29/09/23	0.5 hours
Functional testing – Iteration 1	29/09/23	29/09/23	0.5 hours
Iteration 2 deploy to QA test environment	29/09/23	29/09/23	0.5 hours
Functional testing – Iteration 2	29/09/23	29/09/23	0.5 hours
System testing	30/09/23	30/09/23	0.5 hours
Regression testing	03/10/23	03/10/23	0.5 hours
User Acceptance Testing	03/10/23	03/10/23	0.5 hours
Resolution of final defects and final build testing	04/10/23	04/10/23	0.5 hours
Deploy to Staging environment	04/10/23	04/10/23	1 hours
Performance testing	06/10/23	06/10/23	0.5 hours
Release to Production	11/10/23	11/10/23	1 hours

7.0 FEATURES TO BE TESTED

1. **User Registration and Authentication:**
    - Users should be able to create accounts, log in, and reset their passwords securely.
2. **COVID-19 Data Display:**
    - Real-time COVID-19 data, such as cases, recoveries, and deaths, should be displayed for different regions and time frames.
3. **Information Articles:**
    - The website should provide informative articles on COVID-19, including prevention, symptoms, and guidelines.
4. **Search and Filtering:**

- Users should be able to search for specific COVID-19 information and filter data based on criteria like location and date.

5. **Data Sources and Updates:**
    - The website should cite data sources, and the data should be regularly updated to ensure accuracy.
6. **Mobile Responsiveness:**
    - The website should be mobile-friendly to accommodate users on various devices.

8.0 FEATURES NOT TO BE TESTED

Hardware-Specific Performance:

Reason: Testing the website's performance on specific user hardware configurations is impractical. Instead, performance testing focuses on general device responsiveness and may include lower-end hardware scenarios..

9.0 RESOURCES/ROLES & RESPONSIBILITIES

Team member details and work distribution

Sr No	Name	Roles and Responsibilities
1	Prasanna Munde	Development of backend of website, functional testing.
2	Shreyas Patil	Development of frontend of website, unit testing.
3	Pratik Pawar	Development of frontend of website, integration testing.