**STP Document**



**Table of Contents**

Planned schedule3

Purpose of this document4

Description Of the System 4

Glossary and Abbreviations6

Testing plan 8

Resources

Software/hardware needed

functional tests

non- functional tests

Starting and existing criteria10

**Traceability Table11**

Tests Tree 11

Hazards table 14

**Planned Schedule**

|  |  |  |
| --- | --- | --- |
| A step in the project process | Start date | End date |
| Preparation of STP document | **08.03.2024** | **08.03.2024** |
| Preparation of STD document | **08.03.2024** | **08.03.2024** |
| Round of tests #1 | **08.03.2024** | **08.03.2024** |
| Round of tests #2 | **08.03.2024** | **08.03.2024** |
| Round of tests #3 | **08.03.2024** | **08.03.2024** |
| Round of tests #4 | **09.03.2024** | **09.03.2024** |
| Round of tests #5 | **09.03.2024** | **09.03.2024** |
| Round of tests #6 | **09.03.2024** | **09.03.2024** |

**Purpose Of This Document**

The purpose of this document is to outline a comprehensive testing strategy for the GamerPower website. It will cover various aspects of testing, including functional and non-functional tests, as well as the schedule and resources required for testing.

**Description Of GamerPower**

The GamerPower website provides access to giveaways and promotions related to gaming platforms. It offers a variety of functionalities, including:

1. Retrieving information about giveaways by platform, type, and value.
2. Sorting giveaways based on popularity, date, and other criteria.
3. Grouping giveaways by platform and type for personalized results.
4. Providing specific details about individual giveaways.
5. Checking the sorted order of giveaways based on popularity and date.
6. Handling configurations such as platform, type, and value through a configuration file.
7. Interacting with the API endpoints to fetch data dynamically.

**Key Features and Functions:**

**The system is composed of three main layers:**

1. **Infrastructure Layer: Includes modules for making HTTP requests and reading configuration data.**
2. **Logic Layer: Implements the business logic for interacting with the GamerPower endpoints and processing the data.**
3. **Test Layer: Contains test cases to validate the functionalities of the GamerPower website.**

**Glossary and Abbreviations**

**Glossary**

* GUI (Graphical User Interface): The design of user interfaces based on specified requirements.
* Functional Testing: Verification that fundamental system functions operate correctly.
* Maintenance Testing: Examination of the functionality of a modified system following changes, updates, or alterations in the working environment.
* STP (System Test Plan): A comprehensive project planning document encompassing strategy, schedule, and topic tree.
* STD (System Test Design): Detailed documentation outlining the testing plan.
* Traceability Matrix: A document that correlates any two baselined documents that require a many-to-many relationship to determine the completeness of the relationship.

**Abbreviations**

* QA: Quality Assurance
* CEO: Chief Executive Officer
* HR: Human Resources

**Testing Plan**

**Resources:** 1 tester will be assigned to this project.

**Hardware/ Software Needs:**

1. **Computers:** High-performance desktops or laptops to run test scripts and perform manual testing.
2. **Network Equipment:** Routers and switches to simulate different network conditions for testing.
3. **Virtual Machines:** Set up virtual machines for testing on different operating systems and browser combinations.
4. **Operating Systems:** Windows for testing.
5. **Browsers**: Latest versions of popular browsers (Google Chrome, Microsoft Edge, Safari) for cross-browser compatibility testing.
6. **Database Management System:** Database systems (Postman) for testing data handling and retrieval functionality.

Before the start of testing rounds, **functional tests** will be performed, including:

1. **Unit Testing**: To test individual units or components of the GamerPowerAPI website.
2. **Integration Testing**: To verify the interactions and interfaces between different components or systems within the application.
3. **Regression Testing:** To ensure that new code or changes do not affect the existing functionality of the website.
4. **API Testing:** To validate the functionality of API endpoints by testing their request-response mechanisms.
5. **UI Testing:** To validate that the user interface elements and interactions function correctly according to the design.
6. **End-to-End Testing:** To evaluate the entire system's functionality from start to finish, simulating real user scenarios and interactions.

Afterwards, the following **non-functional tests** will be conducted including:

1. **Performance Tests:** Assess how well the website performs under various conditions, including heavy loads and stressful situations.
2. **Security Testing:** Ensure the website is secure against vulnerabilities such as SQL injection, cross-site scripting, and unauthorized access.
3. **Compatibility Testing:** Ensure the website works seamlessly across different devices, browsers, and operating systems.
4. **Usability Testing**: Evaluate the user-friendliness and overall user experience of the website to ensure it meets user expectations.
5. **Reliability Testing:** Test the stability and reliability of the website under normal and extreme usage conditions.
6. **Scalability Testing:** Test the website's ability to handle increased workload and user traffic without degradation in performance.

**Starting and exiting Criteria**

* **Criteria for starting the tests:**
* 100% of planned sanity tests were carried out and passed successfully.
* 100% of planned functional and non-functional test cases have been created and reviewed.
* A traceability matrix is established, linking each test case to specific requirements.
* The testing environment is prepared and verified.
* Sufficient and accurate test data for both positive and negative scenarios is available.
* The test plan has been reviewed and approved.
* **Completion/Release Criteria:**
* 100% of planned functional and non-functional tests have been executed, and results have been documented.
* 84% of test cases passed successfully.
* The remaining bugs are at low severity levels, with no high-severity issues affecting functionality.

|  |  |  |
| --- | --- | --- |
| **Test Case ID** | **Test Case** | **Functionalities Covered** |
| 1 | test\_get\_all\_giveaways | Retrieving all giveaways |
| 2 | test\_get\_giveaways\_by\_platform | Retrieving giveaways by platform |
| 3 | test\_get\_giveaways\_sorted | Sorting giveaways |
| 4 | test\_get\_giveaways\_by\_type | Retrieving giveaways by type |
| 5 | test\_get\_specific\_giveaway | Retrieving specific giveaway |
| 6 | test\_check\_giveaways\_sorted | Checking sorted giveaways |
| 7 | test\_get\_giveaways\_group | Grouping giveaways |
| 8 | test\_check\_platforms\_group | Checking platforms grouping |
| 9 | test\_choose\_random\_value | Choosing random value |

**Traceability table**

**Test Tree**

1. Retrieve Giveaways
   * Functional Testing:

a. Retrieve All Giveaways

i. Verify successful retrieval of all giveaways.

b. Retrieve Giveaways by Platform

i. Verify successful retrieval of giveaways by platform.

c. Retrieve Giveaways Sorted

i. Verify successful sorting of giveaways.

d. Retrieve Giveaways by Type

i. Verify successful retrieval of giveaways by type.

e. Retrieve Specific Giveaway

i. Verify successful retrieval of specific giveaway.

* + Non-Functional Testing:
    1. a. Check Giveaways Sorted
       - 1. Verify giveaways are sorted by date.
         2. Verify giveaways are sorted by popularity.

b. Check Platforms Group

i. Verify correct grouping of giveaways by platform.

1. Miscellaneous
   * Functional Testing:

a. Choose Random Value

i. Verify successful selection of a random value from a list. b. Return String for Group

i. Verify correct formation of a string for group.

**Hazard Table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Hazard** | **Chance** | **Damage** | **Risk Level** | **Hazard Description** | **Action** | **Description** | **Responsible** |
|  |  | The probability of the hazard occurring, ranging from 0 to 1. | 1-10 | Chance  Multiplied by  Damage | What will happen in case of the hazard | Enclose  Monitoring  Acceptance | -Description  Of prevention method |  |
| 1 | Bad Internet Connectivity | 0.5 | 10 | 5 | Unable to connect as a user | Monitoring | NA | System |
| 2 | New Testers | 0.2 | 8 | 1.6 | Bad testing and coverage | Enclose | Finding a tester for the project length |  |
| 3 | Server Crash | 0.5 | 7 | 7 | Unable to login and retrieve info from the DB | Monitoring | NA | System |
| 4 | Vacations | 1.0 | 7 | 7 | Vacations | Enclose | Postponing / finding replacement |  |
| 5 | Weak Server | 0.5 | 10 | 5 | System Crash | Enclose | Adding servers | System |
| 6 | Employee quitting | 0.5 | 10 | 5 | Lowering Team Morale | Monitoring | Hiring a stable worker | HR |
| 7 | Inexperienced Testers | 0.1 | 10 | 1 | Failure to meet the schedule | Enclose | Hiring Experienced Testers | QA Lead |
| 8 | Contract Termination | 0.3 | 10 | 3 | No income for the company | Monitoring | More flexible customer | CEO |
| 9 | Failure to meet the schedule | 1 | 10 | 10 | Unsatisfied Customer | Enclose | QA Lead bad Management | CEO |
| 10 | Customer requirements document changes frequently during the project | 0.5 | 7 | 3.5 | Failure to be prepared for changes by the customer will not ensure a professional, accurate and correct inspection | Monitoring | An appointment must be made with the customer and it should be noted to him that it will not be possible to make changes after the system is established | CEO |