**Software Test Plan (STP) for Unity Asset Store**

**Table of Contents**

1. **Introduction**
   * **Purpose**
   * **Project Overview**
2. **Scope**
   * **In-Scope**
   * **Out-of-Scope**
3. **Testing Strategy**
   * **3.1 Test Objectives**
   * **3.2 Test Assumptions**
   * **3.3 Data Approach**
   * **3.4 Testing Types**
     + **3.4.1 Functional Testing**
     + **3.4.2 Non-Functional Testing**
       - Performance Testing
       - Security Testing
       - Usability Testing
     + **3.4.3 White-box Testing**
   * **3.5 Testing Levels**
     + **3.5.1 Unit Testing**
     + **3.5.2 User Acceptance Testing**
     + **3.5.3 Regression Testing**
4. **Execution Strategy**
   * **Entry Criteria**
   * **Exit Criteria**
5. **Validation and Defect Management**
   * **Validation of Test Cases / Test Scenarios**
   * **Defect Management**
6. **Environment Requirements**
   * **Test Environments**
   * **Dependencies**

**1. Introduction**

**Purpose**

The purpose of this Software Test Plan (STP) is to outline the testing approach and activities for the Unity Asset Store to ensure its functionality, performance, security, and usability meet the required standards and user expectations.

**Project Overview**

The Unity Asset Store provides a platform for developers to discover and acquire assets, tools, and plugins for Unity game development. This project involves testing the various aspects of the Asset Store to maintain its reliability and usability.

**2. Scope**

**In-Scope**

* Functional Testing of asset browsing, searching, purchasing, and integration.
* UI Testing for layout consistency, accessibility, and responsiveness.
* Security Testing including authentication, data protection, and secure transactions.
* Performance Testing to evaluate load handling, response times, and scalability.

**Out-of-Scope**

* Testing of Unity Editor functionalities not directly related to the Asset Store.
* Testing of third-party integrations or assets not hosted directly on the Unity Asset Store.

**3. Testing Strategy**

**3.1 Test Objectives**

* Validate core functionalities of asset discovery, purchase, and integration.
* Ensure the UI is intuitive and consistent across different devices and browsers.
* Verify robust security measures to protect user data and transactions.
* Assess performance under various loads to maintain responsiveness and scalability.

**3.2 Test Assumptions**

* Unity Asset Store infrastructure and servers are operational and accessible during testing.
* Test environments accurately represent production environments.
* Users have stable internet connections for testing transactions and downloads.

**3.3 Data Approach**

* Utilize a mix of synthetic and anonymized real data to simulate various user scenarios.
* Test with different types of assets, purchase flows, and user configurations to cover diverse use cases.

**3.4 Testing Types**

|  |  |  |
| --- | --- | --- |
| **Test Type** | **Description** | **Responsible Parties** |
| Functional Testing | Validate core asset store operations and functionalities | QA Team |
| Non-Functional Testing | Assess performance, security, and usability aspects | QA Team |
| White-box Testing | Evaluate internal structures, algorithms, and code paths | Development Team |
| Change-related Testing | Verify the impact of changes on existing functionalities. | QA Team |

**3.4.1 Functional Testing**

* **Asset Operations**: Browse, search, purchase, and download assets.
* **Integration Testing**: Verify integration compatibility with Unity Editor versions.
* **Payment Processing**: Test purchase transactions and payment gateways.

**3.5 Testing Levels**  
**3.5.1 Unit Testing**  
- Asset browsing functionality.  
- Search functionality.  
- Purchase and download functionality.  
- User account management.  
  
**3.5.2 User Acceptance Testing**  
**- Overall User Experience:** Ease of browsing and purchasing assets, effectiveness of search functionality, user interaction with the store.  
  
  
**3.5.3 Regression Testing**  
 **Core Asset Store Operations**

* **Asset Browsing:** Ensuring that the browsing feature works correctly and that assets are displayed properly.
* **Asset Search:** Validating the search functionality to ensure that assets can be found using various search terms and filters.
* **Asset Purchase:** Checking that the purchase process is seamless and error-free.
* **Asset Download:** Confirming that downloaded assets are complete and uncorrupted.

**Search functionality.**

* **Basic Search:** Testing the ability to search for assets using simple search terms.
* **Advanced Search:** Verifying that advanced search options (such as filters and categories) function correctly.
* **Search Results Accuracy:** Ensuring that search results are relevant to the search terms used.

**Reviews and Ratings**

* **Adding Reviews:** Ensuring users can add reviews and that these reviews are displayed correctly.
* **Rating Assets:** Verifying that users can rate assets and that the rating system works accurately.
* **Displaying Reviews:** Checking that reviews are displayed in the correct format and order.
* **Editing Reviews:** Validating that users can edit their reviews and that changes are updated correctly.

|  |  |  |
| --- | --- | --- |
| **Tester’s Name** | **Department/ Area** | **Role** |
| Bahaa | QA | QA Tester |
| Shibel | QA | QA Tester |

**4. Execution Strategy**

|  |  |
| --- | --- |
| **Entry Criteria** | **Notes** |
| Test environment(s) is available | |  | | --- | | Ensure that the test environment, including staging environments for Unity Asset Store, is set up and accessible. |  |  | | --- | |  | |
| Test data is available | |  | | --- | | Verify that necessary test data, including sample assets, user accounts, and configurations, is prepared. |  |  | | --- | |  | |
| Code has been merged successfully | |  | | --- | | Confirm that all code changes related to Unity Asset Store functionalities have been merged into the test environment. |  |  | | --- | |  | |
| Development has completed unit testing | |  | | --- | | Ensure that all individual components of the Unity Asset Store have undergone unit testing and are ready for integration testing. |  |  | | --- | |  | |
| Test scripts are completed, reviewed and approved by the Project Team | Validate that all test cases and scripts related to Unity Asset Store testing have been developed, reviewed, and approved by the project team. |

**Entry Criteria**

* Test environments (staging, production-like) are set up and accessible.
* Test data (sample assets, user accounts) is prepared for various testing scenarios.
* Code changes related to Asset Store functionalities are merged and deployed for testing.
* Unit testing for individual components of the Asset Store is completed.

**Exit Criteria**

* All test scripts are executed without any unresolved issues.
* At least 90% of test cases pass successfully, with documented exceptions.
* Critical and high severity defects are addressed and closed.
* All test results and metrics are documented alongside test scripts.
* Test environment cleanup is completed, and backups are made for future testing phases.
* I’m are targeting a 90% completion rate for our project milestones.

|  |  |
| --- | --- |
| **Exit Criteria** | **Notes** |
| 100% Test Scripts executed | |  | | --- | | All test scripts have been executed without any skipped or unresolved scripts. |  |  | | --- | |  | |
| 90% pass rate of Test Scripts | |  | | --- | | Ensure that at least 90% of the executed test scripts have passed successfully, allowing for minor exceptions. |  |  | | --- | |  | |
| No open Critical and High severity defects | |  | | --- | | Verify that all critical and high severity defects have been addressed and closed. |  |  | | --- | |  | |
| All remaining defects are either cancelled or documented as Change Requests for a future release | |  | | --- | | Ensure that any remaining defects are either resolved or documented for future consideration. |  |  | | --- | |  | |
| All expected and actual results are captured and documented with the test script | |  | | --- | | Confirm that all test results, including expected and actual outcomes, are documented alongside the test scripts. |  |  | | --- | |  | |
| All test metrics collected based on reports from daily and Weekly Status reports | |  | | --- | | Gather all relevant test metrics from daily and weekly status reports for analysis and reporting. |  |  | | --- | |  | |
| All defects logged in Defect Tracker/Spreadsheet | |  | | --- | | Ensure that all identified defects are logged and tracked in the designated defect tracking system or spreadsheet. |  |  | | --- | |  | |
| Test environment cleanup completed and a new back up of the environment | Perform a thorough cleanup of the test environment and create a new backup to ensure readiness for future testing phases. |

**5. Validation and Defect Management**

* **Validation of Test Cases / Test Scenarios**:
  + Execute test cases according to defined procedures to validate Asset Store functionalities.
  + Compare actual results with expected outcomes to ensure compliance.
* **Defect Management**:
  + Use a Defect Tracker to log, track, and manage identified issues.
  + Categorize defects based on severity and impact for prioritization and resolution.

|  |  |
| --- | --- |
| **Severity** | **Impact** |
| 1 (Critical) | * Functionality is blocked and no testing can proceed * Application/program/feature is unusable in the current state |
| 2 (High) | * Functionality is not usable and there is no workaround, but testing can proceed |
| 3 (Medium) | * Functionality issues but there is workaround for achieving the desired functionality |
| 4 (Low) | * Unclear error message or cosmetic error which has minimum impact on product use. |

**6. Environment Requirements**

**Test Environments**

* **Staging Environment**: Replica of production for pre-release testing.
* **Production-like Environment**: Simulates real-world conditions for performance testing.
* **Unity Editor Integration Environment**: Testing environment for asset integration with Unity Editor.

**Dependencies**

* Availability of test assets, test accounts, and sample projects.
* Resource availability including testers, tools (automation, security testing), and network connectivity.