**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 meets 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 functionalities of the Asset Store to maintain its reliability and usability.

**2. Scope**

**In-Scope**

* Functional Testing of asset browsing, searching, purchasing, and integration.
* User Authentication Testing, including login with valid and invalid credentials.
* Shopping Cart Functionality Testing, including adding assets to the cart and verifying their presence.
* Asset Management Testing, such as adding free assets to "My Assets."
* Dropdown Menu Functionality Testing, including verifying asset count changes based on selected options.
* Pricing Filter Testing to ensure assets displayed fall within the specified price range.
* Bio Editing Functionality Testing, including editing and saving user bio and handling character limits.

**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.
* Non-functional Testing including performance, security, and usability assessments.
* Load Testing and stress testing to evaluate system behavior under extreme conditions.
* Scalability Testing to determine the system's capacity to handle growth.
* Compatibility Testing across different operating systems and devices not targeted by the current scope.

**3. Testing Strategy**

**3.1 Test Objectives**

* Validate core functionalities of asset discovery, purchase, and integration.
* Verify the user authentication process, including login with valid and invalid credentials.
* Ensure proper functionality of the shopping cart, including adding assets and verifying their presence.
* 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.
* Generate test user accounts with different roles and permissions to simulate varied user interactions.

**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.
  + Add free assets to "My Assets" and verify their listing.
  + Ensure that the dropdown menu functionality accurately changes the number of displayed assets based on selection.
  + Validate the correct application of pricing filters to display assets within specified price ranges.
* **Integration Testing**:
  + Verify integration compatibility with Unity Editor versions.
* **Payment Processing**:
  + Test purchase transactions and payment gateways to ensure successful asset purchases.
* **Profile Management:**
  + Validate the bio editing functionality, including saving changes and handling character limits.

**3.5.1 Unit Testing**

* **Asset Browsing Functionality:**
  + Verify the implementation of asset listing and navigation.
  + Ensure proper rendering of asset details pages.
* **Search Functionality:**
  + Test the search feature for accuracy and relevance.
  + Validate the filtering and sorting mechanisms.
* **Purchase and Download Functionality:**
  + Ensure that the purchasing process works seamlessly, including adding assets to the cart, checking out, and downloading purchased assets.
  + Test the functionality of adding free assets to "My Assets."
* **User Account Management:**
  + Verify user authentication processes, including login and error handling.
  + Test the bio editing feature, including saving changes and handling character limits.

**3.5.2 User Acceptance Testing**

* **Overall User Experience:**
  + Assess the ease of browsing and purchasing assets.
  + Evaluate the effectiveness and user-friendliness of the search functionality.
  + Observe user interactions with the store to ensure intuitive navigation and satisfactory experience.

#### 3.5.3 Regression Testing Core Asset Store Operations

* **Asset Browsing:**
  + Validate correct functionality of browsing features.
  + Ensure assets are displayed accurately and navigation is smooth.
* **Asset Purchase:**
  + Confirm the purchase process is seamless, from adding items to the cart to checkout.
  + Validate transaction handling and error-free payment processing.

**Price Filter**

* **Pricing:**
  + Validate the functionality of setting and applying minimum and maximum price filters.

|  |  |  |
| --- | --- | --- |
| **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. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Customer Demand** | **Description** | **Related Test Cases** | **Priority** | **Status** |
| User Authentication and Login | Ability to login with valid credentials | Test Case 1, Test Case 2 | High | In Progress |
|  | Error handling for invalid credentials | Test Case 1 | High | In Progress |
|  | Secure password handling | Test Case 1 | High | In Progress |
| Asset Browsing and Searching | Functional search by asset name and category | Test Case 4 | High | Not Met |
|  | Advanced search with filters and sorting | Test Case 4 | High | Not Met |
|  | Accurate display of search results | Test Case 4 | High | Not Met |
| Asset Purchase and Download | Seamless purchase process | Test Case 3 | High | Not Met |
|  | Confirmation of purchase | Test Case 3 | High | Not Met |
|  | Reliable asset download | Test Case 3 | High | Not Met |
| UI/UX and Accessibility | Responsive layout on different devices | Test Case 6 | Medium | Not Met |
|  | Accessibility features for disabled users | Test Case 6 | Medium | Not Met |
|  | Consistent UI across platforms | Test Case 6 | Medium | Not Met |
| Security and Privacy | Secure transactions and payment processing | Test Case 8 | High | Not Met |
|  | Protection of user data | Test Case 8 | High | Not Met |
|  | Compliance with data protection regulations | Test Case 8 | High | Not Met |
| Performance and Scalability | Fast loading times for asset pages | Test Case 7 | Medium | Not Met |
|  | Scalability under peak loads | Test Case 7 | Medium | Not Met |
|  | Minimal downtime during updates | Test Case 7 | Medium | Not Met |
| User Reviews and Ratings | Ability to leave reviews for assets | Test Case 5 | Medium | Not Met |
|  | Rating system for assets | Test Case 5 | Medium | Not Met |
|  | Display of reviews in a clear format | Test Case 5 | Medium | Not Met |
| Miscellaneous Features | Add assets to "My Assets" collection | Test Case 10 | Low | Not Met |
|  | Filter assets by price range | Test Case 8 | Low | Not Met |
|  | Integration with Unity Editor | Test Case 9 | Medium | Not Me |

**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 Tracking Table

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Test Case Name | Status | Notes |
| 1 | Login with valid email address and invalid password | Executed | All steps passed |
| 2 | Login with valid email address and valid password | Executed | All steps passed |
| 3 | Add To Cart | Executed | All steps passed |
| 4 | Verify Related Keywords for Selected Category | Executed | All steps passed |
| 5 | Add a free asset to the “my assets” | Executed | All steps passed |
| 6 | Assets Count Changes After Selecting From Dropdown | Executed | All steps passed |
| 7 | Assets Count Matches Chosen Option After Selecting From Dropdown | Executed | All steps passed |
| 8 | Assets Displayed Within Specified Price Range | Executed | All steps passed |
| 9 | Bio Edit Functionality | Executed | All steps passed |
| 10 | Bio Edit exceeding the character limit | Executed | All steps passed |

**7. 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.