4.6 Test Strategy

Complete, multi-layered testing will be employed to certify that the Quiz AI is accurate, reliable, and meets all functional and non-functional requirements. The strategy will use automated and manual testing techniques throughout the development process.

**4.6.1 Testing Methodology**

**The testing approach will be centered on Unit Testing and Burp Suite.**

**Unit Testing:**

**Objective: To confirm every single method, function, and class operates correctly independently of the other system parts. It is highly recommended to detect logic bugs as early as possible during the development cycle.**

**Method: Automated tests will be written by developers for the lowest level of units of code. These are testing functions for text processing, quiz question generation, input validation, and data formatting. External dependencies such as calls to AI APIs or file system calls will be mocked out with mock objects so that tests are fast, consistent, and isolated.**

**Tool: The project will utilize the unit test framework, Python's default unit test tool. The framework provides a solid base upon which to build and run a solid set of tests.**

Penetration Testing & Security Scanning:

Purpose: To actively scan and fix security vulnerabilities in the application before it is deployed.

Procedure: The deployed application will be scanned for typical web vulnerabilities systematically. This includes testing injection flaws (SQLi, XSS), broken authentication, insecure direct object reference, and other on the OWASP Top 10 list.

Tool: The primary tool for manual and automated security testing will be Burp Suite. It will be used to intercept, observe, and manipulate HTTP/S requests between the server and client in an effort to find security vulnerabilities.

4.6.2 Error Identification and Correction

A transparent process will be employed to deal with problems that are found:

Identification: Problems will be identified by automated unit test failures and security scans.

Triage & Logging: Problems from functional bugs to severe security bugs will be logged, prioritized for repair, and sorted based on severity.

Resolution: Bugs will be repaired by developers, while security bugs will be repaired using a patch. Security patches will be tested to ensure that they do not introduce new vulnerabilities.

Validation: The unit test suite and The Burp Suite will be re-run to verify that the fix is working without causing any regressions.

**4.6.3 Test Cases and Validation**

The following table provides examples of test cases for both unit and security testing.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Type | Description | Input Expected | Outcome |
| TC-U-101 | Unit Test | Test input validation for file upload. | A file with type .exe | Function raises a ValueError. |
| TC-U-102 | Unit Test | Test JSON formatting of a quiz question. | Question data object. | A valid JSON string with correct fields. |
| TC-SEC-201 | Security (Burp Suite) | Test for SQL Injection in login form. | username: admin' -- | Returns a generic error message, not a database error. |
| TC-SEC-202 | Security (Burp Suite) | Test for Cross-Site Scripting (XSS) in quiz output. | <script>alert('test')</script> | Input is sanitized; script tags are not executed. wright these a line for each one |