**Ielts Analyzer Testing Document**

Prepared by Nimit Shah

December 4, 2024

Release 1.0

**INDEX**

|  |  |  |
| --- | --- | --- |
| 1. Aim |  | 3 |
| 1. Scope |  | 3 |
| 1. Testers Detail |  | 4 |
| 1. Steps of Testing |  | 5 |
| * 1. Unit Testing |  | 6 |
| * 1. Integration Testing |  | 7 |
| * 1. Validation Testing |  | 8 |
| * 1. Solving Errors |  | 9 |
| * 1. Regression Testing |  | 10 |
| * 1. System Testing |  | 11 |
| * + - Functional Testing |  | 11 |
| * + - Non - Functional Testing |  | 11 |
| * + - * Load Testing |  | 11 |
| * + - * Stress Testing |  | 12 |
| * + - * Security Testing |  | 12 |
| * + - * Volume Testing |  | 12 |
| * + - * Scalability Testing |  | 12 |
| 1. Excel Sheets List |  | 13 |
| 1. Conclusion |  | 14 |

1. **Aim**  
   The aim of this document is to outline the testing strategy, methodology, and procedures to ensure the functionality, performance, usability, security, and compatibility of the website meet specified requirements and deliver a seamless user experience.
2. **Scope**  
   This document covers the testing of all website components, including but not limited to:
3. Functional Testing: Validation of features, navigation, and links.
4. Performance Testing: Evaluation of loading time, scalability, and response under various conditions.
5. Usability Testing: Assessment of user interface and overall user experience.
6. Security Testing: Identification of vulnerabilities and data protection measures.
7. Compatibility Testing: Testing across multiple browsers, devices, and screen resolutions.

Exclusions and limitations, if any, will be documented separately.

1. **Tester Details**
   1. **Tester 1**

* **Name** - Nimit Shah
* **Designation** - Software developer
* **Email** - [nimitshah240@gmail.com](mailto:nimitshah240@gmail.com)
* **Experience in testing** - 2 website
* **Involvement in development** – 100%
  1. **Tester 2**
* **Name** –
* **Designation** – User
* **Email** –
* **Experience in testing** – 0
* **Involvement in development** – 0%

1. **Steps of testing**

The testing process will be conducted in a structured sequence to ensure comprehensive coverage and reliability. Testing details and bugs should be mentioned in the different excel sheets and every bug will resolve and push in master branch through different repositories of each bug. All testing will perform on two platform. 1) Localhost, 2) Netlify app. Testing will be done in two phases, 1) Developer will test the website on localhost and Netlify app, 2) Non-Developer/User will test it on Netlify app. Steps of testing procedure:

1. **Unit Testing:**

Individual components and modules of the website will be tested to verify their functionality and correctness.

1. **Integration Testing:**

Interaction between modules will be tested to ensure seamless data flow and communication.

1. **Validation Testing:**

The website will be tested against functional and non-functional requirements to confirm that it meets specified objectives

1. **Bug Fixing and Regression Testing:**

Identified bugs will be addressed, and regression testing will be performed to ensure that fixes have not introduced new issues or affected existing functionality.

1. **System Testing:**

The entire website will undergo testing under realistic scenarios to evaluate its performance, including load capacity and stress handling.

This sequence ensures that issues are detected and resolved efficiently, leading to a stable and high-performing website.

1. **Unit Testing**

Unit testing focuses on verifying individual components or modules of the website to ensure they perform as expected in isolation. This includes testing functions, conditions, methods, and classes for correct outputs and handling edge cases. It serves as the foundation for detecting and resolving errors early in development cycle.  
**a. Page Testing Sequence:**

The following pages will be tested in sequence to ensure consistency and functionality across the website:

1. **Home**
2. **Selection**
3. **ListView**
4. **DataEntry**
5. **Dashboard**
6. **Trick**
7. **StudentData**

**b. Testing Scope for Pages:**

1. **Design Validation:**

Each page's design will be tested on different devices (desktop, tablet, and mobile) to ensure responsiveness and usability.

1. **CRUD Operations:**

Create, Read, Update, and Delete functionalities for each page will be tested thoroughly to ensure seamless data management and integrity.

This structured approach ensures the website's functionality, user experience, and compatibility across all pages and devices.

1. **Integration testing**

Integration testing ensures that different modules or components of the website work together as intended. This involves testing the data flow, interactions, and dependencies between modules to identify and resolve any issues in their integration. It ensures that the combined functionality of interconnected components aligns with the specified requirements.

**a. Testing Sequence:**

Integration testing will be conducted in the following order to validate data flow and interconnectivity:

1. **Home ➝ Selection**
   * Data, Dashboard, and Trick modules will be tested.
2. **Selection ➝ ListView**
   * Testing Reading and Listening functionalities.
3. **ListView ➝ DataEntry Page**
   * Testing transitions and operations for Reading and Listening functionalities.
4. **Selection ➝ Dashboard**
   * Testing the Dashboard for both Reading and Listening data.
5. **Selection ➝ Trick**
   * Testing Trick functionality for Reading and Listening data.
6. **Sign-In and Sign-Out:**
   * Validating authentication functionality from all pages.
7. **StudentData ➝ Dashboard**
   * Testing Reading and Listening data integration with the Dashboard.
8. **Toast ➝ All pages**
   * Test toast message with all CRUD operation and catches in all pages.
9. **Spinner ➝ All pages**
   * Test spinner on all pages.

This structured sequence ensures that all critical interactions and transitions between modules are thoroughly tested, leading to a reliable and seamless user experience.

1. **Validation Testing**

Validation testing ensures the website meets business and user requirements by verifying that the product behaves as expected. It focuses on confirming that the functional and non-functional aspects align with specifications. This includes checking form inputs, data integrity, navigation flows, and adherence to user expectations.

**a**. **Validation Testing During Unit and Integration Testing:**

1. **During Unit Testing:**

Validation testing will verify the correctness of individual components, such as ensuring form fields accept valid data, mandatory fields cannot be left blank, and error messages display appropriately for invalid inputs.

1. **During Integration Testing:**

Validation testing will ensure that data flow between modules is accurate and adheres to expected conditions, such as ensuring data entered on the **DataEntry** page reflects correctly on the **Dashboard** or **Trick** modules. Cross-module navigation will also be validated to ensure a seamless user experience.

This combined approach ensures that validation testing occurs alongside unit and integration testing to catch and resolve issues early, leading to a robust and user-friendly website.

1. **Solving Errors**

After the completion of all testing phases, the process of resolving identified errors will begin. High-priority bugs, which significantly impact system functionality or user experience, will be addressed first to ensure critical issues are resolved promptly.

1. **Bug Tracking:**
   * All identified bugs will be documented in a new Excel sheet.
   * Each bug will be assigned a unique Bug ID for easy identification and tracking.
2. **Branch Management:**
   * For every bug, a separate branch will be created with the branch name corresponding to the **Bug ID**. For example {git checkout -b ‘001’}
   * Once the bug is resolved, changes will be reviewed and pushed to the master branch.
3. **Commit Guidelines:**
   * Each commit must include a comment specifying:
     + The type of bug (**Functionality** or **Design**).
     + A detailed description of the changes made to resolve the issue. For example {git commit -m ‘Design/Functionality Comment issue was in’}

This process ensures clear traceability of bugs, efficient collaboration, and structured updates to the master branch.

1. **Regression Testing**
2. **Testing Solved Errors:**
   * After resolving all errors, testing will be conducted to confirm the fixes are effective.
3. **Re-Testing of Related Components:**
   * Components related to the fixed errors will be re-tested to ensure the changes have not caused any new issues.
4. **Bug Identification:**
   * If any bug is found during re-testing, it will be documented, specifying whether it is the same bug (recurrence) or a new bug.
   * It will be documented in new tab in **Bug sheet.**
5. **Error Resolution and Regression Testing:**
   * The identified bug will be resolved, and regression testing will be performed to confirm that all functionalities work as expected after the fix.
   * New tab in the same **Regression testing sheet** will be created for new bug’s testing.
   * This cycle of **error resolution** and **regression testing** will continue until no bugs are found.
6. **Regression Testing Documentation:**
   * All regression testing results will be recorded in a dedicated **Regression Testing Excel Sheet**, separate from the primary bug tracking sheet, for better traceability and analysis.

This systematic approach ensures a stable, error-free website before deployment.

1. **System Testing**

System testing is performed after integration testing and involves testing the entire application as a whole to ensure that all components work together correctly. It focuses on validating the system’s compliance with functional and [non-functional requirements](https://www.browserstack.com/guide/non-functional-requirements-examples), such as performance, usability, and security. This type of testing simulates real-world scenarios to identify defects and confirm that the software behaves as expected in a production-like environment, ultimately ensuring the quality and reliability of the system before it is deployed.

**a. Type of System Testing:**

1. **Functional testing**

In this testing, website will be taken as whole. Functional testing verifies that the system performs its intended functions as per specified requirements. It focuses on testing individual features and workflows, such as data submissions, database operations, and navigation. This ensures the application behaves correctly from the end-user perspective.

1. **Non-Functional Testing**

Non-functional testing evaluates aspects of the system unrelated to specific functionality, such as performance, reliability, usability, and scalability. This testing will perform after fully completion of functional testing with 100% success. It ensures the system meets quality standards and handles conditions like high loads, security threats, or different environments. This testing focuses on "how" the system performs rather than "what" it does.

1. **Key Types of Non-Functional Testing:**
2. **Load Testing:**
   * Assesses how the system performs under expected user loads to ensure it operates efficiently.
   * Example: Testing a website with 1,000 concurrent users.
   * How: Mention it later
3. **Stress Testing:**
   * Evaluates the system's behaviour under extreme conditions or beyond normal operational capacity to identify breaking points.
   * Example: Simulating 10,000 users to check system stability during peak loads.
   * How: Mention it later
4. **Security Testing:**
   * Identifies vulnerabilities and ensures the system protects data and resources against potential threats or attacks.
   * Example: Testing for SQL injection, XSS, and unauthorized access.
   * How: Mention it later
5. **Volume Testing:**
   * Examines the system's performance when handling a large volume of data over a period.
   * Example: Testing a database with millions of records to check for efficiency and speed.
   * How: Mention it later
6. **Scalability Testing:**
   * Verifies the system’s ability to scale up or down in response to increased or decreased workloads.
   * Example: Testing how adding more servers impacts system performance.
   * How: Mention it later

These tests ensure the application is robust, reliable, and ready for real-world usage scenarios.

**5. Excel Sheets**

During the testing process, the following Excel sheets will be maintained for efficient tracking and management:

1. **Test Case Scenario Sheet:**
   * Contains detailed test cases, steps, expected results, and actual results for each testing phase. {Mention what data will be there in the sheet}
2. **Bug Sheet:**
   * Documents all identified bugs, including their details, statuses, and resolutions. {Mention what data will be there in the sheet}
   * This sheet will have multiple tabs to organize bugs based on the rounds of testing and bug fixing performed recursively. Each tab will correspond to a specific testing cycle or group of related bugs.
3. **Regression Testing Sheet:**
   * Tracks the results of regression testing performed after bug fixes to ensure stability. {Mention what data will be there in the sheet}
   * Similar to the Bug Sheet, this sheet will have multiple tabs for each round of regression testing and for components affected by bug fixes.
4. **Non-Functional Testing Sheet:**
   * The **Non-Functional Testing Sheet** is designed to document and track the testing of quality attributes that are not directly related to specific functionalities but are critical for overall system performance and user experience. This sheet will include detailed records of various non-functional tests performed, such as load testing, security testing, stress testing, volume testing, scalability testing.

This systematic structure ensures comprehensive tracking and easy navigation through testing data, enhancing collaboration and issue resolution.

**6. Conclusion**

This document outlines a comprehensive testing process designed to ensure the quality, functionality, and reliability of the system. The testing will be conducted in a systematic sequence, including Unit Testing, Integration Testing, Validation Testing, Regression Testing, and System Testing. Each phase will follow specific steps to identify, document, and resolve issues, ensuring the system meets all requirements.

The types of testing to be performed include:

* Functional Testing
* Non-Functional Testing (e.g., Load, Stress, Security, Volume, and Scalability Testing)

To effectively manage and track the testing process, the following Excel sheets will be maintained:

1. **Test Case Scenario Sheet**
2. **Bug Sheet** (with multiple tabs for iterative testing and bug resolution)
3. **Regression Testing Sheet** (with multiple tabs for each testing round and affected components)

By adhering to this structured approach, the testing process will ensure a robust, user-friendly, and error-free system ready for deployment.