REFS Yard Testing Plan

## 1. Introduction

This testing plan outlines the testing activities required to validate the REFS Yard web-based platform, ensuring it meets the functional and non-functional requirements outlined in the requirements analysis. The plan covers testing types, methodologies, tools, and test cases to verify the platform's functionality, performance, usability, security, compatibility, and accessibility.

## 2. Testing Objectives

* Validate all functional requirements (FR1–FR5) to ensure correct system behavior.
* Verify non-functional requirements (NFR1–NFR4) for performance, scalability, and security.
* Ensure a superior user experience through usability and accessibility testing.
* Confirm compatibility across specified browsers and devices.
* Validate business and user requirements through end-to-end testing.

## 3. Testing Scope

The testing scope includes all platform components:

* User interface (search, purchase, account management, recommendation engine).
* Backend services (content upload, database interactions, cloud infrastructure).
* Integration points (payment systems, content delivery).
* Non-functional aspects (performance, security, scalability).

## 4. Testing Types and Test Cases

### 4.1 Functional Testing

**Objective**: Verify that all functional requirements (FR1–FR5) work as specified.  
**Test Cases**:

* **TC-FR1-01**: Verify users can search for references by keywords, categories, and authors, with results matching query criteria.
* **TC-FR1-02**: Validate search filters (e.g., category, author) return accurate results.
* **TC-FR2-01**: Confirm users can purchase and download references in PDF and ePub formats.
* **TC-FR2-02**: Test error handling for failed downloads or payment issues.
* **TC-FR3-01**: Ensure content providers can upload reference materials in supported formats.
* **TC-FR3-02**: Verify content providers can edit or delete uploaded references.
* **TC-FR4-01**: Test user account creation with valid and invalid inputs.
* **TC-FR4-02**: Validate account management features (e.g., profile updates, password reset).
* **TC-FR5-01**: Confirm the recommendation engine suggests relevant references based on user search history.
* **TC-FR5-02**: Test recommendation accuracy for edge cases (e.g., new users).

**Tools**: Selenium (for automation), Postman (for API testing).

### 4.2 Usability Testing

**Objective**: Ensure the platform is intuitive and user-friendly, meeting UR1 (intuitive interface).  
**Test Cases**:

* **TC-UR1-01**: Conduct user testing to verify navigation is intuitive for users with low technical skills.
* **TC-UR1-02**: Test consistency of design elements (e.g., fonts, buttons) across pages.
* **TC-UR1-03**: Validate error messages are clear and actionable.
* **TC-UR1-04**: Assess ease of completing key tasks (e.g., search, purchase) within 3 steps.

**Methodology**: Moderated usability testing with 10–15 participants (academics, students, professionals).  
**Tools**: Figma (for prototype testing), UserTesting.com.

### 4.3 Compatibility Testing

**Objective**: Confirm the platform works across modern browsers and devices, per SR1 and UR2.  
**Test Cases**:

* **TC-SR1-01**: Verify functionality on Chrome, Firefox, and Safari (latest versions).
* **TC-SR1-02**: Test responsiveness on desktop (1920x1080, 1366x768 resolutions).
* **TC-UR2-01**: Validate mobile compatibility on iOS (Safari) and Android (Chrome) devices.
* **TC-UR2-02**: Ensure touch interactions (e.g., swipe, tap) work on mobile devices.

**Tools**: BrowserStack for cross-browser and device testing.

### 4.4 Performance Testing

**Objective**: Validate non-functional requirements (NFR1–NFR3) for speed, scalability, and uptime.  
**Test Cases**:

* **TC-NFR1-01**: Verify search results load within 3 seconds under normal conditions.
* **TC-NFR2-01**: Test system performance with 10,000 concurrent users.
* **TC-NFR2-02**: Conduct stress testing to identify breaking points beyond 10,000 users.
* **TC-NFR3-01**: Monitor platform uptime over a 30-day period to ensure 99.9% availability.
* **TC-NFR3-02**: Test recovery time after simulated server downtime.

**Tools**: JMeter (load testing), AWS CloudWatch (monitoring uptime).

**4.5 Security Testing**

**Objective**: Ensure user data is protected and vulnerabilities are mitigated, per NFR4.  
**Test Cases**:

* **TC-NFR4-01**: Verify end-to-end encryption for user data during transmission.
* **TC-NFR4-02**: Test for SQL injection vulnerabilities in search and login forms.
* **TC-NFR4-03**: Validate session management to prevent unauthorized access.
* **TC-NFR4-04**: Test for cross-site scripting (XSS) vulnerabilities in user inputs.
* **TC-NFR4-05**: Ensure secure payment processing complies with PCI DSS standards.

**Tools**: OWASP ZAP (vulnerability scanning), Burp Suite (penetration testing).

### 4.6 Accessibility Testing

**Objective**: Ensure the platform is usable by people with disabilities, aligning with WCAG standards.  
**Test Cases**:

* **TC-ACC-01**: Verify compliance with WCAG 2.1 Level AA (e.g., color contrast, text size).
* **TC-ACC-02**: Test keyboard navigation for all interactive elements.
* **TC-ACC-03**: Confirm screen reader compatibility (e.g., JAWS, NVDA) for key workflows.
* **TC-ACC-04**: Validate alternative text for images and icons.

**Tools**: WAVE, axe Accessibility Checker.

### 4.7 Localization Testing

**Objective**: Ensure the platform supports multi-language features if added (noted as a potential scope creep challenge).  
**Test Cases**:

* **TC-LOC-01**: Verify UI text displays correctly in at least two languages (e.g., English, Spanish).
* **TC-LOC-02**: Test date, time, and currency formats for different regions.
* **TC-LOC-03**: Ensure search functionality supports non-Latin characters.

**Tools**: Manual testing with native speakers, automated checks via Selenium.

### 4.8 Database Testing

**Objective**: Validate data integrity and performance for backend operations.  
**Test Cases**:

* **TC-DB-01**: Verify data consistency after reference uploads and downloads.
* **TC-DB-02**: Test query execution time for search operations.
* **TC-DB-03**: Validate data integrity during concurrent user updates.
* **TC-DB-04**: Ensure secure storage of user credentials and payment information.

**Tools**: SQL queries, AWS RDS monitoring tools.

### 4.9 Interface Testing

**Objective**: Ensure seamless interaction between platform components (e.g., frontend, backend, payment systems).  
**Test Cases**:

* **TC-INT-01**: Verify data flow between search interface and database.
* **TC-INT-02**: Test integration with payment gateway for successful transactions.
* **TC-INT-03**: Validate error handling for failed API calls.

**Tools**: Postman, REST-assured.

### 4.10 Regression Testing

**Objective**: Ensure new updates do not break existing functionality.  
**Test Cases**:

* **TC-REG-01**: Re-run all functional test cases after each major update.
* **TC-REG-02**: Verify critical workflows (search, purchase, upload) remain unaffected.
* **TC-REG-03**: Test recommendation engine accuracy post-updates.

**Tools**: Selenium (for automated regression testing).

## 5. Testing Environment

* **Frontend**: Deployed on modern browsers (Chrome, Firefox, Safari).
* **Backend**: AWS cloud infrastructure (e.g., EC2, RDS).
* **Test Data**: Synthetic user accounts, sample references, and mock payment data.
* **Staging Environment**: Mirrors production setup for realistic testing.

## 6. Testing Schedule

| **Phase** | **Duration** | **Activities** |
| --- | --- | --- |
| Test Planning | 1 week | Define test cases, set up tools |
| Functional Testing | 3 weeks | Execute FR1–FR5 test cases |
| Usability Testing | 2 weeks | Conduct user testing sessions |
| Compatibility Testing | 2 weeks | Test across browsers and devices |
| Performance Testing | 2 weeks | Load and stress testing |
| Security Testing | 2 weeks | Vulnerability scanning, penetration testing |
| Accessibility Testing | 1 week | WCAG compliance checks |
| Localization Testing | 1 week | Language and format validation |
| Database Testing | 1 week | Data integrity and performance tests |
| Interface Testing | 1 week | Integration testing |
| Regression Testing | Ongoing | Post-update validation |

## 7. Tools and Resources

* **Automation**: Selenium, JMeter, Postman.
* **Security**: OWASP ZAP, Burp Suite.
* **Accessibility**: WAVE, axe.
* **Usability**: Figma, UserTesting.com.
* **Compatibility**: BrowserStack.
* **Monitoring**: AWS CloudWatch.

## 8. Risks and Mitigation

* **Risk**: Incomplete test coverage due to complex requirements.  
  **Mitigation**: Use traceability matrix to map requirements to test cases.
* **Risk**: Performance issues under high load.  
  **Mitigation**: Conduct early load testing and optimize AWS infrastructure.
* **Risk**: Security vulnerabilities missed.  
  **Mitigation**: Follow OWASP guidelines and perform thorough penetration testing.

## 9. Conclusion

This testing plan ensures REFS Yard meets its functional, non-functional, and user requirements through a structured approach. By covering functional, usability, compatibility, performance, security, accessibility, localization, database, interface, and regression testing, the platform will deliver a reliable, secure, and user-friendly experience, aligning with its goal of revolutionizing reference sales.