Test Plan - Movie Ticket Booking Web APP

Document Version: 1.0
Prepared By: RajaRam Kannuri
Date: 30-11-2023

Introduction

This test plan is crafted to guide the systematic testing of web applications for booking movie tickets. It aims to ensure that the application adheres to the highest standards in functionality, usability, performance, and security.

Table of Contents

1. Test Objectives

The primary objectives of this test plan are to:

- Validate the application's core functionalities such as registration, browsing, booking, and ticket management.
- Ensure a high standard of usability and user experience.
- Confirm the application's performance under various conditions.
- Guarantee the security and data integrity throughout the application.

2. Test Scope

This testing initiative will encompass:

2.1. User Registration and Login

- **Functional Testing:** Verify the user can register with valid credentials. Test login functionality with correct and incorrect credentials. Check password reset and email verification processes.
- **Usability Testing:** Assess ease of navigation through registration and login forms. Ensure clear error messages and guidance.
- **Security Testing:** Test for SQL injection, cross-site scripting, and other common security vulnerabilities in registration and login forms.

2.2. Browsing Movie Listings and Showtimes

- **Functional Testing:** Ensure accurate display of movies and showtimes. Test filters and search functionality.
- **Performance Testing:** Evaluate the load time of movie listings, especially under high traffic conditions.
- **Compatibility Testing:** Verify that browsing works seamlessly across different browsers and devices.

2.3. Selecting Seats for a Chosen Movie and Showtime

- **Functional Testing:** Test the seat selection process for different movies and showtimes. Ensure that the selection is accurately reflected in the booking process.
- **Usability Testing:** Assess the user interface for selecting seats. Ensure it is intuitive and provides adequate information about seat availability.

- **Concurrency Testing:** Verify system behavior when multiple users attempt to book the same seat simultaneously.

2.4. Booking Tickets and Making Payments

- **Functional Testing:** Test the entire booking flow, including adding tickets to the cart, applying discounts or promo codes, and completing the payment.
- **Security Testing:** Ensure secure handling of payment information. Test for vulnerabilities in payment processing.
- **Integration Testing:** Verify integration with payment gateways and third-party services.

2.5. Viewing and Canceling Booked Tickets

- **Functional Testing:** Ensure users can view their booked tickets and details. Test the ticket cancellation process and related refunds or penalties.
- **Usability Testing:** Assess the ease of accessing booking history and understanding cancellation policies.
- **Data Integrity Testing:** Ensure that cancellations are accurately reflected in the system, including seat availability updates.

3. Out of Scope

- **Third-Party Services:** Detailed testing of integrated third-party services beyond the interface level, such as in-depth testing of payment gateway internals, is out of scope.
- **Extreme Load Testing:** While performance testing is included, extreme load testing (e.g., simulating conditions far beyond expected peak usage) is out of scope.
- **Long-term Usability Testing:** Longitudinal studies on usability, which involve extended periods of user interaction, are not included.
- **Hardware Testing:** Testing the application on every possible device and hardware configuration is beyond the scope. A representative set of devices will be used.
- **Internationalization and Localization:** Testing the application in languages other than the primary language (if not specified as a requirement) and specific regional adaptations are not included.

- **Complete Backend Testing:** In-depth testing of backend services, like database performance under extreme conditions, is out of the test scope, assuming it falls under a separate backend testing plan.

4. Entry Criteria

Before commencing the testing process, the following criteria must be met:

- Completion of the application development phase with all features implemented.
- Successful unit and integration testing by the development team.
- Fully configured test environment, including the setup of databases, servers, and other infrastructure components.
- Availability of all required testing tools and access rights.

5. Test Strategy

The testing approach includes:

- **Manual Testing:** To cover user experience and exploratory testing aspects.
- **Automated Testing:** For regression, load, and performance testing.
- **Security Testing:** To identify vulnerabilities and data breach potentials.
- **Usability Testing:** Focused on the user interface and user journey.
- **Compatibility Testing:** Across various browsers and devices.

6. Test Environment Setup

The test environment will simulate the production environment and will include:

- **Multiple operating systems:** Windows, macOS, Linux.
- **A range of browsers:** Chrome, Firefox, Edge, Safari.
- **Mobile devices** with different screen sizes and operating systems.

7. Test Schedule

The testing timeline will align with the project development milestones, comprising:

- Preparation phase for test planning and environment setup.
- Execution phase for running test cases.
- Reporting phase for analyzing test results and documenting findings.

8. Test Deliverables

Key deliverables of the testing phase will include:

- A detailed test plan document.
- Comprehensive test cases and test scripts.
- Regular test reports and defect logs.
- Final test summary report.

9. Test Exit Criteria

Testing will be considered complete when:

- All critical bugs are resolved and retested.
- Test coverage meets the predefined threshold.
- Performance and security benchmarks are achieved.

10. Test Scenarios and Cases

For each application feature, detailed test scenarios and cases will be developed, including:

- Positive and negative test cases.
- Boundary condition testing.
- Data-driven test cases for dynamic testing.

11. Automated Testing Approach

11.1.Choice of Tools

- Selenium WebDriver for web UI testing.

- JMeter for performance testing.
- Postman for API testing.
- Qualys Tool for Identifying Security Vulnerabilities

11.2.Test Case Prioritization for Automation

- Focus on repetitive tasks and regression cases.
- Automate complex scenarios involving multiple system integrations.

11.2.1. High Priority for Automation

User Registration and Login (Functional Testing)

- Automate tests for user registration with valid and invalid credentials, and login functionality. These are repetitive and vital for user access.

Browsing Movie Listings and Showtimes (Functional Testing)

- Automate tests for the display of movies and showtimes, including filters and search functionality. Since this feature is frequently used and critical for the user experience, automation ensures consistency and efficiency.

Selecting Seats for a Chosen Movie and Showtime (Functional Testing)

- Automate seat selection tests, ensuring that the selection process is accurately reflected in the booking process. This feature's complexity and frequency of use make it a good candidate for automation.

Booking Tickets and Making Payments (Functional Testing)

- Focus on automating the entire booking flow, including ticket selection, application of discounts, and payment processing. Given the complexity and critical nature of financial transactions, automated tests can help identify issues more reliably.

Viewing and Canceling Booked Tickets (Functional Testing)

- Automate tests for viewing booked tickets and the cancellation process, as these are common user actions. Automation can help ensure that these functionalities are consistently working as intended.

11.2.2. Medium Priority for Automation

Concurrency Testing for Seat Selection

- Automated tests to simulate multiple users booking the same seat simultaneously. This is less frequent but critical for user experience and system integrity.

Security Testing (for all Functional Areas)

- While not all security tests can be automated, tests for common vulnerabilities like SQL injection in user registration and login forms should be prioritized for automation due to their critical nature.

11.2.3. Lower Priority for Automation

Usability Testing (all areas)

- While important, usability testing often requires subjective human assessment that doesn't lend itself easily to automation. Automated tests can be used for basic navigational aspects, but detailed usability evaluations are typically manual.

Data Integrity Testing for Cancellations

- While important, this might be automated after higher-priority tests are in place, as it may be less critical and less frequently executed than the other tests.

10.3. Benefits and Challenges

- **Benefits:** Increased test coverage, reliability, faster feedback.
- **Challenges:** Initial setup cost, maintaining test scripts, ensuring tests remain relevant with application changes.

12. Test Data Management

- **Data Preparation:** Ensure availability of realistic test data for scenarios like user registration, booking, etc.
- **Data Privacy:** Apply data masking and anonymization techniques for sensitive information.
- **Data Storage and Access:** Securely store test data, ensuring access is controlled and limited to authorized personnel.
- **Data Maintenance:** Regularly update and validate test data to reflect changes in application features and user behavior.

13. Resources

- **Test Lead(1):** Oversees the entire testing process, ensuring alignment with objectives and designing and Development of Test Automation Framework.
- **QA Engineers (2):** Create Comprehensive Test Cases and Execute manual and develop automated test cases, report defects.

14. Mitigation Strategies

- **Risk Identification:** Regularly identify and assess potential risks in testing phases.
- **Contingency Planning:** Develop contingency plans for critical risks, including resource reallocation, schedule adjustments, and scope reduction.
- **Communication Plan:** Establish clear communication channels for timely reporting of issues and uncertainties.
- **Backup Resources:** Maintain backup resources such as additional testers or alternative tools to manage unexpected demands.
- **Regular Reviews:** Conduct regular review meetings to monitor risks and implement mitigation strategies proactively.