Software Test Plan

airbnb

Mahmood Odeh

contents

1. Introduction

a. Purpose

The purpose of this document is to:

- Define the scope of testing.
- Describe the testing objectives and approach.
- Identify the resources, roles, and responsibilities for testing.
- Outline the test environment and tools.
- Provide a schedule for testing activities.
- Specify the deliverables and criteria for test completion.

b. Project Overview

The Software Test Plan (STP) document outlines the testing approach and strategy for ensuring the quality of an Airbnb-like website. This document will guide the testing team in planning, executing, and reporting on the testing activities.

c. Website Overview

Airbnb is a platform that connects travelers with hosts who have accommodations available for short-term rental. Airbnb, allows users to search for accommodations based on location, dates, price range, and amenities. Users can view detailed listings, including photos, descriptions, and reviews, and book accommodations directly through the website.

d. Key Features

- .User Registration and Authentication: Users can create accounts, log in, and manage their profiles.
- Accommodation Search: Users can search for accommodations based on various criteria and view search results.
- Accommodation Listings: Hosts can create listings for their accommodations, including photos, descriptions, and pricing.
- Booking and Payment: Users can book accommodations and make payments through the website.
- Reviews and Ratings: Users can leave reviews and ratings for accommodations and hosts.

- Messaging: Users can communicate with hosts through a messaging system.
- Notifications: Users receive notifications for booking confirmations, messages, and other important updates.
- Admin Dashboard: Admins have access to a dashboard for managing listings, users, bookings, and other aspects of the website.

2. Test Strategy

a. Test Objectives

The primary objective of the testing strategy for airbnb is to ensure that the website functions as intended, meets the specified requirements, and provides a positive user experience. The testing will cover functional, non-functional, and usability aspects of the website.

The final product of the test is:

- A working product with minimal bugs that meets the specifications.
- A set of stable test scripts that can be reused for test execution and user acceptance tests.

b. Test Assumption

key assumption:

- Production like data required and be available in the system prior to start of Functional Testing

General assumptions:

- Availability of Test Environments: It is assumed that a dedicated test environment, mimicking the production environment, is available for testing purposes without interference from ongoing development activities.
- Tool Availability: Necessary testing tools are available and configured for use by the testing team.

- Documentation: Comprehensive documentation on application features, known issues, and integration points is available for reference by the test team.
- Dev team will provide Defect fix plans based on the Defect meetings during each cycle to plan.
- Schedule Adherence: All parties involved will adhere to the testing schedule, including timely completion of development sprints, bug fixes, and re-testing phases.
- The defects will be tracked through Jira only. Any defect fixes planned will be shared with Test Team prior to applying the fixes on the Test environment
- The system will be treated as a black box; if the information shows correctly online and in the reports, it will be assumed that the database is working properly.

Functional and Non-Functional Testing:

- The testing team will perform functional and non functional tests at the most of the website.

User Acceptance Testing:

- UAT test execution will be performed by independent users and QA Group will provide their support on creating UAT script.

c. Test Approach

- Manual Testing: Manual testing plays a crucial role in assessing the application's user interface and user

experience. It involves testers interacting with the application, executing test cases, and observing outcomes to identify any discrepancies from expected behavior. This approach is particularly effective in evaluating the usability and aesthetic aspects of the application, such as layout, design consistency, and intuitive navigation.

- Automated Testing: Automated testing complements manual efforts by executing repetitive and regression test cases more quickly and efficiently. Using automated testing tools, scripts were developed to simulate user actions on the Login Page, Sign Up Page, Home Page, Profile Page, and Direct Messages feature. This approach enables the rapid identification of defects and supports continuous integration and delivery processes.

d. Data Approach

1. Test Data Requirements

- User Data: Create test user accounts with varying roles (guests, hosts, admins) to test different functionalities.
- Accommodation Data: Generate test accommodation listings with different attributes (location, price, amenities) to test search and booking features.
- Booking Data: Create test bookings with different dates and payment methods to test the booking process.

2. Test Data Management

- Data Storage: Store test data in a database or file system, ensuring it is easily accessible and can be updated as needed.
- Data Versioning: Version test data to track changes and ensure consistency across test runs.
- Data Masking: Mask sensitive data (e.g., personal information, payment details) to ensure privacy and security during testing.

3. Test Data Generation

- Automated Data Generation: Use automated scripts to generate test data, especially for large datasets or complex scenarios.
- Data Variation: Generate test data that covers a wide range of values and conditions to thoroughly test the application.

4. Test Data Reusability

- Data Templates: Create reusable data templates for common test scenarios (e.g., booking a standard room, updating user profile).
- Data Dependency Management: Manage dependencies between test data to ensure tests can be executed in isolation.

5. Test Data Quality

- Data Validation: Validate test data to ensure it meets the required format and constraints (e.g., valid email addresses, correct currency format).
- Data Cleansing: Cleanse test data to remove duplicates, inconsistencies, or irrelevant information.

6. Test Data Security

- Data Encryption: Encrypt sensitive test data to ensure it is protected during storage and transmission.
- Access Control: Implement access controls to restrict access to test data based on roles and permissions.

e. Levels of Testing

■ Exploratory

PURPOSE: the purpose of this test is to make sure critical defects are removed before the next levels of testing can start.

TESTERS: Testing team.

METHOD: this exploratory testing is carried out in the application without any test scripts and documentation

Timing: At the start of the testing phase.

- **Functional and non Functional Tests**

  Functional Test

  Purpose: The primary purpose of functional testing is to test each function of the software application, by providing appropriate input,and verifying the output against the Functional Requirements.

  TESTERS: Testing team.

  METHOD: The test will be performed according to the Functional test cases that are stored in our Testrail environment.

  Timing: After the Exploratory Testing has been complete.

  Non Functional Test

  Purpose: The primary purpose of non-functional testing is to evaluate the non-functional aspects of a software application, which include performance, usability, reliability, security and localization.

  TESTERS: Testing team.

  METHOD: The test will be performed according to the Non-Functional test cases that are stored in our Testrail environment.

  Timing: After the Functional Testing has been complete

Test Tree

Functional Tests:

a. User Registration and Login
    1. Verify that a new user can register with valid information.
    2. Verify that registration fails with invalid information (e.g., duplicate email).
    3. Verify that a registered user can log in with correct credentials.
    4. Verify that login fails with incorrect credentials.
    5. Verify that a user can reset their password using the forgot Booking Management
    6. Verify that users can view their booking history.
    7. Verify that users can see upcoming and past bookings.
    8. Verify that users can manage their bookings (e.g., cancel, modify dates).
b. Payment Processing
    1. Verify that users can add a payment method to their account.
    2. Verify that users can make a payment for a booking using the selected payment method.
    3. Verify that users receive a confirmation email after making a payment.
    4. Verify that users can view their payment history.
    5. Verify that payment processing is secure and PCI compliant.
c. Profile Management
    1. Verify that users can edit their profile information.
    2. Verify that users can change their account settings (e.g., email preferences, notification settings).

d. Search and filter

    1. Verify that users can search based on price.
    2. search by entering minus price.
    3. search by entering maximum and minimum conversely.

4. Verify that users can clear the search.
5. Verify that users can search by the navigate bar.
6. Verify that users can search by the dates.
7. search by entering by booking wrong dates.
8. Verify that users can search by generation.

Non-Functional Tests:

1. Localization and Internationalization
    a. Verify that all content is translated correctly into supported languages.
    b. Verify that regional settings (e.g., currency, date format) are applied correctly based on user preferences.
2. Mobile Responsiveness
    a. Verify that the website layout adjusts correctly to different screen sizes.
    b. Verify that all functionality is accessible and usable on mobile devices without any issues.
3. Performance Testing
    a. Verify that the website loads quickly on both desktop and mobile devices.
    b. Verify that the website can handle a large number of concurrent users without performance degradation.

- **User Acceptance Test**

  PURPOSE: end users or clients test the software to ensure it can handle required tasks in real-world scenarios, according to specifications.

  TESTERS: Independent outside users .

  METHOD: End users execute the test cases, performing tasks as they would in their day-to-day work. This includes navigating through the application, inputting data, and using the features and functions of the system.

  Timing: After the Functional and Non-Functional tests

3. Execution Strategy

a. Test environment setup

Environment Configuration: Ensure the test environment is configured to match the production environment as closely as possible. Data Setup: Prepare test data, including user accounts, accommodations, bookings, and other relevant data for testing.

b. Test Execution

Unit Testing: Developers will perform unit tests using JUnit/TestNG for backend code and Selenium for frontend code. Integration Testing: Test integration between different modules and components to ensure they work together correctly.

System Testing: Test the entire system to verify that all features and functionalities work as expected. Acceptance Testing: Conduct acceptance tests with stakeholders to validate that the system meets the requirements.

c. Test Automation

Regression Testing: Automate regression tests to ensure that new changes do not break existing functionality.Smoke Testing: Automate smoke tests to quickly verify that the critical functionalities are working after each build.

d. Defect Management

- Defect Identification: Use tools like Jira to track and manage defects.
- Defect Prioritization: Prioritize defects based on severity and impact on the system.
- Defect Resolution: Developers will resolve defects, and testers will verify the fixes.

e. Reporting and Monitoring

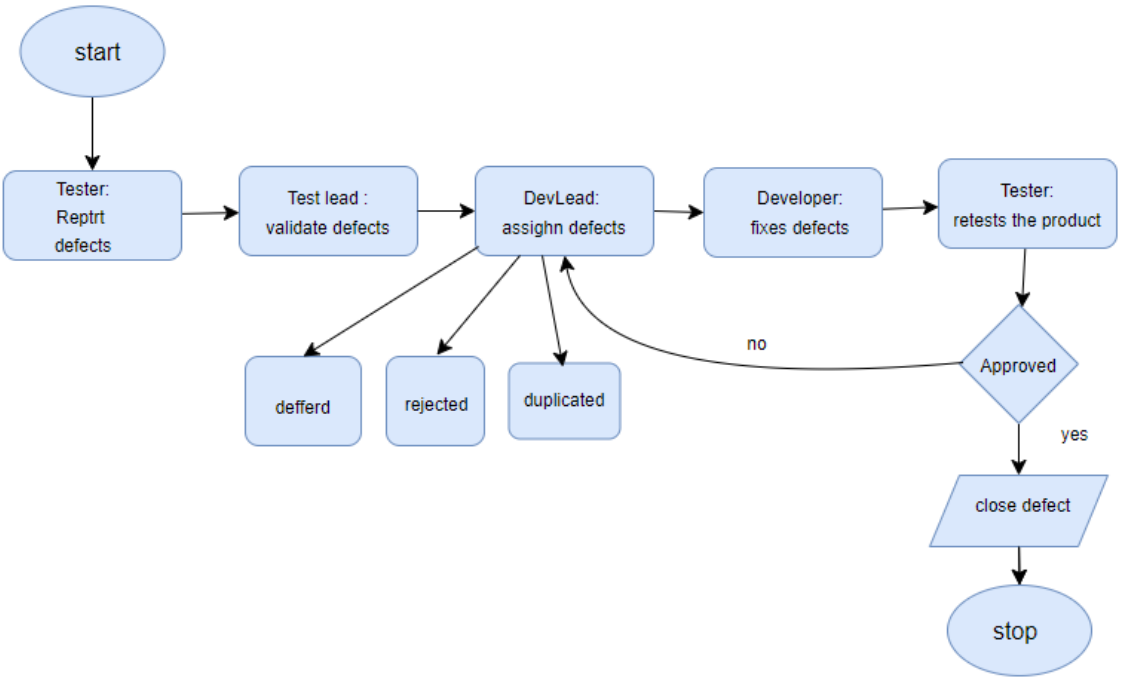
Test Execution Reports: Generate and share test execution reports to track progress and identify issues. Monitoring: Monitor the test execution process to ensure that it is on track and meets the timeline.

f. Continuous Improvement

- Feedback Loop: Collect feedback from stakeholders and team members to identify areas for improvement.
- Process Optimization: Continuously optimize the testing process to improve efficiency and effectiveness.

g. Defect tracking & Reporting

- The presented flowchart illustrates the Defect Tracking Process:



4. TEST MANAGEMENT PROCESS

a. Test Planning

- Define Test Strategy: Define the overall testing strategy, including the use of Selenium Grid for parallel test execution.
- Identify Test Scenarios: Identify test scenarios for each layer of the application (infrastructure, logic, UI).
- Create Test Plan: Create a detailed test plan outlining the testing approach, resources, schedule, and deliverables.

b. Test Design

- Create Test Cases: Develop test cases for each test scenario, specifying the steps to be executed and the expected results.
- Implement Test Logic: Implement the test logic for each test case, including setting up the test environment and configuring Selenium Grid for parallel execution.

c. Test Execution

- Execute Tests: Run the tests using PyCharm and Selenium, utilizing Selenium Grid to run tests in parallel across different browsers and environments.
- Monitor Execution: Monitor the test execution process to ensure tests are running correctly and any issues are promptly addressed.

d. Test Reporting

- Generate Reports: Generate test reports using PyCharm or other reporting tools to track test execution results and identify any failures.
- Analyze Results: Analyze test results to identify trends, patterns, and areas for improvement.

e. Defect Management

- Identify Defects: Use PyCharm or a defect tracking tool to identify and log defects found during testing.
- Prioritize Defects: Prioritize defects based on severity and impact on the application.
- Resolve Defects: Work with the development team to resolve defects, retesting as necessary to verify fixes.

f. Test Closure

- Evaluate Testing: Evaluate the testing process to identify successes and areas for improvement.
- Document Lessons Learned: Document lessons learned from the testing process for future projects.
- Prepare Test Closure Report: Prepare a test closure report summarizing the testing activities, results, and any outstanding issues.

g. Continuous Improvement

- Review and Update: Regularly review and update the test plan, test cases, and test strategy based on feedback and lessons learned.
- Training and Development: Provide training and development opportunities for team members to improve their testing skills and knowledge.

5. Test Environment

Platforms:

- airbnb website
- Windows 11.

Devices:

- A range of desktop computers devices have the firefox,chrome and edge

Network Conditions:

- Tests were performed under various network speeds and conditions (Wi-Fi, 4G, 3G) to assess performance and reliability.

Test Tools:

- pycharm
- selenium
- selenium grid

6. Approvals

Name	Role	Signature
Mahmood Odeh	Project Management	
Etsahe anidgar	Test Lead	
yair amon	Business Analyst	