1. Test Plan Identifier

TestPlan-Shohoz-2024-001

This identifier uniquely distinguishes the test plan for Shohoz, with the number 001 indicating this is the first version for 2024.

2. Introduction

i. Objective:

The purpose of this test plan is to outline the testing scope, approach, tasks, responsibilities, and schedule for ensuring that the Shohoz platform is stable, secure, and provides a seamless user experience. Shohoz is a leading multi-service platform offering a wide variety of services such as bus and launch ticket booking, ride-sharing, and food delivery. This test plan will ensure these core functionalities perform correctly and meet business and user expectations.

ii. Goals:

- **Ensure Functionality**: Verify all features (registration, booking, payment) work seamlessly.
- **Validate User Experience**: Confirm intuitive navigation and responsive design across devices.
- **Test Performance**: Ensure the system handles high traffic (e.g., 1000 concurrent users).
- **Ensure Security**: Protect user data and resolve vulnerabilities (e.g., SQL injection).
- **Validate Integrations**: Ensure smooth functioning with payment gateways and notifications.

3. Test Items

- **User Authentication**:
  - User registration: Signing up via email or phone number.
  - Login: Accessing the user account via username/password.
  - Forgot password: Recovering account credentials.
  - Profile management: Updating user details.

- **Search and Filters**:
  - Service search (bus, launch, ride-sharing) by name, location, and date.
  - Applying filters based on price, distance, rating, and availability.
  - Sorting results by most relevant, price, or duration.
- **Booking**:
  - Booking tickets for bus, launch, or ride services.
  - Adding services to the cart.
  - Proceeding through checkout and payment.
- **Payment Gateway**:
  - Integration with payment services (credit/debit cards, mobile banking).
  - Handling of payment failure, success, and error messages.
- **Wishlist & Cart**:
  - Add services to the cart and wishlist for future use.
  - Managing items within the cart (remove, update quantity).

4. Features To Be Tested

- **User Registration and Login**:
  - Test that new users can successfully register and log in.
  - Test recovery of passwords and changes to profile details.
- **Service Discovery**:
  - Verify the search functionality and ensure filters work correctly.
  - Ensure the user can view service details with accurate and updated information.
- **Booking Process**:
  - Add items to the cart.
  - Test smooth progression to the checkout page.

    - Verify that selected services appear correctly on the checkout page.
- **Payment**:
    - Test all supported payment methods.
    - Test success and failure scenarios, ensuring the system handles both properly.
- **Cross-Platform Compatibility**:
    - Ensure the platform functions correctly across desktop (Windows, macOS), tablet, and mobile devices (iOS, Android).
- **Security**:
    - Test for security vulnerabilities in registration, login, and payment processes, such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

5. Features Not To Be Tested

- **Payment through EMI**: The EMI payment method will not be tested in this cycle.
- **Backend Analytics**: The internal analytics dashboard and reporting tools are not in the scope of this test.
- **Admin Panel**: Any features specific to administrators (e.g., booking management, content management) are not included.

6. Approach

6.1 Testing Methodology:

- **Manual Testing**:
    - For complex user interactions, UI testing, and exploratory testing.
    - Test case execution will be manual for high-traffic functions like booking and payments.
    - **Example**: Manually testing the booking process for different service types to ensure UI elements appear correctly across devices.

- **Automated Testing**:
  - Use **Selenium** for automating regression tests (e.g., login, search, payment process).
  - **Postman** and **RestAssured** will be used to test APIs for bus availability, payment transactions, and user management.
- **Performance Testing**:
  - **JMeter**: Simulate 1000 concurrent users trying to book tickets simultaneously. Measure the response time for each service and ensure the platform can handle high traffic.
- **Security Testing**:
  - Test for vulnerabilities using tools like **OWASP ZAP** and **Burp Suite**.
  - **Example**: Test for SQL injection vulnerabilities in the login and payment pages by entering SQL queries in input fields (e.g., SELECT * FROM users WHERE username='admin' OR '1'='1';).

7. Items Pass/Fail Criteria

- **Pass Criteria**:
  - All core features (registration, booking, payment) work seamlessly without errors.
  - The system handles at least 500 concurrent users during performance tests without degradation in response time.
  - No security flaws are found in the authentication and payment workflows.
- **Fail Criteria**:
  - A critical defect in the booking process (e.g., users cannot complete payment).
  - A performance test shows unacceptable slowdowns (e.g., booking response time > 10 seconds for more than 1000 concurrent users).
  - Security vulnerability (e.g., users can access unauthorized data).

8. Suspension Criteria

Testing will be suspended if:

- **Critical Defects**: If a defect blocks testing of major workflows such as registration, booking, or payment.
- **Environment Failure**: If the test environment is not available or is unstable, preventing testing from proceeding.
- **Unresolved Major Dependencies**: If integration with external services (e.g., payment gateways) fails and is not resolved.

9. Test Deliverables

- **Test Plan**: This document, which defines the strategy and scope of testing.
- **Test Cases**: Detailed test scenarios and steps for validating the functionality of Shohoz.
- **Test Logs**: Logs of executed tests, including pass/fail results.
- **Bug Reports**: A detailed report of defects, their severity, and their impact on the system.
- **Test Summary Report**: A comprehensive report summarizing the testing outcomes, defect statistics, and coverage.
- **User Documentation**: Help files or guides to assist users with navigating the platform.
- **Release Notes**: Information about new features, fixes, and changes included in the release.

10. Testing Tasks

- **Test Case Development**: Write detailed scenarios for registration, login, search, booking, and payment.
- **Test Execution**: Execute functional and performance tests, document results, and log defects.
- **Defect Management**: Track all defects via JIRA, from discovery through resolution.

- **Regression Testing**: Run automated scripts every time a new release or update is pushed to ensure no existing functionality is broken.

11. Environmental Needs

- **Devices**:
  - **Desktop**: Windows 10/11, macOS.
  - **Mobile**: Android 12+, iOS 15+ (smartphones and tablets).
- **Browsers**:
  - Chrome, Firefox, Safari, Edge (latest versions).
- **Testing Tools**:
  - **Selenium**: For browser automation.
  - **JMeter**: For load and stress testing.
  - **OWASP ZAP**: For security scanning.

12. Responsibilities

- **QA Team**:
  - Test case development and execution.
  - Manual and automated test execution.
  - Bug reporting and verification of fixes.
- **Development Team**:
  - Fix defects reported by the QA team.
  - Ensure features are implemented according to the requirements.

13. Training Needs

- Training for the QA team on Shohoz's new features and any specific tools used for testing (e.g., Selenium, JMeter).

- Developers need training on security testing and the importance of secure coding practices.

14. Schedule

Task	**Start Date**	**End Date**	**Duration**
Test Plan Creation	2024-01-02	2024-01-03	2 days
Test Case Development	2024-01-04	2024-01-08	5 days
Test Execution	2024-01-09	2024-01-18	10 days
Bug Reporting & Fixing	2024-01-19	2024-01-25	7 days
Final Report Submission	2024-01-26	2024-01-27	2 days

15. Risks & Contingencies

- **Risk**: Limited testing time due to project deadlines.
    - **Mitigation**: Prioritize critical features and allocate more resources to testing during peak times.
- **Risk**: Payment gateway integration fails or is delayed.
    - **Mitigation**: Use a mock payment gateway for initial testing and delay full payment testing until integration is stable.

16. Approvals

- **Prepared By**: Tanzila Kamal (QA Engineer).
- **Reviewed By**: MOHOSHI HAQUE