Test Plan

Objective

The primary objective of this test plan is to ensure that the web application is production-ready. This includes verifying functionality, performance, security, and user experience while identifying and mitigating potential risks before deployment.

Scope

- Login functionality: Username, password, login button
- Validate that the dashboard and navigation bar function as intended.
- Ensure the ticket purchase flow works
- Ensure the contact form flow works
- Confirm that all features meet functional, non-functional, and UI/UX requirements.
- Include accessibility, performance, and security testing.
- Deploy the application to production following a comprehensive CI/CD workflow.

Assumptions

- The application is bug-free.
- The environment is set up locally and mirrors production.

Types of Testing

Functional Testing

- Verify that the login functionality works as intended
- Navigation bar actions (Home, Tour, More → Contact).
- Tour section interactions (city details, ticket purchase pop-up, field validations).
- Contact Form

Non-Functional Testing

- **Performance Testing**: Ensure the application performance
- **Security Testing**: Protect against vulnerabilities like SQL injection
- **Accessibility Testing**: Confirm WCAG compliance for all UI components

Compatibility Testing:

- Verify compatibility across various browsers and devices
  - Browsers (e.g., Chrome, Firefox, Safari, Edge)
  - Devices (e.g., desktop, mobile, tablets)
  - Operating Systems (e.g., Windows, macOS, Linux, iOS, Android)

Regression Testing

- Ensure that new changes do not impact existing functionality

Exploratory Testing

- Uncover edge cases not defined in test cases.

End-to-End Testing

- Simulate user workflows from start to finish, e.g., logging in and navigating to the dashboard

CI/CD Pipeline:

- Setup automation tests to be triggered as part of CI/CD pipeline

User Acceptance Testing (UAT)

- Verify that the application meets business requirements and expectations

Post-Deployment Validation

- Test in the production environment to confirm successful deployment

Testing Environments

Environment	Purpose	Setup
Local	Initial development and unit testing	Local machine setup
Dev	Smoke and Testing of new features	Pre-configured environment

QA/Staging	Comprehensive functional and non functional testing and regression testing	Production-like environment

Production	Final validation after deployment	Live environment with real data

Testing Requirements

- Access to test environments
- Test data resembling real-world scenarios
- Testing tools (Cypress, Lighthouse, K6, WAVE etc)
- CI/CD pipeline integration for automated testing

Tools and Frameworks

- **Test Automation Framework**: Cypress (for E2E and integration testing)
- **API Testing:** Postman, Cypress
- **Performance Testing**: K6 for Backend APIs and Lighthouse for frontend.
- **Security Testing**: OWASP ZAP
- **Monitoring and Logging**: New Relic or Grafana with Prometheus
- **CI/CD Pipeline**: Jenkins, GitHub Actions, or CircleCI
- **Code Quality and Coverage**: SonarQube
- **Browser Testing**: BrowserStack or Sauce Labs for cross-browser compatibility
- **Accessibility testing:** WAVE API or Wick A11y Testing Environments
- **Bug Reporting & Tracking**: JIRA

Test Plan Phases

Phase 1: Requirement Gathering and Test Planning

- Review product requirements and gather requirements
- Define objectives, scope, and testing criteria.
- Identify risks and mitigation strategies.

Phase 2: Test Case Design

- Create test cases and scripts for functional, performance, and security tests.

Phase 3: Test Execution

- Run automated and manual tests in all environments.
- Identify and log defects, and work on resolutions.

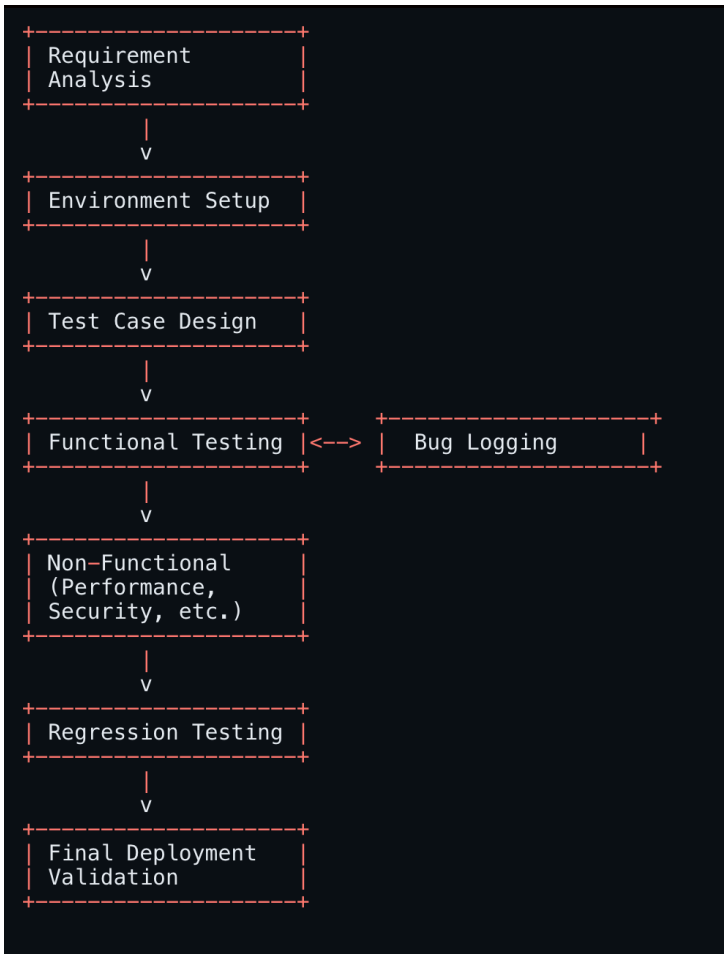
```
Feature: Login functionality

  Scenario Outline: Test login with valid and invalid credentials
    Given the login page is displayed
    When I enter "<username>" and "<password>"
    And I click the login button
    Then I should see "<expectedMessage>" page

    Examples:
      | username       | password     | expectedMessage    |
      | test           | test         | THE BAND           |
      | invalidUser    | validPass123 | 404 Not Found      |
```

Phase 4: Reporting

- Share test results and coverage metrics.
- Highlight risks and unresolved issues.

Phase 5: Post-Deployment Validation

- Execute smoke tests in the production environment.
- Monitor application logs and metrics for anomalies.



CI/CD Workflow

- **Code Commit**: Code pushed to feature branch triggers unit tests.
- **Build**: Application build is initiated in CI (e.g., CircleCI, Jenkins)
- **Static Code Analysis**: Tools like SonarQube verify code quality.
- **Functional Testing**: Automated functional tests run for feature and integration branches.
- **Deployment to QA**: After passing tests, code is deployed to QA.
- **Comprehensive Testing**: Manual and automated tests in QA environment.
- **Approval & Deployment to Prod**: After approval, code is merged into master and deployed to production.
- **Post-Deployment Validation**: Smoke tests and monitoring of production environment.

```
Code pushed → Unit Tests
Build created → Deployed to DEV
Tests in DEV -> Deploy to QA
Tests in QA → Approval → Merge to Master
Deployment to Production → Smoke Tests
```

Bug Template

Field	Description
Bug ID	Unique identifier
Summary	Brief description of the issue
Steps to Reproduce	Detailed steps to reproduce the issue
Expected Result	What should have happened
Actual Result	What actually happened
Serverity	Critical, High, Medium, Low
Environment	Local, Dev, QA, Production

Test Case Design (Sample test cases - Detailed test cases are in Test Case Document)

Feature: LOGIN FUNCTIONALITY

Scenario: Login with valid credentials

1. **Given** the login page is displayed
2. **When** the user enters a valid username and password
3. **When** the user clicks on the login button
4. **Then** the user should see the dashboard page

Scenario: Login with invalid credentials

1. **Given** the login page is displayed
2. **When** the user enters an invalid username and password
3. **When** the user clicks on the login button
4. **Then** the user should see the 404 error page

Scenario: Login with an invalid password

1. **Given** the login page is displayed
2. **When** the user enters a valid username and an invalid password
3. **When** the user clicks on the login button

4. **Then** the user should see the 404 error page

Scenario: Login with an invalid username

1. **Given** the login page is displayed
2. **When** the user enters an invalid username and a valid password
3. **When** the user clicks on the login button
4. **Then** the user should see the 404 error page

Scenario: Login with empty credentials

1. **Given** the login page is displayed
2. **When** the user keeps the username and password field empty
3. **When** the user clicks on the login button
4. **Then** a popup appears with the following text `Please fill out this field`

Scenario: Login with an empty password

1. **Given** the login page is displayed
2. **When** the user enters a valid username and an empty password
3. **When** I click on the login button
4. **Then** a popup appears with the following text `Please fill out this field`

Scenario: Login with an empty username

1. **Given** the login page is displayed
2. **When** the user enters an invalid username and a valid password
3. **When** I click on the login button
4. **Then** a popup appears with the following text `Please fill out this field`

Risk Assessment

Risk	Likelihood	Impact	Mitigation
Bugs in critical workflows	High	High	Prioritize regression and E2E tests.
Performance degradation under load	Medium	High	Run performance tests with realistic load.
Security vulnerabilities	Medium	High	Perform comprehensive security testing.
Deployment failure	Low	High	Validate deployments in staging first.

Entry and Exit Criteria

Entry Criteria

- Development is complete
- Unit and integration tests pass in the local environment
- Test environments are stable and accessible

Exit Criteria

- All critical and high-severity defects are resolved
- Test coverage meets predefined benchmarks
- Stakeholder sign-off is obtained for production deployment

Report and Documentation

- **Test Execution Report**: Includes pass/fail status, defect logs, and coverage metrics.
- **Risk Assessment Document**: Details identified risks and mitigation strategies.
- **Deployment Checklist**: Ensures that all prerequisites are met before production release.

Post-Deployment Validation

- Run smoke tests in production to verify critical functionality (e.g., login, navigation)
- Monitor application logs for errors

- Validate rollback mechanisms in case of failure

Conclusion

This test plan ensures the web application is robust, secure, and meets user expectations before deployment. It incorporates functional, non-functional, and post-deployment validation for comprehensive testing coverage.