ACME Airlines – ABC Flight Booking Portal

Test Strategy

**1) Introduction**

The purpose of the test strategy for the ACME Airlines – ABC Flight Booking Portal is to provide a high-level approach to be employed when testing the software and when evaluating the results of that testing. The testing Strategy will influence tasks related to test planning, test types, test script development, and test execution.

**2) Test Scope and Objectives**

To define the testing scope of ACME Airlines – ABC Booking Portal, the following areas were examined:

1. Testing tasks
2. Test types by task
3. Mocked Data sources/Systems [E.g., Global Distribution System]
4. Application and System interfaces [ E.g., Payment Gateway, Global Distribution System]
5. Testing Environments
6. Testing Tools

The key objectives are as follows:

1. Detailed analysis of the significance of the ABC Flight booking portal to the business and based on the analysis derive the extent of testing required.
2. Identification of the key test deliverables and test coverage to be achieved across for the Application and System under test.
3. List all integration points for instance like the Booking Portal with Global Distribution System, Booking Portal with Payment Gateway etc.
4. Identification of the test environments and mock-ups for API’s and system interfaces.
5. Identify the non-functional requirements like performance and security requirements of the project.

**3)Test Approach**

**3a) Test Levels**

|  |  |
| --- | --- |
| **Test Type** | **Description** |
| Unit Test | Unit Testing focuses on a specific component in the system in isolation for instance testing of the user registration module. |
| Integration Test | Integration Testing focuses on the interaction between multiple components. E.g.: Interfaces between Booking portal and Global Distribution System. |
| System Test | System testing is the functional and non-functional [security, accessibility and performance] testing of the entire deliverable system, and the interfaces between the various components. E.g.: Exercising a complete workflow of the flight booking along with the systems and the APIs |
| Operational Acceptance Test | Operational readiness/acceptance testing is to identify any potential issues with the production environment setup before users accesses the system. |
| Beta / Business Verification Test | Beta/ Business Verification testing is conducted by a small number of experienced users/ product owners in Agile environment try the product in a production mode and report defects and deficiencies. |

**3b) Test Techniques**

|  |  |
| --- | --- |
| **Technique** | **Description** |
| Boundary value analysis [E.g.: Test of User Registration] | Testing modules by supplying inputs at, within and beyond the specified boundaries. |
| Cause-effect Testing [E.g.: Each stages of the Booking Workflow] | Supplying input values to cause all possible output values to occur. |
| Comparison Testing [E.g. Earlier integrations with the Global Distribution system and current] | Comparing output of system under test with another reference system |
| Risk Based Testing [for booking portal e.g., add passenger information] | Risk based testing is prioritizing the feature's, modules and functions of the Application Under Test based on impact and likelihood of failures. |
| Model Based Testing [for integration with the Global Distribution System] | Model based testing is a software testing technique where run time behavior of software under test is checked against predictions made by a model. A model is a description of a system's behavior. |

**4) Testing Environments**

Add, modify or delete testing tasks as appropriate, and document the required platforms and database(s) needed for each task.

The following table documents the testing environment criteria for each testing task:

| Testing Task | Platform/ Databases/Mocks |
| --- | --- |
| Unit Test | Sandbox Environment [limited to Booking Portal alone] |
| Integration Test | Global Distribution System Mocks. CBA Payment Service API Mocks/ Dummy Cards for testing. |
| System Test | Availability of Prod server for exercising limited use cases. |
| Operational Acceptance Test | Prod Server /Databases |
| Beta Test | Prod Server /Databases |

**5) Test Automation and Management Tools**

The following testing tools will be used:

| Testing Tool/Frameworks | Purpose |
| --- | --- |
| Selenium, Java, Cucumber, Maven | UI Automation |
| Rest Assured, Java | API Automation |
| SQL | Database Testing |
| Jmeter | Performance Testing |
| JIRA | Defect and Test Management |
| Confluence | Knowledge Management |

**6)** **Key Deliverables**

1. Set-up of test environment including mocks for the GDS
2. Set-up of test data
3. Perform testing of the entire booking workflow
4. Monitoring and collating results.
5. Initiating rework in the case of defects.
6. Obtaining user acceptance.

**7) Assumptions, constraints and dependencies**

* Major assumption while writing this test strategy is the availability of the test environments/mocks for integration/system testing the project deliverables for instance Global distribution system and Payment gateway.
* One other assumption is availability of the operations engineers and product owners to conduct on prod testing and provide feedback.
* Dependency in the form of other legacy systems migration, timelines for upgrades and data migration.

**Q2) Sample Test Scenarios**

**Test Case 1)**

High Level Description: Verify if a user can register with ABC Airline Portal

Given User types in Airline Portal URL in his browser

And User is provided with login options

When he clicks on Register button

Then he should be able to enter his details in the Registration page

And the registration should be successful

**Test Case 2)**

High Level Description: Verify if the flight information and its prices are retrieved from the Global Distribution System

Given User types in Airline Portal URL in his browser

And User clicks on 'Continue as Guest' link

And User selects Departure and Arrival Airports

When User selects a booking date

Then flights and its prices should be fetched from the Global Distribution System

**Test Case 3)**

High Level Description: Verify if the user can make a payment successfully

Given registered User is on the Airline Portal -> Booking form

And User selects the Best Price and flight time

And User adds the passenger information

When User clicks on 'Make Payment'

Then the User should be navigated to the payment gateway

And User should be able to enter his card information

And the payment should be completed successfully

**Test Case 4)**

High Level Description: Verify if e-mail/SMS is sent to the register user upon successful booking

Given registered User is on the Airline Portal -> Booking form

And User selects the Best Price and flight time

And User adds the passenger information

When User completes the booking

Then an email/SMS should be sent to the registered user's email address/mobile number

**Test Case 5)**

High Level Description: Validate if the guest user can make the booking upon registration

Given registered User is on the Airline Portal -> Booking form

And User selects the Best Price and flight time

And User adds the passenger information

When User clicks on 'Make Payment'

Then an error message should be displayed to inform him that he should register himself before making a payment.

And after registration, user should be allowed to make the payment successfully

**Q3) API Testing**

Following approaches can be applied for the API testing

1. API Specification based testing [Open API spec or WSDL]

* Validating for data elements [JSON Paths and XPaths] as provided in the specification.
* Validating the values corresponding across many datasets for each of the elements.

1. Fault based and Error condition testing – The focus here is to generate API faults or to produce error conditions. E.g. Trying a new value outside the recommended value of an enumerated type
2. Authentication and Security testing – Test for authorization implementations like Basic Auth or OAUTH 2.0
3. Query Parameter Combination – testing for all possible combination of query parameters.

**Q4) Automation Strategy**

Since the testing at times in the project needs to be done directly on Prod and lack of Sandbox environments, automation plays a major role in mocking and simulating prod like scenarios. Use of tools like Fiddler or Traffic parrot to capture the live requests and responses and Rest-Assured for API automation, Cucumber to drive a BDD approach to make it more non-technical and Selenium for UI are some of the steps that I would consider for the automation strategy.