**Case Study**

Acme Airlines is planning to launch a new web portal for flight bookings. This portal will be a responsive web application. It will allow users to book domestic and international flights as a guest as well as allow users to register and store user information for ease of booking in future.

This application will require using Galileo Global distribution System to manage bookings. Galileo doesn't provide a Sandbox environment and only have a production environment to work with.

CBA is partnered to act as a payment gateway. Payment service will be integrated along with flight booking system.

**Q.1 - Please identify your testing strategy and approach to testing this application in an Agile development environment. Please list down any assumptions and questions you have about the application brief.**

Test strategy should contain Testing Scope and Overview, Test Approach, Test Environment,Test Infrastructure, Testing Phases, Testing tools related to Automation and Test management, Scope of Items, Entry and Exit Criteria, Evaluation criteria, Pass/Fail Criteria.

Approach:

During the first iteration identify the testing team, testing tools; identify and break down the requirements and creation of use case scenarios, identify the risks.

Subsequent iterations should be planned considering the below mentioned points

* Breaking down the requirements based on priority for backlog grooming
* UI validations should be handled as part of Automation
* Run functional/unit tests on each module separately
* Each API should be tested for data validation and with multiple data sets
* Should cover both the positive and negative scenarios and should be recorded using appropriate tools
* Integration tests w.r.t. Global Distribution System and Payment Gateway
* Cross browser testing
* Performance testing, load testing, stress testing
* Language/Globalization testing (if applicable)

Assumptions:

• Galileo Global distribution System & CBA payment gateway provides API specifications

**Q.2 - Please list down test scenarios you think will cover the majority of application features. Mention any negative, positive and edge cases based on the application details.**

Below listed are some of the scenarios identified, but not limited to,

Verify that a registered user is able to login successfully and proceed for a new booking.

Verify that a guest/registered user is able to search the flight details based on input criteria.

Verify that the date fields ae working properly dd/mm/yyyy or mm/dd/yyyy and try to play with date formats for leap/non-leap years.

Verify that search results have real-time flight details, timings, and availability.

Verify that the user is able to sort the search results based on price, duration etc.

Verify that clicking the search results open complete details for flight.

Verify that departure & arrival date fields are working as expected (They should not accept any date before current date & arrival date should always be later than departure date/time).

Verify the departure & arrival Airports should not be same/null and validation should be done at UI.

Verify one way & round trip features are working properly.

Verify that user is presented with a graphical view of the airline’s seating arrangement along with seat number and availability status.

Verify that pricing of different types of seats is displayed to the users.

Verify that user cannot select or is not permitted to select seats that are already booked or not allowed for booking.

Verify that after selecting seats, entering passenger details and making [payment](https://en.wikipedia.org/wiki/Payment) the selected seats get booked.

Verify that on successful booking the ticket should be visible and downloadable.

Verify that user also receives confirmation mail/SMS along with tickets on the email provided while filling the details.

Verify that the registered user can view the booking history inside his login

Verify that the user can be able to store the passenger information In master passenger list

Verify that the guest user is redirected towards registration process after successful booking

Verify that a new user is providing email and phone details as part of registration process.

Verify that the required/mandatory fields are marked with \* against each field

Verify that for better user interface dropdowns, radio buttons and checkboxes fields are displayed wherever possible instead of just textboxes

Verify that proper UI validation is in place where ever applicable

Verify the user is providing unique username and validate with existing user list.

Verify that the user registration process completed successfully after providing necessary details.

Verify the maximum number of seats that a user can book, error message should be displayed when selecting more seats than permitted results.

Verify that the user is presented with additional options like extra luggage, extra legroom, foods/beverages etc and selecting the same results in additional cost with the booking amount.

Verify that user also receives confirmation mail along with tickets on the emails provided while filling the details.

Verify that if one user selects the seats and any other user is not able to reserve the same seats

Verify the systems data by cancellation of booking in the middle of transaction

Verify that user can also cancel the tickets booked by entering the mandatory details and the amount after deducting the cancellation fee gets refunded back to user.

**Q.3 - Considering this application has a number of third-party API integrations, how would you go about testing the system?**

List down each & every API integration and try to identify the integration hooks for testing.

Independently test each API Call

Verify the responses for valid & invalid inputs

Verify the security mechanism/encryption used for the API call

Verify the technical failures while the payment made but not confirmed

Verify if the applications have multiple payment options like Credit card and PayPal together, both payment options need to individually tested from end to end.

Verify that user can also cancel the tickets booked by entering the mandatory details and the amount after deducting the cancellation fee gets refunded back to user.

Verify if transaction can be refunded or void successfully

Verify the scenario where the payment made but haven’t got back the success response to application.

Security testing w.r.t passing of payment related confidential information

Verify that the customer gets some kind of transaction confirmation notification like Order confirmation email, etc. if the transaction is successful.

Verify what happens if a payment fails or payment processor stops responding- is there any error message?

Verify what happens when a customer session expires due to idle time

**Q.4 - What will be your automating strategy, approach and choice of tools?**

Automation test strategy should be considered if the project consists of test cases/suites that are repetitive in nature, complex scenarios that are prone to human errors and involves non-functional testing.

**Approach**

* Understanding of project requirements thoroughly
* Benchmarking existing test automation tool for comparison on below considerations

Ease of use, cost involved for procurement, support for web/Desktop/Mobile application requirements, supported programming languages, support for Data Driven testing, Reports Generation, maintenance, cross browser compatibility, DevOps Integration, ROI

* Choice of tools:
  + Selenium – as it supports multiple languages and support for frameworks
  + REST Assured – For REST API Automation Testing
  + Jmeter – for load testing
  + Jenkins – Continuous Integration