Q.1 - Please identify your testing strategy and approach to testing this application in an Agile

development environment.

Answer

1. Test planning
2. Reviewing the test plan by BA
3. Writing testcases for each functionality identified in the testplan
4. Revising the testplan/testcase every sprint.
5. Executing the testcases
6. Raising bugs and follow up of it.
7. Preparing test report at the end of the sprint with details of functionality tested/testcase passed/failed/bugs
8. Identifying the regression scenarios
9. Automating the regression scenarios and running at the end of each sprint.

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.

Diving the functionality this application allows in the testplan and testing each functionality individually.

**TestPlan:**

|  |  |  |
| --- | --- | --- |
|  | Functionality | Description |
| 1 | Portal Homepage | UI testing includes checking of all the expected homepage elements to appear on page and checking all the buttons/textbox functionality. |
| 2 | User Registration Page | 1. If user clicks on registration, check the functionality of user registration. Once registered flight booking has to be done on registered user 2. Else test the functionality of allowing the user as guest and allow him to book ticket as a guest user. |
| 3 | Arrival/departure | 1. The page should arrival/departure dropdown along with different combinations of destinations populated in the destination field. |
| 4 | Booking date | 1. The page should populate the date according to user selection and show all the flights to that particular destination selected. 2. Check sorting according to time and amount. 3. Check for connecting flights feature. |
| 5 | Galileo Global distribution System | API testing to see if the request and response between the systems are as per the expectation. |
| 6. | Select the price and add passenger information | 1. Check if the price is populated according to user’s selection 2. Additional cost of other features should be added in the final price 3. After the price is selected, application should accept the user information. 4. Check if the user information page shows all the fields as per the expectation   . |
| 7 | Make Payments | 1. Payment gateway testing 2. Penetration testing 3. Payment successful test 4. Payment declined test 5. Timeout issue test |
| 8 | Register User/Registered user | 1. If user is not registered, take the user to user registration page, take user inputs and store the information. 2. validation for registration page. 3. Database validation to see if the user details are stored. 4. UI validation to display the registered user details. 5. Check for option to edit user details. 6. Once successfully logged in as registered user/ saving the user details successfully, database validation to check if the journey data is stored in the database according to what the user has selected. 7. Check UI if the journey details are being displayed according to user selection by fetching from database |
| 9 | SMS/email | 1. Once all the previous steps are completed, check if the mail/sms is being triggered to the user to his registered mail id/ phone number. 2. Check for the subject line, message body, icons in the mail 3. Check for sms formatting along with message content |

Q.3 - Considering this application has a number of third-party API integrations, how would you go

about testing the system.

Integration testing once each individual component has been developed.

Mostly achieved via API testing by testing the request and responses and how our system would react to every response and how the UI elements are populated /database gets updated according to the response received from various application

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

Automation using   
1. Selenium-> UI automation  
2. RestAssured -> API testing

3.Cucumber -> BDD approach

4.JAVA/C# -> Wide library support for various types of testing

5.Maven – Build tool

6.GIT- Source control

7.Azure/Jenkins – CI/CD

8.Json/XL – for data

Approach would be BDD Page Object Model Framework, as it is very easy to maintain and for agile systems and changing UI requirements, it’s easy to access class files of each page and modifying rather than modifying the entire test suite.