Section 3: Exploratory Testing

Exploratory testing is a dynamic and unscripted testing approach where testers use their intuition, creativity, and domain knowledge to explore an application and identify potential issues. Given the web application for booking flights and hotels, my approach to exploratory testing would involve the following steps:

1. **Understanding the Application:** Begin by familiarising myself with the application's main functionalities, user interfaces, and workflows. This would include logging in, searching for flights and hotels, making bookings, managing bookings, and making payments.
2. **Exploratory Testing Scenarios:**
    a. ***End-to-End Booking Flow:*** Test the complete booking process, from searching for flights/hotels to confirming the booking. Verify that each step works smoothly and that users can successfully complete the process.
    b. ***Data Validation:*** Enter invalid data (e.g., incorrect dates, negative values) to check how the application handles such inputs. Ensure proper validation and error messages are displayed.
    c. ***Concurrency and Load Testing:*** Open multiple tabs or browsers to simulate concurrent users. Evaluate the application's performance and responsiveness under load.
    d. ***Cancellation and Modification:*** Test the process of cancelling or modifying bookings to ensure these features work as expected.
    e. ***Error Handling:*** Intentionally trigger errors, such as network disconnects or server timeouts, to see how the application handles and recovers from such situations.

Exploratory testing in the context of a flight and hotel booking web application is a powerful approach to uncover a wide range of issues, from usability glitches to critical security vulnerabilities, and helps ensure a robust and user-friendly application.

Section 4: Regression testing

Regression testing is an aspect of quality assurance that ensures that new software updates do not inadvertently introduce new defects or break existing functionalities. Here's my approach to performing regression testing for the updated application:

1. **Test Case Selection:**
    - **Priority-Based Selection:** Identify and prioritise high-risk and critical areas of the application that are more likely to be affected by the update. For example, if the update involved changes to the payment processing module, prioritise test cases related to payments and transactions.
    - **Comprehensive Coverage:** Select a representative set of test cases that cover various features, workflows, and integrations within the application. This ensures a comprehensive check of the application's behaviour.
    - **Previous Defects:** Include test cases related to defects that were previously identified and fixed. This helps ensure that the same issues do not reappear.
2. **Test Data Management:**

- **Data Reusability:** Utilise existing test data that was used during initial testing. This helps maintain consistency and ensures that test cases are executed with realistic data.
- **Variability:** Create variations of the test data to simulate different scenarios, such as edge cases, boundary conditions, and negative inputs. This helps uncover potential issues.

3. **Automated Testing:**
   - **Test Automation Tools:** Utilise test automation tools to automate repetitive and time-consuming test cases. Automated tests can be run quickly and consistently, making them suitable for regression testing.
4. **Regression Test Suite Management:**
   - **Maintain Test Suite:** Keep the regression test suite up to date by regularly reviewing and updating test cases based on the evolving requirements and functionalities of the application.
5. **Defect Tracking and Reporting:**
   - Defect Verification: If any defects were identified and fixed in the previous version, verify that they have been resolved in the updated version.
   - Documentation and Reporting: Document the results of regression testing, including any defects found, their severity, and steps to reproduce. Share the findings with the development team for resolution.

By following this approach, I aim to effectively assess the impact of the software update, maintain the integrity of existing functionalities, and ensure the overall stability and quality of the application.