**a) Questions to Understand the Requirement Better:**

To gain more clarity on the requirement, I would ask the following questions:

1. **Flight Route Clarification**:
   * Should the system handle only direct flights, or should it include connecting flights? cities
2. **Flight Information**:
   * What specific flight details should be displayed (e.g., flight number, departure time, duration, price)?
3. **Search and Results Display**:
   * How quickly should results be displayed to the user?
   * Should the user be able to sort or filter results (e.g., by price, duration, or airline)?
4. **Edge Cases**:
   * How should the system handle situations when no flights are available for a given route and date?
   * Should there be error messages for invalid searches (e.g., same origin and destination, or past dates)?
5. **Business Rules**:
   * Are there any business preferences, like prioritizing certain airlines or partners in the flight search results?

**b) Defining an Appropriate Minimum Viable Product (MVP):**

The MVP should focus on delivering the core functionality with minimal complexity, allowing users to search for flights by route and date. The key elements of the MVP include:

1. **Basic Flight Search Functionality**:
   * Input fields for Origin (departure) and Destination (arrival).
   * A Date Picker for users to select a flight date.
   * A Search Button that triggers the flight search based on the input.
2. **Flight Results**:
   * Display available flights for the selected route and date.
   * Show basic flight details: flight number, departure/arrival time, and duration.
3. **Basic Validation and Error Handling**:
   * Validate that origin and destination are not the same.
   * Display a message if no flights are found for the selected route and date.
4. **Minimal User Interface**:
   * Simple, clean interface for input fields and result display.

c) **Suggestions to Improve Collaboration between Tester and Product Owner**:

1. **Regular Communication**: Schedule regular touchpoints (e.g., weekly meetings or standups) where the tester can share feedback, raise concerns, and get clarifications on requirements.
2. **Early Involvement**: Engage testers early in the requirement gathering and review process so they can contribute to identifying edge cases and potential gaps.
3. **Clear Documentation**: Ensure that requirements are clearly documented with well-defined user stories and acceptance criteria, reducing misunderstandings.
4. **Quick Feedback Loops:** After each testing cycle, provide prompt feedback to the PO regarding any issues or improvements, helping to guide the development process.

d) **User Story & Acceptance Criteria:**

**User Story**:

* As a user, I want to search for flights by route and date So that I can find a flight that suits my travel plans.

**Acceptance Criteria**:

1. **Scenario 1**: Successful Flight Search
   * Given the user has selected valid origin, destination, and date,
   * When the user clicks on the "Search" button, then the system should display the available flights.
2. **Scenario 2**: No Flights Available
   * Given the user has selected a valid route and date,
   * When no flights are available for that route and date, then the system should display a "No flights available" message.
3. **Scenario 3**: Invalid Input (Same Origin and Destination)
   * Given the user has selected the same city for both origin and destination,
   * When the user clicks on "Search", then the system should display an error message asking the user to choose different locations.

**Collaboration with PO**:

Collaborate with the PO to refine the user story, ensure that the scenarios cover all business rules, and define clear acceptance criteria for both functional and non-functional aspects.

**Test Cases:**

1. **TC 1: Valid Flight Search (Normal Operation)**
   1. **Steps**:
      1. Input a valid origin, destination, and date.
      2. Click on "Search".
   2. **Expected Outcome**: The system displays available flights for the specified route and date.
2. **TC 2: No Flights Available**
   1. **Steps**:
      1. Input a valid origin, destination, and date with no flights available.
      2. Click on "Search".
   2. **Expected Outcome**: The system displays a "No flights available" message.
3. **TC 3: Invalid Input (Same Origin and Destination)**
   1. **Steps**:
      1. Input the same city for both origin and destination.
      2. Click on "Search".
   2. **Expected Outcome**: The system displays an error message asking to select different locations.
4. **TC 4: Invalid Date (Past Date)**
   1. **Steps**:
      1. Input a valid route and a past date.
      2. Click on "Search".
   2. **Expected Outcome**: The system displays an error indicating that past dates are not allowed.
5. **TC 5: Empty Fields**
   1. **Steps**:
      1. Leave either the origin, destination, or date field empty.
      2. Click on "Search".
   2. **Expected Outcome**: The system displays an error message asking to fill in the missing fields.

**Test Case Priorities:**

1. **Test Case 1: Valid Flight Search** – **Priority: Normal**
   * This is the core functionality, but not having flight results doesn’t block the entire system.
2. **Test Case 2: No Flights Available** – **Priority: High**
   * Users should know if no flights are available; failing this could lead to confusion.
3. **Test Case 3: Invalid Input (Same Origin and Destination)** – **Priority: Showstopper**
   * If invalid input is accepted, it can break the flight search and confuse users, stopping progress.
4. **Test Case 4: Invalid Date (Past Date)** – **Priority: High**
   * Ensures users can’t search for flights in the past, preventing irrelevant searches.
5. **Test Case 5: Empty Fields** – **Priority: Showstopper**
   * Users must be required to input all necessary information; without this, the feature would be non-functional.

**g) Types of Testing for This Requirement:**

1. **Functional Testing**: To verify that the core functionality of flight searching works as expected.
2. **Usability Testing**: Ensure the interface is user-friendly and intuitive.
3. **Negative Testing**: Ensure proper handling of invalid input like same origin and destination, missing fields, and past dates.
4. **Performance Testing**: To ensure the system can retrieve and display flight results within an acceptable time frame.
5. **Integration Testing**: Test the system’s integration with the flight data provider API to ensure accurate and real-time data.

**h) Exit Criteria for Testing This Requirement:**

1. All critical test cases, including valid searches, error handling, and edge cases, must pass.
2. No high-priority defects remain unresolved.
3. Functional, usability, and performance tests meet the acceptance criteria.
4. Regression testing has been completed to ensure new features have not broken existing functionality.
5. The Product Owner has accepted the testing results, and the feature is ready for release.