**Objective**

A large systems supplier failed to reach a vital contract milestone. The contract required preparing the Flight Management System for its first flight integration with a US military combat helicopter. Late in the project, the supplier realised their resources were insufficient to meet the technical, budgetary, and scheduling constraints. With the possibility of failing to fulfil a key milestone for the United States Army, the supplier turned to Performance Software for assistance. Performance Software was highly recommended because to its high-quality results, quick turnaround, and extensive FMS development and testing experience.

**Pseudo-code for Flight Management System**

// Flight Search Service

function searchFlight(departure, arrival, date) {

if (departure is empty or arrival is empty) {

return "Please enter valid search criteria.";

}

flights = findFlights(departure, arrival, date);

if (flights is empty) {

return "No flights available for the specified criteria.";

} else {

return flights;

}

}

// Flight Booking Service

function bookFlight(flight, passengerDetails, paymentDetails) {

if (passengerDetails is incomplete) {

return "Please complete all required fields.";

}

if (paymentDetails is invalid) {

return "Payment failed. Please check your payment details.";

}

bookingReference = confirmBooking(flight, passengerDetails, paymentDetails);

return "Booking confirmed. Reference number: " + bookingReference;

}

// Flight Schedule Service

function viewSchedule(flightNumber) {

schedule = getSchedule(flightNumber);

if (schedule is empty) {

return "Invalid flight number.";

} else {

return schedule;

}

}

// Flight Cancel Service

function cancelFlight(bookingReference) {

if (bookingReference is invalid) {

return "Invalid booking reference.";

}

cancellationAllowed = checkCancellationTime(bookingReference);

if (!cancellationAllowed) {

return "Cannot cancel the flight as it is too close to the departure time.";

}

cancellationReference = confirmCancellation(bookingReference);

return "Flight canceled. Cancellation reference number: " + cancellationReference; }

**Test cases for flight management system.**

1. Flight search service

2. Flight booking service

3. Flight Schedule service

4. Flight cancel service

**1. Flight Search Service**

**Test Case 1.1: Valid Flight Search**

* **Preconditions:** User is logged into the system.
* **Steps:**
  1. Enter valid departure and arrival locations.
  2. Select a valid date for travel.
  3. Click on the 'Search' button.
* **Expected Result:** System displays a list of available flights matching the search criteria.

**Test Case 1.2: Invalid Flight Search**

* **Preconditions:** User is logged into the system.
* **Steps:**
  1. Enter invalid or non-existent departure and arrival locations.
  2. Select a valid date for travel.
  3. Click on the 'Search' button.
* **Expected Result:** System displays an error message indicating no flights are available for the specified criteria.

**Test Case 1.3: Empty Search Fields**

* **Preconditions:** User is logged into the system.
* **Steps:**
  1. Leave departure and arrival fields empty.
  2. Click on the 'Search' button.
* **Expected Result:** System prompts the user to enter valid search criteria.

**Pseudo Code for the Test Case :-**

function validFlightSearch(departureLocation, arrivalLocation, travelDate) {

// Preconditions: User is logged into the system

if (user.isLoggedIn()) {

// Step 1: Enter valid departure and arrival locations

setField('departure', departureLocation)

setField('arrival', arrivalLocation)

// Step 2: Select a valid date for travel

setField('travelDate', travelDate)

// Step 3: Click on the 'Search' button

clickButton('search')

// Expected Result: System displays a list of available flights matching the search criteria

if (hasSearchResults()) {

return "Test Passed: Available flights displayed"

} else {

return "Test Failed: No flights displayed"

}

} else {

return "Test Failed: User not logged in"

}

}

**// Pseudo-code for Test Case 1.2: Invalid Flight Search**

function invalidFlightSearch(departureLocation, arrivalLocation, travelDate) {

// Preconditions: User is logged into the system

if (user.isLoggedIn()) {

// Step 1: Enter invalid or non-existent departure and arrival locations

setField('departure', departureLocation)

setField('arrival', arrivalLocation)

// Step 2: Select a valid date for travel

setField('travelDate', travelDate)

// Step 3: Click on the 'Search' button

clickButton('search')

// Expected Result: System displays an error message indicating no flights are available

if (isErrorMessageDisplayed("No flights available")) {

return "Test Passed: Error message displayed"

} else {

return "Test Failed: Error message not displayed"

}

} else {

return "Test Failed: User not logged in"

}

}

**// Pseudo-code for Test Case 1.3: Empty Search Fields**

function emptySearchFields() {

// Preconditions: User is logged into the system

if (user.isLoggedIn()) {

// Step 1: Leave departure and arrival fields empty

clearField('departure')

clearField('arrival')

// Step 2: Click on the 'Search' button

clickButton('search')

// Expected Result: System prompts the user to enter valid search criteria

if (isErrorMessageDisplayed("Enter valid search criteria")) {

return "Test Passed: Prompt displayed"

} else {

return "Test Failed: Prompt not displayed"

}

} else {

return "Test Failed: User not logged in"

}

}

**2. Flight booking service**

**Test Case 2.1: Valid Flight Booking**

* **Preconditions:** User has searched for flights and selected a flight.
* **Steps:**
  1. Enter passenger details (name, age, passport number, etc.).
  2. Select seat preferences.
  3. Provide payment details and confirm the booking.
* **Expected Result:** System confirms the booking and provides a booking reference number.

**Test Case 2.2: Booking with Incomplete Passenger Details**

* **Preconditions:** User has searched for flights and selected a flight.
* **Steps:**
  1. Enter incomplete passenger details.
  2. Attempt to confirm the booking.
* **Expected Result:** System prompts the user to complete all required fields.

**Test Case 2.3: Invalid Payment Details**

* **Preconditions:** User has entered valid passenger details.
* **Steps:**
  1. Enter invalid payment details.
  2. Attempt to confirm the booking.
* **Expected Result:** System displays an error message indicating payment failure.

**Pseudo Code for the Test Case :-**

**// Pseudo-code for Test Case 2.1: Valid Flight Booking**

function validFlightBooking(passengerDetails, seatPreferences, paymentDetails) {

// Preconditions: User has searched for flights and selected a flight

if (user.hasSelectedFlight()) {

// Step 1: Enter passenger details

setField('passengerDetails', passengerDetails)

// Step 2: Select seat preferences

setField('seatPreferences', seatPreferences)

// Step 3: Provide payment details and confirm the booking

setField('paymentDetails', paymentDetails)

clickButton('confirmBooking')

// Expected Result: System confirms the booking and provides a booking reference number

if (isBookingConfirmed() && getBookingReferenceNumber() != null) {

return "Test Passed: Booking confirmed and reference number provided"

} else {

return "Test Failed: Booking not confirmed"

}

} else {

return "Test Failed: No flight selected"

}

}

**// Pseudo-code for Test Case 2.2: Booking with Incomplete Passenger Details**

function incompletePassengerDetailsBooking(incompletePassengerDetails, seatPreferences, paymentDetails) {

// Preconditions: User has searched for flights and selected a flight

if (user.hasSelectedFlight()) {

// Step 1: Enter incomplete passenger details

setField('passengerDetails', incompletePassengerDetails)

// Step 2: Attempt to confirm the booking

clickButton('confirmBooking')

// Expected Result: System prompts the user to complete all required fields

if (isErrorMessageDisplayed("Complete all required fields")) {

return "Test Passed: Prompt to complete all required fields displayed"

} else {

return "Test Failed: Prompt not displayed"

}

} else {

return "Test Failed: No flight selected"

}

}

**// Pseudo-code for Test Case 2.3: Invalid Payment Details**

function invalidPaymentDetailsBooking(passengerDetails, seatPreferences, invalidPaymentDetails) {

// Preconditions: User has entered valid passenger details

if (user.hasSelectedFlight()) {

// Step 1: Enter passenger details

setField('passengerDetails', passengerDetails)

// Step 2: Select seat preferences

setField('seatPreferences', seatPreferences)

// Step 3: Enter invalid payment details and attempt to confirm the booking

setField('paymentDetails', invalidPaymentDetails)

clickButton('confirmBooking')

// Expected Result: System displays an error message indicating payment failure

if (isErrorMessageDisplayed("Payment failure")) {

return "Test Passed: Payment failure message displayed"

} else {

return "Test Failed: Payment failure message not displayed"

}

} else {

return "Test Failed: No flight selected"

}

}

3**. Flight Schedule service**

**Test Case 3.1: View Flight Schedule**

**Preconditions:** User is logged into the system.

**Steps:**

Navigate to the flight schedule section.

Enter flight number or select from a list of upcoming flights.

**Expected Result:** System displays the schedule for the selected flight, including departure and arrival times, gate information, and any delays.

**Test Case 3.2: View Non-existent Flight Schedule**

**Preconditions:** User is logged into the system.

**Steps:**

Navigate to the flight schedule section.

Enter an invalid flight number.

**Expected Result:** System displays an error message indicating the flight number is invalid or does not exist.

**Pseudo Code for the Test Case :-**

**// Pseudo-code for Test Case 3.1: View Flight Schedule**

function viewFlightSchedule(flightNumber) {

// Preconditions: User is logged into the system

if (user.isLoggedIn()) {

// Step 1: Navigate to the flight schedule section

navigateToSection('flightSchedule')

// Step 2: Enter flight number or select from a list of upcoming flights

setField('flightNumber', flightNumber)

clickButton('viewSchedule')

// Expected Result: System displays the schedule for the selected flight

if (isScheduleDisplayed(flightNumber)) {

return "Test Passed: Flight schedule displayed"

} else {

return "Test Failed: Flight schedule not displayed"

}

} else {

return "Test Failed: User not logged in"

}

}

**// Pseudo-code for Test Case 3.2: View Non-existent Flight Schedule**

function viewNonExistentFlightSchedule(invalidFlightNumber) {

// Preconditions: User is logged into the system

if (user.isLoggedIn()) {

// Step 1: Navigate to the flight schedule section

navigateToSection('flightSchedule')

// Step 2: Enter an invalid flight number

setField('flightNumber', invalidFlightNumber)

clickButton('viewSchedule')

// Expected Result: System displays an error message indicating the flight number is invalid

if (isErrorMessageDisplayed("Invalid flight number")) {

return "Test Passed: Error message displayed"

} else {

return "Test Failed: Error message not displayed"

}

} else {

return "Test Failed: User not logged in"

}

}

**4. Flight cancel service :-**

**Test Case 4.1: Valid Flight Cancellation**

**Preconditions:** User has a confirmed booking.

**Steps:**

Navigate to the 'My Bookings' section.

Select the flight to cancel.

Confirm the cancellation.

**Expected Result:** System confirms the flight cancellation and provides a cancellation reference number.

**Test Case 4.2: Cancel Flight with Invalid Booking Reference**

**Preconditions:** User is logged into the system.

**Steps:**

Navigate to the 'My Bookings' section.

Enter an invalid booking reference number.

Attempt to cancel the flight.

**Expected Result:** System displays an error message indicating the booking reference is invalid.

**Test Case 4.3: Cancel Flight Close to Departure Time**

**Preconditions:** User has a confirmed booking.

**Steps:**

Navigate to the 'My Bookings' section.

Select a flight that is close to its departure time.

Attempt to cancel the flight.

**Expected Result:** System displays a message indicating that the flight cannot be canceled as it is too close to the departure time.

**Pseudo Code for the Test Case :-**

**// Pseudo-code for Test Case 4.1: Valid Flight Cancellation**

function validFlightCancellation(bookingReference) {

// Preconditions: User has a confirmed booking

if (user.hasConfirmedBooking(bookingReference)) {

// Step 1: Navigate to the 'My Bookings' section

navigateToSection('myBookings')

// Step 2: Select the flight to cancel

selectBooking(bookingReference)

// Step 3: Confirm the cancellation

clickButton('confirmCancellation')

// Expected Result: System confirms the flight cancellation and provides a cancellation reference number

if (isCancellationConfirmed() && getCancellationReferenceNumber() != null) {

return "Test Passed: Flight cancellation confirmed and reference number provided"

} else {

return "Test Failed: Flight cancellation not confirmed"

}

} else {

return "Test Failed: No confirmed booking found"

}

}

**// Pseudo-code for Test Case 4.2: Cancel Flight with Invalid Booking Reference**

function invalidBookingReferenceCancellation(invalidBookingReference) {

// Preconditions: User is logged into the system

if (user.isLoggedIn()) {

// Step 1: Navigate to the 'My Bookings' section

navigateToSection('myBookings')

// Step 2: Enter an invalid booking reference number

setField('bookingReference', invalidBookingReference)

// Step 3: Attempt to cancel the flight

clickButton('cancelBooking')

// Expected Result: System displays an error message indicating the booking reference is invalid

if (isErrorMessageDisplayed("Invalid booking reference")) {

return "Test Passed: Error message displayed"

} else {

return "Test Failed: Error message not displayed"

}

} else {

return "Test Failed: User not logged in"

}

}

**// Pseudo-code for Test Case 4.3: Cancel Flight Close to Departure Time**

function cancelFlightCloseToDeparture(bookingReference) {

// Preconditions: User has a confirmed booking

if (user.hasConfirmedBooking(bookingReference)) {

// Step 1: Navigate to the 'My Bookings' section

navigateToSection('myBookings')

// Step 2: Select a flight that is close to its departure time

selectBooking(bookingReference)

// Step 3: Attempt to cancel the flight

clickButton('cancelBooking')

// Expected Result: System displays a message indicating that the flight cannot be canceled as it is too close to the departure time

if (isErrorMessageDisplayed("Flight cannot be canceled as it is too close to the departure time")) {

return "Test Passed: Message indicating cancellation not possible displayed"

} else {

return "Test Failed: Message not displayed"

}

} else {

return "Test Failed: No confirmed booking found"

}

}

**Flow Chart :-**

Test flow diagram based on given requirements

