April 18th, 2024

**Comfort Airlines**

**Test Plan**

Client: Professor Michael Oudshoorn

Contractors: Byte Me - Joellen A., Kyle B., Hanna Z., Alex H., Joshua T., JR V.

**About Test Plan**

The "Comfort Airlines Test Plan" document, prepared by the Byte Me team, outlines a detailed strategy for testing the functionality and robustness of their airline management software. This comprehensive plan covers a variety of tests from database queries, timetable code functionality, graphical user interface interactions, to script compatibility across different operating systems. Each section delineates specific test scenarios, expected outcomes, and methodologies for ensuring the software performs reliably under various conditions. This document serves as a critical tool for identifying potential issues and verifying that the software meets all specified requirements for managing flight operations efficiently.

Test Plans for GUI:

The test plan for the PHP pages — index.php, route.php, calc.php, and repo.php — focuses on ensuring functionality, usability, and security across various user interactions. These pages serve distinct purposes: index.php displays flight information, route.php handles flight route queries, calc.php analyzes financial data, and repo.php aggregates operational performance metrics. The testing involves checking data input, translation and unit conversions, and response to different flight route scenarios to guarantee a seamless user experience. This plan ensures that each page functions effectively within the system's graphical user interface.

**About Section for index.php:**

**Title:** Flight Information Dashboard

**Description:** This main page serves as a comprehensive dashboard displaying crucial flight-related information. Users can view detailed flight statuses, including scheduled and actual departure and arrival times, alongside distance covered. The page is designed to dynamically adjust content based on user preferences for language and units of measure, enhancing accessibility and personalization. Interactive elements like modal pop-ups provide in-depth details on user interaction, ensuring a rich and responsive user experience.

**Index.php Test Plans based on Results:**

1. **Airport to Airport Without Dates**

* **Objective:** Ensure the system can handle and return relevant flight route information without specific date input.
* **Procedure:** Select "Newark Liberty International Airport (EWR)" and "San Francisco International Airport (SFO)" without specifying a date.
* **Expected Result:** The system returns potential flight routes available, regardless of the date.

1. **Airport to Airport With Dates**

* **Objective:** Test the system's ability to filter and display only routes available on specified dates.
* **Procedure:** Choose "Miami International Airport (MIA)" to "Seattle-Tacoma International Airport (SEA)" and set the travel date to "2024-10-15".
* **Expected Result:** Only routes operating on "2024-10-15" are displayed.

1. **No Airports With Dates**

* **Objective:** Verify system behavior when dates are provided but airports are omitted.
* **Procedure:** Enter the date "2024-07-04" without selecting any airports.
* **Expected Result:** The system prompts the user to select at least one airport.

1. **Airports Within 150 Miles**

* **Objective:** Confirm the system can suggest nearby airports within a 150-mile radius and propose direct or connecting flights.
* **Procedure:** Query for airports within 150 miles of "Chicago O'Hare International Airport (ORD)".
* Expected Result: Nearby airports like "Chicago Midway International Airport (MDW)" are listed with available flight options.

1. **Airports With No Flights Between Them**

* **Objective:** Test system response when no direct or indirect flight routes exist between selected airports.
* **Procedure:** Select "Pensacola International Airport (PNS)" to "Key West International Airport (EYW)".
* **Expected Result:** The system indicates no direct routes and suggests alternatives via major hubs.

1. **Airports Without Any Flights or Dates Selected**

* **Objective:** Assess system response when no inputs are provided.
* **Procedure:** Attempt to search for routes without selecting any airports or dates.
* **Expected Result:** The system prompts the user to provide the necessary information.

1. **Date Selection**

* **Objective:** Test the flexibility and correctness of the date input field.
* **Procedure:** Use the date picker to select "2023-01-01" and "2025-12-31".
* **Expected Result:** The system accepts or rejects dates based on set criteria (e.g., no past dates).

1. **Flights On-Time**

* **Objective:** Ensure accurate reporting of on-time flights.
* **Procedure:** Check flights from "Los Angeles International Airport (LAX)" to "Denver International Airport (DEN)" on a specific day.
* **Expected Result:** The system correctly displays the flights as on-time.

1. **Flights Delayed**

* **Objective:** Verify that the system correctly identifies and reports delayed flights.
* **Procedure:** During known weather disturbances, check flights from "Dallas Fort Worth International Airport (DFW)" to "George Bush Intercontinental Airport (IAH)".
* **Expected Result:** Delays are accurately reflected in the system.

1. **Flights Canceled**

* **Objective:** Test the system's ability to report canceled flights accurately.
* **Procedure:** On a day with expected airline strikes, check routes from "Atlanta Hartsfield-Jackson International Airport (ATL)" to "New York John F. Kennedy International Airport (JFK)".
* **Expected Result:** Cancellations are correctly reported.

1. **Language Translations English to French**

* **Objective:** Ensure the system correctly translates English to French.
* **Procedure:** Change the interface language from English to French while checking flight status from "Paris Charles de Gaulle Airport (CDG)" to "John F. Kennedy International Airport (JFK)".
* Expected Result: All flight details and statuses are accurately translated into French.

**About Section for route.php:**

**Title:** Route Exploration Tool

**Description:** This tool is crucial for exploring potential flight paths between specified departure and arrival airports on selected dates. It connects to a database to fetch flight data and employs a breadth-first search algorithm to offer up to three viable routes. The results are presented in both textual and graphical formats, enhancing the user's understanding of each route through detailed map visualizations. Interactive elements allow users to visualize stopovers and geographical locations, making it an invaluable resource for travel planning and route optimization.

**Route.php for Test Plan:**

1. **Date Selection**

* **Objective:** Ensure the system accurately handles the selection of dates, including validation against incorrect formats and past date restrictions.
* **Procedure:** Interact with the date picker to select various types of dates.
* **Expected Results:**
  + Selecting a valid future date.
  + Attempting to input an invalid date format.
  + Trying to select a date in the past.
* **Sample Outputs:**
  + "Date accepted: 2024-12-15."
  + "Error: Invalid date format. Please use YYYY-MM-DD."
  + "Error: Date cannot be in the past."

1. **Generate Routes**

* **Objective:** Test the functionality of generating routes based on the selected date and any other necessary criteria.
* **Procedure:** After selecting a valid date, activate the 'Generate Routes' function.
* **Expected Result:** The system processes the input and displays available routes for the selected date.
* **Sample Output:** "Routes successfully generated for 2024-12-15. Total routes found: 3."

1. **Clear Routes**

* **Objective:** Confirm that the system can effectively clear any displayed routes with a single action.
* **Procedure:** After routes are displayed, use the 'Clear Routes' function.
* **Expected Result:** All previously displayed route information is cleared from the interface.
* **Sample Output:** "All displayed routes have been cleared."

**About Section for repo.php:**

**Title:** Operational Performance Repository

**Description:** This repository page is a crucial tool for monitoring and analyzing the operational performance of Comfort Airlines over a selected period. It aggregates and displays key metrics such as total flights operated, passenger counts, on-time departures, and on-time arrivals. Additional details on weather impacts and specific delay causes are also provided to aid in decision-making. This page is designed to offer executives and operational managers a clear view of performance trends and potential areas for improvement.

**Repo.php for Test Plan:**

1. **Test Case: Select Tail Number**

**Objective:** Ensure the system correctly handles the selection of aircraft tail numbers, validating that it can display data related to the chosen tail number.

**Procedure:**

* Access the dropdown or input field designated for selecting tail numbers.
* Select a valid tail number from the list.
* Optionally, confirm the selection by clicking a button or simply changing the selection triggers data retrieval.

**Expected Results:**

* The dropdown contains a list of valid tail numbers.
* Upon selecting a tail number, the system retrieves and displays data such as flight history, maintenance records, or performance metrics associated with that tail number.
* If a non-existent or invalid tail number is entered (if manual input is allowed), the system should display an error or no data message.

**Sample Outputs:**

* Valid Tail Number Selected: "Displaying data for Tail Number: N123AB. Total Flights: 250, Last Maintenance: 2024-01-10."
* Invalid Tail Number Entered: "No data found for Tail Number: X999ZZ. Please check the tail number and try again."
* No Tail Number Selected: "Please select a tail number to view data."

**About Section for calc.php:**

**Title:** Financial Overview Calculator

**Description:** This page offers a detailed financial analysis for specific operational periods, focusing on costs, revenues, and profit margins. It is equipped to handle extensive data retrieval from a database and perform complex calculations to present a comprehensive financial breakdown. The results are displayed in an easy-to-understand format, providing valuable insights into operational efficiency and financial health. This tool is essential for financial planning and performance assessment within the airline industry.

**Test Cases:**

1. **Market Share Calculation**

* **Objective:** To test the accuracy of the market share calculation given a multiplier of 1.02.
* **Procedure:** Input a known set of market share data and apply the 1.02 multiplier to the calculation.
* **Expected Result:** The calculated market share should reflect an increase of 2% over the base value.
* **Sample Output:** "Original Market Share: 20%, Calculated Market Share with Multiplier: 20.4%."
* Operational Costs Output
* **Objective:** To verify the correct aggregation and display of total operational costs.
* **Procedure:** Sum the detailed costs from leasing, terminal, and fuel fees to determine the total operational costs.
* **Expected Result:** The total operational costs should match the sum of all individual components.
* **Sample Output:** "Total Operational Costs: $500,000 (Leasing: $200,000, Terminal: $150,000, Fuel: $150,000)."

1. **Leasing Fee Calculation**

* **Objective:** Ensure accurate calculation of leasing fees based on given data.
* **Procedure:** Input known data for aircraft leasing costs and validate the output.
* **Expected Result:** The calculated leasing fee should match the expected value based on the input data.

**Sample Output:** "Leasing Fees: $200,000."

1. **Terminal Fee Calculation**

* **Objective:** Test the calculation of terminal fees to ensure accuracy.
* **Procedure:** Enter specific operational data related to terminal use and check the computed fee.
* **Expected Result:** The terminal fee should accurately reflect the data input, considering any applicable rates.
* **Sample Output:** "Terminal Fees: $150,000."

1. **Fuel Fee Calculation**

* **Objective:** Verify the correct calculation of fuel fees based on consumption and current fuel rates.
* **Procedure:** Provide data for fuel consumption and price per unit to calculate the fuel fee.
* **Expected Result:** The output should correctly calculate the total cost of fuel.
* **Sample Output:** "Fuel Fees: $150,000."
* **Total Costs Verification**
* **Objective:** Confirm that the total costs displayed sum up all individual components accurately.
* **Procedure:** Compare the sum of leasing, terminal, and fuel fees with the displayed total costs.
* **Expected Result:** The total displayed should be an exact sum of the components.
* **Sample Output:** "Total Costs Verified: $500,000 (Matching detailed components)."

Test Plans for Database:

* Query for flight path
  + Expected/Actual outcome:
* Check correct output for price
* Check correct output for distance
* Query for all main tables and all information: SELECT \* FROM table\_name;

`SELECT \* FROM AIRCRAFT;` **(Retrieve all aircraft details)**

`SELECT TailNumber, Speed FROM AIRCRAFT WHERE TotalSeats > 150;` **(Find aircraft with more than 150 seats)**

`INSERT INTO AIRCRAFT (TailNumber, Speed, TotalSeats) VALUES ('TN001', 500, 180);` **(Add a new aircraft)**

`UPDATE AIRCRAFT SET Speed = 520 WHERE TailNumber = 'TN001';` **(Update speed of an aircraft)**

`DELETE FROM AIRCRAFT WHERE TailNumber = 'TN001';` **(Remove an aircraft from the database)**

`SELECT \* FROM AIRPORT;` **(Retrieve all airport details)**

`SELECT AirportCode, NumberOfDepartingPassengers FROM AIRPORT WHERE GateUsed = 'A1';` **(Find airports using gate A1)**

`INSERT INTO AIRPORT (AirportCode, NumberOfDepartingPassengers, GateUsed, Latitude, Longitude) VALUES ('JFK', 1200, 'A2', 40.6413, -73.7781);` **(Add a new airport)**

`UPDATE AIRPORT SET GateUsed = 'B1' WHERE AirportCode = 'JFK';` **(Update gate for JFK airport)**

`DELETE FROM AIRPORT WHERE AirportCode = 'JFK';` **(Remove an airport from the database)**

`SELECT \* FROM FLIGHT;` **(Retrieve all flight details)**

`SELECT FlightNumber, ScheduledDepartureTime FROM FLIGHT WHERE NumberOfPassengers > 100;` **(Find flights with more than 100 passengers)**

`INSERT INTO FLIGHT (FlightNumber, DepartureAirport, DestinationAirport, NumberOfPassengers, ScheduledDepartureTime) VALUES ('FL123', 'JFK', 'LAX', 150, '2024-04-20 10:00:00');` **(Schedule a new flight)**

`UPDATE FLIGHT SET ActualDepartureTime = '2024-04-20 10:30:00' WHERE FlightNumber = 'FL123';` **(Update departure time)**

`DELETE FROM FLIGHT WHERE FlightNumber = 'FL123';` **(Cancel a flight)**

**Join Query to Find Flights from a Specific Airport with Passenger Counts:**

SELECT FLIGHT.FlightNumber, AIRPORT.AirportCode, FLIGHT.NumberOfPassengers

FROM FLIGHT

JOIN AIRPORT ON FLIGHT.DepartureAirport = AIRPORT.AirportCode

WHERE AIRPORT.AirportCode = 'JFK';

**Aggregate Query to Find Maximum, Minimum, and Average Number of Passengers Per Flight for Each Airport:**

SELECT DepartureAirport, MAX(NumberOfPassengers) AS MaxPassengers, MIN(NumberOfPassengers) AS MinPassengers, AVG(NumberOfPassengers) AS AvgPassengers

FROM FLIGHT

GROUP BY DepartureAirport;

**Subquery to Find Flights with More Passengers than the Average of All Flights:**

SELECT FlightNumber, NumberOfPassengers

FROM FLIGHT

WHERE NumberOfPassengers > (SELECT AVG(NumberOfPassengers) FROM FLIGHT);

**Join and Subquery to Find Aircraft Used More Often Than Average:**

SELECT AIRCRAFT.TailNumber, COUNT(FLIGHT.FlightNumber) AS NumberOfFlights

FROM AIRCRAFT

JOIN FLIGHT ON AIRCRAFT.TailNumber = FLIGHT.TailNumber

GROUP BY AIRCRAFT.TailNumber

HAVING COUNT(FLIGHT.FlightNumber) > (SELECT AVG(FlightCount) FROM (SELECT COUNT(FlightNumber) AS FlightCount FROM FLIGHT GROUP BY TailNumber) AS SubQuery);

**Finding Flights with Departure Delays Longer Than 1 Hour:**

SELECT FlightNumber, ScheduledDepartureTime, ActualDepartureTime, TIMESTAMPDIFF(HOUR, ScheduledDepartureTime, ActualDepartureTime) AS DelayInHours

FROM FLIGHT

WHERE TIMESTAMPDIFF(HOUR, ScheduledDepartureTime, ActualDepartureTime) > 1;

**Detailed Information on Flights, Aircraft, and Departure Airports:**

SELECT FLIGHT.FlightNumber, FLIGHT.DepartureDate, FLIGHT.ArrivalDate, AIRCRAFT.TailNumber, AIRCRAFT.TotalSeats, AIRPORT.AirportCode, AIRPORT.Latitude, AIRPORT.Longitude

FROM FLIGHT

JOIN AIRCRAFT ON FLIGHT.TailNumber = AIRCRAFT.TailNumber

JOIN AIRPORT ON FLIGHT.DepartureAirport = AIRPORT.AirportCode;

Test Plans for Timetable Code:

* Objective:
  + To verify if delays on a flight timetable cause flights to extend past 1 AM.
* Scope:
  + This test plan will focus on testing delays on flight schedules and their impact on departure or arrival times extending past 1 AM.
* Test Environment:
  + Access to the flight schedule system.
  + Access to historical flight data.
  + Access to a testing environment to simulate delays if needed.
* Test Scenarios:
  + Scenario 1: Normal Operations
    - Description: Verify that under normal circumstances, no flights depart or arrive past 1 AM.
    - Steps:
      * Access the flight schedule.
      * Verify the departure and arrival times for all flights scheduled for the day.
      * Ensure that no flights have a departure or arrival time past 1 AM.
  + Scenario 2: Simulated Minor Delay
    - Description: Simulate a minor delay and verify its impact on flight schedules.
    - Steps:
      * Introduce a minor delay (e.g., 30 minutes) to a selected flight.
      * Verify the updated departure and arrival times.
      * Ensure that even with the delay, no flights extend past 1 AM.
  + Scenario 3: Simulated Major Delay
    - Description: Simulate a major delay and verify its impact on flight schedules.
    - Steps:
      * Introduce a major delay (e.g., 2 hours) to a selected flight.
      * Verify the updated departure and arrival times.
      * Ensure that with the delay, no flights extend past 1 AM.
  + Scenario 4: Cascading Delays
    - Description: Introduce multiple delays and verify their cumulative impact on flight schedules.
    - Steps:
      * Introduce delays to multiple flights, either simultaneously or sequentially.
      * Verify the updated departure and arrival times for all affected flights.
      * Ensure that even with multiple delays, no flights extend past 1 AM.
* Test Data:
  + Sample flight schedules.
  + Simulated delay times.
* Test Execution:
  + Perform each scenario as outlined above.
  + Record any deviations from expected results.
* Acceptance Criteria:
  + All flights must maintain departure and arrival times before 1 AM, regardless of delays.
  + No flight should extend past 1 AM due to delays.
* Reporting:
  + Document any deviations from expected results.
  + Provide a summary of test outcomes, including any issues encountered and their resolutions.
* Risks and Mitigation:
  + Risk: Inaccurate simulation of delays.
    - Mitigation: Use realistic delay scenarios based on historical data and industry standards.
  + Risk: System limitations causing unexpected behavior.
    - Mitigation: Thoroughly test the system in a controlled environment before conducting live tests.
* Dependencies:
  + Access to the flight schedule system.
  + Availability of historical flight data for analysis and simulation.
* Assumptions:
  + Delays are represented accurately in the flight schedule system.
  + The system handles delay calculations and updates correctly.
* ``

Test Plans for Scripts:

* Test to see if they work in a new environment
* Test to see what Operating Systems the script works on
* Test to make sure that functionality remains the same between using the script and not

Test Plans for Timetable Code:

* Objective:
  + Ensure that the scripts correctly work and the result is identical to running the code manually
* Scope:
  + Ensure that there is no difference
* Test Environment:
  + Access to every resource we provide while using script
* Test Scenarios:
  + Scenario 1: Normal Operations
    - Description: Run the code without the script.
    - Steps: Start the docker and ensure that it is running properly
  + Scenario 2: Use script
    - Description: Run the code with the script
    - Steps: use “./run.sh” to hopefully start the docker and configure everything.
* Test Data:
  + Docker-compose.yml
  + run.sh
  + stop.sh
* Test Execution:
  + Perform each scenario as outlined above.
  + Record any deviations from expected results.
* Acceptance Criteria:
  + Using the script should function the exact same as not using the script
* Reporting:
  + On Mac the script works flawlessly, on Windows there seems to be an error with permissions for the script
* Risks and Mitigation:
  + Risk: Errors on Windows OS
    - Mitigation: Provided in the user manual is the instructions on how to run the code without using the script, while we fix the issue we will provide this for use by those using Windows.
* Dependencies:
  + Access to the docker
* Assumptions:
  + With the webpage loading the code should be executed the same.