Software Engineering Project

AirlineSYS

Submitted By: Divine Okhaus (T00233163)

Computing with Software Development

Date Submitted: 21/04/2024

**Table of Contents**

[1. Introduction/overview 4](#_Toc151732740)

[2. Functional Components 5](#_Toc151732741)

[3. User Requirements 6](#_Toc151732742)

[3.1. AirlineSYS will Manage Routes 6](#_Toc151732743)

[3.2. AirlineSYS will Manage Flights 6](#_Toc151732744)

[3.3. AirlineSYS will Process Bookings 6](#_Toc151732745)

[3.4. AirlineSYS will Perform Administrative Reporting 6](#_Toc151732746)

[4. System Requirements 7](#_Toc151732747)

[4.1. System Level Use Case Diagram 7](#_Toc151732748)

[4.2. Manage Routes 8](#_Toc151732749)

[4.2.1. Add Airport 8](#_Toc151732750)

[4.2.2. Add Operator 11](#_Toc151732756)

[4.2.3. Create Route 14](#_Toc151732757)

[4.2.4. End Route 18](#_Toc151732758)

[4.3. Manage Flights 20](#_Toc151732759)

[4.3.1. Schedule Flight 20](#_Toc151732760)

[4.3.2. Update Flight 23](#_Toc151732766)

[4.3.3. Cancel Flight 25](#_Toc151732767)

[4.4. Process Bookings 27](#_Toc151732768)

[4.4.1. Create Booking 27](#_Toc151732769)

[4.4.2. Update Booking 31](#_Toc151732775)

[4.4.3. Cancel Booking 33](#_Toc151732776)

[4.5. Process Admin 35](#_Toc151732780)

[4.5.1. Yearly Flight Analysis 35](#_Toc151732781)

[4.5.2. Yearly Revenue Analysis 37](#_Toc151732787)

[4.5.3. Yearly Route Analysis 39](#_Toc151732788)

[4.6. Activity Diagram 41](#_Toc151732789)

[5. System Model 43](#_Toc151732790)

[5.1. Level-0 DFD 44](#_Toc151732791)

[5.2. Level-1 DFD 45](#_Toc151732792)

[5.3. Level-2 DFD (Process P1: Manage Airports) 46](#_Toc151732793)

[5.4. Level-2 DFD (Process P2: Manage Flights) 47](#_Toc151732794)

[5.5. Level-2 DFD (Process P3: Process Bookings) 48](#_Toc151732795)

[5.6. Level-2 DFD (Process P4: Process Admin) 49](#_Toc151732796)

[6. Data Model (Class Diagram) 50](#_Toc151732797)

[6.1. Class Diagram 50](#_Toc151732798)

[6.2. Relational Schema 51](#_Toc151732799)

[6.3. Database Schema 52](#_Toc151732800)

[7. Conclusion 55](#_Toc151732801)

# Introduction/overview

The system I will be working on is called AirlineSYS. Its sole purpose will be to provide an airline with a system for adding airports, operators, and creating routes to schedule flights. These scheduled flights will then be available for customers of the airline to book. When creating an airport, operator, and route, various pieces of information will be needed. To schedule a flight, two airports will be selected to create a route for departure and arrival (the system will only facilitate one-way journeys). The operator chosen to schedule the flight will represent the airline operating the plane for the journey. Routes may be ended due to low demand or financial issues. Once a flight is scheduled, if necessary, it can be updated or cancelled. For a booking to occur, various personal information will be required, along with the desired destination.

# Functional Components

# User Requirements

## AirlineSYS will Manage Routes

* + 1. AirlineSYS will Add Airport
    2. AirlineSYS will Add Operator
    3. AirlineSYS will Create Route
    4. AirlineSYS will End Route

## AirlineSYS will Manage Flights

* + 1. AirlineSYS will Schedule Flight
    2. AirlineSYS will Update Flight
    3. AirlineSYS will Cancel Flight

## AirlineSYS will Process Bookings

* + 1. AirlineSYS will Book Flight
    2. AirlineSYS will Update Booking
    3. AirlineSYS will Cancel Booking

## AirlineSYS will Perform Administrative Reporting

* + 1. AirlineSYS will display Yearly Revenue Analysis
    2. AirlineSYS will display Yearly Route Analysis

# System Requirements

This section of the document outlines the system requirements. The primary module at the top level includes Airport, Flights, Booking and Admin.

## System Level Use Case Diagram

The following system level use case diagram illustrates the high-level system requirements.

Airport Manager

Customer

## Manage Routes

The functions that can be performed within this component includes Add Airport, Add Operator, Create Route, and End Route int the Airport File, Routes File, and Operators File.

### Add Airport

This function adds the details of an airport to the Airports file.

|  |  |  |
| --- | --- | --- |
| **Use Case:** Add Airport | | |
|  | | |
| **Use Case Name** | Add Airport | |
| **Use Case Id** | 01 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | N/A | |
| **Description** | This function adds airports to the system | |
| **Preconditions** | Airport must not exist in the system. | |
| **Trigger** | Click the Add Airport button | |
| **Expected Scenario** | **Airport Manager** | **AirlineSYS** |
|  | **Step 1:** Manager invokes Add Airport function.  **Step 3:** Enter the required data for Add Airport:   * AirportCode txt (3) * Name txt (50) * Street txt (50) * City txt (50) * Country txt (50) * Eircode txt (7) * Phone int (10) * Email txt (60)   **Step 4:** Confirms. | **Step 2:** The systemDisplays UI.  **Step 5:** The system validates the data entered:   * All fields must be entered. * AirportCode must be UNIQUE. * AirportCode must be characters with length of 3. * Name must be characters with the maximum length of 50. * Street must be characters with the maximum length of 50. * City must be characters with the maximum length of 50. * Country must be characters with the maximum length of 50. * Eircode must be a valid code with the length of 7 characters. * Phone number must begin with 08 and length of 10 characters. * Email must be a valid email and the maximum length of 60.   **Step 6:** The valid Add Airport details will be saved in the Airport file.   * AirportCode * Name * Street * City * Country * Eircode * Phone * Email   **Step 7:** Display an Appropriate Confirmation Message.  **Step 8:** Reset UI |
| **Alternate Scenarios** | **Airport Manager** | **AirlineSYS** |
| **Invalid Add Airport detail entered** |  | **Step 6:** Invalid data has been detected.  **Step 7:** An appropriate error message will be displayed.  **Step 8:** Return to Step 3. |
| **Conclusions** | An airport has been added to the Airports File. | |
| **Post conditions** | The airport can now be used as a possible route to take or to end. | |
| **Business Rules** | N/A | |
| **Implementation Constraints** | N/A | |



### **Add Operator**

This function adds the details of an operator to the Operators file.

|  |  |  |
| --- | --- | --- |
| **Use Case:** Add Airport | | |
|  | | |
| **Use Case Name** | Add Operator | |
| **Use Case Id** | 02 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | N/A | |
| **Description** | This function adds operators to the system | |
| **Preconditions** | OperatorCode must not exist in the system. | |
| **Trigger** | Click the Add Airport button | |
| **Expected Scenario** | **Airport Manager** | **AirlineSYS** |
|  | **Step 1:** Manager invokes the Add Airport function.  **Step 3:** Enter the required data for Add Operator:   * OperatorCode txt (2) * Name txt (50) * City txt (50) * Country txt (50)   **Step 4:** Click’s Confirm | **Step 2:** The systemDisplay UI.  **Step 5:** System validates the data:   * All field must be entered. * OperatorCode must be characters with length of 2 and UNIQUE. * Name must be characters with a maximum length of 50 * City must be characters with a maximum length of 50. * Country must be characters with a maximum length of 50.   **Step 6:** System saves data in the data in the Operators file:   * OperatorCode * Name * City * Country   **Step 7:** Display the Appropriate Confirmation Message.  **Step 8:** Reset UI |
| **Alternate Scenarios** | **Airport Manager** | **AirlineSYS** |
| **Invalid Add Operator detail entered** |  | **Step 6:** Invalid data has been detected.  **Step 7:** An appropriate error message is displayed.  **Step 8:** Return to Step 3 |
| **Conclusions** | An operator has been added, validated, and saved in the Airports File. | |
| **Post conditions** | The added operator is now available for use. | |
| **Business Rules** | N/A | |
| **Implementation Constraints** | N/A | |

### **Create Route**

This function establishes a route to be used in the system

|  |  |  |
| --- | --- | --- |
| **Use Case:** Create Route | | |
|  | | |
| **Use Case Name** | Create Route | |
| **Use Case Id** | 03 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | N/A | |
| **Description** | This function creates a route in the system. | |
| **Preconditions** | Route must not exist in the system. | |
| **Trigger** | Click the Create Route button | |
| **Expected Scenario** | **Airport Manager** | **AirlineSYS** |
|  | **Step 1:** Manager invokes the Create Route function.  **Step 3:** Enter Required Data:   * DeptAirport txt(3) * ArrAiport txt(3) * Duration number(4) * TicketPrice decimal (6,2)   **Step 3:** Enter Required Data: | **Step 2:** SystemDisplay UI.  **Step 4:** System validates data:   * All fields must be filled. * Departure and Arrival airport must have the length of 3 * The departure airport must not match the entered arrival airport. * Duration > 0 with a length of 4. * TicketPrice must be > 0.   **Step 5:** System sets Route Status to ‘A’ for Active.  **Step 6:** Generate RouteID  **Step 7:** System saves route details in  the Routes file:     * RouteID * DeptAirport * ArrAirport * TicketPrice * Duration * Status   **Step 8:** The system displays an appropriate confirmation message.  **Step 9:** Reset UI. |
| **Alternate Scenarios** | **Airport Manager** | **AirlineSYS** |
| **Departure Airport and Arrival Airport match** |  | **Step 5:** Invalid data has been detected.  **Step 6:** An appropriate error message will be displayed.  **Step 7:** Return to Step 5 |
| **Route Already in the system** |  | **Step 5:** Invalid data has been detected.  **Step 6:** An appropriate error message will be displayed.  **Step 7:** Return to Step 5 |
| **Conclusions** | A route has been created in the Airports File. | |
| **Post conditions** | Route can now be used to schedule flights. | |
| **Business Rules** | There can only be one route with the same departure and arrival airport | |
| **Implementation Constraints** | N/A | |

### **End Route**

This function ends a route saved in the system and sets it to Ended

|  |  |  |
| --- | --- | --- |
| **Use Case:** End Route | | |
|  | | |
| **Use Case Name** | End Route | |
| **Use Case Id** | 04 | |
| **Priority** | Low | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | N/A | |
| **Description** | This function Ends the route in the system. | |
| **Preconditions** | N/A | |
| **Trigger** | Click the Create Route button | |
| **Expected Scenario** | **Airport Manager** | **AirlineSYS** |
|  | **Step 1:** Manager invokes the End Route function.  **Step 4: The** Manager selects the route they wish to end.    **Step 7:** The manager clicks on confirm.  **Step 9:** Selects ‘YES’. | **Step 2:** The system retrieves routes from the Routes File and loads all airports on UI.  **Step 3:** The system displays the UI.  **Step 5:** The system retrieves all the data of the selected route.  **Step 6:** The system loads the retrieved route data on the UI.  **Step 8:** The system prompts message relating to end route confirmation  **Step 10:** The system sets the Status of the select route to ‘E’ for Ended.  **Step 11:** The system prompts ‘The route has been ended’.  **Step 12:** Resets UI. |
| **Alternate Scenarios** | **Airport Manager** | **AirlineSYS** |
| **Clicks on NO** |  | **Step 10:** Returns |
| **Conclusions** | Route was Ended | |
| **Post conditions** | The route selected status can be changed to Ended and will not be used to schedule flights till further notice. | |
| **Business Rules** | If A flight is already scheduled for a route. Ending the route will be aborted and a message will be given. | |
| **Implementation Constraints** | N/A | |

## Manage Flights

The functions that can be performed within this component includes Schedule Flight, Update Flight, Cancel Flight from the Flights File.

### Schedule Flight

This function schedules flights and saves it in the system

|  |  |  |
| --- | --- | --- |
| **Use Case:** Schedule Flight | | |
| <<includes>>  << excludes >> >>>> | | |
| **Use Case Name** | Schedule Flight | |
| **Use Case Id** | 05 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | N/A | |
| **Description** | This function schedules flights, so it can be available for books. | |
| **Preconditions** | Route must exist in the system to schedule a flight | |
| **Trigger** | Click the Schedule Flight Button | |
| **Expected Scenario** | **Airport Manager** | **AirlineSYS** |
|  | **Step 1:** Manager Invokes Schedule flight function.          **Step 8:** Manager selects the desired departure airport and arrival airport.    **Step 11:** Manager pick’s the date for the flight to departure.   * DepartureDate Date   **Step 12:** Manager select’s flight time.   * FlightTime txt(8)   **Step 14:** Manager selects the desired operator   * OperatorCode txt(2)   **Step 16:** Manager enters number of seats.   * NumSeats number (3) | **Step 2:** System displays UI.  **Step 3:** The system retrieves a summary of departure airports, arrival airport from the Routes file and loads on UI.  **Step 4:** System retrieves operator codes from the Operator file and loads on UI.  **Step 5:** The system loads date on UI.  **Step 6:** The system retrieves possible departure times and loads on UI.  **Step 7**: Display UI  **Step 9:** The system retrieves the routeID and duration of the selected departure and arrival airport.  **Step 10:** Displays on UI.  **Step 13:** Calculate estimated arrival time by using the selected flight time and duration.  **Step 15:** The system increments the 4-digit number and combines it with the selected operator code. E.g., FR0001  **Step 17:** System validates the data:   * No fields must be empty * NumSeats > 0 and <=100 * FlightDate > than today   **Step 18:** System sets status of flight to ‘A’ for Available.  **Step 19:** System sets NumSeatsAvail equal to NumSeats.  **Step 2**0**:** System saves data in the Flights file:   * FlightNumber * OperatorCode * RouteID * FlightDate * FlightTime * EstArrTime * NumSeats * NumSeatsAvail * Status     **Step 21:** System displays an Appropriate message.    **Step 22:** Reset UI . |
| **Alternate Scenarios** |  |  |
| **Route doesn’t exist** |  | **Step 10:** Invalid data has been detected.  **Step 11:** Display an appropriate message.  **Step 12:** Return to step 8. |
| **Conclusions** | A flight has been scheduled, validated, and saved in the Flights File. | |
| **Post Conditions** | The flight can now be updated or cancelled. | |
| **Business Rules** | Flight cannot be scheduled for the same today it was created | |
| **Implementation Constraints** | N/A | |



### **Update Flight**

This function can Update the schedule flight function

|  |  |  |
| --- | --- | --- |
| **Use Case:** Update Flight | | |
| <<includes>>  << excludes>> | | |
| **Use Case Name** | Update Flight | |
| **Use Case Id** | 06 | |
| **Priority** | Medium | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | N/A | |
| **Description** | This function updates flights that already exist in the system. | |
| **Preconditions** | FlightNumber must exist in the system | |
| **Trigger** | Click the Schedule Flight Button | |
| **Expected Scenario** | **Airport Manager** | **AirlineSYS** |
|  | **Step 1:** The Manager Invokes the Update Flight function.  **Step 4:** The manager selects the FlightNumber for the flight they wish to update.  **Step 6:** The manager picks the date they wish to update.   * DeptTime | **Step 2:** The system retrieves a summary of flight number from the flights file and loads on UI.  **Step 3:** Displays UI.  **Step 5:** The system retrieves all details of the selected FlightNumber from the Flights file Routes file and Operators file. And displays on UI.   * FlightNumber * RouteID * DeptAirport * OperatorCode * ArrAirport * FlightDate * FlightTime * Duration * EstArrTime * NumSeats   **Step 7:** The system will validate the data:   * DeptTime must be greater than the current time.   **Step 8:** The system displays a confirmation message.  **Step 9:** Reset UI |
|  |  |  |
| **No flight has been selected** |  | **Step 6:** No data has been detected.  **Step 7:** Display an appropriate message.  **Step 8:** Return to step 4. |
| **Conclusions** | A flight has been updated, validated, and saved in the Flights File. | |
| **Post Conditions** | Flight date has been changed | |
| **Business Rules** | N/A | |
| **Implementation Constraints** | N/A | |

### Cancel Flight

This function cancels flight’s that has been scheduled.

|  |  |  |
| --- | --- | --- |
| **Use Case:** Cancel Flight | | |
| <<includes>>  << excludes >> | | |
| **Use Case Name** | Cancel Flight | |
| **Use Case Id** | 07 | |
| **Priority** | Low | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | N/A | |
| **Description** | This function cancel’s flights that already exists in the system. | |
| **Preconditions** | FlightNumber must exist in the system | |
| **Trigger** | Click the Schedule Flight Button | |
| **Expected Scenario** | **Airport Manager** | **AirlineSYS** |
|  | **Step 1:** Manager Invokes Cancel Flight function.  **Step 4:** Manager enters the FlightNumber of the flight they wish to cancel.  **Step 6:** Manager confirms. | **Step 2:** The system retrieves a summary of flight number from the flights file and loads on UI.  **Step 3:** Displays UI.  **Step 5:** The system retrieves all details of the selected FlightNumber from the Flights file Routes file and Operators file. And displays on UI.   * FlightNumber * RouteID * DeptAirport * OperatorCode * ArrAirport * FlightDate * FlightTime * Duration * EstArrTime * NumSeats   **Step 7:** The system changes the status of the flight to ‘C’ for Cancelled.  **Step 8:** The system displays a confirmation message.  **Step 9:** Reset UI. |
| **Alternate Scenarios** |  |  |
| **F**light number not found |  | **Step 6:** The entered FlightNumber was not found/doesn’t exist.  **Step 7:** Display an appropriate message.  **Step 8:** Returns to Step 4. |
| **Conclusions** | The selected Flight has been cancelled and has it status set to ‘C’. | |
| **Post Conditions** | The flight is now cancelled and cannot be booked. | |
| **Business Rules** | If seats have been booked on that Flight, it cannot be cancelled. | |
| **Implementation Constraints** | N/A | |

## Process Bookings

This function books a flight to whatever destination the customer wishes to travel to and its must be within the parameters of the Airline. The Booking can be update and cancelled. Within 48 hours before the flight takes off, cancellation fees will be applied.

### Create Booking

This function books flight and saves it in the system

|  |  |  |
| --- | --- | --- |
| **Use Case:** Create Booking | | |
| <<includes>>  Customer  << excludes >> | | |
| **Use Case Name** | Create Booking | |
| **Use Case Id** | 08 | |
| **Priority** | High | |
| **Source** | N/A | |
| **Primary Business Actor** |  | |
| **Other Participating Actors** | Customer | |
| **Description** | This function allows booking to be processed | |
| **Preconditions** | Flight must have already been scheduled to be booked | |
| **Trigger** | Click the Create Booking Button | |
| **Expected Scenario** | **Customer** | **AirlineSYS** |
|  | **Step 1:** Invokes Create Booking function.  **Step 5:** Selects departure and arrival airport.  **Step 6:** Clicks of search.  **Step 12:** Selects their desired flight containing:   * FlightNumber * DeptAirport * ArrAirport * FlightDate * FlightTime * EstArrTime   **Step 13:** Click on Confirm.  **Step 15:** Click ‘Yes’.  **Step 19:** Click on Confirm.  **Step 25:** Enters required data are entered:   * Forename txt (50) * Surname txt (50) * DateOfBirth Date * Email txt (60) * Phone number (10) * Eircode txt(7)   **Step 26:** Enters Payment for Flight.  **Step 27:** Clicks on Confirm. | **Step 2:** Display UI  **Step 3:** The system retrieves a summary of airport codes from the Airports File and loads UI.  **Step 4:** Displays UI  **Step 7:** The system retrieves all active Routes file using the selected departure and arrival airport.  **Step 8**: Loads on UI.  **Step 9:** The system retrieves scheduled flights and their details from the flight file with the corresponding route.  **Step 10:** Loads on UI  **Step 11:** Displays UI  **Step 14:** The System promptsAre you sure you want to book the selected flight. ‘Yes’ or ‘No’.  **Step 16:** The System prompts  ‘Will you be bringing baggage’. ‘Yes’ or ‘No’.  **Step 17:** The System increments amount of baggage by one.  **Step 18:** The System will display the selected flight data and amount of baggage.  **Step 20:** The System retrieves personal details UI.  **Step 21:** The system retrieves the flight data and loads on UI.  **Step 22:** The System generates potential Seat Number, Booking ID and Passenger ID and loads on UI.  **Step 23:** The System retrieves the payment for the chosen route and loads on UI.  **Step 24:** Display on UI.  **Step 28**: The system validates the data entered:   * All fields must be entered. * Forename must not be numeric and maximum length of 50. * Surname must not be numeric and maximum length of 50. * Date of Birth must date and must calculate to 18 or over. * Email must be valid. * Phone number must be numeric and maximum length of 10. * Eircode must be valid, with a length of 7 * Paymented amount must be numeric and be the exact amount for the flight.   **Step 29**: The System assigns:   * BookingID * PassengerID * SeatNumber   **Step 30:** The System saves Booking data in the Bookings file:   * BookingID * PassengerID * RouteID * FlightNumber * FlightTime * FlightDate * SeatNum * NumBaggage * AmountPaid * Status   **Step 31:** The system saves data in the Passanger File.   * PassangerID * ForeName * Surname * DateOfBirth * Email * Phone * Eircode   **Step 32:** The system confirms booking.  **Step 33:** The System decreases number of seats by 1 (default).  **Step 34:** Reset UI |
| **Alternate Scenarios** | **Customer** | **AirlineSYS** |
| **Invalid Payment** |  | **Step 29:** Invalid data detected  **Step 30:** Display an appropriate message.  **Step 31:** Return to step 25 |
| **Invalid Date of Birth** |  | **Step 26:** Invalid data detected  **Step 27:** Display an appropriate message.  **Step 28:** Return to step 25 |
| **Invalid Phone Number** |  | **Step 26:** Invalid data detected  **Step 27:** Display an appropriate message.  **Step 28:** Return to step 25 |
| **Invalid Eircode** |  | **Step 26:** Invalid data detected  **Step 27:** Display an appropriate message.  **Step 28:** Return to step 25 |
| **Invalid Email** |  | **Step 26:** Invalid data detected  **Step 27:** Display an appropriate message.  **Step 28:** Return to step 25 |
| **Conclusions** | Payment has been validated, flight has been booked, and can now be updated and cancelled. | |
| **Post conditions** | Flight has been booked and confirmation is sent. | |
| **Business Rules** | N/A | |
| **Implementation Constraints** | N/A | |



### Update Booking

This function can Update the schedule flight function

|  |  |  |
| --- | --- | --- |
| **Use Case:** Update Booking | | |
| <<includes>>  Customer  << excludes >> | | |
| **Use Case Name** | Update Booking | |
| **Use Case Id** | 09 | |
| **Priority** | High | |
| **Source** | N/A | |
| **Primary Business Actor** |  | |
| **Other Participating Actors** | Customer | |
| **Description** | This function allows a booked flight to be updated | |
| **Preconditions** | BookingID must exist to update a booking | |
| **Trigger** | Click the Cancel Booking Button | |
| **Expected Scenario** | **Customer** | **AirlineSYS** |
|  | **Step 1:** Invokes Update Booking function.  **Step 3:** Enters the BookingID  **Step 6:** Selects and enters the required data to be Updated:   * ForeName * Surname * DateOfBirth * Email * Phone * Eircode | **Step 2:** Display UI  **Step 4:** The system retrieves a summary of booking details from the Bookings file and loads on UI.  **Step 5:** Display UI  **Step 7:** The system retrieves booking details for the Bookings File and loads on UI.  **Step 8:** The system retrieves booking amount of baggage for the Bookings File and loads on UI.  **Step 9:** Displays UI  **Step 10:** System validates data:   * Forename must not be numeric and maximum length of 50. * Surname must not be numeric and maximum length of 50. * Date of Birth must date and must calculate to 18 or over. * Email must be valid. * Phone number must be numeric and maximum length of 10. * Eircode must be valid, with a length of 7   **Step 10:** The system updates the details of the Bookings file.  **Step 11:** The system updates the detail of the Bookings file.  **Step 12:** System displays an appropriate message.  **Step 13:** Reset UI |
| **Alternate Scenarios** | **Customer** | **AirlineSYS** |
| **Invalid phone number entered** |  | **Step 11:** Invalid data detected  **Step 12:** Display Appropriate message.  **Step 13:** Return to step 6 |
| **Invalid Eircode entered** |  | **Step 11:** Invalid data detected  **Step 12:** Display Appropriate message.  **Step 13:** Return to step 6 |
| **Invalid Email entered** |  | **Step 11:** Invalid data detected  **Step 12:** Display Appropriate message.  **Step 13:** Return to step 6 |
| **Conclusions** | Booking details has been validated, updated, and saved. | |
| **Post conditions** | Booking has been updated. | |
| **Business Rules** | N/A | |
| **Implementation Constraints** | N/A | |

### Cancel Booking

This function cancels a booked flight.

|  |  |  |
| --- | --- | --- |
| **Use Case:** Cancel Booking | | |
| <<includes>>  << excludes >> | | |
| **Use Case Name** | Cancel Booking | |
| **Use Case Id** | 10 | |
| **Priority** | Low | |
| **Source** | N/A | |
| **Primary Business Actor** | N/A | |
| **Other Participating Actors** | Customer | |
| **Description** | This function cancel’s booked flights that already exists in the system. | |
| **Preconditions** | BookingID must exist in the system | |
| **Trigger** | Click the Schedule Flight Button | |
| **Expected Scenario** | **Customer** | **AirlineSYS** |
|  | **Step 1:** Invokes Cancel Flight function.  **Step 3:** Enters BookingID  **Step 7:** Confirms. | **Step 2:** System displays UI.  **Step 4:** The system retrieves all details for the entered BookingID from the Bookings file and loads on UI.   * BookingID * PassengerID * RouteID * FlightNumber * FlightTime * FlightDate * SeatNum * NumBaggage * AmountPaid   **Step 5:** The system retrieves all details for the entered BookingID for the Passengers file and loads on UI.   * ForeName * Surname * DateOfBirth * Email * Phone * Eircode   **Step 6:** Display UI  **Step 8:** The system changes the status of the Booking to ‘Cancelled’.  **Step 9:** The system displays a confirmation message.  **Step 10:** Reset UI |
| **Alternate Scenarios** | **Customer** | **AirlineSYS** |
| **BookingID not found** |  | **Step 5:** Invalid data entered  **Step 6:** Display an appropriate message.  **Step 7:** Returns to Step 3. |
| **Conclusions** | The booked flight has been cancelled and has it status set to ‘C’. | |
| **Post Conditions** | The booking is now cancelled and cannot be used board a flight. | |
| **Business Rules** | N/A | |
| **Implementation Constraints** | N/A | |



## Process Admin

The functions that can be performed within this component includes Yearly Revenue Analysis and Yearly Route Analysis.

### Yearly Revenue Analysis

This function does the Yearly Revenue Analysis

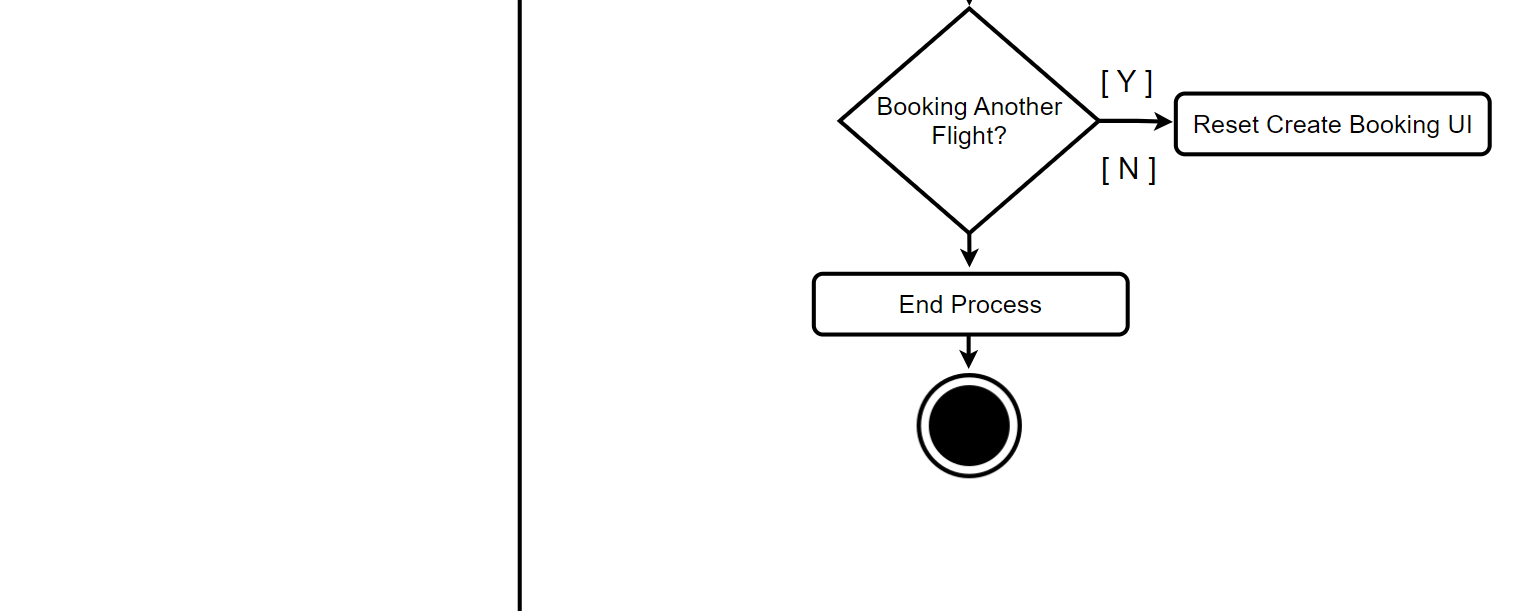
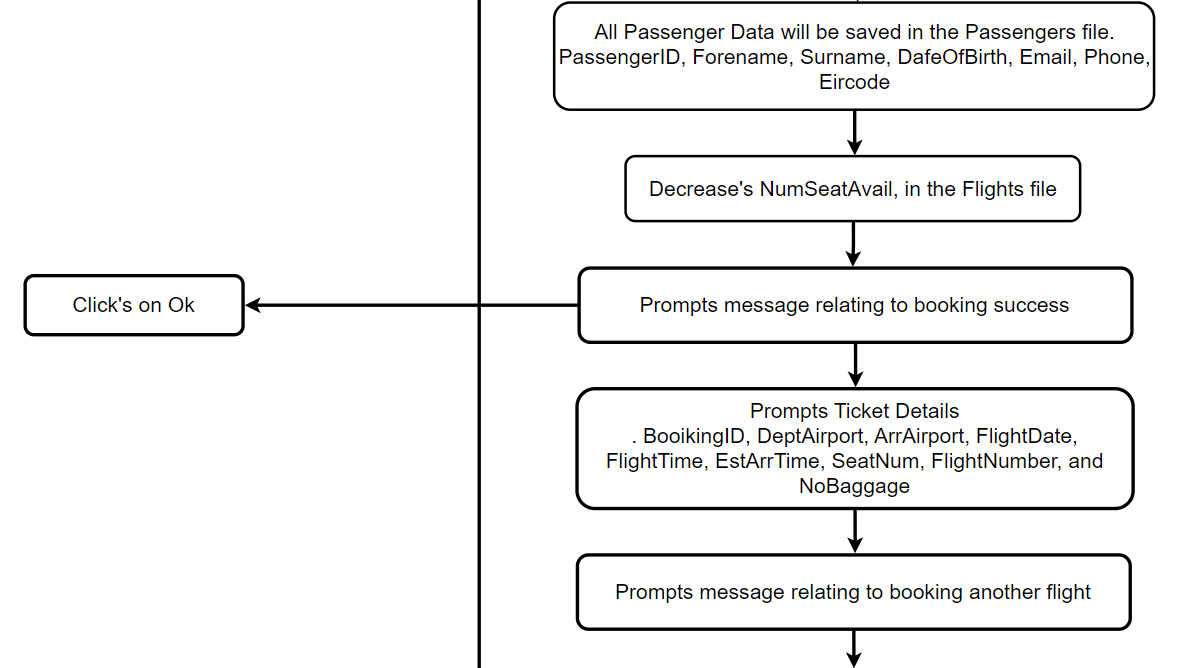
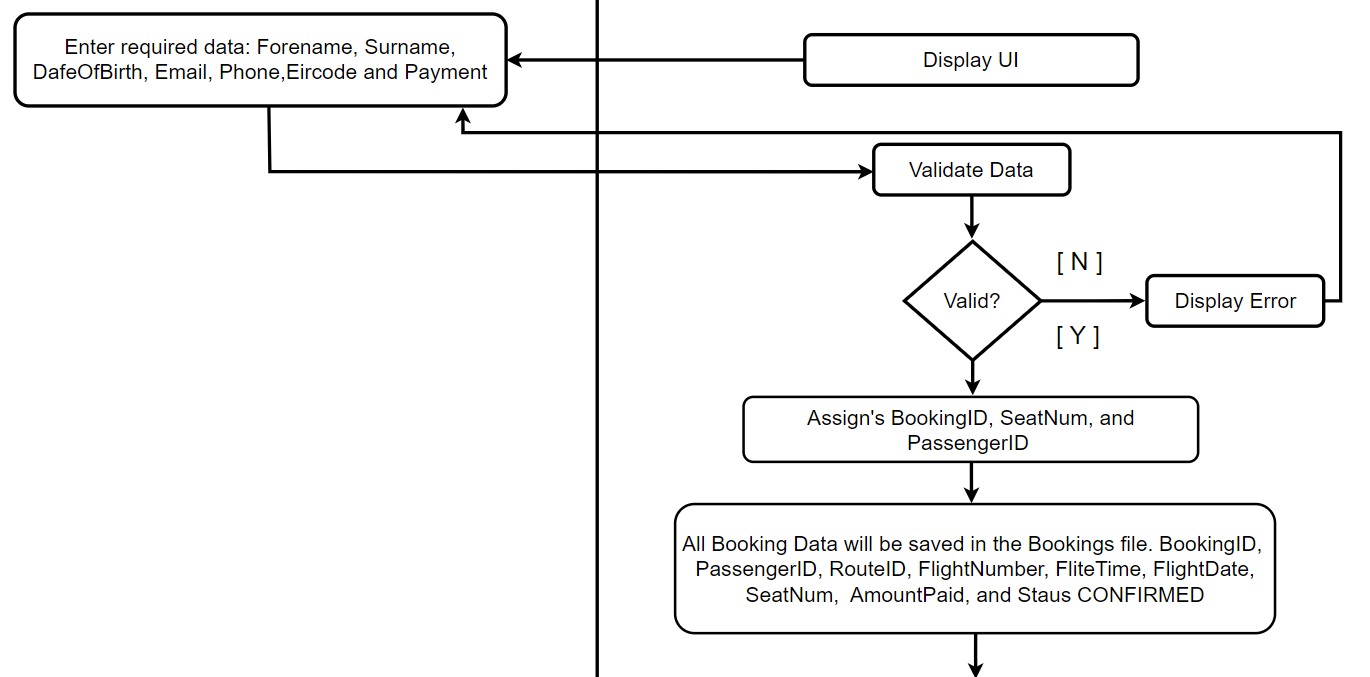
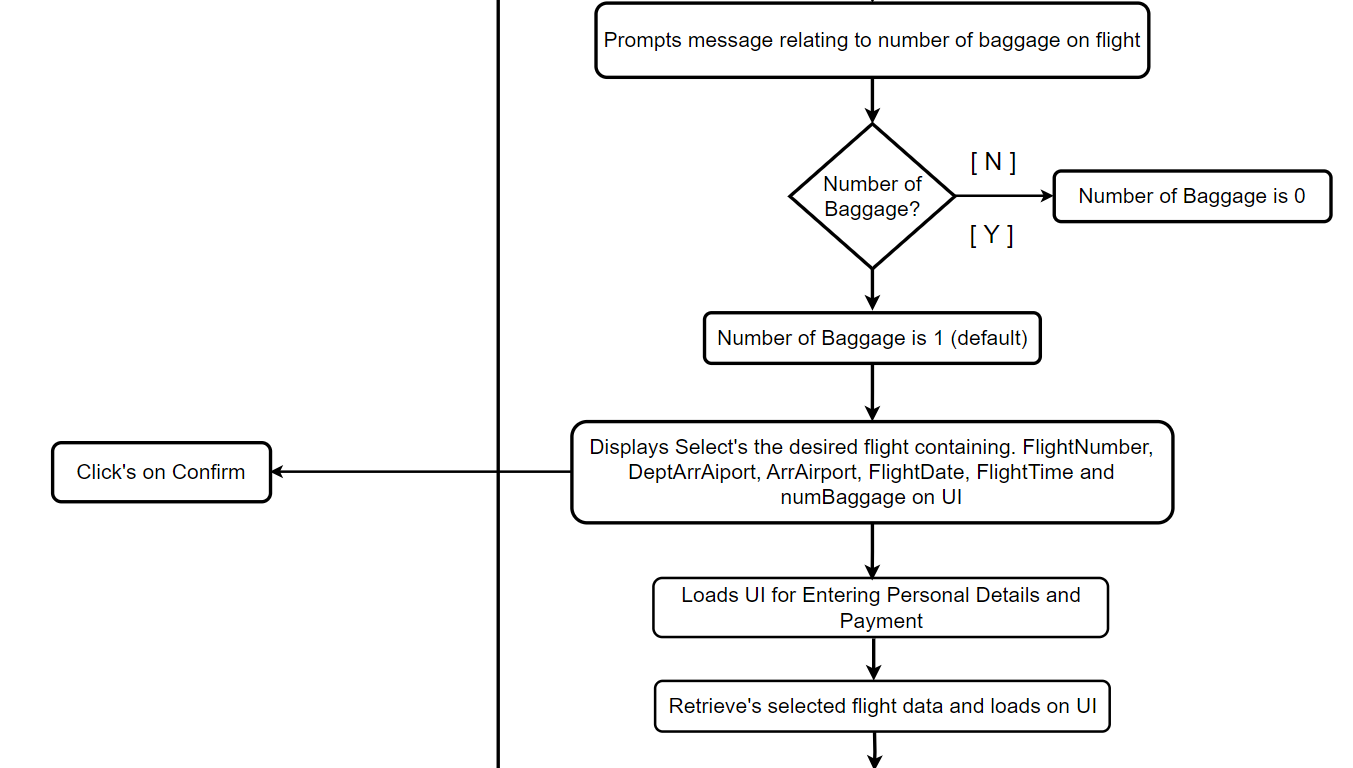
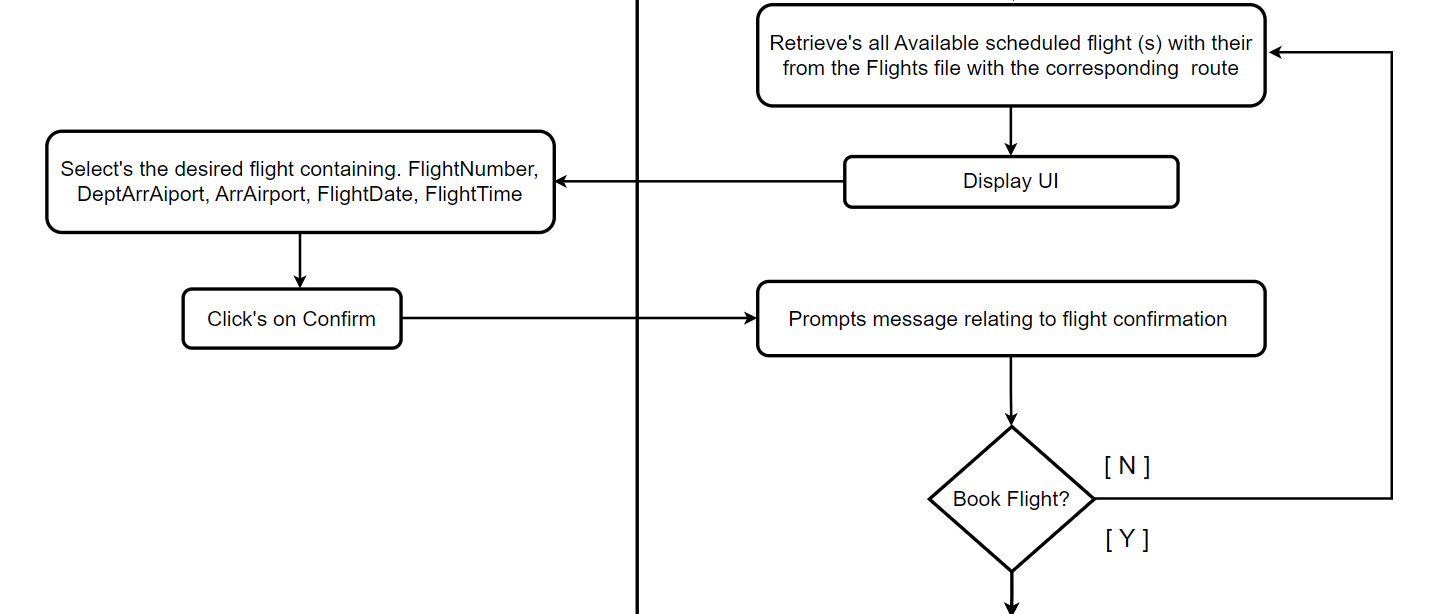
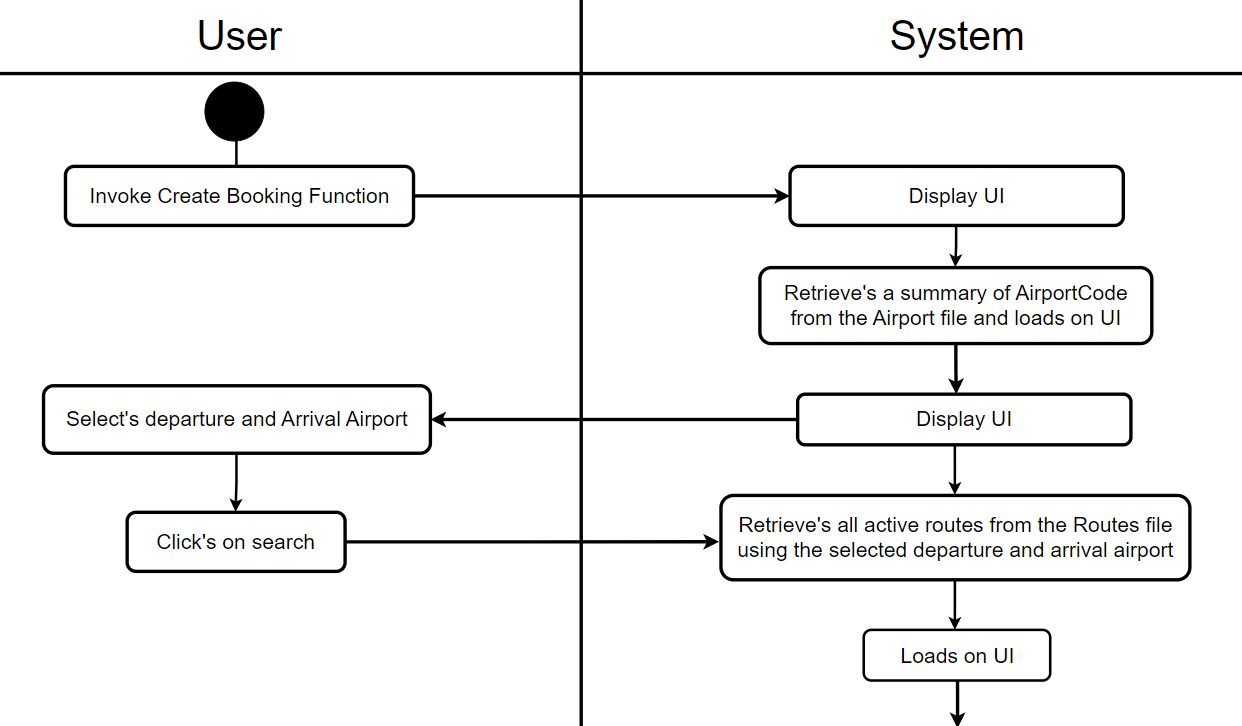
|  |  |  |
| --- | --- | --- |
| **Use Case:** Yearly Revenue Analysis | | |
|  | | |
| **Use Case Name** | Yearly Revenue Analysis | |
| **Use Case Id** | 12 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | N/A | |
| **Description** | This function does the Yearly Revenue Analysis | |
| **Preconditions** |  | |
| **Trigger** | Click the Yearly Revenue Analysis Button | |
| **Expected Scenario** | **Airport Manager** | **AirlineSYS** |
|  | **Step 1:** Manager invokes Yearly Revenue Analysis function.  **Step 4:** Manager selects the required year for analysis. | **Step 2:** Loads year on UI  **Step 3:** System displays UI.  **Step 5:** The system retrieve’s a summary of:   * Payment details from the Bookings file.   **Step 6:** The system will display UI of selected year.  **Step 8:** The system will retrieve a report of yearly booking analysis by showing.   * Amount. Made * The month it was made   **Step 9:** Thereport UI will be displayed.  **Step 10:** The system will display a confirmation message.  **Step 11:** Results UI. |
| **Alternate Scenarios** | **Airport Manager** | **AirlineSYS** |
| **No data was found for the inputted year** |  | **Step 5:** No data was found for that year.  **Step 6:** The system will display an appropriate message.  **Step 7:** return to step 4 |
| **Conclusions** | The inputted yearly revenue analysis will be displayed | |
| **Post conditions** | Year has been analysed | |
| **Business Rules** | N/A | |
| **Implementation Constraints** | N/A | |

### Yearly Route Analysis

This function does the Yearly Route Analysis

|  |  |  |
| --- | --- | --- |
| **Use Case:** Update Aircraft Type | | |
|  | | |
| **Use Case Name** | Yearly Route Analysis | |
| **Use Case Id** | 13 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | N/A | |
| **Description** | This function does the Yearly Route Analysis | |
| **Preconditions** |  | |
| **Trigger** | Click the Yearly Route Analysis Button | |
| **Expected Scenario** | **Airport Manager** | **AirlineSYS** |
|  | **Step 1: I**nvokes Yearly Route function.  **Step 3:** Managers select the required year for analysis. | **Step 2:** Loads year on UI  **Step 4:** The system retrieves a summary of the route taken from the Flights file and Routes file.  **Step 5:** The system will generate a report of yearly flight analysis.   * Number of Journeys * The Route   **Step 6:** The report UI will be displayed.  **Step 7:** The system will display a confirmation message.  **Step 8:** Results UI. |
| **Alternate Scenarios** | **Airport Manager** | **AirlineSYS** |
| **No data was found for the selected year** |  | **Step 4:** No data was found for that year.  **Step 5:** The system will display an appropriate message.  **Step 6:** return to step 3 |
| **Conclusions** | The inputted yearly revenue route will be displayed | |
| **Post conditions** | Route has been Analysed | |
| **Business Rules** | N/A | |
| **Implementation Constraints** | N/A | |

## Activity Diagram



# System Model

The following dataflow diagrams have been produced for the system:

***External Entities:***

Customer

***Date Stores:***

D1 Airports File

D2 Routes File

D3 Flights File

D4 Operators File

D5 Bookings File

D6 Passengers File

D7 FlightTimes File

***Processes:***

**P1 Manage Airports**

P1.1 Add Airport

P1.2 Add Operator

P1.3 Create Route

P1.4 End Route

**P2 Manage Flights**

P2.1 Schedule Flight

P2.2 Update Flight

P2.3 Cancel Flight

**P3 Process Bookings**

P3.1 Create Booking

P3.2 Update Booking

P3.3 Cancel Booking

**P4 Process Admin**

P4.1 Yearly Revenue Analysis

P4.2 Yearly Route Analysis

## Level-0 DFD

AirlineSYS

Customer

Booking Details

Tickets Details

## Level-1 DFD

Airport Details

Payment Details

Customer

Ticket Details

Booking Details

Payment Details

Booking Details

Operator Details

Operator Details

Flight Details

Route Details

Route Details

P3

Process

Bookings

D52

Bookings File

P1

Manage

Airports

D1

Airport File

D4

Operators File

P2

Manage

Flights

D3

Flights File

D2

Routes File

P4

Process

Admin

D3

Flights File

Route Details

FlightTime File

FlightTime Details

D7

Passenger Details

Passengers File

D6

Route Details

D2

Routes File

## Level-2 DFD (Process P1: Manage Airports)

Airport Details

Airport Details

Route ID

Route Details

P1.1

Add Airport

P1.2

Add Operator

P1.3

End

Route

D2

Routes File

D4

Operators File

Airport File

D1

Operator Details

Route Details

P1.1

Create

Route

## Level-2 DFD (Process P2: Manage Flights)

D3

Flights File

Updated Flights Details

Flights Details

Flights Details

Flight Number

P2.1

Schedule Flight

P2.2

Update

Flight

P2.3

Cancel

Flight

D4

Operators File

D2

Routes File

Operator Details

FlightTime Details

FlightTime File

FlightTime Details

D7

Flights Details

Route Details

FlightTime Details

FlightTime File

D7

D1

Airports File

## Level-2 DFD (Process P3: Process Bookings)

Booking ID

Current Booking Details

Current Booking Details

Booking Details

Updated Booking Details

P3.3

Cancel

Booking

P3.1

Create

Booking

P3.2

Update

Booking

D5

Bookings File

Customer

Books Flight

New Ticket Details

Ticket Details

Updated Passenger Details

Passenger Details

Passengers File

D6

Flight Details

Flight Details

Flight Details

Flights File

D3

Route Details

D2

Routes File

## Level-2 DFD (Process P4: Process Admin)

D5

Bookings File

D2

Routes File

Booking Details

P4.2

Yearly

Revenue

Analysis

P4.3

Yearly

Route

Analysis

Route Details

Booking Details

# Data Model (Class Diagram)

In this Class Diagram it shows there are various table needed for a flight to be book for a destination. The class name clearly describes what they are associated with. Operator had information on operation. Route has information on routes. Airport has information on airports. Flight has information on a scheduled flight. FlightTime represents the departure times available for flights to the schedules. Booking has information on what is flight will be taken and the amount paid for the flight and various other thing. While Passenger has information on the personal information of a passenger that booked the flight.

* 2. **Class Diagram**

- FlightTimes: String

+ getFlightTimes():List<FlightTimes>

FlightTime

has

has

0.. \*

1

0.. \*

0.. \*

1

has

1

2..2

has

1

Booking

- BookingID\*: int

- PassengerID: int

- RouteID: int

- FlightNumber: String

- FlightTime: String

- FlightDate: Date

- SeatNum: int

- NumBaggage: int

- AmountPaid: decimal

- Status: String

+ getNextBookingID() : int

+ getNextSeatNumber(flightNumber: string, routeID: int) : int

+ addBooking() : void

+ updateBooking() : void

+ cancelBooking(bookingID: int, forename: string, surname: string, email: string) : void

+ decreaseNumAvailSeat(flightNumber: string, seats: int) : bool

+ increaseNumAvailSeat(flightNumber: string, seats: int) : bool

+ findBookingDetails(bookingID: int) : DataTable

1

has a

- RouteID\*:int

- DeptAirport: String

- ArrAirport: String

- TicketPrice: decimal

- Duration: int

- Status: String

+ getNextRouteID(): int

+ createRoute(): void

+ getRoutes(): List<Route>

+ endRoute(int routeID): void

+ getAllRouteDetails(): List<Route>

+ doesRouteExist(deptAirport: string, arrAirport: string): bool

+ getRouteID(dept: string, arr: string): int

+ getDuration(dept: string, arr: string): int

+ getAirportsByRouteID(routeID: int): Route

Route

0.. \*

- PassengerID\*: int

- Forename: String

- Surname: String

- DateOfBirth: Date

- Email: String

- Phone: String

- Eircode: String

+ getRoutePrice(deptAirport string, arrAirport: string): decimal

+ addPassenger(): void

+ updatePassenger(passengerID: int): void

Passenger

0.. \*

1

Operated by

Operator

- OperatorCode\*: String

- Name: String

- City: String

- Country: String

+ addOperator():void

+ getOperator(): List < Operator>

+ checkOperatorExists(string operatorCode) bool

- FlightNumber\*: String

- OperatorCode: String

- RouteID: int

- FlightDate: Date

- FlightTime: String

- EstArrTime: String

- NumSeats: int

- NumSeatAvail: int

- Status: String

+ scheduleFlight() : void

+ getFlightNumber(selectedOperatorCode: string) : string

+ updateFlight(flightNumber: string) : void

+ getAllFlightDetails() : List<Flight>

+ cancelFlight(flightNumber: string) : void

+ getAvailableFlights(routeID: int) : List<string[]>

+ getActiveFlights() : List<Flight>

Flight

Airport

- AirportCode\*: String

- Name: String

- Street: String

- City: String

- Country: String

- Eircode: String

- Phone: String

- Email: String

+ addAirport(): void

+ getAvailAirports(): List< string>

+ checkAirportExists(string): bool

* 1. **Relational Schema**

Airports (AirportCode, Name, Street, City, Country, Eircode, Phone, Email)

Operators (OperatorCode, Name, City, Country)

Routes (RouteID, DeptAirport, ArrAirport, TicketPrice, Duration, Status)

FlightTimes (FlightTime)

Flights (FlightNumber, OperatorCode , RouteID, FlightDate, FlightTime, EstArrTime, NumSeats, NumSeatAvail, Status)

Passengers (PassengerID, Forename, Surname, DateOfBirth, Email, Phone, Eircode)

Bookings (BookingID, PassengerID, RouteID, FlightNumber, FlightTime, SeatNum NumBaggage, AmountPaid, Status)

## Database Schema

**Schema**: AirlineSYS

**Relation:** Airports

**Attributes:**

AirportCode VARCHAR2 (3) NOT NULL

Name VARCHAR2 (50) NOT NULL

Street VARCHAR2 (50) NOT NULL

City VARCHAR2 (50) NOT NULL

Country VARCHAR2 (50) NOT NULL

Eircode VARCHAR2 (7) NOT NULL

Phone VARCHAR2 (10) NOT NULL

Email VARCHAR2 (60) NOT NULL

Primary Key AirportCode

**Relation:** Operators

**Attributes:**

OperatorCode VARCHAR2 (2) NOT NULL

Name VARCHAR2 (50) NOT NULL

City VARCHAR2 (50) NOT NULL

Country VARCHAR2 (50) NOT NULL

Primary Key OperatorCode

**Relation:** Routes

**Attributes:**

RouteID NUMBERIC (2) NOT NULL

DeptAirport VARCHAR2 (3) NOT NULL

ArrAirport VARCHAR2 (3) NOT NULL

TicketPrice NUMERIC (6,2) CHECK TicketPrice > 0 and TicketPrice <=999 NOT NULL

Duration NUMERIC (4) NOT NULL

Status CHAR (1) NOT NULL CHECK Status IN ('A', 'I') NOT NULL

Primary Key RouteID

Foreign Key DeptAirport References Airports

Foreign Key ArrAirport References Airports

**Relation:** FlightTimes

**Attributes:**

FlightTime VARCHAR2(8) NOT NULL

Primary Key FlightTime

**Relation:** Flights

**Attributes:**

FlightNumber VARCHAR2 (6) NOT NULL

OperatorCode VARCHAR2(2) NOT NULL

RouteID NUMERIC (2) NOT NULL

FlightDate Date NOT NULL

FlightTime VARCHAR2 (8) NOT NULL

EstArrTime VARCHAR2 (8) NOT NULL

NumSeats NUMERIC(3) CHECK NUMSEATS > 0 AND NUMSEATS <= 100 NOT NULL

NumSeatAvail NUMERIC(3) CHECK NUMSEATAVAIL >= 0 NOT NULL

Status CHAR (1) NOT NULL NOT NULL

Primary Key FlightNumber

Foreign Key RouteID References Routes

Foreign Key OperatorCode References Operators

Foreign Key FlightTime References FlightTimes

**Relation:** Passengers

**Attributes:**

PassengerID NUMERIC (2) NOT NULL

Forename VARCHAR2 (50) NOT NULL

Surname VARCHAR2 (50) NOT NULL

DateOfBirth Date NOT NULL

Email VARCHAR2 (50) NOT NULL

Phone VARCHAR2 (50) NOT NULL

Eircode VARCHAR2(7) NOT NULL

Primary Key PassengerID, BookingID

**Relation:** Bookings

**Attributes:**

BookingID NUMERIC (2) NOT NULL

PassengerID NUMERIC (2) NOT NULL

RouteID NUMERIC (2) NOT NULL

FlightNumber VARCHAR2 (6) NOT NULL

FlightTime VARCHAR2 (8) NOT NULL

FlightDate Date NOT NULL

SeatNum INT NOT NULL NOT NULL

NumBaggage INT NOT NULL

AmountPaid DECIMAL (6,2) NOT NULL

Status VARCHAR (9) CHECK STAUS IN ('CONFIRMED','CANCELLED') NOT NULL

Primary Key BookingID

Foreign Key PassengerID Refences Passengers

Foreign Key RouteID Refences Routes

Foreign Key FlightNumber Refences Flights

Foreign Key FlightTime References FlightTimes

# Conclusion

In conclusion, the system functioned as intended, fulfilling its primary purpose of providing an airline with a system for managing airports, operators, and routes to schedule flights. These scheduled flights were successfully made available for customers to book. Throughout the process of creating airports, operators, and routes, various pieces of information were needed. Flight scheduling involved selecting two airports to active routes, which was a departure and arrival airport, with the system catering to one-way journeys. The chosen operator for scheduling represented the airline responsible for the flight's operation. Routes could be ended due to factors like low demand or financial issues. Additionally, once flights were scheduled, updates or cancellations were possible if required. A Booking included providing various personal information along with the desired destination.