Flight Management System

Version 1.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 11/Apr/2015 | 1.0 | Created the use case specification and description document | Michael Kong |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Brief Description 4

1.2 Glossary of Terms 4

2. Use Case Descriptions 4

2.1 Use Case: Login 4

2.2 Use Case: Enter Wrong Credentials 4

2.3 Use Case: Signup 5

2.4 Use Case: Enter Invalid Details 5

2.5 Use Case: Cancel Booking 5

2.6 Use Case: View Flight History 6

2.7 Use Case: Modify Account Details 6

2.8 Use Case: Make Booking 7

2.9 Use Case: Add Persons 7

2.10 Use Case: Add Customers 8

2.11 Use Case: Enter Invalid Username 8

2.12 Use Case: Book Services 8

2.12 Use Case: Book Services 8

# Introduction

## Brief Description

This document covers all the use cases involved in the Flight Management System. Each use case description will cover exactly one use case as presented in the use case diagram. A single use case description contains the use case name, the use case ID, the stakeholders and goals, the description of the use case in question, the participating actors, the trigger that activates this use case, and the flow of events including normal flows, sub-flows and exceptional flows.

## Glossary of Terms

**Airport details** – Refers to the details of an airport that are: airport ID, airport name, city, country, IATA/FAA code, latitude, longitude, altitude, time zone, DST and TZ database time zone.

**Flight details** – Refers to the details of a flight that are: flight ID, plane type, route number, departure time and arrival time.

**Person details –** Refers to the details of the extra person that the Customer actor or Travel Agency actor might add in a booking that are: title, first name, last name, gender, date of birth, phone number, email, street address, state, city, country, credit card type, credit card number, and whether they hold a passport.

**Plane details** – Refers to the details of a plane that are: the aircraft model, number of first class seats, number of business class seats, number of premium economy class seats and number of ecnomy seats.

**User** – When in used in context of a use case description, it refers to the actors of that use case in question.

**User** **credentials** – This refers to the username and password of the system’s user that will be used as the primary means of identifying each user of the system.

**User details** – Refers to the details of the user that are: title, first name, last name, e gender, date of birth, phone number, email, street address, state, city, country, credit card type, credit card number, and whether they hold a passport.

# Use Case Descriptions

## Use Case: Login

|  |  |
| --- | --- |
| **Name:** Login | **ID:** 1 |
| **Stakeholders and goals:**  Airline customers and airline staff – Need to login to the system to access the functions of the system. | |
| **Description:** A user must enter their user credentials into the system through the interface to access the functionality of the system. | |
| **Actors:** Travel agency, Customer, Normal Staff, Service System Manager, Flight Manager, System Administrator, Reservation System Manager, Profile System Manager, Reporting System Manager | |
| **Trigger:** A user accesses the login interface of the system. | |
| **Normal flow:**   1. User enters user credentials into the system. 2. System checks the user credentials with the database. 3. If the user credentials are correct, the user is logged in and system displays the appropriate user main menu for that particular user. 4. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:**  2a Incorrect user credentials entered: The *Enter Wrong Credentials* use case is performed. | |

## Use Case: Enter Wrong Credentials

|  |  |
| --- | --- |
| **Name:** Enter Wrong Credentials | **ID:** 2 |
| **Stakeholders and goals:**  Airline customers and airline staff – Inform the user that the user credentials that were entered into the system are incorrect and perform the Login use case again. | |
| **Description:** A user has entered incorrect user credentials into the system. | |
| **Actors:** Travel agency, Customer, Normal Staff, Service System Manager, Flight Manager, System Administrator, Reservation System Manager, Profile System Manager, Reporting System Manager | |
| **Trigger:** A user of the system enters the wrong credentials into the system. | |
| **Normal flow:**   1. System finds a mismatch in the user credentials and cannot find the corresponding user credentials. 2. System displays an error message on the user interface. 3. User performs *Login* use case. 4. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:** None | |

## Use Case: Signup

|  |  |
| --- | --- |
| **Name:** Signup | **ID:** 3 |
| **Stakeholders and goals:**  Airline customers – A new airline customer wants to create a new account in the system to access the functionality of the system. | |
| **Description:** A customer or travel agency wants to create a new account | |
| **Actors:** Travel agency, Customer | |
| **Trigger:** A user accesses the signup interface of the system. | |
| **Normal flow:**   1. User enters user details into the system. 2. System checks validity of user details. 3. User performs *Login* use case. 4. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:**  2a Invalid details are entered: The *Enter Invalid Details* use case is performed. When that use case has been performed, redo *Signup* use case. | |

## Use Case: Enter Invalid Details

|  |  |
| --- | --- |
| **Name:** Enter Invalid Details | **ID:** 4 |
| **Stakeholders and goals:**  Airline customers – Inform the user that the user details that were entered into the system are invalid and perform the Signup use case again. | |
| **Description:** A user has entered invalid user details into the system | |
| **Actors:** Travel agency, Customer | |
| **Trigger:** A user of the system enters invalid details into the system. | |
| **Normal flow:**   1. System finds that the details do not match the expected format for a particular field. 2. System displays an error message on the user interface. 3. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:** None | |

## Use Case: Cancel Booking

|  |  |
| --- | --- |
| **Name:** Cancel Booking | **ID:** 5 |
| **Stakeholders and goals:**  Airline customers – Wants to cancel a flight booking that was previously made. | |
| **Description:** The user wants to cancel a flight booking, but the user will be charged a cancellation fee. | |
| **Actors:** Travel agency, Customer | |
| **Trigger:** A user of the system chooses the Cancel Booking option from the user interface. | |
| **Normal flow:**   1. User selects Cancel Booking option from the main menu. 2. System displays all flights that are booked in a numbered list. 3. User selects the number associated with the booking to cancel. 4. System acknowledges the cancellation and charges a cancellation fee to the user’s credit card. 5. System returns to the main menu. 6. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:** None | |

## Use Case: View Flight History

|  |  |
| --- | --- |
| **Name:** View Flight History | **ID:** 6 |
| **Stakeholders and goals:**  Airline customers – Wants to view the flights that were previously made. | |
| **Description:** The user wants to view the view all flights that they have flown on. | |
| **Actors:** Travel agency, Customer | |
| **Trigger:** A user of the system chooses the View Flight History option from the user interface. | |
| **Normal flow:**   1. User selects View Flight History option from the main menu. 2. System displays all flights that the user has flown on in a numbered list. 3. System returns to the main menu. 4. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:** None | |

## Use Case: Edit Account Details

|  |  |
| --- | --- |
| **Name:** Edit Account Details | **ID:** 7 |
| **Stakeholders and goals:**  Airline customers – Wants to change the details of their account. | |
| **Description:** The user wants to edit their user details and have it updated in the system. | |
| **Actors:** Travel agency, Customer | |
| **Trigger:** A user of the system chooses the Modify Account Details option from the user interface. | |
| **Normal flow:**   1. User selects Modify Account Details option from the main menu. 2. System displays the modifiable user details in a numbered list. 3. User selects the user detail that they want to modify by the number. 4. System displays a prompt to retrieve input. 5. User enters new detail. 6. System saves modified user detail to database. 7. System returns to the main menu. 8. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:** None | |

## Use Case: Make Booking

|  |  |
| --- | --- |
| **Name:** Make Booking | **ID:** 8 |
| **Stakeholders and goals:**  Airline customers – Wants to make a booking for a flight. | |
| **Description:** The user wants to make a booking for a flight and perform payment for their booking. | |
| **Actors:** Travel agency, Customer | |
| **Trigger:** A user of the system chooses the Book Flight option from the user interface. | |
| **Normal flow:**   1. User selects Book Flight option from the main menu. 2. System displays prompt for flight destination and origin. 3. User enters desired flight destination and origin. 4. System displays the list of flights available corresponding to the given flight destination and origin in a numbered list. 5. User selects flight to book by the corresponding number. 6. System performs the appropriate flow according to the type of user:    1. If the user is a Customer Actor, the *Add Persons* use case is performed.    2. If the user is a Travel Agency Actor, the *Add Customers* use case is performed. 7. System displays booking summary and prompt if user wants to book services. 8. If user chooses not to book services, system prompts user to make payment. 9. User pays by credit card and can choose whether to use frequent flier points to get a discount. 10. System records payment details and updates database. 11. System sends flight information to user via email. 12. System returns to main menu. 13. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:**  9a User chooses to book services: The *Book Services* use case is performed. | |

## Use Case: Add Persons

|  |  |
| --- | --- |
| **Name:** Add Persons | **ID:** 9 |
| **Stakeholders and goals:**  Airline customers – Wants to add persons to the booking. | |
| **Description:** The user wants to make a booking for additional people on their behalf. | |
| **Actors:** Travel Agency, Customer | |
| **Trigger:** Customer has selected a flight in the Make Booking use case OR Travel Agency makes booking on behalf of other persons. | |
| **Normal flow:**   1. System prompts user if they want to add persons to the booking. 2. If user does not want to add persons to booking, go to step 7. 3. System prompts user for the person details. 4. User fills in the person details. 5. System prompts user whether there is another person. 6. If user enters that there is another person, perform step 3-5 again. 7. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:**  2a Invalid details are entered: The *Enter Invalid Details* use case is performed. When that use case has been performed, go to step 3. | |

## Use Case: Add Customers

|  |  |
| --- | --- |
| **Name:** Add Customers | **ID:** 10 |
| **Stakeholders and goals:**  Airline travel agencies – Wants to add customers to the booking. | |
| **Description:** The user wants to make a booking for customers. | |
| **Actors:** Travel Agency | |
| **Trigger:** A user of the system has selected a flight in the Make Booking use case. | |
| **Normal flow:**   1. System prompts for the usernames of the customers to book for. 2. User enters usernames of the customers. 3. *Add Persons* use case is performed. 4. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:**  2a Invalid usernames are entered: The *Enter Invalid Username* use case is performed. When that use case has been performed, go to step 1. | |

## Use Case: Enter Invalid Username

|  |  |
| --- | --- |
| **Name:** Enter Invalid Username | **ID:** 11 |
| **Stakeholders and goals:**  Airline travel agencies – Informs the user that the username does not exist in the system. | |
| **Description:** The user has entered invalid username into the system and the system will display an error. | |
| **Actors:** Travel Agency, Normal Staff | |
| **Trigger:** A user of the system has entered a username/usernames that does not exist in the system. | |
| **Normal flow:**   1. System cannot find the username/usernames provided. 2. System displays an error message on the user interface. 3. User acknowledges the error message. 4. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:** None | |

## Use Case: Book Services

|  |  |
| --- | --- |
| **Name:** Book Services | **ID:** 12 |
| **Stakeholders and goals:**  Airline customers – Wants to book services to add on to the flight booking. | |
| **Description:** The user wants to make booking for services that they can have on the flight. | |
| **Actors:** Travel agency, Customer, Normal Staff | |
| **Trigger:** A user of the system chooses to book services when prompted in the Make Booking use case. | |
| **Normal flow:**   1. System displays all services in a numbered list. 2. User selects the numbers of the services that they want. 3. System records the services for this user. 4. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:** None | |

## Use Case: Cancel Customer Booking

|  |  |
| --- | --- |
| **Name:** Cancel Customer Booking | **ID:** 13 |
| **Stakeholders and goals:**  All staff – Want to cancel a booking of a customer. | |
| **Description:** The user wants to cancel the booking of a customer | |
| **Actors:** Normal Staff | |
| **Trigger:** A user of the system chooses the Cancel Customer Booking option from the user interface. | |
| **Normal flow:**   1. System prompts user for customer’s username. 2. User enters the username of the customer. 3. System displays the current bookings of the customer with the corresponding username in a numbered list. 4. User selects the list number of the booking to cancel it. 5. System records the changes in the database. 6. System sends email to customer who has had their booking cancelled. 7. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:**  2a Invalid username is entered: The *Enter Invalid Username* use case is performed. When that use case has been performed, go to step 1 | |

## Use Case: Edit Services in Customer Booking

|  |  |
| --- | --- |
| **Name:** Edit Services in Customer Booking | **ID:** 14 |
| **Stakeholders and goals:**  All staff – Want to edit the services that are currently booked by a customer. | |
| **Description:** The user wants to edit services in a booking. | |
| **Actors:** Normal Staff | |
| **Trigger:** A user of the system chooses the Modify Services in Customer Booking option from the user interface. | |
| **Normal flow:**   1. System prompts user for customer’s username. 2. User enters the username of the customer. 3. System displays the current bookings of the customer with the corresponding username in a numbered list. 4. User selects the list number of the booking. 5. System displays the list of services that the customer booked for that booking in a numbered list. 6. User enters a whether they want to add, change or delete a service.    1. Add service – Sub-flow S1    2. Change service – Sub-flow S2    3. Delete service – Sub-flow S3 7. System returns to the main menu. 8. End | |
| **Sub-flows:**  S1: Add service   1. System displays a list of services currently available in a numbered list. 2. User enters numbers of the corresponding services. 3. System accepts input and saves the addition of service to database.   S2: Change service   1. User selects the number of the corresponding service that is currently booked. 2. System displays a list of services currently available in a numbered list. 3. User enters numbers of the corresponding services. 4. System accepts input and saves the change of service to database.   S3: Delete service   1. User selects the number of the corresponding service that is currently booked. 2. System accepts input and saves the deletion of service to database. | |
| **Alternative/Exceptional flows:** None | |

## Use Case: Add Services

|  |  |
| --- | --- |
| **Name:** Add Services | **ID:** 15 |
| **Stakeholders and goals:**  Service System Manager – Want to add services that customer can book for a flight. | |
| **Description:** The user wants to add services to the current list of services so that customers can have a larger variety of services to choose from. | |
| **Actors:** Service System Manager | |
| **Trigger:** A user of the system chooses the Add Services option from the user interface. | |
| **Normal flow:**   1. System prompts user for the service name, service cost and availability. 2. User enters the required details. 3. System prompts user if there is another service to add. 4. If user wants to add another service, repeat steps 1-3; else go to step 5. 5. System saves all added services to database. 6. System returns to the main menu. 7. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:** None | |

## Use Case: Remove Services

|  |  |
| --- | --- |
| **Name:** Remove Services | **ID:** 16 |
| **Stakeholders and goals:**  Service System Manager – Want to remove services that customer can book for a flight. | |
| **Description:** The user wants to remove services from the current list of services. | |
| **Actors:** Service System Manager | |
| **Trigger:** A user of the system chooses the Remove Services option from the user interface. | |
| **Normal flow:**   1. System displays all services in a numbered list. 2. User selects the list numbers of the services to remove. 3. System removes all selected services in the database. 4. System returns to the main menu. 5. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:** None | |

## Use Case: Edit Service Price

|  |  |
| --- | --- |
| **Name:** Edit Service Price | **ID:** 17 |
| **Stakeholders and goals:**  Service System Manager – Want to edit prices of the services that are saved in the system. | |
| **Description:** The user wants edit the prices of existing services. | |
| **Actors:** Service System Manager | |
| **Trigger:** A user of the system chooses the Edit Service Price option from the user interface. | |
| **Normal flow:**   1. System displays all services in a numbered list. 2. User selects the list number of the service to adjust its price. 3. System prompts the user for the new price for the selected service. 4. User enters the new price for the selected service. 5. System saves the new price of the service to the database. 6. System prompts user whether they want to adjust the price of another service. 7. If the user wants to adjust the price of another service, then repeat steps 1-6; else go to step 8. 8. System returns to the main menu. 9. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:** None | |

## Use Case: Edit Service Availability

|  |  |
| --- | --- |
| **Name:** Edit Service Availability | **ID:** 18 |
| **Stakeholders and goals:**  Service System Manager – Want to edit the availability of the services for different types of flights. | |
| **Description:** The user wants to edit the availability of services to “international” or “all” flights. | |
| **Actors:** Service System Manager | |
| **Trigger:** A user of the system chooses the Adjust Service Availability option from the user interface. | |
| **Normal flow:**   1. System displays all services in a numbered list. 2. User selects the list number of the service to adjust availability. 3. System prompts user to input the availability of the selected service. 4. User enteres the new availability for the selected service. 5. System saves the new availability of the service to the database. 6. System prompts the user whether they want to adjust the availability of another service. 7. If the user wants to adjust the availability of another service, then repeat steps 1-6; else go to step 8. 8. System returns to the main menu. 9. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:** None | |

## Use Case: Edit Routes

|  |  |
| --- | --- |
| **Name:** Edit Routes | **ID:** 19 |
| **Stakeholders and goals:**  Flight Manager – Want to edit the routes that a flight can take. | |
| **Description:** The user wants to add, modify or delete the routes that a flight can take which include the origin airport and the destination airport of the route. | |
| **Actors:** Flight Manager | |
| **Trigger:** A user of the system chooses the Edit Routes option from the user interface. | |
| **Normal flow:**   1. System prompts user if they want to add, modify or delete route. 2. System displays the appropriate flow according to the user’s choice.    1. Add route – Sub-flow S1    2. Edit route – Sub-flow S2    3. Delete route – Sub-flow S3 3. System returns to main menu. 4. End | |
| **Sub-flows:**  S1: Add route   1. System prompts user for the origin airport and destination airport. 2. User enters the origin and destination airport. 3. If route already exists, system displays an error message and go to step 1; else go to step 4. 4. System prompts user for codeshare and number of stops. 5. User enters codeshare and number of stops for the route. 6. System saves the new route to database.   S2: Edit route   1. System prompts user for the origin airport and destination airport. 2. User enters the origin and destination airport. 3. If route is not found, display an error message and go to step 1; else go to step 4. 4. System displays the origin airport, destination airport, codeshare and number of stops in a numbered list and prompts user to choose which detail to edit. 5. User enters which detail they want to edit. 6. System prompts user for new detail. 7. User enters new detail. 8. If the new detail is an airport code and it does not exist in the database, system displays an error message and go to step 4; else go to step 9. 9. If the route already exists, system displays an error message and go to step 4; else go to step 10. 10. System updates the database with the new route information.   S3: Delete route   1. System prompts user for the origin airport and destination airport. 2. User enters the origin and destination airport. 3. If route is not found, system displays an error message and go to step 1; else go to step 4. 4. System deletes route from the database. | |
| **Alternative/Exceptional flows:** None | |

## Use Case: Edit Fleet

|  |  |
| --- | --- |
| **Name:** Edit Fleet | **ID:** 20 |
| **Stakeholders and goals:**  Flight Manager – Want to edit the planes that are curently in the system. | |
| **Description:** The user wants to add, modify or delete the planes. | |
| **Actors:** Flight Manager | |
| **Trigger:** A user of the system chooses the Edit Fleet option from the user interface. | |
| **Normal flow:**   1. System prompts user if they want to add, modify or delete plane. 2. System displays the appropriate flow according to the user’s choice.    1. Add plane – Sub-flow S1    2. Edit plane – Sub-flow S2    3. Delete plane – Sub-flow S3 3. System returns to main menu. 4. End | |
| **Sub-flows:**  S1: Add plane   1. System prompts user for the plane details. 2. User enters the plane details into the system. 3. If the system detects a mismatch in the input with the expected input, system displays error message and go to step 1; else go to step 4. 4. System saves the new plane to database.   S2: Edit plane   1. System prompts user for the plane details. 2. User enters the aircraft type into the system. 3. If the system is not able to find the aircraft type entered, system displays error message and go to step 1; else go to step 4. 4. System displays seat types and the number of seats for that type in a numbered list, and prompts the user for 5. User selects the seat type by the number. 6. System prompts user for seat addition or remove. (Positive number for adding seats and negative number for removing seats) 7. If the number of seats to remove is less than the actual number, system displays an error message and go to step 4; else go to step 8. 8. System saves the modified seats to database.   S3: Delete plane   1. System prompts user for the plane details. 2. User enters the plane details into the system. 3. If the system detects a mismatch in the input with the expected input, system displays error message and go to step 1; else go to step 4. 4. If the system detects that the number of seats entered by the user is greater than that number stored in the database, system displays an error message and go to step 1; else go to step 5. 5. System deletes the plane from the database. | |
| **Alternative/Exceptional flows:** None | |

## Use Case: Edit Flight Schedule

|  |  |
| --- | --- |
| **Name:** Edit Flight Schedule | **ID:** 21 |
| **Stakeholders and goals:**  Flight Manager – Want to edit the flights in the system. | |
| **Description:** The user wants to add, modify or delete flights. | |
| **Actors:** Flight Manager | |
| **Trigger:** A user of the system chooses the Edit Flight Schedule option from the user interface. | |
| **Normal flow:**   1. System prompts user if they want to add, modify or delete flight. 2. System displays the appropriate flow according to the user’s choice.    1. Add flight – Sub-flow S1    2. Edit flight – Sub-flow S2    3. Delete flight – Sub-flow S3 3. System returns to main menu. 4. End | |
| **Sub-flows:**  S1: Add flight   1. System prompts user for the flight details. 2. User enters the flight details into the system. 3. If the system detects a mismatch in the input with the expected input, system displays error message and go to step 1; else go to step 4. 4. System saves the new flight to database.   S2: Edit flight   1. System prompts user for the flight ID, source airport and destination airport. 2. User enters the flight ID, source airport and destination airport into the system. 3. If the system cannot find the flight with the corresponding flight ID, source airport and destination airport, system displays error message and returns to step 1; else go to step 4. 4. System displays all the flight details except for the flight ID in a numbered list and system prompts user to choose which detail they want to modify. 5. User enters the detail to modify by the associated number. 6. System prompts user for the updated detail. 7. User enters updated detail into the system. 8. If the system detects an input mismatch, system displays error message and go to step 4; else go to step 9. 9. System saves modifications to database.   S3: Delete flight   1. System prompts user for the flight ID, source airport and destination airport. 2. User enters the flight ID, source airport and destination airport into the system. 3. If the system cannot find the flight with the corresponding flight ID, source airport and destination airport, system displays error message and returns to step 1; else go to step 4. 4. System deletes the flight from the database. | |
| **Alternative/Exceptional flows:** None | |

## Use Case: Edit Airports

|  |  |
| --- | --- |
| **Name:** Edit Airports | **ID:** 22 |
| **Stakeholders and goals:**  !!!Flight Manager – Want to edit the flights in the system. | |
| **Description:** The user wants to add, modify or delete airports. | |
| **Actors:** Flight Manager | |
| **Trigger:** A user of the system chooses the Edit Airport option from the user interface. | |
| **Normal flow:**   1. System prompts user if they want to add, modify or delete aiport. 2. System displays the appropriate flow according to the user’s choice.    1. Add airport – Sub-flow S1    2. Edit flight – Sub-flow S2    3. Delete flight – Sub-flow S3 3. System returns to main menu. 4. End | |
| **Sub-flows:**  S1: Add airport   1. System prompts user for the airport details. 2. User enters the airport details into the system. 3. If the system detects a mismatch in the input with the expected input, system displays error message and returns to step 1; else go to step 4. 4. If the system detects a duplicate airport ID or airport name in the database, system displays an error and returns to step 1; else go to step 5. 5. System saves the new airport to database.   S2: Edit airport   1. System prompts user for the airport ID. 2. User enters the airport ID into the system. 3. If the system cannot find the airport with the corresponding airport ID, system displays error message and returns to step 1; else go to step 4. 4. System displays all the airport details except for the airport ID in a numbered list and system prompts user to choose which detail they want to modify. 5. User enters the detail to modify by the associated number. 6. System prompts user for the updated detail. 7. User enters updated detail into the system. 8. If the system detects an input mismatch, system displays error message and returns to step 4; else go to step 9. 9. System saves modifications to database.   S3: Delete airport   1. System prompts user for the aiport ID. 2. User enters the airport ID into the system. 3. If the system cannot find the airport with the corresponding airport ID, system displays error message and returns to step 1; else go to step 4. 4. System deletes the airport from the database. | |
| **Alternative/Exceptional flows:** None | |

## Use Case: Move Passengers Between Flights

|  |  |
| --- | --- |
| **Name:** Move Passengers Between Flights | **ID:** 23 |
| **Stakeholders and goals:**  !!!Flight Manager – Want to edit the flights in the system. | |
| **Description:** The user wants to move passengers between flights. | |
| **Actors:** Reservation System Manager | |
| **Trigger:** A user of the system chooses the Move Passengers Between Flights option from the user interface. | |
| **Normal flow:**   1. System prompts user for the flight ID, source airport and detination airport. 2. User enters flight ID, source airport and detination airport into the system. 3. If the system is unable to find a flight with the details entered, system displays error message and returns to step 1; else go to step 4. 4. System displays a list of customers on the flight with the id, first name and last name, and prompts user for the ids of the customers to move to another flight. 5. User enters the ids of the customers to move to another flight. 6. If the system detects that some of the ids are not valid (are not on the flight or do not exist), system displays error message and return to step 4; else go to step 7. 7. System prompts user for the flight ID, source airport and detination airport of the flight to transfer customers to. 8. User enters flight ID, source airport and destination airport into the system. 9. If the system is unable to find a flight with the details entered, system displays error message and returns to step 7; else go to step 10. 10. For each customer, system prompts user to enter the seat number for the customer. 11. If the seat number is already occupied or the seat number is invalid, system displays an error message and returns to step 10 for the particular customer that it failed at; else go to step 12. 12. System saves the changes to database. 13. System returns to main menu. 14. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:** None | |

## Use Case: Set Ticket Prices

|  |  |
| --- | --- |
| **Name:** Set Ticket Prices | **ID:** 24 |
| **Stakeholders and goals:**  !!!Flight Manager – Want to edit the flights in the system. | |
| **Description:** The user wants to set the prices of tickets for flights. | |
| **Actors:** Reservation System Manager | |
| **Trigger:** A user of the system chooses the Set Ticket Prices option from the user interface. | |
| **Normal flow:**   1. System prompts user for the flight ID, source airport and detination airport. 2. User enters flight ID, source airport and detination airport into the system. 3. If the system is unable to find a flight with the details entered, system displays error message and returns to step 1; else go to step 4. 4. System displays current price and prompts user for new price. 5. User enters new price. 6. If system detects that the input is invalid, system displays an error message and returns to step 4; else go to step 7. 7. System saves updated price to the database. 8. System returns to main menu. 9. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:** None | |

## Use Case: Move Passengers within a Flight

|  |  |
| --- | --- |
| **Name:** Move Passengers within a Flight | **ID:** 25 |
| **Stakeholders and goals:**  !!!Flight Manager – Want to edit the flights in the system. | |
| **Description:** The user wants to move passengers within a flight. | |
| **Actors:** Reservation System Manager | |
| **Trigger:** A user of the system chooses the Set Ticket Prices option from the user interface. | |
| **Normal flow:**   1. System prompts user for the flight ID, source airport and detination airport. 2. User enters flight ID, source airport and detination airport into the system. 3. If the system is unable to find a flight with the details entered, system displays error message and returns to step 1; else go to step 4. 4. System displays a list of customers on the flight with the id, first name and last name, and prompts user for the ids of the customers to move within the flight. 5. User enters the ids of the customer to move within the flight. 6. If the system detects that some of the ids are not valid (are not on the flight or do not exist), system displays error message and return to step 4; else go to step 7. 7. For each customer, system prompts user to enter the seat number for the customer. 8. If the seat number is already occupied or the seat number is invalid, system displays an error message and returns to step 7 for the particular customer that it failed at; else go to step 9. 9. System saves the changes to database. 10. System returns to main menu. 11. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:** None | |

## Use Case: Edit Watch and No Fly List

|  |  |
| --- | --- |
| **Name:** Edit Watch and No Fly List | **ID:** 26 |
| **Stakeholders and goals:**  !!!Flight Manager – Want to edit the flights in the system. | |
| **Description:** The user wants to move passengers within a flight. | |
| **Actors:** Reservation System Manager | |
| **Trigger:** A user of the system chooses the Set Ticket Prices option from the user interface. | |
| **Normal flow:**   1. System prompts user for the flight ID, source airport and detination airport. 2. User enters flight ID, source airport and detination airport into the system. 3. If the system is unable to find a flight with the details entered, system displays error message and returns to step 1; else go to step 4. 4. System displays a list of customers on the flight with the id, first name and last name, and prompts user for the ids of the customers to move within the flight. 5. User enters the ids of the customer to move within the flight. 6. If the system detects that some of the ids are not valid (are not on the flight or do not exist), system displays error message and return to step 4; else go to step 7. 7. For each customer, system prompts user to enter the seat number for the customer. 8. If the seat number is already occupied or the seat number is invalid, system displays an error message and returns to step 7 for the particular customer that it failed at; else go to step 9. 9. System saves the changes to database. 10. System returns to main menu. 11. End | |
| **Sub-flows:** None | |
| **Alternative/Exceptional flows:** None | |