**USE CASE DESCRIPTION**

**(AIR TICKET RESERVATION SYSTEM)**

1. **LOGIN OR SIGN-UP USE CASE**

|  |
| --- |
| **Introduction** : This function allows a registered passenger to login his account using his ID airline and password. If a passenger is not registered, the website shall allow the passenger to enroll first. |
| **Actors** : Passenger, Administrator, Departmental Staff |
| **Pre-Condition** : The actor must be logged to a suitable internet connection.if the passenger is already logged in he needs to log in .Other users have the option to sign up. |
| **Post-Condition** : After successful login signup passenger proceeds to flight booking page |
| **EVENT FLOW** |
| **Basic Flow** :   1. **Already Registered :** 2. The system prompts the login screen to the passenger asking for username and 3. password. 4. The passenger enters his login ID and password. 5. The system validates the actor’s password and logs him/her into the system. 6. **New Passenger :**  * If not already registered passenger registers using new credentials . * Login ID is alpha-numeric and exactly 6 character long * The password is 8-15 character long, must be alphanumeric and have at least one special character . |
| **Alternate Flow** :  **Alternate Flow 1** :  **Invalid details**   * If the passenger fills in the invalid details as in invalid login id,name or any other details then the passenger has to again fill in the details.   **Alternate Flow 2** : **Not Signed Up**   * The passenger can sign up with the system if he hasn’t enrolled before.   **Alternate Flow 3** : **Actor Terminates/Exits**   * This allows the actor to exit anytime during the use case and the state of the system remains unchanged |
| **Special Requirement** : None. |
| **Associated Use Case** : None. |

1. **MAINTAIN FLIGHT DETAILS USE CASE**

|  |
| --- |
| **Introduction** : This use case documents the steps that must be followed to maintain the flight details. |
| **Actors** : Administrator, Departmental Staff. |
| **Pre-Condition** : The actors must be logged into the system. |
| **Post-Condition** : The actors must be successfully able to maintain the flight details and the state of the system is changed otherwise it remains the same. |
| **EVENT FLOW** |
| **Basic** **Flow** : The actors are able to maintain the flight details as in :   * The actor enters in the details of flights as in the flights number and other details and the details regarding the flights are maintained. * Then the actor have the option of adding a flight,updating a flight details,removing a flight detail. : * **Adding a flight** : The actor can enter new flight details. * **Updating a flight** : The actor can  update existing flight details as in its timing , the route and the delays if any. * **Removing a flight** : The actor can remove the flight details in case if it is not functional or it is having another issue. |
| **Alternate** **Flow1** : **Invalid Details**   * If the actor fills in the invalid details of the flight then the state remains same and the actors have to fill in the details again.   **Alternate Flow 2** : **Actor Terminates/Exits**   * This allows the actor to exit anytime during the use case and the state of the system remains unchanged. |
| **Special Requirement** : None. |
| **Associated Use Case** : Login. |

1. **MAINTAIN PASSENGER DETAILS USE CASE**

|  |
| --- |
| **Introduction** : This use case documents the steps that must be followed to maintain the user/passenger  details. |
| **Actors** : Administrator |
| **Pre-Condition** : The actors must be logged into the system. |
| **Post-Condition** : The actors must be successfully able to maintain the passenger details and the state of the system is changed otherwise it remains the same. |
| **EVENT FLOW** |
| **Basic** **Flow** : Actors are able to log in and then various options are displayed as :   * Then the actor have the option of adding a passenger,updating a passenger details,removing a passenger detail. : * **Adding a passenger** : The actor can enter new passenger details. * **Updating a passenger** : The actor can  update existing passenger details as in its email , the number and others if any. * **Removing a passenger** : The actor can remove the passenger details in case they have cancelled the ticket or it is having another issue.   **Basic Flow 1: Add a Passenger**   * The system displays the actors to enter the details of the passenger.   These are as:   * Name * Phone Number * Age * Email Address * Address * Pin-Code   After the details are filled then the system generates a unique id to the passenger.  **Basic Flow 2:**  **Update a Passenger**   * The system requests the actors to enter the valid credentials of the passenger of whom he wishes to update the details. * The actors then enter the passenger credentials. * Then he adds the necessary changes that he wishes to update. * Once the details are updated then the system responds by displaying the changes.   **Basic Flow 3 : Remove a Passenger**   * The system prompts the actors to enter the passenger details. * The actor then enters the login credentials. * Then he deletes the user and confirms the deletion. * And the system notifies the passenger about deletion of the record. |
| **Alternate** **Flow1** : **Invalid Details**   * If the actor fills in the invalid details of the passenger then the state remains the same and the actors have to fill in the details again.   **Alternate Flow 2: Passenger Already Existed**   * If in the add passenger subflow the passenger already has an account then the system terminates by saying that the passenger already existed.   **Alternate Flow 3** : **Passenger Not Found**   * If in the remove passenger subflow is not found then the system terminates.   **Alternate Flow 4** :  **Update Cancelled**   * If in the update subflow the actors wish not to update the details then they can cancel the updates.   **Alternate Flow 5**: **Actor Terminates/Exits**   * This allows the actor to exit anytime during the use case and the state of the system remains unchanged. |
| **Special Requirement** : None. |
| **Associated Use Case** : Login. |

1. **MAINTAIN STAFF DETAILS USE CASE**

|  |
| --- |
| **Introduction** : This use case documents the steps that must be followed to maintain the staff  details. |
| **Actors** : Administrator. |
| **Pre-Condition** : The actor must be logged into the system. |
| **Post-Condition** : The actor must be successfully able to maintain the staff details and the state of the system is changed otherwise it remains the same. |
| **EVENT FLOW** |
| **Basic** **Flow** : The admin are able to login and  maintain the staff details as in :   * Then the actor have the option of adding a staff,updating a staff details,removing a staff detail. : * **Adding staff** : The actor can enter new staff details. * **Updating staff** : The actor can  update existing staff details as in its email , number,staff id and others if any. * **Removing staff** : The actor can remove the staff details in case they have left the system  or they have been removed.   **Basic Flow 1: Add Staff**   * The system displays the actors to enter the details of the staff to be added.   These are as:   * Name * Phone Number * Age * Email Address * Address * Pin-Code * Joining Report * Qualification   After the details are filled then the system generates a unique id to the staff member and provides him with permission to the system to make the allowed changes.  **Basic Flow 2:**  **Update staff**   * The system requests the actor to enter the valid credentials of the staff member of whom he wishes to update the details. * The actor then enters the staff member credentials. * Then he adds the necessary changes that he wishes to update. * Once the details are updated then the system responds by displaying the changes.   **Basic Flow 3 : Remove Staff**   * The system prompts the actors to enter the staff details. * The actor then enters the login credentials. * Then he deletes the staff and confirms the deletion. * And the system notifies the staff about deletion of the record. |
| **Alternate** **Flow1** : **Invalid Details**   * If the actor fills in the invalid details of the staff then the state remains the same and the actors have to fill in the details again.   **Alternate Flow 2** : **Staff Not Found**   * If in the remove staff subflow is not found then the system terminates.   **Alternate Flow 3: Staff Already Existed**   * If in the add staff subflow the staff already has an account then the system terminates by saying that the staff already existed.   **Alternate Flow 4** :  **Update Cancelled**   * If in the update subflow the actors wish not to update the details then they can cancel the updates.   **Alternate Flow 5** : **Actor Terminates/Exits**   * This allows the actor to exit anytime during the use case and the state of the system remains unchanged. |
| **Special Requirement** : None. |
| **Associated Use Case** : Login. |

1. **VIEW FLIGHT DETAILS USE CASE**

|  |
| --- |
| **Introduction** :   This use case documents or includes the steps that must be  followed to view the flight details. |
| **Actors** : Passenger, Administrator, Departmental Staff |
| **Pre-Condition** : The actor must be logged into the system. |
| **Post-Condition** : If the passenger is successful in adding the flight details then the state of the passenger changes and moves on to the next state otherwise the state of the system remains unchanged. |
| **EVENT FLOW** |
| **Basic Flow** :   * The system request the actor to enter his login credentials * The actor enters the flight details. * The passenger selects flight and the seat.   **Basic Flow 1: Enter the login details.**   * The actor enter the login details and then the system requests the actors to choose the option they want to choose.   **Basic Flow 2: Enter the Flight Details.**   * The actor then enters the flight details and the details of the flight is displayed to the actor. |
| **Alternate Flow** :  **Alternate Flow 1** :  **Invalid details**   * If the actor fills in the invalid details then they have to again fill in the details.   **Alternate Flow 2: Invalid Flight Details**   * If the flight details entered by the actor are invalid then the use case terminates by saying invalid flight details.   **Alternate Flow 3** : **Passenger Terminates and Exits**   * The passenger can exit anytime during the use case and the state of the system remains unchanged. |
| **Special Requirement** : None. |
| **Associated Use Case** : Login. |

1. **BOOK A TICKET USE CASE**

|  |
| --- |
| **Introduction** :   This use case documents or includes the steps that must be  followed for booking flight tickets. |
| **Actors** : Passenger. |
| **Pre-Condition** :   * The actor i.e the passenger must have logged into the system * The actor must have viewed flight details . |
| **Post-Condition** : If the passenger is successful in selecting flight details then the state of the passenger changes and moves on to the next state otherwise the state of the system remains unchanged. |
| **EVENT FLOW** |
| **Basic Flow** :   * The passengers choose from flight details generated from previous use case i.e view flight details . * The passenger s chooses a flight from the available list. * The passenger selects a seat and books a ticket.   **Basic Flow 1: Login to the System**   * The passenger logs into the system by entering the valid credentials.   **Basic Flow 2: View Flight Details**   * The passenger then views the flight details and chooses the flight.   **Basic Flow 3: Enter the payment details.**   * After selecting the flight the passenger then fills in his payment details and then confirms the ticket. |
| **Alternate Flow** :  **Alternate Flow 1** :  **Invalid details**   * If the passenger fills in the invalid details as in invalid flight details or the login details then the passenger has to again fill in the details.   **Alternate Flow 2** : **Seat Availability**   * The passenger will have to reselect the seats if no seats are available corresponding to given travel dates .   **Alternate Flow 3: Invalid Payment/Bank Details**   * If the passenger fills in the wrong bank details then the use case terminates by saying invalid bank or payment details.   **Alternate Flow 4** : **Passenger Terminates and Exits**   * The passenger can exit anytime during the use case and the state of the system remains unchanged. |
| **Special Requirement** : None. |
| **Associated Use Case** : View Flight Details,Login,Payment. |

1. **PAYMENT USE CASE**

|  |
| --- |
| **Introduction** :   This use case documents or includes the steps that must be  followed to pay for the flight ticket |
| **Actors** : Passenger. |
| **Pre-Condition** :   * The actor i.e the passenger must be logged into the system * The actor i.e the passenger must have booked tickets . |
| **Post-Condition** : If the passenger is successful in making payment corresponding to his flight, the tickets are generated . |
| **EVENT FLOW** |
| **Basic Flow** :   * The passengers checks his selected flight before making payment and proceeds . * Customer selects mode of transaction and makes payment * On success a confirmation message is generated |
| **Alternate Flow** :  **Alternate Flow 1** :  **Invalid transaction details**   * If the passenger fills in the invalid payment details then the use case terminates and passenger needs to re-initiate the transaction.   **Alternate Flow 2** : **Passenger Terminates and Exits**   * The passenger can exit anytime during the use case and the transaction will be cancelled. |
| **Special Requirement** : None. |
| **Associated Use Case** : Book tickets. |

1. **ADD LUGGAGE DETAILS USE CASE**

|  |
| --- |
| **Introduction** :   This use case documents or includes the steps that must be  followed for adding the luggage details . |
| **Actors** : Passenger. |
| **Pre-Condition** :   * The actor i.e the passenger must be logged into the system, * The actor must have made payment. |
| **Post-Condition** : If the passenger is successful in adding the luggage details then the state of the passenger changes and moves on to the next state otherwise the state of the system remains unchanged. |
| **EVENT FLOW** |
| **Basic Flow** :  The passengers adds no. of items and their basic description such as colour, type, size etc. |
| **Alternate Flow** :  **Alternate Flow 1** :  **Invalid details**   * If the passenger fills in the invalid details as entering 2 luggage items and entering details of 3 items then the passenger has to again fill in the details.   **Alternate Flow 2** : **Passenger Terminates and Exits**   * The passenger can exit anytime during the use case and the state of the system remains unchanged. |
| **Special Requirement** : None. |
| **Associated Use Case** : Payment. |

1. **ADD PASSPORT DETAILS USE CASE**

|  |
| --- |
| **Introduction** :   This use case documents or includes the steps that must be  followed by adding the passport details. |
| **Actors** : Passenger. |
| **Pre-Condition** :   * The actor i.e the passenger must be logged into the system, * The actor must have made payment. |
| **Post-Condition** : If the passenger is successful in adding the passport details then the state of the passenger changes and moves on to the next state otherwise the state of the system remains unchanged. |
| **EVENT FLOW** |
| **Basic Flow** :   * The passengers logs into the system and chooses the add passport option * The passengers add their passport details into the system. |
| **Alternate Flow** :  **Alternate Flow 1** :  **Invalid details**   * If the passenger fills in the invalid details as in invalid login id,name or any other passport details then the passenger has to again fill in the details.   **Alternate Flow 2** : **Passenger Terminates and Exits**   * The passenger can exit anytime during the use case and the state of the system remains unchanged. |
| **Special Requirement** : None. |
| **Associated Use Case** : Payment,Book Ticket. |

1. **CANCEL FLIGHT TICKET USE CASE**

|  |
| --- |
| **Introduction:** This use case includes the steps that must be followed to cancel the ticket. |
| **Actors** : Administrator, Passenger. |
| **Pre-Condition** :   * The actors must be logged into the system. * The passenger should have made payment. |
| **Post-Condition** : The actors are able to cancel the flight ticket and the state of their travelling changes and if they are not able to cancel the ticket then the state remains unchanged. |
| **EVENT FLOW** |
| **Basic Flow** :   * The passenger fills in their details and then various options are displayed; they then choose the cancel ticket option and apply for the refund. * If the admin is cancelling the passenger ticket he then enters their details and cancels the ticket providing the passenger with an update along with a reason. |
| **Alternate Flow 1** :  **Invalid Details** :   * If the actors fill in the invalid details then they have to re-enter their details.   **Alternate Flow 2**  : **Actor Terminates/Exits** :   * This allows the actors to exit the system at any state and the state of their ticket remains unchanged. |
| **Special Requirement** : None. |
| **Associated Use Case** : Payment. |

1. **REFUND USE CASE**

|  |
| --- |
| **Introduction** :   This use case documents or includes the steps that must be  followed to refund the passenger his amount. |
| **Actors** : Administrator |
| **Pre-Condition** :   * The actor i.e the administrator must be logged into the system * The passenger has cancelled the ticket . |
| **Post-Condition** : If the passenger had made payment and cancelled the flight ticket then his amount is refunded . |
| **EVENT FLOW** |
| **Basic Flow** :   * The administrator verifies passenger details and flight details from the cancel ticket and initiates refund . * The amount is refunded after deducting penalty charges |
| **Alternate Flow** :  **Alternate Flow 1** :  **Invalid details**   * If the passenger filled in the invalid details i.e payment not made or invalid flight details then the refund is cancelled .   **Alternate Flow 2** : **Passenger Terminates and Exits**   * The passenger can exit anytime during the use case and the state of the system remains unchanged. |
| **Special Requirement** : None. |
| **Associated Use Case** : Cancel ticket. |