# Planning

# Functional Requirement Analysis

## Actor identification

There are mainly following actors involved in the scenario:

Human actors:

* Passengers
* Staff
* Management staff
* System admin
* Team software engineers

Non-human actors:

* Plane
* System database

## User stories

* The system admin should be able to create new flights that could be viewed and searched by the passengers;
* The system admin should be able to create new routes to the same destination so that passengers can plan their trip more efficiently;
* The system admin should be able to temporarily assign a flight to a new destination to accommodate for random specific cases;
* The staff should be able to handle passenger payment at front-desk;
* The management staff should always be able to view the most up-to-date schedule for all flights;
* The system admin should be able to delete an existing flight;
* The system admin should be able to read all transaction records no matter successful or failed, with various payment options enabled, and the admin should be able to generate and retrieve all the payment records;
* The passenger should be able to create an account in the booking system;
* The passenger can search for flights by filtering on destination, route and leaving dates through the system;
* The passenger can book a flight through the system by using various payment methods;
* The passenger can check the updated the flight schedule that he/she booked;
* The system database should assign each flight a unique ID, so that flights of multiple planes (by same/different airline) leaving/arriving at the same airport for same destination will be stored as different records in the database;
* The system software engineer should be able to adapt to new changes/situations of the airport flights quickly to better cater for the flight service.

## Use cases

***Use case 1: Admin login***

Goal: To enable the admin to log in to the system

Primary actor: the system admin

Secondary actor: database

Trigger: The system admin entered credential and clicked “log in” button

Precondition: The system admin account has been created and is not logged in yet

Flow of event:

1. The admin selects “to login”;

3. The admin enters credential into the login form;

4. The system will verify the credential to the record stored in database;

5. If the system admin enters the wrong credential the use case will redirect to extension 1E. Otherwise, the flow is continued.

6. Use case end.

Extensions:

1E. Admin login failed:

a. The system prompts that a system login is failed;

b. Choose “Login again” or enter the credential again. If selecting “Login again”, continue use case 2;

c. The system asks for credentials again;

d. The admin can proceed by following step 2 of FOE. Otherwise, the use case ends.

***Use case 2: Passenger login***

Goal: To enable the passenger users to log in to the system

Primary actor: the passenger member

Secondary actor: database

Trigger: The passenger member entered credential and clicked “log in” button

Precondition: The passenger member has already registered in the system and is not logged in yet

Flow of event:

1. The passenger member selects “to login”;

3. The passenger member enters credential into the login form;

4. The system will verify the credential to the record stored in database;

5. If the passenger member enters the wrong credential the use case will redirect to extension 2E. Otherwise, the flow is continued.

6. Use case end.

Extensions:

2E. Passenger member login failed:

a. The system prompts that a member login is failed;

b. Choose “Login again” or enter the credential again. If selecting “Login again”, continue use case 2;

c. The system asks for credentials again;

d. The passenger member can proceed by following step 2 of FOE. Otherwise, the use case ends.

***Use case 3: Manage flights***

Goal: To add new flights, delete or edit existing flights;

Primary actor: the system admin

Secondary actor: Database

Trigger: the system admin select a flight and choose an action and then click “submit”

Precondition: log in successfully to the system.

Flow of event:

1. The system admin is presented with a list of flights, and he selects one from the list;
2. The system admin then specifies the action: add, edit or delete the flight from the list.

A. Add a new flight:

1.1. The system admin is presented with a form about the new flight information.

1.2. The Administrator types in the flight’s information such as destination, seat capacity, plane, departure time, etc.

1.3. Click “Submit” button

1.4. If the entered flight data is correct, the flow continues at 1.5. Otherwise, the use case will enter extension 1A

1.5. The new flight information is saved into the database

B. Edit existing flight’s data:

1.1. The information about the selected flight is presented in the form to the system admin

1.2. The administrator scans through the data entries and change the entries that need to be modified.

1.3. Click “Submit” button

1.4. If the edited flight data is correct, the flow continues at 1.5. Otherwise, the use case will enter extension 1A

1.5. The updated flight information is saved into the database

C. Delete an existing flight:

1.1. The system admin chooses the flights he wants to delete.

1.2. The system admin clicks “delete”

1.3. The system will prompt the admin for confirmation before deleting. If the administrator agrees to proceed, the use case will continue at step 1.4. Otherwise, the use case will redirect to step 1.5.

1.4. Prompting the admin that the flight has been deleted.

1.5. The interface returns to the flight list.

1. The use case end.

Extension:

1A. The data entered about a flight is incorrect:

1. The system will prompt and alert that the entered new data is incorrect with details.

2. The system admin needs to re-enter the information;

3. The use case resumes at 1.4.

***Use case 4: Check flight schedule***

Goal: To plan a journey.

Primary actor: Passenger member

Secondary actor: Database

Trigger: The passenger clicks on “view flight schedule” button

Precondition: The passenger has successfully logged into the system

Flow of event:

1. Passenger member enters the credentials to login

2. Passenger specify a date range, and click “view schedule” button, all the flights that have been booked by the passenger will be listed chronologically;

3. Passenger member can select each of the upcoming flight to view the updated information such as seat availability;

4. Passenger can proceed with other features such as seat reservation;

Extension:

1A. No booked flights for the passenger:

I. The system will prompt the user that “there is no booked flight yet”.

II. The system will redirect the passenger to the booking page.

III. The passenger can choose to book a flight.

IV. The use case end.

1B. The booked flight for the passenger had been delayed or cancelled.

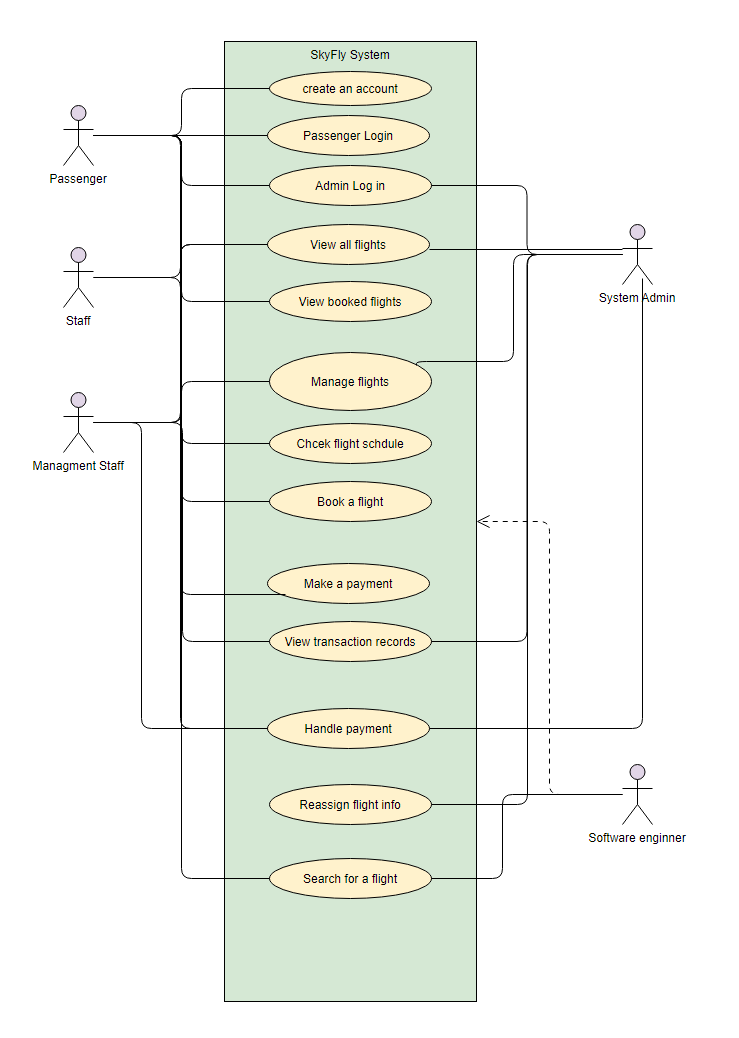
I. The system will prompt the passenger about the incident and apologize;

II. Passenger will need to confirm the message;

III. Passenger will be redirected to the booking page with recommendation on the same departing data and destination;

IV. The use case resumes at step 2.

## Use case diagram



# Non-Functional Requirement Analysis

## Usability requirement

* The system should be able to display all the flight information, preferably with the pagination enabled so passengers can better navigate;
* The system should provide a complex and complete filter for user to narrow down the intended flight;
* The system must be able to allow users to redo/undo their previous actions;
* The system must be covering all the major airports/flight company and plane models worldwide;
* The system must be able to generate a schedule or flight report to the user, in the form of either PDF or excel spreedsheets.

## Performance requirement

## Reliability requirement