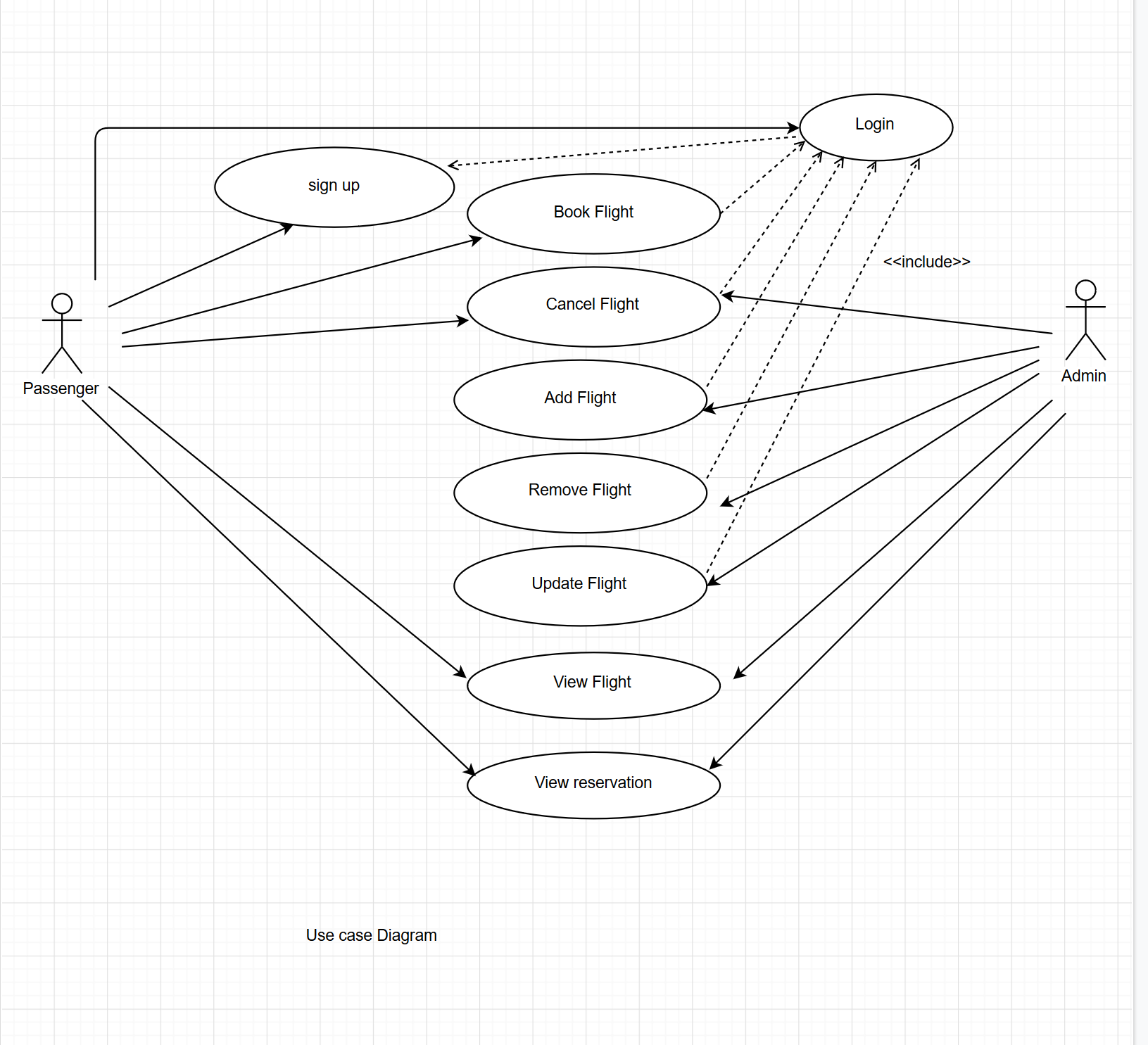
**Airplane online booking System**

***Software requirement specification (SRS)***

**1.**Group Github repository

<https://github.com/zelalemzergaw/Software-Enginering-2019>

**2**.Use case diagram



**2.Airline Flight Booking System: Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case Number: 1** | | | |
| **Name** | | **Book Flight** | |
| **Brief description** | | This use case helps the user(passenger) to book a flight ticket | |
| **Actors** | | Passenger(user) | |
|  | | | |
| **Preconditions:** The user not necessarily to login to book flight. | | | |
| **Flows of events** | | | |
| 1. **Basic Flows** | | | |
| **Step** | **User Actions** | | **System Actions** |
| 1 | The user(passenger) search a flight in searching box. | | The system retrieves that information from the database and returns to the user(passenger). |
| 2 | The user(passenger) fills out his information to book a flight ticket. | | The system registers his information to data base. |
| **Postconditions:** The passenger gets his confirmation number | | | |
| **Business Rules:** passenger must have valid payment method (debit card or credit card) | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case Number: 2** | | | | |  | |
| **Name** | | | View Flight | |  | |
|  | |  | | |  | |
| **Brief description this** use case helps users watch their flight bookings | | | | | | |
|  | |  |  | |  | |
| **Actors** | | | Passenger | |  | |
|  | |  | | |  | |
|  | |  |  | |  | |
| **Preconditions** | | | | |  | |
| The user should input the booking Id to view the booking. | | | | |  | |
|  | |  | | |  | |
| **Flows of Events:** | | | | |  | |
| **1. Basic Flows** | | | | |  | |
|  | |  | | |  | |
|  |  | | |  | |
| **Step** | |  | **User Actions** | | **System Actions** | |
|  | |  |  | |  | |
| 1 | |  | The passenger Enters the booking Id into the search box. | | The system searches the database for that booking Id then returns a result with all the booking information along with the result. | |
|  | |  |  | |  | |
|  | |  |  | |  | |
| 2 | |  | The passenger views the information of that flight/ booking. | | The system presents the detailed booking information to the passenger with the associated booking Id like the flight number, date of departure and arrival, airport, airline, booking id and etc. | |
|  | |  |  | |  | |
|  | |  |  | |  | |
|  | |  |  | |  | |
|  | |  |  | |  | |
|  | |  | | |  | |
| **Post conditions** | | | | |  | |
| **Business Rule** | | | | |  | |
| Every booking Id is unique | | | | |  | |
|  | |  | | |  | |

**Airline Flight Booking System: Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case Number: 1** | | | |
| **Name** | | Add Flight | |
| **Brief description** | | This use case allows the admin to Add flight in the system | |
| **Actors** | | Admin | |
|  | | | |
| **Preconditions: admin must have logged in to the system** | | | |
|  | | | |
| **Flows of events** | | | |
| 1. **Basic Flows** | | | |
| **1.1.0 Add Flight** | | | |
| Step | User Actions | | System Actions |
| 1 | The admin calls the add flight command. | | The system displays the add flight form with the field flight name. |
| 2 | The Admin fills out the form and requests the system to save the details | | The system verifies that there’s no other add flight in the database with the same name and saves the flight and returns the success message on success, generate an ID automatically to the flight or a fail message in case of failure. In case another airline exists with the name, the system returns the message indicating a duplicate entry exists. |
| **Postconditions:** The add flight is persisted in the system | | | |
|  | | | |
| **Business Rules:** No duplicate flight. A unique id is identified each flight. | | | |
|  | | | |
| **1.1.1 Update flight** | | | |
| **step** | **User Actions** | | **System Action** |
| 1 | The admin selects to view a list of flight | | The system returns a list of all flights. |
| 2 | The admin selects the flight he wants to update | | The system displays an editable flight form pre-populated with the flight details |
| 3 | The admin updates the name of the flight they want to update and requests system to save the new name. | | The system updates the record and returns the success message or a fail message on exception |
|  | | | |
| **Postconditions:** The name of the flight will be updated. | | | |
|  | | | |
| **1.1.3 Delete Flight** | | | |
| **step** | **User Actions** | | **System Action** |
| 1 | The admin selects to view a list of flights | | The system returns a list of all flights |
| 2 | The admin selects the flight he wants to delete | | The system displays a confirmation dialogue window |
| 3 | The admin selects OK on the confirmation dialog window to confirm deleting the profile | | The system deletes the flight and returns message a success confirms. |
| **Postconditions:** An flight will be deleted | | | |

1. **Supplementary specification**

A supplementary specification captures the system requirements that are not readily captured in the use cases of the use-case model. It gives a general additional idea about the nonfunctional requirements and features such as Usability Reliability performance and all other features.

**Usability**

The online flight booking is a user-friendly system. In order to use the system, the user no need to train the software can guides him what he need to do. Our web app give for our passenger can give him better information.so the user can easily understand the app easily.

**Reliability**

Our system is consistent, could function very well and trusted.

**Performance**

Our system is compatible with any operating system. The tasks performed real fast, like less than 3 seconds to execute. It allows to multiple users to access with out being the system down.

**5.Glossarly**

It is problem domain term

**Booking a flight** – is the order of reservation of seat for flight.

**Flight** – travelling using airplanes

**Departure time**- it is the time at which flight starts or when the plane start leaving

**Boarding time-**the time at which check in starts

**Arrival time** - it is the time at which the lane start landing to its destination