

CS-319 Object Oriented Software Engineering

Analysis Report

Airplane Reservation System

Fall 2015

**Group 6 - Sec 03**

Burak Sefa SERT

Sanem ELBASI

Can Mert YILDIZ

**Table of Contents**

**1. Introduction 3   
2. Overview 3  
3. Requirement Analysis 3  
 3.2. Functional Requirements 3  
 3.3. Non-Functional Requirements 5  
 3.4. Constraints 5  
 3.5. Scenarios 5  
 3.6. Use Case Models 7  
 3.7. User Interface 13  
4. Analysis 31  
 4.1. Object Model 31  
 4.1.1. Domain Lexicon 31  
 4.1.2. Class Diagram(s) 31  
 4.2. Dynamic Models 35  
 4.2.1. State Chart 35  
 4.2.2. Sequence Diagram 36  
5. Conclusion 38**

**1. Introduction**

As our project, we decided to design and implement an airplane reservation system, in which user -on behalf of a customer- can reserve a place from among available places of a plane with a given date, departure and destination location.

This report describes the functional, non-functional and pseudo requirements and it also contains use-cases, scenarios and UML diagrams. This reports also give an intuition about how the program will work and what features it has.

**2. Overview**

First of all, to be clearer, let us define our target end user profile. There will be two different type of end-users who differ on their privileges. One of them, “User”, is simply a clerk in office of airline company and his main task is regulating (i.e. making, cancelling and querying) reservations for customers. On the other hand, “Admin” is the other person whose range of authority is greater and his main task is to decide who will have access to the system. List of Admin’s ability is the following:

Adding and deleting a plane from the system

Adding and deleting a user from the system

Changing schedule of a plane

In order the store all information (reservations, user information etc.), we will be using an online database so that changes will be visible to all users who uses application from different computers. Database system will be MySQL database.

**3.2. Functional Requirements**

### **3.2.1 Admin**

**Login:**

* Admin will be able to login the system using his/her employee id and password.
* If the id or password is entered incorrectly, the system shall be able to give an error message specifying either id or password is incorrect.

**Display Information about Plane and Seats:**

* After the login, admin will have an option to see existing planes, their times and also allocated seats in them.

**Add Plane or Change Schedule:**

* Admin will be able to change the schedule by giving an existing plane to another hour in the schedule if it is empty.
* Admin will be able to add/delete destination and departure cities.
* Admin will be able to add a new plane to the system by indicating the type of the plane and the number of the seats in the plane.

**Add/Drop User:**

* Admin will be able to add new employee to the system with employee name and email address.
* The system will automatically generate an employee id for the new user and will display the new id to the admin.
* Admin will be able to delete a user.

**Adjust Ticket Prices**

* If it is necessary, admin will be able to change ticket prices depending on the airline company.

### **3.2.2 User**

**Login:**

* User, after registered to the system, can login to the system using employee id and password in order to make reservation.
* If the id or password is entered incorrectly, the system shall be able to give an error message specifying either id or password is incorrect.

**Set Account Details**

* User, after registered to the system, able to change his/her account details including mail address, name and password.

**Select Departure/Destination Place:**

* User will be able to select passenger’s destination and departure place of plane.
* System shall be able to show specific place list consisting of departure and destinations places which user will be able to select from.
* System will not list the departure places which don’t have available flights.

**Select Date:**

* User will be able to select the departure date of the flight.
* User will be able to select both departure and return date if the round trip option is checked.
* System shall be able to show flight list on the specified date with specified departure and destination places.
* System will not list the dates which don’t have available flights.

**Select Time:**

* User will be able to select a time interval to narrow the options on the flight list of the given date.

**Select Seat:**

* After specifying time and place, user will see the available seats in that plane.
* User shall be able to select any empty seat.
* After selection, this seat will not be empty anymore.

**Show Reservations / Cancel Reservation:**

* User will be able to make a reservation with customers’ name, surname, social security number and email address.
* User will be able to see any reservation with social security number of the customer.
* User will be able to cancel any reservation on the reservation list.

**3.3 Non-functional Requirements**

**3.3.1 Well Designed Interface**

We want to make the GUI of the program both appealing and user friendly so that neither user nor admin have difficulty to use the program.

**3.2.2 Extendibility**

We aim to design the program in such a way that implementing future changes will not result in the redesigning the whole program.

**3.2.3 Cost-Efficiency**

It is also one of our concern that making the program cost effective. Our purpose is keeping the cost of the program as low as possible.

**3.4. Constraints**

* The program will be implemented in java
* Language of the system will be English
* MySQL Database will be used to store data
* The program will be a desktop application

**3.4 Scenarios**

Scenario #1: Making Reservation

* Customer, John, goes to the ticket office in order to make a reservation.
* Office clerk, Mary, asks John date, destination and departure point.
* John gives her the necessary information.
* Mary, searches for available trips that meet John criterions, informs him about available the schedule.
* John opts for a departure time.
* According to John’s selection, Mary shows John available seats.
* John selects a seat from amongst available seats

Scenario #2: Cancelling/Changing Reservation

* Customer, John, goes to ticket office in order to change his reservation.
* Office clerk,Mary, asks John date, destination point, departure point and seat number of the reservation that he wants to change.
* John gives her the necessary information.
* Mary cancels the current reservation of John by using her user privileges.

--------- (Steps above is enough for cancelling the reservation.)-----------

* Then, Mary asks John the date, destination point and departure point of the new reservation.
* John provides the information that Mary asks for.
* Mary, searches for available trips and informs him about available the schedule.
* John opts for a departure time.
* According to John’s selection, Mary shows John available seats.
* John selects a seat from among available seats.

Scenario #3: Deleting/Adding User To The System

* The old office clerk, Mary, quits the job.
* The chief, also the admin, Jasmine deletes Mary from the system by specifying her employee number in order to prevent any unwanted situation.
* The new office clerk, Anna, replaces Mary’s position.
* The admin adds Anna to the users by entering her employee number and password that Anna decides to the system.
* Now Anna is capable of making reservation.

Scenario #4: Deleting Plane from The System

* One of the plane of the airline company becomes unusable due to a malfunction of the radar system of the plane.
* Company decides to remove this plane from the fleet.
* The company director informs admin about the change.
* Admin deletes the plane from the system.

Scenario #5: Adding Plane To The System

* In order to compensate the plane that is removed from the fleet, company buys a new plane.
* After the testing process, plane is ready for the work.
* Company director informs admin about the change.
* Admin adds the new plane to the system.

**3.5 Use Case Models**

Remainder: In those use cases “user” represents the office clerk who uses the program and make reservations according to customer wish.

**Use Case Diagram:**



Figure 3.5.1 Use Case Diagram

**Use Case #1**

**Use Case Name:**Login

**Actor**: Admin or User(Both are eligible)

**Pre-Condition**: Admin or user has opened the program

**Post-Condition**: Admin or user logged in

**Main Flow of Events:**

User or admin enters his/her employee id and password on text fields

User or admin hits the login button

User or admin logs in successfully if his/her information is correct.

Otherwise, the program throws a proper message.

**Use Case #2**

**Use Case Name:** MakeReservation

**Actor**: User

**Pre-Condition:** User has opened the programand logged in

**Post-Condition:** User has made reservation OR

User has been informed about why he is unable to make reservation

**Main Flow Of Events:**

User selects time, date, departure and destination points according to customer’s wish.

The program displays planes and seats which are suitable.

User informs customer about available seats.

Customer makes a selection from among those seats .

User completes reservation process by entering necessary information of the customer to the system.

**Alternative Flow of Events:**

Customer is informed that there is no available seats for given time,date,departure and destination points. As a result either

-Customer stops interaction OR,

-Customer requests reservation for different time.

**Use Case #3**

**Use Case Name:** Cancel Reservation

**Actor:**User

**Pre-condition:** User has opened program

The reservation, that is going to be cancelled, exists.

**Post-condition**: User cancels reservation OR

The program informs user about why he cannot cancel the reservation.

**Main Flow Of Events:**

User enters his employee id and password.

User selects the reservation that customer wants to cancel and sends cancellation request by specifying date, destination point, departure point and the seat number.

Admin confirms the changes.

The system returns success message if the cancellation request is made 24+ hours before the departure time of plane OR,

The system returns error message that it is too late to make that transaction.

**Use Case #4**

**Use Case Name:** ViewFlightInformation

**Actor**: Admin or User(Both are eligible)

**Pre-condition**: Admin or user has opened program and logged in.

**Post-condition**: Admin or user gets the information

**Main Flow Of Events:**

Admin or user hits the ViewFlightInformation button on the menu.

The program displays a flight whose ID is specified by admin or user (i.e. departure time and location, destination location, total number of seats and reserved number of seats)

**Use Case #5**

**Use Case Name:** AddPlane

**Actor:** Admin

**Pre-Condition**: Admin has opened the program and logged in.

**Post-Condition**: Admin add a plane to the system or,

the program informs admin about why he cannot do so.

**Main Flow Of Events:**

Admin goes add a new plane section which is located under Plane Menu.

Admin enters the plane type, number of seats and the planes unique id in add plane section.

The program returns error message if the plane already exists.

Otherwise the program returns success message which indicates the planes has been added.

**Use Case #6**

**Use Case Name:** DeletePlane

**Actor:** Admin

**Pre-Condition**: Admin has opened the program and logged in.

**Post-Condition**: Admin delete a plane to the system or,

the program informs admin about why he cannot do so.

**Main Flow Of Events:**

Admin goes delete new plane section which is located under Plane Menu.

Admin enters the unique id in delete plane section.

The program returns error message if the plane does not exists.

Otherwise the program returns success message which indicates the planes has been deleted.

**Use Case #7**

**Use Case Name**: CreateUser

**Actor**: Admin

**Pre-Condition**: Admin has opened the program and logged in

**Post-Condition**: Admin adds a new user who can make reservations OR,

he gets a message why he cannot add the user.

**Main Flow Of Events:**

Admin goes to add a new user section which is located under User Menu.

Admin enters employee id and password of the new user.

The program returns error message if the user already exists.

Otherwise the program returns success message which indicates that the new user has been added.

**Use Case #8**

**Use Case Name**: DeleteUser

**Actor**: Admin

**Pre-Condition**: Admin has opened the program and logged in

**Post-Condition**: Admin deletes a specific user OR,

he gets a message why he cannot delete the user.

**Main Flow Of Events:**

Admin goes to delete a user section which is located under User Menu.

Admin enters employee id the new user.

The program returns error message if the user does not exists.

Otherwise the program returns success message which indicates that the new user has been deleted.

**Use Case #9**

**Use Case Name:** ChangeSchedule

**Actor**: Admin

**Pre-Condition**: Admin has opened the program and logged in

**Post-Condition**: Admin changes schedule of the plane OR,

he gets a message why he cannot change the schedule

**Main Flow of Events:**

Admin goes to ChangeFlight section which is located under Flight Menu.

Admin enters the Id of the plane which he wants to change.

Admin enters the new schedule for that plane.(i.e. new departure time, destination and departure points)

Admin confirms the changes.

The program shows a success message if the plane exist or throws an error message if it does not.

**Use Case #10**

**Use Case Name:** Update Personal Info

**Actor**: Admin or User(Both are eligible)

**Pre-Condition**: Admin or User has opened the program and logged in

**Post-Condition**: Admin or User changes his/her personal information such as e-mail password etc.

**Main Flow of Events:**

Admin or user goes to account settings menu.

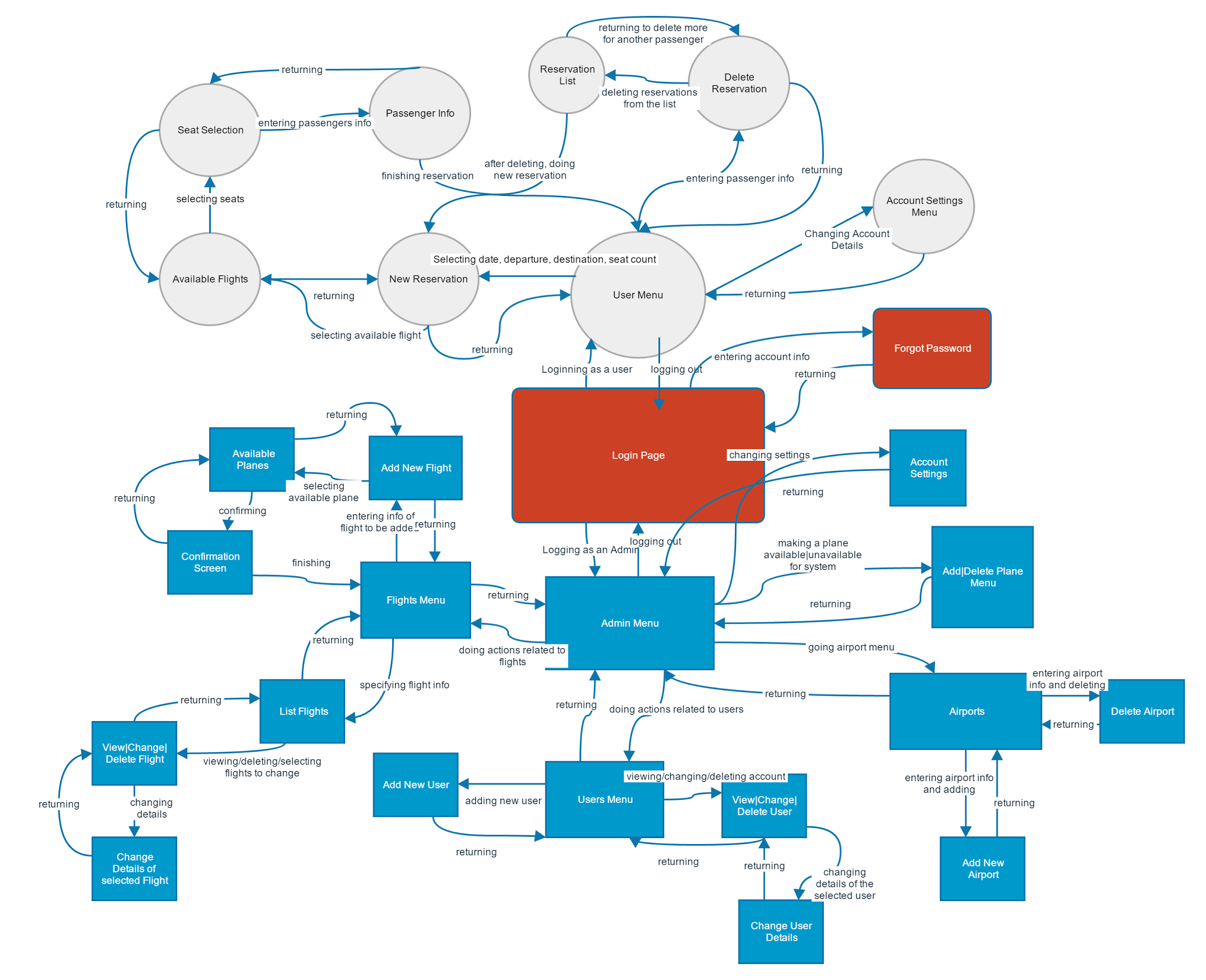
Admin or user changes personal information which he wants to change.

Admin or user confirms and saves changes.

The program shows success message.

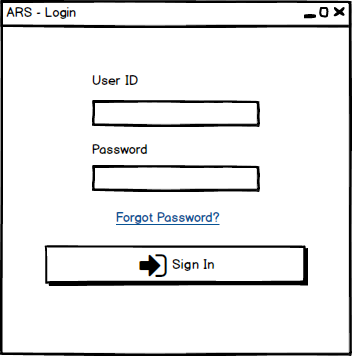
**3.7. User Interface - Navigational Paths and Screen Mock-ups**

**Navigational Path**



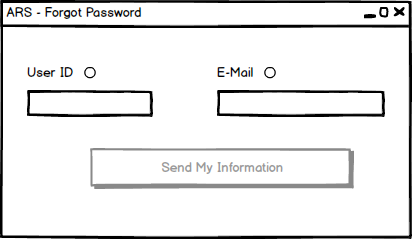
**Figure 3.7.1**

**Screen Mock-ups**



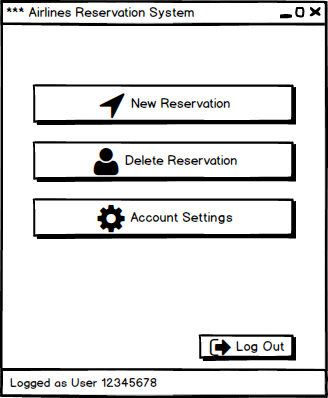
**Figure 3.7.2 - Login Page**

**Login page** is the first screen that will be shown. User must enter his/her ID and password which obtained from administrator in order to sign in. In case of entering wrong information, a warning window will appear.



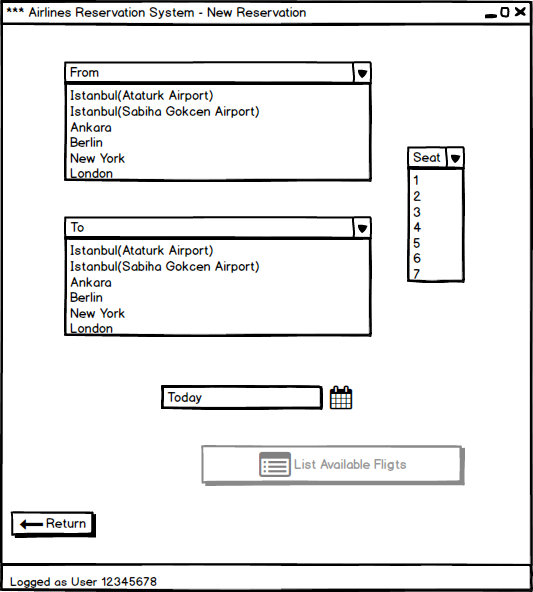
**Figure 3.7.3 - Forgot Password**

**Forgot Password:** For the users who forgot his/her ID or password, it is possible to get missing information by entering ID or E-mail. The system sends the missing information to mail address of the user.



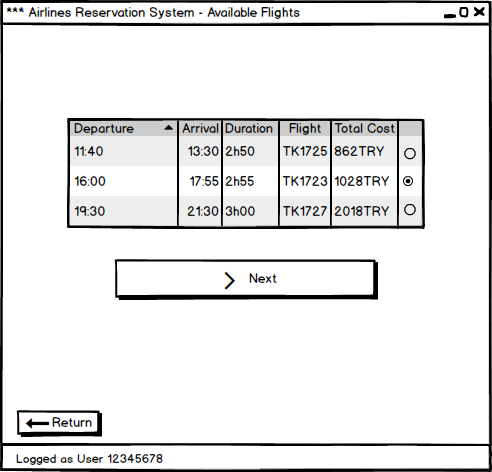
**Figure 3.7.3 - User Main Menu**

**User Main Menu:** After logging in, both users and admins have a simple main menu. At the left bottom corner of the all windows, it is possible to see the ID and login type. Possible operations of the user is grouped under these 3 buttons and after every 3 button, different operations start systematically branching. The user have to perform all operations related to passenger and these details keep the reservation system simple and fast.



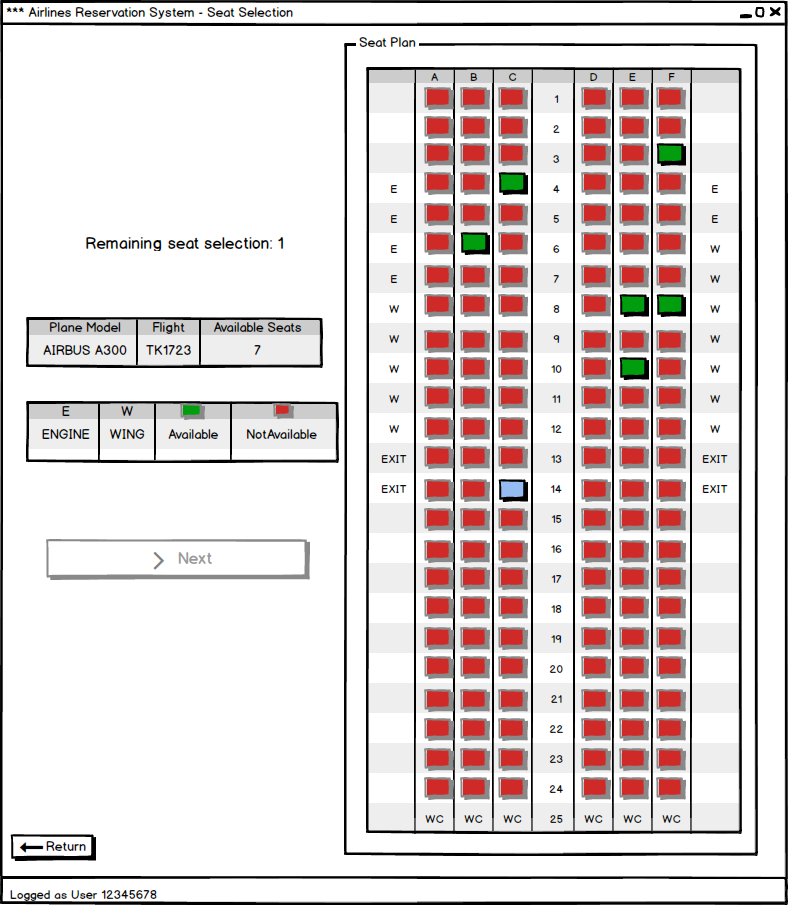
**Figure 3.7.4 - New Reservation Menu**

**New Reservation Menu:** User enters the information related to reservation depending on the expectations of the passenger. New reservation menu lists available airports and seat counts with combo boxes. This menu also has a calendar interface to select the date for flight. After main menu, all of the windows have a return button. It enables user to return previous menu.



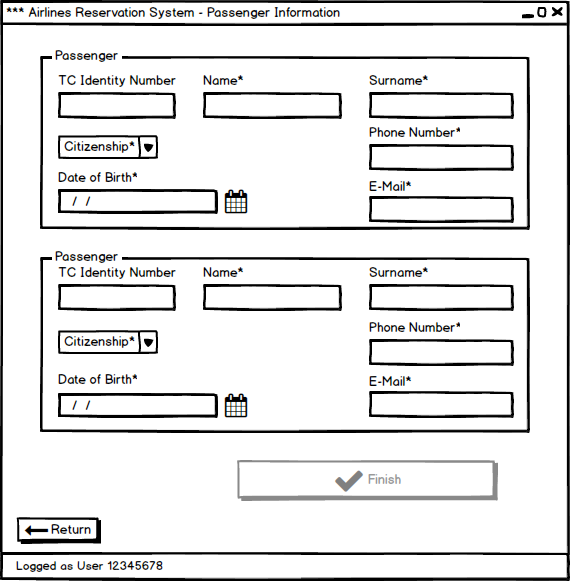
**Figure 3.7.5 - Available Flight List**

**Available flights:** After specifying information of the flight, available flights appear in this windows. We assume that from this point customer (passenger) is able to see the screen of the user to select a proper flight. User is able to select flight with radio buttons. After selection, status of the next button becomes active and status of the radio buttons of the other flight options become inactive.



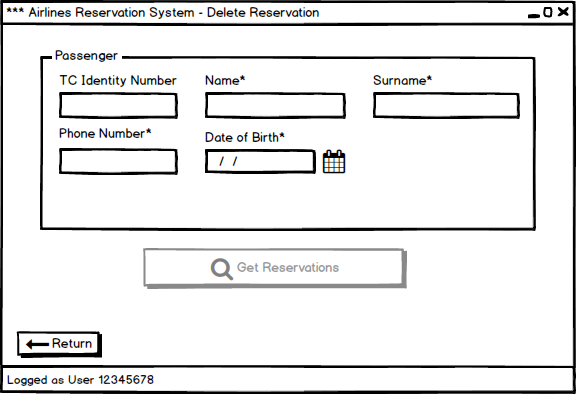
**Figure 3.7.6 - Seat Selection**

**Seat selection:** In this menu, the seat plan of the current flight appears. Tables give information related to icons and other symbols. Until remaining seat selection becomes 0, the next button does not become active. User is able to select seats by clicking the green boxes. After clicking, processed seats become unavailable for other users. This is the last menu which customer can see.



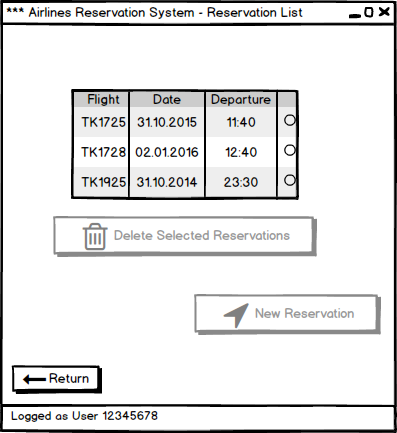
**Figure 3.7.7 - Passengers Information for Reservation**

**Passengers Information:** Depending on the passenger count, input text fields with enough number appear. The fields with “\*” character must be filled by the user. Otherwise, the finish button does not become active. “TC identity number” is an optional field for Turkish Republic citizens. The name, surname, phone number and date of birth data stored by the application make the passengers unique. In case of any wrong input, system shows an error pop-up.



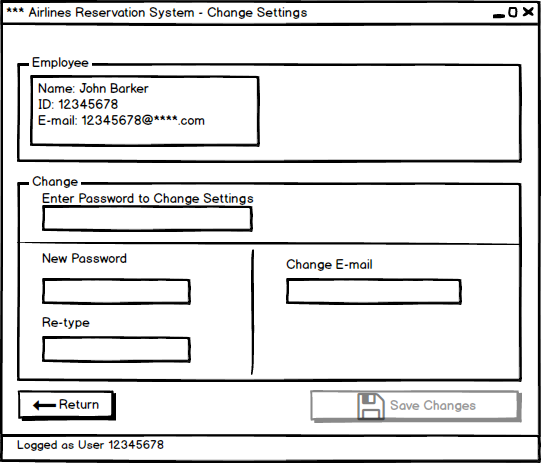
**Figure 3.7.8 - Delete Reservation**

**Delete Reservation:** User is able to delete reservation(s) of the passenger by entering the unique information of the passenger.



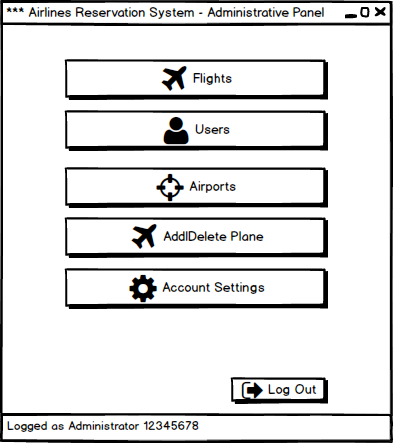
**Figure 3.7.9 - Reservation List of the Current Passenger**

**Reservation List:** Depending on the request of the passenger, user selects the flights which will be deleted by clicking radio buttons. After at least one selection, delete button becomes active. Instead of creating “change reservation” menu, we combined “change” operation with “delete” operation. After delete operation, user has a direct access to new reservation menu.

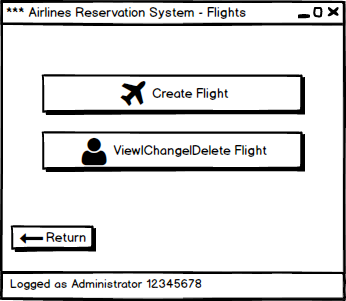


**Figure 3.7.10 - Account Settings for User**

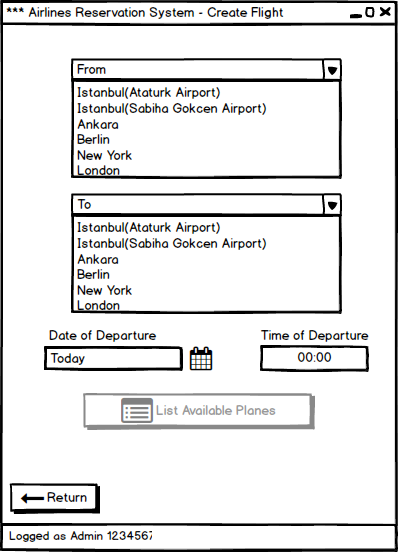
**Account Settings for User:** The user is able to change her/his password or e-mail. Since the IDs are generated from the system, it is not possible to change it. Both password change and mail change, the system asks user for his/her password considering security and privacy issue.



**Figure 3.7.11 - Administrator Main Menu**

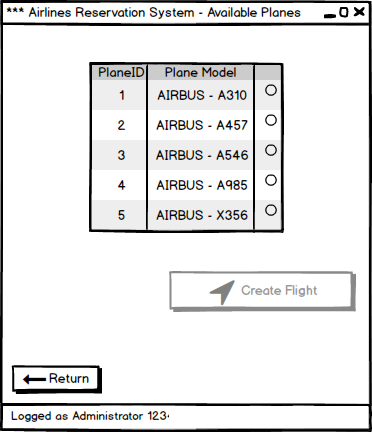


**Figure 3.7.12 - Flight Operations Menu**



**Figure 3.7.13 - Create Flight Menu**

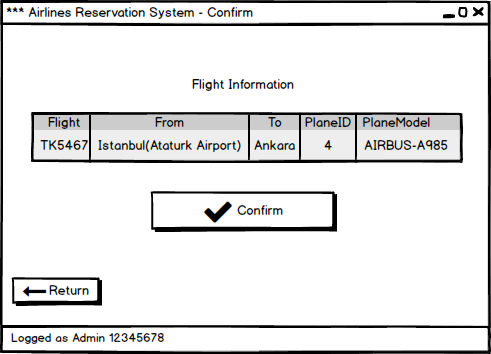
**Create Flight Menu:** Administrator is able to create new flights depending on the requests of the airline company. S/he enters the necessary information related to flight.



**Figure 3.7.14 - Available Plane List**

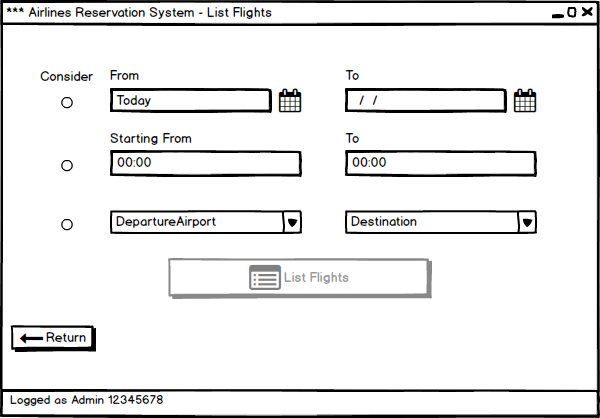
**Available plane list:** After specifiying flight information, available plane list appears.

Administrator is able to select one of them by clicking radio button. After this, other radio buttons become inactive and create flight button becomes active.



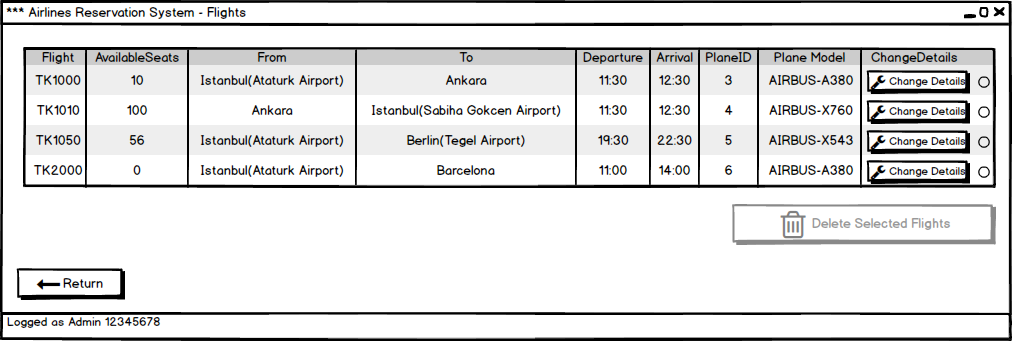
**Figure 3.7.15 - Confirmation of the Flight to be created**

**Confirmation:** Since creating flight is a serious operation, system shows again the details of the flight together which will be created and waits for confirmation from administrator. After confirmation the application returns main administrator menu.



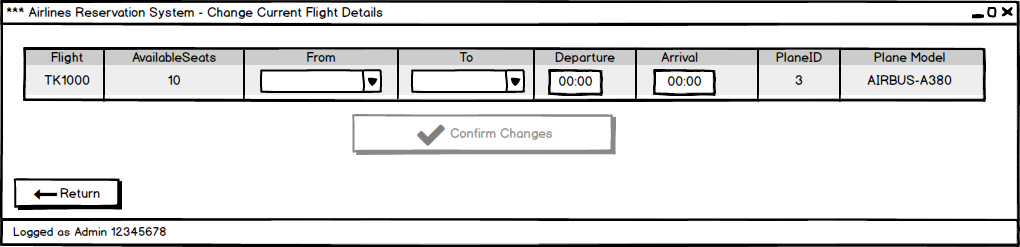
**Figure 3.7.16 - List Flights depending on specifications**

**List Flights:** Before the operations related to existing flights, the administrator specifies the flights which will be listed. S/he is able to add search options by clicking the radio buttons.



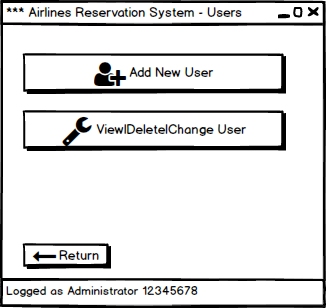
**Figure 3.7.17 - Flight List**

**Flight List:** The administrator is able to delete selected flights. Radio buttons let the administrator select more than one flight. After at least selection, delete button becomes active. It is also possible changing details of a flight by clicking changing details button.

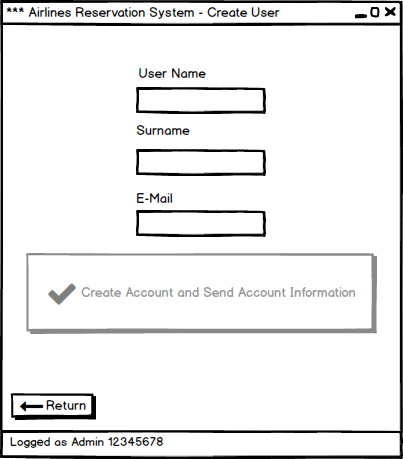


**Figure 3.7.18 - Change Current Flight Details**

**Change Current Flight Details:** The administrator is able to change destination and departure airports and deadlines. This operations results with cancelling all reservations of the current flight.

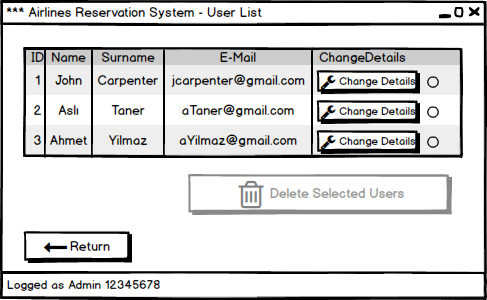


**Figure 3.7.19 - User Operations Menu**



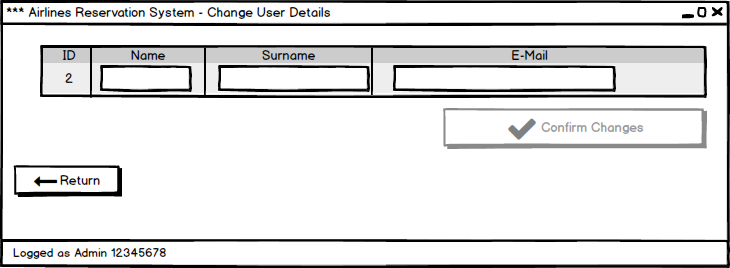
**Figure 3.7.20 - Create User Menu**

**Create User Menu:** Only the administrator is able to add a new user. After filling the text fields and pressing create button, the user receive his/her account information.



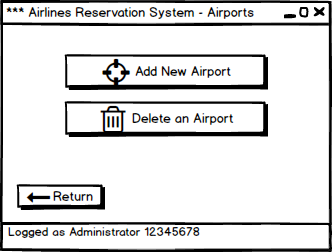
**Figure 3.7.21 - User List to Change/Delete Operations**

**User List:** Only administrator can delete the users. He or she is able to select more than one user by using radio buttons.

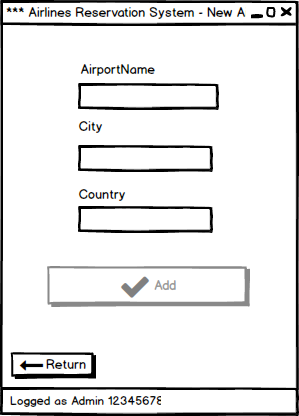


**Figure 3.7.22 - Change User Details Menu**

**Change User Details:** Administrator is able to change information of the users by filling related text fields.

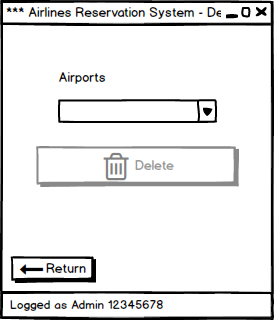


**Figure 3.7.23 - Airport Operations Menu**



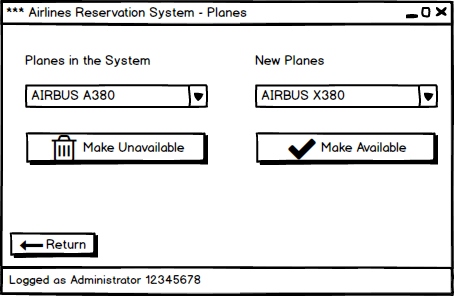
**Figure 3.7.24 - New Airport Menu**

**New Airport:** The administrator is able to add new airports to the system depending on the requests of the airline company.



**Figure 3.7.25 - Delete Airport Menu**

**Delete Airport:** The administrator is able to delete an airport depending on the requests of the airline company. S/he selects an airport from the list in the combo box and presses delete button.



**Figure 3.7.26 - Add/Delete Plane Menu**

**Add/Delete Plane:** The administrator is able to make planes available or unavailable for flights. The system has a default list of planes and after this operations, planes and seat plans become available when administrator creates a flight.

**4. Analysis  
4.1. Object Model  
4.1.1. Domain Lexicon**

**Admin:**  An employee of the airline company who has privileged rights. Admin will use the application to organize the flight schedule.

**User:** An employee of the airline company who does makes the reservations of the passengers.

**Passenger:** A customer of the airline company who is either about to make a reservation or has a reservation.

**ID:** An ID is the unique identifier of the airline company’ employees.

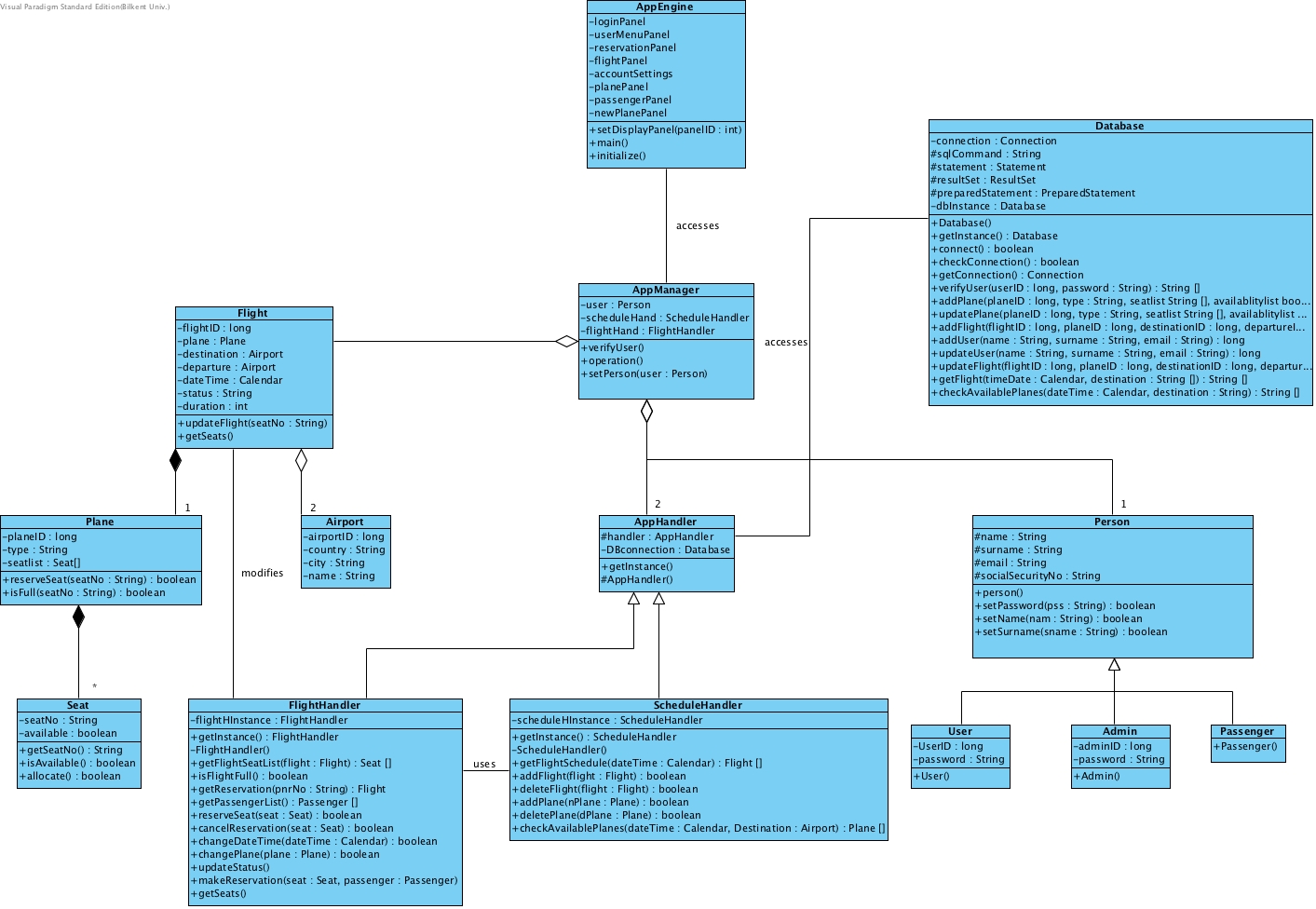
**Password:** Combination of numbers and characters that is necessary to be able to login the system. Each ID has a password.

**Destination:** It is the name of the airport that the flight will arrive.

**Departure:** It is the name of the airport that the flight will depart

**Reservation:** Withholding seats.

**4.1.2. Class Diagram(s)**



**Figure 4.1.2.1**

To make the class diagram clear, GUI classes and panels are not included. Brief description of classes and their most important methods in a nutshell is given below:

**AppEngine**: It is the most important and main class in our design. It is responsible for taking user inputs from mouse and keyboard. It is also responsible for the panel changes.

**AppManager**: This class has access to 2 AppHandler objects and gets necessary information from its operations. This class also has entity classes and initialize their objects. It also checks whether the entered id and passwords are correct or not and holds the object of the active user.

**AppHandler**: This class is a super class and holds the Database object to get the related informations from database and make the necessary changes on the database.

**ScheduleHandler**: This is a child class of AppHandler and is responsible for all of the database-application interactions about flight schedules. Its most important methods are getFlights which will return all of the flights on a specific date and to a specific destination from specific departure airport and getAvailablePlanes which will return the available planes that can be assigned to a new flight at a specific date from specific departure airport .Also it can use FlightHandler to update the flight objects.

**FlightHandler**: This is a child class of AppHandler and is responsible for all of the database-application interactions about specific flight. Its most important methods are getSeats which will return the available seats on the plane and make reservation which will reserve the seat for a passenger.

**Flight**: This is a model class that holds the information of a flight such as departure and a destination airports, date, time and plane. It also has a unique identifier flightID. Flight consist of Plane and Airport objects.

**Airport**: This is a model class that holds the information of an airport such as country, city and name. It also has a unique identifier airportID.

**Plane**: This is a model class that holds the information of a plane such as the seats and type of the plane. It also has a unique identifier planeID. Plane consists of Seat objects.

**Seat**: This is a model class that holds the information of a seat such as the availability of the seat and the passenger on the seat if the is is reserved.

**Database**: This class holds the connection to the database and is responsible from every query to the database. Its most important methods are getFlights which will return all of the flights on a specific date and to a specific destination from specific departure airport and getAvailablePlanes which will return the available planes that can be assigned to a new flight at a specific date from specific departure airport. the return type of its methods are generally String[] so that this class does not need to know any model class.

**PasswordChecker: This class is a static class its only purpose is to check the entered ID and password and let AppManager to know what is the type of the user.**

**Person**: This is a super class that holds the information about a person such as name, surname, email and social security number.

**User**: This is a model class who is child of the Person class. User class holds the information about a user such as the ID and the password of the user.

**Admin**:This is a model class who is child of the Person class. Admin class holds the information about a user such as the ID and the password of the admin.

**Passenger**: This is a model class who is child of the Person class.

**4.2. Dynamic Models  
4.2.1. State Chart**

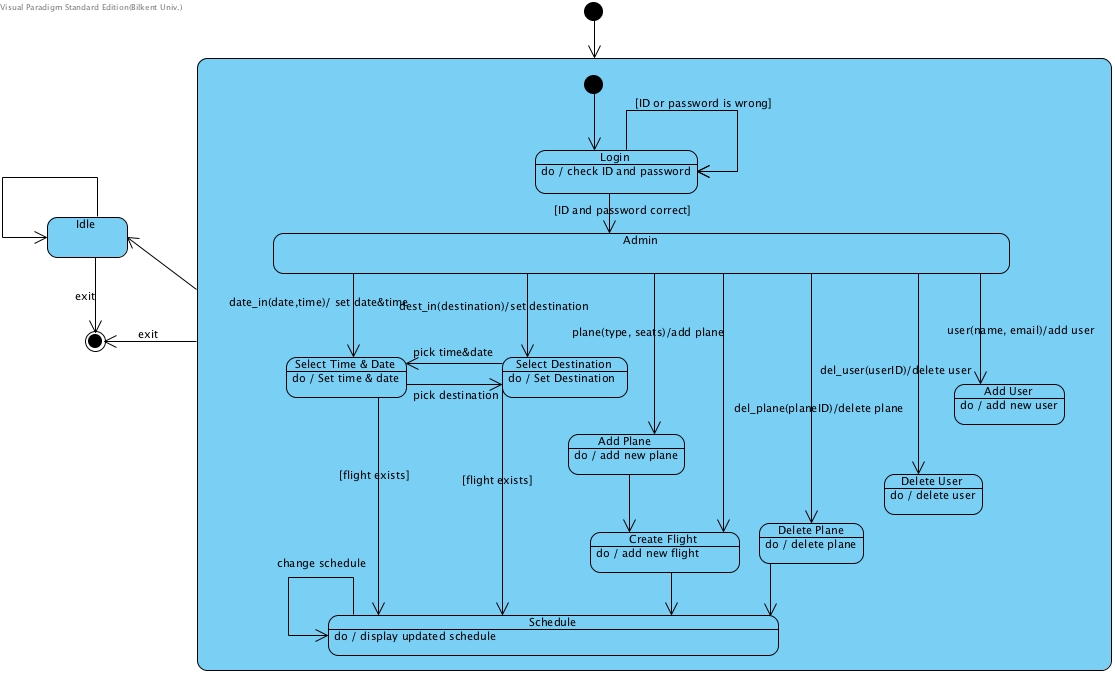


Figure 4.2.1.1- State Chart diagram for Admin

When an admin enters the application, at the menu page, he cannot register because to register he will use invalid id which is existing in the system. When he enters admin ID and password correctly, he will see the admin page. In this page, the admin can select to change schedule of busses, add new plane or flight, drop a plane or flight and make changes on the schedule also can see the schedule at a given date and time.

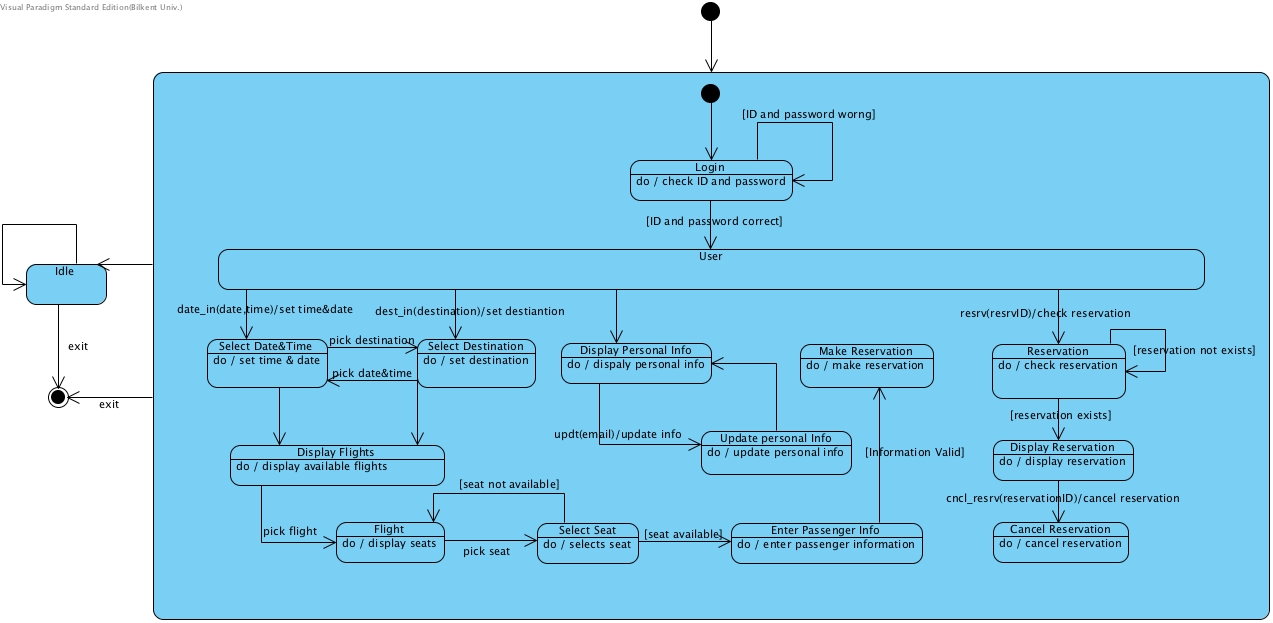


Figure 4.2.1.2 State Chart diagram for User

When an user opens the application he cannot register because his ID is already registered in the system. After he enters his ID and password, he will come to user page. In this page, he can reserve seat, drop a seat and also, he can change his password.

**4.2.2. Sequence Diagram  
Login to the System:**

Figure Sequence Diagram for Login

When a user logins to the system **AppManager will use PasswordChecker to verify the user and get the type of the user. PasswordChecker calls the verifyUser method of the Database to check the password and returns the type of the user and the users’ data to the AppManager.** If the user is an employee of the company and has User account then the new User object is created by AppManager and this object is hold by the AppManager to be able to restrict the functions according to the user type.

**Reserving Seat:**

Figure Sequence Diagram for Reservation

When user selects the date, time, departure and destination the AppManager will ask ScheduleHandler to bring the flight schedule according to given data. ScheduleHandler then uses Database class to access the database and check for the flight available. After getting the data from Database ScheduleHandler will Ask FlightHandler to create the Flights. When the ScheduleHandler gets the Flight objects it will return them to the AppManager. AppManager then call the setDisplayPanel method of the AppEngine to change to the flight panel then will initialize the panel according to the Flight objects. AppEngine calls FlightManager’s getSeats method. FlightManager gets the seat array of the Plane by calling the get seats method of the plane and returns the seat array to the AppEngine to be displayed.

User picks the flight from the panel. AppManager calls the getSeats method of the FlightManger. FlightManager gets the Seat array from the Plane by getSeats method and returns it to the AppManager. AppManager then calls the AppEngine’ setDisplayPanel method to display the interior of the plane.

User picks a seat to be reserved. AppManager calls the reserveSeat method of the FlightManger with the selected Seat. FlightManager calls the reserveSeat method of Database. If the Database returns true then FlightManager calls the reserveSeat method of Plane. If the Plane’s reserveSeat method also returns true then AppManager calls the AppEngine’ setDisplayPanel method to display the reservation succes. If any of the boolean returns return false then the AppEngine will display failure.

**Adding New Flight:**

Figure Sequence Diagram for Adding New Flight

Admin enters date, time, destination and departure airports. AppManager calls checkAvailablePlanes method of the ScheduleHandler. ScheduleHandler uses the Database method with the same name to get the available planes. ScheduleHandler then creates the plane objects and returns the array to the AppManager. AppManager calls setDisplayPanel method of the AppEngine to display the available planes.

Admin selects a plane. AppManager calls FlightHandler’s newFlight method. FlightHandler creates a Flight object and sends the Flight object to the ScheduleHandler to add the Flight to the schedule with addFlight method. ScheduleHandler calls the Database’ addFlight method and if this method returns true. FlightHandler returns true to the AppManager. AppManager calls the setDisplay method of AppEngine, if the return is true then AppEngine will display a success else failure.

**Conclusion**

To sum up, our airplane reservation system will be an desktop application in which reservations can be regulated. This will make reservation easy to follow and change. Also, user-friendly environment makes both user and admin work easy and faster.

Throughout analysis process, we have learned many useful things that, potentially, has significant importance in our future business life. Planning and analyzing are the phases that determines the path that we should follow throughout the development process. State charts and sequence diagrams are the tools that visualize plan that we have made therefore, they have great importance. This process has greatly helped us to comprehend how to use those tools.

Last but not least, analysis part teach us how to start a project from scratch. If we started our project before doing any analysis, we would have completely lost and we would not be able to proceed.