GROUP 10

PROJECT PROPOSAL 2

- **Group Members**

ALP EMİR BİLEK

MOHAMMAD ASHRAF YAWAR

İLKAN MERT OKUL

RAHMET ALİ ÖLMEZ

ECE ERER

ELİF KELEŞ

SENİHA SENA TOPKAYA

BİLAL BAYRAKDAR

NEVZAT SEFEROĞLU

Problem Definition

As days go by, more and more people are using airplanes for transportation. This increase demands for more planes, airports and employees, as the capacity of the current resources start to not suffice the needs. All this growth and increased complexity becomes a huge problem for the passengers, employees, managers and companies.

For the airline companies and airports to be able to keep up with the increase of demand, they need to boost the productivity of their personnel and managers. High productivity can be achieved by making every available operation easy to access. Some of the operations that airports have to offer include:

- Coordinating the airplanes to help them take off and land safely.
- Providing resources for airline companies to manage flight related operations such as ticket sales and check-ins.
- Implementing a display for available tickets, and the details of flights.
- Giving flight information for the passengers who already have tickets, when requested.
- Providing restaurants, banks, stores etc. to meet the needs of the passengers.
- Employing managers and personnel to organize all the resources.

Making these operations easy to access can be a problem if they are not organized and managed professionally, especially when millions of passengers are flying from a single airport each year. Thus, this problem requires a sophisticated, easy-to-use management system.

Users of the System

Users of this system are, customers, administrators of airline companies and airport, flight managers and check-in/ check-out personnel of airline companies, stores managers and company managers of the airport.

Customers can choose one of ‘the customer with ticket’ and a ‘customer without a ticket’ options to enter the system.

Airport Users

- **Airport Administrator** can add and remove Airline Company Manager and Store Manager.
- **Airline Company Manager** can add and remove Airline Company.
- **Store Manager** can add and remove Store.
- **Airport Traffic Tower** can check the flight information and manage the landing places.

Airline Company Users

- **Airline Administrator** can add & remove Flight Manager and Airline Personnel.
- **Flight Manager** can check flight information.
- **Airline Personnel** can manage check-in processes.

Customer Users

- **Ticketed Passenger** can check-in, access flight information and display all stores.
- **Guest** can access flight information on the system.

• Requirements (detailed)

User Requirements

- All the users should have a menu with choices to pick from.
- The user interface should be clear to understand and easy to use.
- A graphical user interface could be implemented, to improve the convenience of the system.

System Requirements

- The system will have separate menus for each user type, including administrators, managers, employees and customers. The menus will have a clear, understandable user interface for the system to be user-friendly.

- The system will provide an interface for both airports and airlines, as these closely work together for the passengers to be able to fly.

Functional Requirements

- The menus for registered users will be accessed only if the correct password is entered. Unless the correct password is entered, a warning will be shown to the user, stating that the password is incorrect.
- The users will only be able to access the operations related to themselves. In other words, no user will be able to make operations that do not belong to them. Also, no private data should be accessed other than the related user.
- Administrators will be able to add managers and employees, employees will add customers, airline companies' manager will add airline companies and store managers will add stores to the system.
- Flight managers will be able to access the flight information of the passengers and present it to them.
- Check-in personnel, as the name explains itself, help the passengers to check-in. They can also access to the flight information.
- The airport traffic tower will be able to check flight information and coordinate the flights to determine where they will be landing.
- Customers (passengers) will be able to browse available tickets and see their flight information. They will be able to buy tickets and register to the system to see their flight information from the system.

Non-Functional Requirements

- Aggregation should be used efficiently across the system where new managers, employees, customers etc. are created for better code reusability.
- All possible exceptions should be handled, and an error message should be shown to the user for each exception.

- ## Implemention Details

USER-

In this module, we create an User interface that includes needed methods signatures. This interface that is this module has some operations about users.

Some kind of users as Airport Admins or Company Managers are related with the system directly so they can access so many informations about the system, and also they have ids and passwords to enter the system.

Other side of the user module provide customers to acces some informations about airport and flights.

In this case they are some differences between customers. Some of them are just demand a ticket and wait for the acception. Some of them are just guest who curious about what kind of facilities the airport has and wants to learn more.

User interface includes these functions:
menu();
login();
add();
remove();
display();
getName();
getSurname();
getIdt();
setName(String name);
setSurname(String surname);
setIdt(int idt);

We assumed there will be 3 types of customers in our system.
First one is Guest;

- Can viewcompanies in the airport
- Can view flight options of the companies

- Can view many options from the menü
- Can demand a ticket

When the guest wants to get a ticket then he/she will wait for a while
In this case the system will take the guest’s information(name, surname ect.)
When the request is approved the guest will become Ticketed.
Second one is Ticketed;

- Can access his/her own ticket
- Can access flight informations
- Can access every screen which guest can view and more
- Can have an customer id since he/she was registered at the system

The third one is Customer only;
This class includes so many features.

- Admins and Managers can enter the system bay using their passwords
- They can access thier related screens
- They can also make some changes in the system (add, remove)
- All the final datas will be taken about flights, tickets, airport situations
- All cases will be considered, and if it is okay, the Guest will be converted to Ticketed

So there are two container in our User module;

Collection< Ticketed > listOfTicketed;

Collection< Guest > listOfGuest;

Main purpose of this module is providing efficiency for both registered users and customers. Also it provides different options that provide flexible usage.

AIRLINE

1- Arraylist to keep destination , Aircrafts and for employee to provide random access for employee recuiting and dissmissal purposes.

2- use binary search tree to assign Aricrafts to a particular flight and compare them in passenger capacity ,fuelTankVolume.

3- use queue to add and remove pilots in an airline company for the seek of Effeciency and to access elements in the fastest possible manner.

4-Destination structure keeps the data related with airline route.

Those routes should be able to removed or added to specific position on that list.Also ,

Our frequent operation will be get() .Therefore , ArrayList will be more efficient for now.

AIRPORT

-The basest type of the System.Anything will be managed from Airport class.

-Stores many lists for Customers, Airline Companies and Shops.

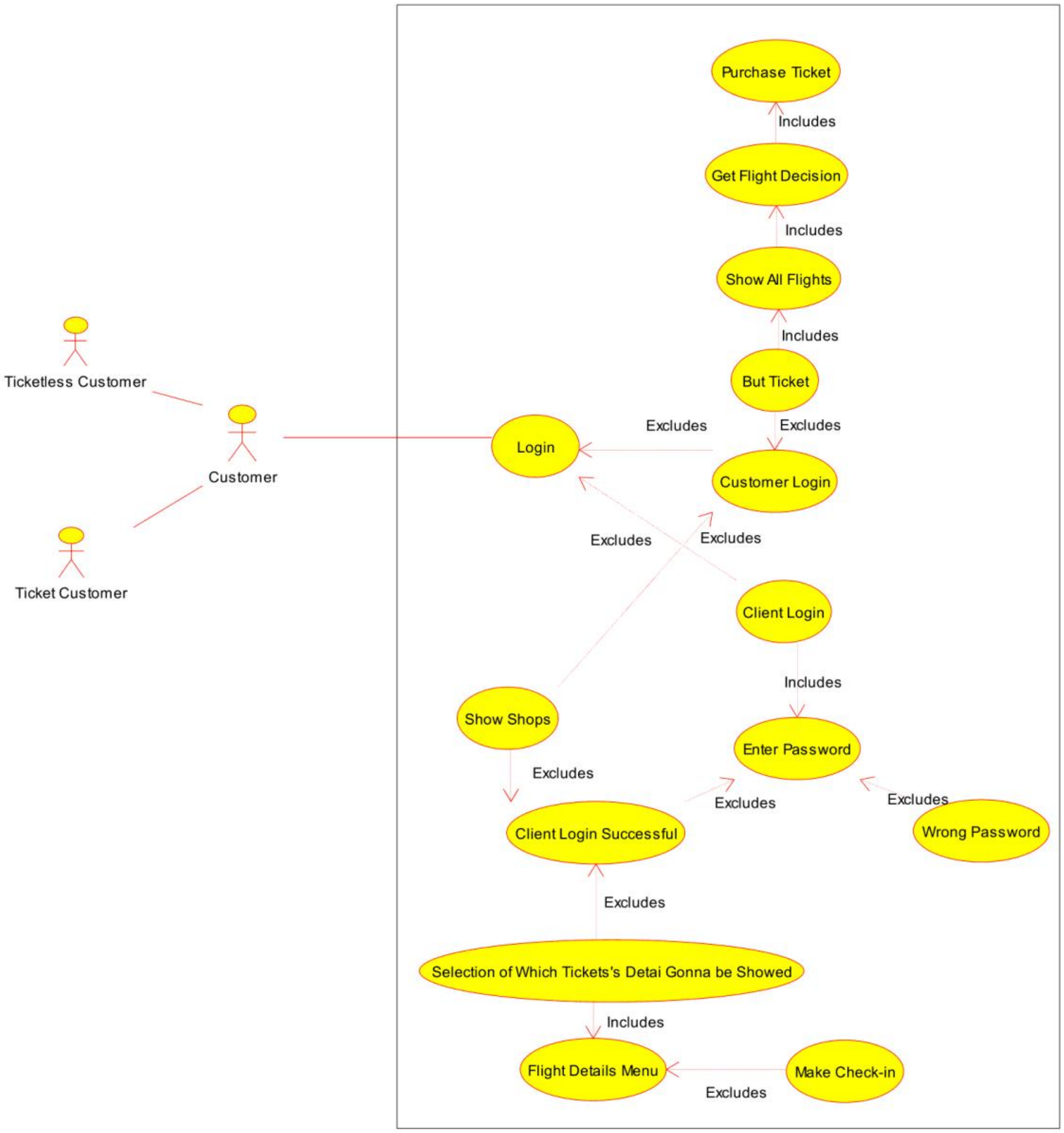
-These types have aparted Managers(ShopManager,CompanyManager)

-Managers can add, remove objects from their own branch lists.

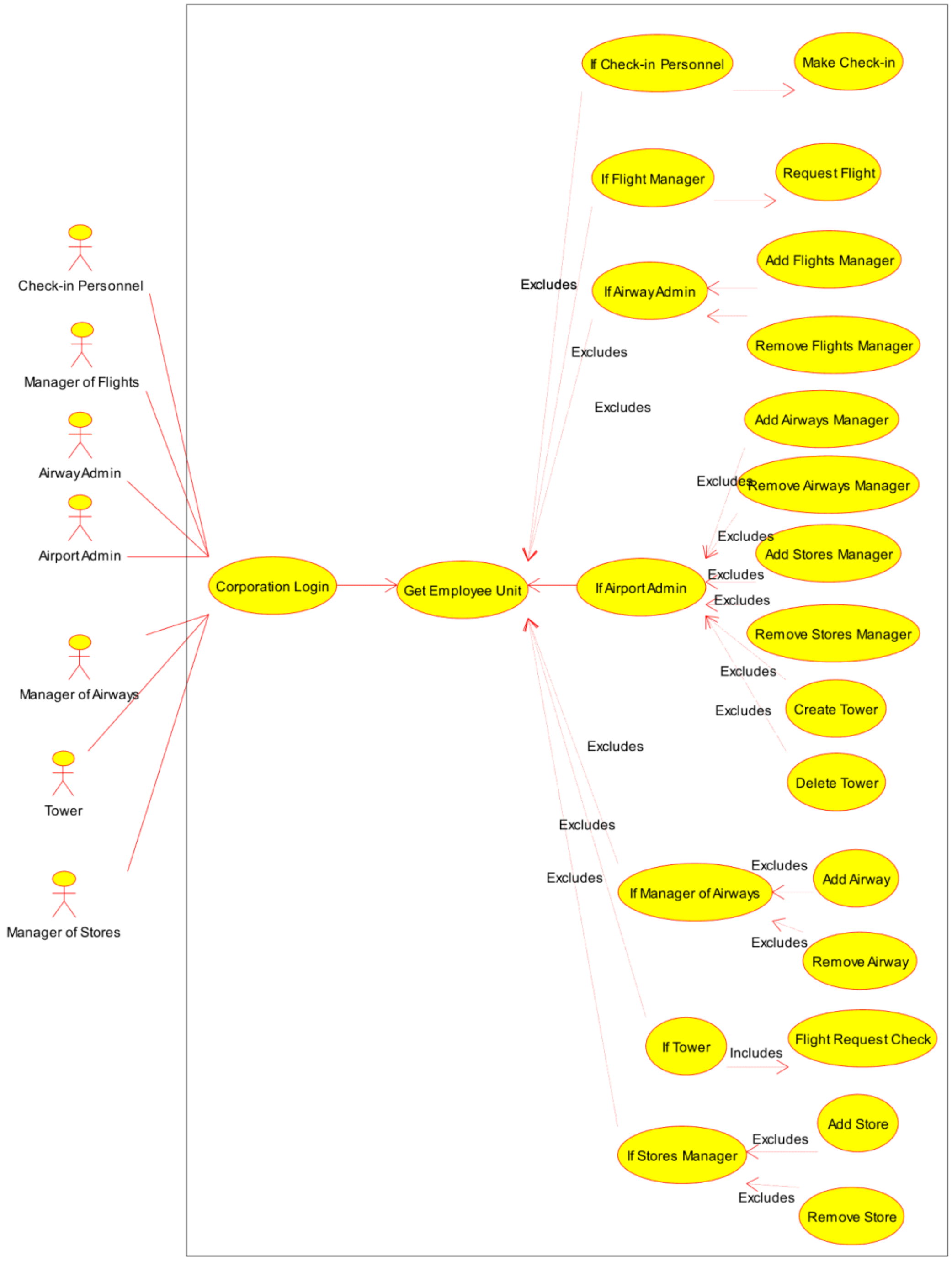
-AirportAdmin is the superior of that managers.It can manage the whole system.

- ## <u>Use-case diagrams</u>

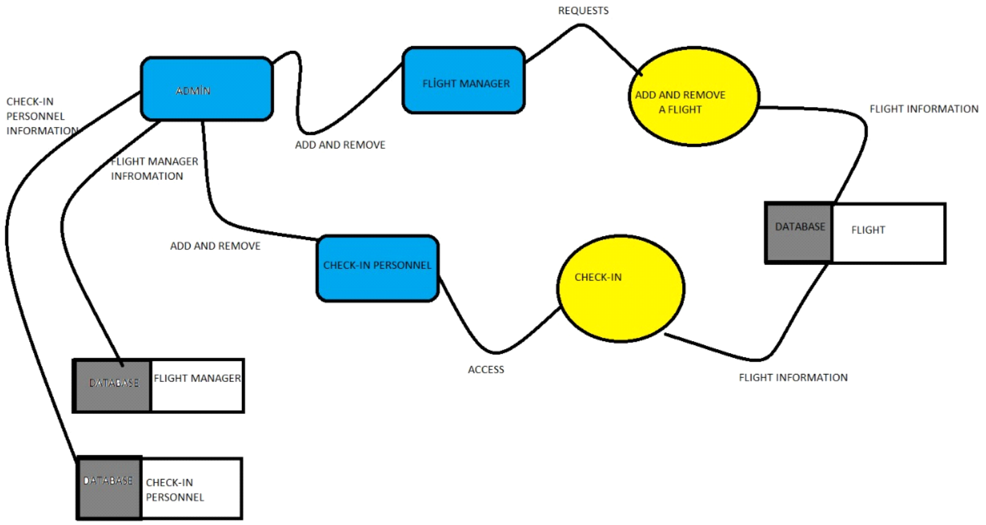
Customer Login



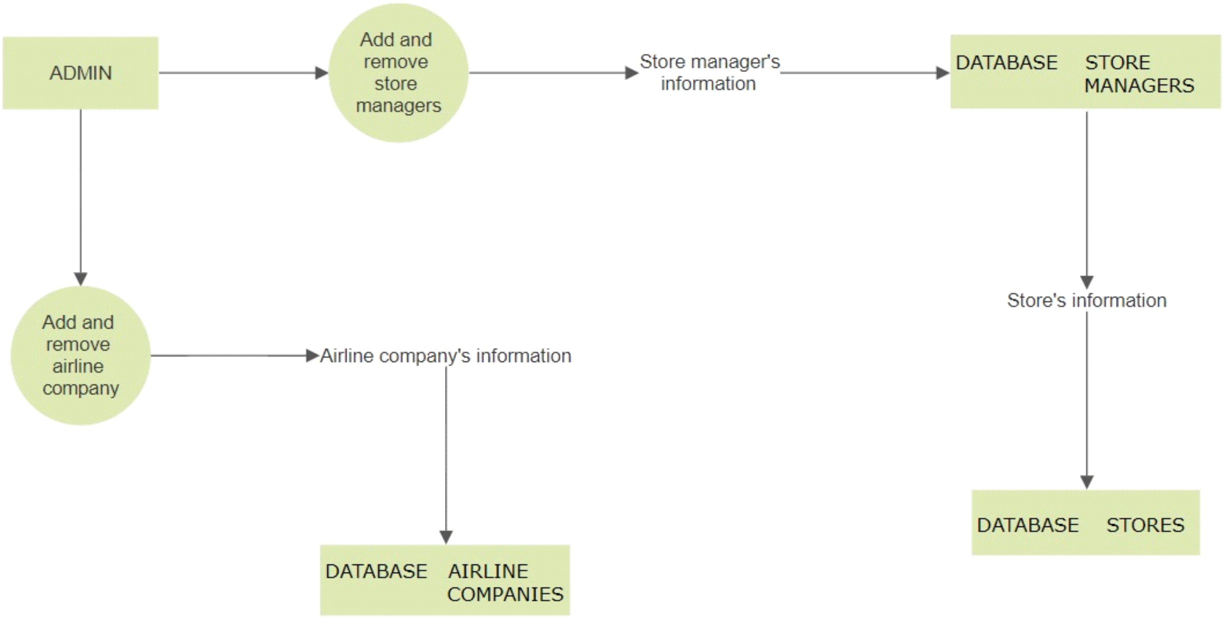
Employee Login Screen



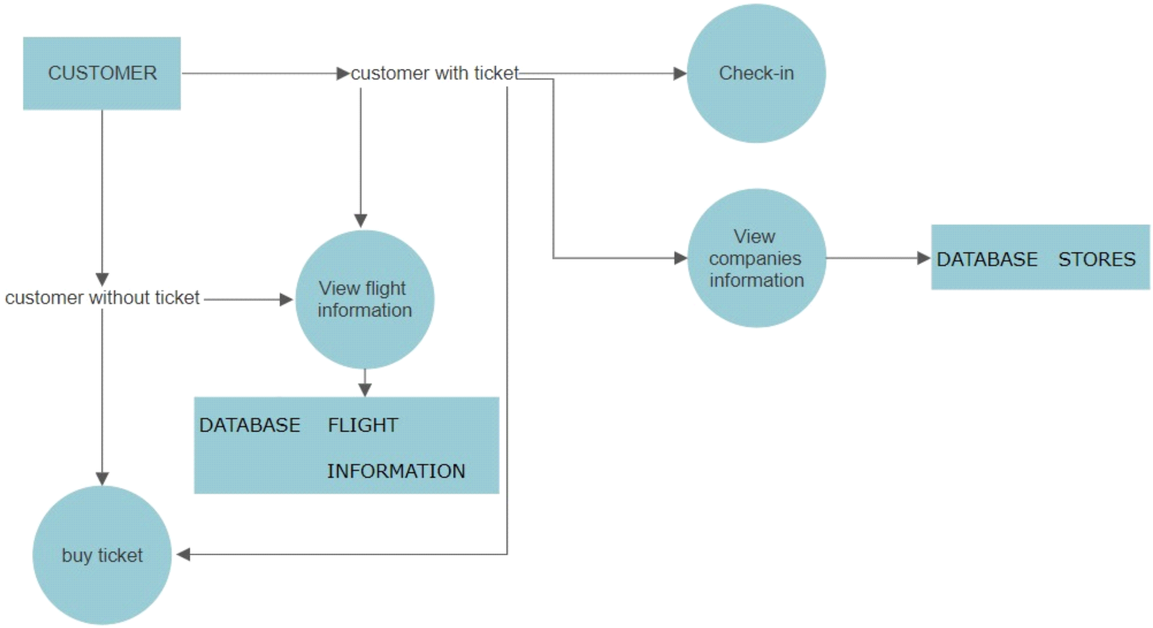
- **Module diagram (including system modules and their interactions)**



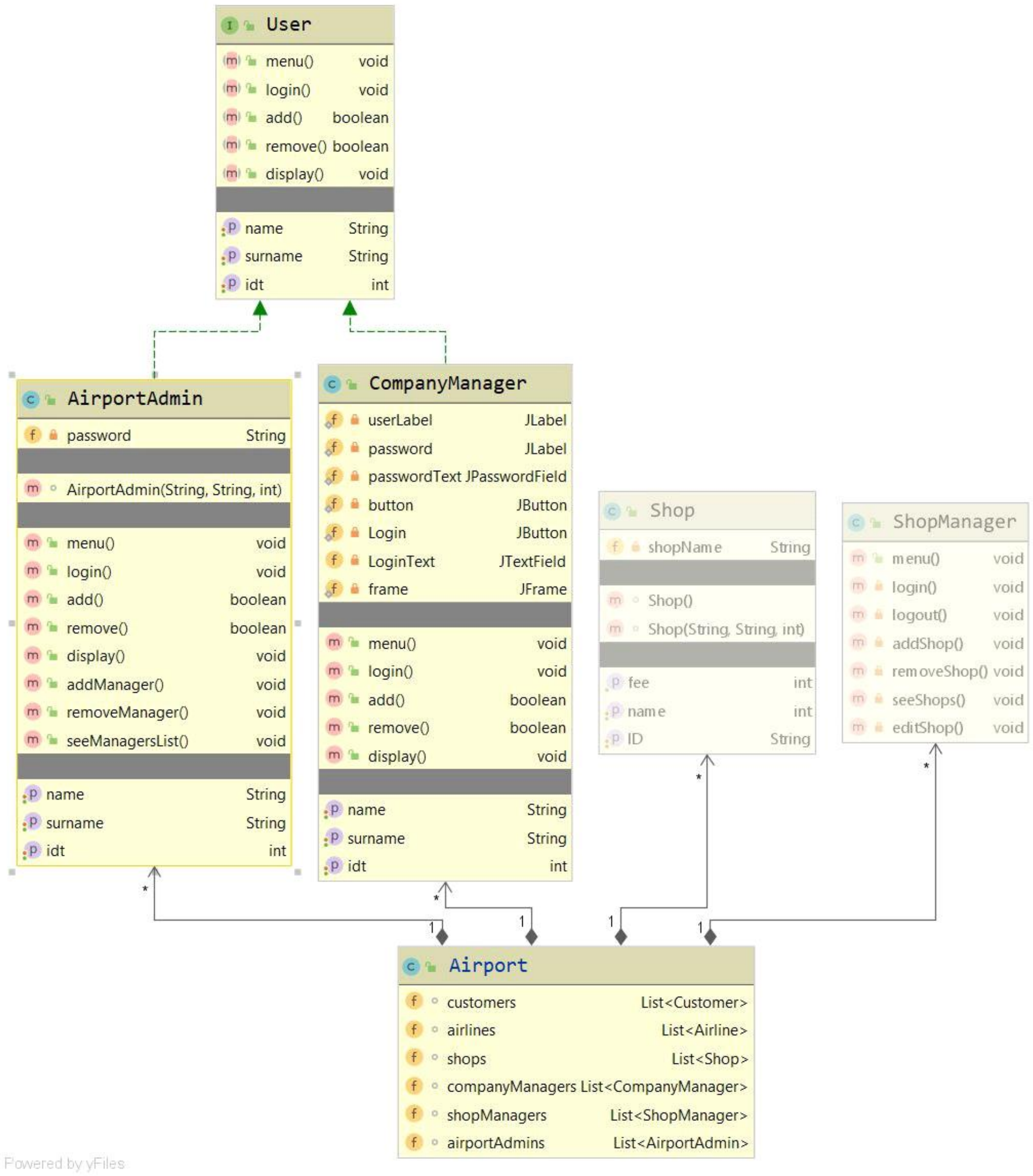
Airport Module Diagram

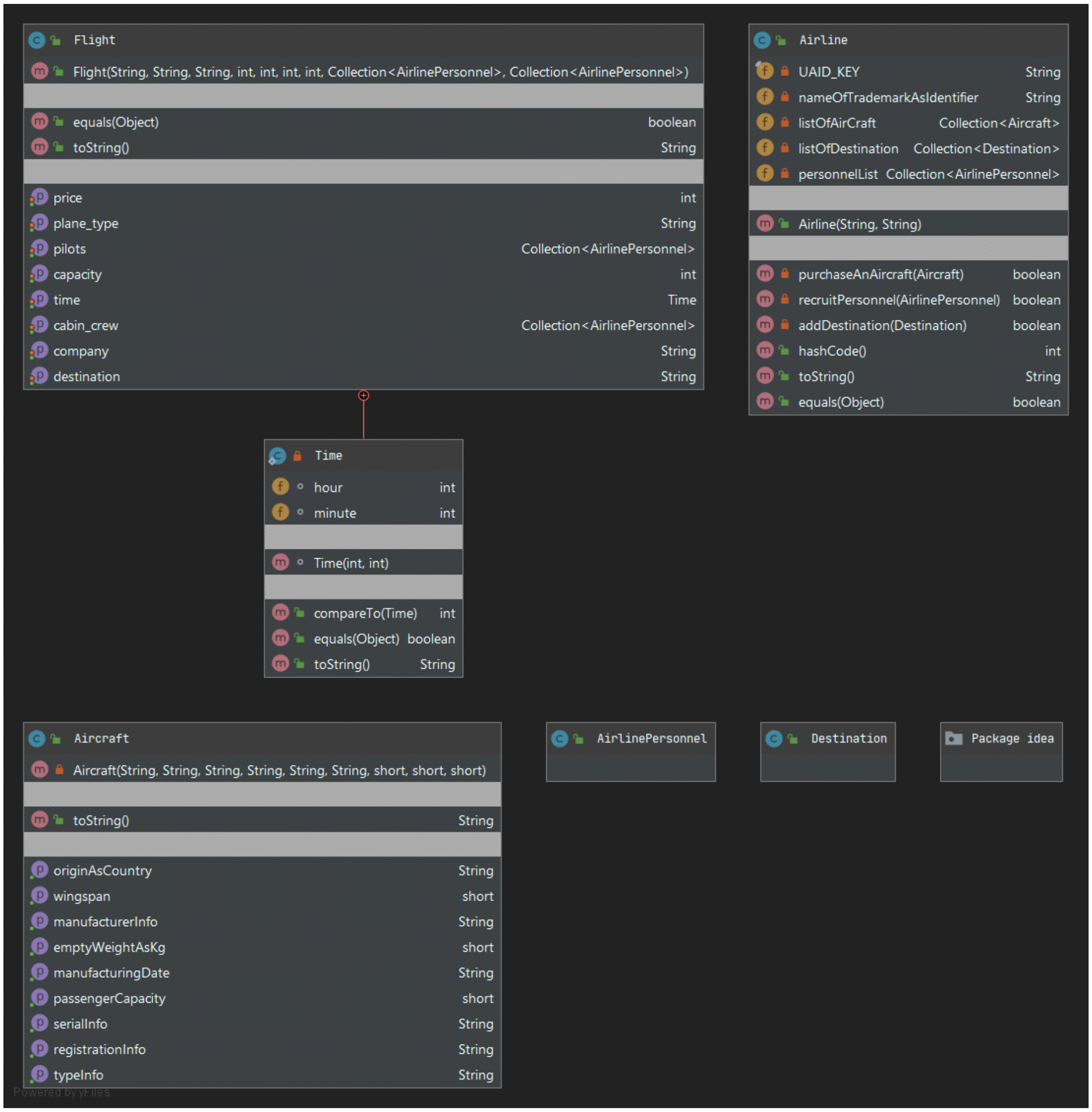

ADMIN
Add and remove store managers
Store manager's information
DATABASE STORE MANAGERS
Store's information
Add and remove airline company
Airline company's information
DATABASE STORES
DATABASE AIRLINE COMPANIES

Customer Module Diagram



CLASS DIAGRAMS





Flight

Flight(String, String, String, int, int, int, int, Collection<AirlinePersonnel>, Collection<AirlinePersonnel>)

equals(Object) boolean

toString() String

price int

plane_type String

pilots Collection<AirlinePersonnel>

capacity int

time Time

cabin_crew Collection<AirlinePersonnel>

company String

destination String

Airline

UAID_KEY String

nameOfTrademarkAsIdentifier String

listOfAirCraft Collection<Aircraft>

listOfDestination Collection<Destination>

personnelList Collection<AirlinePersonnel>

Airline(String, String)

purchaseAnAircraft(Aircraft) boolean

recruitPersonnel(AirlinePersonnel) boolean

addDestination(Destination) boolean

hashCode() int

toString() String

equals(Object) boolean

Time

hour int

minute int

Time(int, int)

compareTo(Time) int

equals(Object) boolean

toString() String

Aircraft

Aircraft(String, String, String, String, String, String, short, short, short)

toString() String

originAsCountry String

wingspan short

manufacturerInfo String

emptyWeightAsKg short

manufacturingDate String

passengerCapacity short

serialInfo String

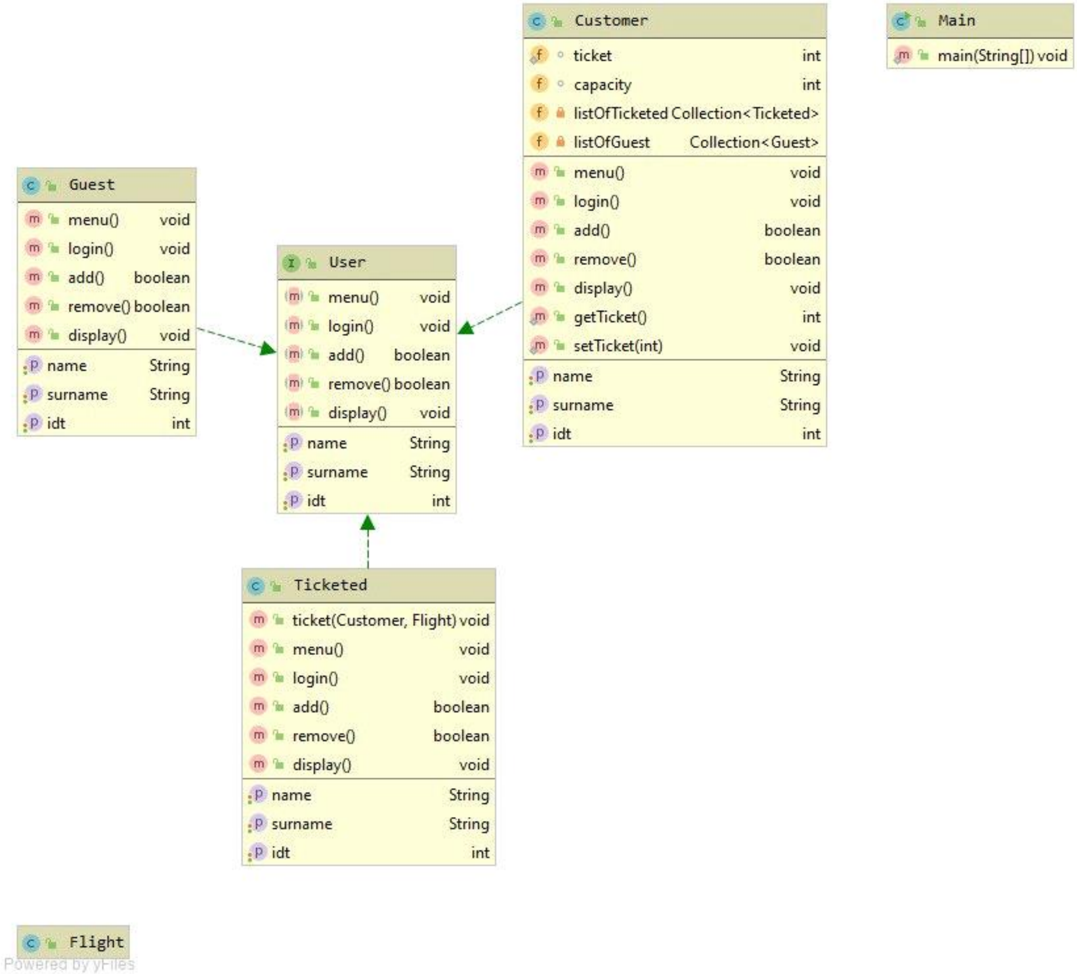
registrationInfo String

typeInfo String

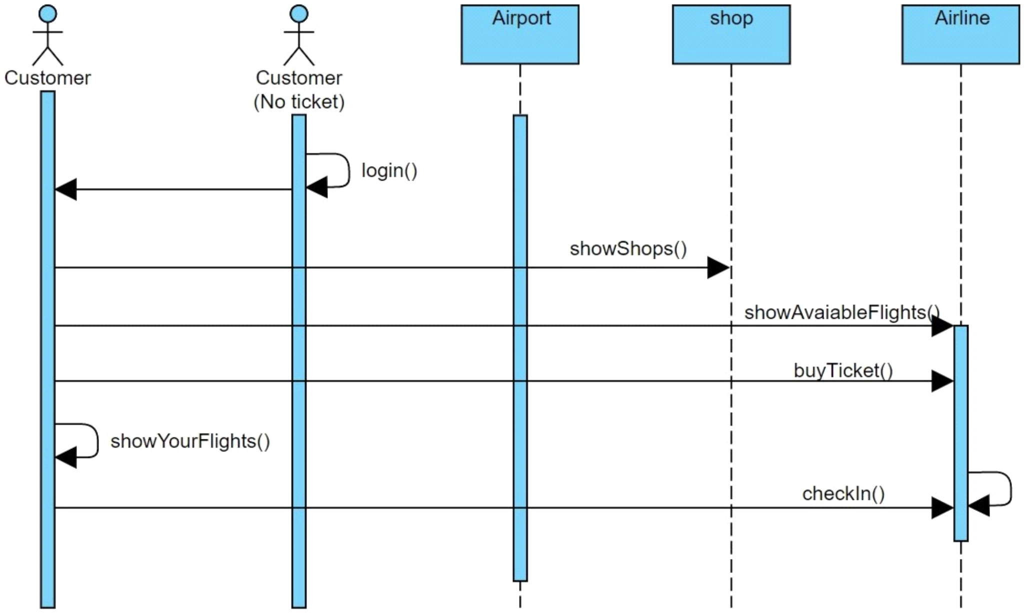
AirlinePersonnel

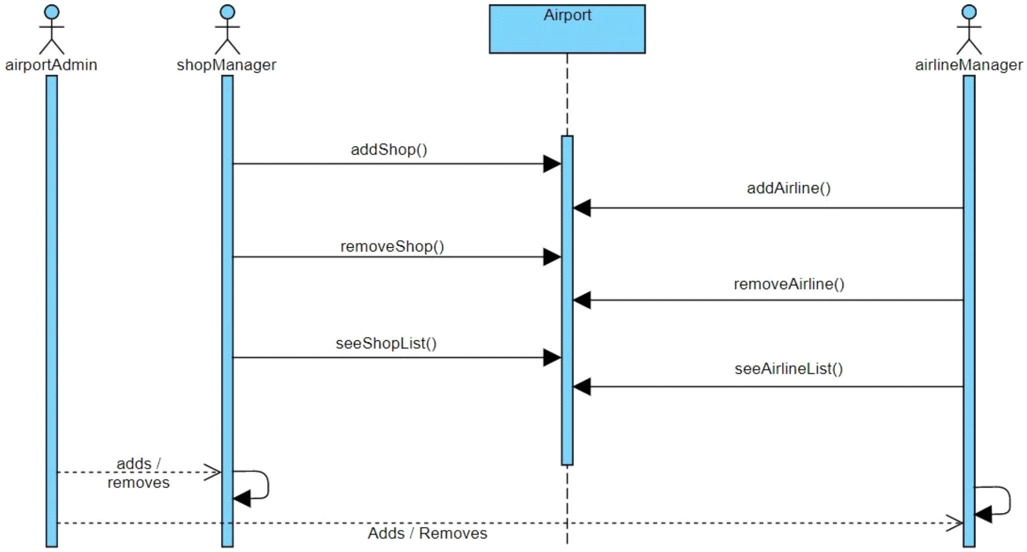
Destination

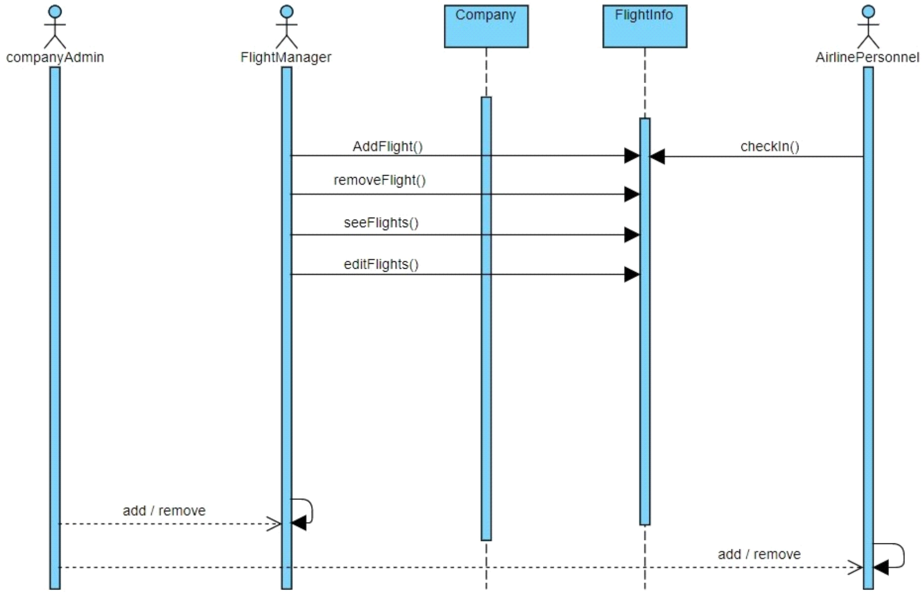
Package idea


Customer
ticket int
capacity int
listOfTicketed Collection<Ticketed>
listOfGuest Collection<Guest>
menu() void
login() void
add() boolean
remove() boolean
display() void
getTicket() int
setTicket(int) void
name String
surname String
idt int
Main
main(String[]) void
Guest
menu() void
login() void
add() boolean
remove() boolean
display() void
name String
surname String
idt int
User
menu() void
login() void
add() boolean
remove() boolean
display() void
name String
surname String
idt int
Ticketed
ticket(Customer, Flight) void
menu() void
login() void
add() boolean
remove() boolean
display() void
name String
surname String
idt int
Flight
Powered by yFiles

SEQUENCE DIAGRAMS:






companyAdmin
FlightManager
Company
FlightInfo
AirlinePersonnel
AddFlight()
checkIn()
removeFlight()
seeFlights()
editFlights()
add / remove
add / remove

Airport Test Case

Test ID	Scenario	Expected Result
1	Hiring a new CompanyManager	A new CompanyManager added to List
2	Login to System	Checks the userId and Password
3	Making a Selection from MENU	Jumps the selected Label
4	Adding couple new Airline Companies	New Airlines added to List
5	Displaying the Airline Companies	Listing the Airline Companies on the screen
6	Removing a new Airline Company	Firing the given Company from the Airport
7	Adding a new airport administrator	Airport administrator is added to the list
8	Removing an airport administrator	Airport administrator is removed from the list
9	Adding a new shop to the system	A shop is added to the list
10	Removing a shop from the system	The shop is removed from the list
11	Adding a shop manager	An instance of a shop manager is added to the list
12	Removing shop manager	Shop manager is removed from the list
13	Attempting to remove an object (i.e. Airport admin.) when the list is empty	An error message is shown to user

Airline Test Cases

Test ID	Scenario	Expected Result
01	Recuiting a new employee	A new employee added to company
02	Dissmissal of an employee	An existent employee removed from company
03	Accepting customer's flight request	Proper request of customer is accepted and ticket is created for this flight
04	Rejecting customer's flight request	Unproper request of customer is rejected
05	Destination adding for flight routes	New destination is added to destination container
06	Destination removing from flight routes	An existent destination is removed from destination container
07	New flight creation request from airport	New fligth is added to fligth container
08	Create UAID key	New uaid key is created for new ticket
09	Aricraft listing/sorting/removing/adding	Given processes are handled
10	Employee listing/sorting/removing/adding	Given processes are handled
11	Destination listing/sorting/removing/adding	Given processes are handled
12	Ticket listing/sorting/removing/adding	Given processes are handled
13	Flight listing/sorting/removing/adding	Given processes are handled

Customer Test Case

Test ID	Scenario	Expected Result
01	Buying ticket	The customer receives the ticket.
02	Log in	The customer logs in to the system with her id and password.
03	Requesting ticket	The customer looks at the flight information and requests a ticket for the date s\he wishes. If there is a ticket for that date, s\he can buy it later.
04	Check-in	Registration of customers is done before the flight.
05	Cancel ticket	Customer's previously bought ticket is cancel.