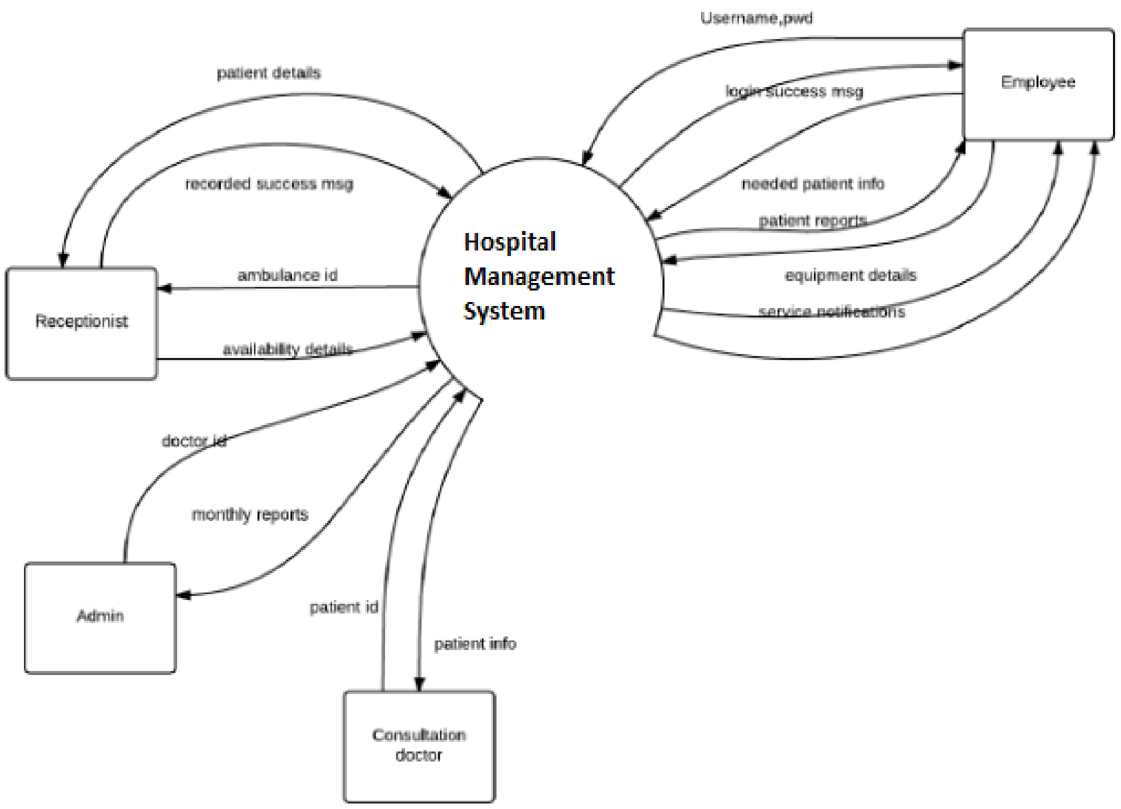
Task 1:

Your boss has given you functional requirements of a hospital management system and you must give implementations for each of these requirements. For reference, the basic flow of such a system is shown in the following diagram:



Functional Requirements:

Registration

1- Add patients
This function will allow front-desk staff to add new patients to the system.

2- Add doctors
This function will allow front-desk staff to add new doctors to the system.

3- Add staff
This function will allow the admin/owner to add new staff to the system.

4- Assign ID to patient
This function will allow front-desk staff to give each patient a ID and add it to the patient's record. The patient throughout his /her stay in hospital shall use this ID.

5- Assign Username and Password to doctor

This function will allow front-desk staff to assign a unique username and password to doctor. The doctor throughout his /her job in hospital will use this username and password.

Check In and Check Out

1- Assign Doctor to a Patient
This function will allow front-desk staff to assign a relevant doctor to every new patient.

2- Delete Patient ID
The administrative staff in the ward must be allowed to delete the ID of the patient from the system when the patient checks out.

3- Add to beds-available list
The administrative staff in the ward must be allowed to put the beds just evacuated in beds-available list.

Report Generation

1- Patient information
This function will generate reports on patients about the following information: patient's ID, patient's name, ward name, bed number and the doctor's name, which was assigned.

2- Bed Availability
The system will generate reports on bed availability about the following information: ward name, bed number, occupied/unoccupied.

Task 2:

Consider a storage structure. In this structure, the first value stored is the first one retrieved i.e. first-in-first-out (FIFO) policy. It is like a line of customers in a bank: The first one to join the queue is the first one served. Besides a constructor, your program should have two functions: one called insert () to put a value in your storage structure and one called read () to get value from the storage structure. You will need two variables for the storage structure: one called head to point to the front of the storage structure, and one called tail to point to the rear end of the storage structure. Values are placed at the tail (like the last customer getting in line at the bank from the rear end) and removed from the head. The head and tail variables will change whenever values are added and removed from the storage structure.

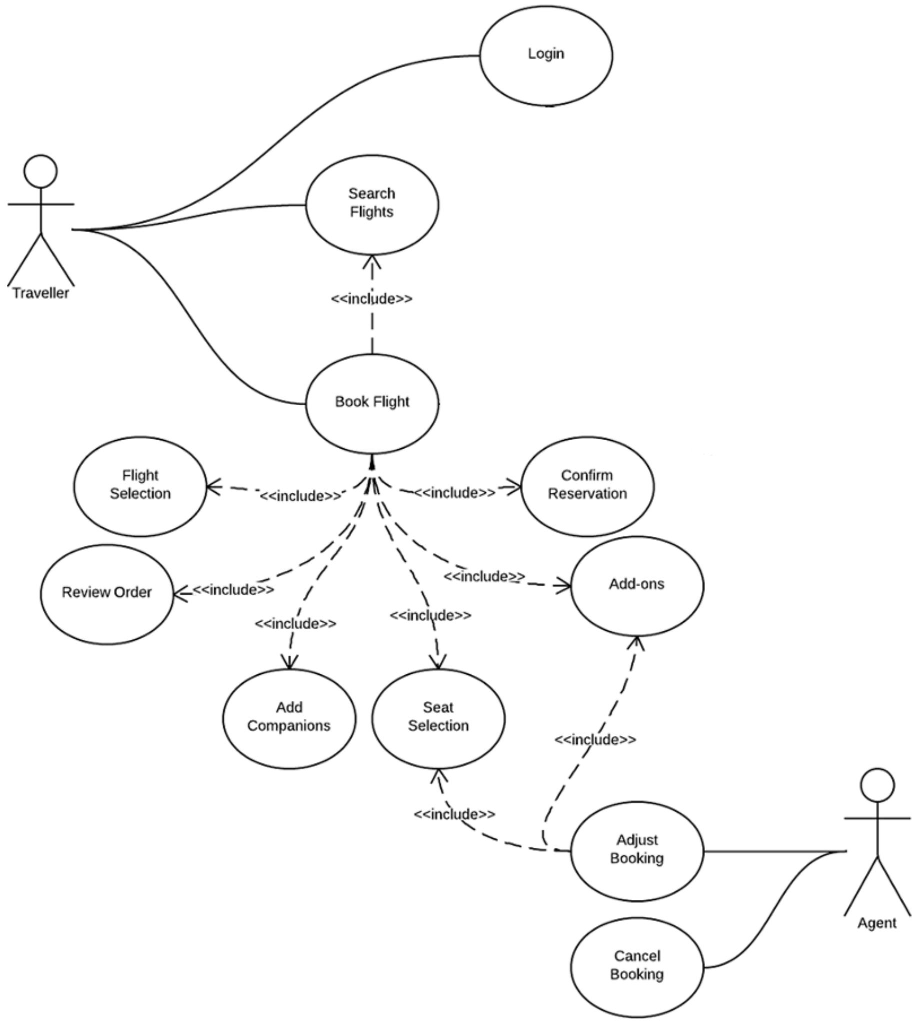
1. Implement the storage structure using any appropriate concepts you learned in this course.
Note: The values in the storage structure can be of any type.

2. Implement a function to find any given value in the storage structure. Take a value (any type) to find as user input.

3. Implement a function to split the storage structure into two equal parts. If there are odd number of values in storage structure then disregard the last stored value

Task 3:

You are required to deliver a project to manage air ticket reservation system. For this, you are given the following functional requirements that must be fulfilled:



FUNCTIONAL REQUIREMENTS:

1. The system admin can create flight and modify its details

2. The admin also has information about the airplanes and availability of seats for each flight

3. The user/customer need to provide all the necessary details in order to make reservations or bookings for flights

4. A function must be give to enable user to search for available flights through departure time, their current city and destination city

5. The customer can book flights in three different categories (VVIP, Business & Economy). They can also add options (in-flight meal & extra legroom) if they book a flight in Business or VVIP class).

6. The customer can also book a flight thorough a booking agent

7. The customer can check the flights he has booked but cannot modify the details. However, he can cancel his booking till three days before the flight. Cancellation will cost him 2% of the ticket amount.

Task 4:

A multi−set is a collection of items much like a set, in that order of items is not important. However, it differs from a set in that items can be repeated in a multi−set. An example of a multi−set is:

{1, 2, 5, 5, 6, 7, 7, 7}

You need to implement the following functions associated with the multi-set.

1) **Insert method:** The insert method should insert a new item to the multi−set at the beginning if it does not already exist. Otherwise, it should add the item in the multi−set at the end.

2) **Remove method:** The remove method will remove one item (the one insert last) from the multi-set. If there is just one item in the multi-set, then simply remove that item.

3) **Merge method:** The merged multi-set will have only one copy of each item found in two different multi-set.

   Example:

Set_1	=	{1, 3, 4, 5, 6, 3, 7}
Set_2	=	{3, 5, 6, 4, 6, 8}
Merged_Set	=	{1, 3, 4, 5, 6, 7, 8}

Note:

1. In your implementation, any OOP concept must not be violated.
2. The multi-set can be of any type (int, char, double, etc.).
3. The size of the multi-set can vary.