T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY of ENGINEERING
DEPARTMENT of COMPUTER ENGINEERING

HOSPITAL AUTOMATION SYSTEM

CSE 222 DATA STRUCTURES AND ALGORITHMS
PROJECT REPORT 2

GROUP
14

LECTURER
Prof. Dr. Fatih Erdoğan SEVİLGEN

TEACHING ASSISTANT
Başak KARAKAŞ
Muhammed Burak KOCA

KOCAELİ, 2021

İçindekiler Tablosu

1. GROUP MEMBERS

Group 14 Members

Burak YILDIRIM	1901042609
Cem BOZKURT	1801042090
Çağla ŞAHİN	171044050
Merve DUR	141044022
Mete GONCA	161044075
Ömer Faruk SAYAR	171044038
Şeyda Nur DEMİR	121044042

2. PROBLEM DEFINITION

Hospital Automation System

One of the main problems with daily routine of hospital management is scheduling patient appointments. With increasing number of patients and their appointment requests, it is hard to plan these routine.

As a solution to this problem, we are planning to develop a hospital management system. With this management system, patients will be able to schedule appointments and doctors will see these requests. Also, patients can schedule appointments via receptionist, face to face. Patients can be vaccinated by nurses, also can take car from nurses when they are inpatient. Administrator have all authority. It is the managing role that has ability to access and edit each entry on the management system.

3. USERS OF THE SYSTEM

Administrator

Administrator can add/remove staff, add/remove bed to/from dorm, see list of hospital staffs and see the current situation in dorm.

Doctor

Doctors can accept patients, see their appointments, see patients' medical informations, see patients' previous appointments, make appointments to patients, set patients' current status (e.g., discharged, inline) and see admitted patients.

Nurse

Nurses can vaccinate patients in queue and take care of inpatients.

Receptionist

Receptionists can open accounts for patients, confirm appointments of patients.

Patient

Patients can get an appointment, cancel an appointment, see their appointments and see their prescription.

4. REQUIREMENTS IN DETAILS

4.1. Functional Requirements

- There is a hospital management system.
- There are users of the system.
  These user roles are administrator, doctor, nurse, receptionist and patient roles.
- Users have an account in the system.
- Users see an interactive menu when opened the system.
  User must register to the system to use it.
- Users can sign up to the system by entering their information using menu.
- Users can sign in to the system by entering their username and password using menu.
  User must be already registered to the system to sign in.
- Users can sign out of the system by following the menu.
  User must be already logged in to the system to sign out.
- System also calculate the time interval to inform the users about appointments.

- All users can use menu, but each user see different menu as its user role.
- Administrator has all authority to use this system.
- Manager of the hospital has the administrator user role.
- Doctor has limited authority, doctors of the hospital have the doctor user role.
- Nurse has limited authority, nurses of the hospital have the nurse user role.
- Receptionist has limited authority, receptionist of the hospital has the receptionist user role.
- Patient has limited authority, all patients of the hospital has the patient user role, patients differs by their patience degree, they may be inpatient or outpatient.
  Patients may be anyone who wanting to be vaccinated.
- Doctors, nurses and receptionists are staff of the hospital.

- Administrator can add or remove beds.
- Administrator can list all staff.
- Administrator can search for a specific staff.
- Administrator can see dorm status.
- Administrator can hire of fire (add or remove) a staff.
- Administrator decide (set) the minimum vaccinate age.

- Doctor can call patient.
- Doctor can view all own appointments.
- Doctor can view medical information of the its patient.
- Doctor can also view previous appointment of the its patient.
- Doctor can change (set) its patients' status after examine.
  Doctor can also give prescription to the patient.
- Doctor can list all inpatients.
- Doctor can cancel (clear) its all appointments.

- Nurse can vaccinate patients who wanting to be vaccinated.
- Nurse can take care who calling to care.

- Receptionist can register a patient to the system.
- Receptionist can confirm appointments.

- Patient can add or remove appointment.
- Patient can view its all appointments.
- Patient can view its all prescriptions.

4.2. Non-Functional Requirements

- Scalability : We will use various data structures,
  such as arrays, queues, maps, lists, etc. to hold big data.
- Portability : We will use Java programming language.
  Java code can be executed on several platforms, Windows, Linux, Sun Solaris, Mac OS, etc.
  The Java code is compiled by the compiler and converted into bytecode.
  This bytecode is a platform independent code.
- Security : Each system user will have mail and password to login.
  We will use hashing to keep the passwords.
- Maintainability : We will follow java code name conventions.
  We will add javadoc documentations for all classes, methods, variables etc. and we will write tests.
- Flexibility : This system can be used in any hospital.
- Reliability/Robustness : We will do exception handling gracefully and check unexpected inputs.

5. USED DATA STRUCTURES

Binary Search Tree

We will use binary search tree to store information about the users of the system and for search purposes.

Priority Queue

We will use PriorityQueue to store appointments based on their dates to get the first patient with closest date.

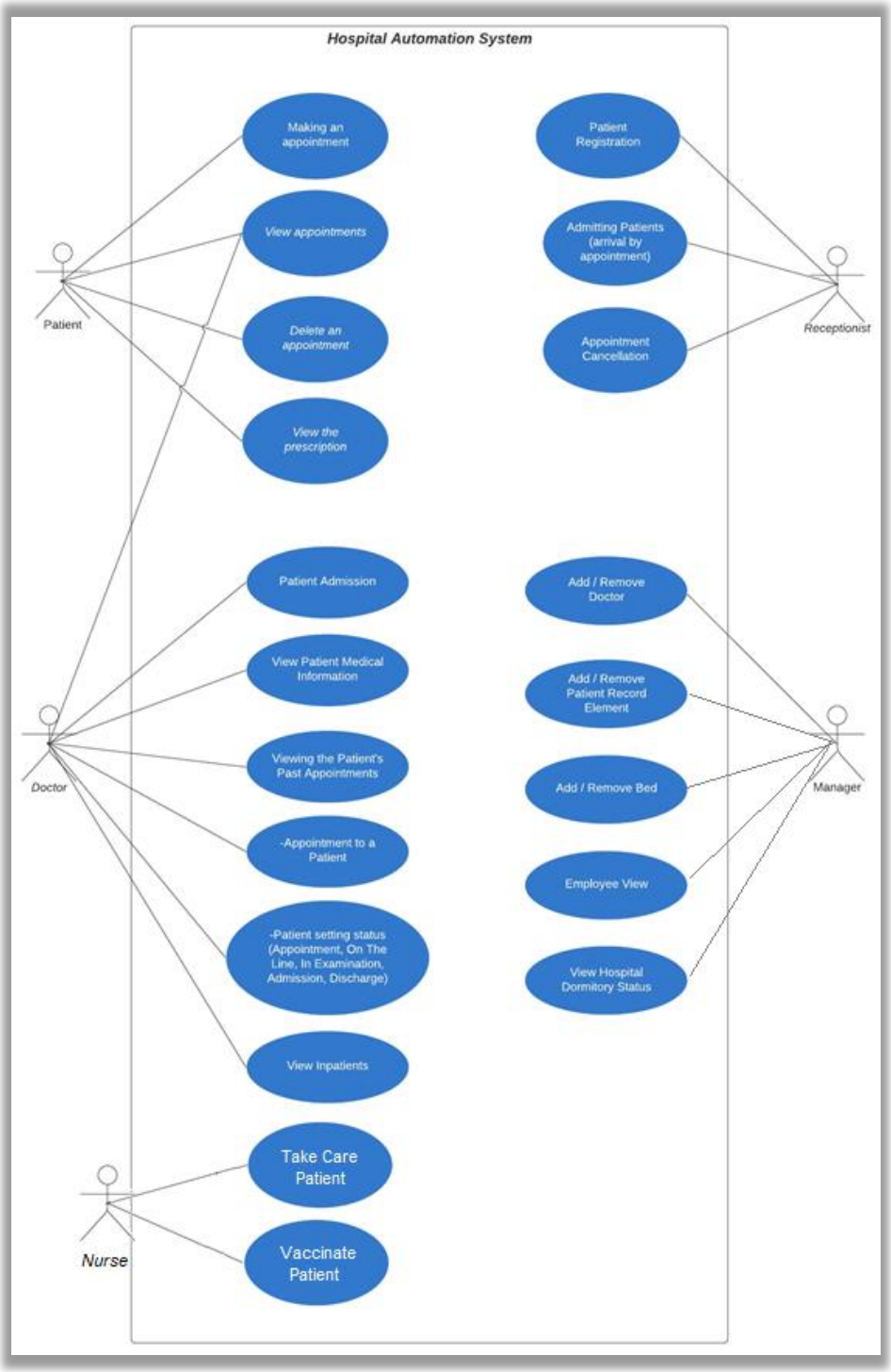
Stack

We will use Stack to store the patients' previous appointments to access them from end to beginning.

List

We will use List (ArrayList) to store the information other than the ones mentioned above because it is easy to use and have constant time methods.

6. USE CASE DIAGRAMS

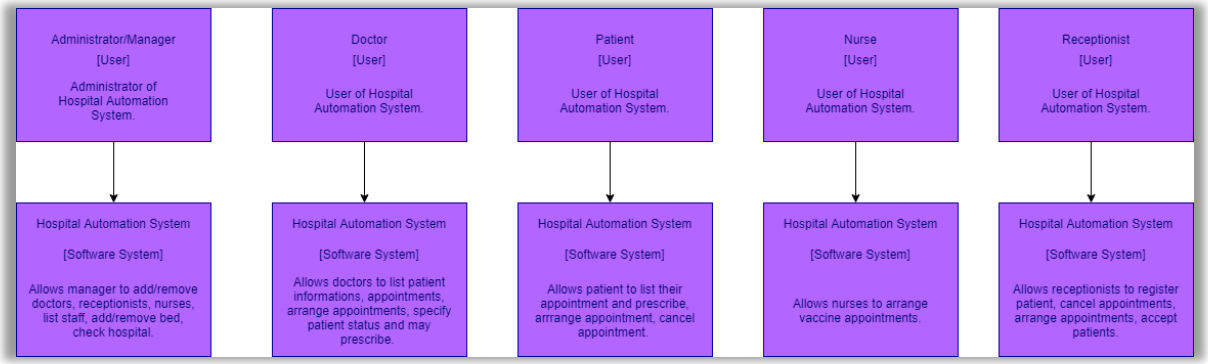


7. C4 MODEL OF THE SYSTEM

7.1. Level 1 : System Context Diagram

This is a system context diagram for a hospital automation system. It shows the people who use it.

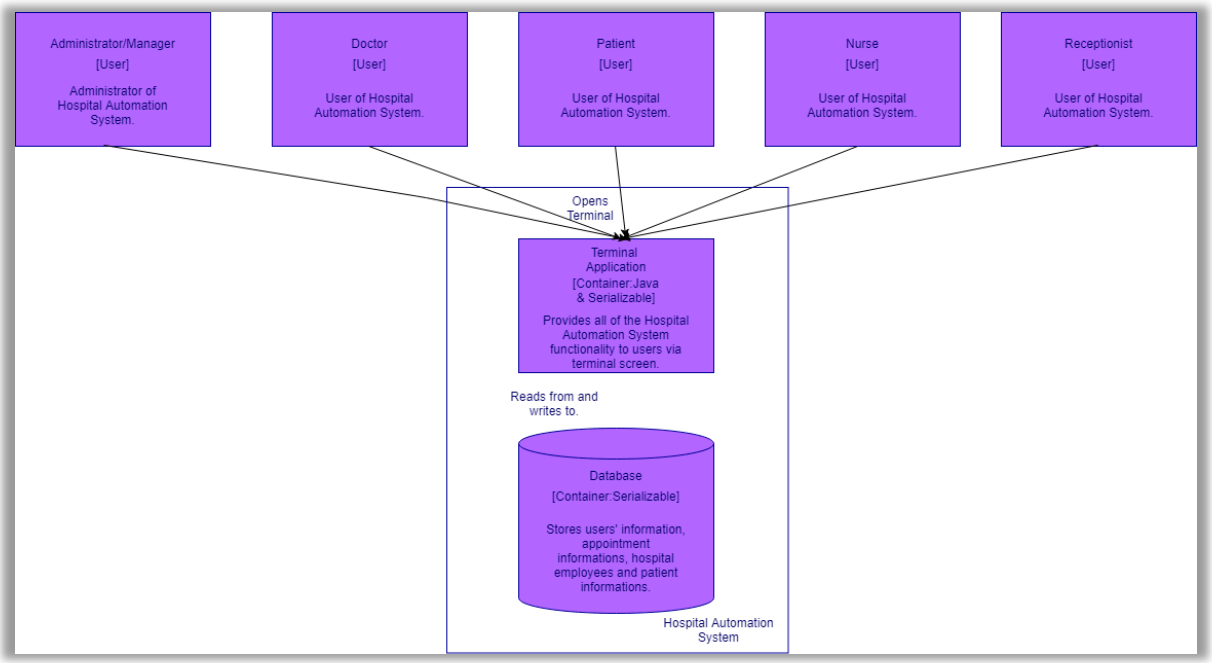
Doctors of the hospital use the automation system list patients information, appointments, arrange appointments, specify patents status and may prescribe. Patients of the hospital list their appointment and prescribe, arrange appointment, cancel appointment. Manager of the hospital adds/removes doctors, adds/removes receptionist, list staff, adds/removes bed, checks hospital. Receptionist of the hospital register patient, cancel appointments, arrange appointments, accept patients.



7.2. Level 2 : Container Diagram

This is the container diagram for a hospital automation system. It shows the high-level shape of the software architecture and how responsibilities are distributed across it. It also shows the major technology choices and how the containers communicate with one another.

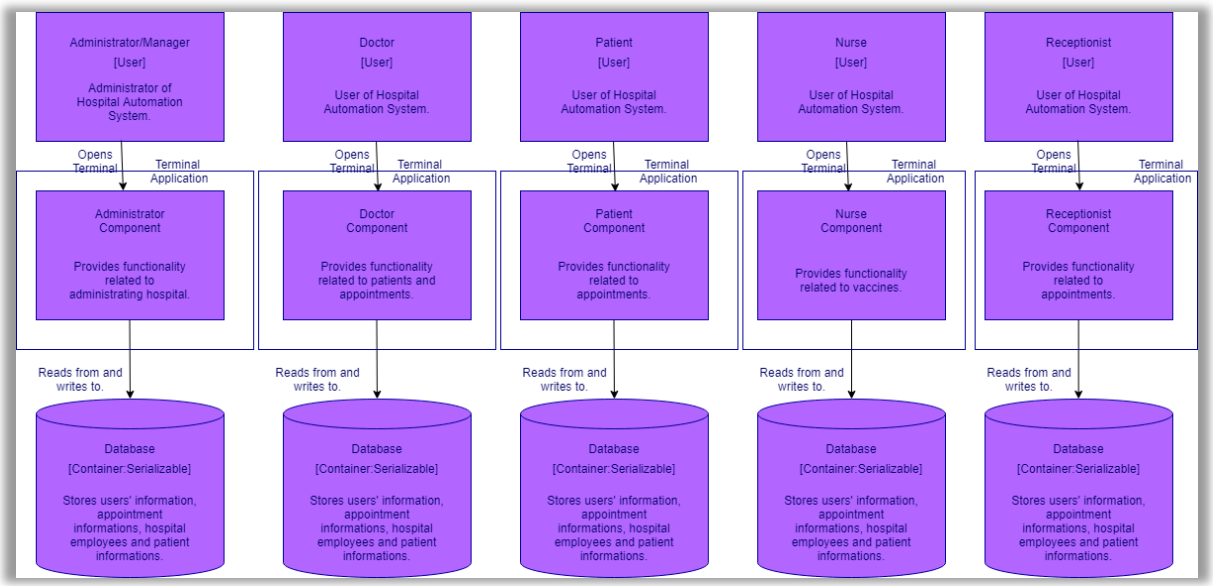
Hospital automation system is a terminal application, uses Java for backend, also stores needed informations in a Json file locally. If it is needed, this automation system can provide a graphical user interface.



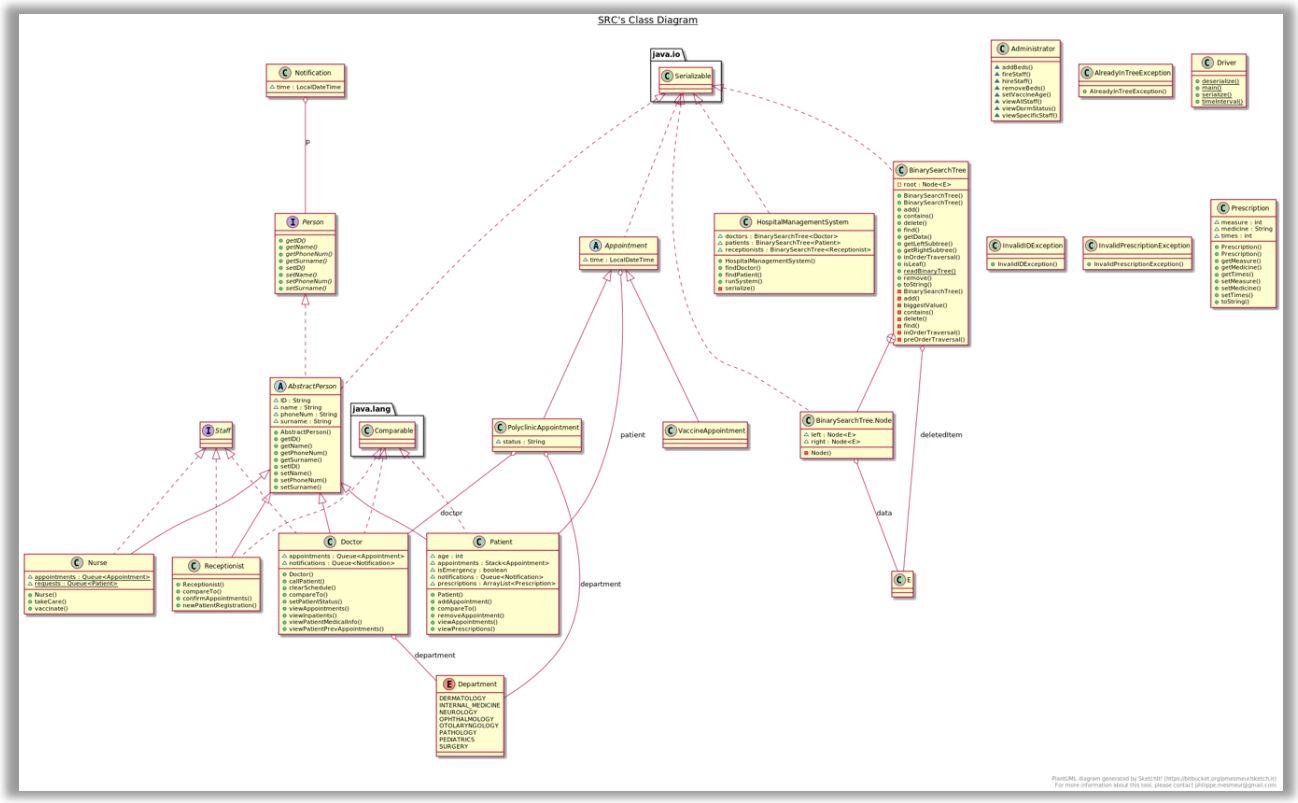
7.3. Level 3 : Component Diagram

This is the component diagram for a hospital automation system. It shows how a container is made up of a number of "components", what each of those components are, their responsibilities and the technology/implementation details.

Hospital automation system components provides users functionally related to their user roles. Administrator component provides functionally related to administrating hospital. Doctor component provides functionally related to patients and appointments. Patient component provides functionally related to appointments. Nurse component provides functionally related to vaccines. Receptionist component provides functionally related to appointments.

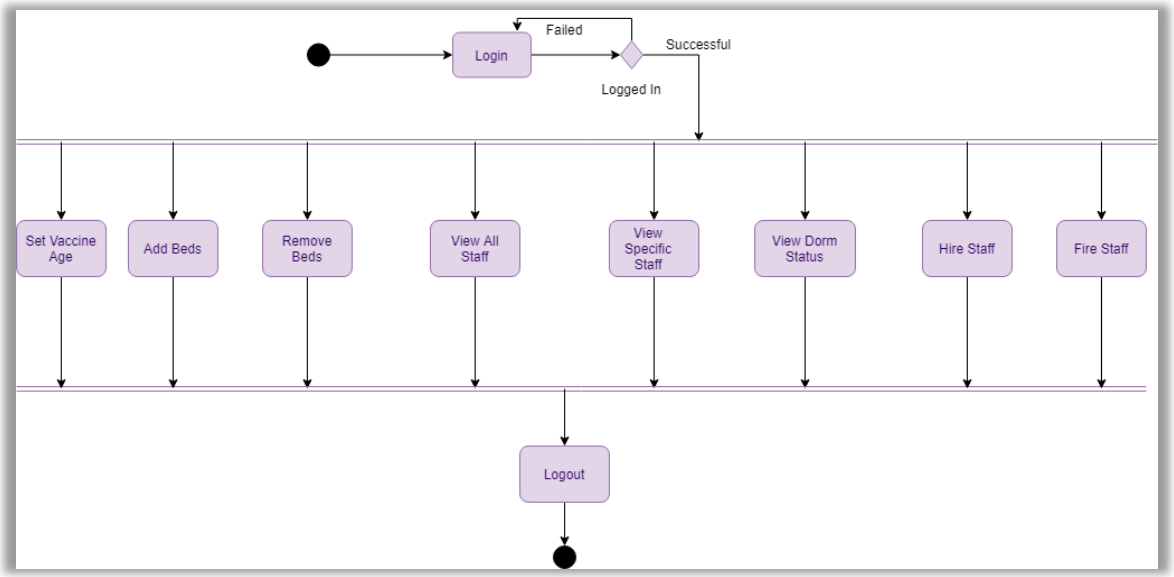


8. CLASS DIAGRAMS

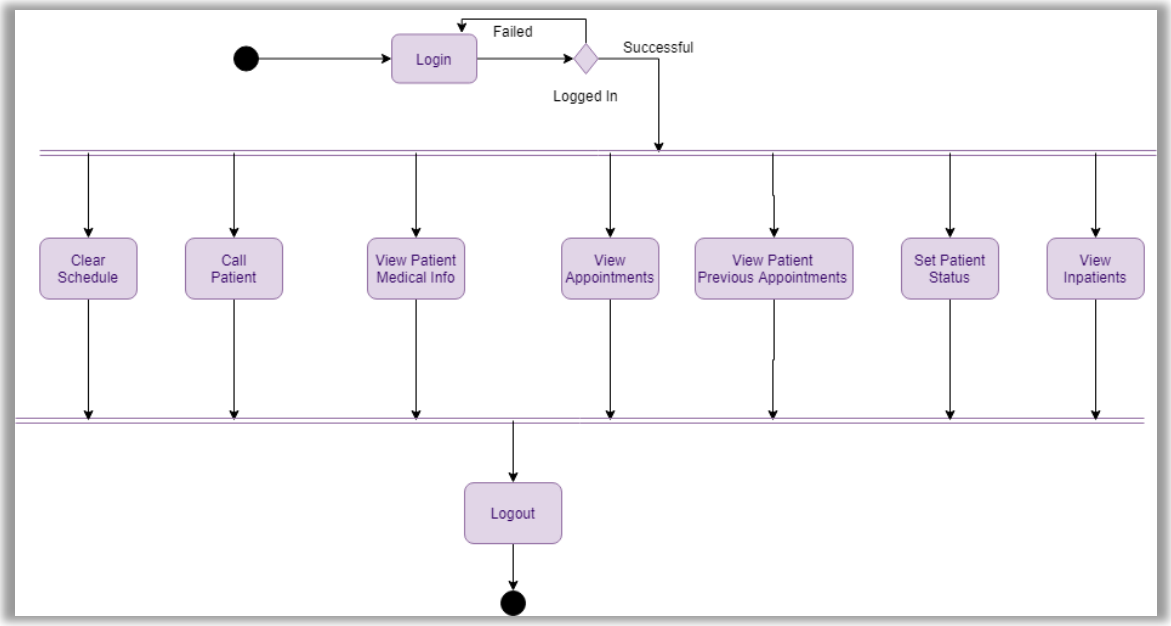


9. ACTIVITY DIAGRAMS

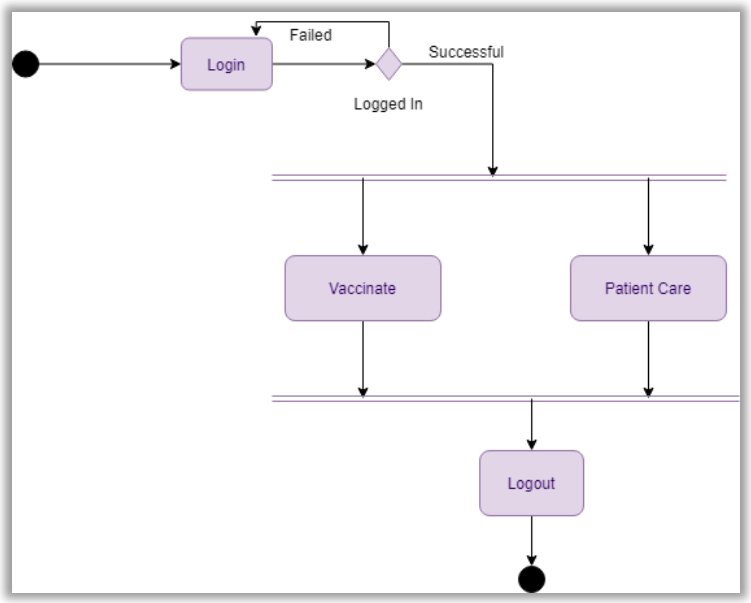
Administrator



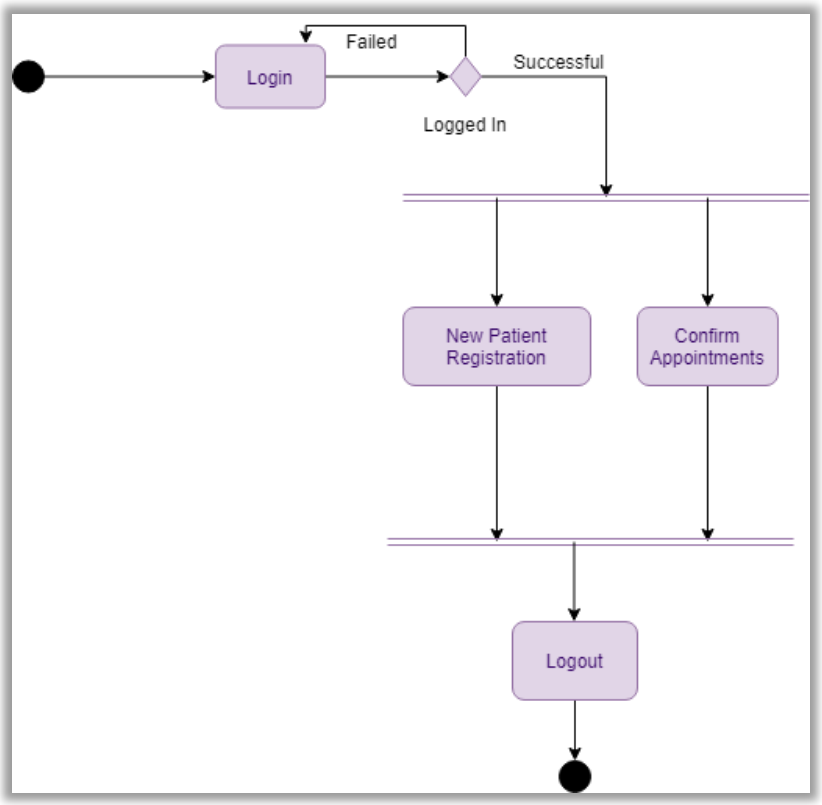
Doctor



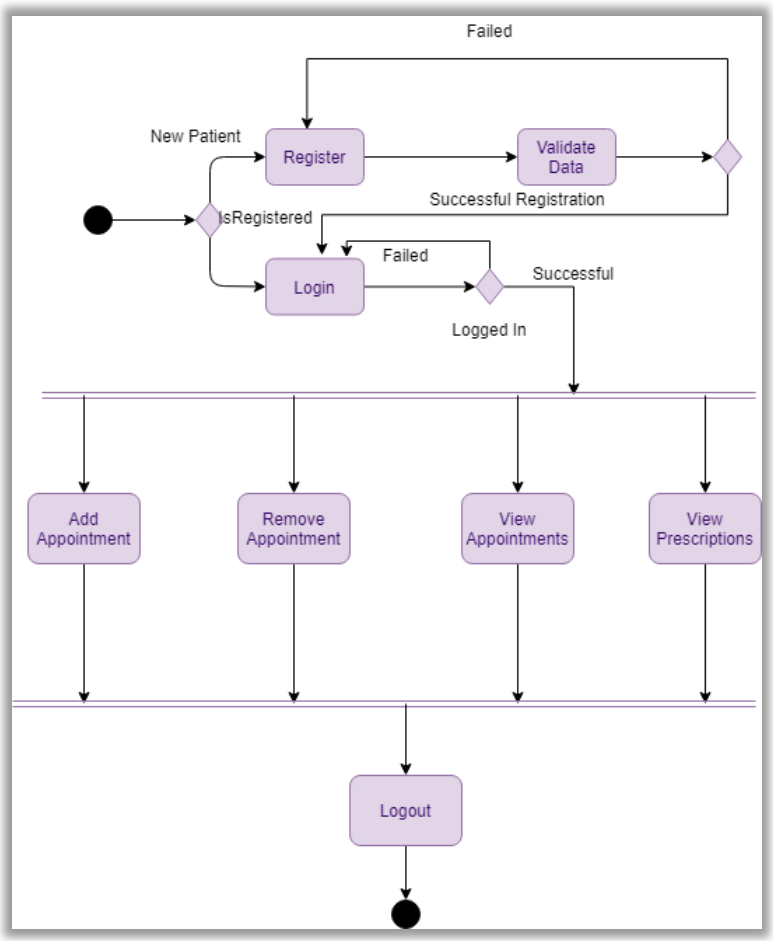
Nurse



Receptionist

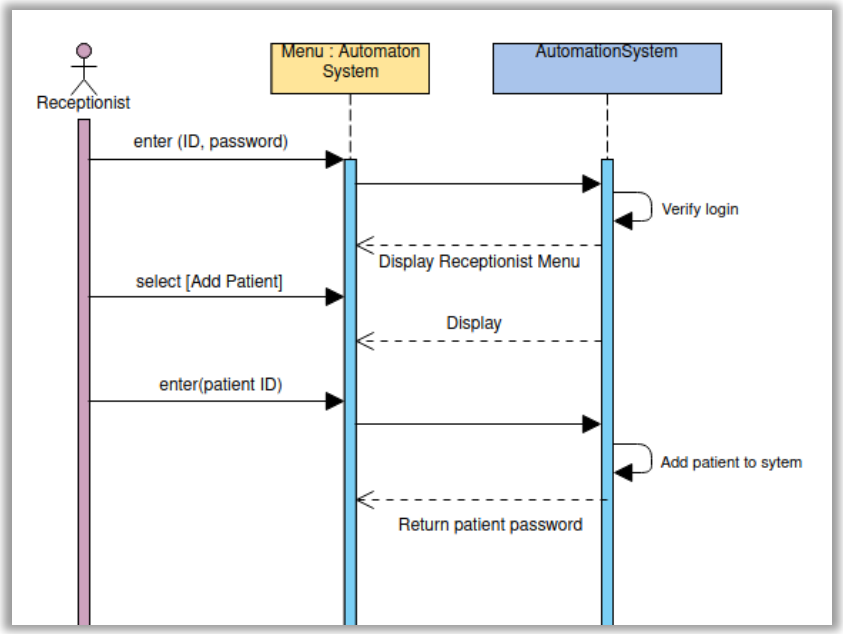


Patient

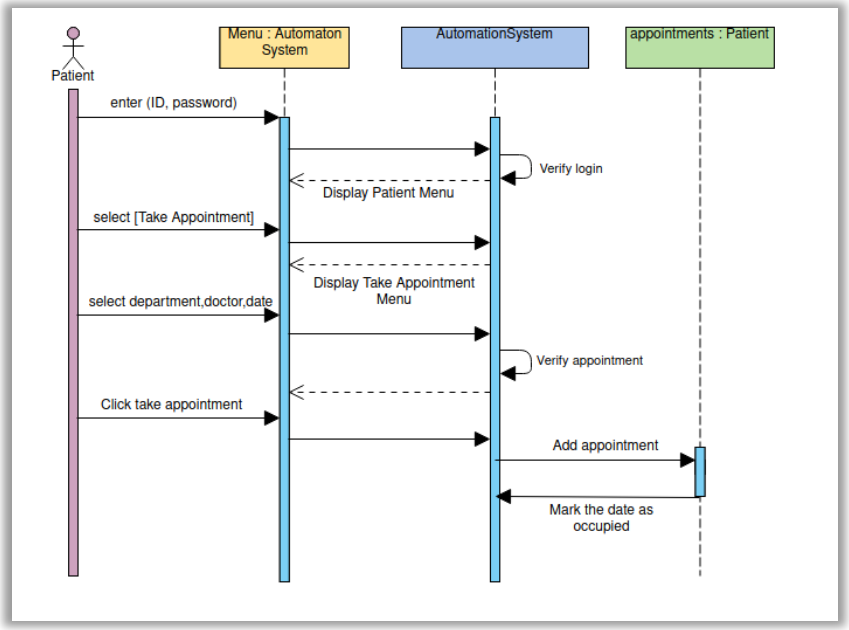

Failed
New Patient
Register
Validate Data
Successful Registration
IsRegistered
Failed
Login
Successful
Logged In
Add Appointment
Remove Appointment
View Appointments
View Prescriptions
Logout

10.SEQUENCE DIAGRAMS

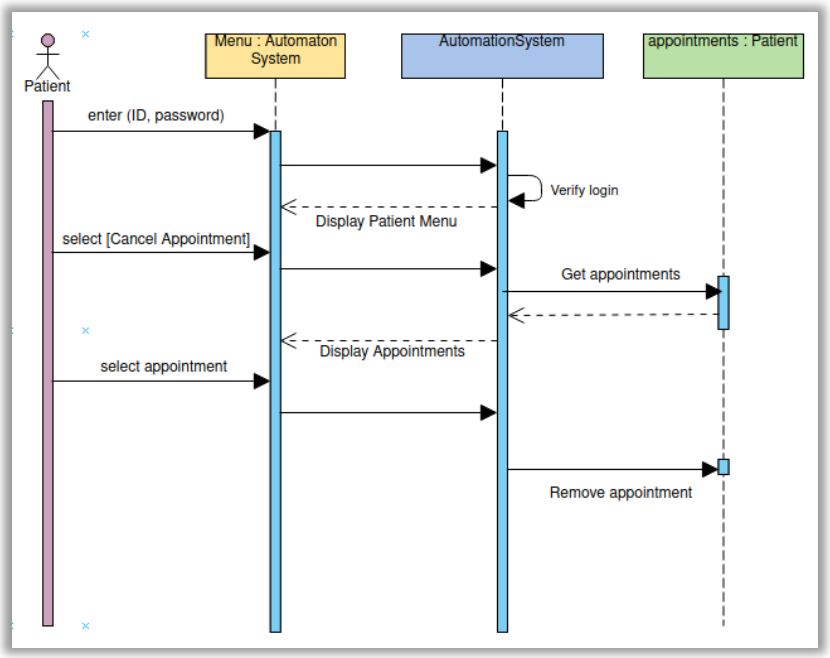
Add New Patient



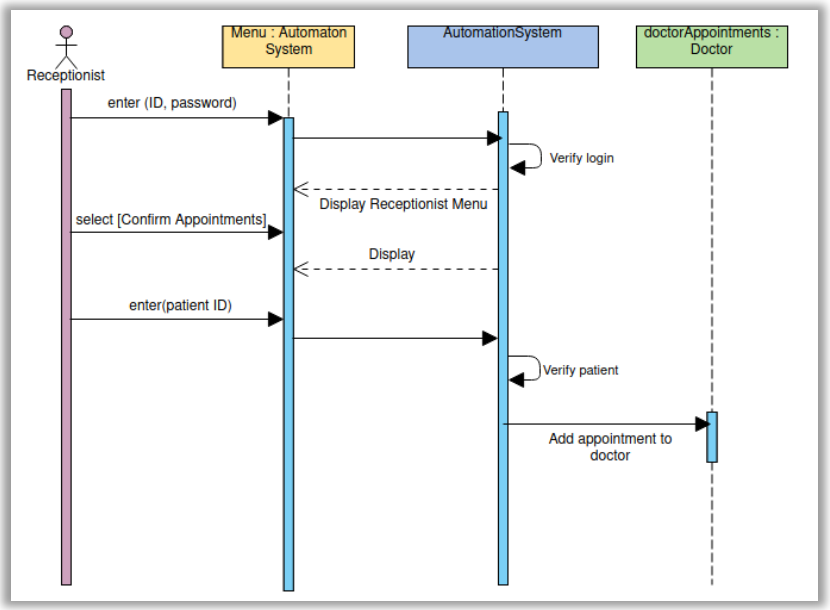
Get Appointment



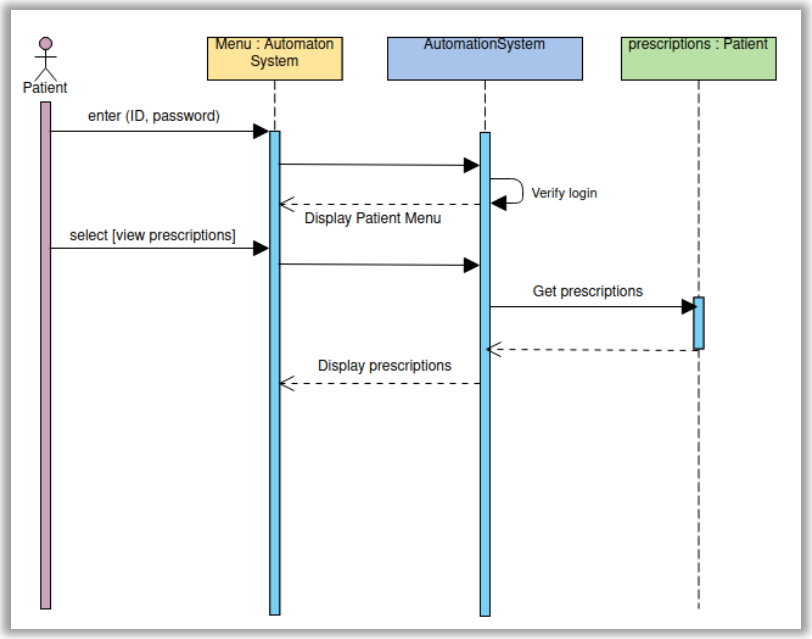
Cancel Appointment



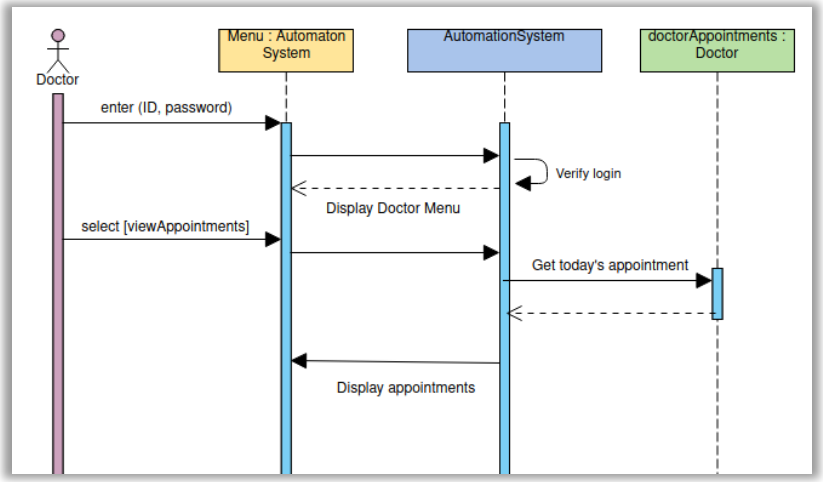
Confirm Appointment



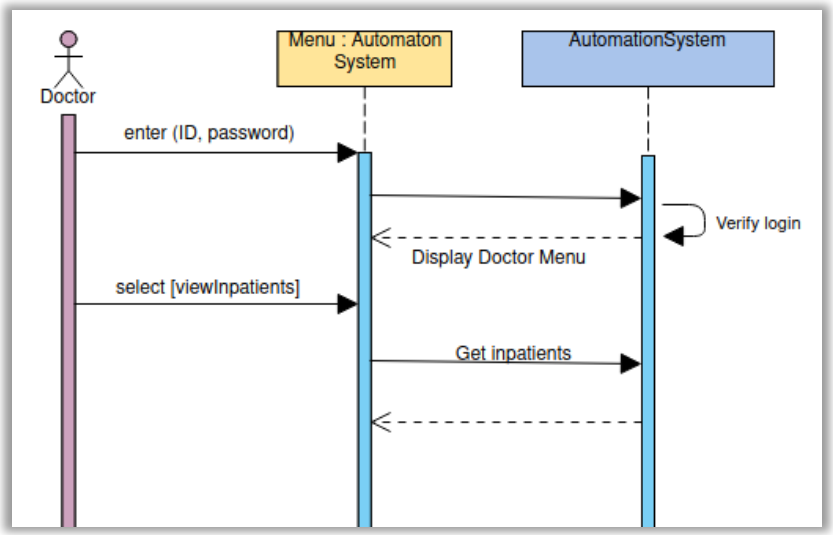
View Prescription



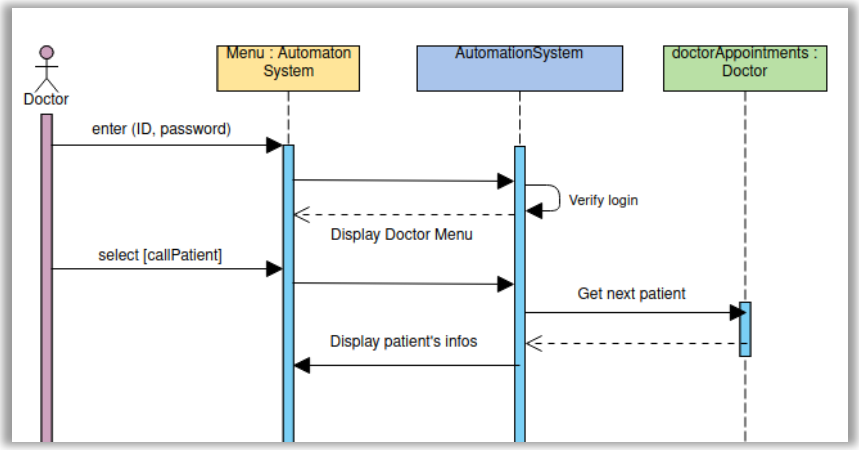
View Doctor's Daily Appointments



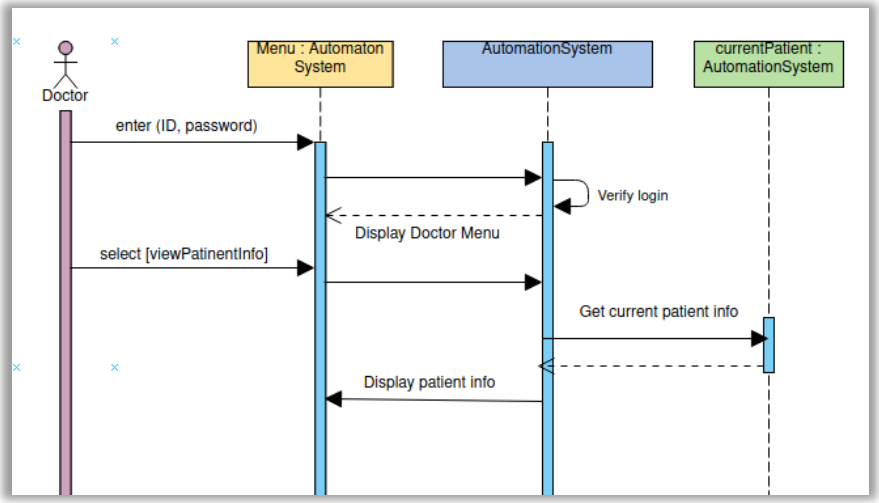
View Inpatients



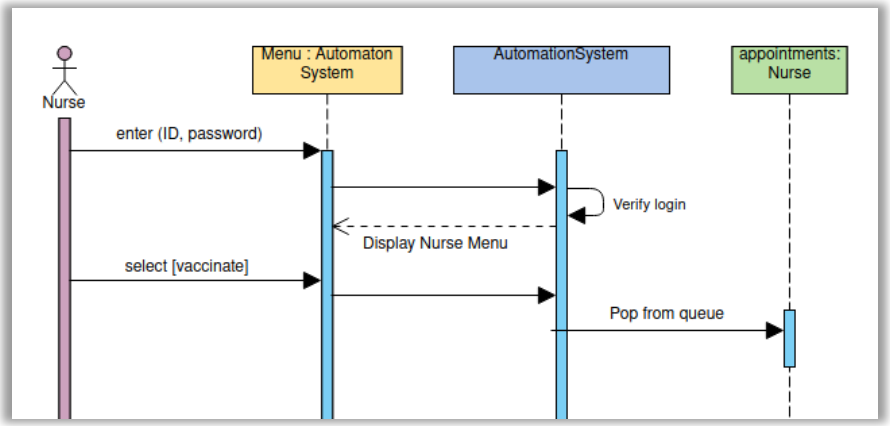
Call Patient



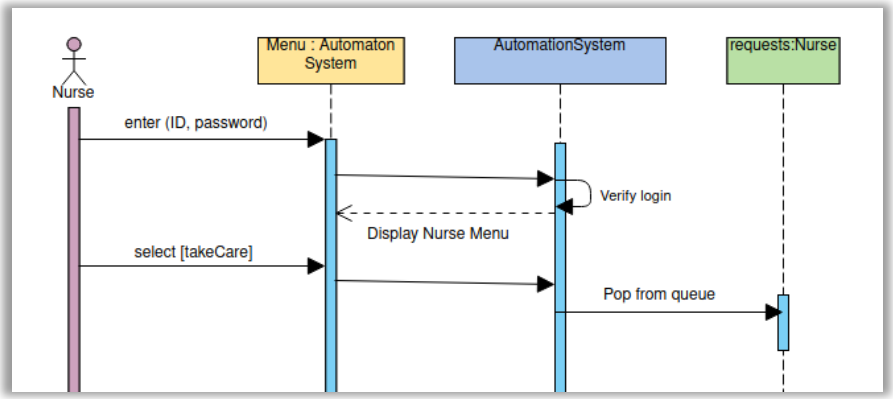
View Patient's Info



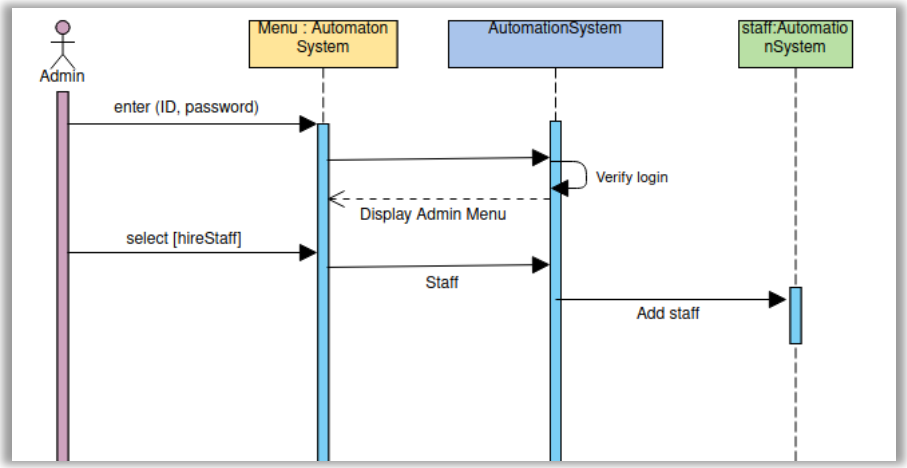
Vaccinate



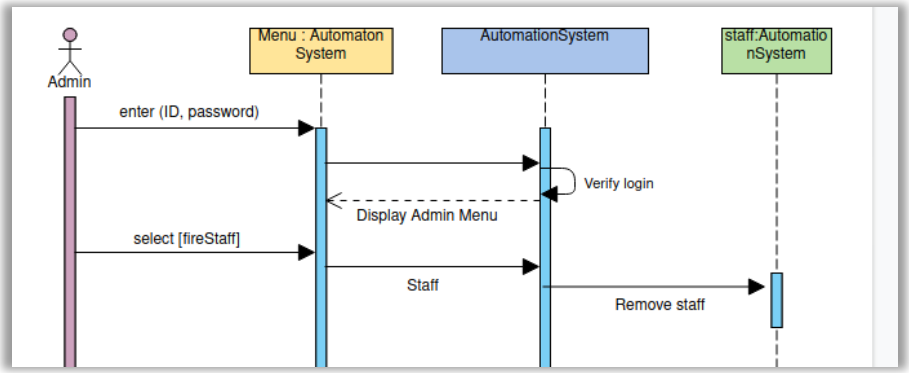
Patient Care



Hire Staff



Fire Staff



11.NON-TRIVIAL IMPLEMENTATION DETAILS

11.1. Binary Search Tree

add(E item) Method

This method adds the given element to the tree if it is not already present in the tree. If it is, throws an exception.

contains(E item) Method

This method checks if the given element is present in the tree. If it is, returns true and if it is not, returns false.

find(E item) Method

This method returns the given element in the tree if it is present in the tree. If it is not, returns null.

peek() Method

This method returns the element at the top of the tree.

getLeftSubtree() Method

This method returns the left child of the root node as a binary search tree.

getRightSubtree() Method

This method returns the right child of the root node as a binary search tree.

isLeaf() Method

This method returns true if the root node has no children, otherwise returns false.

11.2. Administrator

addBeds(int num) Method

Adds as many beds as specified to the dorm.

removeBeds(int num) Method

Removes as many beds as specified from the dorm.

viewAllStaff() Method

Prints a list of all the hospital staff with their info.

viewSpecificStaff(Staff staff) Method

Prints the info of specified staff

viewDormStatus() Method

Prints the occupancy rate of the dorm.

hireStaff(Staff staff) Method

Adds the given staff to the system.

fireStaff(Staff staff) Method

Removes the given staff from the system.

setVaccineAge(int age, HospitalManagementSystem system) Method

Sets the minimum age of patients who can be vaccinated.

11.3. Doctor

callPatient() Method

Pops the next patient from the priority queue.

viewAppointments() Method

Prints all the appointments of the doctor which are in the day that this method used.

viewPatientMedicalInfo(Patient patient) Method

Prints the given patient's all medical information and previous prescriptions.

viewPatientPrevAppointments(Patient patient) Method

Prints the given patient's previous appointments.

setPatientStatus(Patient patient) Method

Sets the status of the given patient based on whether he/she is discharged or hospitalized.

viewInpatients() Method

Prints a list of patients which are hospitalized.

clearSchedule() Method

Cancels all of the doctor's appointments on the day this method called.

11.4. Nurse

vaccinate() Methods

Vaccinates the next patient in the queue.

takeCare() Methods

Takes care the next patient in the queue.

11.5. Receptionist

newPatientRegistration(String name, String surname, String ID, String phoneNum, Hospital_Management_System system) Method

Adds a new patient to the system.

confirmAppointments(String ID, Hospital_Management_System system) Method

Confirms the appointment of the patient whose ID is the given ID and adds the appointment to the appointment priority queue of the doctor of the appointment.

11.6. Patient

addAppointment(Appointment appointment) Method

Adds the given appointment to the patient's appointment priority queue.

removeAppointment(Appointment appointment) Method

Removes the given appointment from the patient's appointment priority queue.

viewAppointments() Method

Prints a list of appoinments of the patient.

viewPrescriptions() Method

Prints a list of prescriptions of the patients.

11.7. Time Interval Calculator

timeInterval(LocalDateTime date1, LocalDateTime date2) Method

Calculates the time between given dates.

12.TEST CASES

Binary Search Tree

Test Scenario	Test Case	Pre Conditions	Test Step	Test Data	Expected Result	Actual Result	Status
Binary Search Tree Methods							
Check Add Functionality	Check response on adding an element that is already present in the tree	BinarySearchTree object must be constructed and 15 must be present in the tree	Call add(E item) method with an item that is present in the tree	Item: 15	add method should throw AlreadyInTreeException		
Check Add Functionality	Check response on adding an element that is not present in the tree	BinarySearchTree object must be constructed and 16 must not be present in the tree	Call add(E item) method with an item that is not present in the tree	Item: 16	Adding should be completed successfully		
Check Contains Functionality	Check response on checking if an element that is not present in the tree is present in the tree	BinarySearchTree object must be constructed and 17 must not be present in the tree	Call contains(E item) method with an item that is not present in the tree	Item: 17	contains method should return false		
Check Contains Functionality	Check response on checking if an element that is present in the tree is present in the tree	BinarySearchTree object must be constructed and 18 must be present in the tree	Call contains(E item) method with an item that is present in the tree	Item: 18	contains method should return true		
Check Find Functionality	Check response on finding an element that is not present in the tree	BinarySearchTree object must be constructed and 19 must not be present in the tree	Call find(E item) method with an item that is not present in the tree	Item: 19	find method should return null		
Check Find Functionality	Check response on finding an element that is present in the tree	BinarySearchTree object must be constructed and 20 must be present in the tree	Call find(E item) method with an item that is present in the tree	Item: 20	find method should return 20		
Check Peek Functionality	Check response on getting the peek element of an empty binary search tree	BinarySearchTree object must be constructed and must be empty	Call peek() method	Empty binary search tree	peek method should return null		
Check Peek Functionality	Check response on getting the peek element of a non-empty binary search tree	BinarySearchTree object must be constructed and must be non-empty	Call peek() method	Non-empty binary search tree	peek method should return the element at the top of the tree		
Check Get Left Subtree Functionality	Check response on getting the left child of the root which has left child as a binary search tree	BinarySearchTree object must be constructed and root node must have left child	Call getLeftSubtree() method	Root node with left child	getLeftSubtree method should return the left child of root node as a binary search tree		
Check Get Left Subtree Functionality	Check response on getting the left child of the root which does not have left child as a binary search tree	BinarySearchTree object must be constructed and root node must not have left child	Call getLeftSubtree() method	Root node without left child	getLeftSubtree method should return null		
Check Get Right Subtree Functionality	Check response on getting the rigth child of the root which has right child as a binary search tree	BinarySearchTree object must be constructed and root node must have right child	Call getRightSubtree() method	Root node with right child	getRightSubtree method should return the right child of root node as a binary search tree		
Check Get Right Subtree Functionality	Check response on getting the right child of the root which does not have right child as a binary search tree	BinarySearchTree object must be constructed and root node must not have right child	Call getRightSubtree() method	Root node without right child	getRightSubtree method should return null		
Check isLeaf Functionality	Check response on checking if the root of this tree which is a leaf is a leaf node	BinarySearchTree object must be constructed and root node must not have children	Call isLeaf() method	Root node without children	isLeaf method should return true		
Check isLeaf Functionality	Check response on checking if the root of this tree which is not a leaf is a leaf node	BinarySearchTree object must be constructed and root node must have at least one child	Call isLeaf() method	Root node with at least one child	isLeaf method should return false		
Check isLeaf Functionality	Check response on checking if the root of this tree which is null is a leaf node	BinarySearchTree object must be constructed and must be empty	Call isLeaf() method	Empty Binary Search Tree	isLeaf method should return false		
Check Remove Functionality	Check response on removing an element that is not present in the tree	BinarySearchTree object must be constructed and must 21 must not be present in the tree	Call remove(E item) method with item that is not present in the tree	Item: 21	remove method should return false		
Check Remove Functionality	Check response on removing an element that is present in the tree	BinarySearchTree object must be constructed and must 22 must be present in the tree	Call remove(E item) method with item that is present in the tree	Item: 22	remove method should return true		
Check Delete Functionality	Check response on deleting an element that is not present in the tree	BinarySearchTree object must be constructed and must 23 must not be present in the tree	Call delete(E item) method with item that is not present in the tree	Item: 23	delete method should return null		
Check Delete Functionality	Check response on deleting an element that is present in the tree	BinarySearchTree object must be constructed and must 24 must be present in the tree	Call delete(E item) method with item that is present in the tree	Item: 24	delete method should return 24		

Person

Test Scenario	Test Case	Pre Conditions	Test Step	Test Data	Expected Result	Actual Result	Status
Person Class Methods							
Check getName functionality	Check if the schedule is suitable	Person object must be constructed.	Call getName() method	(no parameter)	returns Person name in Person object as a string.		
Check getSurname functionality	Check if there is appointment	Person object must be constructed.	Call getSurname() method	(no parameter)	returns Person surname in Person object as a string.		
Check *getID* functionality	Check if there is appointment	Person object must be constructed.	Call getID() method	(no parameter)	returns Person ID in Person object as a string.		
Check *getPhoneNum* functionality	Check if there is prescriptions.	Person object must be constructed.	Call *getPhoneNum()* method	*(no parameter)*	returns Person phonenumber Person object as a string.		
Check *setName* functionality	Check if the method assign the given times integer into the relevant variable in Prescription object.	Person object must be constructed.	Call *setName()* method	*String name*	assigns the given parameter into String name variable in Person object.		
Check *setSurname* functionality	Check if there is appointment	Person object must be constructed.	Call setSurname() method	String Surname	assigns the given parameter into String surname variable in Person object.		
Check *setID* functionality	Check if there is prescriptions.	Person object must be constructed.	Call setID() method	*String ID*	assigns the given parameter into String ID variable in Person object.		
Check *setPhoneNum* functionality	Check if the method assign the given times integer into the relevant variable in Prescription object.	Person object must be constructed.	Call setPhoneNum() method	*String phoneNum*	assigns the given parameter into String phoneNum variable in Person object.		

Prescription

Test Scenario	Test Case	Pre Conditions	Test Step	Test Data	Expected Result	Actual Result	Status
Prescription Class Methods							
Check *getMedicine* functionality	Check if the method return medicine name as a String	Prescription object must be constructed.	Call getMedicine() method	(no parameter)	returns medicine name in Prescription object as a String.		
Check *getTimes* functionality	Check if the method return times variable as an Integer	Prescription object must be constructed.	Call getTimes() method	(no parameter)	returns the usage duration of the given medicine denoted as times in the Prescription object.		
Check *getMeasure* functionality	Check if the method return measure variable as an Integer	Prescription object must be constructed.	Call getMeasure() method	(no parameter)	returns the usage amount of the given medicine denoted as measure in the Prescription object.		
Check *setMedicine* functionality	Check if the method assign the given medicine name into the relevant variable in Prescription object.	Prescription object must be constructed.	Call *setMedicine()* method	*String medicine*	assigns the given parameter into *String medicine* variable in Prescription object.		
Check *setTimes* functionality	Check if the method assign the given times integer into the relevant variable in Prescription object.	Prescription object must be constructed.	Call *setTimes()* method	*int times*	assigns the given parameter into *int times* variable in Prescription object.		
Check *setMeasure* functionality	Check if the method assign the given measure integer into the relevant variable in Prescription object.	Prescription object must be constructed.	Call *setMeasure()* method	*int measure*	assigns the given parameter into *int measure* variable in Prescription object.		

Doctor

Test Scenario	Test Case	Pre Conditions	Test Step	Test Data	Expected Result	Actual Result	Status
Doctor Class Methods							
Check *callPatient* functionality	Check calling the next patient according to his / her id.	Patient object must be constructed.	Call callPatient() method	*(no parameter)*	The relevant patient is called according to the ID given from the registered patients.		
Check *viewAppointments* functionality	Check view all appointments	Patient object must be constructed.	Call viewAppointments() method	*(no parameter)*	All appointments are listed according to the doctor's ID		
Check *viewPatientMedicalInfo* functionality	Check view all health information for a specific patient	Patient object must be constructed.	Call viewPatientMedicalInfo() method	*PatientInterface patient*	All health information is displayed according to the ID of the patient.		
Check *viewPatientPrevAppointm* functionality	Check viewing the patient's previous appointments	Patient object must be constructed.	Call viewPatientPrevAppointments method	*PatientInterface patient*	All previous appointments according to the patient's ID are displayed		
Check *setPatientStatus* functionality	Check after the control of the patien the patient's condition was treated / changed as an inpatient	Patient object must be constructed.	Call setPatientStatus() method	*PatientInterface patient*	The patient's condition is changed by the doctor after the checkup.		
Check *viewInpatients* functionality	Check viewing inpatients	Patients object must be constructed.	Call viewInpatients() method	*(no parameter)*	All inpatients are listed		
Check *clearSchedule* functionality	Check deletion of all doctor's appointments	Schedule list must be constructed.	Call clearSchedule() method	*(no parameter)*	The appointments of the doctor are deleted.		
Check *compareTo* functionality			Call compareTo() method	*Doctor o*			

Nurse

Test Scenario	Test Case	Pre Conditions	Test Step	Test Data	Expected Result	Actual Result	Status
Nurse Class Methods							
Check setMeasure functionality	If there is someone waiting for vaccine.	Nurse object must be constructed.	Call vaccinate() method	(no parameter)	pop next person in appointments.		
Check setMeasure functionality	If there is no one waiting for vaccine	Nurse object must be constructed.	Call vaccinate() method	(no parameter)	throws exception		
Check takeCare functionality	If there is someone in request queue	Nurse object must be constructed.	Call takeCare() method	(no parameter)	pop next person in request queue		
Check takeCare functionality	If there is no one in request queue	Nurse object must be constructed.	Call takeCare() method	(no parameter)	throws exception		

Receptionist

Test Scenario	Test Case	Pre Conditions	Test Step	Test Data	Expected Result	Actual Result	Status
Receptionist Class Methods							
Check *newPatientRegistration* functionality	Check registering the new patient to the system	Patient object must be constructed.	Call newPatientRegistration() method	*String name, String surname, String ID,*	The patient is added to the system		
Check *confirmAppointments* functionality	Check confirmation of the appointments of patients who have made an online	Patient object must be constructed.	Call confirmAppointments() method	*String phoneNum, String ID, Hospital_Management_System*	The relevant patient is called according to the ID given from the registered		
Check compareTo functionality	appointment when they come		Call compareTo() method	*Doctor o*	patients.		

Patient

Test Scenario	Test Case	Pre Conditions	Test Step	Test Data	Expected Result	Actual Result	Status
Patient Class Methods							
Check *addAppointment* functionality	Check if the schedule is suitable	Patient object must be constructed.	Call addAppointment() method	(no parameter)	Add an appointment for patient		
Check addAppointment functionality	if schedule is not suitable	Patient object must be constructed.	Call removeAppointment () method	(no parameter)	throw exception		
Check removeAppointme functionality	Check if there is appointment	Patient object must be constructed.	Call removeAppointment () method	(no parameter)	Remove an appointment		
Check removeAppointme functionality	if there is no appointment	Patient object must be constructed.	Call removeAppointment () method	(no parameter)	throws exception		
Check *viewAppointments* functionality	Check if there is appointment	Patientobject must be constructed.	Call viewAppointments() method	(no parameter)	Listing current appointments		
Check *viewAppointments* functionality	if there is no appointment	Patientobject must be constructed.	Call viewAppointments() method	(no parameter)	throws exception		
Check *viewPrescriptions* functionality	Check if there is prescriptions.	Patient object must be constructed.	Call *viewPrescriptions()* method	*(no parameter)*	Viewing prescriptions		
Check *viewPrescriptions* functionality	if there is no prescriptions	Patient object must be constructed.	Call *viewPrescriptions()* method	*(no parameter)*	throws exception		

END OF THE REPORT

Last Update Date : 10th May, 2021 01:00 AM

GROUP
14

LECTURER
Prof. Dr. Fatih Erdoğan SEVİLGEN

TEACHING ASSISTANT
Başak KARAKAŞ
Muhammed Burak KOCA

KOCAELİ, 2021