BIRZEIT UNIVERSITY

Faculty of Engineering & Technology

Electrical & Computer Engineering Department

SMART MEDICINE WITH VITAL SIGNS MONITORING USING A SMARTWATCH APP

George

Ahmad

Ahmad Barhoum

Table of Contents

1. Phase One

2. Phase Two

2.1. Introduction

Our design's main focus is to help both users and doctors manage and market their health care services in a way that is integrated. The need for this system comes from the fact that traditional methods are hard for patients, doctors, and all employees because most center staff work by hand, which takes time, effort, and resources. Also, the system will keep an eye on vital signs and let emergency specialists know when they are needed. Also, most of us look up information online before making decisions these days, so a good healthcare system is just as important. Our software system will have a place for the answers to these and other problems.

By looking at the current system, we can see that the main problem with how healthcare providers and patients talk to each other is that a new patient can't book an appointment online. Instead, he has to go to the center and talk to a receptionist. After that, he has to give the employee all of his personal information, like his name, address, and email, so that they can fill out the reservation. But even with old patients, things don't get better. They have to make reservations over the phone or in person. This way of making reservations is hard, especially for employees, and it can lead to a lot of problems. Start with the large and frequent number of calls from patients who want to make an appointment, and try to imagine how the employee will be able to find time between all of the records to make this appointment. On the other hand, these calls can be stressful for employees because they can't keep track of and control the number of calls or manage their work well. Another problem is that data about patients are stored in hard-copy files like Excel sheets, which makes it likely that data will be lost or damaged. The process takes a long time, and there is no privacy involved.

By using our system, old data that was saved on hard copies like Excel sheets can be automatically turned into soft copies. This way, the old data can still be used without having to manually add it.

Appointments can be made by new or old patients without going to the center. Appointments can be made at any time on the website. After choosing the clinic and doctor, they will be able to change their minds. Before choosing a doctor, they will also be able to see all the educational information about each one. The patient must fill out a form with the necessary information, which is then put into a database that is connected to the system. When a patient's appointment time is getting close, our system will be able to send them a text message to their phone number. This way of making appointments will avoid conflicts and save time for both the patient and the receptionist.

When new patients or visitors come to the center or website for the first time, things will go smoothly. When you go to the website, you'll see a very short guide video at the top. It tells you a little about the center and how to make your first appointment. Finding the center location on the map, figuring out the distance from the current location, and other smart-related features will be added to the system.

Doctors will also benefit a lot from the system. They are lucky to have a machine learning program built into the system. This program lets them enter the symptoms the patient describes, and then the program predicts the patient's problem. This will be done on an online server or remote computer, so there is no need for a high-powered computer in the center. After each appointment, doctors can write the patient report directly in a field for that patient and add an attachment to it. This way, patients can always know what's going on with their health.

The system also has monitoring software that checks a person's breathing rate, BPM, and oxygen levels. This software is linked to a smartwatch that can measure these things. Then, a machine learning algorithm can be used to look at the data and make predictions about the patient's health. In case of an emergency, the smartwatch's gyroscope can have a fall detection feature that can send a message to the emergency team and doctors so that they can help the patient.

2.2. Functional Requirements

This subsection contains the requirements for the app, there are different categories of requirements (user/system) we will include in this section.

2.2.1. User Requirements

1. The software must provide a means of representing and accessing the data.
2. The user can add the doctor they'd like to be called if something strange happens.
3. The user can find out when doctors and other health care providers are open.
4. The system's user interface should be easy to use quickly and very well.
5. Each user should be able to log in to the system using their own private credentials, and each user should have a set number of permissions.

2.2.2. System Requirements

1. The system should have a way to send out an alert or SOS in case of an emergency.
2. The system should evaluate the pulse rate as well as the breathing rates, as well as any issues.
3. Every user should be able to make a profile for themselves.
4. For each profile, the system will keep track of history and store information.
5. The user should be able to look through all of the doctors and health care providers who are available.
6. By filling out a form, the patient can request and book an appointment online. The patient will then get a message saying that the form was sent successfully.
7. The user should be able to talk to doctors and other people who work in health care.

8. The user should be able to talk to doctors and other people who work in health care.
9. Doctors and other health care providers should be able to see the user's medical history.
10. The user should be able to choose who can see their own history and who can't.
11. The system should work on the Apple smartwatch with the right sensors, like an optical heart sensor and a breathing rate sensor.
12. When exhaustion is found, the user should get an alert.

2.3. Non-Functional Requirements

In this section, we'll talk about the requirements that don't have to do with function (NFR). NFRs describe the system's properties, like how secure, reliable, fast, easy to maintain, scalable, and easy to use it is. They act as limits on the design of the system across all of the different backlogs.

1. The system should be easy for a regular user to figure out how to use.
2. As the smartwatch doesn't have a lot of space, the software must be very slight, around 50 MB.
3. After turning on the car, it should take no more than 5 seconds to start the program.
4. When the user needs to take a break, the system must notify them.
5. The system should be protected and secure from any threats to data.

2.4. Software Development Process

The software needs to work for doctors, medical staff, and even patients, all of whom have different levels of technical knowledge. So, the design of healthcare software should meet the needs of its many users and be easy to use. When making this software system, many problems will need to be solved. Because of these problems, the agile development method is a great fit for software used in healthcare. The Agile development process lets smaller, more frequent releases be made, each with fewer requirements than the last. The benefit is that software can be made quickly, and different requirements can be reordered. Agile development also uses scrum, which is a great planning method because it is step-by-step. For example, in the first sprint, sketches will be used to show the customer an interactive prototype. This gives the customer a clear picture of the project's interface, so there won't be any problems at the sprint demo. Using it, the main parts of the system can be done in separate steps. This means that changes don't require a complete refactoring of software development schedules to account for new or shifting priorities, and all problems can be fixed continuously by looking back on the sprint. One of the best things about Agile is how it handles risks that were not expected. In this area, healthcare software can benefit from the iterative development cycles of agile. Unexpected risks can be dealt with quickly because Agile iterations are small and risks are minimized and manageable as part of each iteration.

3. Phase Three

3.1 Scenario Analysis

3.1. Scenario for reserving an appointment (George)
Initial assumption: The user wants to reserve an appointment using the system
Normal: The user will first choose the "Reserve Appointment" option. The system will then show a list of doctors and their specialties. The user can then choose the best time and day for the appointment based on when the doctor is free. Then, the actor moves on to the payment page, where the patient can choose to pay online or with cash. If the patient wants to pay with cash, they can do so when they meet at the hospital or clinic, or they can enter their information and pay online.
Alternative: If the user wants to get rid of an appointment, they can look at all of them and decide which one to get rid of by looking at the information next to each one. If the user wants to change the doctor for a given appointment, they can keep the same date and time but change the doctor if it works with the doctor's schedule. If the user wants to change the date and time of the appointment but keep the same doctor, they can move the appointment to a new date and time while keeping the same doctor if that works with the doctor's schedule.
Error: If you want to pay with a credit card and the credit card does not have enough money to make the reservation, the system will display the message "Payment failed, the card does not have enough money."

3.2. Description of the Use Cases

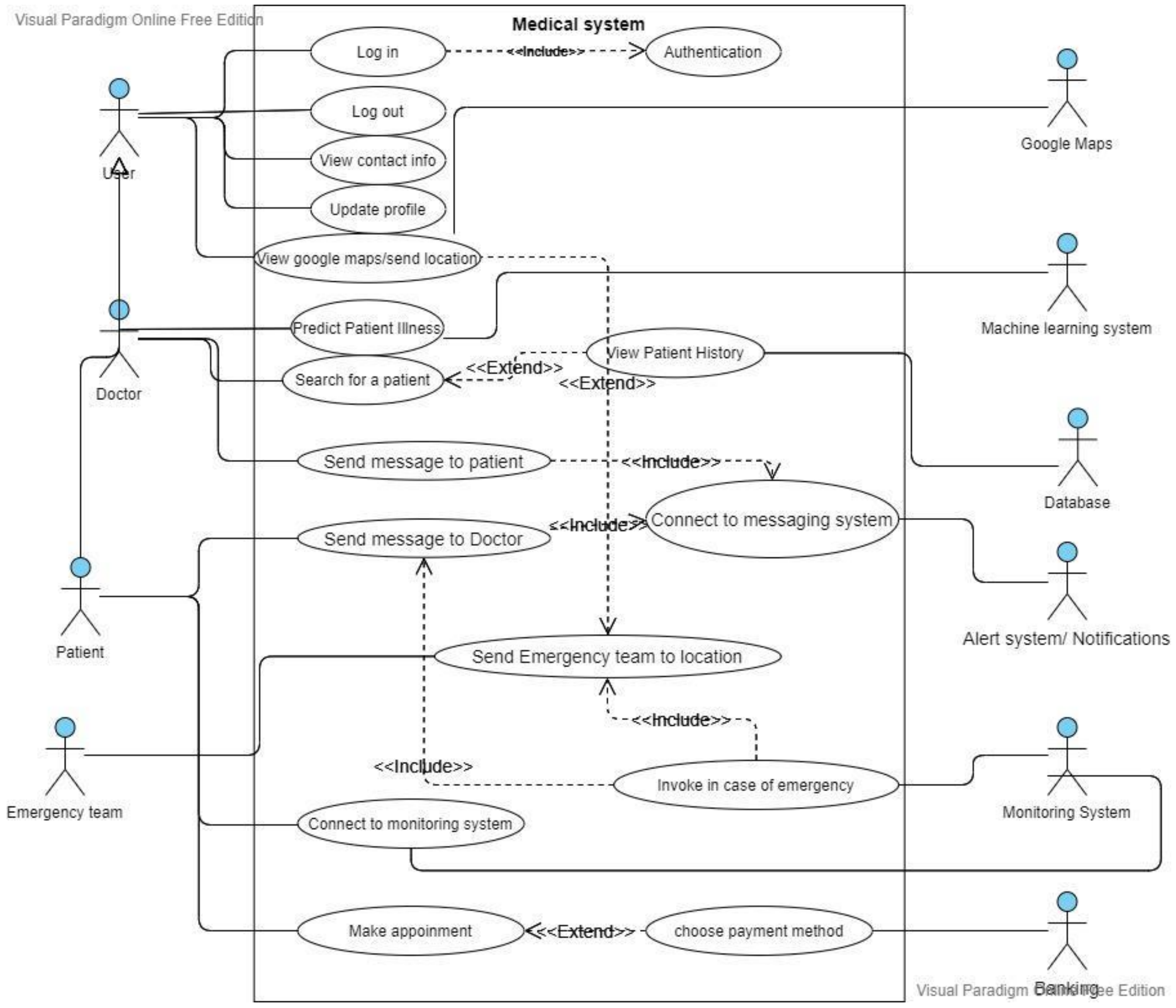


Figure 1: Use Case for the system

1. Login: Users will be able to log in and use the system as they see fit.
2. Logout: Users should be able to sign out of the system when they're done using it.
3. Update profile: The system should let users change their own information, like their username, password, ID number, date of birth, ID number, and address.
4. View map/send location: The system will let users use Google Maps to look at the map, and it will also automatically give directions.
5. Authentication: After a user logs in, the system should give them a different kind of authentication based on their job title in the database system.
6. Get notified: The system should let both the patient and the doctor know when there are changes.
7. Predict a patient's illness: The doctor should be able to put a list of the patient's symptoms into a machine learning system that will try to figure out what's wrong with the patient.

8. Find a patient: This use case will make it possible for doctors to look for a patient. One way to do this is to look at the patient's medical history.
9. View patient History: This use case shows the patient's medical history. It includes information like the patient's full name, age, and contact information, as well as a list of previous appointments with details about each one.
10. Send a message to the doctor: The patient should be able to send a message or notification to the doctor at any time, or the monitoring system should be able to call the doctor in an emergency and send a message to the doctor.
11. Connect to the messaging system: Using the notifications system, the system should be able to put patients in touch with their doctors.
12. Invoke in case of emergency: If there is an emergency, the system will send a message to the doctor and send the emergency team to the location that is marked on Google Maps.
13. Make an appointment: The system must let the receptionist or patient book, cancel, or change an appointment.

3.3. Use Case Specifications

3.3.1. Login Use Case specification

This Use case Scenario is done by Ahmad

3.3.1.1. Brief Description

Users will be able to log in and use the system as they see fit. All users are the ones in this use case.

3.3.1.2. Flow Actions

This use case can be used when the user selects option ‘Login’ option

3.3.1.2.1. Basic flow - Successful Login

1. The user fills out the login form and clicks "login." The form has the user's username and password.
2. The system takes the information and checks it against the database system, which has all the information about all users. This information includes the username, full name, password, email, date of birth, id number, whether or not the person is an employee, and the address.
3. If the user is signed in, the system shows the user interface, and the user should be able to use the system.

3.3.1.2.2. Alternative Flows

Describes a use case scenario that is different from the basic flow and helps a user reach his or her goal.

3.3.1.2.2.1. New User

1. If the user has never used the system before, he or she chooses "sign up."
2. The system shows a form with text fields for the user's username, full name, password, email address, date of birth, whether or not the person is an employee, id number, and address.
3. The user fills out the form and chooses "enter."
4. The system checks that the information is correct by making sure that none of the text fields are blank, that the email is unique and has a "@," and that the ID number is also unique.
5. The system adds the information to the database and checks the user's identity.
6. The system should be easy for the user to use.

7. When making a new account, the employee field will be set to NO, (patient), since the system already has accounts for other actors.
8. If there's a new employee, HR can add his account to the system as an employee (doctor, receptionist, etc...)

3.3.1.2.2.2. Wrong Password

1. if the user enters the wrong password
2. The system tells the user "Your password is not correct" and clears out the password field. It also gives the user three more chances to get it right.
3. If the user chooses the right one or alternative flow, we go back to the basic flow. 2.2.3 If he didn't remember his password

3.3.1.2.2.3. Forgot Your Password

1. if the user forgets his password, he can select the ‘Forgot your password? Option.
2. The system asks the user for his email address. The user types in his email address and clicks the "send confirmation number" button.
3. The system shows a text field for the confirmation number and sends the confirmation number to the email.
4. The user types in the number, click "reset the password," and then types in his new password. The system replaces the old password in the database with the new one.
5. The system goes to the main login screen, where the user should be able to enter the username and the new password. 6. The system verifies the user's identity, and only then should the user be able to use the system.

3.3.1.2.2.4. Account Already Exists

In the new user flow, the system checks to see if the ID number entered is already in the database. If it is, it tells the user that their account already exists by sending them a message.

3.3.1.3. Special requirements

1. It should be easy to log in, and the system should show the main user interface in less than one second.
2. It must take less than 1.5 seconds for the database to send and receive data.
3. It should take less than 2 seconds to send the confirmation number to reset the password. 4. There are some rules about the password. The password must be longer than 7 characters and contain at least one capital letter and two numbers.

3.3.1.4. Entry Conditions

There are no preconditions

3.3.1.5. Exit Conditions

There are no postconditions

3.3.2. Reserve an appointment Use Case specification

This Use case Scenario/Description is done by George

1. Actors

The Actors in this use case are the Patients.

2. Brief Description

The Actor should be able to reserve an appointment, edit an appointment and cancel an appointment

3. Flow actions

3.1 Basic Flow: Adding an appointment

1-The actor chooses "book an appointment."
2-A list of doctors is shown by the system.
3-The list has the doctor's name and the field he works in.
4-The actor chooses which doctor he wants to see.
5-The actor chose "view schedule."
6-The System shows the actor's timetable for the doctor they chose.
7: The doctor's schedule shows when he is available and how much it will cost to see him.
8-The actor can only choose times when the doctor is available.
8-The actor picks one of the times that the doctor is available.
9-The Actor chooses to keep paying.
10-The Actor is given a list of ways to pay by the System.
11: You can use a credit card or cash.
12-The actor picks "Credit Card" from the menu.
13: The actor is asked to put in his card details.
14: The information includes his first and last name, his card number, the card's CVV number, and the date it will expire.
15: The performer clicks "confirm."
16: That much money should be taken out of his bank account.
17: The system shows a message that says, "Your reservation went through."
18: For each reservation, steps 2–14 are done again.

3.2 Alternative flows

3.2.1 Deleting an appointment:

1-The actor clicks on "View my appointments."
2-The system shows a list of the next appointments for that patient.
3-Next to each appointment on that list is some information.
4-The information is the appointment date, the name of the doctor, and whether or not the person wants to delete or change the appointment.
5-The Actor chooses "cancel appointment" from the list of options.
6-A confirmation message, "Are you sure you want to cancel this appointment?" appears on the screen.
7- The Actor chooses "yes"
8- The system gets rid of the appointment.
9- The System sends the actor's bank account 70% of the amount paid.
10-Steps 2 through 8 are done again for every appointment that is taken out of the system.

3.3.2 Editing the doctor in an appointment

1-The actor clicks on "View my appointments."
2-The system shows a list of the next appointments for that patient.
3-Next to each appointment on that list is some information.
4-The information includes the date of the appointment, the name of the doctor, how much it will cost, and whether or not the person wants to delete or change the appointment.
5-The Actor chooses the option to "edit the appointment."
6-The system shows the user a list of the information they can change.
7: The information is about the doctor and the date of the appointment.
8: The actor chooses to change the doctor.
9: The system shows a list of doctors and the fields in which they work.
10-The actor chooses the doctor he wants to work with.
11: The system tells the actor when the doctor's appointments are and how much they will cost.
12-The actor picks a time from the schedule for an appointment.
13-The system shows the difference between the old appointment cost and the new appointment cost.
14: The performer clicks "Apply Changes."
15-The system shows a message asking if you are sure you want to change the appointment.
16: The Actor chooses "yes."
17: The system sends or takes from the actor's bank account the difference between the old and new appointments.
18: If the edited appointment costs more, the actor will be charged money, and if it costs less, the actor will get money back. If they both cost the same, the actor will not be charged or given any money back.
19: The System saves the appointment that was changed and gets rid of the old one.

3.3.3 Editing the time in an appointment:

1-The actor clicks on "View my appointments."
2-The system shows a list of the next appointments for that patient.
3-Next to each appointment on that list is some information.
4-The information is the appointment date, the name of the doctor, and whether or not the person wants to delete or change the appointment.
5-The Actor chooses the option to "edit the appointment."
6-The system shows the user a list of the information they can change.
7: The information is "The doctor, Date of reservation."
8: The actor chooses to change the date of the reservation.
9: The System shows the actor that he has an appointment on the doctor's calendar.
10-The actor picks the new time for the appointment.
11: The performer clicks "Apply Changes."
12-The System shows a message asking if you're sure you want to change this appointment.
13: The actor chooses "yes."
14: The changed appointment is saved by the system.

3.3.4 The patient doesn’t have enough balance in his bank account:

In the "Adding an appointment" flow, if you choose to pay with a credit card but the card doesn't have enough money to make a reservation, the system will show the message "Payment failed, the card doesn't have enough money."

3.3.5 The patient chooses to pay with cash:

In step 12 of the "Adding an appointment" flow, the actor chooses "Cash." The system then shows the message "You will need to pay at the Center." The actor then chooses "Confirm," and the system shows the message "Your reservation was successful."

4. Special requirements

1-When someone wants to see a list of doctors, the system should be able to get that list in less than 0.1 seconds.
2-If you want to know when a doctor is available. The system should be able to show any doctor's schedule in less than 0.5 seconds.

5. Entry conditions

5.1 Log in

The Actor must be logged in to the system to reserve

6. Exit conditions

6.1 Apply changes to the database

The changes should be made to the database, and the system should schedule

6.2 Bank account changes

If a transaction takes place, the amount of money will be taken out of or put into the actor's bank account.

Class Diagram

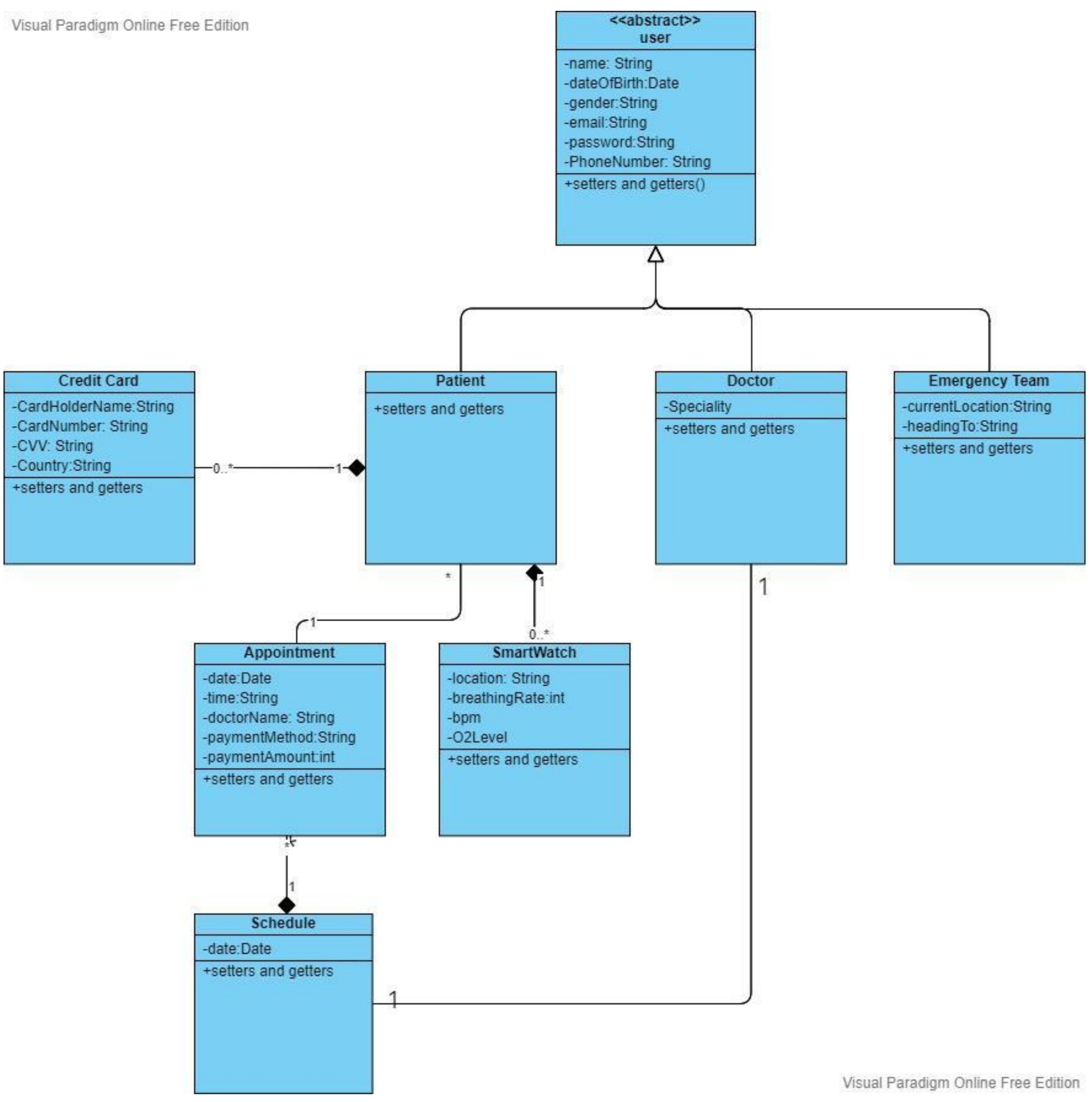


Figure 2 Class diagram

State Diagram for the object: Appointment (George)

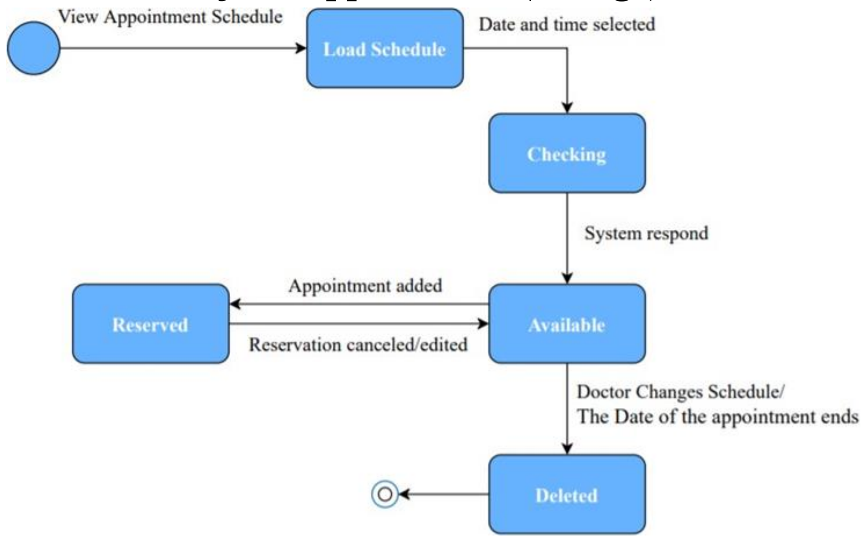


Figure 3 state diagram

This activity diagram is done by George

The following activity diagram shows the flow of actions with the system behavior of the Reserving an appointment Use case. The actor is the Patient.

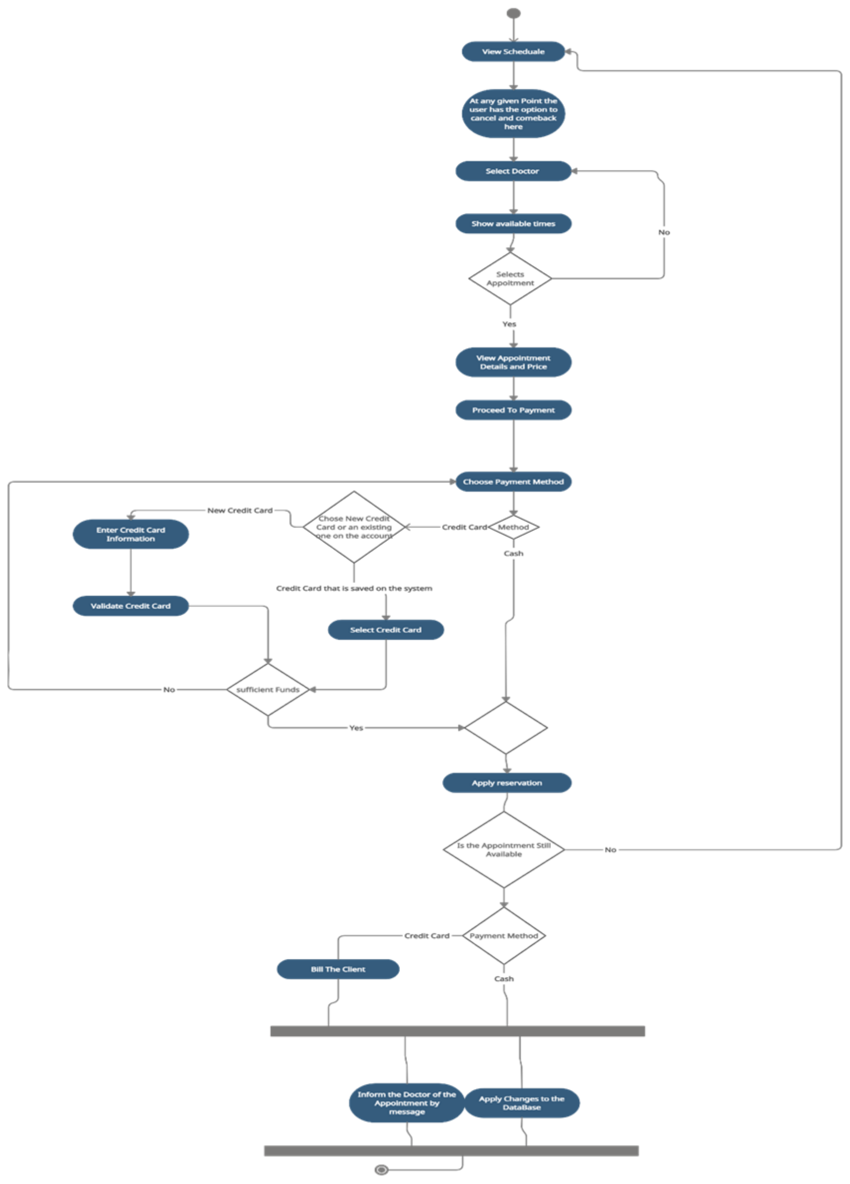

View Scheduale
At any given Point the user has the option to cancel and comeback here
Select Doctor
Show available times
Selects Appoitment
No
Yes
View Appointment Details and Price
Proceed To Payment
Choose Payment Method
Method
Credit Card
Cash
Chose New Credit Card or an existing one on the account
New Credit Card
Credit Card that is saved on the system
Enter Credit Card Information
Validate Credit Card
Select Credit Card
sufficient Funds
No
Yes
Apply reservation
Is the Appointment Still Available
No
Payment Method
Credit Card
Cash
Bill The Client
Inform the Doctor of the Appointment by message
Apply Changes to the DataBase

Figure 4: Activity diagram

This activity diagram is done by Ahmad

The following activity diagram shows the flow of actions with the system behavior of the login use case. The actor s the user

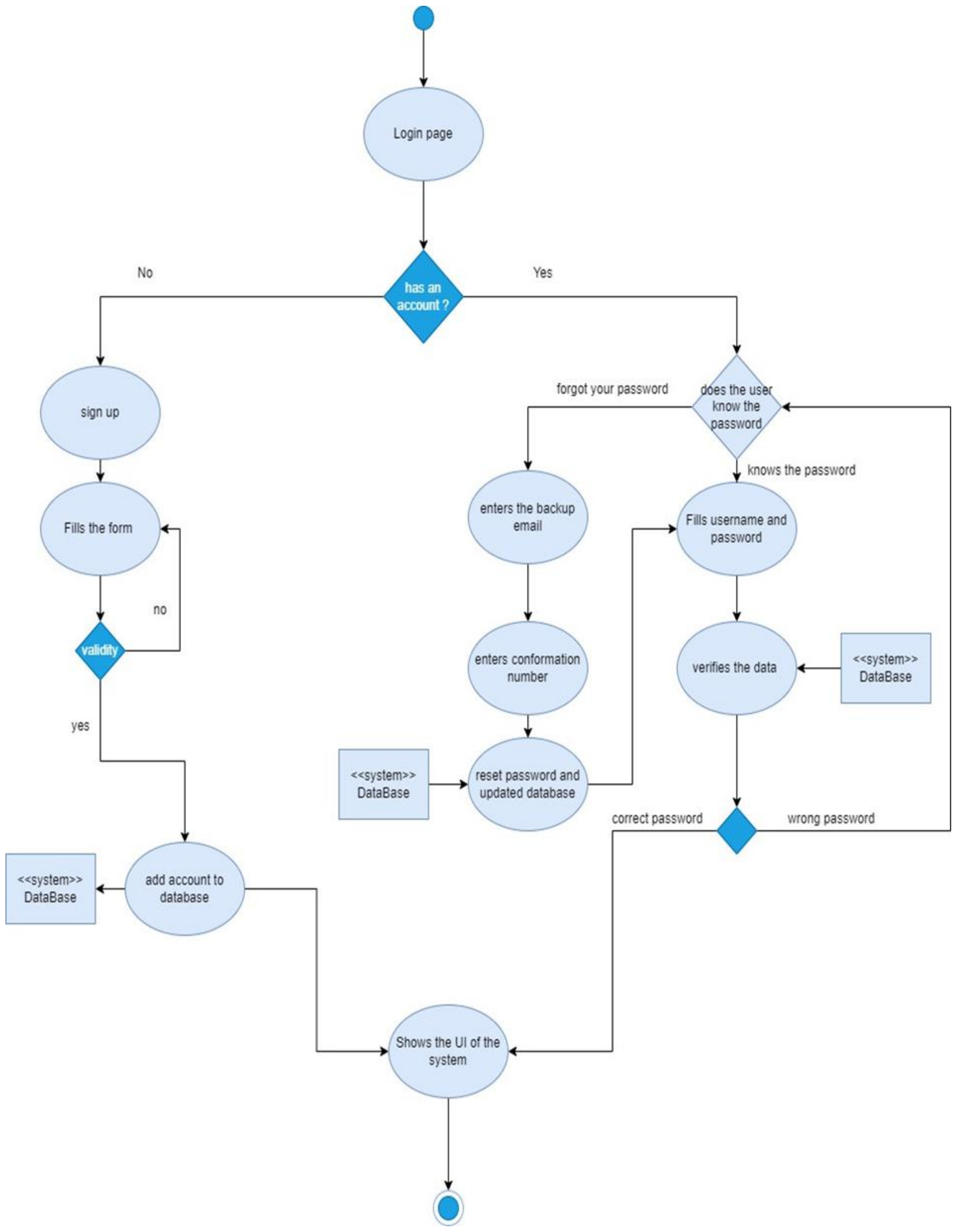


Figure 5: Activity diagram

3. 3. Reserving An appointment sequence diagram By George

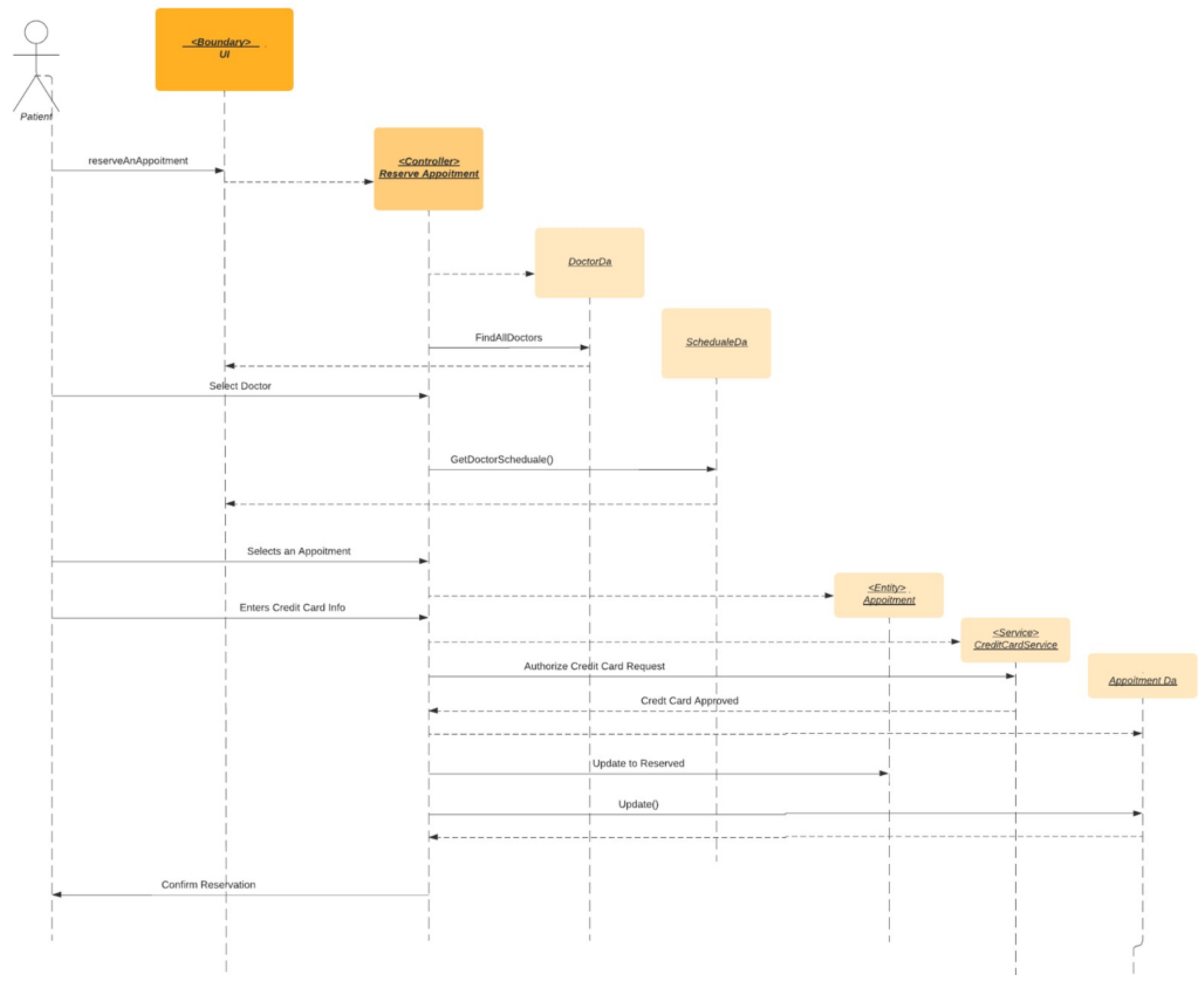


Figure 6: sequence diagram

Basic Flow for reserving an appointment

1-The Actor selects “reserve appointment”.
2-The System Displays a List of doctors.
3-The list has the name of the doctor and his field.
4-The Actor selects the doctor he wants.
5-The Actor select “view schedule”
6-The System displays the schedule of the doctor that the actor chose.
7-The schedule of the doctor indicates when he is available and shows the appointment cost.
8-The actor can only select times when the doctor is available.
8-The Actor selects one of the available times of the doctor.
9-The Actor selects to continue to pay.
10-The System gives the Actor a list of options to pay with.
11-Those options are Credit Card, Cash.
12-The actor selects “Credit Card”.
13-The actor is asked to enter his card information.
14-The information is his First and last name, card number, card CVV, and card expiry date.
15-The actor selects “confirm”.

16-The amount of money should be withdrawn from his bank account.
17-The System displays a message “Your reservation is successful”.
18-Steps 2-14 are repeated for each reservation.

1. This Sequence diagram describes the login (new user) use case, Done by Ahmad

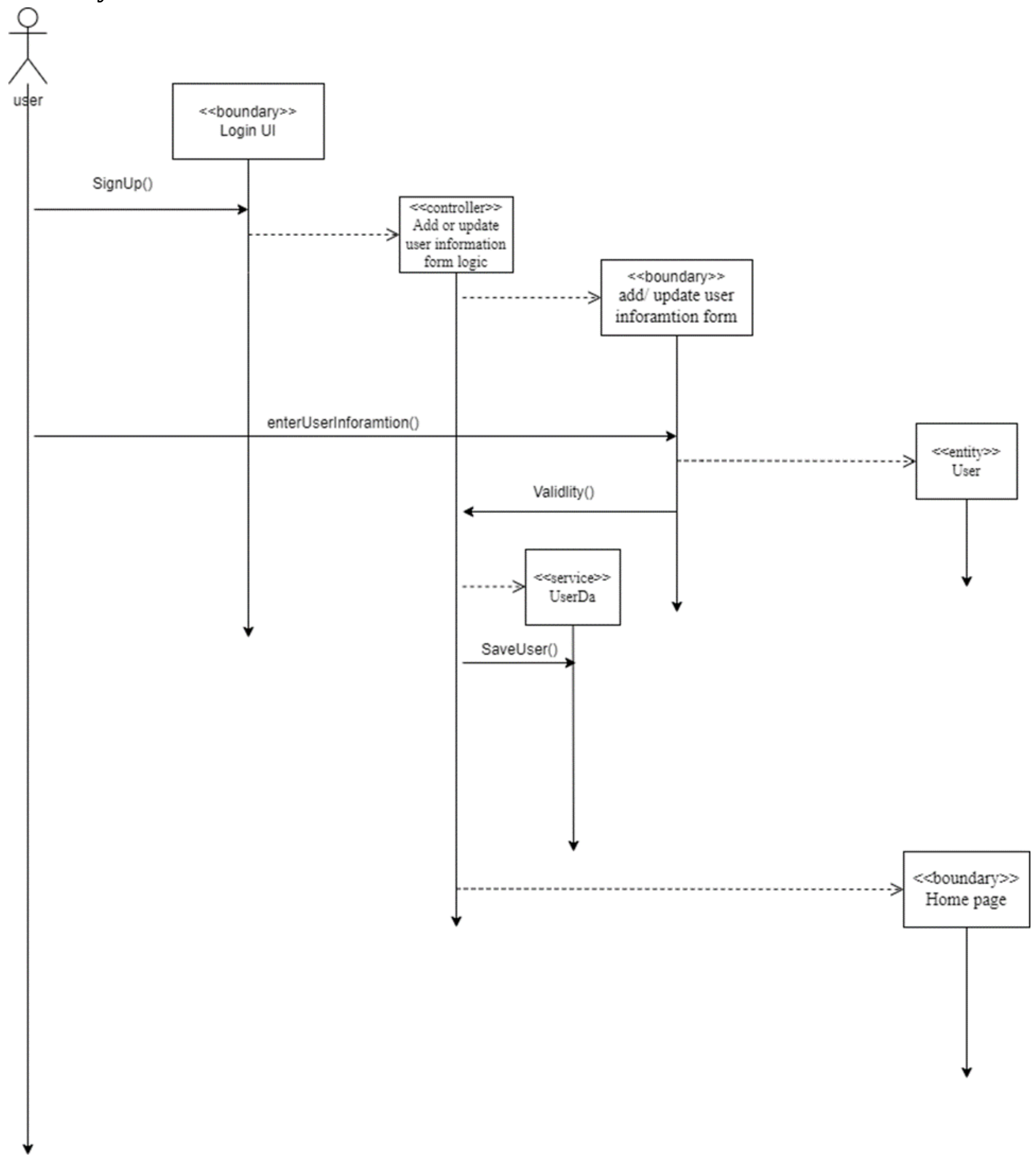


Figure 7: sequence diagram

New user

1. if the user is new to the system, he selects the ‘sign up’ option

2. the system shows a form that contains the following text fields: Username, full name, password, email, date of birth, if its an employee or not, id number, and the address

3. the user fills the form and selects the ‘enter’ option

4. The system checks the validity of the information that all text fields are not empty and the email should be unique and has an ‘@’ and also the id number must be unique

5. the system adds the data to the database system and verifies the user

6. the user should be able to use the system

7. when making a new account is employee field is going to be NO,(patient) since other actors' accounts are pre-defined in the system

8. if there is a new employee HR can add his account to the system as an employee(doctor, receptionist, etc...)

Predicting health and illness using machine learning (Ahmad Barhoum 1183231)

1. Machine learning system have access to all the data all the time.
2. A doctor can see the results of the analysis done by the system for every patient.
3. The analysis result is sent at the end of each month to the patient (on the 28th of each month).
4. Doctors and patients can access the analysis for the previous months.
5. The system sends 3 types of alerts based upon how urgent the case is. So an email (blue case): a report for the patient will be sent if the result of the analysis is at low risk.
   (yellow case): This is the middle case. A report will be sent to the patient and to the last doctor he visited, demonstrating the analysis and prediction.
   (Red case): in this case, the system predicts that the patient is at a high risk area, so an email will be sent to the last 3 doctors he visited and to the patient asking the user to make an appointment with the doctor.
   Alerts will be sent every 5 hours to the patient until he makes the appointment.
6. If the patient do not make an appointment after 4 alerts sent to him, system will alert the emergency team to take the suitable solution by them.
7. A doctor can check the previous history of the patient to see if the prediction made by the automated system is correct.

Use Case diagram for the machine learning system:

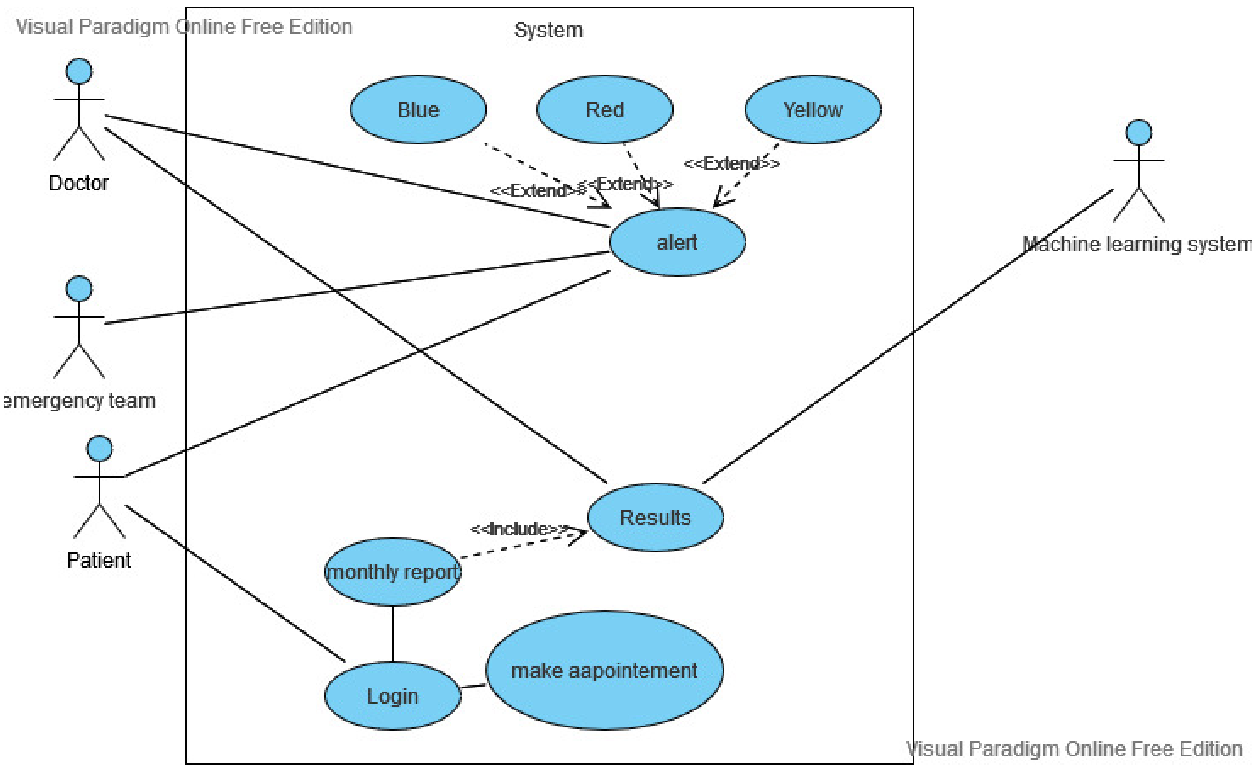
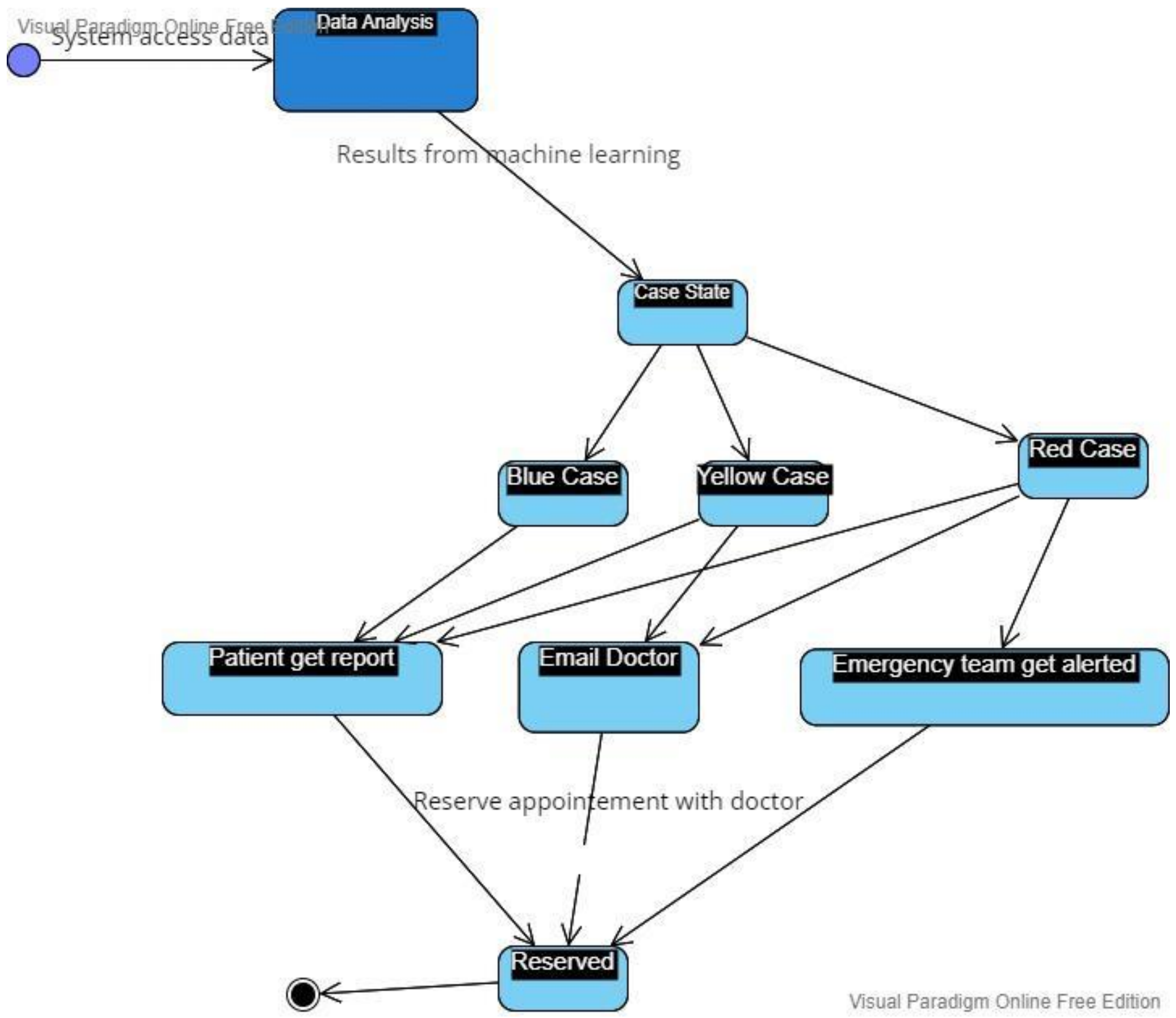


Figure 8: use case diagram for machine learning system

Scenario for red alert by machine learning system: (Author Ahmad Barhoum 1183231)
Initial assumption: the machine learning analysis results predict a high risk case for the patient.
Normal: at first, when the system recognizes it is a high risk case based on predescribed medical conditions, then the system will send an alert to the 3 last doctors the patient visited and the patient.
The alert will be sent to the patient every 5 hours until the patient makes an appointment with any doctor to check more for his case.
Alternative: if a patient does not recognize the alerts by notification, an email will be sent to him to make him aware of his situation and encourage him to make an appointment with the doctor.
An alert will be sent to the emergency team if the patient does not respond after 4 alerts displaying to them an alert "the patient X has a high risk medical condition and did not respond to previous alerts sent to him by system, Please check with him", in order to take the suitable solution with the patient.
Error: If the system does not send an alert, the monthly report will include all the information and the situation of the patient, if he is at high risk or not.

State Diagram (Ahmad Barhoum 1183231)


Visual Paradigm Online Free Edition
System access data
Data Analysis
Results from machine learning
Case State
Blue Case
Yellow Case
Red Case
Patient get report
Email Doctor
Emergency team get alerted
Reserve appointement with doctor
Reserved
Visual Paradigm Online Free Edition
Figure 9: state diagram for machine learning system

Sequence diagram :(Ahmad Barhoum)

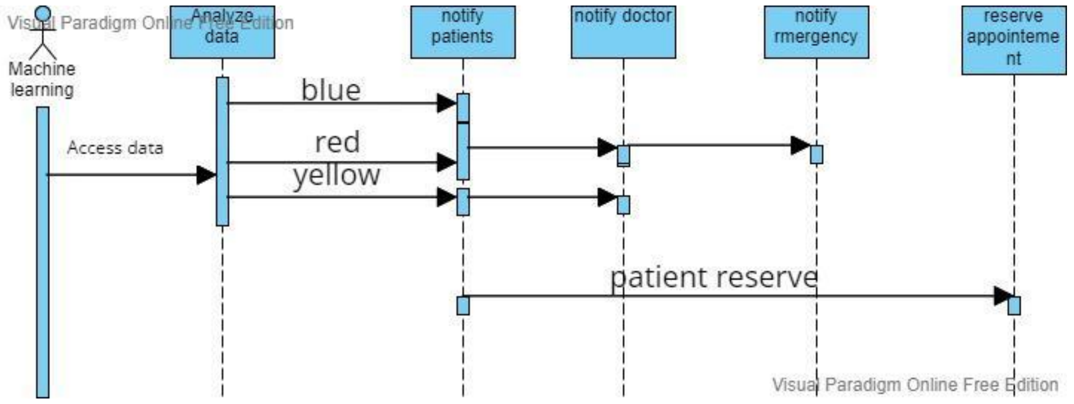


figure 10: sequence diagram

Activity Diagram (Ahmad Barhoum)

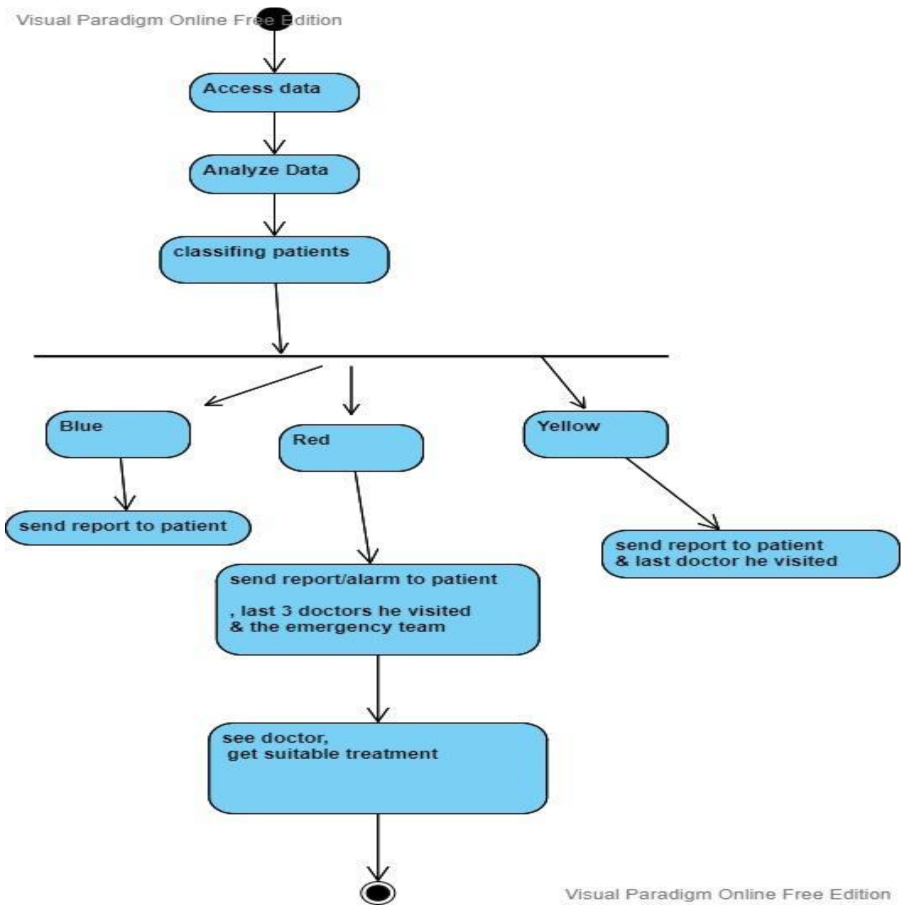


figure 11: activity diagram

4. Phase Four: System Modeling and Design

4.1 System Design Goals

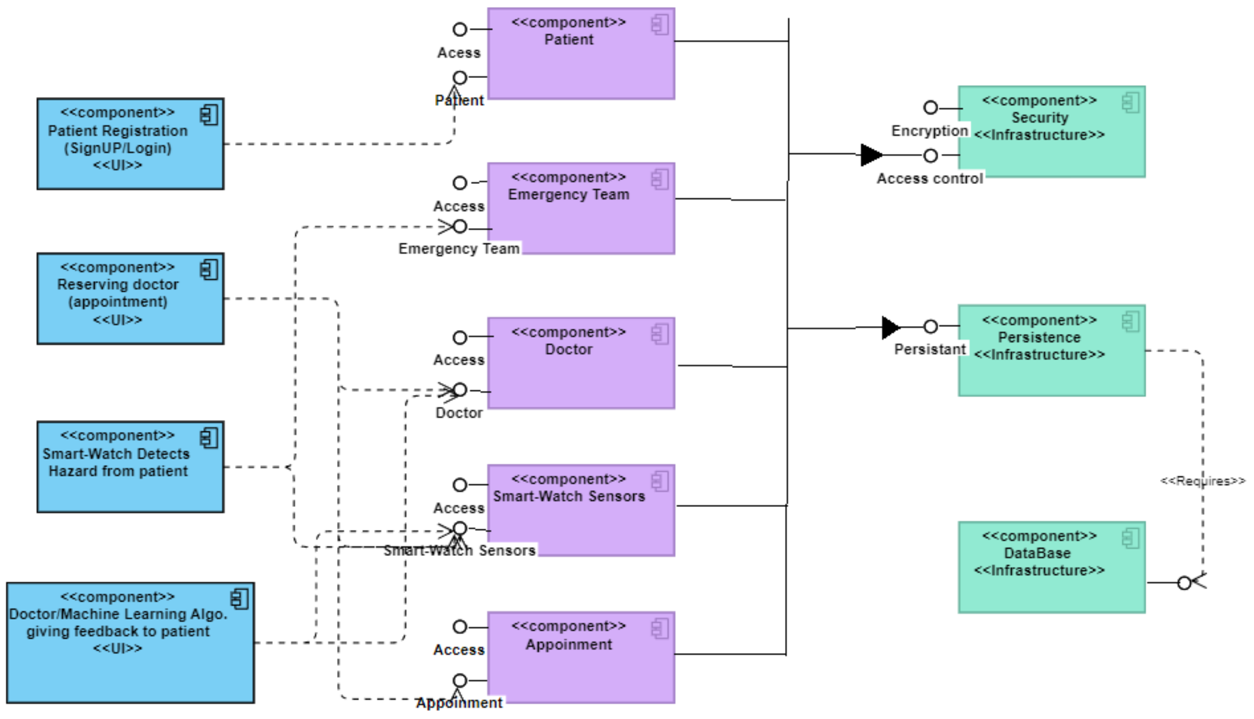
Two general design goals:
1-**High Cohesion**:
The classes that have functionality that are highly connected, so we can put them together to achieve high cohesion. .
2- **Low coupling**:
Our system will rely on models that are self-contained, meaning that any modification in one component will have little or no impact on the others.

One specific system design goal:
User-friendliness:

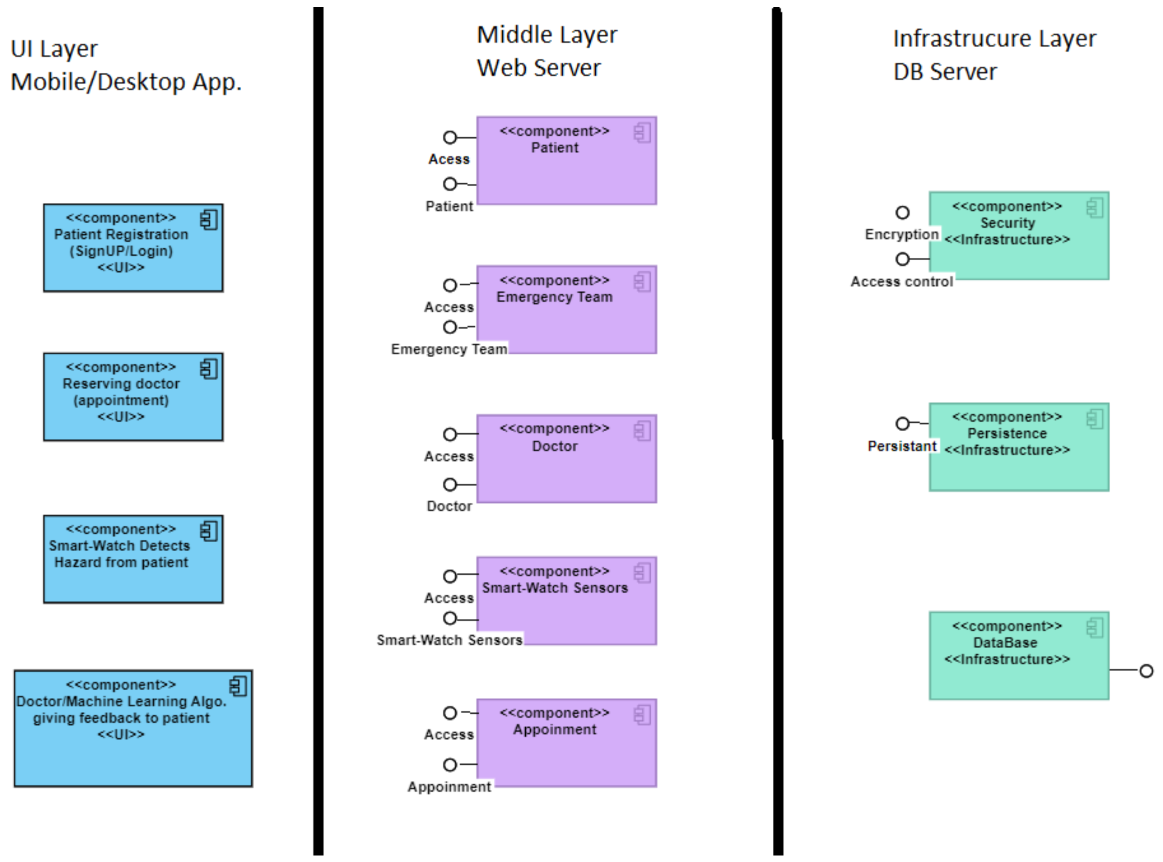
The system should be easy for a regular user to figure out how to use. When it comes to user-friendliness, the most important factor is an easily comprehensible and quick-to-operate website.

Security:
The system deals with personal data so any leaks will not be tolerated and the system has to be resistant to attacks

4.2 System and Component modeling



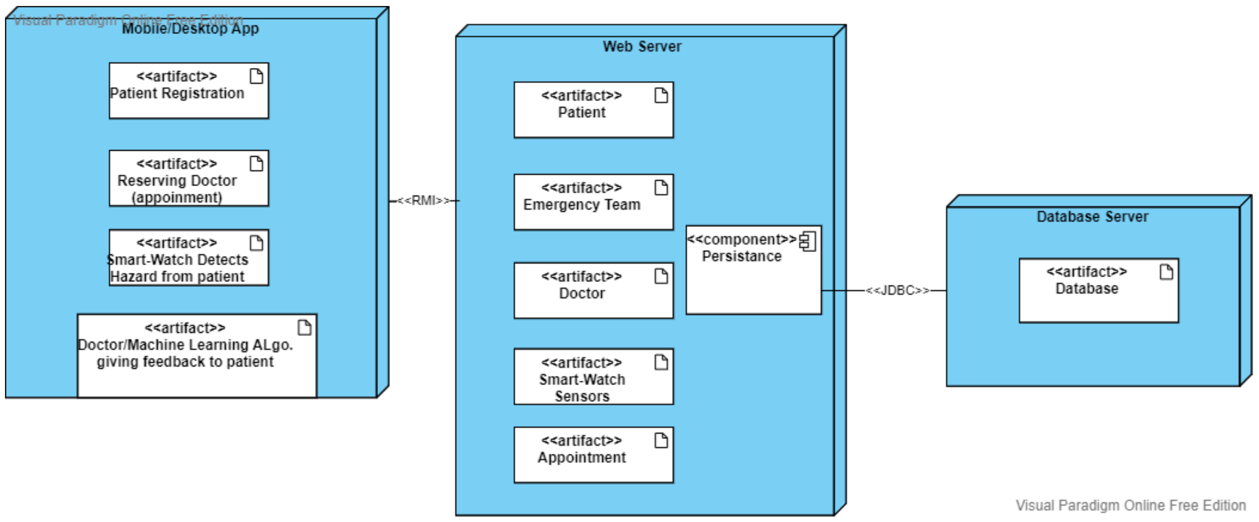
4.3 System and Architectural Design


UI Layer
Mobile/Desktop App.
<<component>> Patient Registration (SignUP/Login) <<UI>>
<<component>> Reserving doctor (appointment) <<UI>>
<<component>> Smart-Watch Detects Hazard from patient
<<component>> Doctor/Machine Learning Algo. giving feedback to patient <<UI>>
Middle Layer
Web Server
<<component>> Patient
Acess
Patient
<<component>> Emergency Team
Access
Emergency Team
<<component>> Doctor
Access
Doctor
<<component>> Smart-Watch Sensors
Access
Smart-Watch Sensors
<<component>> Appoinment
Access
Appoinment
Infrastrucure Layer
DB Server
<<component>> Security <<Infrastructure>>
Encryption
Access control
<<component>> Persistence <<Infrastructure>>
Persistant
<<component>> DataBase <<Infrastructure>>

Architecture refers to the process of establishing a set of hardware and software components, as well as their interfaces, in order to lay the groundwork for the creation of a computer system. As a result, we choose to employ the layered approach.
Because it separates the little elements to deliver a complete function, architecture to operate the program with low coupling and high cohesiveness

4.4 System and Deployment modeling

Deployment diagram


Visual Paradigm Online Free Edition
Mobile/Desktop App
<<artifact>> Patient Registration
<<artifact>> Reserving Doctor (appoinment)
<<artifact>> Smart-Watch Detects Hazard from patient
<<artifact>> Doctor/Machine Learning ALgo. giving feedback to patient
<<RMI>>
Web Server
<<artifact>> Patient
<<artifact>> Emergency Team
<<artifact>> Doctor
<<artifact>> Smart-Watch Sensors
<<artifact>> Appointment
<<component>> Persistance
<<JDBC>>
Database Server
<<artifact>> Database
Visual Paradigm Online Free Edition