

**Computer Engineering**

**Software Engineering Course - CEN 302**

Authors :

Jaser Sokoli

Alvi Lika

Neritan Hoxha  
Rigels Hita

Ibrahim Pisha  
Kris Ferko

*This page intentionally left blank*

**Table of Contents**

[1.Executive Summary 3](#_Toc510459586)

[1.1 Project Overview 3](#_Toc510459587)

[1.2 Purpose and Scope of this Specification 3](#_Toc510459588)

[2. Product/Service Description 4](#_Toc510459589)

[2.1 Product Context 4](#_Toc510459590)

[2.2 User Characteristics 4](#_Toc510459591)

[2.3 Assumptions 4](#_Toc510459592)

[2.4 Constraints 4](#_Toc510459593)

[2.5 Dependencies 5](#_Toc510459594)

[3. Requirements 5](#_Toc510459595)

[3.1 Functional Requirements 5](#_Toc510459596)

[3.2 Non-Functional Requirements 7](#_Toc510459597)

[3.2.1 User Interface Requirements 7](#_Toc510459598)

[3.2.2 Usability 7](#_Toc510459599)

[3.2.3 Performance 7](#_Toc510459600)

[Capacity 7](#_Toc510459601)

[Availability 8](#_Toc510459602)

[Latency 8](#_Toc510459603)

[3.2.4 Manageability/Maintainability 8](#_Toc510459604)

[3.2.4.1 Monitoring 8](#_Toc510459605)

[3.2.4.2 Maintenance 8](#_Toc510459606)

[3.2.4.3 Operations 8](#_Toc510459607)

[3.2.5 System Interface/Integration 8](#_Toc510459608)

[Network and Hardware Interfaces 9](#_Toc510459609)

[Systems Interfaces 9](#_Toc510459610)

[3.2.6 Security 9](#_Toc510459611)

[Protection 9](#_Toc510459612)

[Authorization and Authentication 9](#_Toc510459613)

[3.2.7 Data Management 9](#_Toc510459614)

[3.2.8 Standards Compliance 10](#_Toc510459615)

[3.2.9 Portability 10](#_Toc510459616)

[3.3 Domain Requirements 10](#_Toc510459617)

4. Software Design / Diagrams

4.1 User Scenarios

4.2 User Scenarios Extended

4.3 User Cases

4.4 User Cases Diagrams

4.5 Activity Diagrams

4.6 State Diagrams

4.7 Collaboration Diagrams

4.8 Sequence Diagrams

4.9 Class Diagrams

4.10 DB Schema Diagram

4.11 ER Diagram

4.12 Component Diagram

4.13 Object Diagram

4.15 Deployment Diagram

5. Implementation Technology

6.Project Planning

APPENDIX

# 1.Executive Summary

## Project Overview

This project conforms with a simple application which is a Web-Based and consists as a simplifier management system for doctor visits . One of the main usage of our project is to leave an appointment online and keep track of records for each patient in a online form for a period of time and stored and connected to your health card instead of actual use , on paper.

In order to finish the idea we need to cowork with actual city clinics so we can't miss any detail. Family doctors will be able to leave appointmens for their patients to forward medication if their state is more serios , for example to the 'Specialist Doctor' or even 'the hospital'. They will be able to leave notes for each patient and their medical recors or their specific details.

Appointments can be cancelt by the patiens through the email with a automated response . An display of his timetable will we offered besides the one of Hospitals' doctors or Specialist doctor which he uses to book his patients visits or further treatment.

All this data can be also accesed for urgent cases because his personal and medical datas are displayed there . We intend to wipe of queue so the public can have simpler ,faster and better healthcare .

## Purpose and Scope of this Specification

The purpose of this specification is to assess the current state of the product design and to document the entire process based on design issues and the audience.

This specification encompasses several aspects of the process being discussed in an as broad scope as possible. Thus in this scope we address the following:

* In depth documentation of the features of the product
* Technical overview of the system processes and views
  + This is discussed in Part 2.1 and throughout the document
* User and System Requirements
* Components & Functional/non-functional requirements
  + These are discussed in Part 3 in some detail
* Definition of users’ means of using and accessing the product
  + Use cases/scenarios discussed in Part 4
* Dependencies and Constraints
  + These are discussed in Part 2.4/5 of the Document

Aspects not included in the scope are as follows:

* Legislative requirements for the product
* Auditing and financial considerations of the product

# Product/Service Description

Long queues of people in medical institutions in Elbasan is a prevalent problem which requires an immediate solution. The Compentent Public bodies for giving this services in Elbasan are:

”The house doctor”, ”Ambulance” and the ”Hospital”.

It operates on a hierarchical structure starting from “House doctor” until the hospital.

“MMS”(Medical Management System) will aim to reduce the obstacles faced by the patients of this institution and improve the data-maintenance process. It will be used by "Bashkia Elbasan" to efficiently avoid the queues on providing the health service for the patients.

It should help in the operations within the institution like: keeping the track of the patients examinations history; dynamic calendar of doctors and easily accesable from the patients.

The “house doctor”will be able to schedue an appoiment online using this facility by watching on the doctors calendar for empty places In his calencar. No log in needed for the clients but all the data must be stored on their unique health insurance number for example(id:1234).

All the doctors must have their log ins where they can see their calendar and in the end of the examination they will be able to leave a feedback or a recipt wich can also be printable.

## Product Context

Our software is directly related to the "Elbasan Hospital". It is concepted to be an independent system that will make the direct connection between the client and the patient.

## User Characteristics

There are four types of users that will interact with the system :

* patients ,
* doctors ,
* admin
* secretary

1. Patients will able to book a free time slot with a doctor without login, but just by using his unique insurance number. He can use this health card number to check all his examinations he has ever done in the medical system.
2. Doctors will have their own accounts given by the admin also they will have a time Slot Management using Calendar, Writing Client Report on occupied time slot.
3. Secretary will have her own account given by admin and Can forcefully occupy and cancel doctor time slot.
4. Administrator manages the staff management system. He can add a new staff member or remove one. He randomly generates a password and sends it confidentially to worker. The worker itself can reset the password.

## Assumptions

## It is assumed that some actions performed behind the scenes are performed regularly according to law. Therefore users under Drejtoria e Informacionit, who confirm the assistance, according to “Vendimit te Keshillit te Ministrave Nr.787 date 14.12.2005”. Therefore a client must have a health insurance card. For security issues, it is taken for granted that after administrator assigns a worker to a directory, he send confidentially by mail the random generated password.

## Constraints

The project is constrained by the Internet connection. Since the application fetches data from the database over the Internet, it is crucial that there is stable Internet connection for the application to function.For this project since it is a web based application we have decited to use Pure PHP .This application will be apple to work on different Operating Systems for example on Windows 7,8,10 , Windows XP, Linux,Max etc.

## Dependencies

In order for this web application to work we must frequently gain access on the health insurance card’s database.

# Requirements

## Functional Requirements

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Req#** | **Requirement** | **Comments** | **Priority** | **Date Rvwd** | **SME Reviewed / Approved** |
| R\_0\_1 | Different level users according to their position in the administrate | According to their position, each will have their own profile | 1 | 02/04/2018 | Jaser Sokoli |
| R\_0\_2 | Privileges according to their position. | - | 1 | 02/04/2018 | Jaser Sokoli |
| R\_0\_3 | Exchange of information | Information between levels of hierarchy  . | 2 | 02/04/2018 | Jaser Sokoli |
| R\_0\_4 | Limited access of information | The information will be visible only to the profile of intented user.  . | 1 | 02/04/2018 | Jaser Sokoli |
| R\_0\_5 | Interactivity between the unit and the doctors on duty. | Giving tasks and reporting in real time | 1 | 02/04/2018 | Jaser Sokoli |
| R\_0\_6 | Interactivity with the patients. | The patients can report with the help of the android app | 1 | 02/04/2018 | Jaser Sokoli |
| R\_0\_7 | Reporting to a higher hierarchical level. | - | 1 | 02/04/2018 | Jaser Sokoli |
| R\_0\_8 | Sharing the information restrictively. | The information shared will be higly protective. | 1 | 02/04/2018 | Jaser Sokoli |
| R\_0\_9 | Task problems | If a visit is not completed or is facing difficulties(problems) they should report to the higher up organization and give reason. | 1 | 02/04/2018 | Jaser Sokoli |
| R\_1\_0 | Flow of information | Flow of information between different levels of the hierarchy, directly or indirectly. | 1 | 02/04/2018 | Jaser Sokoli |
| R\_1\_1 | Tasks between levels of hierarchy | Assigning tasks to different level of hierarchy to the lower tier level of this hierarchy. | 1 | 02/04/2018 | Jaser Sokoli |
| R\_1\_2 | Detailed report after terminating tasks | Doctor is required to provide a detailed report to the visit, in order to fill a detailed recipt | 2 | 02/04/2018 | Jaser Sokoli |

## Non-Functional Requirements

### User Interface Requirements

1. Different screen resolutions and apperance based on devices
2. Receive native push notifications in real time
3. Sliding navigation drawer for the product
4. Static navigation drawer for the web page
5. Responsive simple design

### *Usability*

* Accessibility
  + The software shall be easy to access remotely and at all times, since both patients and doctors will use the application on their devices.
  + This software shall be easy to access with only your health card number as an indentification protocoll .

* Responsiveness
  + The software shall be responsive both in data transactions, especially because of the reliance on the servers .

* Flexibility
  + The software shall be easy to update in order to accommodate new requirements like new doctors or secretaries or administrators.
  + The software shall be designed in such a way that allows us to make changes without starting from the beginning.

* Effectiveness
  + The software shall provide both staff and clients with practical tools of managing their data and also with a convenient way of communicating their needs across a platform that avoids wasted time on ques of people.

* Efficiency

* Performance Requirements: The performance of the software is going to depend on the server used. JavaScript and PHP will be used which means that it will be light weighted and the browser won’t crash.
* Space Requirements: We are striving to build a good, strong server. It will be able to support a number of simultaneous users.
* As the software is supposed to perform a country size scale, the servers that will be used at least 1 Terabyte of memory, fast, preferable with SSD, and the operating system is thought to be Linux Ubuntu.
* The central server, which is located to the Central General Directory, which will operate and access the information from all subdirectories will require at least 100 Terabyte of memory.
* As the software is planned to be fast, there should an internet connection of at least 20 dedicated Megabytes of internet, both in download and upload .

### *Performance*

#### Capacity

The backend is built on top of Google’s infrastructure and thus scales very well horizontally.

* Database writes are limited to 2300 per second
* Software starting time will be in no time , other procedures will take time accoring to user internet strength
* Maximum concurrent connections for mobile/web clients are limited to 100000 per database.
* The project is web base so everythings is connected to web server
* Maximum API request size is 8-10 MB
* Maximum number of documents that can be passed to a Commit operation in a transaction is 500
* Maximum number of composite indexes for a database is 200
* Maximum function call depth is 15-20

#### Availability

* The app will be live 24/7
* Is has a very low probability of downtime,
* It will be region independent, but available in Albanian only , and in English the Home page.
* Impact of downtime will be very minimal around 0.05%, considering the high reliability of the Google infrastructure.
* Its availability is measured by its performance when a subsystem fails, its ability to resume service in a state close to the state of the system at the time of the original failure, and its ability to perform other service-affecting tasks (such as software upgrade or configuration changes) is a manner that eliminates or minimizes down time.

### *Manageability/Maintainability*

#### *Monitoring*

The system will be subject to periodic evaluation by our Tester and our Supervisor and This evaluation will be performed by measuring the error logs generated by our product and sent to use automatically.

For us to be able to fix those errors , we shall be able to make some procedures with prompts and validations of our data .

#### Maintenance

We are going to design in a Modular way our product so the administrator or we can perform and evident operations for maintenance . So the system is isolatet and easy maintained.

#### Operations

* Reporting in real time.
* Sharing info restrictively.
* Backup and recovery information.
* Assigning tasks

### System Interface/Integration

Specify the use of other required products (e.g., a database or operating system), and interfaces with other systems (e.g., UWHires package interfaces with PubCookie and ODS, HEPPS system interfaces with Budget system). For each interface, define the interface in terms of message format and content. For well-documented interfaces, simply provide a reference to the documentation.

Outline each interface between the product and the hardware or network components of the system. This includes configuration characteristics (e.g., number of ports, instruction sets), what devices are to be supported, and protocols (e.g., signal handshake protocols).

#### Network and Hardware Interfaces

The app will use either Wi-Fi, or mobile data to connect to the internet. Other network related issues are automatically handled by Firebase Infrastructure, including connection monitoring, operation queueing during offline periods, etc.

#### Systems Interfaces

The users and doctors will be able to authenticate using the following methods, but the app uses a unique identifier which is not affected by the possibly different sign in method.

* classic username and password
* email and password
* Gmail account
* Facebook
* Twitter
* Phone number

The signing of the consent will be done electronically, complying with all legislative regulations, according the specified template by the clinic.

### *Security*

* Specify the factors that will protect the system from malicious or accidental access, modification, disclosure, destruction, or misuse. For example:
* encryption
* activity logging, historical data sets
* restrictions on intermodal communications
* data integrity checks
* Security provided to the access of user private information.
* Security provided to access of data.
* Restriction provided to access of data.
* There should also be provided a term of agreement to the user, in order to access their information and the data posted to the server.
* The android application should also include a list of modules and components that will be required to be accessed from the application, as known as permissions

### *Data Management*

The datas in this product are managed only by the administrator and they include user personal datas and their records in medical files.

We can use the " Cloud Firestore" products,in which differently from the Realtime Database, data is stored in Documents and Collections, thus offering better query support, and a offline first approach.

### *Standards Compliance*

Only Legislative Requirements:

* Accounting Requirements: Verification with Id card no.
* Safety/Security Requirements: System encrypted.

### *Portability*

* Use of Firebase products, which offer flexibility in different platforms, such as: Android, iOS, Node.js, Java, Python and GO
* Real time updates

## Domain Requirements

*Everything related to the domain that might be needed in the project shall be mentioned in here. Sometimes the domain Requirements might be thought as part of either functional or non-functional requirements.*

**4. Software Designs/Diagrams**

4.1 User Scenarios

**User**

|  |  |  |
| --- | --- | --- |
| Nr. | User Story Name | Description |
| 1. | Pacient identification | Using the card id and selecting a date the user si offered a list of available time period to book a visit |
| 2. | Pacient selecting the timetable for the visits | After the list is showned the user selects the preferred time period |
| 3. | Pacient booking the visit | The user is promted to enter an optional email, to confirm the visit and if the email is provided the details are sent in the email |
| 4. | Pacient cancels a visit from email | If the email is provided, the user can cancel the visit directly |
| 5. | Pacient call the hospital to cancel the visit | If the user doesn’t have an email, calls the hospital to cancel the visit |

**Admin**

|  |  |  |
| --- | --- | --- |
| Nr. | User Story Name | Description |
| 1. | Admin log in | Using a password and an email the admin access the system |
| 2. | Create hospital entity | By providing the necessary data the admin creates the hospital as an entity |
| 3. | Update hospital information | In case of errors in the data of the hospital, the admin can correct the information |
| 4. | Create doctor | By providing the necessary data the admin creates the doctor, and associates him with the specific hospital |
| 5. | Update doctor data | If necessary the admin can change the doctor data, and also map him with another hospital |
| 6. | Delete doctor | If a doctor is no longer available in the system, the admin can delete the doctor |
| 7. | Assign the visit of a doctor to another doctor | In case of removal of a doctor, all the visits of the doctor are appointed to another doctor |
| 8. | Create secretary | By providing the necessary data the admin creates the secretary and map it with a specific hospital |
| 9. | Update secretary data | If necessary the admin can change the secretary data, and also map him with another hospital |
| 10. | Delete a secretary | If a secretary is no longer available in the system, the admin can delete the secretary |
| 11. | Create a pacient | By providing the necessary data the admin creates the patient and map to a specific doctor |
| 12. | Update a pacient | If necessary the admin can update the patient data or map to another hospital |
| 13. | Delete a pacient | If necessary the admin can delete the patient |
| 14. | Update profile | The admin can update his data in the system |

**Secretary**

|  |  |  |
| --- | --- | --- |
| Nr. | User Story Name | Description |
| 1. | Secretary log in | Using a password and an email the secretary access the system |
| 2. | Cancel a visit | In case that a patient call to cancel a visit, the secretary cancel the visit from the system |
| 3. | Update profile | The secretary can update his profile |
| 4. | Create visit | The secretary can create visit for the unregistered patient on timetable |

**Doctor**

|  |  |  |
| --- | --- | --- |
| Nr. | User Story Name | Description |
| 1. | Doctor log in | Using a password and an email the doctor access the system |
| 2. | Check the visits for a specific date | The doctor can see all the visit for a specific date of his selection |
| 3. | Dashboard access | In the dashboard there is a chart containing the number of visits for each day over a specified period |
| 4. | Update profile | The doctor can update his profile |
| 5. | Check the details of the visit | The doctor can check the previous visits details |
| 6. | Create the details of the visit | The doctor fills in the symptomps and recipe for a specific visit, if the visit is not delegated to another doctor |
| 7. | Update the visit details | If necessary the doctor can update the details of a specific visit |
| 8. | Delegate a visit to another doctor | In cases that the visit cannot proced the doctor can delegate the visit to another doctor explaining the cause of this delivery |

**4.2 User Scenarios**

**Patient**

1. Scenario – Patient identification

a.User is requested to enter his medical card ID

b. If the users card ID matches in the database a list of available timetables is showed

c.User is requested to select a date

1. Scenario – Patient is not recognised

a.User is requested to enter his medical card ID

b.If the users card ID does not match in the database user is not recognized

c.A promt will be shown

d.He will be asked to re-enter hic medical card ID(since he has made a typing mistake)

1. Scenario – Patient books a visit

a.Patient is requested to enter an optional e-mail

b.Patient books the visit.

c.Patient receives an e-mail with the details of the visit if the e-mail is provided.

1. Scenario- Patient calncels a visit(e-mail)

a.If the user has provided an e-mail a link is sent to his e-mail.

b.the patient opens the details of the visit.

c.the patient cancels the visit.

1. Scenario- Patient cancels a visit(phone)

a.The patients calls the hospital in wich the appointment was made.

b.The patient verifies the visit.

c.The patient requests to cancel his visit.

**User**

1. Scenario- User log in

a.User is requested an e-mail

b.User is requested a password

c. If his credentials match with those in database , user is successfully logged in

d.Depending on the user category the corresponding page is opened.

1. Scenario- User fails to log in

a.User is requested an e-mail

b.User is requested a password

c. If his credentials do not match with those in database , user is not logged in

d. A prompt will be shown.

e. He will be asked to re-enter his data

1. Scenario- User updates his profile

a.User can change his e-mail

b.User can change his password

c.If the profile is successfully updated the user is notified

d.In case the e-mail already exists the user is notified

**Admin**

9. Scenario – Admin creates the hospital

a.A name is requested

b.An address is requested

c.A level is requested

d.Admin creates the hospital

10.Scenario - Admin updates the hospital details

a.Admin can change the hospital name

b.Admin can change the hospital address

c.Admin can change the hospital level

d.Admin updates the hospital

11. Scenario – Admin creates doctor

a.Doctor is required a name

b.Doctor is required a last name

c.Doctor is required a hospital to be mapped

d.Doctor is required a department

e.Doctor is required a description

f.Doctor is required an e-mail

12. Scenario – Admin updates doctor

a.Admin updates doctors name

b.Admin updates doctors last name

c.Admin changes doctors hospital level

d.Admin changes doctors department

e.Admin changes doctors description

f.Admin requests e-mail change to the doctor.

13. Scenario- Admin deletes doctor

a.Admin selects a doctor

b.If the doctor does not have visits, admin succesfuly deletes the doctor

c.If the doctor has visits the admin is notified

14. Scenario – Admin assigns visits to another doctor

a.Admin selects doctor

b.Admin selects the doctor to assign the visits

c.Admin succesfuly assigns the visits

15. Scenario – Admin creates secretary

a.Secretary is required a name

b.Secretary is required a last name.

c.Secretary is required the hospital to be mapped.

d.Secretary is required a description

e.Secretary is required an e-mail

16. Scenario – Admin updates secretary

a.Admin changes secretary name.

b.Admin changes secretary last name.

c.Admin changes secretary hospital level position

d.Admin request e-mail change

17. Scenario – Admin deletes secretary

a.Admin selects a secretary.

b.Admin succesfuly deletes secretary.

18. Scenario - Admin creates patient

a.Patient is required a name

b.Patient is required a last name

c.Patient is required a medical card ID

d.Patient is required a birthdate

e.Patient is guided in the hospital to be mapped

f.Patient is guided to the doctor to be mapped

19. Scenario - Admin updates patient

a.Admin changes patient name

b.Admin changes patient last name

c.Admin changes patient birthdate

d.Admin changes the doctor to be mapped

e.Admin changes the hospital to be mapped

20. Scenario - Admin deletes patient

a.Admin selects a patient.

b.Admin succesfuly deletes a patient.

**Secretary**

21. Scenario – Secretary cancels a visit

a.Secretary receives a call from a patient

b.Secretary verifies the visit

c.Secretary succesfuly cancels a visit

22. Scenario – Secretary creates a visit

a.Secretary enters a patient ID

b.Secretary selects a date for the visit

c.Secretary selects appointment time on the doctors timetable

d.Secretary succesfuly creates the visit

**Doctor**

23. Scenario – Doctors checks daily visits

a.Doctor selects date

b.Doctor is showed the list of visits for specific date

24. Scenario – Doctor creates patient details

a.Doctor selects a patient

b.Doctor submits symptoms details

c.Doctor submits a receipt details

d.Doctors creates specific details for specific patient

25. Scenario – Doctor updates patient details

a.Doctor selects a patient

b.Doctor updates symptoms details

c.Doctor updates receipt details

d.Doctor succesfuly updates details

26. Scenario – Doctor delegates patient

a.Doctors selects the hospital level to delegate the patient

b.Doctor selects the doctor to be delegated to

c.Doctor fills a form about the reason of the delegation

d.Doctor succesfuly delegates the patient.

**4.3 User cases**

|  |  |
| --- | --- |
| Name | Patient identification |
| Summary | Patient identifies himself and opens appointments calendar |
| Actor | Patient |
| Description | Each patient accesses the system by providing a valid health-card number |
| Precondition | The patient should already have a health card |
| Alternatives | - |
| Post condition | The user can make an appointment now |

|  |  |  |
| --- | --- | --- |
| Name | Patient books a visit | |
| Summary | In case patient wants a visit he chooses a date | |
| Actor | Patient | |
| Description | Patients chooses a date and can enter an optional email for an online appointment details | |
| Precondition | Patient must first identifie and then select a date for the visit | |
| Alternatives | - | |
| Post condition 1 | Visit booked | |
| Post condition 2 | Patient receives an e-mail with the details of the visit if the e-mail is provided | |
| Name  Name | **Patient cancels a Patient cancels a visit(via e-mail)** |  |
| Summary | Patient through email details cancels his appointment | |
| Actor | Patient | |
| Description | Patient clicks on the provided link in the email | |
| Pre condition | If the user has provided an e-mail a link is sent to his email | |
| Alternative | Cancel it through phone | |
| Post condition | Visit canceled | |

|  |  |  |
| --- | --- | --- |
| Name | Patient cancels a visit(via phone) |  |
| Summary | Patient calls to cancel the visit | |
| Actor | Patient-Secretary | |
| Description | Patient calls the secretary to cancel the visit through the phone | |
| Pre condition | A phone number is provided after booking the visit | |
| Alternative | Cancel it through email | |
| Post condition | Visit canceld | |

|  |  |
| --- | --- |
| Name | User log in |
| Summary | User enters the system by providing genuine credentials |
| Actor | User |
| Description | User is requested an e-mail  User is requested an password |
| Pre condition | If his credentials match with those in database , user is succesfully logged in |
| Alternative | - |
| Post condition | Depending on the user category the corresponding page is opened |

|  |  |
| --- | --- |
| Name | User Updates his profile |
| Summary | User makes changes to his profile details |
| Actor | User |
| Description | User can change his e-mail  User can change his password |
| Pre condition | User should log in |
| Alternative | Admin updates his profile |
| Post condition | If the profile is successfully updated the user is notified.  In case the e-mail already exist the user is notified |

|  |  |
| --- | --- |
| Name | Admin creates the hospital |
| Summary | Admin requestes some details in order to create a hospital as an entity |
| Actor | Admin |
| Description | A name is requested.  An address is requested.  A level is requested |
| Pre condition | The User must have admin credencials |
| Alternative | We continuo with the avaible hospitals |
| Post condition | Now we can enter hospital details |

|  |  |
| --- | --- |
| Name | Hospital Update |
| Summary |  |
| Actor | Admin |
| Description | Admin can change the hospital name.  Admin can change the hospital address.  Admin can change the hospital level. |
| Pre condition | Admin must first create the hospital |
| Alternative | - |
| Post condition | Now hospital system is avaible |

|  |  |
| --- | --- |
| Name | Admin creates doctor |
| Summary | Admin fulfills the neccesary details to create a doctor for the hospital |
| Actor | Admin |
| Description | Doctor is required a name.  Doctor is required a last name.  Doctor is required a hospital to be mapped.  Doctor is required a departament.  Doctor is required a description  Doctor is required an e-mail |
| Precondition | Hospital needs a Doctor so it can work |
| Alternative | Should go to a hospital where system works |
| Post condition | Doctor is ready to be updated |

|  |  |
| --- | --- |
| Name | Admin updates Doctor |
| Summary | The Admin manages Doctors data specification |
| Actor | Admin |
| Description | Admin updates doctors name  Admin updates doctors last name  Admin changes doctors hospital level  Admin changes doctors departmen  Admin changes doctors description  Admin requestes e-mail change to doctor |
| Pre condition | Specific datas must be changed and updated so everythings is okay |
| Alternative | All the datas must be filled so the system can work |
| Post condition | Hospital System now works and can start the visits |

|  |  |
| --- | --- |
| Name | Admin deletes doctor |
| Summary | Some doctor is ready to retire or has changed hospital or simply has quited his job to go to asylum in germany so we must delete its user |
| Actor | Admin |
| Description | Admin selects a doctor |
| Pre condition | If the doctor does not have visits, admin succesfully deletes the doctor |
| Alternative | If the doctor has visits the admin is notified |

|  |  |
| --- | --- |
| Name | Admin assigns visits to another doctor |
| Summary | In case vists transfer is required, the admin will handly the visit transferring from a doctor to another |
| Actor | The Admin |
| Description | All the visits are transferred if the college if he has empty slots on his timetable otherwise admin is notified |
| Precondition | The doctor must have visits and he is inable to do the visits in order to make the transfer |
| Alternatives | The transfer can be made only once |
| Post condition | The visits are transferred succesfuly |

|  |  |
| --- | --- |
| Name | Admin creates secretary |
| Summary | In case new secretary needs to be added in the software, the admin will create the profile. |
| Actor | The Admin |
| Description | Admin is responsible for creating the secretary profile |
| Precondition | A new secretary needs a profile in order to have access on the system |
| Alternatives | If the e-mail the secretary is providing already corresponds in the database the admin will be notified |
| Post condition | The secretary can access the system |

|  |  |
| --- | --- |
| Name | Admin updates secretary |
| Summary | In case the details of the secretary needs to be updated, the admin will update it. |
| Actor | The Admin |
| Description | Admin is responsible for updating the secretary profile |
| Precondition | A secretary needs a profile in order to have access on the info |
| Alternatives | If the e-mail the secretary is providing already corresponds in the database the admin will be notified |
| Post condition | The secretary can access the system using the new info provided |

|  |  |
| --- | --- |
| Name | Admin deletes secretary |
| Summary | Whenever a secretary should be removed from the system, the admin is in charge for this. |
| Actor | The Admin |
| Description | Admin can delete a user profile in case it is no longer needed. |
| Precondition | The institution decides that a secretary profile is no longer needed. |
| Alternatives | In case the secretary is using the system while the admin is deleting, it cannot be deleted. |
| Post condition | The user is removed from the database. |

|  |  |
| --- | --- |
| Name | Admin creates patient |
| Summary | In case new patient need to be added in the software, the admin will create them profiles.. |
| Actor | The Admin |
| Description | Administrator is responsible for creating new MMS user profiles. |
| Precondition | The patient need specific info in order to access the hospital |
| Alternatives | - |
| Post condition | The user can access the system. |

|  |  |
| --- | --- |
| Name | Admin updates patient |
| Summary | Whenever a patient profile should be modified, the admin is in charge for this. |
| Actor | The Admin |
| Description | Administrator is responsible for editing patient data, in case that profile needs an update. |
| Precondition | - |
| Alternatives | The updated profile cannot be saved if there are empty fields. |
| Post condition | The user profile is updated. |

|  |  |
| --- | --- |
| Name | Admin deletes patient |
| Summary | Whenever a patient profile should be removed from the system, the admin is in charge for this. |
| Actor | The Admin |
| Description | Administrator can delete a patient profile in case it is no longer needed |
| Precondition | The hospital/doctor decides that a user profile is no longer needed. |
| Alternatives | In case the user is using the system while the admin is deleting, it cannot be deleted. |
| Post condition | The user is removed from the database |

|  |  |
| --- | --- |
| Name | Secretary cancels a visit |
| Summary | Whenever a patient makes a call to cancer a visit, the secretary is responsible for canlceling their appointmnt |
| Actor | Secretary |
| Description | Secretary can cancel a visit if a request is made and the info is verified |
| Precondition | The patient must make a call and confirm his information |
| Alternatives | If the person does not confirm his information he should not be removed from the timetable |
| Post condition | The secretary successfully cancels a visit |

|  |  |
| --- | --- |
| Name | Secretary creates a visit |
| Summary | Whenever the patient goes directly to the secretary to make an appointment, in case there is empty place in the timetable, the secretary can create an appointment |
| Actor | Secretary |
| Description | Secretary can create a visit if the right documentation is provided in the reception |
| Precondition | The patient must have the correct documentation with him. |
| Alternatives | If the person does not confirm his information he should not be added on the timetable |
| Post condition | The secretary successfully creates a visit |

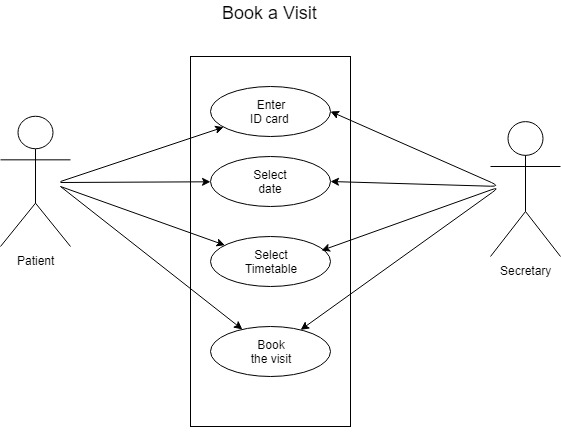
|  |  |
| --- | --- |
| Name | Doctors checks daily visits |
| Summary | The doctor has the possibility to check his timetable and the previous timeables he has had using the corresponding date |
| Actor | Doctor |
| Description | Doctor can check his timetable |
| Precondition | - |
| Alternatives | - |
| Post condition | The doctor succcesfuly checks his timetable |

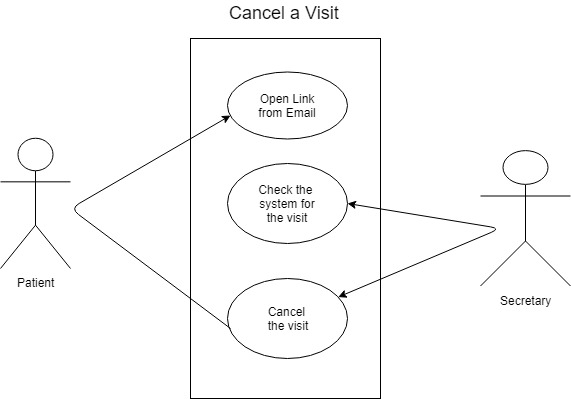
|  |  |
| --- | --- |
| Name | Doctor creates patient details |
| Summary | The doctor has the possibility to select a patient and to submit in a specific form his symptoms and the recipt he has given to him. |
| Actor | Doctor |
| Description | Doctor gives the form of symptoms and the recipt of specific patient |
| Precondition | There must be an active patient |
| Alternatives | - |
| Post condition | The doctor creates patient details |

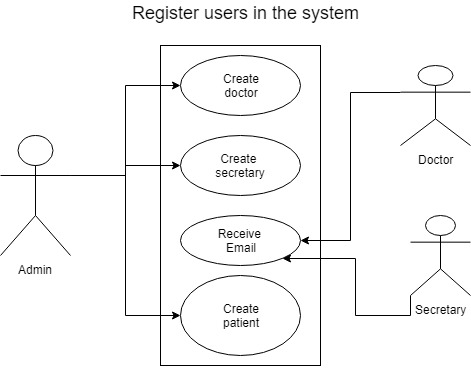
|  |  |
| --- | --- |
| Name | Doctor updates patient details |
| Summary | The doctor has the possibility to select a patient and to update his details about the symptoms and the recipt |
| Actor | Doctor |
| Description | Doctor updates the information he has given is cases of mistakes or symptoms change on a second visit |
| Precondition | There must be already data about symptoms and recipt in order to be updated |
| Alternatives | - |
| Post condition | The doctor succcesfuly updates patient details |

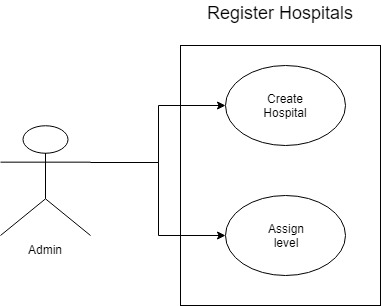
|  |  |
| --- | --- |
| Name | Doctor delegates patient |
| Summary | The doctor has the possibility to delegate a patient to another college if the symphtom is out of his knowledge. |
| Actor | Doctor |
| Description | Doctor fills a form explaining the reason of delegatin the patient. |
| Precondition | There must be the an active patient |
| Alternatives | - |
| Post condition | The doctor succcesfuly delegates the patient |

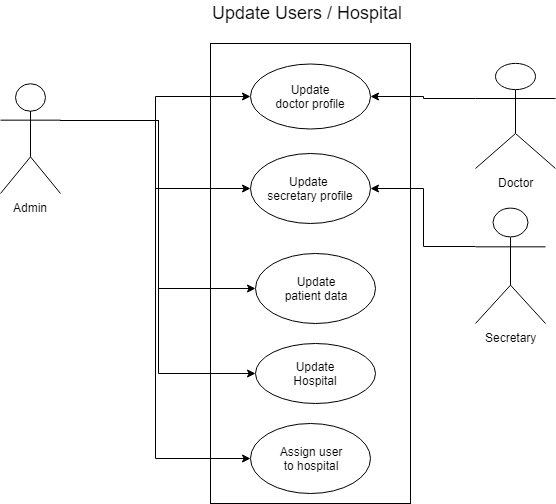
**4.4 User Case Diagram**

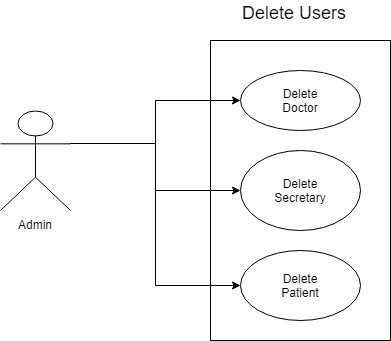
****

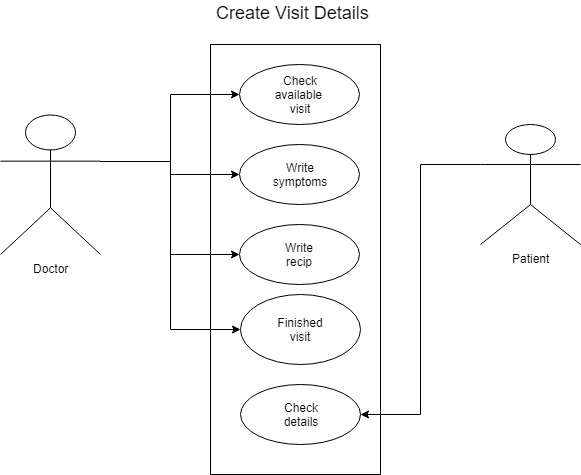
****

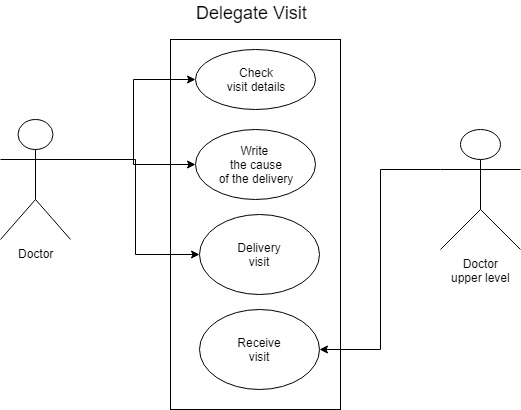
****

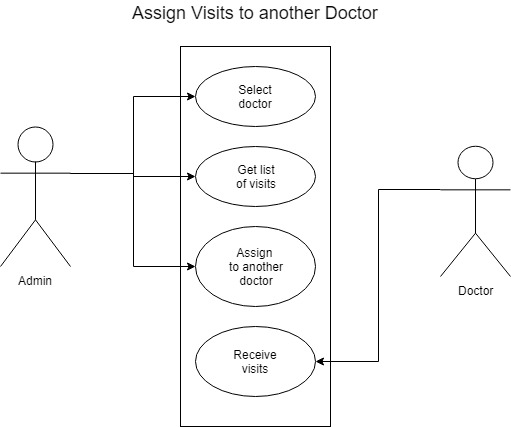
****

****

****

****

****

****

4.5 Activity Diagram

1.png

2.png

3.png

4.png

5.png

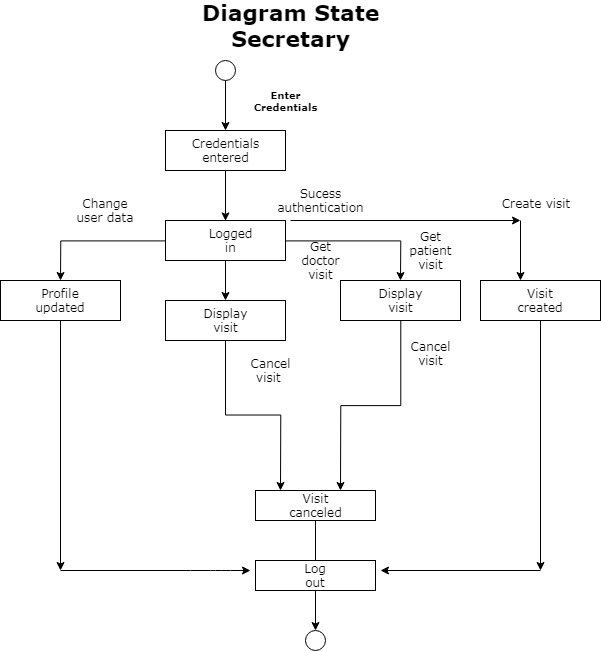
6.png

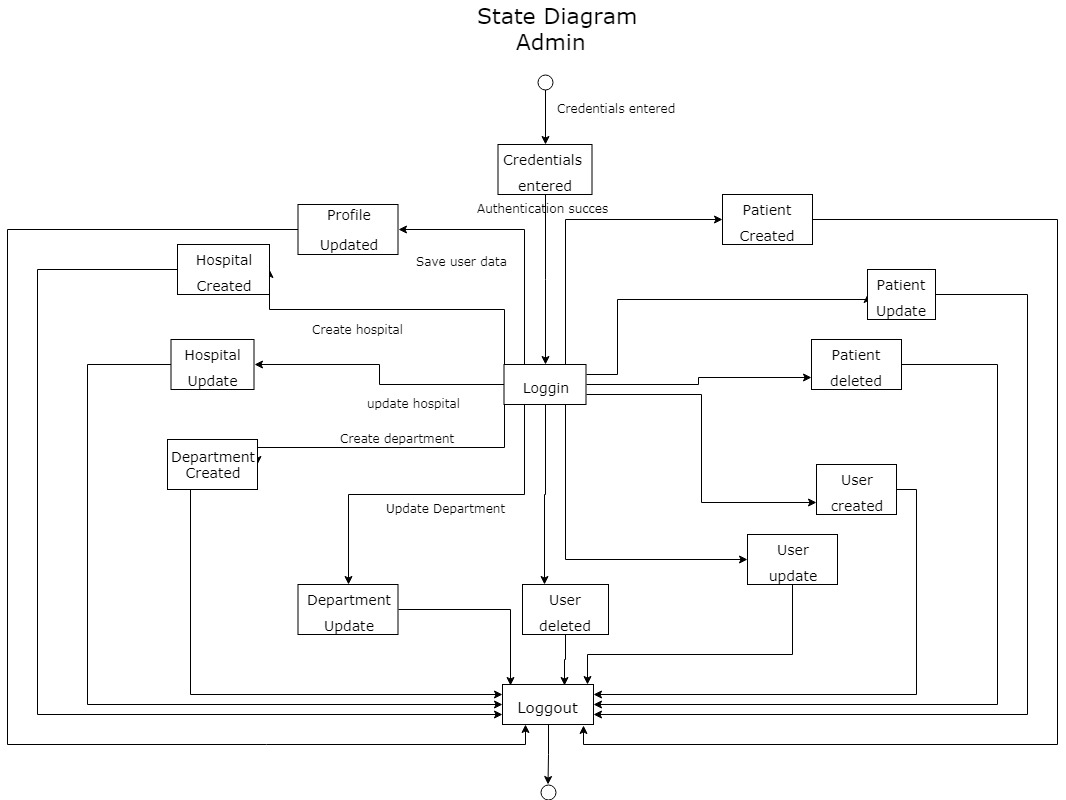
7.png

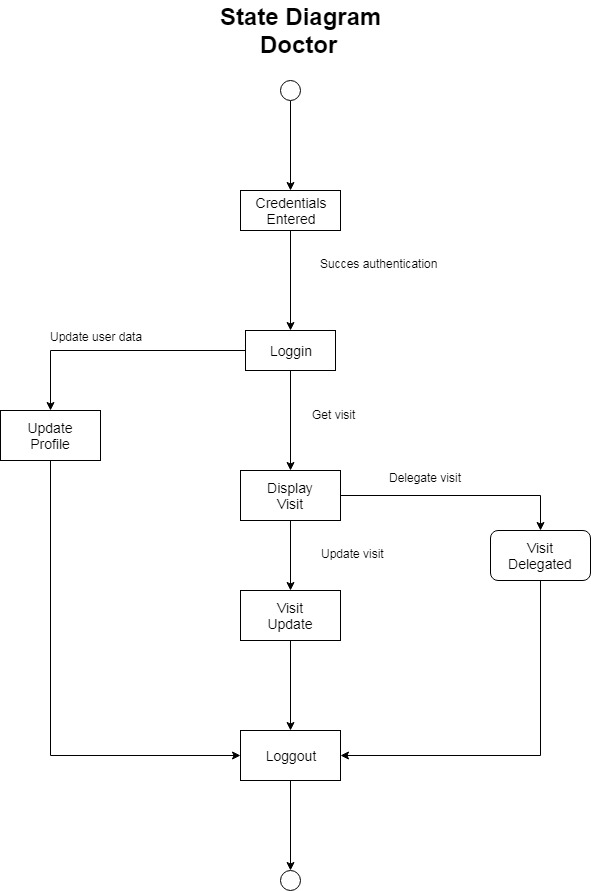
8.png

9.png

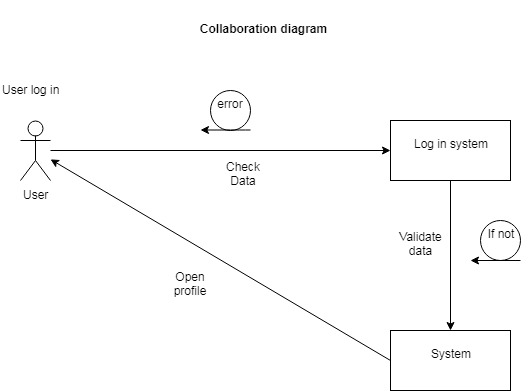
**4.6 State Diagrams**

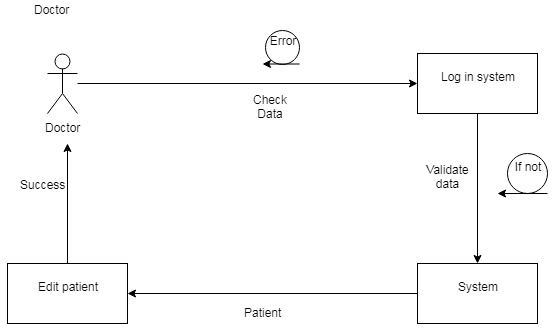


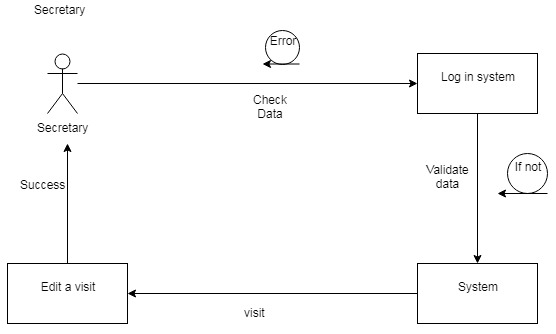


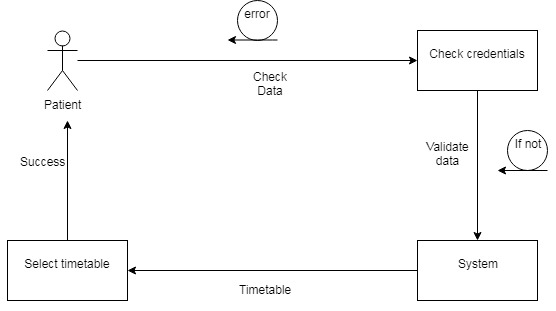


**4.7 Collaboration Diagrams**

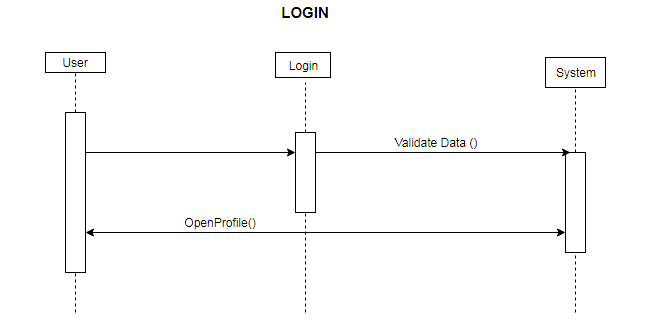


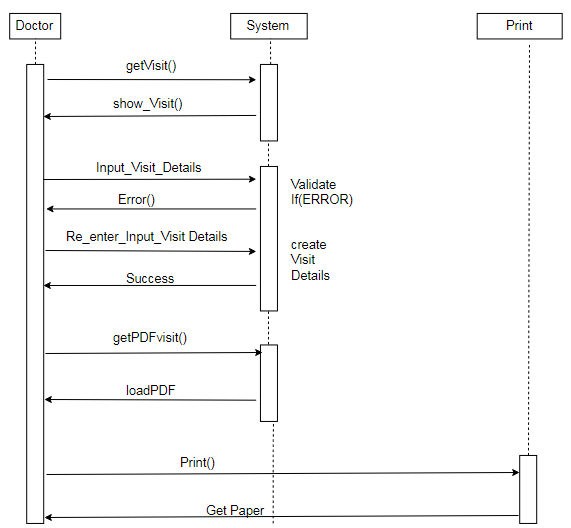


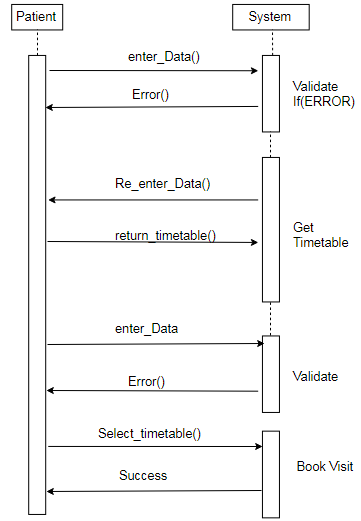


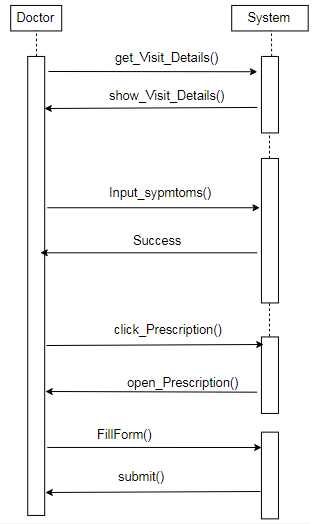


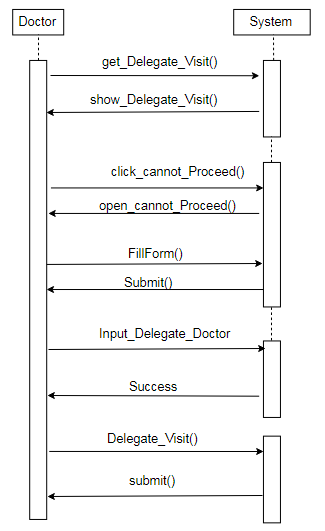
4.8 Sequence Diagrams

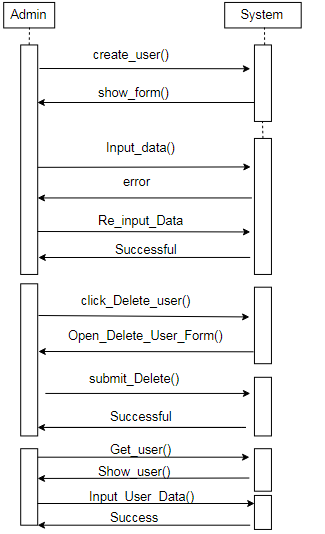


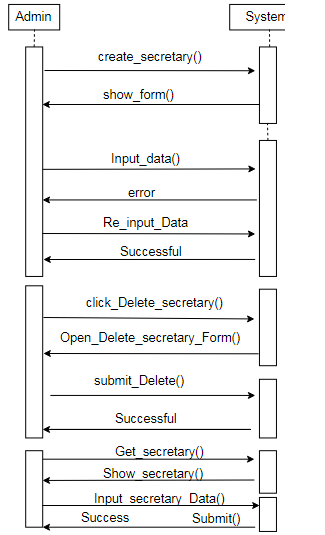


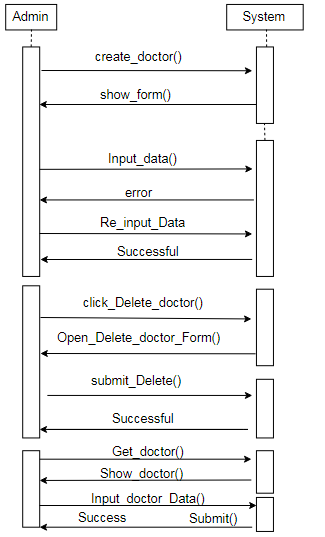


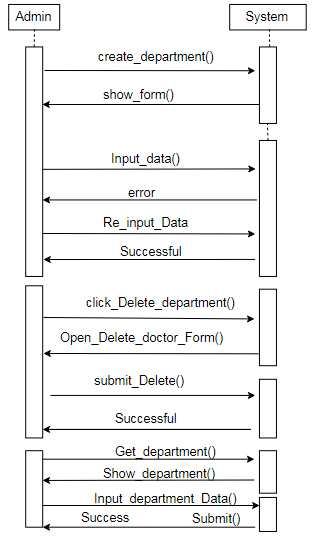


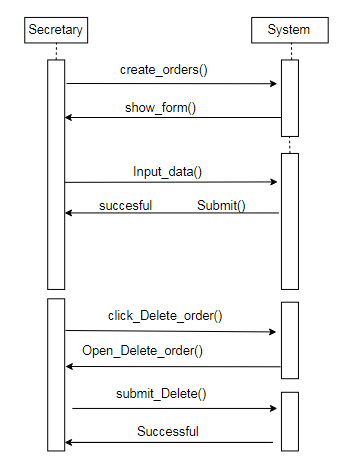




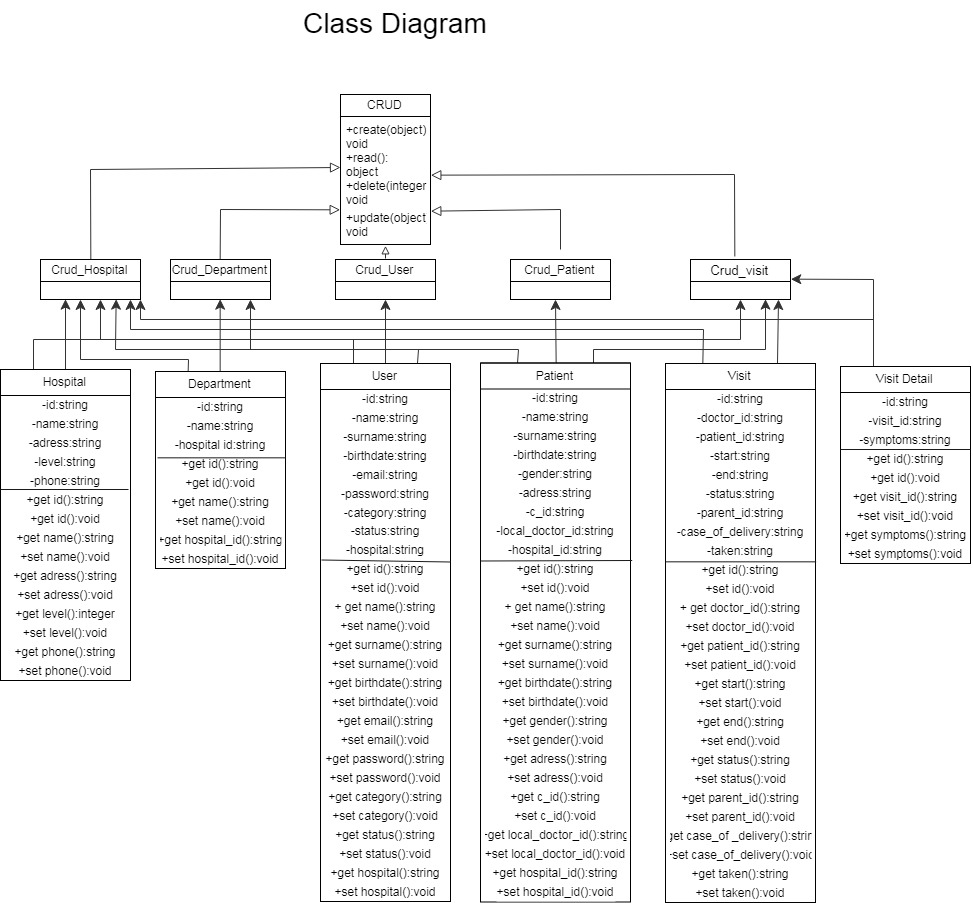


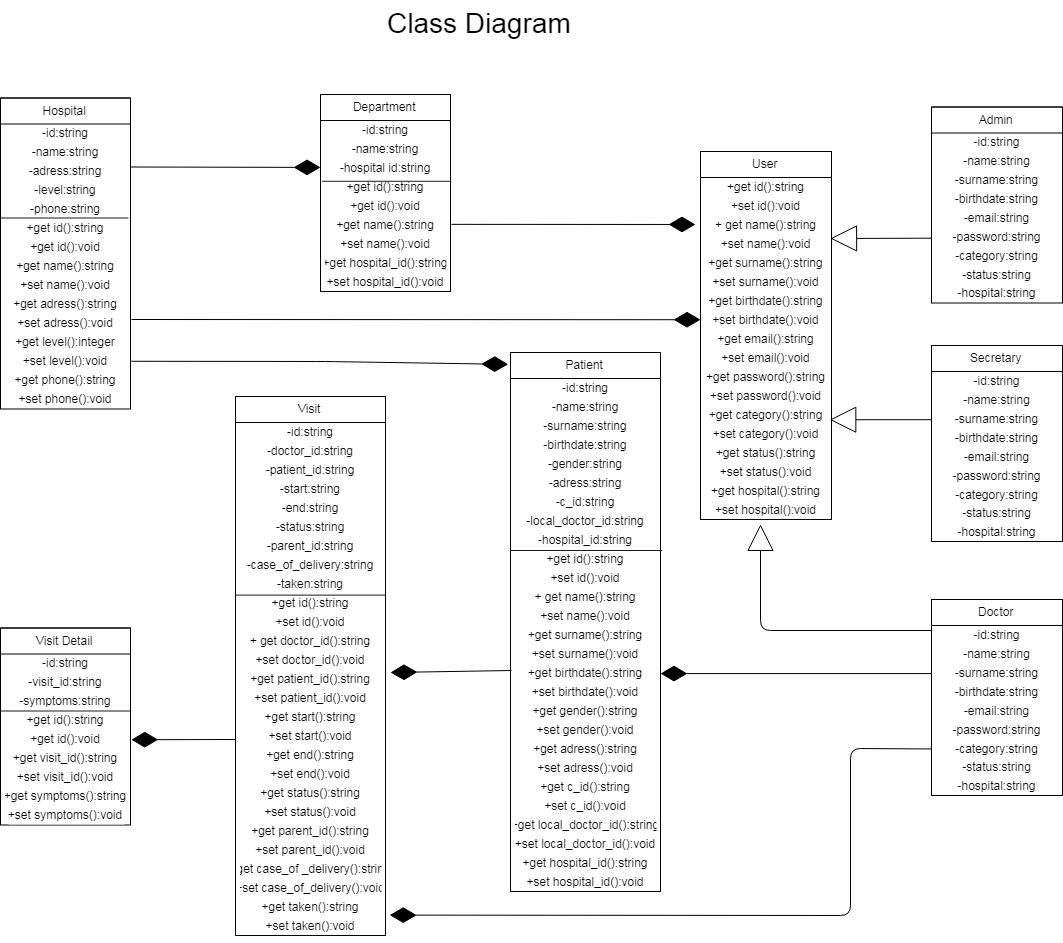




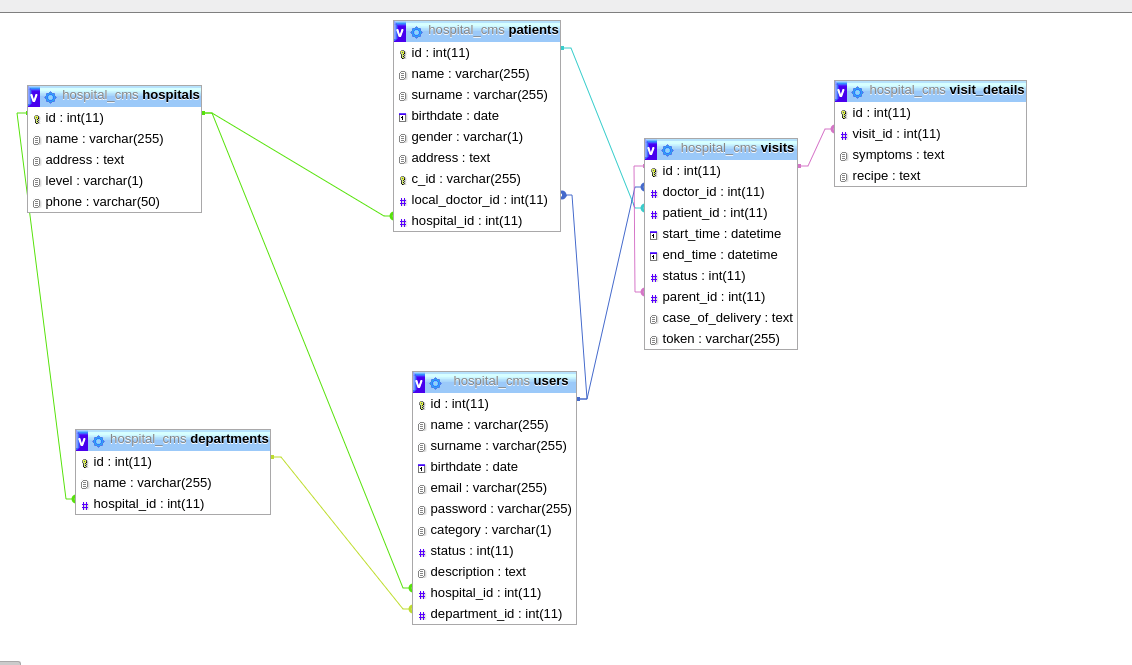


4.9 Class Diagram

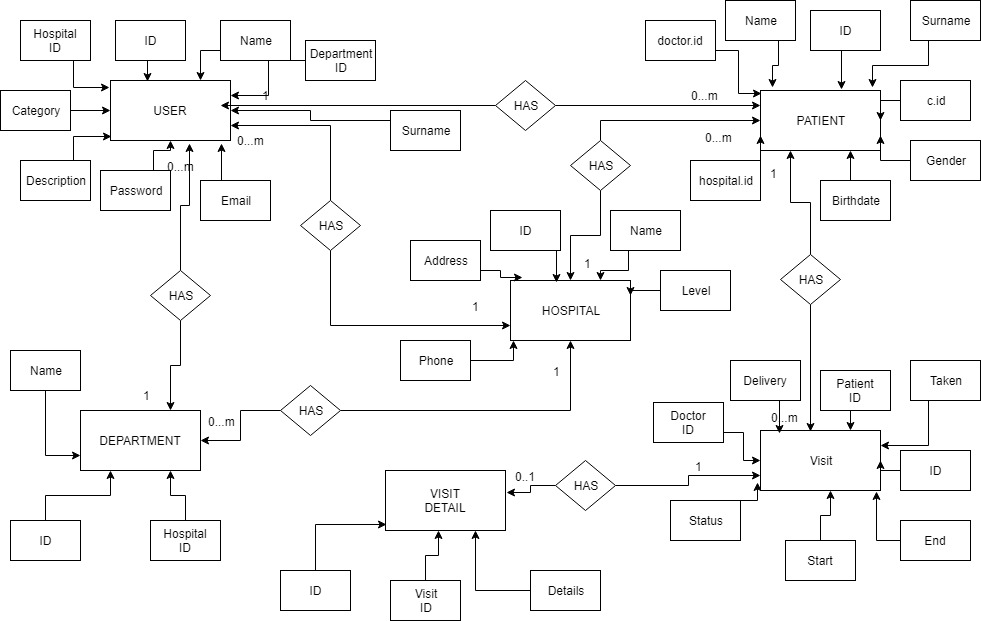




4.10 DB Schema



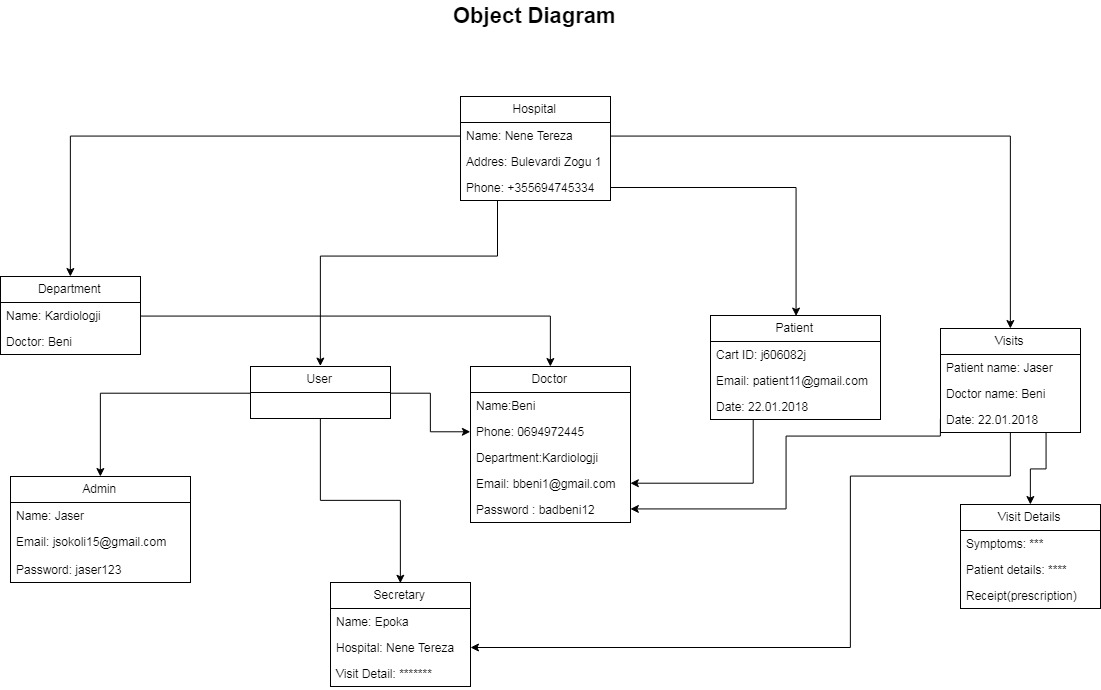
4.11 ER Diagram



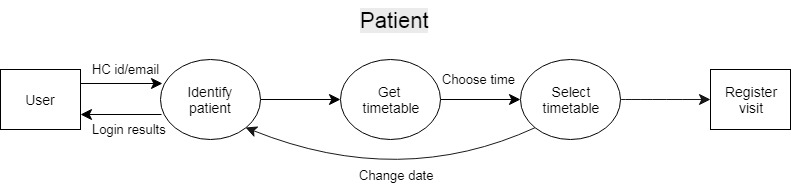
4.12 Component Diagram

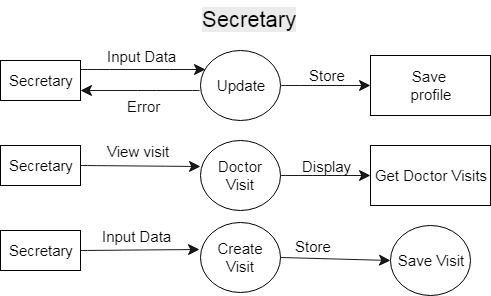
ComponetDiagram.png

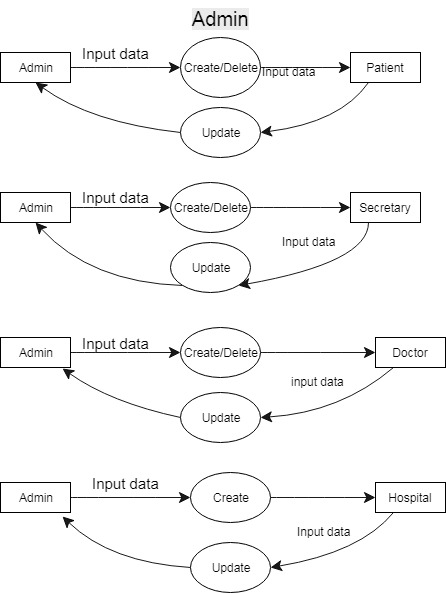
4.13 Object Diagram

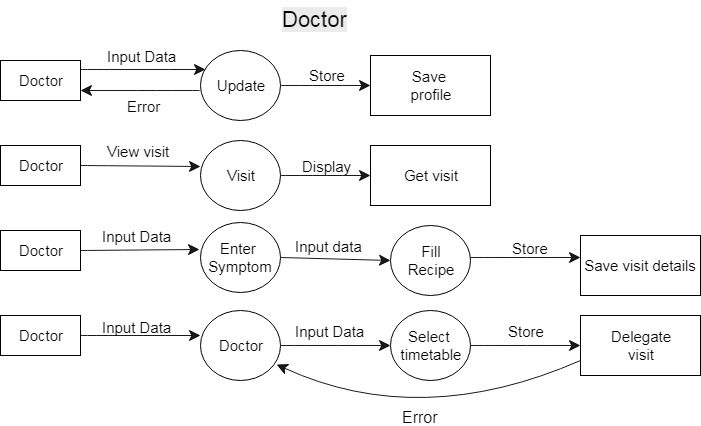


4.14 Data Flow Diagram

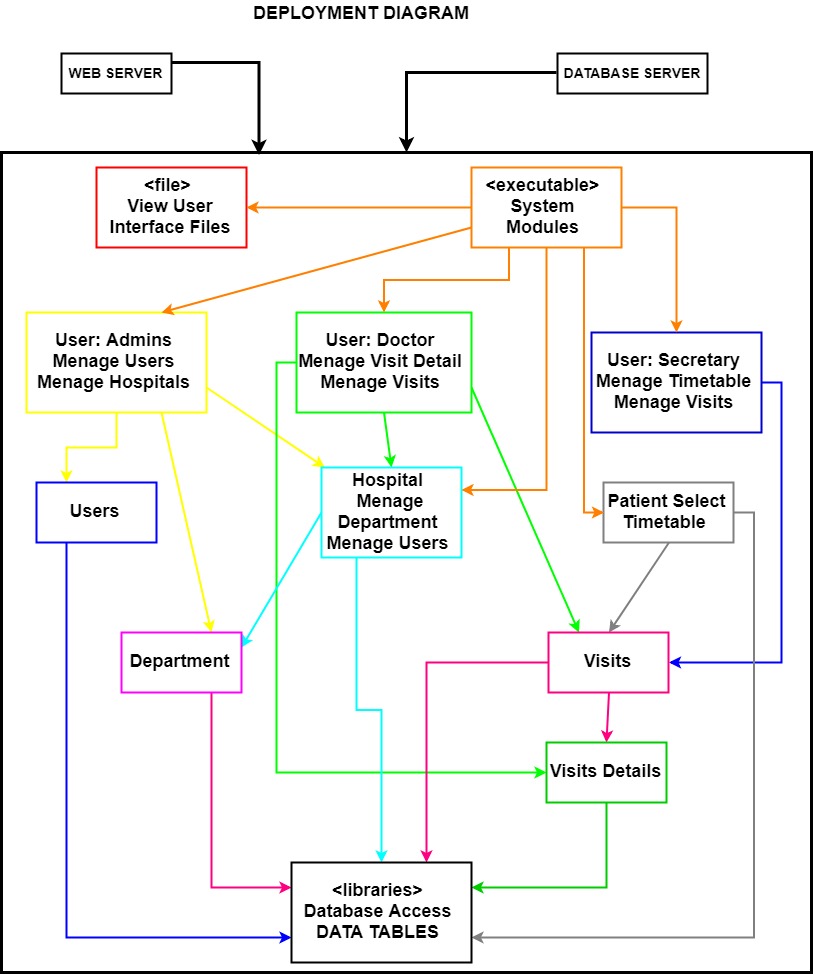








4.15 Deployment Diagram



5. Implementation Technology

SRC

DepartmentQueries

<?php

require\_once dirname(dirname(\_\_FILE\_\_))."/database/Database.php";

require\_once dirname(dirname(\_\_FILE\_\_))."/database/Department.php";

class DepartmentQueries

{

private $connection;

public function \_\_construct()

{

$database = new Database();

$this->connection = $database->getConnection();

}

public function getDepartmentById($id, $hospital\_id = null) {

$hospital\_id\_query = "";

if(!empty($hospital\_id)){

$hospital\_id\_query = " and hospital\_id = '$hospital\_id'";

}

$query = "Select \* from departments where id = '$id'" . $hospital\_id\_query;

$perform = mysqli\_query($this->connection, $query);

if ($perform) {

if(mysqli\_num\_rows($perform) == 1) {

$result = mysqli\_fetch\_assoc($perform);

$id = $result['id'];

$name = $result['name'];

$hospital\_id = $result['hospital\_id'];

return new Department($id, $name, $hospital\_id);

}

return null;

}

return false;

}

function createDepartment(Department $department){

$name = $department->getName();

$hospital\_id = $department->getHospitalId();

$query = "Insert into departments(name, hospital\_id) VALUES ('$name', '$hospital\_id')";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

return true;

}

return false;

}

function updateDepartment(Department $department) {

$id = $department->getId();

$name = $department->getName();

$query = "Update departments set name = '$name' where id = '$id'";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

return true;

}

return false;

}

public function getConnection() {

return $this->connection;

}

public function getDepartmentsWithDoctors($hospital\_id){

$query = "Select departments.\*,

CONCAT('[', IFNULL(

(

SELECT GROUP\_CONCAT(

CONCAT(

'{',

'\"id\":', users.id, ',',

'\"name\":', CONCAT('\"',users.name, '\"'), ',',

'\"surname\":', CONCAT('\"',users.surname,'\"'),

'}'

)

)

from users

where users.hospital\_id = '$hospital\_id'

and users.department\_id = departments.id

and users.category = 'd'

)

, '') , ']') as doctors

from departments where hospital\_id = '$hospital\_id'";

$perform = mysqli\_query($this->connection, $query);

if($perform){

$departments = [];

while ($result = mysqli\_fetch\_assoc($perform)){

$id = $result['id'];

$name = $result['name'];

$doctors = $result['doctors'];

$departments[] = [

'department' => new Department($id, $name, $hospital\_id),

'doctors' => json\_decode($doctors, true)

];

}

return $departments;

}

return false;

}

}

HospitalQueries

<?php

require\_once dirname(dirname(\_\_FILE\_\_))."/database/Database.php";

require\_once dirname(dirname(\_\_FILE\_\_))."/database/Hospital.php";

require\_once dirname(dirname(\_\_FILE\_\_))."/database/Department.php";

class HospitalQueries

{

private $connection;

public function \_\_construct()

{

$database = new Database();

$this->connection = $database->getConnection();

}

public function getHospitalById($id) {

$query = "Select \* from hospitals where id ='$id'";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

if(mysqli\_num\_rows($perform) == 1) {

$result = mysqli\_fetch\_assoc($perform);

$id = $result['id'];

$name = $result['name'];

$address = $result['address'];

$level = $result['level'];

$phone = $result['phone'];

return new Hospital($id, $name, $address, $level, $phone);

} else {

return false;

}

}

return null;

}

public function getHospitalWithUsers(){

$query = "select hospitals.\*, concat('[', ifnull((

select group\_concat(

CONCAT(

'{',

'\"id\":', users.id, ',',

'\"name\":', CONCAT('\"',users.name,'\"'), ',',

'\"surname\":', CONCAT('\"', users.surname, '\"'),

'}'

)

)

from users where users.hospital\_id = hospitals.id and users.category = 'd'

),'') , ']') as doctors

from hospitals";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

$hospitals = [];

while ($result = mysqli\_fetch\_assoc($perform)) {

$id = $result['id'];

$name = $result['name'];

$address = $result['address'];

$level = $result['level'];

$phone = $result['phone'];

$doctors = json\_decode($result['doctors'], true);

$doctors\_data = [];

foreach ($doctors as $doctor) {

$doctors\_data[] = [

'id' => $doctor['id'],

'name' => $doctor['name'],

'surname' => $doctor['surname']

];

}

$hospitals[] = [

'hospital' => new Hospital($id, $name, $address, $level, $phone),

'doctors' => $doctors\_data

];

}

return $hospitals;

}

return false;

}

public function getHospitalData($id = null, $with\_data = false) {

$id\_query = "";

if(!empty($id)) {

$id\_query = "where id = '$id'";

}

$with\_data\_query = "";

if($with\_data){

$with\_data\_query = ",

concat('[', ifnull(

(SELECT group\_concat(

CONCAT(

'{',

'\"id\":', departments.id, ',',

'\"name\":', CONCAT('\"', departments.name, '\"'),

'}'

)

)

from departments where departments.hospital\_id = hospitals.id)

, ''), ']') as departments";

}

$query = "Select hospitals.\*

{$with\_data\_query}

from hospitals

". $id\_query;

$perform = mysqli\_query($this->connection, $query);

if($perform) {

$hospitals = [];

while ($result = mysqli\_fetch\_assoc($perform)){

$id = $result['id'];

$name = $result['name'];

$address = $result['address'];

$level = $result['level'];

$phone = $result['phone'];

$department\_data = null;

if($with\_data){

$departments = json\_decode($result['departments'], true);

$department\_data = [];

foreach ($departments as $department) {

$dep\_id = $department['id'];

$dep\_name = $department['name'];

$department\_data[] = new Department($dep\_id, $dep\_name, $id);

}

}

$hospitals[] = [

'hospital' => new Hospital($id, $name, $address, $level, $phone),

'departments' => $department\_data

];

}

return $hospitals;

}

return false;

}

public function updateHospital(Hospital $hospital) {

$id = $hospital->getId();

$name = $hospital->getName();

$address = $hospital->getAddress();

$level = $hospital->getLevel();

$phone = $hospital->getPhone();

$query = "Update hospitals set name = '$name', address = '$address', level = '$level', phone = '$phone'

where id = '$id'";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

return true;

}

return false;

}

public function createHospital(Hospital $hospital) {

$name = $hospital->getName();

$address = $hospital->getAddress();

$level = $hospital->getLevel();

$phone = $hospital->getPhone();

$query = "Insert into hospitals(name, address, level, phone)

values('$name', '$address', '$level', '$phone')";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

return true;

}

return false;

}

public function getConnection() {

return $this->connection;

}

}

.

PatientQueries

<?php

require\_once dirname(dirname(\_\_FILE\_\_))."/database/Database.php";

require\_once dirname(dirname(\_\_FILE\_\_))."/database/Patient.php";

class PatientQueries

{

private $connection;

public function \_\_construct()

{

$database = new Database();

$this->connection = $database->getConnection();

}

public function getConnection() {

return $this->connection;

}

public function getPatientById($id, $hospital\_id = null, $doctor\_id = null) {

$hospital\_query = "";

if(!empty($hospital\_id)){

$hospital\_query = " and hospital\_id = '$hospital\_id'";

}

$doctor\_query = "";

if(!empty($doctor\_id)){

$doctor\_query = " and local\_doctor\_id = '$doctor\_id'";

}

$query = "Select \* from patients where id = '$id'" . $hospital\_query . $doctor\_query;

$perform = mysqli\_query($this->connection, $query);

if($perform) {

if(mysqli\_num\_rows($perform) == 1) {

$result = mysqli\_fetch\_assoc($perform);

$id = $result['id'];

$name = $result['name'];

$surname = $result['surname'];

$birthdate = $result['birthdate'];

$gender = $result['gender'];

$address = $result['address'];

$c\_id = $result['c\_id'];

$local\_doctor\_id = $result['local\_doctor\_id'];

$hospital\_id = $result['hospital\_id'];

return new Patient($id, $name, $surname, $birthdate, $gender, $address, $c\_id, $local\_doctor\_id, $hospital\_id);

}

return null;

}

return false;

}

public function getPatientByCardId($card\_id) {

$query = "Select \* from patients where c\_id = '$card\_id'";

$perform = mysqli\_query($this->connection, $query);

if($perform){

if(mysqli\_num\_rows($perform) == 1) {

$result = mysqli\_fetch\_assoc($perform);

$id = $result['id'];

$name = $result['name'];

$surname = $result['surname'];

$birthdate = $result['birthdate'];

$gender = $result['gender'];

$address = $result['address'];

$c\_id = $result['c\_id'];

$local\_doctor\_id = $result['local\_doctor\_id'];

$hospital\_id = $result['hospital\_id'];

return new Patient($id, $name, $surname, $birthdate, $gender, $address, $c\_id, $local\_doctor\_id, $hospital\_id);

}

return null;

}

return false;

}

public function getPatientByHospitalId($hospital\_id) {

$query = "Select \* from patients where hospital\_id = '$hospital\_id'";

$perform = mysqli\_query($this->connection, $query);

if($perform){

$patients = [];

while ($result = mysqli\_fetch\_assoc($perform)) {

$id = $result['id'];

$name = $result['name'];

$surname = $result['surname'];

$birthdate = $result['birthdate'];

$gender = $result['gender'];

$address = $result['address'];

$c\_id = $result['c\_id'];

$local\_doctor\_id = $result['local\_doctor\_id'];

$hospital\_id = $result['hospital\_id'];

$patients[] = new Patient($id, $name, $surname, $birthdate, $gender, $address, $c\_id, $local\_doctor\_id, $hospital\_id);

}

return $patients;

}

return false;

}

public function createPatient(Patient $patient) {

$name = $patient->getName();

$surname = $patient->getSurname();

$birthdate = $patient->getBirthdate();

$gender = $patient->getGender();

$address = $patient->getAddress();

$c\_id = $patient->getCId();

$local\_doctor\_id = $patient->getLocalDoctorId();

$hospital\_id = $patient->getHospitalId();

$query = "Insert into patients (name, surname, birthdate, gender, address, c\_id, local\_doctor\_id, hospital\_id)

VALUES ('$name', '$surname', '$birthdate', '$gender', '$address', '$c\_id', '$local\_doctor\_id', '$hospital\_id')";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

return true;

}

return false;

}

public function updatePatient(Patient $patient) {

$id = $patient->getId();

$name = $patient->getName();

$surname = $patient->getSurname();

$birthdate = $patient->getBirthdate();

$gender = $patient->getGender();

$address = $patient->getAddress();

$c\_id = $patient->getCId();

$local\_doctor\_id = $patient->getLocalDoctorId();

$hospital\_id = $patient->getHospitalId();

$query = "Update patients set name = '$name', surname = '$surname', birthdate = '$birthdate',

gender = '$gender', address = '$address', c\_id = '$c\_id', local\_doctor\_id = '$local\_doctor\_id',

hospital\_id = '$hospital\_id' where id = '$id'";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

return true;

}

return false;

}

public function deletePatient($patient\_id) {

$query = "Delete from patients where id = '$patient\_id'";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

return true;

}

return false;

}

public function changePatientsDoctor($previousDoctorId, $doctorId) {

$query = "Update patients set local\_doctor\_id = '$doctorId' where local\_doctor\_id = '$previousDoctorId'";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

return true;

}

return false;

}

}

.

SendMail

<?php

require\_once dirname(\_\_FILE\_\_)."/PHPMailer-master/PHPMailerAutoload.php";

require\_once dirname(\_\_FILE\_\_)."/PHPMailer-master/class.phpmailer.php";

class SendMail

{

public static function mail($email, $password, $name, $action){

if($action == 'create') {

$body = "Welcome to Hospital CMS. Please use this email account and the password : '$password' to login";

} else {

$body = "Your account credentials were changed. Please use this email account and the password : '$password' to login";

}

$mail = new PHPMailer;

$mail->isSMTP();

$mail->Debugoutput = 'html';

$mail->Host = 'smtp.gmail.com';

$mail->Port = 587;

$mail->SMTPSecure = 'tls';

$mail->SMTPAuth = true;

$mail->Username = "hospitalcms.app@gmail.com";

$mail->Password = "hospitalcms";

$mail->setFrom('hospitalcms.app@gmail.com', 'Hospital CMS');

$mail->addReplyTo('hospitalcms.app@gmail.com', 'Hospital CMS');

$mail->addAddress($email, $name);

$mail->Subject = 'New Password for your Hospital CMS Account';

$mail->Body = $body.PHP\_EOL." Please change the password after you login";

$mail->AltBody = "password for your account : $password";

$mail->send();

return $mail->ErrorInfo;

}

public static function visitMail($email, $name, $body) {

$mail = new PHPMailer;

$mail->isSMTP();

$mail->Debugoutput = 'html';

$mail->Host = 'smtp.gmail.com';

$mail->Port = 587;

$mail->SMTPSecure = 'tls';

$mail->SMTPAuth = true;

$mail->Username = "hospitalcms.app@gmail.com";

$mail->Password = "hospitalcms";

$mail->setFrom('hospitalcms.app@gmail.com', 'Hospital CMS');

$mail->addReplyTo('hospitalcms.app@gmail.com', 'Hospital CMS');

$mail->addAddress($email, $name);

$mail->Subject = 'Visit details for your Hospital CMS visit';

$mail->Body = nl2br($body);

$mail->AltBody = "visit details for your visit: " . nl2br($body);

$mail->send();

return $mail->ErrorInfo;

}

}

UserQueries

<?php

require\_once dirname(dirname(\_\_FILE\_\_))."/database/Database.php";

require\_once dirname(dirname(\_\_FILE\_\_))."/database/User.php";

class UserQueries

{

private $connection;

public function \_\_construct()

{

$database = new Database();

$this->connection = $database->getConnection();

}

public function getConnection() {

return $this->connection;

}

public function getUserById($id) {

$query = "Select \* from users where id = '$id'";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

if(mysqli\_num\_rows($perform) == 1) {

$result = mysqli\_fetch\_assoc($perform);

$id = $result['id'];

$name = $result['name'];

$surname = $result['surname'];

$birthdate = $result['birthdate'];

$email = $result['email'];

$password = $result['password'];

$category = $result['category'];

$status = $result['status'];

$description = $result['description'];

$hospital\_id = $result['hospital\_id'];

$department\_id = $result['department\_id'];

return new User($id, $name, $surname, $birthdate, $email, $password, $category, $status, $description,$hospital\_id, $department\_id);

}

} else {

return false;

}

return null;

}

public function getUserByEmail($email) {

$query = "Select \* from users where email = '$email'";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

if(mysqli\_num\_rows($perform) == 1) {

$result = mysqli\_fetch\_assoc($perform);

$id = $result['id'];

$name = $result['name'];

$surname = $result['surname'];

$birthdate = $result['birthdate'];

$email = $result['email'];

$password = $result['password'];

$category = $result['category'];

$status = $result['status'];

$description = $result['description'];

$hospital\_id = $result['hospital\_id'];

$department\_id = $result['department\_id'];

return new User($id, $name, $surname, $birthdate, $email, $password, $category, $status, $description,$hospital\_id, $department\_id);

}

} else {

return false;

}

return null;

}

public function getUsersByHospital($hospital\_id, $user\_id = null) {

$user\_query = "";

if(!empty($user\_id)){

$user\_query = " and id = '$user\_id'";

}

$query = "Select \* from users where hospital\_id = '$hospital\_id'" . $user\_query;

$perform = mysqli\_query($this->connection, $query);

if($perform) {

$users = [];

while ($result = mysqli\_fetch\_assoc($perform)){

$id = $result['id'];

$name = $result['name'];

$surname = $result['surname'];

$birthdate = $result['birthdate'];

$email = $result['email'];

$password = $result['password'];

$category = $result['category'];

$status = $result['status'];

$description = $result['description'];

$hospital\_id = $result['hospital\_id'];

$department\_id = $result['department\_id'];

$users[] = new User($id, $name, $surname, $birthdate, $email, $password, $category, $status, $description, $hospital\_id, $department\_id);

}

return $users;

}

return false;

}

public function createUser(User $user, $hospital\_id, $department\_id) {

$name = $user->getName();

$surname = $user->getSurname();

$birthdate = $user->getBirthdate();

$email = $user->getEmail();

$password = $user->getPassword();

$category = $user->getCategory();

$description = $user->getDescription();

if(!empty($description)){

$description = "'$description'";

} else {

$description = "null";

}

if(!empty($department\_id)) {

$department\_id = "'$department\_id'";

} else {

$department\_id = "null";

}

$query = "Insert into users (name, surname, birthdate, email, password, category, status, description, hospital\_id, department\_id)

values ('$name', '$surname', '$birthdate', '$email', '$password', '$category', '1', $description, '$hospital\_id', $department\_id )";

$perform = mysqli\_query($this->connection, $query);

if($perform){

return true;

}

return false;

}

public function updateUser(User $user) {

$id = $user->getId();

$name = $user->getName();

$surname = $user->getSurname();

$birthdate = $user->getBirthdate();

$email = $user->getEmail();

$password = $user->getPassword();

$category = $user->getCategory();

$status = $user->getStatus();

$description = $user->getDescription();

$hospital\_id = $user->getHospitalId();

$department\_id = $user->getDepartmentId();

if(!empty($description)){

$description = "'$description'";

} else {

$description = "null";

}

if(!empty($hospital\_id)){

$hospital\_id = "'$hospital\_id'";

} else{

$hospital\_id = "null";

}

if(!empty($department\_id)){

$department\_id = "'$department\_id'";

} else {

$department\_id = "null";

}

$query = "Update users set name = '$name', surname = '$surname', birthdate = '$birthdate',

email = '$email', password = '$password', category = '$category',

status = '$status', description = $description, hospital\_id = $hospital\_id,

department\_id = $department\_id where id = '$id'";

$perform = mysqli\_query($this->connection, $query);

if($perform){

return true;

}

return false;

}

public function deleteUser($user\_id) {

$query = "Delete from users where id = '$user\_id'";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

return true;

}

return false;

}

public function getUsersByHospitalAndDepartment($hospital\_id, $department\_id) {

$query = "SELECT users.id as user\_id,

users.name as user\_name,

users.surname as user\_surname,

users.birthdate as user\_birthdate,

users.email as user\_email,

users.password as user\_password,

users.category as user\_category,

users.status as user\_status,

users.description as user\_description,

hospitals.id as hospital\_id,

hospitals.name as hospital\_name,

hospitals.address as hospital\_address,

hospitals.level as hospital\_level,

hospitals.phone as hospital\_phone,

departments.id as department\_id,

departments.name as department\_name

from users, hospitals, departments

where users.hospital\_id = hospitals.id

and users.department\_id = departments.id

and hospitals.id = '$hospital\_id'

and departments.id = '$department\_id'";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

$user\_data = [];

while ($result = mysqli\_fetch\_assoc($perform)) {

$user\_id = $result['user\_id'];

$user\_name = $result['user\_name'];

$user\_surname = $result['user\_surname'];

$user\_birthdate = $result['user\_birthdate'];

$user\_email = $result['user\_email'];

$user\_password = $result['user\_password'];

$user\_category = $result['user\_category'];

$user\_status = $result['user\_status'];

$user\_description = $result['user\_description'];

$hospital\_id = $result['hospital\_id'];

$hospital\_name = $result['hospital\_name'];

$hospital\_address = $result['hospital\_address'];

$hospital\_level = $result['hospital\_level'];

$hospital\_phone = $result['hospital\_phone'];

$department\_id = $result['department\_id'];

$department\_name = $result['department\_name'];

$user\_data[] = [

'user' => new User($user\_id, $user\_name, $user\_surname, $user\_birthdate, $user\_email, $user\_password, $user\_category, $user\_status, $user\_description, $hospital\_id, $department\_id),

'hospital' => new Hospital($hospital\_id, $hospital\_name, $hospital\_address, $hospital\_level, $hospital\_phone),

'department' => new Department($department\_id, $department\_name, $hospital\_id)

];

}

return $user\_data;

}

return false;

}

public function getDoctorWithoutVisitOverlap($hospital\_id, $department\_id = null, $start, $data = [], $user\_id) {

$dataInput = implode("', '", $data);

$dataQuery = " and visits.start\_time in ('$dataInput')";

$departmentQuery = "";

if(!empty($department\_id)) {

$departmentQuery = " and department\_id = '$department\_id'";

}

$query = "Select \* from users where hospital\_id = '$hospital\_id'

{$departmentQuery}

and category = 'd'

and id != '$user\_id'

and not exists (

SELECT '1' from visits where visits.start\_time >= '$start' and visits.doctor\_id = users.id

{$dataQuery}

{$departmentQuery} limit 1

)

";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

$doctors = [];

while ($result = mysqli\_fetch\_assoc($perform)) {

$id = $result['id'];

$name = $result['name'];

$surname = $result['surname'];

$birthdate = $result['birthdate'];

$email = $result['email'];

$password = $result['password'];

$category = $result['category'];

$status = $result['status'];

$description = $result['description'];

$hospital\_id = $result['hospital\_id'];

$department\_id = $result['department\_id'];

$doctors[] = new User($id, $name, $surname, $birthdate, $email, $password, $category, $status, $description, $hospital\_id, $department\_id);

}

return $doctors;

}

return false;

}

public function getDoctors() {

$query = "Select \* from users where category = 'd' and department\_id is not null";

$perform = mysqli\_query($this->connection, $query);

if($perform){

$doctors = [];

while ($result = mysqli\_fetch\_assoc($perform)) {

$id = $result['id'];

$name = $result['name'];

$surname = $result['surname'];

$birthdate = $result['birthdate'];

$email = $result['email'];

$password = $result['password'];

$category = $result['category'];

$status = $result['status'];

$description = $result['description'];

$hospital\_id = $result['hospital\_id'];

$department\_id = $result['department\_id'];

$doctors[] = new User($id, $name, $surname, $birthdate, $email, $password, $category, $status, $description, $hospital\_id, $department\_id);

}

return $doctors;

}

return false;

}

}

Validation

<?php

function check\_name($name){

$pattern = "/^[A-Z]{1}[a-z]+$/";

return preg\_match($pattern, $name);

}

function check\_email($email){

$pattern = '/^[a-z\d\.\_-]+@([a-z\d-]+\.)+[a-z]{2,6}$/i';

return preg\_match($pattern, $email);

}

function check\_username($username){

$pattern = "/^[a-zA-Z]{1}[a-zA-Z0-9\_-]+$/";

return preg\_match($pattern, $username);

}

function check\_phone($phone){

$pattern = "/^[0-9]{12}$/";

return preg\_match($pattern, $phone);

}

function check\_password($password){

$pattern = "/^[a-zA-Z0-9\_-]+$/";

return preg\_match($pattern, $password);

}

function check\_description($des){

$pattern = "/^[a-zA-Z0-9\s\.,]+$/";

return preg\_match($pattern, $des);

}

.

VisitDetailsQueries

<?php

require\_once dirname(dirname(\_\_FILE\_\_))."/database/Database.php";

require\_once dirname(dirname(\_\_FILE\_\_))."/database/VisitDetail.php";

class VisitDetailsQueries

{

private $connection;

public function \_\_construct()

{

$database = new Database();

$this->connection = $database->getConnection();

}

public function getVisitDetailByVisitId($visit\_id){

$query = "Select \* from visit\_details where visit\_id = '$visit\_id'";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

if(mysqli\_num\_rows($perform) == 1) {

$result = mysqli\_fetch\_assoc($perform);

$id = $result['id'];

$visit\_id = $result['visit\_id'];

$symptoms = $result['symptoms'];

$recipe = $result['recipe'];

return new VisitDetail($id, $visit\_id, $symptoms, $recipe);

}

return null;

}

return false;

}

public function createVisitDetail(VisitDetail $visitDetail){

$visit\_id = $visitDetail->getVisitId();

$symptoms = $visitDetail->getSymptoms();

$recipe = $visitDetail->getRecipe();

$query = "Insert into visit\_details(visit\_id, symptoms, recipe)

VALUES ('$visit\_id', '$symptoms', '$recipe')";

$perform = mysqli\_query($this->connection, $query);

if($perform){

return true;

}

return false;

}

public function updateVisitDetail(VisitDetail $visitDetail){

$id = $visitDetail->getId();

$symptoms = $visitDetail->getSymptoms();

$recipe = $visitDetail->getRecipe();

$query = "Update visit\_details set symptoms = '$symptoms',

recipe = '$recipe' where id = '$id'";

$perform = mysqli\_query($this->connection, $query);

if($perform){

return true;

}

return false;

}

public function deleteVisitDetailsByPatientId($patient\_id) {

$query = "Delete from visit\_details where visit\_id in (

select id from visits where patient\_id = '$patient\_id'

)";

$perform = mysqli\_query($this->connection, $query);

if($perform){

return true;

}

return false;

}

public function deleteVisitDetailsByVisitId($visit\_id) {

$query = "Delete from visit\_details where visit\_id = '$visit\_id'";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

return true;

}

return false;

}

}

VisitPdf

<?php

require\_once dirname(\_\_FILE\_\_)."/fpdf16/fpdf.php";

class VisitPdf extends FPDF

{

function \_\_construct ($orientation = 'P', $unit = 'pt', $format = 'Letter', $margin = 40) {

$this->FPDF($orientation, $unit, $format);

$this->SetTopMargin($margin);

$this->SetLeftMargin($margin);

$this->SetRightMargin($margin);

$this->SetAutoPageBreak(true, $margin);

}

function Header() {

$this->SetFont('Arial', 'B', 20);

$this->SetFillColor(36, 96, 84);

$this->SetTextColor(225);

$this->Cell(0, 30, "Hospital CMS", 0, 1, 'C', true);

}

function Footer() {

$this->SetFont('Arial', '', 12);

$this->SetTextColor(0);

$this->SetXY(0,-60);

$this->Cell(0, 20, "", 'T', 0, 'C');

}

}

VisitQueries

<?php

require\_once dirname(dirname(\_\_FILE\_\_))."/database/Database.php";

require\_once dirname(dirname(\_\_FILE\_\_))."/database/Visit.php";

class VisitQueries

{

private $connection;

public function \_\_construct()

{

$database = new Database();

$this->connection = $database->getConnection();

}

public function getDoctorVisits($doctor\_id, $timestamp = null, $data = null) {

$timeQuery = "";

if(!empty($timestamp)) {

$timeQuery = " and start\_time >= '$timestamp'";

}

$dataQuery = "";

if(!empty($data)) {

$dataInput = implode("', '", $data);

$dataQuery = " and start not in('$dataInput')";

}

$query = "Select \* from visits where doctor\_id = '$doctor\_id'" . $timeQuery . $dataQuery;

$perform = mysqli\_query($this->connection, $query);

if($perform) {

$visits = [];

while ($result = mysqli\_fetch\_assoc($perform)) {

$id = $result['id'];

$doctor\_id = $result['doctor\_id'];

$patient\_id = $result['patient\_id'];

$start = $result['start\_time'];

$end = $result['end\_time'];

$status = $result['status'];

$parent\_id = $result['parent\_id'];

$case\_of\_delievery = $result['case\_of\_delivery'];

$token = $result['token'];

$visits[] = new Visit($id, $doctor\_id, $patient\_id, $start, $end, $status, $parent\_id, $case\_of\_delievery, $token);

}

return $visits;

}

return false;

}

public function getDoctorVisitForDate($doctor\_id, $date = null) {

if(empty($date)) {

$date = date('Y-m-d') . " 00:00:00";

} else {

$date .= " 00:00:00";

}

$end\_date = new DateTime($date);

$end\_date = $end\_date->modify("+1 days");

$end\_date = $end\_date->format('Y-m-d H:i:s');

$query = "Select visits.\*, (

SELECT CONCAT(

'{',

'\"name\":', CONCAT('\"',patients.name,'\"'), ',',

'\"surname\":', CONCAT('\"', patients.surname, '\"'),

'}'

) from patients where patients.id = visits.patient\_id limit 1

) as patient from visits where doctor\_id = '$doctor\_id' and start\_time >= '$date' and end\_time <= '$end\_date' order by start\_time";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

$visit\_data = [];

while ($result = mysqli\_fetch\_assoc($perform)){

$id = $result['id'];

$doctor\_id = $result['doctor\_id'];

$patient\_id = $result['patient\_id'];

$start = $result['start\_time'];

$end = $result['end\_time'];

$status = $result['status'];

$parent\_id = $result['parent\_id'];

$case\_of\_delievery = $result['case\_of\_delivery'];

$token = $result['token'];

$patient\_data = $result['patient'];

$patient = json\_decode($patient\_data, true);

$visit\_data[] = [

'visit' => new Visit($id, $doctor\_id, $patient\_id, $start, $end, $status, $parent\_id, $case\_of\_delievery, $token),

'patient' => $patient

];

}

return $visit\_data;

}

return false;

}

public function getPatientVisitByDate($patient\_id, $hospital\_id, $date = null){

if(empty($date)) {

$date = date('Y-m-d') . " 00:00:00";

} else {

$date .= " 00:00:00";

}

$end\_date = new DateTime($date);

$end\_date = $end\_date->modify("+1 days");

$end\_date = $end\_date->format('Y-m-d H:i:s');

$query = "select visits.\*, (

SELECT CONCAT(

'{',

'\"name\":', CONCAT('\"',users.name,'\"'), ',',

'\"surname\":', CONCAT('\"', users.surname, '\"'),

'}'

) from users where users.id = visits.doctor\_id

) as doctor from visits

where patient\_id ='$patient\_id'

and start\_time >= '$date'

and end\_time <= '$end\_date'

and visits.doctor\_id in(

SELECT id from users where hospital\_id = '$hospital\_id'

) order by start\_time";

$perform = mysqli\_query($this->connection, $query);

if($perform){

$visit\_data = [];

while ($result = mysqli\_fetch\_assoc($perform)){

$id = $result['id'];

$doctor\_id = $result['doctor\_id'];

$patient\_id = $result['patient\_id'];

$start = $result['start\_time'];

$end = $result['end\_time'];

$status = $result['status'];

$parent\_id = $result['parent\_id'];

$case\_of\_delievery = $result['case\_of\_delivery'];

$token = $result['token'];

$doctor\_data = $result['doctor'];

$doctor = json\_decode($doctor\_data, true);

$visit\_data[] = [

'visit' => new Visit($id, $doctor\_id, $patient\_id, $start, $end, $status, $parent\_id, $case\_of\_delievery, $token),

'doctor' => $doctor

];

}

return $visit\_data;

}

return false;

}

public function transferVisit($visitDoctorId, $doctorIdToTransfer, $start) {

$query = "Update visits set doctor\_id = '$doctorIdToTransfer' where start\_time >= '$start' and doctor\_id = '$visitDoctorId'";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

return true;

}

return false;

}

public function deletePatientVisit($id, $start = null) {

$start\_query = "";

if(!empty($start)) {

$start\_query = " and start >= '$start'";

}

$query = "Delete from visits where patient\_id = '$id'". $start\_query;

$perform = mysqli\_query($this->connection, $query);

if($perform){

return true;

}

return false;

}

public function getVisitById($id, $hospital\_id = null, $without\_children = false) {

$hospital\_query = "";

if(!empty($hospital\_id)){

$hospital\_query = " and exists (

Select 1

from users where visits.doctor\_id = users.id

and users.hospital\_id = '$hospital\_id'

limit 1

)";

}

$child\_query = "";

if($without\_children) {

$child\_query = " and not exists (

Select 1

from visits v

where v.parent\_id = '$id'

limit 1

)";

}

$query = "Select \* from visits where id = '$id'" . $hospital\_query . $child\_query;

$perform = mysqli\_query($this->connection, $query);

if($perform){

if(mysqli\_num\_rows($perform) == 1) {

$result = mysqli\_fetch\_assoc($perform);

$id = $result['id'];

$doctor\_id = $result['doctor\_id'];

$patient\_id = $result['patient\_id'];

$start = $result['start\_time'];

$end = $result['end\_time'];

$status = $result['status'];

$parent\_id = $result['parent\_id'];

$case\_of\_delievery = $result['case\_of\_delivery'];

$token = $result['token'];

return new Visit($id, $doctor\_id, $patient\_id, $start, $end, $status, $parent\_id, $case\_of\_delievery, $token);

}

return null;

}

return false;

}

public function createVisit(Visit $visit) {

$doctor\_id = $visit->getDoctorId();

$patient\_id = $visit->getPatientId();

$start = $visit->getStart();

$end = $visit->getEnd();

$token = self::generateRandomString(20);

$parent\_id = $visit->getParentId();

if(is\_null($parent\_id)) {

$parent\_id\_q = "null";

} else {

$parent\_id\_q = "'$parent\_id'";

}

$query = "Insert into visits(doctor\_id, patient\_id, start\_time, end\_time, token, parent\_id)

VALUES ('$doctor\_id', '$patient\_id', '$start', '$end', '$token', $parent\_id\_q)";

$perform = mysqli\_query($this->connection, $query);

if($perform){

return true;

}

var\_dump($query);

var\_dump(mysqli\_error\_list($this->connection));

return false;

}

public function deleteVisit($id, $hospital\_id = null) {

$hospital\_query = "";

if (!empty($hospital\_id)){

$hospital\_query = " and exists(

Select 1

from users where visits.doctor\_id = users.id

and users.hospital\_id = '$hospital\_id'

limit 1

)";

}

$query = "Delete from visits where id = '$id'" . $hospital\_query;

$perform = mysqli\_query($this->connection, $query);

if($perform) {

return true;

}

return false;

}

public function updateVisit(Visit $visit){

$id = $visit->getId();

$status = $visit->getStatus();

$case\_of\_delivery = $visit->getCaseOfDelivery();

$query = "Update visits set status = '$status', case\_of\_delivery = '$case\_of\_delivery' where id = '$id'";

$perform = mysqli\_query($this->connection, $query);

if($perform){

return true;

}

var\_dump(mysqli\_error\_list($this->connection));

return false;

}

public function getVisitStats() {

$date = date('Y');

$query = "Select MONTH(start\_time) as month, count(id) as number

from visits

where YEAR(start\_time) = '$date'

group by MONTH(start\_time)";

$perform = mysqli\_query($this->connection, $query);

if($perform) {

$data\_to\_return = [];

for ($i = 1; $i <= 12; $i++){

$data\_to\_return["$i"] = 0;

}

while ($result = mysqli\_fetch\_assoc($perform)) {

$month = $result['month'];

$number = $result['number'];

$data\_to\_return["$month"] = $number;

}

return $data\_to\_return;

}

return false;

}

private function generateRandomString($length = 10) {

$characters = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';

$charactersLength = strlen($characters);

$randomString = '';

for ($i = 0; $i < $length; $i++) {

$randomString .= $characters[rand(0, $charactersLength - 1)];

}

return $randomString;

}

}

6. Project Planning

**Project Name:** MMS (Medical Management System)

**Members:** Jaser Sokoli, Alvi Lika, Neritan Hoxha, Kris Ferko, Rigels Hita, Ibrahim Pisha

**Start Day:** 13.03.2018

**End Day:** 3.06.2017

**Total Days:** 88 Days

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr | Activity | Duration | Dependencies | Worked |
| 0 | Proposed topics for project | 7 |  | All members |
| 1 | Technology | 6 | 1 | All members |
| 2 | Project description | 4 | 1 | All members |
| 3 | Structure, functionality | 7 | 3 | All members |
| 4 | Assignment on requirements (specification) | 3 | 4 | All members |
| 5 | Sequence & collaboration diagrams | 7 | 4,5 | All members |
| 6 | State diagrams | 4 | 5,6 | All members |
| 7 | Behavior of objects in use cases | 3 | 5,6,7 | All members |
| 8 | Structure of the database, the main tables | 17 | 8 | All members |
| 9 | Login & register user | 7 | 9 | All members |
| 10 | Design | 9 | 10 | All members |
| 11 | Crud interface | 5 | 9 | All members |
| 12 | Implementation technology &task network | 2 | 2,6,7 | All members |
| 13 | Profiles | 3 | 10,12 | All members |
| 14 | Appendix & finish code | 4 | 11,13,14 | All members |

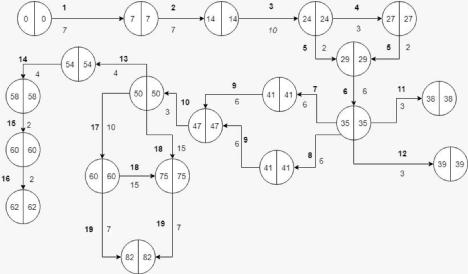
***Gantt Chart***

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **NR** | **Activity** | **Week 1** | **Week 2** | **Week 3** | **Week 4** | **Week 5** | **Week 6** | **Week 7** | **Week 8** | **Week 9** | **Week 10** |
| 1 | Proposed topic for the project |  |  |  |  |  |  |  |  |  |  |
| 2 | Research |  |  |  |  |  |  |  |  |  |  |
| 3 | Requirements analysis |  |  |  |  |  |  |  |  |  |  |
| 4 | User scenarios |  |  |  |  |  |  |  |  |  |  |
| 5 | User cases |  |  |  |  |  |  |  |  |  |  |
| 6 | Use cases diagrams |  |  |  |  |  |  |  |  |  |  |
| 7 | Activity diagrams |  |  |  |  |  |  |  |  |  |  |
| 8 | State diagrams |  |  |  |  |  |  |  |  |  |  |
| 9 | Sequence diagrams |  |  |  |  |  |  |  |  |  |  |
| 10 | Collaboration diagrams |  |  |  |  |  |  |  |  |  |  |
| 11 | ERD |  |  |  |  |  |  |  |  |  |  |
| 12 | DFD |  |  |  |  |  |  |  |  |  |  |
| 13 | Class diagrams |  |  |  |  |  |  |  |  |  |  |
| 14 | Object diagram |  |  |  |  |  |  |  |  |  |  |
| 15 | Component Diagram |  |  |  |  |  |  |  |  |  |  |
| 16 | Deployment Diagram |  |  |  |  |  |  |  |  |  |  |
| 17 | Documentation |  |  |  |  |  |  |  |  |  |  |

.

**Network Diagram**

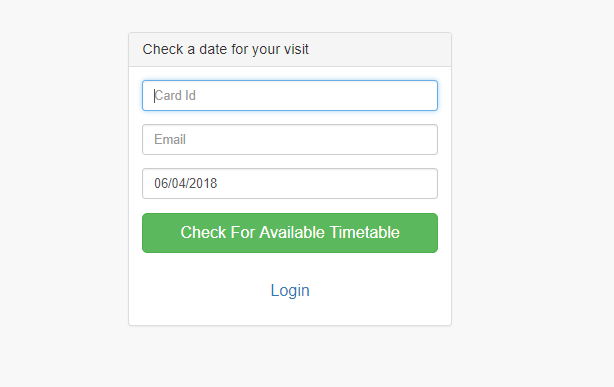
To visualize the planning process and the critical sections of the development, the following diagram has been used.



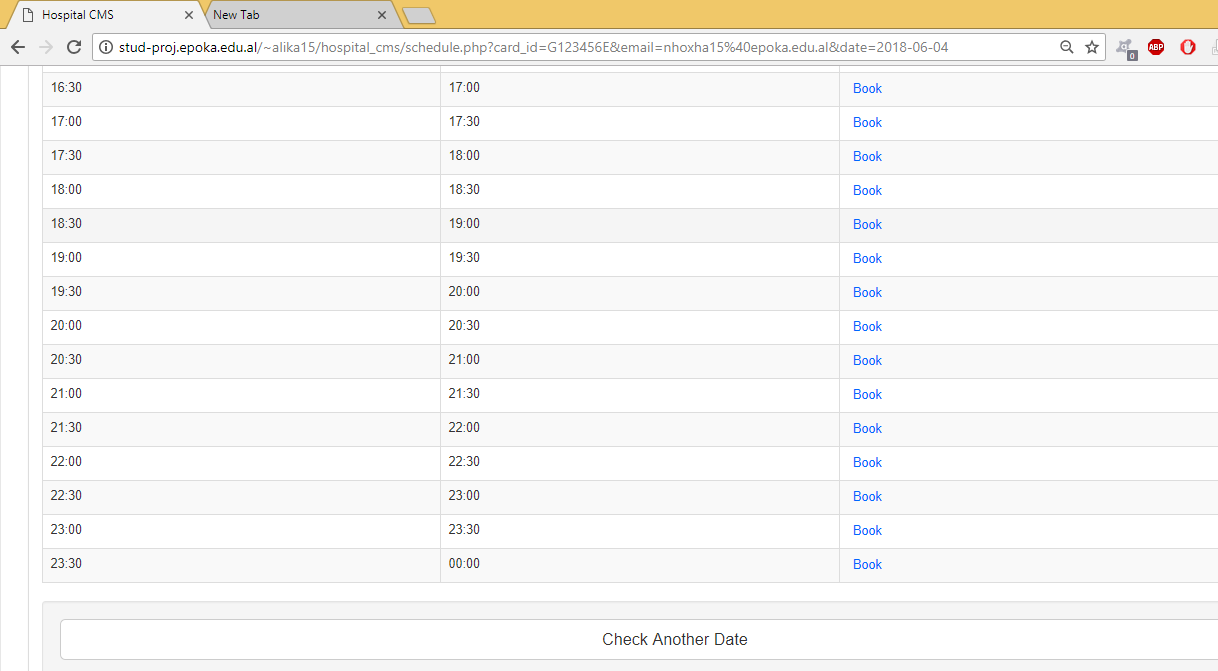
APPENDIX

Screenshots:

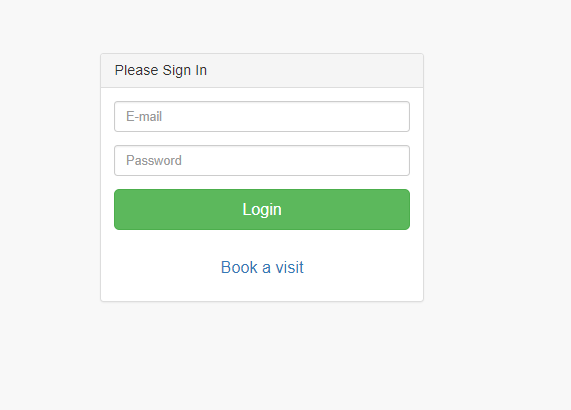
Patient/Log in



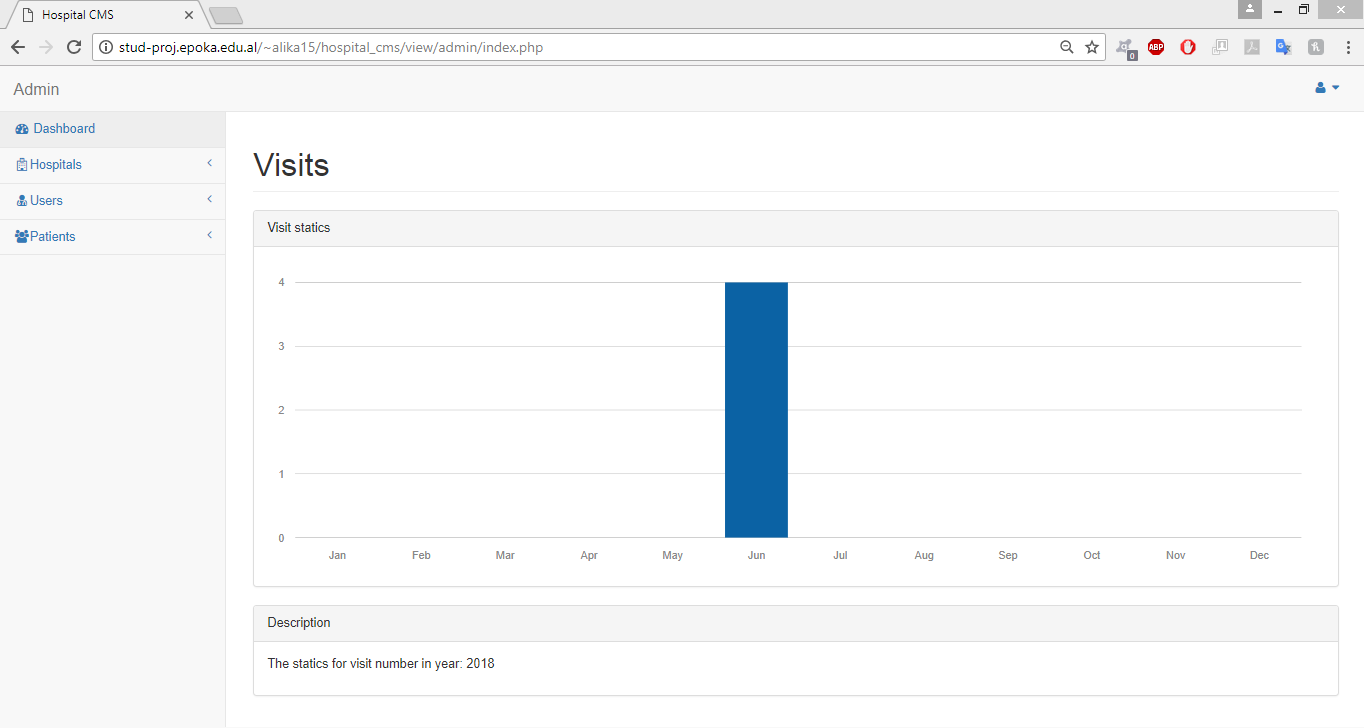
Patient/Booking



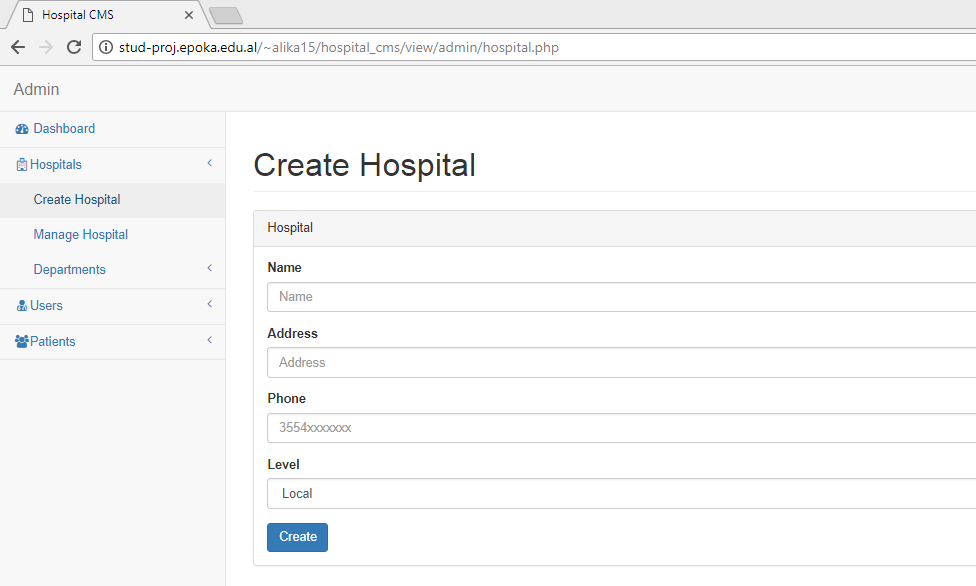
User/Log in



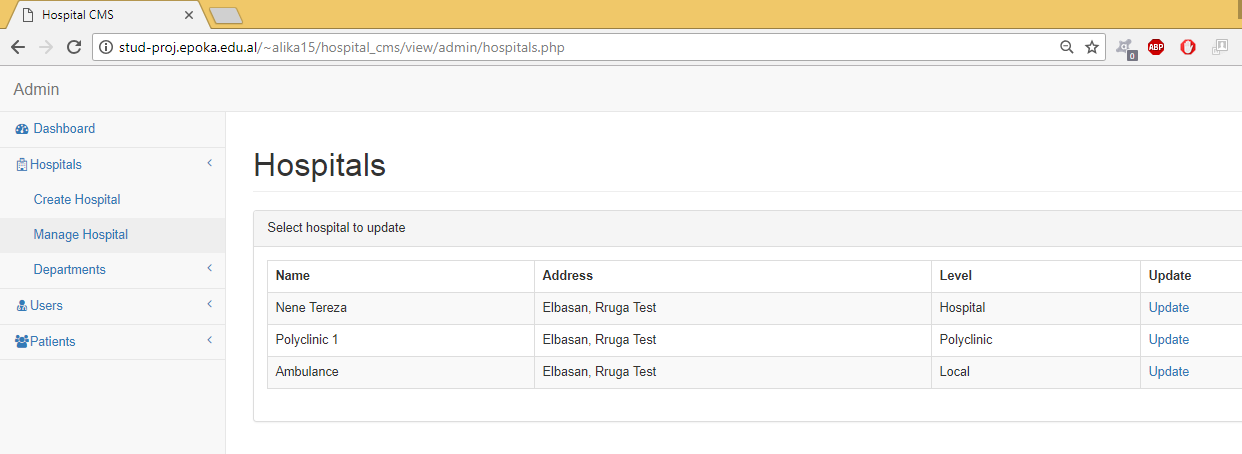
Admin/Dashboard



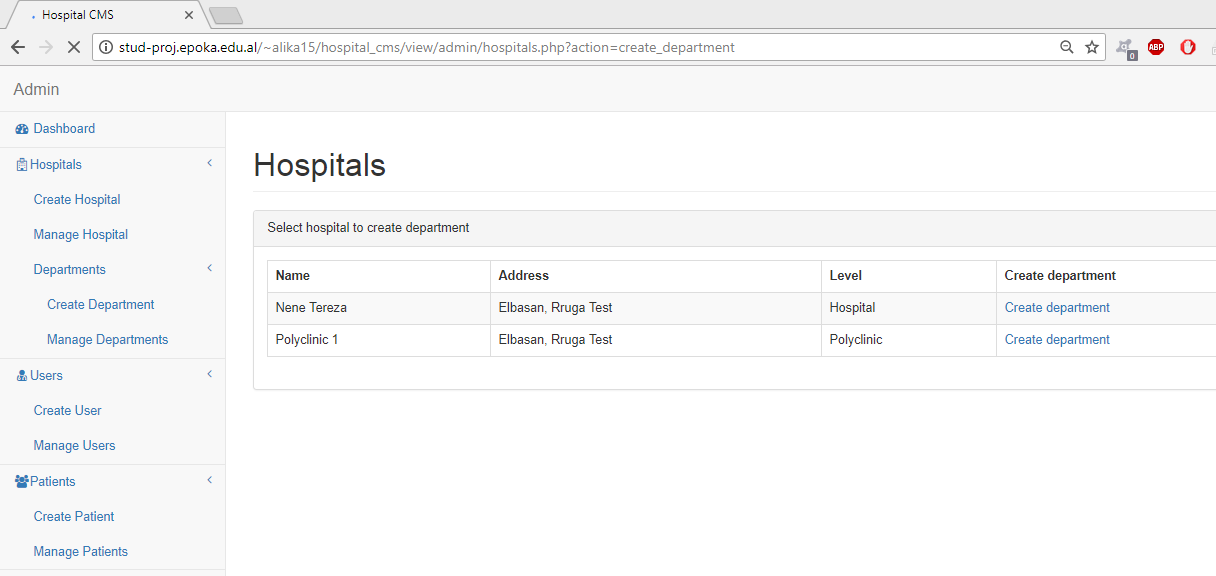
Admin/Create Hospital



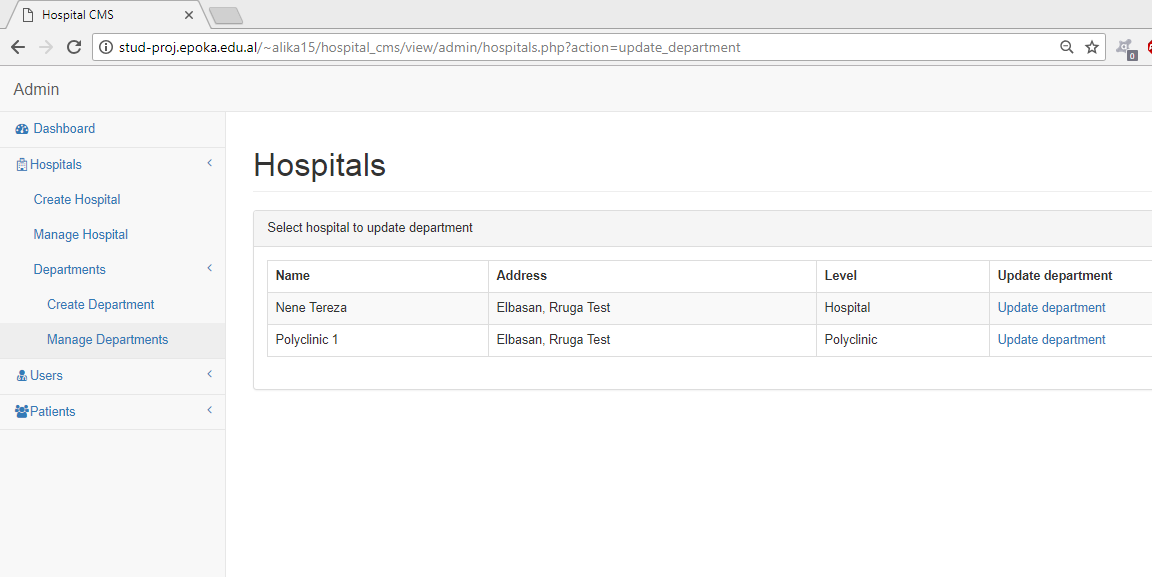
Admin/Manage Hospitals



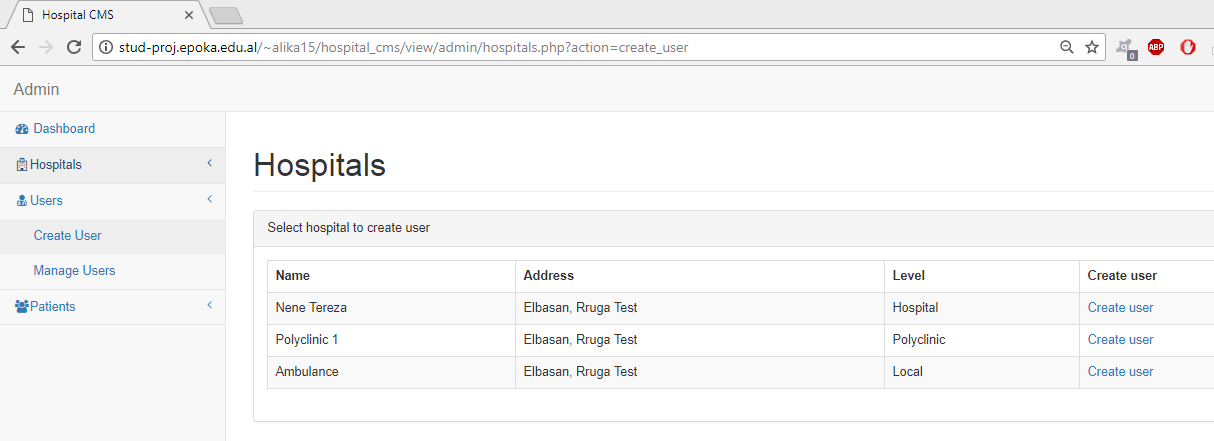
Admin/Create Department



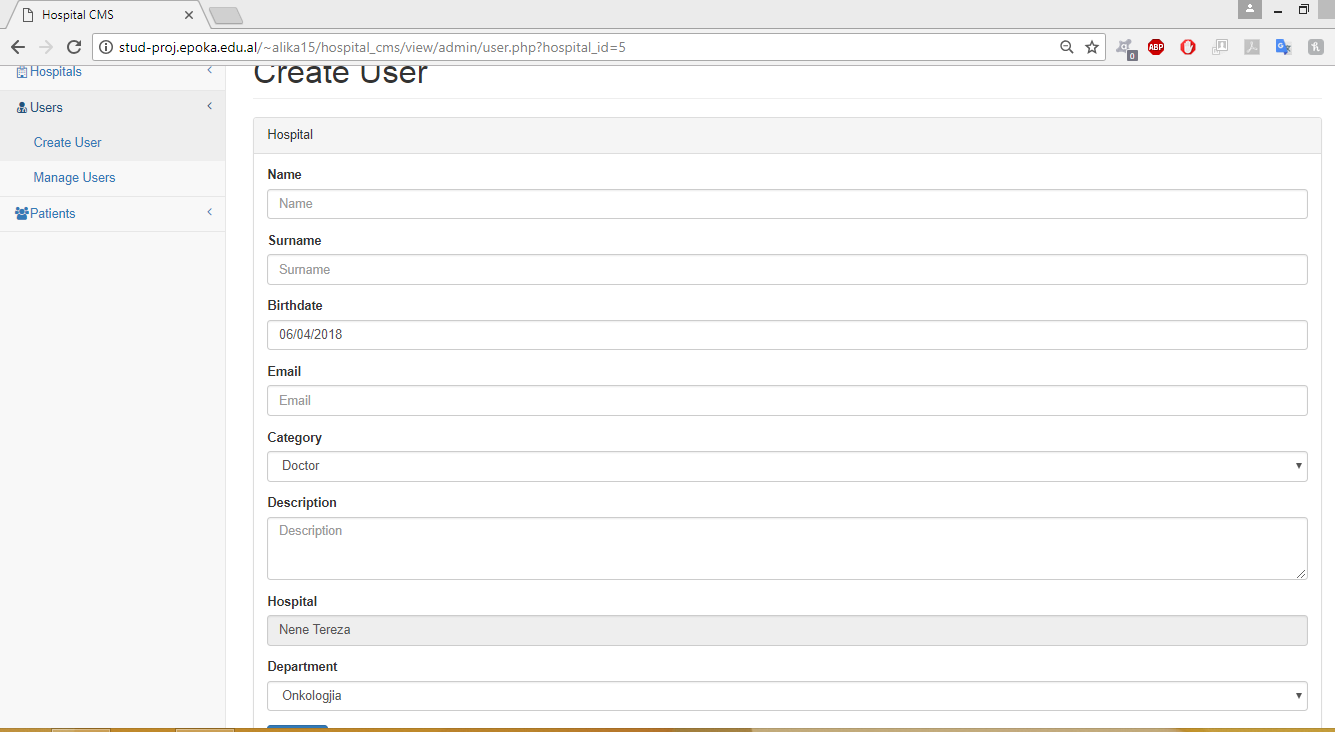
Admin/Manage Department



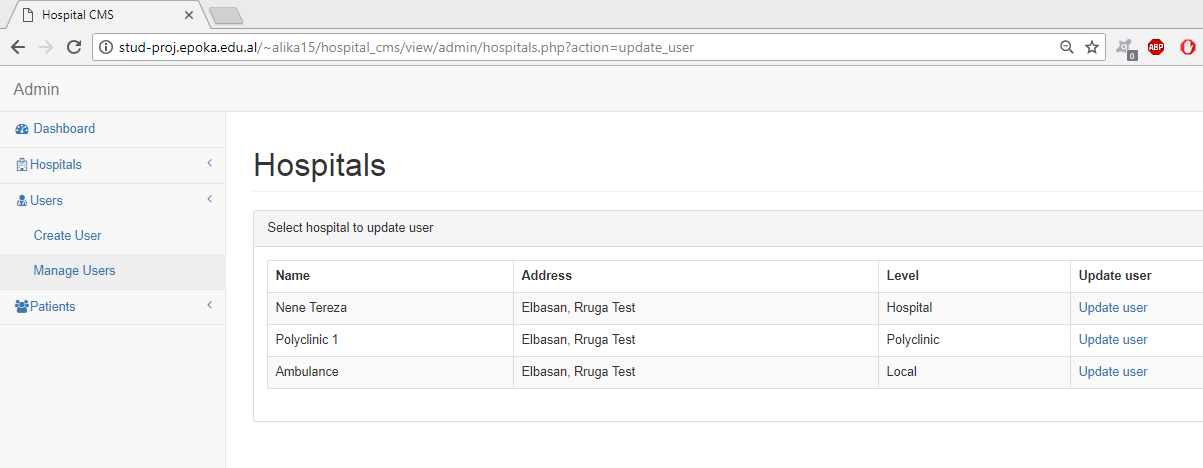
Admin/Create User



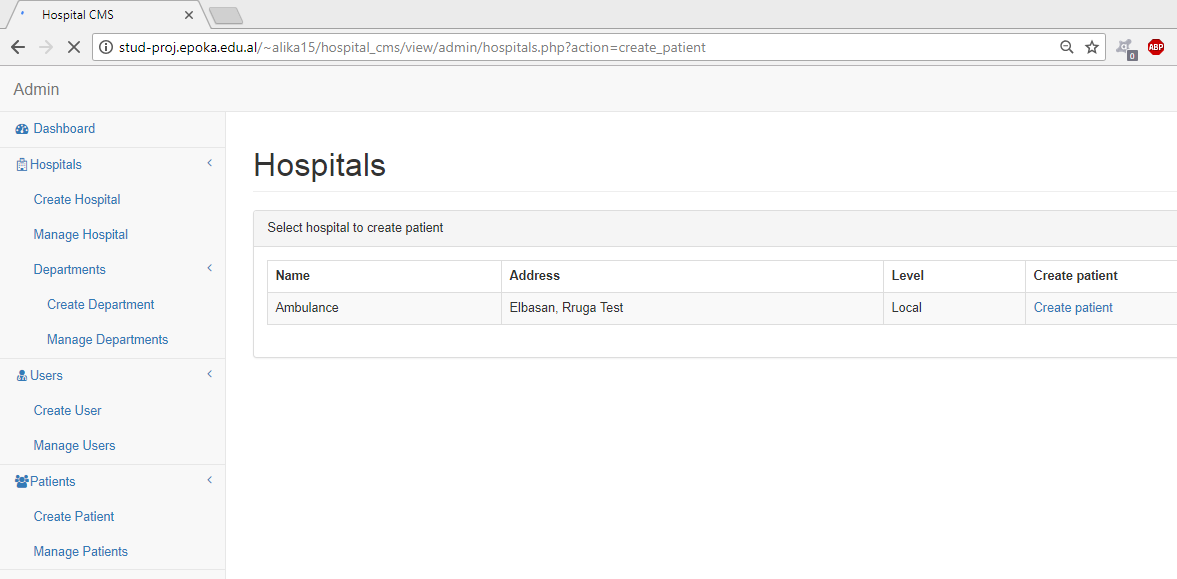
Admin/Create User Details



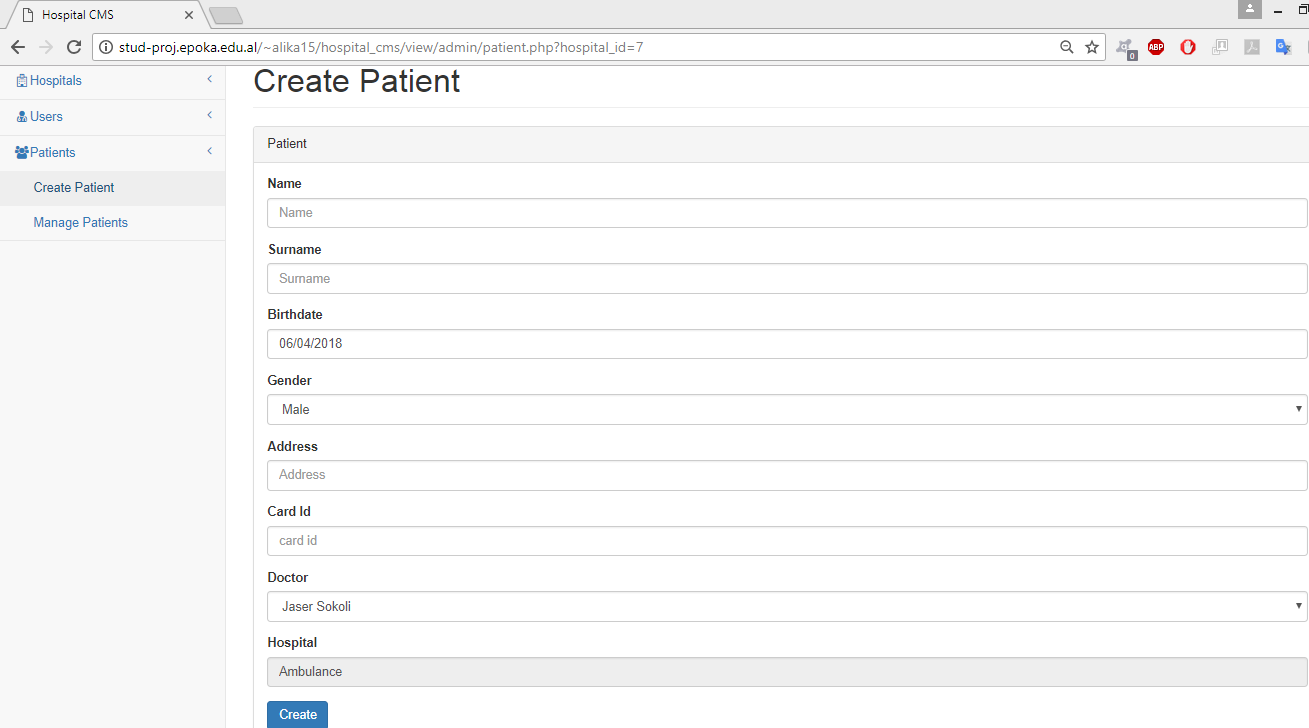
Admin/Manage User



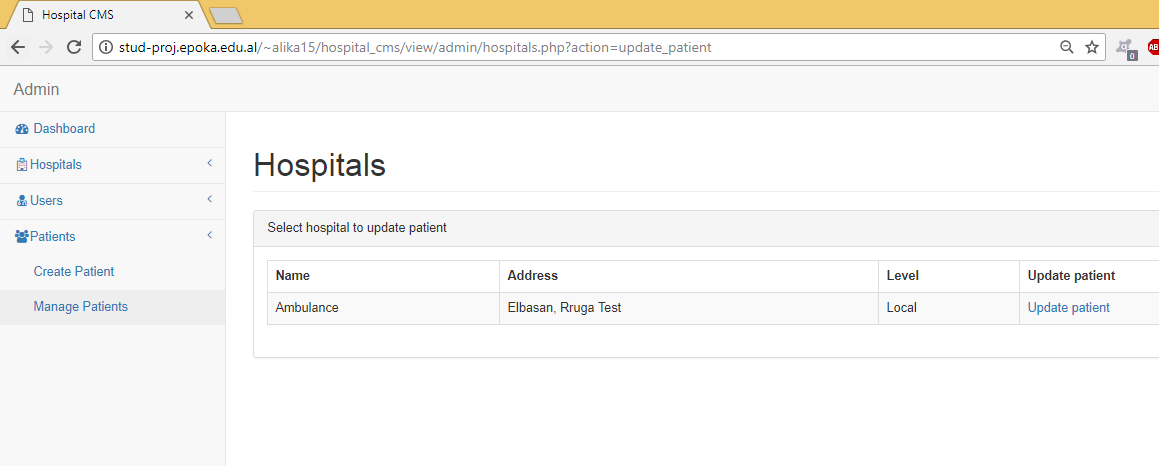
Admin/Create Patient



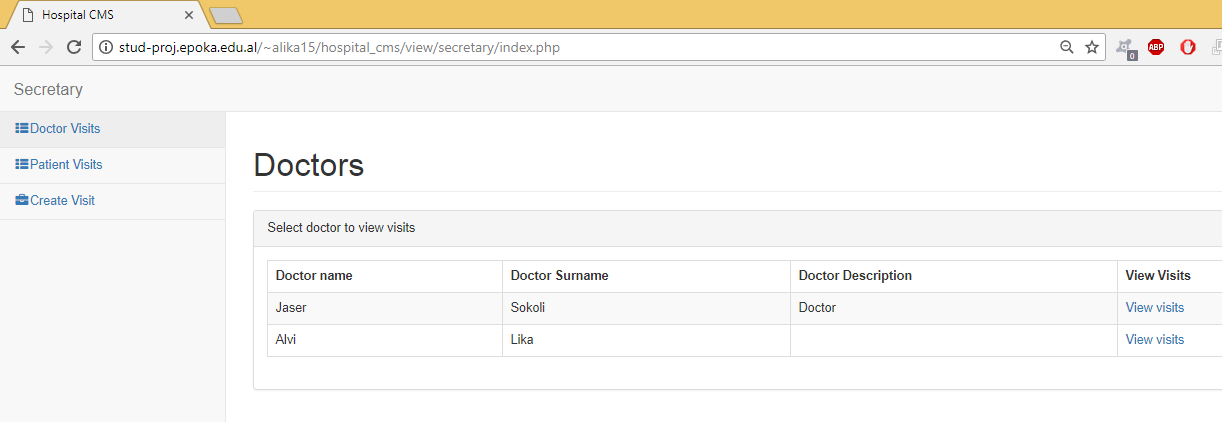
Admin/Create Patient Details



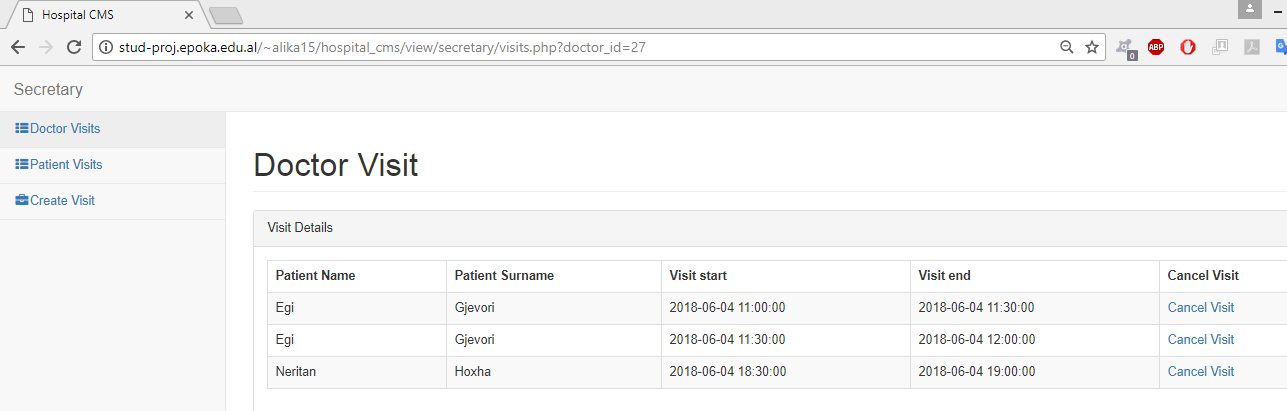
Admin/Manage Patient



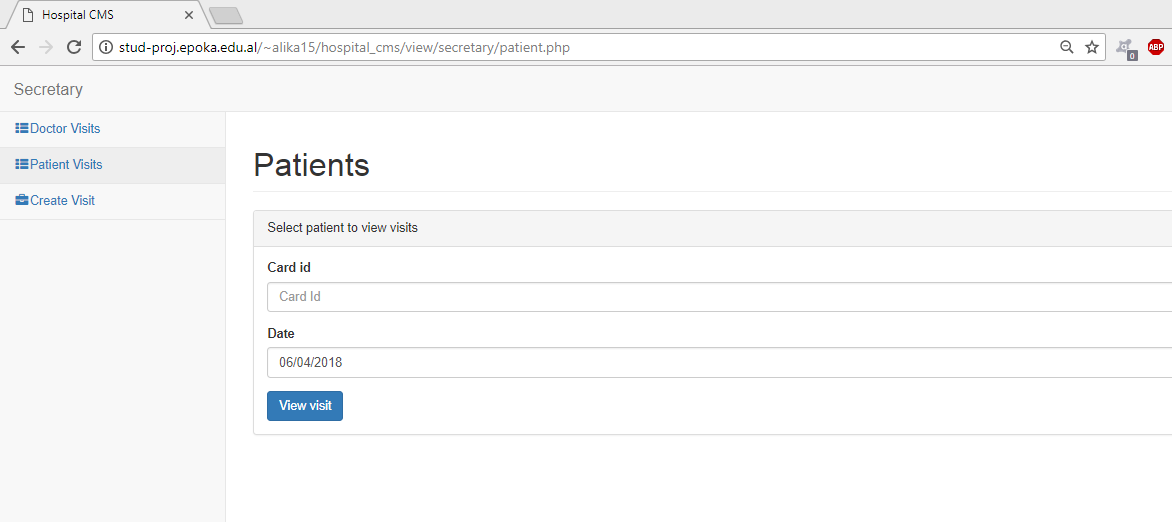
Secretary/Doctor Visits



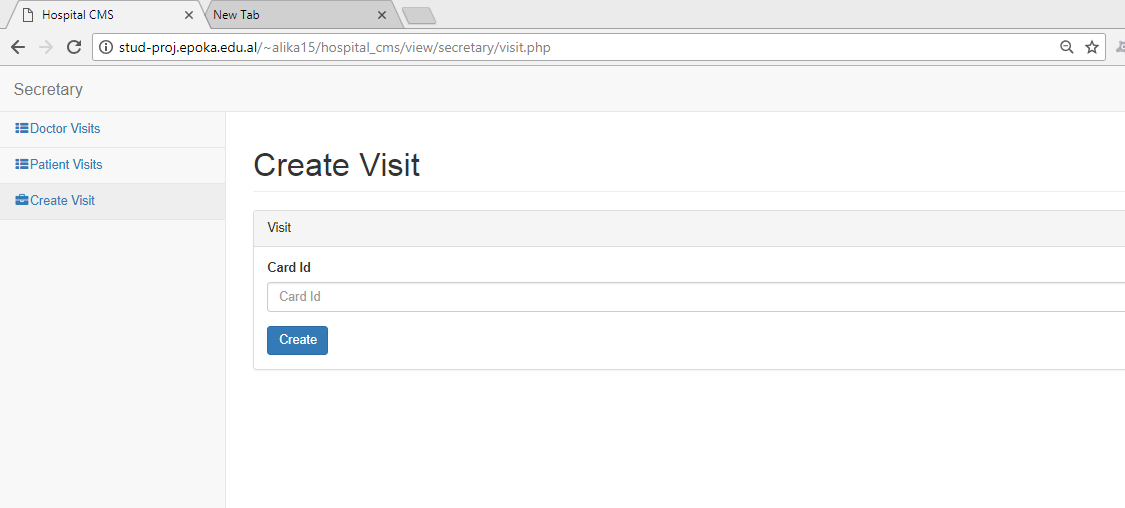
Secretary/Doctor Visit/Timetable

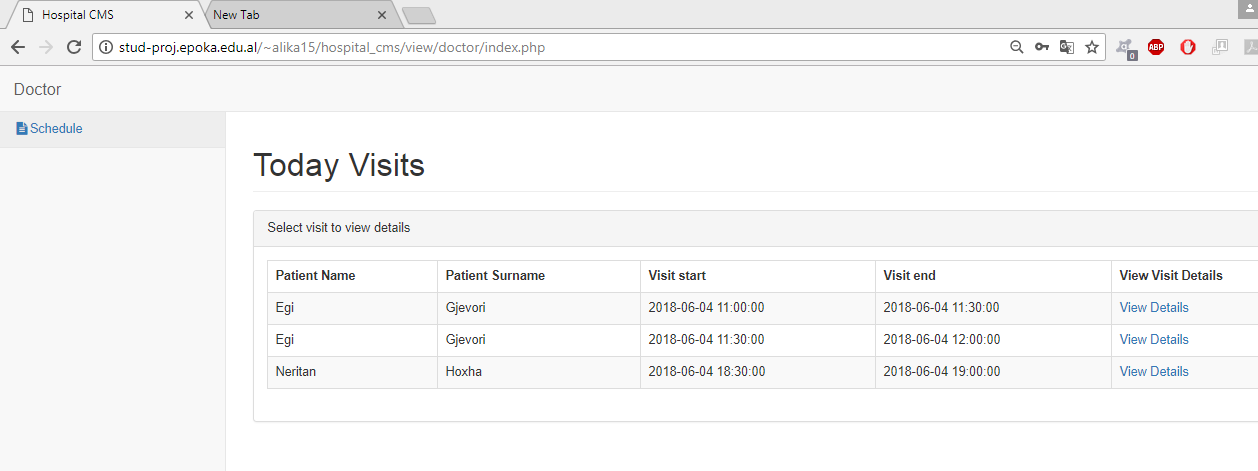


Secretary/Patient Visits



Secretart/Create Visit



Doctor/Today Visits

Doctor/Visit Details

