Zewail City of Science and Technology

مدينة زويل

ZEWAIL CITY
ESTABLISHED 2000
INAUGURATED 2011

مدينة زويل للعلوم والتكنولوجيا

CIE203

Software Engineering

StethoMan

Software Design Document

SIGSEGV

Mohamed Maher

CIE 203: Phase 1 – SIGSEGV
Project: <StethoMan>

Software Design Document

December & 2016

Contents

CIE 203: Phase 1 – SIGSEGV
Project: <StethoMan>

Software Design Document

Instructions [To be removed]

Team

ID	Name	Email	Mobile
201301463	Mohamed Maher	m.maher525@gmail.com	01141763525
201304603	Youhana Mourad	s-youhana.mikhaiel@zewailcity.edu.eg	01281596847
201304322	Omar Tarek Merghany	s-omar.merghany@zewailcity.edu.eg	01117496377
201305531	Randa Selim Negm	s-randa.negm@zewailcity.edu.eg	

Document Purpose and Audience

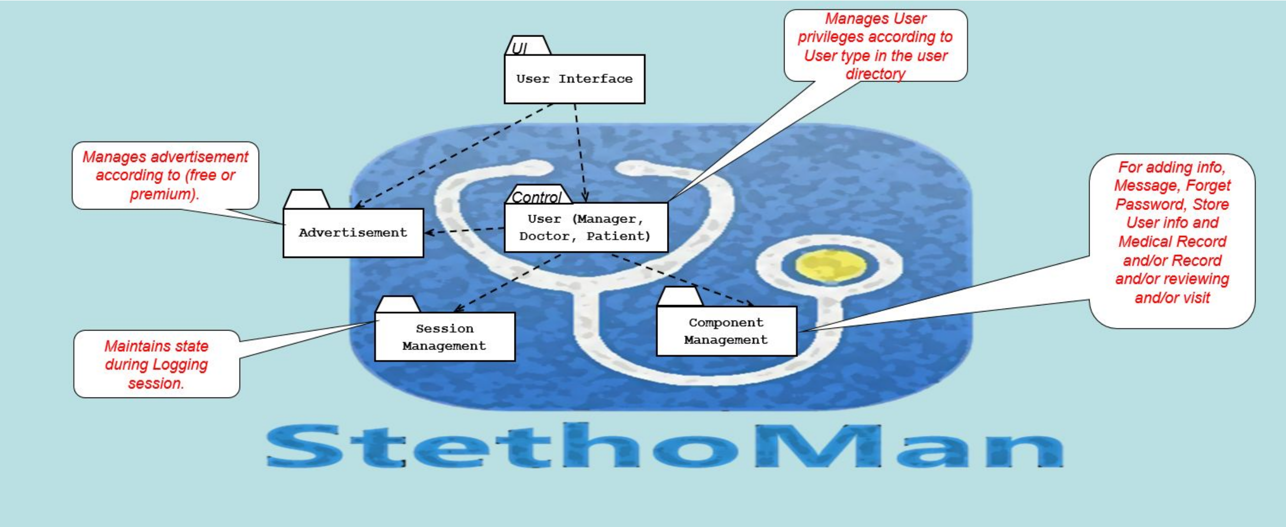
- **This document is the Software Design Document which contains documentation for design phase of the software and contains several diagrams like Class Diagram, Entity Relations Diagram, System decomposition diagram and used Algorithms pseudocode or flowcharts.**
- **It targets Manger and developers of the software in order to visualize the solution before implementation and to complete any missing details that need to be defined from the problem domain.**

CIE 203: Phase 1 – SIGSEGV
Project: <StethoMan>

Software Design Document

System Models

I. System Decomposition -- Software Used: Visual Paradigm



Manages User privileges according to User type in the user directory

Manages advertisement according to (free or premium).

For adding info, Message, Forget Password, Store User info and Medical Record and/or Record and/or reviewing and/or visit

Maintains state during Logging session.

Layers & Services:

- **For achieving runtime efficiency, we will use Open Architecture (Transparent Layering) as a relationship between the subsystems.**
- **Given that, subsystems should have as maximum cohesion and minimum coupling as possible**
- **To achieve Service Portability, Efficient Performance, Flexibility and Reliability we will use Client-Server Model**

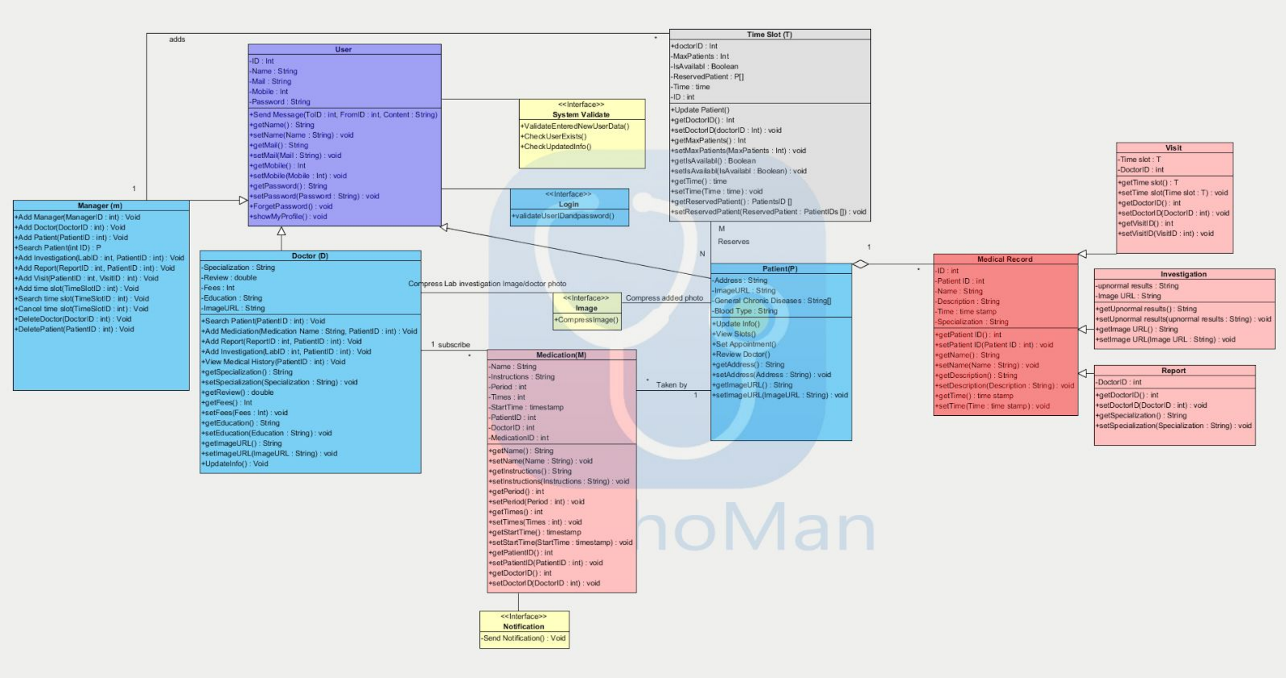
Layer number	Subsystem ID	Services & Description
Layer 1	User Interface	This subsystem is the interface that appears to the user, it contains part for User logging and/or Advertisement (in case of free trial)
Layer 2	Advertisement	This subsystem manages the advertisement appearance. ***(compile time dependency)***

Layer 2	User (Manager, Doctor, Patient)	This subsystem manages the user privileges and display the user interface. ***<u>(run time dependency)</u>*** It handles the case of Forget Password (which is requested from the user to the manager) and Medical Record (which stores results of archival records, the record is added by the Doctor or Manager only, and can be viewed by patient, doctor, manager) It also stores all the user data in the Database and contacts info it can be updated by the User himself or by the manager.
Layer 3	Component Management	This subsystem is responsible for adding info, Message and/or Record and/or reviewing and/or visit. It is also responsible for adding new Doctor, Patient, Manager in case the current user is a Manager. ***<u>(run time dependency)</u>***
Layer 3	Session Management	This subsystem maintains state during Logging session. It handles the changes between the user interface and the Database. ***<u>(compile time dependency)</u>***

CIE 203: Phase 1 – SIGSEGV
Project: <StethoMan>

Software Design Document

II. Class diagrams (download visual paradigm file from here) Or (Full Image Resolution) --- Software Used: Visual Paradigm


adds
1
*
User
-ID : Int
-Name : String
-Mail : String
-Mobile : Int
-Password : String
+Send Message(ToID : int, FromID : int, Content : String)
+getName() : String
+setName(Name : String) : void
+getMail() : String
+setMail(Mail : String) : void
+getMobile() : Int
+setMobile(Mobile : Int) : void
+getPassword() : String
+setPassword(Password : String) : void
+ForgetPassword() : void
+showMyProfile() : void
Manager (m)
+Add Manager(ManagerID : int) : Void
+Add Doctor(DoctorID : int) : Void
+Add Patient(PatientID : int) : Void
+Search Patient(int ID) : P
+Add Investigation(LabID : int, PatientID : int) : Void
+Add Report(ReportID : int, PatientID : int) : Void
+Add Visit(PatientID : int, VisitID : int) : Void
+Add time slot(TimeSlotID : int) : Void
+Search time slot(TimeSlotID : int) : Void
+Cancel time slot(TimeSlotID : int) : Void
+DeleteDoctor(DoctorID : int) : Void
+DeletePatient(PatientID : int) : Void
Doctor (D)
-Specialization : String
-Review : double
-Fees : Int
-Education : String
-ImageURL : String
+Search Patient(PatientID : int) : Void
+Add Mediciation(Medication Name : String, PatientID : int) : Void
+Add Report(ReportID : int, PatientID : int) : Void
+Add Investigation(LabID : int, PatientID : int) : Void
+View Medical History(PatientID : int) : Void
+getSpecialization() : String
+setSpecialization(Specialization : String) : void
+getReview() : double
+getFees() : Int
+setFees(Fees : Int) : void
+getEducation() : String
+setEducation(Education : String) : void
+getImageURL() : String
+setImageURL(ImageURL : String) : void
+UpdateInfo() : Void
<<Interface>>
System Validate
+ValidateEnteredNewUserData()
+CheckUserExists()
+CheckUpdatedInfo()
<<Interface>>
Login
+validateUserIDandpassword()
Time Slot (T)
+doctorID : Int
-MaxPatients : Int
-IsAvailabl : Boolean
-ReservedPatient : P[]
-Time : time
-ID : int
+Update Patient()
+getDoctorID() : Int
+setDoctorID(doctorID : Int) : void
+getMaxPatients() : Int
+setMaxPatients(MaxPatients : Int) : void
+getIsAvailabl() : Boolean
+setIsAvailabl(IsAvailabl : Boolean) : void
+getTime() : time
+setTime(Time : time) : void
+getReservedPatient() : PatientsID []
+setReservedPatient(ReservedPatient : PatientIDs []) : void
M
Reserves
N
1
Compress Lab investigation Image/doctor photo
<<Interface>>
Image
+CompressImage()
Compress added photo
Patient(P)
-Address : String
-ImageURL : String
-General Chronic Diseases : String[]
-Blood Type : String
+Update Info()
+View Slots()
+Set Appointment()
+Review Doctor()
+getAddress() : String
+setAddress(Address : String) : void
+getImageURL() : String
+setImageURL(ImageURL : String) : void
1 subscribe
*
Medication(M)
-Name : String
-Instructions : String
-Period : int
-Times : int
-StartTime : timestamp
-PatientID : int
-DoctorID : int
-MedicationID : int
+getName() : String
+setName(Name : String) : void
+getInstructions() : String
+setInstructions(Instructions : String) : void
+getPeriod() : int
+setPeriod(Period : int) : void
+getTimes() : int
+setTimes(Times : int) : void
+getStartTime() : timestamp
+setStartTime(StartTime : timestamp) : void
+getPatientID() : int
+setPatientID(PatientID : int) : void
+getDoctorID() : int
+setDoctorID(DoctorID : int) : void
*
Taken by
1
<<Interface>>
Notification
-Send Notification() : Void
*
Medical Record
-ID : int
-Patient ID : int
-Name : String
-Description : String
-Time : time stamp
-Specialization : String
+getPatient ID() : int
+setPatient ID(Patient ID : int) : void
+getName() : String
+setName(Name : String) : void
+getDescription() : String
+setDescription(Description : String) : void
+getTime() : time stamp
+setTime(Time : time stamp) : void
Visit
-Time slot : T
-DoctorID : int
+getTime slot() : T
+setTime slot(Time slot : T) : void
+getDoctorID() : int
+setDoctorID(DoctorID : int) : void
+getVisitID() : int
+setVisitID(VisitID : int) : void
Investigation
-upnormal results : String
-Image URL : String
+getUpnormal results() : String
+setUpnormal results(upnormal results : String) : void
+getImage URL() : String
+setImage URL(Image URL : String) : void
Report
-DoctorID : int
+getDoctorID() : int
+setDoctorID(DoctorID : int) : void
+getSpecialization() : String
+setSpecialization(Specialization : String) : void

Class ID	Class Name	Subsystem ID	Description & Responsibility
U001	User	User Interface	This class contains the Name, ID, Email, Mobile and password and the user can send message through it (the message gets arguments: From, To and content)
Man001	Manager	User	This class is responsible for displaying the Manager Interface and show the action that can be performed by the manager (adding new: manager, doctor, patient, report, Investigation, visit also it is responsible for cancelling time slot, deleting doctor, patient and can perform search time slot action_

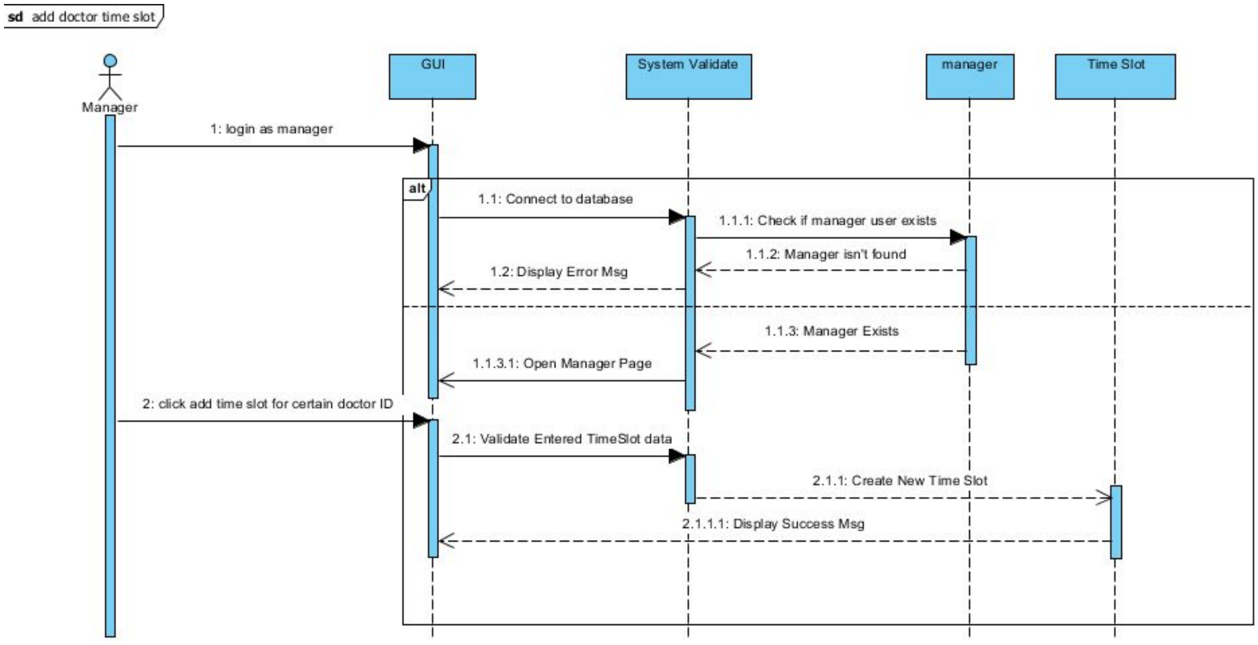
D001	Doctor	User	This class is responsible for displaying the Doctor Interface and show the action that can be performed by the Doctor (adding medication, report, investigation, view patient medical history, search patients).
P001	Patient	User	This class is responsible for displaying the Patient Interface and show the action that can be performed by the Patient (update info, view time slots, set appointment and review doctor).
MR001	Medical Record	Component Management	This class is responsible for displaying any medical record (Report, Investigation and visit) of the patient
V001	Visit	Component Management	This class is responsible for setting a patient for the patient with specific Doctor
Inv001	Investigation	Component Management	This class is responsible for storing the investigation of the patient
R001	Report	Component Management	This class is responsible for storing the medical record of the patient
Med001	Medication	Component Management	This class is responsible for setting the Medication to the patient with instructions and time.
N001	Notification	Component Management	This class is responsible for sending notification to the User with a specific reminder
Im001	Image	Component Management	This class is responsible for displaying and compressing the image of the user
V001	Validate	User	This class is responsible for validation of the user and to decide the user access to the database

Software Design Document

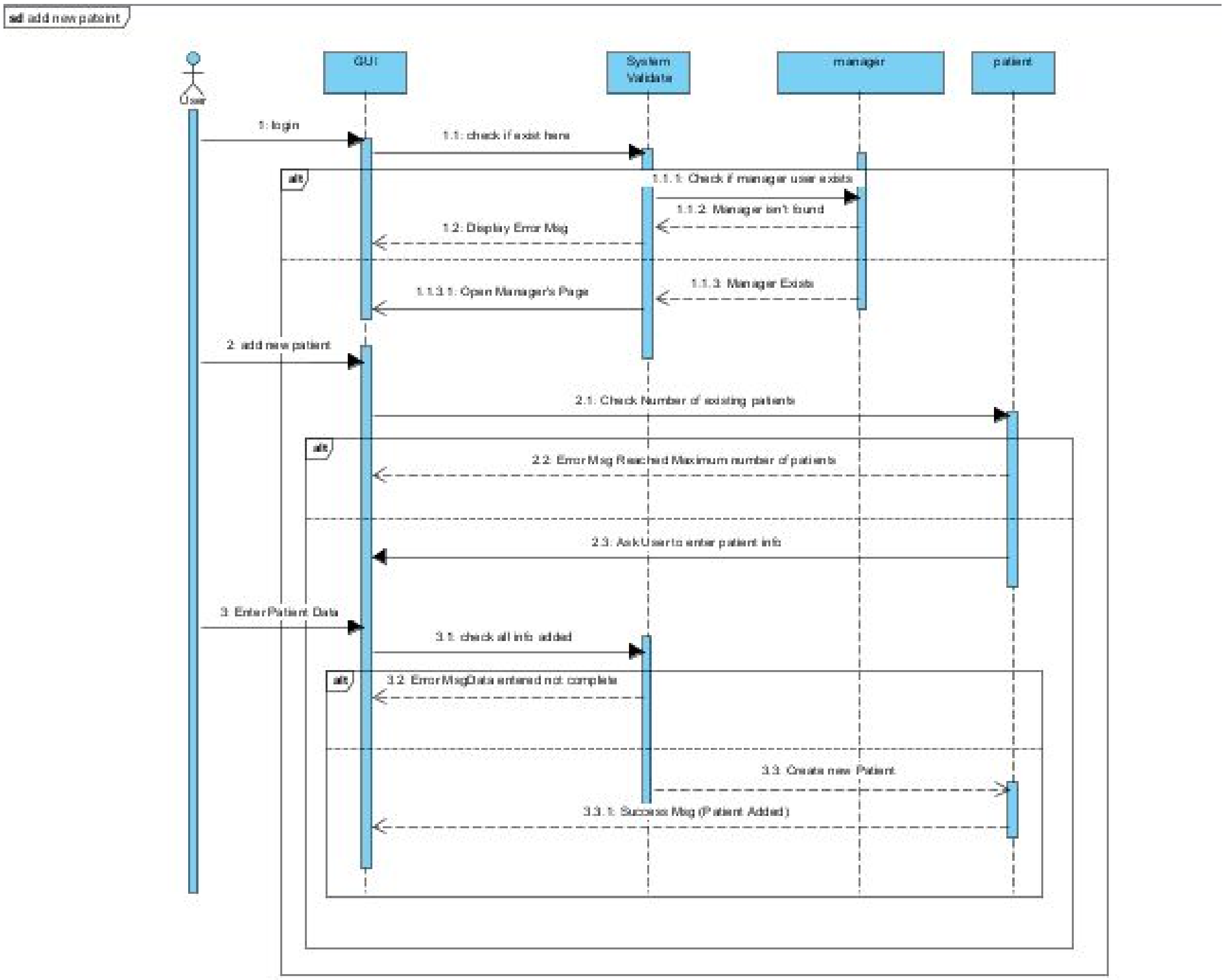
TS001	Time Slot	Component Management	This class is responsible for scheduling a time slot between patient and doctor

III. Sequence diagrams

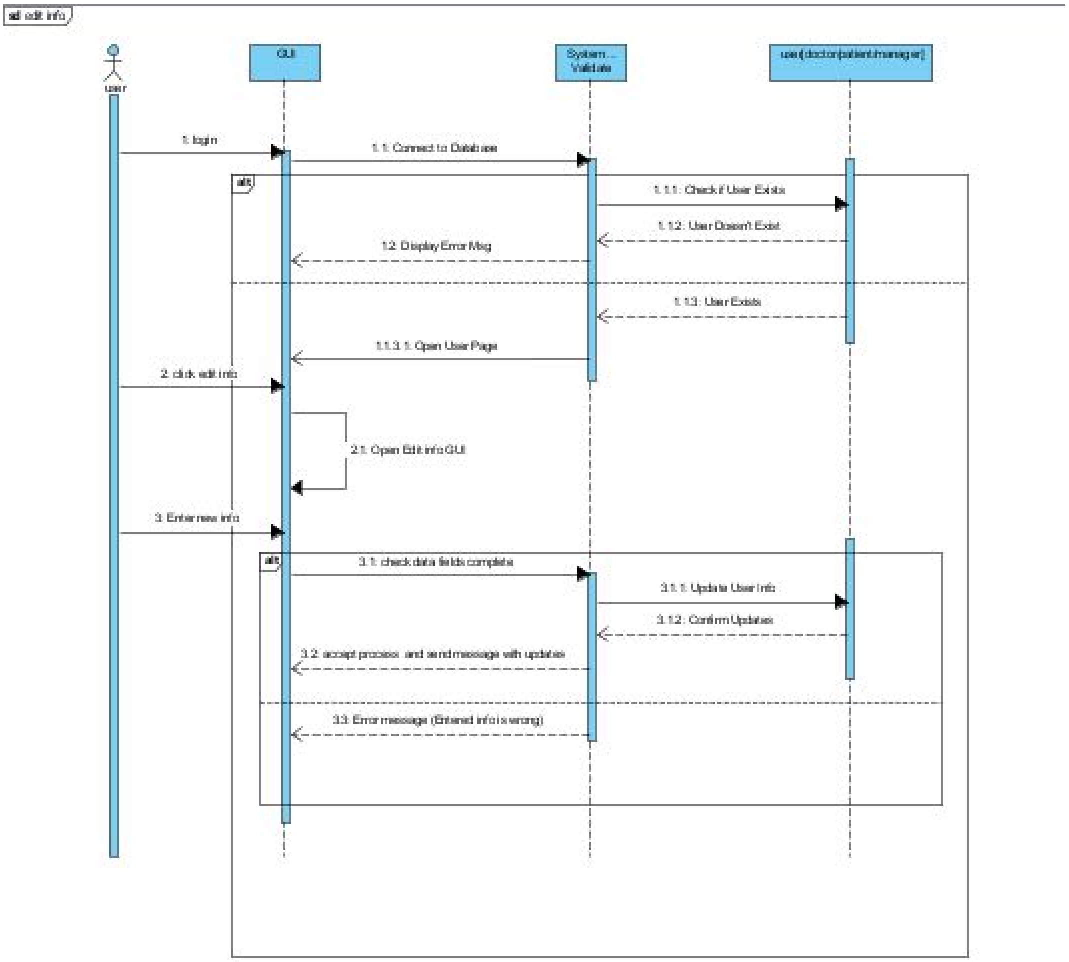
- **Sequence diagram (SQ001):**


sd add doctor time slot
Manager
GUI
System Validate
manager
Time Slot
1: login as manager
alt
1.1: Connect to database
1.1.1: Check if manager user exists
1.1.2: Manager isn't found
1.2: Display Error Msg
1.1.3: Manager Exists
1.1.3.1: Open Manager Page
2: click add time slot for certain doctor ID
2.1: Validate Entered TimeSlot data
2.1.1: Create New Time Slot
2.1.1.1: Display Success Msg

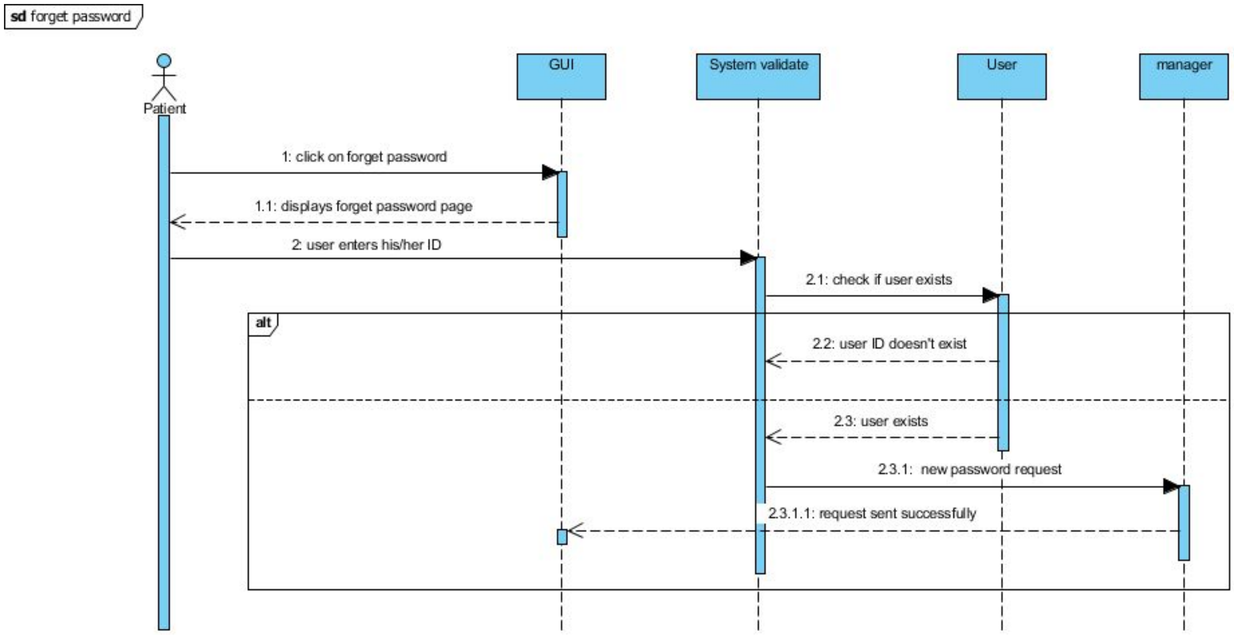
- **Sequence diagram (SQ002):**


sd add new pateint
User
GUI
System Validate
manager
patient
1: login
1.1: check if exist here
alt
1.1.1: Check if manager user exists
1.1.2: Manager isn't found
1.2: Display Error Msg
1.1.3: Manager Exists
1.1.3.1: Open Manager's Page
2: add new patient
2.1: Check Number of existing patients
alt
2.2: Error Msg Reached Maximum number of patients
2.3: Ask User to enter patient info
3: Enter Patient Data
3.1: check all info added
alt
3.2: Error MsgData entered not complete
3.3: Create new Patient
3.3.1: Success Msg (Patient Added)

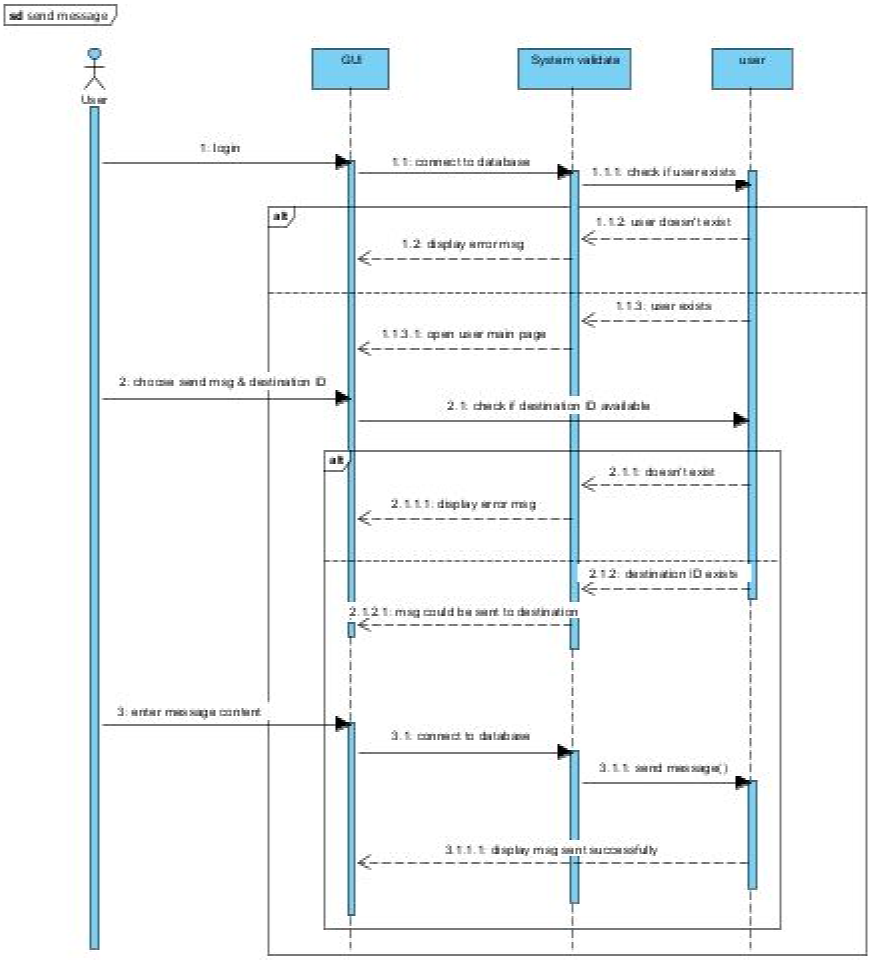
- **Sequence diagram (SQ003):**



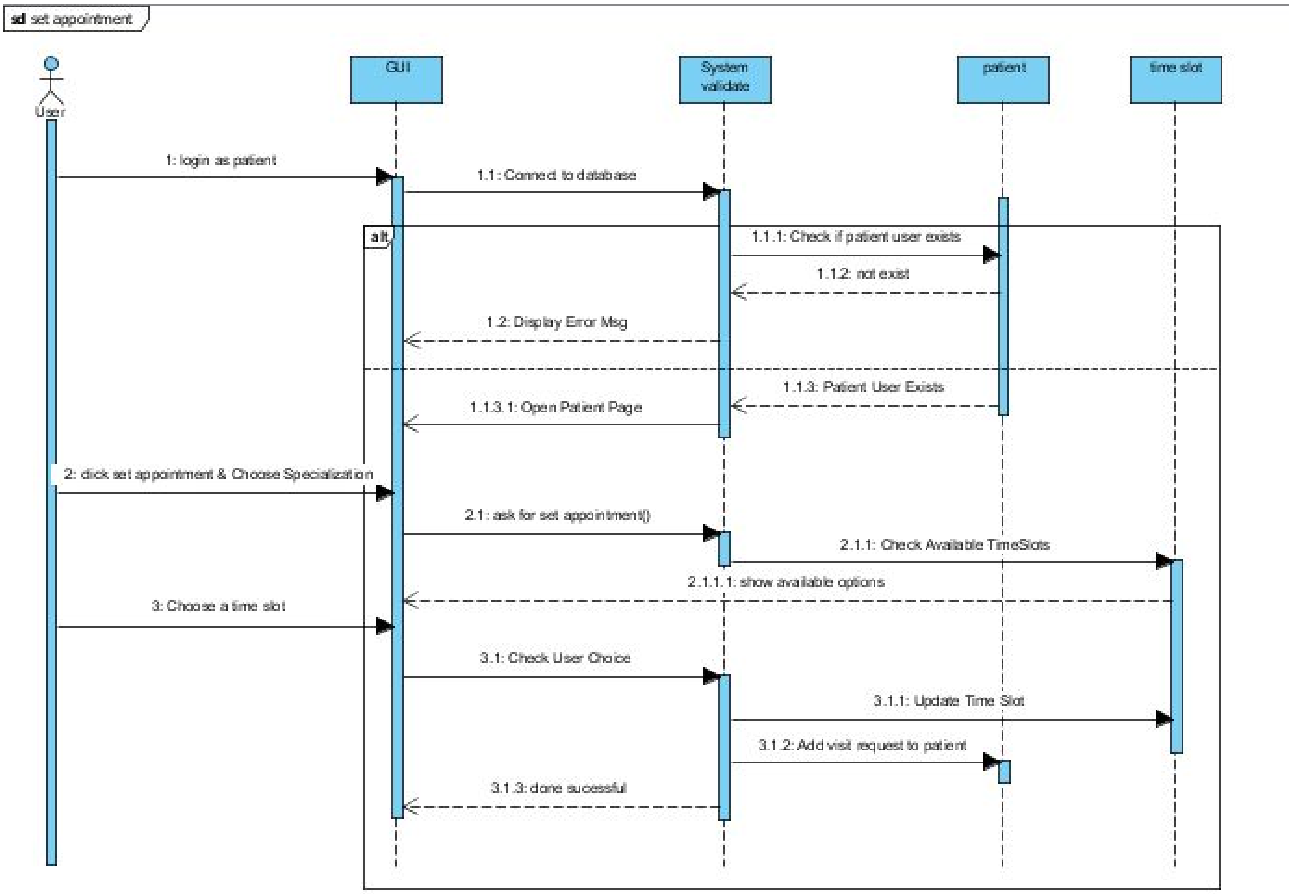
- **Sequence diagram (SQ004):**


sd forget password
Patient
GUI
System validate
User
manager
1: click on forget password
1.1: displays forget password page
2: user enters his/her ID
2.1: check if user exists
alt
2.2: user ID doesn't exist
2.3: user exists
2.3.1: new password request
2.3.1.1: request sent successfully

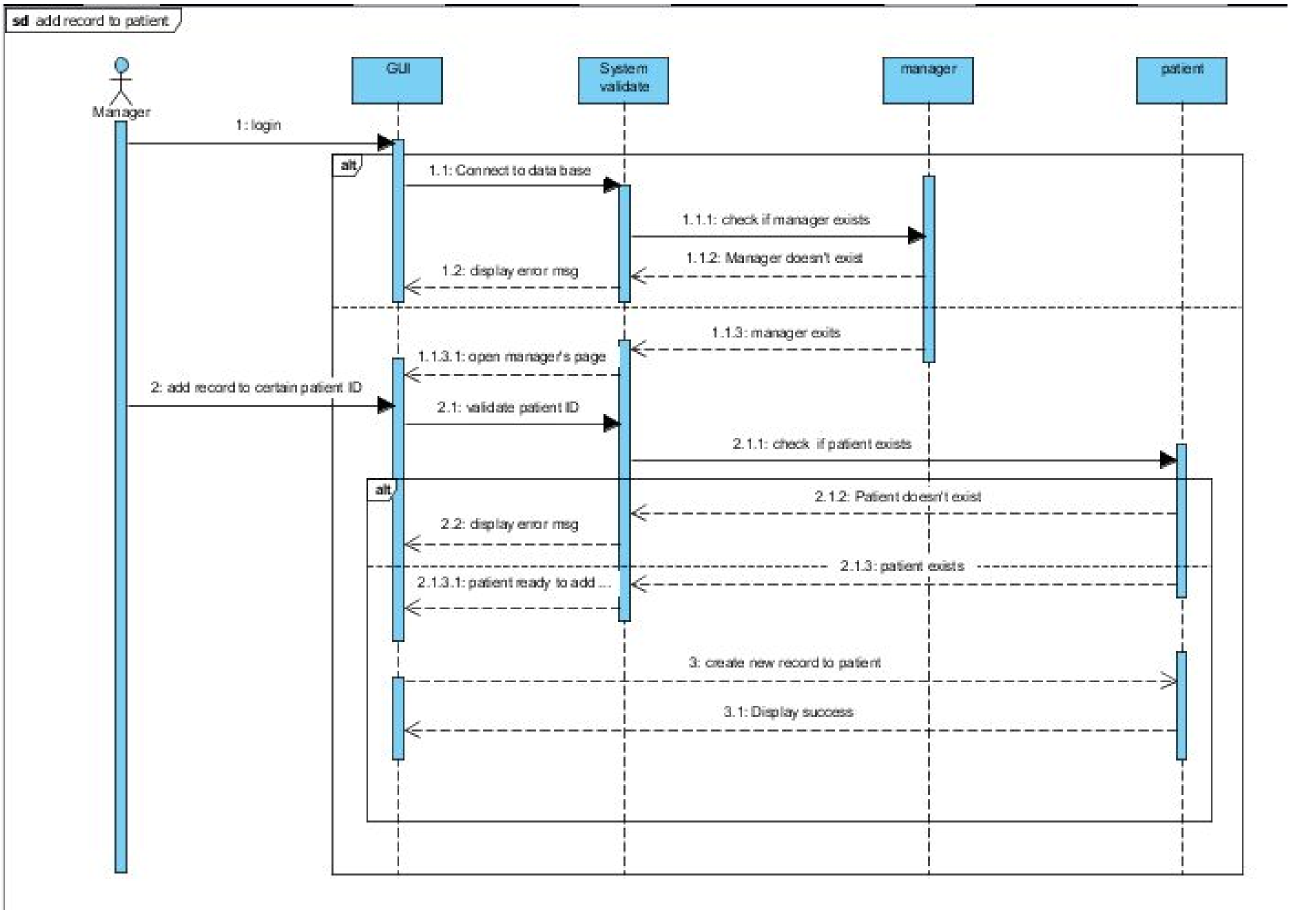
- **Sequence diagram (SQ005):**


sd send message
User
GUI
System validate
user
1: login
1.1: connect to database
1.1.1: check if user exists
alt
1.1.2: user doesn't exist
1.2: display error msg
1.1.3: user exists
1.1.3.1: open user main page
2: choose send msg & destination ID
2.1: check if destination ID available
alt
2.1.1: doesn't exist
2.1.1.1: display error msg
2.1.2: destination ID exists
2.1.2.1: msg could be sent to destination
3: enter message content
3.1: connect to database
3.1.1: send message()
3.1.1.1: display msg sent successfully

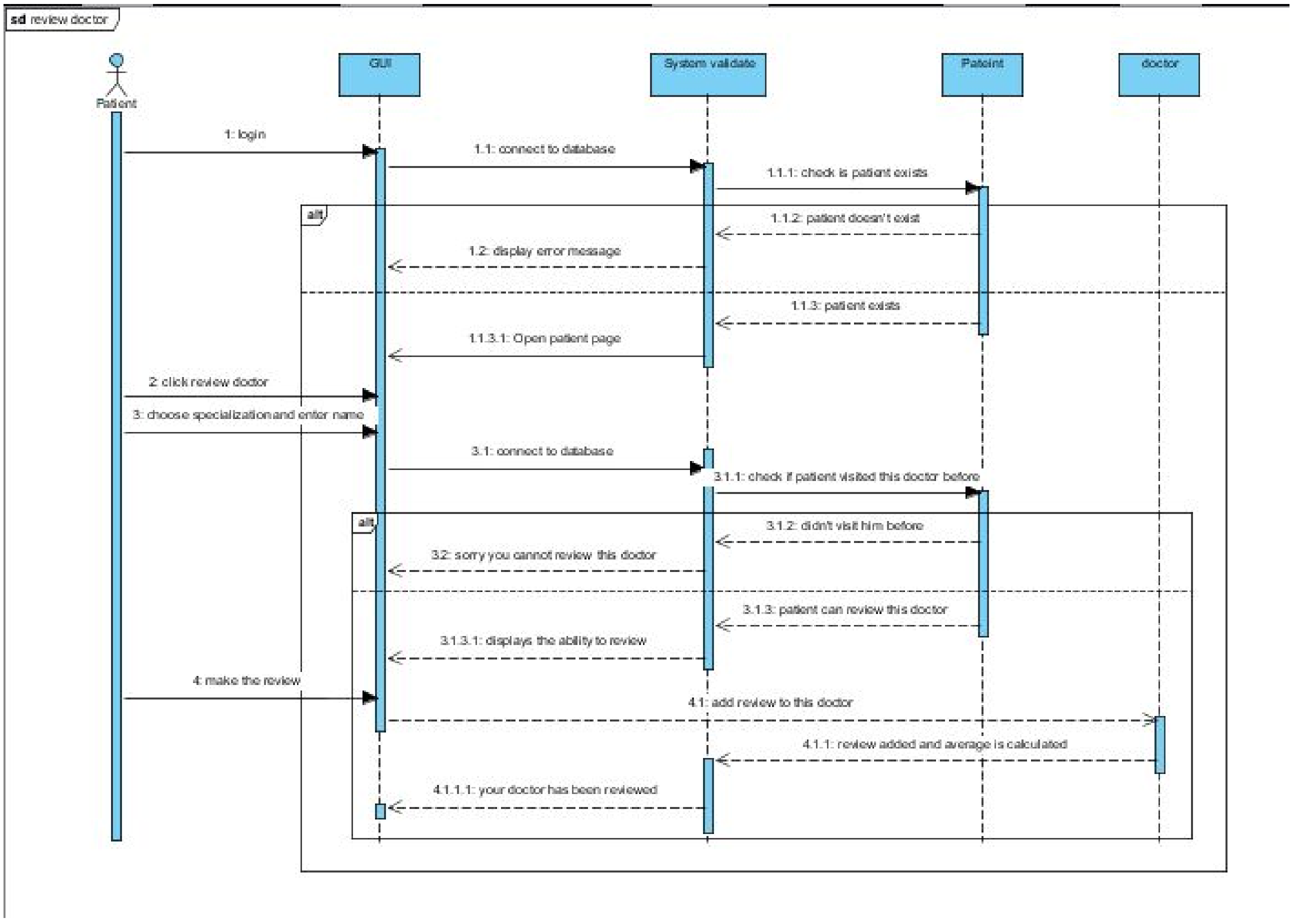
- **Sequence diagram (SQ006):**


sd set appointment
User
GUI
System validate
patient
time slot
1: login as patient
1.1: Connect to database
alt
1.1.1: Check if patient user exists
1.1.2: not exist
1.2: Display Error Msg
1.1.3: Patient User Exists
1.1.3.1: Open Patient Page
2: click set appointment & Choose Specialization
2.1: ask for set appointment()
2.1.1: Check Available TimeSlots
2.1.1.1: show available options
3: Choose a time slot
3.1: Check User Choice
3.1.1: Update Time Slot
3.1.2: Add visit request to patient
3.1.3: done sucessful

- **Sequence diagram (SQ007):**


sd add record to patient
Manager
GUI
System validate
manager
patient
1: login
alt
1.1: Connect to data base
1.1.1: check if manager exists
1.1.2: Manager doesn't exist
1.2: display error msg
1.1.3: manager exits
1.1.3.1: open manager's page
2: add record to certain patient ID
2.1: validate patient ID
2.1.1: check if patient exists
alt
2.1.2: Patient doesn't exist
2.2: display error msg
2.1.3: patient exists
2.1.3.1: patient ready to add ...
3: create new record to patient
3.1: Display success

- **Sequence diagram (SQ008):**


sd review doctor
Patient
GUI
System validate
Pateint
doctor
1: login
1.1: connect to database
1.1.1: check is patient exists
alt
1.1.2: patient doesn't exist
1.2: display error message
1.1.3: patient exists
1.1.3.1: Open patient page
2: click review doctor
3: choose specialization and enter name
3.1: connect to database
3.1.1: check if patient visited this doctor before
alt
3.1.2: didn't visit him before
3.2: sorry you cannot review this doctor
3.1.3: patient can review this doctor
3.1.3.1: displays the ability to review
4: make the review
4.1: add review to this doctor
4.1.1: review added and average is calculated
4.1.1.1: your doctor has been reviewed

CIE 203: Phase 1 – SIGSEGV
Project: <StethoMan>

Software Design Document

Class - Sequence Usage Table

Class Name	Sequence Diagrams	Overall used methods
Validate	SQ001, SQ002, SQ003, SQ004 SQ005, SQ006, SQ007, SQ008	ValidateEnteredData()

Manager	SQ001, SQ002, SQ004, SQ007	AddTimeSlot()
		AddInvestigation()
		AddReport()
		AddVisit()
		AddPatient()

Time Slot	SQ001, SQ006	UpdatPatient()
		getDoctorID()

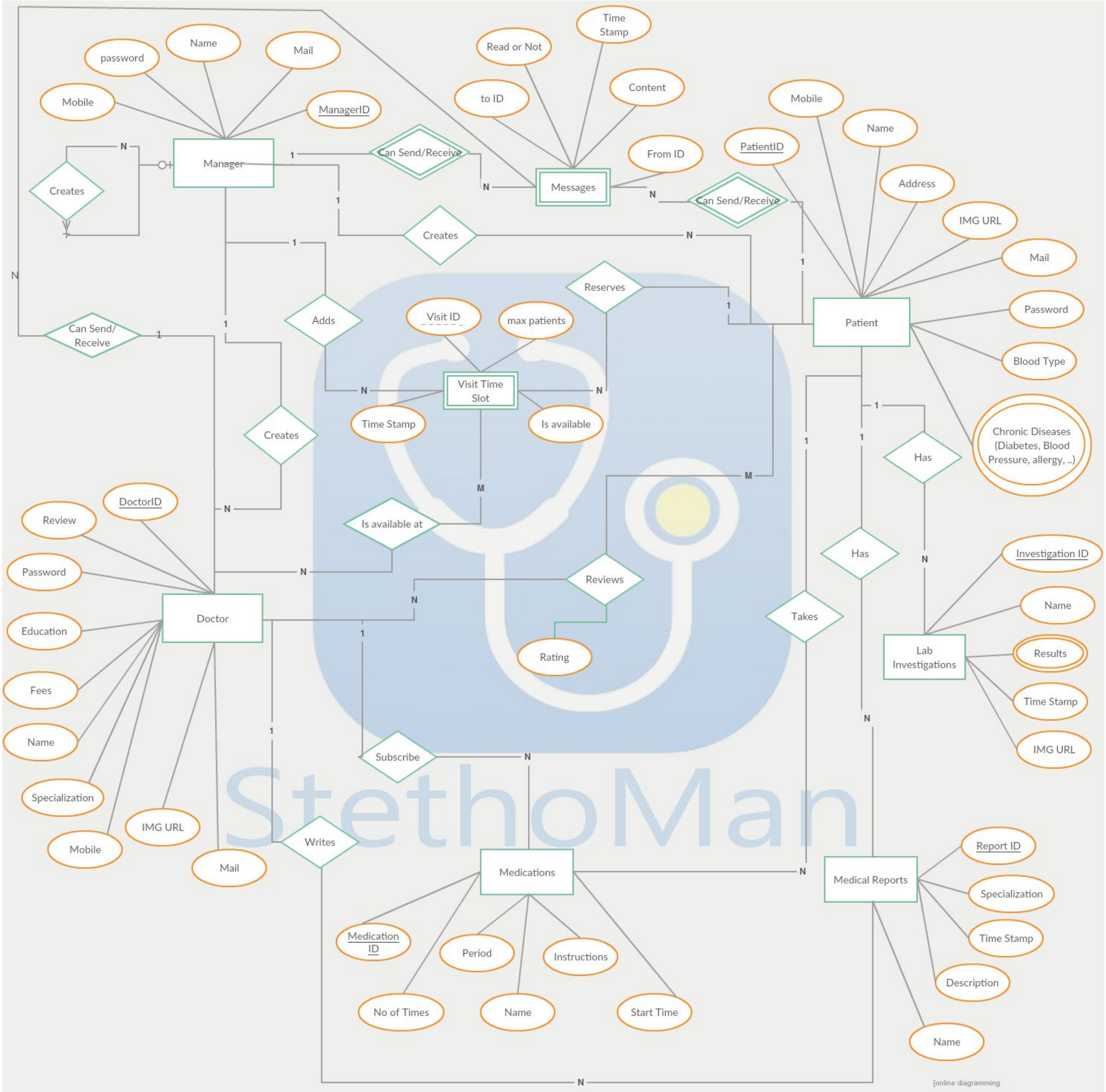
Patient	SQ002, SQ006, SQ007, SQ008	UpdateInfo()
		SetAppointment()
		ReviewDoctor()

User	SQ003, SQ005	setName()
		setMail()
		setMobile()
		setPassword()
		sendMessage()

CIE 203: Phase 1 – SIGSEGV
Project: <StethoMan>

Software Design Document

IV. Physical Entity-Relationship Diagram (Click Here to view at complete resolution) -- Software Used: Creatly.com

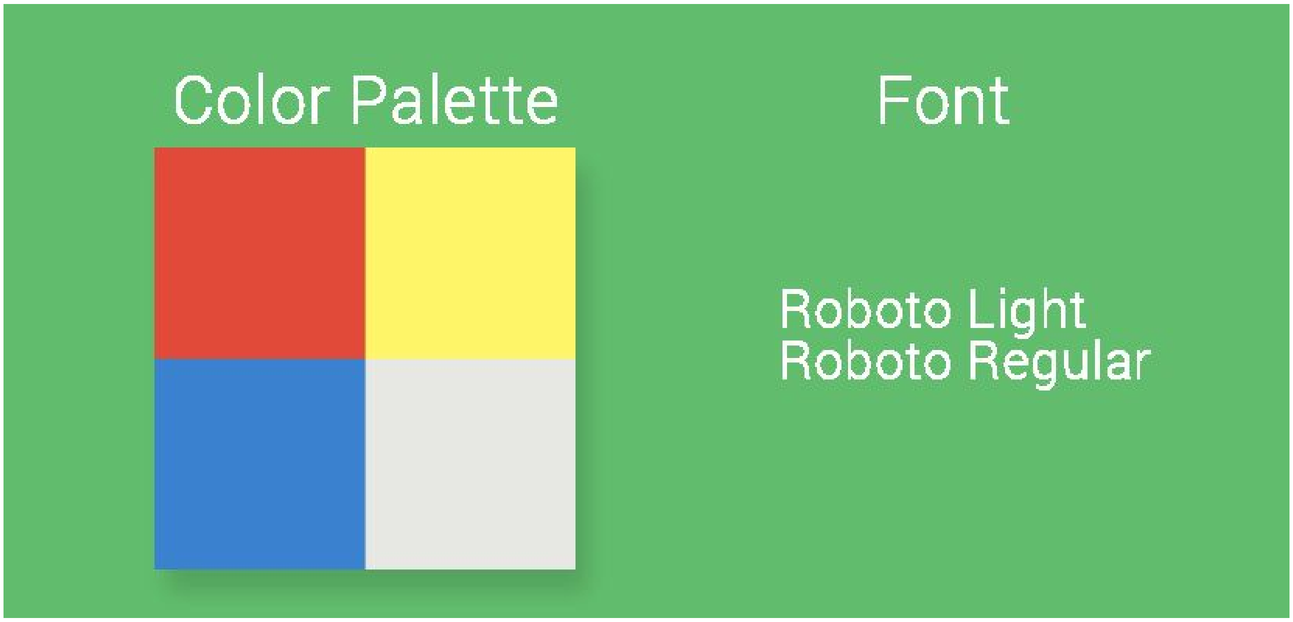


V. User Interface Design

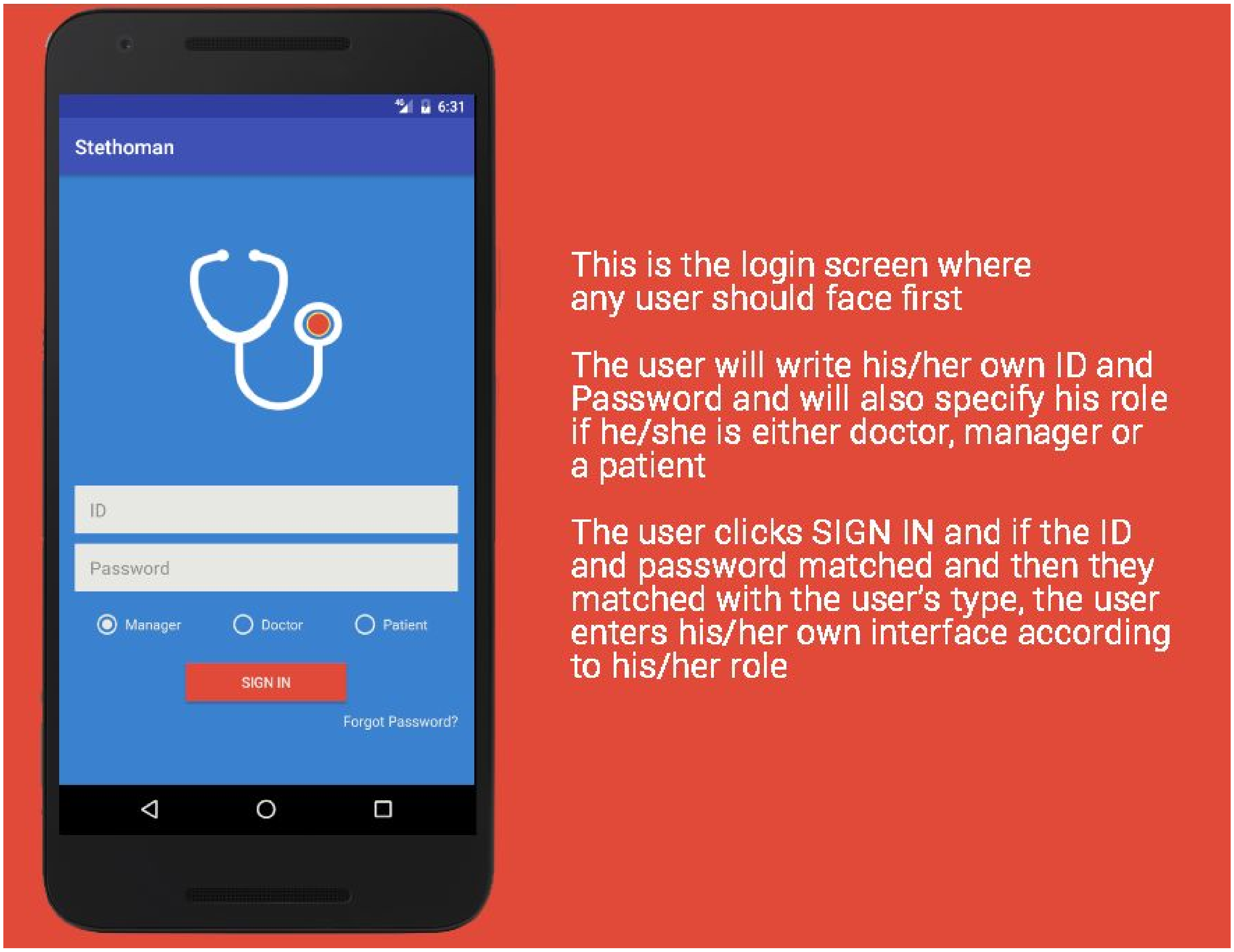
The user interface was designed first using Adobe Photoshop CC and then then the UI was implemented on Android Studio v2.2.2 using XML and Java.

A color palette was made and the font “Roboto” was chosen to create the brand identity of our application.

Our UI supports from Android v4.0.0 “Lollipop” and it also supports many dimensions for different mobile phones using android operating system such as: 480x800, 720x1280, 2560x1600, 768x1280, 1200x1920, 1080x1920, 2048x1536 which is nearly 98% of all tablets and mobile phones using Google playstore.



- **Screen 1 – Login Screen**

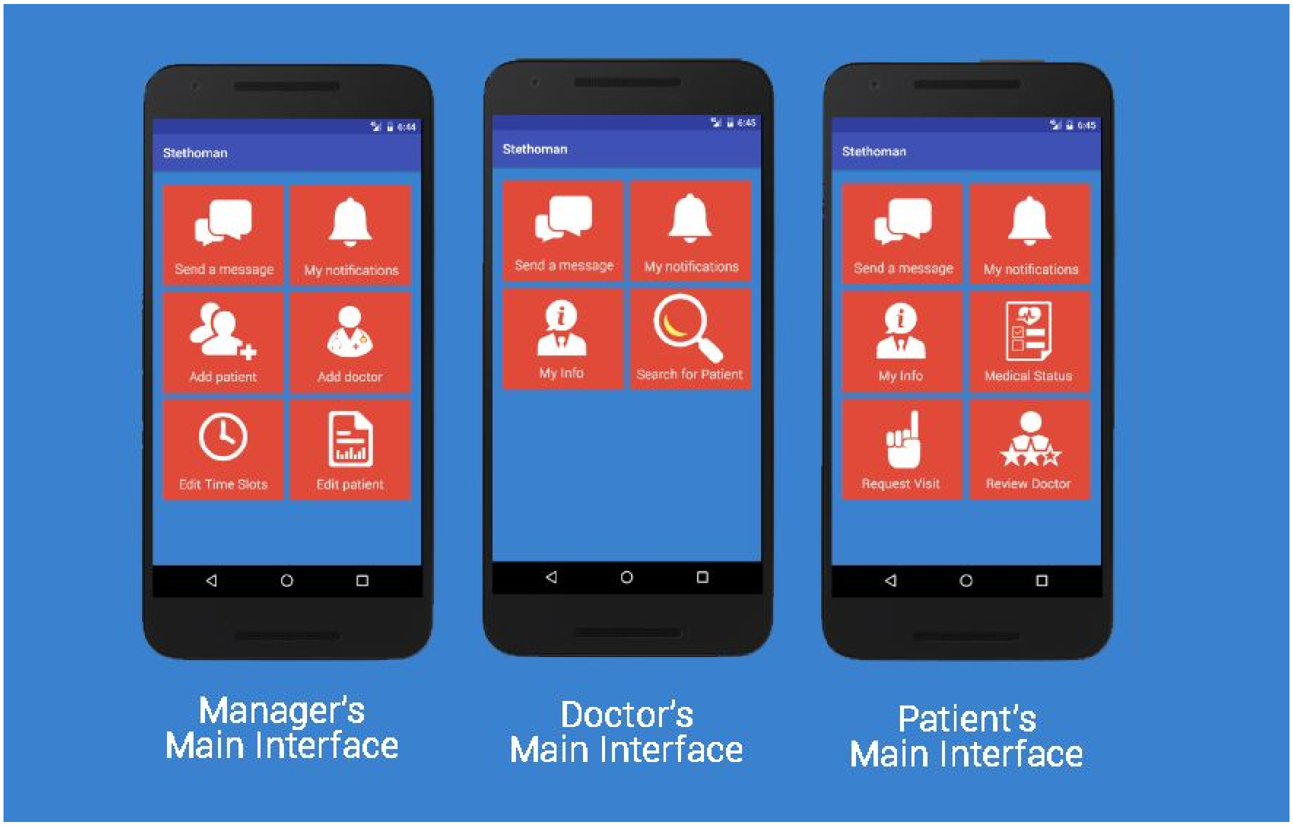


This is the login screen where any user should face first

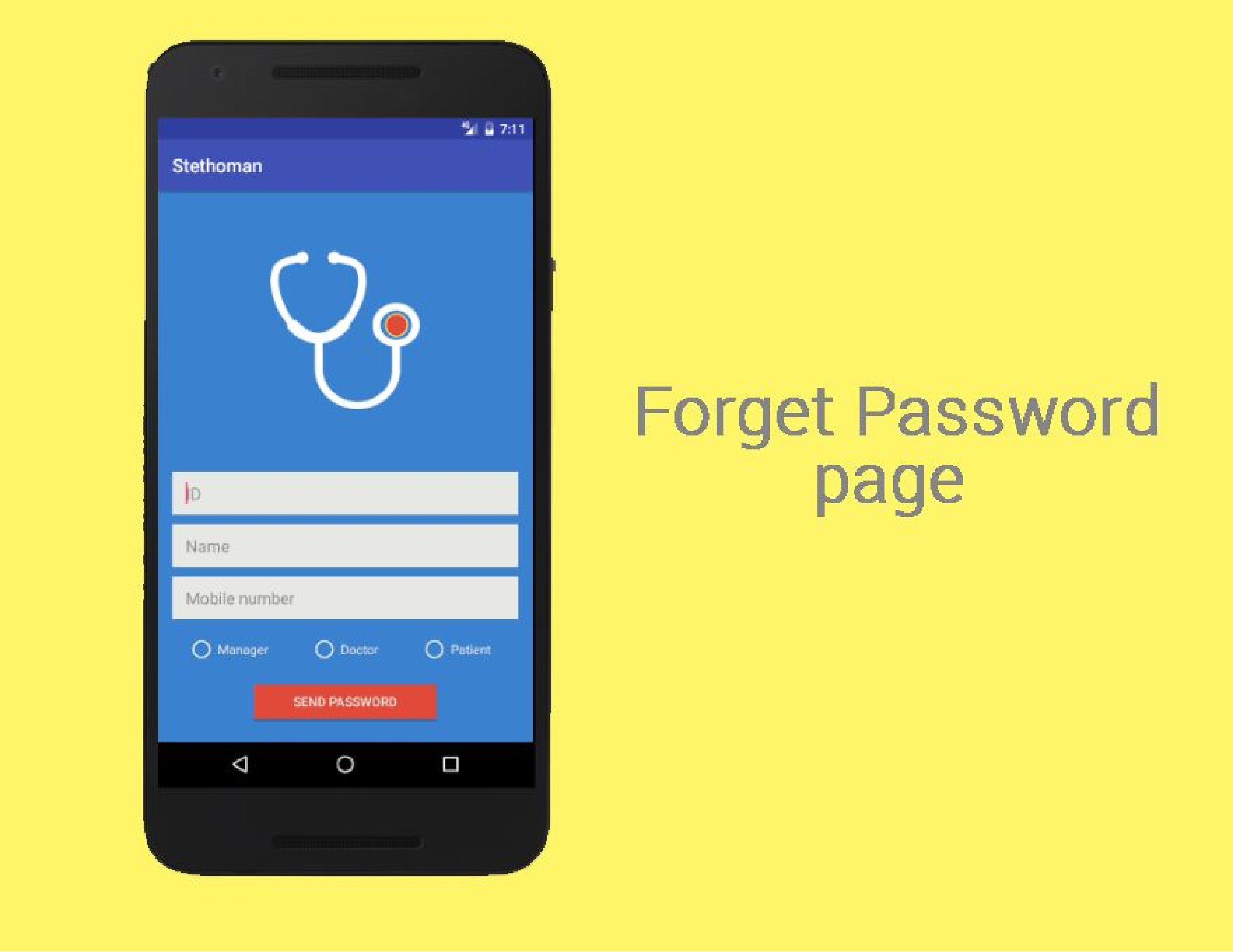
The user will write his/her own ID and Password and will also specify his role if he/she is either doctor, manager or a patient

The user clicks SIGN IN and if the ID and password matched and then they matched with the user's type, the user enters his/her own interface according to his/her role

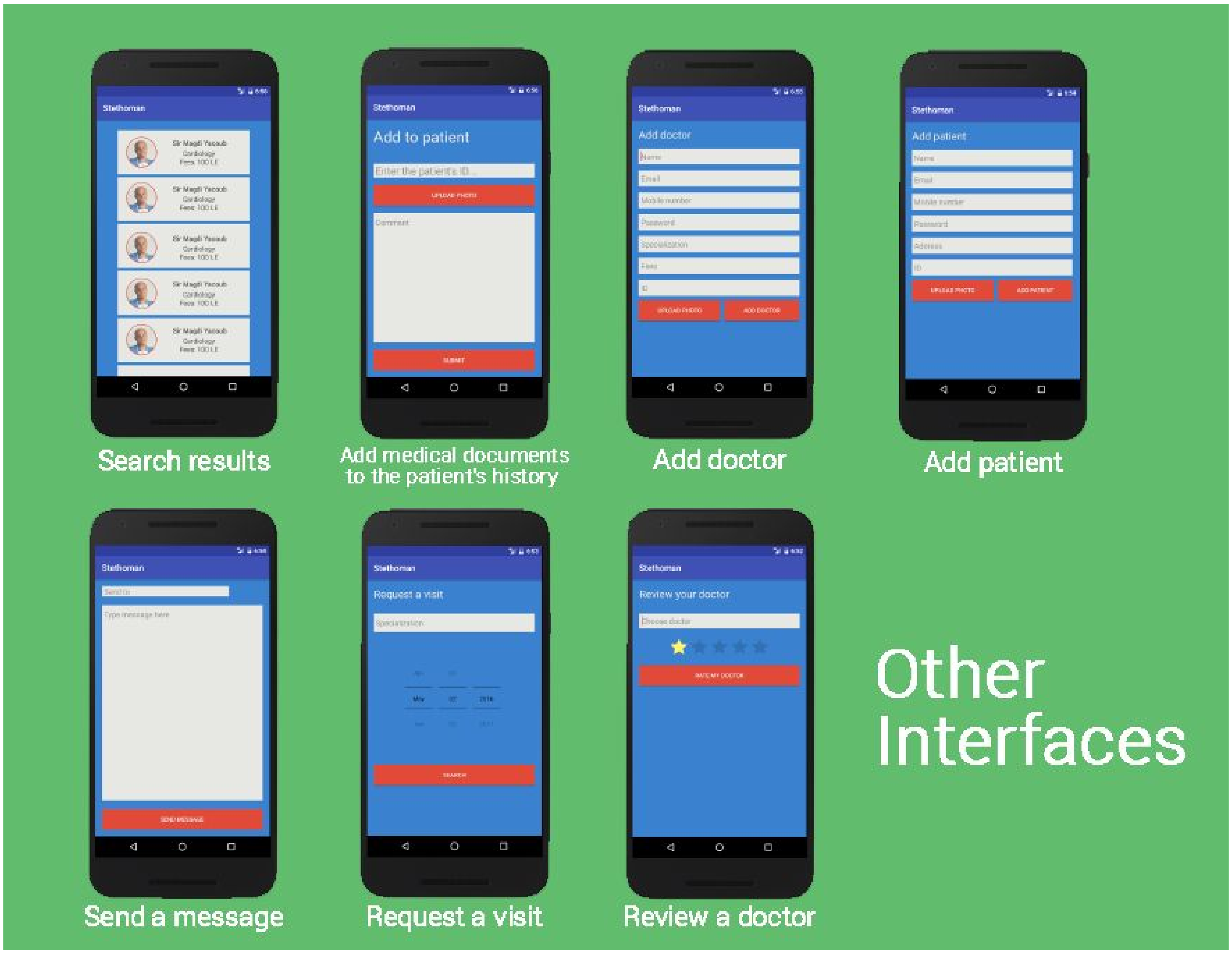
- **Screen 2 – Main Interface**



- **Forget password screen**

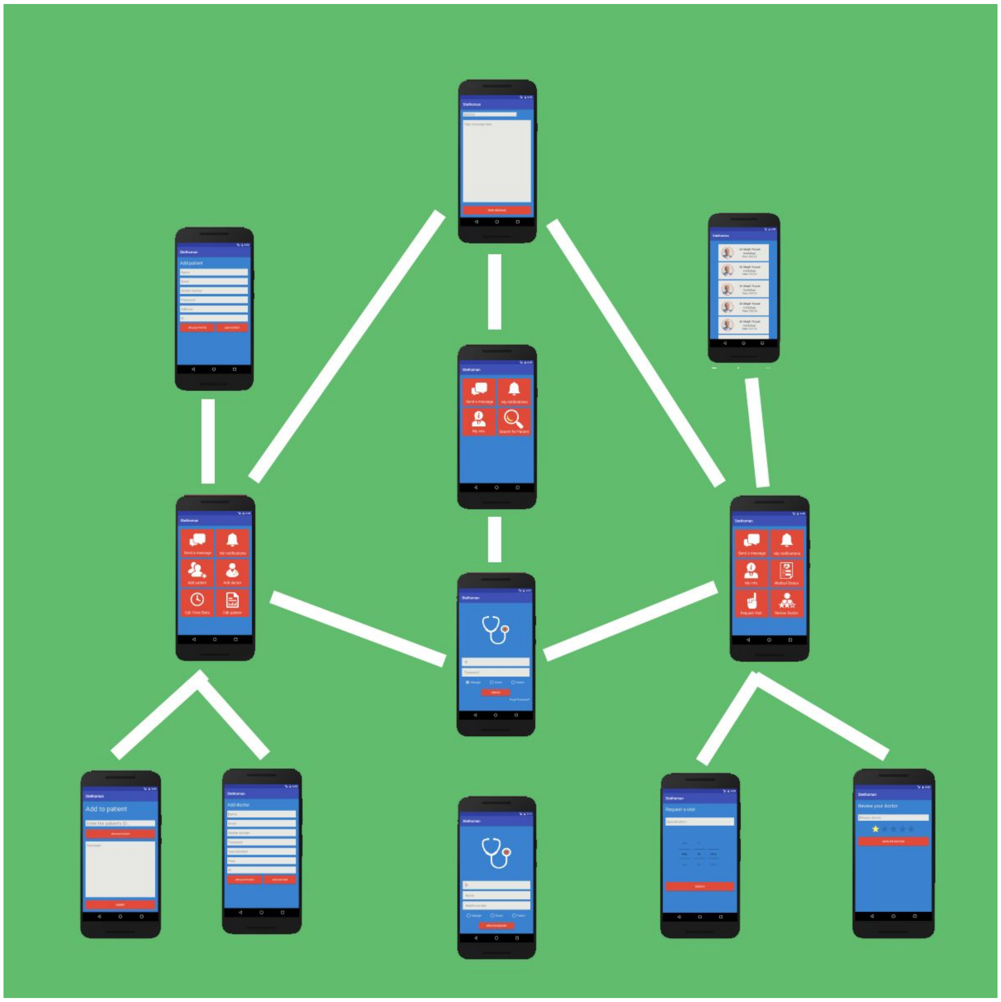


- **Other screens**



- **Navigation tree:**

1. The login interface has a clickable button “Sign in” and a clickable text “Forgot Password?”. “Sign in” leads to one of the three main pages of the three types of users and “Forgot Password” leads to the forgot password activity.
2. The manager’s main page has six buttons each of them lead to a different activity. Send message button leads to send message activity, view notifications button leads to notifications page and so on as was introduced earlier in the images above.
3. The same happens with the doctor’s and the patient’s main pages. The three types of users share both view notifications send message activities.



VI. Algorithms and Data Structures

***Log IN:

```
-User Enter His User ID, Password and User Type (Patient, Doctor, Manager)
-Connect to database
-Connect to Table (Patient, Doctor, Manager) according to user type
-Select row of User with same ID and Password
-If no results found
        -Display Error to user to check his information or select forget password
Else
        -Log in to user profile showing his profile interface
```

***Forget Password:

-User Enters what he remembers from (UserID, Password, Mail, Mobile Phone)

-Connect to database (Messages table)

-Create a new row in table with target "Manager" and content with user entered info

-Display a message that his request has been sent successfully and a manager will contact him

***Add new User:

-Check a Manager is logged in

-User Choose User Type and fill in his information

```
-If there is missing info
    -Display message to user to fill in missing fields
-Else
    -Connect to Database and user type table
    -Search for a user with same entered info
    -If another user with same info exists
        -display a message user already exists and display user info
    -Else
        -Add new row in table with entered data
```

***Delete (Patient / Doctor):

```
-Check a Manager is logged in
-User Choose User Type(Patient/Doctor) and fill in his information
-If there is missing info
    -Display message to user to fill in missing fields
-Else
    -Connect to Database and user type(Doctor/Patient) table
    -Search for a user with same entered info
    -If user with same info exists
        -Delete user and all its records
    -Else
        -Display error message that there is no user with entered info
```

***Add Medical Record by Manager:

```
-Check that Logged in user is a manager
-User Enters Patient ID, type of medical record, data of medical record
-If there is missing info
    -Display message to user to check missed fields
-Else
    -Connect to database and patients table
    -If Patient ID doesn't exist in table
        -Display message to user to check user id
    -Else
        -Display Message to User with Patient info and asks for confirmation
        -Open Medical Records table and add the record info
        -Display a success message to user
```

***Send a Message

```
-Check That a User is logged in
-User Enters Target and message content
```

```
-Connect to Database
-Search for Target ID
-If no results Exist
        -Display error message that target is wrong
-Else
        -Connect to Messages table in database
        -Add new row with message details
        -Send a notification to target user and display a success message to sending user
```

***Read a message

```
-Connect to Database
-Search for messages with target id = logged in user id
-Add messages to a stack
-While stack is not empty
        -Pop message to user
```

***Stack data structure is used here because messages should be displayed to user from newest to oldest

***Add doctor Time Slot:

```
-Check that logged in user is a manager
```

```
-User Enters Doctor ID, time stamp of time slot, maximum no. of patients to register at this slot
-Connect to database
-Check for doctor in doctors table
-If no results exist
        -Display error message that doctor can't be found
-Else
        -Connect to time slots table
        -add new row with doctor time slot
        -Display success message
```

***Cancel Slot:

```
-Check that logged in user is a manager
-Connect to database and time slots table and store all time slots newer than current timestamp in stack
-While stack is not empty
        -Pop a time slot details to user
-User choose a slot to cancel
-Connect to visits table
- Store all patients' id in addition to doctor id where timeslot id = to the cancelled id
- Send message to all users and to doctor that time slot is cancelled
```

-Delete selected rows from visits table

***Confirm Visit Record by manager:

-Check that logged in user is a manager

-Connect to Database, visits table

-Search for all visits rows that aren't confirmed and time stamp is newer than current time

-Display all visits

-Manager selects a visit to confirm

-Update data in visit row to be confirmed and Send message to patient to confirm visit

***View Patient Medical History by doctor:

```
-Check that logged in user is doctor
-User Enters patient id
-Connect to database, visits table
-Search for visits record with patient id = entered id and doctor id = logged in id
-If no results found
    -Display an error, Doctor doesn't have permission to view patient history
-Else
    -Connect to medical records tables (Lab investigations, Medical Reports)
    -Search for rows with patient id and stores them in stack
    -While Stack isn’t empty
        -Pop out medical record to be displayed to doctor
```

***Stack data structure is used here because Medical records should be displayed to user from newest to oldest

***Add Medical Record by Doctor:

```
-Check that Logged in user is a doctor
-User Enters Patient ID, type of medical record, data of medical record
-If there is missing info
    -Display message to user to check missed fields
-Else
    -Connect to database and visits table
    -If patient ID doesn't exist
        - Display a message that patient doesn’t exist.
    -Else If Patient ID = entered ID and Doctor ID = logged in user id doesn’t exist in table
        -Display message doctor doesn’t have permission to add record
    -Else
        -Display Message to User with Patient info and asks for confirmation
        -Open Medical Records table and add the record info
        -Display a success message to user
```

***Add Medication by Doctor:

```
-Check that Logged in user is a doctor
-User Enters Patient ID, medication info, start time, end time and period of taking the medication
-If there is missing info
    -Display message to user to check missed fields
-Else
    -Connect to database and visits table
    -If patient ID doesn't exist
        - Display a message that patient doesn’t exist.
    -If Patient ID = entered ID and Doctor ID = logged in user id doesn’t exist in table
        -Display message doctor doesn’t have permission to add record
    -Else
        -Display Message to User with medication info and asks for confirmation
        -Open Medications table and add the record info
        -Display a success message to user
```

***Alert Patient with Medication:

```
-If a logged in user is a patient
    -Connect to database, medications table
    -Search for rows with patient id = logged in id, and time greater than current time, not checked yet and stores them
    -While there are still new medications
        -flag the medication
        -Add new alarms to user at periods of taking the medication from the start date to the end date
```

***Review a Doctor:

```
-Checked that logged in user is a patient
-Connect to database, visits table
-Search for a visit with patient id = logged in id and doctor id = chosen doctor id
-If no results exist
    -display error message that patient doesn't have permission to review this doctor
-Else
    -Connect to reviews table and search for rows with patient id = logged in id and doctor id = chosen doctor id
    -If a result exists
        -Display an error that patient has reviewed doctor before
    -Else
        -User Enters his review
        -Add new row in reviews table with patient review
```

```
-Open Doctors Table and Search for Doctor ID
                -Update doctor rating
                -Display success message to user
```

<u>*Set an appointment by patient:</u>**

```
-Check Logged in user is a patient
-User selects specialization
-Connect to database and time slots table
-Search for time slots with same specialization, not closed and has timestamp (time of visit)
newer than current time
-If no results exist
        -Display a message to user that no available doctor at this specialization
Else
        -Display all time slots available
        -User selects a time slot to reserve
        -Connect to visits table
        -Search for patient ID and time slot ID
        -If a result found
                -Display error message that patient has registered in this slot before
        -Else
                -Add new row in visits table with patient id and visit info
                -Update time slot info to increase one reserved patient in that time slot
                -If number of reserved patients = maximum number of allowed patients to reserve
                    -Update time slot and check it as closed
```

<u>*Compress Image:</u>**

```
- Check that User has uploaded an image with dimensions greater than 80 KB
-If Image is Greater than 80KB Apply averaging filter
    -Calculate image current dimensions to the required dimensions to make it as desired size
    -OW = old width, NW = new width, OL = old length, NL = new length
    -CW = OW/NW "step Width" and CL = OL/NL "step Length"
    -While r = 0 to OL
        -for i=0 to OW and i = i + CW
            -for j=i to i + CW and j = j+1
                - sum all pixels values
            -end
            - Get average of Pixel Values and stores new pixel at row "r" and column
        -end
    - Perform same operations on rows instead of columns
    -Make the new compressed image
```

- If Image is doctor photo

-Rename image with doctor ID and stores it in doctors images folder

-Else If Image is patient photo

-Rename image with patient ID and stores it in patients images folder

-Else If image is related to patient lab investigation

-Rename Image with patientID_LabinvestigationID and stores in a folder named with patient ID

Ownership Report

Item	Owners
Entity Relation Diagram & Algorithms	*Mohamed Maher*
System Decomposition & Class Diagram	*Omar Merghany*
UI	*Youhana Mourad*
Sequence Diagram	*Randa Selim Negm*

Policy Regarding Plagiarism:

1. تشجع الكلية على مناقشة الأفكار و تبادل المعلومات و مناقشات الطلاب حيث يعتبر هذا جوهريا لعملية تعليمية سليمة
2. ساعد زملاءك على قدر ما تستطيع و حل لهم مشاكلهم فى الكود و لكن تبادل الحلول غير مقبول و يعتبر غشا.
3. أى حل يتشابه مع أى حل آخر بدرجة تقطع بأنهما منقولان من نفس المصدر سيعتبر أن صاحبيهما قد قاما بالغش.
4. قد توجد على النت برامج مشابهة لما نكتبه هنا أى نسخ من على النت يعتبر غشا يحاسب عليه صاحبه.
5. إذا لم تكن متأكدا أن فعلا ما يعد غشا فلتسأل المعيد أو أستاذ المادة.
6. فى حالة ثبوت الغش سيأخذ الطالب سالب درجة المسألة ، و فى حالة تكرار الغش سيرسب الطالب فى المقرر.

References

- http://www.mhhe.com/engcs/compsci/pressman/graphics/Pressman5sepa/common/cs1/design.pdf

Authors

-