REPORT

TOPIC 19: SMART HEALTH PREDICTION SYSTEM

Member List
Nguyen Minh Trang – ITDSIU19020
Phan Vo Phuong Tung – ITDSIU19025
Bui Thi Xuan Lan – ITDSIU19007
Phan Ho Hoang Phuoc – ITDSIU18044

Github Link: https://github.com/nmtrang/smart-health-prediction-database

(THIS PAGE IS INTENTIONALLY LEFT BLANK)

Table of Contents

I. INTRODUCTION

Nowadays, it is quite challenging to predict one’s performance based on their health. There have been so many cases where people find out they carry a serious problem due to poor medical performance. Since massive health data is being recorded every day, medical centers can utilize it to summarize, find patterns, reasons, and solutions in order to preclude the possibility of worse consequences in the foreseeable future.

II. OBJECTIVES

The main objective of our project is to understand and develop an application using database management concepts accompanied by Intellij IDEA (Java) and SQL.

This project requires us to predict the disease from patient-input symptoms where the disease is predicted in the form of information and knowledge, the term “ knowledge discovery process (KDD)” is coined. The raw data used in this project is collected from the web and any relevant data has been selected to be processed further. With the data we have collected it is then inputted into the database using SQL.

With that being said, the process in which a doctor can determine the illness that the patient is afflicted with is still a prolonged and strenuous task.

A smart health prediction system is known as a health-care system designed to aid health-care professionals in their decision-making process in medical situations.

With this system in place, it will greatly reduce the time in the drawn out process of identifying patterns of disease and help to work out an optimal solution. For this system to be operational it requires substantial quantities of data crucial to be used on predicting a patient's health.

III. SELF-EVALUATION

Strengths:

- Members agree on and set team goals supported outcomes and results, instead of just on the quantity of labor being done. a transparent plan can then be set about how the members are visiting achieve these objectives, as a group, likewise as each individual's contribution. This provides the team with clear direction and provides the team something to aim for collectively.
- Members feel a way of belonging to the team, are committed to their work, and really care about the success of the team.
- Team members are always happy to help others after they need a aid with work.
- Everyone is exclusive and can be ready to offer their own experiences and knowledge that others might not possess. So, members help one another plenty in designing the realistic database, furthermore because the coding part.

Weaknesses:

- Some first few weeks were chaos in organizing and planning the project because some tasks might clash with one another, which results in disconnected pieces of labor.
- At first, some team members interrupted or talked over each other. There was consistent silence from them during meetings, allusions to problems but failure to formally address them, or false consensus.

IV. CONTRIBUTION

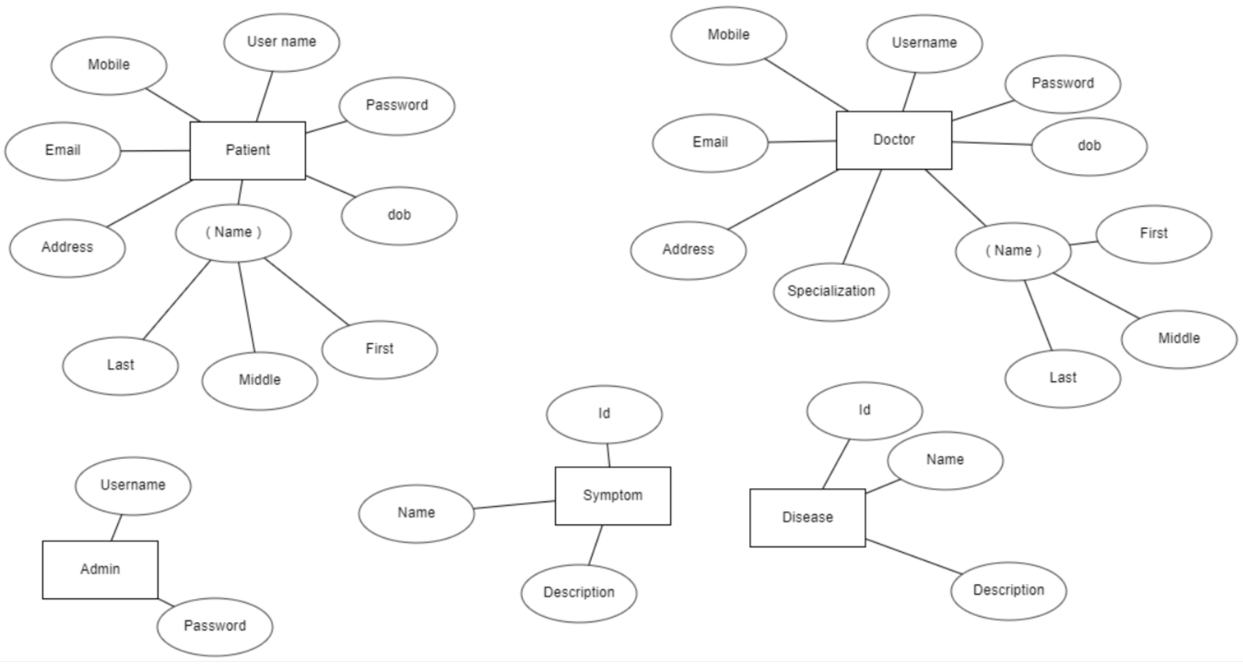
Name	Detailed tasks	Contribution (%)
Bui Thi Xuan Lan	Planning, organizing Java code, suggesting systematic ideas to build a good solid program.	30%
Nguyen Minh Trang	Design database from first draft to real product; planning, leading and keeping track of each individual's task.	25%

Phan Vo Phuong Tung	Design database from first draft to real product; give some logical methods on how to effectively predict the disease.	25%
Phan Ho Hoang Phuoc	Test the product and suggest improvement as well as solutions to fix bugs; plan on writing a detailed report.	20%

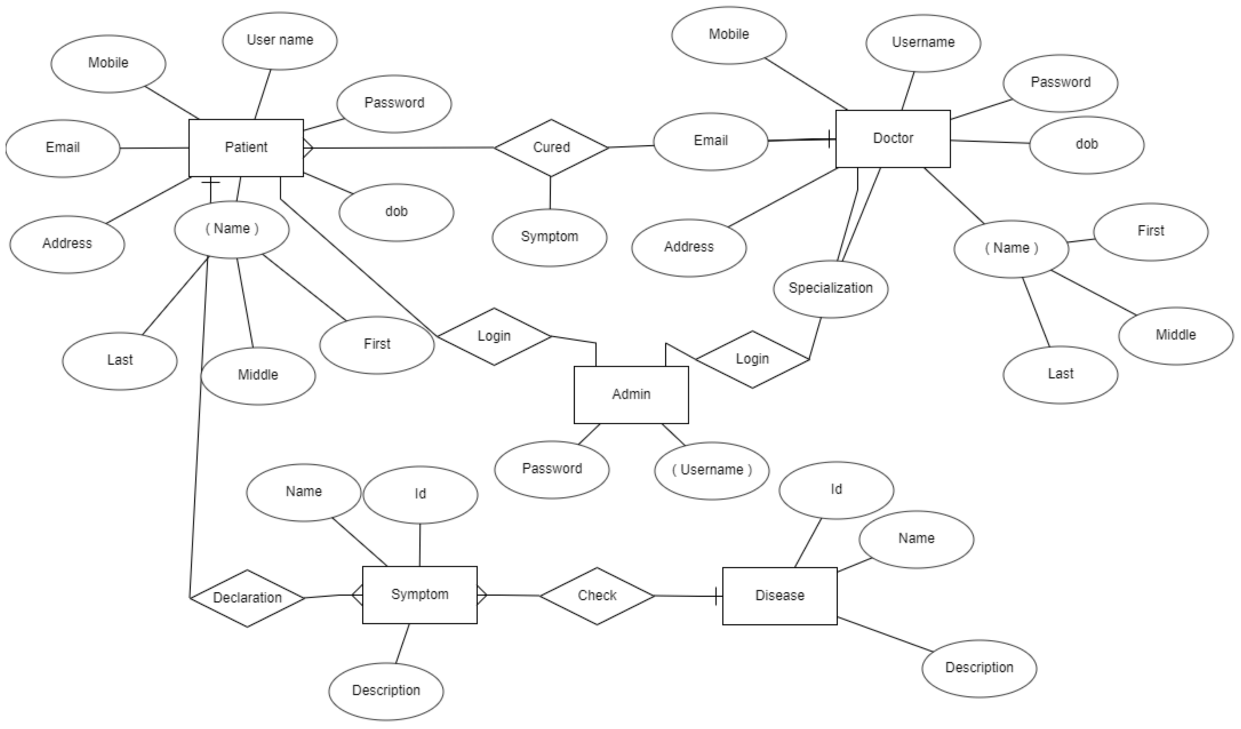
V. DEVELOPMENT

1. ERD

- First, we think about how the smart health prediction system works? We imagine a patient making an account with their information and username, password then login to the system, choose their symptoms and the system will output the disease of this patient and let the patient choose one of the doctors who specializes in that disease and make an appointment or give feedback. A doctor can see their patients and add diseases or symptoms. Admin can see the information of patients, doctors, diseases which symptoms, and feedback from patients.
- Next, we draw an ERD on the paper. We think all patients, doctors should have their information like id, first name, middle name, last name, phone, email, address, dob (date of birth) and then username, password. Furthermore, we consider that symptoms and diseases should have their own name, id and description of what they are. Finally, we need an admin who can access everything in the system and the admin does not need the information so we leave it just username and password. At the end, we have an entity on the paper like this:



- The next process is finding the relationship between these entities, doctors have to know some information about their patients, so we need a relationship between these entities like Cured, the doctor knows the symptoms of the patient and one doctor can cure many patients so we set the relationship to “one-to-many”. Next thing is the relationship between symptoms and diseases, each symptom belongs to each disease, like we know sneezing and runny nose are related to flu or fever, so a relationship like Check will check what diseases these symptoms belong to. A patient can declare many symptoms so the relationship is “one to many”. Finally, we consider the relationship between the admin and patients, doctors; login may be the suitable relationship for this case. Therefore, we update our ERD like this:



2. SQL

Script to create tables

```
CREATE TABLE [dbo].[Admin](
  [username] [nchar](20) NOT NULL,
  [password] [nchar](20) NOT NULL,
 CONSTRAINT [PK_Admin] PRIMARY KEY CLUSTERED
(
  [username] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

```

```
CREATE TABLE [dbo].[Declaration](
  [patient_id] [int] NOT NULL,
  [symptom_id] [int] NOT NULL,
  [date] [date] NOT NULL
) ON [PRIMARY]
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

```

```sql
CREATE TABLE [dbo].[Diagnosis](
  [patient_id] [int] NOT NULL,
  [disease_id] [int] NOT NULL,
  [date] [date] NOT NULL
) ON [PRIMARY]
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO


```

```sql
CREATE TABLE [dbo].[Disease](
  [d_id] [int] IDENTITY(1,1) NOT NULL,
  [d_name] [nchar](20) NOT NULL,
  [description] [nchar](1000) NULL,
 CONSTRAINT [PK_Disease] PRIMARY KEY CLUSTERED
(
  [d_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

```

```sql
CREATE TABLE [dbo].[DiseaseClassification](
  [disease_id] [int] NOT NULL,
  [specialty_id] [int] NOT NULL,
 CONSTRAINT [PK_DiseaseClassification] PRIMARY KEY CLUSTERED
(
  [disease_id] ASC,
  [specialty_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

```

```sql
CREATE TABLE [dbo].[Doctor](
  [d_id] [int] IDENTITY(1,1) NOT NULL,
  [d_firstname] [nchar](20) NOT NULL,
  [d_middlename] [nchar](20) NULL,
  [d_lastname] [nchar](20) NOT NULL,
  [phone] [nchar](11) NOT NULL,
  [email] [nchar](50) NULL,
  [address] [nchar](100) NOT NULL,
  [dob] [date] NOT NULL,
  [specialty_id] [int] NOT NULL,
  [username] [nchar](20) NOT NULL,
  [password] [nchar](20) NOT NULL,
 CONSTRAINT [PK_Doctor] PRIMARY KEY CLUSTERED
(
  [d_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

```
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

```

```
CREATE TABLE [dbo].[Feedback](
  [patient_id] [int] NOT NULL,
  [doctor_id] [int] NOT NULL,
  [description] [nchar](1000) NOT NULL,
  [date] [date] NOT NULL
) ON [PRIMARY]
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

```

```
CREATE TABLE [dbo].[Patient](
  [p_id] [int] IDENTITY(1,1) NOT NULL,
  [p_firstname] [nchar](20) NOT NULL,
  [p_middlename] [nchar](20) NULL,
  [p_lastname] [nchar](20) NOT NULL,
  [phone] [nchar](11) NOT NULL,
  [email] [nchar](50) NULL,
  [address] [nchar](100) NOT NULL,
  [dob] [date] NOT NULL,
  [username] [nchar](20) NOT NULL,
  [password] [nchar](20) NOT NULL,
 CONSTRAINT [PK_Patient] PRIMARY KEY CLUSTERED
(
  [p_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

```

```
CREATE TABLE [dbo].[PatientDoctor](
  [patient_id] [int] NOT NULL,
  [doctor_id] [int] NOT NULL,
 CONSTRAINT [PK_PatientDoctor] PRIMARY KEY CLUSTERED
(
  [patient_id] ASC,
  [doctor_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

```

```
CREATE TABLE [dbo].[Specialty](
  [s_id] [int] IDENTITY(1,1) NOT NULL,
  [s_name] [nchar](30) NOT NULL,
 CONSTRAINT [PK_Speciality] PRIMARY KEY CLUSTERED
(
  [s_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

```

```
CREATE TABLE [dbo].[Symptom](
  [s_id] [int] IDENTITY(1,1) NOT NULL,
  [s_name] [nchar](20) NOT NULL,
  [s_description] [nchar](1000) NULL,
 CONSTRAINT [PK_Symptom] PRIMARY KEY CLUSTERED
(
  [s_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

```

```
CREATE TABLE [dbo].[SymptomClassification](
  [symptom_id] [int] NOT NULL,
  [disease_id] [int] NOT NULL,
 CONSTRAINT [PK_SymptomClassification] PRIMARY KEY CLUSTERED
(
  [symptom_id] ASC,
  [disease_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

```

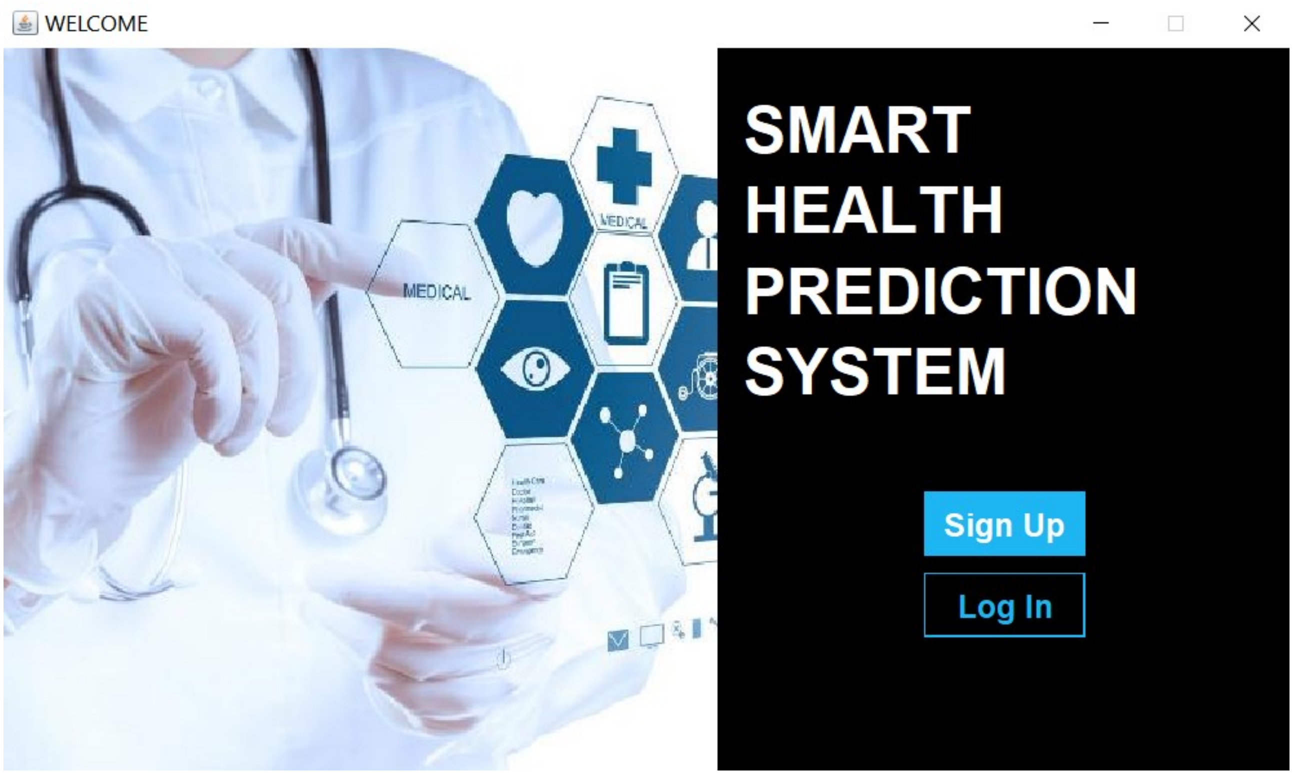
Add the Foreign Key

```
ALTER TABLE [dbo].[Declaration] ADD  CONSTRAINT [DF_Declaration_date]  DEFAULT (getdate()) FOR
[date]
GO
ALTER TABLE [dbo].[Diagnosis] ADD  CONSTRAINT [DF_Diagnosis_date]  DEFAULT (getdate()) FOR
[date]
GO
ALTER TABLE [dbo].[Feedback] ADD  CONSTRAINT [DF_Feedback_date]  DEFAULT (getdate()) FOR [date]
GO
ALTER TABLE [dbo].[Declaration]  WITH CHECK ADD  CONSTRAINT [FK_Declaration_Patient] FOREIGN
KEY([patient_id])
REFERENCES [dbo].[Patient] ([p_id])
GO
ALTER TABLE [dbo].[Declaration] CHECK CONSTRAINT [FK_Declaration_Patient]
GO
ALTER TABLE [dbo].[Declaration]  WITH CHECK ADD  CONSTRAINT [FK_Declaration_Symptom] FOREIGN
KEY([symptom_id])
REFERENCES [dbo].[Symptom] ([s_id])
GO
```

```
ALTER TABLE [dbo].[Declaration] CHECK CONSTRAINT [FK_Declaration_Symptom]
GO
ALTER TABLE [dbo].[Diagnosis]  WITH CHECK ADD  CONSTRAINT [FK_Disease_Diagnosis] FOREIGN KEY([disease_id])
REFERENCES [dbo].[Disease] ([d_id])
GO
ALTER TABLE [dbo].[Diagnosis] CHECK CONSTRAINT [FK_Disease_Diagnosis]
GO
ALTER TABLE [dbo].[Diagnosis]  WITH CHECK ADD  CONSTRAINT [FK_Patient_Diagnosis] FOREIGN KEY([patient_id])
REFERENCES [dbo].[Patient] ([p_id])
GO
ALTER TABLE [dbo].[Diagnosis] CHECK CONSTRAINT [FK_Patient_Diagnosis]
GO
ALTER TABLE [dbo].[DiseaseClassification]  WITH CHECK ADD  CONSTRAINT [FK_DC_Disease] FOREIGN KEY([disease_id])
REFERENCES [dbo].[Disease] ([d_id])
GO
ALTER TABLE [dbo].[DiseaseClassification] CHECK CONSTRAINT [FK_DC_Disease]
GO
ALTER TABLE [dbo].[DiseaseClassification]  WITH CHECK ADD  CONSTRAINT [FK_DC_Specialty] FOREIGN KEY([specialty_id])
REFERENCES [dbo].[Specialty] ([s_id])
GO
ALTER TABLE [dbo].[DiseaseClassification] CHECK CONSTRAINT [FK_DC_Specialty]
GO
ALTER TABLE [dbo].[Doctor]  WITH CHECK ADD  CONSTRAINT [FK_Doctor_Specialty] FOREIGN KEY([specialty_id])
REFERENCES [dbo].[Specialty] ([s_id])
GO
ALTER TABLE [dbo].[Doctor] CHECK CONSTRAINT [FK_Doctor_Specialty]
GO
ALTER TABLE [dbo].[Feedback]  WITH CHECK ADD  CONSTRAINT [FK_Feedback_Doctor] FOREIGN KEY([doctor_id])
REFERENCES [dbo].[Doctor] ([d_id])
GO
ALTER TABLE [dbo].[Feedback] CHECK CONSTRAINT [FK_Feedback_Doctor]
GO
ALTER TABLE [dbo].[Feedback]  WITH CHECK ADD  CONSTRAINT [FK_Feedback_Patient] FOREIGN KEY([patient_id])
REFERENCES [dbo].[Patient] ([p_id])
GO
ALTER TABLE [dbo].[Feedback] CHECK CONSTRAINT [FK_Feedback_Patient]
GO
ALTER TABLE [dbo].[PatientDoctor]  WITH CHECK ADD  CONSTRAINT [FK_PD_Doctor] FOREIGN KEY([doctor_id])
REFERENCES [dbo].[Doctor] ([d_id])
GO
ALTER TABLE [dbo].[PatientDoctor] CHECK CONSTRAINT [FK_PD_Doctor]
GO
ALTER TABLE [dbo].[PatientDoctor]  WITH CHECK ADD  CONSTRAINT [FK_PD_Patient] FOREIGN KEY([patient_id])
REFERENCES [dbo].[Patient] ([p_id])
GO
ALTER TABLE [dbo].[PatientDoctor] CHECK CONSTRAINT [FK_PD_Patient]
GO
ALTER TABLE [dbo].[SymptomClassification]  WITH CHECK ADD  CONSTRAINT [FK_SC_Disease] FOREIGN KEY([disease_id])
REFERENCES [dbo].[Disease] ([d_id])
GO
ALTER TABLE [dbo].[SymptomClassification] CHECK CONSTRAINT [FK_SC_Disease]
GO
ALTER TABLE [dbo].[SymptomClassification]  WITH CHECK ADD  CONSTRAINT [FK_SC_Symptom] FOREIGN KEY([symptom_id])
REFERENCES [dbo].[Symptom] ([s_id])
GO
ALTER TABLE [dbo].[SymptomClassification] CHECK CONSTRAINT [FK_SC_Symptom]
GO
USE [master]
GO
```

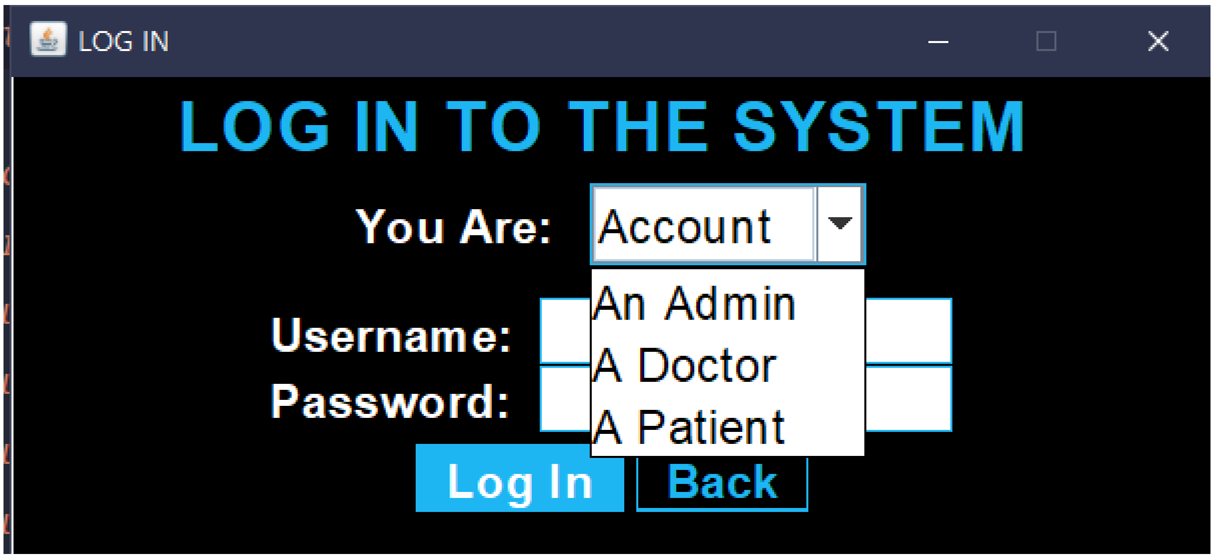
```
ALTER DATABASE [SHPS] SET  READ_WRITE
GO

```

VI. DEMONSTRATION



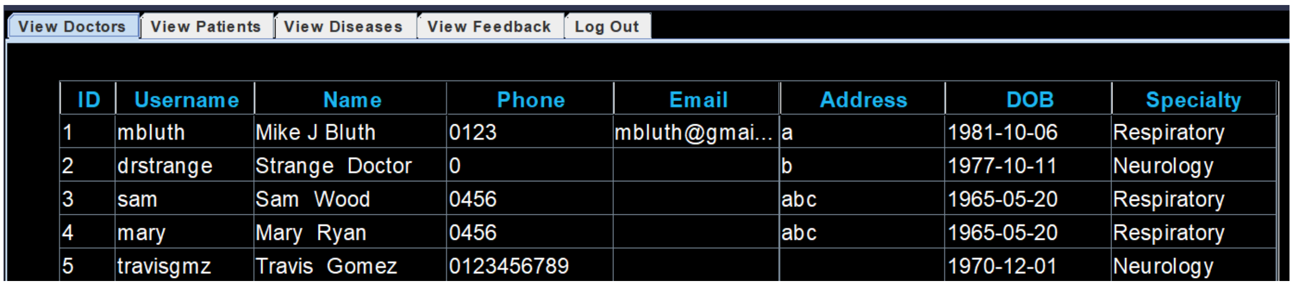
The GUI consists of 3 main sections specified for different purposes:

- Admin: who can manage everything in the database.
- Doctor: who can check their information as well as their patients’ information.
- Patient: who can check their information and most importantly use the main feature of the system to predict their disease based on the symptoms.



Login options depending on 3 account types.

What's inside the Admin Dashboard?



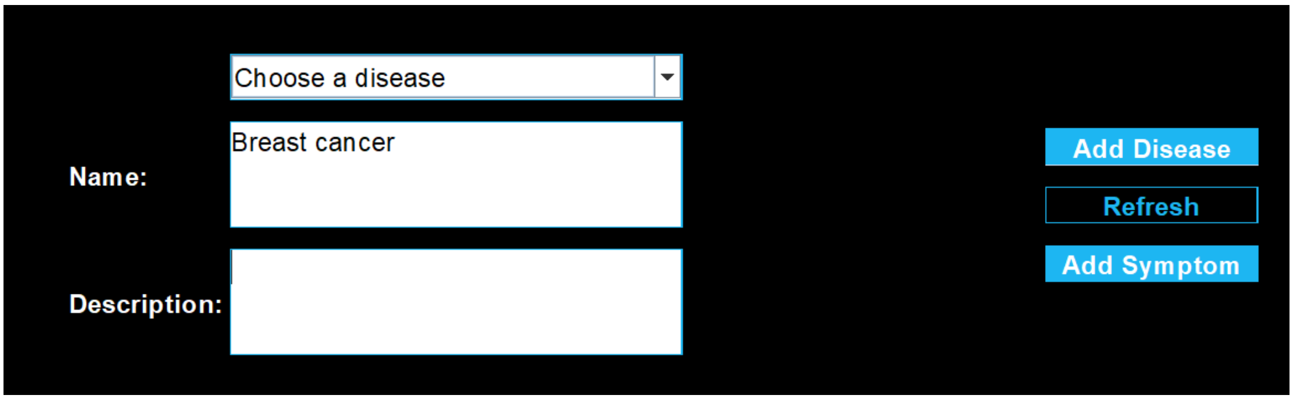
ID	Username	Name	Phone	Email	Address	DOB	Specialty
1	mbluth	Mike J Bluth	0123	mbluth@gmai...	a	1981-10-06	Respiratory
2	drstrange	Strange Doctor	0		b	1977-10-11	Neurology
3	sam	Sam Wood	0456		abc	1965-05-20	Respiratory
4	mary	Mary Ryan	0456		abc	1965-05-20	Respiratory
5	travisgmz	Travis Gomez	0123456789			1970-12-01	Neurology

Admin can view all available doctors and their information as well as patients' information. An admin can also view and add diseases and symptoms. Last but not least, admin can take a look at patients' feedback to a specific doctor with recorded time.
Login: admin - 123

What's inside the Doctor Dashboard?

A doctor can search for his/her patients based on appointments to view all the information about them including the feedback that was given to him/her

A doctor can also manage to add new diseases with their specific symptoms as well. **Note:** diseases that have been added will also be added in their specialty. Example: Mike Bluth's speciality is Respiratory so disease A will be added and classified as a disease in the Respiratory category.
Login: mbluth - 123

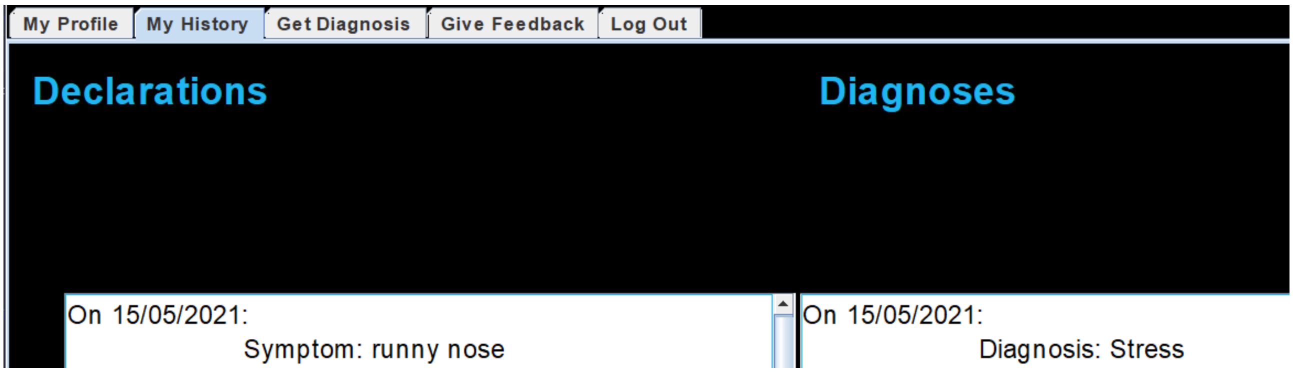


Name: Breast cancer

Description:

What's inside the Patient's Dashboard?

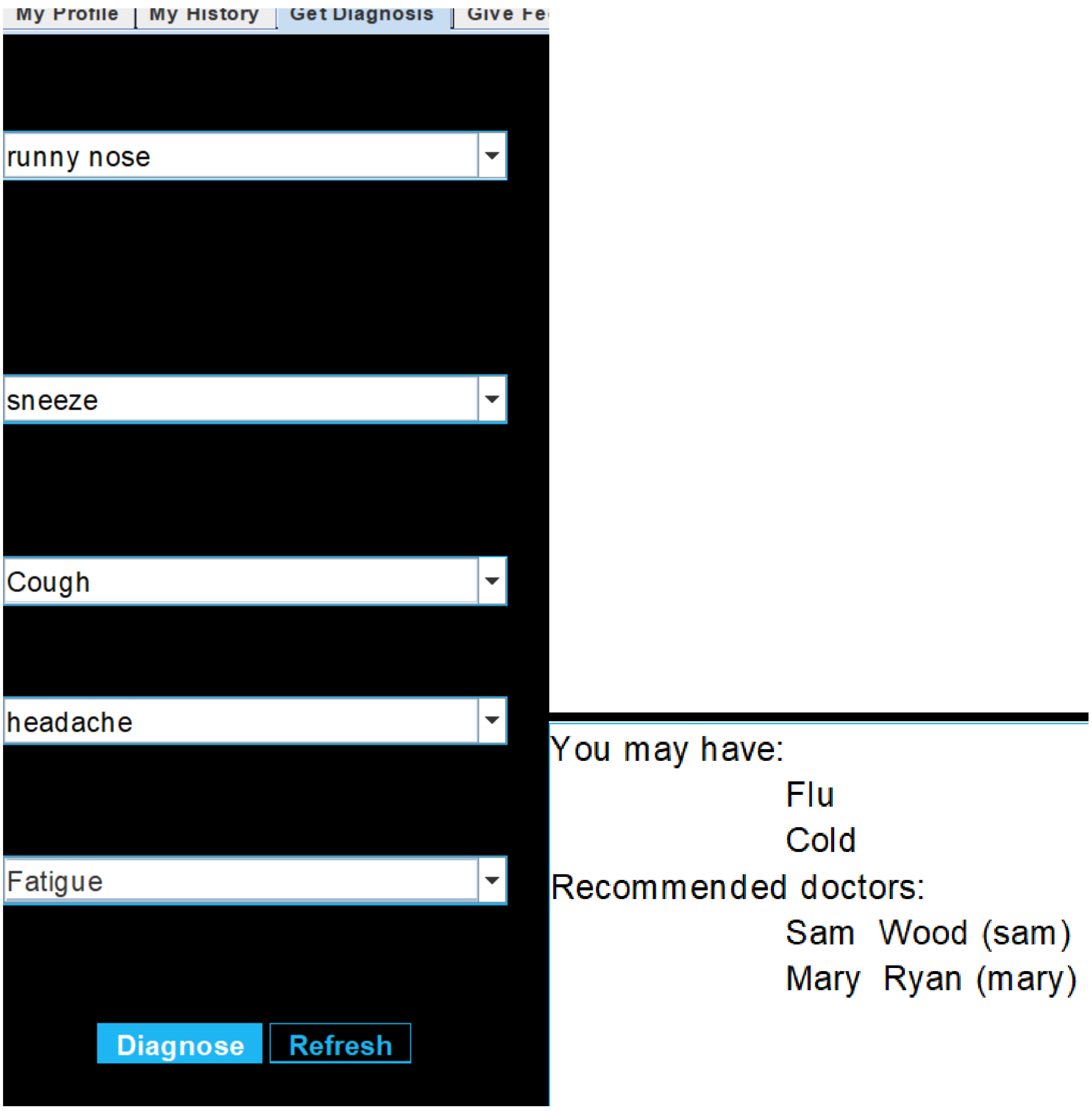
Patients can have a glance at their history activity. They can view what they had declared and what they had diagnosed.
Login: janedoe - 123



On 15/05/2021:
Symptom: runny nose

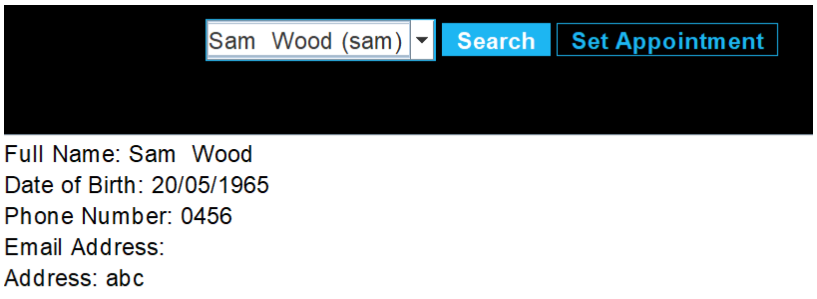
On 15/05/2021:
Diagnosis: Stress

Patients get to choose the symptoms they have experienced and get diagnosed immediately along with recommended doctors.



You may have:
Flu
Cold
Recommended doctors:
Sam Wood (sam)
Mary Ryan (mary)

What patients have to do next is just choose one doctor and set the appointment.



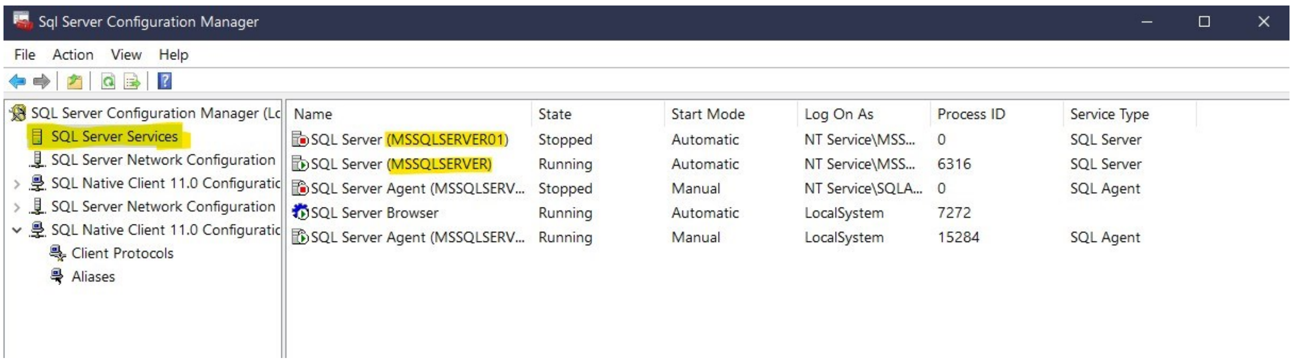
Full Name: Sam Wood
Date of Birth: 20/05/1965
Phone Number: 0456
Email Address:
Address: abc

HOW TO RUN THE PROGRAM?

- First off, make sure to backup the database using SHPS.bak in your local machine in order to access it in the application.

- For developers, you can run this program in your favourite IDE, specifically, the Main class. **Note:** Please make sure you already added the SQL JDBC JAR file in your Project Settings section. Otherwise, you won't be able to connect to the database.
- The default url to connect to the database is
  "jdbc:sqlserver://localhost;database=SHPS;integratedSecurity=true;"

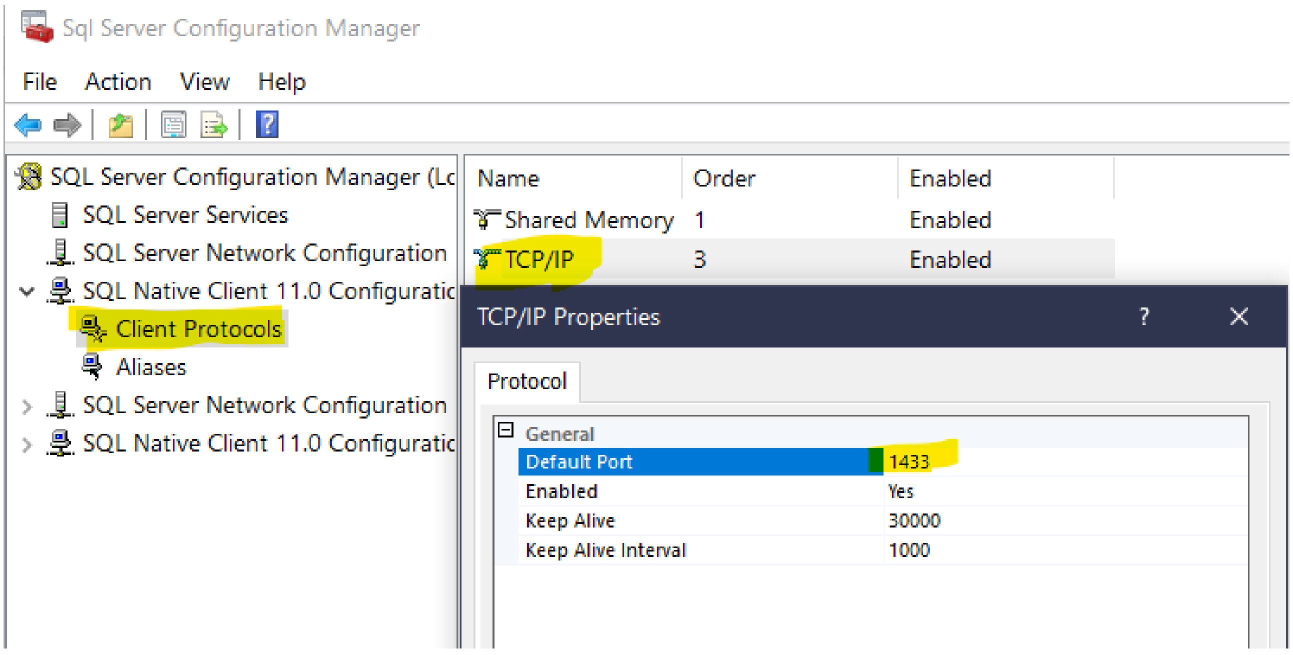
- The above url is supposed to find your default localhost name, default instance and default port of your local. If you have any problem while connecting the database, please make sure to adjust the url to fit your machine's requirements.
  How to fix?:
  - To find your localhost name, press hotkeys Windows + R and type cmd to run Command Prompt. Type hostname then press Enter to receive your localhost name.
  - To find your instance name, open SQL Server Configuration Manager. The instance name is highlighted in the right panel of the window.



SQL Server Configuration Manager (Lc
SQL Server Services
SQL Server Network Configuration
SQL Native Client 11.0 Configuratic
SQL Server Network Configuration
SQL Native Client 11.0 Configuratic
Client Protocols
Aliases

Name	State	Start Mode	Log On As	Process ID	Service Type
SQL Server (MSSQLSERVER01)	Stopped	Automatic	NT Service\MSS...	0	SQL Server
SQL Server (MSSQLSERVER)	Running	Automatic	NT Service\MSS...	6316	SQL Server
SQL Server Agent (MSSQLSERV...	Stopped	Manual	NT Service\SQLA...	0	SQL Agent
SQL Server Browser	Running	Automatic	LocalSystem	7272	
SQL Server Agent (MSSQLSERV...	Running	Manual	LocalSystem	15284	SQL Agent

- Same for finding default port. In this case, it's 1433.


Sql Server Configuration Manager
File Action View Help
SQL Server Configuration Manager (Lc
SQL Server Services
SQL Server Network Configuration
SQL Native Client 11.0 Configuratic
Client Protocols
Aliases
SQL Server Network Configuration
SQL Native Client 11.0 Configuratic
TCP/IP Properties
Protocol
General

Name	Order	Enabled
Shared Memory	1	Enabled
TCP/IP	3	Enabled

Default Port	1433
Enabled	Yes
Keep Alive	30000
Keep Alive Interval	1000

VII. CONCLUSION

- The project has successfully brought out the real value to the users as many important features are available for managing a health database system in various aspects.
- The project was designed in a clean and coherent manner, both on and under the surface. The user interface is simple, understandable at first glance. The system working underneath performs well enough to help the program run without bugs.

What needs to be improved?

- More advanced features along with advanced database design should be developed in order to bring great user experience.
- Patients should have more ways of inputting their symptoms instead of dropdown boxes. The program was hard-coded with some diseases instead of flexibly predicting it.

THE END