American University of Madaba
الجامعة الاميركية في مادبا

Department of Computer Science

Faculty of Information Technology

HOSPITAL/CLINIC INFORMATION SYSTEM

Laithe Ghaleb Said Said 1720031

Supervisor: Dr.Ahmad Ababneh

Contents

Table of Figures

1 Acknowledgment

I would like to express my deep gratitude and appreciation towards my mentors who have focused their efforts and provided me with the necessary guidance throughout my work on this project, as well as their dedication to my education throughout my bachelor's degree study.

Firstly, I am thankful for my supportive supervisor Dr.Ahmad Ababneh, and for his constant support, supervision, and patience throughout my project. His willingness to give time so generously has been very much appreciated, as well as his dedication and engagement in developing my project.

Secondly, I would like to express my appreciation towards Dr.Ahmad Al Daraiseh and Dr.Mohammad Daoud for their continuous help and guidance. In addition to all professors at the American University of Madaba who have always been supportive and encouraging throughout my study.

Finally, I would like to sincerely thank my families and close friends who have provided me with unwavering encouragement and reassurance. As none of this could have been accomplished without all these people's support.

2 Abstract

The processing of medical data is important, especially during the current circumstances where Covid-19 spread over the world and increases the number of patients in the hospital and clinic. Due to the increased demands on processing medical data, I built an information system that keeps and processes the patient medical records, reserves appointments for the patients, and processes the necessary payment transaction. I used the agile development approach during the development process. The Agile model added more flexibility in collecting the system's requirements. The system is equipped with a relational database, and we used the C# programming language to implement our system with MYSQL Workbench 8.0 CE to manage and control the database.

3 Introduction

All of us have the thought that the most important aspect of life is health. Billions of people around the world visit hospitals and clinics each day. In some countries like Jordan, there is even something called medical tourism where people travel to a country to obtain medical treatment.

Two reasons pushed me to think to develop this project. The first one is the COVID-19 pandemic, as I am writing this proposal, the whole world is going through the COVID-19 pandemic where over 120 million cases have been recorded (developers, 2021). The second reason is the large number of people who have other diseases and illnesses, and the hospitals need to keep and process their files.

3.1 Project aim

Processing a huge number of records for patients, doctors, and other staff members is a cumbersome task due to the time and resource consumption. I aim to develop an information system to completely computerize the operation of storing and organizing patients' data. Developing this information system makes the task of entering, processing, and accessing data much more convenient. Also, the proposed information system will automate records of data for patients, doctors, and other staff members.

The information system will be very user-friendly, interactive, and simple to use. Simplicity ensures that anyone can understand and use this information system, and this feature is very important because the program will not be used by computer scientists, it will be used by clinics and hospitals' staff.

What makes this IS so unique? Well, this IS will differ in having an extremely basic user-friendly UI that is responsive and fast. This information system is aimed at making the management of hospitals extremely easy and reliable. Most information systems have a very high-tech graphical user interface that confuses some users rather then getting straight to the point with basic graphics. So, my IS will have a strong structured back end with a minimalist front end UI getting the user straight to point on what they need. This is what makes my IS unique and different to many other information systems already on the market.

3.2 Candidate Solutions and Methodology

One of the most important aspect of this information system will be the data stored. Therefore, I will design a database for this information system using the database approach based on Structured Query Language.

Regarding the programming language, I will use the C# language in the visual studio. The reason for choosing C# is most hospitals and clinics use Windows as their OS. And C# is one of the strongest languages for building Windows desktop applications.

The Agile approach will be used as the software development methodology in this information system; this gives me flexibility in:

1. Changing the application as needed and working in small sprints to ensure simplicity and modularity.
2. Building robust structured code as there will be testing every time.
3. Ensuring a good quality final product because it imposes a continuous improvement.

3.3 Research Questions

1. What is HOSPITAL/CLINIC INFORMATION SYSTEM?

   The Hospital/Clinic information system is a windows application aimed at organizing the data of patients, credentials of doctors, storing and calculating payments for patients and employees, and booking appointments.

2. What is the aim of this project?

   The project aims to present a fast, reliable, simple, and user-friendly information system to aid in organizing the needs whether at clinics or hospital levels.

3. Why is it important for us?

   The health care system was under pressure during the Covid-19 pandemic. This pandemic has forced large loads of peoples on hospitals and clinics. Therefore, it is very important to keep the

medical data organized and stored in a fashionable manner and making these clinics and hospitals' work run more efficiently.

4. What are we going to use for building it?

   Programming Languages:

   - C#
   - MYSQL

   Programming Environments and platforms:

   - MYSQL Workbench 8.0 CE for the database part (back end).
   - Microsoft Visual Studio for the GUI (back end) and programming the GUI in C# programming language (Back end).

3.4 Schedule

Below will be a timetable showing the task and duration of each process for this project. I will stick to this table religiously to stay organized and not lose track and to inform you about each stage of development:

Figure 1 Timetable

Task	Start date	End date	Duration
Proposal Preparation	Week 1	Week 3	3 Weeks
Analyzing and Designing the IS	Week 4	Week 5	1 Week
Implementing the Design	Week 5	Week 9	4 Weeks
Implementing the Database	Week 9	Week 11	3 Weeks
Testing	Week 11	Week 12	1 Week
Writing IS interim written report (Manuscript)	Week 12	Week 15	3 Weeks
Present IS	Week 16	Week 16	1 Day

Note: - All these dates include regular meetings with the supervisor regularly.

4 Agile Development process

Software engineering provides us different approaches for software development. Each approach fits in a specific problem domain and works under specific conditions and constraints. I chose the agile development process to give more flexibility in the requirement collection and system modeling phases.

Agile development uses a continuous iterated process of development and testing during software development to ensure processing the needs of the client. Agile helps complete many small projects in sprints whereas a process like a waterfall method of software development divides a project into completely separated phases.

Sprint is a time of two weeks where at the beginning of each sprint a list of tasks is prioritized by the customer and completed in the time period of two weeks. At the end of the sprint, the development team and the customer meet to evaluate work and progress and make notes for a future sprint.

In Agile development, the client has always involved in the development process, which makes the customer gets exactly what he wants and know what is going on.

Agile development process also gives us the advantage of changing the requirements at any time. It focuses mainly on customer satisfaction and emphasizes rapid deployment.

(Martin, 2002)

5 Requirements' collection and Analysis

The two well-known approaches for requirements collection are interactive and Unobtrusive approaches. In this project, we chose to select an interactive method based on interviews of the candidate users of the system.

For the purpose of data collection, I chose to refer to an organization in Ohio, USA Professional Psychiatric Services aka PPS. Choosing this organization gave me access to more advanced facilities. As an example, for the convenience of being able to contact the person working at that facility on demand. Having the ability of on-demand inquiries was crucial as I am using the agile software devolvement methodology due to changes, testing, and continual improvement during the development of my information system. In a meeting with the employee at the clinic, the first and most important part of the data that I needed to gather from him was how the data in the clinic was organized. This data was crucial because once gathered I had an idea of how my Database should be organized.

For the design purposes of the IS GUI, a meeting was held with an employee at the clinic. The following requests were made by him:

- Make it extremely simple. He informed me during the meeting "The IS should be extremely simple to understand and to use to the point where a 5th grader should be able to understand and use it". So, during our meeting, he emphasized on how simple my IS should be. He also requested that there should not be many paths in the application it should be straightforward.
- Limit the number of windows that pop up.
- At the facility, there is an employee that works at the facility where this person was hired for specifically reporting to the company the IS bugs and improvements on the current system they have.
- Data should be organized by Date. Does not want to see infinite scrolling of data for the patient.
- He does not care to see graphics as much as caring about a simple but strong structured IS that runs flawlessly.
- The doctor's page should be simple where the doctor's basic information: picture, name, certificates, and degree. Doctors should be identified by ids.
- The payment system should be two parts:
    - Doctors Salary and deductions for leaves
    - Fees that should be paid by the patient.

5.1 Functional & Non-Functional Requirements

First, let me explain what functional and non-functional requirements are. Functional requirements specify or describe what the system will do or how it will behave. For example, a system must have the ability to take in input from the user and store it in the database, or the system must send an email to the user whenever a certain condition is met. So, as you can see, we are describing what the system should accomplish. As for non-functional requirements, describes how the system shall perform or behave a certain task. An example would be the system shall be efficient and reliable. Another example would be the database shall have encryption in order to have the users' confidentialities stored securely. Below are two lists of Functional and Non-functional requirements (Wasson, 2006):

5.1.1 Functional Requirements

- Keeping a store of patient's medical information, general information, and session notes, while having the ability to making amendments when needed to their data.
- Booking appointments and creation of new patients.
- The admin has the functionality to add and delete new employees and staff members and making accounts for them to access what they need.
- Calculating and storing financial data for patient payments in general and employee salaries.
- The user can view appointments, delete, and add new appointments as needed.
- The user can view financial details for patients and employees.
- The user can view the employee details.
- The ability to view all the patient's information whether general information, medical information, or session notes.

5.1.2 Non-Functional Requirements

- The application should have a sense of security (encryption and decryption of passwords).
- The application needs to be fast.
- The application should be user-friendly and easy to understand by anyone.
- Simplicity is key with the GUI.
- All data stored in the database will be secured to maintain data integrity.
- The code will be written clearly and as simple as possible to make future changes the code effortless as possible.

5.2 Scenarios discovered during the requirement collection phase

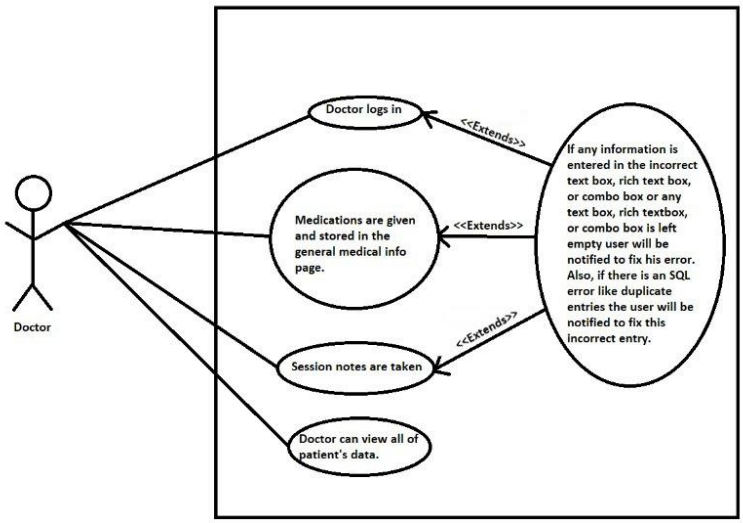
(uml-use-case-diagram, n.d.)

During the design phase I also went ahead and made four use case diagrams (figures 2 – 5) to depict different types of scenarios for this IS. But first what is a use case diagram and what do all the symbols in a use case diagram represent.

A use case diagram is a form of system requirements processing for the new software under development. A use case diagram consists of three major notations. First, the actor which in the diagrams below are represented by a stickman, an actor is someone that interacts with the system.

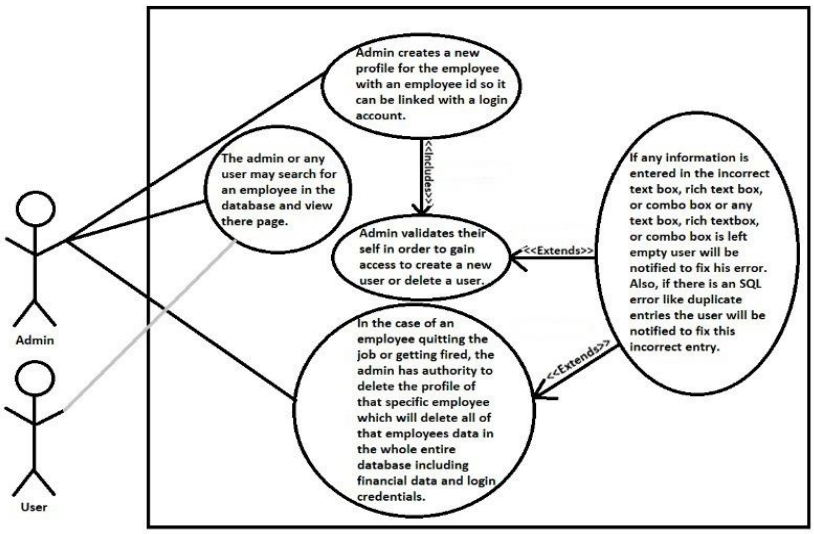
Second notation is use case which is represented by the ovals consisting of system functions. A use case may not always be linked to an actor, but an actor is always linked to a use case. We have the communication link which is a solid line as you can see in the use case diagrams below. With these links there is a use case relationship which defines the relationship between to cases. The use case relationships I used are extends and includes. Extends relationship is used when a use case adds steps leading to another use case. Includes relationship shows a case is broken down into smaller cases. Lastly, we have the big rectangle that contains all the use cases as you can see in the diagrams below which is called the boundary system which defines the system.

Figure 2 Use Case Diagram for Doctor



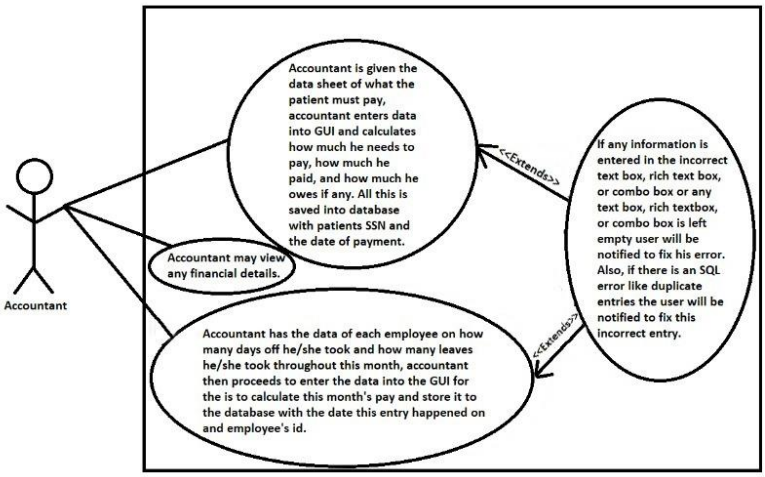
This specific use case is explaining when a doctor logs in and what a doctors can do. If a doctor logs in unsuccessfully then he will get an error which is represented by the big oval in the above figure. Then, He has 3 other cases where the doctor may add session notes, medications and medical info, or viewing data.

Figure 3 Use Case Diagram for Admin & User



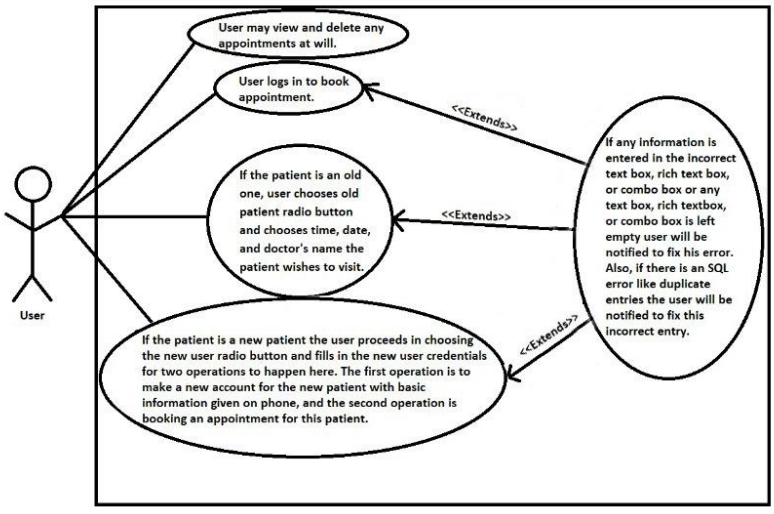
In this use case diagram as you can see, we have two actors, the first being the admin which has a relationship with the creating a new employee with login credentials, searching for an employee, and deleting all the data for a certain employee on the system. The second actor is denoted by user and has a relationship with the case of having the ability to search for employees and viewing their information page.

Figure 4 Use Case Diagram for Accountant



This use case diagram is describing the segment of the relationship between the actor accountant and the cases. The accountant has a relationship with having the ability to view and add payments for patients and employees for the whole system and these cases have an extended relationship with the last case which is if the employee enters invalid data in the wrong spots.

Figure 5 Use Case Diagram for User

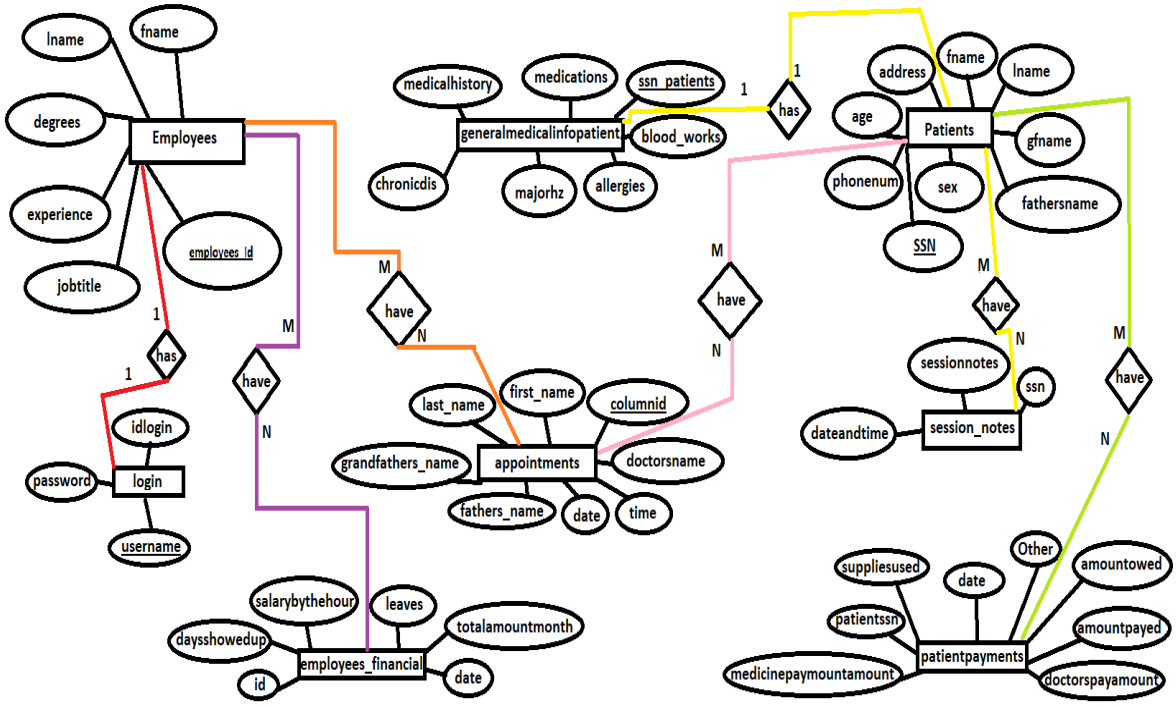


The last use case diagram represents the cases that could happen with the actor in case of booking an appointment. In this diagram the actor may delete an appointment, book an appointment whether for a new or old patient, and viewing appointments.

6 System Design Phase

The requirement collection phase of my project gave me a complete picture of all the entities in the problem domain. And I reflected my understanding in the following Entity Relationship Diagram (figure 6):

Figure 6 ERD for Database



(Earp, 2003)

6.1.1 Patient Data

First, I took down notes on the type of data and how it is organized for the patients. Patients' data should be organized into two sections, left being general information ex: - first name, father's name, grandfather's name, last name, address, phone number, SSN, age, and sex. This was later organized in the patient's table. The right section consisted of the medical medications, blood works, allergies, medical history of patient including (major hospitalization, medical history, and chronic diseases), and session notes with the date. Left and right does not mean that the data is stored physically on the right side or left side of a paper or chart, it is to express how they refer to different information for the patient.

The right section helped me to form two different tables. The first table being the general medical information for the patient named ‘generalmedicalinfopatient’, the second table is named ‘session notes’ which included the session notes for the patients with the date and SSN for each patient.

The ‘generalmedicalinfopatient’ table includes a field named ssnpatients which is linked to the table patients ssn, the options chosen for the foreign key where cascade on update and delete. Fields in ‘generalmedicalinfopatient’ are: ssn_patients, medications, blood_works, allergies, majorhz, medicalhistory, chronicdis. The table named patients includes the fields: fname, fathersname, gfname, lname, ssn, phonenum, age, sex, address. This table is not linked with a foreign key to another table.

6.1.2 Sessions’ data

A separate table for the session notes was created for the purpose of storing session notes for each patient’s session. Three fields were added: ssn, sessionnotes, and dateandtime. The field ssn was made into a foreign key and linked to the table patients and field ssn. The options chosen for the foreign key where cascade on update and delete.

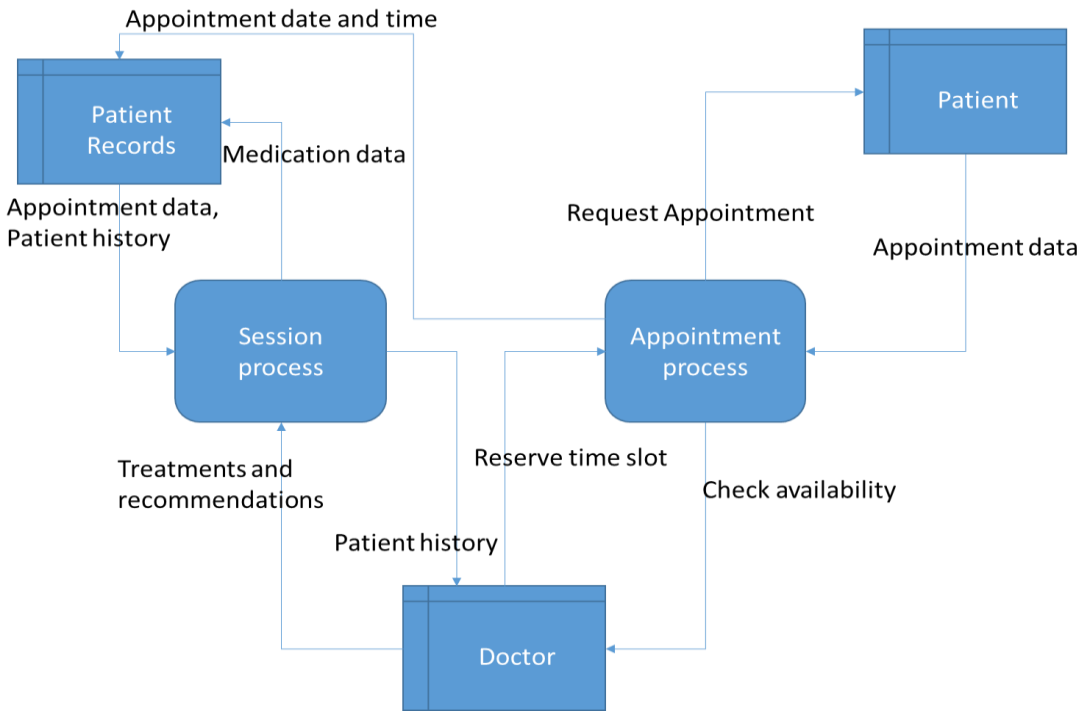
6.1.3 Appointment data

For the table appointments, I created a table named appointments with the fields: columnid (this column is autoincremented and for the purpose of being able to know which row I selected in the UI and delete that row), first_name, fathers_name, grandfathers_name, last_name, date, time, and doctorsname. This table is made for strictly storing the data of the appointments booked by the patients and with which doctor it was booked.

6.1.3.1 Appointment and sessions processes

Three external entities are interacting during the appointment reservation and treatment session; the Patient, patient records and the doctor, the following data flow diagram (Figure 7) summaries the two processes.

Figure 7 Data Flow Diagram



The appointment process starts when the patient requests an appointment, then the appointment system supply the patient with the date and time after checking the availability of the request doctor, and the appointment process also updates the patient's record to register the appointment details if this is a new patient, in case of an old patient the record of the old patient remains the same.

The session process provides the doctor with the available historical data about the patient. After the session, the doctor supplies the treatment details, recommendations, session notes and accordingly the session process updates the patient records, general medical information, and session notes.

6.1.4 Employees Data

Now for the employees, the data they stored and displayed on their website consisted of first name, last name, Education and Credentials, and experience. Ex. "The clinic had a former FBI agent that was a forensic phycologist hostage negotiator, and the description was used on his profile on the website" showed to me by the employ I was referring to. I then formed a table in my database named employees which consisted of the following fields: employee's id, first name, last name, degrees, experience (this will describe the employees experience and his or her description).

The table login was created with only 3 fields: username, password, idlogin. This table was created for the purpose of login credentials of employees. The field idlogin is a foreign key link with the field in employees named employees_id. The options chosen for the foreign key where cascade on update and delete.

6.1.5 Financial data

Now for the financial section in the database I had two separate tables made, the first table named employees_financial which is strictly for storing the data regarding employees. This table includes the following fields: id, daysshowedup, salarybythehour, leaves, totalamountmonth, and date. The field ssn is the foreign key of a table referencing to the table employees field name employees_id. The options chosen for the foreign key where cascade on update and delete. For the second table a table was created named patientpayments which is dedixated for storing the financial data of patients. This table included the following fields: doctorspayamount, medicinepaymountamount, patientssn, suppliesused, amountpayed, amountowed, Other (this is if the patient had something else to pay for not specified), and date. The field patientssn was chosen as the foreign key which is referring the table patients field name ssn. The options chosen for the foreign key where cascade on update and delete.

7 Design and Implementation of GUI

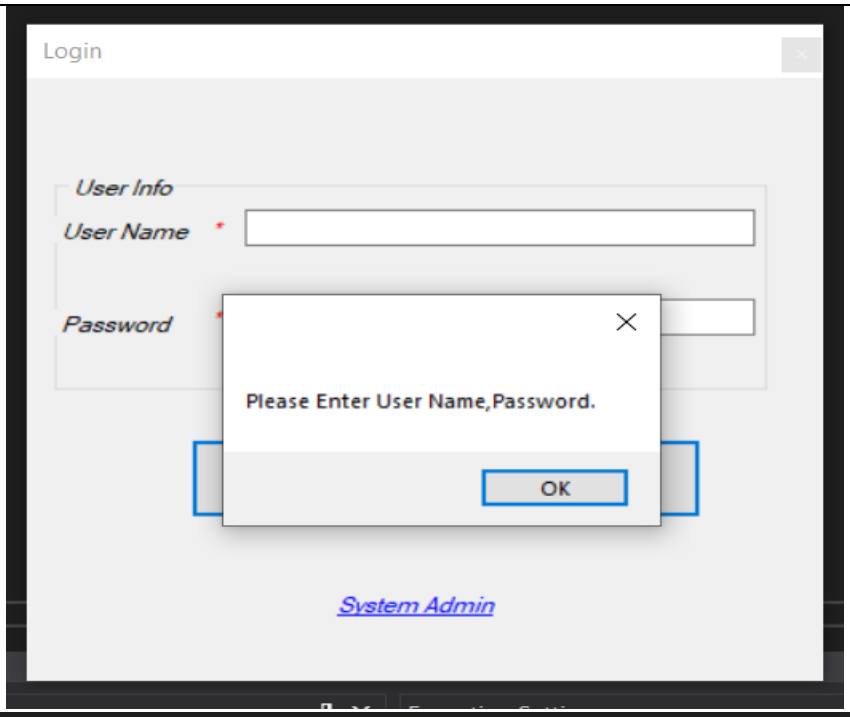
7.1 Repeated code

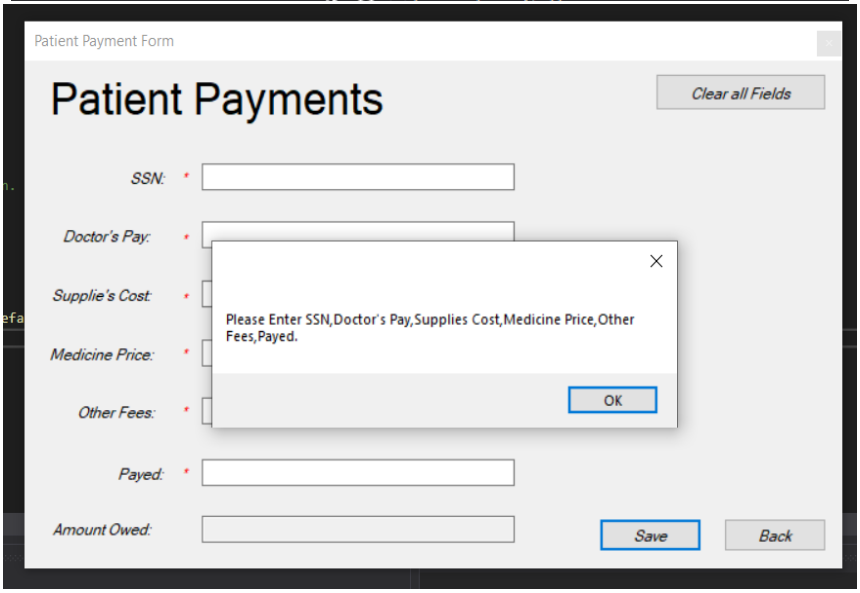
Now I will be explaining the implementation of the GUI. First, I will take segments of code out of my project that were repeated throughout the project and explain them. Then I will show pictures for every single form and how it runs while explaining the design.

First, I will explain the form validation. Form validation is when the data entered by the user is entered incorrectly, for example, a user enters a patient's SSN by accident into the first name textbox. If the data entered is validated before the code continues inserting this data to the DB it could cause the IS to crash because you are inserting an integer into a column with a datatype of VAR. if the IS does not crash, the system will send data to incorrect places in the DB caused by human error. The first method that I repeated throughout the code was using if statements to check if the textbox is empty. This segment of code is part of the login form:

```
string result = "Please Enter ";
            if (textBox1.Text == string.Empty)
            {
                result += "User Name,";
                label10.Visible = true;
            }
            if (textBox2.Text == string.Empty)
            {
                result += "Password,";
                label3.Visible = true;
            }
            if (result != "Please Enter ")
            {
                result = result.Remove(result.LastIndexOf(","));
                MessageBox.Show(result + '.');
            }
            else
                  {

      //continues with code

                    }
```

7.2 Login Page Validation





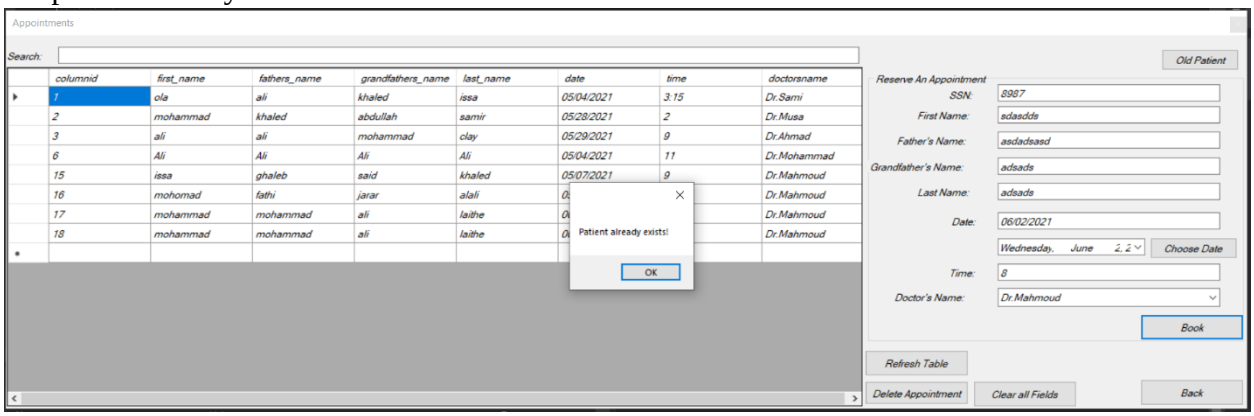
The first picture above shows a message box that will appear prompting the user to enter the empty fields if the user forgets to enter login credentials. The second picture shows the same concept but for the patients' payments form. Using this type of form validation for checking empty textboxes was the best as it does not trap the user in a certain textbox. This concept of form validation is repeated throughout the project in any form where the user is required to input data.

Another part of code that is repeated throughout the code is the try{} catch(){} statement. This statement allows the programmer to enter the code to be executed in the try block, then it will be tested as executed, if an error occurs the catch block will be executed. I used this when executing SQL statements

in my code in case there is a SQL to display an error message instead of the IS crashing from duplicate entries.

```
try
{
    string MyConnection2 = "datasource=localhost;port=3306;username=root; password=123456789ls";
    string Q = "insert into clinichospitaldb.patients(fname,fathersname,gfname,lname,ssn)values" +
    "('" + this.textBox1.Text + " " + "','" + this.textBox2.Text + " " + "','" +
    this.textBox3.Text + " " + "','" + this.textBox4.Text + " " + "','" + this.textBox9.Text + "');";
    MySqlConnection MyConn3 = new MySqlConnection(MyConnection2);
    MySqlCommand MyCommand3 = new MySqlCommand(Q, MyConn3);
    MySqlDataReader MyReader3;
    MyConn3.Open();
    MyReader3 = MyCommand3.ExecuteReader();     // Here our query will be executed and data saved into the database.
    MessageBox.Show("New patient added!");
    while (MyReader3.Read())
    {
        Console.WriteLine(String.Format("{0}", MyReader3[0]));
    }
    MyConn3.Close();
    string Query = "insert into clinichospitaldb.appointments(first_name,fathers_name,grandfathers_name,last_name,date,time,doctorsname)values" +
        "('" + this.textBox1.Text + " " + "','" + this.textBox2.Text + " " + "','" +
        this.textBox3.Text + " " + "','" + this.textBox4.Text + " " + "','" + this.textBox5.Text +
        "','" + this.textBox6.Text + "','" + this.comboBox1.SelectedItem + "');";
    MySqlConnection MyConn2 = new MySqlConnection(MyConnection2);
    MySqlCommand MyCommand2 = new MySqlCommand(Query, MyConn2);
    MySqlDataReader MyReader2;
    MyConn2.Open();
    MyReader2 = MyCommand2.ExecuteReader();     // Here our query will be executed and data saved into the database.
    MessageBox.Show("Appointment booked!");
    while (MyReader2.Read())
    {
        Console.WriteLine(String.Format("{0}", MyReader2[0]));
    }
    MyConn2.Close();

}
catch (Exception)
{
    MessageBox.Show("Patient already exists!");
}
```

The following picture is form 3 where the user books appointments for patients, as you can see if the user enters a duplicate and a SQL error occurs the program does not crash. The catch block is executed displaying a message box informing that this is not a new patient you are adding this an old patient and the patient already exists in the DB.



So, as you can see if this patient already exists in the patients' table and the user is trying to add a new patient that already exists the error will be caught, and the user will be informed with the error.

Down Below are segments of code that were used throughout the IS with Explanations. This will help on understanding how the project runs so when I show snapshots of the code and GUI it will make more sense.

1. These libraries are essential for establishing connections and manipulating data in the DB.

```
using MySql.Data;
using MySql.Data.MySqlClient; //to connect to the database and get some info about the MySQL server.
```

2. Providing elements of the MYSQL data provider. Like DB password and username etc.

```
string MyConnection2 =
"datasource=localhost;port=3306;username=root;password=123456789ls";
```

3. This is the creation of an instance object class MySqlConnection to make a connection String to establish a connection to the database.

```
MySqlConnection MyConn2 = new MySqlConnection(MyConnection2);
```

4. This is the insert statement that will be stored in variable Q of type string that includes the data typed in the textboxes by the user and go into the database.

```
string Q = "insert into clinichospitaldb.patients(fname,fathersname,gfname,lname,ssn)values" +
                    "('" + this.textBox1.Text + " " + "','" + this.textBox2.Text + " " + "','" +
                    this.textBox3.Text + " " + "','" + this.textBox4.Text + " " + "','" +
this.textBox9.Text + "');";
```

5. Initializes a new instance of the MySqlCommand class with the text of the query stored in this case variable q and the MySqlConnection stored in a variable of type string MYConnection2.

```
MySqlCommand MyCommand3 = new MySqlCommand(Q, MyConn3);
```

6. This is an instance of the data reader class for the MYSQL.

```
  MySqlDataReader MyReader3;
```

7. These are methods from class MySqlConnection that open and closes connections to the sMYSQL server.

```
MyConn3.Open();//opens connection
```

```
MyConn2.Close();//closes connection
```

8. Here our query will be executed and data saved into the database.

```
 MyReader3 = MyCommand3.ExecuteReader();
```

9. This is a message box to alert the user if something they entered was not successful.

```
MessageBox.Show("New patient added!");
```

10. A while loop to read and retrieve a row from the query results.

```
  while (MyReader3.Read())
                    {
                        Console.WriteLine(String.Format("{0}", MyReader3[0]));
                    }
```

11. An If statement to check if the textbox is only receiving numbers, if anything else is received, the system alerts the user with a message box and delete the non-number data entered.

```
if (System.Text.RegularExpressions.Regex.IsMatch(textBox2.Text, "[^0-9]"))
            {
                MessageBox.Show("Please enter only numbers.");
                textBox2.Text = textBox2.Text.Remove(textBox2.Text.Length - 1);
            }
```

12. This segment of code clears all fields(text boxes) on the form.

```
Action<Control.ControlCollection> func = null;

            func = (controls) =>
            {
                foreach (Control control in controls)
                    if (control is TextBox)
                        (control as TextBox).Clear();
                    else
                        func(control.Controls);
            };

            func(Controls);
```

13. the following segment of code is for the back button. It hides the form and makes an instance of another form and opens the previous form. Using hide is better than using close when intending to reopen the window. It's faster because this.Hide(); doesn't physically delete the form from memory while close deletes the form from memory and if it needs to be reopened it need to completely recreate the form.

```
            this.Hide(); // hide function hides this form. always use hide instead of close
function when intending on going back to this form.
            Form9 parent = new Form9(); //creating instance of class Form1
            parent.ShowDialog();// shows form 1 with show dialogue function
```

14. The following line of code loads data into the table.

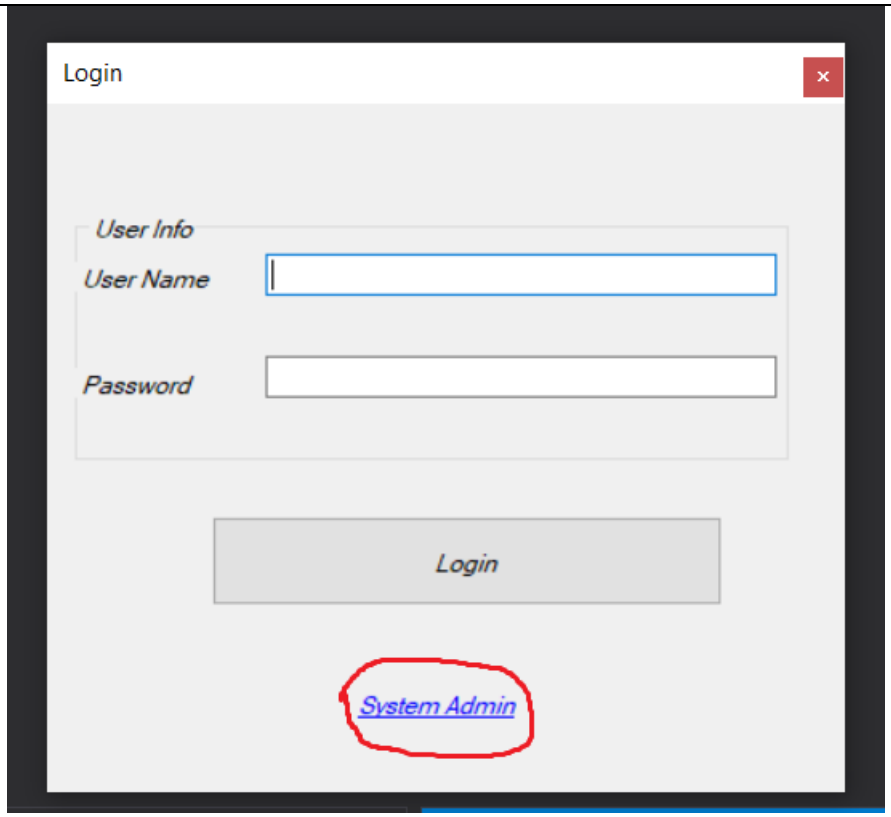
```
// TODO: This line of code loads data into the 'clinichospitaldbDataSet.appointments' table. You
can move, or remove it, as needed.
            this.appointmentsTableAdapter.Fill(this.clinichospitaldbDataSet.appointments);
```

15. `this.CenterToScreen();`. This code centers the window's start position to the center of the screen.

```
2 references
public Form4()
{
    InitializeComponent();
    this.CenterToScreen(); //centers form in middle of screen
}
```

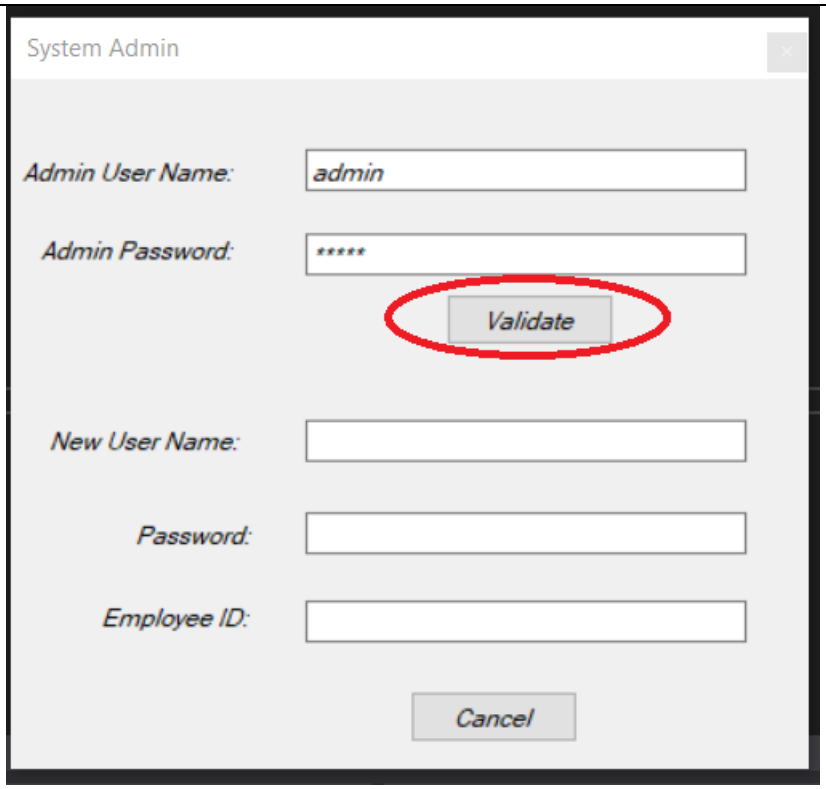
```
public Login()
{
    InitializeComponent();
    this.CenterToScreen();
}
```

7.3 Login Page Description

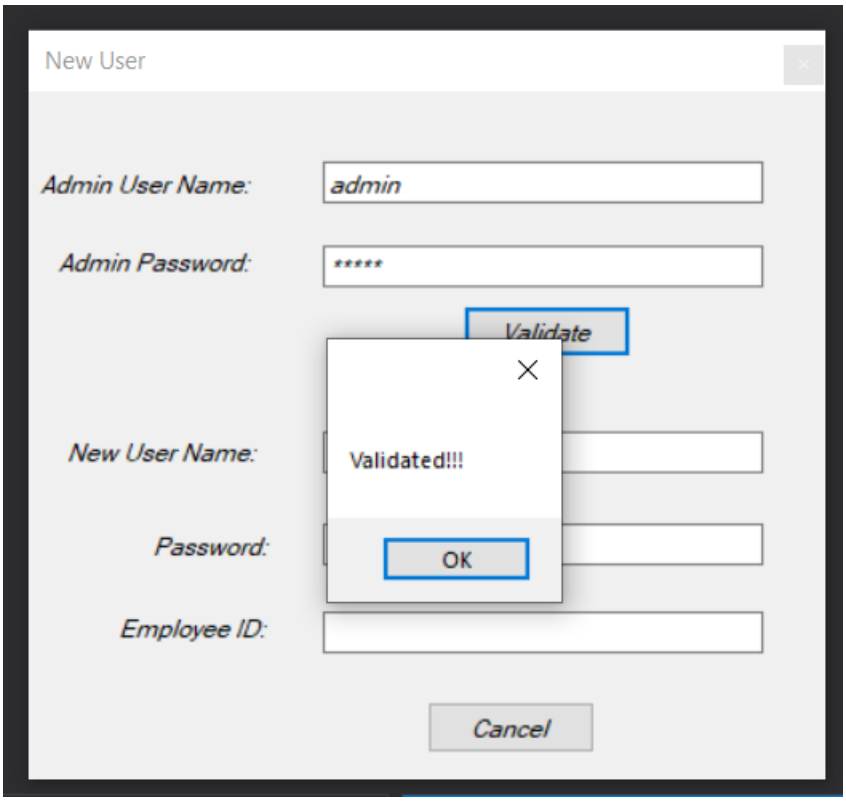


This is the login page; it is the first window to run when opening the program. Clicking on the link System Admin will take you to the next page where only the admin can add new users and delete users.

7.4 System Admin Page Description(Add a New User or Delete An Old User)



This is the system admin window in this page the admin enters the username and password to make a new user or delete an employee's account. As you can see the delete users link is not visible nor is the save button before admin validation.

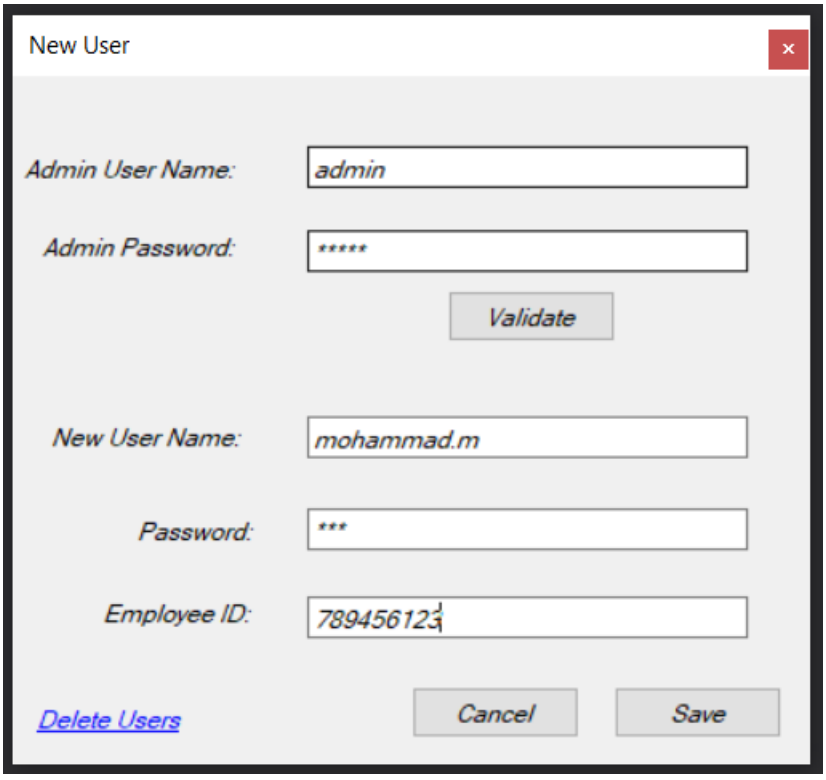


Here the admin is validated and prompted with a message box that the validation was successful. Notice in the select statement how the password is encrypted using md5. Down below is the code that is executed for the validation of the admin:

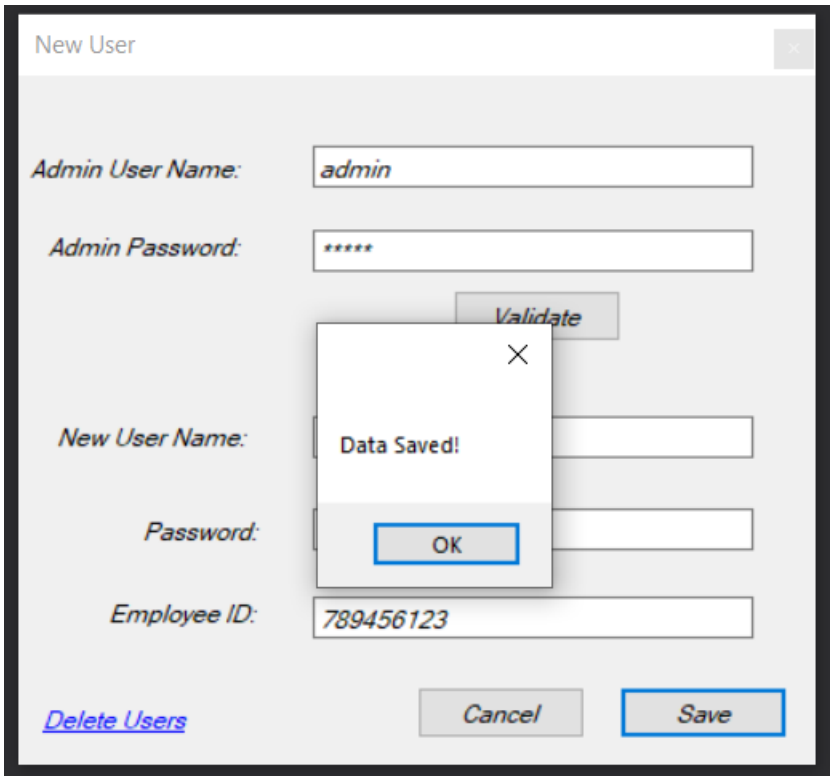
```
            if (String.IsNullOrEmpty(textBox1.Text))
            {
                MessageBox.Show("Admin user name empty.");
                label9.Visible = true;
            }
            else if (String.IsNullOrEmpty(textBox2.Text))
            {
                MessageBox.Show("Admin password empty.");
                label6.Visible = true;
            }

            else
            {
                string query = "SELECT username, password FROM clinichospitaldb.login where
username=" +
         "'" + textBox1.Text + "'and password = md5('" + textBox2.Text + "')";
                MySqlConnection connection2 = new
MySqlConnection("datasource=localhost;port=3306;username=root;password=123456789ls");
                MySqlDataReader mdr;
                MySqlCommand command = new MySqlCommand(query, connection2);
                connection2.Open();
                var username = "";
                var password = "";
```

```
            mdr = command.ExecuteReader();
            // reads the data and fills the combo box and listbox

            if (mdr.HasRows)
            {
                while (mdr.Read())
                {
                    username = mdr["username"].ToString();
                    password = mdr["password"].ToString();

                }

                MessageBox.Show("Validated!!!");
                connection2.Close();
                button1.Visible = true;
                label9.Visible = false;
                label6.Visible = false;
                linkLabel2.Visible = true;
            }
            else
            {
                MessageBox.Show("Invalid admin username or password.");
            }
        }
```

The first if statements are for validation if they are left empty the stars will appear and a message box will alert the admin what textbox is left empty. After the if statements is the SQL segment where the username and password are matched against the credentials in the database.



In this window, since the admin is successfully validated, the delete users link has appeared and the save button as well. Now the admin can make a login account to an already existing doctor. So, if the employee does not exist in the employee table admin cannot make a login for the user.



The operation is successful, and the admin is prompted with a message box that the data is saved.

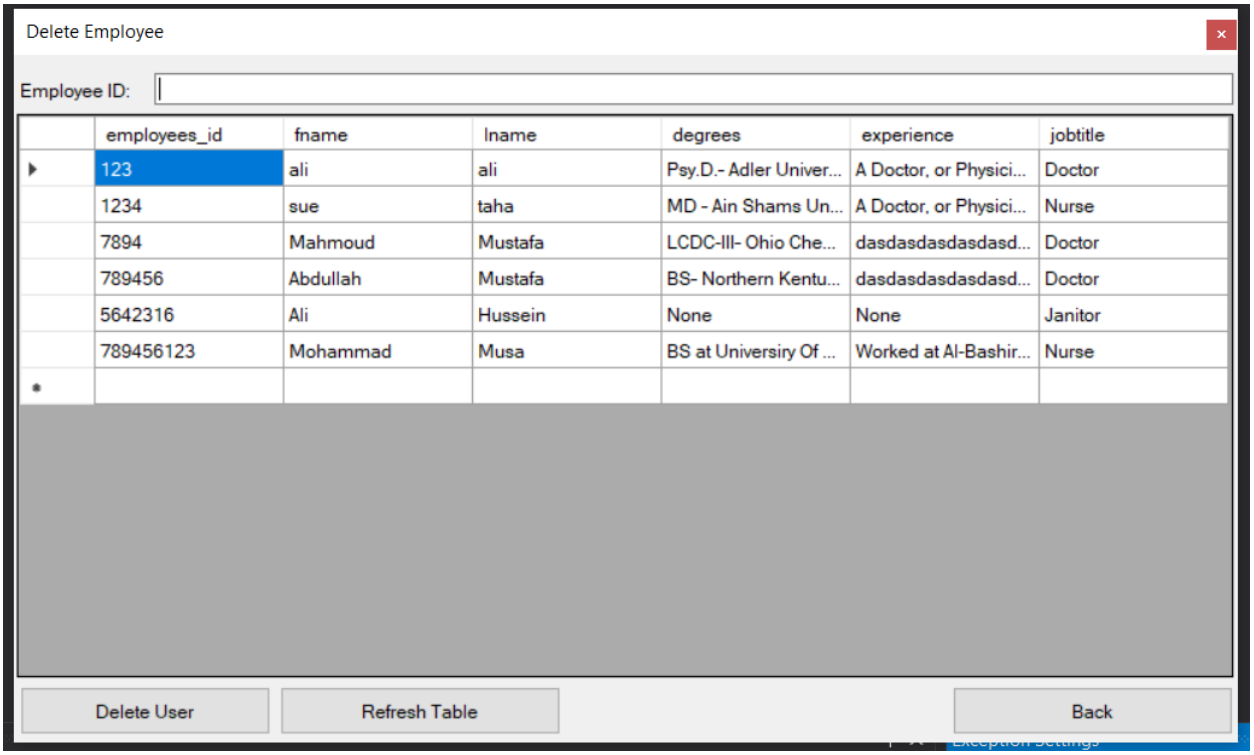
.

```
if (String.IsNullOrEmpty(textBox3.Text))
          {
              MessageBox.Show("New user name empty.");
              label7.Visible = true;
          }
          else if (String.IsNullOrEmpty(textBox4.Text))
          {
              MessageBox.Show("Password empty.");
              label8.Visible = true;
          }
          else if (String.IsNullOrEmpty(textBox5.Text))
          {
              MessageBox.Show("Employeed ID empty.");
              label10.Visible = true;
          }

          else
          {
              try
              {
```

```
                    string Connection1 = "datasource=localhost;port=3306;username=root;
password=123456789ls";

                    string Query = "insert into
clinichospitaldb.login(username,password,idlogin)values" +
                        "('" + this.textBox3.Text + "'," + "md5('"+this.textBox4.Text  + "'),'"
+ this.textBox5.Text + "');";

                    MySqlConnection Conn1 = new MySqlConnection(Connection1);
                    MySqlCommand Command1 = new MySqlCommand(Query, Conn1);
                    MySqlDataReader Reader1;
                    Conn1.Open();
                    Reader1 = Command1.ExecuteReader();
                    Conn1.Close();
                    MessageBox.Show("Data Saved!");


                    this.Close();
                    Login l1 = new Login(); //creating instance of class Form1
                    l1.ShowDialog();// shows form 1 with show dialogue function
                    this.Close();
                }
                catch (Exception)
                {

                    MessageBox.Show("User already exists.");
                }
            }
```

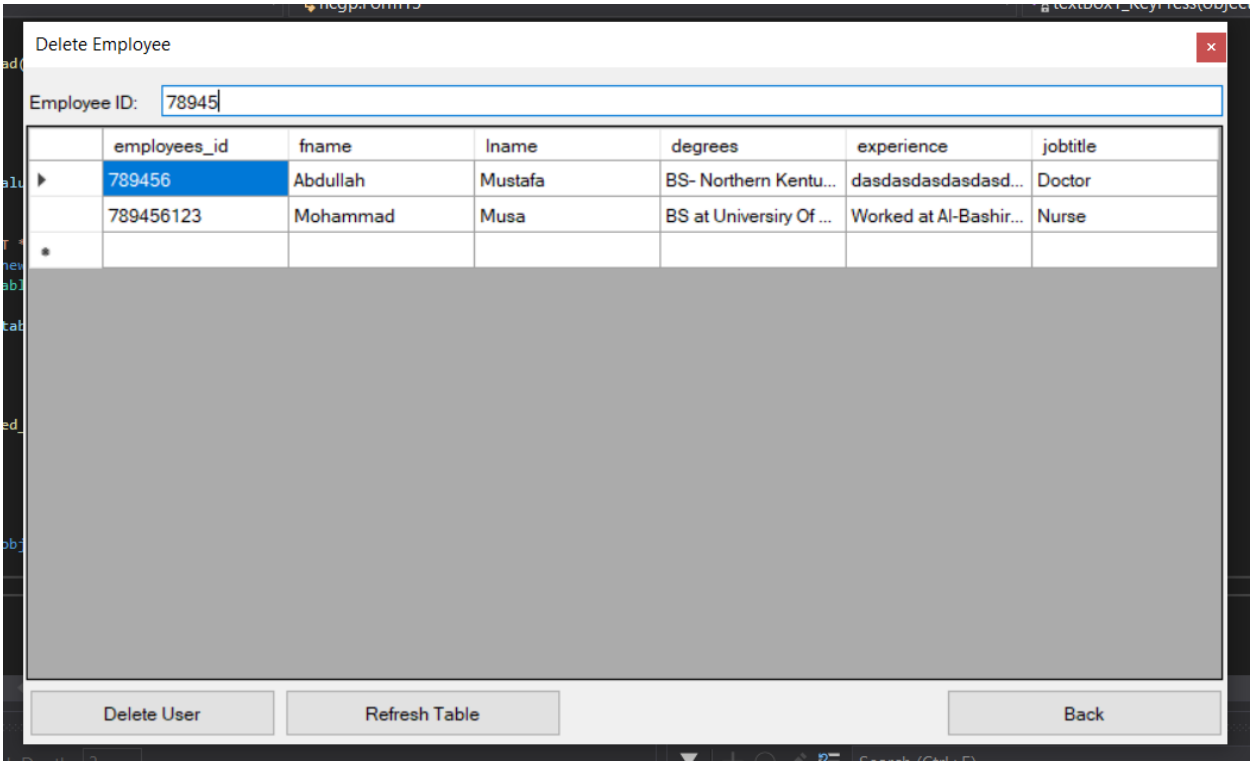
the picture below is the login table. It shows the new user is added and the passwords are encrypted.

	username	password	idlogin
▶	admin	21232f297a57a5a743894a0e4a801fc3	NULL
	mohammad.m	698d51a19d8a121ce581499d7b701668	789456123



This is the delete employee's page. Notice this is displaying the employees' table and not the login table, that is because when an employee leaves the organization all his data must be wiped:

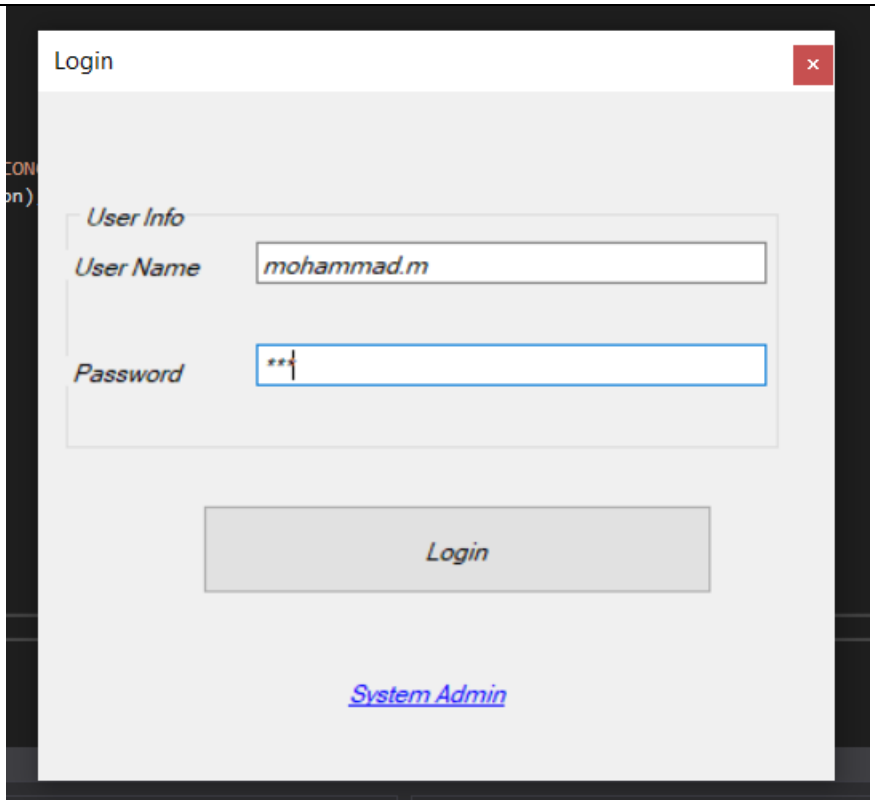
```
   private void button2_Click(object sender, EventArgs e)
        {
            DataRow row = (dataGridView1.SelectedRows[0].DataBoundItem as DataRowView).Row;

            using (MySqlConnection sqlConn = new
MySqlConnection("datasource=localhost;port=3306;username=root;password=123456789ls"))
            {
                sqlConn.Open();

                using (MySqlCommand sqlCommand = new MySqlCommand("DELETE FROM
clinichospitaldb.employees  WHERE employees_id = " + row["employees_id"], sqlConn))
                {
                    sqlCommand.ExecuteNonQuery();
                }
            }
            dataGridView1.Rows.RemoveAt(this.dataGridView1.SelectedRows[0].Index);
            MessageBox.Show("User deleted.");

        }
```



Here I used a textbox to filter data to make it more user-friendly and easier to look for employees. This method is called lazy loading or filtering. Below is the code for this operation notice how every time a key is pressed it is using a select statement to filter and display in the data grid view:

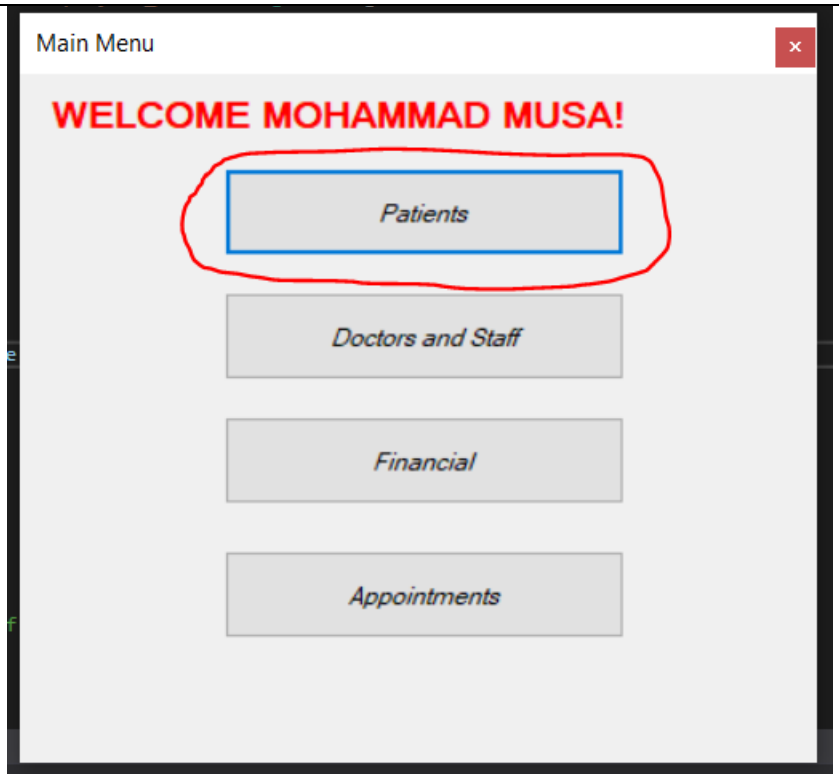
```
    private void textBox1_Search_Load(object sender, EventArgs e)
        {
            searchData("");
        }

        public void searchData(string valueToFind)
        {

            string searchQuery = "SELECT * FROM clinichospitaldb.employees WHERE
CONCAT(employees_id) LIKE '%" + valueToFind + "%'";
            MySqlDataAdapter adapter = new MySqlDataAdapter(searchQuery, connection);
            DataTable table = new DataTable();
            adapter.Fill(table);
            dataGridView1.DataSource = table;

        }

        private void textBox1_TextChanged_1(object sender, EventArgs e)
        {
            searchData(textBox1.Text);
        }
```

7.5 Logging in



User logging in with username and password admin just made.

7.6 Main Menu Page

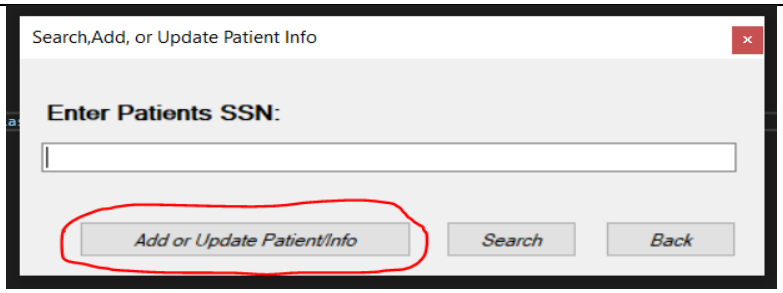


Here is the main menu page where the user can navigate to any section they need. Here is the code:

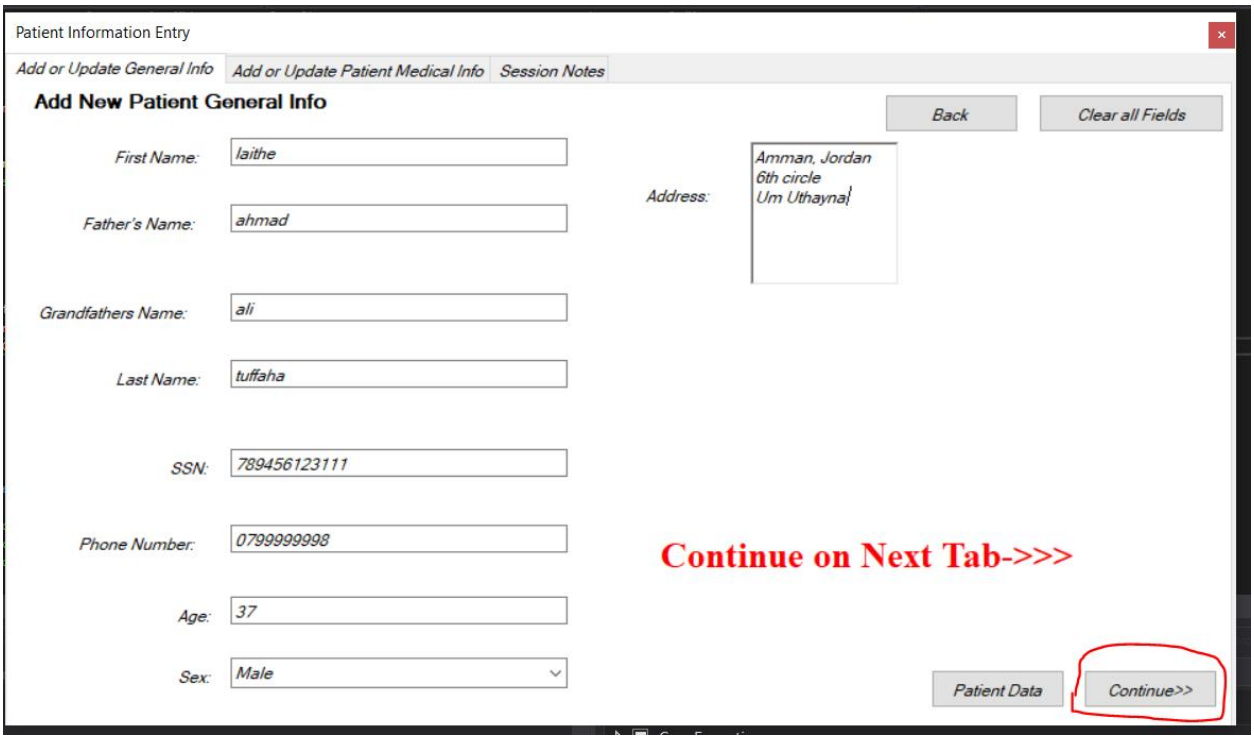
```
   private void Form1_Load(object sender, EventArgs e)
        {
            MySqlConnection connection = new
MySqlConnection("datasource=localhost;port=3306;username=root;password=123456789ls");
            MySqlDataReader mdr;
            var first_name = "";
            var last_name = "";

            string query = "SELECT fname, lname FROM clinichospitaldb.employees where
employees_id='" + Login.idlogin + "'";
            MySqlCommand command = new MySqlCommand(query, connection);
            connection.Open();
            mdr = command.ExecuteReader();
            // reads the data and fills the combo box and listbox

            if (mdr.HasRows)
            {
                while (mdr.Read())
                {

                    first_name = mdr["fname"].ToString();
                    last_name = mdr["lname"].ToString();
                    label1.Text = "WELCOME "+ first_name.ToUpper() + " " +
last_name.ToUpper()+"!";

                }
            }
```

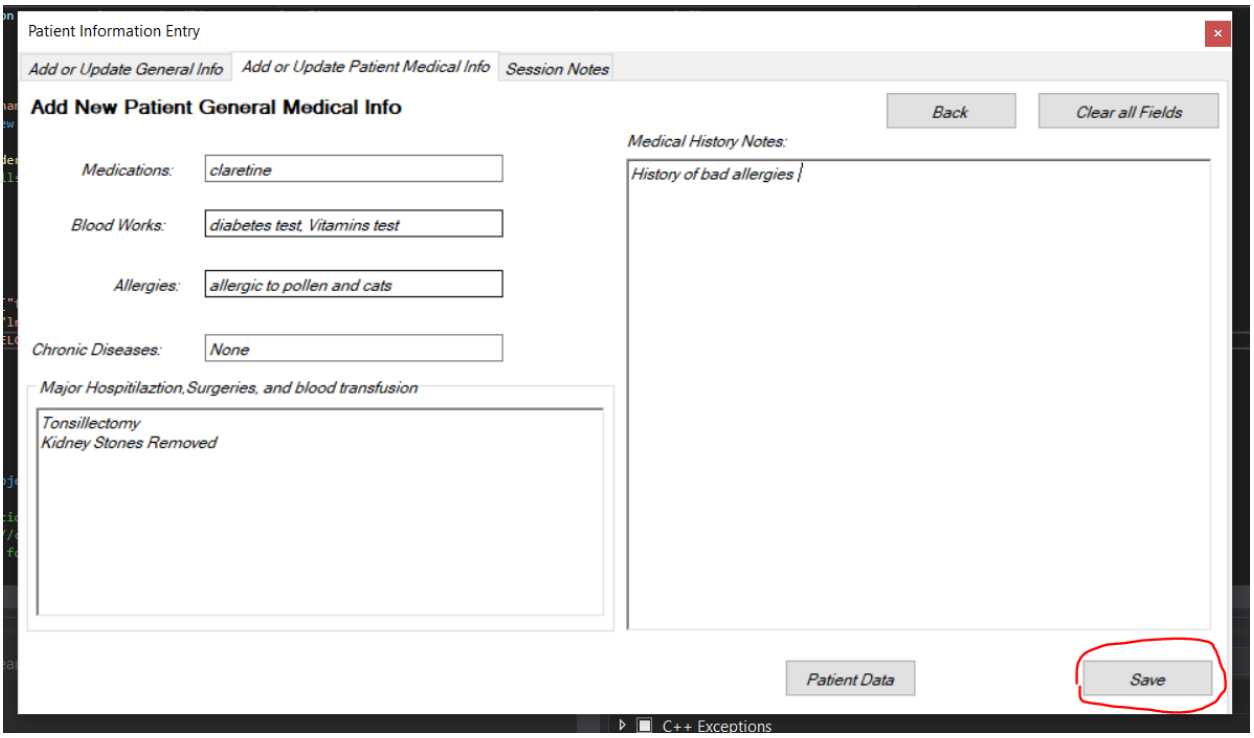
```
            connection.Close();
        }
```

7.7 Add or Search for Patients

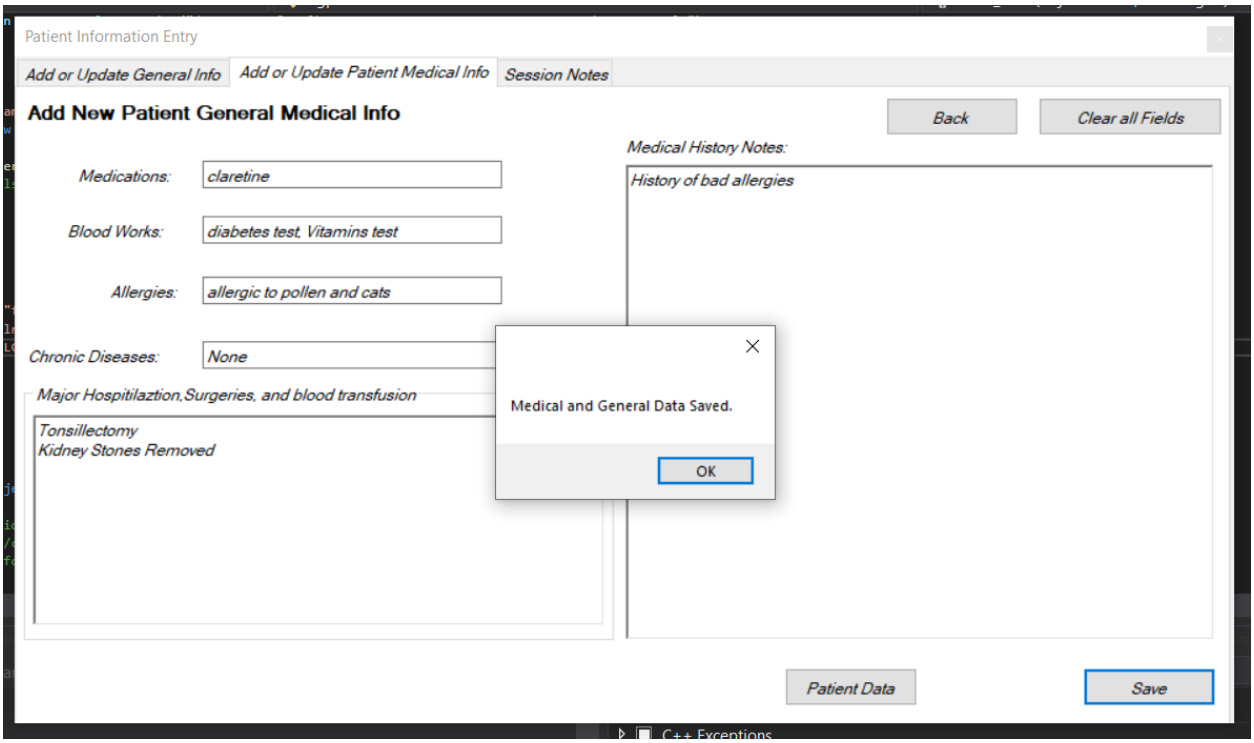


This is the patient info page that is displayed after clicking the patient's button on the main menu. After clicking the add or update patient/info button I have circled above the following page loads:





This is the second tab that is named add or update patient medical info. The data in this page will be sent to the generalmedicalinfopatient table. This page also consists of patient data button for loading patient's data for updating, back button, and clear all fields button. The user then clicks the save button.

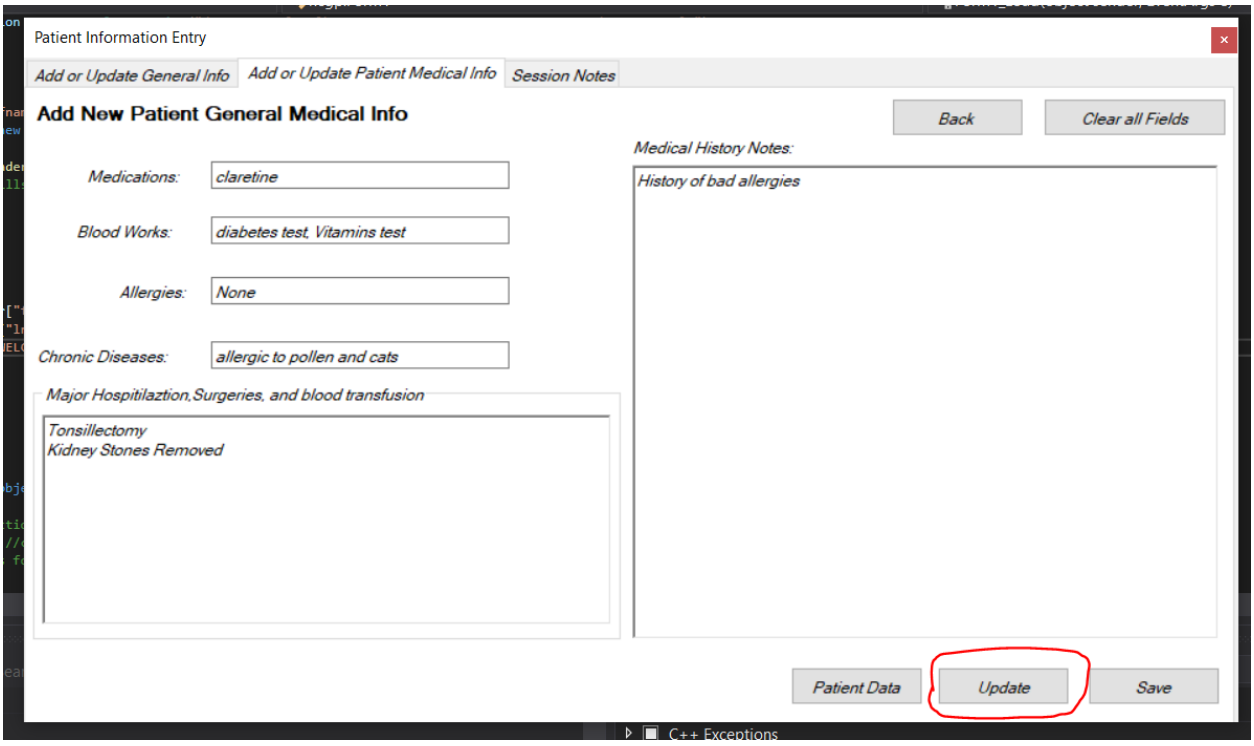


The data is then sent to the DB for both tabs general info and medical info using the following code:

```
string result = "Please Enter ";
            if (textBox1.Text == string.Empty)
            {
                result += "First Name,";
                label21.Visible = true;
            }
            if (textBox2.Text == string.Empty)
            {
                result += "Father's Name,";
                label22.Visible = true;
            }
            if (textBox3.Text == string.Empty)
            {
                result += "Grandfather's Name,";
                label23.Visible = true;
            }
            if (textBox4.Text == string.Empty)
            {
                result += "Last Name,";
                label24.Visible = true;
            }
            if (textBox5.Text == string.Empty)
            {
                result += "SSN,";
                label25.Visible = true;
            }
            if (textBox6.Text == string.Empty)
            {
                result += "Phone Number,";
                label26.Visible = true;
            }
            if (textBox7.Text == string.Empty)
            {
                result += "Age,";
                label27.Visible = true;
            }
            if (comboBox1.SelectedItem==null)
            {
                result += "Sex,";
                label28.Visible = true;
            }
            if (richTextBox1.Text == string.Empty)
            {
                result += "Address,";
                label29.Visible = true;
            }
            if (textBox9.Text == string.Empty)
            {
                result += "Medications,";
                label30.Visible = true;
            }
            if (textBox10.Text == string.Empty)
            {
                result += "Blood Works,";
                label31.Visible = true;
            }
```

```
            if (textBox11.Text == string.Empty)
            {
                result += "Allergies,";
                label32.Visible = true;
            }
            if (textBox12.Text == string.Empty)
            {
                result += "Chronic Diseases,";
                label33.Visible = true;
            }
            if (richTextBox3.Text == string.Empty)
            {
                result += "Major Hospitlization,";
                label34.Visible = true;
            }
            if (richTextBox2.Text == string.Empty)
            {
                result += "Medical History Notes,";
                label35.Visible = true;
            }

            if (result != "Please Enter ")
            {
                result = result.Remove(result.LastIndexOf(","));
                MessageBox.Show(result + ". If no data please specify as none.");
            }
            else

            {
                try
                {
                    string MyConnection1 = "datasource=localhost;port=3306;username=root;
password=123456789ls";
                    string Query = "insert into clinichospitaldb.patients(fname, fathersname,
gfname, lname, ssn, phonenum, age, sex, address)values" +
                  "('" + this.textBox1.Text + "','" + this.textBox2.Text + "','" +
this.textBox3.Text +
                  "','" + this.textBox4.Text + "'," + this.textBox5.Text + "," +
                  this.textBox6.Text + "," + this.textBox7.Text + ",'" +
this.comboBox1.SelectedItem + "','" + this.richTextBox1.Text + "');";


                    MySqlConnection MyConn1 = new MySqlConnection(MyConnection1);
                    MySqlCommand MyCommand1 = new MySqlCommand(Query, MyConn1);
                    MySqlDataReader MyReader1;
                    MyConn1.Open();
                    MyReader1 = MyCommand1.ExecuteReader();     // Here our query will be
executed and data saved into the database.
                    while (MyReader1.Read())
                    {
                        Console.WriteLine(String.Format("{0}", MyReader1[0]));
                    }
                    MyConn1.Close();
                    string MyConnection2 = "datasource=localhost;port=3306;username=root;
password=123456789ls";

                    string q2 = "insert into clinichospitaldb.generalmedicalinfopatient
(ssn_patients, medications,blood_works,chronicdis,allergies,majorhz,medicalhistory)values" +
```

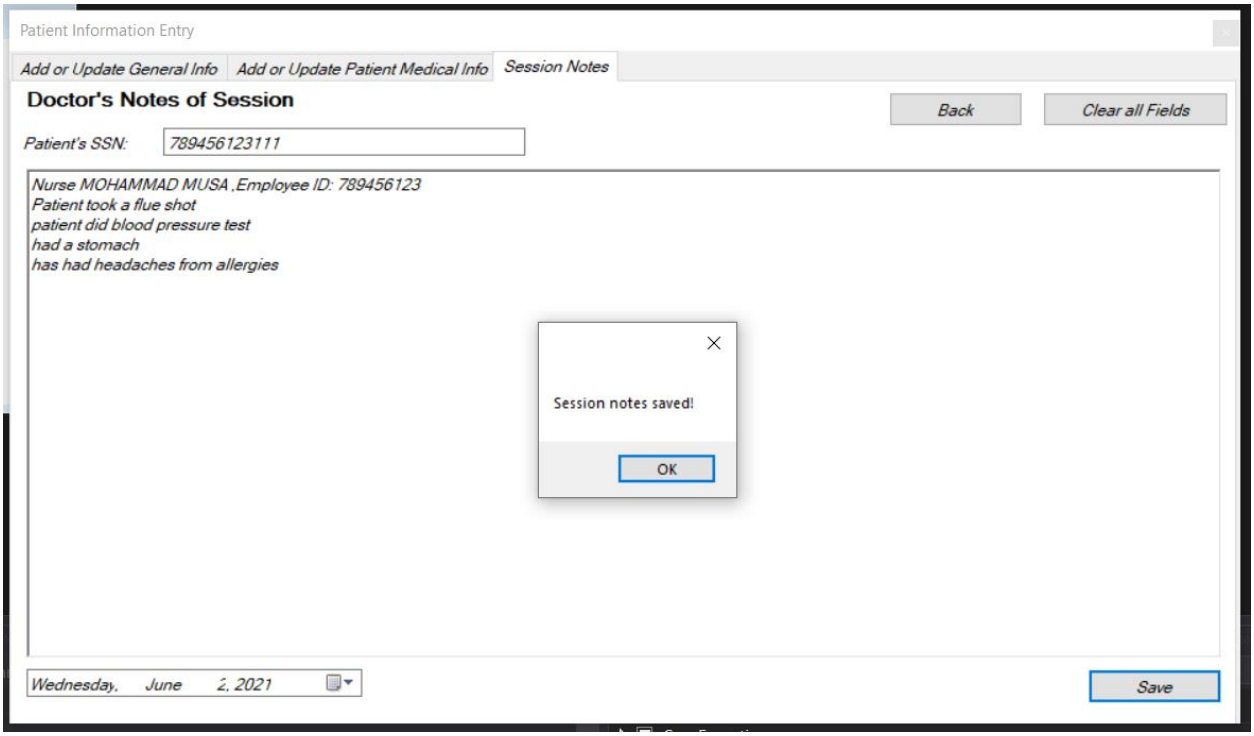
```
                    "('" + this.textBox5.Text + "','" + this.textBox9.Text + "','" +
this.textBox10.Text + "','" + this.textBox11.Text +
                    "','" + this.textBox12.Text + "','" + this.richTextBox3.Text + "','" +
this.richTextBox2.Text + "');";
                    MySqlConnection MyConn2 = new MySqlConnection(MyConnection2);
                    MySqlCommand MyCommand2 = new MySqlCommand(q2, MyConn2);
                    MySqlDataReader MyReader2;
                    MyConn2.Open();
                    MyReader2 = MyCommand2.ExecuteReader();     // Here our query will be
executed and data saved into the database.
                    MessageBox.Show("Medical and General Data Saved.");
                    while (MyReader2.Read())
                    {
                        Console.WriteLine(String.Format("{0}", MyReader2[0]));
                    }

                    MyConn2.Close();
                }
                catch (Exception)
                {

                    MessageBox.Show("SSN already exixts!");

                }
```



As you can see an update button appeared since there is an SSN entered in the SSN text box on the first page and the user clicked on patient data. So, now if the user needs to change any data for that particular patient, he can just change the data above in first two tabs and then click update, then the following code

will be executed with 2 update SQL statement two update generalmedicalinfopatient table and patients table:

```
try {
            //This is my connection string i have assigned the database file address path
            string connect = "datasource=localhost;port=3306;username=root;
password=123456789ls";
            //This is my update query in which i am taking input from the user through windows
forms and update the record.
            string q = "update clinichospitaldb.generalmedicalinfopatient set medications='" +
this.textBox9.Text  + "',blood_works='" + this.textBox10.Text + "',allergies='" +
this.textBox11.Text + "',majorhz='" + this.richTextBox3.Text + "',medicalhistory='" +
this.richTextBox2.Text + "', chronicdis = '" + this.textBox12.Text + "' where ssn_patients='" +
this.textBox5.Text + "';";
            //This is  MySqlConnection here i have created the object and pass my connection
string.
            MySqlConnection MyCon = new MySqlConnection(connect);
            MySqlCommand MyComm = new MySqlCommand(q, MyCon);
            MySqlDataReader MyRead;
            MyCon.Open();
            MyRead = MyComm.ExecuteReader();
            while (MyRead.Read())
            {
            }
                //This is my update query in which i am taking input from the user through
windows forms and update the record.
                string q1 = "update clinichospitaldb.patients set fname='" + this.textBox1.Text
+ "',fathersname= '" + this.textBox2.Text + "',gfname ='" + this.textBox3.Text +
                 "',lname='" + this.textBox4.Text + "',ssn='" + this.textBox5.Text +
"',phonenum='" +
                  this.textBox6.Text + "',age='" + this.textBox7.Text + "',sex='" +
this.comboBox1.SelectedItem + "',address='" + this.richTextBox1.Text +
                "' where ssn='" + this.textBox5.Text + "';";
                //This is  MySqlConnection here i have created the object and pass my connection
string.
                MySqlConnection MyCon1 = new MySqlConnection(connect);
                MySqlCommand MyComm1 = new MySqlCommand(q1, MyCon1);
                MySqlDataReader MyRead1;
                MyCon1.Open();
                MyRead1 = MyComm1.ExecuteReader();
                MessageBox.Show("Data Updated");
                while (MyRead1.Read())
                {
                }
                MyCon1.Close();
            } //Connection closed here
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
```



In the session notes, the textbox needs to have the patient's SSN entered in it for the current session. The first line in the rich textbox has the doctor's first name, last name, and id of the doctor that logged in. When the doctor clicks the save button the session notes are saved with the current sessions date. The first part of code I will show you is how the rich textbox prints the doctor logged in current info into the rich textbox:

```
MySqlConnection connection = new MySqlConnection("datasource=localhost;port=3306;username=root;password=123456789ls");
            MySqlDataReader mdr;
            var first_name = "";
            var last_name = "";
            var jobtit = "";
            string query = "SELECT jobtitle, fname, lname FROM clinichospitaldb.employees where employees_id='" + Login.idlogin + "'";
            MySqlCommand command = new MySqlCommand(query, connection);
            connection.Open();
            mdr = command.ExecuteReader();
            // reads the data and fills the combo box and listbox

            if (mdr.HasRows)
            {
                while (mdr.Read())
                {
                    jobtit = mdr["jobtitle"].ToString();
                    first_name = mdr["fname"].ToString();
                    last_name = mdr["lname"].ToString();
                    richTextBox4.AppendText(jobtit+ " " + first_name.ToUpper() + " " + last_name.ToUpper()+" ,Employee ID: "+Login.idlogin+"\r\n");

                }
            }
            connection.Close();
```

This second part of the code is when the user clicks the save button:

```
  try
            {
                string result = "Please Enter ";
                if (textBox13.Text == string.Empty)
                {
                    result += "Patient's SSN or ";
                    label36.Visible = true;
                }
                if (richTextBox4.Text == string.Empty)
                {
                    result += "Session Notes or";

                }
                if (result != "Please Enter ")
                {
                    result = result.Remove(result.LastIndexOf(" or"));
                    MessageBox.Show(result + '.');
                }
                else
                {
                    string MyConnection1 = "datasource=localhost;port=3306;username=root;
password=123456789ls";
                    string dt = dateTimePicker1.Value.ToString("MM/dd/yyyy hh:mm:ss");
                    string Queryy = "insert into clinichospitaldb.session_notes(ssn,
sessionnotes,dateandtime)values" +
                        "('" + textBox13.Text + "','" + this.richTextBox4.Text + "','" + dt +
"');";

                    //Console.WriteLine(Query);
                    MySqlConnection MyConn1 = new MySqlConnection(MyConnection1);
                    MySqlCommand MyCommand1 = new MySqlCommand(Queryy, MyConn1);
                    MySqlDataReader MyReader1;
                    MyConn1.Open();
                    MyReader1 = MyCommand1.ExecuteReader();     // Here our query will be
executed and data saved into the database.
                    MessageBox.Show("Session notes saved!");
                    while (MyReader1.Read())
                    {
                        Console.WriteLine(String.Format("{0}", MyReader1[0]));
                    }
                    MyConn1.Close();
                    textBox13.Clear();
                    richTextBox4.Clear();
                }
            }
            catch (Exception)
            {
                MessageBox.Show("Patient not found.");
            }
```

Here are pictures of the data in the tables:

1. patients table

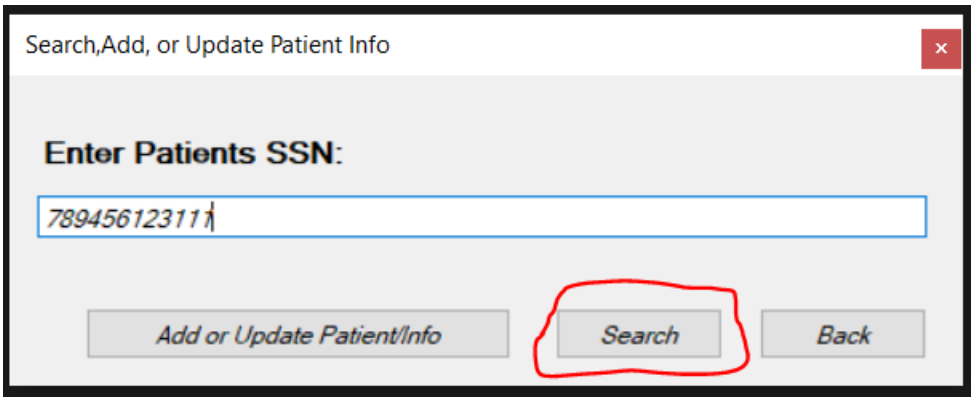
laithe	ahmad	ali	tuffaha	789456123...	799999998	37	Male	Amman, Jorda...

2. generalmedicalinfopatient table

789456123111	claretine	diabetes test, Vitamins test	None	Tonsillectomy	Kidney S...	History of bad allergies	allergic to pollen and cats

3. session_notes table

	ssn	sessionnotes	dateandtime
▶	789456123111	Nurse MOHAMMAD MUSA ,Employee ID: 78945...	06/02/2021 06:57:27

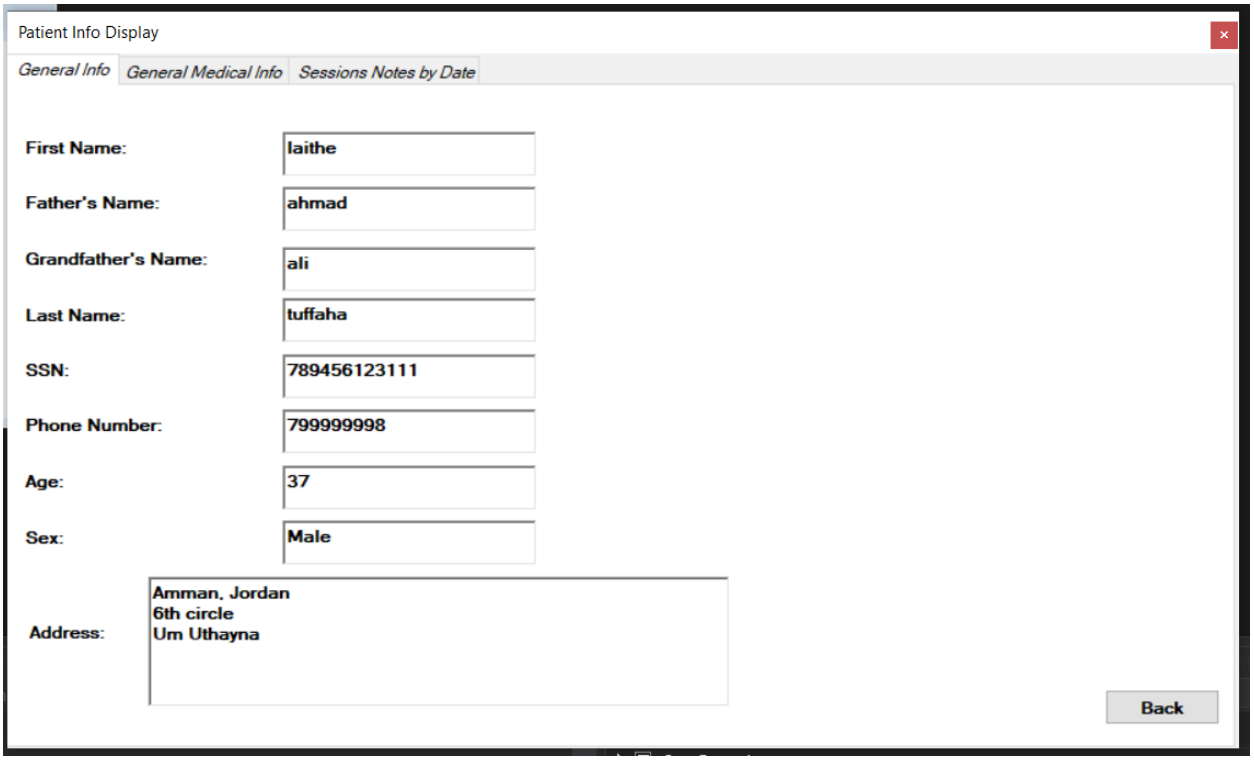


Now I will show the code for the search button circled in red and what happens when the patient that was just entered to the database will look like.

```
13 references
public partial class Form5 : Form
{

    public static string ssn;
```

```
            ssn = textBox1.Text;

            this.Hide(); // hide function hides this form. always use hide instead of close
function when intending on going back to this form.
            Form10 f10 = new Form10();//creating instance of class Form7
                        f10.ShowDialog();//shows form 7 with show dialogue function
```

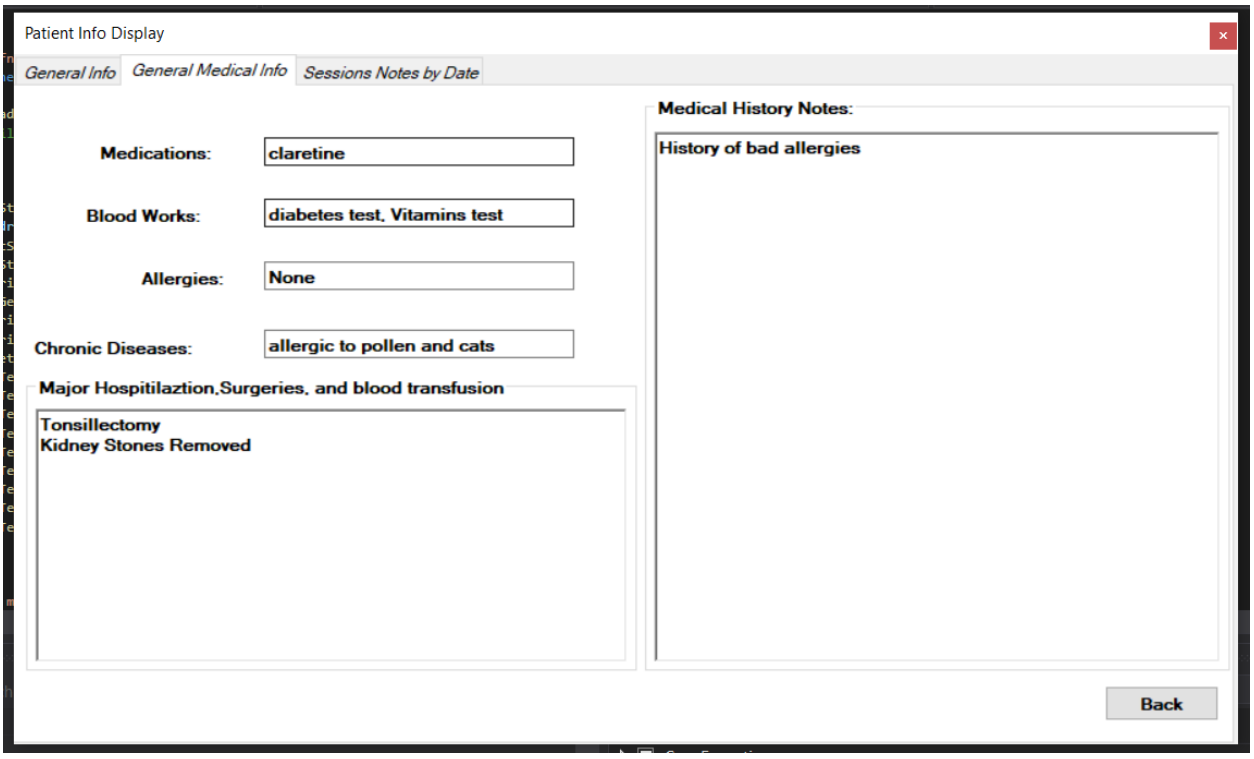
A global variable is declared at the top of the code and the SSN entered into the text box is stored in this global variable so it can be accessed in the next page, and this window is hidden, and the next form is opened.



As you can see the form is split into 3 tabs that only allow the user to view data entered. Using a select statement the data is accessed using a select statement and the SSN passed from the global variable made in the previous form. Code:

```
MySqlDataReader mdr;
                string query = "Select fname,fathersname,gfname,lname,ssn,phonenum,
age,sex,address from  clinichospitaldb.patients where ssn = " + Form5.ssn;
                MySqlCommand command = new MySqlCommand(query, connection);
                connection.Open();
                mdr = command.ExecuteReader();
                // reads the data and fills the combo box and listbox

                while (mdr.Read())
                {
                    var fname = mdr.GetString(0);
                    var fathersname = mdr.GetString(1);
                    var gfname = mdr.GetString(2);
                    var lname = mdr.GetString(3);
                    var ssn = mdr.GetString(4);
                    var phonenum = mdr.GetString(5);
                    var age = mdr.GetString(6);
                    var sex = mdr.GetString(7);
                    var address = mdr.GetString(8);
                    richTextBox1.AppendText(fname);
                    richTextBox2.AppendText(fathersname);
                    richTextBox3.AppendText(gfname);
                    richTextBox4.AppendText(lname);
                    richTextBox5.AppendText(ssn);
                    richTextBox6.AppendText(phonenum);
                    richTextBox7.AppendText(age);
                    richTextBox8.AppendText(sex);
```

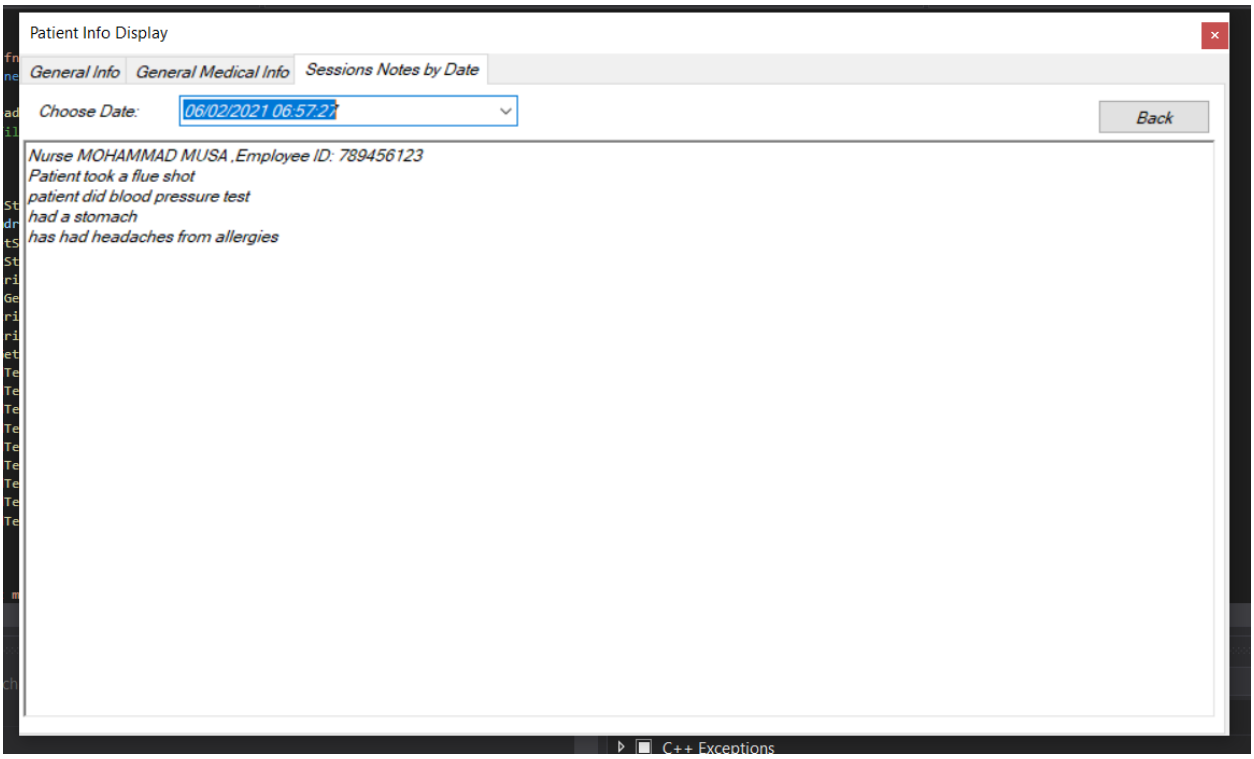
```
            richTextBox9.AppendText(address);
        }
        connection.Close();
```



The same goes here just a different select statement accessing from the generalmedicalinfopatient table. Here is the code for this segment:

```
            string query2 = "Select
medications,blood_works,allergies,majorhz,medicalhistory,chronicdis from
clinichospitaldb.generalmedicalinfopatient where ssn_patients = " + Form5.ssn;
            MySqlCommand command2 = new MySqlCommand(query2, connection);
            MySqlDataReader mdr2;
            connection.Open();
            mdr2 = command2.ExecuteReader();
            // reads the data and fills the combo box and listbox
            while (mdr2.Read())
            {
                //var ssn_patients = mdr2.GetString(0);
                var medications = mdr2.GetString(0);
                var blood_works = mdr2.GetString(1);
                var allergies = mdr2.GetString(2);
```

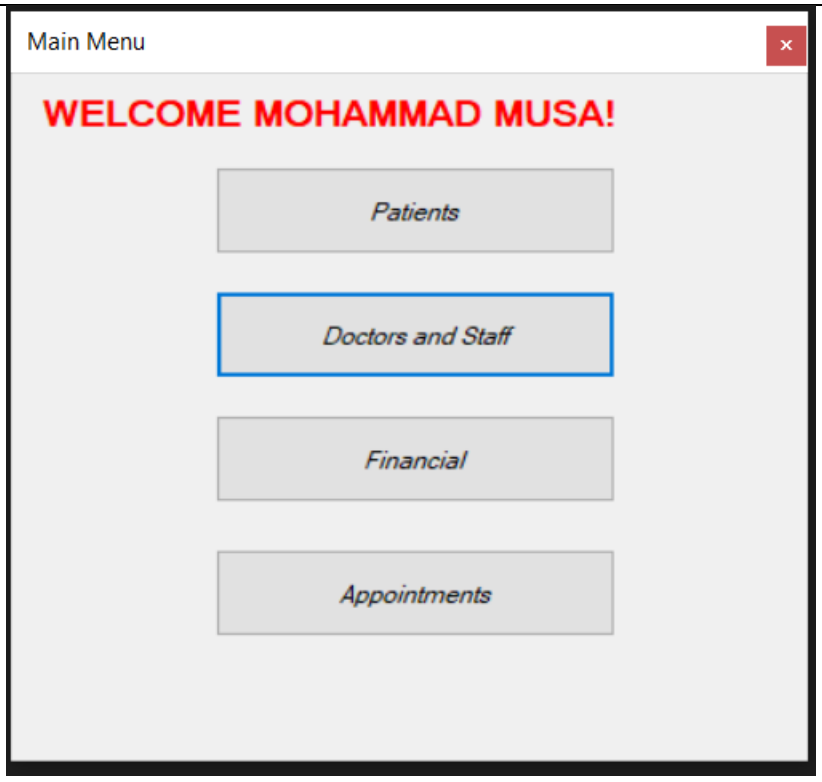
```
                var majorhz = mdr2.GetString(3);
                var medicalhistory = mdr2.GetString(4);
                var chronicdis = mdr2.GetString(5);

                textBox9.AppendText(medications);
                textBox10.AppendText(blood_works);
                textBox11.AppendText(allergies);
                richTextBox11.AppendText(majorhz);
                richTextBox10.AppendText(medicalhistory);
                textBox12.AppendText(chronicdis);
            }
            connection.Close();
            string selectQuery4 = "SELECT * FROM clinichospitaldb.session_notes where ssn =
" + Form5.ssn;
            connection.Open();
            MySqlCommand command4 = new MySqlCommand(selectQuery4, connection);
            MySqlDataReader reader4 = command4.ExecuteReader();
            while (reader4.Read())
            {

                comboBox1.Items.Add(reader4.GetString("dateandtime"));
            }
            connection.Close();
        }
```



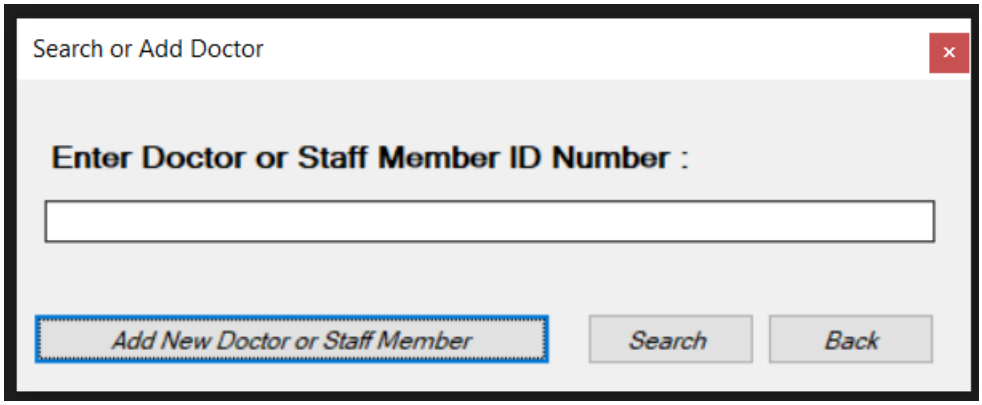
Here nothing is displayed until the user chooses a date to access session notes for the patient. Therefore, the SSN from the search textbox page is used and then the rich textbox will be filled with notes according to which date the user selected in the combo box. Here is the code for this segment:

```
string selected = comboBox1.Text;
            string query3 = "SELECT ssn, dateandtime,sessionnotes FROM
clinichospitaldb.session_notes where dateandtime='" + selected + "';";
            MySqlCommand command3 = new MySqlCommand(query3, connection);
            MySqlDataReader mdr3;
            connection.Open();
            mdr3 = command3.ExecuteReader();
            while (mdr3.Read())
            {

                var dateandtime = mdr3.GetString(1);
                var sessionnotes = mdr3.GetString(2);
                richTextBox12.Text=sessionnotes;

            }
            connection.Close();
```

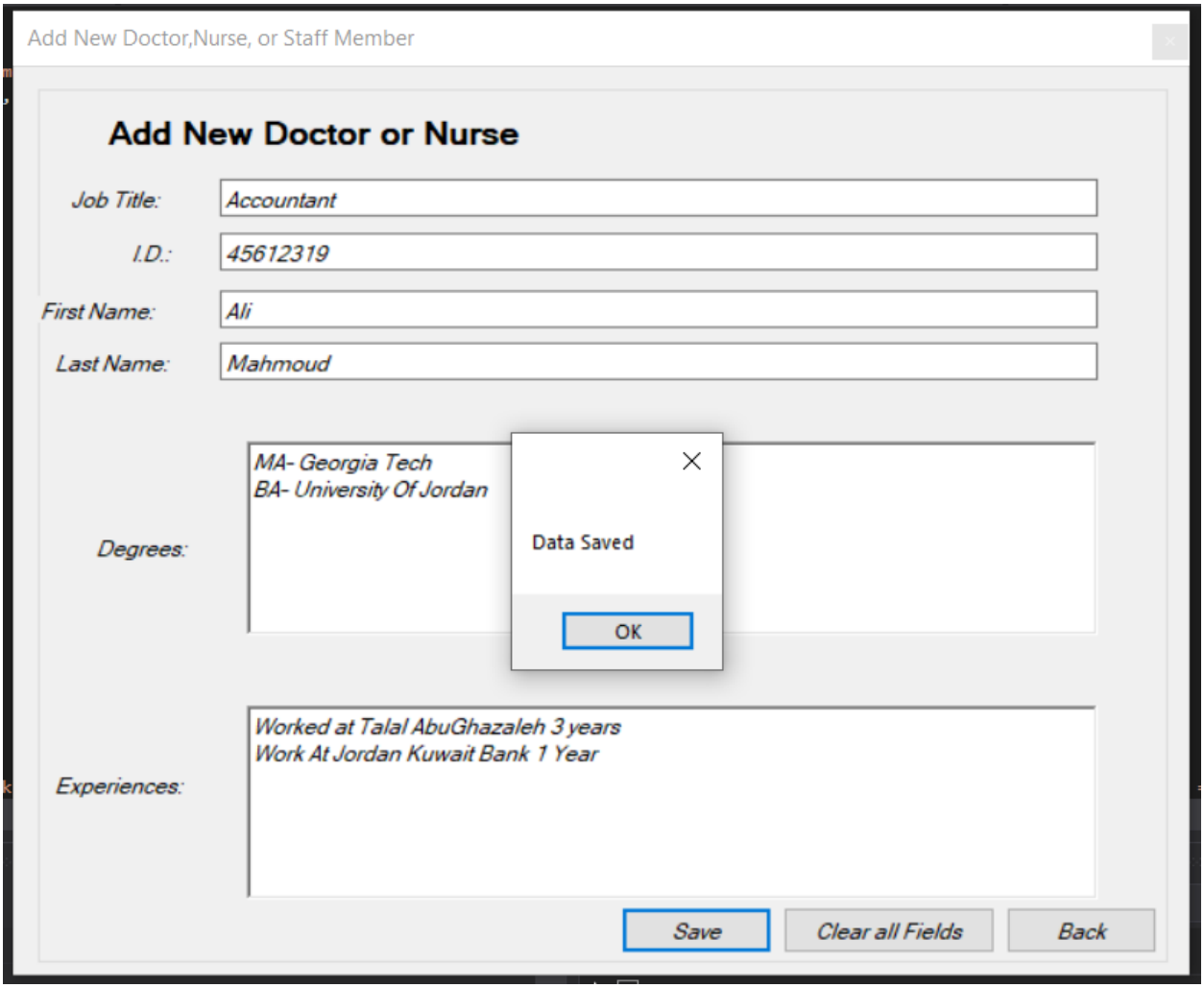
7.8 Doctors and Staff



Now back to the main menu for the second button Doctors and staff.



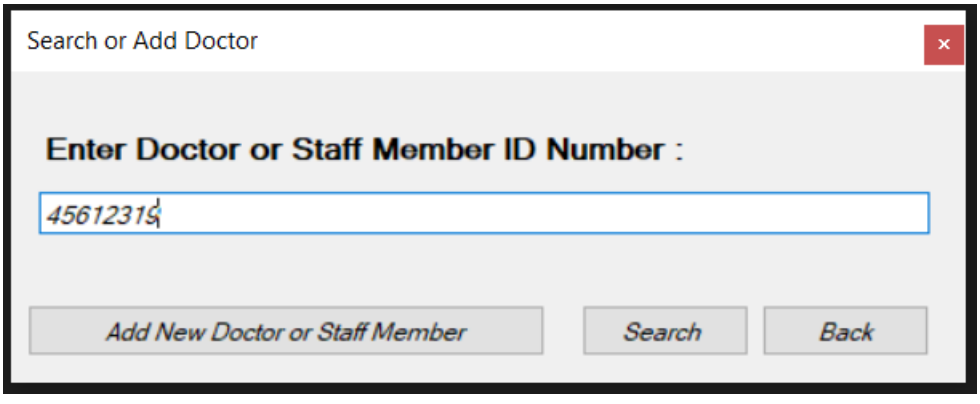
Once clicked the same design as the patient's page is opened with the same functionality. Either add a doctor or staff member or search.



Here you can see the user created a new staff member successfully and prompted with data saved. This is done with an insert statement inserting this data into the employees table. This page also consists of a back button and clear all fields button to clear all textboxes and rich textboxes. Here is the code for the save button:

```
   string result = "Please Enter ";
            if (textBox7.Text == string.Empty)
            {
                result += "Job Title,";
                label6.Visible = true;
            }
            if (textBox1.Text == string.Empty)
            {
                result += "ID,";
                label7.Visible = true;
            }
            if (textBox2.Text == string.Empty)
            {
                result += "Grandfather's Name,";
                label8.Visible = true;
            }
            if (textBox6.Text == string.Empty)
            {
                result += "Last Name,";
                label9.Visible = true;
            }
            if (richTextBox1.Text == string.Empty)
            {
                result += "Date,";
                label12.Visible = true;
            }
            if (richTextBox2.Text == string.Empty)
            {
                result += "Time,";
                label13.Visible = true;
            }
            if (result != "Please Enter ")
            {
                result = result.Remove(result.LastIndexOf(","));
                MessageBox.Show(result + '.'+" If nothing to be filled in degrees or expriences
please enter none.");
            }
            else
            {
                try
                {

                    string Query = "insert into clinichospitaldb.employees
(employees_id,fname,lname,degrees,experience,jobtitle)values" +
                        "('" + this.textBox1.Text + "','" + this.textBox2.Text + "','" +
this.textBox6.Text + "','" + this.richTextBox1.Text + "','" + this.richTextBox2.Text + "','" +
textBox7.Text + "');";
                    MySqlConnection MyConn = new MySqlConnection(MyConnection);
                    MySqlCommand MyCommand = new MySqlCommand(Query, MyConn);
                    MySqlDataReader MyReader;
                    MyConn.Open();
                    MyReader = MyCommand.ExecuteReader();     // Here our query will be executed
and data saved into the database.
                    MessageBox.Show("Data Saved");
                    while (MyReader.Read())
                    {
                        Console.WriteLine(String.Format("{0}", MyReader[0]));
                    }
```

```
            MyConn.Close();
        }
        catch (Exception)
        {
            MessageBox.Show("ID already exists!");
        }
        label6.Visible = false;
        label7.Visible = false;
        label8.Visible = false;
        label9.Visible = false;
        label12.Visible = false;
        label13.Visible = false;

    }
```

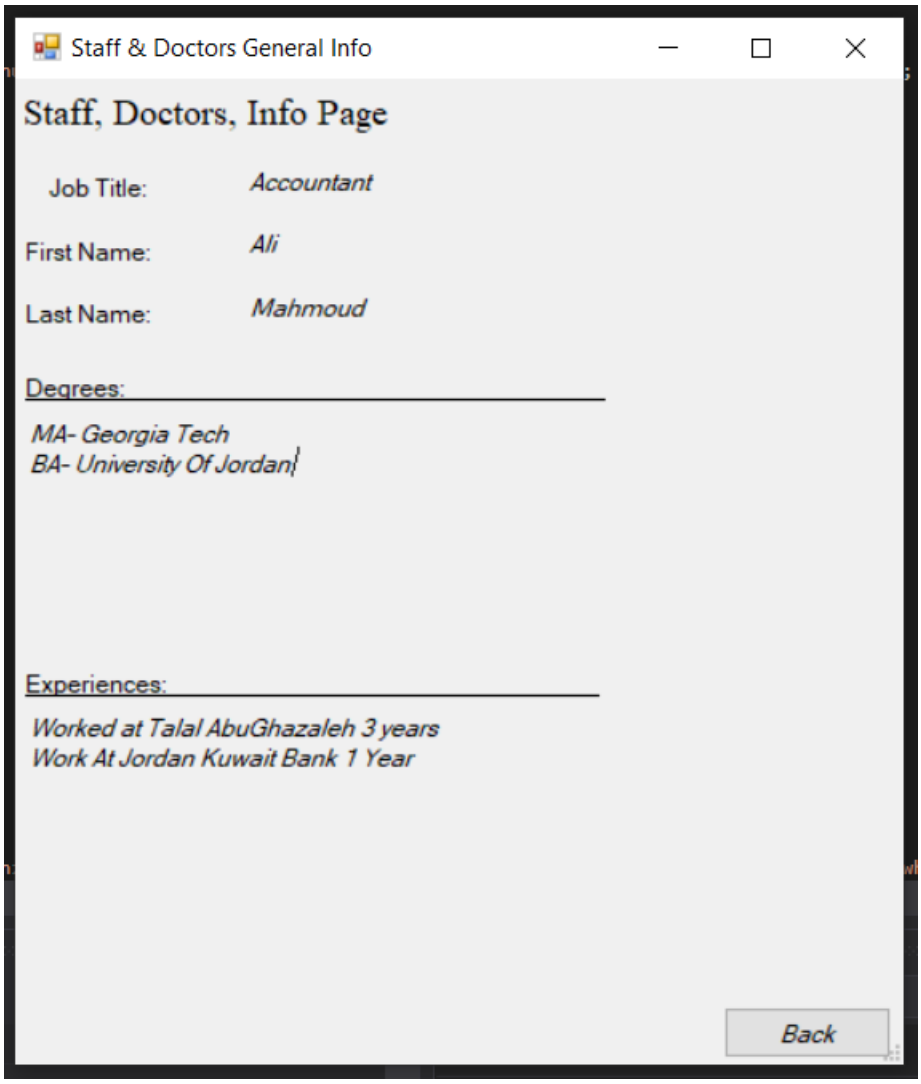


Now back to search or add doctor window, the user here wants to search the employee he has just created. Here's samples of the code to show how a global variable is declared so the id can be passed:

```
6 references
public partial class Form6 : Form
{
    public static string id;
```

```
id = textBox1.Text;
```

```
                this.Hide(); // hide function hides this form. always use hide instead of close
function when intending on going back to this form.
                Form12 f12 = new Form12();//creating instance of class Form7
                f12.ShowDialog();//shows form 7 with show dialogue function
```

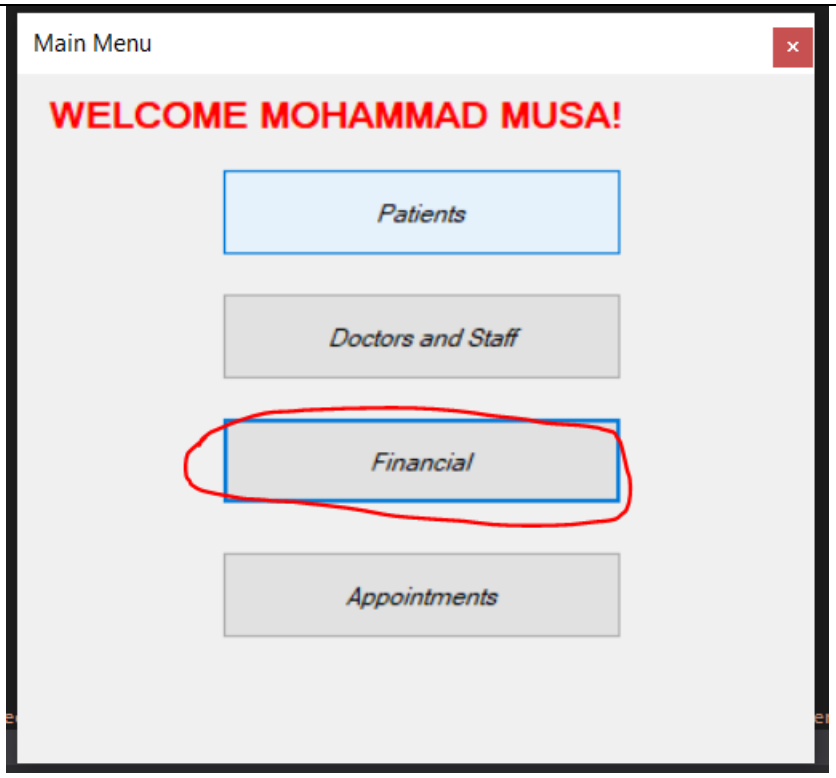


The data is then fetched using a select statement and displayed on this page:

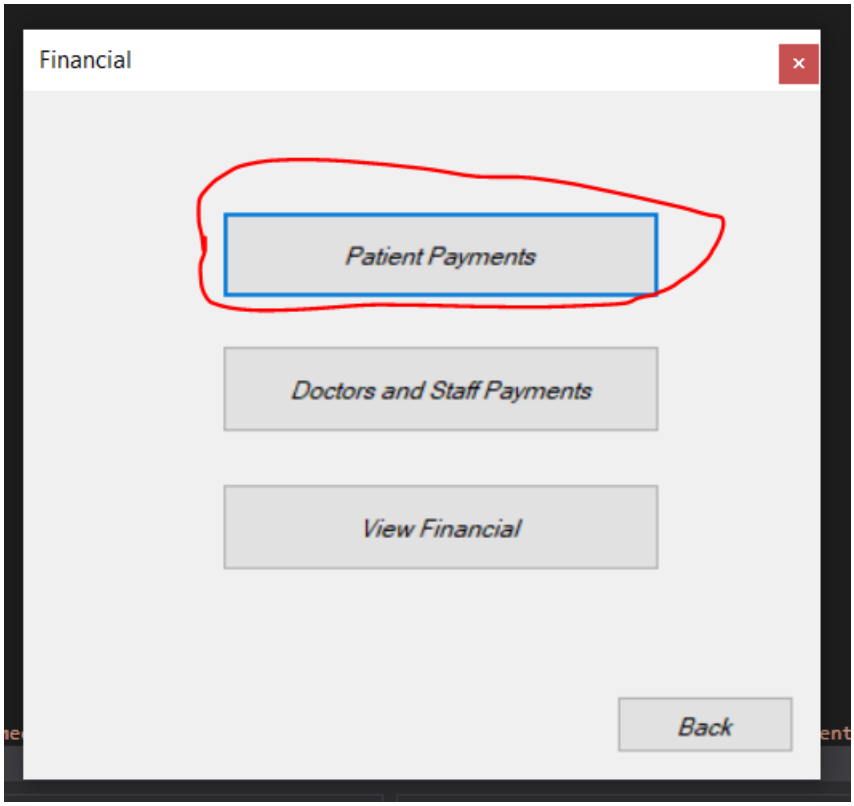
```
MySqlConnection connection = new
MySqlConnection("datasource=localhost;port=3306;username=root;password=1234567891s");

        public Form12()
        {
            InitializeComponent();
            this.CenterToScreen(); //centers form in middle of screen
        }

        private void Form12_Load(object sender, EventArgs e)
        {       connection.Open();

            //if (Form6.chosen == "doctors_nurses")
            //{
```

```
                MySqlDataReader mdr;
                string query = "Select employees_id,fname,lname,degrees, experience,jobtitle
from clinichospitaldb.employees where employees_id = " + Form6.id;
                MySqlCommand command = new MySqlCommand(query, connection);

                mdr = command.ExecuteReader();
                // reads the data and fills the combo box and listbox

                while (mdr.Read())
                {
                    var fname = mdr.GetString(1);
                    var lname = mdr.GetString(2);
                    var degrees = mdr.GetString(3);
                    var experience = mdr.GetString(4);
                    var jobtitle = mdr.GetString(5);

                    textBox1.AppendText(jobtitle);
                    textBox2.AppendText(fname);
                    textBox3.AppendText(lname);
                    richTextBox1.AppendText(degrees);
                    richTextBox2.AppendText(experience);

                }
```

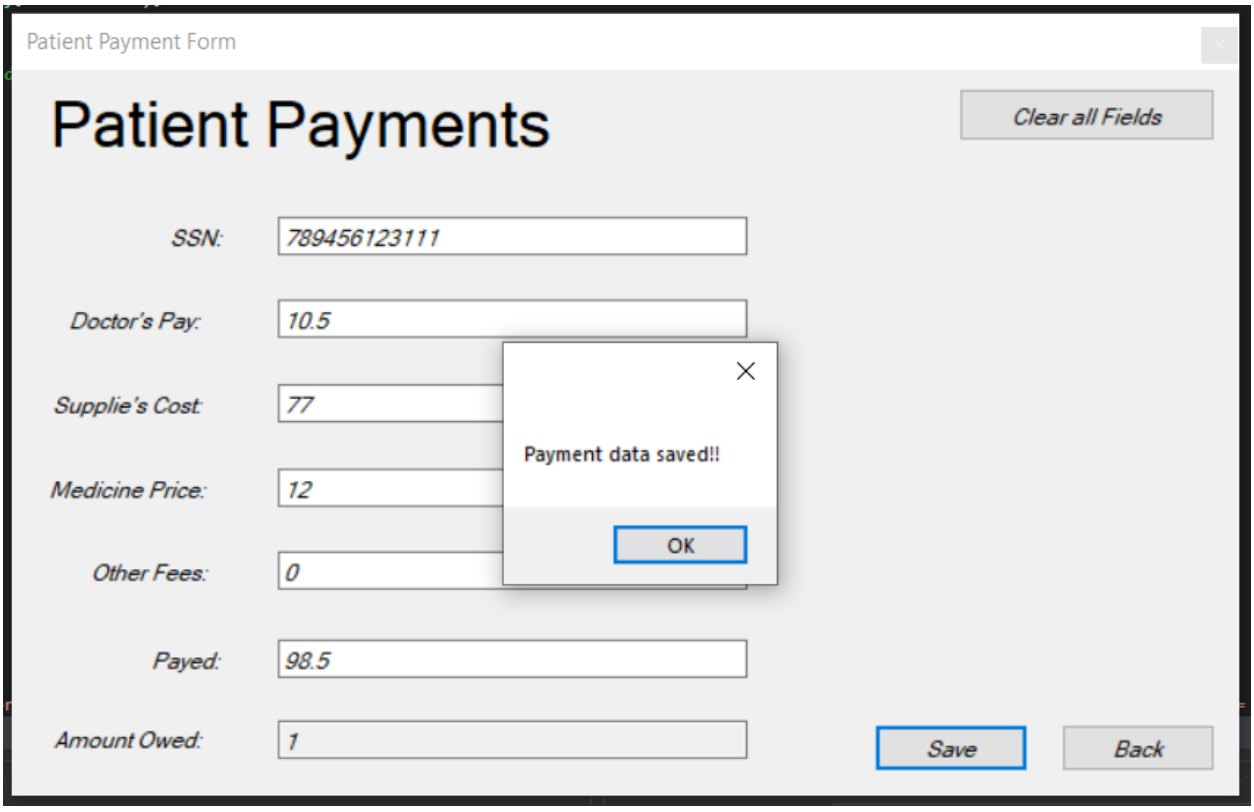
7.9 Financial



Now back to the main menu page to click on the button circled in red to get to the financial section of the IS.



This is the financial section, first, let us see the button I have circled in red and go to the patient payments page.



Here you can see the SSN of the patient that was previously made is entered in the first textbox. These textboxes have two forms of validation, first it should not be empty, and the second form of validation is that only numbers should be entered in text boxes. A formula is coded in to enter how much the patient paid and how much he owes. The amount owed is displayed in last textbox when save button is clicked. An insert statement sends all this data to the patientpayments table. Here is the code for this page:

```
private void textBox1_TextChanged(object sender, EventArgs e)
     {
         if (System.Text.RegularExpressions.Regex.IsMatch(textBox1.Text, "[^0-9]"))
         {
             MessageBox.Show("Please enter only numbers.");
             textBox1.Text = textBox1.Text.Remove(textBox1.Text.Length - 1);
         }


     }

     private void button2_Click(object sender, EventArgs e)
     {
         string result = "Please Enter ";
         if (textBox1.Text == string.Empty)
         {
             result += "SSN,";
             label10.Visible = true;
         }
         if (textBox2.Text == string.Empty)
         {
             result += "Doctor's Pay,";
             label9.Visible = true;
         }
         if (textBox3.Text == string.Empty)
         {
             result += "Supplies Cost,";
             label11.Visible = true;
         }
         if (textBox4.Text == string.Empty)
         {
             result += "Medicine Price,";
             label12.Visible = true;
         }
         if (textBox5.Text == string.Empty)
         {
             result += "Other Fees,";
             label13.Visible = true;
         }
         if (textBox6.Text == string.Empty)
         {
             result += "Payed,";
             label14.Visible = true;
         }
         if (result != "Please Enter ")
         {
             result = result.Remove(result.LastIndexOf(","));
             MessageBox.Show(result + '.');
         }
```

```
            else
            {

                try
                {
                    double owed = (Convert.ToDouble(this.textBox2.Text) +
Convert.ToDouble(this.textBox3.Text) + Convert.ToDouble(this.textBox4.Text) +
Convert.ToDouble(this.textBox5.Text)) - Convert.ToDouble(this.textBox6.Text);

                    string MyConnection1 = "datasource=localhost;port=3306;username=root;
password=123456789ls";
                    string query = "insert into clinichospitaldb.patientpayments
(patientssn,doctorspayamount,suppliesused,medicinepaymountamount,Other,amountpayed,amountowed,da
te)values('" + this.textBox1.Text + "', '" + this.textBox2.Text + "', '" + this.textBox3.Text +
"','" + this.textBox4.Text + "','" + this.textBox5.Text + "','" +
                    this.textBox6.Text + "','" + owed + "','" + DateTime.Now.ToString("yyyy-MM-
dd")/*date*/ + "');";

                    MySqlConnection MyConn1 = new MySqlConnection(MyConnection1);
                    MySqlCommand MyCommand1 = new MySqlCommand(query, MyConn1);
                    MySqlDataReader MyReader1;
                    MyConn1.Open();
                    MyReader1 = MyCommand1.ExecuteReader();     // Here our query will be
executed and data saved into the database.

                    while (MyReader1.Read())
                    {
                        Console.WriteLine(String.Format("{0}", MyReader1[0]));
                    }
                    textBox7.Text = Convert.ToString(owed);
                    MessageBox.Show("Payment data saved!!");
                    MyConn1.Close();


                    this.Hide(); // hide function hides this form. always use hide instead of
close function when intending on going back to this form.
                    Form9 parent = new Form9(); //creating instance of class Form1
                    parent.ShowDialog();// shows form 1 which is parent form with show dialogue
function
                }
                catch (Exception)
                {
                    MessageBox.Show("SSN not Found.");
                }

            }
        }

        private void label7_Click(object sender, EventArgs e)
        {

        }
        private void button10_Click(object sender, EventArgs e)
        {
            Action<Control.ControlCollection> func = null;

            func = (controls) =>
            {
```

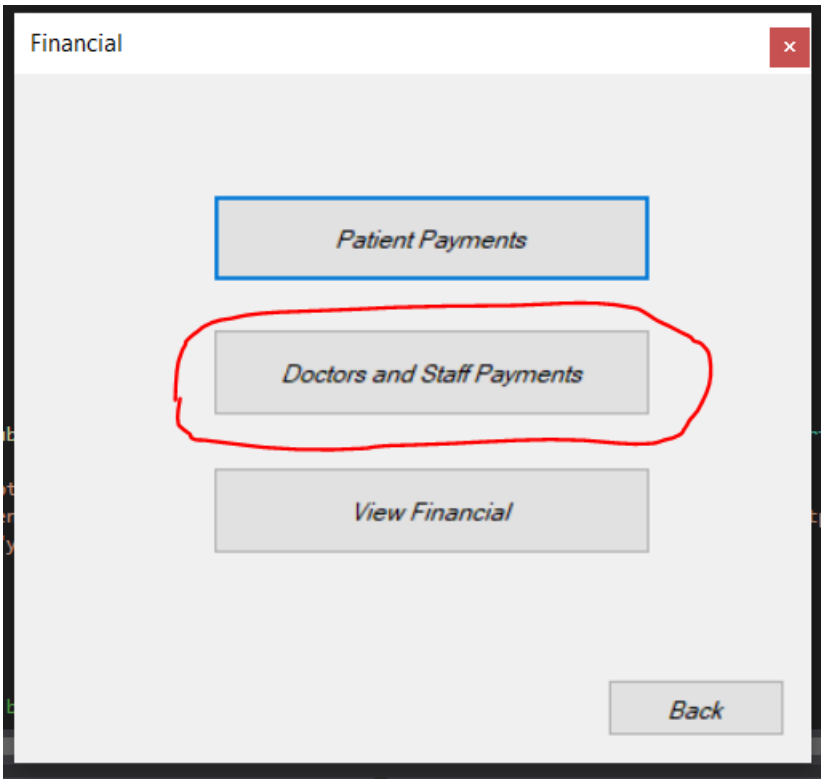
```
            foreach (Control control in controls)
                if (control is TextBox)
                    (control as TextBox).Clear();
                else
                    func(control.Controls);
        };

        func(Controls);
        label10.Visible = false;
        label9.Visible = false;
        label12.Visible = false;
        label13.Visible = false;
        label14.Visible = false;
        label11.Visible = false;

    }

    private void textBox2_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (e.KeyChar == '.')
        {
            if (this.Text.IndexOf(".") >= 0 || this.Text.Length == 0)
            {
                e.Handled = true;
            }
        }
        else if ((e.KeyChar < '0' || e.KeyChar > '9') && e.KeyChar != '\b')
        {
            e.Handled = true;
        }
    }

    private void textBox2_Validating(object sender, CancelEventArgs e)
    {
        float value;
        if (!float.TryParse(textBox2.Text, out value)) { textBox2.Text = ""; }
    }

    private void textBox3_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (e.KeyChar == '.')
        {
            if (this.Text.IndexOf(".") >= 0 || this.Text.Length == 0)
            {
                e.Handled = true;
            }
        }
        else if ((e.KeyChar < '0' || e.KeyChar > '9') && e.KeyChar != '\b')
        {
            e.Handled = true;
        }
    }

    private void textBox3_Validating(object sender, CancelEventArgs e)
    {
        float value;
        if (!float.TryParse(textBox3.Text, out value)) { textBox3.Text = ""; }
    }
```

```
private void textBox4_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == '.')
    {
        if (this.Text.IndexOf(".") >= 0 || this.Text.Length == 0)
        {
            e.Handled = true;
        }
    }
    else if ((e.KeyChar < '0' || e.KeyChar > '9') && e.KeyChar != '\b')
    {
        e.Handled = true;
    }
}

private void textBox4_Validating(object sender, CancelEventArgs e)
{
    float value;
    if (!float.TryParse(textBox4.Text, out value)) { textBox4.Text = ""; }
}

private void textBox5_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == '.')
    {
        if (this.Text.IndexOf(".") >= 0 || this.Text.Length == 0)
        {
            e.Handled = true;
        }
    }
    else if ((e.KeyChar < '0' || e.KeyChar > '9') && e.KeyChar != '\b')
    {
        e.Handled = true;
    }
}

private void textBox5_Validating(object sender, CancelEventArgs e)
{
    float value;
    if (!float.TryParse(textBox5.Text, out value)) { textBox5.Text = ""; }
}

private void textBox6_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == '.')
    {
        if (this.Text.IndexOf(".") >= 0 || this.Text.Length == 0)
        {
            e.Handled = true;
        }
    }
    else if ((e.KeyChar < '0' || e.KeyChar > '9') && e.KeyChar != '\b')
    {
        e.Handled = true;
    }
}
```

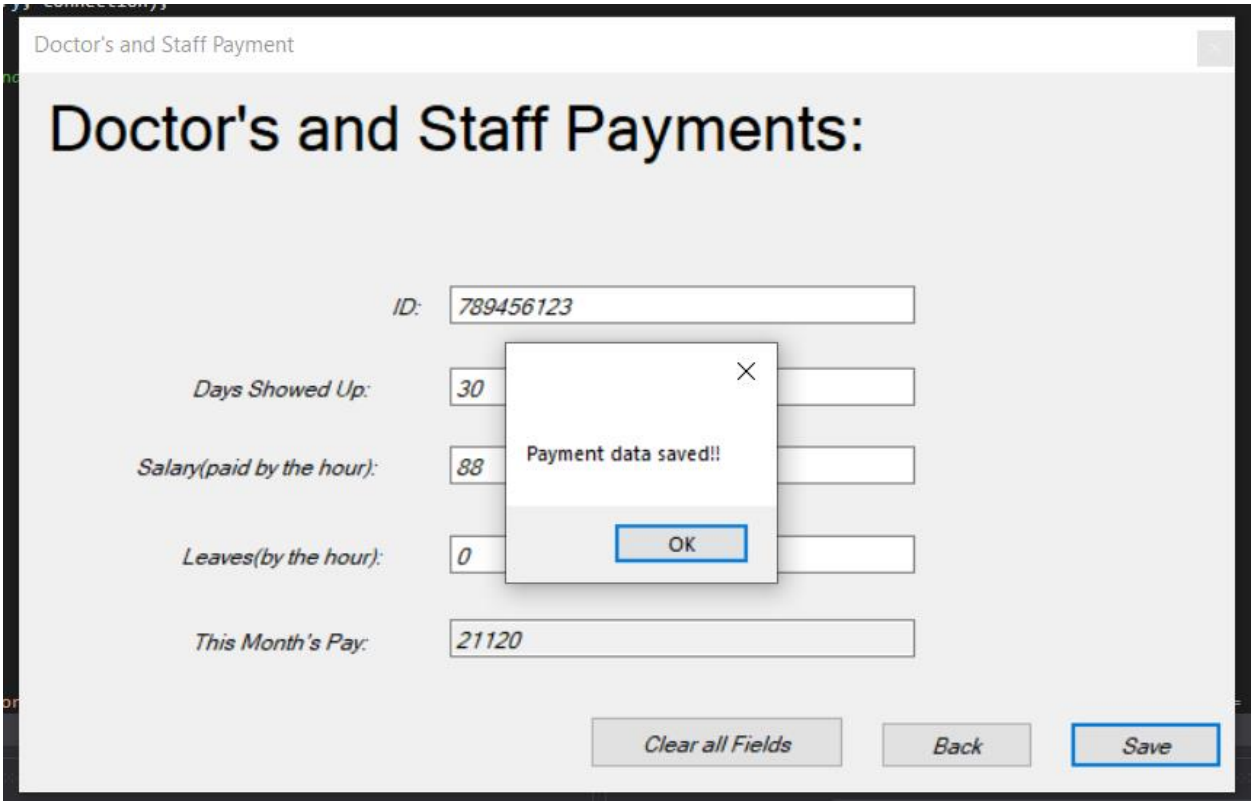
```
        private void textBox6_Validating(object sender, CancelEventArgs e)
        {
            float value;
            if (!float.TryParse(textBox6.Text, out value)) { textBox6.Text = ""; }
        }
    }
}
```

Here is how the data looks like in the table:

11	12	789456123111	77	99	1	0	2021-06-02

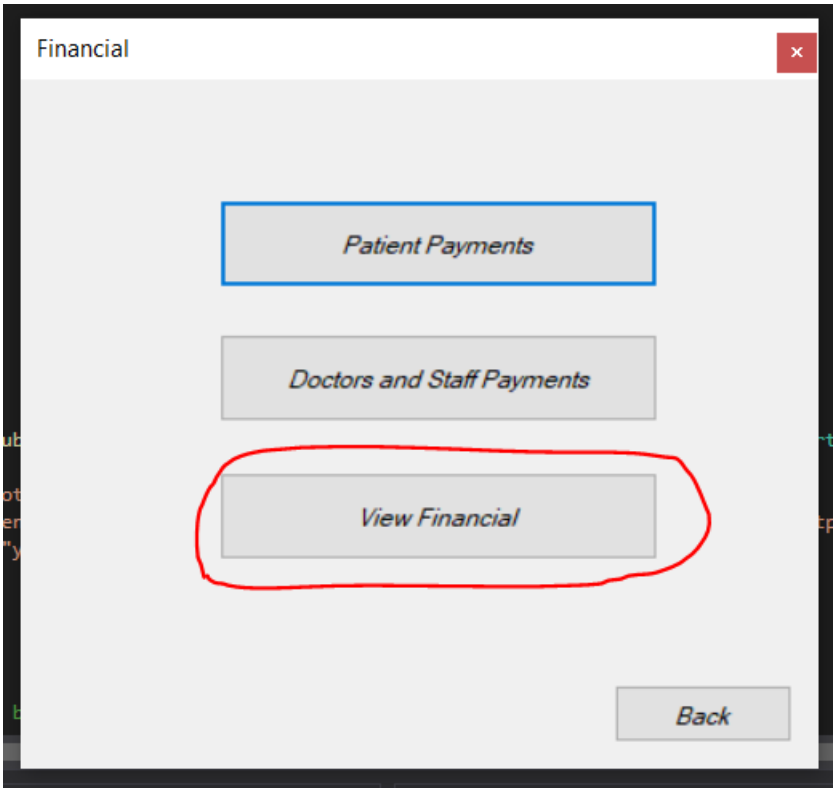


Now for the employees' financial section.



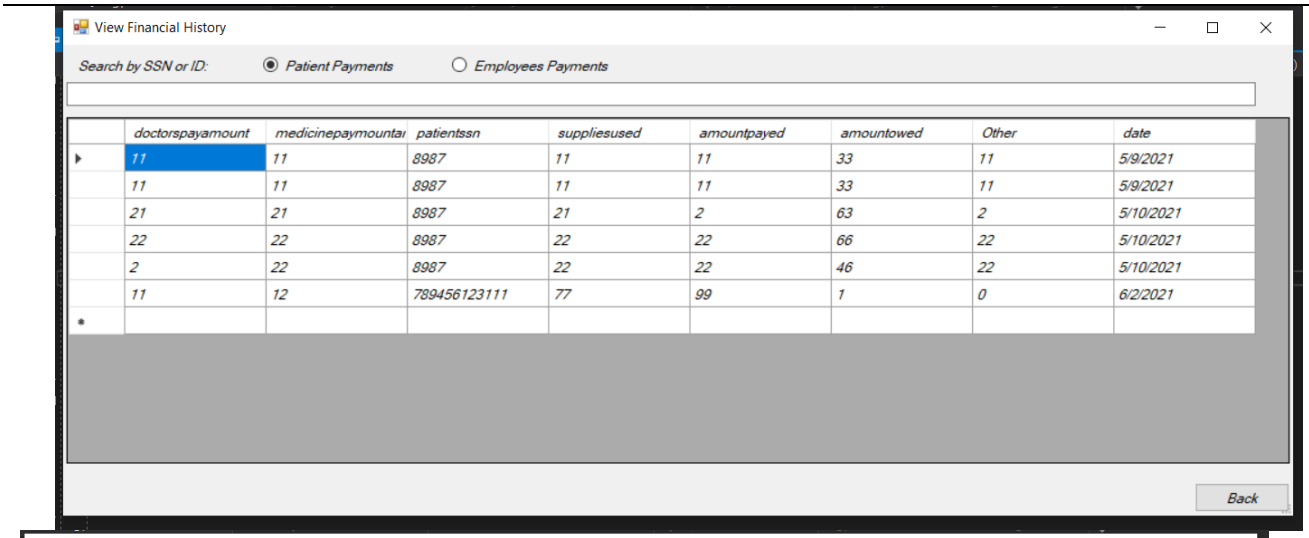
As we can see this is the same as the patient payments form with the same code. The only difference is this operation sends the data to the employees financial table. Also, the formula for calculating these months' pay is different. Here is the code difference between this page and patient payments page:

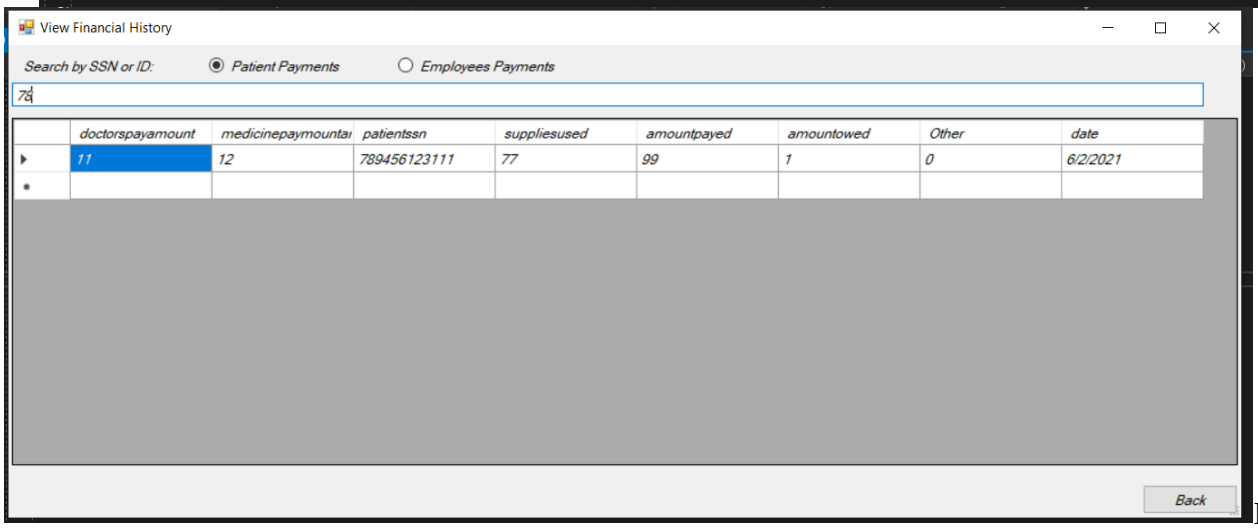
```
int days = Convert.ToInt32(this.textBox2.Text);
                    float salary = Convert.ToInt32(this.textBox3.Text);
                    int lvs = Convert.ToInt32(this.textBox4.Text);
                    float tmp = ((days * 8) - lvs) * salary;
                    textBox5.Text = Convert.ToString(tmp);
                    string MyConnection1 = "datasource=localhost;port=3306;username=root;
password=123456789ls";
                    string query = "insert into clinichospitaldb.employees_financial
(id,daysshowedup,salarybythehour,leaves,totalamountmonth,date) values('" + this.textBox1.Text +
"','" + this.textBox2.Text + "','" + this.textBox3.Text +
                                "','" + this.textBox4.Text + "','" + this.textBox5.Text + "','" +
DateTime.Now.ToString("yyyy-MM-dd")/*date*/ + "');";
```



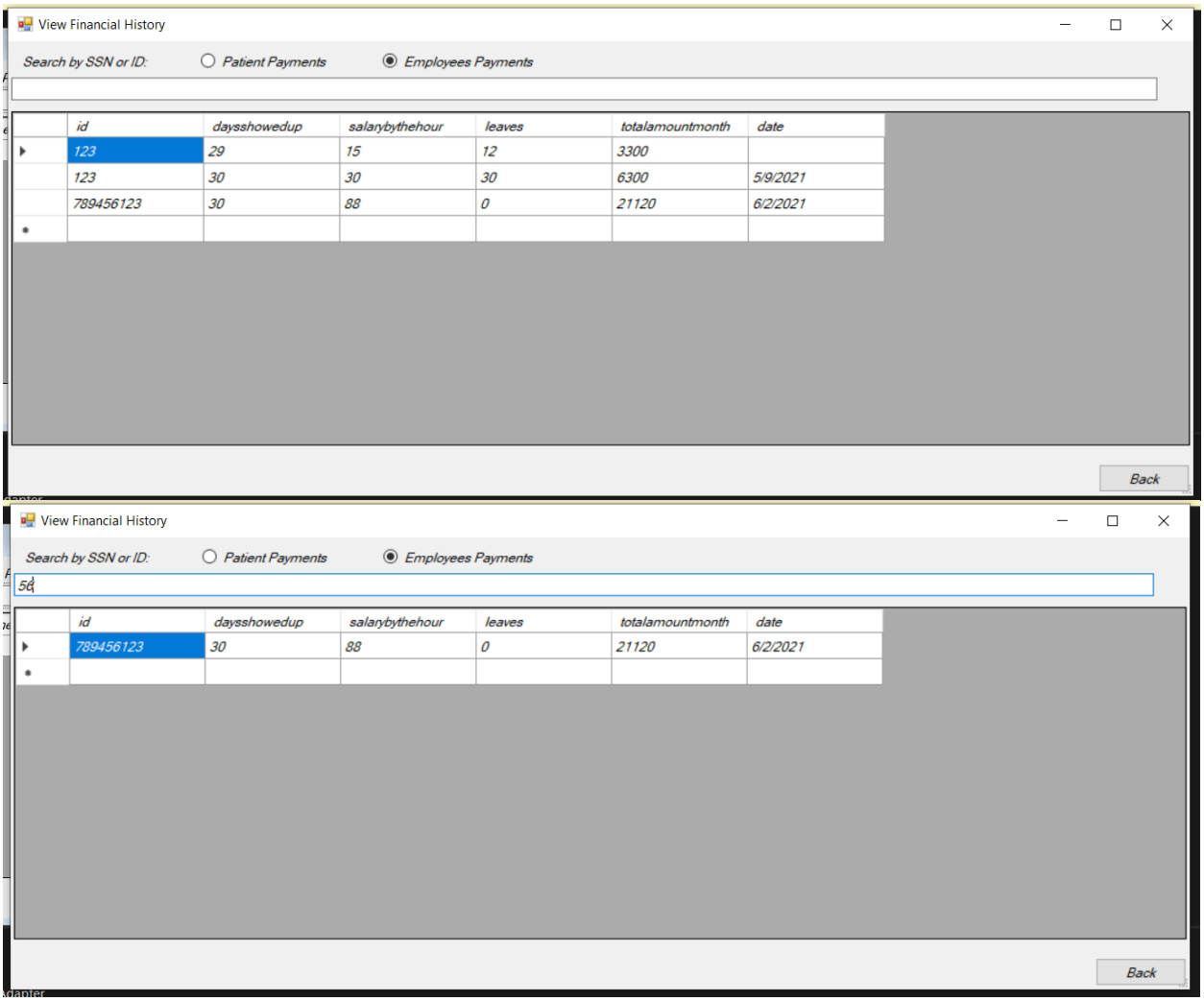
Now the last button on the financial window is the view financial button. This will take us to a new window that displays data grid views for patient payments and employee payments tables.

7.10 Viewing Financial



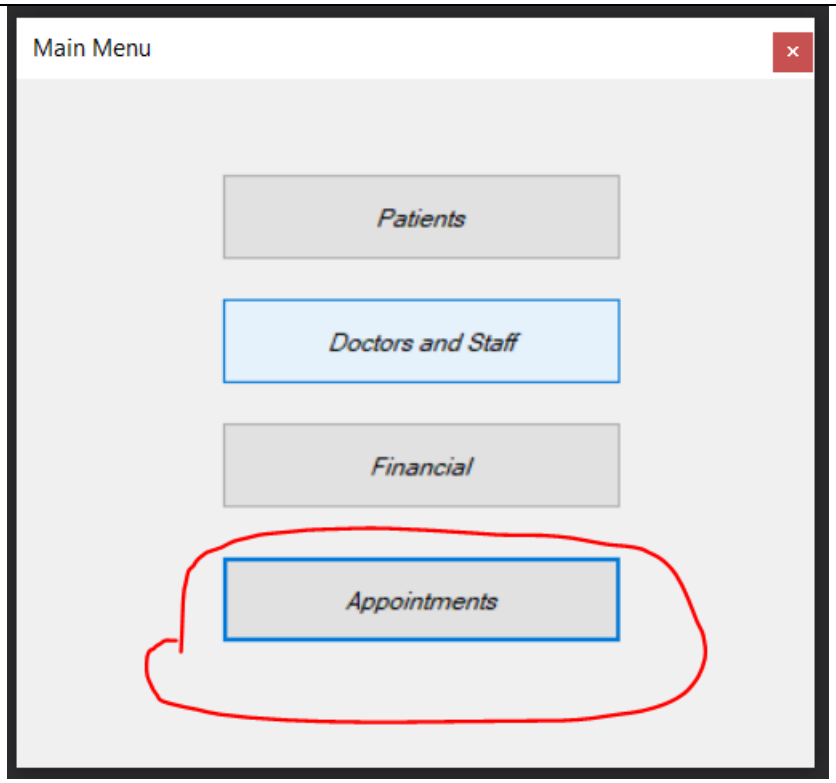


If the user chooses the first radio button patient payments table will be displayed. The text box uses filtering do search for a certain SSN.



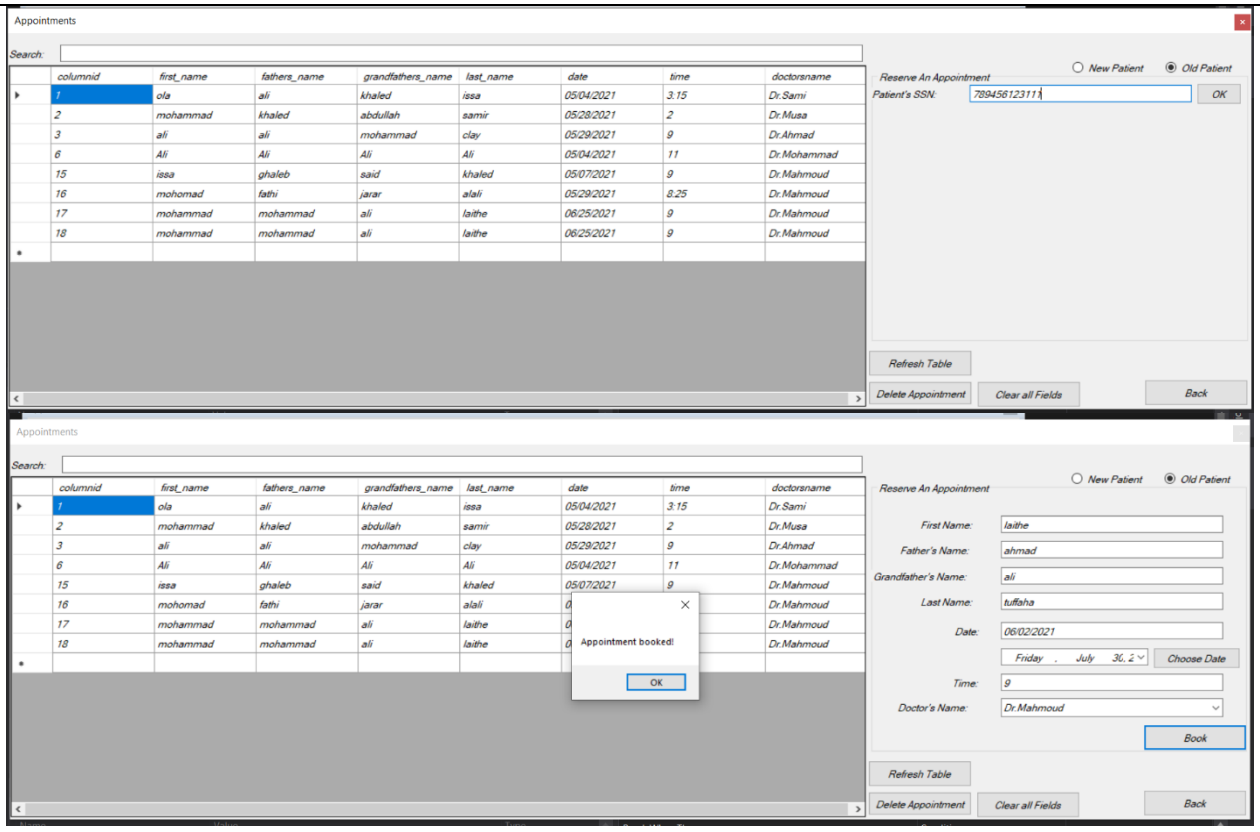
If the user chooses the second radio button employees' payments table will be displayed. The text box uses filtering do search for a certain Id.

7.11 Booking Appointments



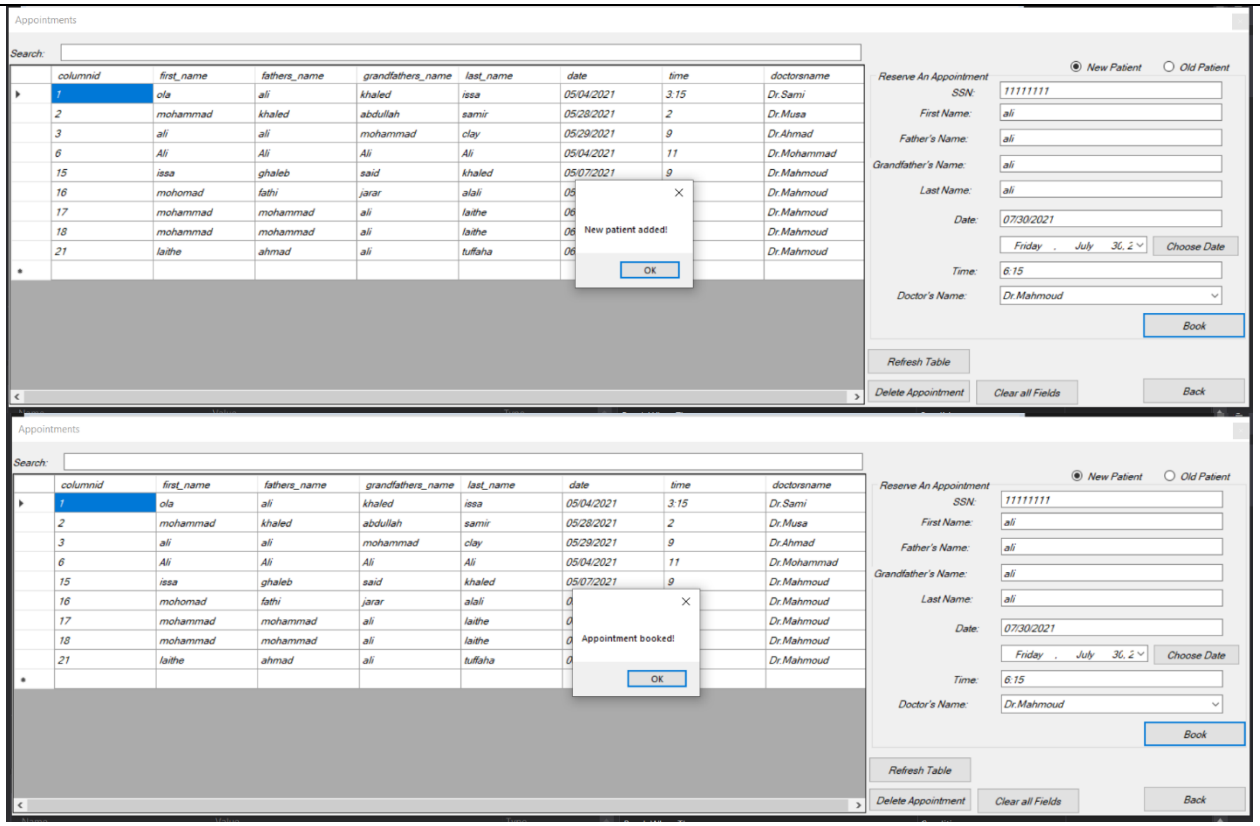
Now for the last button on the main menu is the appointments button.

7.12 Booking Appointments for Old Patients

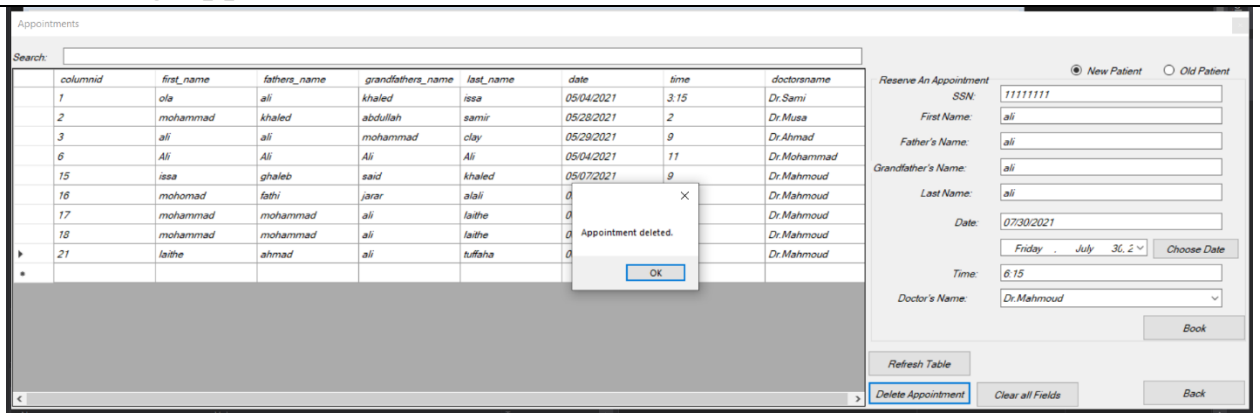


If the user is informed by the patient that this is an old patient the second radio button is chosen, and the patient's SSN is entered into the text box. The SSN in the textbox is then used in a select statement to select that patient's information from patients table and auto entered in the text boxes. The names of the doctors are loaded into the combo box by retrieving them with a select statement from the employees table and then adding them to the combo box, every time a doctor is added it is added automatically when this page is opened. ONLY employees with the job title doctor will be added. Then the user will only have to enter date, time, and choose doctors name from combo box. Once the user clicks the book button the data is taken from the textboxes and combo box and inserted into appointments table in the DB.

7.13 Booking Appointments for New Patients



7.14 Deleting Appointments



In the first two pictures as you can see the user has selected that this is a new patient. This is crucial on new patients because we need to add new patients to the patient table and fill their new data in. So, when the new patients radio button is selected two lines of SQL are executed, the first line to insert the new patient in the patients table and the second to send the data to appointment table to book an appointment. The last picture is deleting an appointment from the data grid view and the table appointments simply by selecting the appointment you want to delete. It will delete it from the database and the data grid view. The refresh table button is to refresh the table with the data in the database.

1. This is the old patient's appointment in the appointment table.

21	laithe	ahmad	ali	tuffaha	06/02/2021	9	Dr.Mahmoud

2. This is the new patients appointment in the appointment table.

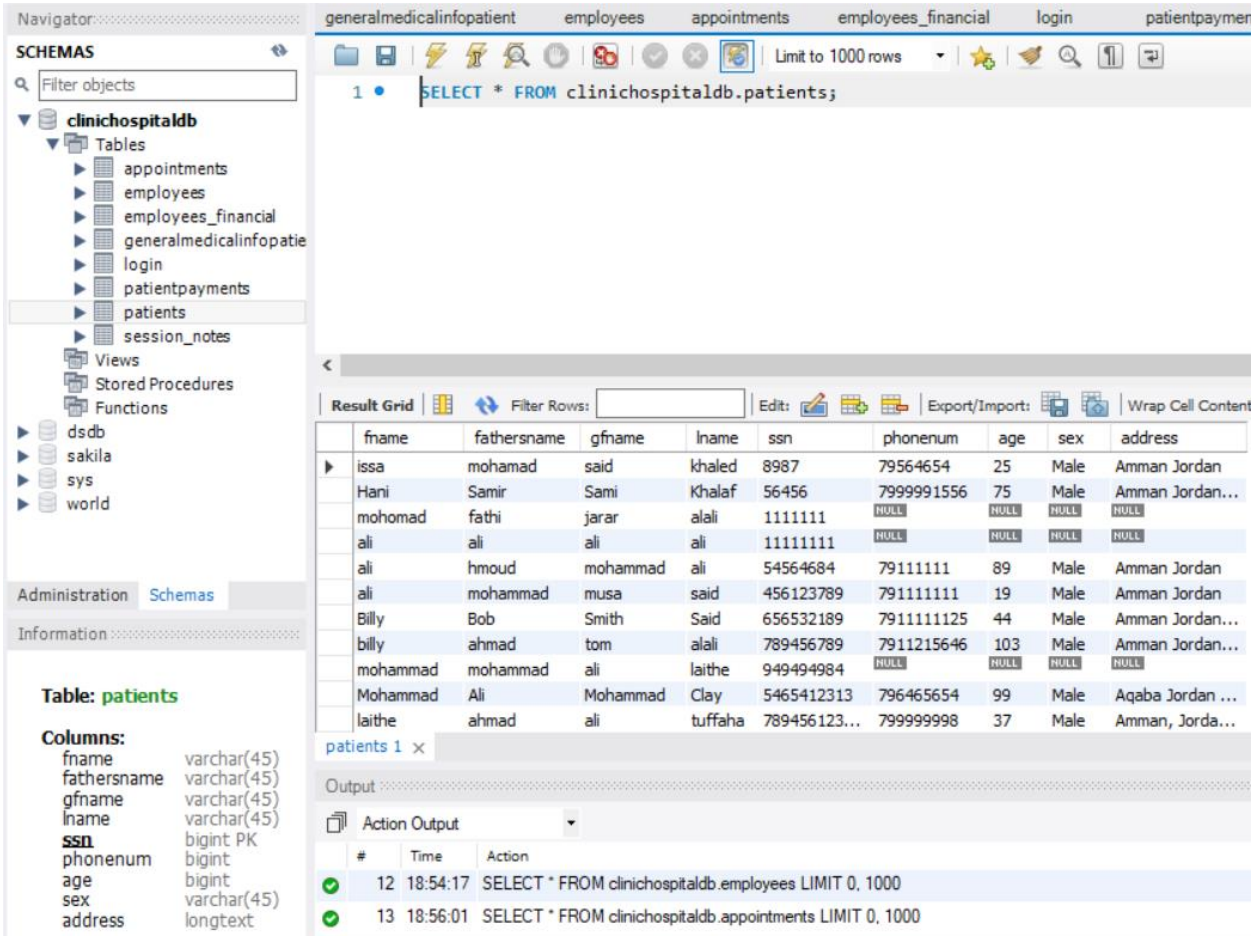
6	Ali	Ali	Ali	Ali	05/04/2021	11	Dr.Mohammad

3. This is the new patients added to the patients table with basic data entered only. Once the patient shows up the user can update this patients info.

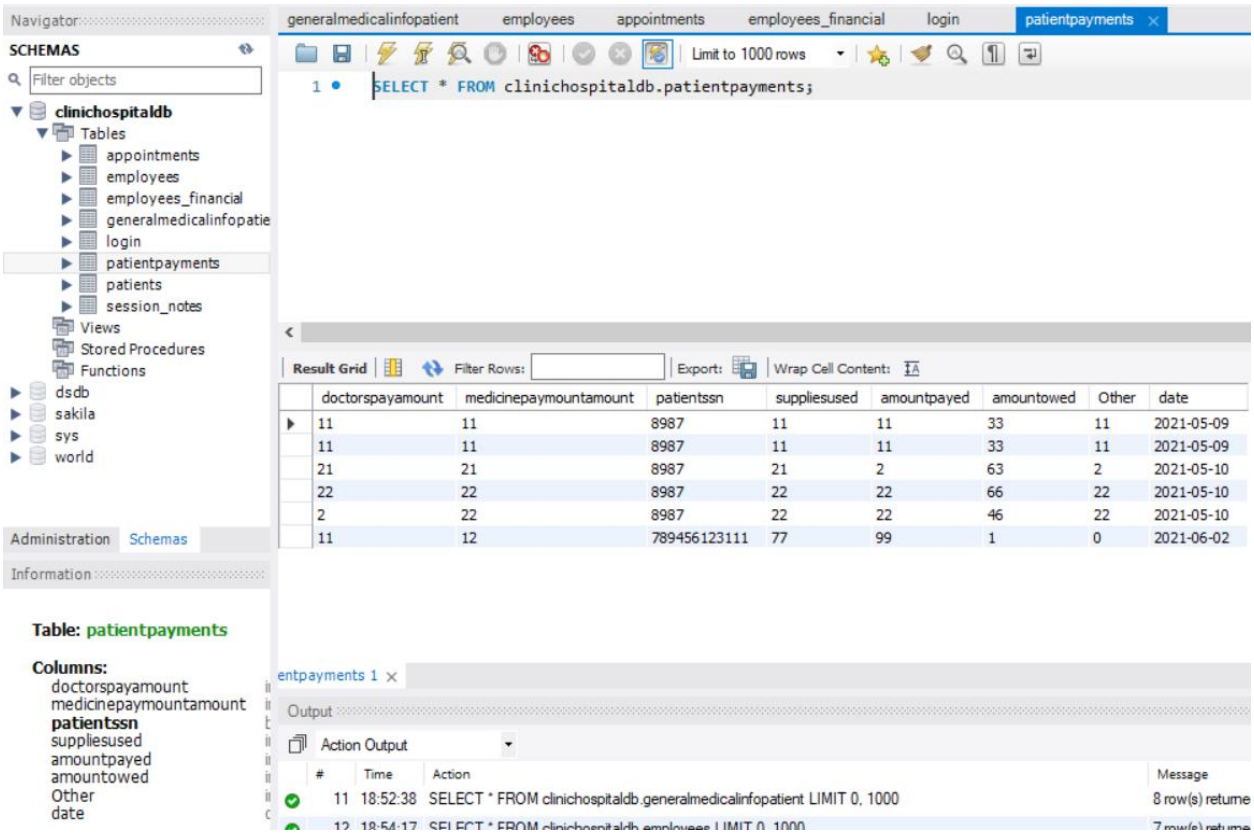
ali	ali	ali	ali	11111111	NULL	NULL	NULL	NULL

- one page. Example in the following figure

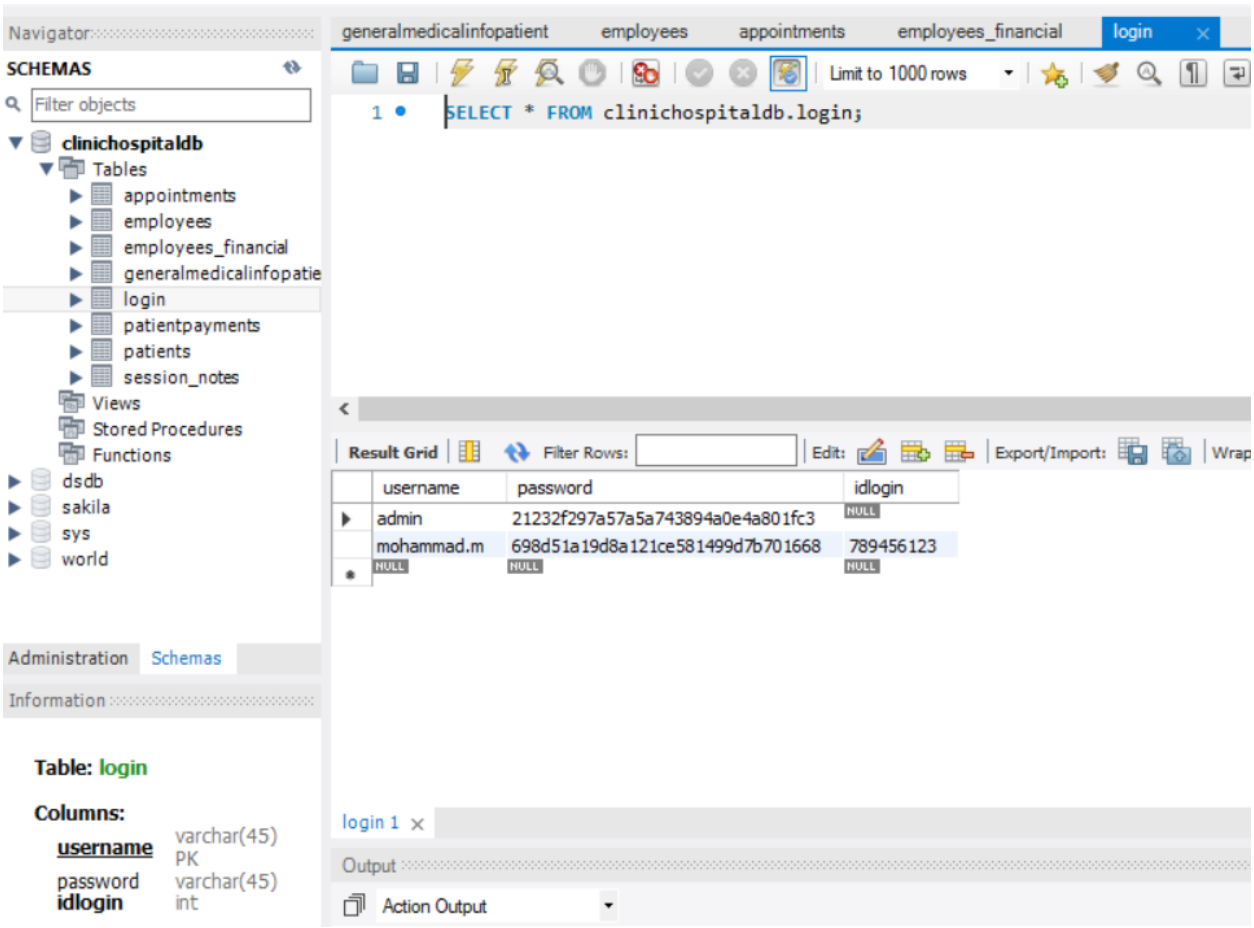
8 Pictures of Table in the Database:



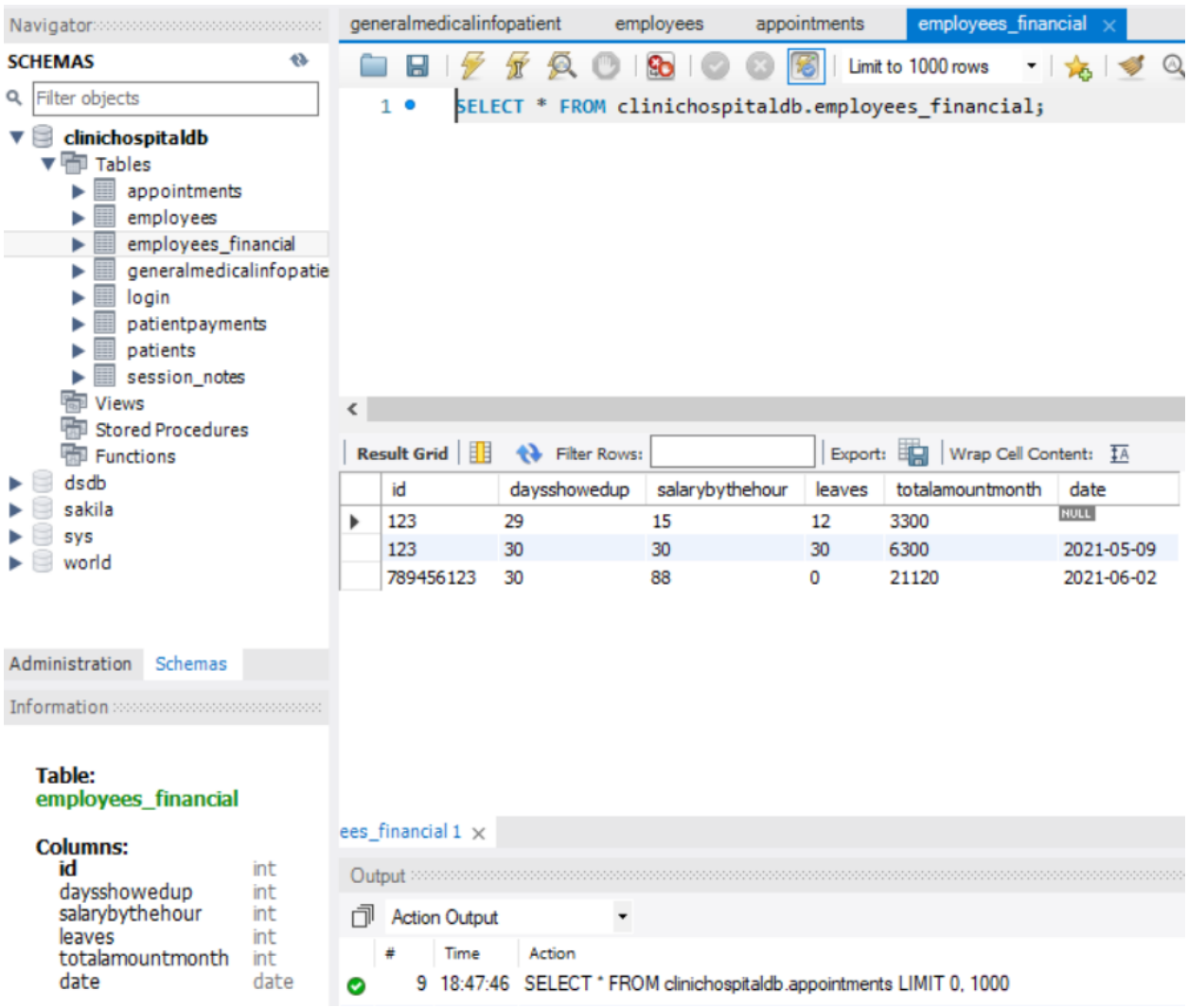
The image above is a screenshot of the patients table.



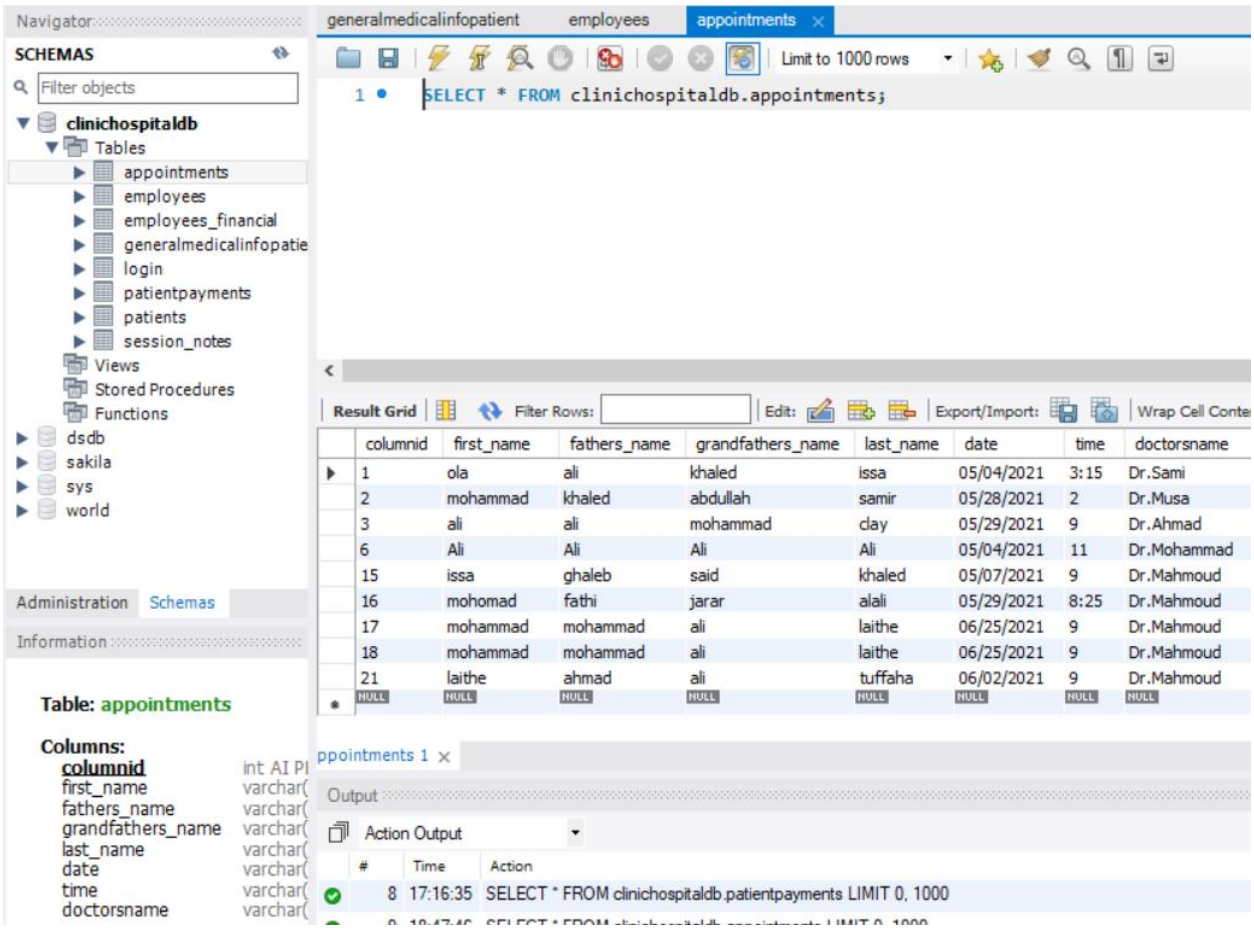
The image above is a screenshot of the patient payments table.



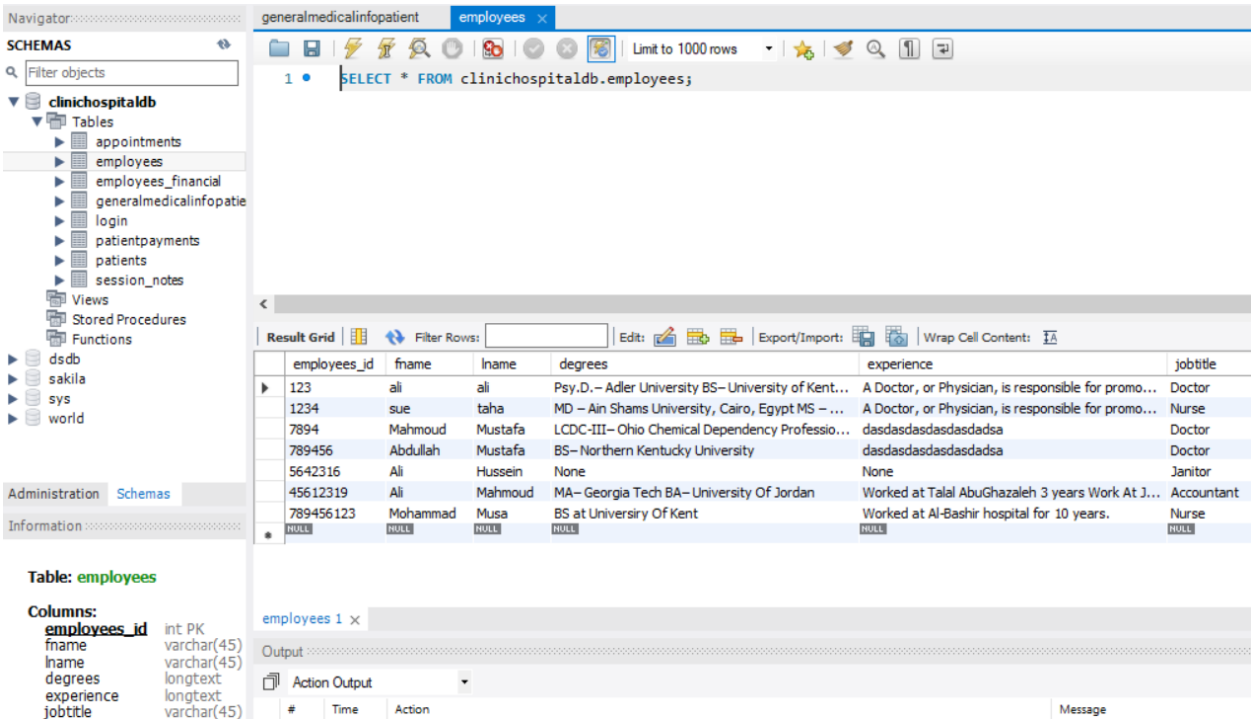
The image above is a screenshot of the login table.



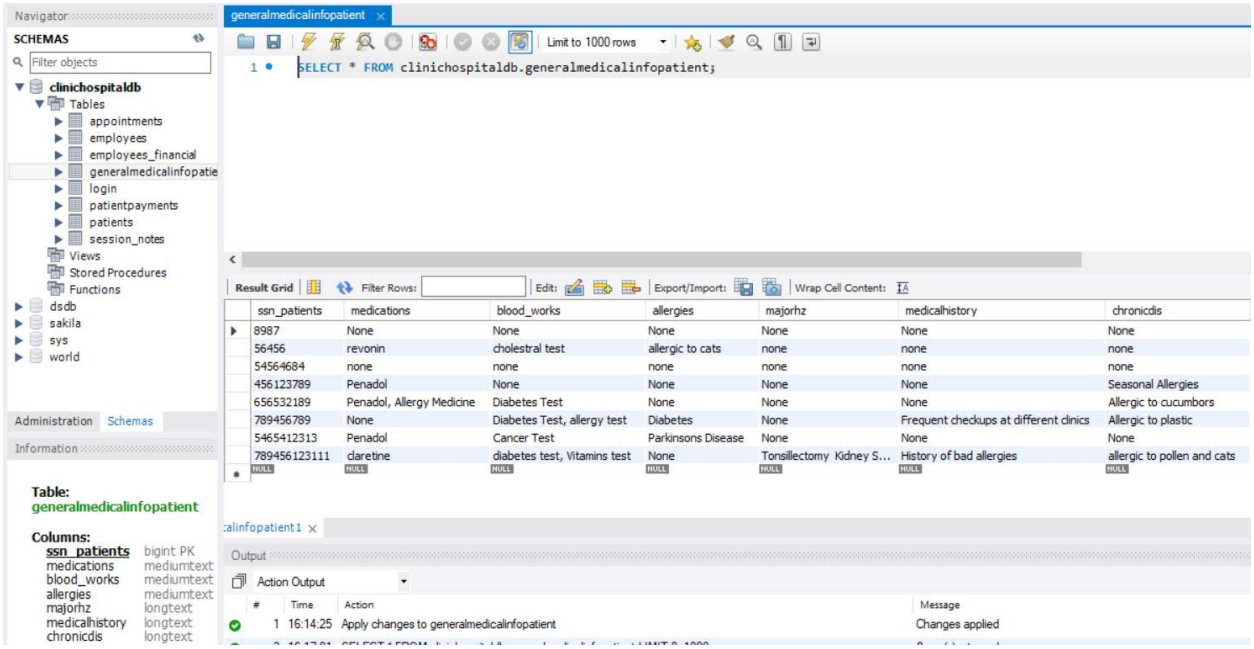
The image above is a screenshot of the employee's financial table.



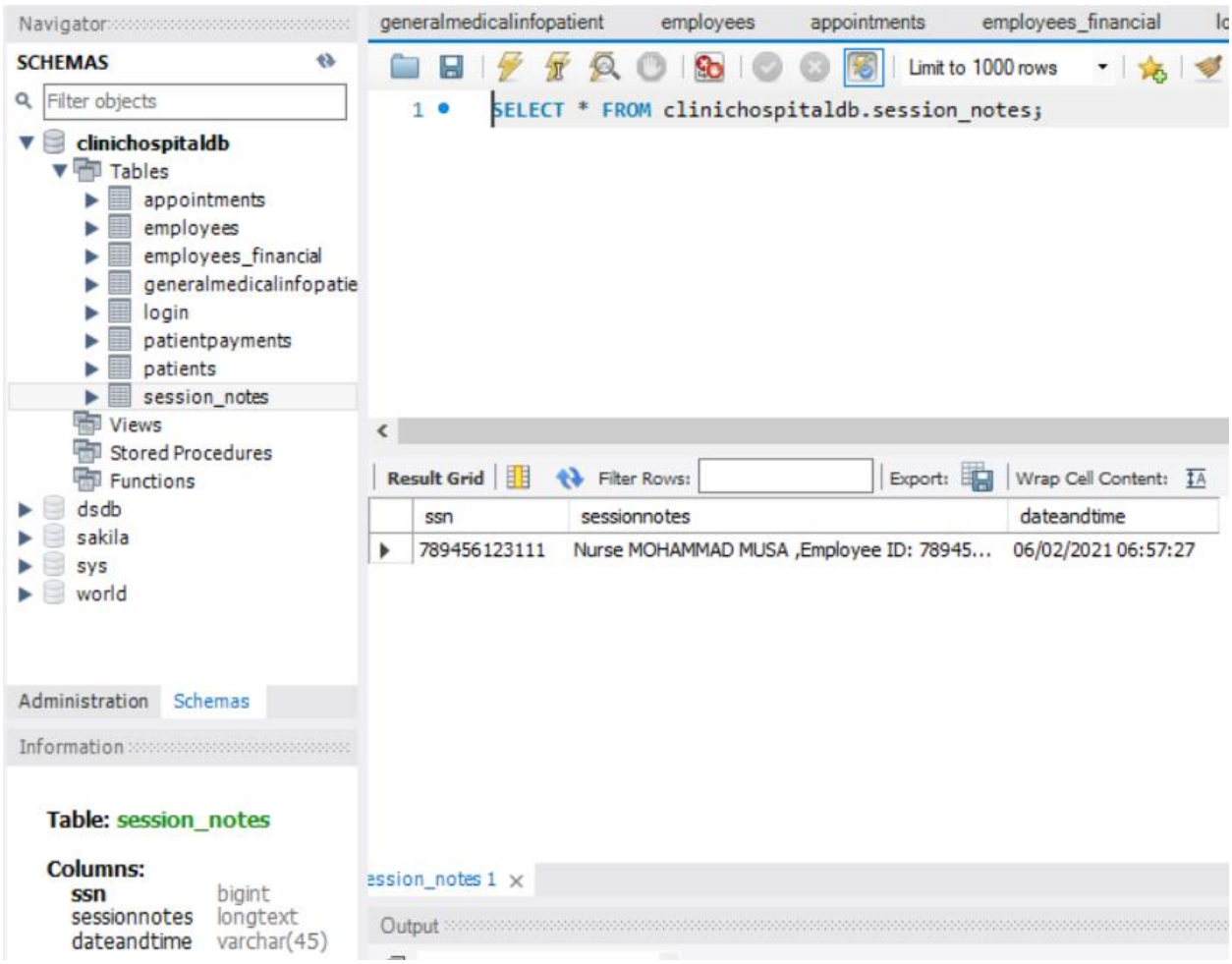
The image above is a screenshot of the appointment table.



The image above is a screenshot of the employees' table.



The image above is a screenshot of the patient's general medical information table.

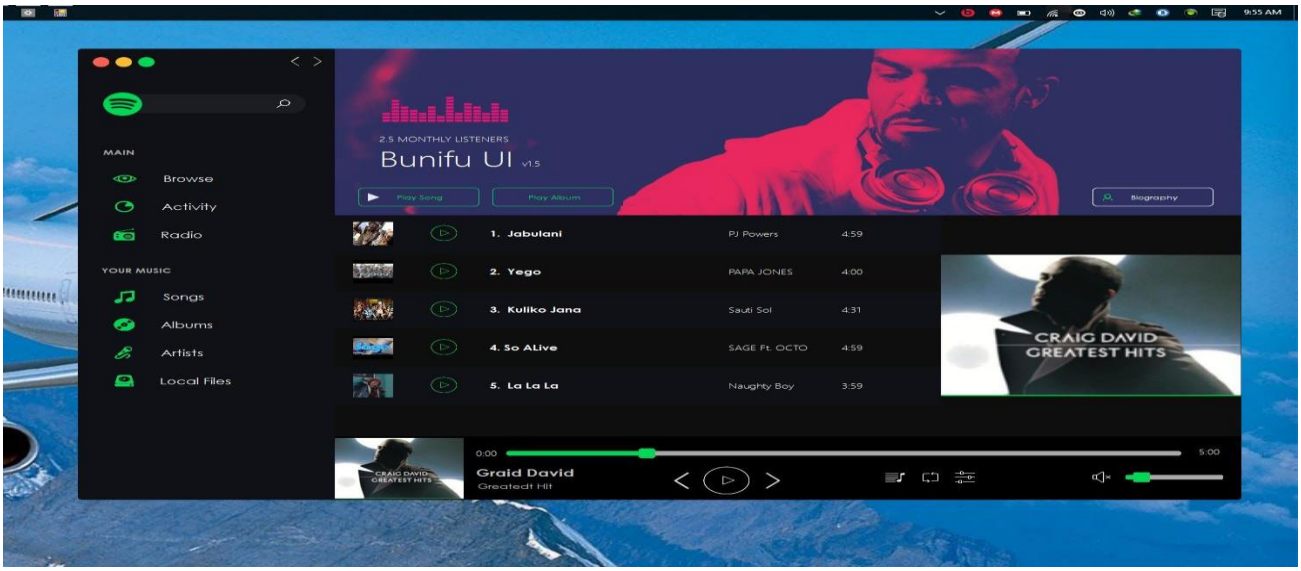


The image above is a screenshot of the session notes table.

9 Improvements

Every system even after development will require improvements, especially after the product has been sold. In this section I would change to enhance and improve the users experience while using my IS:

- This system contains four parts: booking appointments, financial, patients data storage, doctors data storage; So, as you can see this product is aimed for bigger enterprises leaving the smaller clinics purchasing more than they need. In the future as being able to dominate a larger portion of the market, splitting my IS into 4 different IS's is a better option because if a doctor in a small clinic with only one employee to keep track of appointments and financial and is only dealing with one patient at a time that doctor will not need to purchase the whole system instead he can purchase the patients part as a separate IS and the appointments part of the IS. Or, if a doctor is at a clinic by himself who does not have appointments and just needs to keep track of patients and what antibiotics he took, what happened during the sessions, and blood works etc. The doctor will only need to purchase the patient's part of the IS. This will make the IS less complicated for users that need only specific specifications from the full IS and it will make it cheaper for them.
- Another improvement would be upgrading from forms hiding and closing to making a single page form with tabs on the left side using the same and simple graphics. This will make navigation easier on user and the IS more user friendly. This will also make IS run faster because everything loads at once because it is all on



10 Conclusion

In this project, I designed, developed, and implemented an information system called Hospital/Clinic Information System. The testing experiments showed that our system is effective in managing patient records and appointment details. Also, the system processes the financial data of both the patients and the working staff. The database of the developed system has the capability to organize large quantities of medical data.

Even though I have completed the IS, but there is still room for improvements and updates especially after receiving feedback from the end-users once this software is released to the public. In the future, I would like to make improvements and contribute even more to help the health sector of the world. Thank you for taking the time to read through my documentation and manual.

11 References

- developers, I. t. (2021). *coronavirus*. Retrieved from worldometers: https://www.worldometers.info/coronavirus/
- Earp, S. B. (2003). *Database Design Using Entity-Relationship Diagrams.* Boca Raton: Auerbach Publications.
- Martin, R. C. (2002). *Agile Software Development.* Boston: Pearson Education, Inc.
- *uml-use-case-diagram*. (n.d.). Retrieved from lucidchart: https://www.lucidchart.com/pages/uml-use-case-diagram
- Wasson, C. S. (2006). *System Analysis, Design, and Development* (1st ed.). New Jersey: John Wiley & Sons.
- Professional Psychiatric Services Cincinnati, OH
- https://stackoverflow.com/
- https://dev.mysql.com/