Group 1 Final Report: Healthcare Data Management and Database

Abhishek Pokharel, Elvis Soto, Rabin Singh, and Robel Gebremichael

St. Joseph University - New York

COM-370-B01: Advanced Computer Programming

Professor Taghi Ozbeki

1 May 2023

Abstract

This paper presents the creation of a hospital database management system developed using MySQL and XAMPP, and displayed through GUI, both drag and drop, and Object Oriented Programming (OOP) in a Java development environment. It highlights the significance of OOP, SQL database concepts, and their application together as a way to display, delete, update or insert massive amounts of data. The paper showcases how to utilize MySQL and Java OOP to create both a dynamic and efficient application for hospital database management. The findings demonstrate the successful application of object-oriented programming and database design principles to address complex healthcare management challenges.

Keywords: database, data management, foreign key, primary key, key violation, uml diagram

Contents

Abstract **2**

Introduction **4**

Database Tables and Attributes **5**

- *Patient Database Table*
- *Doctor Database Table*
- *Appointment Database Table*

Keys and Relationship **6**

Key Violation **8**

- *Primary Key Violation*
- *Foreign Key Violation*

Creating Records through GUI **9**

Updating the Record through GUI **13**

Deleting the record through GUI **16**

Error Handling **20**

Developers Point of View and Application Constraints **21**

Conclusion **22**

References **23**

Group 1 Report: Healthcare Data Management and Database

Introduction

We have created a Java application that brings the power of a reliable database system to healthcare organizations. Our application simplifies the management of patient and doctor information, as well as appointments. It is designed to streamline hospital administration, ensuring efficient access and manipulation of critical information of doctors and patients. Along with that, it makes the patient and doctors easier and efficient to access their health information while keeping the track of appointments.

With our application, hospital administrators can easily organize and update important records. The database contains all the necessary details about patients, doctors, and appointments, making it easier to manage hospital operations. Staff members cna quickly access patient and doctor information, which helps them make informed decisions and provide better care. Our application also allows administrators to retrieve additional information based on appointment details. For example, they can find out a doctor's specialty or a patient's insurance company by using the appointment IDs. This saves time and makes managing appointments smoother.

In this report, we provide a detailed explanation of our database system and application that works through the database. We discuss the details of the database through different tables, attributes, primary and foreign keys. Along with that, we explore the functionalities of the application's graphical user interface (GUI), including updating and deleting data within the database. Finally, we address the critical topic of error handling in the application, ensuring the quality user experience.

Database Tables and Attributes

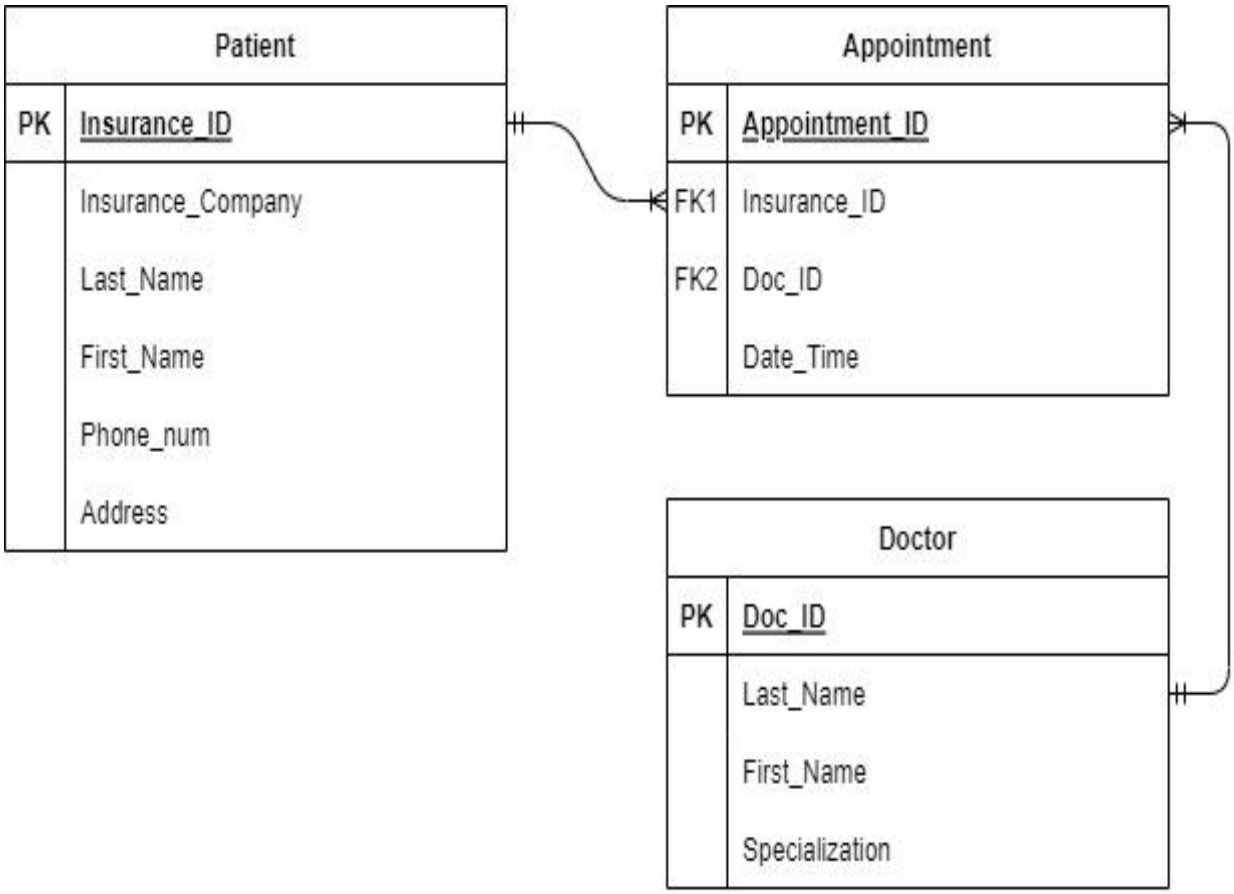


Figure 1: Database Table and Attributes

Patient Database Table

Keeps records on each patient that comes to the hospital. This table stores the patient's first and last name, their phone number, address, insurance company, and Insurance ID. These attributes are important for this table because they give up the patient's name, their contact information like their phone number and address, and their insurance information which is important information needed for them to receive care.

Doctor Database Table

To assign the doctors, check their availability, and keep records for further processes. We have four attributes: Doc_ID(PK), last name, first name, specialization. The Doc_ID is the

unique IDentifier column for doctor's IDentification which also serves as a primary key. Last name, first name columns have similar functionality but they are not necessarily unique. The specialization column is to assign the doctors by specialization and also keep record of availability in specific departments.

Appointment Database Table

This table holds the patient's insurance ID, doctors' ID and the appointment date and time. It helps organize which doctor the patient is seeing on what specific date and time. The appointment ID makes each appointment details unique even if the same patient has multiple appointments with the same/different doctors.

Keys and Relationship

Primary Key (PK) for the Appointment table is Appointment_ID, Doctor table is Doc_ID and Patient table is Insurance_ID. Table 1 depicts the information in a table format. These fields have been chosen as unique identifiers for their respective tables and are used to join data between tables as needed (Hollender, 2016). Each Primary Key is named consistently and has been set up with the appropriate data type and constraints for the database system being used.

Table_Name	Primary_Key
1. Appointment	Appointment_ID
2. Doctor	Doc_ID
3. Patient	Insurance_ID

Table 1: Primary Key

In this database, there exist Foreign Keys (FK) only for the Appointment table which are Doc_ID and Insurance_ID. These fields reference the Primary Keys of the Doctor and Patient

tables, respectively, and are used to establish relationships between the tables. The Foreign Keys ensure that data entered into the Appointment table is consistent with the data in the related tables, and they help to enforce referential integrity. The other tables in the database may also have Foreign Keys that reference the Appointment table, depending on the specific relationships between them.

There are two PK - FK relationships in the proposed database. One between Patient and Appointment, and another between Doctor and Appointment. Appointment receives a FK from both Patient and Doctor. We need this relationship because these FKs link the doctors and patients in the Appointment table to doctors and patients in the Patient and Doctor table (AssafMSFT, 2023). This allows us to find the doctors or patients from certain appointments in the appointment table and find out more information from them in the Doctor or Patient table, like their full names, addresses or doctor specialty.

This is a sample database designed for managing patient and doctor information and appointments. The database was generated using MySQL, a popular open-source relational database management system (Oracle, 2018). By using this sample database, healthcare providers can easily manage patient appointments, view patient and doctor information, and update records as needed. Please note that this is just a sample database and can be customized based on specific needs and requirements.

←T→			Appointment_ID	Insurance_ID	Doc_ID	Date_Time
Edit	Copy	Delete	1006	XM382748J	435	2023-01-09 12:15:00
Edit	Copy	Delete	1007	XM275429L	673	2023-01-12 10:05:00
Edit	Copy	Delete	1008	XM178390Y	847	2023-01-13 15:00:00

Sample 1: Database Entry Sample - Appointment Table

			Doc_ID	Last_Name	First_Name	Specialization
Edit	Copy	Delete	435	Johnson	Fred	Dermatology
Edit	Copy	Delete	673	Buble	Richard	Anesthesiology
Edit	Copy	Delete	847	Jackson	Owen	Internal Medicine

Sample 2: Database Entry Sample - Doctor Table

			Insurance_ID	Insurance_Company	Last_Name	First_Name	Phone_Num	Address
Edit	Copy	Delete	XM178390Y	GoodHealth	Swan	Amy	903-274-1937	3022 New City
Edit	Copy	Delete	XM275429L	CarePlus	Griffin	Emily	917-498-2351	40 Wall St.
Edit	Copy	Delete	XM382748J	Fidelis	Swanson	Joe	303-643-8000	23 Fulton St.

Sample 3: Database Entry Sample - Patient Table

Key Violation

Primary Key Violation

In the Doctor Table, there is a possibility of a primary key violation for the Doc_ID column, which serves as the primary key. Since the same doctor can treat multiple patients, it is possible to have multiple entries with the same Doc_ID. In the Patient Table, primary key violations can occur when different Patient_FirstName or Patient_LastName values are entered for the same insurance ID. The Insurance_ID column is the primary key in this table.The Appointment Table also faces the risk of primary key violation. This occurs when different Date_time values are entered for the same Appointment_Id. In other words, there are two different appointment times for the same appointment.

Foreign Key Violation

A foreign key violation occurs when there is an attempt to insert or update a record in a table that violates the referential integrity constraints defined by a foreign key. In this context, a foreign key is a field in one table that refers to the primary key of another table. In the

appointment table, the Doc_ID section is likely a foreign key that references the primary key of a table containing information about doctors. This foreign key constraint ensures that appointments are only made with existing doctors.

A sample foreign key violation in this context could occur if someone attempted to insert an appointment record with a Doc_ID value that does not exist in the referenced table. For instance, if the appointment record specifies Doc_ID = 99, but there is no corresponding record in the doctor table with a primary key value of 999, a foreign key violation would occur. This violation would prevent the insertion or update of the appointment record until the issue with the foreign key constraint is resolved.

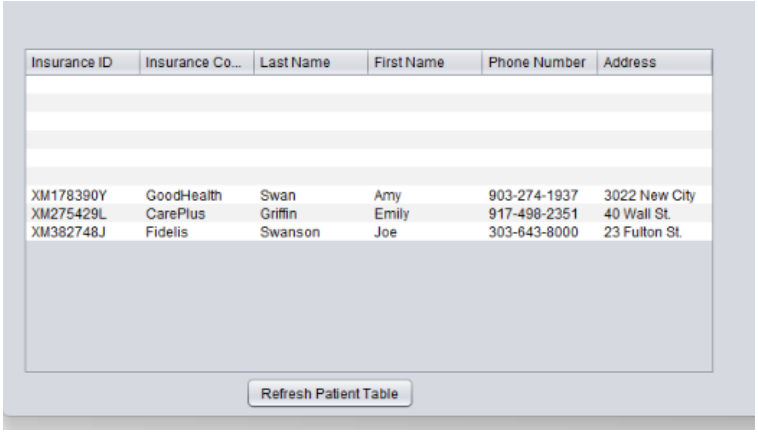
Creating Records through GUI

To create a record, the application follows a series of steps. First, it takes the user input and retrieves the data in the appropriate format. Then, it validates the input by checking data types, empty fields, and enforcing any necessary constraints to ensure data integrity. Once the data is validated, it is prepared for storage as a new record. This involves converting data types and formatting the data accordingly.

Next, the application establishes a connection to the database using Java Database Connectivity (JDBC). With the connection in place, it constructs an SQL insert statement that matches the structure of the destination table for the new record. This statement is executed, adding the new record to the database table.

Handling exceptions related to this process will be discussed later in the paper. Below is a sample illustrating the operation:

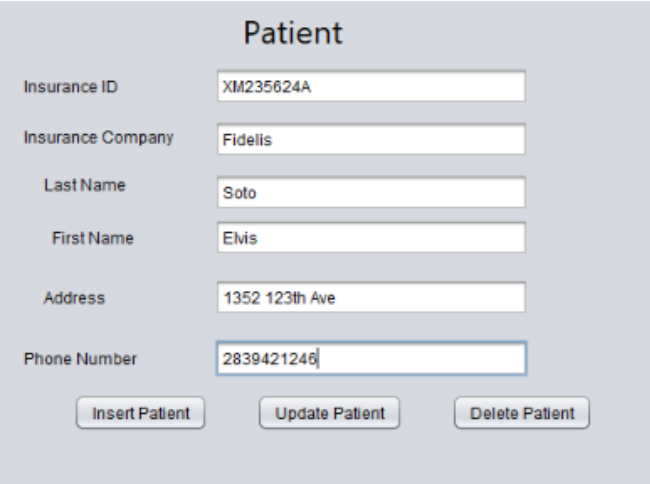
Patient Table



Insurance ID	Insurance Co...	Last Name	First Name	Phone Number	Address
XM178390Y	GoodHealth	Swan	Amy	903-274-1937	3022 New City
XM275429L	CarePlus	Griffin	Emily	917-498-2351	40 Wall St.
XM382748J	Fidelis	Swanson	Joe	303-643-8000	23 Fulton St.

Refresh Patient Table

Above is the patient table with 3 rows in it before any new inserts are made



Patient

Insurance ID: XM235624A

Insurance Company: Fidelis

Last Name: Soto

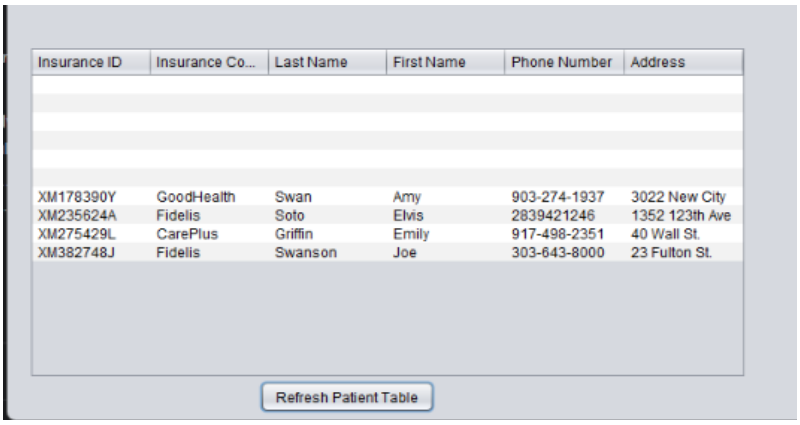
First Name: Elvis

Address: 1352 123th Ave

Phone Number: 2839421246

Insert Patient | Update Patient | Delete Patient

This is the new row of data being prepared to be inserted into the patient table

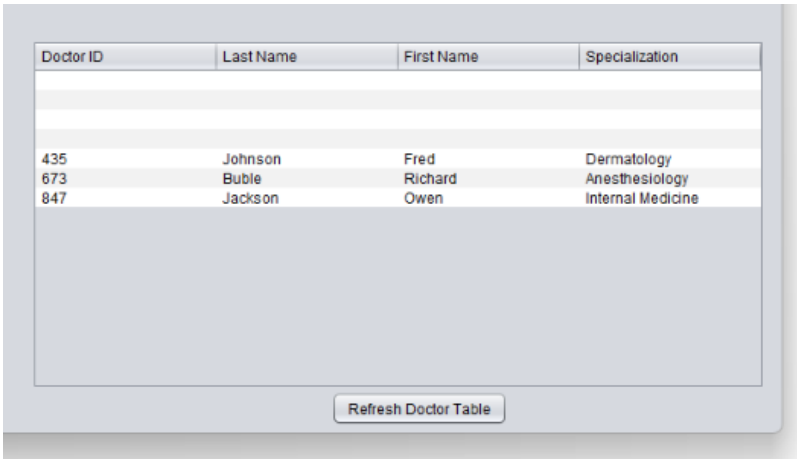


Insurance ID	Insurance Co...	Last Name	First Name	Phone Number	Address
XM178390Y	GoodHealth	Swan	Amy	903-274-1937	3022 New City
XM235624A	Fidelis	Soto	Elvis	2839421246	1352 123th Ave
XM275429L	CarePlus	Griffin	Emily	917-498-2351	40 Wall St.
XM382748J	Fidelis	Swanson	Joe	303-643-8000	23 Fulton St.

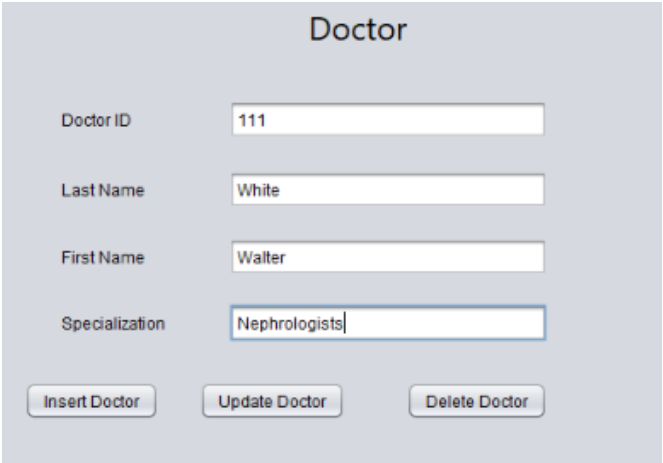
Refresh Patient Table

This is the patient table after that previous insert was made

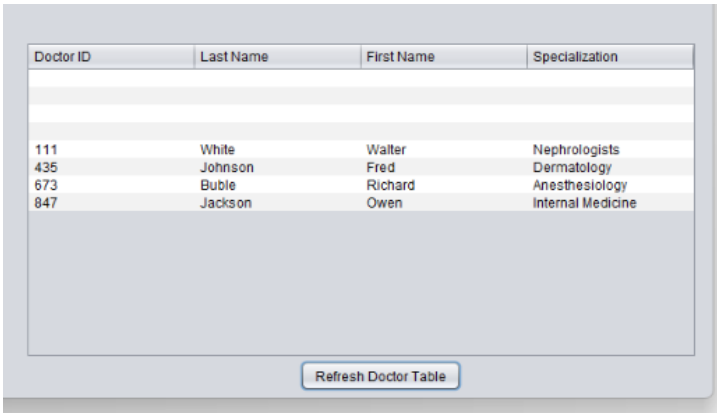
Doctor Table



This is the Doctor table with three rows of data before any inserts are made

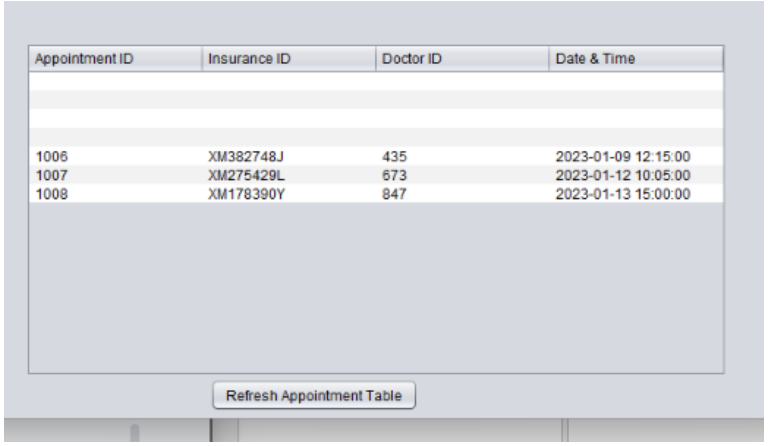


This is the data that is about to be inserted into the Doctor table



This is the Doctor table after that insert has been made. Now the Doctor table has 4 rows.

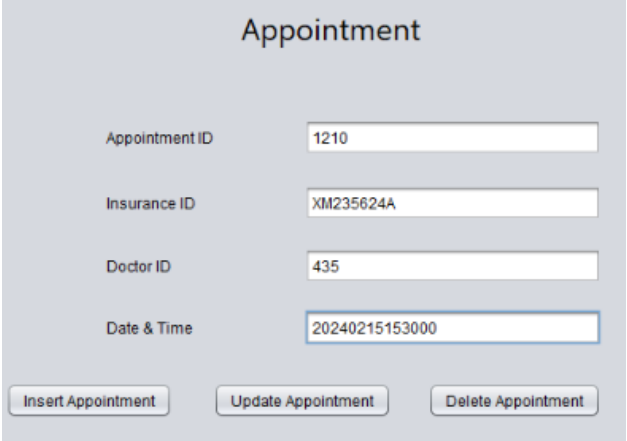
Appointment Table



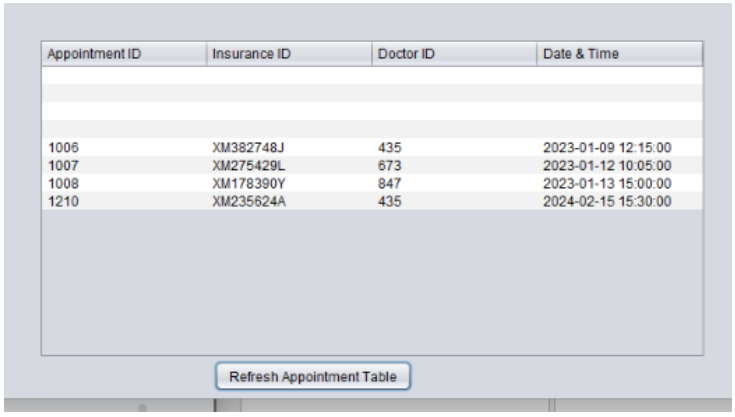
Appointment ID	Insurance ID	Doctor ID	Date & Time
1006	XM382748J	435	2023-01-09 12:15:00
1007	XM275429L	673	2023-01-12 10:05:00
1008	XM178390Y	847	2023-01-13 15:00:00

Refresh Appointment Table

This is the appointment table with only three rows without new insertion.



This is the new insert being made into the Appointment table.



Appointment ID	Insurance ID	Doctor ID	Date & Time
1006	XM382748J	435	2023-01-09 12:15:00
1007	XM275429L	673	2023-01-12 10:05:00
1008	XM178390Y	847	2023-01-13 15:00:00
1210	XM235624A	435	2024-02-15 15:30:00

Refresh Appointment Table

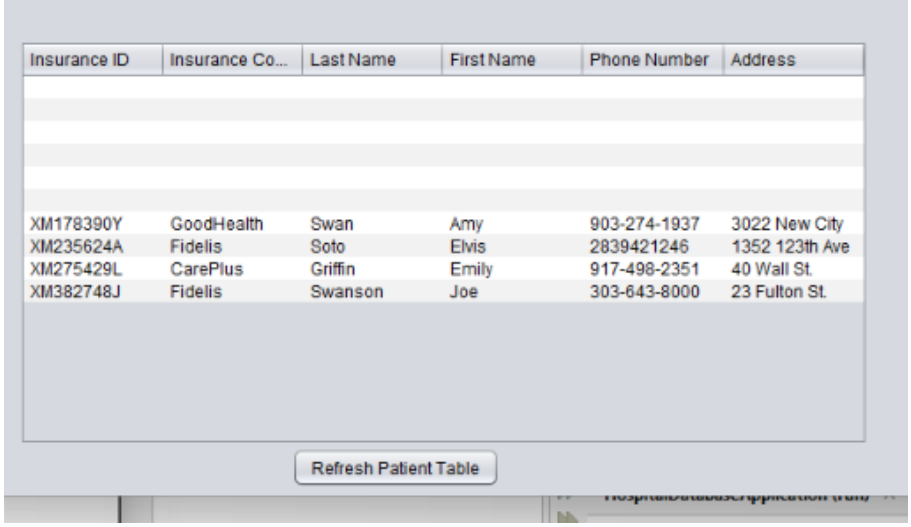
This is the Appointment table after that new row of data was inserted.

Updating the Record through GUI

In our application, updating a record is a very easy process. First the application displays the existing record by retrieving it from the database and populating the GUI components with the current data. This provides the user with a clear view of the existing values before making any modifications. To modify the record, the user can simply click the button in the GUI to make changes to the displayed record.

Once the modifications are made, the application follows the same steps as described in the previous section for creating a record. It validates the updated input, prepares the data for the record update, establishes a connection to the database using JDBC, and executes the SQL update statement. These steps guarantee data integrity and facilitate the easy updating of the record.

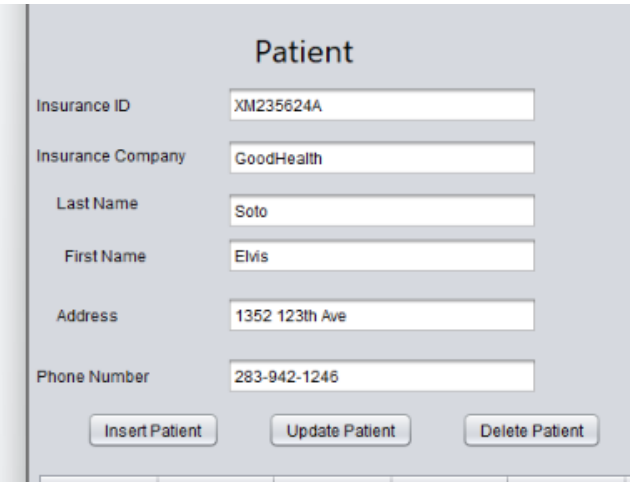
Patient Table



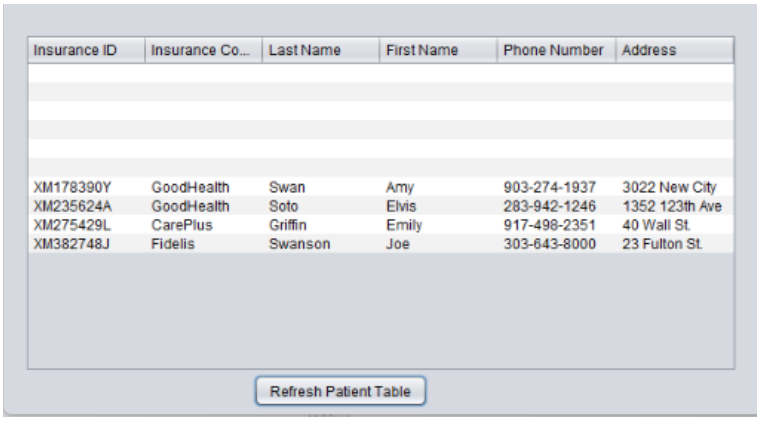
Insurance ID	Insurance Co...	Last Name	First Name	Phone Number	Address
XM178390Y	GoodHealth	Swan	Amy	903-274-1937	3022 New City
XM235624A	Fidelis	Soto	Elvis	2839421246	1352 123th Ave
XM275429L	CarePlus	Griffin	Emily	917-498-2351	40 Wall St.
XM382748J	Fidelis	Swanson	Joe	303-643-8000	23 Fulton St.

Refresh Patient Table

In the above image, the Patient table is shown with 4 rows

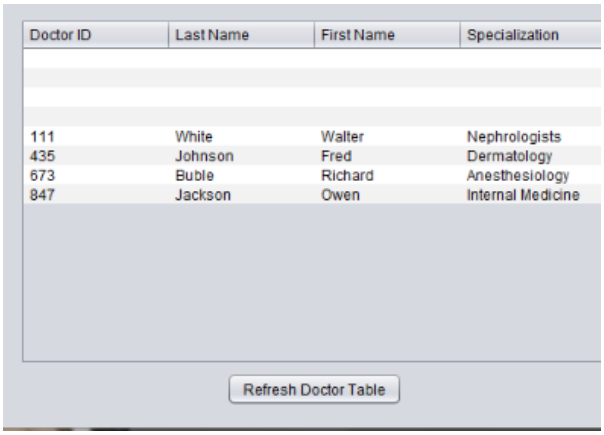


The user is updating this record by changing the Insurance Company of this patient.

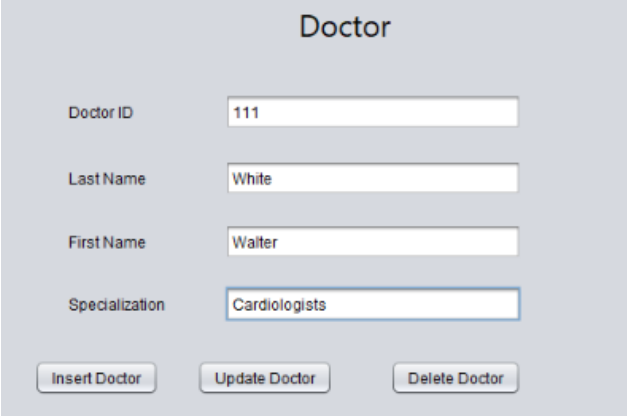


In this image, the insurance company of that patient was indeed changed.

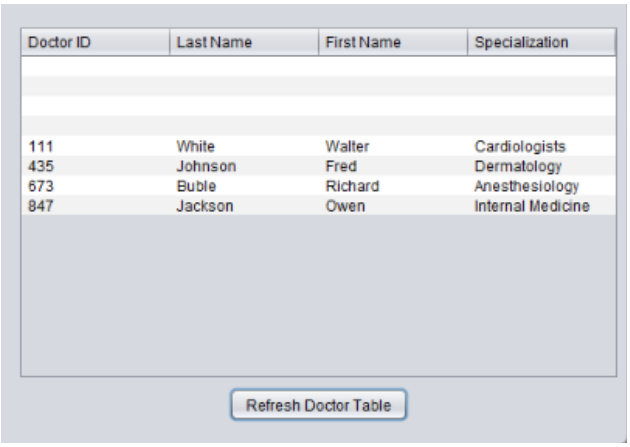
Doctor Table



This is an image of the Doctor Table before any updates were made.



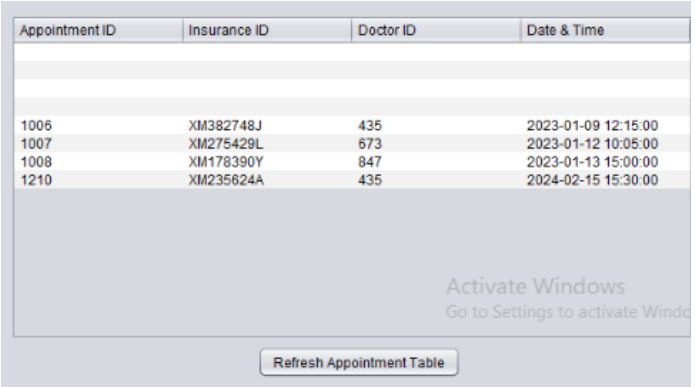
In this image above, the user is changing the Specialization of this doctor.



Doctor ID	Last Name	First Name	Specialization
111	White	Walter	Cardiologists
435	Johnson	Fred	Dermatology
673	Buble	Richard	Anesthesiology
847	Jackson	Owen	Internal Medicine

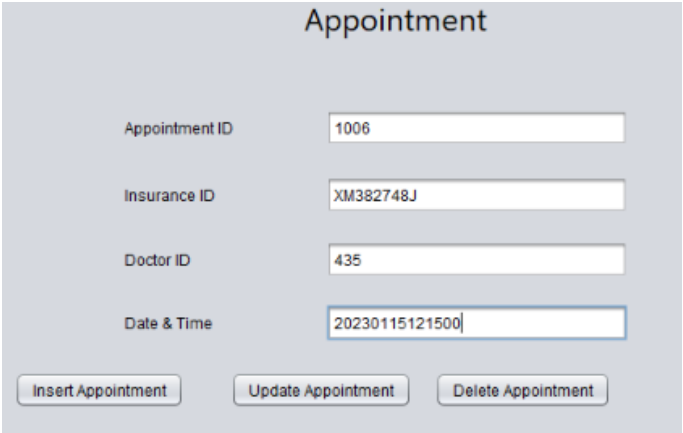
The doctor table now reflects the change and that doctor now has a different specialization.

Appointment Table

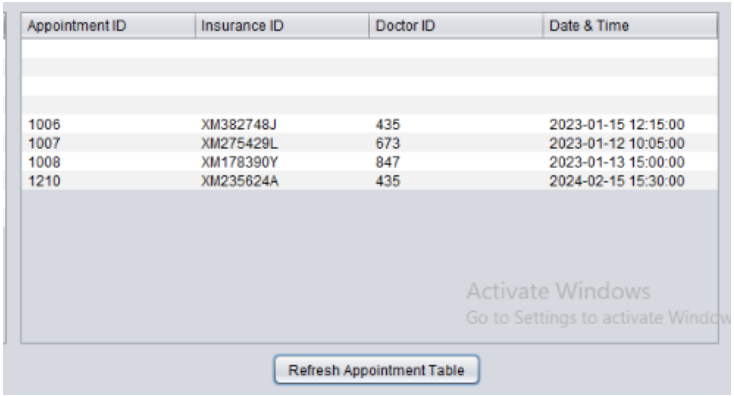


Appointment ID	Insurance ID	Doctor ID	Date & Time
1006	XM382748J	435	2023-01-09 12:15:00
1007	XM275429L	673	2023-01-12 10:05:00
1008	XM178390Y	847	2023-01-13 15:00:00
1210	XM235624A	435	2024-02-15 15:30:00

This is an image of the Appointment Table before any new updates are made.



The user in the image above is changing what day the appointment is.

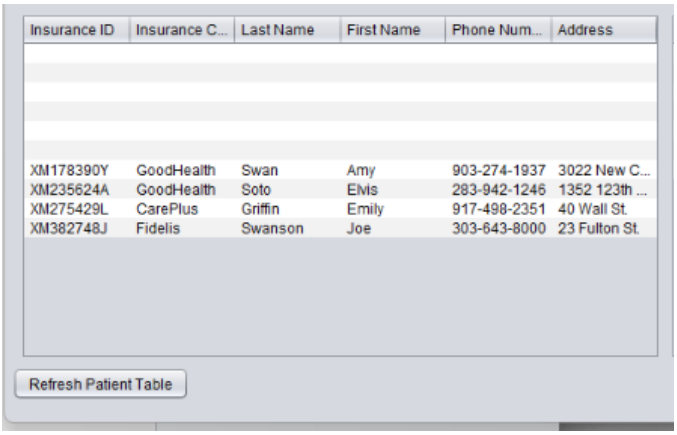


Appointment ID	Insurance ID	Doctor ID	Date & Time
1006	XM382748J	435	2023-01-15 12:15:00
1007	XM275429L	673	2023-01-12 10:05:00
1008	XM178390Y	847	2023-01-13 15:00:00
1210	XM235624A	435	2024-02-15 15:30:00

With the update, the appointment date for that specific appointment has been changed.

Deleting the record through GUI

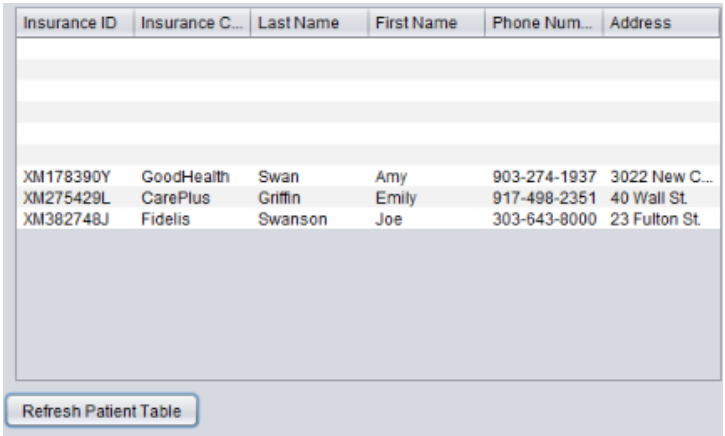
Patient Table



Insurance ID	Insurance C...	Last Name	First Name	Phone Num...	Address
XM178390Y	GoodHealth	Swan	Amy	903-274-1937	3022 New C...
XM235624A	GoodHealth	Soto	Elvis	283-942-1246	1352 123th ...
XM275429L	CarePlus	Griffin	Emily	917-498-2351	40 Wall St.
XM382748J	Fidelis	Swanson	Joe	303-643-8000	23 Fulton St.

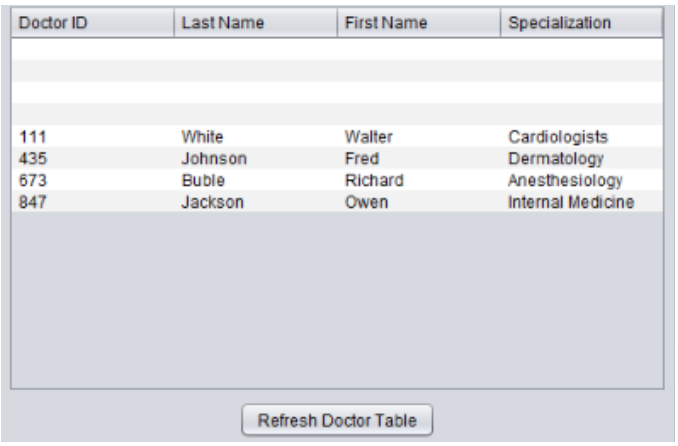
This is the Patient table before any delete statements have happened.

Inputting the primary key of the patient table of a patient to delete a record.

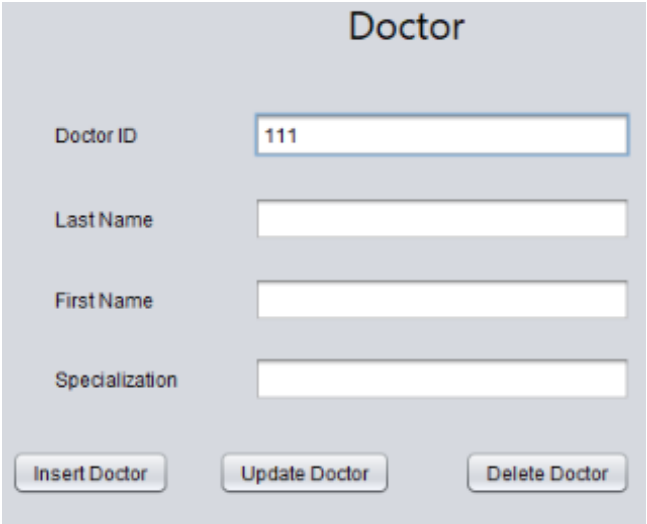


Now the Patient table is back to having only three rows.

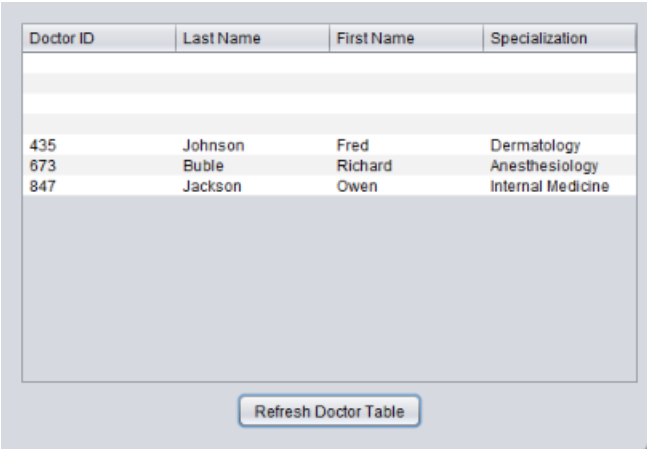
Doctor Table



This is the Doctor Table before any delete statements have been run against it.

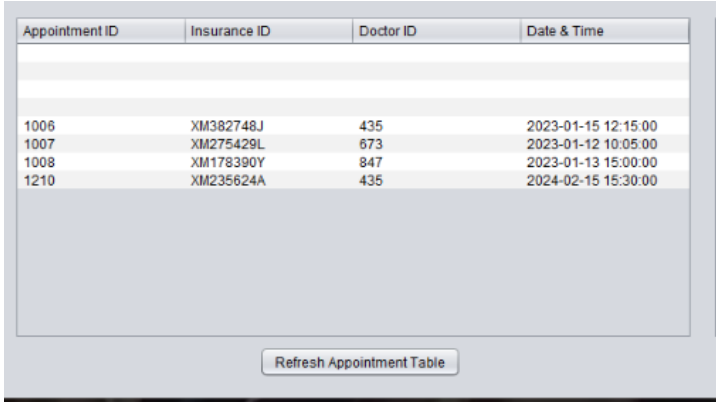


The user here is putting in the Doctor ID of a certain doctor that they are preparing to delete from the database.



Now the result of the previous delete statement has occurred and there are only three rows in the Doctor table.

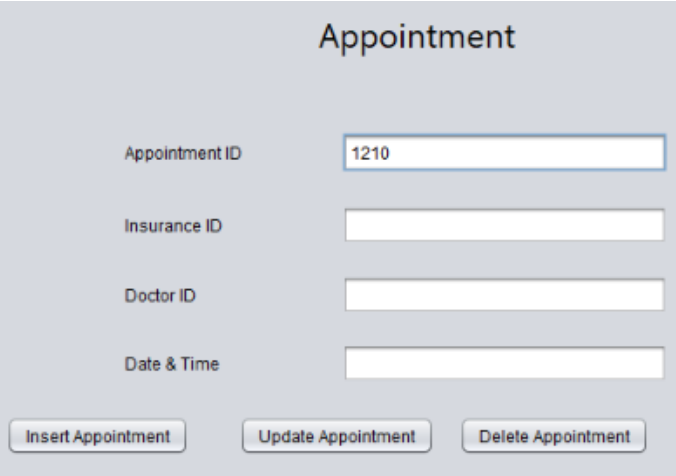
Appointment Table



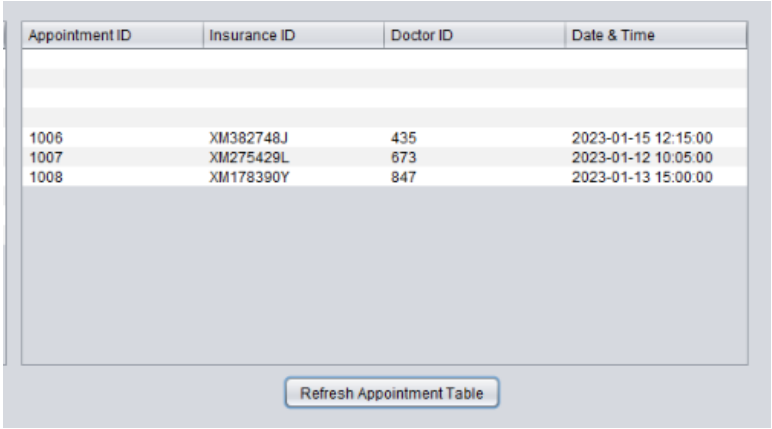
Appointment ID	Insurance ID	Doctor ID	Date & Time
1006	XM382748J	435	2023-01-15 12:15:00
1007	XM275429L	673	2023-01-12 10:05:00
1008	XM178390Y	847	2023-01-13 15:00:00
1210	XM235624A	435	2024-02-15 15:30:00

Refresh Appointment Table

This is the appointment table above before any delete statements have run against it.



User inputting the Appointment ID of a certain appointment in order to delete it.



Appointment ID	Insurance ID	Doctor ID	Date & Time
1006	XM382748J	435	2023-01-15 12:15:00
1007	XM275429L	673	2023-01-12 10:05:00
1008	XM178390Y	847	2023-01-13 15:00:00

Refresh Appointment Table

This is the view after the appointment ID 1210 is deleted.

Error Handling

Error handling is the process of identifying, anticipating, and resolving errors or exceptions that can occur during the execution of a program or application. Errors could arise due to various reasons, such as invalid user input, hardware or network failures, resource unavailability, programming mistakes or unexpected system conditions. In our application, we have mainly focused on handling invalid user input by providing appropriate responses and preventing the program from producing incorrect results and also crashing. Here are the five error handling examples in our application.

1. **Error handling for Primary Key Violation:** Database constraint is set on the primary key column to ensure that it always contains unique values by setting the primary key column as a UNIQUE and PRIMARY KEY constraint when creating the table. When you try to insert a record with a duplicate primary key value, the database will raise an error.
2. **Error handling for Foreign Key Violation:** Database constraint is set on the Foreign key column to ensure that it always references an existing record in the referenced table by setting the foreign key column as a FOREIGN KEY constraint when creating the table. When you try to insert a record with a non-existent foreign key value, the database will raise an error.
3. **Try-catch SQL exception Handling:** If connection cannot be made to the designated database it will throw a message to the user that asks to check the username, password.
4. **Overriding the methods for error handling:** some of the methods are overridden to avoid the error if error for different number of arguments.
5. **Data Type Error Handling:** Error message window will pop up with the right message for the user if they input for example letters in the phone number field. We are using regular expressions to validate the input pattern to avoid data error.

Developers Point of View and Application Constraints

From a developer's perspective, we are humbled by the achievements of our hospital database management system. Through the utilization of MySQL and Java programming, we have strived to create a solution that addresses real-world challenges in healthcare management. Throughout the development process, we recognized the significance of effective data management and handling. By incorporating primary keys, foreign keys, and relationships, we aimed to ensure data integrity within the system. We are particularly proud of the user-friendly GUI we have implemented, which simplifies record organization and updates for hospital administrators. Also, allows patients and doctors to keep track of their information.

Some important lessons that we learned are the value of time management and never promising more than you are able to realistically achieve. There were some ideas we had early on that would have simply taken too much time to achieve or distracted from the core purpose of what this hospital database management application should be. We also learned how important communication can be, especially when it comes to a project like this which can not be done in a single day alone. Through equally dividing the tasks between the members of the team, we were able to make an application that we are all proud of.

This project also has some constraints that should be kept in mind. Since we are not making a shuttle to the moon, it is important to remember that this is only a relatively simple database application that does not have all the security that a real life hospital application would have. The GUI is not as complex or aesthetically pleasing as we may have wanted, and at times certain components of the GUI can freak out and move around unexpectedly, causing a negative user experience. This is only a beginner's database application, but with this knowledge that we

gained from this project under our belt, we now all feel more confident in our ability as Java developers.

Conclusion

Overall, the paper emphasized the significance of object-oriented programming and database concepts in addressing real-world problems efficiently. It provided a detailed study of a hospital database management system, which utilized both MySQL for optimization and Java programming for automation, to develop a dynamic and effective hospital management system along with a Java application with a graphical user interface (GUI). The system efficiently manages patient and doctors information, as well as appointments, easing hospital administration and improving access to critical data. The application demonstrates the effective collaboration of object-oriented programming and SQL database concepts to develop a full-fledged application , showcasing the healthcare management challenges. The use of primary keys, foreign keys, and relationships between tables ensures data integrity and referential integrity. The system provides functionality for creating, updating and deleting records through the GUI, and error handling mechanisms are implemented to handle invalid user input. Overall, the developed database management system facilitates the hospital administration, staff and patients.

References

AssafMSFT, W. D. (2023). *Primary and foreign key constraints - SQL server*. SQL Server | Microsoft Learn. Retrieved March 24, 2023, from https://learn.microsoft.com/en-us/sql/relational-databases/tables/primary-and-foreign-key-constraints?view=sql-server-ver16

Hollender, J. (2016, January 1). *Violation of primary key constraint. cannot insert a duplicate key in the object*. Stack Overflow. Retrieved March 24, 2023, from https://stackoverflow.com/questions/36043439/violation-of-primary-key-constraint-cannot-insert-duplicate-key-in-object

Oracle. (2018). *MySQL 8.0 Reference Manual :: 11.2.2 the date, DATETIME, and timestamp types*. MySQL. Retrieved March 24, 2023, from https://dev.mysql.com/doc/refman/8.0/en/datetime.html#:~:text=The%20DATETIME%20type%20is%20used,both%20date%20and%20time%20parts