Distribution of Effort for Deliverable 4

Name	Revisions to Deliverable 3 (%)	Report – Application Implementation Section (%)	Report - Conclusion (%)	Implementation (%)	Demonstration Video (%)
Lana Adams	**50**	**50**	**50**	**50**	**50**
James Trafny	**50**	**50**	**50**	**50**	**50**

Indicate the percentage of effort per team member for each of these sections of the deliverable.

MediDB
Lana Adams and James Trafny
March 26th, 2018

Introduction:

MediDB will be an application that allows users to easily track and manage their medications and prescriptions. Medical terminology and abbreviations can be intimidating to laymen, and instructions written in medical jargon can be difficult to remember, especially for elderly patients or people who suffer from cognitive impairments. There are many dangerous drug interactions that patients rely on their doctors for warning them about. However, if there are issues with continuity of care or if the doctor is unaware of all the medications a patient is taking, these drug interactions may happen by mistake.

Considering all these issues, MediDB aims to achieve these primary goals:

1) Reduce the probability of dangerous drug interactions by alerting the user to potential risks in their regimen.
2) Produce a printable schedule for medications, reducing the risk of misreading abbreviated instructions (i.e. PO BID [orally, twice a day].)
3) Remind patients of prescription renewals based on RX refill dates.
4) Create a printable medication history to easily send to medical providers.

Possible secondary goals include:

1) Drug names from description and vise versa.
2) Translate between generic, name-brand, and chemical names.
3) Check for medication interactions based on body metrics such as height, weight, gender, medical history, or pregnancy status.
4) Notifications for medication reminders.

To accomplish these primary and secondary goals, we will need to store data about patients, patient drug regimens, and drug information. Regarding patient information, we will need: a user ID, name, date of birth, gender, weight, and possibly some medical history. For the patient drug regimen, we will need: name of medication, dosage, frequency, and amount. For drug information, we will need: generic name, brand name, chemical name, dose thresholds, and drug interactions. The user ID can be generated by hashing the user name and used internally as a key.

Using a database on the back end will be the most appropriate data structure for our need. This is because we will be storing largest amounts of related tuples that need to be cross referenced with each other when we look for drug interactions or when we prepare medication schedules.

Project Description:

From 2009-2012 the Center for Disease Control recorded that 47.3% of noninstitutionalized Americans used one prescription drug within 30 days of being asked, and 10.1% had taken five prescriptions or more (Center for Disease Control, 2016). Each prescription comes with a booklet of information explaining how to take the medicine, what to look out for in side effects, and any possible interactions with other medicine. This application will collect patients' medical information, and offer a printable schedule for medications, remind patients to take their medications, and provide a medical history report upon request.

The goal of the system is to eliminate the need to memorize the pages of prescription information; from multiple medicines and their different schedules. The system will facilitate adding new medications to the patient's schedule and reduce the chances of taking medicines that negatively interact.

This application will be constructed best using databases because of the large amount of data being accessed at different times. A database will make the information being stored easier to cross references as efficiently as possible.

User Classes:

Patient/User – Will need to use interfaces to modify any tables, limiting to what they can edit. For example, the patient should be able to update their weight, but not change their user ID.

Administrator – IT will need to be able to modify all tables for maintenance.

Stored Information:

Patient/User Table:
user ID
userPW
firstName
lastName
date of birth
gender
weight

Regimens Table:
medication name
dosage
frequency
count

Drug Information Table:
generic name
brand name
chemical name
dose thresholds
drug interactions

Input, Update, and Retrieval Scenarios:

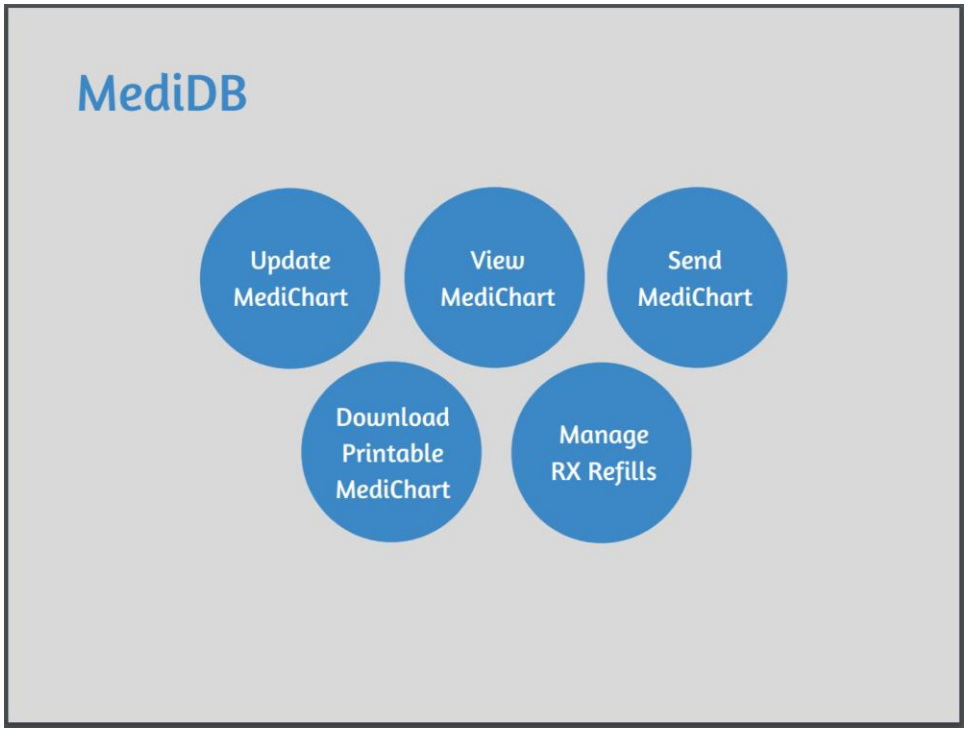
A new user may wish to register with the application. To do this, we will need to update the User Table with input from the new user interface. The user may then

want to add a medication to their current drug regimen. To do this, we will need to update the Regimens Table with input from the new medication interface.

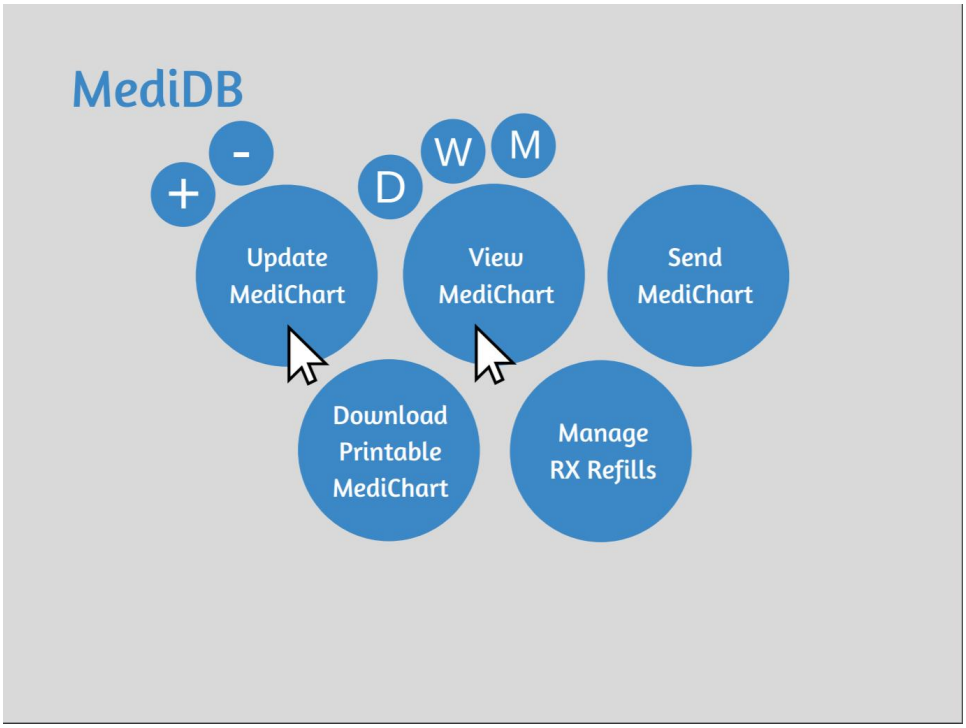
A user may have a medication called “Acetaminophen”, and they cannot remember what it if for. Using an interface for translating brand and generic medication names, the system would retrieve the name “Tylenol”. If the user then wants to print out their weekly medication regimen, the system would retrieve and format a list for the user.

An administrator may wish to add a new category of medications. The administrator will need to update the Drug Information Tables to include the new drug classification.

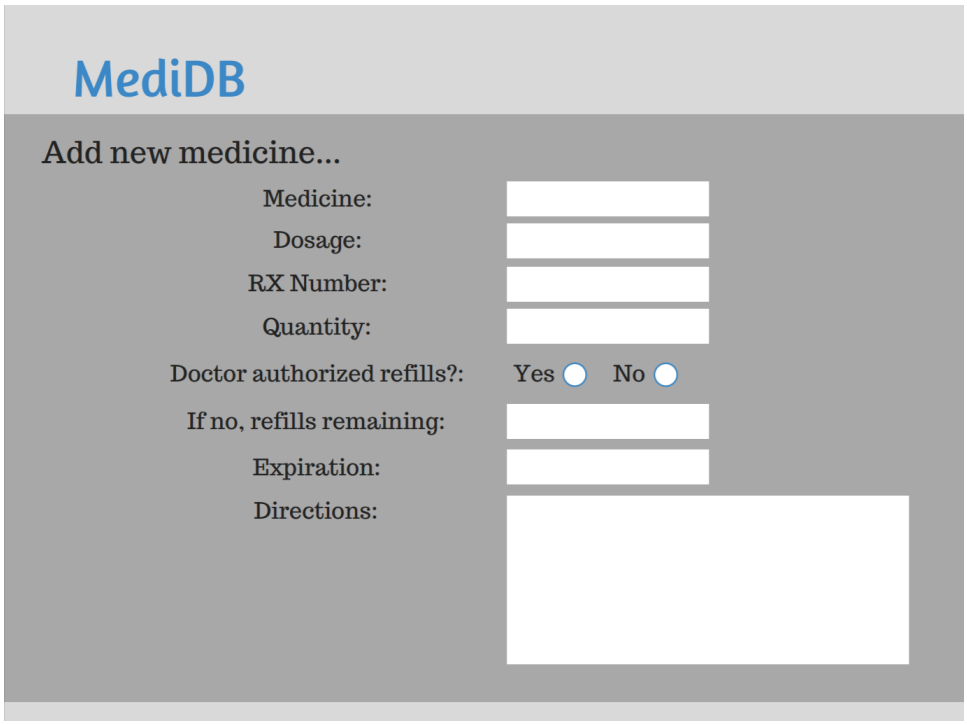
Mockup of Application



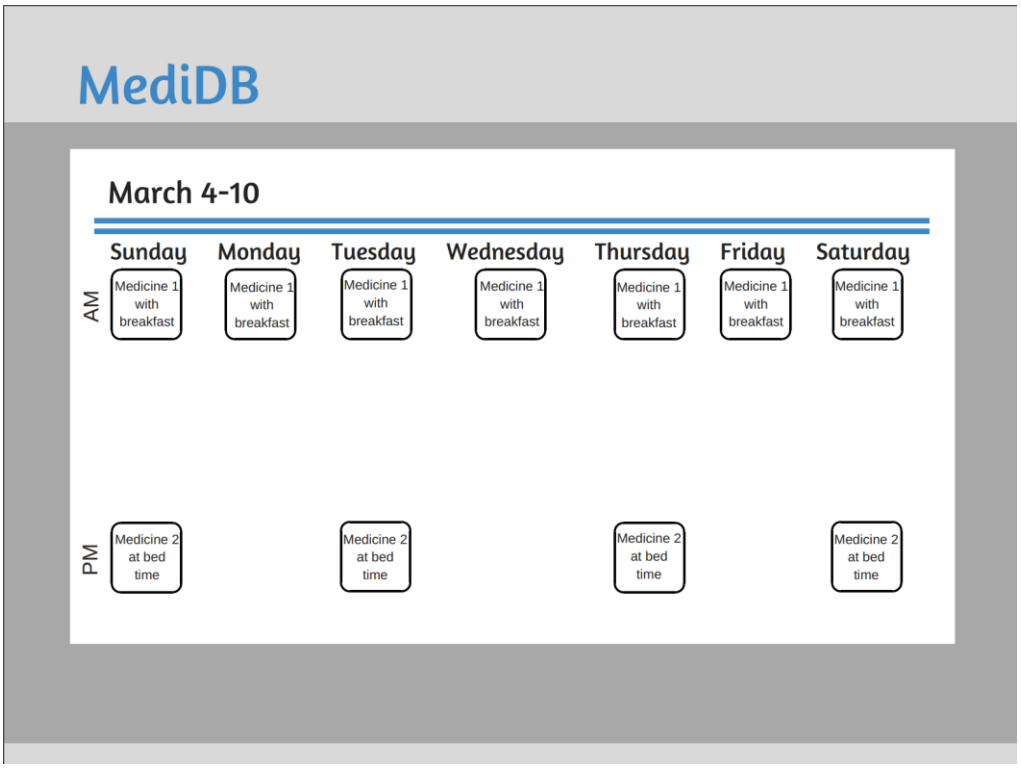
The main screen for the Patient.
They will be able to click on any of the circles to complete whichever task they wish.
The "Update" and "View" will have multiple options for the user to select from.
"Send" opens a pop-up where the user will be able to enter their providers email.
“Download” will download a printable PDF form for the user's monthly MediChart.
"Manage" will allow the user to turn on/off prescription refill notifications.



This screen shows the options associated with "Update" and "View".
To update, the patient can either add (+) a new medicine or chose to remove (-) one.
To view, the patient can choose between daily (D), weekly (W), or monthly (M) views.



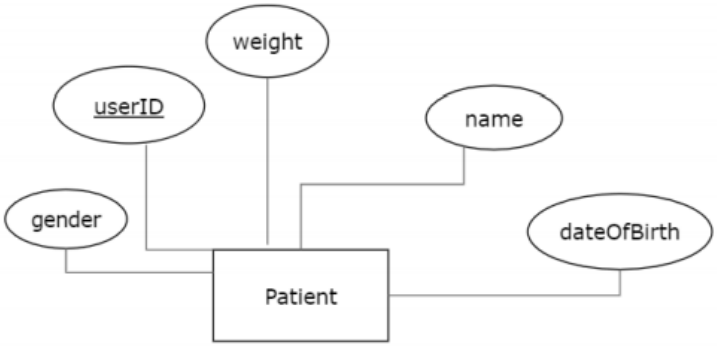
When the patient chooses to add a new medicine, they will be required to fill out the information above. This information will be entered into the patient's database and will be crossed referenced with other medications in the system. If there is an interaction found between the patients current and new medication, an alert will be sent via email.



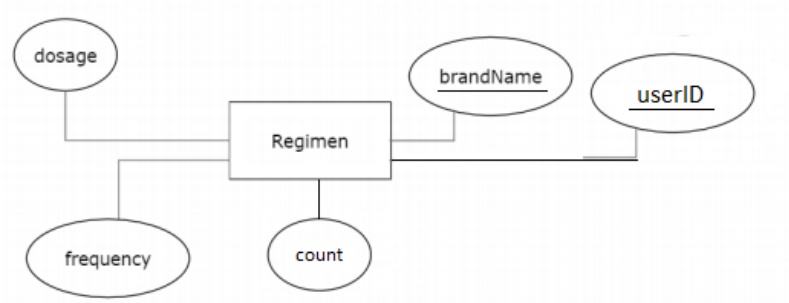
The patient's weekly MediChart would be shown as above. The daily and monthly view will be of similar format.

Design of Relations

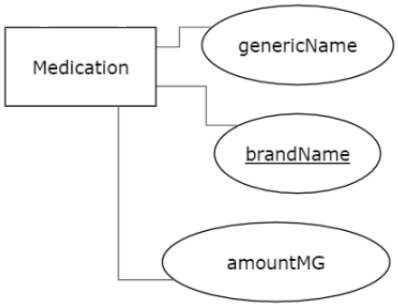
E/R Modeling



Each patient represents an entity in our database. The patient entity has a unique userID that is used as the key, along with other attributes directly related to the patient.

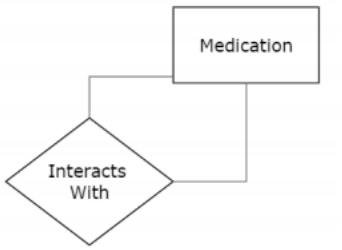


A regimen is made up of individual users and the medication they are taking. The medication, dosage, and frequency are all attributes of a medication regimen.



Medications are an entity in our database with the brand name of the medication acting as the key. A brand name is a trademarked name owned by the manufacture of the medication and is sufficiently unique to be used as a key. For example, there are many brand names for acetaminophen, but only one Tylenol. The other

attributes associated with this entity are the physical characteristics of the medication.



Medications interact with other medications from the medications table. This type of entity relationship needs no other attributes.

Local Relational Schemas

Patient Information Schema

Patients(userID, userPW, firstName, lastName, dateOfBirth, weight, gender)
The given relation meets 3rd Normal Form. Each patient is given a unique user ID to identify them. The patient's name is for personalizing the output to the user and is not to be used as a key. The patient's basic information also belongs to the patient table as they are attributes directly associated with each patient. This information is needed to find possible biometric interactions with medications.

Medication Regimen Schema

Regimen(userID, brandName, dosage, frequency, count)
The given relation meets 3rd Normal Form. If a patient wishes to see their medication regimen, they will expect to see the medication they are taking (brandName), how much they take (dosage), and how often they take it (frequency). This data stored along with the userID can be parsed into the schedule viewer for patients. The count represents the number of doses left in the bottle for the patient.

Medication Information Schema

Medications(brandName, genericName, amountMG)
The given relation meets 3rd Normal Form. Each medication will have specific information that pertains to that specific medication; including its trademarked brand name as a key, and a generic name for identification, and each pills amount of medication per pill in milligrams.

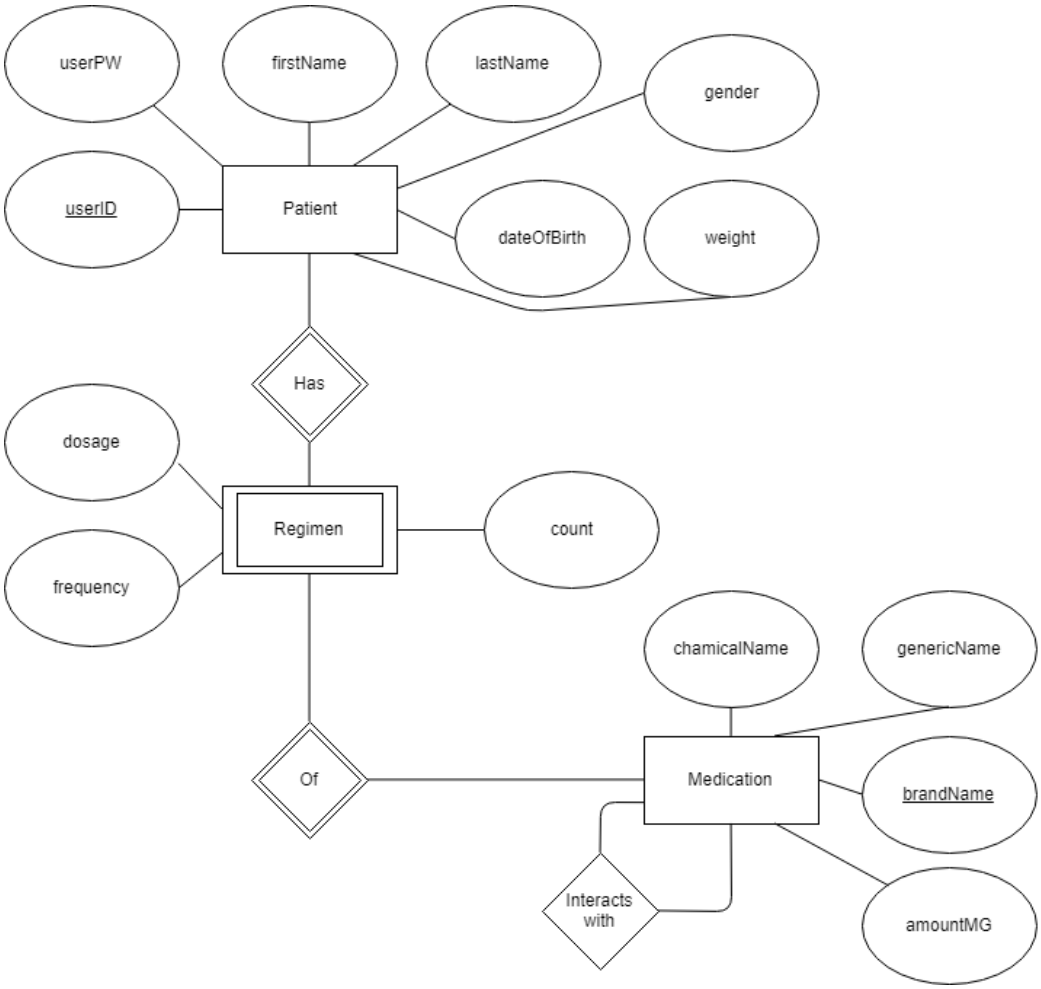
Drug Interaction Schema

Interactions(medication1, medication2)
The given relation meets 3rd Normal Form. Drug interactions should be captured when the patient tries adding an interacting medication to their regimen. In the

case that an interaction if found, it should be relayed to the patient and show them which medications are causing the issue.

Global Relational Schema

Patients(<u>userID</u>, userPW, firstName, lastName, dateOfBirth, weight, gender)
Regimen(<u>userID</u>, <u>brandName</u>, dosage, frequency, count)
Medications(<u>brandName</u>, genericName, amountMG)
Interactions(medication1, medication2)



Implementation of Database

The database for this project is hosted on a local instance of a MySQL Router. Initializing the database includes some initial data including a user and some medications added to the database. Each attribute was given a data-type each table had its keys and foreign keys selected, as well as setting the default results of a delete or update function.

Definition of SQL Database Schema

Database schema

```
-- MediDB Schema
-- James Trafny and Lana Adams
--


-- Clear DB
DROP DATABASE IF EXISTs MediDB;
CREATE DATABASE IF NOT EXISTS MediDB;
USE MediDB;
```

```
-- User/Patient table
-- Patients(userID, userPW, firstName, lastName, dateOfBirth, weight, gender)
CREATE TABLE Patients(
        userID  VARCHAR(50) PRIMARY KEY,
        userPW  VARCHAR(50),
        firstName VARCHAR(50),
        lastName VARCHAR(50),
        dateOfBirth DATE,
        weight FLOAT(6,2),
        gender CHAR(1)
    );
```

```
-- Medications table
-- Medications(brandName, genericName, amountMG)
CREATE TABLE Medications(
        brandName VARCHAR(50),
        genericName VARCHAR(50),
        amountMG FLOAT(6,2),
        PRIMARY KEY(brandName)
    );
```

```
-- Interactions table
-- Interactions(medication1, medication2)
CREATE TABLE Interactions(
        medication1 VARCHAR(50),
        medication2 VARCHAR(50),
        FOREIGN KEY (medication1) REFERENCES Medications(brandName)
        ON DELETE CASCADE,
        FOREIGN KEY (medication2) REFERENCES Medications(brandName)
        ON DELETE CASCADE
    );
```

```
-- Regimen table
-- Regimen(userID, brandName, dosage, frequency, count)
CREATE TABLE Regimen(
        userID VARCHAR(50),
        brandName VARCHAR(50),
        dosage FLOAT(6,2),
        frequency VARCHAR(3),
        count INTEGER,
        FOREIGN KEY (userID) REFERENCES Patients(userID)
        ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (brandName) REFERENCES Medications(brandName)
        ON DELETE CASCADE ON UPDATE CASCADE
    );
```

```
-- Initial Data
-- Medication
INSERT INTO medications VALUES
       ('Lipitor', 'Atorvastatin Calcium', 100),
       ('Synthroid', 'Levothyroxine', 100),
       ('Prinivil', 'Lisinopril', 100),
       ('Prilosec', 'Omeprazole', 20),
       ('Glucophage', 'Metformin', 32),
       ('Norvasc', 'Amlodipine', 100),
       ('Zocor', 'Simvastatin', 300),
       ('Lortab', 'Hydrocodone/Acetaminophen', 230),
       ('Toprol XL', 'Metoprolol ER', 225),
       ('Cozaar', 'Losartan', .5),
       ('Zithromax', 'Azithromycin', .5),
       ('Ambien', 'Zolpidem', 25),
       ('Microzide', 'Hydrochlorothiazide', 225),
       ('Lasix', 'Furosemide', 800),
       ('Lopressor', 'Metoprolol', 25),
       ('Protonix', 'Pantoprazole', 25),
       ('Neurontin', 'Gabapentin', 45),
       ('Amoxil', 'Amoxicillin', 800),
       ('Deltasone', 'Prednisone', 250),
       ('Zoloft', 'Sertraline', 100),
       ('Flomax', 'Tamsulosin', 10),
       ('Flonase', 'Fluticasone', 10),
       ('Pravachol', 'Pravastatin', 25),
       ('Ultram', 'Tramadol', 50),
       ('Singulair', 'Montelukast', 100),
       ('Lexapro', 'Escitalopram', 225),
       ('Coreg', 'Carvedilol', 300),
       ('Xanax', 'Alprazolam', 25),
       ('Coumadin', 'Warfarin', 50),
       ('Mobic', 'Meloxicam', 800),
       ('Plavix', 'Clopidogrel', 10),
       ('Augmentin XR', 'Amoxicillin/Potassium Clavulanate ER', 225),
       ('Zyloprim', 'Allopurinol', 125),
       ('Wellbutrin', 'Bupropion', 10),
       ('Zestoretic', 'Lisinopril/HCTZ', 5),
       ('Celexa', 'Citalopram', 1),
       ('Tenormin', 'Atenolol', 1000),
       ('Cymbalta', 'Duloxetine', 2000),
       ('Prozac', 'Fluoxetine', 12.5),
       ('Tricor', 'Fenofibrate', .5),
       ('Effexor', 'Venlafaxine', .8),
       ('Adderall', 'Amphetamine/Dextroamphetamine', 20),
       ('Flexeril', 'Cyclobenzaprine', 10),
       ('Medrol', 'Methylprednisolone', 25),
       ('Klor-Con', 'Potassium Chloride', 50),
       ('Tylenol', 'Acetaminophen', 325),
       ('Advil', 'Ibuprofin', 200);


 INSERT INTO Patients VALUES
       ('1337', 'pass', 'James', 'Trafny', '1988-03-24', 170.00, 'm'),
       ('1338', 'word', 'Lana', 'Adams', '1992-03-06', 170.00, 'f');
```

Sample Data

```
INSERT INTO regimen VALUES
        (1337, 'Ambien', 25, 'PRN', 30),
        (1337, 'Advil', 200, 'BID', 60),
        (1338, 'Adderall', 20, 'BID', 45),
        (1338, 'Amoxil', 800, 'BID', 14);
```

```
INSERT INTO patients VALUES
        (0001, 1234, 'Ada', 'Lovelace', '1815-12-10', 155, 'f'),
        (0002, 1234, 'Alan', 'Turing', '1927-06-23', 150, 'm'),
        (0003, 1234, 'Linus', 'Torvalds', '1969-12-28', 195, 'm'),
        (0004, 1234, 'Steve', 'Wozniak', '1950-08-11', 245, 'm');
```

```
INSERT INTO interactions VALUES
        ('Tylenol', 'Advil'),
        ('Augmentin XR', 'Pravachol'),
        ('Prozac', 'Synthroid'),
        ('Zithromax', 'Ultram');
```

Schema Test Results

SELECT * FROM medidb.regimen;

Result Grid | Filter Rows: | Exp

userID	brandName	dosage	frequency	count
1337	Ambien	25.00	PRN	30
1337	Advil	200.00	BID	60
1338	Adderall	20.00	BID	45
1338	Amoxil	800.00	BID	14

SELECT * FROM medidb.patients;

Result Grid | Filter Rows: | Edit: | Expor

userID	userPW	firstName	lastName	dateOfBirth	weight	gender
1	1234	Ada	Lovelace	1815-12-10	155.00	f
1337	pass	James	Trafny	1988-03-24	170.00	m
1338	word	Lana	Adams	1992-03-06		
2	1234	Alan	Turing	1927-06-23		
3	1234	Linus	Torvalds	1969-12-28	195.00	m
4	1234	Steve	Wozniak	1950-08-11	245.00	m
NULL	NULL	NULL	NULL	NULL	NULL	NULL

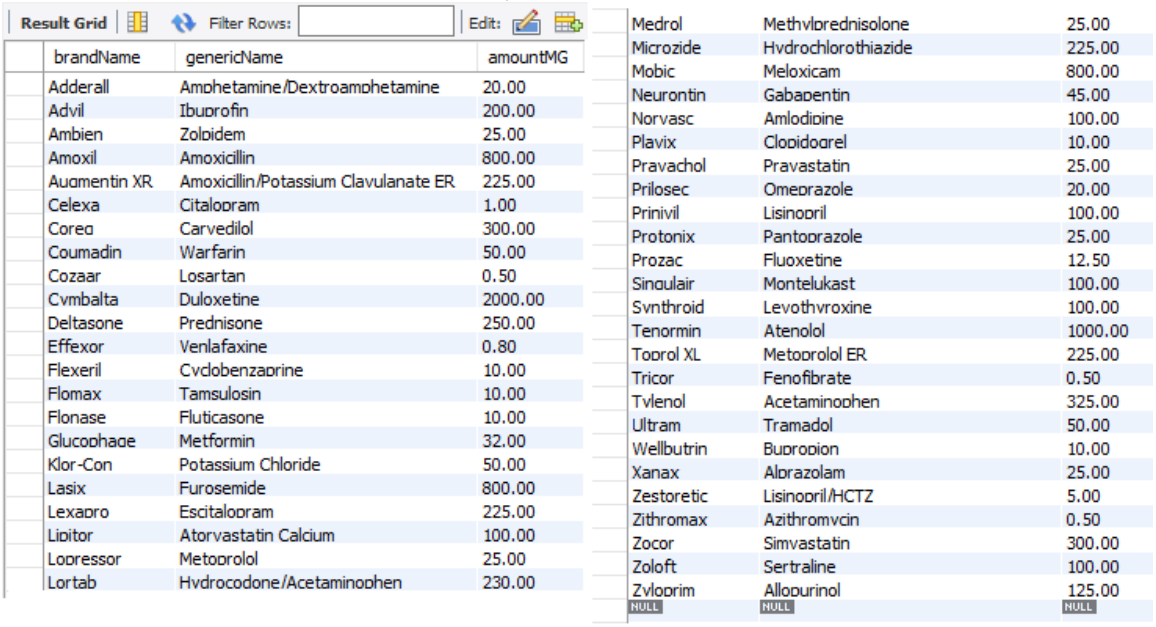
1992-03-06

SELECT * FROM medidb.interactions;

Result Grid | Filter Rows:

medication1	medication2
Tylenol	Advil
Augmentin XR	Pravachol
Prozac	Synthroid
Zithromax	Ultram

SELECT * FROM medidb.medications;

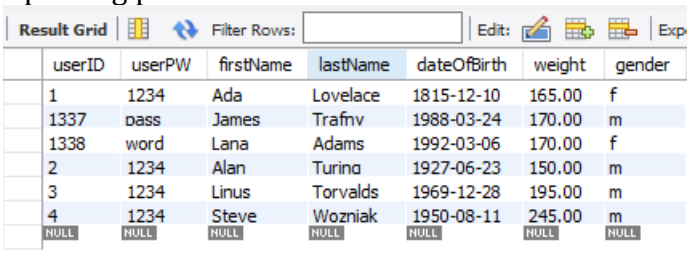


brandName	genericName	amountMG
Adderall	Amphetamine/Dextroamphetamine	20.00
Advil	Ibuprofin	200.00
Ambien	Zolpidem	25.00
Amoxil	Amoxicillin	800.00
Augmentin XR	Amoxicillin/Potassium Clavulanate ER	225.00
Celexa	Citalopram	1.00
Coreg	Carvedilol	300.00
Coumadin	Warfarin	50.00
Cozaar	Losartan	0.50
Cymbalta	Duloxetine	2000.00
Deltasone	Prednisone	250.00
Effexor	Venlafaxine	0.80
Flexeril	Cyclobenzaprine	10.00
Flomax	Tamsulosin	10.00
Flonase	Fluticasone	10.00
Glucophage	Metformin	32.00
Klor-Con	Potassium Chloride	50.00
Lasix	Furosemide	800.00
Lexapro	Escitalopram	225.00
Lipitor	Atorvastatin Calcium	100.00
Lopressor	Metoprolol	25.00
Lortab	Hydrocodone/Acetaminophen	230.00
Medrol	Methylprednisolone	25.00
Microzide	Hydrochlorothiazide	225.00
Mobic	Meloxicam	800.00
Neurontin	Gabapentin	45.00
Norvasc	Amlodipine	100.00
Plavix	Clopidogrel	10.00
Pravachol	Pravastatin	25.00
Prilosec	Omeprazole	20.00
Prinivil	Lisinopril	100.00
Protonix	Pantoprazole	25.00
Prozac	Fluoxetine	12.50
Singulair	Montelukast	100.00
Synthroid	Levothyroxine	100.00
Tenormin	Atenolol	1000.00
Toprol XL	Metoprolol ER	225.00
Tricor	Fenofibrate	0.50
Tylenol	Acetaminophen	325.00
Ultram	Tramadol	50.00
Wellbutrin	Bupropion	10.00
Xanax	Alprazolam	25.00
Zestoretic	Lisinopril/HCTZ	5.00
Zithromax	Azithromycin	0.50
Zocor	Simvastatin	300.00
Zoloft	Sertraline	100.00
Zyloprim	Allopurinol	125.00

Database Operations and Testing

```
UPDATE Patients
SET weight = 165.00
WHERE userID = 1;
```

Updating patient information



userID	userPW	firstName	lastName	dateOfBirth	weight	gender
1	1234	Ada	Lovelace	1815-12-10	165.00	f
1337	pass	James	Trafny	1988-03-24	170.00	m
1338	word	Lana	Adams	1992-03-06	170.00	f
2	1234	Alan	Turing	1927-06-23	150.00	m
3	1234	Linus	Torvalds	1969-12-28	195.00	m
4	1234	Steve	Wozniak	1950-08-11	245.00	m

```
UPDATE Regimen
SET count = 31
WHERE userID = 1337
AND brandName = 'Advil';
```

Updating pill count

Result Grid | Filter Rows: | Exp

userID	brandName	dosage	frequency	count
1337	Ambien	25.00	PRN	30
1337	Advil	200.00	BID	31
1338	Adderall	20.00	BID	45
1338	Amoxil	800.00	BID	14

Add new medicine

```
INSERT INTO Regimen VALUES
        (1337, 'Flonase', 10, 'PRN', 60);
```

Result Grid | Filter Rows: | Exp

userID	brandName	dosage	frequency	count
1337	Ambien	25.00	PRN	30
1337	Advil	200.00	BID	31
1338	Adderall	20.00	BID	45
1338	Amoxil	800.00	BID	14
1337	Flonase	10.00	PRN	60

```
DELETE FROM Regimen
WHERE userID = 1337
AND brandName = 'Flonase';
```

Deleting a medicine from the user's regimen

Result Grid | Filter Rows: | Exp

userID	brandName	dosage	frequency	count
1337	Ambien	25.00	PRN	30
1337	Advil	200.00	BID	31
1338	Adderall	20.00	BID	45
1338	Amoxil	800.00	BID	14

Checking for Interactions

```
SELECT * FROM Interactions
WHERE medication1 = 'Tylenol'
OR medication2 = 'Tylenol';
```

Result Grid | Filter Rows

medication1	medication2
Tylenol	Advil

Application Implementation

MediDB is a graphical user program designed to manage a user's medication regimen. The application was built in Java, using MySQL as the database back end. The database was generated using an SQL script to instantiate an initial set of users and list of medications. An instance of the database must be running on the user's local machine. The class files making up the Java application are not compiled, because the user (Dr. Stansbury) will need to edit the username and password in the class files to run it on his machine.

When the application is running (from MediChartGUI) the user is presented with a log-in screen. Here, the user can create a new user, or remove one from the system. Entering a valid username and password will let the user into their management screen, an invalid username/password combination will allow the user to re-try, but not log them in.

In the management screen, the user can view their current medication regimen, or make changes to their regimen. When editing their own regimen, the user's choices are limited by a drop-down menu that only displays medications currently in their regimen. If the user wants to add a medication to their regimen, the system limits the users choices with a drop-down menu to only medication that are in the database system. If the user wishes to view their regimen, they are presented with a list of all the medications they are currently using. The user can also edit medications in their regimen to update the pill count of each.

createAccount.java

Description	This is the JavaFX file that manages the view for creating a new user in the system.
User Input	User ID (4 digit integer) Password (user defined) First Name (string) Last Name (string) Date Of Birth (yyyy-mm-dd) Weight (double) Gender (string)
User Output	n/a
Database Operations	Through MediDB.java this class takes in the user input to create a new row in the Patient table using addNewPatient().

manageRXPopup.java

Description	This is the JavaFX file that manages the view for updating the count for the selected medicine.

User Input	Brand Name (drop down menu) Count (integer)
User Output	n/a
Database Operations	Through MediDB.java this class updates the pill count information in the Regimens table using addNewPatient()..

MediChartGUI.java

Description	This is the javaFX file that is responsible for launching the entire program
User Input	User ID (4 digit integer) Password (string)
User Output	n/a
Database Operations	Through MediDB.java this class checks the password stored with the given user ID and allows the user to log in if they match using logIn(). If the password is incorrect, it resets the password field for the user to try again.

MediDB.java

Description	This file is responsible for interacting with the database. It contains all methods that require SQL for the GUI to operate.
User Input	Handled via GUI.
User Output	Boolean values that indicate success of failure of SQL operations. Array Lists used to display information to the user.
Database Operations	logIn() - responsible for validating user login addNewPatient() - creates a new row in Patient table deletePatient() - removes row from Patient table addToRegimen() - creates a new row in Regimen table removeFromRegimen() - removes row from Regimen table getFullRegimen() - returns an Array List of user's Regimen table rows getPartRegimen() - returns an Array List of Medications within the Regimen table with matching user ID updatePillCount() - updates the count of the medicine listed in Regimen table with matching user ID getMedicationList() - returns an Array List of all Medications

medidbschema.sql

Description	This file defines the schema used by MediDB.java
User Input	n/a
User Output	n/a
Database Operations	Used by MySQL to create the database.

removeMedicine.java

Description	This GUI file shows the menu for removing a medication from the current users regimen. The drop-down box only allows the user to select medications that are currently in their regimen.
User Input	Medication to remove (via drop-down menu).
User Output	None.
Database Operations	Using a call to MediDB, this file is invoking a DELETE operation on the regimens table on a row matching the two keys needed (userID and brandName).

sendPopUp.java

Description	This file is not working in the current form of the application, it would be used for sending the users regimen via email to a recipient.
User Input	String of recipients email address.
User Output	An email sent to the recipient.
Database Operations	This file would need to go through MediDB.java to get the current users regimen, but this file is not implemented.

updatePopUp.java

Description	This pop up happens after the user clicks the “Update Medicine” button, and presents the user with two options, add or remove a medication from their regimen.
User Input	Button press, add or remove.
User Output	New view.
Database Operations	None

updateUserInput.java

Description	This file is responsible for displaying the screen to add a new medication to the user’s regimen.
User Input	The user will have a drop-down menu to select which medication they wish to add to their regimen, along with the dosage,

	frequency, and cardinality.
User Output	None
Database Operations	Using a call to MediDB.java, the INSERT operation is used to add a new row to the regimens table. For this table, each row has a unique userID and brandName associated as the keys. brandName is a foreign key that must be present in the medications table. The JavaFX menu for selecting which medication to add should prevent the user from entering something that is not in the medications table.

userHome.java

Description	This JavaFX file presents the user with a main screen, where they can use the buttons on the screen to interact with the system.
User Input	Button press events.
User Output	New screens and views after button presses.
Database Operations	None

viewRegimen.java

Description	This JavaFX class is used to display the users current medication regimen.
User Input	None
User Output	A list of medications currently in the users regimen, along with the frequency and dosage.
Database Operations	This file makes a call to MediDB.java, getFullRegimen(userID), and gets an ArrayList in return containing strings representing the rows of the regimen table with the current user's ID as the primary key.

Conclusion

The functionality in this version of MediDB demonstrates most of the SQL functionality we intended to when starting this project; including adding and removing users, updating user specific regimens, updating user specific medications, and maintaining a cross reference to a master list of medications. The GUI also has several non-functioning buttons that indicate where we would spend more time developing functionality. These buttons include an option to send your medication regimen to an email address, and to download their regimen as a PDF.

The main SQL functionality that we would like to add is a cross-reference that checks for drug interactions. This was a functional requirement that we set out to achieve but did not have the time to implement. An interactions table was created but not filled out, and not referenced in the Java code.

On the Java side, we would have liked to display the patient's regimen in a more user-friendly way, by displaying it as a weekly calendar instead of just listing it out. A menu bar would also be appropriate for this kind of application, but the functionality was difficult to implement in the short time period. There is also a problem in the design of our Java implementation, namely an SQL connection object gets instantiated in each class that needs a connection. While our approach technically works, a more appropriate design pattern would be the singleton design pattern. The way our system is developed now increases the coupling of our system by a large factor and makes maintenance of SQL calls more difficult.

References

[1] Center for Disease Control. *Prescription drug use in the past 30 days, by sex, race and Hispanic origin, and age: United States, selected years 1988-1994 through 2009-2012*[PDF]. Clinfton, GA: 2016

Appendix A: Source File Listing

createAccount.java

```
/**
 * Lana Adams, James Trafny
 * CS 317 - Final project
 *
 * This application is designed to store and manage patient medicine information.
 *
 * This is the pop up which will allow the user to create an account
 */

import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class createAccount {



	public static boolean display(String title) {
		String url = "jdbc:mysql://localhost:3306/medidb?useSSL=false";
		String user = "root";
		String pswd = "BlueJean1018";
		MediDB mediDB = new MediDB(url, user, pswd);

		Stage window = new Stage();
		window.initModality(Modality.APPLICATION_MODAL);
		window.setTitle(title);
		window.setMinWidth(500);
		window.setMinHeight(600);

		//Create label, text field, buttons
		Label userLabel = new Label("User ID (####):");
		TextField userID = new TextField();
		Label pwLabel = new Label("Password:");
		TextField pwInput = new TextField();
		Label fnLabel = new Label("First Name: ");
		TextField fnInput = new TextField();
		Label lnLabel = new Label("Last Name: ");
		TextField lnInput = new TextField();
		Label dobLabel = new Label("Date Of Birth (YYYY-MM-DD): ");
		TextField dobInput = new TextField();
		Label weightLabel = new Label("Weight: ");
		TextField weightInput = new TextField();
		Label genderLabel = new Label("Gender (m/f): ");
		TextField genderInput = new TextField();



		Button okBt = new Button("OK");

		GridPane gridPane = new GridPane();
		gridPane.setVgap(20);
		gridPane.setHgap(20);

		//Add buttons
```

```
                gridPane.add(userLabel, 0, 0);
                gridPane.add(userID, 1, 0);
                gridPane.add(pwLabel, 0, 1);
                gridPane.add(pwInput, 1, 1);
                gridPane.add(fnLabel, 0, 2);
                gridPane.add(fnInput, 1, 2);
                gridPane.add(lnLabel, 0, 3);
                gridPane.add(lnInput, 1, 3);
                gridPane.add(dobLabel, 0, 4);
                gridPane.add(dobInput, 1, 4);
                gridPane.add(weightLabel, 0, 5);
                gridPane.add(weightInput, 1, 5);
                gridPane.add(genderLabel, 0, 6);
                gridPane.add(genderInput, 1, 6);
                gridPane.add(okBt, 1, 7);
                gridPane.setAlignment(Pos.CENTER);
                Scene scene = new Scene(gridPane);
                scene.getStylesheets().add("stylesheetMDBpopup.css");
                window.setScene(scene);
                window.show();









                okBt.setOnAction(e -> {
                        int user_ID = Integer.parseInt(userID.getText());
                        int weight = Integer.parseInt(weightInput.getText());
                        mediDB.addNewPatient(user_ID, pwInput.getText(),
fnInput.getText(), lnInput.getText(), dobInput.getText(), weight,
genderInput.getText());
                        window.close();
                });




                return true;


        }

}
```

manageRXPopup.java

```
/**
 * Lana Adams, James Trafny
 * CS 317 - Final project
 *
 * This application is designed to store and manage patient medicine information.
 *
 * This is the pop up which will allow the user update the count of their
 * medicine by brand name
 */
import java.util.ArrayList;
import java.util.prefs.Preferences;

import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class manageRXPopup {

	public static boolean display(String title) {

		String url = "jdbc:mysql://localhost:3306/medidb?useSSL=false";
		String user = "root";
		String pswd = "BlueJean1018";
		MediDB mediDB = new MediDB(url, user, pswd);

		Stage window = new Stage();
	 window.initModality(Modality.APPLICATION_MODAL);
		window.setTitle(title);
		window.setMinWidth(500);
		window.setMinHeight(300);

		ChoiceBox<String> choiceBox = new ChoiceBox<>(); //brandname drop down

		//Create label, text field, buttons
		Label medLabel = new Label("Medication (Brand Name): ");
		//TextField brandName = new TextField();
		Preferences up = Preferences.userRoot();
		int user_id = 0;
		int userid = up.getInt("userID", user_id);
		ArrayList<String> results = mediDB.getPartRegimen(userid);
		for (String med:results) {
			choiceBox.getItems().add(med);
		}

		Label countLabel = new Label("Count: ");
		TextField count = new TextField();
		Button okBt = new Button("OK");

		GridPane gridPane = new GridPane();
		gridPane.setVgap(20);
		gridPane.setHgap(20);

		//Add buttons
		gridPane.add(choiceBox, 1,           0);
		gridPane.add(okBt, 1, 2);
		gridPane.add(medLabel, 0, 0);
		gridPane.add(countLabel, 0, 1);
		gridPane.add(count, 1, 1);
		gridPane.setAlignment(Pos.CENTER);
```

```
                Scene scene = new Scene(gridPane);
        scene.getStylesheets().add("stylesheetMDBpopup.css");
                window.setScene(scene);
                window.show();




                okBt.setOnAction(e -> {
                        int newCount = Integer.parseInt(count.getText());
                        mediDB.updatePillCount(newCount, userid, choiceBox.getValue());
                        window.close();
                });

                return true;

}
}
```

MediChartGUI.java

```
/**
 * Lana Adams, James Trafny
 * CS 317 - Final project
 *
 * This application is designed to store and manage patient medicine information.
 *
 * This is the login stage for the application, the user
 * will enter his/her username and password, then click log in
 * to continue to the home page.
 */
import java.util.prefs.Preferences;

import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Pos;
import javafx.scene.Node;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.control.Tooltip;
import javafx.scene.image.Image;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundImage;
import javafx.scene.layout.BackgroundPosition;
import javafx.scene.layout.BackgroundRepeat;
import javafx.scene.layout.BackgroundSize;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.GridPane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;

public class MediChartGUI extends Application{

	String url = "jdbc:mysql://localhost:3306/medidb?useSSL=false";
	String user = "root";
	String pswd = "BlueJean1018";
	MediDB mediDB = new MediDB(url, user, pswd);


	//gridpane for the login
	private GridPane loginPane;

	//text fields for user inputs
	static TextField userID = new TextField();
	/**
	 * creates a TextField() to allow user to input password
	 */
	static PasswordField userPassword = new PasswordField();

	/**
	 * button to login
	 */
	private Button loginBt;

	/**
	 * button to get help
	 */
	private Button helpBt;

	/**
```

```
         * button to create account
         */
        private Button createAccountBt;

        /**
         * label to show user where to input password
         */
        private Label pwLabel;

        /**
         * label to show user where to input email

         */
        private Label userLabel;

        public MediChartGUI() {
                //instantiate the login pane
                loginPane = new GridPane();
                loginPane.setVgap(20);
                loginPane.setHgap(20);


                //login labels
                pwLabel = new Label("Password:");
                userLabel = new Label("User Name:");


                //instantiate the buttons and text fields
                loginBt = new Button("Login");
                helpBt = new Button("Help");
                createAccountBt = new Button("Create Account");
                //ToolTips for buttons
                Tooltip tt1 = new Tooltip("Click to log in");
                Tooltip.install(loginBt, tt1);
                Tooltip tt2 = new Tooltip("Click for more information");
                Tooltip.install(helpBt, tt2);
                //layout
                loginPane.setAlignment(Pos.CENTER);
                loginPane.setOpacity(100);
                loginPane.add(userLabel , 0, 0);
                loginPane.add(userID, 1, 0);
                loginPane.add(pwLabel, 0, 1);
                loginPane.add(userPassword, 1, 1);
                loginPane.add(loginBt, 1, 3);
                loginPane.add(helpBt, 1, 5);
                loginPane.add(createAccountBt, 1, 4);
        }

        public void start(Stage MediChartGUI) {

                //Application root
                BorderPane borderPane = new BorderPane();
                borderPane.setCenter(loginPane);


                //Button Handlers
                helpBt.setOnAction(new EventHandler<ActionEvent>() {
                        public void handle(ActionEvent e) {
                                //Display help window
                                Alert helping = new Alert(AlertType.INFORMATION);
                                helping.setTitle("MediDB Help");
                                helping.setHeaderText("Enter your 4-digit User ID, your
password, and click Login");
                                helping.showAndWait();

                        }
                });


                //Application layout
```

```
                Scene scene = new Scene(borderPane, 500,500);
         scene.getStylesheets().add("stylesheetMDB.css");
                MediChartGUI.setTitle("MediDB"); // Set the stage title
                MediChartGUI.setScene(scene); // Place the scene in the stage
                MediChartGUI.show(); // Display the stage



////////////////////////////////////////////////////////////////////////////
                //login button
                userHome userHome = new userHome();


                loginBt.setOnAction(new EventHandler<ActionEvent>() {
                        public void handle(ActionEvent e) {
                                try {
                                        //catch input
                                        String userName = userID.getText();
                                        int user_id = Integer.parseInt(userName);
                                        String password = userPassword.getText();
                                        boolean loginAttempt = mediDB.logIn(user_id,
password);
                                        if (loginAttempt) {
                                                Preferences up = Preferences.userRoot();
                                                up.putInt("userID", user_id);
                                                userHome.start(MediChartGUI);

                                        }else {
                                                userPassword.clear();
                                        }

                                } catch (Exception e1) {
                                        // TODO Auto-generated catch block

                                }
                        }});

                createAccountBt.setOnAction(e -> {
                boolean result = createAccount.display("Create New Account");
        });

        }

        public static void main(String[] args) {
                launch(args);
        }

}
```

MediDB.java

```
import java.sql.*;
import java.sql.Date; //explicit import to disambiguate util.Date and sql.Date
import java.util.*;

public class MediDB {

        Connection sql;

        public MediDB(String url, String user, String pswd) {

                /** Creates an instance of the database connection **/
                try {
                        this.sql = DriverManager.getConnection(url, user, pswd);
                } catch (SQLException e) {
```

```
                        System.out.println(e.getMessage()); // Handle exceptions
                }
        }

        /*
         * Verify log in credentials
         */
        public boolean logIn(int userID, String userPW) {
                boolean success = false;
                try {
                        // Define the query as a string
                        String logInQuery = "SELECT userPW FROM patients WHERE userID =
?";

                        // Request a Statement object from SQL class
                        PreparedStatement pstmt = sql.prepareStatement(logInQuery);
                        pstmt.setInt(1, userID);

                        // Execute the query
                        ResultSet results = pstmt.executeQuery();

                        // Compare given pass to actual pass
                        while (results.next()) {
                                String correctPW = results.getString("userPW");
                                if (correctPW.equals(userPW)) {
                                        // System.out.println("Success");
                                        success = true;
                                } else {
                                        // System.out.println("Fail");
                                        success = false;
                                }
                        }
                } catch (SQLException e) {
                        System.out.println(e.getMessage()); // Handle exceptions
                }
                // System.out.println(success);
                return success;
        }

        /*
         * Add a new user to the database
         */
        public boolean addNewPatient(int userID, String userPW, String firstName, String
lastName, String dateOfBirth,
                        double weight, String gender) {
                boolean success = false;

                try {

                        // Create insert query string
                        String insertString = "INSERT INTO patients VALUES (?, ?, ?, ?,
?, ?, ?)";

                        // Create prepared statement for insert and assign it the values.
                        PreparedStatement insertStatement =
sql.prepareStatement(insertString);
                        insertStatement.setInt(1, userID);
                        insertStatement.setString(2, userPW);
                        insertStatement.setString(3, firstName);
                        insertStatement.setString(4, lastName);
                        insertStatement.setDate(5, Date.valueOf(dateOfBirth));
                        insertStatement.setDouble(6, weight);
                        insertStatement.setString(7, gender);

                        // Perform the insert and confirm success.
                        int rows = insertStatement.executeUpdate();
                        success = true;
                        if (rows != 1) {
                                System.out.println("ALERT: Insertion failed.");
                                success = false;
```

```
                        }

                } catch (SQLException e) {
                        System.out.println(e.getMessage()); // Handles exceptons.
                }
                return success;
        }

        /*
         * Remove a user to the database, also removes them from regimen
         */
        public boolean deletePatient(int userID) {
                boolean success = false;

                // Prepare the delete statement strings with ? for the key attributes of
                // title and year
                String deletePatientString = "DELETE FROM patients WHERE userID = ?";
                String deleteRegimenString = "DELETE FROM regimen WHERE userID = ?";

                // Declare prepared statement variables.
                PreparedStatement deletePatientStmt;
                PreparedStatement deleteRegimenStmt;

                try {

                        // Start Transaction by Setting Auto Commit to False
                        // Note: we must re-enable auto-commit when we are done to
restore
                        // the system to the default
                        // state.
                        sql.setAutoCommit(false);

                        // Perform the first delete by preparing and executing the
statement
                        deletePatientStmt = sql.prepareStatement(deletePatientString);
                        deletePatientStmt.setInt(1, userID);
                        deletePatientStmt.executeUpdate();

                        // Perform the second delete by preparing and executing the
                        // statement.
                        deleteRegimenStmt = sql.prepareStatement(deleteRegimenString);
                        deleteRegimenStmt.setInt(1, userID);
                        deleteRegimenStmt.executeUpdate();

                        // Commit the change to make the change live to the database.
                        sql.commit();
                        success = true;

                } catch (SQLException e) {

                        // If an exception occurs, we will roll back the transaction to
                        // avoid having an error state.

                        try {
                                System.out.println("Rolling Back Transaction.");

                                // Performs the roll back.
                                sql.rollback();
                                success = false;
                        } catch (SQLException e2) {
                                System.out.println(e2.getMessage()); // Handle exception
                        }

                } finally {

                        // Finally is called after the try ends by reaching the end of
its
                        // code or from a return statement.
                        // We include it here so that we can reenable autocommit and take
it
```

```
                        // out of transaction mode.

                        try {
                                // Re-enable autocommit
                                sql.setAutoCommit(true);
                        } catch (SQLException e) {
                                System.out.println(e.getMessage());
                        }
                }
                return success;
        }


        /*
         * Add to user regimen
         */
        public boolean addToRegimen(int userID, String brandName, double dosage, String
frequency, int count) {
                boolean success = false;
                try {

                        // Create insert query string
                        String insertString = "INSERT INTO regimen VALUES (?, ?, ?, ?,
?)";

                        // Create prepared statement for insert and assign it the values.
                        PreparedStatement insertStatement =
sql.prepareStatement(insertString);
                        insertStatement.setInt(1, userID);
                        insertStatement.setString(2, brandName);
                        insertStatement.setDouble(3, dosage);
                        insertStatement.setString(4, frequency);
                        insertStatement.setInt(5, count);

                        // Perform the insert and confirm success.
                        int rows = insertStatement.executeUpdate();
                        success = true;
                        if (rows != 1) {
                                System.out.println("ALERT: Insertion failed.");
                                success = false;
                        }

                } catch (SQLException e) {
                        System.out.println(e.getMessage()); // Handles exceptons.
                }
                return success;
        }

        /*
         * Remove a medication from user regimen
         */
        public boolean removeFromRegimen(int userID, String brandName) {
                boolean success = false;

                // Prepare the delete statement strings with ? for the key attributes of
                // title and year
                String deleteMedString = "DELETE FROM regimen WHERE userID = ? AND
brandName = ?";

                // Declare prepared statement variables.
                PreparedStatement deleteMedStmt;

                try {

                        // Start Transaction by Setting Auto Commit to False
                        // Note: we must re-enable auto-commit when we are done to
restore
                        // the system to the default
                        // state.
                        sql.setAutoCommit(false);
```

```
                        // Perform the first delete by preparing and executing the
statement
                        deleteMedStmt = sql.prepareStatement(deleteMedString);
                        deleteMedStmt.setInt(1, userID);
                        deleteMedStmt.setString(2, brandName);
                        deleteMedStmt.executeUpdate();

                        // Commit the change to make the change live to the database.
                        sql.commit();
                        success = true;

                } catch (SQLException e) {
                        // If an exception occurs, we will roll back the transaction to
                        // avoid having an error state.
                        try {
                                System.out.println("Rolling Back Transaction.");
                                // Performs the roll back.
                                sql.rollback();
                                success = false;
                        } catch (SQLException e2) {
                                System.out.println(e2.getMessage()); // Handle exception
                        }

                } finally {
                        // Finally is called after the try ends by reaching the end of
its
                        // code or from a return statement.
                        // We include it here so that we can reenable autocommit and take
it
                        // out of transaction mode.
                        try {
                                // Re-enable autocommit
                                sql.setAutoCommit(true);
                        } catch (SQLException e) {
                                System.out.println(e.getMessage());
                        }
                }
                return success;
        }

        /*
         * Get user's regimen table
         */
        public ArrayList<String> getFullRegimen(int userID) {
                ArrayList<String> fullResult = new ArrayList<String>();

                try {
                        // Define the query as a string
                        String query = "SELECT * FROM regimen WHERE userID = ?";

                        // Prepare a Statement object from SQL class
                        PreparedStatement stmt = sql.prepareStatement(query);
                        stmt.setInt(1, userID);

                        // Execute the query
                        ResultSet results = stmt.executeQuery();

                        // Print each record
                        while (results.next()) {

                                // Regimen(int userID, String brandName, FLOAT(6.2)
dosage,
                                // String frequency, int count)
                                //
                                // ResultSet also has get methods for each data type
based upon
                                // the index of the returned item
                                // in the tuple. Please note that the indexing starts at
1 and
```

```
                                // not zero.
                                String partResult = new String(

results.getString("brandName") + "    " +

results.getFloat("dosage") +  "    " +

results.getString("frequency") + "    " +

results.getInt("count"));
                                fullResult.add(partResult);
                        }
                } catch (SQLException e) {
                        System.out.println(e.getMessage()); // Handle exceptions
                }
                return fullResult;
        }

        /*
         * Get user's medication names from regimen
         */
        public ArrayList<String> getPartRegimen(int userID) {
                ArrayList<String> patientMeds = new ArrayList<String>();

                try {
                        // Define the query as a string
                        String query = "SELECT brandName FROM regimen WHERE userID = ?";

                        // Prepare a Statement object from SQL class
                        PreparedStatement stmt = sql.prepareStatement(query);
                        stmt.setInt(1, userID);

                        // Execute the query
                        ResultSet results = stmt.executeQuery();

                        // Print each record
                        while (results.next()) {
                                //Put results into array list for javafx
                         patientMeds.add(results.getString("brandName"));
                        }
                } catch (SQLException e) {
                        System.out.println(e.getMessage()); // Handle exceptions
                }
                return patientMeds;
        }

        /*
         * Increment or decrement pill count in a user regimen
         */
        // pub

        /*
         * Update pill count
         */
        public boolean updatePillCount(int newCount, int userID, String brandName) {
                boolean success = false;
                try {
                        // Query String
                        String query = "UPDATE regimen SET count = ? WHERE userID = ? AND
brandName = ?";

                        // Prepared Statement
                        PreparedStatement pstmt = sql.prepareStatement(query);
                        pstmt.setInt(1, newCount);
                        pstmt.setInt(2, userID);
                        pstmt.setString(3, brandName);

                        // Execute the update
                        pstmt.executeUpdate();
                        success = true;
```

```
                } catch (SQLException e) {
                        System.out.println(e.getMessage()); // Handles exception
                }
                return success;
        }

        /*
         * Print out all medications in the system
         */
        public ArrayList<String> getMedicationList() {
                ArrayList<String> meds = new ArrayList<String>();
                try {
                        // Define the query as a string
                        String query = "SELECT * FROM medications";

                        // Request a Statement object from SQL class
                        Statement stmt = sql.createStatement();

                        // Execute the query
                        ResultSet results = stmt.executeQuery(query);

                        // Print each record
                        while (results.next()) {

                                // Medications(VARCHAR(50) brandName, VARCHAR(50)
genericName,
                                // FLOAT(6.2) amountMG)
                                //
                                // ResultSet also has get methods for each data type
based upon
                                // the index of the returned item
                                // in the tuple. Please note that the indexing starts at
1 and
                                // not zero.
                           meds.add(results.getString("brandName"));
                        }
                } catch (SQLException e) {
                        System.out.println(e.getMessage()); // Handle exceptions
                }
                return meds;
        }
}
```

medidbschema.sql

```
-- MediDB Schema
-- James Trafny and Lana Adams
-- aka the best team you ever did see!
--              ^^^ @Lana, don't delete this!

-- Fresh start!
DROP DATABASE IF EXISTs MediDB;
CREATE DATABASE IF NOT EXISTS MediDB;
USE MediDB;

-- User/Patient table
-- Patients(userID, userPW, firstName, lastName, dateOfBirth, weight, gender)
CREATE TABLE Patients(
userID  VARCHAR(50) PRIMARY KEY,
userPW  VARCHAR(50),
firstName VARCHAR(50),
lastName VARCHAR(50),
dateOfBirth DATE,
weight FLOAT(6,2),
gender CHAR(1)
    );

-- Medications table
-- Medications(brandName, genericName, amountMG)
CREATE TABLE Medications(
brandName VARCHAR(50),
genericName VARCHAR(50),
amountMG FLOAT(6,2),
PRIMARY KEY(brandName)
    );

-- Interactions table
-- Interactions(medication1, medication2)
CREATE TABLE Interactions(
medication1 VARCHAR(50),
medication2 VARCHAR(50),
FOREIGN KEY (medication1) REFERENCES Medications(brandName)
ON DELETE CASCADE,
FOREIGN KEY (medication2) REFERENCES Medications(brandName)
ON DELETE CASCADE
    );

-- Regimen table
-- Regimen(userID, brandName, dosage, frequency, count)
CREATE TABLE Regimen(
userID VARCHAR(50),
brandName VARCHAR(50),
dosage FLOAT(6,2),
frequency VARCHAR(3),
count INTEGER,
FOREIGN KEY (userID) REFERENCES Patients(userID)
ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (brandName) REFERENCES Medications(brandName)
ON DELETE CASCADE ON UPDATE CASCADE
    );

-- Initial Data
-- Medication
INSERT INTO medications VALUES
('Lipitor', 'Atorvastatin Calcium', 100),
('Synthroid', 'Levothyroxine', 100),
('Prinivil', 'Lisinopril', 100),
('Prilosec', 'Omeprazole', 20),
('Glucophage', 'Metformin', 32),
('Norvasc', 'Amlodipine', 100),
```

```
('Zocor', 'Simvastatin', 300),
('Lortab', 'Hydrocodone/Acetaminophen', 230),
('Toprol XL', 'Metoprolol ER', 225),
('Cozaar', 'Losartan', .5),
('Zithromax', 'Azithromycin', .5),
('Ambien', 'Zolpidem', 25),
('Microzide', 'Hydrochlorothiazide', 225),
('Lasix', 'Furosemide', 800),
('Lopressor', 'Metoprolol', 25),
('Protonix', 'Pantoprazole', 25),
('Neurontin', 'Gabapentin', 45),
('Amoxil', 'Amoxicillin', 800),
('Deltasone', 'Prednisone', 250),
('Zoloft', 'Sertraline', 100),
('Flomax', 'Tamsulosin', 10),
('Flonase', 'Fluticasone', 10),
('Pravachol', 'Pravastatin', 25),
('Ultram', 'Tramadol', 50),
('Singulair', 'Montelukast', 100),
('Lexapro', 'Escitalopram', 225),
('Coreg', 'Carvedilol', 300),
('Xanax', 'Alprazolam', 25),
('Coumadin', 'Warfarin', 50),
('Mobic', 'Meloxicam', 800),
('Plavix', 'Clopidogrel', 10),
('Augmentin XR', 'Amoxicillin/Potassium Clavulanate ER', 225),
('Zyloprim', 'Allopurinol', 125),
('Wellbutrin', 'Bupropion', 10),
('Zestoretic', 'Lisinopril/HCTZ', 5),
('Celexa', 'Citalopram', 1),
('Tenormin', 'Atenolol', 1000),
('Cymbalta', 'Duloxetine', 2000),
('Prozac', 'Fluoxetine', 12.5),
('Tricor', 'Fenofibrate', .5),
('Effexor', 'Venlafaxine', .8),
('Adderall', 'Amphetamine/Dextroamphetamine', 20),
('Flexeril', 'Cyclobenzaprine', 10),
('Medrol', 'Methylprednisolone', 25),
('Klor-Con', 'Potassium Chloride', 50),
('Tylenol', 'Acetaminophen', 325),
    ('Advil', 'Ibuprofin', 200);

 INSERT INTO Patients VALUES
(0001, 1234, 'Ada', 'Lovelace', '1815-12-10', 155, 'f'),
    (0002, 1234, 'Alan', 'Turing', '1927-06-23', 150, 'm'),
    (0003, 1234, 'Linus', 'Torvalds', '1969-12-28', 195, 'm'),
    (0004, 1234, 'Steve', 'Wozniak', '1950-08-11', 245, 'm'),
('1337', 'pass', 'James', 'Trafny', '1988-03-24', 170.00, 'm'),
    ('1338', 'word', 'Lana', 'Adams', '1992-03-06', 170.00, 'f');

INSERT INTO regimen VALUES
(1337, 'Ambien', 25, 'PRN', 30),
    (1337, 'Advil', 200, 'BID', 60),
(1338, 'Adderall', 20, 'BID', 45),
    (1338, 'Amoxil', 800, 'BID', 14);

INSERT INTO interactions VALUES
('Tylenol', 'Advil'),
    ('Augmentin XR', 'Pravachol'),
    ('Prozac', 'Synthroid'),
    ('Zithromax', 'Ultram');
```

removeMedicine.java

```
/**
* Lana Adams, James Trafny
* CS 317 - Final project
*
* This application is designed to store and manage patient medicine information.
*
* This is the pop up which will allow the user to remove a medicine to their regimen
*/

import java.util.ArrayList;
import java.util.prefs.Preferences;

import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class removeMedicine {

	public static boolean display(String title) {

		String url = "jdbc:mysql://localhost:3306/medidb?useSSL=false";
		String user = "root";
		String pswd = "BlueJean1018";
		MediDB mediDB = new MediDB(url, user, pswd);

		Stage window = new Stage();
		window.initModality(Modality.APPLICATION_MODAL);
		window.setTitle(title);
		window.setMinWidth(500);
		window.setMinHeight(300);

		ChoiceBox<String> choiceBox = new ChoiceBox<>(); //brandname drop down


		//Create label, text field, buttons
		Label bnLabel = new Label("Brand Name: ");

		Preferences up = Preferences.userRoot();
		int user_id = 0;
		int userid = up.getInt("userID", user_id);
		ArrayList<String> results = mediDB.getPartRegimen(userid);
		for (String med:results) {
			choiceBox.getItems().add(med);
		}


		Button deleteBt = new Button("Delete");

		GridPane gridPane = new GridPane();
		gridPane.setVgap(20);
		gridPane.setHgap(20);

		//Add buttons
		gridPane.add(choiceBox, 1,	0);
		gridPane.add(deleteBt, 2, 0);
		gridPane.add(bnLabel, 0, 0);
		gridPane.setAlignment(Pos.CENTER);
```

```
                    Scene scene = new Scene(gridPane);
                    scene.getStylesheets().add("stylesheetMDBpopup.css");
                    window.setScene(scene);
                    window.show();

                    deleteBt.setOnAction(e -> {
                            mediDB.removeFromRegimen(userid, choiceBox.getValue());
                            window.close();
                    });

                    return true;

}

}
```

sendPopUp.java

```
/**
 * Lana Adams, James Trafny
 * CS 317 - Final project
 *
 * This application is designed to store and manage patient medicine information.
 *
 * This is the pop up which will allow the user to enter an email
 * to have their medicine list sent to their doctor
 */
import javafx.stage.*;
import javafx.scene.*;
import javafx.scene.layout.*;
import javafx.scene.control.*;
import javafx.geometry.*;
public class sendPopUp {

	public static boolean display(String title) {
		Stage window = new Stage();
		window.initModality(Modality.APPLICATION_MODAL);
		window.setTitle(title);
		window.setMinWidth(500);
		window.setMinHeight(300);

		//Create label, text field, buttons
		Label enterEmail = new Label("Doctor's Email: ");
		TextField drEmail = new TextField();
		Button send = new Button("Send");

		GridPane gridPane = new GridPane();
		gridPane.setVgap(20);
		gridPane.setHgap(20);

		//Add buttons
		gridPane.add(drEmail, 1,	0);
		gridPane.add(send, 2, 0);
		gridPane.add(enterEmail, 0, 0);
		gridPane.setAlignment(Pos.CENTER);
		Scene scene = new Scene(gridPane);
		scene.getStylesheets().add("stylesheetMDBpopup.css");
		window.setScene(scene);
		window.show();

		send.setOnAction(e -> {
			window.close();
		});

		return true;

}}
```

updatePopUp.java

```
/**
 * Lana Adams, James Trafny
 * CS 317 - Final project
 *
 * This application is designed to store and manage patient medicine information.
 *
 * This is the pop up which will allow the user to decide between adding or
 * removing a medication from his/her list
 */
/**
 * modified version of
 * https://github.com/buckyroberts/Source-Code-
 * from-Tutorials/blob/master/JavaFX/006_communicatingBetweenWindows/ConfirmBox.java
 */

import javafx.stage.*;
import javafx.scene.*;
import javafx.scene.layout.*;
import javafx.scene.control.*;
import javafx.geometry.*;

public class updatePopUp {

	public static boolean display(String title) {
		Stage window = new Stage();
		window.initModality(Modality.APPLICATION_MODAL);
		window.setTitle(title);
		window.setMinWidth(500);
		window.setMinHeight(300);

		//Create two buttons
		Button addMed = new Button("Add\nMedicine");
		Button removeMed = new Button("Remove\nMedicine");

		GridPane gridPane = new GridPane();
		gridPane.setVgap(20);
		gridPane.setHgap(20);

		//Add buttons
		gridPane.add(addMed, 0,	0);
		gridPane.add(removeMed, 1, 0);
		gridPane.setAlignment(Pos.CENTER);
		Scene scene = new Scene(gridPane);
		scene.getStylesheets().add("stylesheetMDBpopup.css");
		window.setScene(scene);
		window.show();

		addMed.setOnAction(e -> {
			boolean result = updateUserInput.display("Add New Medicine");
			window.close();
		});

		removeMed.setOnAction(f -> {
			boolean result = removeMedicine.display("Remove Medicine");
			window.close();
		});

		return true;
	}


}
```

updateUserInput.java

```
/**
 * Lana Adams, James Trafny
 * CS 317 - Final project
 *
 * This application is designed to store and manage patient medicine information.
 *
 * This is the pop up which will allow the user to add a new medicine to their regimen
 */
import java.util.ArrayList;
import java.util.prefs.Preferences;

import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class updateUserInput {

	public static boolean display(String title) {

		String url = "jdbc:mysql://localhost:3306/medidb?useSSL=false";
		String user = "root";
		String pswd = "BlueJean1018";
		MediDB mediDB = new MediDB(url, user, pswd);

		Stage window = new Stage();
		window.initModality(Modality.APPLICATION_MODAL);
		window.setTitle(title);
		window.setMinWidth(500);
		window.setMinHeight(500);

		ChoiceBox<String> choiceBox = new ChoiceBox<>(); //brandname drop down

		//Create label, text field, buttons
		Label bnLabel = new Label("Brand Name: ");

		ArrayList<String> results = mediDB.getMedicationList();
		for (String med:results) {
			choiceBox.getItems().add(med);
		}

		choiceBox.getItems().add("DRUGZ");
		Label doseLabel = new Label("Dosage: ");
		TextField doseInput = new TextField();
		Label mgLabel = new Label("mg");
		Label freqLabel = new Label("Frequency: ");
		TextField freqInput = new TextField();
		Label countLabel = new Label("Count: ");
		TextField countInput = new TextField();

		//create the button
		Button addBt = new Button("Add");

		//create the girdpane
		GridPane gridPane = new GridPane();
		gridPane.setVgap(20);
		gridPane.setHgap(20);

		//Add buttons
```

```
                    gridPane.add(bnLabel, 0, 0);
                    gridPane.add(choiceBox, 1, 0);
                    gridPane.add(doseLabel, 0, 1);
                    gridPane.add(doseInput, 1, 1);
                    gridPane.add(mgLabel, 2, 1);
                    gridPane.add(freqLabel, 0, 2);
                    gridPane.add(freqInput, 1, 2);
                    gridPane.add(countLabel, 0, 3);
                    gridPane.add(countInput, 1, 3);
                    gridPane.add(addBt, 1, 4);
                    gridPane.setAlignment(Pos.CENTER);
                    Scene scene = new Scene(gridPane);
                    scene.getStylesheets().add("stylesheetMDBpopup.css");
                    window.setScene(scene);
                    window.show();

                    Preferences up = Preferences.userRoot();
                    int user_id = 0;
                    int userid = up.getInt("userID", user_id);



                    addBt.setOnAction(e -> {
                            int count = Integer.parseInt(countInput.getText());
                            int dosage = Integer.parseInt(doseInput.getText());
                            mediDB.addToRegimen(userid, choiceBox.getValue(), dosage, freqInput.getText(), count);
                            window.close();
                    });

                    return true;



}

}
```

userHome.java

```
/**
 * Lana Adams, James Trafny
 * CS 317 - Final project
 *
 * This application is designed to store and manage patient medicine information.
 *
 * This is the home stage for the application, the user
 * can choose to update, send, download, view, or manage
 * their medicine list by clicking the corresponding button.
 */

import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Menu;
import javafx.scene.control.MenuBar;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.GridPane;
import javafx.stage.Popup;
import javafx.stage.Stage;
import javafx.stage.Window;
import javafx.scene.control.MenuItem;
import javafx.scene.control.Tooltip;

public class userHome extends Application{
	////////////////////////////////////////////////////////////////////////////////
	//grid pane for button options
	private BorderPane borderPane;
	private GridPane centerBPane;
	////////////////////////////////////////////////////////////////////////////////
	//menu bar set up
	private MenuBar menubar;
	private Menu menuFile, menuHelp, menuOptions;
	//menu items
	private MenuItem miClose, miLogout, miManage, miUpdate, miView, miSend, miDownload;
	private MenuItem miAbout;
	////////////////////////////////////////////////////////////////////////////////
	/**
	 * Button for user to update their current medicine.
	 */
	private Button updateBt;

	/**
	 * Button for user to view their chart
	 */
	private Button viewBt;

	/**
	 * Button for user to email chart
	 */
	private Button sendBt;

	/**
	 * Button to download PDF of chart
	 */
	private Button downloadBt;

	/**
	 * Button for user to manage their refills
	 */
	private Button manageRXBt;
```

```
////////////////////////////////////////////////////////////////////////////////

public userHome() {
        ////////////////////////////////////////////////////////////////////////////////
        //instantiate the button pane
        centerBPane = new GridPane();
        centerBPane.setVgap(20);
        centerBPane.setHgap(20);
        centerBPane.setAlignment(Pos.CENTER);

        ////////////////////////////////////////////////////////////////////////////////
        //instantiate buttons
        updateBt = new Button("Update\nMedicine");
        viewBt = new Button("View\nMediChart");
        sendBt = new Button("Send\nMediChart");
        downloadBt = new Button("Download\nTo PDF");
        manageRXBt = new Button("Manage\nRX");

        //add tooltips
        Tooltip tt1 = new Tooltip("Click to add or remove medicine");
        Tooltip.install(updateBt, tt1);
        Tooltip tt2 = new Tooltip("Click to view your weekly MediChart");
        Tooltip.install(viewBt, tt2);
        Tooltip tt3 = new Tooltip("Click to send your medicine list via email");
        Tooltip.install(sendBt, tt3);
        Tooltip tt4 = new Tooltip("Click to download a PDF of your current medicine list");
        Tooltip.install(downloadBt, tt4);
        Tooltip tt5 = new Tooltip("Click to change your medicine information");
        Tooltip.install(manageRXBt, tt5);

        ////////////////////////////////////////////////////////////////////////////////
        //layout
        centerBPane.add(updateBt, 1, 0);
        centerBPane.add(viewBt, 3, 0);
        centerBPane.add(sendBt, 5, 0);
        centerBPane.add(downloadBt, 2, 1);
        centerBPane.add(manageRXBt, 4, 1);

        ////////////////////////////////////////////////////////////////////////////////
        //menu bar
        menuFile = new Menu("File");
        menuHelp = new Menu("Help");
        menuOptions = new Menu("Options");

        menubar = new MenuBar();

        //menuFile.getItems().addAll(miLogout, miClose);
        //menuHelp.getItems().add(miAbout);
        //menuOptions.getItems().addAll(miManage, miUpdate, miView, miSend, miDownload );

        menubar.getMenus().addAll(menuFile, menuOptions, menuHelp);
        ////////////////////////////////////////////////////////////////////////////////


}

public void start(Stage userHome) {

        borderPane = new BorderPane();
        Scene scene = new Scene(borderPane, 500, 500);
        scene.getStylesheets().add("stylesheetMDB2.css");
        //menubar
        borderPane.setTop(menubar);
        borderPane.setCenter(centerBPane);


        userHome.setScene(scene);
```

```
                        userHome.setTitle("MediDB");
                        userHome.setMaximized(false);
                        userHome.setMaximized(true);
                        userHome.show();

                        ////////////////////////////////////////////////////////////////////////////
                        //login button
                        updateBt.setOnAction(e -> {
      boolean result = updatePopUp.display("Update MediChart");
    });

                        sendBt.setOnAction(f -> {
                                boolean result = sendPopUp.display("Send to Doctor");
                        });

                        manageRXBt.setOnAction(g -> {
                                boolean result = manageRXPopup.display("Update Medicine Count");
                        });

                        viewBt.setOnAction(h -> {
                                boolean result = viewRegimen.display("Your Regimen");

                        });

            }

            public static void main(String[] args) {
                        launch(args);

            }

}
```

viewRegimen.java

```
/**
 * Lana Adams, James Trafny
 * CS 317 - Final project
 *
 * This application is designed to store and manage patient medicine information.
 *
 * This is the pop up which will allow the user to view their regimen
 */
import java.util.ArrayList;
import java.util.prefs.Preferences;

import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.VBox;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class viewRegimen {

	static String url = "jdbc:mysql://localhost:3306/medidb?useSSL=false";
	static String user = "root";
	static String pswd = "BlueJean1018";
	static MediDB mediDB = new MediDB(url, user, pswd);

	public static boolean display(String title) {
		Stage window = new Stage();
		window.initModality(Modality.APPLICATION_MODAL);
		window.setTitle(title);
		window.setMinWidth(500);
		window.setMinHeight(300);

		VBox vBox = new VBox();
		vBox.setAlignment(Pos.CENTER);
		Preferences up = Preferences.userRoot();
		int user_id = 0;
		int userid = up.getInt("userID", user_id);

		ArrayList<String> results = mediDB.getFullRegimen(userid);
		Label labels[] = new Label[results.size()];
		int index = 0;
		for (String row:results) {
			labels[index] = new Label(row);
			vBox.getChildren().add(labels[index]);
			index++;
		}



		Scene scene = new Scene(vBox);
		scene.getStylesheets().add("stylesheetMDBpopup.css");
		window.setScene(scene);


		//call to mediDB

		window.show();


		return true;
	}}
```