**Office Scheduler Application Overview**

**Cody Dempsey**

**Written: 12/1/2023**

**Edited: 6/6/24**

Table of Contents

**Design Document3**

Class Design3

Class Diagram5

UI Design6

UI Diagrams8

**Unit Test12**

Overview and Purpose12

Plan12

Procedure and Outcomes16

**Source Code21**

**User Guides22**

Installation Guide22

MySQL22

Office Scheduler33

Application Guide34

Design Document

Class Diagram

The class diagram below depicts the classes and their relationships in the Office Scheduler application. Inheritance is shown between the Patient class and the AdultPatient and PediatricPatient subclasses. The subclasses are differentiated by the polymorphic method PatientAgeType() which returns different values depending on whether it is called from an adult or pediatric patient object. This polymorphism is utilized by comparing the returned value of PatientAgeType() to ensure that pediatric patients can only meet for appointments with physicians that specialize in pediatrics and adult patients may only meet with adult-specialized physicians.

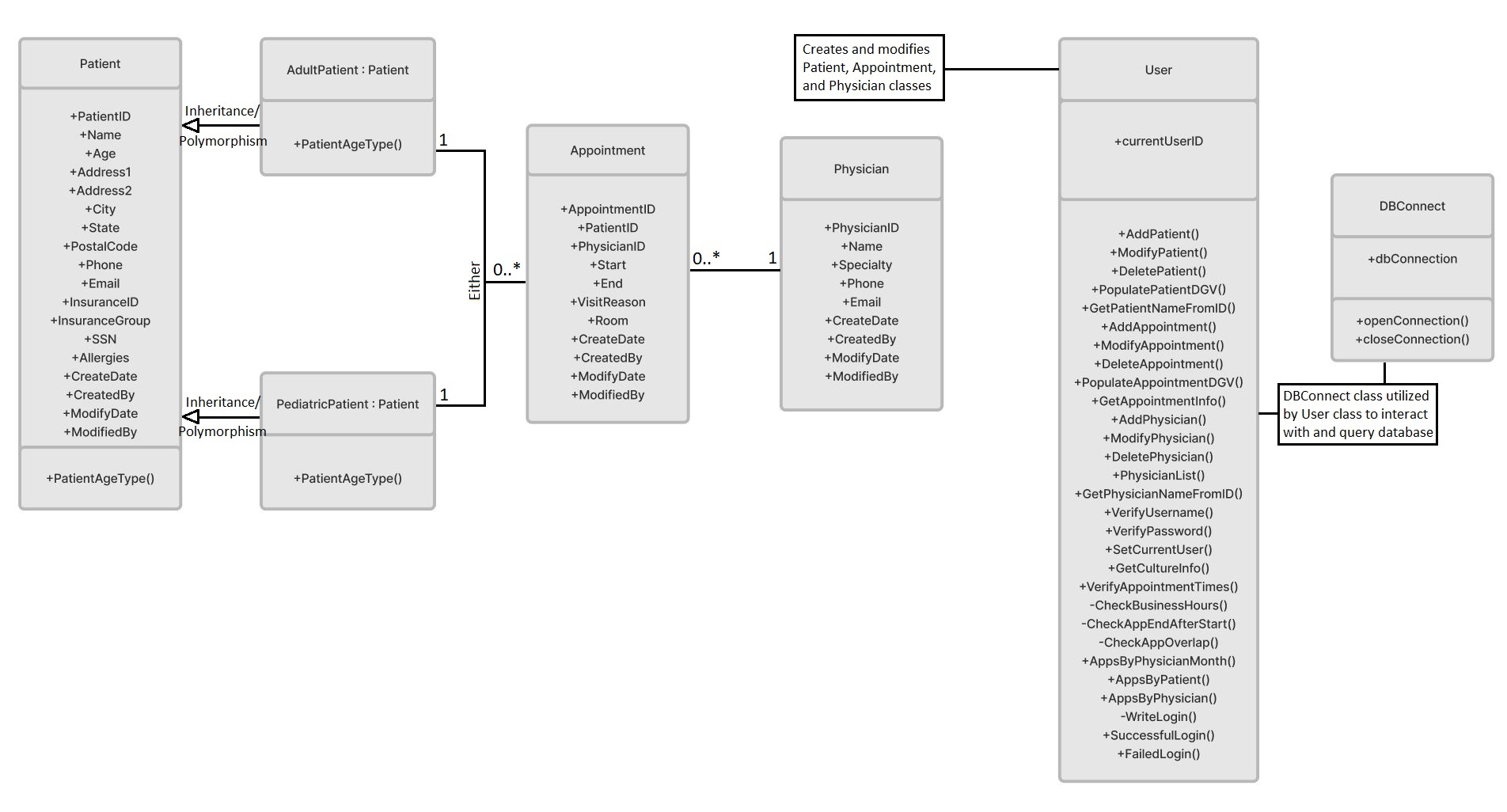
The Appointment class relates to several other classes. It must have one and only one physician. It must also have one and only one of either Patient subclass: AdultPatient or PediatricPatient. Either of the Patient subclasses and the Physician class may have zero-to-many Appointments.

The User class functions primarily as a utility class but maintains a relationship to the application’s Physician, Appointment, and Patient classes. When a Physician, Appointment, or Patient class is created or updated, it is done so by an end-user of the application. This user’s unique ID—the public attribute currentUserID belonging to the User class—is stored in the CreatedBy and/or ModifiedBy attributes of the class object that is being created or modified.

Otherwise, the User class serves as the layer that queries and interacts with the MySQL database upon which the Office Scheduler application relies. This results in a relatively large class with a lengthy list of static methods which are called through form interactions and make queries, additions, modifications, and deletions to the database. Most of these methods could be split up into their relevant classes (e.g.: AddPatient(), ModifyPatient(), and DeletePatient() could be contained in the Patient class) but a design decision was made to contain them within a single class due to their similarity in structure and syntax.

The DBConnect class object is created when the application launches to establish the initial connection to the database. This must be done at launch so the user’s credentials can be validated against records in the database’s user table to allow login. DBConnect relates to the User class as its public attribute, dbConnection, is used in the construction of every MySQLCommand object within User’s methods.

Office Scheduler Class Diagram



UI Diagrams

Displayed below are the low and high-fidelity wireframes of the Office Scheduler’s main and add appointment menus. The low-fidelity main menu shows the appointment calendar which takes up most of the space in the application window. To the calendar’s left are its date controls which allow the user to filter the appointments shown by day, week, and month. Below that are controls to add, modify, and delete patients and physicians as well as the reports button. These controls will open separate windows that will not be presented here due to their similarity to the add appointment window which will be discussed here later.

Beneath the appointment calendar are controls to add, modify, and delete appointments. These buttons will open a new window to either enter the details of a new appointment or edit the details of an appointment that is selected from the appointment calendar.

The bottom right of the main menu contains the search field and search controls, which allow the user to search through the appointments displayed in the calendar by physician name or patient name. The button to exit the application is also in this area.

Following this main menu wireframe is the low-fidelity wireframe for the window used to add a new appointment. At the top of this window is a list of all patient records and their details to select and assign to the appointment. Below is a combo box to assign the appointment’s physician. The appointment details sector will contain a reason for the visit as well as the examination room that will be used for the appointment.

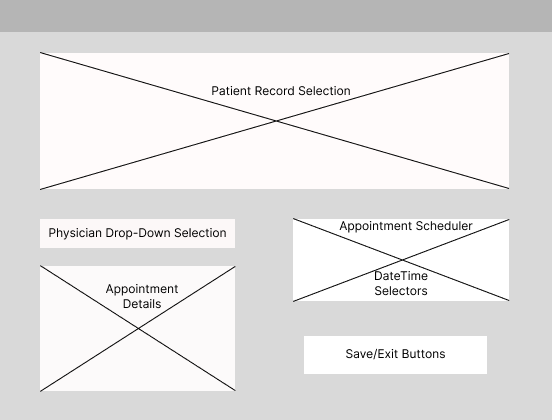
To the right of the window is the DateTime picker for the appointment. This will allow the user to select a date and time for both the start and end of the appointment. Finally, buttons for saving the appointment or exiting the form are in the bottom right.

Beneath the low-fidelity wireframes are their respective high-fidelity counterparts. These were created in Visual Studio 2022 as Windows Forms items due to the application’s ease of use for creating mockups of this nature. As a result, these wireframes are utilized directly in the final application.

Main Menu Low-Fidelity Wireframe

Lo-Fi Main Menu


Add Appointment Low-Fidelity Wireframe



Main Menu High-Fidelity Wireframe

A screenshot of a computer

Description automatically generated

Add Appointment High-Fidelity Wireframe

A screenshot of a computer

Description automatically generated

Unit Test

Overview and Purpose

The medical office has six rooms available for procedures, consultations, and other appointment needs. These rooms are labeled numerically 1 through 6. Due to privacy concerns and practicality, no two appointments can share the same room at the same time. The application has logic in place to check that neither physicians nor patients are double booked. However, when an appointment is created or modified, logic must also be in place to check rooms for existing appointments regardless of the attending physician or patient.

Plan

The Office Scheduler is a C# Windows Forms application written using Visual Studio 2022. Visual Studio provides means for creating unit test classes and methods to test methods within a project. However, due to incompatibilities with the .NET framework version used for this application, Visual Studio’s unit testing features is unavailable for use on this project. Instead, a manual testing method has been chosen to test the application’s ability to detect existing appointments in one of the medical office’s rooms and disallow an appointment to be scheduled in that room during that time.

There are two items required for the first part of the unit test: an appointment scheduled in an exam room, and another appointment which we will attempt to schedule in the same room during the same timeframe. Because the application checks for double-booking a physician and a patient before it checks whether a room is booked, the second appointment must be for a different physician and patient than the first.

For the second part of the unit test, the second appointment will be scheduled during the same timeframe as the first but in a separate room that has no appointments scheduled during that time. Because the application will still detect a double-booked physician or patient, the second appointment must still be for a different physician and patient than the first.

Finally, two appointments for the same room will be scheduled during different timeframes. Because the times will not overlap, the appointments can be scheduled for any physician or patient. Here, the edge case of having two appointments for the same room adjacent to each other on the schedule will be tested.

The “Add Appointment” window is the feature of the application where this testing will take place. Optionally, the initial appointment can be inserted into the database using MySQL Workbench, but **appointment validation is performed client-side** and testing must be performed within the Office Scheduler application. The “Room” field will be the specific control for testing in this window. This field is a text box whose contents are validated when “Save” is clicked. The validation first checks if the text entered is an integer with a value between 1 and 6 inclusively. Next, the database is queried to check for appointments for the chosen room during the timeframe selected with the DateTime picker in this window.

The test should provide two possible outcomes. The first outcome is a MessageBox that informs the user that there is already an appointment for the scheduled time in the room they are attempting to schedule the new appointment. This MessageBox will provide the room number in question and request that the user review the appointment calendar and try again. The second outcome is a new appointment—one that does not overlap any other appointments in the room selected—added to the appointment calendar with the patient, physician, visit reason, room, start, and end times provided by the user.

The deliverable for this test is certifying the functionality of the “Check Room Overlap” code segment contained in the “TimeCheck” region of the User class. This method of manually checking the validation functionality of form fields is applicable for the entire test plan of the project. No iteration is used as the method for testing is manual input.

The tasks for this unit test are:

1. Determine and record the testing procedure.
2. Ensure the testing environment is set up and available per the needs listed below.
3. Perform the test.
4. Record the results.
5. Certify code functionality or rework code and repeat the test.

The needs for this test are the deployed MySQL database, the installation of the Office Scheduler application on a Windows 10/11 PC, and confirmation of communication between the application and the MySQL database; outlined in the setup guides later in this document.

The test passes if the application successfully prevents a user from adding an appointment that is assigned to a room that already has an existing appointment during the same timeframe. The application must not allow any overlap in scheduled times for the room and must notify the user of the scheduling error.

The test fails if the application allows a user to successfully add an appointment to the calendar that is assigned to a room that already has an existing appointment during the same timeframe.

The specific segment of code that is being tested in this unit test is shown below.

A computer screen shot of a program code

Description automatically generated

Procedure and Outcomes

The testing procedure is performed manually within the application itself and utilizes the following steps:

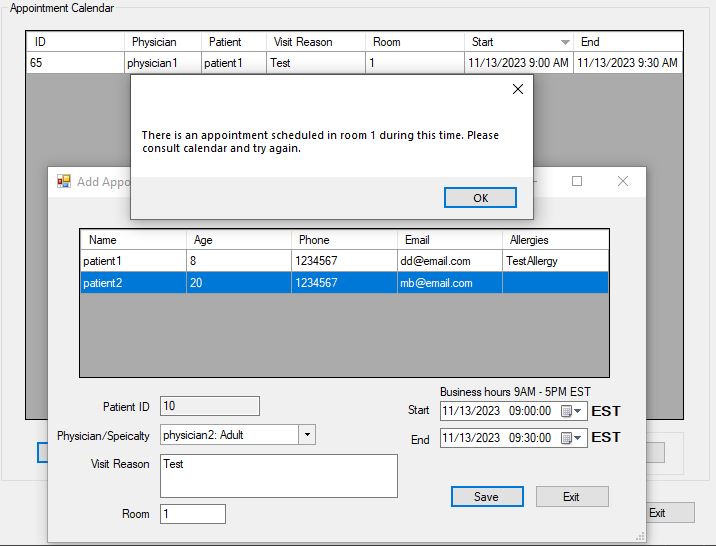
1. Schedule an appointment in exam room 1 for physician1 and patient1 for 11/13/23 from 9AM to 9:30AM.
2. Schedule an appointment in exam room 1 for physician2 and patient2 for 11/13/23 from 9AM to 9:30AM.
3. Examine the application’s output and ensure only the appointment from step 1 remains for further testing.
4. Schedule an appointment in exam room 2 for physician2 and patient2 for 11/13/23 from 9AM to 9:30AM.
5. Repeat step 3.
6. Schedule an appointment in exam room 1 for physician2 and patient2 for 11/13/23 from 9:30AM to 10AM.
7. Repeat step 3.
8. Schedule an appointment in exam room 1 for physician2 and patient2 for 11/13/23 from 9:35AM to 10AM.
9. Examine the application’s output and consolidate the results.

Screenshots of each step’s results are shown below.

Step 1:

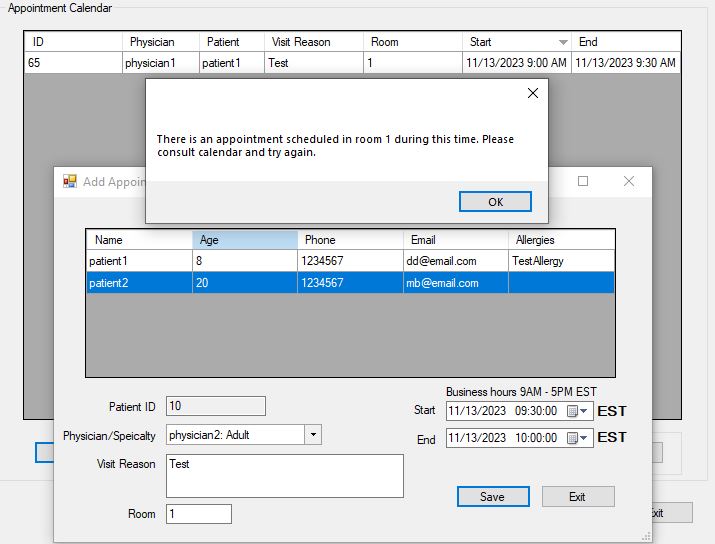
A screenshot of a computer

Description automatically generated

Step 2-3:

Step 4-5:A screenshot of a computer

Description automatically generated

Step 6-7 (Note that the application allows for **no** overlap even when the new appointment’s start time is the same as the previous appointment’s end time. This is by design to allow for time between appointments for medical staff to set up the examination room.)

Steps 8-9:A screenshot of a computer

Description automatically generated

The unit test concluded successfully. The user was alerted each time they attempted to create an appointment that overlapped the time of another and was in the same room. The application allowed the user to create appointments that had overlapping times but were in separate rooms or appointments in the same room with a brief period between the end of the first appointment and the start of the next.

User Guide

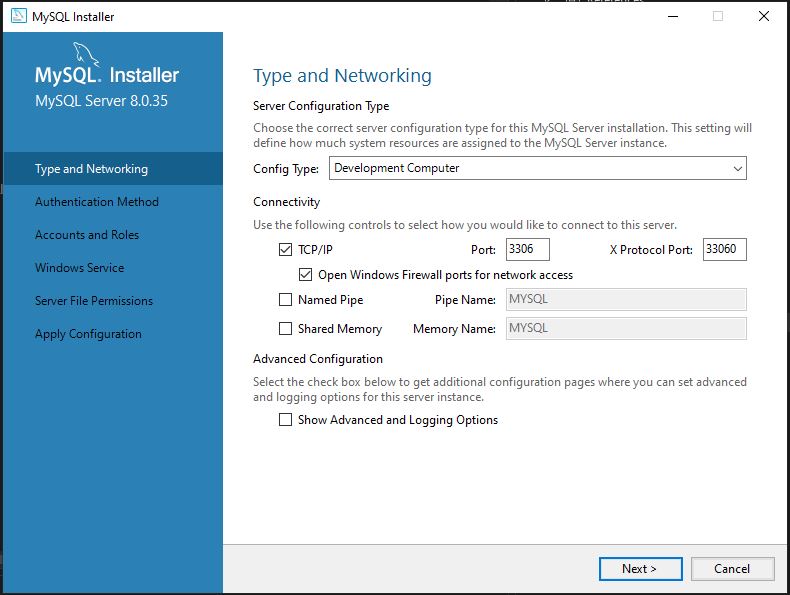
Installation Guide

MySQL

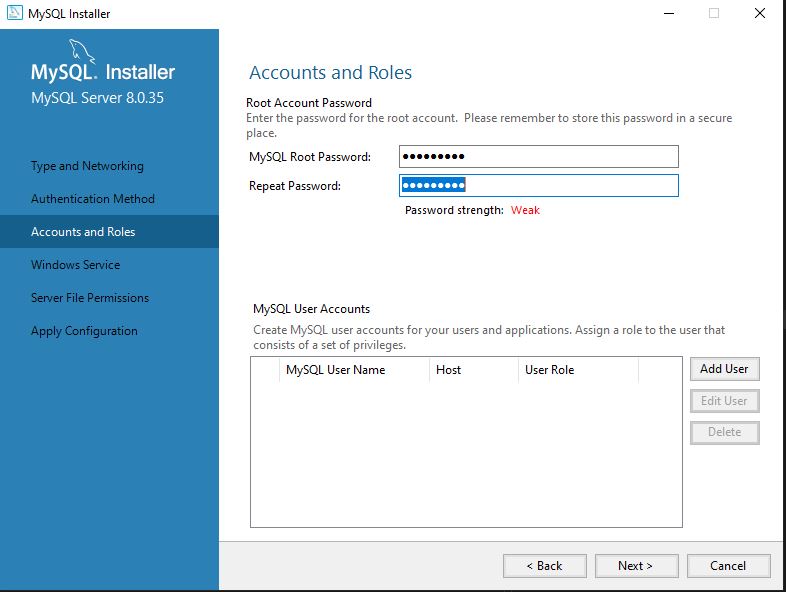
The following installation guide is for assessment evaluation purposes. Actual installation will differ slightly, and the steps will be developed alongside the Xxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxx’s server and desktop teams to adhere to their protocols.

This Windows Forms application requires a Windows PC with an updated version of Windows 10 or 11. A MySQL Community database will need to be installed locally for the application to function. Steps for installing the MySQL database are outlined here:

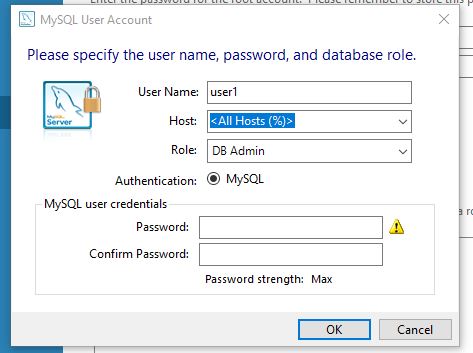
* Download “MySQL Installer for Windows” from: <https://dev.mysql.com/downloads/>
* Run installer and select Full Install.
* Click execute to install all listed products.
* Proceed to configuration.
* Ensure the Type and Networking page appears as below:



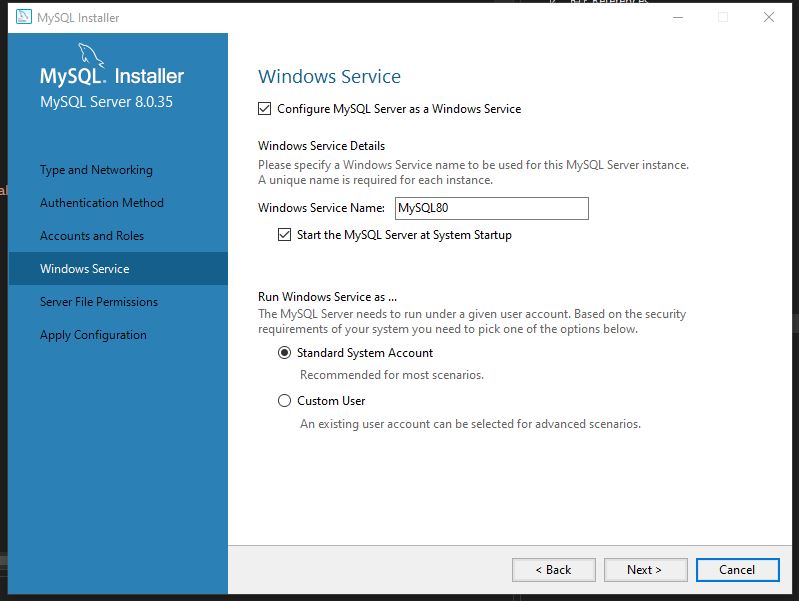
* Select “Use Strong Password Encryption for Authentication (RECOMMENDED)” on the next page.
* On the page shown below, enter “password1” without quotes as the MySQL Root Password. Then select “Add User”.



* Enter the username “user1” without quotes as shown below. **Do not enter a password.**



* Click “OK”, then “Next” on the Accounts and Roles page.
* Ensure the Windows Service page appears as below:



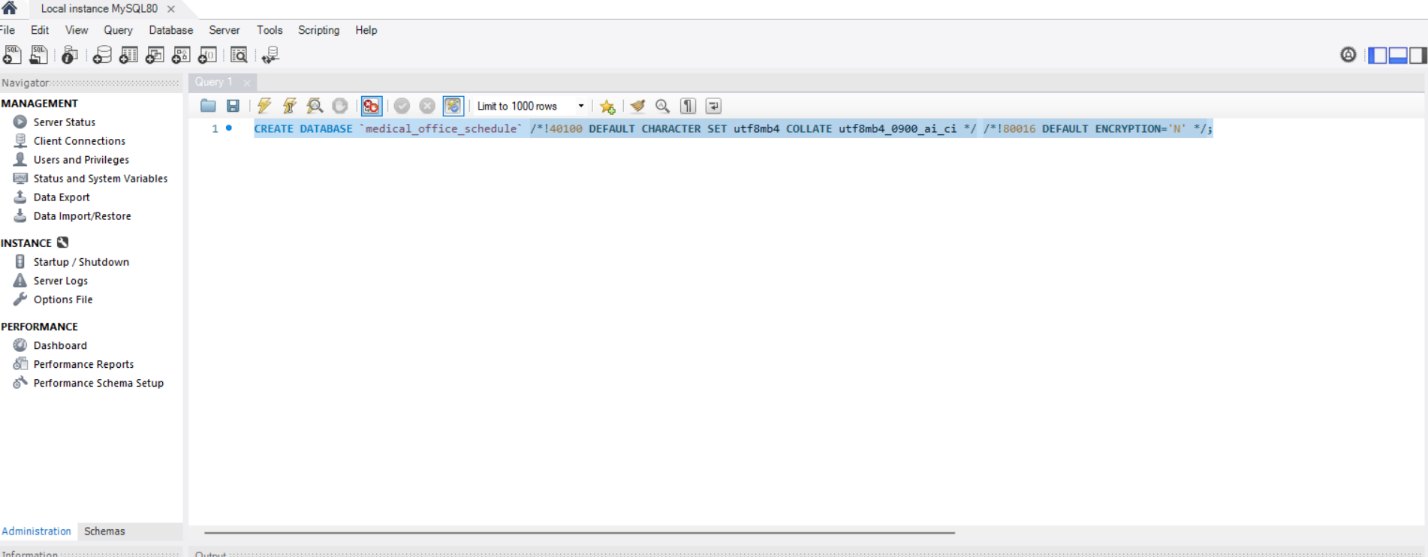
* Click “Next”.
* Select: “Yes, grant full access to the user running the Windows Service (if applicable) and the administrators group only. Other users and groups will not have access.”
* Click “Next”.
* Click “Execute” to apply the configuration.
* Click “Finish”.
* Click “Next”, then cancel the MySQL Router Configuration.
* Click “Next”, then cancel the Samples and Examples Configuration.
* Click “Finish”. Launch MySQL Workbench if it doesn’t launch automatically.
* Double click “Local instance MySQL80” under MySQL Connections as shown below and enter password1 as the password if prompted.

A screenshot of a computer

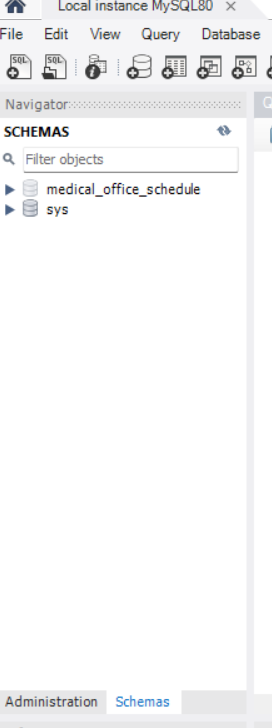
Description automatically generated

* Paste the following CREATE DATABASE command into the query window as shown below and click the lightning bolt icon to execute the script. This will create a database called “medical\_office\_schedule”

CREATE DATABASE `medical\_office\_schedule` /\*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci \*/ /\*!80016 DEFAULT ENCRYPTION='N' \*/;



* Select the Schemas tab from the bottom of the Navigator pane on the left. Click on the refresh icon, then double click on medical\_office\_schema to make it the active schema. It should become bolded.



* Paste the following CREATE TABLE commands into the query window and execute them with the lightning bolt button. **They must be executed in the order below.**

CREATE TABLE `patient` (

`idpatient` int NOT NULL AUTO\_INCREMENT,

`name` varchar(45) DEFAULT NULL,

`age` int DEFAULT NULL,

`addressline1` varchar(45) DEFAULT NULL,

`addressline2` varchar(45) DEFAULT NULL,

`city` varchar(45) DEFAULT NULL,

`state` varchar(45) DEFAULT NULL,

`postalcode` varchar(45) DEFAULT NULL,

`phone` varchar(45) DEFAULT NULL,

`email` varchar(45) DEFAULT NULL,

`insuranceid` varchar(45) DEFAULT NULL,

`insurancegroup` varchar(45) DEFAULT NULL,

`ssn` int DEFAULT NULL,

`allergies` varchar(250) DEFAULT NULL,

`createdate` datetime DEFAULT NULL,

`createdby` int DEFAULT NULL,

`modifydate` datetime DEFAULT NULL,

`modifiedby` int DEFAULT NULL,

PRIMARY KEY (`idpatient`)

) ENGINE=InnoDB AUTO\_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `physician` (

`idphysician` int NOT NULL AUTO\_INCREMENT,

`name` varchar(45) DEFAULT NULL,

`specialty` enum('Pediatric','Adult') DEFAULT NULL,

`phone` varchar(45) DEFAULT NULL,

`email` varchar(45) DEFAULT NULL,

`createdate` datetime DEFAULT NULL,

`createdby` int DEFAULT NULL,

`modifydate` datetime DEFAULT NULL,

`modifiedby` int DEFAULT NULL,

PRIMARY KEY (`idphysician`)

) ENGINE=InnoDB AUTO\_INCREMENT=6 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci COMMENT=' ';

CREATE TABLE `appointment` (

`idappointment` int NOT NULL AUTO\_INCREMENT,

`patientid` int NOT NULL,

`physicianid` int NOT NULL,

`start` datetime NOT NULL,

`end` datetime NOT NULL,

`visitreason` varchar(250) DEFAULT NULL,

`room` varchar(45) DEFAULT NULL,

`createdate` datetime DEFAULT NULL,

`createdby` int DEFAULT NULL,

`modifydate` datetime DEFAULT NULL,

`modifiedby` int DEFAULT NULL,

PRIMARY KEY (`idappointment`),

KEY `patientid\_idx` (`patientid`),

KEY `physicianid\_idx` (`physicianid`),

CONSTRAINT `patientid` FOREIGN KEY (`patientid`) REFERENCES `patient` (`idpatient`) ON DELETE RESTRICT,

CONSTRAINT `physicianid` FOREIGN KEY (`physicianid`) REFERENCES `physician` (`idphysician`) ON DELETE RESTRICT

) ENGINE=InnoDB AUTO\_INCREMENT=68 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `user` (

`iduser` int NOT NULL AUTO\_INCREMENT,

`username` varchar(45) NOT NULL,

`password` varchar(45) NOT NULL,

`admin` tinyint(1) NOT NULL,

PRIMARY KEY (`iduser`)

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

* Paste the following INSERT statement into the query window and execute the command with the lightning bolt button. This prepares a test user and simulates how the medical center’s database administrators will add users to the application.

INSERT INTO user VALUES (NULL, ‘test’, ‘test’, 1);

Office Scheduler

The MySQL database should now be configured properly for the Office Scheduler application. Steps for installing the application for evaluation purposes are outlined here:

* Navigate to the downloaded folder > bin > Release
* Within the Release folder, locate the file named “Office Scheduler.exe”
* The application can be launched by double-clicking this file, or a shortcut can be created by right clicking on the file and selecting “Create shortcut”. The shortcut can be moved to the desktop and used to launch the application.

Application Guide

* **Login**

Enter your username and password on this screen. For evaluation purposes, username “test” and password “test” are automatically populated in these fields. If you get an invalid username/password error, please review the MySQL installation instructions.

**Note:** Username and password administration is not supported within the application. The medical center’s database administrators handle adding and removing users for all applications. Requests for new users and password changes will be submitted to them by office staff via the medical center’s IT ticketing system.

A screenshot of a login box

Description automatically generated

* **Main Menu**

The appointment calendar is displayed here. Use the search field beneath it to search for appointments by physician or patient name. Use the calendar filter to the left to search by all appointments, month, week, or day. The add, modify, and delete buttons beneath the appointment calendar will add new appointments, or modify/delete the selected appointment from the calendar.

The patient control and physician control buttons allow you to add, modify, and delete patient and physician records respectively.

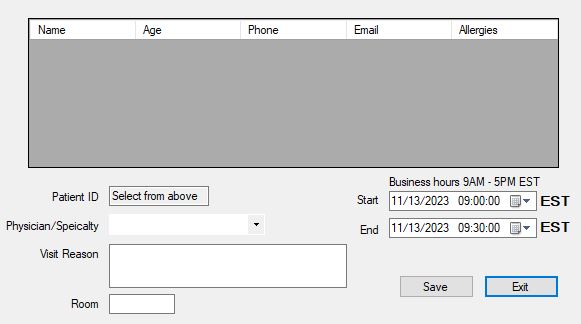
The reporting tool button in the bottom left will open the reports window.

A screenshot of a computer

Description automatically generated

* **Add/Modify/Delete Appointments**

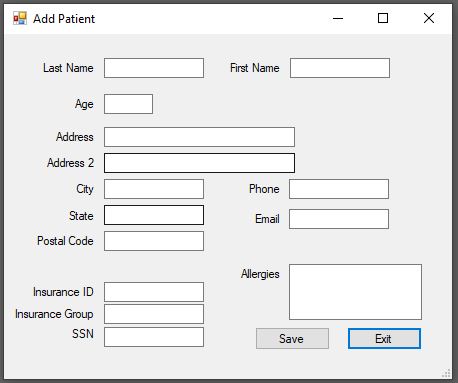
The add appointment window is displayed below. It is identical to the modify appointment window. Select a patient from the records at the top of the window. Assign a physician to the appointment from the Physician drop-down menu. Visit Reason is an optional field. A room number between 1 and 6 must be entered and start/end dates and times must be selected. Click Save to save the new appointment or the changes made to an existing one. If no patients or physicians are present, add them using their respective controls on the main menu.



* **Add/Modify/Delete Patients**

The add patient and modify/delete patient windows are displayed below. All fields are required except the Address 2 and Allergies fields.

To modify or delete a patient, first select their record from the top of the modify/delete window. Edit their information or click Delete to delete them. Click Save to add the new patient or save modifications to an existing one.



A screenshot of a computer

Description automatically generated

* **Add/Modify/Delete Physician**

The add and modify/delete physician windows are displayed below. All fields are required. Select a physician from the top of the window to edit their information or delete the record.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* **Reporting Tool**

The reporting tool window is displayed below. Select a physician and a month from the “Number of Appointments by Physician/Month” control to view appointments for that physician for the chosen month.

Select a physician from the next control to show a complete schedule for that physician.

Select a patient from the bottom control to show a complete schedule for that patient.

A screenshot of a computer screen

Description automatically generated