C868 - Software Capstone Project Summary

Task 2 - Section C

Capstone Proposal Project Name: Retire Me Now Scheduling Software Application

Student Name: Nazareth Aguilera

Table of Contents

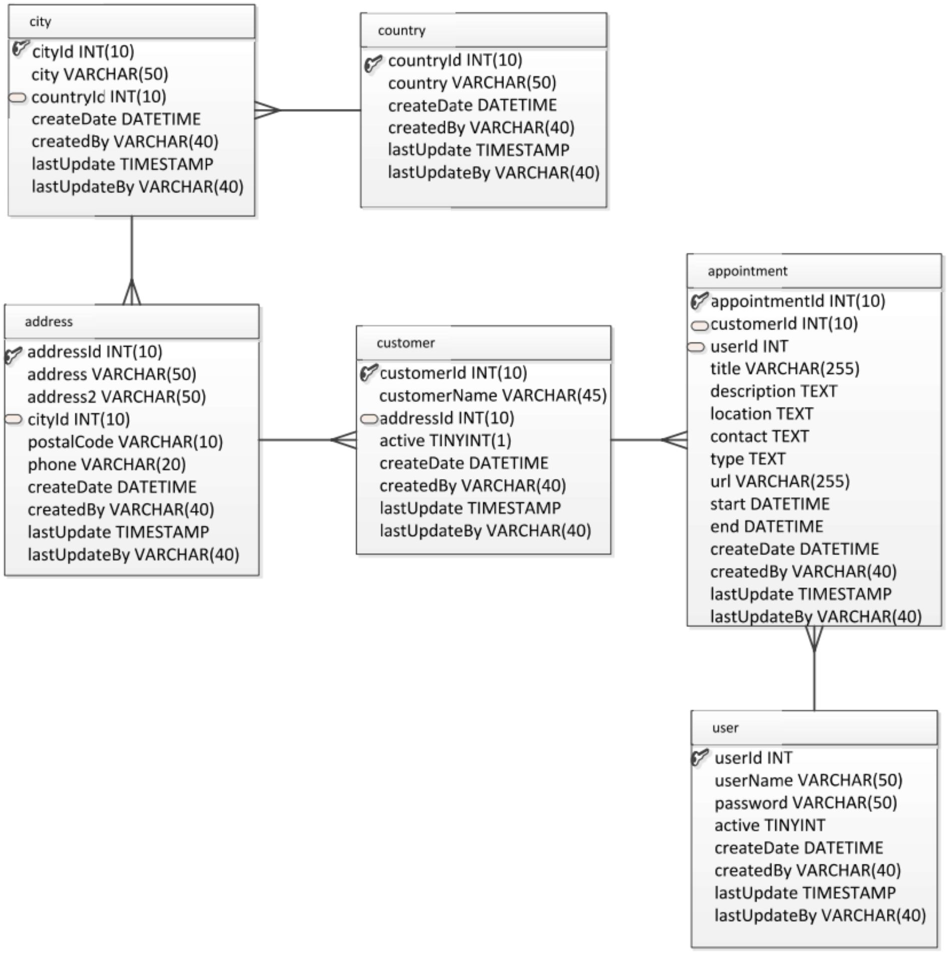
Application Design

Summary

This section contains all relevant design documents created for the Retire Me Now Scheduling Software Application. These documents include the Entity Relationship Diagram, the Class Diagram and Wireframes. With well understood requirements, the development team is able to expedite the development of the end product for the customer.

Entity Relationship Diagram

An entity in the context of a database is designed to represent a person or object in reality. The fields belonging to the entity define different traits that the entity might possess. This diagram, also known as an Entity Relationship Diagram, is a way of mapping out how the different entities in the database might relate to one another. This diagram is used to facilitate how the database tables are created.



Class Diagram

Similar to the concept of an entity for a database, a class in C# is a representation of a person or object in reality. A class includes a name, properties that reflect characteristics, and methods that represent an action an instance of a class can perform. This diagram shows how different classes interact with each other and also includes classes that stand alone but are necessary to the development of the application.

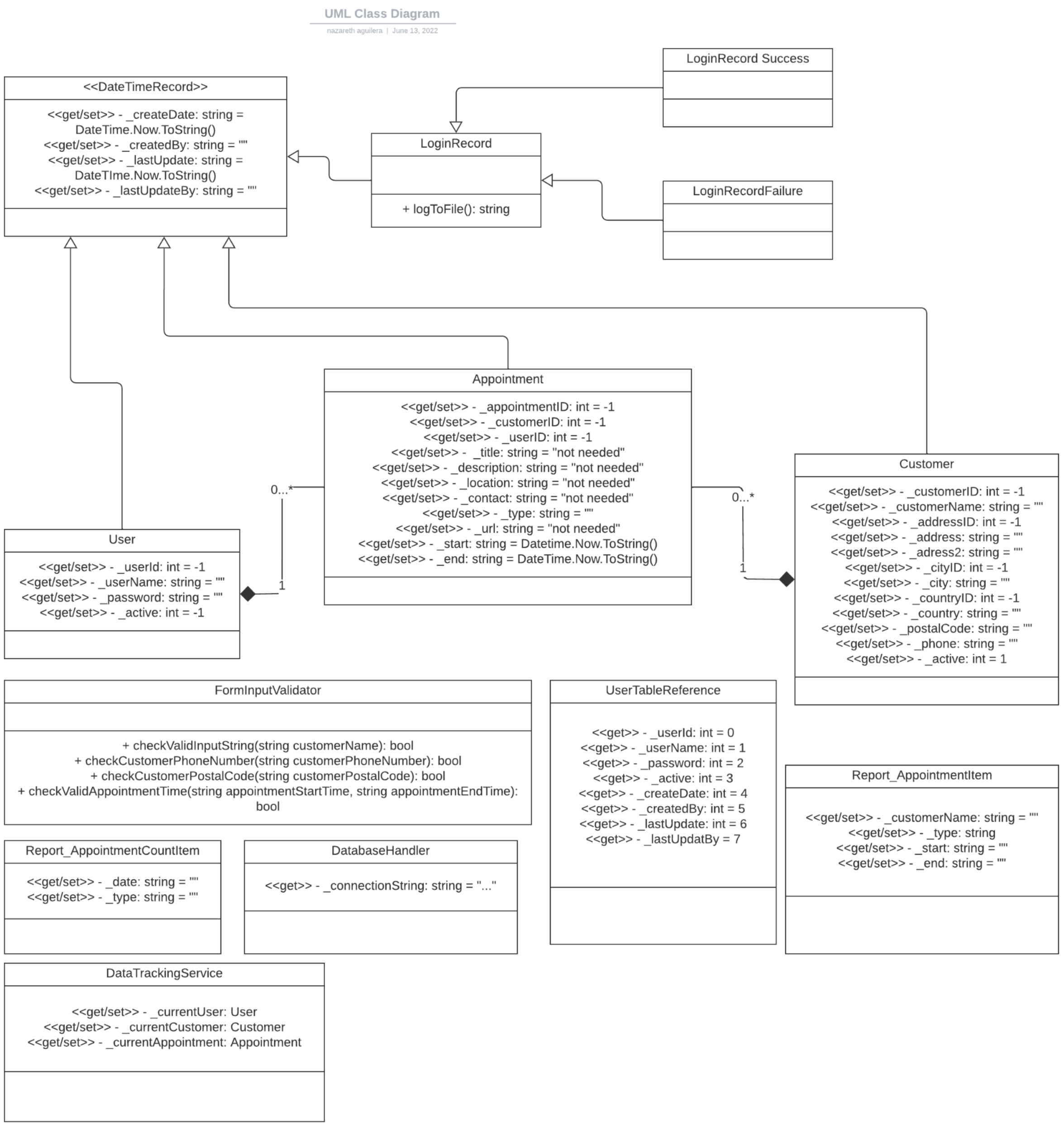
As a note for the diagram shown below, although getters and setters in programming can technically be classified as functions, to help remove redundancy I have included them in the properties section preceding the property that they manipulate. The getters and setters of a class are denoted by the <<get/set>> syntax.

To also help with redundancy, many of the classes inherit properties from the abstract class DateTimeRecord. Inheritance is a programming concept where a parent-child relationship is formed where the child can obtain properties and methods from their parent class, while also adding in new properties and methods of their own.

In the case of the classes LoginRecordSuccess and LoginRecordFailure, they both not only inherit from LoginRecord but the logToFile() method that they inherit, they each have their own twist on the result of calling this method. In programming, inheriting a method like this and then changing it’s behavior is known as an example of polymorphism.

One last note for the diagram presented below, the _connectionString property located in the DatabaseHandler class is shortened for security reasons. The connection string which holds the credentials to access the database is shown here but has been replaced by “...”.

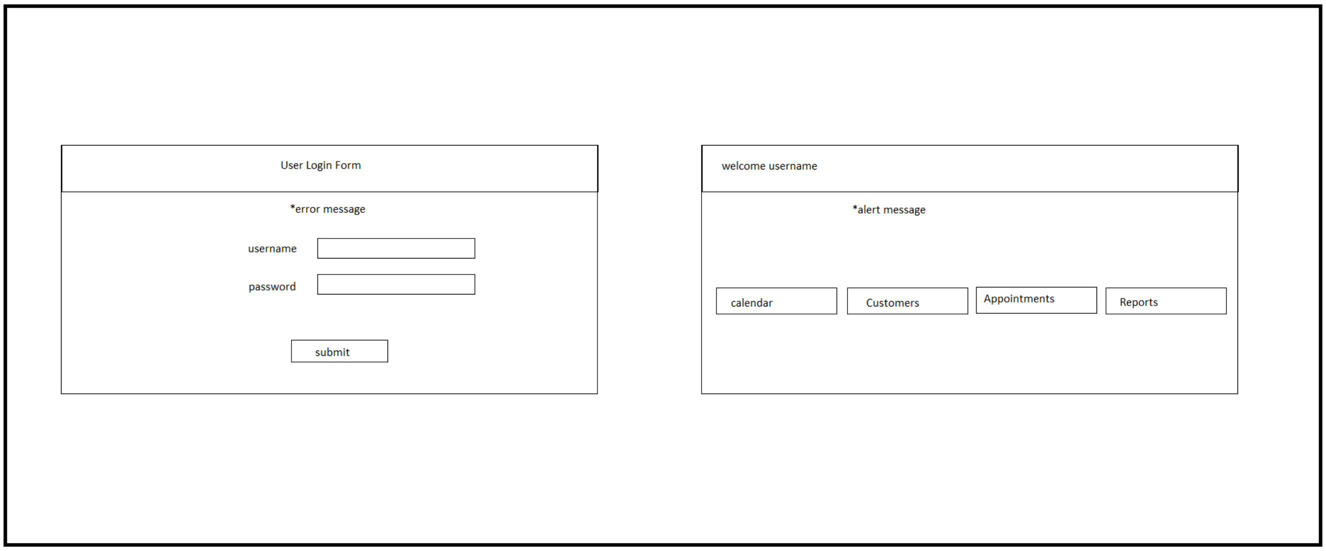
Retire Me Now Scheduling Software Application


UML Class Diagram
nazareth aguilera | June 13, 2022
<<DateTimeRecord>>
<<get/set>> - _createDate: string = DateTime.Now.ToString()
<<get/set>> - _createdBy: string = ""
<<get/set>> - _lastUpdate: string = DateTIme.Now.ToString()
<<get/set>> - _lastUpdateBy: string = ""
LoginRecord Success
LoginRecord
+ logToFile(): string
LoginRecordFailure
Appointment
<<get/set>> - _appointmentID: int = -1
<<get/set>> - _customerID: int = -1
<<get/set>> - _userID: int = -1
<<get/set>> - _title: string = "not needed"
<<get/set>> - _description: string = "not needed"
<<get/set>> - _location: string = "not needed"
<<get/set>> - _contact: string = "not needed"
<<get/set>> - _type: string = ""
<<get/set>> - _url: string = "not needed"
<<get/set>> - _start: string = Datetime.Now.ToString()
<<get/set>> - _end: string = DateTime.Now.ToString()
0...*
1
0...*
1
User
<<get/set>> - _userId: int = -1
<<get/set>> - _userName: string = ""
<<get/set>> - _password: string = ""
<<get/set>> - _active: int = -1
Customer
<<get/set>> - _customerID: int = -1
<<get/set>> - _customerName: string = ""
<<get/set>> - _addressID: int = -1
<<get/set>> - _address: string = ""
<<get/set>> - _adress2: string = ""
<<get/set>> - _cityID: int = -1
<<get/set>> - _city: string = ""
<<get/set>> - _countryID: int = -1
<<get/set>> - _country: string = ""
<<get/set>> - _postalCode: string = ""
<<get/set>> - _phone: string = ""
<<get/set>> - _active: int = 1
FormInputValidator
+ checkValidInputString(string customerName): bool
+ checkCustomerPhoneNumber(string customerPhoneNumber): bool
+ checkCustomerPostalCode(string customerPostalCode): bool
+ checkValidAppointmentTime(string appointmentStartTime, string appointmentEndTime): bool
UserTableReference
<<get>> - _userId: int = 0
<<get>> - _userName: int = 1
<<get>> - _password: int = 2
<<get>> - _active: int = 3
<<get>> - _createDate: int = 4
<<get>> - _createdBy: int = 5
<<get>> - _lastUpdate: int = 6
<<get>> - _lastUpdatBy = 7
Report_AppointmentItem
<<get/set>> - _customerName: string = ""
<<get/set>> - _type: string
<<get/set>> - _start: string = ""
<<get/set>> - _end: string = ""
Report_AppointmentCountItem
<<get/set>> - _date: string = ""
<<get/set>> - _type: string = ""
DatabaseHandler
<<get>> - _connectionString: string = "..."
DataTrackingService
<<get/set>> - _currentUser: User
<<get/set>> - _currentCustomer: Customer
<<get/set>> - _currentAppointment: Appointment

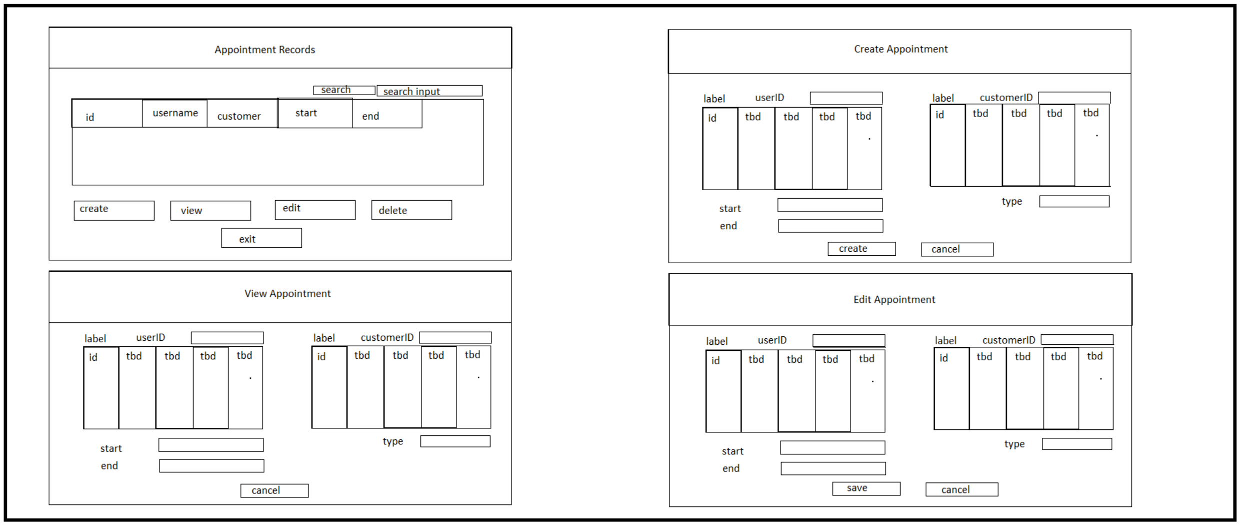
Wireframes

As per the Waterfall model, the wireframes for the Retire Me Now Scheduling Software Application are developed during the Design Phase. Wireframes give a visual representation of how the layout of the application's user interface should be created. These wireframes allow the team to explore how a user may navigate through the application. The wireframes were developed by the development team's User Experience Designer with assistance from the Software Developer. All wireframes should reference the Requirements document to ensure that each feature added addresses a customer need.

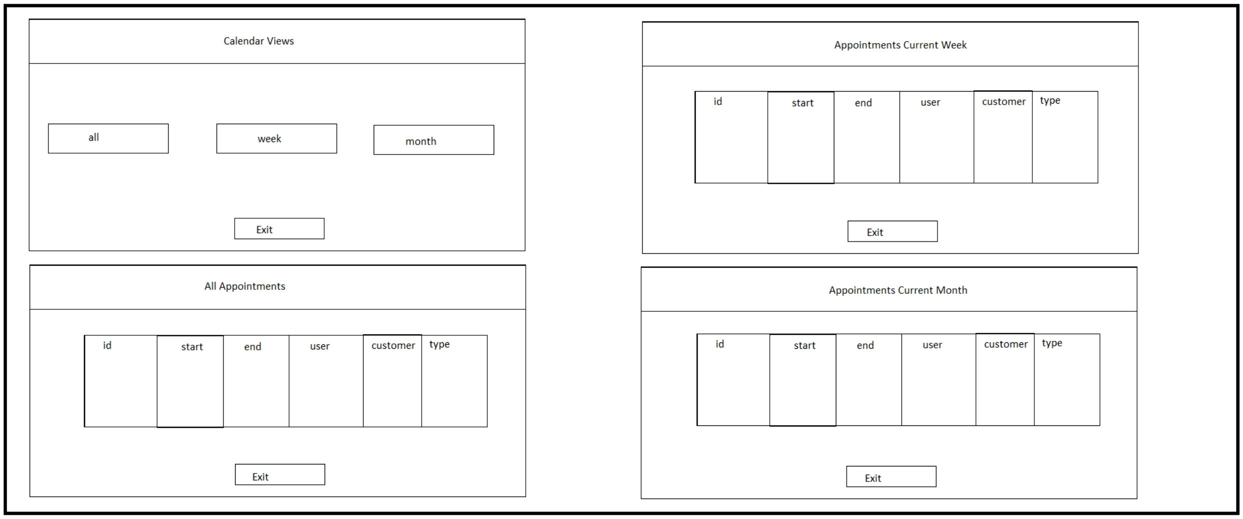
Login / Dashboard Wireframe



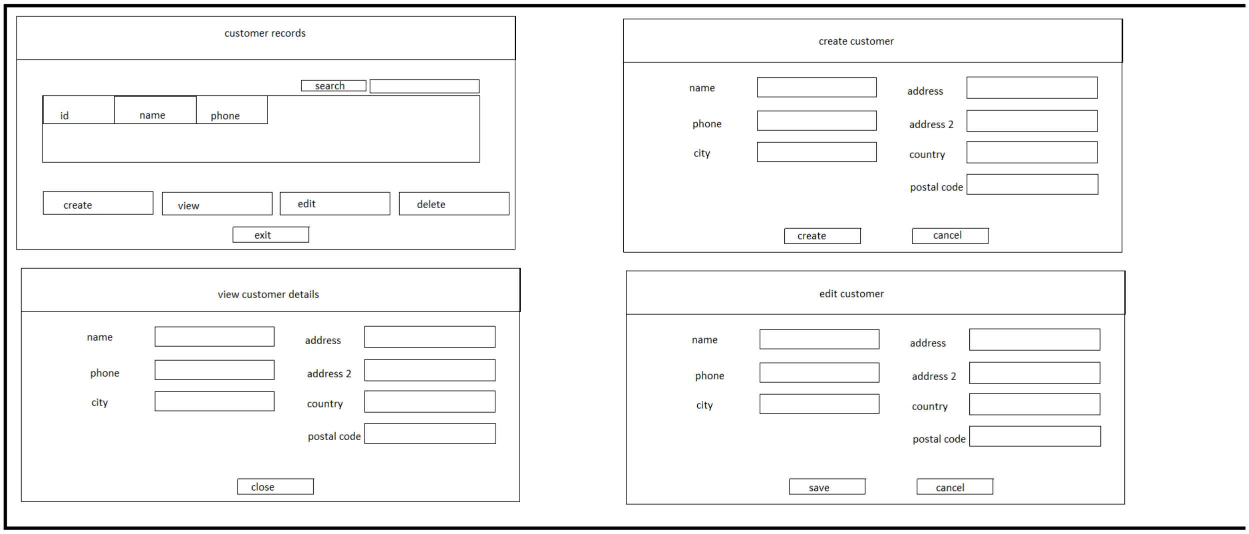
Appointment Records - Create, View, Edit Wireframes



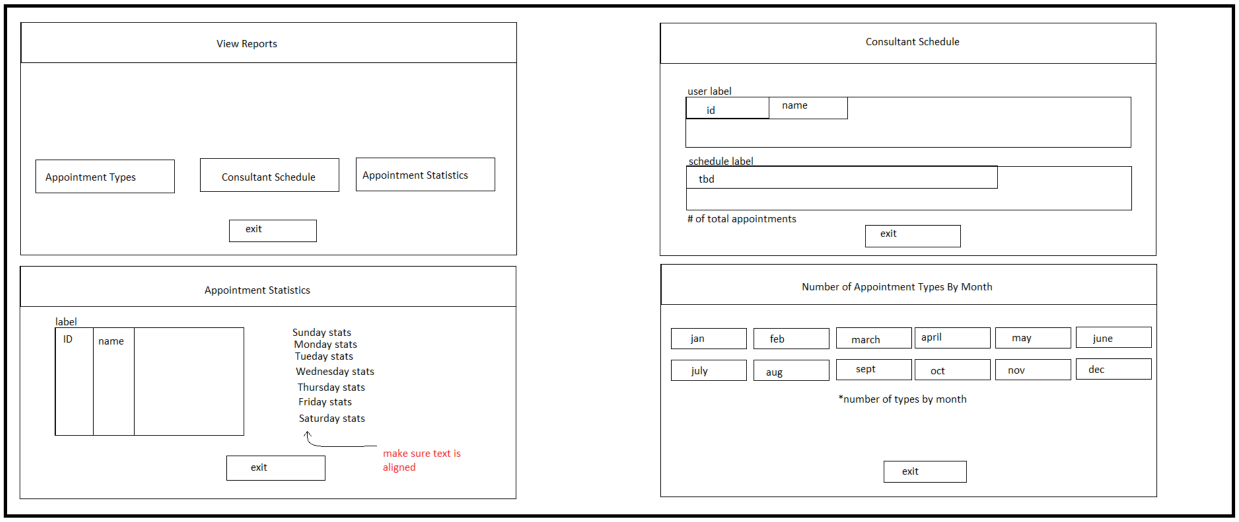
Calendar Views - All Appointments, Week, Month Wireframes



Customer Records - Create, View, Edit Wireframes



Reports - Consultant Schedule, Appointment Statistics, Appointment Types Wireframes



Unit Test Plan

Introduction

During the Testing Phase of the Software Development Life Cycle, a test plan is used to facilitate how the application will be scrutinized in order to ensure that the application only behaves with intended design. For this section, a Unit Test Plan is made that will facilitate how the unit tests will be conducted. A unit test checks code segments of the application and ensures that it behaves as expected. By developing a plan to test many code segments that the application is composed of, it helps remove software defects that cause unintentional behavior.

The unit test described below will show how the customer phone number input is tested to ensure data integrity.

Purpose

When creating a customer, one of the fields requires the user to input a customer's phone number. Since the customer phone number field is a text box, it leaves the possibility that the customer provides invalid information that the database does not expect. A means of validating that data is necessary to ensure data integrity. For this application, MSTEST is a unit test framework that allows the development team to apply various scenarios to a function and check to see if the code will behave as expected and return a valid outcome. There are six unit tests provided that all passed verification.

Overview

The addition of the unit tests had the development team refactor the code to be test-friendly. By moving many of the form validation checks from the forms into the FormInputValidator class, this allowed the codebase to take advantage of code reuse.

Unlike some of the other inputs in the customer form, the phone number is required to only be numbers. The text box will also return a string value, but the phone number needs to be checked to ensure that it represents a number value. The phone number input also needs to meet a length of digits criteria as no phone number can be less than 7 digits and must account for the length of out-of-country phone numbers.

Unit Test Plan

Items

In order to complete the tests, MSTest must be added into the project. Once MSTest has been added, a reference to the Retire Me Now Scheduling Software Application must be added to it's dependencies. After including the application to the dependencies, it must be included into the unit test's headers.

Features

Each test includes:

- A test name
- A hypothetical input that the user could provide.
- A reference to the FormInputValidator class to call the method checkCustomerPhoneNumber();
- An assert method that performs the test and checks if the output of the method is the same as the expected output.

Deliverables

A list result that displays the outcome of each unit test.

Tasks

In order to complete these unit tests:

1. Include all necessary dependencies as stated in Items of this section.
2. In Microsoft Visual Studio, find the toolbar at the top and click on Tests.
3. Click "Run All Tests".
4. Await results.
5. If any tests do not pass, make changes necessary to the method and test again.
6. Change possible input that fulfills the test method's general description of the method.
    a. For example, TestPhoneNumber_10Digits() should have input that is at least 10 digits long.

Needs

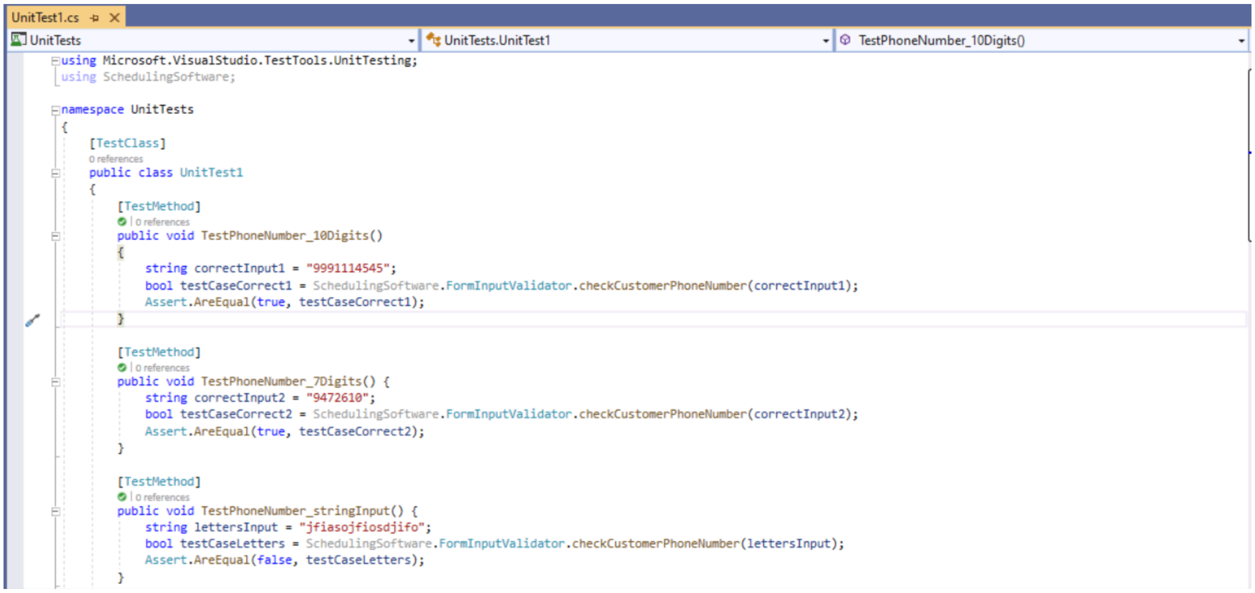
1. Microsoft Visual Studio 2019
2. MSTest, testing .NET 5.0 (current)
3. The Scheduling Software Application included in the headers.
4. The Scheduling Software Application included as a reference to the unit test's dependencies.
5. The FormInputValidator class, using the .checkCustomerPhoneNumber() method.

Pass/Fail Criteria

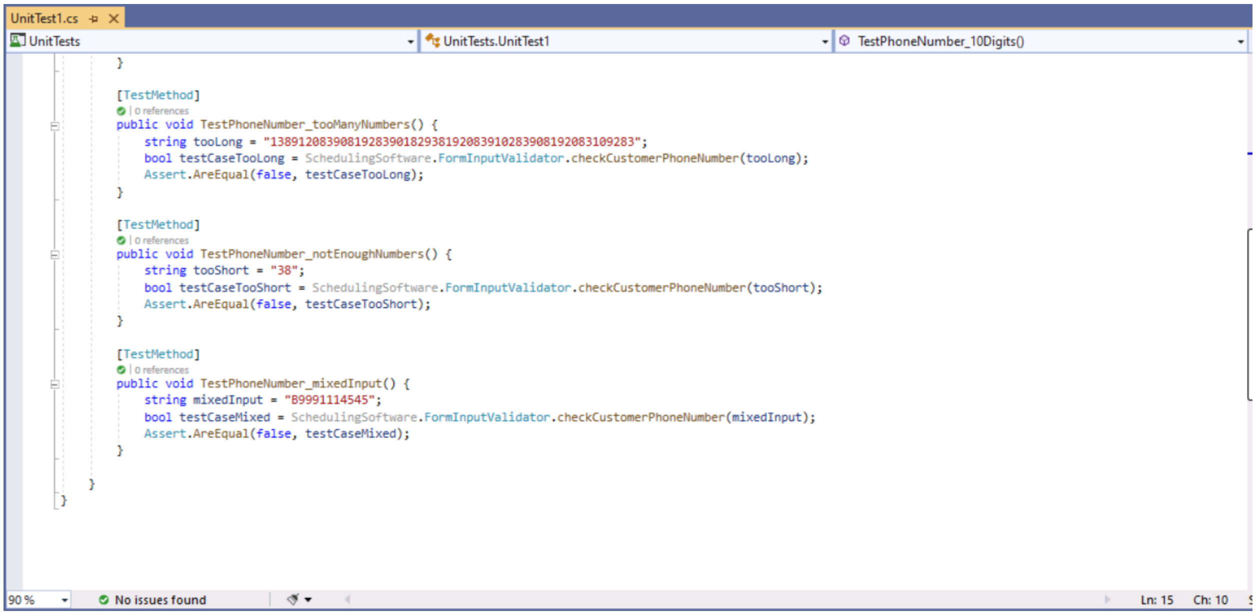
All tests ran successfully as denoted by the green checkmarks in the Results section.

Specifications

The image below shows the code used to run the unit tests.



```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using SchedulingSoftware;

namespace UnitTests
{
    [TestClass]
    0 references
    public class UnitTest1
    {
        [TestMethod]
        | 0 references
        public void TestPhoneNumber_10Digits()
        {
            string correctInput1 = "9991114545";
            bool testCaseCorrect1 = SchedulingSoftware.FormInputValidator.checkCustomerPhoneNumber(correctInput1);
            Assert.AreEqual(true, testCaseCorrect1);
        }

        [TestMethod]
        | 0 references
        public void TestPhoneNumber_7Digits() {
            string correctInput2 = "9472610";
            bool testCaseCorrect2 = SchedulingSoftware.FormInputValidator.checkCustomerPhoneNumber(correctInput2);
            Assert.AreEqual(true, testCaseCorrect2);
        }

        [TestMethod]
        | 0 references
        public void TestPhoneNumber_stringInput() {
            string lettersInput = "jfiasojfiosdjifo";
            bool testCaseLetters = SchedulingSoftware.FormInputValidator.checkCustomerPhoneNumber(lettersInput);
            Assert.AreEqual(false, testCaseLetters);
        }
```



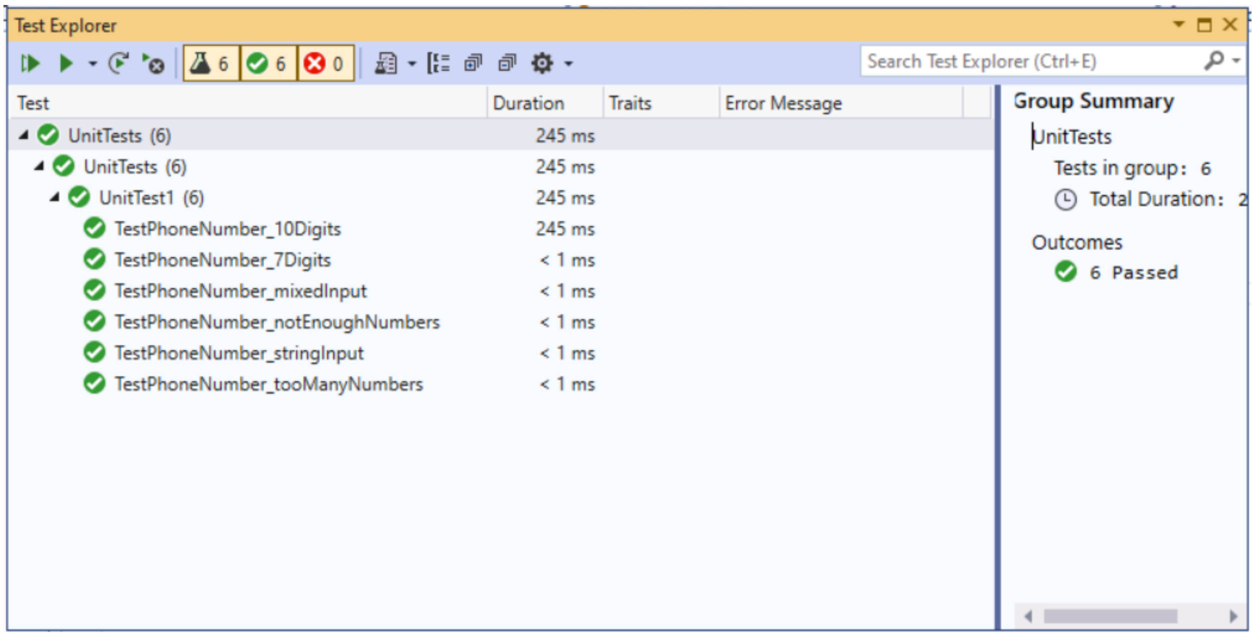
```
        }

        [TestMethod]
        | 0 references
        public void TestPhoneNumber_tooManyNumbers() {
            string tooLong = "1389120839081928390182938192083910283908192083109283";
            bool testCaseTooLong = SchedulingSoftware.FormInputValidator.checkCustomerPhoneNumber(tooLong);
            Assert.AreEqual(false, testCaseTooLong);
        }

        [TestMethod]
        | 0 references
        public void TestPhoneNumber_notEnoughNumbers() {
            string tooShort = "38";
            bool testCaseTooShort = SchedulingSoftware.FormInputValidator.checkCustomerPhoneNumber(tooShort);
            Assert.AreEqual(false, testCaseTooShort);
        }

        [TestMethod]
        | 0 references
        public void TestPhoneNumber_mixedInput() {
            string mixedInput = "B9991114545";
            bool testCaseMixed = SchedulingSoftware.FormInputValidator.checkCustomerPhoneNumber(mixedInput);
            Assert.AreEqual(false, testCaseMixed);
        }

    }
}
```

Procedures

To run the unit tests, the MSTest testing framework was properly configured to reference the application. All necessary components are listed in the Needs portion of this section. Then, locate the Tests button located on the toolbar at the top of the screen in Visual Studio. Click “Run All Tests”. The MSTest will begin running all the tests that are marked by the [TestMethod] attribute. The results of this procedure are listed below under the Results portion of this section.

Results

The results of the unit test are displayed in the image below.



Test	Duration	Traits	Error Message
✓ UnitTests (6)	245 ms		
✓ UnitTests (6)	245 ms		
✓ UnitTest1 (6)	245 ms		
✓ TestPhoneNumber_10Digits	245 ms		
✓ TestPhoneNumber_7Digits	< 1 ms		
✓ TestPhoneNumber_mixedInput	< 1 ms		
✓ TestPhoneNumber_notEnoughNumbers	< 1 ms		
✓ TestPhoneNumber_stringInput	< 1 ms		
✓ TestPhoneNumber_tooManyNumbers	< 1 ms		

Group Summary

UnitTests

Tests in group: 6

Total Duration: 2

Outcomes

✓ 6 Passed

Source Code

The code for the Retire Me Now Schedule Software Application has been included in the .zip file that was submitted alongside this document. The folder is labeled “Nazareth Aguilera Capstone Project - v4 - final draft”.

Link to Live Version

There is no live version of this application currently available. This is a stand-alone application and must be created in Visual Studio. Please download the necessary files to properly compile the provided .zip file. A client_schdule schema must also be created in the MySQL Workbench that allows a connection through a local connection string. Use the following credentials to begin:

- Username: test
- Password: test

Application Maintenance Guide

Prerequisites

The following is a list of software development environments needed for maintenance.

- Windows 10+ Operating System
- Microsoft Visual Studio 2019
- C# Programming Language
- .NET Framework 4.7+
- MSTest (for unit testing)
- MySQL Workbench 8.0

Installation and Debugging

1. Ensure that all the necessary files have been downloaded to your local machine. The project files can be found in the .zip folder labeled “Nazareth Aguilera Capstone Project - v4 - final draft”.
2. Extract all files into a known location on your computer.
3. In Visual Studio 2019, Click Open Project. Then inside the “Nazareth Aguilera Capstone Project - v4 - final draft”, you will find the file called “SchedulingSoftware.sln”.
4. Select the file and click open.
5. The application should be open in Visual Studio 2019 and ready for you to begin maintenance.

User Guide

Introduction

This User Guide is meant to be used as a manual to help the user learn how to use each feature of the Retire Me Now Scheduling Software Application. Each section of this guide states what the purpose of the feature is and how the user may use it. Included is a short description of the application page, an image of what the application page looks like, and the available actions that the user can perform on a given application page.

Installation

The Retire Me Now Scheduling Software Application should already be installed for you by an IT professional. Contact our organization or your office’s IT Support Specialist for further assistance.

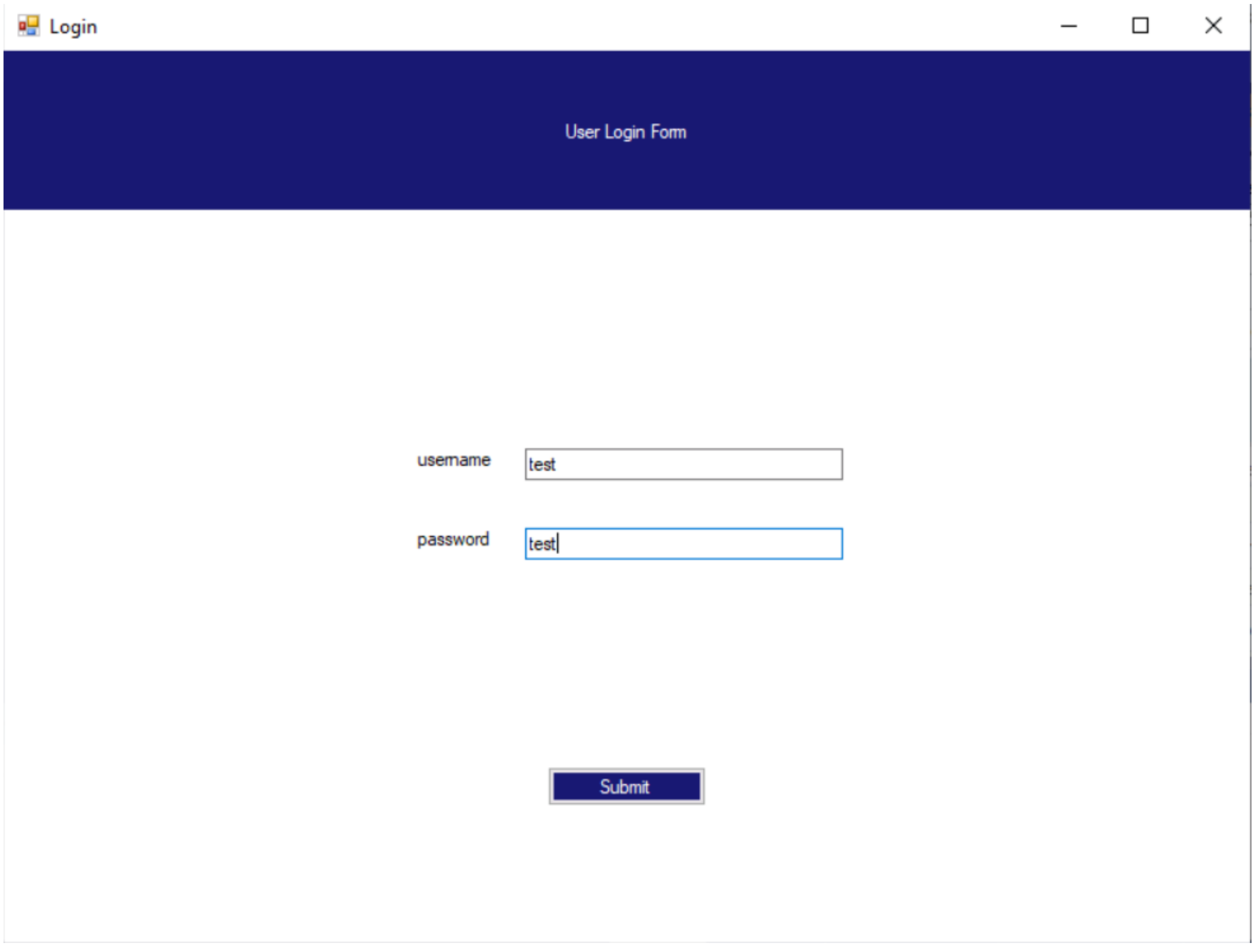
User Creation

The Retire Me Now Scheduling Software Application does not currently have a way to create a new user for the system through its user interface. A new user must be created through a database query using MySQL Workbench. All users using this application should already have an account set up for you by an IT professional. Contact our organization or your office’s IT Support Specialist for further assistance.

Login

The User Login Form is the entry point of the Retire Me Now Scheduling Software Application. This is the first screen presented to the user on startup. The user will be able to use their login credentials in order to navigate to the Dashboard. The User Login Form has a language detection that will ask the user for their credentials in Spanish if the user's Operating System settings are set to a language other than English.

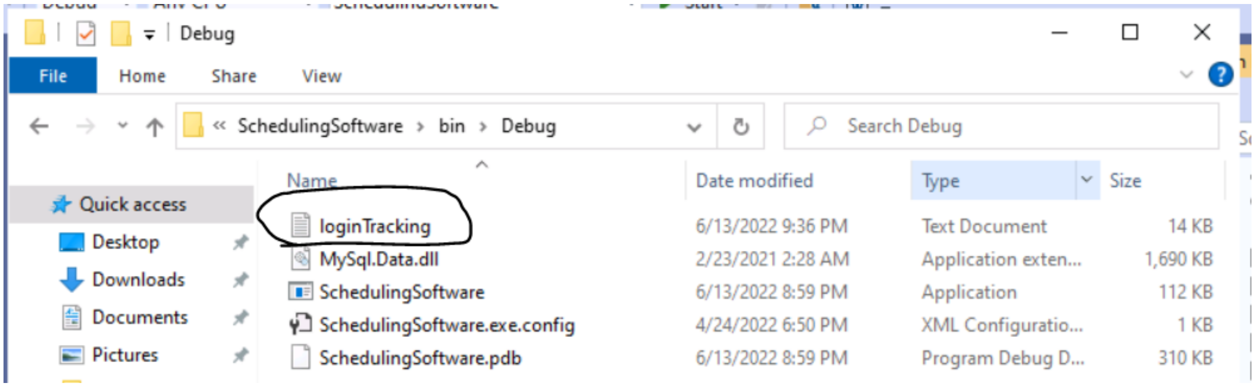
If the user tries to login using invalid credentials, an error message will appear.



username test

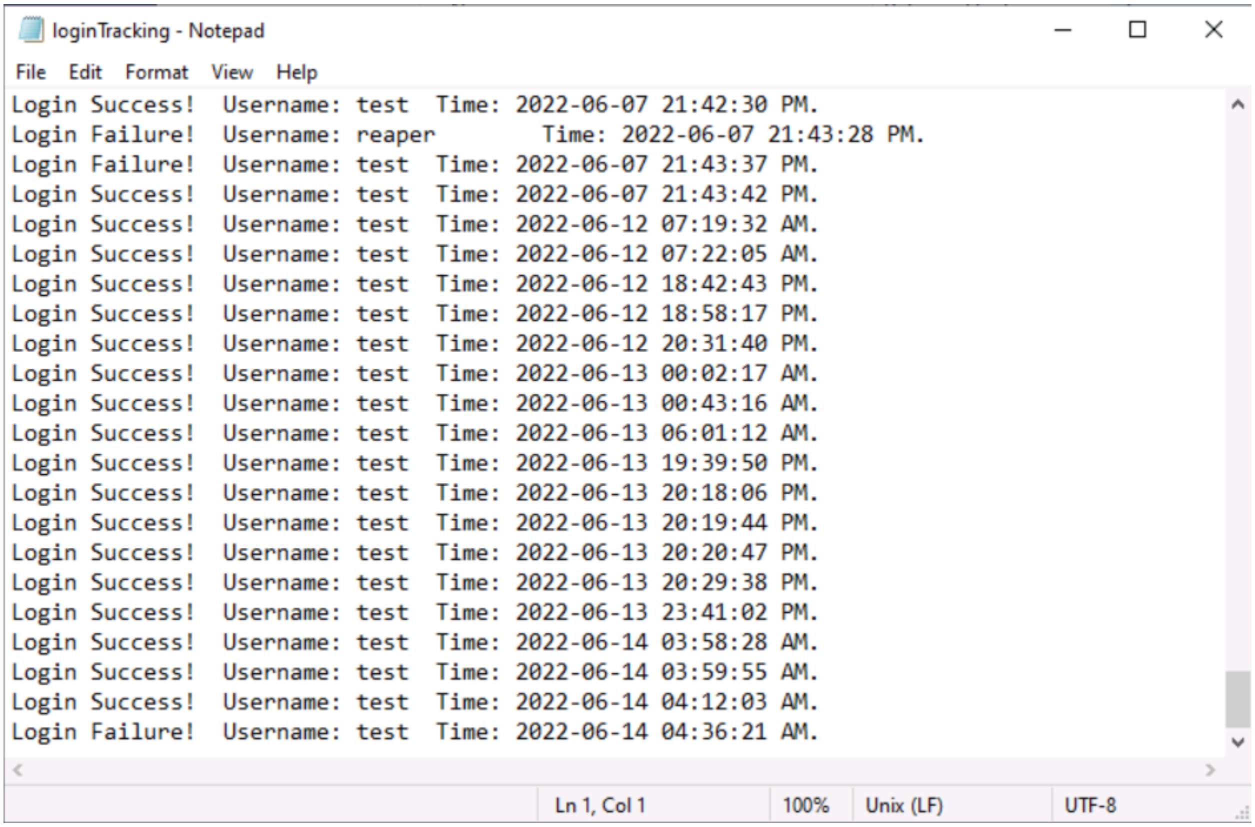
password test

Each attempt to login by the user is recorded in a file named “loginTracking.txt”. This file can be reached by finding the root directory of the application, then access each subsequent folder in this order: Scheduling Software > bin > Debug > loginTracking.txt.



Name	Date modified	Type	Size
loginTracking	6/13/2022 9:36 PM	Text Document	14 KB
MySql.Data.dll	2/23/2021 2:28 AM	Application exten...	1,690 KB
SchedulingSoftware	6/13/2022 8:59 PM	Application	112 KB
SchedulingSoftware.exe.config	4/24/2022 6:50 PM	XML Configuratio...	1 KB
SchedulingSoftware.pdb	6/13/2022 8:59 PM	Program Debug D...	310 KB

The attempted logins will be displayed here. The status of the login, the username attempted, and the time (in UTC) is recorded.



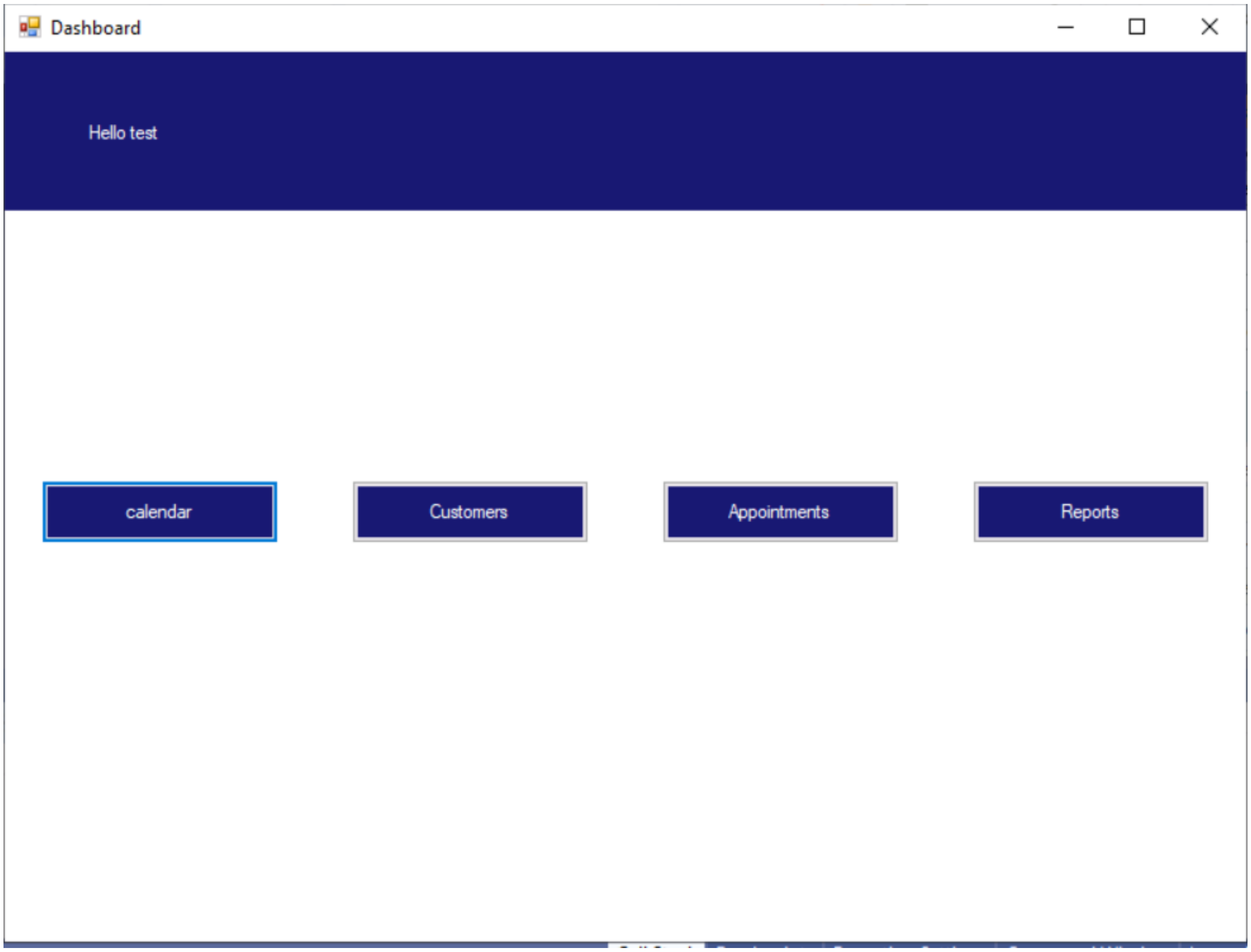
```
Login Success!  Username: test  Time: 2022-06-07 21:42:30 PM.
Login Failure!  Username: reaper        Time: 2022-06-07 21:43:28 PM.
Login Failure!  Username: test  Time: 2022-06-07 21:43:37 PM.
Login Success!  Username: test  Time: 2022-06-07 21:43:42 PM.
Login Success!  Username: test  Time: 2022-06-12 07:19:32 AM.
Login Success!  Username: test  Time: 2022-06-12 07:22:05 AM.
Login Success!  Username: test  Time: 2022-06-12 18:42:43 PM.
Login Success!  Username: test  Time: 2022-06-12 18:58:17 PM.
Login Success!  Username: test  Time: 2022-06-12 20:31:40 PM.
Login Success!  Username: test  Time: 2022-06-13 00:02:17 AM.
Login Success!  Username: test  Time: 2022-06-13 00:43:16 AM.
Login Success!  Username: test  Time: 2022-06-13 06:01:12 AM.
Login Success!  Username: test  Time: 2022-06-13 19:39:50 PM.
Login Success!  Username: test  Time: 2022-06-13 20:18:06 PM.
Login Success!  Username: test  Time: 2022-06-13 20:19:44 PM.
Login Success!  Username: test  Time: 2022-06-13 20:20:47 PM.
Login Success!  Username: test  Time: 2022-06-13 20:29:38 PM.
Login Success!  Username: test  Time: 2022-06-13 23:41:02 PM.
Login Success!  Username: test  Time: 2022-06-14 03:58:28 AM.
Login Success!  Username: test  Time: 2022-06-14 03:59:55 AM.
Login Success!  Username: test  Time: 2022-06-14 04:12:03 AM.
Login Failure!  Username: test  Time: 2022-06-14 04:36:21 AM.
```

Available actions:

1. Enter username and password in their respective input fields.
2. Click the Submit button to attempt to login to the application.
3. Click the X located in the upper right corner to close the application.

Dashboard

The Dashboard is the main menu of the application while the user is logged in. The user will always return to the Dashboard after exiting the other features of this application. The Dashboard displays a welcome message with the user's username. An alert message will also appear if the user has an appointment with a customer within the next 15 minutes of the user logging into the application.

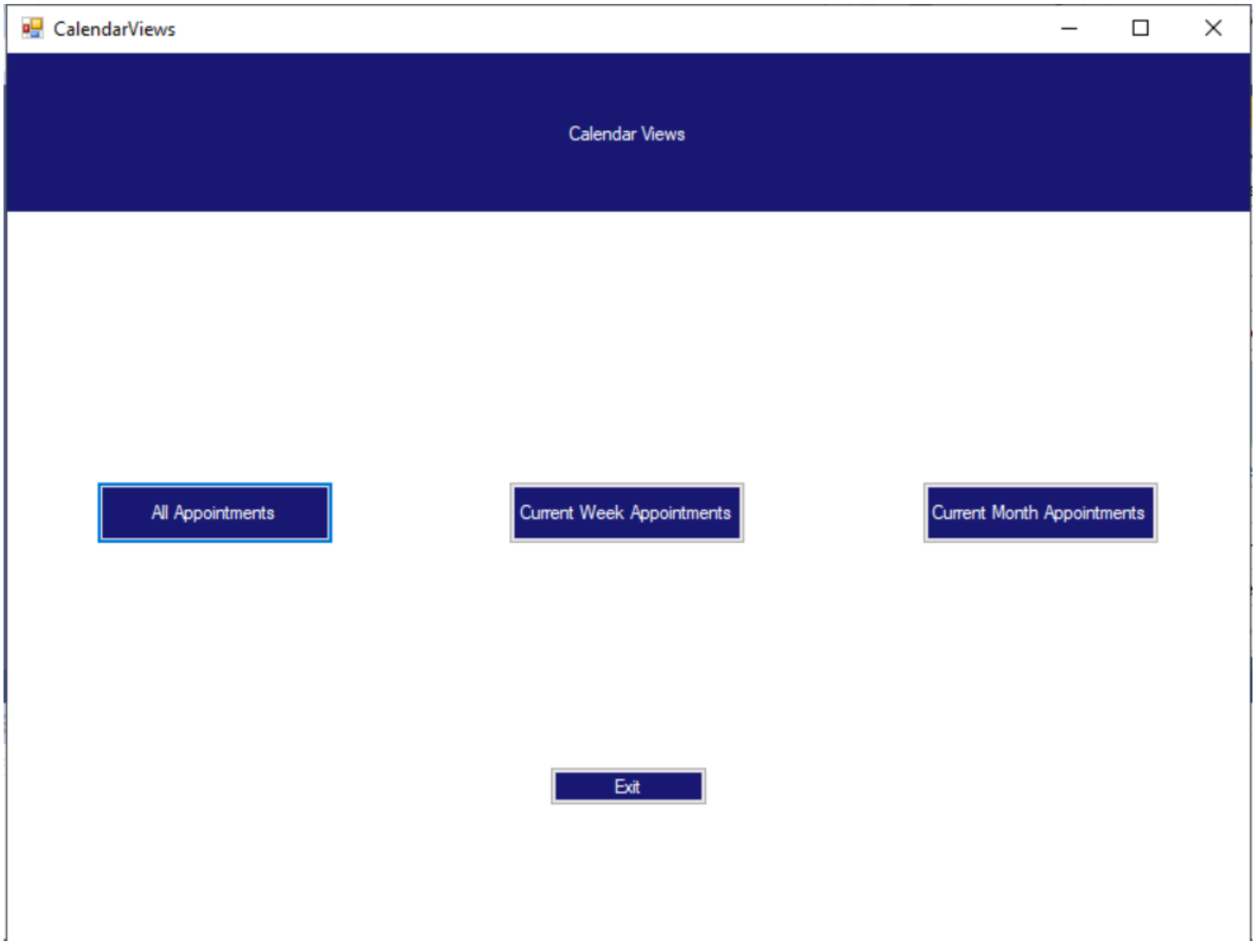


Available actions:

1. Click the Calendar button to open the Calendar Views menu.
2. Click the Customers button to open the Customer Records menu.
3. Click the Appointments button to open the Appointment Records menu.
4. Click the Reports button to open the Reports menu.
5. Click the X in the upper right corner to close the application.

Calendar

The Calendar menu allows the user to view different options for filtering the list of appointments. The user can choose between viewing all appointments, viewing only appointments in the current week, or viewing appointments in the current month.

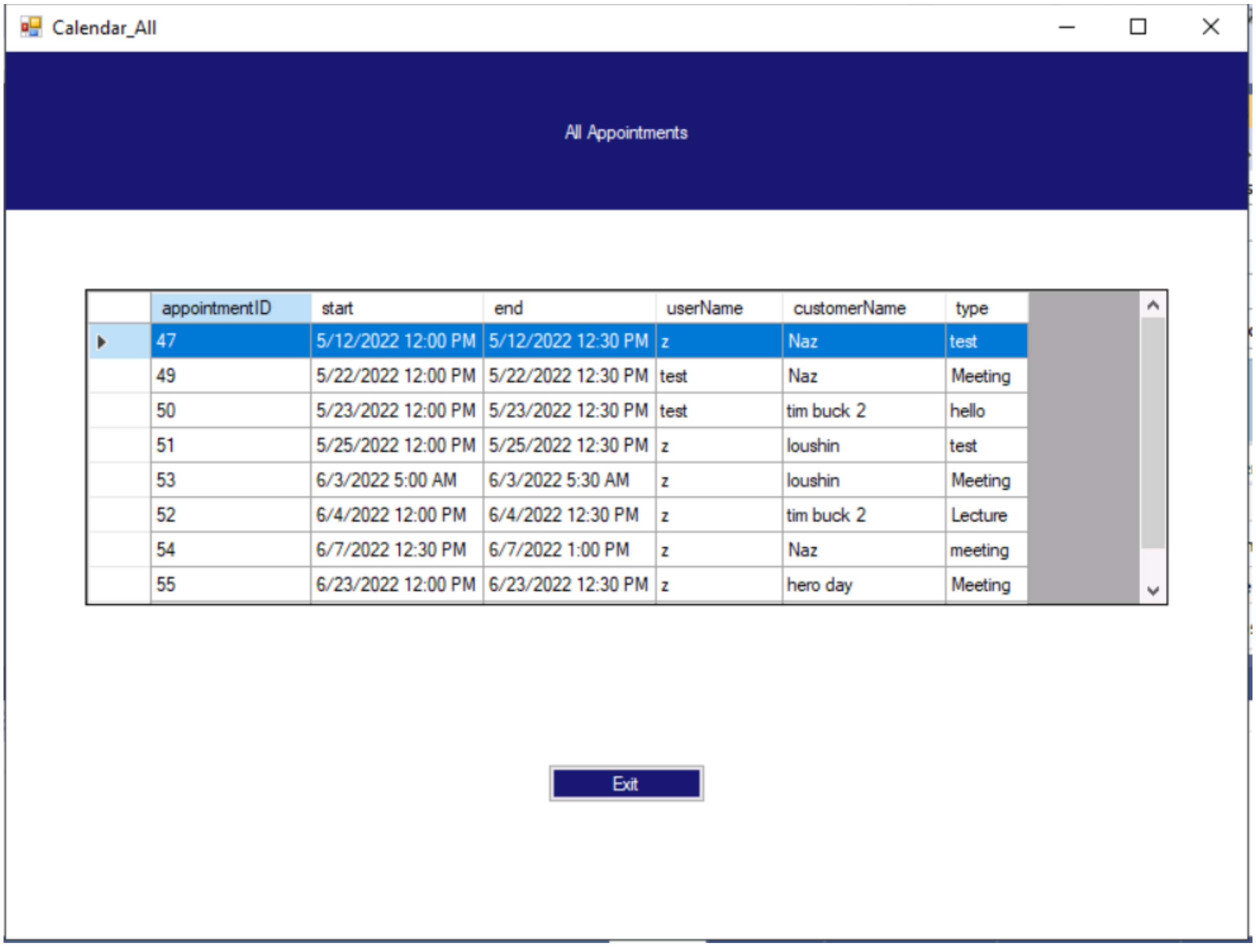


Available actions:

1. Click the All Appointments button to view a listing of all appointments that have been scheduled.
2. Click the Current Week Appointments button to view a listing of all appointments that are set during the current week.
3. Click the Current Month Appointments button to view a listing of all appointments that are set during the current week.
4. Click the Exit button or the X located in the upper right corner to return to the main menu.

Calendar - All Appointments

The All Appointments screen will display all appointments that have been scheduled. Current Week and Current Month Appointments are functionally similar and are not included in this guide. Please use this section as a guide for available actions for those screens as well.



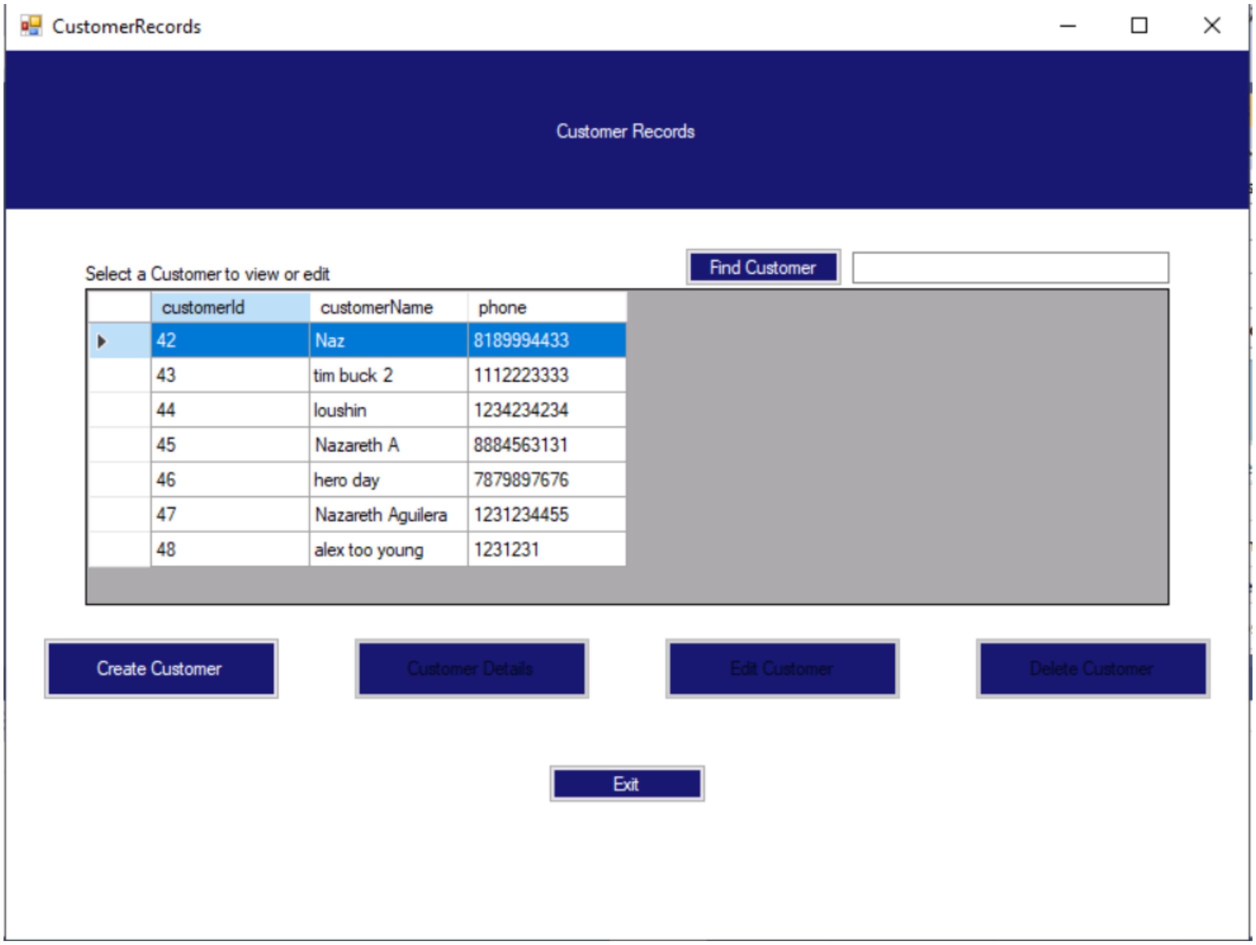
appointmentID	start	end	userName	customerName	type
47	5/12/2022 12:00 PM	5/12/2022 12:30 PM	z	Naz	test
49	5/22/2022 12:00 PM	5/22/2022 12:30 PM	test	Naz	Meeting
50	5/23/2022 12:00 PM	5/23/2022 12:30 PM	test	tim buck 2	hello
51	5/25/2022 12:00 PM	5/25/2022 12:30 PM	z	loushin	test
53	6/3/2022 5:00 AM	6/3/2022 5:30 AM	z	loushin	Meeting
52	6/4/2022 12:00 PM	6/4/2022 12:30 PM	z	tim buck 2	Lecture
54	6/7/2022 12:30 PM	6/7/2022 1:00 PM	z	Naz	meeting
55	6/23/2022 12:00 PM	6/23/2022 12:30 PM	z	hero day	Meeting

Available actions:

1. Click on a cell in the dataGridView to highlight an appointment.
2. Click the Exit button or the X located in the upper right corner to return to the Calendar menu.

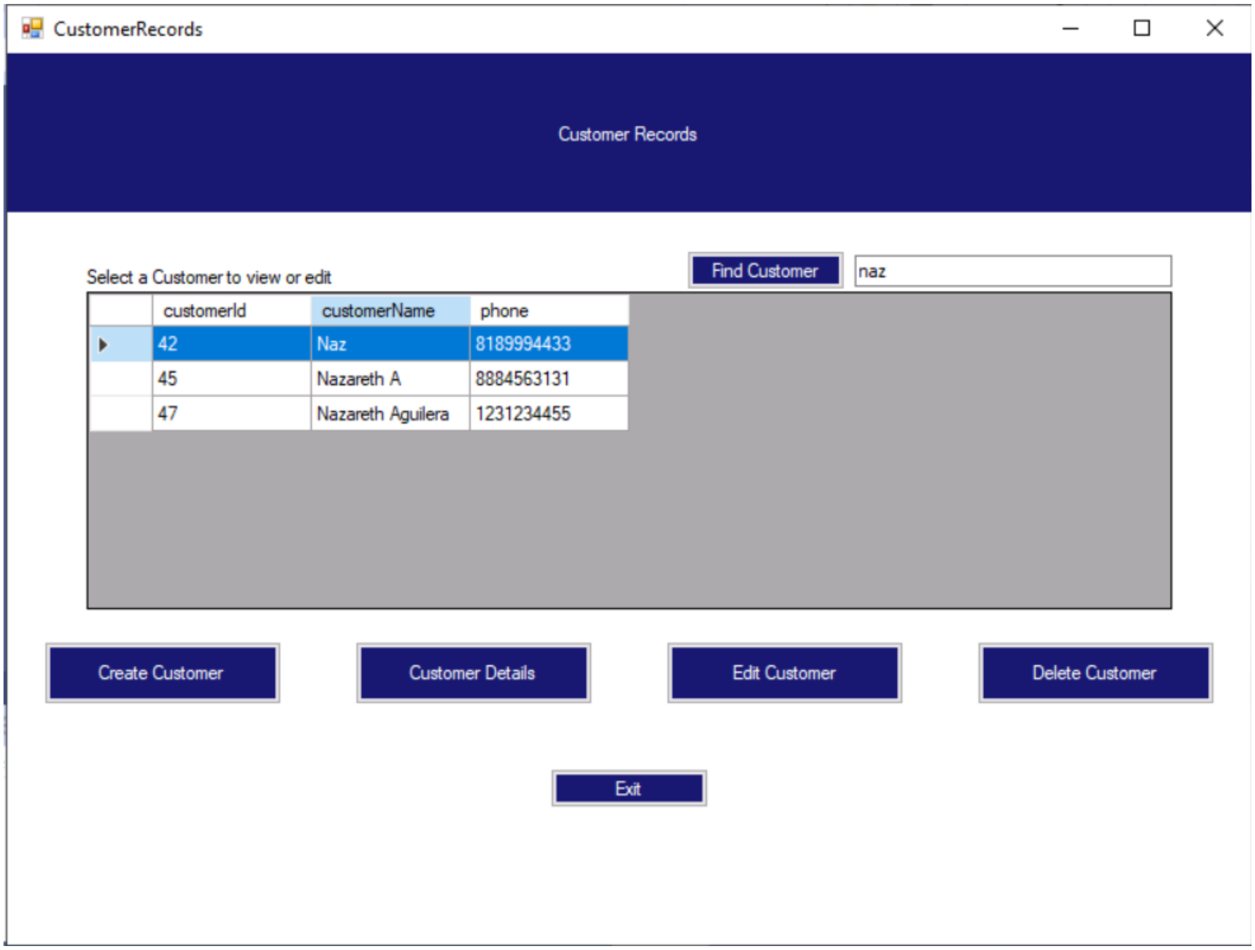
Customer Records

This menu is where the user can perform actions involving the creation, editing, and deleting of a Customer. The user may also view a Customer for more detail. The dataGridView displays all Customers currently in the database. The user must select a Customer before the buttons to View, Edit, or Delete the customer become available. The user can also filter the dataGridView with the Find Customer search function located in the top right corner.



customerId	customerName	phone
42	Naz	8189994433
43	tim buck 2	1112223333
44	loushin	1234234234
45	Nazareth A	8884563131
46	hero day	7879897676
47	Nazareth Aguilera	1231234455
48	alex too young	1231231

The user can type into the Find Customer input field to filter the list of customerNames. See the following image for an example:



Find Customer: naz

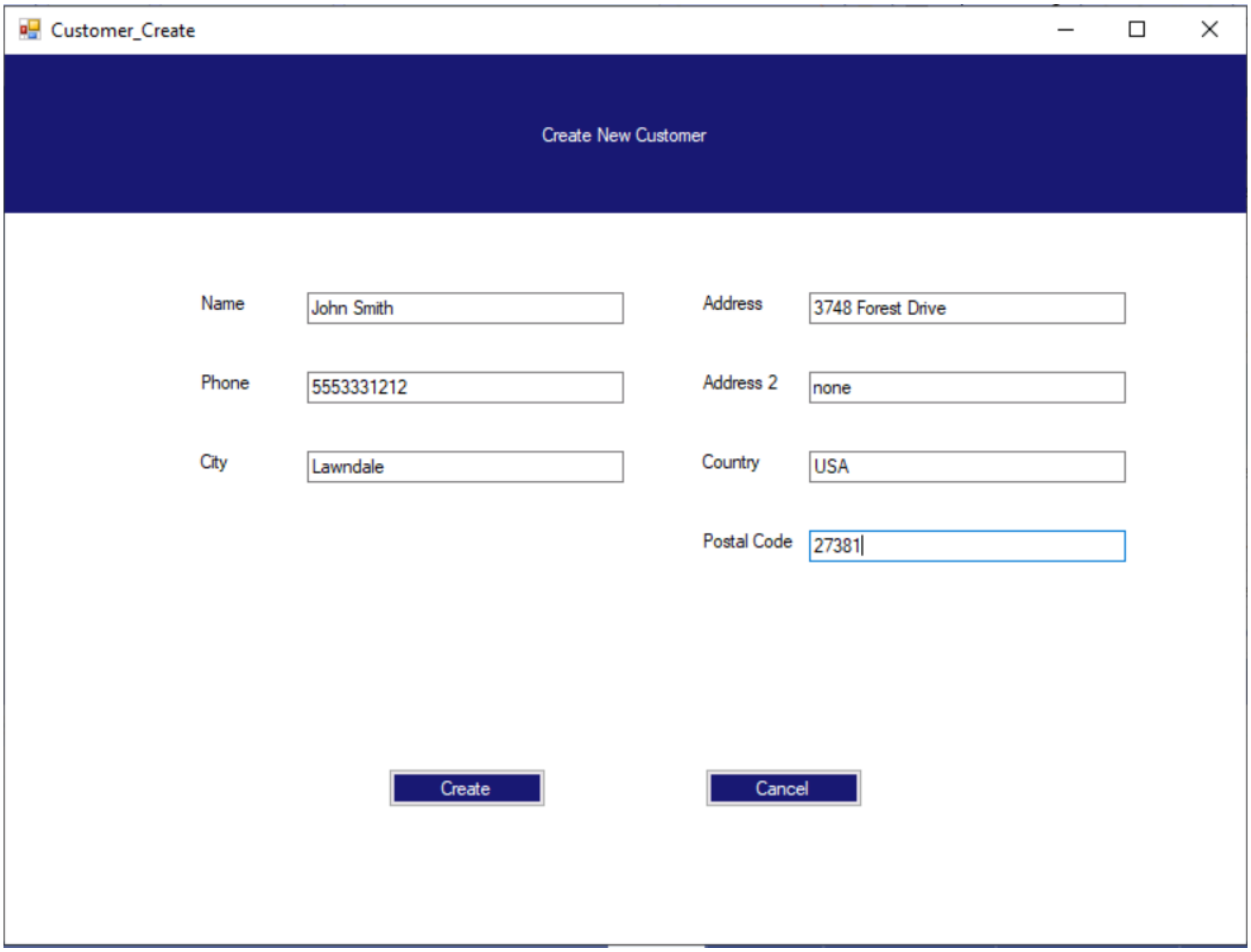
customerId	customerName	phone
42	Naz	8189994433
45	Nazareth A	8884563131
47	Nazareth Aguilera	1231234455

Available actions:

1. Click on a cell in the dataGridView to enable the View Customer Details, Edit Customer and Delete Customer buttons.
2. Click Create Customer to be prompted a form to create a new customer.
3. Click View Customer Details to be prompted a form that will display customer information.
4. Click Edit Customer to be prompted a form that will allow the user to change information about the selected customer.
5. Type in the Find Customer Textbox to filter the dataGridView listings by the customerName.
6. Click the Exit button or the X located in the upper right corner to return to the main menu.

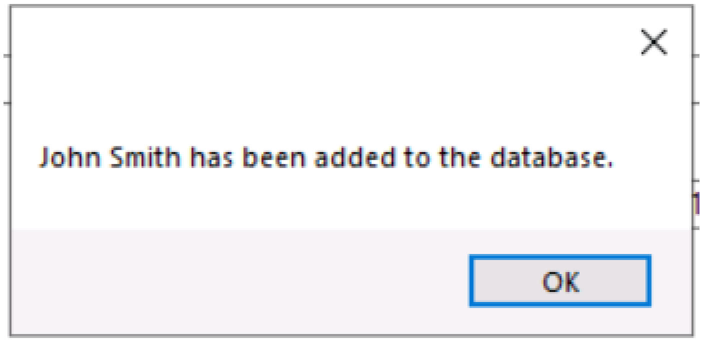
Customer Records - Create a new Customer

In this section, the user can create a new customer to be added to the database. Initially the input fields will have a red color which signifies that the input is invalid. Once the user enters valid input into the form, the field will change to a white background color which signifies that the input is valid. Once the user is done filling out the form, the Create button will be enabled. The user can then create the customer and will be returned to the Dashboard.



Name: John Smith
Phone: 5553331212
City: Lawndale
Address: 3748 Forest Drive
Address 2: none
Country: USA
Postal Code: 27381

Once the user clicks the Create button, the user will be prompted by a message that tells the user if a customer has been successfully added to the database. The prompt will tell the user if there is an error. In the case that a customer has been successfully added to the database, the user will receive a message similar to the following:



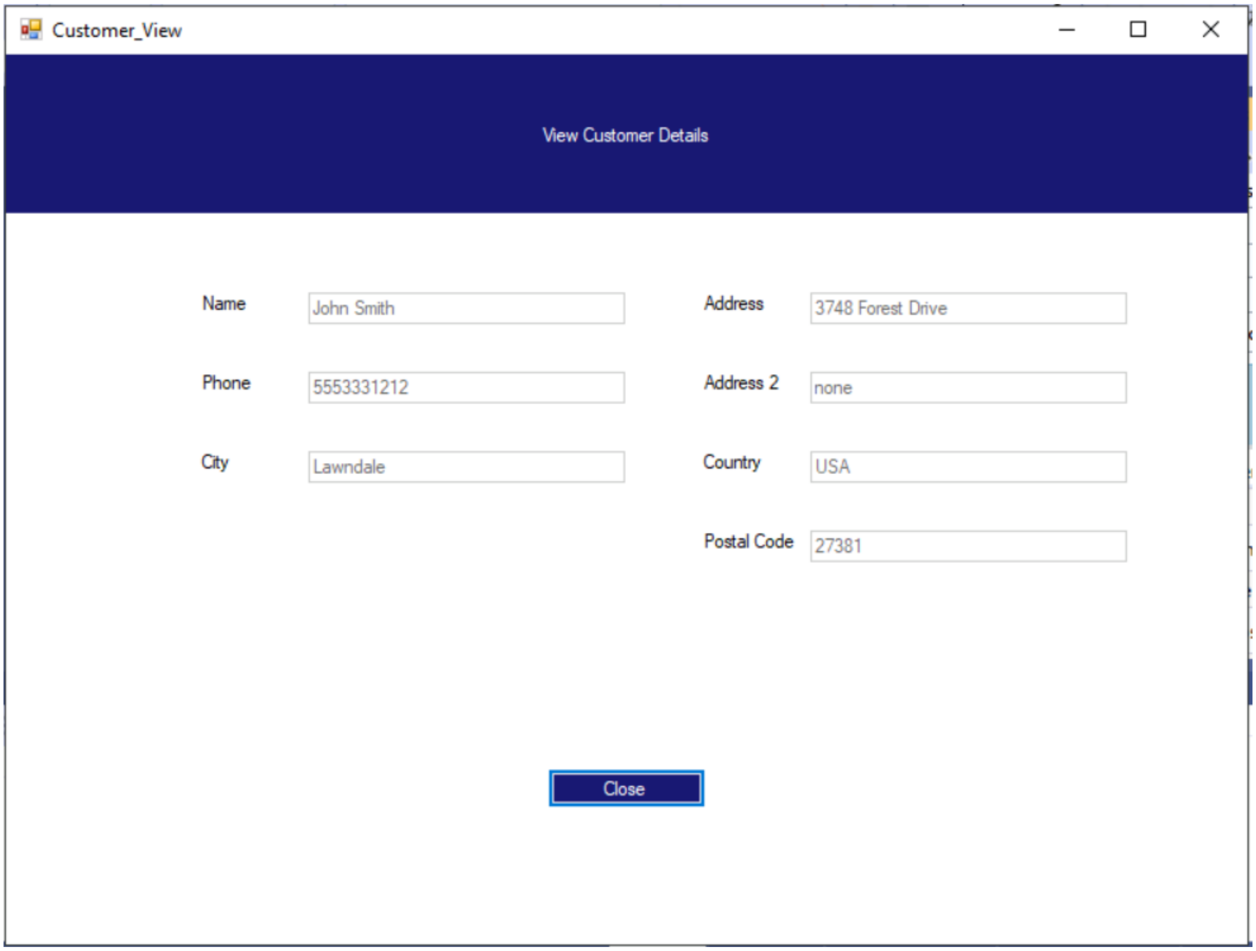
John Smith has been added to the database.

Available actions:

1. Fill out all form fields until the background color of each form is white.
2. Click the Create button to submit the form and add the user to the database.
3. Click the Cancel button or the X located in the upper right corner to return to the Customer Records menu.

Customer Records - View Customer Details

In this section, the user can view further information about a customer that is not shown on the Customer Records menu.



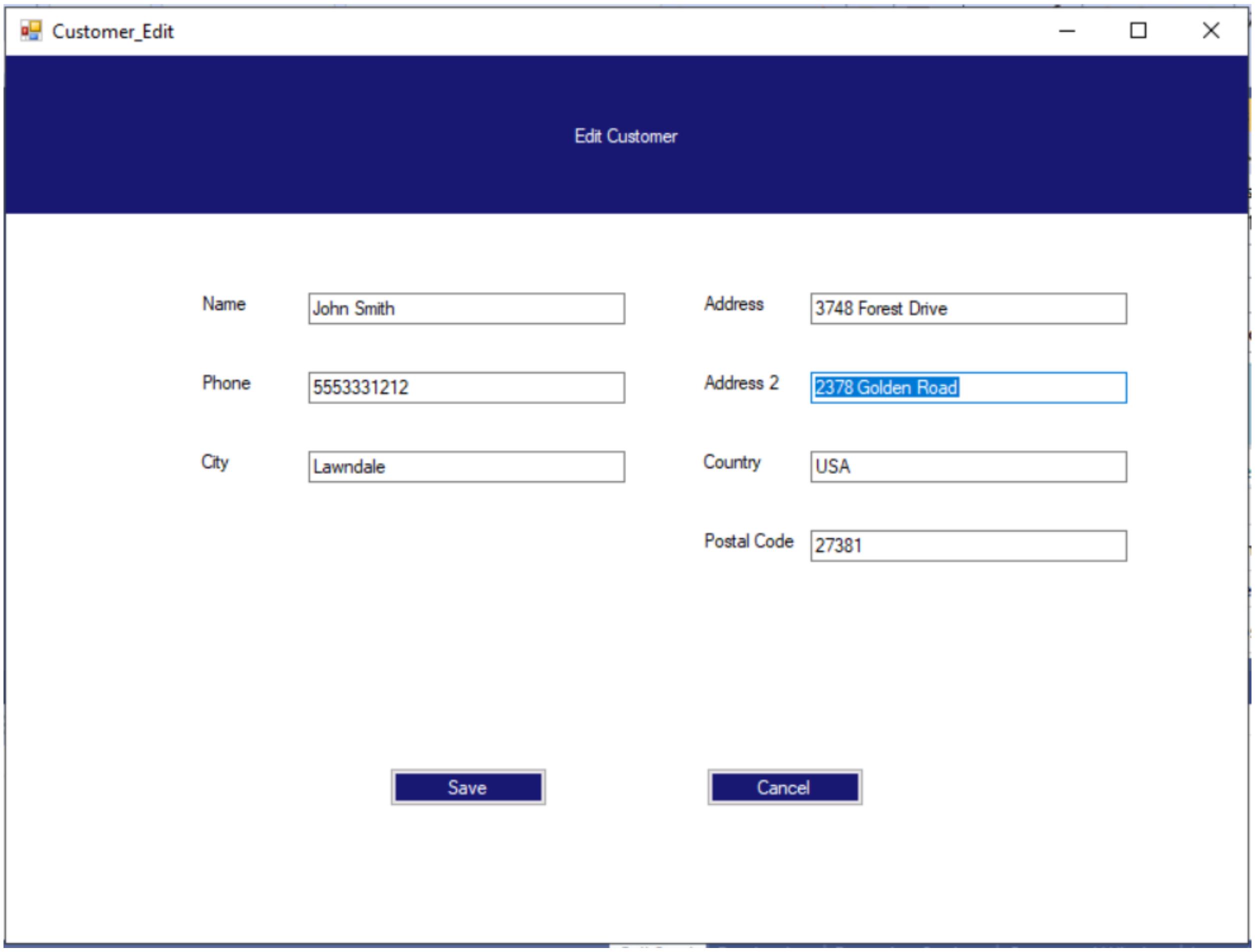
Name: John Smith
Phone: 5553331212
City: Lawndale
Address: 3748 Forest Drive
Address 2: none
Country: USA
Postal Code: 27381

Available actions:

1. Click the Close button or the X located in the upper right corner to return to the Customer Records menu.

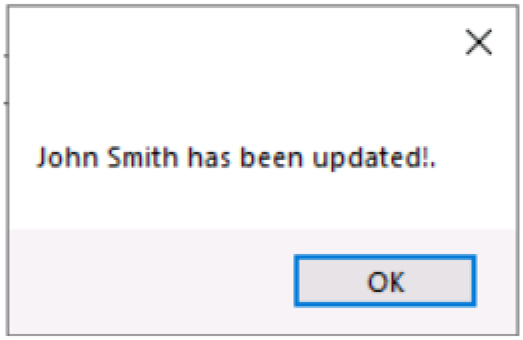
Customer Records - Edit Customer

In this section, the user can edit the information about a user that was selected on the Customer Records menu. If the user changes any input field so that the information becomes invalid, the input field will become red, signifying that the data is invalid. If there are any invalid input fields, the Save button will become disabled. If all the input fields are valid, they will have a white colored background color and the Save button will become enabled.



Name: John Smith
Phone: 5553331212
City: Lawndale
Address: 3748 Forest Drive
Address 2: 2378 Golden Road
Country: USA
Postal Code: 27381

Once the user clicks the Save button, a prompt will appear that shows the status of the request to update the user. If the update is unsuccessful, an error message will appear. If the update is successful, the user will receive a message similar to the following:



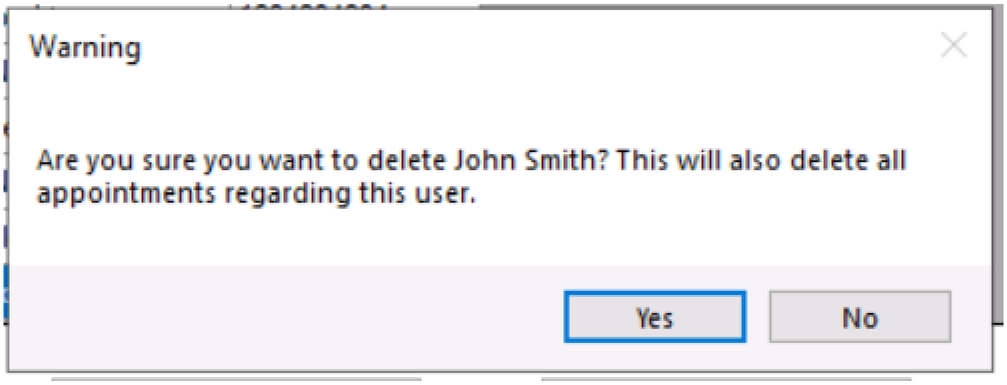
John Smith has been updated!.

Available actions:

1. Change the text of the input fields as needed.
2. Click the Save button to update the customer information.
3. Click the Cancel button or the X located in the upper right corner to return to the Customer Records menu.

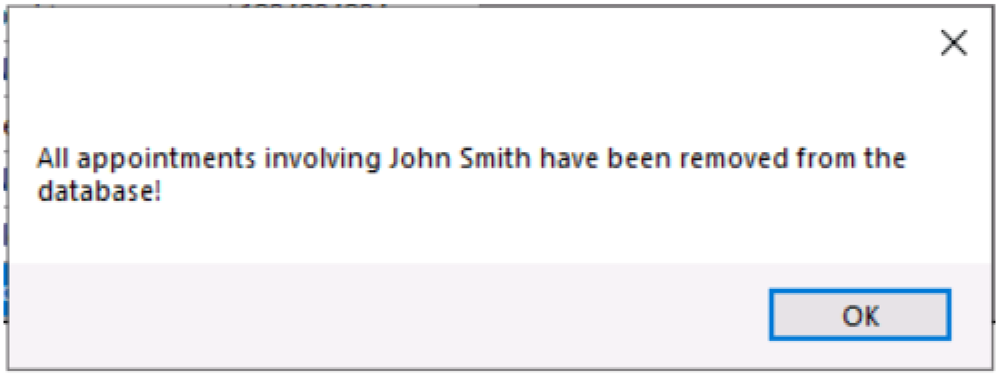
Customer Records - Delete Customer

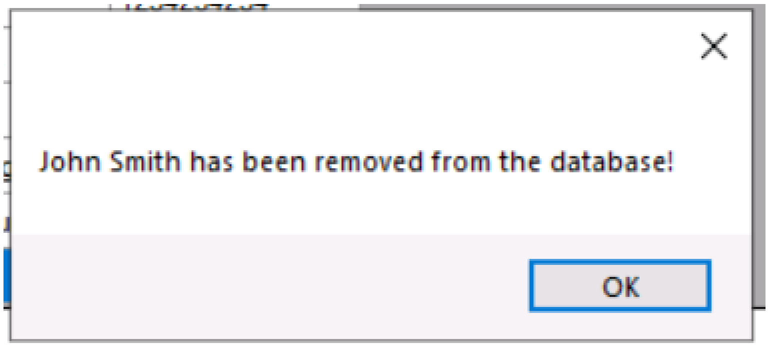
By clicking the Delete Customer button on the Customer Records menu, the user will be able to delete a customer from the database. Unlike the View Customer Details and Edit Customer buttons, the user will not be navigated to another screen. The user will receive the following prompt:


Warning

Are you sure you want to delete John Smith? This will also delete all appointments regarding this user.

If the user selects No, the delete action will be canceled and the prompt will close. If the user selects Yes, the database will attempt to delete the customer from the database. If the deletion is unsuccessful, an error message will appear. If the deletion is successful, all appointments involving the customer will be deleted from the database. Then, the customer will be deleted. The user will receive the following messages:


All appointments involving John Smith have been removed from the database!

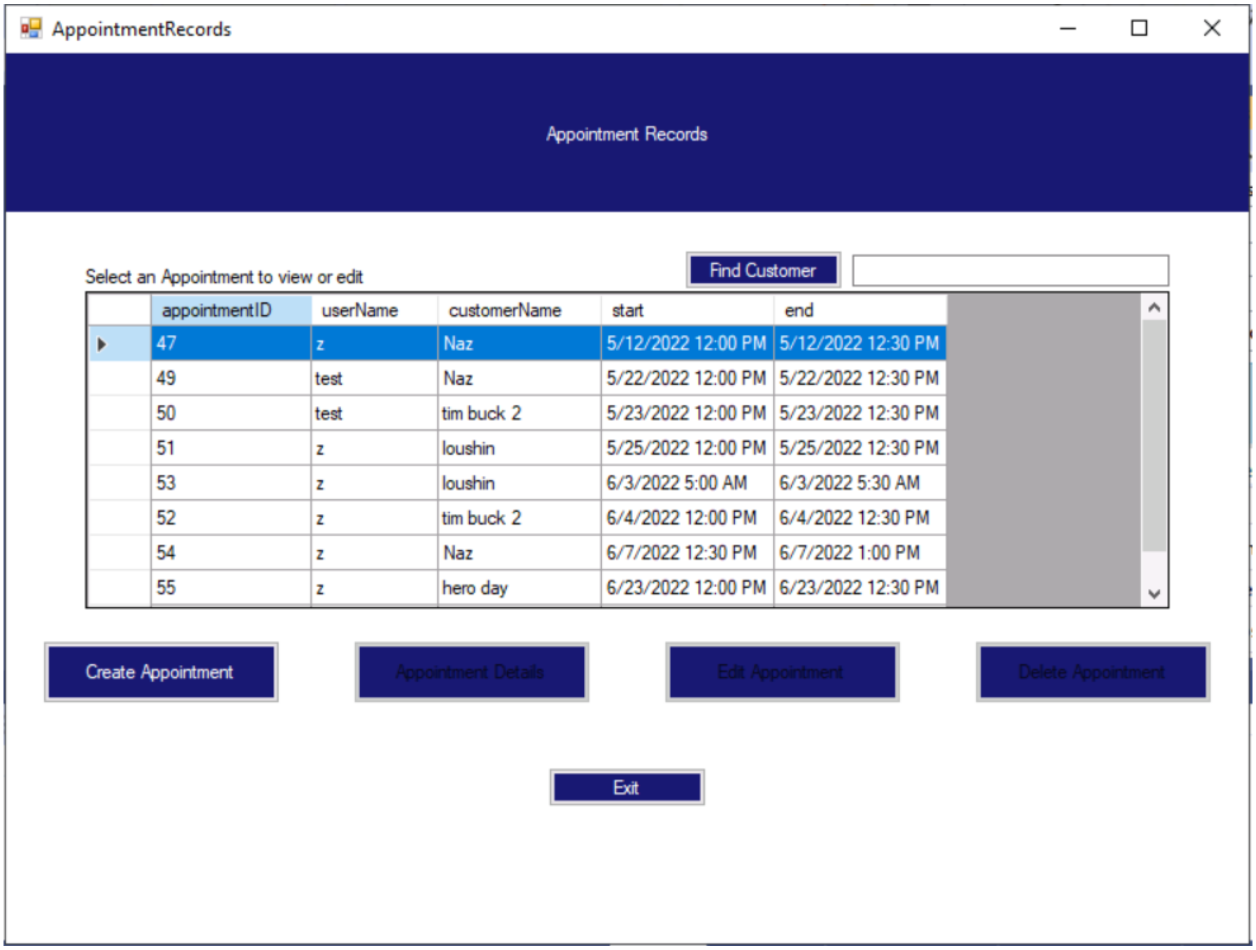

John Smith has been removed from the database!

Available actions:

1. Click the Yes button to confirm the deletion of the customer.
2. Click the No button to cancel the deletion.

Appointment Records

This menu is where the user can perform actions relating to creating, editing and deleting appointments set between a customer and a user. The user can also view further information in the Appointment Details section. The dataGridView displays all Appointments currently in the database. The user must select an appointment before the buttons to View, Edit, or Delete the appointment become available. The user can also filter the dataGridView with the Find Customer search function located in the top right corner. Please see the Customer Records section of this User Guide for an example of how the Find Customer search function works..



appointmentID	userName	customerName	start	end
47	z	Naz	5/12/2022 12:00 PM	5/12/2022 12:30 PM
49	test	Naz	5/22/2022 12:00 PM	5/22/2022 12:30 PM
50	test	tim buck 2	5/23/2022 12:00 PM	5/23/2022 12:30 PM
51	z	loushin	5/25/2022 12:00 PM	5/25/2022 12:30 PM
53	z	loushin	6/3/2022 5:00 AM	6/3/2022 5:30 AM
52	z	tim buck 2	6/4/2022 12:00 PM	6/4/2022 12:30 PM
54	z	Naz	6/7/2022 12:30 PM	6/7/2022 1:00 PM
55	z	hero day	6/23/2022 12:00 PM	6/23/2022 12:30 PM

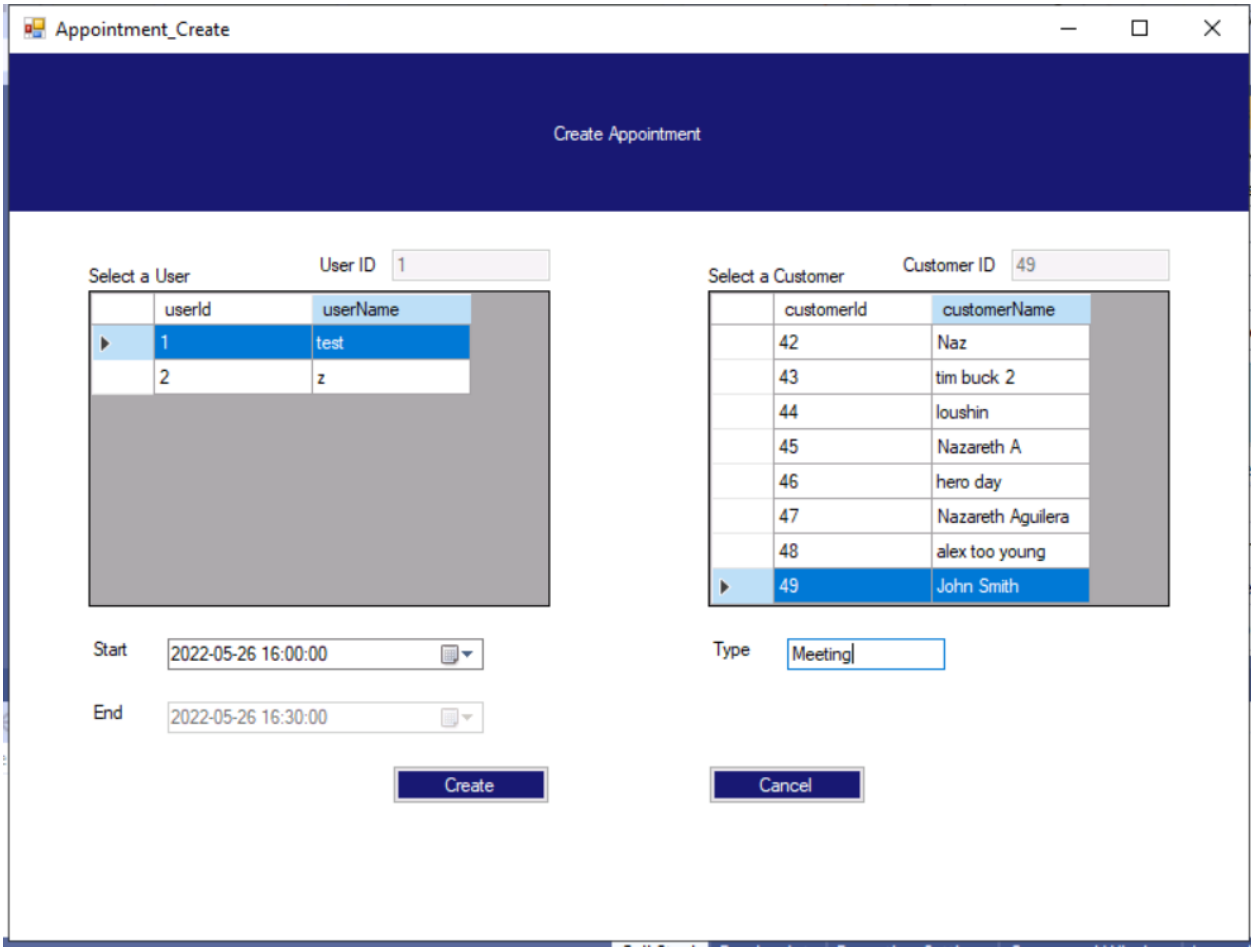
Available actions:

1. Click on a cell in the dataGridView to enable the Appointment Details, Edit Appointment and Delete Appointment buttons.
2. Click Create Appointment to be prompted a form to create a new appointment.
3. Click Appointment Details to be prompted a form that will display appointment information.
4. Click Edit Appointment to be prompted a form that will allow the user to change information about the selected appointment.

5. Type in the Find Customer Textbox to filter the dataGridView listings by the customerName.
6. Click the Exit button or the X located in the upper right corner to return to the main menu.

Appointment Records - Create a new Appointment

In this section, the user can set up a new appointment between a user and a customer. Please note that a user can set up a new appointment for a customer with any other customer, not just the logged in user. The user will select an appointment start time. The end time is set to automatically end 30 minutes after. The user can describe the nature of the appointment by leaving a short description in the Type input field. Once all the inputs have been validated, the create button will be enabled.



Select a User

User ID: 1

	userId	userName
▶	1	test
	2	z

Select a Customer

Customer ID: 49

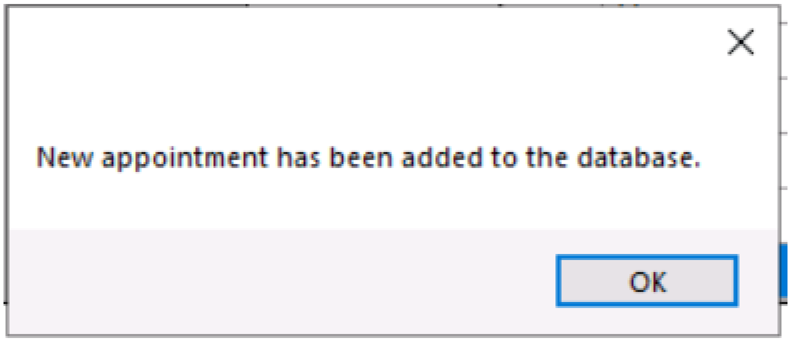
	customerId	customerName
	42	Naz
	43	tim buck 2
	44	loushin
	45	Nazareth A
	46	hero day
	47	Nazareth Aguilera
	48	alex too young
▶	49	John Smith

Start: 2022-05-26 16:00:00

End: 2022-05-26 16:30:00

Type: Meeting

If the creation of a new appointment is unsuccessful, an error message will appear. If the creation of a new appointment is successful, the user will receive the following message:



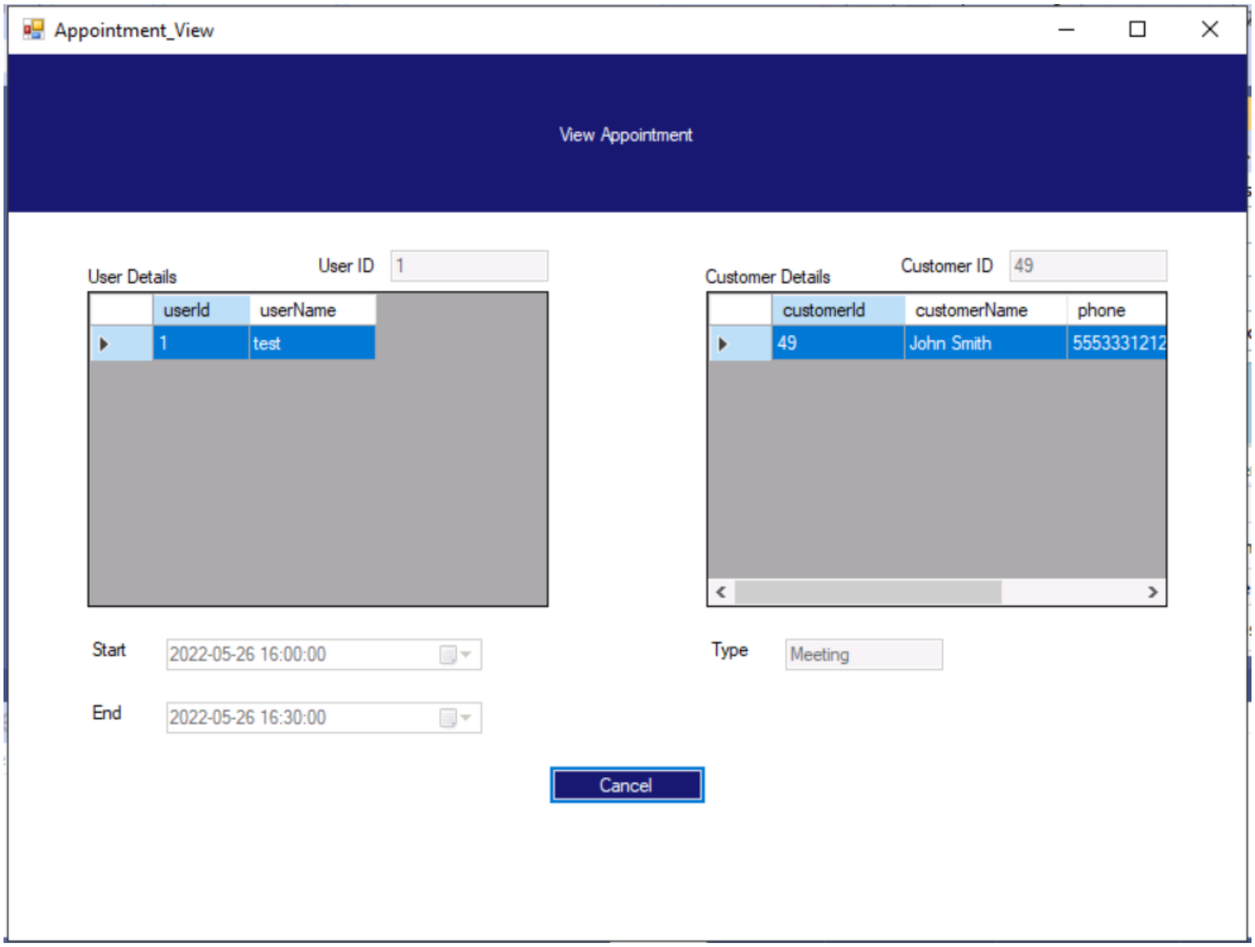
New appointment has been added to the database.

Available actions:

1. Select a user to be added to the appointment by clicking on a cell in the dataGridView (left).
2. Select a customer to be added to the appointment by clicking on a cell in the dataGridView (right).
3. Select a start time for the appointment.
4. Enter a short description describing the purpose of the meeting in the Type input field.
5. Click the Create button to add the new appointment to the database.
6. Click the Cancel button or the X located in the upper right corner to return to the Appointment Records menu.

Appointment Records - View Appointment Details

In this section, the user can view further information about the appointment that was selected on the Appointment Records menu.



User Details

User ID: 1

userId	userName
1	test

Customer Details

Customer ID: 49

customerId	customerName	phone
49	John Smith	5553331212

Start: 2022-05-26 16:00:00

End: 2022-05-26 16:30:00

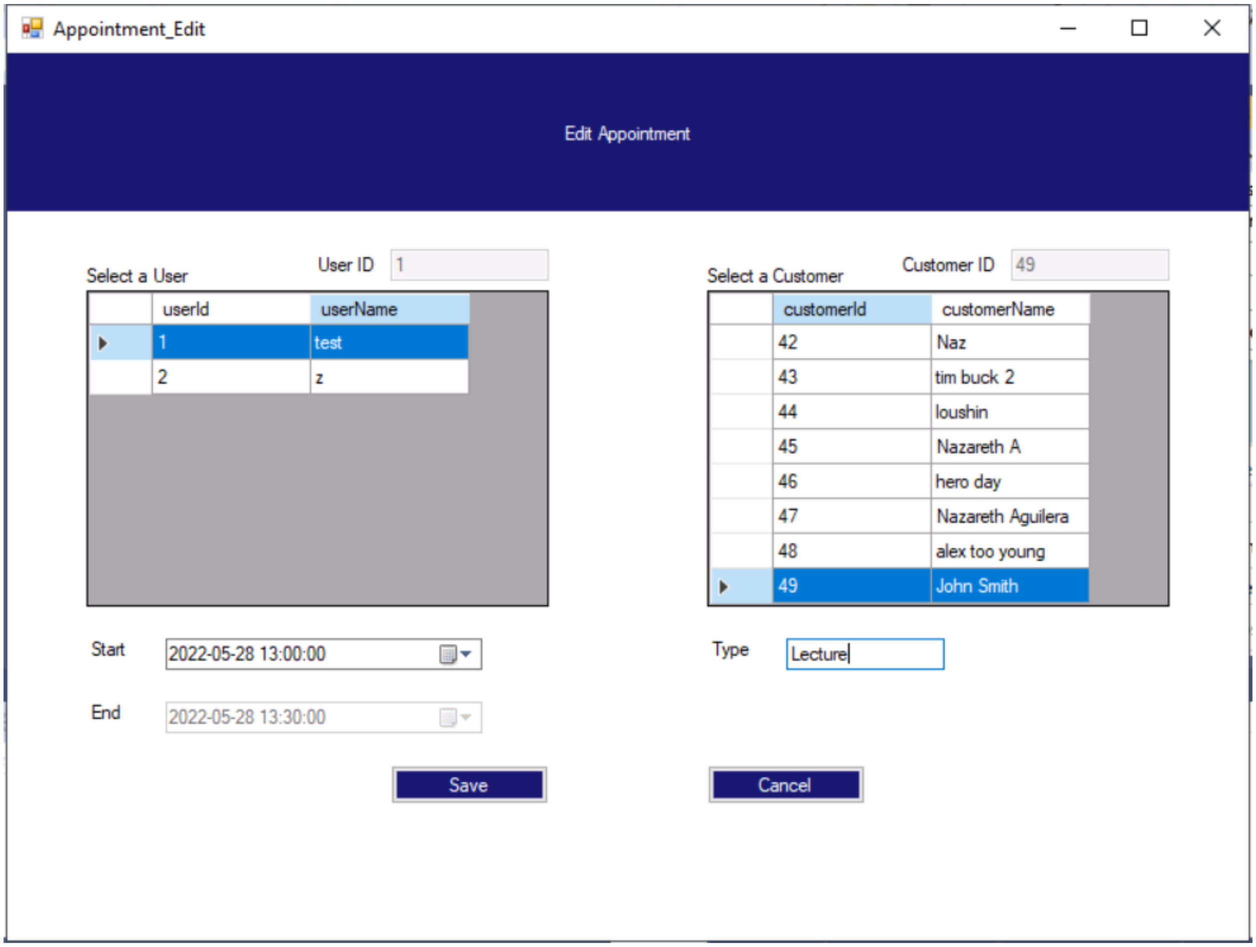
Type: Meeting

Available actions:

1. Click the Cancel button or the X located in the upper right corner to return to the Appointment Records menu.

Appointment Records - Edit Appointment

In this section, the user can edit the information regarding the appointment that was selected on the Appointment Records menu. The user can change the user selected for the appointment, change the customer for the appointment, change the appointment date and time, and also change the description for the appointment. If any of the changes become invalid, the Save button will become disabled. If the changes are all valid, the Save button will be enabled allowing the user to update the appointment in the database.



Select a User

User ID: 1

	userId	userName
▸	1	test
	2	z

Select a Customer

Customer ID: 49

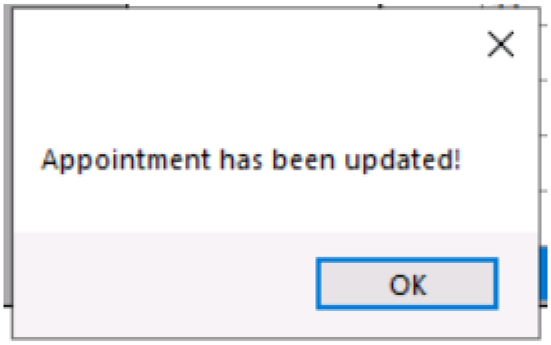
	customerId	customerName
	42	Naz
	43	tim buck 2
	44	loushin
	45	Nazareth A
	46	hero day
	47	Nazareth Aguilera
	48	alex too young
▸	49	John Smith

Start: 2022-05-28 13:00:00

End: 2022-05-28 13:30:00

Type: Lecture

If the update is unsuccessful, an error message will appear. If the update is successful, the user will receive the following message:



Appointment has been updated!

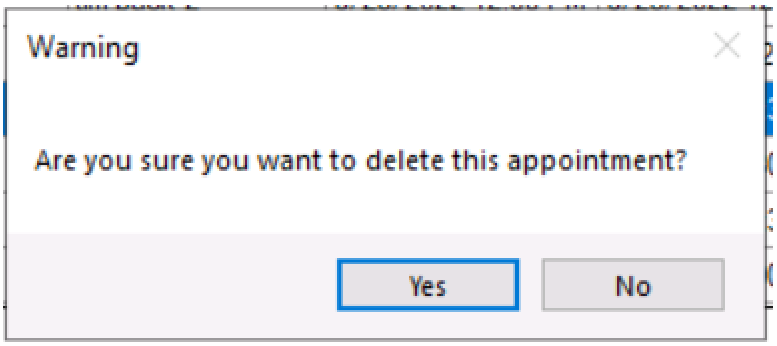
Available actions:

1. Select a user to be updated to the appointment by clicking on a cell in the dataGridView (left).
2. Select a customer to be updated to the appointment by clicking on a cell in the dataGridView (right).
3. Update the start time for the appointment.
4. Update the short description describing the purpose of the meeting in the Type input field.
5. Click the Save button to update the appointment in the database.

1. Click the Cancel button or the X located in the upper right corner to return to the Appointment Records menu.

Appointment Records - Delete Appointment

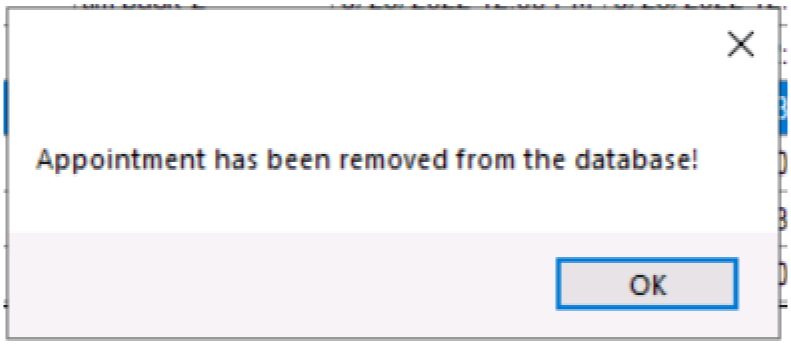
By clicking the Delete Appointment button on the Appointment Records menu, the user will be able to delete an appointment from the database. Unlike the Appointment Details and Edit Appointment buttons, the user will not be navigated to another screen. The user will receive the following prompt:



Warning

Are you sure you want to delete this appointment?

If the user selects No, the delete action will be canceled and the prompt will close. If the user selects Yes, the database will attempt to delete the appointment from the database. If the deletion is unsuccessful, an error message will appear. If the deletion is successful, all appointments involving the appointment will be deleted. The user will receive the following messages:



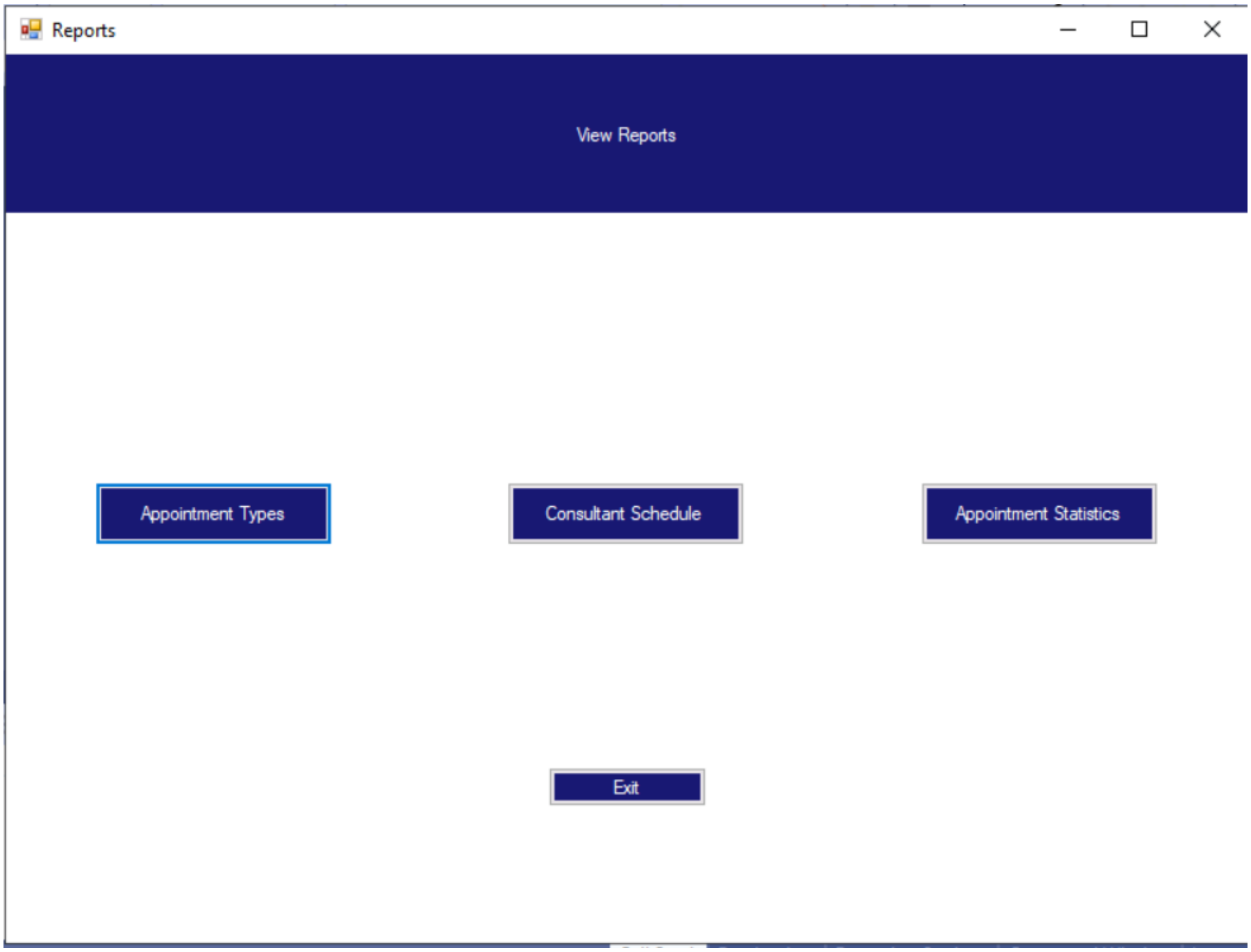
Appointment has been removed from the database!

Available actions:

1. Click the Yes button to confirm the deletion of the appointment.
2. Click the No button to cancel the deletion.

Reports

The Reports menu allows the user to view different options for reports generated by the application. The user can choose between viewing the Appointment Types report, viewing the Consultant Schedule report, or viewing the Appointment Statistics report.

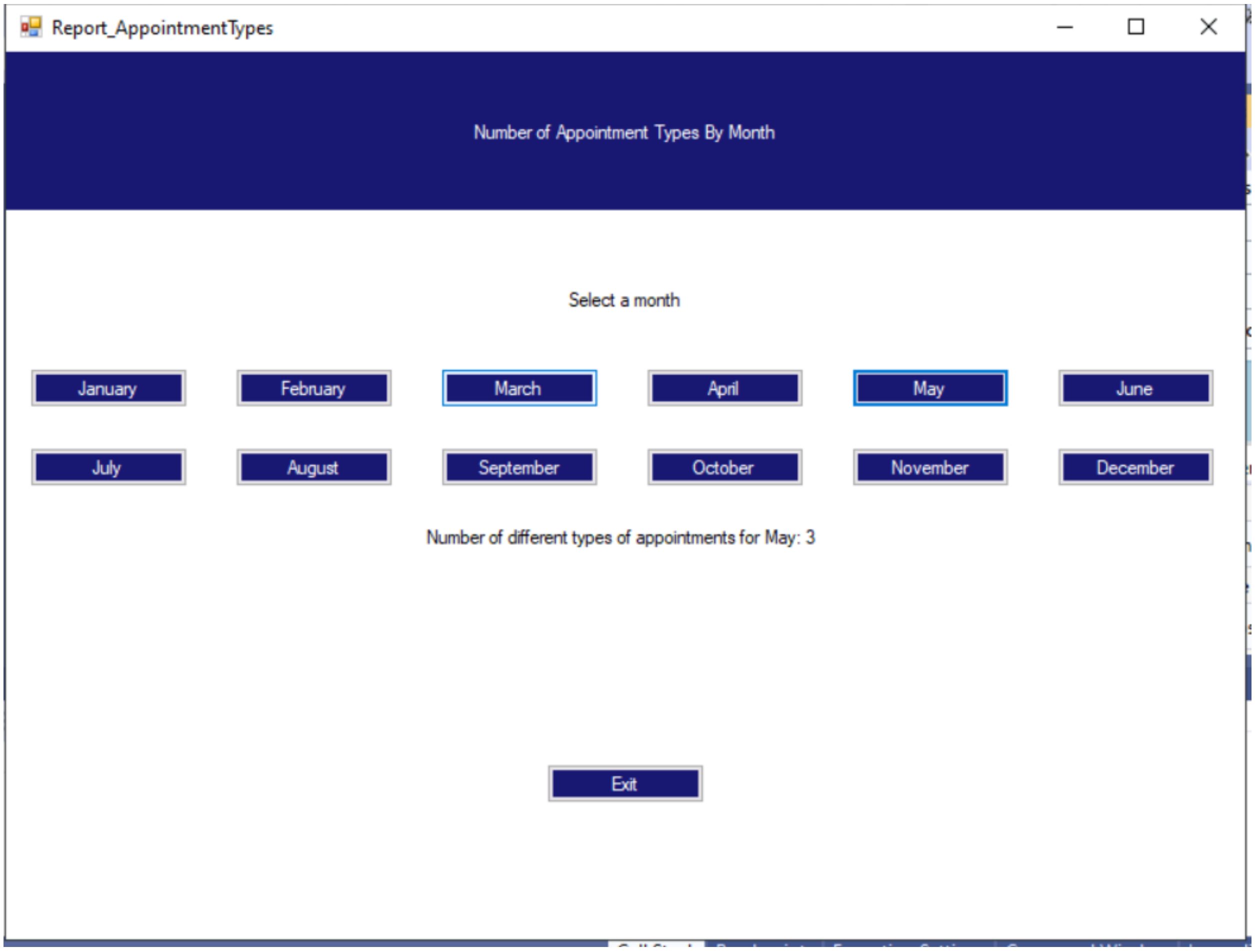


Available actions:

1. Click the Appointment Types button to view a report showing the number of different types of appointments that occur each month.
2. Click the Consultant Schedule button to view a report showing the schedule of a consultant and the total number of appointments that consultant has been scheduled for.
3. Click the Appointment Statistics button to view a report of the days of the week that a user is most scheduled for.
4. Click the Exit button or the X located in the upper right corner to return to the main menu.

Reports - Appointment Types by Month

In this section, the user can view a report which shows the number of different types of appointments that have been set in a particular month. The user is presented with twelve buttons which represent the twelve months of the year. When the user clicks on one of these buttons, a line of text will appear that calculates the total number of different appointments and displays that number.



Report_AppointmentTypes

Number of Appointment Types By Month

Select a month

Number of different types of appointments for May: 3

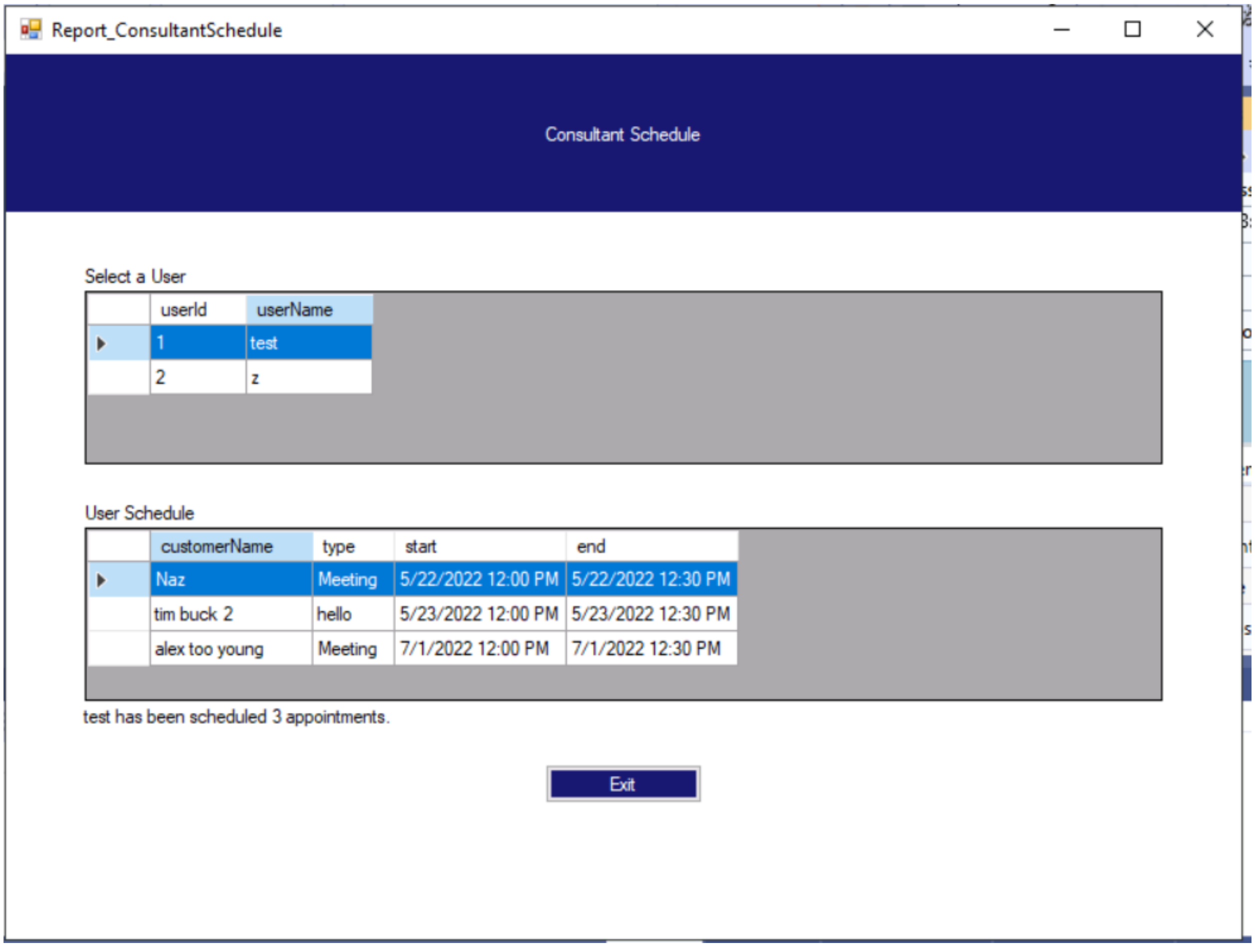
Available actions:

1. Click on the January button to view the number of different types of appointments set for the month of January.
2. Click on the February button to view the number of different types of appointments set for the month of February.
3. Click on the March button to view the number of different types of appointments set for the month of March.
4. Click on the April button to view the number of different types of appointments set for the month of April.

5. Click on the May button to view the number of different types of appointments set for the month of May.
6. Click on the June button to view the number of different types of appointments set for the month of June.
7. Click on the July button to view the number of different types of appointments set for the month of July.
8. Click on the August button to view the number of different types of appointments set for the month of August.
9. Click on the September button to view the number of different types of appointments set for the month of September.
10. Click on the October button to view the number of different types of appointments set for the month of October.
11. Click on the November button to view the number of different types of appointments set for the month of November.
12. Click on the December button to view the number of different types of appointments set for the month of December.
13. Click the Exit button or the X located in the upper right corner to return to the Reports menu.

Reports - Consultant Schedule

In this section, the user can view a report that shows the full appointment schedule for a user throughout the entire lifetime of the user. The user will begin by selecting a user in the top dataGridView. Once selected, the bottom dataGridView will be populated with that user's appointment schedule. A line of text will appear at the bottom showing the total number of appointments that the user has been scheduled for.



Select a User

	userId	userName
▶	1	test
	2	z

User Schedule

	customerName	type	start	end
▶	Naz	Meeting	5/22/2022 12:00 PM	5/22/2022 12:30 PM
	tim buck 2	hello	5/23/2022 12:00 PM	5/23/2022 12:30 PM
	alex too young	Meeting	7/1/2022 12:00 PM	7/1/2022 12:30 PM

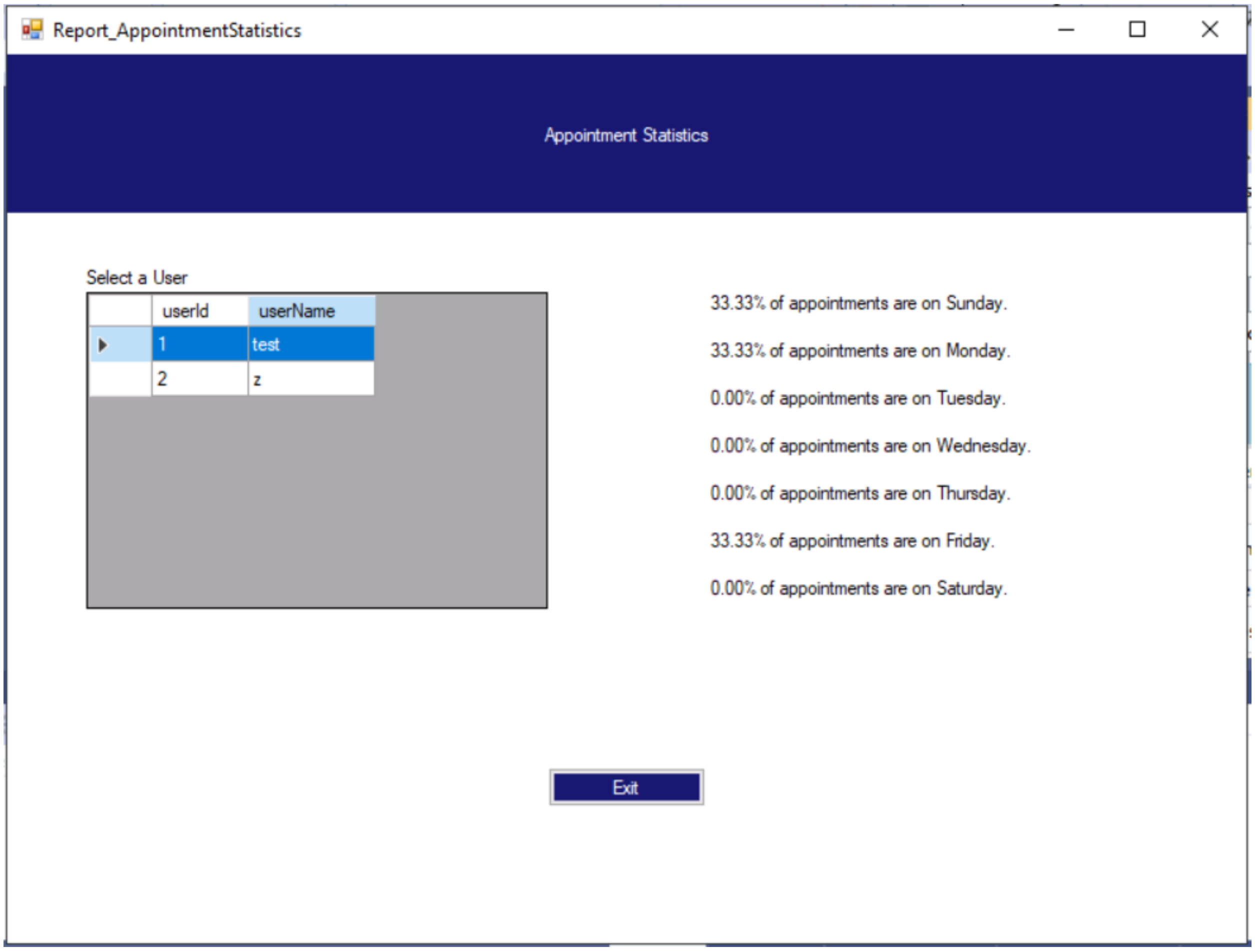
test has been scheduled 3 appointments.

Available actions:

1. Click on a cell in the top dataGridView to show that user's complete schedule in the bottom dataGridView.
2. Click the Exit button or the X located in the upper right corner to return to the Reports menu.

Reports - Appointment Statistics

In this section, the user can view a report that shows the distribution of appointments set throughout the week for all appointments that the user is set for. By clicking on a user in the dataGridView on the right, the report will calculate what percentage of their appointments occur on a given day of the week.



	userId	userName
▸	1	test
	2	z

33.33% of appointments are on Sunday.

33.33% of appointments are on Monday.

0.00% of appointments are on Tuesday.

0.00% of appointments are on Wednesday.

0.00% of appointments are on Thursday.

33.33% of appointments are on Friday.

0.00% of appointments are on Saturday.

Available Actions:

1. Click on a cell in the dataGridView on the right to generate a statistical report on which days of the week the user is set up for their appointments.
2. Click the Exit button or the X located in the upper right corner to return to the Reports menu.