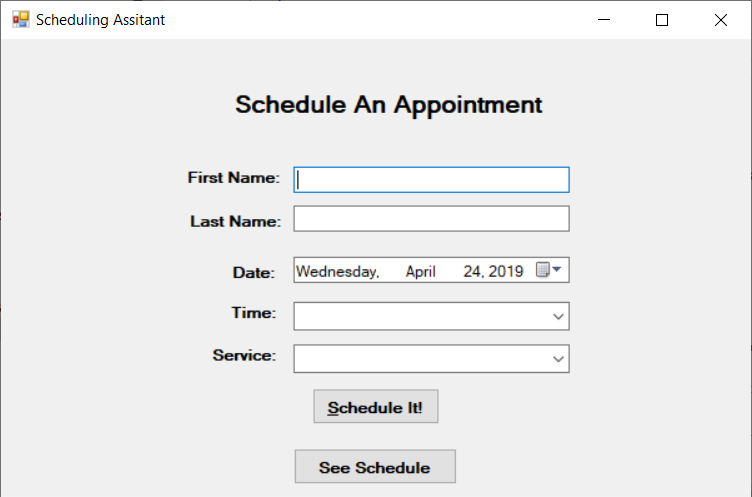
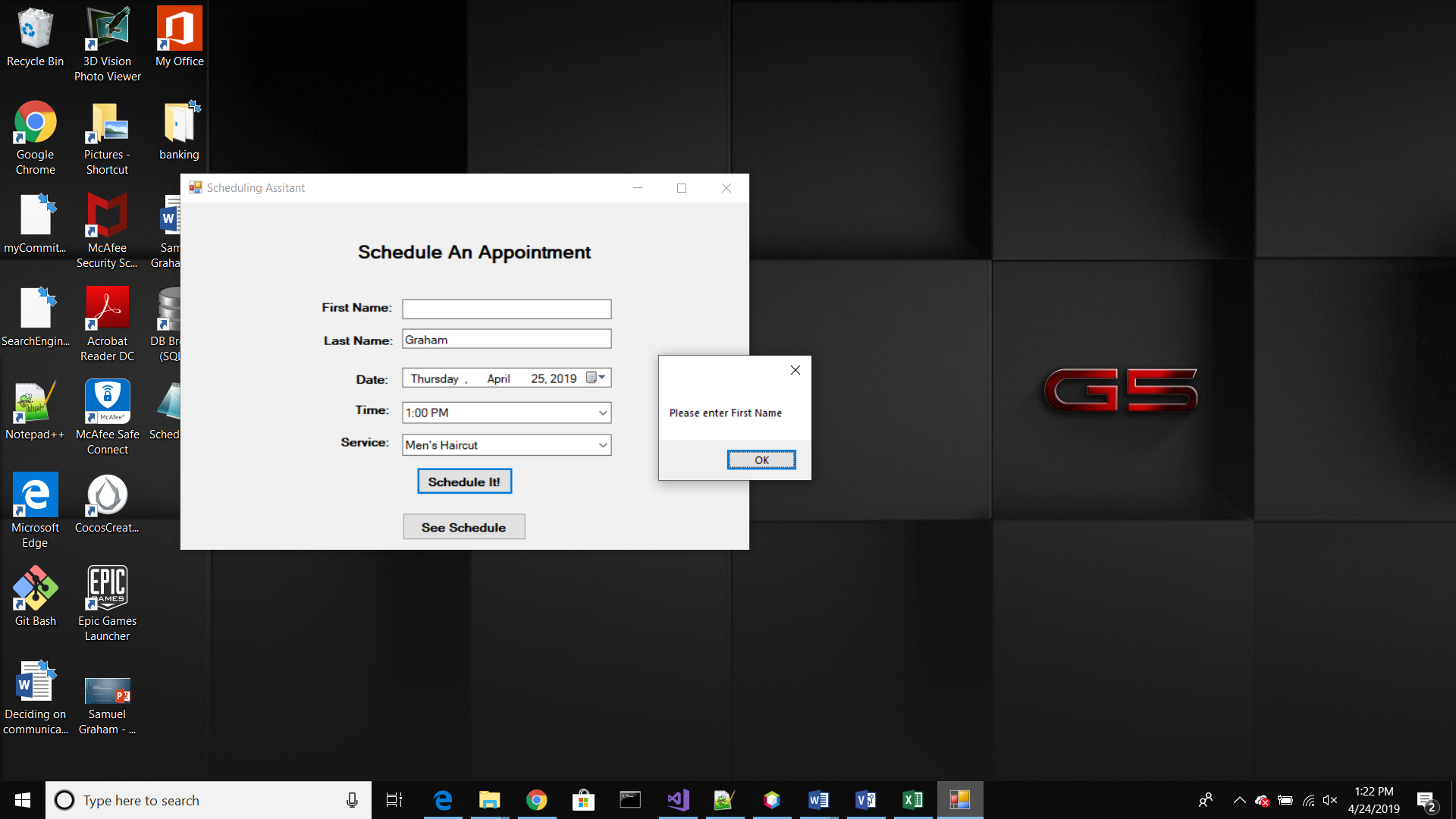
Please let me know if you would like me to zip the project and send to you. (note: the database is embedded but currently using an absolute path)

Code for this project is on pages after screenshots.

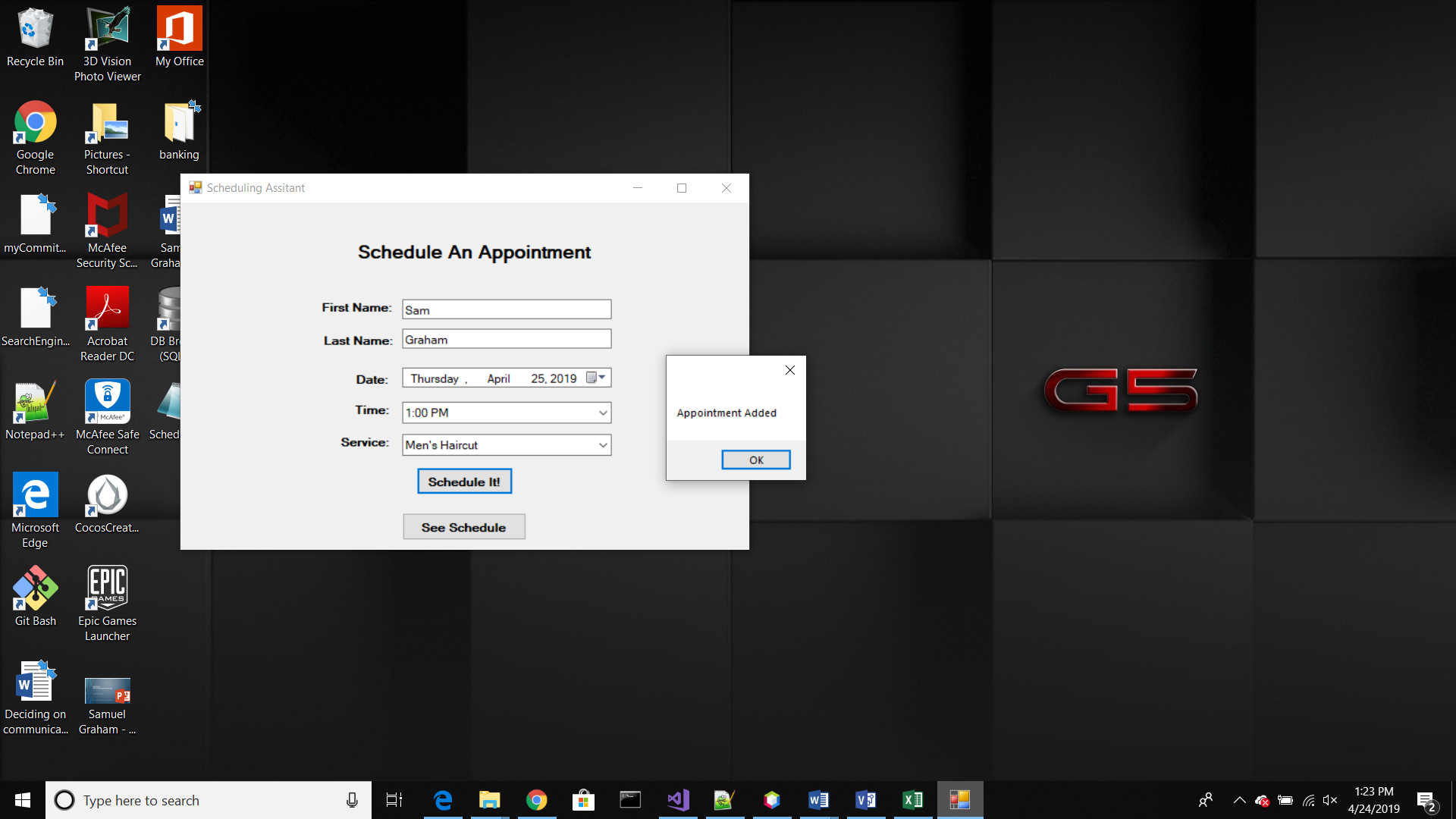
**GUI Screen:**



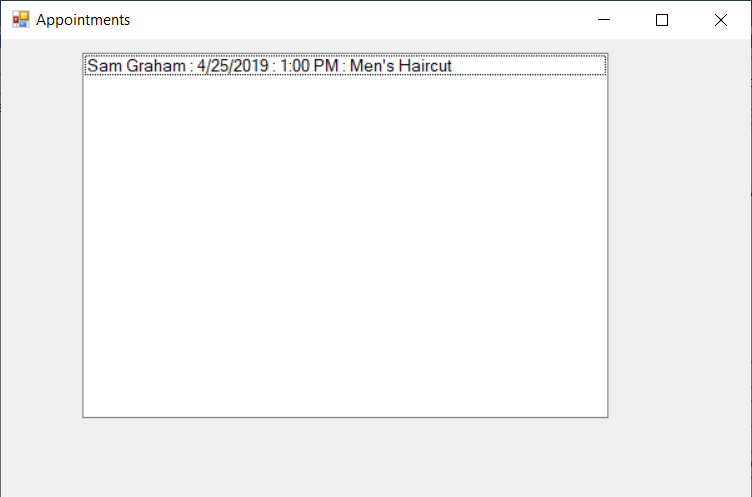
**Blank input not allowed:**



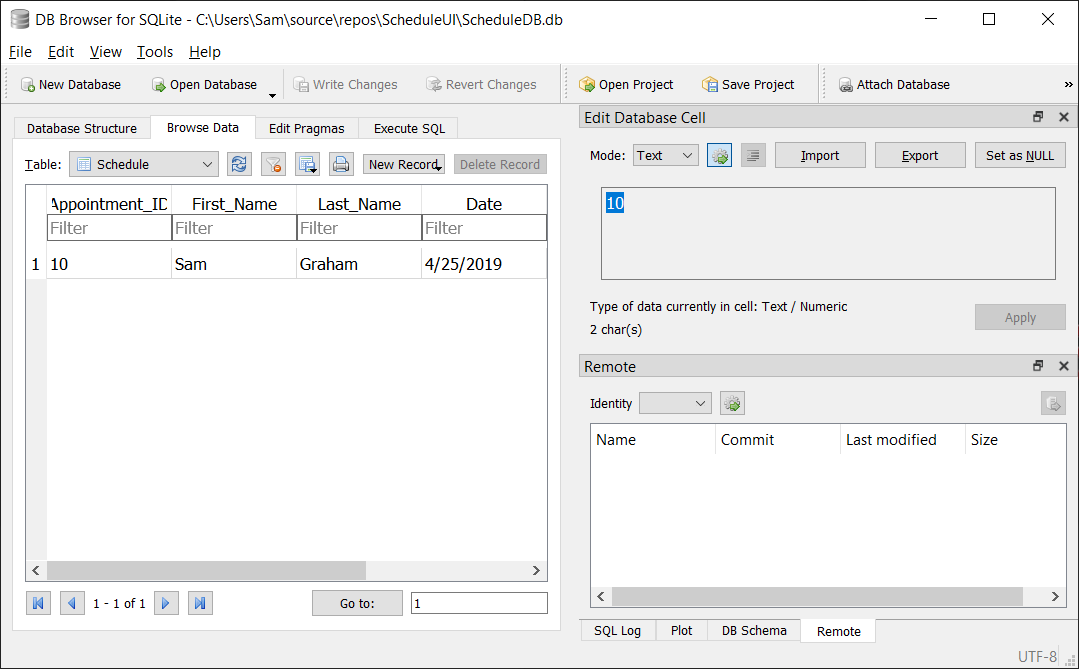
**User informed of schedule success:**



**See Schedule Button:**



**Database Data in DB Browser:**



// **GUI FORM CODE:**

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace SchedulePersist

{

public partial class UIform : Form

{

public UIform()

{

InitializeComponent();

}

/\*\*

\* Adds appointment using information from the GUI.

\*/

private void AddAppointment()

{

AppointmentModel a = new AppointmentModel

{

First\_Name = FirstNametxt.Text,

Last\_Name = LastNametxt.Text,

Date = datePicker.Value.Date.ToString("d"),

Time = ComboTime.GetItemText(ComboTime.SelectedItem),

Service = ComboService.GetItemText(ComboService.SelectedItem)

};

SqliteDataAccess.SaveAppointment(a);

MessageBox.Show("Appointment Added");

//clear the form for next input

FirstNametxt.Text = "";

LastNametxt.Text = "";

datePicker.Text = "";

ComboTime.SelectedIndex = -1;

ComboService.SelectedIndex = -1;

}

// Handles Schedule It! click. Validates fields are not empty then

// adds appointment to database.

private void Schedule\_Button\_Click(object sender, EventArgs e)

{

if (validateInput() == true)

AddAppointment();

}

// Handles See Schedule Button. Opens new form displaying

// scheduled appointments.

private void BtnSeeSchedule\_Click(object sender, EventArgs e)

{

ScheduleUI.AppointmentForm form = new ScheduleUI.AppointmentForm();

form.Show();

}

/\*\*

\* Verifies the user filled in the form correctly. Will work from the

\* bottom of the form up. Reverse order of if statements to work from top

\* of form down.

\*/

private bool validateInput()

{

bool mayProceed = true;

int problem = 0;

if (FirstNametxt.Text == "")

problem = 1;

if (LastNametxt.Text == "")

problem = 2;

if (datePicker.Text == "")

problem = 3;

if (ComboTime.SelectedIndex == -1)

problem = 4;

if (ComboService.SelectedIndex == -1)

problem = 5;

switch (problem)

{

case 1:

MessageBox.Show("Please enter First Name");

FirstNametxt.Focus();

mayProceed = false;

break;

case 2:

MessageBox.Show("Please enter Last Name");

LastNametxt.Focus();

mayProceed = false;

break;

case 3:

MessageBox.Show("Please pick a date");

datePicker.Focus();

mayProceed = false;

break;

case 4:

MessageBox.Show("Please choose a time");

ComboTime.Focus();

mayProceed = false;

break;

case 5:

MessageBox.Show("Please choose a service");

ComboService.Focus();

mayProceed = false;

break;

}

return mayProceed;

}

/\*\*

\* Remove times that are already scheduled from the drop down list when

\* user selects a date. Needs some work to be implemented correctly.

\*/

private void datePicker\_ValueChanged(object sender, EventArgs e)

{

string date = datePicker.Value.Date.ToString("d");

List<AppointmentModel> notAvailable = SqliteDataAccess.takenAppointments(date);

foreach (AppointmentModel unavailableTime in notAvailable)

foreach (string time in ComboTime.Items)

if (ComboTime.GetItemText(ComboTime.SelectedItem) == unavailableTime.ToString())

{

ComboTime.Items.Remove(ComboTime.SelectedIndex);

}

ComboTime.Update();

}

}

}

// **Code to interact with database**

using Dapper;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SQLite;

using System.Linq;

using System.Configuration;

namespace SchedulePersist

{

/\*\*

\* Provides methods to interact with the ScheduleDB database. See App.config file

\* for path settings of database.

\*/

public class SqliteDataAccess

{

/\*\*

\* Provides a list of Appointments.

\*/

public static List<AppointmentModel> LoadAppointment()

{

using (IDbConnection cnn = new SQLiteConnection(LoadConnectionString()))

{

var output = cnn.Query<AppointmentModel>("select \* from Schedule", new DynamicParameters());

return output.ToList();

}

}

/\*\*

\* Saves an Appointment to the database.

\*/

public static void SaveAppointment(AppointmentModel appointment)

{

using (IDbConnection cnn = new SQLiteConnection(LoadConnectionString()))

{

cnn.Execute("insert into Schedule (First\_Name, Last\_Name, Date, Time, Service) values " +

"(@First\_Name, @Last\_Name, @Date, @Time, @Service)", appointment);

}

}

/\*\*

\* Queries database for available appointments given the date as argument

\*/

public static List<AppointmentModel> takenAppointments(string date)

{

using (IDbConnection cnn = new SQLiteConnection(LoadConnectionString()))

{

var takenAppointments = cnn.Query<AppointmentModel>($"select Time from Schedule where Time = {date}", new DynamicParameters());

return takenAppointments.ToList();

}

}

/\*\*

\* Connect to the database

\*/

private static string LoadConnectionString(String id = "Default")

{

return ConfigurationManager.ConnectionStrings[id].ConnectionString;

}

}

}

// **Code for AppointmentForm**

using SchedulePersist;

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace ScheduleUI

{

public partial class AppointmentForm : Form

{

public AppointmentForm()

{

InitializeComponent();

}

/\*\*

\* Executes when form loads. Populates the list box with scheduled appointments

\*/

private void appointmentForm\_Load(object sender, EventArgs e)

{

List<AppointmentModel> appointments = SqliteDataAccess.LoadAppointment();

List<string> appointmentStringList = new List<string>();

foreach (AppointmentModel scheduledAppointment in appointments)

{

appointmentStringList.Add(scheduledAppointment.AppointmentDetails());

}

ListBxAppointments.Items.Clear();

foreach (string appointmentString in appointmentStringList)

{

ListBxAppointments.Items.Add(appointmentString);

}

}

}

}

// **Code for AppointmentModel**

namespace SchedulePersist

{

/\*\*

\* Creates an appointment object.

\*/

public class AppointmentModel

{

public int appointmentId { get; set; }

public string First\_Name { get; set; }

public string Last\_Name { get; set; }

public string Date { get; set; }

public string Time { get; set; }

public string Service { get; set; }

/\*\*

\* Display appointment object as string

\*/

public string AppointmentDetails()

{

string appointmentString = First\_Name + " " + Last\_Name + " : " +

Date + " : " + Time + " : " + Service;

return appointmentString;

}

}

}

**// App.config file (ablsolute path for DB connection) it is embedded but had**

**// trouble with relative path**

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<connectionStrings>

<add name="Default" connectionString="Data Source=C:\Users\Sam\source\repos\ScheduleUI\ScheduleDB.db;Version=3;" providerName="System.Data.SqlClient"/>

</connectionStrings>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6.1" />

</startup>

</configuration>