PAS Report

Program functionality

The program is used to display/edit information about patients in different locations in a hospital, and brief information about doctors.

This is the dashboard view of the program. From here, a user has the function to check for a specific subset of persons inside the database (Patients in Ward, ICU and doctors.) The user can also check the search button where all people in the database are displayed. The user also has the option to add either a patient or a doctor to the database. Numbers keep track of how many users are in each subset of the database.

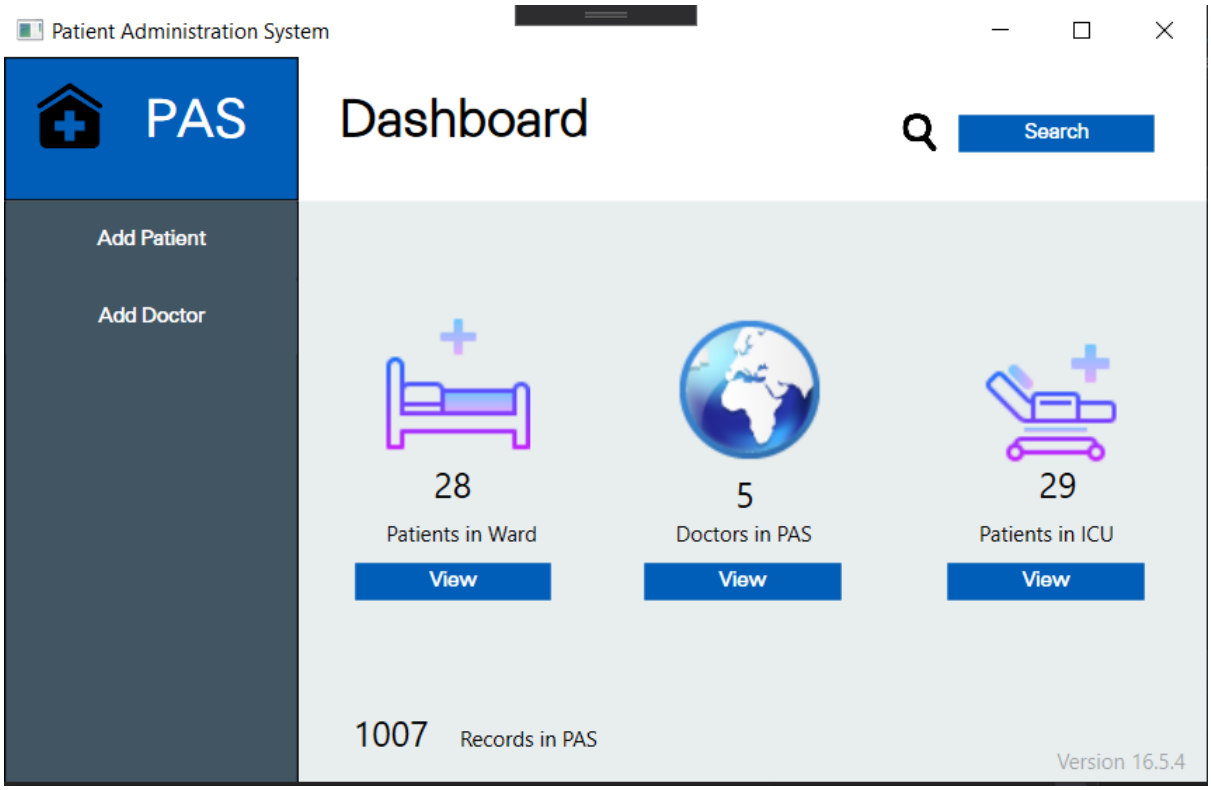

Figure 1-PAS Dashboard

When a user clicks on a button to view the users in the database, they have displayed them within a data table, and are given the option to filter the search based on their needs. If the user wants to select a specific user, they can double click on the data table field.

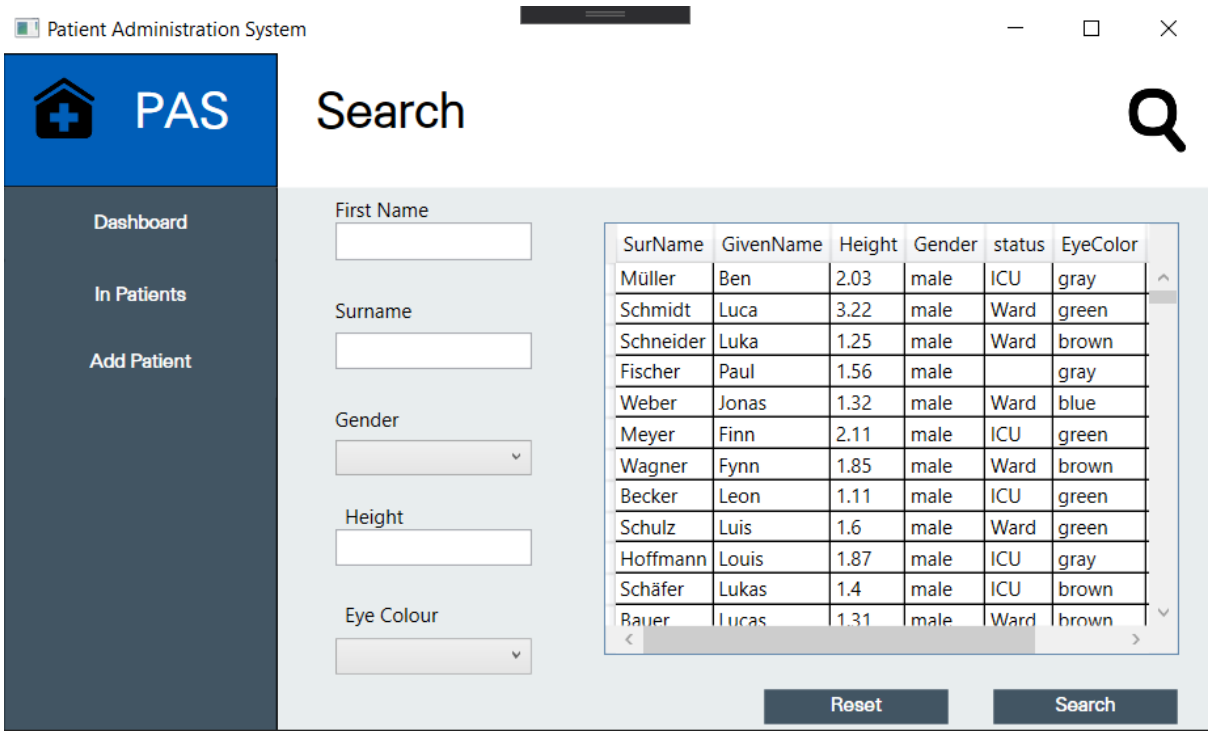

Figure 2- PAS Basic Search

A user then has the option to edit the information about the specific patient as shown. The user can also 'promote' the patient to a doctor position with the button as displayed. The patient can also be deleted.

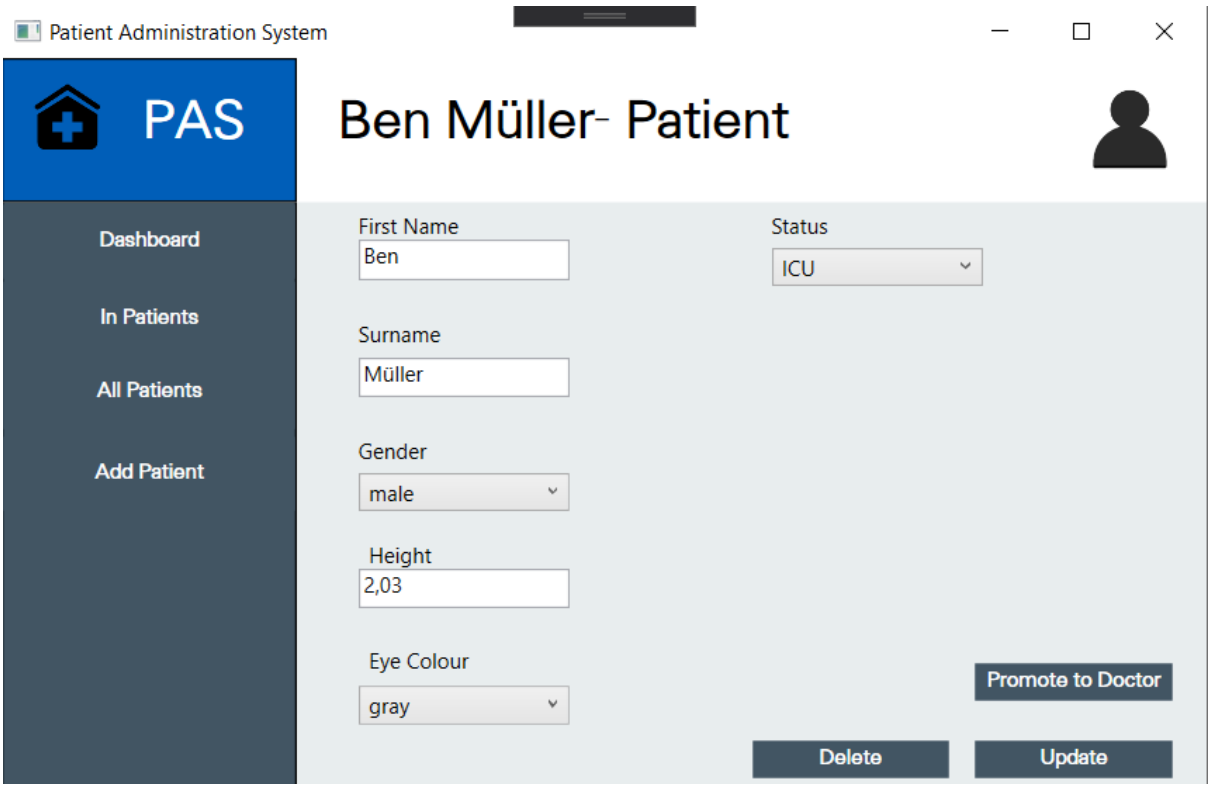

Figure 3 - The user info of a patient.

Classes and Interaction

Inheritance

I first made an abstract class of Person, and I also created a doctor.

```
public class Person
{
    2 references
    public string SurName { get;  set; }
    2 references
    public string GivenName { get;  set; }
    2 references
    public decimal Height { get;  set; }
    2 references
    public string Gender { get;  set; }
    2 references
    public string EyeColor { get;  set; }
```

Figure 4 - The base class with all fields get and set

My base class in this situation is the person. They are the parent, and the class that is being inherited from. This person class has access modifiers shown at 'public'

The derived class in this situation is the doctor. They are the child, and the class that inherits from another class. The doctor will inherit all the same fields that the person has such as surname, given name and eyecolor. They will also have their own individual fields unique to them and in this case it is work.

```
class Doctor : Person
{

    0 references
    public string work { get; set; }
}
```

Figure 5 – ':' specifies which class is inheriting which

Class and Object

```
//assinging to class
Person p = new Person();
p.SurName = SurnameInput.Text;
p.GivenName = FNameInput.Text;
```

Figure 6 - assigning person attributes to the class

Above you can see p, which is an instance type of class Person. A new method is assigned to "p" by calling a special method of the class called constructor. The class of person interacts with my SQliteDataAccess class.

This class makes the connection to my database, and the person class is used to add or update user info. The doctor class is used when a doctor is added.

Windows

In my program, there are 4 XMAL windows are they are as follows:

- MainWindow.xmal
    - This window is my dashboard. This is what is run when the user first starts the program.
- Search.xmal
    - This window is used when the search button is called. From the dashboard, the search can be called in 4 different variations, with an ordinary search or searching specifically for something in the database. Figure 7 shows how the ordinary search is opened, by passing a blank query.

```
String query = null;
Search se = new Search(query);
se.Show();
this.Close();
```

Figure 7 - passing a query as a parameter

- AddPatient.xmal
    - This window is used when adding a patient to the database. There are slight modifications to it also for when adding a doctor.
- UserInfo.xmal
    - This window is displayed when a user clicks on a row in the DataTable on the search.xmal. This window will display all of the user information that can be edited and deleted from the database. The user also has options for promoting a patient to a doctor, but this button is not visible for doctors.

Lecture topics covered

Polymorphy

Polymorph was used to create an abstract base class for each individual person. As stated above, this will occur when we have classes that are related to each other by inheritance and in the case of my program this is the doctor to the person class. The doctor inherits all of the fields from the person.

WPF-Application

The WPF-Application is used as my base, in order to display the user interface. It is what helps bridges the divide between the user and functionality.

User-Controls

User controls are added for the user to interact with the WPF-Application and make requests. As shown by my dashboard page, the user has controls to view specific patients/doctors in the database, add patients/doctors or to search.

Events

Events are used throughout my application. An example is on a button click; the event handler is launched to then make an action. Figure 8 shows what when the button is clicked on. The RoutedEventArgs contains state information and event data associated with a routed event

```
1 reference
private void AddPatient_Button(object sender, RoutedEventArgs e)
{
```

Figure 8 - The add patient event

There are many event handlers available for elements in the WPF forms, for example different types of movement of the mouse, dragging of the element and previews. Another example is on my search.xaml with my DataGrid. The data grid has an event for a mouse double click.

```
1 reference
private void SearchResults_MouseDoubleClick(object sender, MouseButtonEventArgs e)
{
    //determining the selected result, and passing the id into the user info.
    DataRowView select = SearchResults.SelectedItem as DataRowView;
```

Figure 9 - Double click event

Figure 9 shows this event handler, with MouseButtonEventArgs providing data for the mouse button related events.

Opening and closing

Figure 10 shows the opening and closing of a window, and this happens every time a Button.Click event is used. The example shows the addpatient.xmal window being instantiated with a bool of false being passed, and the property show being set. 'this' instance of the current window is closed with the .close property.

```
AddPatient AddPat = new AddPatient(false);
AddPat.Show();
this.Close();
```

Figure 10 - Opening and closing windows

Interfaces

Given more time to think of methods or taking a different approach to using my classes I would have implemented an interface.

LINQ-Queries

Given my program makes use of SQLite, LINQ-Queries were a very similar implementation but never got around to being done. They would have been used to display statistics of the database such as ‘number of doctors with brown eyes’.

These further additions were discovered too late too implement, given my process of working the application. It felt more fitting to provide a working application then something not fully functioning, as I try accommodate the changes these would both make to my program in the space of time I had.

New topics covered

SQLlite - Connection

With my implementation of a database, I had to learn the corresponding SQLite commands to link my project to the database.

The first, most important command is **SQLiteConnection**.

```
public SqliteDataAccess()
{
    //connection to my pas-database
    sqlite = new SQLiteConnection("Data Source=pas-database.db");
    Connect();
```

Figure 11 - SQLiteDataAccess

Figure 11 shows what happens when an instantiation of my SqliteDataAccess class is made. The private SQLiteConnection ‘sqlite’ is set to the location of my database. The connect function is then launched.

The connect function will simply try open the sqlite connection with a try and catch.

SQLlite – Commands

When using SQlite, creating commands is essential to retrieving information for the database.

```
public int LoadIcuPatients()
{
    int Rowcount = 0;
    SQLiteCommand cmd = new SQLiteCommand(sqlite);
    cmd.CommandText = "SELECT COUNT (id) FROM person WHERE status = 'ICU'";

    Rowcount = Convert.ToInt32(cmd.ExecuteScalar());


    return Rowcount;
```

Figure 12 - SQLite command for ICU patient number

Figure 12 shows the function for the dashboard to show the number of ICU patients in the database. An SQLite command is made and is associated with the specified connection (sqlite).

.commandText will set the command string and executeScalar here is used to return the first column of the first row of the resultset or null if it is not available.

SQLlite – Reader

```
public SQLiteDataReader Query(string QueryString)
{


    SQLiteDataReader Reader = null;
    if (sqlite != null)
    {

        SQLiteCommand Command = new SQLiteCommand(QueryString, sqlite);


        try
        {
            Reader = Command.ExecuteReader();
        }
```

Figure 13 - SQLiteReader implimentation

Figure 13 shows how an SQLiteDataReader functions. If the connection is not null, the command is set and the reader executes with the command.

SQLlite – Prepared statements

Because I am dealing with a database and potentially sensitive hospital information, it is best to provide some level of sql injection protection when dealing with input into the database.

Figure 14 shows what happens when I add a person from the addPatient.xaml. This method is also used on update.

```
SQLiteCommand Command = new SQLiteCommand(AddString, sqlite);

Command.Parameters.AddWithValue("@surname", p.SurName);
Command.Parameters.AddWithValue("@givenname", p.GivenName);
Command.Parameters.AddWithValue("@height", p.Height);
Command.Parameters.AddWithValue("@gender", p.Gender);
Command.Parameters.AddWithValue("@status", p.Status);
Command.Parameters.AddWithValue("@eyecolor", p.EyeColor);
```

Figure 14 - prepared statement on person add

SQLlite – Data Adapters and DataTables

```
public DataTable DataTableQuery(String QueryString)
{
    DataTable Table = null;
    if (sqlite != null)
    {
        //creating a table and then filling it with the query info
        SQLiteCommand Command = new SQLiteCommand(QueryString, sqlite);
        SQLiteDataAdapter Adapter = new SQLiteDataAdapter(Command);
        Table = new DataTable();
        try
        {
            Adapter.Fill(Table);
```

Figure 15 - filling a data table for my search results

On my search window, I need to display as many results match the query that the user sends to my database. A data table here is a good implementation to represent one table of in memory data.

The data adapter here is used to create a connection between the command, and my now table. The fill property in the try on figure 15 adds or refreshes rows in a specific range of the data set to match those in the data source using the data table name.