CA400- Technical Manual

Project Title: Mini Mental State Examination Application

Student Name:	Matthew Nolan
Student ID:	16425716
Student Name:	Michael O'Hara
Student ID:	16414554
Project Supervisor:	Prof. Gareth Jones
Date Completed:	

Table of Contents

1. Overview

1.1 Overview

The project is based on the Mini Mental State Exam (MMSE). The MMSE is a 30-point questionnaire which is used in clinical and research practices to measure cognitive impairment in patients. It is also used in medicine to screen for dementia. The main purpose of this exam is to estimate the severity and progression of cognitive impairment and to monitor the course of cognitive changes in a patient over time. This makes the test an effective method to document a patient's response to treatment. The examination includes questions that gauge the patient's sense of date and time, sense of location, short-term memory, basic mathematics, naming objects and complex cognitive functions like drawing.

This document will cover the key design concepts, how these translated into the development of the application. The challenges during the development life cycle will also be covered. The users of the application must be able to at the very core level be able to carry out the Mini-Mental State Exam and save the results and then be able to go back and the view the results. In order to do this they will be login to the application and select a patient which the exam will be carried out on, start the exam and progress through it and then finish the exam which saves the results, then select that specific exam in order the view the results

The project is an application built using Xamarin Forms for the User Interface. The flexibility of Xamarin allowed for development for both the Android and iOS platforms and the Universal Windows Platform without the need for developing in the separate environments. Using this also allowed for UI components to be converted to platform specific elements at runtime.

The backend is composed of C# and MySQL for the source code and database, respectively. The choice to use C# was made to enhance the difficulty of the project as it was not a familiar language to us and also because of the numerous benefits it offers such as being rich in class libraries containing vast amounts of functions

1.2 Motivation

The way in which the exam is currently administered is manually using a hard copy of the exam. The idea is to modernize the way in which the MMSE is conducted by creating an application which will allow the test to be carried out in a more efficient manner. The application produced will digitise the MMSE and allow for the option, in some cases, for the patient to be able to carry out the exam themselves. The goal for the project is to construct a reliable and user-friendly application that will be able to efficiently monitor levels of cognitive impairments in patients, making the application a suitable information management system for any institution which uses the MMSE

1.3 Research

Before work began on the application, there was some research to be done to gather information in a few key areas. The areas are as follows:

- Creating and hosting Databases
- Cross platform client development
- Application Security
- Modernize pen and paper to application
- Accessibility and User-Friendly UI Design

From looking online and the general idea was that developing the application using Xamarin and C# would allow for seamless development for iOS and Android. Xamarin is an open source app platform

from Microsoft for building modern & performant iOS and Android apps with C# and .NET. It enables this by providing tools and libraries for features such as:

- Libraries for common patterns, such as ModelView ViewModel (MVVM)
- Editor extensions to provide syntax highlighting, code completion, designers, and other functionality specifically for developing mobile pages

(https://dotnet.microsoft.com/learn/xamarin/what-is-xamarin)

Another big help for this project was that AWS has a free tier which would allow for running and hosting a MySQL database which would then be integrated with the source code in order to store/update patient and doctor information. AWS is the world's most comprehensive and broadly adopted cloud platform, offering over 175 fully and featured services from data centres globally. Some of the services offered by AWS cover the areas of:

- Machine Learning
- Databases
- Gaming Technologies
- Blockchain

The service we were most interested in learning about was what they could offer in terms of Database hosting and management. We investigated it and on the beginner tier they offer 12 months of Relational Database Service. This allowed us to set up a MySQL database for storing records and have them be accessible from any instance of the application (with correct login credentials).

There was a significant amount of research done into methods to make the User Interface more accessible and user friendly for both medical professionals and the cognitively impaired, while retaining a high level of functionality and productivity. Research into this topic was done by reading a number of articles on Google Scholar and IEEE Xplore. One of the key articles that was used which outlines the key factors which affect users suffering from cognitive impairment from 2013 can be found here:

(https://www.researchgate.net/publication/236861806_Issues_with_Designing_Dementia-Friendly_Interfaces)

It outlines key factors that need to be taken account of such as limited motor function or decreased visual impairment. We used the information we found in the articles we used to aid in the design of our User Interface.

Another article that was found and used can be found here:

(https://ieeexplore.ieee.org/abstract/document/4224212?casa_token=RkBdbm47t5cAAAAA:uF8TCELN-8u_6J743PfXXUcywNFVGXq0cdrgevydqyXdEltpaS6Oo0VS0y8lD9770ouuG5YXPA)

2. Design

2.1 Design Overview/Concept

This is an overview of what the design concept and use case of the application for a medical professional would be when dealing with a patient.

When the app is initially launched the user is greeted by a Login Screen. Here the user will enter their login details and be given access to the application. If the user does not have an account, they will create one and will be given an ID number which will be used to link the doctor to the patient when carrying out the test.

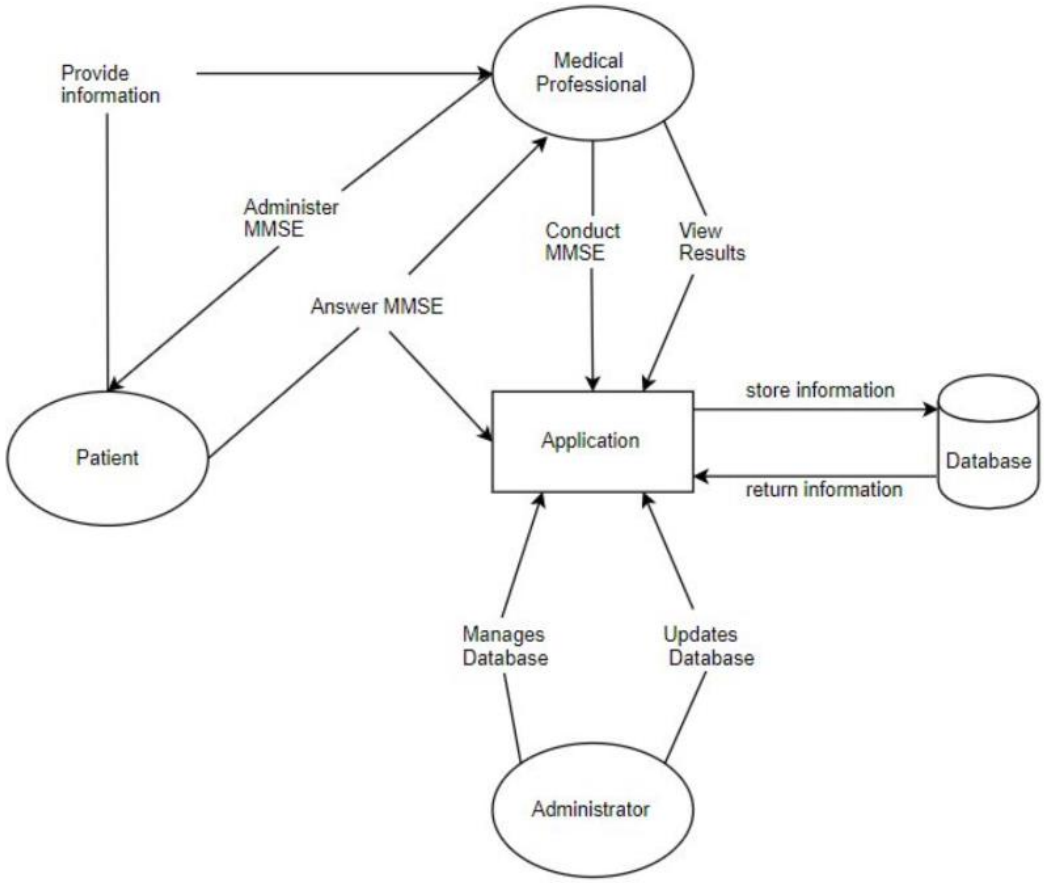
Once the user enters the application, they will be greeted by a home screen or dashboard. From here the user will be able to access the functions which our application offers.

They will also be able access a list of patients in their care. This was to facilitate the reviewing of results after the fact to gather an appropriate diagnosis. This option allows the doctor to review and compare results to previous results (if the patient has carried out the exam before). In this section the doctor is able to add a new patient to the database. In this section the doctor will add in the patients first and last name, their age. Then as the patients carries out exams their profile will keep a record of when that test was and what score they got in that test.

The doctor can also view a calendar from this screen which will allow them to see upcoming consultations with patients or it will allow them to schedule the next consultation with a patient by adding it into the calendar.

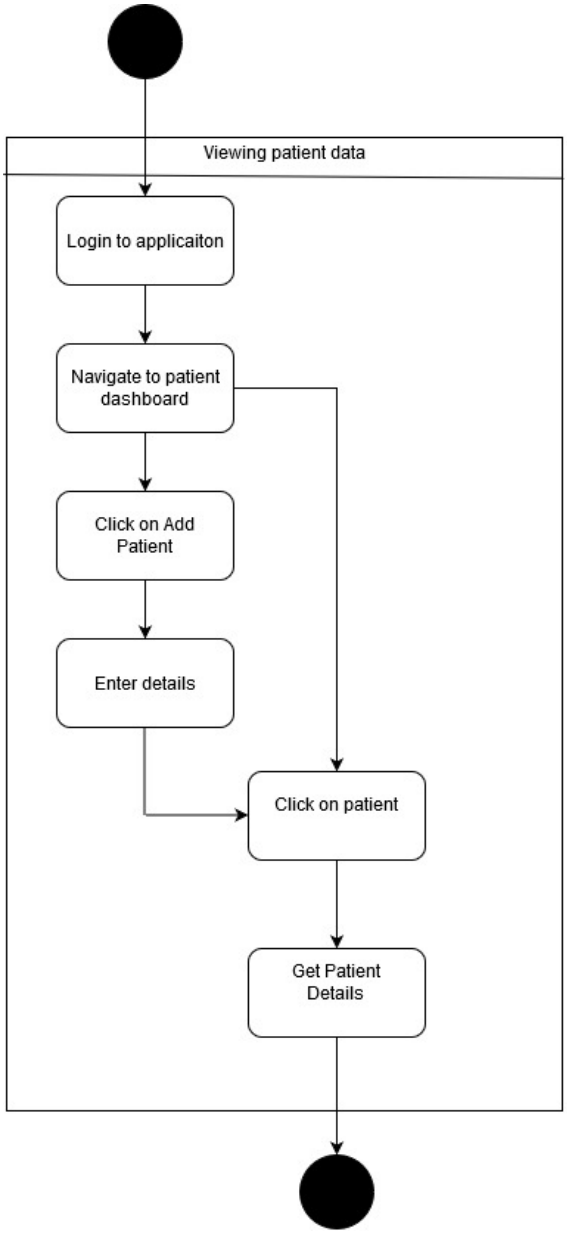
The user can start up the test and enter in the patient's ID and name in order to link the results to said patient. This will bring the medical professional to the exam page. The exam is broken down into 6 sections, which are *Orientation, Registration, Attention and Calculation, Recall, Language and Copying.* These sections each contain a number of questions which are designed to test the mental faculties of the patient in order to determine the level of cognitive impairment in the patient. The questions range from questions like *"What is todays date? Or What is the name of this hospital/location"* all the way to instructions like "*Copy this image or follow this 3-stage command".* The answers will then be saved and graded by the application and return the result on the screen to the medical professional. This eliminates the need for the doctor to manually correct the exam and calculate the results for the patient.

2.2 Context Diagram



Above is the context diagram for the Application. This diagram outlines the relationship the system has with the other entities inside the system. We see the users, in this case the Doctors and Patients using the system in order to carry out the exam. It shows the application interacting with the Database, by storing information or querying the database to get the information.

2.3 State Diagram



The above is a state diagram going through viewing a patient's data inside the application

Technologies Used

2.4 User Interface

The application Design is built using Xamarin for designing the User Interface. There was a number of key reasons this was chosen as the main approach

2.4.1 Single Technology Stack

As Xamarin uses C# complemented with the .NET framework it allows for easy creating apps of each of the mobile platforms such as iOS and Android. This allowed for reusing a lot of source code and sped up the development life cycle. It also meant there was no need to switch development environments when developing for each platform and development for all platforms could occur natively from Visual Studio.

2.4.2 Simplified Maintenance

It also simplified the maintenance and updating life cycle of the application. Due to its cross-platform nature deploying changes or updates was done by applying them directly to the source file and then they were automatically applied to both iOS and Android.

2.4.3 Native User Experiences

It allowed for creating user experiences that mirror native android or iOS development due to the ability to use platform-specific UI elements. This is achieved by using Xamarin forms which converts app UI components into the platform-specific interface elements at runtime.

2.4.4 Hardware Support

Because of the use of Xamarin the applications had native-level app functionality. It eliminated all hardware compatibility issues, by using plugins and specific APIs, to work with common devices functionality across the platforms we were developing for. Along with the access to platform-specific APIs, Xamarin supports linking with native libraries. This opened up more options for better customization and native-level functionality.

2.5 Backend

The backend of the application is a combination of a MySQL Database which is hosted by AWS and the source code which is written in C#.

2.5.1 MySQL

A MySQL database is used as the relational database management system for the project. The hosting of the database is done using the free tier of Amazon Web Service. This was done to keep the database hosted and running at all times. The reasons for choosing AWS to host the MySQL database are as follows

2.5.1.1 Ease of Use

AWS was chosen as it is a very powerful web hosting platform. It also does not require a huge amount of tech skill or expertise to use. The AWS Management Console or well-documented web services APIs to can be used to easily access AWS's application hosting platform.

2.5.1.2 Security & Encryption

Another reason for using AWS is because of its high level of security and ability to protect information. AWS provides a more reliable security measure that is guaranteed to keep your data safe and secure. With 12 data centres scattered across the globe, and another 5 slated to open this year, this is as safe as it gets for your private data and information.

2.5.1.3 Scalable and high performance

Using AWS tools, Auto Scaling, and Elastic Load Balancing, your application can scale up or down based on demand. Backed by Amazon's massive infrastructure, you have access to compute and storage resources when you need them. (https://aws.amazon.com/application-hosting/benefits/)

2.5.2 C#

The reasons for choosing C# for the back-end code logic are outlined below

2.5.2.1 Object-Oriented

C# is an object-oriented programming language. It supports Data Encapsulation, inheritance, polymorphism, interfaces. C# being object-oriented makes development and maintenance easier when compared to other languages which are procedure-oriented programming language

2.5.2.2 Ease of Development

C# language has a rich class of libraries that make many functions easy to be implemented. This compounded with the ability to add more libraries through the use of NuGet packages meant we had a wide range of possibilities to aid the development of our application

3. Implementation

This section will discuss how the main design concepts and user tasks were implemented in the code. There will be snippets of code relating to the frontend and the backend. These 2 sections were developed in isolation from each and then brought together in the later stages of development.

3.1 Side Menu

```
[XamlCompilation(XamlCompilationOptions.Compile)]
4 references
public partial class MenuPage : ContentPage
{
    1 reference
    MainPage RootPage { get => Application.Current.MainPage as MainPage; }
    List<HomeMenuItem> MenuStructure;
    0 references
    public MenuPage()
    {
        InitializeComponent();
        MenuStructure = new List<HomeMenuItem>
        {
            new HomeMenuItem { ID = MenuItemType.Home, Title = "Home"},
            new HomeMenuItem { ID = MenuItemType.Patient, Title = "Patients"},
            new HomeMenuItem { ID = MenuItemType.Exam, Title = "Exam"},
        };

        ListViewMenu.ItemsSource = MenuStructure;
        ListViewMenu.SelectedItem = MenuStructure[0];
        ListViewMenu.ItemSelected += async (sender, e) =>
        {
            if (e.SelectedItem == null)
                return;

            var id = (int)((HomeMenuItem)e.SelectedItem).ID;

            await RootPage.NavigateFromMenu(id);

        };
    }

    0 references
    private void LogoutClicked(object sender, EventArgs e)
    {
        LoginPage main = new LoginPage();
        Application.Current.MainPage = main;
    }
}
```

Above is the code behind the sidebar menu of the application. The code initialises a list of type HomeMenuItems. Each entry refers to a different class of type Page, allowing for easy navigation between pages. The Menu page is a MasterDetail page, with the list being the Master and each page in the navigation list being the Detail. The logout button is located at the bottom of the sidebar. When the logout button is clicked the application main page is set to the Login page and the current user is set to blank, resulting in the user being logged out.

3.2 Adding Doctor Info to the DB

```
1 reference
private DbResult RegisterUser()
{
    DbResult res = new DbResult();
    string cs = @"server=oharam29-nolanm45-mmse-app.c4zhfzwco8qq.eu-west-1.rds.amazonaws.com;Port=3306;database=patient_info;user id=oharam
    MySqlConnection con = new MySqlConnection(cs);
    try
    {

        if (con.State == ConnectionState.Closed)
        {
            con.Open(); // open connection
            MySqlCommand cmd = new MySqlCommand("INSERT INTO TBL_USER(UserName, PassWord) VALUES(@user, @pass)", con); // sql query string
            cmd.Parameters.AddWithValue("@user", Username); // add parameters
            cmd.Parameters.AddWithValue("@pass", Password);
            cmd.ExecuteNonQuery(); // execute the query

            res.msg = "User added successfully";
            res.success = true;

        }
    }
    catch (MySqlException ex)
    {
        res.msg = "Error occured, please try again";
        res.success = false;

        throw (ex);

    }
    finally
    {
        con.Close(); // close connection
    }
    return res;
}
```

The code above shows the backend of the application when a doctor is registering to use the application. At the top of the function the string called ‘CS’ is the connections string for the AWS hosted database. In a future version of the application the database log in details would not be present in the connection string. Due some difficulties with SQLite not being an adequate database solution for the application as was originally thought, and at this late a stage in the project there was no way to rectify this.

Moving on through the code, the MySqlConnection package is used for managing any interaction with the database. Connecting to the database is wrapped inside a try and catch which would allow the code catch ‘MySqlExceptions’ and throw an error message to the user telling them to try again. Inside the try there is an ‘if’ statement to check the connection state to the database and if it is closed then it would move inside the if and open the connection to the database. Then the data is inserted into the database using the ‘INSERT’ command and “Parameters.AddWithValue()” method to add the values that the user enters. It also sets the result success parameter to true and then in the finally block connection.Close() is called to close the connection to the database and finally we return ‘res’ be used in the function which displays the results to the user and handles the navigation to the next page accordingly

3.3 Querying the DB during login

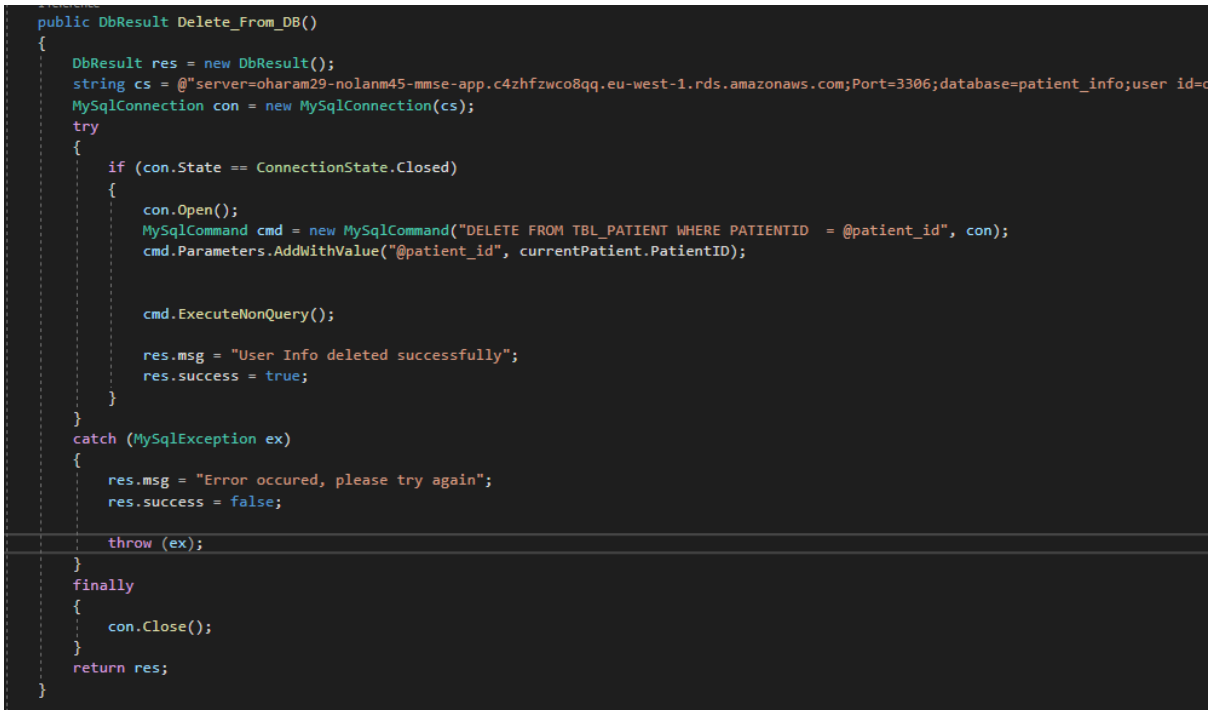
```
1 reference
private DbResult DB_Validate()
{
    List<User> Users = new List<User>();
    string cs = @"server=oharam29-nolanm45-mmse-app.c4zhfzwco8qq.eu-west-1.rds.amazonaws.com;Port=3306;database=pat
    MySqlConnection con = new MySqlConnection(cs);
    try
    {
        if (con.State == ConnectionState.Closed)
        {
            con.Open(); // open db connection
            string query = "SELECT UserName, PassWord FROM TBL_USER"; // sql query string
            using (MySqlCommand command = new MySqlCommand(query, con))
            {
                using (MySqlDataReader reader = command.ExecuteReader())
                {
                    while (reader.Read()) // read in query results
                    {
                        User U = new User
                        {
                            UserName = reader.GetString(0), // get the corresponding values
                            PassWord = reader.GetString(1)
                        };
                        Users.Add(U); // make a list of all users
                    }
                }
            }
            DbResult result = new DbResult();
            foreach (User user in Users) // iterate through the list of users to find desired user
            {
                if (user.UserName == Username)
                {
                    if (user.PassWord == Password)
                    {
                        result.msg = "Login Success"; // if username and password are correct log user in
                        result.success = true;
                    }
                    else // else do not log user in
                    {
                        result.msg = "Incorrect Password";
                        result.success = false;
                    }
                    break;
                }
                else
                {
                    result.msg = "Incorrect Username";
                    result.success = false;
                }
            }
            return result;
        }
```

This code is the backend code which runs when an already registered user tries to login to the application. This code will initialise a list of users which have the parameters UserName and PassWord. This method re-uses the same code for connecting to the database. Once again using ‘MySqlCommand’ but this time there is a string called ‘query’ which gets all the usernames and passwords in the User table. MySqlDataReader is then used in order to read through the results which were returned. A new user is then created for each row in the database which was returned and set the UserName and Password for that user and then add that user to the list of users.

After each of the users has be set and added to the list. A for loop will read through all the usernames in the list and see if one matches the one the user entered. The same thing is done for password. If there is a match for both the application will return a message saying, “Login successful”. The code also sets result to true and returns that to be used later. If one of the parameters doesn’t match, result.message is set to a message saying, “Incorrect

Username/Password" (depending on which one was wrong). This will be used in the next function to run in the code which will display the message to the user and navigate to the page which is required next.

3.4 Deleting a Patients information from the Database

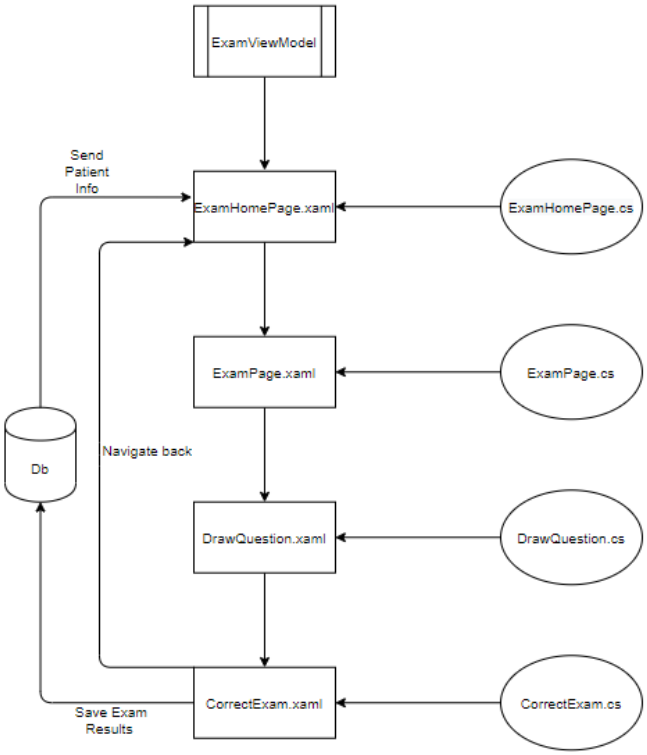


```
public DbResult Delete_From_DB()
{
    DbResult res = new DbResult();
    string cs = @"server=oharam29-nolanm45-mmse-app.c4zhfzwco8qq.eu-west-1.rds.amazonaws.com;Port=3306;database=patient_info;user id=
    MySqlConnection con = new MySqlConnection(cs);
    try
    {
        if (con.State == ConnectionState.Closed)
        {
            con.Open();
            MySqlCommand cmd = new MySqlCommand("DELETE FROM TBL_PATIENT WHERE PATIENTID  = @patient_id", con);
            cmd.Parameters.AddWithValue("@patient_id", currentPatient.PatientID);


            cmd.ExecuteNonQuery();

            res.msg = "User Info deleted successfully";
            res.success = true;
        }
    }
    catch (MySqlException ex)
    {
        res.msg = "Error occured, please try again";
        res.success = false;

        throw (ex);
    }
    finally
    {
        con.Close();
    }
    return res;
}
```

The above code is the backend code which runs when a medical professional clicks the delete patient info button on the patient's profile from within the application. This code follows the same premise for connecting to the database as the last two code snippets as it was felt this was a good and efficient way to do so. It was decided it was better to connect to the database only when there was a need to directly perform an action on it and close the connection after the action was completed rather than leave the connection open as the MySql database has connection pooling internally to manage to the manage concurrent connections.

This code runs a SQL query which removes the record of the patient with the same patient ID that matches the open patient record. It will also set res.msg to "User info deleted successfully". And sets reg.success to true in order this to be checked in the function which will display a message to the user and handles the navigation after this function is finished executing.

3.5 Conducting the Exam



When implementing the conducting exam functionality a Model View ViewModel approach was adopted. The design in the above diagram encapsulates the approach taken. At the top of the diagram is the ExamViewModel. TheExamViewModel is used throughout the Exam. It contains references to the current patient, current doctor, each exam answer and the Exam Score. ExamViewmodel gets initiated and using Binding context in Xamarin, each answer gets bound to a corresponding value in the viewmodel. The Viewmodel is of the type “INotifyPropertyChanged”, which is an event based class. This means that whenever an answer is typed into a bound textbox an event is fired which updates the variable in the ExamViewModel.

The ExamHomePage is the starting page of the exam. Before an exam can start a patient must be selected, using the search bar. The search bar will only show patients which are made by the user. This allows for each medical professional to have their own set of patients. Once a patient is chosen the user can select start exam. Start exam will navigate the user to the exam page.

The Exam page hosts the question and textbox for answers for the exam. As mentioned each answer textbox is bound to a variable in the viewmodel. The final question in the exam requires the user to be able to draw on the screen. Drawing.xaml contains this functionality. This page contains a “Finish Exam” button, once clicked the user is brought to the correct exam page. Correct exam page gives the user the ability to mark the answers right or wrong, automatically updating the Exam Score in the ExamViewModel. When finished correcting the Exam view model is then saved to the database.

4. Testing

This section will outline the tests which were carried out on the application during the development lifecycle. The database tests which will be discussed later are fairly simple in nature, this is the due the fact that during development the database had to be changed to suit the needs of the application so these tests were made very late in the development lifecycle. That being said the results the tests provided allowed for the connection of the application to the database.

4.1 Environment Capability Testing

Environment testing is the testing of the application in a number of different environments to ensure the application can function as desired on a number of different devices such as windows laptops, Android devices and iOS devices. Due to flexibility of Xamarin it is designed to work on all of these devices. The reason for this type of testing was to account for different software version and screen sizes. For this project, a majority of this type of testing was carried out on Android Devices and Windows laptops due these being the devices readily available.

The Android testing was carried out on 2 different smartphones with varying software versions. These being the Huawei P20 Pro running EMUI Version 9.1 based on Android Version 9, the OnePlus 8 Pro running Oxygen OS Version 10.5.8 based on Android version 10. These were the 2 devices that for available for the duration of the project.

There was also testing done on a Google Pixel 2 running Android 8 using he built in emulator inside of visual studio, to account for older devices which may be used to run the application. These tests ensured the application ran smoothly across a range of devices during the implanting of new features during the development.

4.2 Database Tests

These are just a few basic tests which were used to make the sure the database and the backend code were functioning correctly.

```
public void Test_Connection()
{
    string cs = @"server=oharam29-nolanm45-mmse-app.c4zhfzwco8qq.eu-west-1.rds.a
    MySqlConnection con = new MySqlConnection(cs);
    try
    {
        con.Open();
    }
    catch(MySqlException ex)
    {
        Console.WriteLine("TEST FAILED - Unable to connect");
        throw (ex);
    }
    Console.WriteLine("TEST PASS - Successfully connected to the DB");
}
```

These are some of the tests which were performed on how the backend interacts with the database to make sure they were both working correctly. The above code snippet is the test used to verify the application is able to connect to the database. This test was used when the database was first linked to the backend.

```
public void Test_Query()
{
    string cs = @"server=oharam29-nolanm45-mmse-app.c4zhfzwco8qq.eu-west-1.rds.amazona
    MySqlConnection con = new MySqlConnection(cs);
    con.Open();
    MySqlCommand cmd2 = new MySqlCommand("SELECT stringTest FROM TBL_UNITTEST", con);

    using (MySqlDataReader reader = cmd2.ExecuteReader())
    {
        reader.Read(); // read in query results
        String test_string = reader.GetString(0);
        if(test_string != "test")
        {
            Console.WriteLine("TEST FAILED - Value does not match query");
        }
        else if(test_string == "")
        {
            Console.WriteLine("TEST FAILED - Value does not match query");
        }
    }
    Console.WriteLine("TEST PASSED - Value in database matches test value");
}
```

The above code fragment is the test used when testing to make sure the backend can query the database in order to get information from the database. It will query the database looking for the column “stringTest” which contains the string “test”. The code then converts the result of the query to a string and will compare the string to what the string should be. If the strings match, then we know the code is functioning correctly.

```
public void Test_Insert()
{
    string cs = @"server=oharam29-nolanm45-mmse-app.c4zhfzwco8qq.eu-west-1.rds.amazonaws.com;Port=3306;
    MySqlConnection con = new MySqlConnection(cs);
    con.Open();
    MySqlCommand cmd1 = new MySqlCommand("INSERT INTO TBL_UNITTEST(insertTest) VALUES(insert)", con);


    MySqlCommand cmd2 = new MySqlCommand("SELECT insertTest FROM TBL_UNITTEST", con);

    using (MySqlDataReader reader = cmd2.ExecuteReader())
    {
        reader.Read(); // read in query results
        String test_string = reader.GetString(0);
        if (test_string != "insert")
        {
            Console.WriteLine("TEST FAILED - Value not inserted");
        }
        else if (test_string == "")
        {
            Console.WriteLine("TEST FAILED - Value not inserted");
        }
    }
    Console.WriteLine("TEST PASSED - Value inserted successfully");

}
```

The above code fragment was used to test the ability of the backend to insert data into the database. It begins by connecting to the database and once connected it will insert a string into the column “insertTest”. It will then query the database to verify that the value was inserted correctly. This test was used in conjunction with a test which will delete the record which was inserted in order to keep the test table from being encumbered with useless information

4.3 User Testing

Once the ethical approval process was completed the goal was to produce a working prototype of the application to allow users to begin testing and providing feedback. Due the strict social distancing measures implemented by the government at the tail end of march the testing process had to be cut down to a smaller group. The testing process was modified to be carried out via the use of email and zoom.

The group of testers was composed of various students from a number of different courses in DCU which were available via zoom to help test the application. The testers were asked to carry out a number of tasks such as :

- Login to a pre created doctor profile
- Create a patient entry for themselves
- Carry out the exam
- View their results

Once these tasks had been completed the testers were asked to fill out a questionnaire to provide feedback on the application and the experiences, they had with it. A majority of the feedback that was received was positive, but it also highlighted a number of things which required attention. From the feedback that was received from these questionnaires, it was discovered that some pf the testers had issues with some of the questions loading and an issue with seeing the newly created patient when returning to patient screen from creating patient.

This feedback was invaluable in the development process as it allowed for the ironing out of bugs and defects which would not have been found till much later if there was not any user testing done.

5 Challenges Overcome

This section will outline the challenges that were faced during the development lifecycle of the project. The challenges which were faced during the project are broken down into 2 sections, they are Real World Challenges and Development Challenges. Real World Challenges are challenges which were outside control of any of the parties involved in the development of the project i.e. the outbreak of Covid-19. Development challenges are challenges which were encountered directly during the software development of the application.

5.1 Real World Challenges

Even though this is not considered a direct challenge to the development of the application the Covid-19 outbreak was a challenge which had to be overcome during the life cycle of the project. Had the outbreak not occurred we would have worked on the project in the School of Computing computer labs but as it did occur, there was a need to adjust to working together remotely. Zoom was a key part in helping to maintain communication via video and it allowed co-ordination on what needed to be done in order to continue development.

Michael is also a part time worker in a shop which is deemed an essential service and as a result of lockdown the shop got busier as everyone was out purchasing their items day to day needs meaning that all workers were asked to do more hours meaning his time became divided between working and completing studies. Through careful planning and time management sufficient work done in order to complete the tasks which were set out to complete work on the application.

5.2 Development Challenges

5.2.1 C#/Xamarin Full Stack Application

The first challenge that was faced during this project was developing a full stack application in C#/Xamarin. A big reason this was a challenge is the fact that C# and Xamarin forms are not covered in the main course content in Computer Applications. That being said the decision to develop an application using these technologies was for the purpose of acquiring a better understanding of different practices that can be applied in future software projects.

Due to the unfamiliarity with both C# and Xamarin, the development at the beginning of the project was slow and inconsistent. The was caused by our lack of knowledge of the syntax but as time passed and confidence was gained the development sped up and became more efficient.

5.2.2 Hosting the Database

Another big challenge that was faced was where the Database would be hosted in order to allow connectivity at all times whenever it is needed. It was known that this would be an issue from the start of the project, but it was believed that the use of SQLite as the database of choice would solve this. Unfortunately, this would lead to separate instances of the DB for every different device the app is used on rather than one shared database each device connects to.

This led to finding out about using AWS as the method for hosting a MySQL database which each instance of the application will connect to and send data to and retrieve data from. This would also for a user to login on multiple devices if one were to stop working or to be misplaced. The AWS web console in conjunction also allowed for fast management of the database

5.2.3 Integrating Source Code/DB with UI

Another big challenge was the integrating of the source code containing the backend functions and the code for connecting to, querying, and writing the database with code written in Xamarin which defined the User Interface. Working with each part of the code had been done in isolation in

previous modules and even on INTRA placements, integrating all the part together presented new challenges to the software development process.

After a couple of hours working through each element of the UI and linking the backend code to that related to that element, we had a functional application.

6 Future Work

6.1 Converting application to companion

Due to the way hospitals are run, hard copies of records are used to track of everything. Given the wide range of technologies available in today's age the field of medicine should also be able to follow suit. One possibility is that the application could be updated to be transformed into a companion application for medical professionals. The medical professional would be able to take notes and add information to a patient file straight from the application. They would be able to view data on all the patients in their care all in the one place rather than needing to go their office to get the files on different patient. This would reduce the time between seeing different patients.

6.2 Adding more examinations

Another avenue that could be ventured down would be adding more exams that the medical professional can carry out. These could be added into the application under a separate menu heading and the results for these tests could be added into different tables in the database. Given the different nature of these exams, they would need some tinkering to be adapted to the application format.

Some of these could include:

6.2.1 NIH Stroke Scale

The national Institutes of Health Stroke Scale is a tool that is used by all major healthcare providers to objectively quantify the impairment caused by a stroke. Its composed of 11 different items, each of which scores a specific ability between a 0 and 4. Scores in each area of 0 indicate normal function in that area and that a higher score is indicative of a level of impairment. After each section has been completed the scores are added up and a result is determined. The maximum possible score is 42, with the minimum score being a 0.

6.2.2 Self-administered Gerocognitive Examination

The self-administered Gerocognitive Examination is a brief cognitive assessment instrument for mild cognitive impairment (MCI) and early dementia. It was created by Douglas Scharre, who is a Professor of Clinical Neurology and Psychiatry at Ohio State University Wexner Medical Centre

6.2.3 Informant Questionnaire on Cognitive Decline in the Elderly

This a questionnaire that can be filled out by relative or other supporter of an elderly person and this can determine whether that person has declined in cognitive functioning. This exam is used as a screening test for dementia. This questionnaire contains 26 everyday situations where a person has to use their memory or intelligence. Example of such situations include: "Remembering where to find things which have been put in a different place from usual". The scored for IQCODE is scored by averaging the ratings across the 26 situations. A person with no cognitive decline will score an average of 3 across the board. This exam could be used in combination with the Mini-Mental State Examination to improve the detection of dementia.

7 Appendices

Xamarin - https://dotnet.microsoft.com/apps/xamarin

C# - https://docs.microsoft.com/en-us/dotnet/csharp/

AWS - https://aws.amazon.com/products/?pg=WIAWS-N&tile=learn_more

MySQL - https://en.wikipedia.org/wiki/MySQL