**Declaration Cover Sheet for Project Submission**

**SECTION 1** *Student to complete*

| |
|---|
| **Name: Andrew Bradley** |
| **Student ID: X12118770** |
| **Supervisor: Paul Hayes** |

**SECTION 2 Confirmation of Authorship**

*The acceptance of your work is subject to your signature on the following declaration:*

I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature: Andrew Bradley

Date: 10/05/16

NB. If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the College's Disciplinary Committee. Should the Committee be satisfied that plagiarism has occurred this is likely to lead to your failing the module and possibly to your being suspended or expelled from college.

**Complete the sections above and attach it to the front of one of the copies of your assignment**

National College of Ireland
BSc in Computing
2014/2015

Andrew Bradley
X12118770
Andrew.bradley@student.ncirl.ie

# Imperium

Technical Report

**Table of Contents**

**Executive Summary**

Imperium is a reliable diabetes management tool that simplifies diabetes care by allowing people to record and monitor their blood sugar level quickly and easily, and facilitating communication between individuals and healthcare providers. In addition to assisting with logging blood sugar readings the mobile application, offers more advanced functionality if the user wishes to use it. This includes the ability to view historical results in an array of different graphs, which allows the user to quickly identify trends, such that patterns of problematic readings can be addressed. Equally, stable readings will help to reinforce the benefits of a healthy lifestyle, which should in turn encourage the user to continue with that approach.

To facilitate communication between the user and their doctor/healthcare professional (HCP) it allows results to be shared both via email with a csv attachment directly from the App, and also by means of the external sync option. The latter allows the user to sync all recorded results from the devices local database with an external website. The doctor can then log in to this website and see real time data and advise the patient on any changes needed.

An appointments section helps the user keep track of upcoming appointments and a notification of an upcoming appointment servers as a reminder to either send or upload your latest results.

For development Visual Studio is being used and languages used are C# and XAML. SQLite is being used for the database which is stored on the devices and Microsoft Azure will host the website. The software architecture being followed is the MVVM (Model – View- ViewModel) approach allowing the GUI to be easily resized while sharing the rest of the code across different devices.

# 1 Introduction

## *1.1 Background*

I have had diabetes for the past 17 years and so I was personally motivated to build an application to assist with the daily care required to control blood sugar levels. Such control is one of the most important aspects to the management of diabetes, as stable normal levels are essential to good health. The App needed to be both reliable and user friendly, with streamlined functionality in order to ensure that it can easily be incorporated into most, if not all, the user's daily routines. Given the frequency with which blood sugar have to be checked, time is of the essence to encourage the regular tracking of levels that is needed to maintain good control.

Currently I have to record my results either on paper or excel and then print them out and bring them with me when going to the hospital. Usually this means transferring the previous two months' worth of readings from my tester and then printing them out the night before the hospital appointment. This is time consuming and cumbersome. It would be ideal to be able to log results on-the-go at the same time as I take my reading, and know that this is then available for my doctor and myself to access and analyse as needed.

Having tried some Apps that are available I found that most take an unnecessary amount of time to enter a reading as you have to scan through and find the exact time and date for every entry. As these readings are taken several times a day, this lengthy approach outweighed any benefits that the Apps offered and I found that they did not encourage me to keep my recordings up to date.

I wanted my App to allow me record my blood sugars with as few button presses as possible and to provide me with an easy way to share my readings with my doctor. My first idea involved setting up a number of pre-time slots such as Morning, Lunch etc. which could be selected when entering a reading. The option to add exact time can be used if preferred but as readings are, for the most part, taken at certain regular intervals throughout the day, the one click easy approach is expected to be the most used.

I am also required to attend the hospital multiple times a year for various checkups and blood tests. The App will have an area to enter appointments and will give reminders the

day before as to an impending appointment. From this reminder you can choose to email results to a specified address such as your doctors. It is a timely development given that the health service is to introduce electronic health records and individual health identifiers (IHI) in the near future. I believe that such an electronic integrated system points to the way in which all future healthcare will be delivered and engaged with.

### *1.2 Aims*

The aim of the project is to develop a Windows phone\tablet App that allows a user to record his or her blood glucose levels and, most importantly, to review these results in order to assist them to maximise their blood sugar control. Such control is vital to good health. As levels have to be checked several times a day, every day, one of the primary uses of the App is to minimise the amount of time needed to manage glucose levels. Users can download the App to their Windows Phone or tablet and then launch the application. From here they can setup pre-determined time slots such as breakfast, lunch, dinner etc. or morning, afternoon, evening etc., with 12.30pm to 2.00pm being assigned for lunch, for example, or as best suits their own daily routine. This first step allows each individual to personalise the App to suit their own circumstances. Readings added as lunch will in future be recorded as 12.30pm. Once set up is complete, readings can be recorded at the click of a single button 'Add result', saving valuable time each day. This streamlined approach may also encourage more regular recording of levels. A degree of flexibility will also be built in. To capture late readings, you can alternatively click on one of the time slots 'Lunch', to record it in this time period 'after the fact'.

As users can find themselves in a situation where they have to transfer a lot of readings at once there will be an option to select a start and end date and also apply a period filter such as Lunch. This will allow the user to enter a reading and the App will move on to the next date already filtered to Lunch so the user just enters the figure and presses next. This will allow for quick entry of any backlog of results the user may have.

Aside from recording a user's readings the App will also provide useful feedback, by displaying graphs indicating trends from a user's readings and these will be filterable with a start and end date. The graphs will provide an at a glance health report for user,

doctor and parents of children and allows you to easily identify problem areas without having to go through lists of results.

Another option the Application will have is the ability to send results to your doctor or healthcare professional. Such transfers will be in csv format and again you can select a start and end date and then enter an email address the results will be sent to. Alternatively you can pre-set an email address and have it populated automatically when you enter the send email page.

The App will have a calendar function were you can enter upcoming hospital appointments, and reminders can be set for any imminent appointments. This should be particularly helpful, given the wide range of healthcare professionals involved in providing diabetes care.

The final section of the App will allow the user to link the App to a website. If they are availing of this option they will have to enter a username and password and also their hospital number. This will then setup an account on the website and sync all results entered in the App with the websites database. Here they can view trends and graphs of their historical results. Doctors/HCPs can also login to this website and view results in real-time.

It is anticipated this feature will be of particular benefit to young people in encouraging, and facilitating, their engagement with their HCP in the management of their condition.

### *1.3 Technologies*

I will be using Microsoft Visual Studio to create the App for windows phones\tablets and coding in C# and XAML.

The website will also be built using Visual Studio and hosted on Microsoft Azure and will also feature C# along with ASP.NET and HTML.

The database for the App will be SQLite and for the website will be SQL.

The App can be used as a standalone App if the user does not want to use the external sync option but if they do they will need to connect to Wi-Fi or a 3\4G connection.

# 2 System

## *2.1 Requirements*

### 2.1.1 Functional Requirements

- Record blood sugar readings – The most important thing the App must accomplish is to allow a user to enter his or her blood sugar glucose readings quickly. This will be the default setting on the reading screen.
- Record reading by time slot – The user must be able to enter readings by time slots. This enhances a user's potential to review previous recordings.
- Record reading by date\time entry – The user must be able to enter readings by entering an exact date and time.
- Setup time slots for readings – The user must be able to setup time slots to allow for quick entry of readings into one of these pre-set time slots.
- Enter historical readings by time slot – The user must be able to enter a range of results for a select or all time slots.
- Email results – The user must be able to email results from the App to a specified email address.
- View charts – The user must be able to view all blood sugar readings for a specified time period in a graph format.
- View data – The user must be able to view the data behind the charts above. This will be in the form of a data table.
- Enter appointments – The user must be able to input hospital appointments on a calendar.
- Set reading type – The user must be able to select mmol or mg for reading type.
- Setup website sync – The user can setup a username and password to be used to access the website.
- Sync with website – The user can sync their data back to a website
- Login to website – The user can login to a website to view results\graphs from a desktop\laptop.

### 2.1.2 Data Requirements

There is a requirement for the user to save their blood glucose results within the App itself and to also have the option to sync to an external website were their doctor can view their latest results.

For the local data, the App will save the results on the device itself using SQLite. For the external data the local data will be uploaded to a Microsoft Azure SQL database. This will be handled through Azure mobile services.

### 2.1.3 User Requirements

To use the App on a phone or tablet the user will need:

- A device running Windows 8.1. Any phone, tablet or pc\laptop running this OS that downloads the App will be able to launch the App with no problems;
- Access to the internet either through 3G\4G or WIFI will be needed to download the App to the device. Once downloaded the App can function without internet connectivity;
- A windows store account will be needed to be able to download the App

### 2.1.4 Environmental Requirements

The App has been designed to run on Windows 8.1 and Windows 10. The device must be running either of these operating systems otherwise you will not be able to install the App.

To develop the App the following were required:

- Laptop running Windows 10
- Visual Studio 2015 community edition

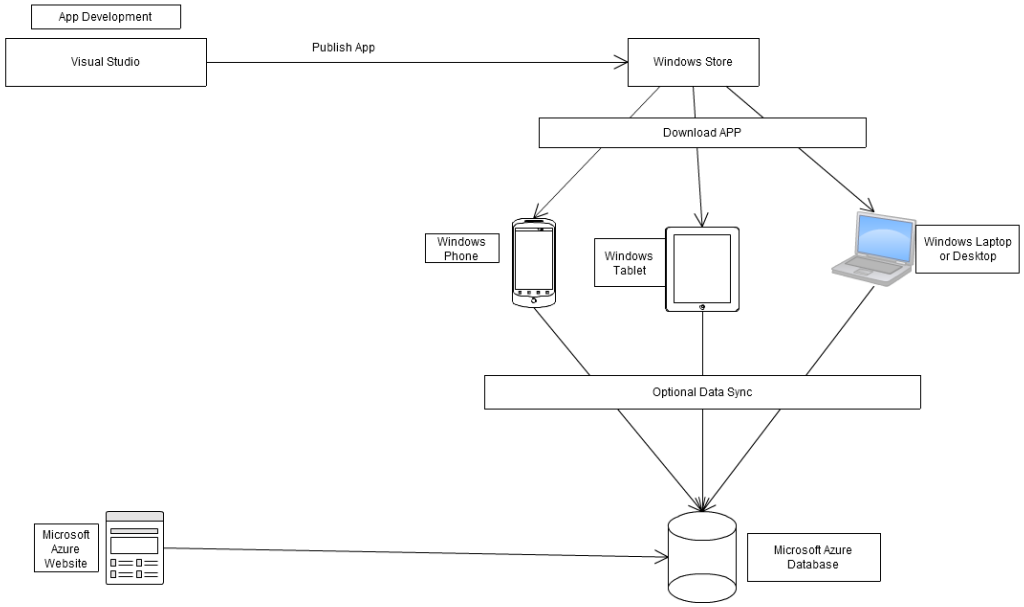
To run the emulator in Visual Studio the development machine must have:

- Hyper-V support
- 6GB or more of RAM
- 64 bit pro version of windows 8 or higher

## 2.2 Design and Architecture

### 2.2.1 Logical View

The below illustration\diagram shows how the App will be installed and used by users. The App will be developed in Visual Studio in C# and XAML for the GUI pages and will be published to the Windows Store and be available to download to the user's device. If they choose to use the external sync option they can sync their App data with the website's database allowing them or their doctor to view their results in here as well as the App.




### 2.2.2 Hardware Architecture

The phone used for the purpose of this project will be a Microsoft Lumia 640 LTE and the tablet will be a LINUX 10" Windows tablet. The phone will be running Windows 8.1, and the tablet windows 10, and both will have the App side loaded during the production

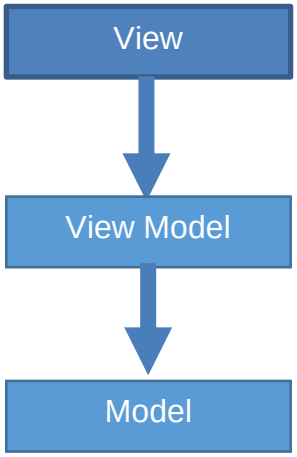
stage. For outside of this project scope any phone or table running Windows 8.1 or Windows 10 can download and run the App from the windows store.

Lumia 640 Phone


Linux 10 Tablet

### 2.2.3 Software Architecture

The App will be developed in Visual Studio community 2015 using the MVVM (Model-View-ViewModel) pattern which is based on the MVC (Model-View-Controller) pattern. With this pattern the GUI development is separate to the business or back end logic. You can easily create a different GUI targeted at a specific device such as a desktop without having to change the other two layers, so you can re-use most of your work.

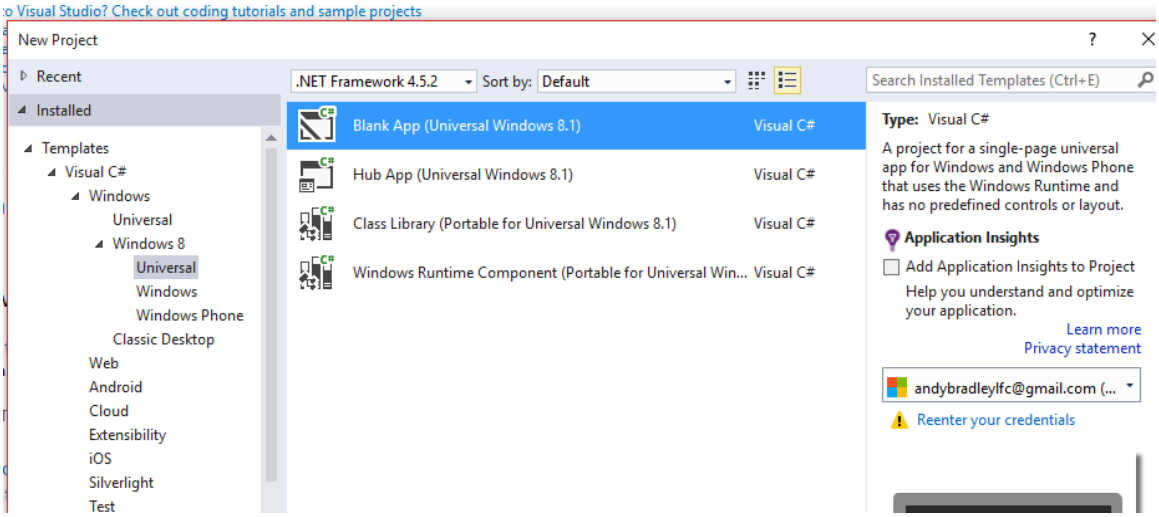



The Model section is the part that the App will consume. This will hold the actual data.

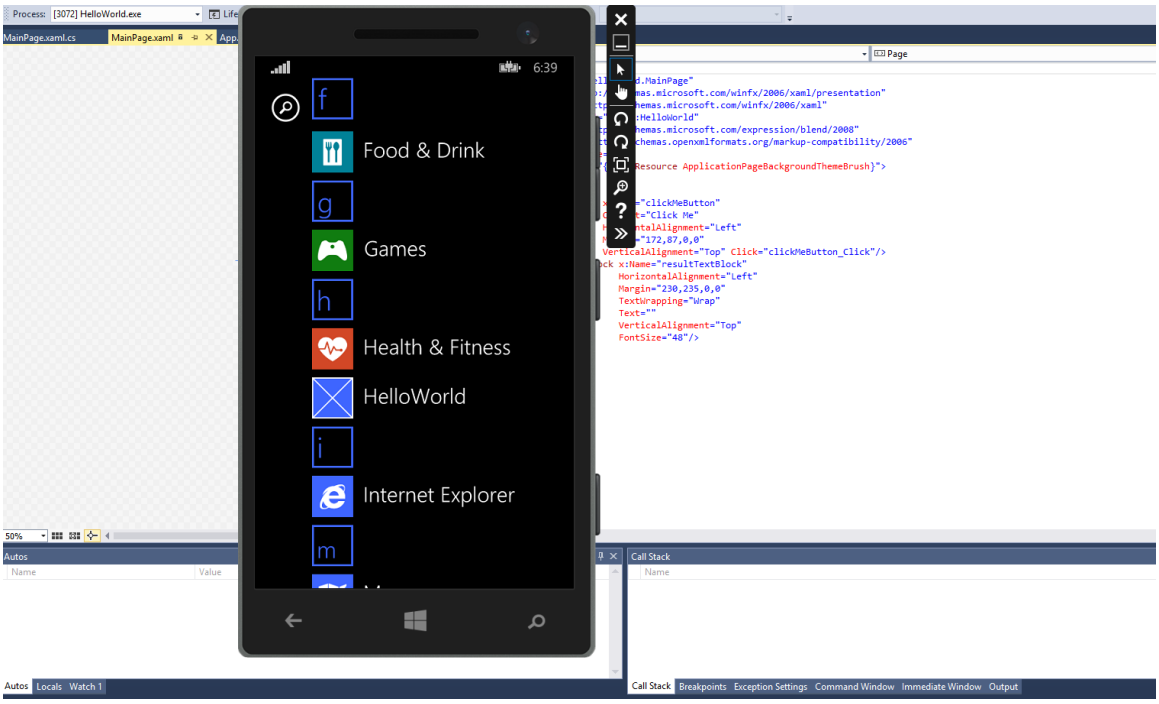
The View section is were the UI pages are held. Every page the user see's is a view in the MVVM terminology. Data can be formatted in the view section to be displayed a certain way and the view will be done in XAML code. The view is bound to a view model.

The View Model is the link between the view and the model. A view is linked to a view model which then gets the data for the view from the model.

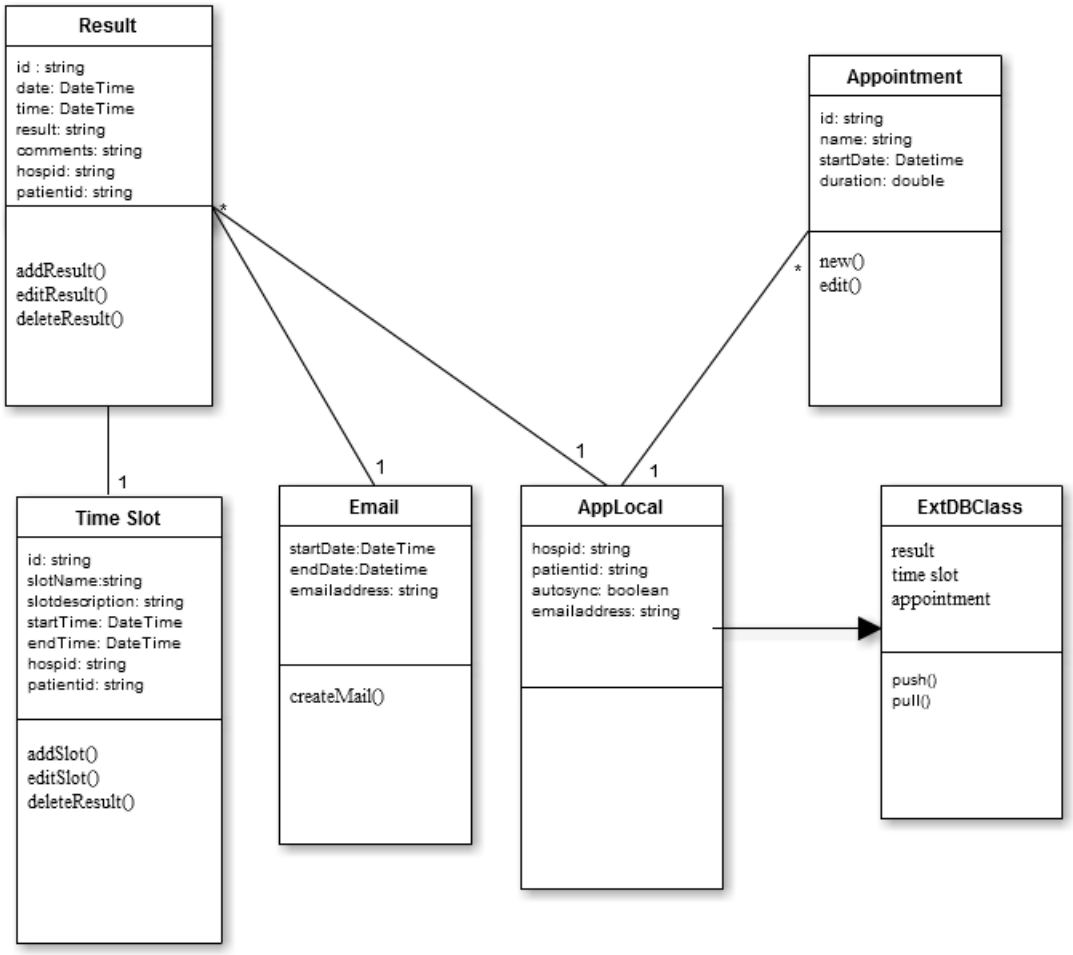
The project will be based on the C# Universal App selection. With this method the published App will run on a phone or tablet\desktop without having to specifically design a new project for each.




To test the App from Visual Studio an emulator will be used. This gives the full functionality of a phone without having to connect an actual device each time you add to or edit the App's code. Your machine must have Hyper-V support and 6GB or more of RAM to be able to run the emulator.



### 2.2.4 System Architecture

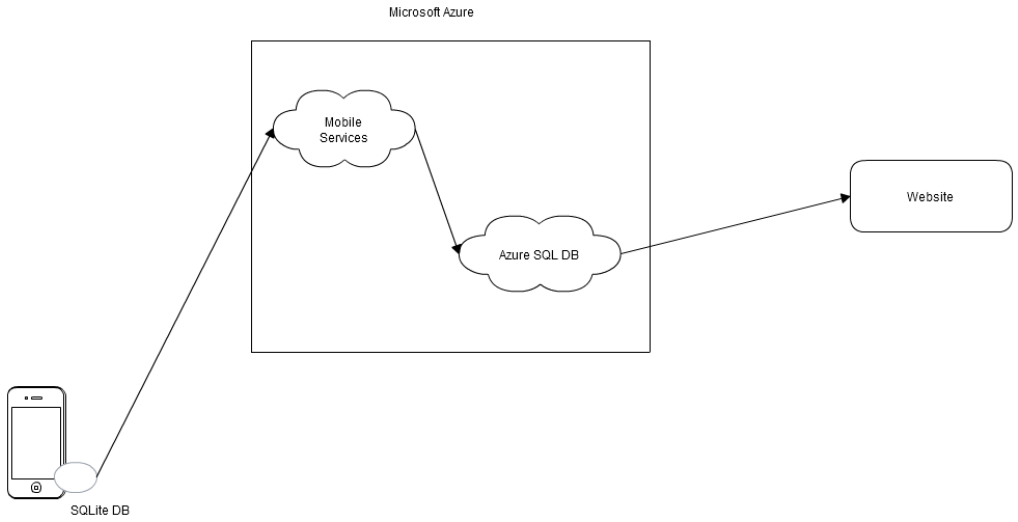



### 2.2.5 Security Architecture

As the App will not contain anything other than the user's blood glucose results there is no need to set a login every time the user wants to access the App as this will take away from the easy and quick to use design. The user should have their phone set to auto-lock so if the phone or tablet is stolen the App will not be accessible if they cannot log in to the phone itself. If they do get access, the data is meaningless to anyone other than the user or their doctor.

### 2.2.6 API Interface

The App will have a function to upload data from the local SQLite database on the phone to a Microsoft Azure SQL stored SQL database. This will be handled through Azure mobile services and the operation will send data only in the initial release.

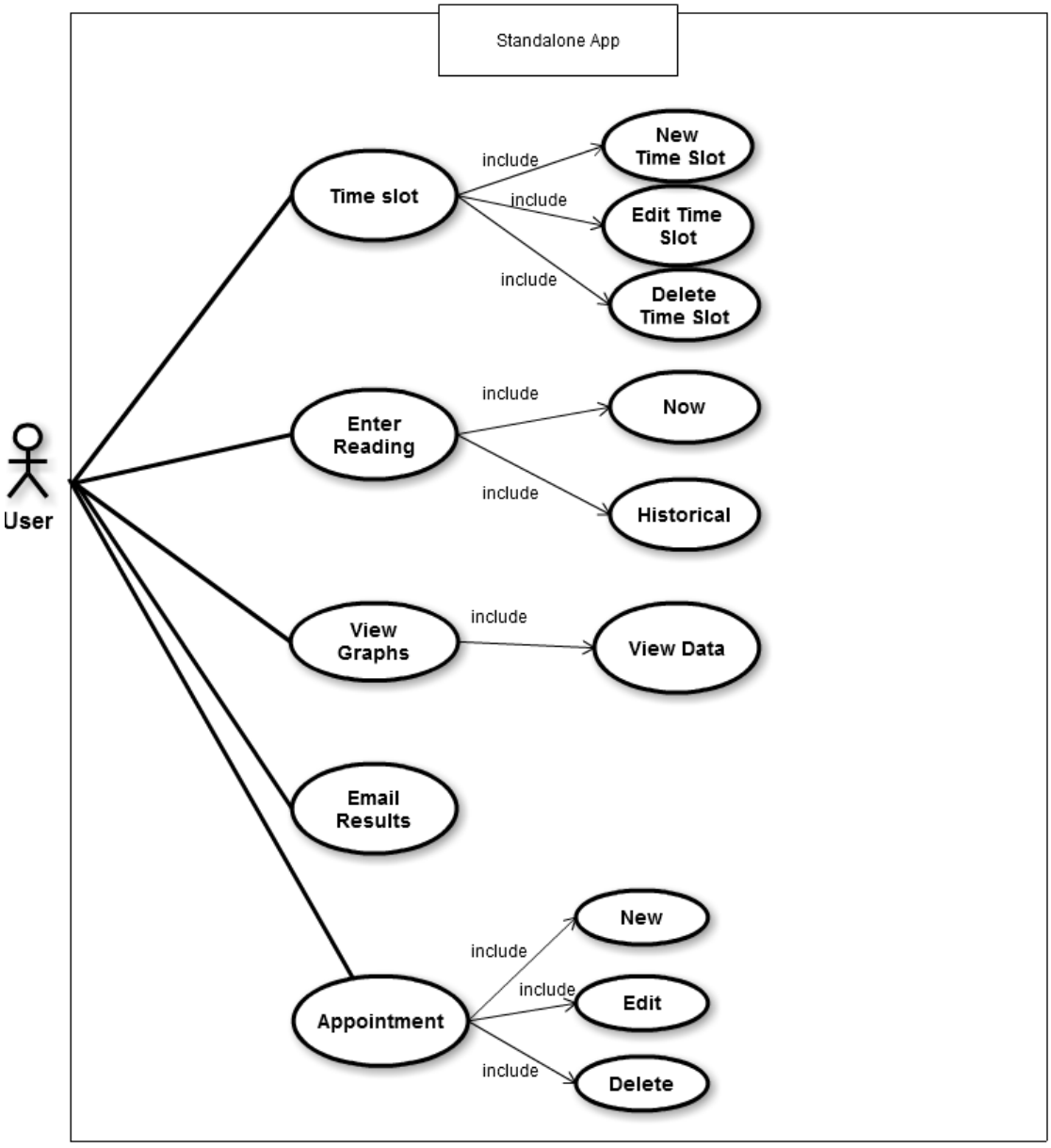



To complete this the WindowsAzure.MobileServices.SQLiteStore Nuget package needs to be added to the App's package.

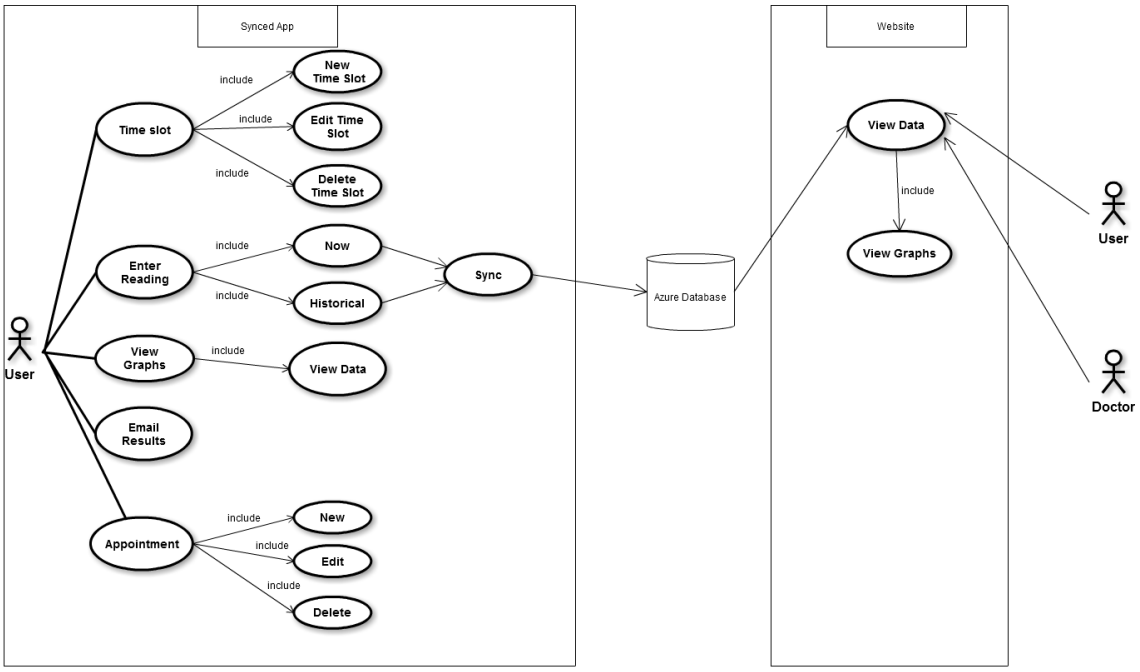
If the user intends on using this feature when they choose to sync the local data, a call is made to IMobileServicessyncContext.PushAsyn(), this will send all the local data to the Azure database.

## *2.3 Use Cases*

The below use case provides an overview of the system from the user's perspective if they prefer to only use the App and do not want to take advantage of the Azure database sync.




This use case illustrates how the user can sync their data with the Azure database and share their results with their doctor.




### 2.3.1 Requirement 1: Enter Reading

Description & Priority

The main goal of the App is to allow a user to enter their blood glucose reading and to store this reading in the App's database.

Use Case

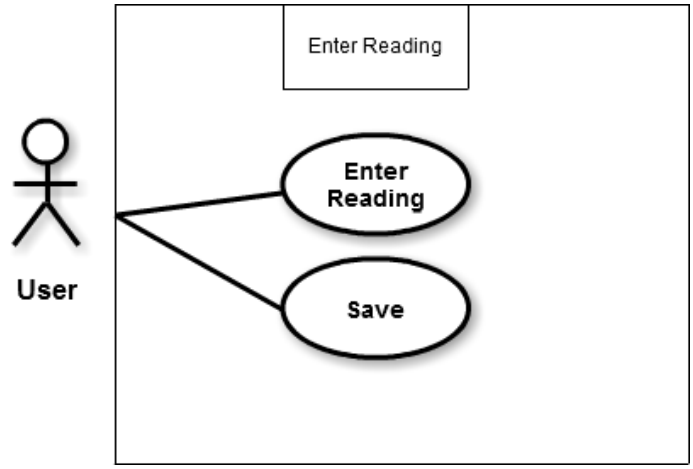
**Scope**

The scope of this use case is to allow the user to enter their Blood Glucose readings.

**Description**

This use case describes the user entering their blood sugar readings.

**Use Case Diagram**




**Flow Description**

**Precondition**

The user has downloaded and opened the App on their smartphone or tablet.

**Activation**

This use case starts when the user has opened the App and selected the enter Reading option.

**Main flow**

1. The user has opened the App and selected enter reading option
2. The user enters a reading and presses ok (See A1 No reading entered)
3. The system stores reading in the database

**Alternate flow**

A1: No reading entered

1. The system indicates that no blood glucose reading was entered
2. The system asks the user to make sure there is a reading entered
3. The use case continues at position 2 of the main flow

**Exceptional flow**

**Termination**

User is back at the main menu screen

**Post condition**

User can select an option from the main menu screen.

### 2.3.2 Requirement 2: Historical

Description & Priority

This will allow the user to easily enter a batch of results, this can happen when they have been unable to enter the reading to the App at the same time as recording the level.

Use Case

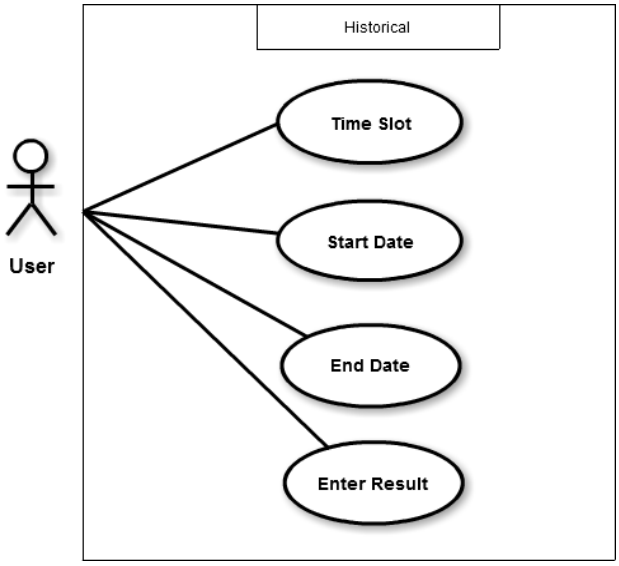
**Scope**

The scope of this use case is to allow the user to enter a batch of historical readings

**Description**

This use case describes the user entering a batch of their historical readings in an easy way

**Use Case Diagram**




**Flow Description**

**Precondition**

The user is in the enter reading section of the App

**Activation**

This use case starts when the user has selected the historical button in the reading section of the App

**Main flow**

1. The user clicks on the historical option
2. The user selects a time slot(s)
3. The user selects a start date
4. The user selects an end date (A1 End date before Start date)
5. The user enters a reading
6. The user selects ok (A2 Check Data)

**Alternate flow**

A1: End date before Start date

1. The system indicates that the end date is before the start date
2. The system asks the user to select another end date
3. The use case continues at position 4 of the main flow

**Alternate flow**

A2: Check Data

1. The system indicates that there is not enough selections made
2. The system asks the user to check all fields are populated
3. The use case continues at position 2 of the main flow

**Termination**

User is back at the historical page

**Post condition**

The App waits for another user selection

### 2.3.3 Requirement 3: Setup Time Slot

Description & Priority

The user has the capability to setup pre-determined time slots to aid in entering the day's readings.

Use Case

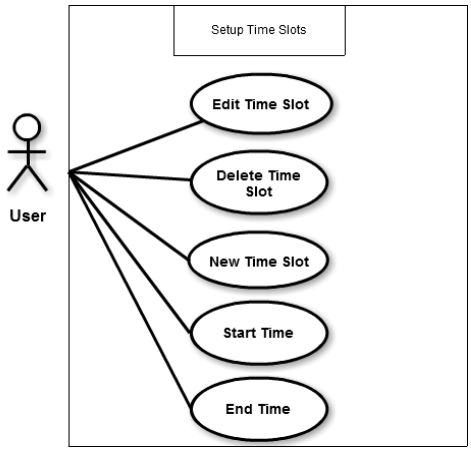
**Scope**

The scope of this use case is to allow the user to enter time slots for the easy recording of readings after the fact.

**Description**

This use case describes the act of setting up time slots.

**Use Case Diagram**




**Flow Description**

**Precondition**

The user has to be in the setup a time slot section

**Activation**

This use case starts when the user selects the setup a time slot option

**Main flow**

1. The user selects the time slot section
2. The user selects setup new time slot option (E1 all time slots taken)
3. The user enters a name for the slot (A1 Name already entered)
4. The user enters a from and to time period for the slot (A2 Time already on a slot)
5. The system stores the slot in the database.

**Alternate flow**

A1: Name already entered

1. The system indicates that the time slot name has already been entered
2. The system asks the user to enter another name
3. The use case continues at position 3 of the main flow

**Alternate flow**

A2: Time already on a slot

1. The system indicates that the time entered is already allocated to another time slot
2. The system asks the user to enter another time range for this slot
3. The use case continues at position 4 of the main flow

**Exceptional flow**

E1: All time slots taken

1. The system indicates that all time periods are already assigned to slots
2. The system says to edit or delete existing slots before adding any more new ones
3. The use case ends

**Termination**

User is back at the time slot section of the App

**Post condition**

The App waits for another user selection

### 2.3.4 Requirement 4: Email Results

Description & Priority

The user has the capability to email results to a specified email address.

Use Case

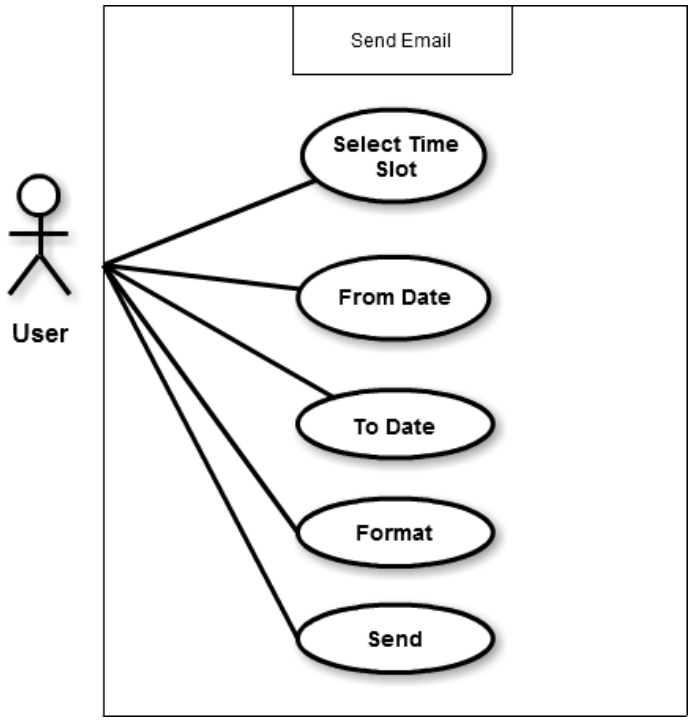
**Scope**

The scope of this use case is to allow the user to send an email containing their results in excel or pdf format.

**Description**

This use case describes the act of sending results by email.

**Use Case Diagram**




**Flow Description**

**Precondition**

The user has to be in the send email section

**Activation**

This use case starts when the user selects to email results

**Main flow**

1. The user selects the send an email option
2. The user selects format
3. The user selects a from time period (E1 no results exist)
4. The user selects a to time period (A1 to time period cannot be before from time period)
5. The system attempts to send the results (E2 no email client setup)

**Alternate flow**

A1: To time period cannot be before from time period

1. The system indicates that the to time period cannot be less than the from time period
2. The system asks the user to enter another to time period.
3. The use case continues at position 4 of the main flow

**Exceptional flow**

E1: No results exist

1. The system indicates that no results have been entered into the App
2. The system advises the user to enter at least one result before selecting the emailing option.
3. The use case ends

**Exceptional flow**

E2: No results exist

1. The system indicates that there is no email client setup on the phone or tablet.
2. The system advises the user to setup an email client before using the email option.
3. The use case ends

**Termination**

User is back at the email section of the App

**Post condition**

The App waits for another user selection

### 2.3.5 Requirement 5: Graphs

Description & Priority

The user can view their results in a graph based format. This will provide a quick at a glance view of how they are performing and any noticeable issues will be clearly visible.

Use Case

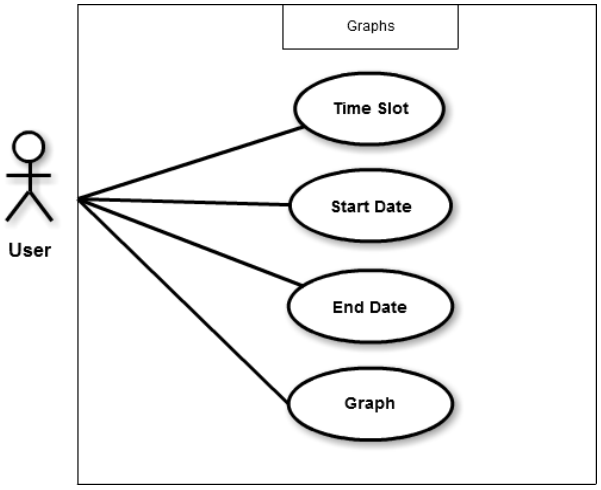
**Scope**

The scope of this use case is to allow the user to view all results for a selected time period and time slot in a graph based format.

**Description**

This use case describes the act of view a user's results in the form of a graph

**Use Case Diagram**




**Flow Description**

**Precondition**

The user has to be in the graph section

**Activation**

This use case starts when the user selects the graph option from the main menu

**Main flow**

1. The user selects the graph section of the App
2. The user selects a time slot or a range of time slots
3. The user selects a start time period
4. The user selects a finish time period (A1 Finish time before start time)
5. The system displays a graph for the results and time periods chosen

**Alternate flow**

A1: Finish Time before Start time.

1. The system indicates that the finish time period is before the start time period
2. The system asks the user to enter another finish time period
3. The use case continues at position 4 of the main flow
4. 4 of the main flow

**Termination**

User is back at the graph section of the App

**Post condition**

The App waits for another user selection

### 2.3.6 Requirement 6: Sync

Description & Priority

The user has the option to sync their stored results with an external database and then view these results on a website

Use Case

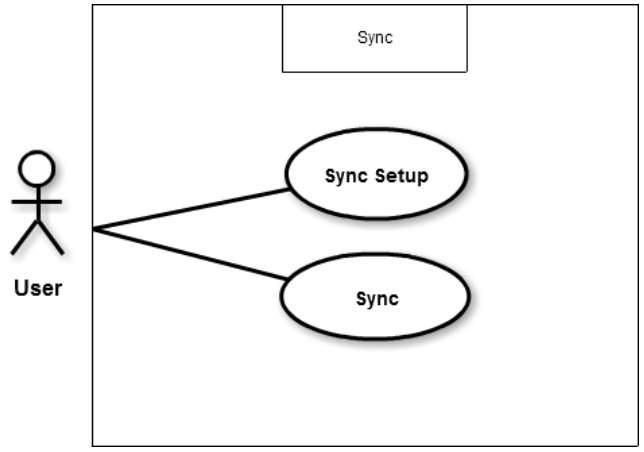
**Scope**

The scope of this use case is to allow the user to sync their results with an external database

**Description**

This use case describes the act of syncing a user's results with an external database

**Use Case Diagram**




**Flow Description**

**Precondition**

The user has to be in the other options menu

**Activation**

This use case starts when the user selects the sync option within the other options menu

**Main flow**

1. The user selects the sync option (E1 no login created)
2. The system connects to the external database (E2 External Resource unavailable)
3. The system uploads any un-synced data (E3 Connection lost)
4. The system indicates sync successful

**Exceptional flow**

E1: No login created

1. The system indicates that there is no login created
2. The system advises the user to run the sync setup
3. The use case ends

**Exceptional flow**

E2: External resource unavailable

1. The system indicates that the external resource is currently unavailable
2. The system advises the user to try again later
3. The use case ends

**Exceptional flow**

E3: Connection lost

1. The system indicates that the connection to the external resource has been lost
2. The system advises the user to try the sync again
3. The use case ends

**Termination**

User is back at the other options screen

**Post condition**

The App waits for another user selection

### 2.3.7 Requirement 7: View Data

Description & Priority

Additionally to the graph view, the user can view all of their previous results in text format. This will be in the form of a data table.

Use Case

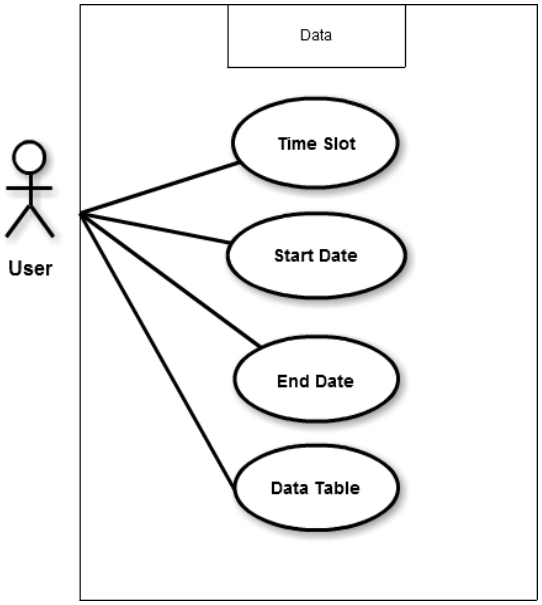
**Scope**

The scope of this use case is to allow the user to view their results in data view to see the actual results behind the graph

**Description**

This use case describes the act viewing the user's results in a data table

**Use Case Diagram**




**Flow Description**

**Precondition**

The user has to be in the graph page

**Activation**

This use case starts when the user selects the view data option from the graph page

**Main flow**

1. The user selects the View Data option (A1 no option selected)
2. The system opens up with options defaulted to the ones selected on the graph view

**Alternate flow**

A1: No data selected

1. The system indicates that no options were selected
2. The system asks the user to make sure all option fields are populated
3. The use case continues at position 1 of the main flow

**Termination**

User is back at the graph page

**Post condition**

The App waits for another user selection

### 2.3.8 Requirement 8: Appointments

Description & Priority

The user can enter any upcoming hospital appointments and receive reminders for upcoming appointments.

Use Case

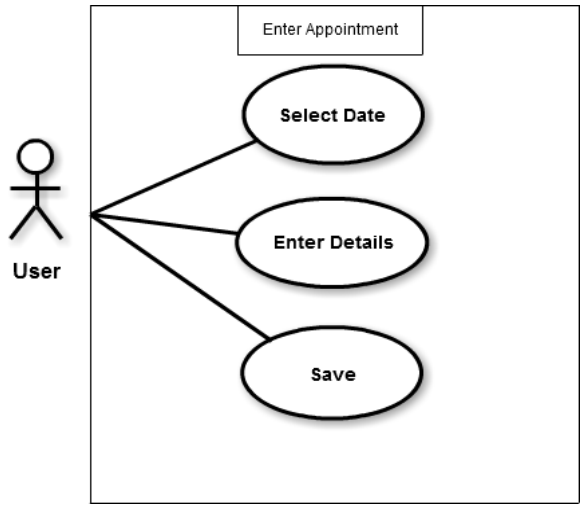
**Scope**

The scope of this use case is to allow the user to enter any hospital appointments that they have

**Description**

This use case describes the act of entering hospital appointments

**Use Case Diagram**




**Flow Description**

**Precondition**

The user has to be in the appointment page

**Activation**

This use case starts when the user selects the new appointment option

**Main flow**

1. The user selects the enter new option
2. The system opens a calendar and the user enters appointment details
3. The user saves the appointment (A1 no appointment entered)

**Alternate flow**

A1: No appointment entered

1. The system indicates that no appointment details were entered
2. The system asks the user to make sure details are entered
3. The use case continues at position two

**Termination**

User is back at the appointments page

**Post condition**

The App waits for another user selection

### 2.3.9 Requirement 9: Login to Website

Description & Priority

The user once they have setup the sync function can sync their results with an external database, and then view these results and charts based on these results on a desktop\laptop.

Use Case

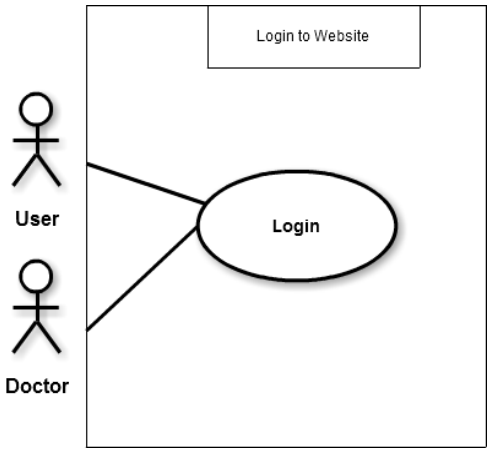
**Scope**

The scope of this use case is to allow the user to login to a website to view their results

**Description**

This use case describes the act of logging in to the website

**Use Case Diagram**




**Flow Description**

**Precondition**

The user has to be on the website login page

**Activation**

This use case starts when the user attempts to log in

**Main flow**

1. The user enters their username that was setup on the App
2. The user enters their password which was setup on the App
3. The user clicks to login (A1 user doesn't exist)
4. The user is logged in to the site

**Alternate flow**

A1: User doesn't exist

1. The system indicates that the username doesn't exist
2. The system asks the user to re-enter a valid username
3. The use case continues at position 1

**Termination**

User is logged in to the website

**Post condition**

The user can view results on the webpage

## *2.4* Implementation

### 2.4.1 Technology Overview

Visual Studio 2015 was the program used to develop the majority of the application and website. The language used was C# with XAML used for the GUI on the App and Razor (cshtml) used for the GUI on the website. The MMVM approach was taken which allowed the development of both the phone and windows App within the one project.

SQLite is used for the local storage in the App, with SQL being used for the online storage, hosted by Microsoft Azure.

### 2.4.2 Technologies

#### 2.4.2.1 Microsoft Visual Studio

Visual Studio 2015 was used as the main software to develop the App. I chose Visual Studio as we had experience from using it throughout the college course and also it could be used to develop both my App and Website but also can easily link in to Microsoft Azure to test my link to the SQL database hosted on Azure.

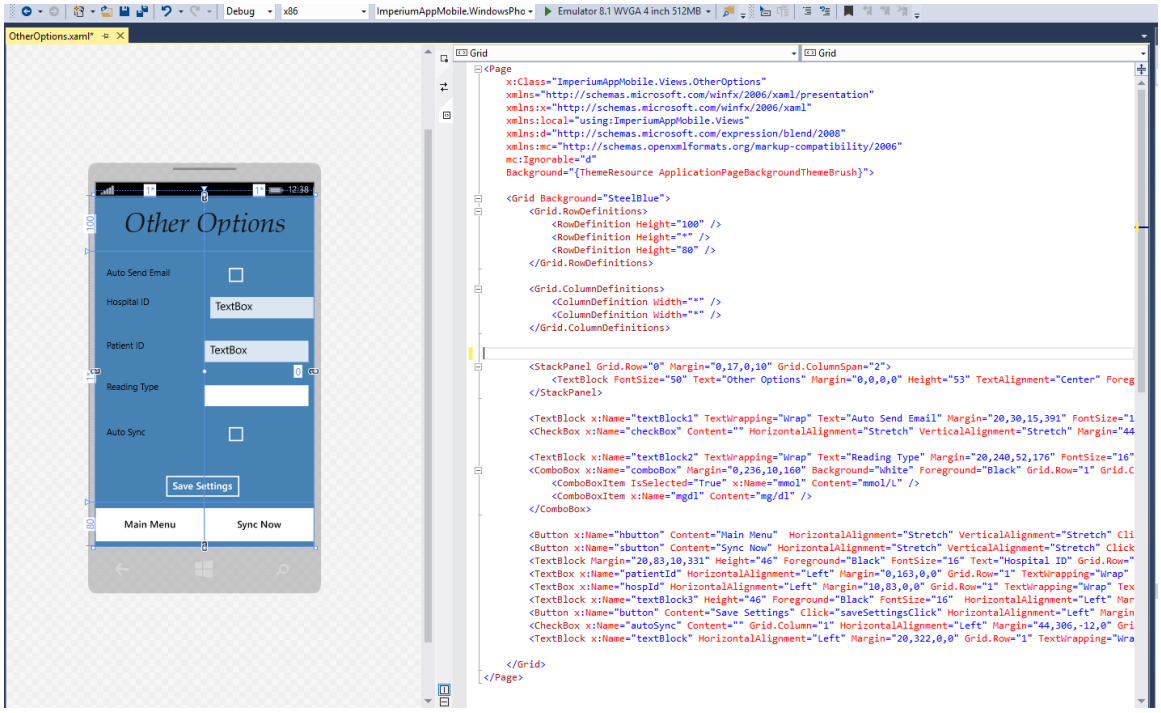
Another reason was for the built in emulator meaning I would not have to connect my phone each time I wanted to test new functionality and instead could run it under the emulator which mimics the windows phone version I was coding for.

#### 2.4.2.2 C#

Windows Apps allow you to code in a number of different languages, I chose to do my App and website in C#. I decided on this based on the fact we had a module in the previous year that was based on C# that I enjoyed and also during my research for the App a lot of Microsoft's own tutorials were coded using C#.

#### 2.4.2.3 XAML

XAML is the language used to develop the GUI sections of Windows Apps. It is an XML-based markup language that was developed by Microsoft. Below is the XAML used for the 'my options' page. It illustrates how the preview pane provides immediate feedback of how your layout will look.




#### 2.4.2.4 SQLite & SQL

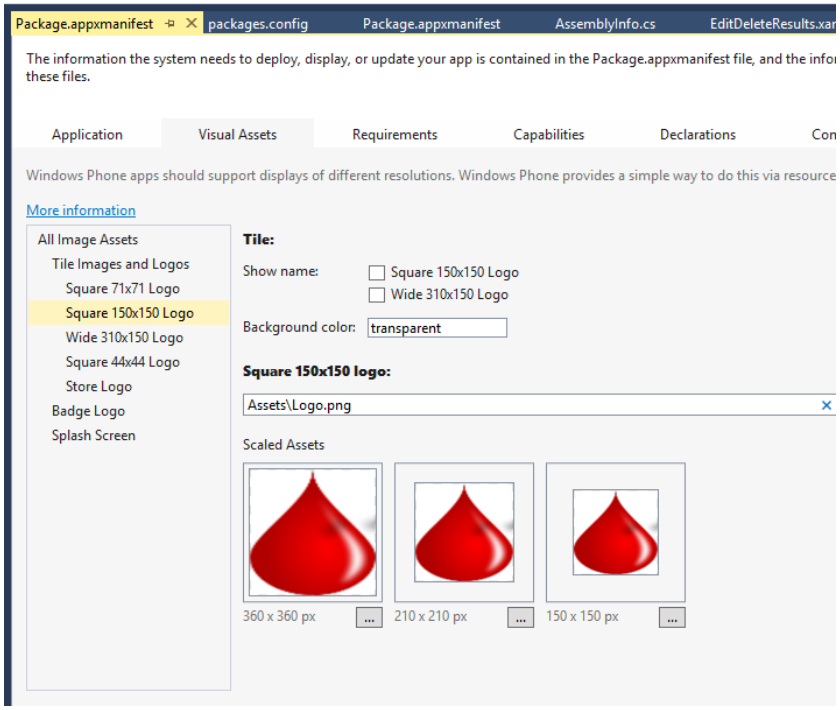
SQLite is used to store the data locally on the device. The SQLite packages are added to the windows and windows phone solutions and then you configure CRUD operations for the local database. For my App the IMobileServiceSyncTable interface is used to perform operations on the database as this uses SyncContext and this keeps the local database and the Azure SQL database in sync.

#### 2.4.2.5 Microsoft Azure App Services

I used Azure App mobile services to synchronise my App's local data with the hosted Azure SQL database. This allowed me to build the backend on the Azure portal and download the code, to begin building the App on top of this backend in Visual Studio.

#### 2.4.2.6 GNU Image Manipulation Program

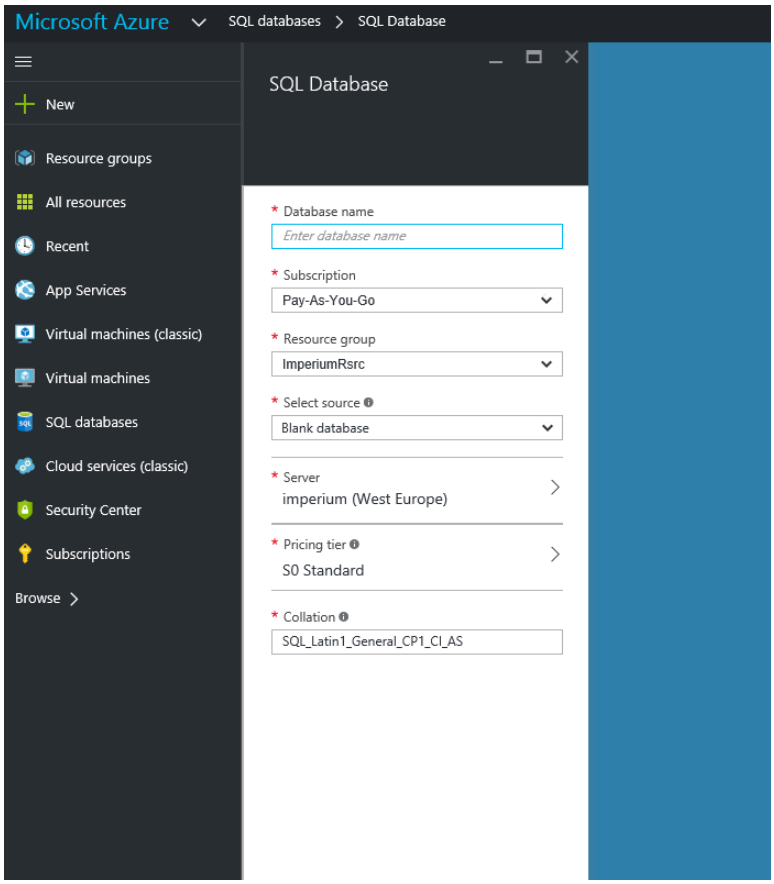
GIMP is an open source image editor that I used to rescale the App logo for the different devices it would be used on. This is then set in the package manifest for the App.



### 2.4.3 Procedures

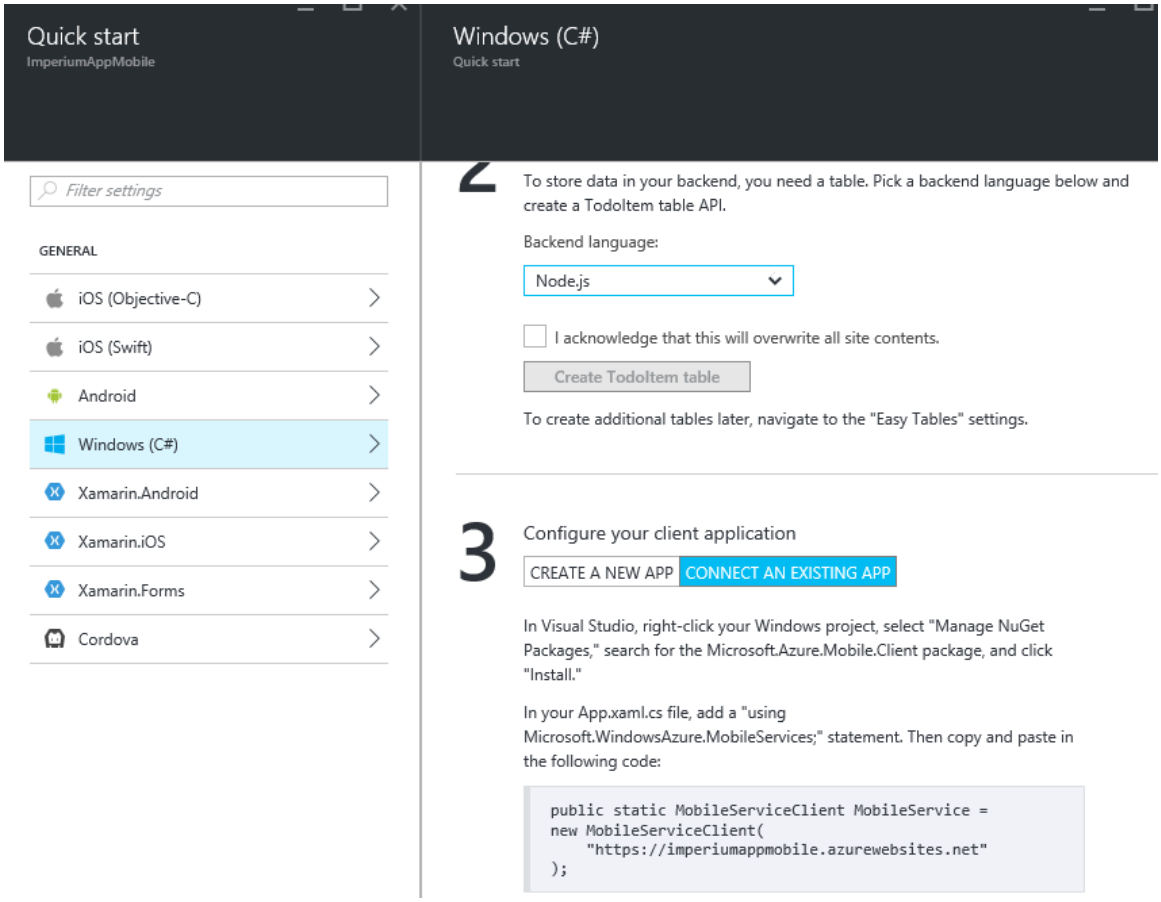
#### 2.4.3.1 Setting up Azure

To enable the synchronization to Azure I first created the SQL database to be hosted on Azure. This is the databased that the local App data is to be synchronised with and also the database that is the websites backend. I logged into the Azure portal and selected SQL database. Then selected add –create and entered the database details.




Once the database is setup, the App service can be created. This service handles the synchronisation of data between the SQLite database on the device the App is running on and the SQL database hosted in the portal. In the portal I selected App services and add. I entered the details and passwords and then assigned the database that was created in the previous step. Once the service was created I selected to download the code, this will

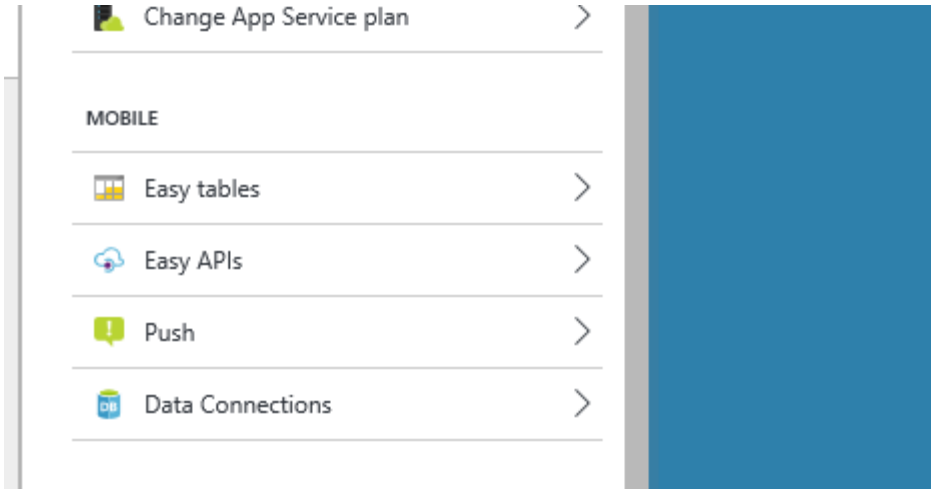
download a Visual Studio project with the App service already setup. As the model used in the download is the same as the MVVM model I wanted to use in Visual Studio, I selected the create a new App option.




Once the App service was up and running I had to add and configure the tables needed for the Azure SQL database. This can be done from the Azure portal or from Visual Studio. I found the easiest way was to configure this direct from the Azure portal itself.

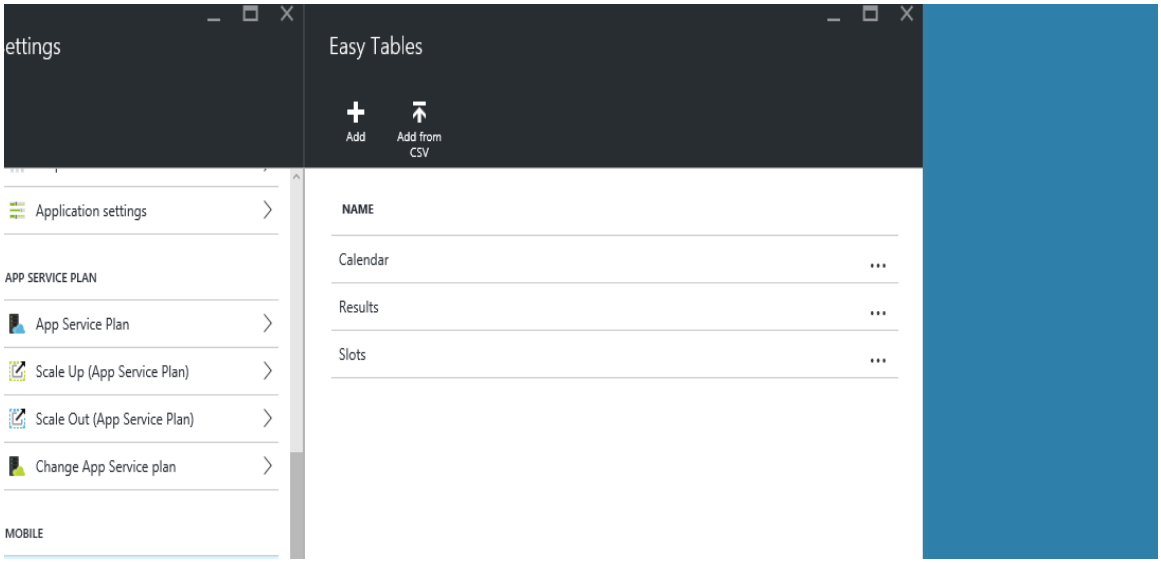
To setup the SQL tables, from within the App service I went to

Mobile – Easy Tables.



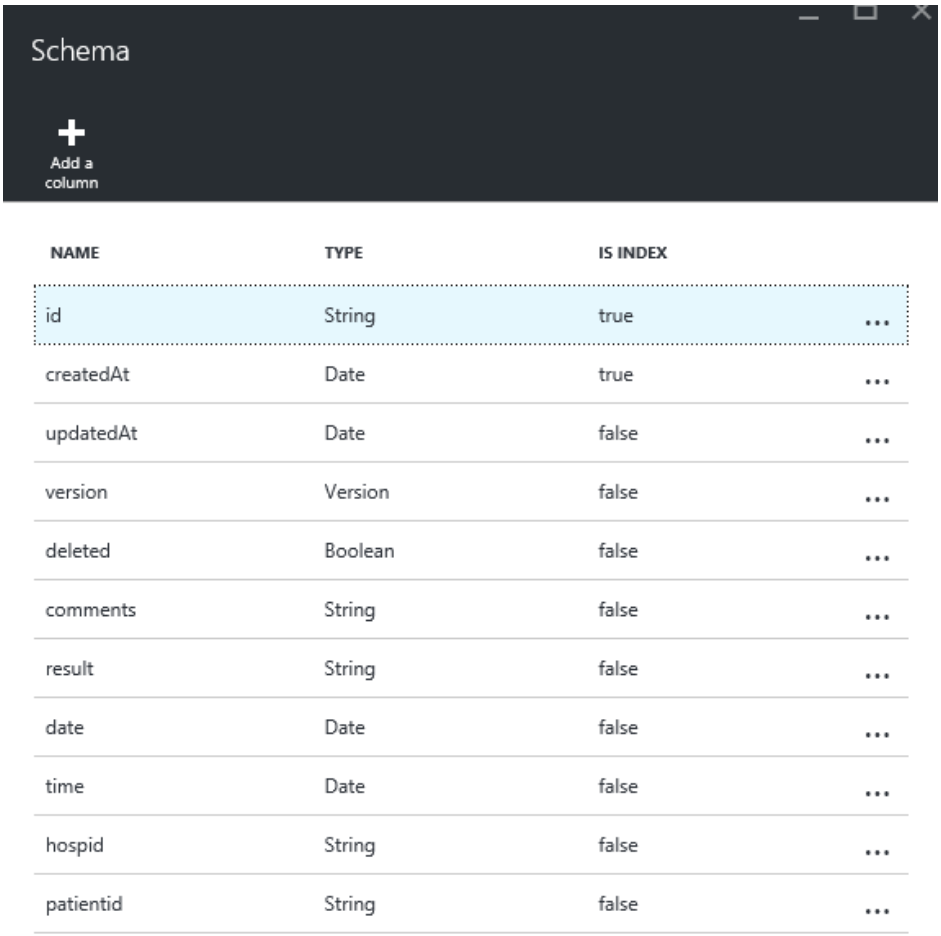

From here I added the tables and then set the schema for the table. This should match the setup in your table classes in the Visual Studio project.

For this App I am using three tables that will be synced between the device and Azure SQL

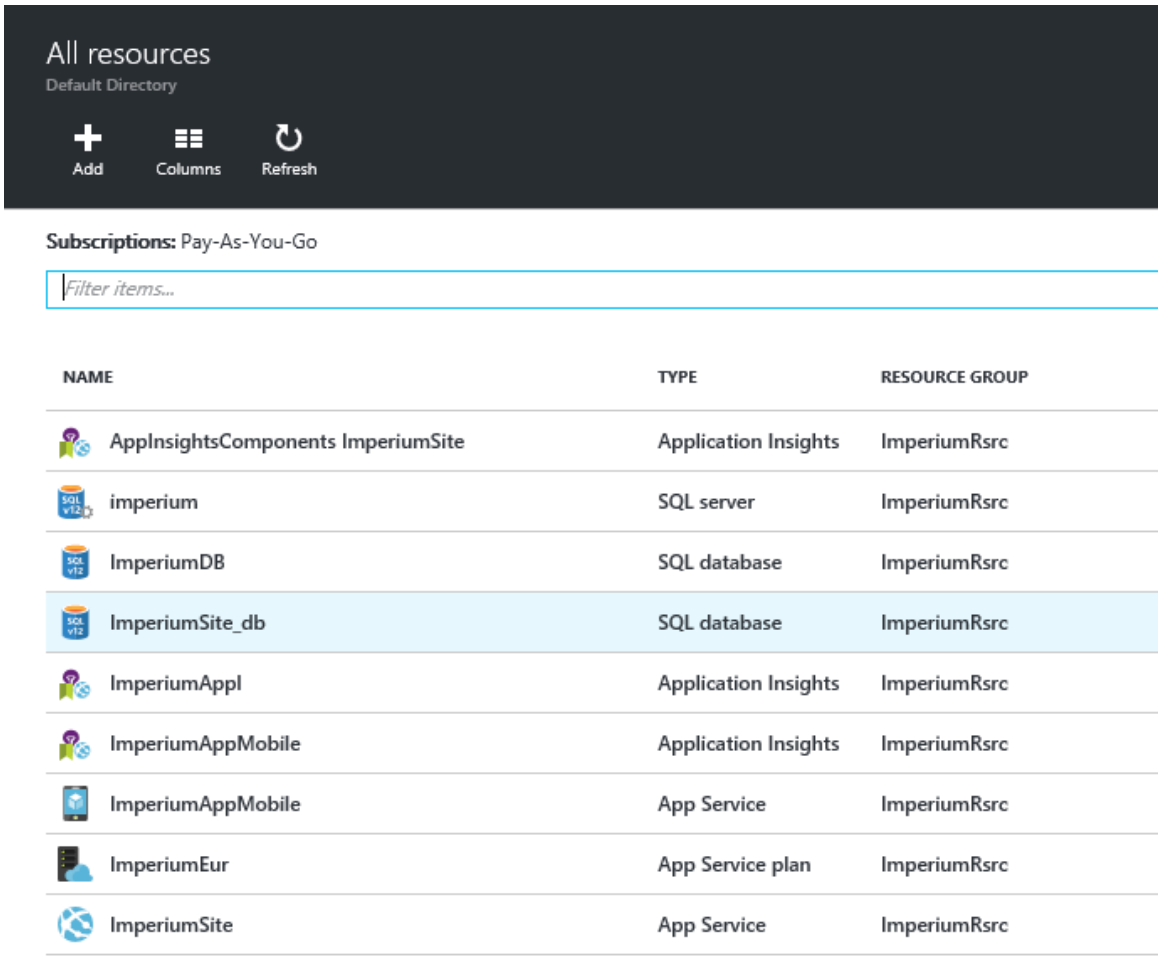



And then once the table was setup, the schema is configured. Below is the schema for the results table. There is some additional Azure fields here like createdAt and updatedAt which are populated automatically on data entry or modification, so these fields do not have to exist in the database classes within the App. I had originally set up my id as int on

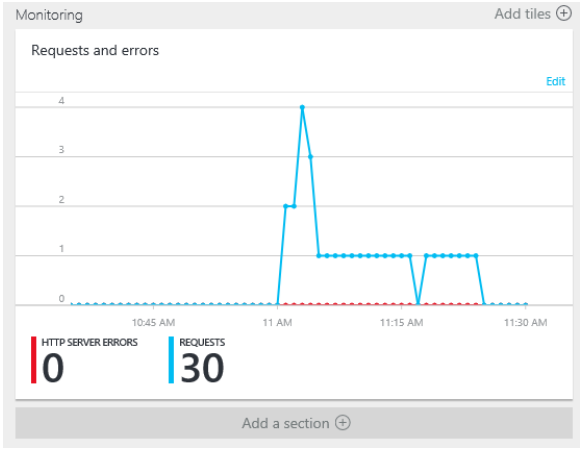
my Visual Studio project but changed to string as this is what SQL Azure uses by default. I thought it best to keep them in line and there is no specific reason for my id to be set as an int within the project.




When everything was setup in the Azure portal all the services currently running where viewable in the portal. There are some that are setup automatically as others are setup, such as the SQL server which is initialized when the SQL database is created



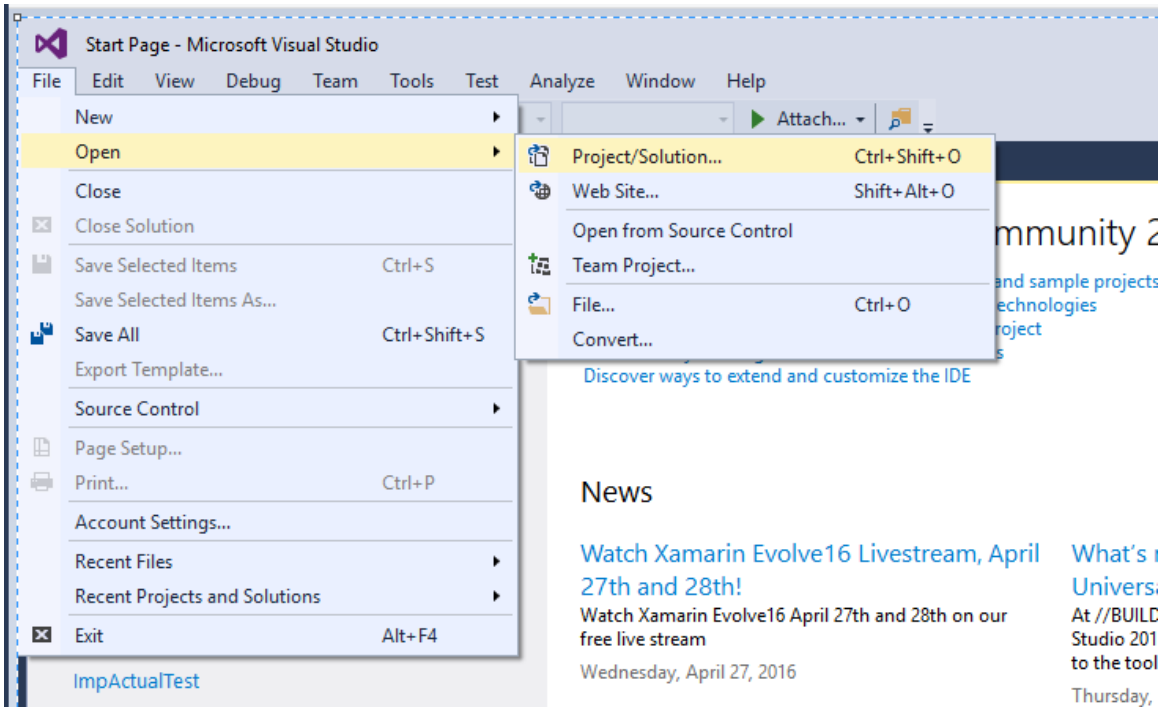

You can also see the traffic used against these services and any request errors that have occurred.



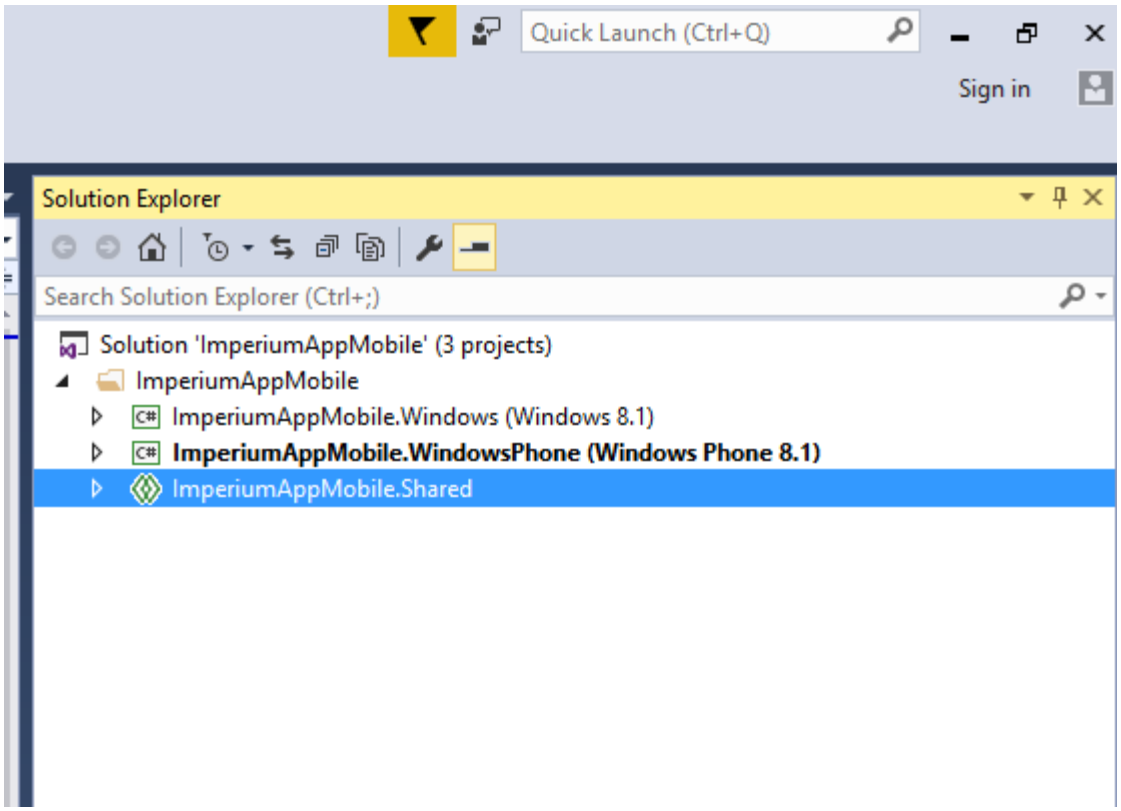
#### 2.4.3.2 Project Creation

The App itself is fully created in Visual Studio. To start I downloaded the Visual Studio project from the Azure portal and started with this. This created a MVVM style project with the backend I created on the Azure portal already setup. I then built the App itself on top of this and allowed for local storage on the device as well.

Start Visual Studio and go to file – open – and select the downloaded project from the portal.




The below screenshot shows my project split between windows\windows phone and shared sections




The link to the Azure App service is in the app.xaml.cs page within the share folder but other than this it is a blank project with the MVVM style implemented into above three staging areas.

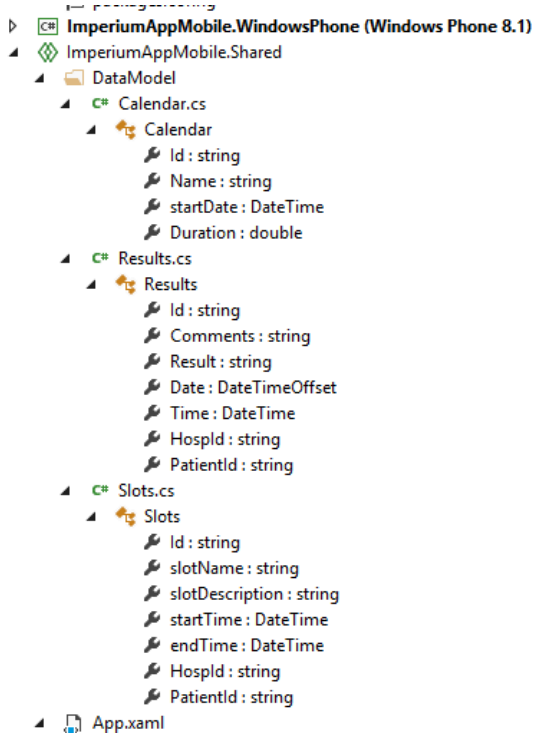
Link to App service

App.xaml.cs

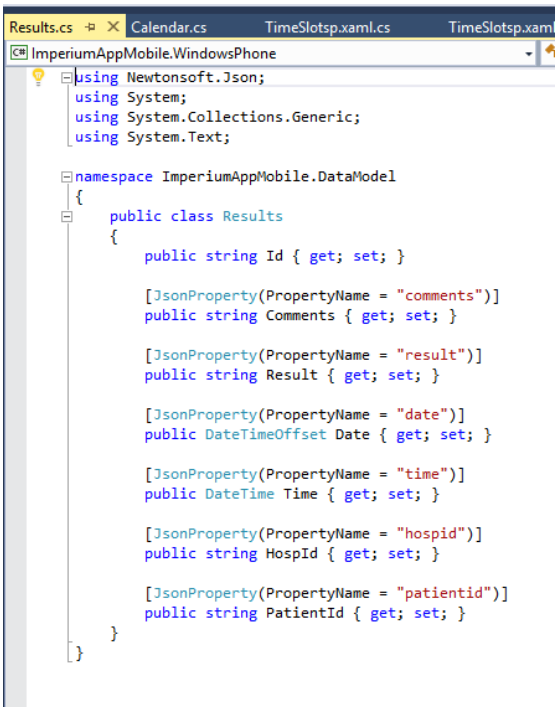
ImperiumAppMobile.Windows | ImperiumAppMobile.App

```
using Microsoft.WindowsAzure.MobileServices;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.ApplicationModel;
using Windows.ApplicationModel.Activation;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;

namespace ImperiumAppMobile
{
    /// <summary>
    /// Provides application-specific behavior to supplement the default Application class.
    /// </summary>
    sealed partial class App : Application
    {

        // This MobileServiceClient has been configured to communicate with the Azure Mobile App.
        // You're all set to start working with your Mobile App!
        public static MobileServiceClient MobileService = new MobileServiceClient(
            "https://imperiumappmobile.azurewebsites.net"
        );
```

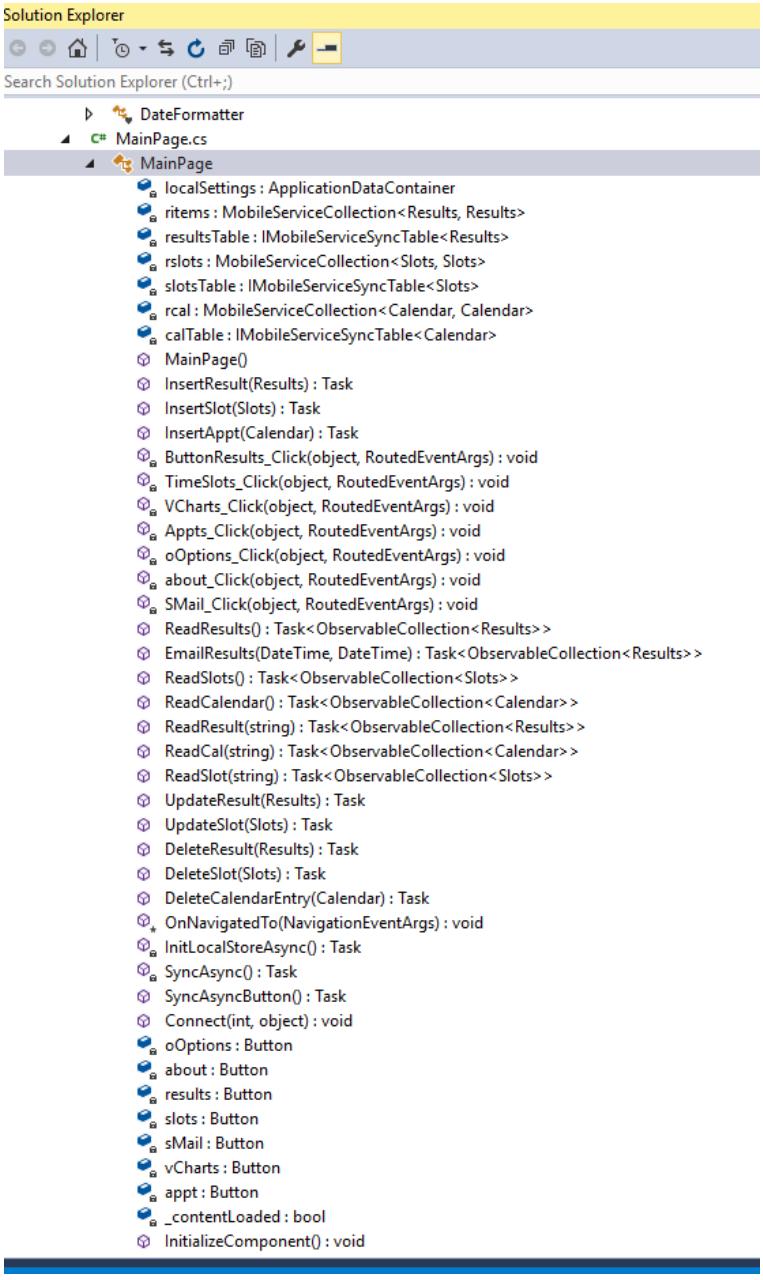
I then created a DataModel Folder in the shared section. This will be were my database classes are stored.  Each field should match the same datatype as the SQL Azure database.




This is the results.cs file



That is the database class's setup in the datamodel, which represents the model portion of the MVVM model. The next stage was to setup the viewmodel portion of the MVVM approach. The downloaded project from Azure already had the view model file created called MainPage.cs so this file was added to. Below is all the functions I added to this file to cater for all the different functionality of the App. This vewmodel acts as the middle between the data model and the GUI (view). When the user clicks an item on screen to read some data the request is handled through the mainpage.cs file.



This of course was completed piece by piece as each XAML (view) page was added and functionality was required for it but the above picture is of the final mainpage.cs file. The file includes the following functions:

- Function to insert a new result
- Function to insert a new slot
- Function to insert a new appointment
- Function to handle button press fro mmain menu to open the page requested
- Function to read a single result from the database that accepts a recordId passed by the request
- Function to read a single slot from the database that accepts a recordId passed by the request
- Function to read all results from the database and sort by date and time
- Function to read all slots fro mthe database and sort by time
- Function to read all calendar entries created by the App
- Function to read all results for a specific time period with the date parameters passed by the requester, this is used when emailing results and creating charts
- Function to update specific result
- Function to update specific slot
- Function to delete specific result
- Function to delete specific slot
- Function to delete specific appointment
- Function to initilaize a sync to Azure mobile services

Below are samples of some of these functions. The first is the handler for what happens when the Send Mail button is clicked from the main menu. The App directs to the SendEmail XAML page. The next three sections are requests to the database to get results and slots data and returns them to the requester in an observable collection. The middle one is used for emailing results so takes in start and end time parameters to filter the data to this time period. Each request for data is also filtered to the specific patient id so they only see there data. Patient id is saved in the Apps local settings so is accessible from any class.

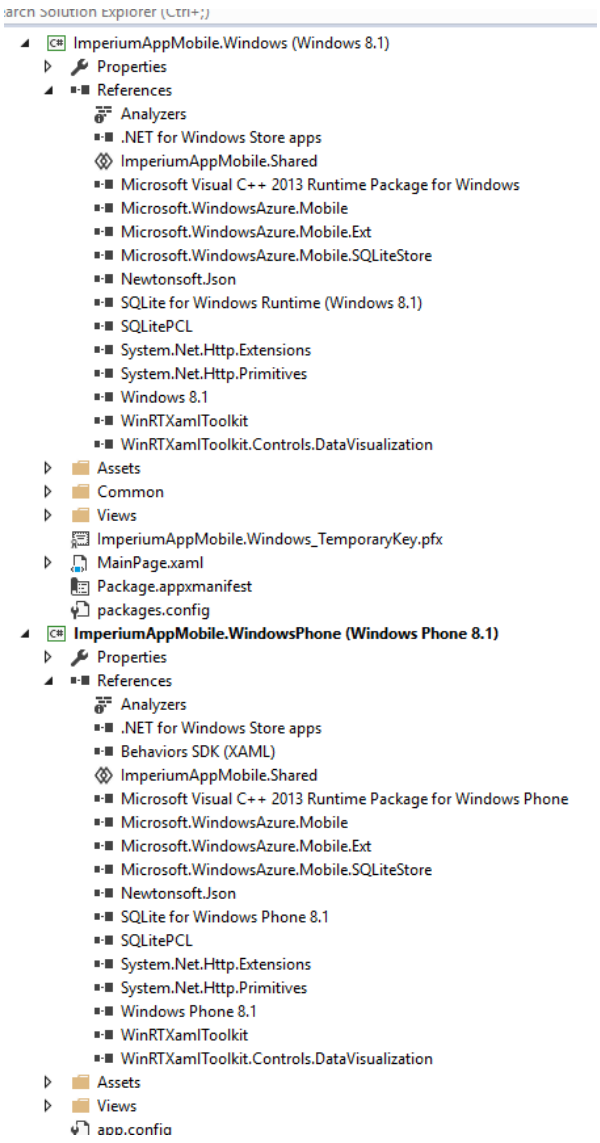
```
private void SMail_Click(object sender, RoutedEventArgs e)
    {
        //Navigate to SendEmail page when button is clicked
        Frame.Navigate(typeof(SendEmail));
    }
```

```
        public async Task<ObservableCollection<Results>> ReadResults()
        {
            //gets a collection of results were patient id passed from the user filters the reults
to there results only
            ritems = await resultsTable.Where(ritems => ritems.PatientId ==
localSettings.Values["patientId"].ToString()).ToCollectionAsync();

           await SyncAsync();
            ObservableCollection<Results> ResultsList = new
ObservableCollection<Results>(ritems);
            //returns the list of results
            return ResultsList;
        }




        public async Task<ObservableCollection<Results>> EmailResults(DateTime
startDate, DateTime endDate)
        {
            //gets a collection of results for the dates entered and passed to the request by
the user
            ritems = await resultsTable.Where(ritems => ritems.PatientId ==
localSettings.Values["patientId"].ToString() && ritems.Date >= startDate && ritems.Date
<= endDate).OrderByDescending(i => i.Date).ThenBy(i => i.Time).ToCollectionAsync();

            await SyncAsync();
            ObservableCollection<Results> ResultsList = new
ObservableCollection<Results>(ritems);
            return ResultsList;
        }



        public async Task<ObservableCollection<Slots>> ReadSlots()
        {

            rslots = await slotsTable.Where(ritems => ritems.PatientId ==
localSettings.Values["patientId"].ToString()).ToCollectionAsync();
            await SyncAsync();
            ObservableCollection<Slots> SlotsList = new
ObservableCollection<Slots>(rslots);
            return SlotsList;


        }
```

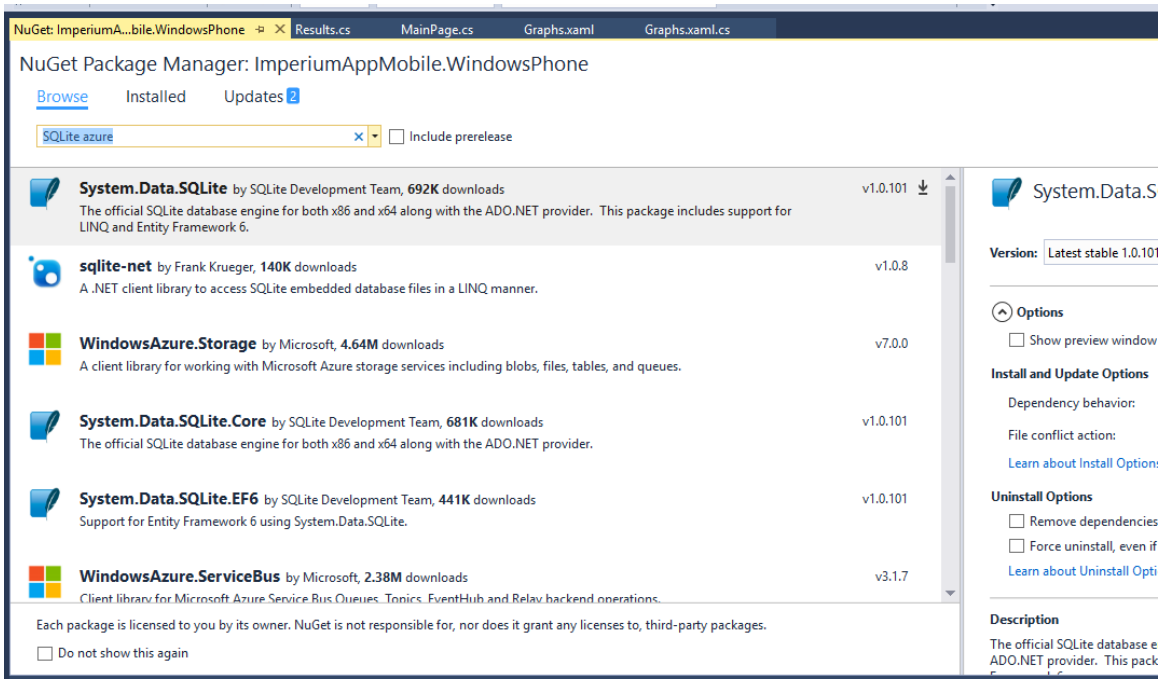
After this the shared section is pretty much finished.

A number of packages had to be added to each solution such as SQLite to cater for the local storage on the device the App is running on and Azure and Json packages to handle the synchronization to Azure SQL database through Azure mobile App services. There are also two packages for WinRTXamlkit which enables the use of libraries for displaying charts in the App.

Notice how the libraries have to be added to both the Windows project and the Windows phone project. This separation is done for two reasons; 1) to make the package size smaller as users may only be creating for one style of project and not require both, and 2) in some instances the packages are different to cater for different approaches to the phone and windows models.

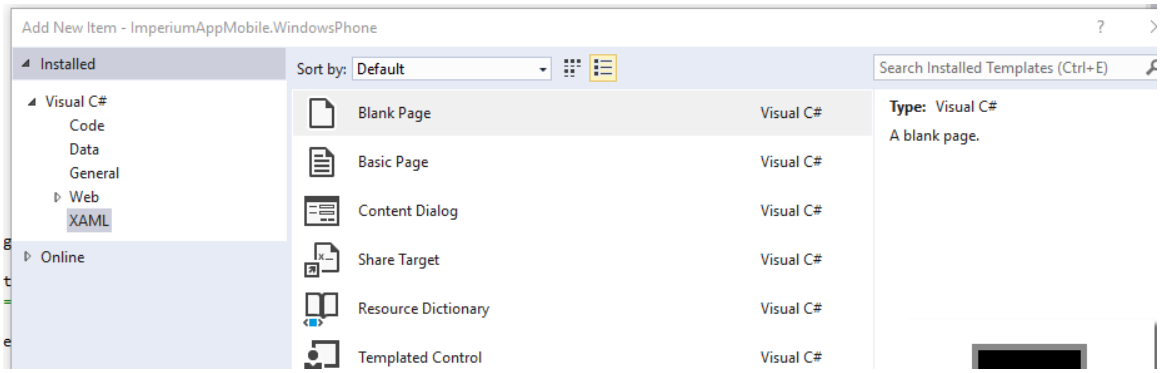


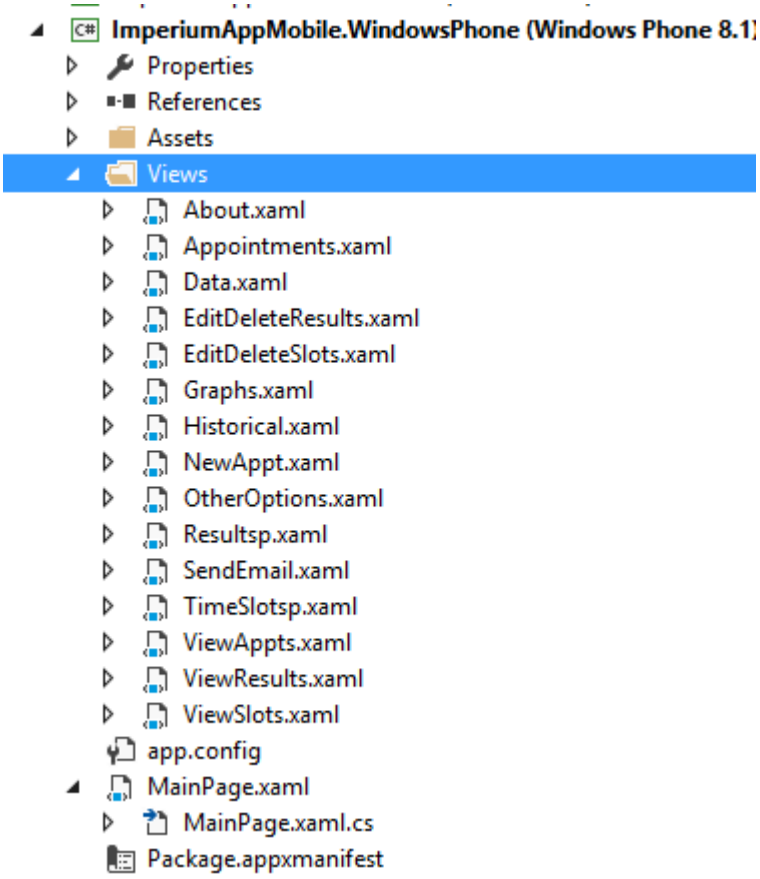
Packages are installed through the NuGet Package manager.




With the model and view model setup and all the packages I needed installed, I then created the views. This part like the package installs has to be done in both the windows section and the windows phone section. This is the advantage of using the MVVM approach as the model and view model are shared between both projects so the only separation is for the views (XAML) pages. This allows for the creation of different screen layout for the smaller phone screens in the windows phone section and the larger tablet\laptop screens in the windows section.

Next I went in to each specific section and created the views. Every page in the App needs to have its own view. Below are the views I created for the phone version. To create a view right click the views folder and select Add – New Item – XAML – Blank Page



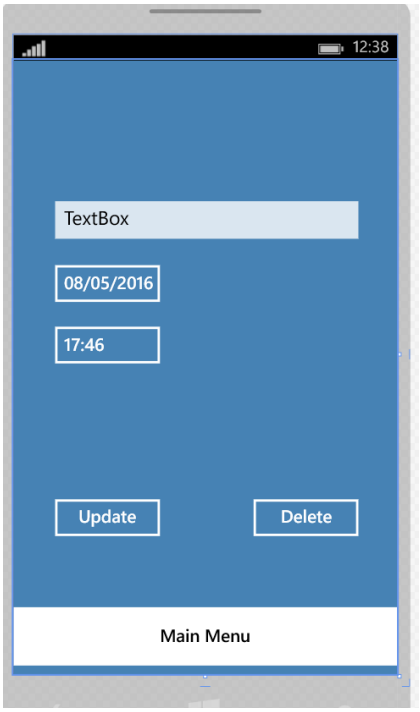




To give an example I will focus on the editdeleteresults.xaml page. The XAML code is

```
<Page
   x:Class="ImperiumAppMobile.Views.EditDeleteResults"
```

```
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:ImperiumAppMobile.Views"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid Background="SteelBlue">
        <Grid.RowDefinitions>
            <RowDefinition Height="100" />
            <RowDefinition Height="*" />
            <RowDefinition Height="80" />
        </Grid.RowDefinitions>
        <Button x:Name="homeButton" Content="Main Menu"
HorizontalAlignment="Stretch" VerticalAlignment="Stretch" Margin="0,0,0,0"
Grid.Row="2" Click="homeButtonClick" Background="White" Foreground="Black"/>
        <TextBox x:Name="resultTextBox" HorizontalAlignment="Left" Margin="43,47,0,0"
Grid.Row="1" TextWrapping="Wrap" Text="TextBox" VerticalAlignment="Top"
Width="316"/>
        <TimePicker x:Name="resultTimePicker" HorizontalAlignment="Left"
Margin="43,168,0,0" Grid.Row="1" VerticalAlignment="Top"/>
        <DatePicker x:Name="resultDatePicker" HorizontalAlignment="Left"
Margin="43,104,0,0" Grid.Row="1" VerticalAlignment="Top"/>
        <Button x:Name="updateButton" Content="Update" HorizontalAlignment="Left"
Margin="43,348,0,0" Grid.Row="1" VerticalAlignment="Top" Click="updateButtonClick"/>
        <Button x:Name="deleteButton" Content="Delete" HorizontalAlignment="Left"
Margin="250,348,0,0" Grid.Row="1" VerticalAlignment="Top"
Click="deleteButtonClick"/>

    </Grid>
</Page>
```

Which looks like




Then in the code behind the XAML page there are a number of calls to the functions created in the helper file that are called depending on the actions of the user.

The code for this particular page is:

```
using ImperiumAppMobile.DataModel;
using System;
using System.Collections.ObjectModel;
using System.Linq;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Navigation;

namespace ImperiumAppMobile.Views
{

    public sealed partial class EditDeleteResults : Page
    {
        //create object so local settings saved for the app can be accessed
        Windows.Storage.ApplicationDataContainer localSettings =
Windows.Storage.ApplicationData.Current.LocalSettings;
```

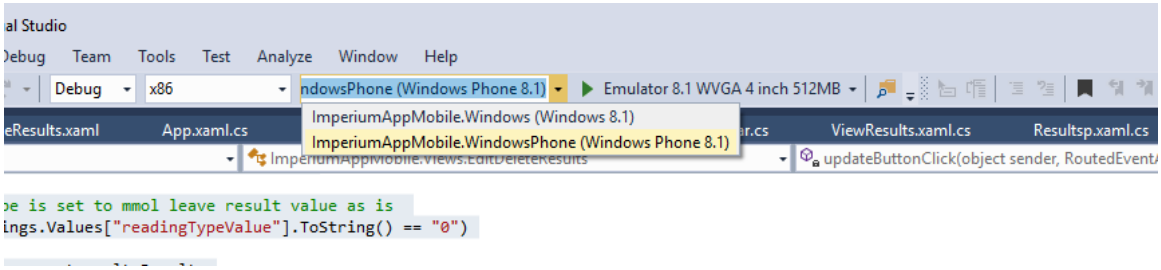
```
        //create collection of results class
        ObservableCollection<Results> DB_ResultList = new
ObservableCollection<Results>();

        //create and set string to be used to store specific result id to be passed to the
database for querying specific result
        string Selected_ResultId = "0";

        MainPage Db_Helper = new MainPage();

        Results currentresult = new Results();

        public EditDeleteResults()
        {
            this.InitializeComponent();
        }

        protected async override void OnNavigatedTo(NavigationEventArgs e)
        {
            string cResult = " ";
            Selected_ResultId = (e.Parameter.ToString());
            DB_ResultList = await Db_Helper.ReadResult(Selected_ResultId);
            //currentresult = await Db_Helper.ReadResult(Selected_ResultId);
            currentresult = DB_ResultList.FirstOrDefault();
            //List<Results> fred = DB_ResultList.ToList();

            //If value type is set to mmol leave result value as is
            if (localSettings.Values["readingTypeValue"].ToString() == "0")
            {
                cResult = currentresult.Result;
            }
            else
            {
                //if value type is not set to mmol the change result value to be mg\dl
                cResult = (double.Parse(currentresult.Result) * 38.67).ToString("0.0");
            }
            resultTextBox.Text = cResult;
            resultDatePicker.Date = currentresult.Date;
             resultTimePicker.Time = currentresult.Time.TimeOfDay;


        }

        private void homeButtonClick(object sender, RoutedEventArgs e)
        {
            Frame.Navigate(typeof(MainPage));
        }

        private async void updateButtonClick(object sender, RoutedEventArgs e)
        {
```

```
            currentresult.Result = resultTextBox.Text;
            await Db_Helper.UpdateResult(currentresult);
            Frame.Navigate(typeof(ViewResults));
        }

        private async void deleteButtonClick(object sender, RoutedEventArgs e)
        {
            await Db_Helper.DeleteResult(currentresult);
            Frame.Navigate(typeof(ViewResults));
        }
    }
}
```

Once all the XAML pages were complete the same steps were repeated for the Windows section of the project. When the files are being created for the Window store, one is created for the phone and one for windows. Both include the shared section in their build.

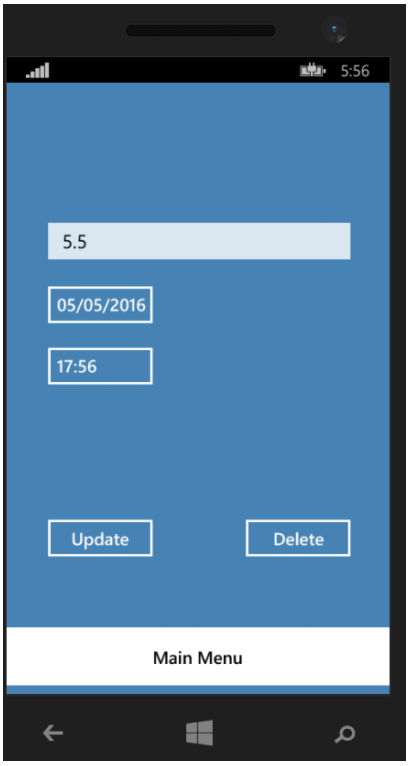
Once pages were complete I ran the emulator from Visual Studio to get immediate feedback of any issues or confirming that it was functioning correctly.

Both the Windows and Windows phone version can be tested by selecting one in the menu illustrated below

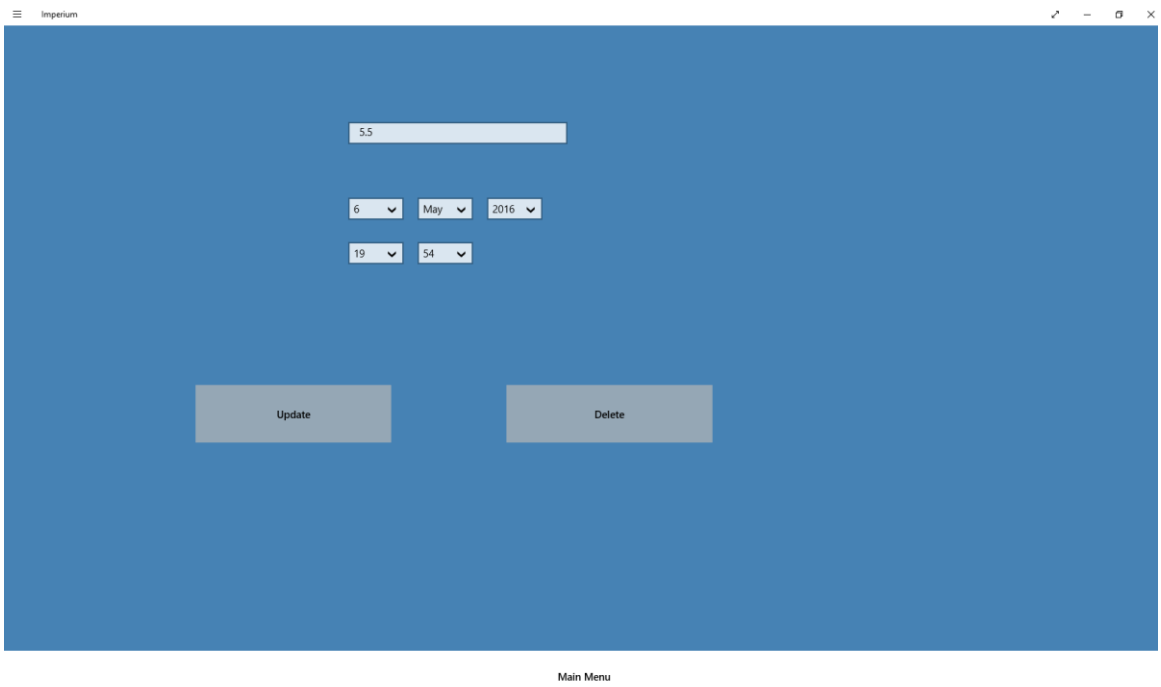



Then the emaultor will launch for the desired option as follows.

Phone:


5:56
5.5
05/05/2016
17:56
Update
Delete
Main Menu

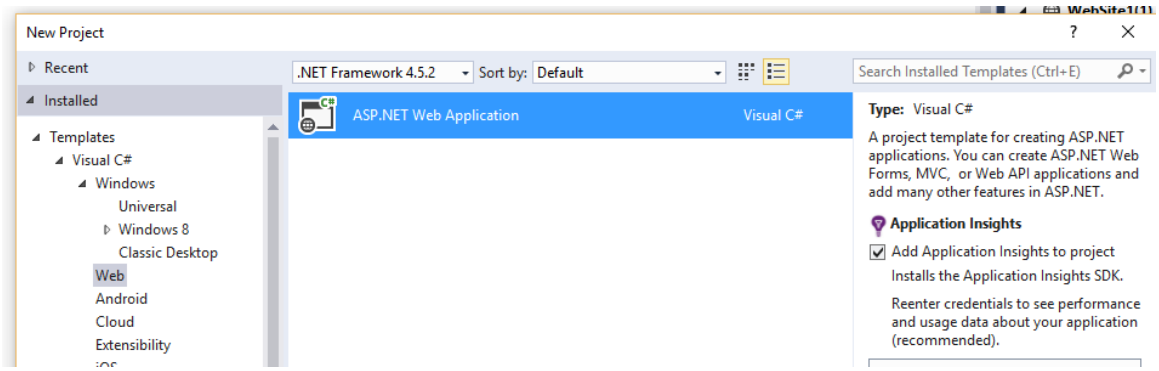
Windows:


Imperium
5.5
6
May
2016
19
54
Update
Delete
Main Menu

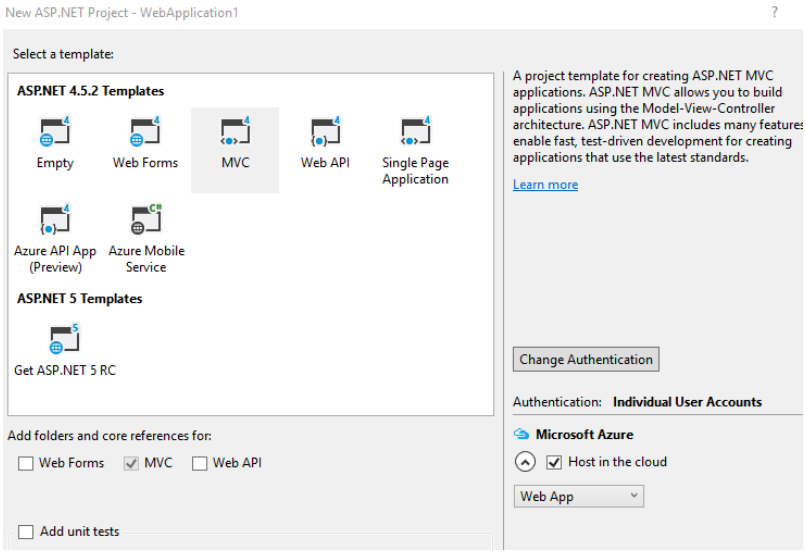
#### 2.4.3.3 Website

The website was created in Visual Studio and then published to Azure. The SQL database in Azure was used as the backend so when the data is synced from the phone it is available for the doctor or healthcare professional on the website.

To create the website I opened Visual Studio and went to file – new – project – and selected web – ASP.NET Web application




Then I selected MVC template with authentication set as individual user accounts.



Then I needed to connect to the same Azure SQL database that my App was syncing with. To do this I created a new folder called infrastructure and then right clicked on the folder add – new scaffold item. I selected MVC 5 controller with views, using entity framework. I then added the connection details to the Azure SQL database.

Once the database connection was setup I then added to the existing controller called home controller and set the view for when the result link is pressed.

```
public ActionResult Display_Results_Data(string searchString)
{


ViewBag.Message = "Results_Data";

    if (!String.IsNullOrEmpty(searchString))
    {
        return View(db.Results.Where(s =>s.patientid.Contains(searchString)).OrderBy(d => d.patientid).ThenBy(d => d.date).ToList());
    }
    else {

        return View(db.Results.OrderBy(d => d.patientid).ThenBy(d => d.date).ToList());

    }
```

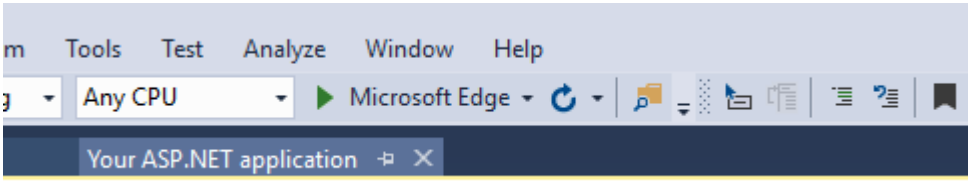
The same was done for the chart view. Once the controller was setup I then created a view for the results page and added a table to list the results. I also added a search box to make it easy to filter on a patient id.

```
<h2>Display_Results_Data</h2>

<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.date)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.time)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.result1)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.comments)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.hospid)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.patientid)
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>

                @item.date.Value.ToString("MMM dd yyy")
            </td>
            <td>
                @item.time.Value.ToString("HH:mm")
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.result1)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.comments)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.hospid)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.patientid)

        </tr>
```

Again similar steps were followed for the charts section.

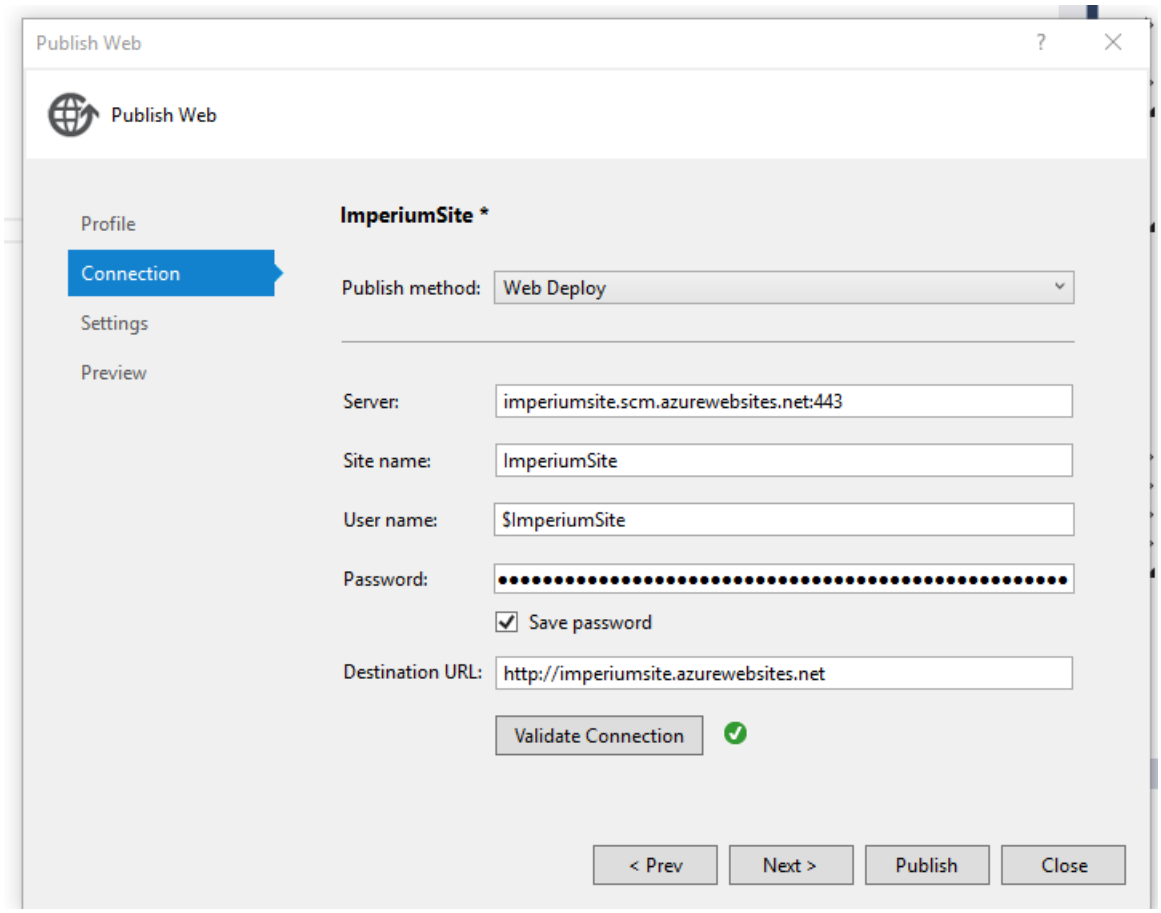
Once the website build was finished in Visual Studio I was able to test by running locally.

To start testing click on the green arrow

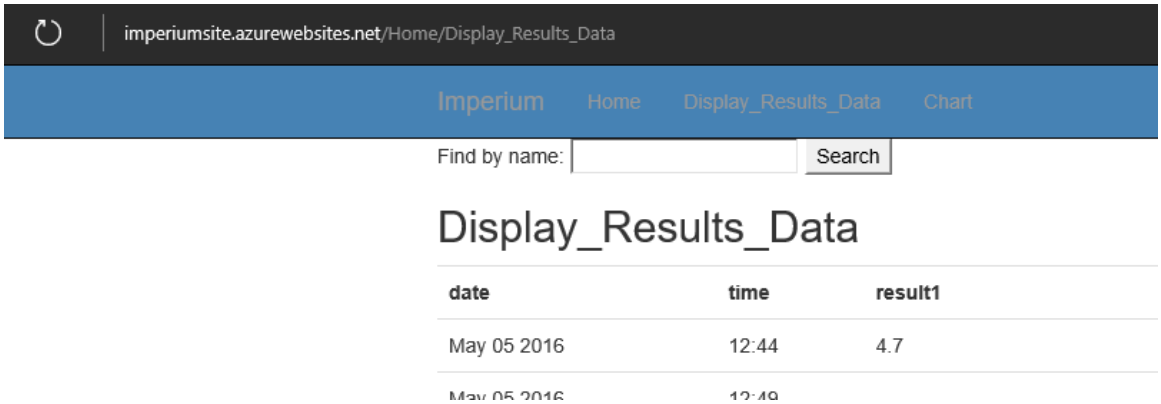


Results page running on local host


When I was happy with the site from within the Visual Studio project I went to build menu and chose publish




The website was then available online hosted in Azure.




2.4.3.5 Code Snippets

To store local user settings such as hospital Id and patient Id

```
Windows.Storage.ApplicationDataContainer localSettings =
Windows.Storage.ApplicationData.Current.LocalSettings;
```

Then the settings are saved to this location

```
 hospId.Text = localSettings.Values["hospId"].ToString();
patientId.Text = localSettings.Values["patientId"].ToString();
```

To enter a new result into the database

Behind the results XAML page create a result object and store the users entered values in it plus any local settings save in the location in above code snippet.

```
 var results = new Results { Comments = commentsBox.Text, Result = cResult, HospId = hospId, PatientId = patientId, Date = datePick.Date, Time = fredd };
```

Then call the routine saved in the shared location (MainPage.cs) and pass the results object to it

```
await Db_Helper.InsertResult(results);
```

The insert routine in the shared folder

```
public async Task InsertResult(Results results)
    {
        // This code inserts a new result into the database.
        await resultsTable.InsertAsync(results);
        ritems.Add(results);

        //cater for sync to sql database on Azure
        await SyncAsync();
    }
```

There is a similar routine for slots and appointments.

Routine for returning results from the database filtered to the user's hospital and patient id

```
public async Task<ObservableCollection<Results>> ReadResults()
    {
        //gets a collection of results were patient id passed from the user filters the reults to there results only
        ritems = await resultsTable.Where(ritems => ritems.PatientId == localSettings.Values["patientId"].ToString()).ToCollectionAsync();

       await SyncAsync();
        ObservableCollection<Results> ResultsList = new ObservableCollection<Results>(ritems);
        //returns the list of results
        return ResultsList;
    }
```

Then in the code behind the view results XAML page these results are assigned to a list box and displayed in order by date and time

```
MainPage dbresults = new MainPage();

        if (localSettings.Values.ContainsKey("hospId") && localSettings.Values.ContainsKey("patientId"))
        {
            DB_ResultList = await dbresults.ReadResults();//Get all DB contacts
```

```
            listBoxobj.ItemsSource = DB_ResultList.OrderByDescending(i =>
i.Date).ThenBy(i => i.Time).ToList();

        }
```

To return a single result

```
DB_ResultList = await Db_Helper.ReadResult(Selected_ResultId);

currentresult = DB_ResultList.FirstOrDefault();
```

Then in MainPage.cs

```
public async Task<ObservableCollection<Results>> ReadResult(string resultid)
    {
        var query = await resultsTable.Where(Results => Results.Id ==
resultid).ToCollectionAsync();
        await SyncAsync();
        //ListItems.ItemsSource = query;
        return query;
    }
```

To sync with Azure, a check is done in the user's local settings to see if they have auto sync enabled

```
 if (localSettings.Values["autoSend"].ToString() == "True")
            {
                await App.MobileService.SyncContext.PushAsync();
                await resultsTable.PullAsync("results", resultsTable.CreateQuery());
                await slotsTable.PullAsync("slots", slotsTable.CreateQuery());
                await calTable.PullAsync("calendar", calTable.CreateQuery());
            }
```

## *2.5 Testing*

The built in emulator with Visual Studio is being used at each development stage of the App to provide real-time feedback of how the App performs on both a phone and a tablet of various screen sizes.

### 2.5.1 Developer Testing

Initial testing was carried out by myself. I outlined all the functionality of the App and documented the steps needed to complete each one. A value of pass or fail was assigned to each one.

Examples of issues found during testing:

Issue – Date formatting

Description – On the results page when viewing the results list, the value for the date also included a time value of 00:00

Cause – This was due to SQL on SQL storing all date values as DateTime so the local SQLite database also had to be set to date time.

Fix – A formatting class was created and the XAML page that displayed the results was pointed to this formatting class. The dates in the XAML code could then be set to the correct format by adding the code Text="{Binding Date, Converter={StaticResource ResourceKey=FormatConverter}, ConverterParameter=\{0:dd-MMMM-yy\}}"

Issue – Extra results showing in list view

Description – Upon saving and loading some results there were some results for another test showing

Cause – There was an error in code in that the filter in the database query was added after the results are stored in the collection meaning that unwanted results were not being filtered out

Fix – Moved the filter to earlier in the line of code

```
ritems = await resultsTable.Where(ritems => ritems.PatientId ==
localSettings.Values["patientId"].ToString()).ToCollectionAsync();
```

Issue – Time slots end time

Description – App was accepting an end time earlier than the start time

Cause – No check in place to make sure start time is before end time

Fix- Added a check to the save button

```
if (startTime.Time < endTime.Time)
```

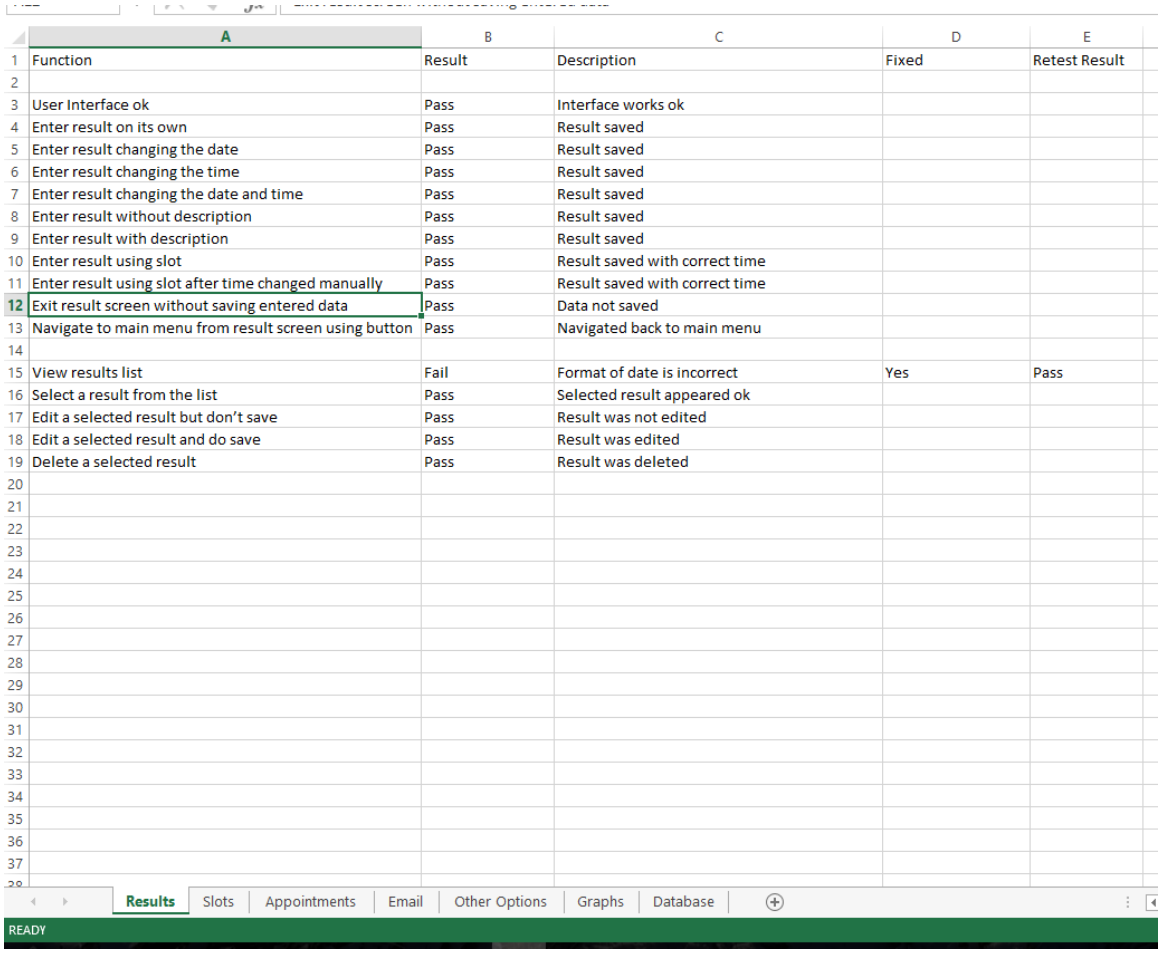
Issue – Results view list text was overlapping

Description – The result and date were overlapping in the result view

Cause – Spacing in the XAML layout wasn’t enough to cater for both pieces of text

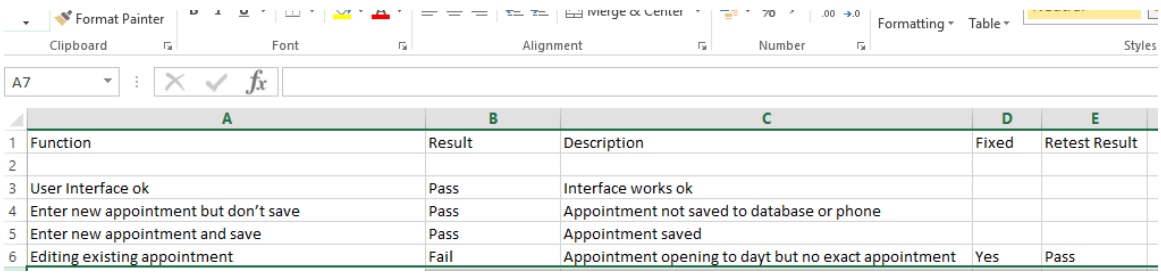
Fix – Added a second grid column to the XAML layout and placed the result text in here

Some screenshots of my own sheets for developer testing:



| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Function | Result | Description | Fixed | Retest Result |
| 2 | | | | | |
| 3 | User Interface ok | Pass | Interface works ok | | |
| 4 | Enter result on its own | Pass | Result saved | | |
| 5 | Enter result changing the date | Pass | Result saved | | |
| 6 | Enter result changing the time | Pass | Result saved | | |
| 7 | Enter result changing the date and time | Pass | Result saved | | |
| 8 | Enter result without description | Pass | Result saved | | |
| 9 | Enter result with description | Pass | Result saved | | |
| 10 | Enter result using slot | Pass | Result saved with correct time | | |
| 11 | Enter result using slot after time changed manually | Pass | Result saved with correct time | | |
| 12 | Exit result screen without saving entered data | Pass | Data not saved | | |
| 13 | Navigate to main menu from result screen using button | Pass | Navigated back to main menu | | |
| 14 | | | | | |
| 15 | View results list | Fail | Format of date is incorrect | Yes | Pass |
| 16 | Select a result from the list | Pass | Selected result appeared ok | | |
| 17 | Edit a selected result but don't save | Pass | Result was not edited | | |
| 18 | Edit a selected result and do save | Pass | Result was edited | | |
| 19 | Delete a selected result | Pass | Result was deleted | | |







| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Function | Result | Description | Fixed | Retest Result |
| 2 | | | | | |
| 3 | User Interface ok | Pass | Interface works ok | | |
| 4 | Enter new appointment but don't save | Pass | Appointment not saved to database or phone | | |
| 5 | Enter new appointment and save | Pass | Appointment saved | | |
| 6 | Editing existing appointment | Fail | Appointment opening to dayt but no exact appointment | Yes | Pass |

### 2.5.2 Customer Testing

For end user testing I loaded the App on to several people's phones and tablets. It was a bit of an issue finding people as most people have an Android or Apple phone. Most people I found used the windows phone for work.

You can find the user testing document in the appendix.

From the results of the user testing everyone was happy with the GUI with one tester pointing out that the send mail button brings you back to the email screen were it could bring you straight back to the main menu screen.

For the results section again everyone was happy with no additional comments entered.

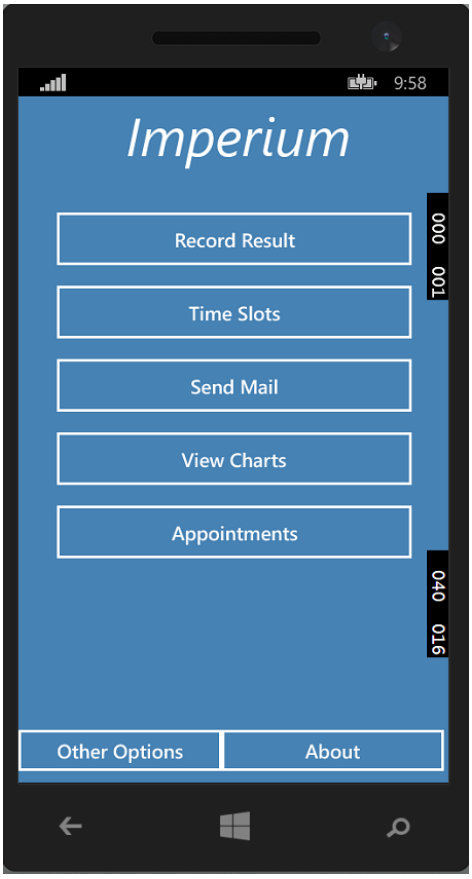
For the slots again everything worked as expected but there were two users who queried if it would be possible for the list of slots to be visible on the create a slot screen to save you having to go into the list view from there to see what slots are already setup

For the email section everyone was happy that it functioned correctly with the only comment being it was a very useful feature.

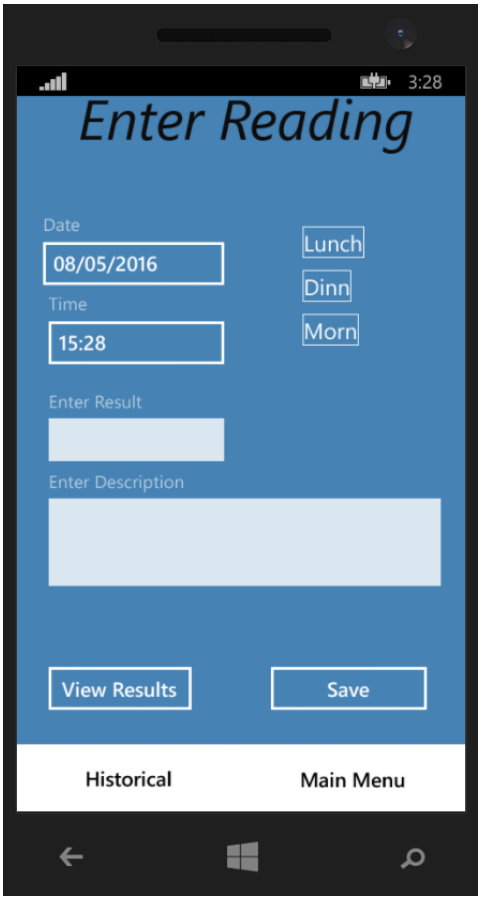
For the charts section the only negative response received was in reply to the 'was it easy to identify trends' question. In the additional comments it was said that the screen size is not ideal. The user was testing on a phone so I made a change such that only basic charts could be used on the phone with the bigger charts being available on the tablet and website.

For the appointments section everyone was happy with no additional comments received.

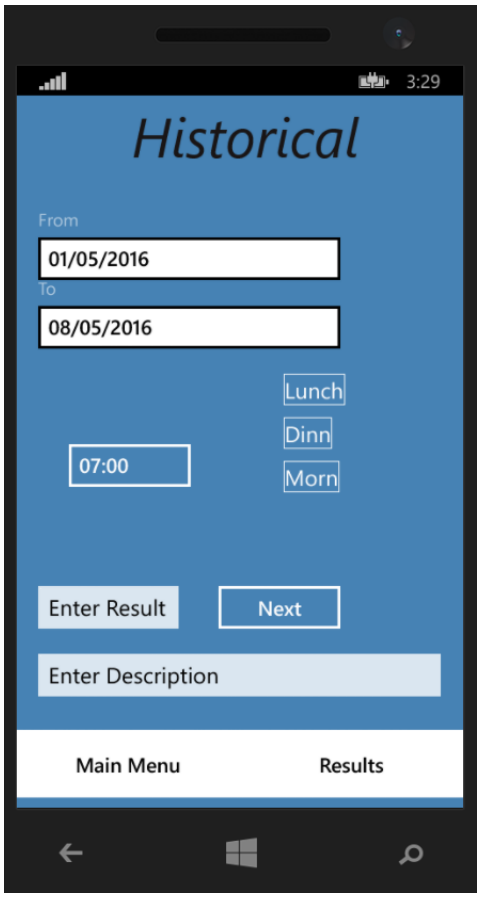
### *2.6 Graphical User Interface (GUI) Layout*




**Main Page:** This is the initial screen loaded when the App is started on a Windows phone. There is a link back to this screen from every other page in the App.

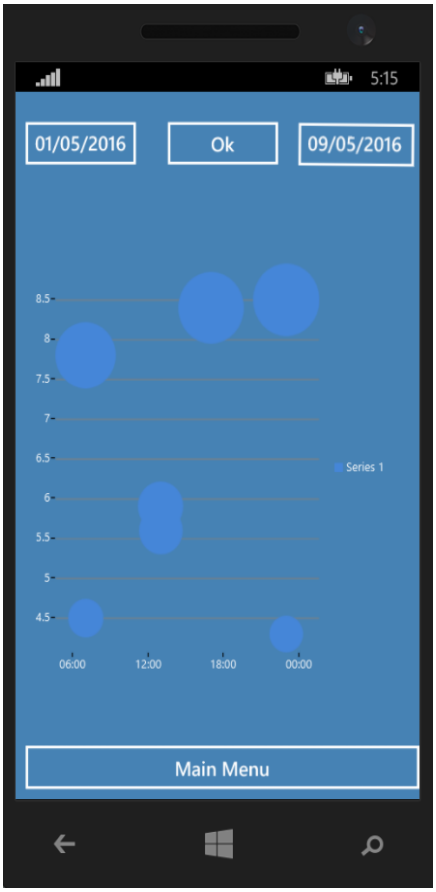



**Enter a Reading:** This is the page used to enter a user's blood sugar readings. There is a button selection on the top which will be defaulted to 'Now' when the user enters the page. This allows the user to just enter a result in the text box and save it to the database. This allows minimal button presses to enter your latest result. Alternatively you can select one of your pre-set time slots and then enter your result. This will record the result at the earliest (from) time period. So if you have lunch set at 12 - 2, it will record it as 12. Finally if you do not have or want to use a slot you can enter the exact date and time if you wish and then enter and save your reading.

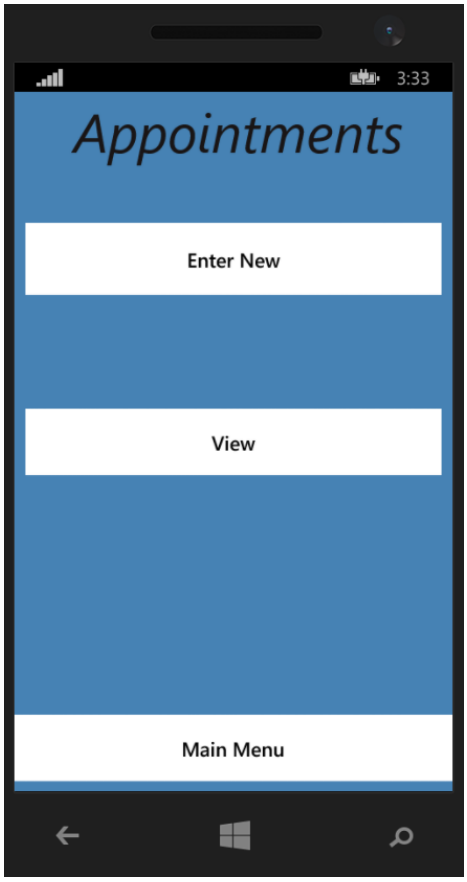



**Historical Readings:** On this page you can quickly enter a range of historical results. In the select a time slot section you can choose a single, a range, or all your time slots. Then you can select a start and end date. Then enter your results and click the tick, then enter next result and click the tick again. This will record your entries in the order you have selected i.e if you select morning time slot and then select from 05/07/15 to 05/08/15, it will record the first reading against the morning slot on the 5th and then the next on the 6th and so on.

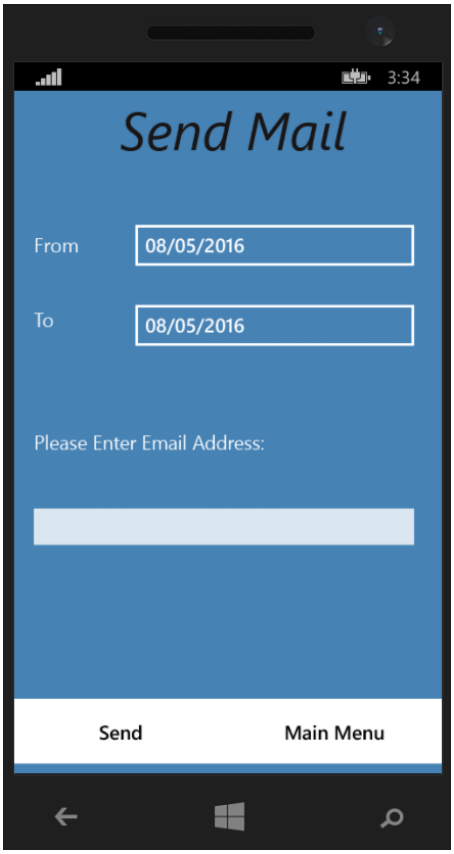

**Setup Time Slots:** This is were the user will setup the time slots used throughout the App. The user can click any existing time slots to edit the name or times, or enter a new slot name in the space provided and select a start and end time and save it as a new slot



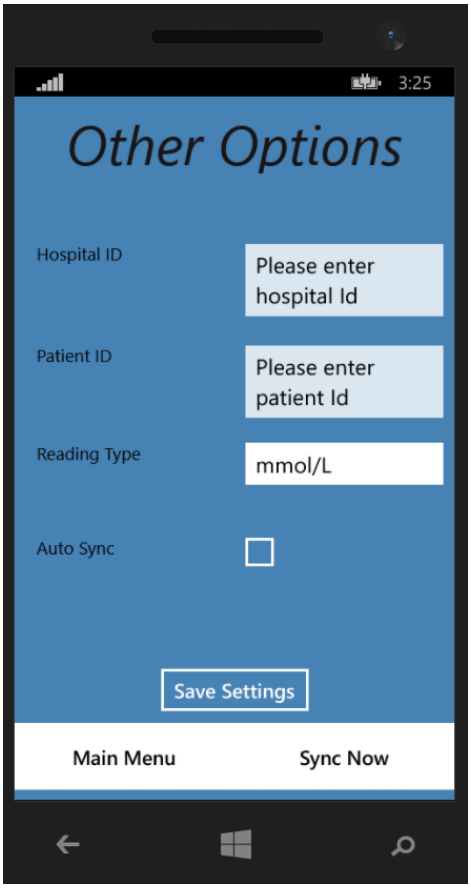

**Graphs:** In this page the user can view a graph of their blood sugar results and see trends like what time of day it tends to go above or below normal. The graph can be filtered by time slots and date. The user can then view the actual data behind the graph by clicking on the ‘view data’ button.



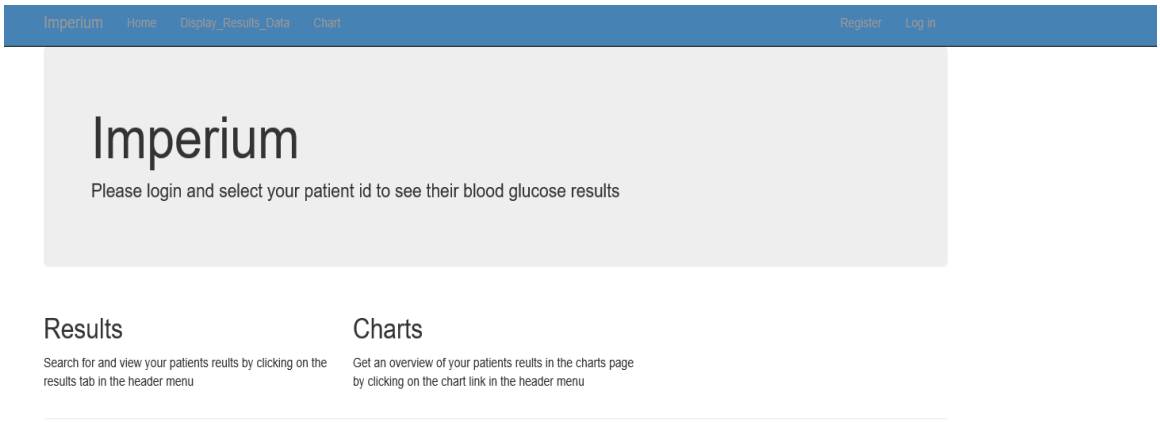

**Appointments:** This is were the user will enter upcoming appointments. They can enter a new one which will bring them to sub-page were they select a date and start\end time. They can also choose to set up a reminder for the appointment. The other option in this page is to view already entered appointments. This will bring up a calendar and the user can select an appointment to view it.



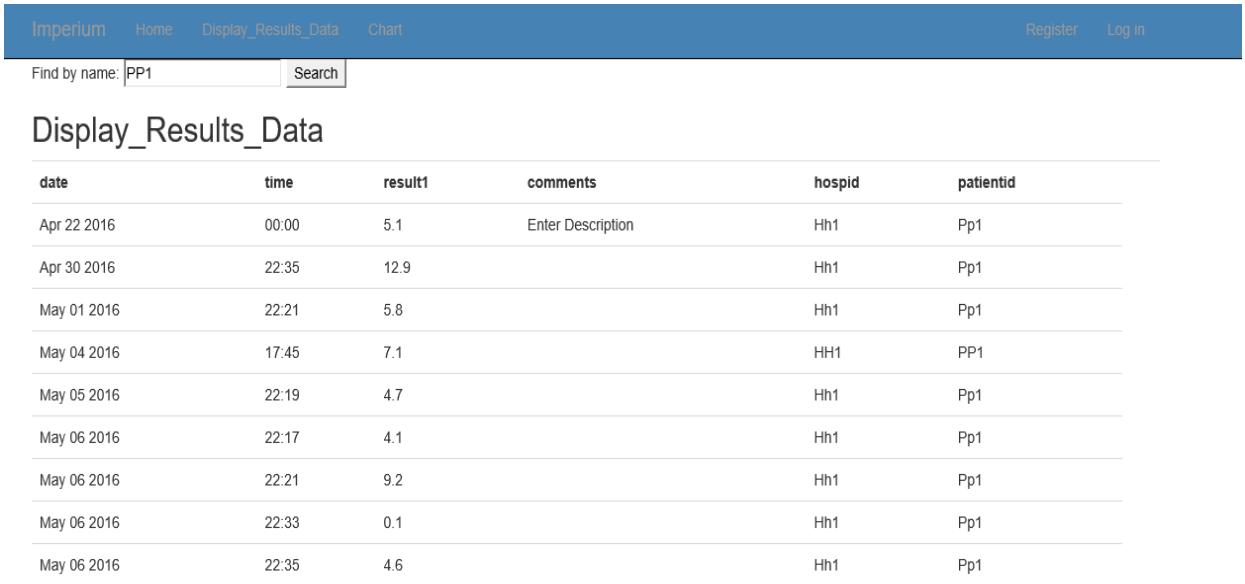

**Send Email:** This screen allows the user to select a date range and filter by time-slot and also choose a format of excel or pdf . They then can press send to send the email of results in the chosen format.




**Other Options:** This page allows the user to select other options for the App such as whether to auto send results email on the morning of any scheduled appointments. They can also select the reading format of mmol or mg. This is also the screen used to setup the sync options and were the button is to manually kick off a sync.




**Website Home Page:** This will be the login page of the website. The user or doctor logs in and can see results as long as that user has chosen to setup the sync option within the App.




| date | time | result1 | comments | hospid | patientid |
|---|---|---|---|---|---|
| Apr 22 2016 | 00:00 | 5.1 | Enter Description | Hh1 | Pp1 |
| Apr 30 2016 | 22:35 | 12.9 | | Hh1 | Pp1 |
| May 01 2016 | 22:21 | 5.8 | | Hh1 | Pp1 |
| May 04 2016 | 17:45 | 7.1 | | HH1 | PP1 |
| May 05 2016 | 22:19 | 4.7 | | Hh1 | Pp1 |
| May 06 2016 | 22:17 | 4.1 | | Hh1 | Pp1 |
| May 06 2016 | 22:21 | 9.2 | | Hh1 | Pp1 |
| May 06 2016 | 22:33 | 0.1 | | Hh1 | Pp1 |
| May 06 2016 | 22:35 | 4.6 | | Hh1 | Pp1 |

**Website Pages:** This is the websites other pages. Above is the results page were all synced results from the App will be listed and searchable by hospital id. There is an option to view graphs of the same data and a link to the home page which will give some information about the website

### *2.7 Evaluation*

The App has been fully test both by the developer and user testing. (Please refer to testing section of the document for more on each of these). All along the development cycle the Visual Studio emulator was used to test each function while it was being developed. Along with the user testing I have been using the App daily to record my own blood glucose levels and also the other features of the App.

### *2.8 Conclusions*

I have really enjoyed the process of creating the App from the design phase right through to using it on my phone. There have been plenty of challenges to overcome but working through any issues has really improved my understanding of coding and the designing and documentation required has made me realise the importance of these planning stages to assist the development of a better product at the end of the process.

Overall, any person who uses the App should experience an easy to use yet powerful App which they can choose to use solely for the purpose of recording their sugar levels for themselves, or they may choose to use the additional functions, which are intended to offer a more interactive approach to maintaining optimal blood sugar control.

I think the App will be a welcome addition to the market and will offer a helpful assistance for people with diabetes.

## 3 Further Development or Research

- Future releases for IOS and Android phones and tablets. This will expand the range of potential users and then the App will be available on the three biggest Operating Systems used in phones and tablets.

- Desktop Application that will sync with the App through Bluetooth. This will further make it easier to sync the results with your doctor. On arrival the App (Installed on the doctor's machine) can automatically pull down a user's results with the click of a button.

- There is a plan to add a news section to the App. This will pull Diabetes related news from a google news API. The user clicks on it and it will be filtered to any relevant stores in the previous 24 hours.

- In future releases there will be sections for additional areas related to Diabetes management such as foot and eye care, and to record foods consumed throughout the day.

- There are plans for the health service to move from a largely paper-based system to one that uses electronic records. As diabetes care is delivered in hospital and community settings, its services stand to gain a lot from such a development. This App can be further developed to work with any such new system. It should also be compatible with the soon to be introduced Individual Health Identifier (IHI).

- There is the potential for this App to support the delivery of remote care for diabetes and possibly other chronic diseases. For diabetes alone, it may be possible to shift care from a hospital setting to the community which benefits both service providers and patients alike. The App might one day allow users to experience an entire continuum of care, much of it from their own home as opposed to episodic care, given that it allows a wealth of information of blood sugar readings to be communicated to any HCP at the touch of a button and on a regular basis if need be. It should provide comfort to those newly diagnosed with diabetes, or to those who experience it for a short period of time such as those with gestational diabetes, the prevalence of which is increasing sharply in Ireland.

# 4 References

Model your App's Architecture – Research on an applications Architecture. [ONLINE] Available at: https://msdn.microsoft.com/en-us/library/dd490886.aspx [Accessed 22 November 2015].

A technique for Architecture and Design – Research on Architecture & design. [ONLINE]

Available at: https://msdn.microsoft.com/en-us/library/ee658084.aspx [Accessed 23 November 2015].

Windows Phone Architecture – Research on Windows Phone Architecture [ONLINE]

Available at: https://sysdev.microsoft.com/en-us/Hardware/oem/docs/Getting_Started/Windows_Phone_architecture_overview [Accessed 26 November 2015].

Using offline data sync in Mobile Services – Syncing an App's local database with an Azure database using Mobile Services [ONLINE]

Available at: https://azure.microsoft.com/en-us/documentation/articles/mobile-services-windows-store-dotnet-get-started-offline-data/ [Accessed 4 December 2015].

Bob Taylor – Windows Phone 8.1 Development for absolute Beginners

Available at: https://channel9.msdn.com/Series/Windows-Phone-8-1-Development-for-Absolute-Beginners [Accessed 10th January 2016]

# 5 Appendix

## *5.1 Project Proposal*

Project Proposal

Imperium

Andrew Bradley, X12118770, andrew.bradley@student.ncirl.ie

BSc (Hons) in Computing

Networking and Mobile Technologies

Date 29/09/15

Objectives

The objective of this project is to create an App that assists a person with diabetes to track their blood sugar readings. This is one of the most important aspects to the management of the diabetes, as stable blood sugar levels are essential to good health. The App needs to be both reliable and user friendly, with streamlined functionality in order to ensure that it can easily be incorporated into most, if not all, the user's daily routines. Given the frequency with which blood sugars have to be checked, time is of the essence to encourage the regular tracking of levels that is needed to maintain good control.

As such, there should be as few button presses as possible required to open the App and record the latest blood sugar reading. This will be done by allowing the user to set up time slots for the times they would generally take their blood sugars so that when they open the App, they need only select a pre-set time slot such as Breakfast or Morning and then simply enter their reading and press save. This will eliminate the need to select dates and times for every reading the user enters, and, as readings are for the most part taken at certain intervals through the day, this can be used for the majority of the readings.

In addition, it can act as a reminder mechanism, by alerting the user to any missed time slots, within, say 15 minutes after that slot has ended. There will be a button to enter 'extra' readings which can be used to enter any unscheduled readings. For this, the user will have to select a date and time from the drop-down menu and then enter the reading.

Aside from recording a user's readings the App will also provide useful feedback, by displaying KPI's and graphs indicating trends in a user's readings and these will be filterable with a start and end date. The graphs will provide an at a glance health report for user, Doctor and parents of children and allows you to easily identify problem areas without having to go through lists of results.

The App can be used as a standalone recording tool or you can enable the App to sync data with a website. On the website you can view your readings history and have extra KPI's\graphs if you wish to see it on a bigger screen. A username and password will have to be setup on the App to enable this option and to access the website, in order to keep the data secure. If you would like to give access to your Doctor he/she can then

login and have access to all your blood sugar readings aiding him in helping you stay in control of your blood sugars.  It is anticipated that this feature will be of particular benefit to young people in encouraging, and facilitating, their engagement with their health care practitioner in the management of their condition.

There will be an option to email results in either word, excel or pdf formats and you can specify which date periods you want to include in the report. There will also be an option to add any hospital appointments to a calendar and again prompts can be setup to notify users of upcoming appointments.

Overall, any person who uses the App should experience an easy to use yet powerful App which they can choose to use solely for the purpose of recording their sugar levels for themselves, or they may choose to use the additional functions, which are intended to offer a more interactive approach to optimal blood sugar control.

Background

Having had Diabetes for the past 17 years I wanted to build an application to record my blood sugars with as few button presses as possible, and also provide me with an easy way to share my readings with my doctor.

Currently I have to record my results either on paper or excel and then print them out and bring with me when going to the hospital. Usually this means transferring the previous 2 months' worth of readings from my tester and then printing out the night before the hospital appointment. It would be ideal to be able to enter them into the App at the same time as I take my reading, and know that this is then available for my doctor and myself to look at later.

Having tried some Apps that are available I found that most take an unnecessary amount of time to enter a reading as you have to scan through and find the exact time and date for every entry. I want to be able to setup a number of slots such as Morning, Lunch etc. and then select this and enter a reading. The option to add exact time can be used if preferred but the one click easy approach is intended to be the most used.

I am also required to attend the hospital multiple times a year for various checkups and blood tests. The App will have an area to enter appointments and will give reminders the day before as to an impending appointment, from this reminder you can choose to email results to a specified address such as your doctors. I believe that such an integrated system points to the way in which all future healthcare will be delivered and engaged with.

Technical Approach

I will be using Microsoft Visual Studio to create the App for windows phones\tablets and coding in C#. I will build the App and then carry out testing in the Windows phone Emulator that comes with Visual Studio.

I will also use Microsoft Visual Studio to build my website and websites database. This will then be published online to allow my Application to communicate with it via web services. The App can be used as a standalone App if the user does not want to use this extra functionality but if they do they will need to connect to Wi-Fi or a 3\4G connection. There will be an option to delay syncing until you are on a Wi-Fi connection if the user does not want to use their data connection in case of any network charges.

Special resources required

Hardware:

- Phone running windows 8 or above.
- Tablet running windows 8 or above.

Software:

- Visual Studio

Technical Details

C#

HTML

SQL

ASP

Web Services

Evaluation

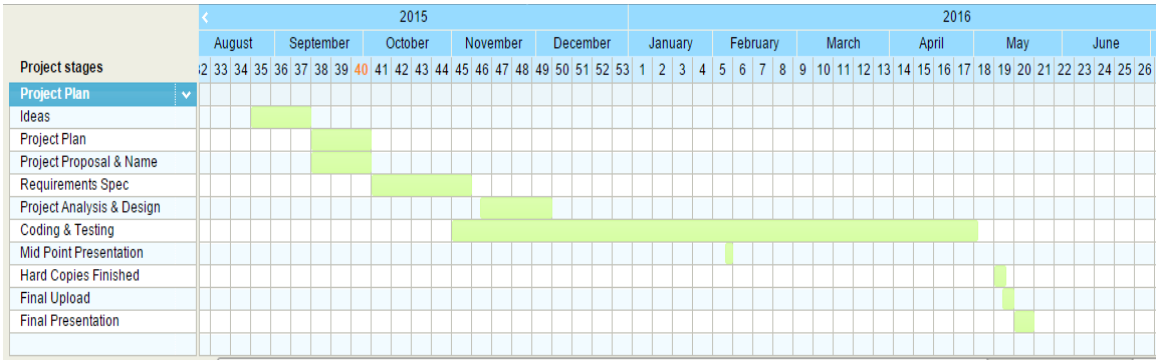
Once the development/building stage is complete, I plan to use the App in my daily routine. As I record my blood sugar levels throughout the day I will also input them into the App.  I will then be checking the website to make sure the results are being synced correctly and also the KPI screens and graphs are functioning okay.

I hope to engage with my Doctor to see if he will agree to log in and view my results on the website and provide feedback on improvements that could be made to benefit him more. I will also be asking my Doctor if he can recommend any other patients who would be willing to download my App and start using\testing. This of course would rely on the patient having access to a windows phone or tablet. Failing this they could enter directly into the website but ideally end user testing should be done using the phone\tablet based App.

## *5.2 Project Plan*

Project Plan

| Project Task | Start | End |
|---|---|---|
| Ideas | 24/08/15 | 11/09/15 |
| Project Plan | 14/09/15 | 02/10/15 |
| Project Proposal & Name | 14/09/15 | 02/10/15 |
| Requirements Spec | 05/10/15 | 06/11/15 |
| Project Analysis & Design | 11/11/15 | 04/12/15 |
| Coding & Testing | 02/11/15 | 02/05/16 |
| Mid-Point Presentation | 06/02/15 | 06/02/15 |
| Hard Copies Finished | 09/05/16 | 11/05/16 |
| Final Upload | 11/05/16 | 14/05/16 |
| Final Presentation | 14/05/16 | 21/05/16 |




## *5.3 Monthly Journals*

Reflective Journal

Student name: Andrew Bradley - X12118770

Programme: BSc in Computing

Month: September

My Achievements

In the month of September I had to pick a project and then upload my project proposal and plan. Over the summer months I was loosely thinking of a couple of ideas but now I had to separate non-runners from the good ideas and start to really think of what could be both an achievable and successful project and then to commit to it.

I had 3 projects that were runners, the first was related to a request in work, were users wanted a portable way to pick goods from a tablet rather than printing out pick sheets. An App on the tablet that linked back to our ERP system was suggested but the costs quoted from our ERP provider were too high. I was going to look at developing the App for my project but after consideration decided against it as A) there would be less resources available online for our ERP system if I ran into trouble and we don't have any in-house programmers for assistance and B) it could be troublesome to show in the presentation as I would need a guaranteed connection into the workplace.

The second option was for a local community App, were you can sign in with your address or Eircode and get automatically added to your local group\community. The App would send alerts of goings on in the community and any security alerts. You could also search for recommended tradesmen in your local area. After considering I felt it may be a little light in terms of functionality and also to fully test you would need a large group from different areas and an admin to control each group.

The third and chosen idea was an App for Diabetes. I have Diabetes myself and have always found it very annoying having to gather my blood sugar results together the night before a visit to the Outpatients clinic. I wanted to primarily make it as easy as possible to record and evaluate my blood sugar readings and also to share these with my Doctor as quickly as possible. I wanted the App to become part of a user's Diabetes management.

I done up my project proposal based on the Diabetes App and have started to think, of additional features I can add, while trying to keep it within a realistic\achievable target. To this end I am looking forward to being assigned a supervisor to discuss my plan and to briefly go over my other 2 suggestions as a final ok that I chose the correct project.

To start with the programming side of things as I had recently upgrade my laptop to Windows 10, I wanted to ensure Visual Studio was working ok. To this end I downloaded

the latest version and have begun doing some of the Tutorials around Windows App's that were posted on Moodle.

My Reflection

I felt it was of most benefit to start writing down the beginning of a project plan for each of my ideas and once down on paper I could get a clearer picture of what each one entailed.

On the other side of that I perhaps wish I had started writing down my thought processes for each idea earlier as I could have focused on my final choice earlier in the summer.

Intended Changes

The next upload due is the project specifications document. I intend to work on this daily, and to write down as much as possible over the next few weeks so as I am in a position were I can remove things at the end if needed, rather than struggle to write everything down at the last minute.

I am planning to complete a number of online tutorials for making a Windows App in Microsoft Visual Studio and to look at hosting options for my website.

Supervisor Meetings

We will be assigned a supervisor in the next few weeks.

Reflective Journal

Student name: Andrew Bradley - X12118770

Programme: BSc in Computing

Month: October

My Achievements

For this month we have had to get the project requirements specification document done. On first opening it seemed like a lot of sections were involved in the template provided, but I planned to take a section every few days so I wasn't over-whelmed towards the end of the month.

I started the introduction section first as the requirements in this section were clear as most had been put in the project proposal document.

I then decided to research some online tools to aid with my diagrams. I found 2 that looked good and decided to go with them. I would use Nulab for the use case diagrams and the class diagram. For the GUI interface I found a program called NinjaMock that had templates for a windows phone App so was ideal for my project. It was good to see the project coming to life if only in a series of images it still gave me a sense of how the project would look and feel. I ended up with 8 different GUI screens and will have some smaller sub-menus within some of them as well. I was surprised with how many screens you can actually think off and there was a temptation to add more but I decided to keep it realistic for the time-frame involved.

It was a similar situation with my use case scenarios. From initially thinking will I get the 4- 5 recommended, before I knew it I was on number 9. Again there was a temptation to keep adding more but to not over stretch on my targets I decided to leave it at that and re-visit nearer the end of the project if I find there is time left over for additional functionality.

I have also been creating some test projects in Visual Studio and following online tutorials on Windows App creation. I have completed a number of them and got them running on the emulator so am gaining confidence for starting coding on my own Application.

My Reflection

I enjoyed getting the GUI screens and the use cases drawn up. It started to give me a clearer picture of my end goals and my thought process's towards beginning the programming is more clear. It also makes me realize that there is quite a lot to be done, but I am looking forward to the point of seeing my mocked up GUI's become a reality on the phone and tablet.

Intended Changes

The next upload we have due is for the Project Analysis Design document. I intend to have this done early in the month as I really want to get into the coding aspect of my project.

For the coding I have already completed a number of online tutorials for making Windows App's in Visual Studio and I intend to continue down this road at the beginning of next month before starting on the actual App.

For my website hosting I have been looking at Microsoft Azure and I intend to look into the 6 month student pass that they have available. Hopefully this is available and if activated at the beginning of next year should be still active for the end of the project. Failing this I will look into other alternatives.

Supervisor Meetings

I met with Paul Hayes on Tuesday 13th October. He had some good ideas for additional options in my App including to sync with a desktop based version of the software automatically via Bluetooth. I will look at this to see if I can add this to my Apps functionality within the time-frame of the project.

He said I can email him if I have any more questions and that he will be following progress from these monthly journal uploads.

Reflective Journal

Student name: Andrew Bradley - X12118770

Programme: BSc in Computing

Month: November

My Achievements

For November we have had to get the project design specification document done and uploaded. This involved more detailed diagrams and forced me to look more closely at the key components to my App. From doing research and online samples I have decided

to go with the MVVM architecture design when coding my App. This separates the sections and can allow for changes to UI and other sections without needing a complete re-design of the App itself. I will also use the universal design as this will allow the App to run on multiple devices without having to worry about a re-code for each type of device.

I spent a lot of time this month doing some small one page Apps in Visual Studio to get an idea of what is involved. From this process I found that the emulator section of Visual Studio were you can test the running of your App didn't work on my laptop. I had been using a work laptop up to that point to run the emulator but was now running on my own laptop. You have to have the pro version of windows to make use of this feature. After much searching for alternatives my choice seemed to be to connect the phone directly and load the App each time I wanted to test a change or upgrade. I had recently upgraded to the free windows 10 version on my laptop and decided to upgrade this to pro to get the emulator functionality. After this I was able to run and test each program I had created and it was good to see the end result of the tutorials I had completed so far.

I started a trial of version of Azure to be able to look at the Azure Mobile Services which I intend to use to sync my offline phone data to an Azure SQL database. I was able to create a basic App in Visual Studio and sync some data from the local SQLite database to the Azure SQL database. I will now look into the options I have available to me to get something activated when the trial version runs out.

My Reflection

I found doing this document has now made it very clear in my head how the entire App will tie together and also in deciding the final architecture I will be using to code the App in Visual Studio.

I was happy to get the emulator up and running on my laptop as it allows me to see the results of the coding I am doing on a replica of a phone on screen. This will save a lot of time over the course of the project as I won't have to continuously connect the actual hardware to see the end result.

Although I would have liked to have gotten more actual coding done in November I am happy to now have the 3 document uploads complete so December and January can concentrate on getting into the code and hopefully having an early working example in place for the mid-term presentation in February.

I have done a lot of reading on the different designs for Windows App. I briefly looked at doing the App for windows 10 but as I had no windows 10 phone or tablet I would not be able to see the end result on a device without purchasing new hardware so I decided against it. I am going to develop for Windows 8.1 and use the universal pattern so the App can be run on a phone, desktop or tablet without the need to re-design everything.

Intended Changes

In December I intend to concentrate on the coding as much as possible and start proper development on my application. I hope to make as much use as possible of the Christmas holidays from work to get ahead of schedule on the development and also hope to fit in some studying for the one exam we have in January for this Semester.

By the end of December\beginning of January I would hope to have a basic version of my App running in the Visual Studio emulator and then I will look to focus on the sync to the Azure database through Azure Mobile services after that.

Supervisor Meetings

As agreed in the meeting Paul Hayes will review my uploaded documents on a monthly basis.

Reflective Journal

Student name: Andrew Bradley - X12118770

Programme: BSc in Computing

Month: December

My Achievements

In December I have created my Universal App in Visual Studio and have created both the main page of the App and then added the packages for SQLite to the project. I have successfully created my database which will checks if one exists and if it doesn't it gets created on the first opening of the App. I have also created my results table and have linked it to the results view page of the App. I have then tested adding, editing, deleting and viewing a list of entries in this results table. Most of these files have been created in the shared section of the universal App, the actual pages users will see have been done

in the mobile specific section and I will add another view page for those using on a tablet/laptop to make use of the bigger screen.

I was most happy this month to get my database up and running. Now that the underlying structure is there I should be able to add my other tables following the same process and to get one table functioning correctly was a good moment in the project.

My Reflection

It is good to see the beginning of my own App rather than the tutorials I have been programming up until now. Even though it is only the main page with one other sub section working to see it in the emulator is good and keeps you motivated to get more complete.

I was happy to get the SQLite database working as this was an important task to complete early on in the development as testing each page cannot be done properly without some data to show.

I have made some slight amendments to my design document as when creating the database structure I came across a couple of fields that would be better off in their own table as they are likely to change quite often and to go back and change each entry in a table wasn't the best option. Now they can be referenced from their own table each time they are required by a report etc.

Intended Changes

For the rest of December I plan to make use of the holidays and continue to work on the development of the App. At the end of January, I hope to have all the database tables created and linked to the corresponding views in the App. After that I will concentrate on the individual functions like running reports filtered to time slots and date periods. I expect the first week in January to mostly be taken up with revision as we have the only exam of this Semester on the 8th January, but after this there should be a few weeks to just concentrate on the project.

Supervisor Meetings

As agreed in the meeting Paul Hayes will review my uploaded documents on a monthly basis.

Reflective Journal

Student name: Andrew Bradley - X12118770

Programme: BSc in Computing

Month: January

My Achievements

In January I have now finished the creation of the XAML pages for my App. The links between each page are working and all navigation buttons are functioning correctly. Furthermore the results page is adding in to the database and is correctly recording the date the result went in.

I have also started to add the functionality for the emailing page and have successfully sent an email from within the App to the specified email address.

Another area I have started to add functionality for is the graphs. I have found a package you can add to Visual Studio that includes templates for various graphs so have loaded this to my App and have gotten a basic line graph working from my data stored in the database.

My Reflection

For the early part of January I spent most of my time going over material for the one exam we had for the first Semester. After that the focus moved on to the technical report which is the final document upload before the mid-point presentations.

I would have liked to be further along in the development stage but it is important to get the documentation completed and uploaded as the mid-point presentation is worth 25% of the final marks. After the presentation the focus will shift primarily to the development.

I am having some trouble getting my time slot buttons to work on the results page. The Grid view layout is not formatting them as expected and the functionality behind them is not yet working correctly. I am hopeful this can be resolved in the coming weeks.

I was happy to get some of the functionality like emailing and the displaying of graphs working and also to see all the user interface complete and navigation links working successfully.

Intended Changes

Having now completed the XAML pages for the App, I will continue to link in the different parts to the tables in the database. I have the email functionality working in the App also so I will focus on getting pdf creation developed so the attachment can be added to the email.

By the end of February I would hope to have the graph and data pages completed and also the emailing of the pdf completed.

Supervisor Meetings

As agreed in the meeting Paul Hayes will review my uploaded documents on a monthly basis.

Reflective Journal

Student name: Andrew Bradley - X12118770

Programme: BSc in Computing

Month: February

My Achievements

The biggest achievement in February was to get the mid-point presentation and document uploads complete. The presentation went well with some good feedback. I had a working demo on my phone to show at the end so it felt like real progress was being made. There was a number of CA's due during February for the 2 other modules also so a bit of time was diverted and spent preparing and doing those.

From the App point of view I was happy to see the App running on an actual phone and being able to test some of the functionality like email from the phone itself and also making sure the database on the phone is storing and loading the data correctly, when the phone is switched off and on etc.

My Reflection

Now that the mid-point presentation is complete, focus is fully on completing the App's functionality. The realization of how little time is now left is becoming apparent and my focus is just to put as much of my free time as possible into completing the project and the work for the 2 other subjects we have this Semester.

I would like to make a big step forward during the reading week and hope to close out some of the problem areas I have with the development so far.

Also from feedback from the mid-point presentation I want to get a good test plan in place so I can ask my doctor if other patients will be willing to use the App and provide feedback about any issues or possible improvements.

Intended Changes

Intended work for March is to complete all the functionality aspects of the App and then to move on to the syncing process with the external website. I had an issue with the displaying of the graphs were they work fine on the emulator but behave differently on an actual device so I want to get to the bottom of this also.

Supervisor Meetings

As agreed in the meeting Paul Hayes will review my uploaded documents on a monthly basis.

Reflective Journal

Student name: Andrew Bradley - X12118770

Programme: BSc in Computing

Month: March

My Achievements

For March most of my efforts were on getting my project done for multimedia and 2 CA's for distributed systems.

There wasn't a lot done on the App itself as my planned time to do it was taken up with the other subjects this month but with them done it should clear a lot of time for the App itself.

My Reflection

I was a little disappointed in how little time I had to spend on the App as the other subject work took up so much time this month. I had to spend a lot of time on the distributed systems CA's in particular as I found this subject quite difficult this Semester.

I now will focus on the 2 exams we have in early April and have a schedule setup for when the exams are finished as there is a lot of work still to complete in the App. With everything else effectively finished at that stage I am hoping to make good progress and have the App fully complete by the upload date in May although it does some a little daunting at this stage when I look at what still needs to be finished.

Intended Changes

From mid – April (after the exams) I am aiming to have everything in the App fully functional and have users testing by the end of the month. My first version that was used in the mid-point has performed well so I am hopeful of a successful final version.

Supervisor Meetings

As agreed in the meeting Paul Hayes will review my uploaded documents on a monthly basis.

Reflective Journal

Student name: Andrew Bradley - X12118770

Programme: BSc in Computing

Month: April

My Achievements

April was all about getting the App finished and ready for testing in early May. I was pleased with how progress went and am feeling good going into May. I have gone through hand tested each function of the App and have fixed up the few small issues I have found so far. Items completed in April include:

Email date filter and email of attachment fully functional

Graph date filter and display fully functional

Historical result entry algorithm fully functional

Appointment link to local calendar fully functional

Always on sync and manual sync option both fully functional

Azure website link to Azure SQL database fully functional

My Reflection

I am happy with the progress I made towards the end of April. A couple of issues I was having were resolved and I was happy to start using the fully functioning App on my mobile so I can fully test over the next week.

Intended Changes

Complete all functions of the App and also the website for displaying user's results to the Doctor.

Supervisor Meetings

As agreed in the meeting Paul Hayes will review my uploaded documents on a monthly basis.

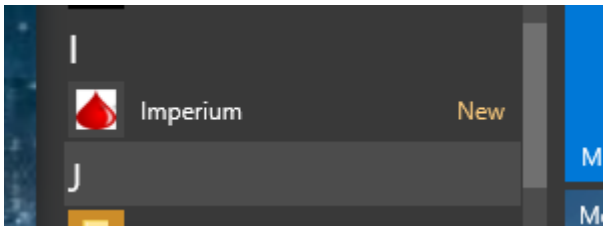
## *5.4 Other Material Used*

### 5.4.1 Testing Documents for End Users

### 5.4.2 Instruction Manual

**Imperium Instruction Manual**

Once you have downloaded the App from the windows store select it from your list of Apps. It will be under I and have the blood drop logo.



When the App is launched you will in the main menu section. From here you can choose from the following menus

- Record Result
- Time slots
- Send Mail
- View Charts
- Appointments
- Other Options
- About

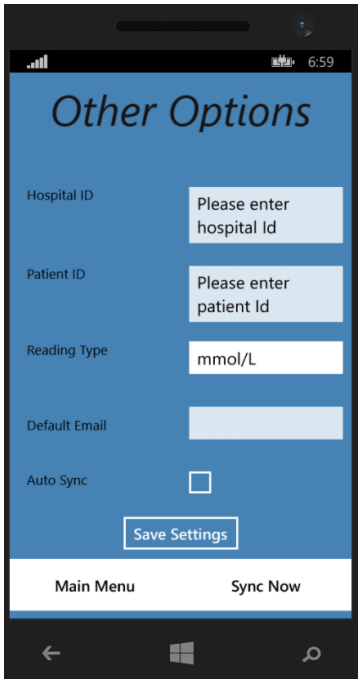
The first thing you should do is enter your hospital id and patient id. This makes your results unique when syncing with Azure database, so the doctor can quickly filter to your results only based on your patient id. You will have a patient id assigned to you were you receive your Diabetes care so if you do not know it you can get it from them.

Other Options

In here you can set the values of below fields. These are store in the Apps local settings and are retained even if the device is switched off. Also in this page is the sync now button which when pressed will sync any un-synced data with the SQL database on Azure

- Hospital ID - Hospital you receive your diabetes care
- Patient Id – your unique patient id with the hospital above
- Reading Type – can be set to mmol or mg/dl
- Auto Sync – this determines if the App automatically syncs to Azure or waits for you to manually select the sync now option
- Default Email – enter your doctors email to save you having to type it in each time you want to send an email

Once you have entered your preference for above press the save settings button.




Time Slots

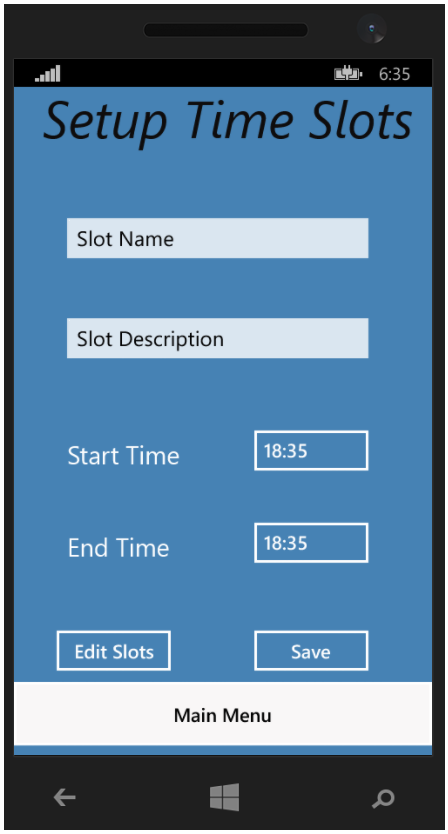
On this page you can

- Enter a name for the time slot
- Enter a description for the time slot
- Enter a start time for the time slot
- Enter an end time for the time slot
- Edit existing slots

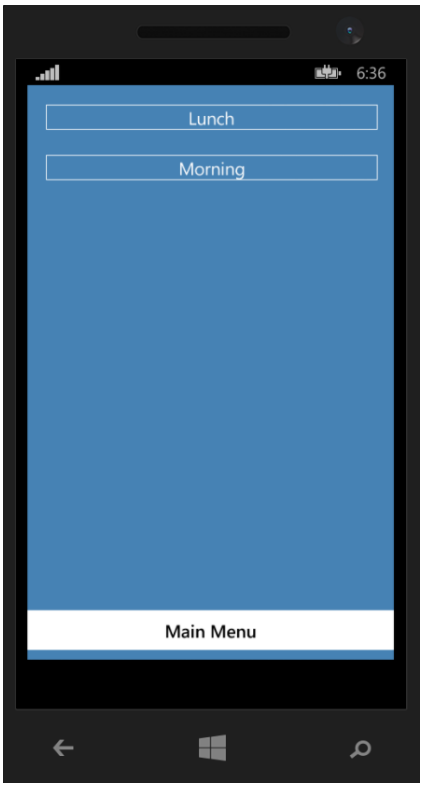
Fill out all the fields and then press the save button to store your slot in the database.

To edit existing slots press the edit slots button and then select the slot you want to edit and update the slot with the new information and press save. If you want to remove the slot press the delete button.

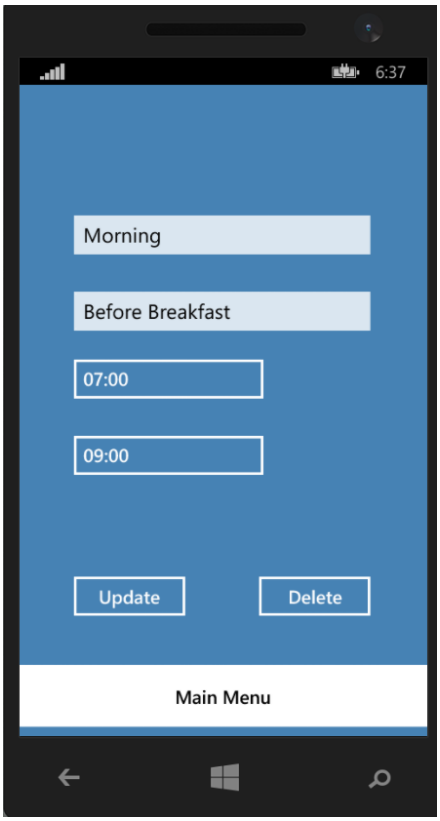
Time slot screen




Edit time slot list




Edit or Delete



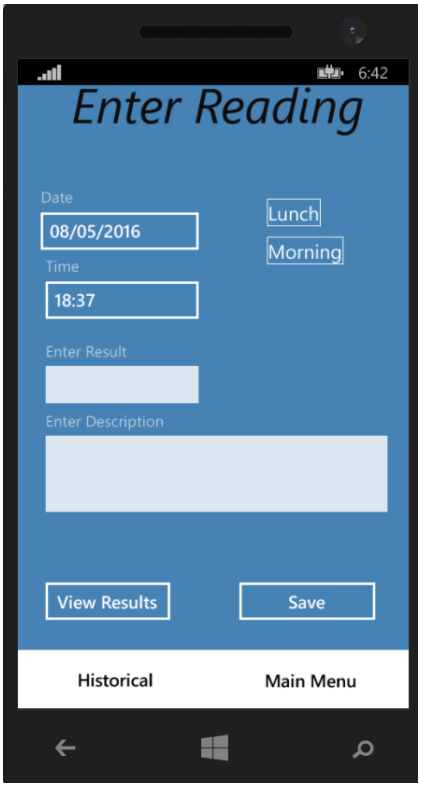

Record Result

On this page you can

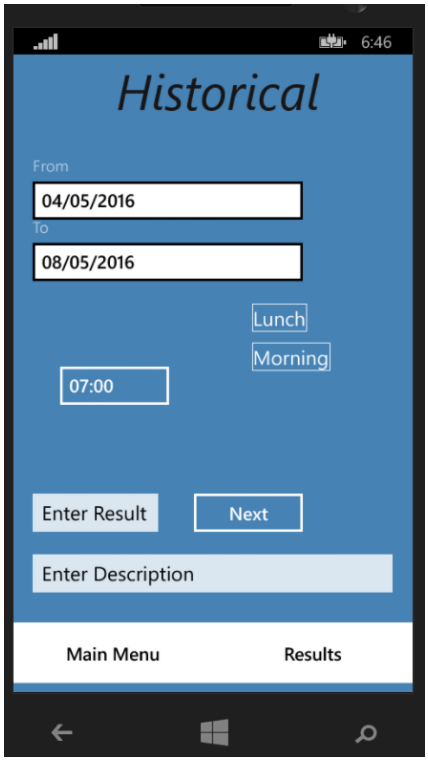
- Record a result
- Edit existing results
- Record a group of historical results

When you click on the result page the date and time are set to the current date time. To record a result for the present just simply enter the result and press save. If you need to change the date and time just select the date or time picker and choose the desired date or time.

If you are recording results for earlier in the day and have a time slot setup for this time period just simply press the time slot to set the time picker to that slots start time. This saves you having to change the time picker when entering earlier results and requires only a single button press to set the desired time.




If you have a list of historical results to enter then press the historical button along the bottom of the page. From here just enter your start and end date and either the time or time slot you will be entering for. The just simply enter your result and press next. The date will automatically move to the next day and the time will remain in the slot you selected. Keep entering your results and pressing next until the to date is reached in which case the App will prompt that the routine has finished.



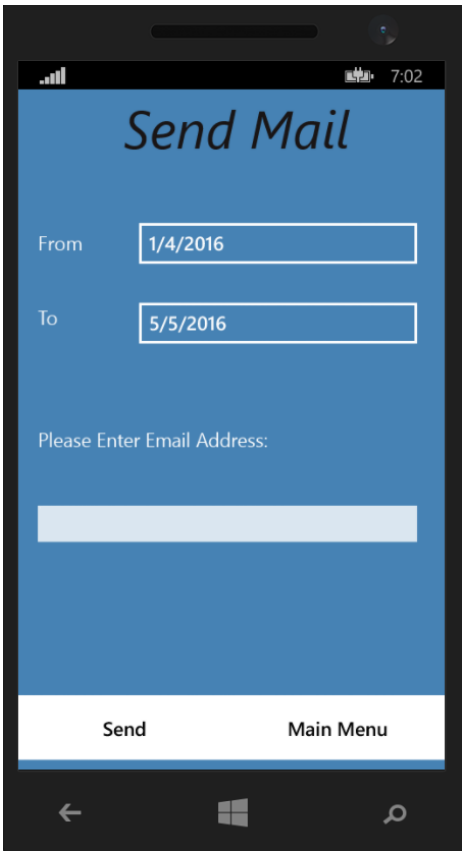

To view your entered results from the results page select view results near the bottom of the screen. From here you can view or select any result that is stored in the database and edit and delete as required.

Send Email

On this page you can

- Enter a start date
- Enter a to date
- Enter an email address

When you enter this page if you have entered a default email address on the other options page this will automatically be populated on this page for you. You can change if you want also. Select a from and to date from the pickers and then if no email address already entered, enter one. Press send and the App will generate a csv of all your results for the dates specified and email to the address chosen.

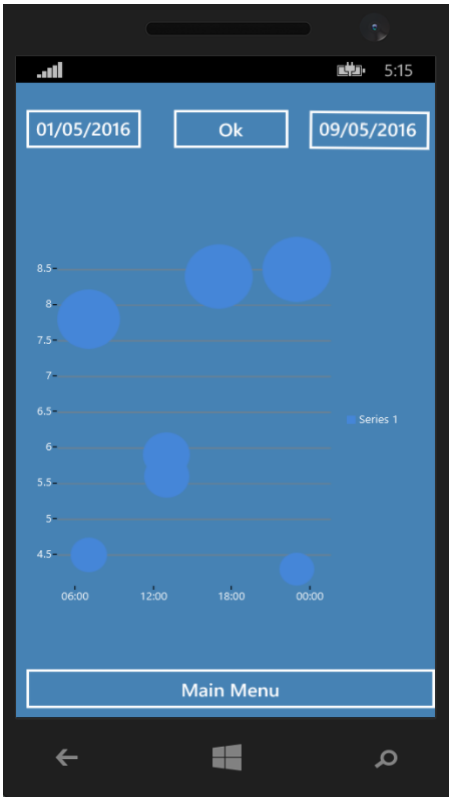



View Charts

On this page you can

- Enter a start date
- Enter an end date

After you enter a start and end date press ok and the chart will display showing all your results for the time period. If you prefer to view your charts on a bigger screen you can log in to the website to view your data in bigger charts also



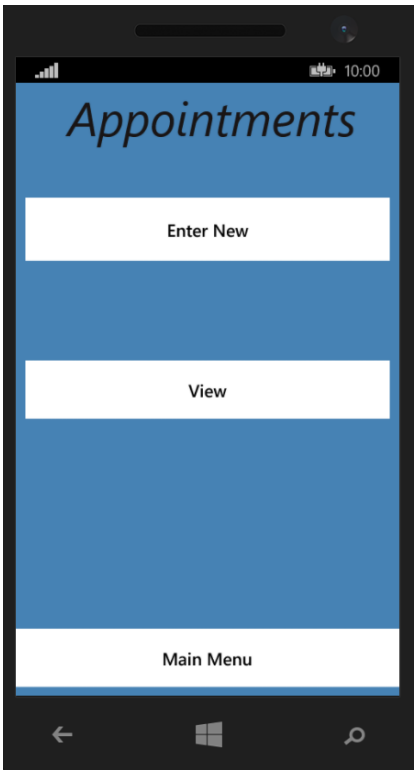

Appointments

On this page you can

- Enter a new appointment
- View existing appointment

When you chose to enter a new appointment fill out the fields on screen and then press save. An appointment will be saved to the devices calendar and a reminder will be automatically set.

When you select to view appointments you will be shown a list of appointments entered from this App. By clicking one you will be brought straight to that entry in the Apps local calendar.




About

On this page you can

- View information about the App

### 5.4.3 User Testing Document

Imperium App Testing

| | Please circle one: | |
|---|---|---|
| Device tested on: | Phone | Tablet |
| Windows version tested on: | W8.1 | W10 |

GUI

| | | |
|---|---|---|
| Did the menus function correctly | Yes | No |
| Did you find the GUI was too small | Yes | No |
| Did you find the GUI was too big | Yes | No |
| Did you find the GUI very responsive | Yes | No |
| Was the layout easy to follow | Yes | No |

Any other comments about the GUI

_____

_____

_____

_____

_____

Results Section

| | | |
|---|---|---|
| Was a result easy to record | Yes | No |
| Was it easy to record historical results | Yes | No |
| Was it easy to follow the procedure | Yes | No |
| Were results saved correctly | Yes | No |

Any other comments about the results page

_____

_____

_____

_____

_____

Slots section

| | | |
|---|---|---|
| Was the setup easy to do | Yes | No |
| Were the slots displayed correctly | Yes | No |
| Were they a useful addition | Yes | No |

Any other comments about the slots section

_____

_____

_____

_____

_____

Send Email

| | | |
|---|---|---|
| Was it easy to follow the procedure | Yes | No |
| Did the email go ok | Yes | No |
| Was the attachment ok | Yes | No |

| | | |
|---|---|---|
| Was the function useful | Yes | No |

Any other comments on the email page

_____

_____

_____

_____

_____

View Charts

| | | |
|---|---|---|
| Did you find the charts useful | Yes | No |
| Was it easy to identify trends | Yes | No |
| Did the chart load quickly enough | Yes | No |

Any other comments on the chart page

_____

_____

_____

_____

_____

Appointments

| | | |
|---|---|---|
| Was the appointments a useful feature | Yes | No |
| Was it easy to use | Yes | No |
| Did the reminder pop up | Yes | No |

Any other comments on the appointment page

_____

_____

_____

_____

_____

### *5.5 Key Terms*

The following table provides definitions for terms relevant to this document.

| Term | Definition |
|---|---|
| *App* | *Application* |
| *GUI* | *Graphical User Interface* |
| *MVVM* | *Model-View-ViewModel* |
| *XAML* | *eXtensible Application Markup Language* |
| *C#* | *Programming language used for most of the development* |
| *SQL Server* | *Database* |
| *SQLite* | *Database* |