

To what extent is designing and creating a social media app in Visual Studio difficult for a person with prior experience with coding in Python and SQL?

Introduction

In this project I will explore, in depth, my experience with creating a social media app in Visual Studio. My original idea was to create a video game in unity, however the idea was too close to what I wanted to do for my NEA for computer science, so instead I decided to look into other aspects of computer science. This led me to choosing to create an app. I then had to refine my idea to make sure it was niched down enough. I couldn't make a mobile game so I thought about other types of apps which led me to thinking about creating a social media app.

In this EPQ I will completely dissect the process of creation outlining what I found most and least difficult when creating this app and how I did it. I will begin by guiding you through how to install the many packages and software you need to create the app, links provided, and then show you how I set up my first project.

There are many reasons why this is worth investigating as a lot of factors will affect how difficult I find this. The biggest of these being the Programming language; unlike my past projects that are coded in python this will be coded in c#. The big difference between c# and python is the programming paradigm. Python supports all four programming paradigms, mainly using Imperative, Functional and Procedural; I had not used OOP(Object Oriented Programming) in python as there was no need in the previous projects I had completed. On the other hand, c# uses majoritively OOP, object oriented programming, which is one I am very inexperienced with and so it will make completing the EPQ much more difficult.

Another reason could be getting the software to work. Most software has default settings that are compatible with most uses of their software, however with some software, such as visual studio, they require additional settings and extensions that enable the required features for development.

Methodologies

For this project I will be using an agile methodology. How it works is that I will originally make one prototype. The prototype will contain all the basic functionality needed for my app to be usable without being over developed. The prototype will then be tested in order to make sure the functionality is up to standard. I will then evaluate the app on how it functions and what can be improved. The other prototypes will follow a similar template until I have reached a suitable final template. The agile methodology was created by Ken Schwaber and Jeff Sutherland in the spring of 2000 which was curated as a way of speeding up the process of software development {1}. The method is statistically proven to be most successful with a 42% success rate compared to other methodologies such as the waterfall method that has only a 14% success rate and is also used by a plethora of the leading tech companies e.g Apple, IBM and Microsoft.{2}

The final version will then be given to my friends and other fellow classmates to try and give feedback on.

Basis of evaluation

I am a grade 9 computer science student who only has past experience with 2 programming languages, Python and SQL. How difficult I find creating the app will heavily differ from someone who may have had experience with other languages such as c# or someone who has learnt programming to a higher degree than me, for example at university. To evaluate how well I've done I must base it off of certain criteria to give a more accurate assessment.

To evaluate the success of the app I will use these criteria:

- Did the app work as intended?
- Was the app easy to use?
- Does the app look good or professional?

My evaluation won't strictly be based on just these criteria and may reference a multitude of other points that determine how successful my app went; however these criteria will still be important as they provide a base that will be developed on. As Well as these criteria I will also use the testing from my classmates and get their point of views on the app in order to get a completely unbiased opinion of the app.

Installation

First key component to create this app is the development program also known as an IDE.

An IDE, or an Integrated Development Environment, is a software that enables a developer to create, compile and run code. In this case I will be using an IDE to create the code for my app, compile it and run it. There are features that I require from an IDE that are fundamental to an IDE. They must allow you to type in code and check it for any syntax errors (simple spelling and punctuation errors) and can also point out logic errors if needed that may break your program. They must be able to compile your code spotting any errors in terms of referencing (programs use libraries, stores of functions, to allow the program to use certain features for example the System library which if not referenced would cause the program to crash due to the inability to access the referenced functions). They must also have to have the ability to run the program, for example opening the app on a phone, or allowing you to play the game you created. Now there are many IDEs for C# and they all have their pros and cons, for example Rider.

Rider is an IDE developed by JetBrains, a PLC based in the Czech republic, which specialises in the development of tools for software developers. They created Rider in order to enable people to create applications and programs based on the .NET Framework and from looking at reviews of the IDE from G2.com we can see that It is generally a very good IDE providing users with excellent features such as “in-built package manager tools” which become really important when developing.

A competitor to Rider, and the more favourable one judging from review, is Visual Studio. Released in 1997 by Microsoft, Visual studio quickly became one of if not the most used IDE in the world with a 28% market share, 14% greater than the second place IDE. Although their market share is decreasing, it is not at an alarming rate with it only decreasing by 0.1% over the last year. In comparison, the second place IDE, also developed by Microsoft, Visual Studio Code has a 13.99% market share but has increased by 1.1% in the last year. Looking at reviews

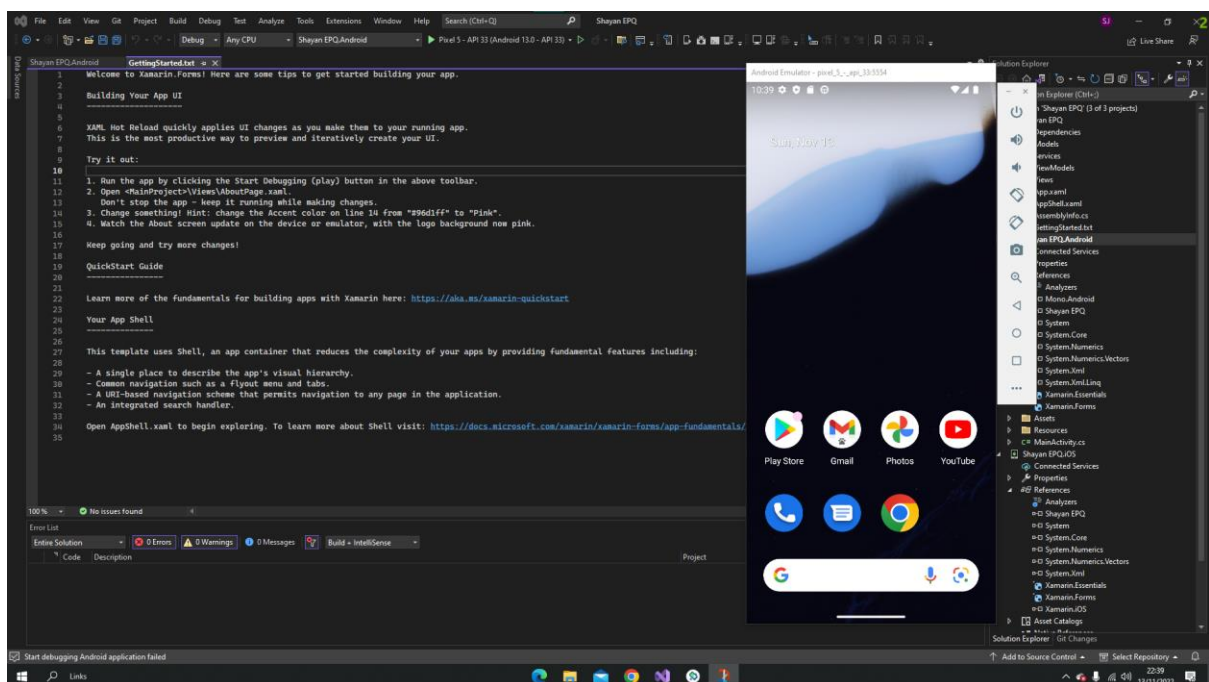
On top of being more favourable by most, Visual studio is much more advanced and versatile than rider allowing for a plethora of high level programming languages including some of the most popular and commonly found examples including python, html and c++ . This contrasts riders ability of only being able to code in .NET compatible coding languages such as Javascript and c#.

Due to rider's low user base there is an incredibly small number of reviews that can be found and this means that reviews can be slightly skewed. For example, using gartner.com we can find only 3 reviews submitted for Rider in contrast to visual studios 1235 ratings. This larger sample size means that the overall rating for visual studio is a lot more accurate than for rider meaning a fair comparison of the 2 is very difficult to be made.

Another comparison point is the price. Rider is a completely paid IDE meaning you must pay a monthly or yearly subscription to use it and although they provide a 30 day free trial, I would still have to pay as developing this app will almost certainly not be completed within that time frame. Visual Studio is completely free for the community version and a one time purchase of 573 for the professional version; I do not need the professional version however as everything I need is provided in the community version.

My choice of using VS was pretty much already decided before starting, as seen in my title, as it was an IDE I had used before when creating python programs and compared to Rider had a much larger community which actively provided support and produced mass content using the IDE.

Once I had downloaded visual studio from this website {3} I needed to download the right packages in order to be able to actually emulate and create my app. By following the video referenced here {4} I was able to download all required packages needed to create my app. Also, it guided me through how to set everything up in order to start developing which I found incredibly useful. Along with the basic setup, this video informs you on how to set up an emulated phone on your computer through visual studio, which will allow you to test your app on your pc.

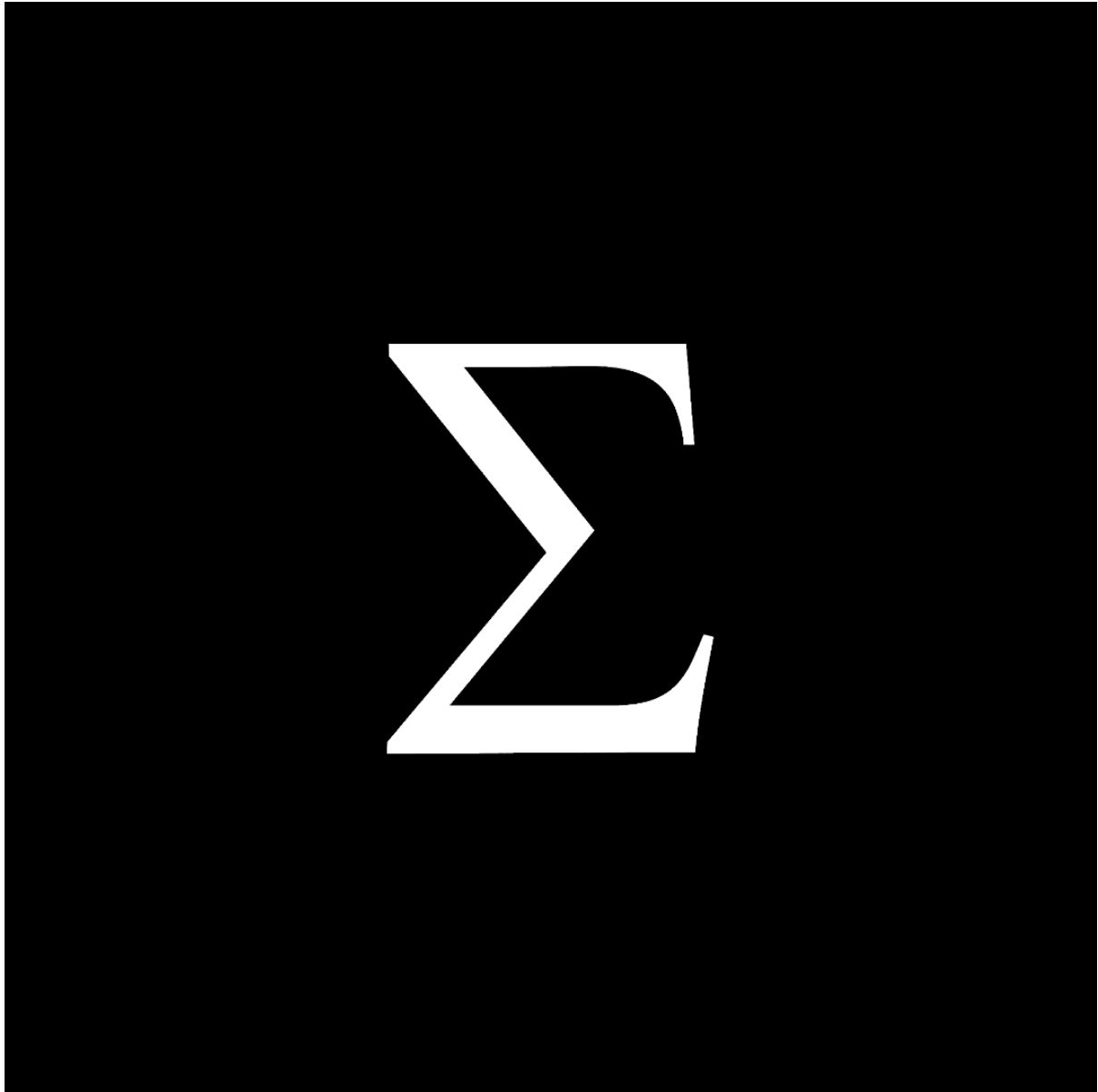


What you see above is the result of following the video along with a bit of tampering on my end. Everything worked as it should however the phone emulation was a problem. Using the emulated phone to try my app was taxing on my pc and causing it to run very slowly and so I turned to solutions, which led me to discovering the video referenced here {5}. The video is a guide on how to debug, run the app to identify and remove facts, on my physical phone rather than an emulated one which meant that my pc would not have to experience any cuts in performance.

Installation was not a difficult task and is by far the easiest thing to get working compared to other parts of the app that I developed.

The Design

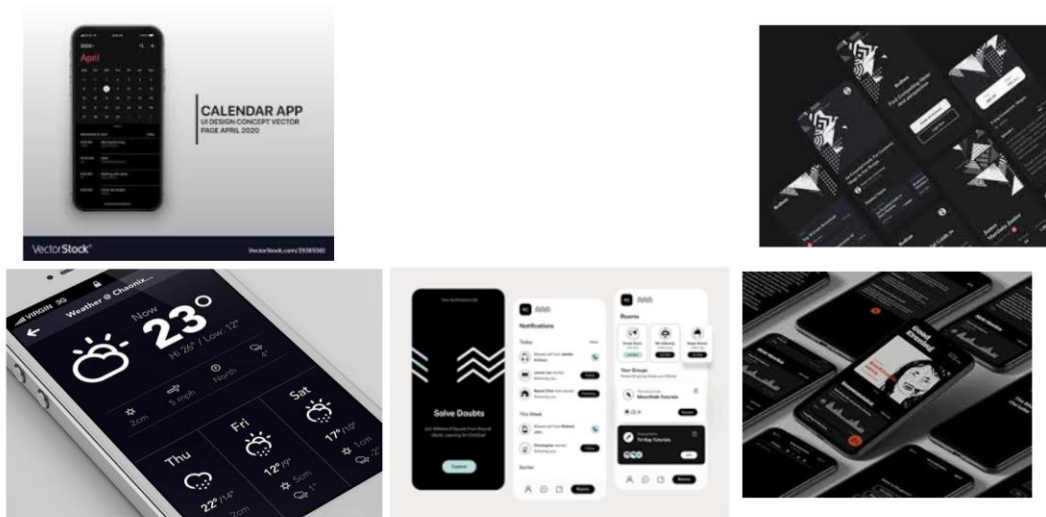
When you look at other brands, such as Tiktok or Instagram, they have logos that are iconic and easily recognisable. For my app I chose the greek letter sigma, seen in the image below. Originally the Sigma icon had no meaning as I thought it just looked nice, however as I began to develop, I thought about what social media really is. It is a means of bringing people together and so the letter sigma makes sense as it is used to denote the sum function, adding together



This logo is very simplistic, and is easily recognisable.

Theming wise, I used a predominantly black and white colour scheme, as this brings attention to the posts rather than the UI. It also provides a better experience when in use as the darker screen reduces the amount of eyestrain at night or in dark places.

Here is a mood board on how I want my app's UI to look like.



A few success criteria for the app's UI are that:

- The UI must look clean and not overcrowded
- It must be easy to navigate
- It should follow a specific style and stay consistent

I will test these criteria on my feedback that I receive from the testers and from my own judgement.

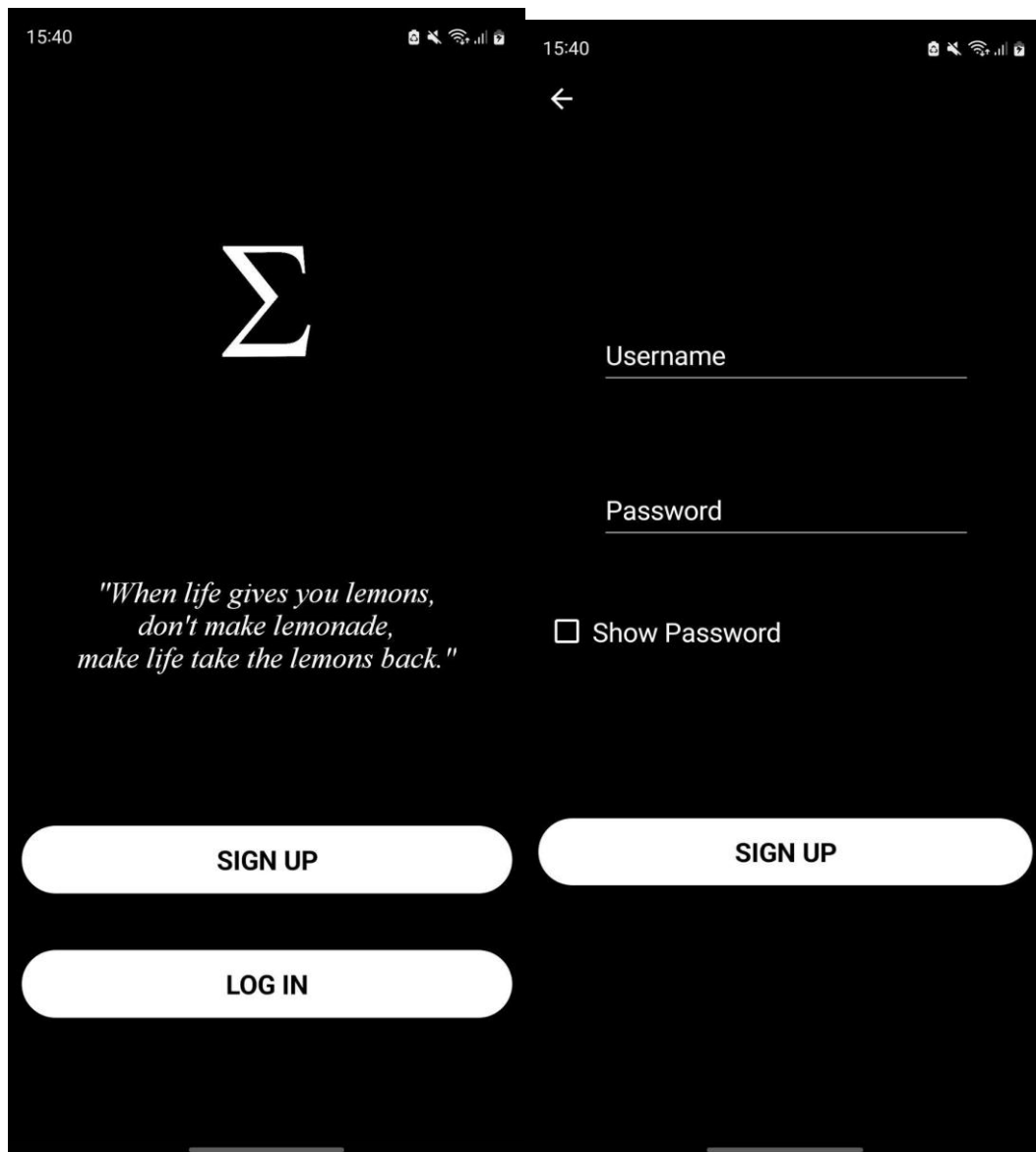
Development

Prototype 1

Login/ Register

Creating the Login and Register Page was very simple. Looking at most social media apps there is not much difference between how any of them organise them. All that is included is a username and password. For registration, the majority of apps tend to take more than just a username and password and would most likely include personal details such as Full names, date of birth and email address. I will not ask for any personal info as I am not a corporation that requires such details, additionally I believe users will find it more difficult to trust my app if they had to put personal information into it.

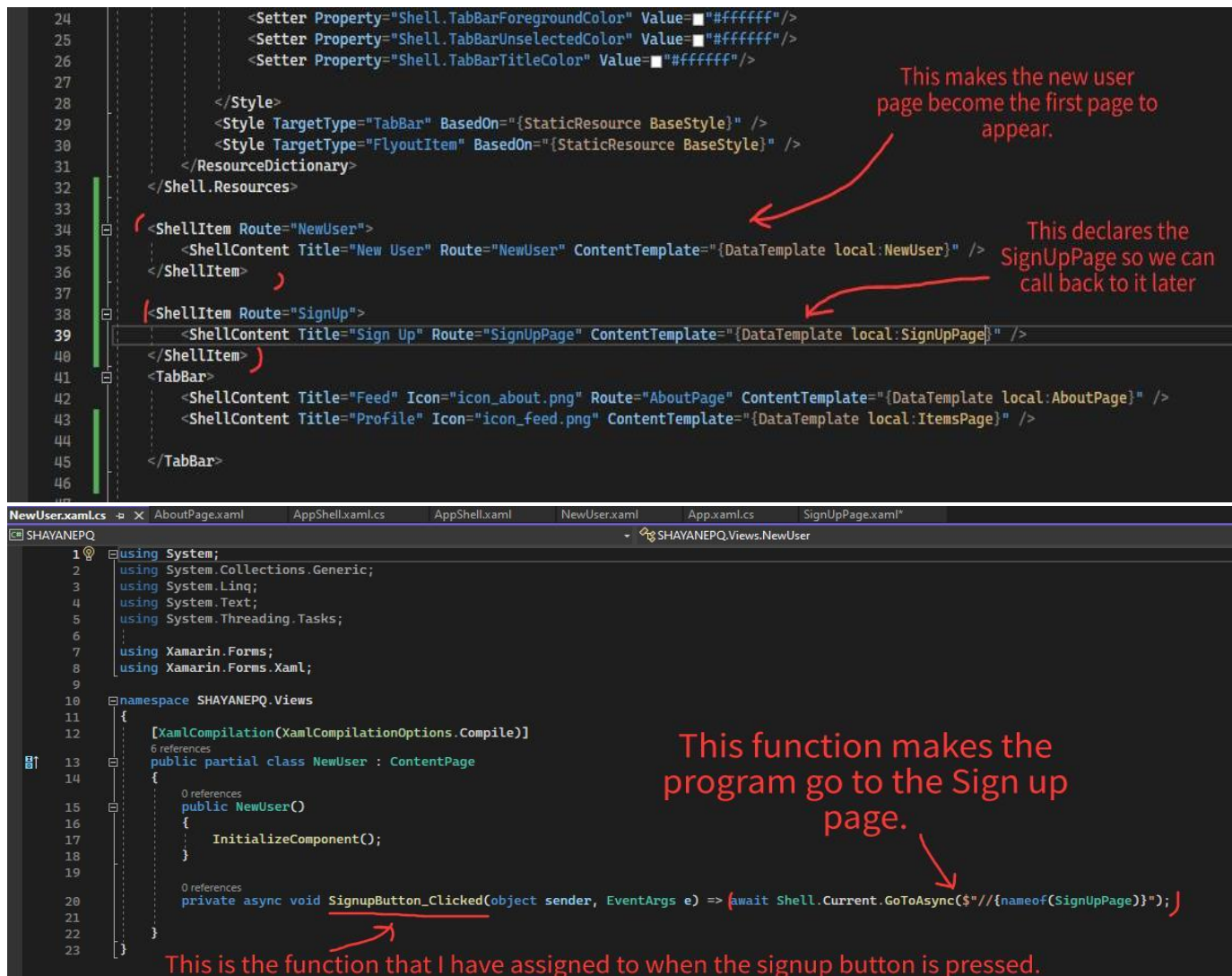
To create the page I researched into any premade login pages as it would allow me to save time for the more complex side of the app. I found an overabundance of videos explaining how to create login pages and so I looked through a few until I found one that looked professional. This was the video used. {6}



I followed the video and paid attention to the code used as it was important and would be used throughout the app. I used a lot of the template given but added my own changes ranging from the separate sign up and login page and other stylistic changes for example the colours.

This UI not only abides to all the criteria I set, but additionally, due to the amount of black colouring, reduces the amount of strain on the eyes and saves battery due to less pixels being on {7}. . However, the video did not show how to create a back end to the UI and so the buttons didn't do anything.

I needed my buttons to navigate the user from page to page and so it was crucial to the whole app that I would learn how to do it. This video {8} by James Montemagno, a Lead program manager at Microsoft who is well versed in Xamarin, guided me through shell navigation.



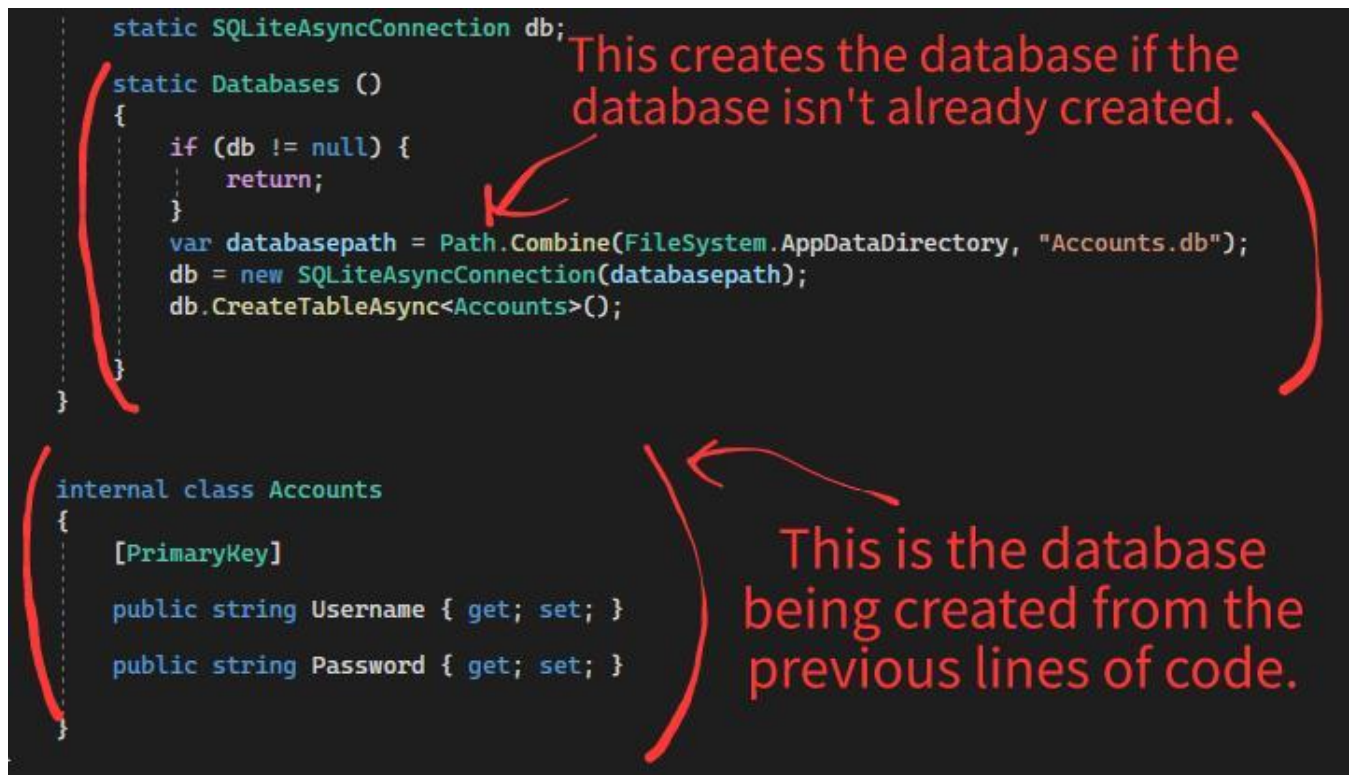
Now that I could get user details I needed to be able to store them

Databases

A database is defined as a structured set of data held in a computer, especially one that is accessible in various ways. To put it simply, It is an area to store data. I need a database to store account info so that accounts can be saved permanently in case someone deletes the app and re downloads it and wants to log back in. Additionally, the database will hold every post and comment on the site. These will be organised using the primary key, an attribute attached to a

record which can be used for searching, for each post and comment as well as secondary keys for grouping comments to posts.

After watching this video {9} I was able to find out how to create databases in visual studio as well as manage them. Following through the video I was able to learn all about databases in visual studio including how to create databases including the tables and columns they are made up of.



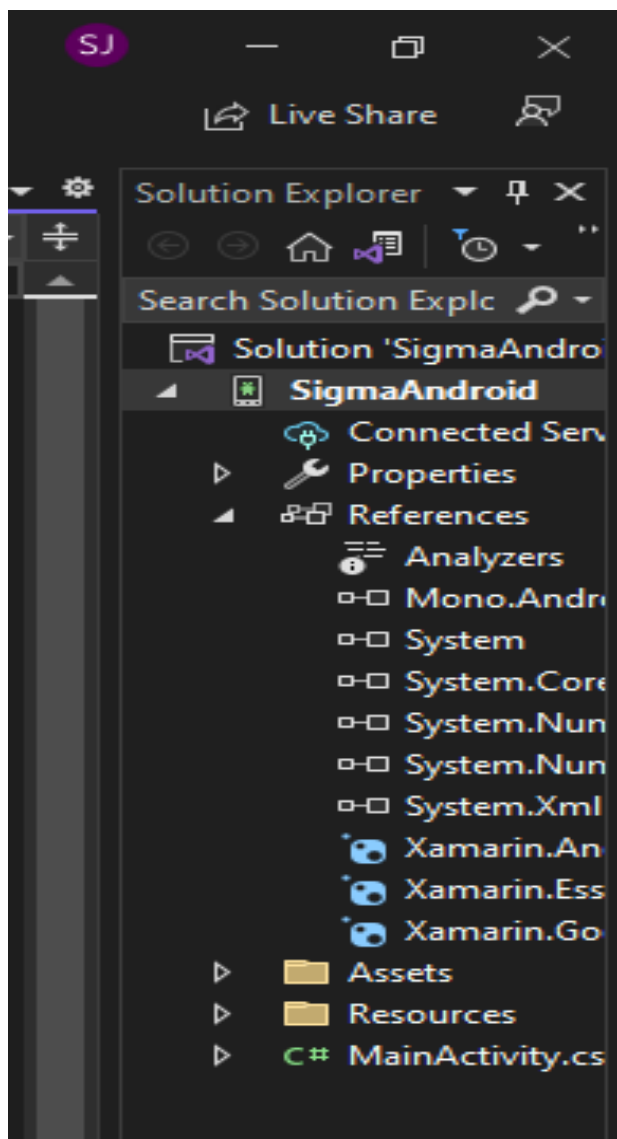
The image above shows the code used in my app to create a database with the table "Accounts" and columns "Username" and "Password".

2 weeks after I had managed to create a database, I had encountered a logical error in my program which was disallowing my app from functioning as it should. The databases I was creating were purely local to the app; this means that each and every version of the app that is downloaded by a user will have their own local version of the database. This will not work as for the sharing of posts to work, there needs to be one sole database that has shared access between each user so that any user can access other users posts. Additionally, a local database would mean that when the app is deleted, so too is the database which is counterintuitive as it goes against the purpose of the login database I stated earlier.

This evaluation has led me to the realisation that this should be the point at which the first prototype should end. I managed to get the ui and button functionality working as well as the shell page navigation which are all fundamental parts of my app. What I have not done so well is my database and so for my next prototype I will work to improve where this first prototype went wrong.

Prototype 2

From the start of this new prototype I have decided to change a few things. Firstly, the original app I was creating before was made for both android and IOS, however due to the inability to get apps on IOS, without uploading to the App store, I have decided to remove it altogether.



The second change is that instead of creating the login system, I will start developing the main part of the app first. This is due to the fact that even if I am not able to complete the login system before the deadline I could still make the app usable as an anonymous social media app where everyone gets an auto generated username e.g Guest 1234.

The third change is that there is now a Devlog ,a document designed to monitor updates made to a piece of software or program, for prototype 2.

As of the 29/03/2023, every change made to my program will be in the DevLog with explanations for each of the changes in there. Any changes that I would like to explain in detail will be discussed in this document. This ensures that you as the reader will know when progress was made and what progress was made which is critical to my initial question of how difficult creating a social media app will be.

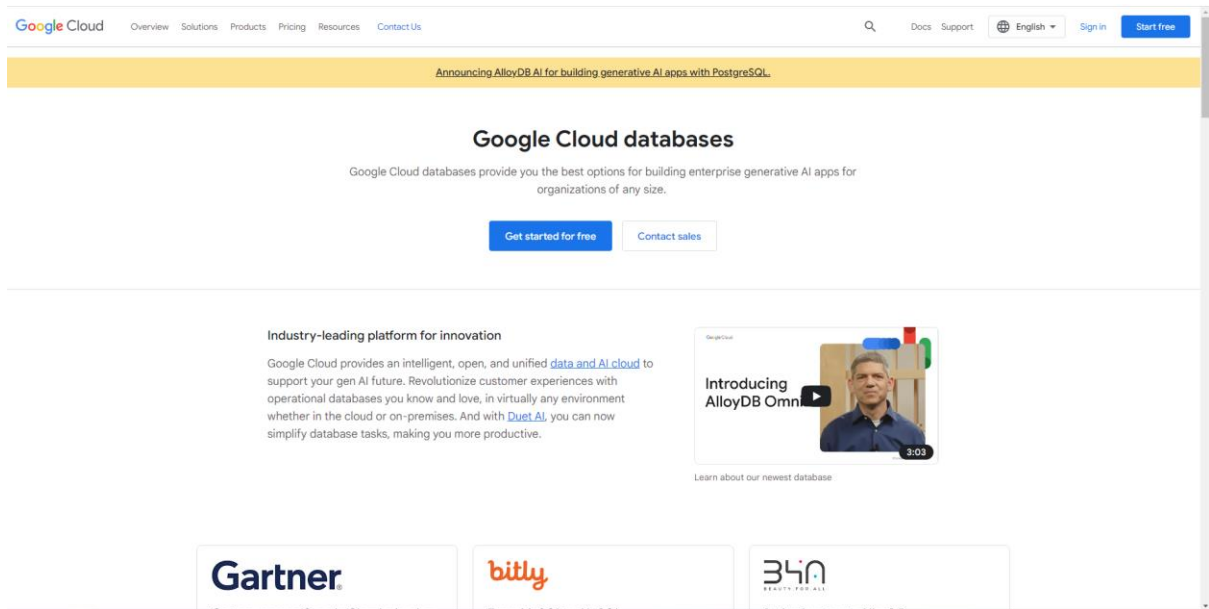
Routing

Originally in prototype 1 I used a mix of shell routing and navigation however navigation pages had seemed to become problematic and so in prototype 2 I have phased it out completely. Routing works similar to navigation where it navigates through shell items to get to different pages, however with routing there is no back button. This will return control to the programmer on how users navigate through the app restricting the user. Although restricting may seem problematic it actually will help stop users from encountering errors when using the app.

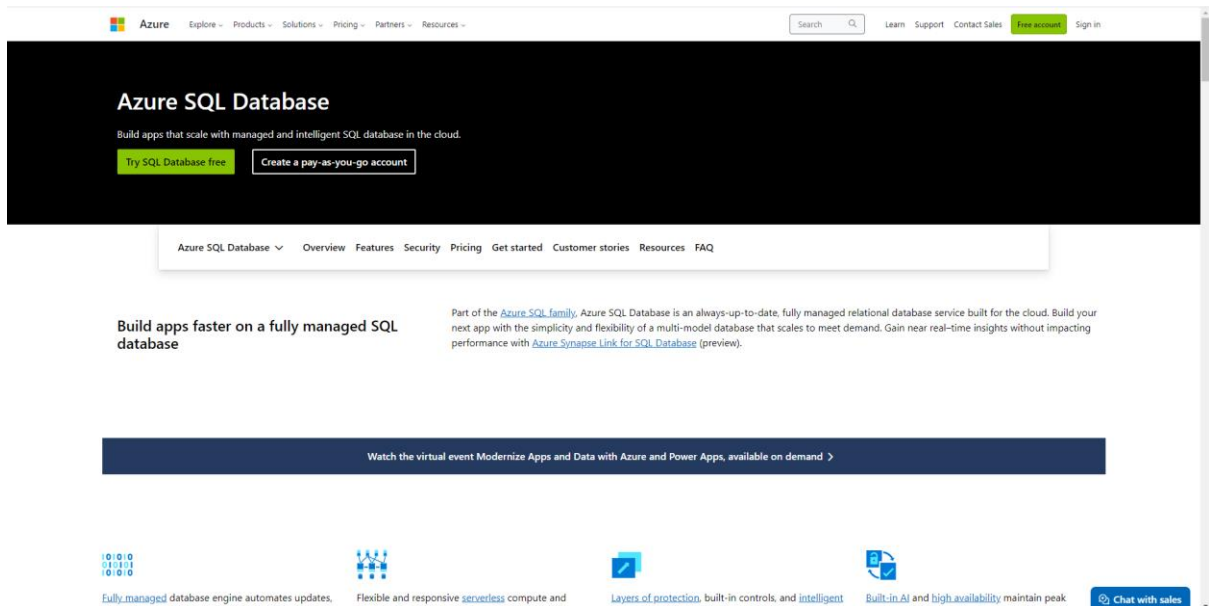
Databases

As aforementioned, the fatal flaw of my first prototype database was the area it was stored. A locally stored database is not compatible with my app and how I wanted it to function and so it was imperative that it be changed.

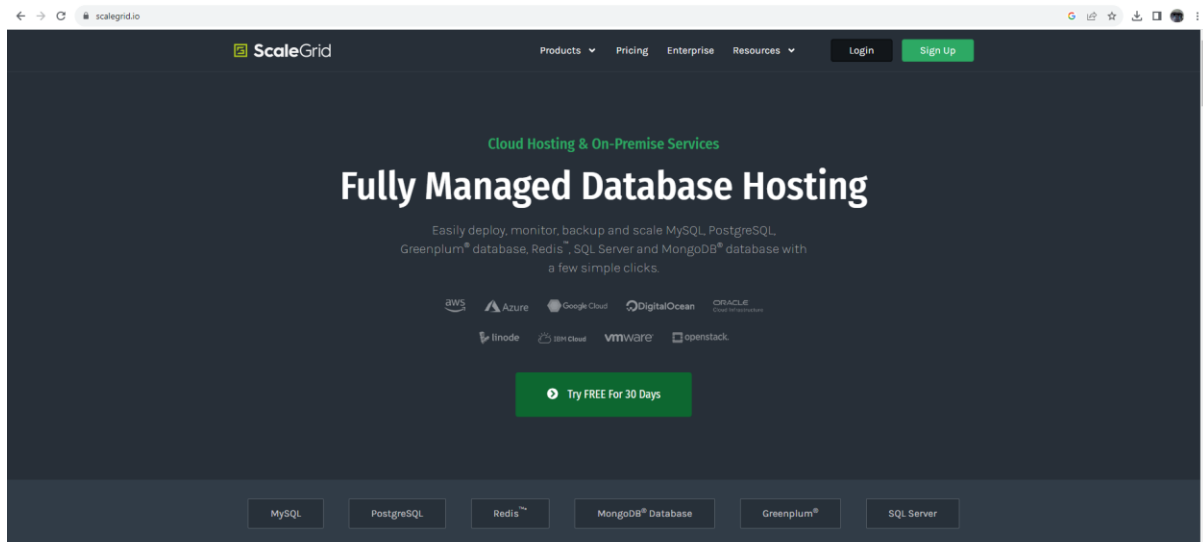
To counteract this problem I came to the conclusion that I will need to somehow run the database as a server through the internet in order to be able to access and make changes anywhere. I first thought about running the database of the cloud and decided to do some research on how I could do that. There were many options to run the database of the cloud, for example Oracle {10}, Google {11} or Microsoft {12}. However, A big problem with this method is that I would have to pay monthly subscriptions in order to keep these running and that would not be within my budget.



(Google's Cloud service database page)



(Microsoft's Cloud service, Azure, database page)



Database-as-a-Service Benefits
(ScaleGrid, an oracle based database hosting service)

	CORE (v)	RAM (GB)	STORAGE (GB)	MONTHLY PRICE (USD)
MICRO Standard.EA.Flex (1/8 baseline utilization)	1	2	50 IOPS: 100	\$35
SMALL Standard.EA.Flex (1/2 baseline utilization)	2	4	50 IOPS: 3000	\$60
MEDIUM Standard.EA.Flex	1	8	60 IOPS: 3600	\$104
LARGE Standard.EA.Flex	2	16	120 IOPS: 9000	\$208
XLARGE Standard.EA.Flex	2	30	240 IOPS: 18000	\$348
X2XLARGE Standard.EA.Flex	4	60	480 IOPS: 35000	\$672
X4XLARGE Standard.EA.Flex	8	120	700 IOPS: 35000	\$1456
VERY XLARGE			800	

Database: MySQL, PostgreSQL, MongoDB® Database, Redis™

Cloud: AWS, AWS High Performance, Azure, DigitalOcean, GCP, Linode, Oracle Cloud

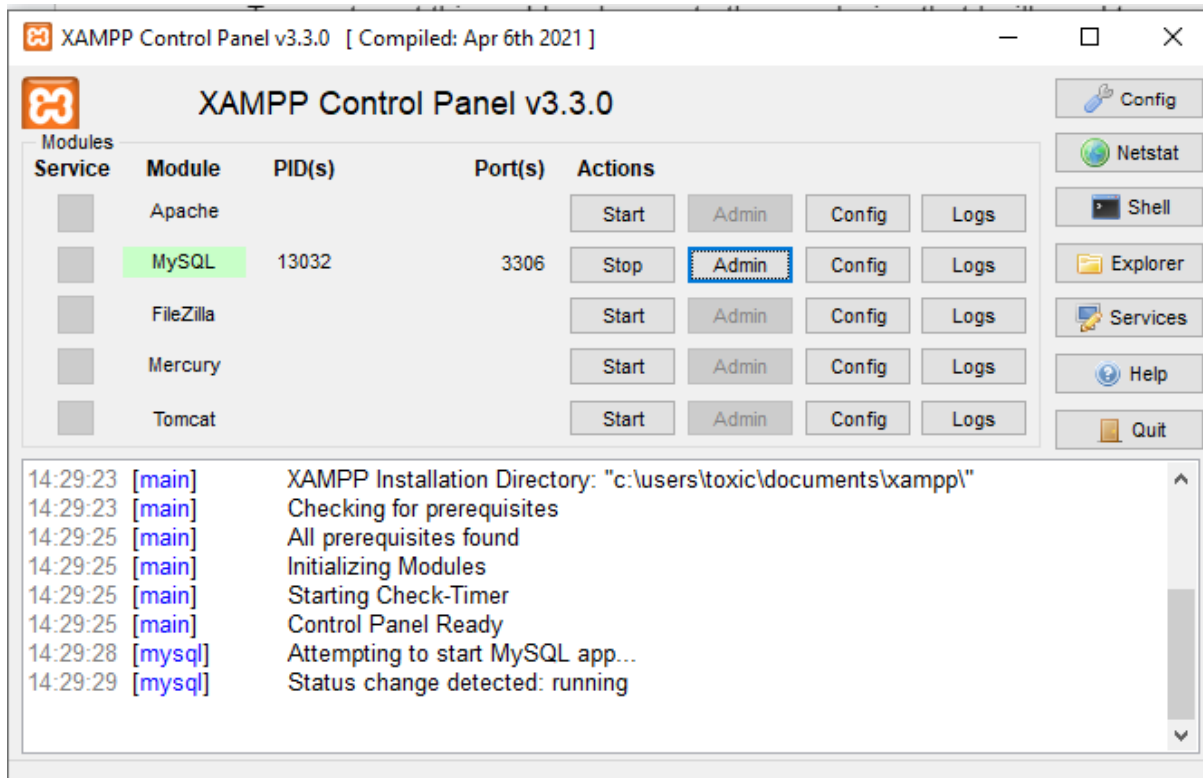
Replicas: Standalone, Source-Replica-Replica

\$35/month (Minimum 120 hours) [Start your free Month](#)

(ScaleGrid Pricing)

Looking at options, I came to the realisation that none of the cloud options would be suitable.

Servers are essentially computers whose sole purpose is to allow the transfer of data to and from them and that made me think about my own computer. I researched ways of turning my computer into a server which led me to XAMPP. XAMPP is a free program that can be downloaded from this site {13}. It is very useful as it provides you with the ability to run an sql server off of your pc. It also allows you to run other types of servers such as apache for websites and other servers however for now I will only be using the SQL server.

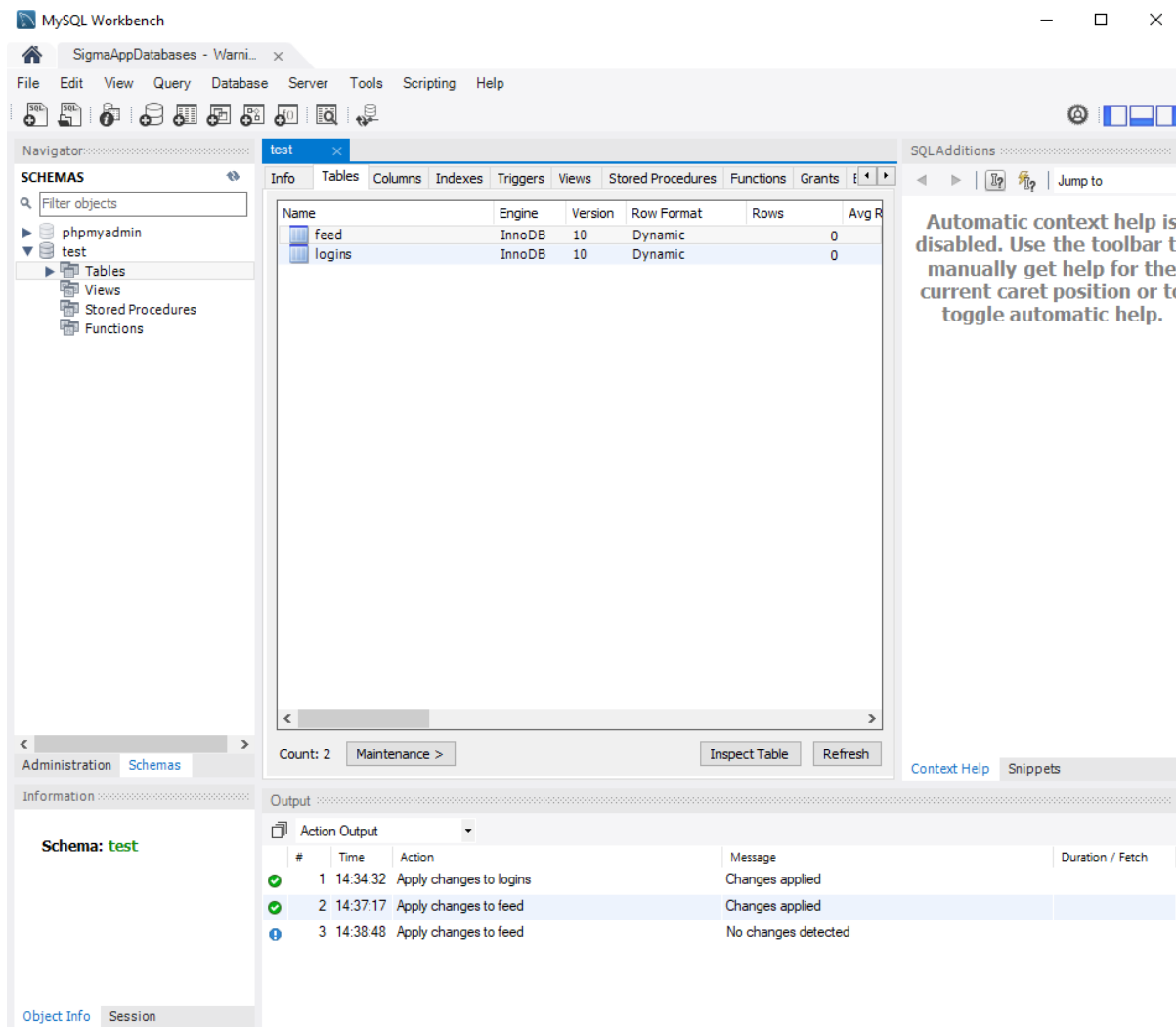


(XAMPPs default control panel)

This is what the user interface looks like with the mysql server running and after following a tutorial on youtube {14} I was able to set up the database so that it was fit for purpose.

With the Sql server running, I needed to access my database so that I could create my tables and columns. For this I used My Sql Workbench, a free software that allows you to connect to Sql servers in order to access and manage databases. I downloaded it here {15}.

With this I was able to create 2 tables, one for logins and one for posts.



(MySQL Workbench default UI)

With the database completed I diverted my attention to getting it connected to my app. This video {16} was the best and most valid way I could find to connect the two and so I did.

Seeing as the database is the most integral part of the app, once completed and functional the rest of the app should be relatively easy.

Progress Loss

On the 2nd of may I had a problem with my pc. It kept crashing and with no way of being able to recover it I was forced into a complete wipe. This meant I had lost all of my data on prototype 2. Deciding not to look at this through a negative view, It seemed like I should take advantage of this opportunity to start a new prototype. Most of the time before the wipe had been spent working on the database, however I never managed to get it working as there were still many complications I was experiencing and I could not determine whether it was a fault of my code or the software itself.

Back Ups

The progress loss had pointed out a flaw in my work. Apart from being stored locally, there was no other version being saved elsewhere. This meant that my progress was constantly at risk of being lost and to counteract that I began to look into saving backups somewhere other than my computer's local data storage. For the App itself I researched whether there were any built in methods of backing up my app in visual studio. I found out there were. This website informed me on how I could upload my app straight to github through an extension {17}. I managed to successfully upload my whole app onto github which means that from now on I will have a backup online that will prevent further progress loss

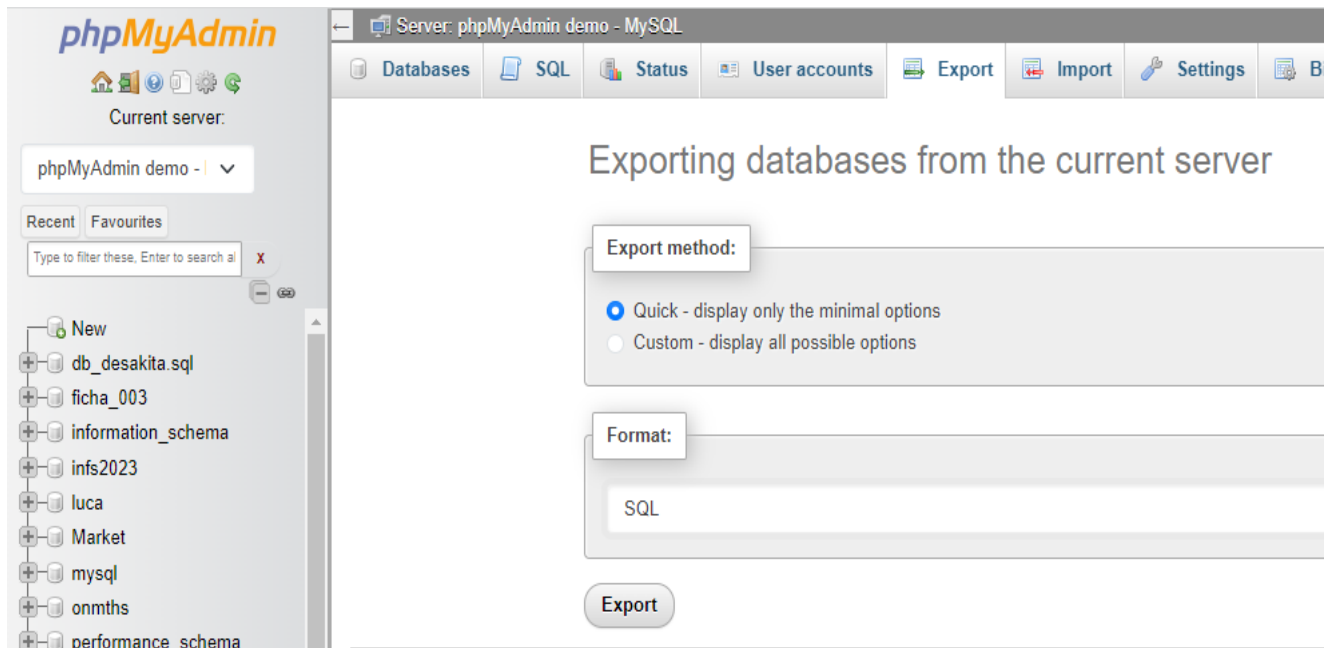
The screenshot shows the GitHub interface for a repository named 'Sigma'. The repository is public and has a 'master' branch with 1 branch and 1 tag. The file structure is as follows:

File/Folder	Description	Last Commit
Assets	Add project files.	last week
Properties	Add project files.	last week
Resources	Add project files.	last week
.gitattributes	Add .gitattributes and .gitignore.	last week
.gitignore	Add .gitattributes and .gitignore.	last week
MainActivity.cs	Add project files.	last week
Sigma.csproj	Add project files.	last week
Sigma.sln	Add project files.	last week

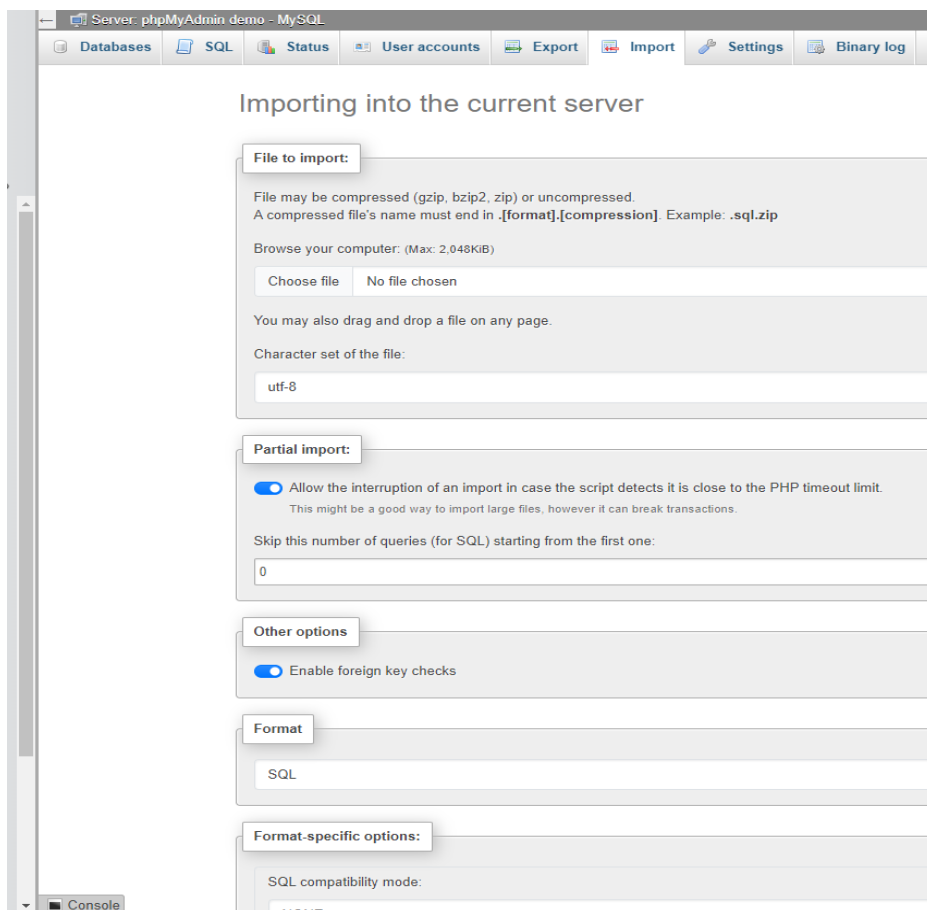
At the bottom, there is a prompt to 'Add a README' to help people understand the project.

(Sigma Github Repository)

To Backup the database, I used the built in database manager PhPMyAdmin that provides a way of exporting databases into a sql file that I can later import back in incase of data loss.



(PhPMysql database export)

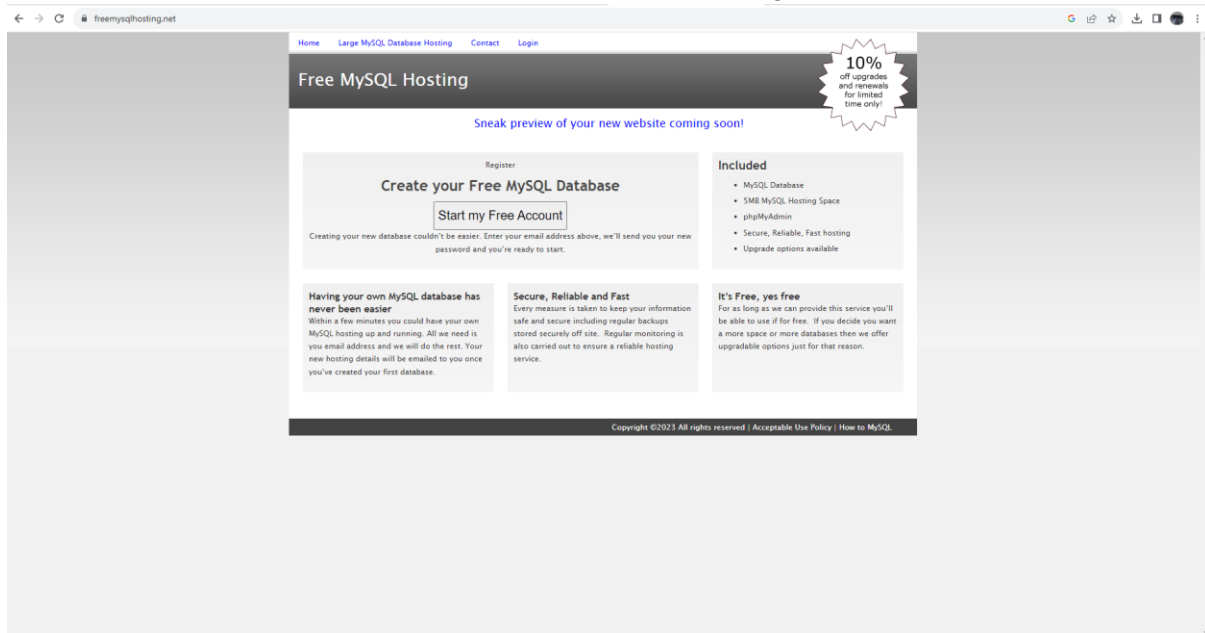


(PhPMysql database import)

Prototype 3

The first thing I would do for this prototype which would build on the previous one would be to determine where the problems were and how they could be fixed. I had gone through numerous different videos and websites explaining how to allow connection to the database remotely which was pivotal for the premise of a social media app. (references to XAMPP Binding website and videos). I spent time testing every part of my code, from the client side connection from my application to the server side connection from the database. Around Mid August, I came to the conclusion after vigorous testing that the problem was server side. I had spent a lot of time first testing the code, interchanging different libraries and different functions in order to try to get a connection to work. I then turned to the server side. I researched the different config settings and what each meant so I would have a better understanding of how my server was functioning. I had found a setting that I thought would fix everything, the bind address; the bind address determines what IPs can connect, "bind", to the server and after looking at multiple different values I could use to open up the access to the server and testing each value, I found that none worked. I decided that it was best to look into other software available to me to use.

I began looking into free online database hosting and came across the website <https://www.freemysqlhosting.net/>. After creating an account and connecting to it through my app, I had confirmation that the previous problem was in fact server side. However, this solution was only temporary as I soon realised that there was an extremely small limit on the amount of data that it could hold, 5mb, due to its free pricing.



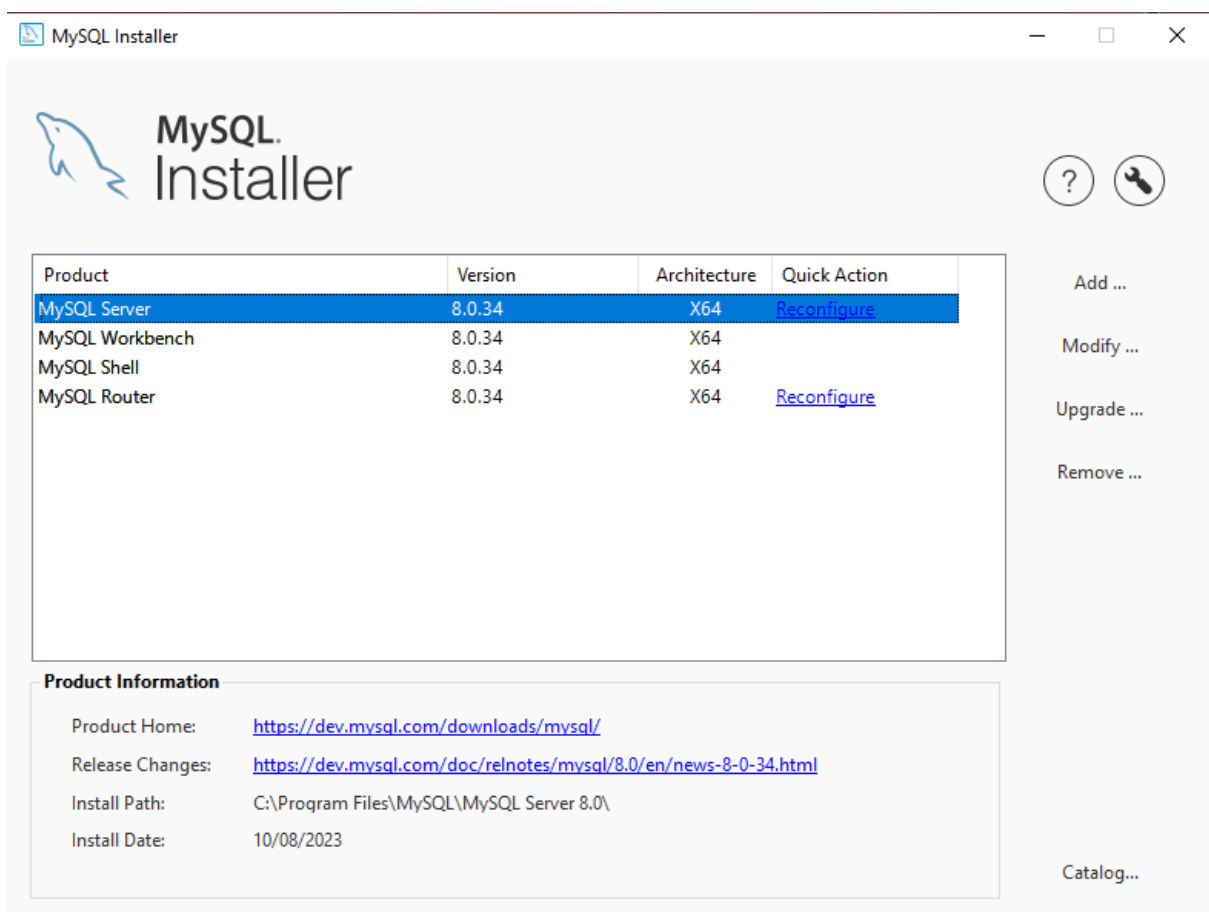
(Free MySQL Hosting website)




Included

- MySQL Database
- 5MB MySQL Hosting Space
- phpMyAdmin
- Secure, Reliable, Fast hosting
- Upgrade options available

(5mb limit)

After more searching I reverted back to my original idea of running a mysql server off of my pc which is where I ended up finding this video that directed me through creating a server on mysql installer, previously used to download mysql workbench {19}. I additionally had to download a new plugin rather than mysql.data as it was not functioning in my app. I found a new plugin called mysql connector and after using it it worked, however, there are still no remote connections.



	MySQLConnector by Bradley Grainger A truly async MySQL ADO.NET provider, supporting MySQL Server, MariaDB, Percona Server, Amazon Aurora, Azure Database for MySQL and more.	2.2.7
	NETStandard.Library by Microsoft A set of standard .NET APIs that are prescribed to be used and supported together. 18a36291e48808fa7ef2d00a764ceb1ec95645a5	2.0.3
	NuGet.Frameworks by Microsoft	6.6.1

```

1
2  using MySqlConnection;
3  using Sigma.DataModels;
4  using System;
5  using System.Collections.Generic;
6  using System.IO;
7  using Xamarin.Forms;
8  using Xamarin.Forms.Xaml;
9
10 namespace Sigma.Views
11 {

```

Remote connections

Fortunately for me, I was able to get remote connections working on the server. To do so I first had to open the server port to the internet. Port forwarding, a method of logging into your router settings and opening the port, allowed me to get remote connections however there was still a client side problem, the IP. Originally I used the local IP that every device on my router recognises.

```

var connstring = "SERVER=192.168.1.32;UID=admin;DATABASE=sigma;PASSWORD=sigma;PORT=3306";
var connection = new MySqlConnection(connstring);
connection.Open();
string posts = "SELECT PostID, PostTitle, PostBody, usernameDB, pfp FROM posts ";
MySqlCommand postcmd = new MySqlCommand(posts, connection);
MySqlDataReader postreader = postcmd.ExecuteReader();

```

However, for remote connections from other networks this IP would not work and so the public IP of my router was needed. After using the new public IP, I was able to connect through any wifi I wanted.

```

var connstring = "SERVER=2.101.11.158;UID=admin;DATABASE=sigma;PASSWORD=sigma;PORT=3306";
var connection = new MySqlConnection(connstring);
connection.Open();
string posts = "SELECT PostID, PostTitle, PostBody, usernameDB, pfp FROM posts ";
MySqlCommand postcmd = new MySqlCommand(posts, connection);
MySqlDataReader postreader = postcmd.ExecuteReader();

```

File Handling

File handling was quite a lot more complex than I originally thought it would be. In my past experiences with python, file handling has been relatively simple. You simply copy the path of the text file from your pc and use that as the location of access for the file.

In Xamarin however, you can't simply just copy the path of the file. Local files for Xamarin have to be created within the code rather than creating them yourself through Xamarin.Essentials. An example code of a local Xamarin file path can be seen below.

```
string postfile = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "post.txt");
string usernamefile = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "username.txt");
```

I managed to do this using the system.IO package that came built in in visual studio. W3 schools is an excellent resource for programmers as it not only gives tutorials for a variety of programming languages but also gives intractable examples that you can experiment with to get used to the code {20}. Throughout the EPQ, there have been numerous times in which I needed to use it and it always managed to work for me. In every section of the references there will probably be at least one W3 schools tutorial cited, which further emphasises how important of a resource it was.

```
if (File.Exists(postfile))
{
    File.WriteAllText(postfile, string.Empty);
}
else
{
    File.CreateText(postfile);
}
```

Somewhere near the end of the EPQ I experienced some problems with file handling. The app was crashing a lot and researching the errors led me to believe that there was a problem when reading or writing to files. Specifically, the sharing of paths and that meant I needed to change things. This is when I researched further into file handling on Xamarin and came across StreamWriter and StreamReader. A beneficial upgrade, it enabled me to push past the path sharing problems and brought back functionality to the app.

```
StreamWriter postsave = new StreamWriter(postfile, false);
postsave.Write(post);
postsave.Close();
await Shell.Current.GoToAsync(nameof(ProfileCheck));
```

```
StreamReader pfpsave = new StreamReader(pfpfile);  
string pfppath = pfpsave.ReadLine();  
string pfppfull = pfppath + ".png";
```

TESTING

The stakeholders for my social media app are teenagers around my age, so it was the most sensible option to extract a focus group from my classmates that would tryout and give feedback on my app. The four individuals that I chose are avid social media users and have been for a considerably large amount of time.

I decided that just testing with my classmates was not sufficient and I would additionally need a more qualified person to test my app. Asking through my family, I found out that one of my uncles, Zeeshan Janjua, had actually got a degree in multimedia from Brunel University. I contacted him explaining my app and organising a day to test.

On the 27/09/23 I met with my uncle and let him use the app. We met up in an outdoor public communal area and we sat down to discuss and test the app. From the onset, he had straight away noticed a flaw in my UI design that I had not noticed. Due to us sitting outside, the sun was causing the screen to glare. This meant that the icons at the bottom of the app that were of a grey colour, seemed invisible in front of the black background behind. I had managed to overlook this due to the fact that pretty much every time I had tested my app, it had been in optimal lighting conditions indoors and with my phone's brightness on full.

After we had moved to a location in the shade, he continued to use the app and test it out and had majoritively positive remarks on my app. He complimented the colour theme of my app, comparing it to apps such as Instagram and YouTube that have opted to create dark mode themes for apps; however noted that the colour scheme also was slightly lacking in certain areas. For example, in the display alerts, seen below, that I used to notify the user that their post had been posted he notes that the button to remove notification box was black and due to the background being a dark grey, it was difficult to see; he also mentioned how it looked a bit off compared to the rest of the UI due to the grey used being a seemingly random compared to the shades used throughout the rest of the UI. Overall for the UI, he said that there could be an improvement in terms of colour. Adding buttons or text with a vibrant colour to them will make them pop, leading to the user to not only notice them but make them more inclined to read or use them.

Another complaint for the app was crashing. For the most part, I had managed to remove most areas where the app would crash but there were still areas where it would crash. When trying to log in, my uncle had entered his details and after pressing the login button, was sent straight out of the app and onto my phone's home page meaning it had crashed. When he opened the app again it said he had logged in and so we moved on however he still noted that as something to look into.

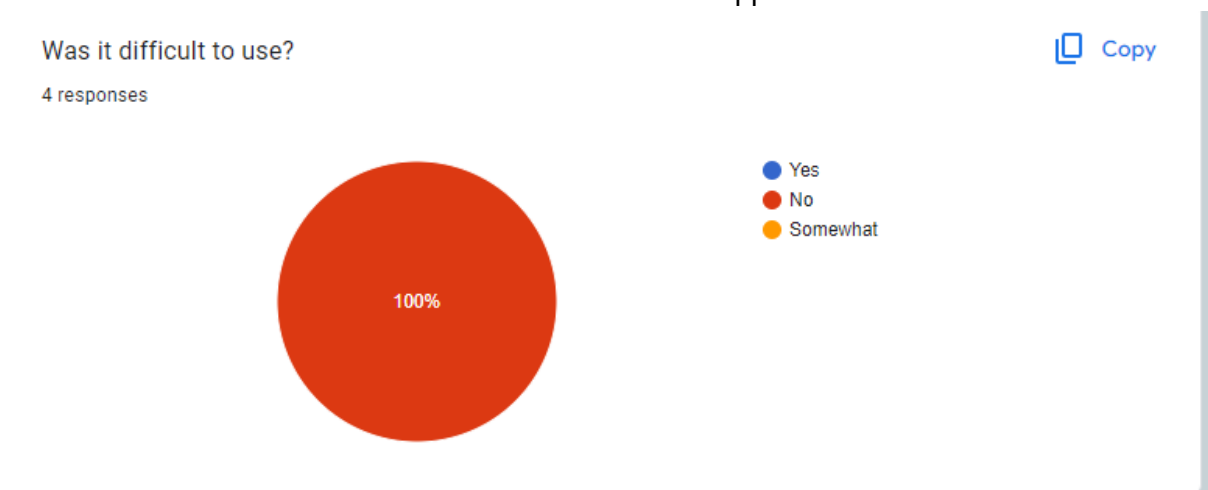
Moving onto my classmates' responses, I had set up a Google Forms questionnaire which asked a few questions that would give me an insight into how successful my app was.

The first question asked the tester what their first impression of the app was. The majority of the response seemed positive with people talking about how the UI "Simple" and "Professional" and allround the app ran smoothly providing an enjoyable experience to the user.

The second question asked about what the tester liked about the app which featured similar responses to the first question with the majority of responses complimenting the design choice of the app regarding its theme and formatting. Additional comments were made by testers such as BS who found that the messaging speed was quick and enabled him full use of his keyboard, not limiting him to just letters and numbers. Another tester, Mihai enjoyed the autogenerated profile pics that were given to users as well as the ability to comment on posts.

The next question asked the tester about what they disliked when using the app. Here the main problem seemed to be the crashing which we had seen beforehand when testing with my uncle. This was mainly found within the login system however the crashing was only occasionally for most people and still allowed to use the app for the most part. Another problem that one of the testers experienced was a formatting error. After downloading the app, MN had not seen the app as he had expected to see it. Text was in the wrong places and would sometimes be colliding with other UI objects such as profile pictures which although did not stop him from being able to use the app, still made the app quite unpleasant to use.

For the next question I had made a multiple choice question on how difficult the app was. The answer was a unanimous no as no one found the app difficult at all to use.



My next question asked the tester what improvements could be made to the app. The first response I got gave a very valid point about the feed page on my app. Unlike most social media apps that display the newest posts first, my app seemed to have the oldest first and the newest posts all the way at the bottom. This was bothersome for the user as they had to scroll all the way to the bottom to see the newest posts. Another common remark I received

was that the UI colours, despite being very good, still lacked a bit of colour. This was also seen in the interview with my uncle and the addition of some slight colour would be quite beneficial to the look of the app. Apart from that the only other improvement I got was to add private messaging to the app which can be found in most modern apps nowadays. This feature was originally thought about when creating the app but was never implemented due to time constraints.

The final question was yet again a multiple choice which got the testers overall opinion of the app. The majority of people had an overall good perception of the app with a 75% good rating. The other 15% was poor.

EVALUATION

Making an App is difficult. When I say difficult I mean it; it's not something you can just start doing knowing it will be done. It's a long process of learning and implementing constantly in a cycle of development. Throughout this EPQ I have experienced an abundance of emotions that have had a heavy toll on my mental health, mainly stress. The constant on edge feeling of uncertainty on whether you will actually complete the app, or end up failing, struggling to troubleshoot the numerous errors that you encounter. At the same time, a feeling of triumph consumes you whenever you manage to push past these errors; an indication of progress. The EPQ has taught me about the reality of software engineering. Before going into this project, I had some presumptions on what it would be like; I believed it would be simple and quick. My original plan of completing the EPQ before April which then seemed very doable, now seems like a joke.

If I were to restart this EPQ from the beginning there would be many differences in how I would conduct things whether that be the order in which things are completed or how I completed things.

A big mistake I made during the creation of this app was not researching everything beforehand. My way of researching involved only researching when a certain feature was needed. This was ineffective as I would come to find out later a good example being databases. When it first came time to add databases to my app I did not look at my app as a whole picture and rather at the specific thing I was trying to do at the time. This caused me to only research ways in which I could create a database rather than what would actually be needed, which was a database server. If I had done all my research beforehand I would have realised that in order for the app to be functional, the database could not be locally stored on the app and I would have begun research into database servers much earlier.

On the topic of the database, I believe that if instead of starting with creating the ui and all the easy visual aspects of my app, I had begun by testing and ensuring that all the functionality of my app would work ie having a working database, having a working feed, being able to add and remove from the database. I could've had a working version of the app much closer to the time in which I had actually planned to finish development of the app.

I would also change the UI slightly as evident by testing it could have benefited with some colour and so for my design stage I would have included that. My design section could have been improved quite a bit in general as it lacked much detail. Maybe going through similar social media apps and analysing why their UI works would have given me a better understanding of what the UI should look like and therefore provided a better base for my app's UI.

During the EPQ I had been making updates to my production log. Near the beginning of the EPQ I had a workshop with Dr Victoria Yuskaitis where I had a one on one session discussing my EPQ topic. The workshop was helpful and I had received feedback which was mentioned in my production log. Also in my production log, were some points of improvement that I could do which would improve my idea. In my production log I had responded to her feedback of niching my app down by adding a tag system where each person can add tags to their profile and then they would see posts with the same tags. Unfortunately, because of the difficulty of this task, as well as the time restraints I was given, I was not able to add this system to my app. I think this would be a good feature that would improve my app and if I were to redo my EPQ I would definitely add it.

Overall however, I believe that the app was a success. Going back to the success criteria I set at the start of this EPQ we can see that everyone of them was met:

- Did the app work as intended? Yes. All features that were originally intended on being implemented, were implemented and did function as required. This cannot only be proved by the video evidence provided but also from the testing I and the testers did.
- Was the app easy to use? Yes. From the testing seen before we can see that everyone agreed that the app was easy to use and that they experienced no difficulty navigating through the various pages of the app.
- Does the app look good or professional? Yes. Again, as seen through testing, the app does meet this criteria and although there could be slight improvement, the overall consensus was that it looked both good and "professional"

Success Criteria

The following section will be a collection of features that I have implemented into my app as well as the medium in which I will prove their functionality. All images and videos will be provided along with the actual EPQ, the artefact file and the EPQ DevLog. Proof Files can be found in the Success Criteria Folder.

Feature	How it will be proved	Proof file name
Splash screen functionality	Screen recording	Splash Screen.mp4
Page switching functionality	Screen recording	Page Switching.mp4
Feed functionality as well as feed refresh functionality	Screen recording	Feed Page and Refresh.mp4
Search page functionality	Screen recording	Search Page.mp4
Comment page and comment posting functionality	Screen recording	Comment Page.mp4
New user page, Login Page and Sign in page functionality	Screen recording	Login and sign in functionality.mp4
Profile page functionality, including pfp, feed, and logout functionality	Screen recording	Profile Page.mp4
Create page functionality	Screen recording	Post creation and deleting.mp4

CONCLUSION

Going back to the original question at hand, this EPQ was completed in the attempt to determine how difficult it is to create a social media app. To conclude, for the duration of this EPQ I have had to go through numerous prototypes, each with their own problems which I had to research and solve. In the beginning, it was relatively easy making the app; the code was basic as I was only creating UI and basic button functionality, I experienced close to no problems within my code, and problems that did occur were easy to find solutions to and fix a lot of the time being able to solve them without the need for looking for solutions online and rather solving them myself.

Progressing through the development process, I saw the app get more and more difficult to work on, at some points getting into stand stills where I would be stuck doing months of research, trying to solve a problem, testing every part of the problem to make sure I was doing everything right in order to find out where exactly the problem was and how to fix it. Especially during the middle of my development, around the time when I was trying to create a working database server, I was stopped altogether from developing the app itself and was forced to research every way my server could be failing, changing config settings, network router settings etc. That was definitely the most difficult part in creating my app and took the longest time to get past.

Near the end of the EPQ, once the database server was working, development was much easier and I was able to make lots of progress with my app and managed to finish it in time to test and evaluate.

Overall, the app was difficult. Even with my prior experiences with both python and SQL the process of learning this new language and getting software to work in tandem with my code was a much harder task than I had originally perceived it to be before beginning this project. I recognise that the process of creating an app is difficult and that substantial planning is required before development in order to produce a successful app.

References

Methodologies

- {1} [study.com agile methodology](https://study.com/learn/lesson/agile-methodology.html) Accessed 01/03/2023
- {2} <https://www.zippia.com/advice/agile-statistics/> Accessed: 01/03/2023

Installation

- {3} <https://visualstudio.microsoft.com/> Accessed 9/11/22
- {4} <https://www.youtube.com/watch?v=NTe4C80OPlw> Accessed 9/11/2022
- {5} <https://youtu.be/zvp7wvbyceo> Accessed 5/12/22

Development

Prototype 1

LogIn/Register

- {6} <https://youtu.be/-Nao5QH5FBU> Accessed 28/12/2022
- {7} <https://www.visioncenter.org/blog/dark-mode-eye-health/#:~:text=Here%20are%20some%20potential%20advantages,to%20read%20on%20your%20device> Accessed: 29/03/23
- {8} https://youtu.be/yIbqWHB_gMI Accessed 28/12/2022

Databases

- {9} <https://youtu.be/XFP8Np-uRWc> Accessed 28/12/22

Prototype 2

Databases

- {10}<https://www.oracle.com/uk/database/> Accessed 22/04/2023
- {11} <https://cloud.google.com/sql-server> Accessed 22/04/2023
- {12}<https://azure.microsoft.com/en-us/products/azure-sql/database> Accessed 22/04/2023
- {13} <https://www.apachefriends.org/> Accessed 22/04/2023
- {14}<https://youtu.be/qqA8AJofRyE> Accessed 22/04/2023
- {15}<https://www.mysql.com/products/workbench/> Accessed 22/04/2023
- {16}<https://youtu.be/jcrvDTp-II8> Accessed 22/04/2023

Back Ups

- {17}<https://www.softwaretestinghelp.com/github-extension-for-microsoft-visual-studio/>
Accessed 04/05/2023

Prototype 3

- {19}<https://youtu.be/k5tlCuneiSU?feature=shared> Accessed 18/08/2023

File Handling

- {20}https://www.w3schools.com/cs/cs_files.php Accessed 4/09/2023