2020

ISAD157 – Computer & Information Security

NAME: MATT CAINE
STUDENT ID: 10672334

Final Prototype .Zip on Private GitHub: https://github.com/Matt-Caine/ISAD157

Introduction

The following report relates to my ISAD157 Module. It will outline the different requirements, the process of designing, the database, as well as the design and functionality requirements of my C# application. The aim of this assignment is to focus on the analysis and design of a data application that will store Facebook users, their interconnecting relationships, messages and workplace. I will be using MySQL workbench as the mechanism to store the data and C# - written in Visual Studio, to illustrate a sample interface for application, this will likely not be a fully finished application, but hopefully will show the possible functionality and detail, that can be grown upon, if it were to be made fully.

The first step of this process is to extract the requirements from the supplied User table.
From "Table 1" – The Facebook User Profile I was given in the Assignment Brief I've extracted these requirements:

- Store Personal User Data
- Store relationships between users
- Store Messages sent between said users

The full functioning C# application should therefore be able to:

- Display stored user data
- Update stored user data
- Delete data of a user
- Delete friendships between users
- Create a new user
- Create friendships between users

Normalization

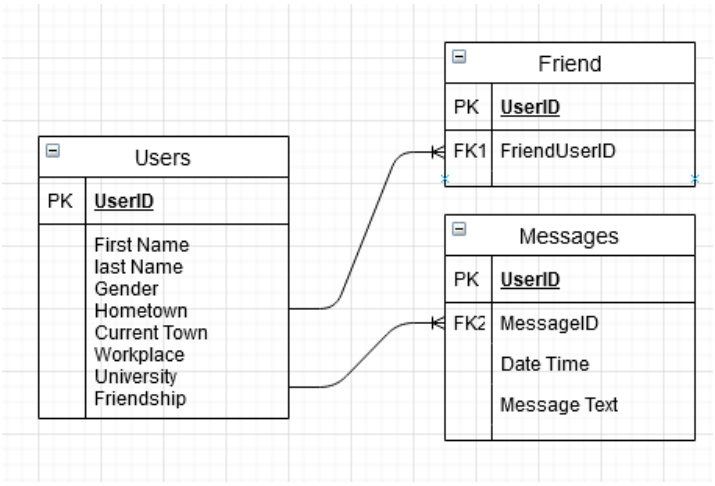
The first column below shows all the data fields I have pulled out of "Table 1" in its unnormalized form, from here I have normalised it to 3 degrees, finishing on 3NF.

UNF	1NF	2NF	3NF
UserID:	**UserID:**	**UserID:**	**UserID:**
First Name	First Name	First Name	First Name
last Name	last Name	last Name	last Name
Gender	Gender	Gender	Gender
Hometown	Hometown	Hometown	Hometown
Current Town	Current Town	Current Town	Current Town
Workplace			
Start Date	**UserID:**	**UserID:**	**UserID:**
End Date	WorkplaceID:	WorkplaceID:	WorkplaceID:
University	Start Date	Start Date	Start Date
Start Date	End Date	End Date	End Date
End Date			
Friendship	**UserID:**	**UserID:**	**UserID:**
Friend UserID	UniversityID:	UniversityID:	UniversityID:
MSG Time/Date	Start Date	Start Date	Start Date
MSG Text	End Date	End Date	End Date
	UserID:	**UserID:**	**UserID:**
	FriendshipID:	FriendshipID:	FriendshipID:
	Friend UserID	Friend UserID	Friend UserID
	MSG Time/Date	MsgID:	MsgID:
	MSG Text	MSG Time/Date	MSG Time/Date
		MSG Text	MSG Text

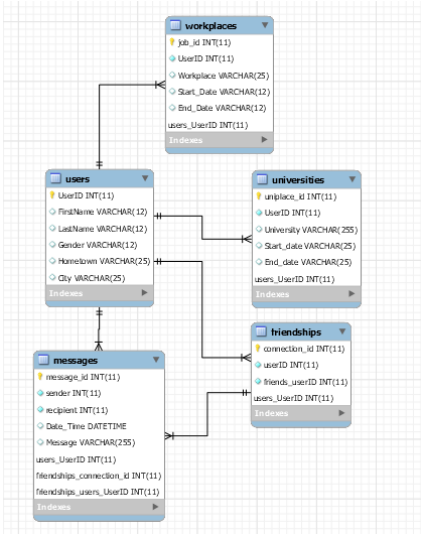
https://github.com/Matt-Caine/ISAD157

UML Diagrams

Entity Relationship Diagrams:



Initial

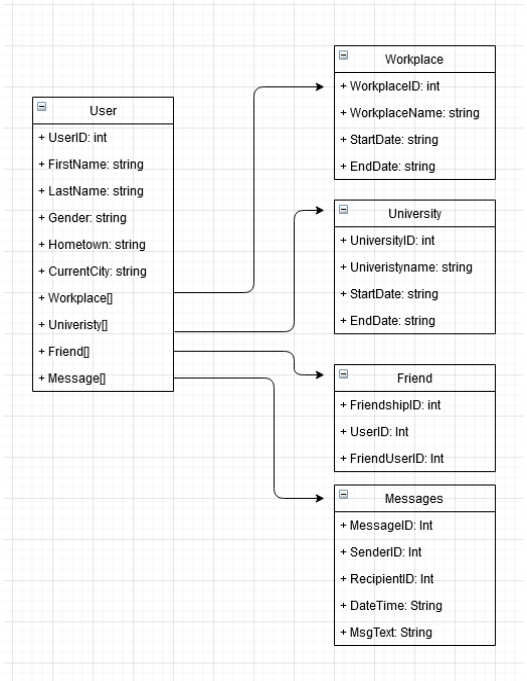


Final

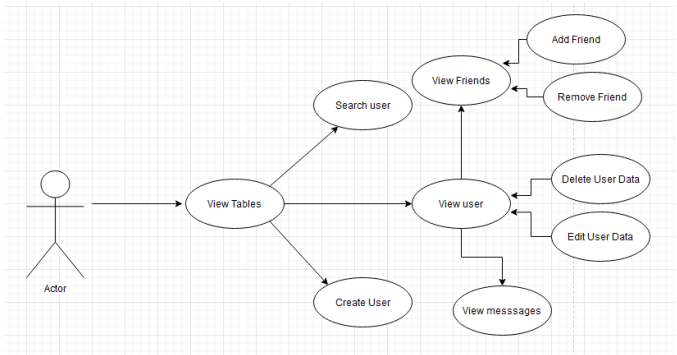
The ERD above is my initial entity relationship diagram, as you can see there is currently only 3 declared entities; Users, Friend and Messages. As you can see from the final ERD on the right more entities appear as we normalise the data. By breaking up the original tables we can assign keys for each individual friend connection and each message sent for example

Class Diagram:

To the right is my class diagram, the main class is the user and then the arrows point to where the 4 other entities are coming from. E.g. from Workplace we get a unique WorkplaceID which contains dates and the workplace name.



Use Case:

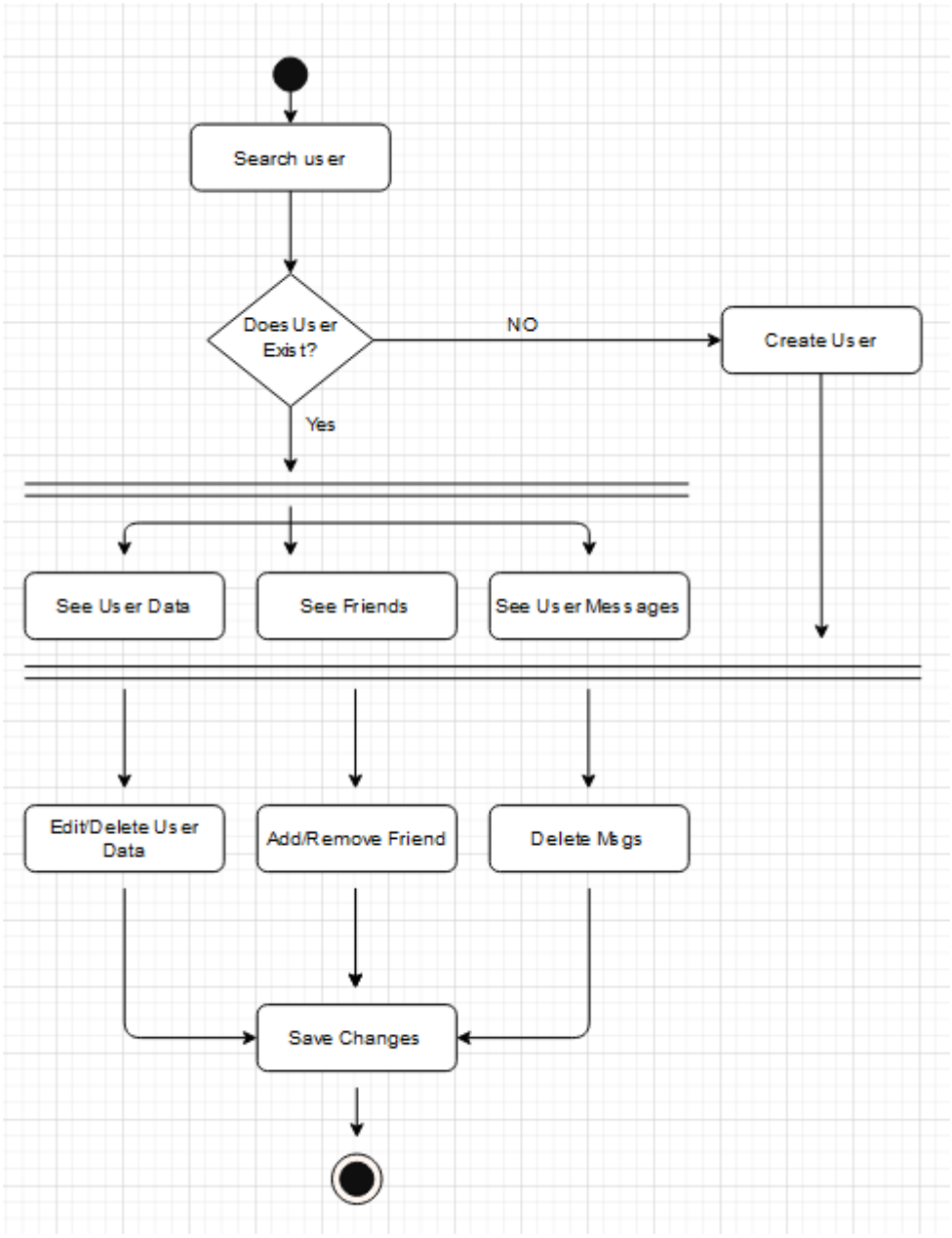


Above is the use case diagram for my C# application

Actor: “As the database manager I need to be able to search and select a user, view their information and delete or edit data if necessary. In addition to this if a user is not in the database, I need to be able to create a new user and fill in all their data so that it matches the rest of the database.

https://github.com/Matt-Caine/ISAD157

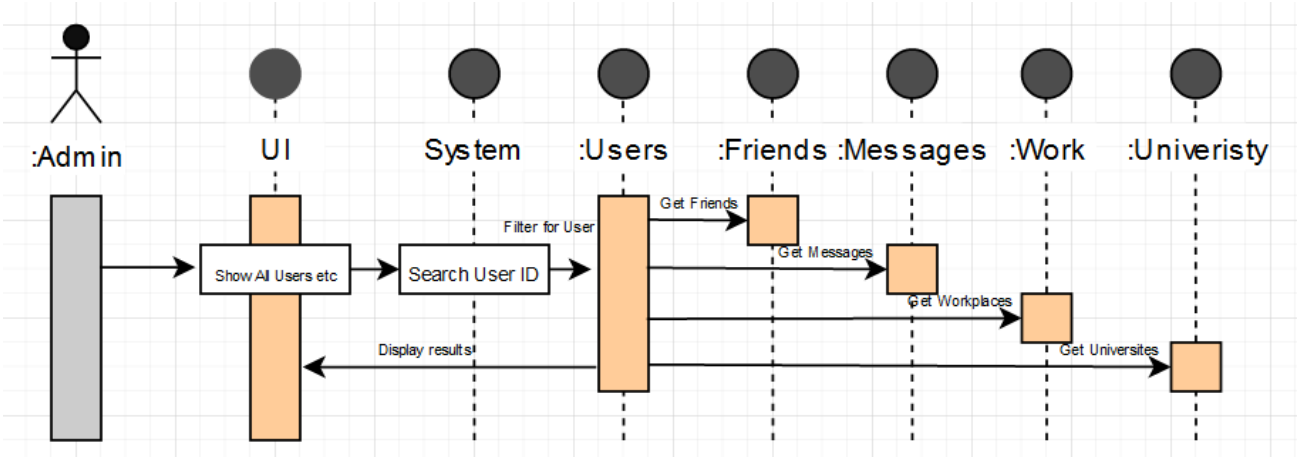
Activity Diagram:



Above is a simplified activity diagram that a user of my C# application might go through. The application is not too complicated, in its more basic use it simply reads from MySQL server and outputs into simple UI form. There are a few aspects missing from this diagram. For example, the function when adding a new friend connection, you would want the system to check if the user your adding exists and if not give the option to create a new user, or simply notify the application user that that person doesn't exist.

https://github.com/Matt-Caine/ISAD157

Example Sequence Diagram



Above is sequence diagram that explains the interaction between the Application Admin and parts/functions of the program. The only thing the Admin will see is the UI of my application which I will have design plans further down in this report, most of the functionality of my program happens in the backend of the software and the results of the Admin's quarry's are pushed back out and displayed through the GUI.

MySQL Workbench Example Code

After Creating and importing the data I was left to do some tidying up of the tables (mostly removing black Cells such as the ones created from users not having workplaces or university places) and as such ran the following lines of SQL commands;

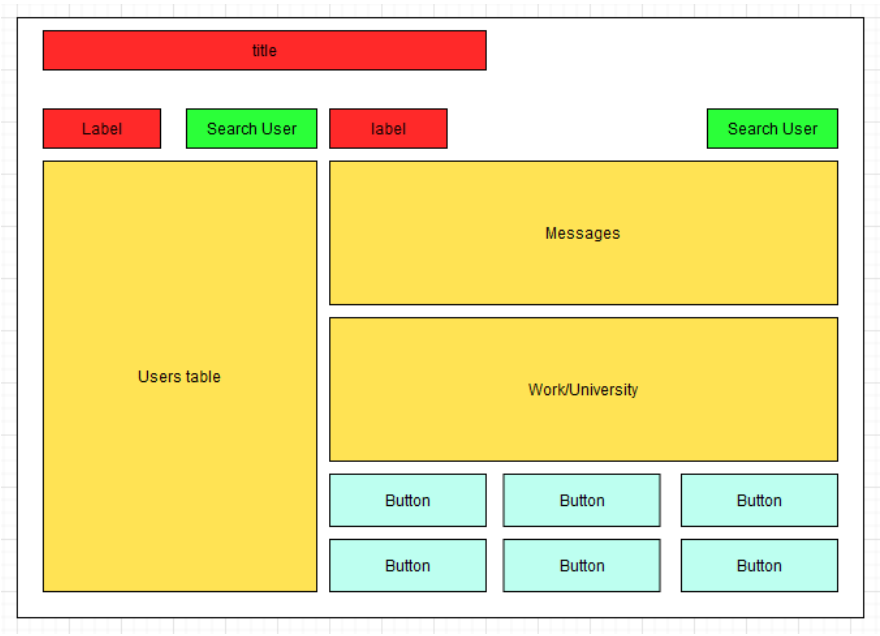
----------Sample (Not All)----------

```
CREATE TABLE users #Create Table
UserID INT NOT NULL AUTO_INCREMENT,
FirstName VARCHAR(12) NOT NULL,
LastNameVARCHAR(12) NOT NULL, #Create Column length of 12
Gender VARCHAR(12) NOT NULL,
HometownVARCHAR(25) NOT NULL,
CityVARCHAR(25)
PRIMARY KEY (UserID) #Set Key
CREATE TABLE workplaces
Job_ID INT NOT NULL AUTO_INCREMENT,
UserID INT(11) NOT NULL,
Workplace VARCHAR (25) NOT NULL,
Start_Date VARCHAR (12) NOT NULL,
End_Date VARCHAR (12) NOT NULL,

-----------------------------------------------------

ALTER table workplace #Select the table to edit
ADD start_date varchar(12) NOT NULL #Add Start_Date Column to work

AFTER workplace;
ALTER table workplace
AFTER start_date; #insert a new column after the recently created Start date
column called End_Date
ADD end_date varchar(12) NOT NULL #Add End Date Column to work

SET SQL_SAFE_UPDATES = 0; #Disable safe mode to allow deletion of rows
ALTER table workplaces
DELETE FROM workplaces WHERE Workplace IS NULL; #Remove rows with blank cells

ALTER table university

DELETE FROM university WHERE University IS NULL; #Remove rows with blank cells
-----------------------------------------------------
```

Application Design

Initial Ideas

In this section of the report I will be detailing and laying out my proposed User interface for my C#/MySQL Application.



My first design can be seen on the right. I've tried to make it as minimalistic as possible, this is for ease of use and aesthetics. The yellow boxes are the DataGrid's that will display the data read from the MySQL Database.

The "user" grid is the most prominent, since I figured that would be the most commonly interacted Grid view. Moving on, we have the two other data grids on the left. These DataGrid's are "User messages" and "work and university", I feel like Work and University are commonly seen together hence they are in one grid. This may only be a design step and is not set in stone.

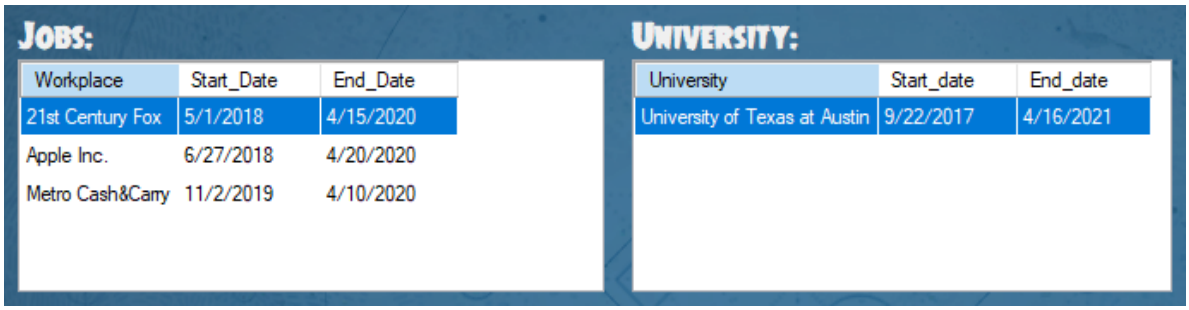
The red boxes are placeholders for text, these serve no software functionally, however they help the user of the software understand what is on the screen, this includes title of the program and labels for what the data grids are displaying. Finally, we have the functionality parts of the program, these are displayed by the green boxes and the light blue buttons. The green boxes are showing where I want search boxes to be for the ability to filter the data grids to get the information you want. Next we have the blue boxes which are buttons, these buttons should include Add User, Edit User, Delete Users and Delete Messages.

I feel this minimalist design is good for user accessibility and most people will find my program intuitive and won't have issues while trying to do the tasks they want. Related to my accessibility goal, I will try limit the amount of unnecessary actions, hence I wont be having buttons to search what is entered in the search boxes and will instead have the data grids update as the users types, this will help with ease of use.

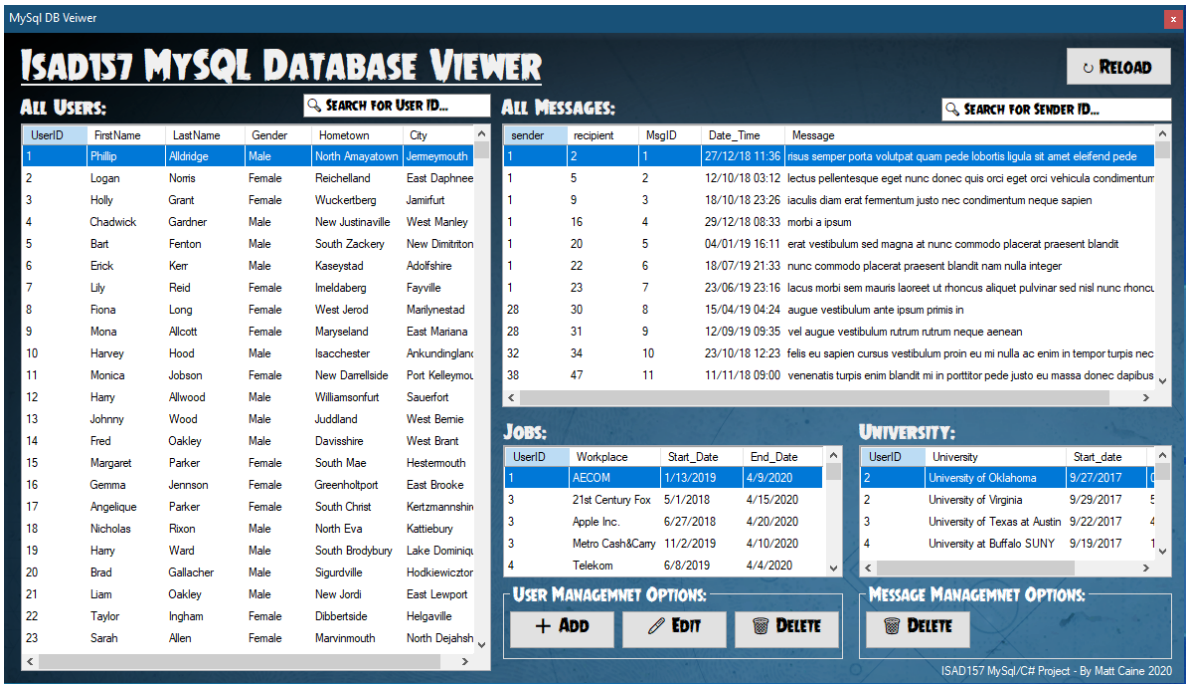
Fabrication

From starting to fabricate my designs my first thought is that I want the buttons to be obvious, so I have added a group box around each set to link them together, so far I have "User Management Options" and also "Message M Management Options". In addition to this I have also added symbols to each of the buttons, this is to make it clear the function of each of them, E.g. A Plus for add, A pen for Edit and A bin symbol for delete.

My next change I made while putting together my GUI, was separating the Users Jobs and the universities the User attended. I feel like this was down to making the UI more user friendly and making obvious what was being displayed to the user.

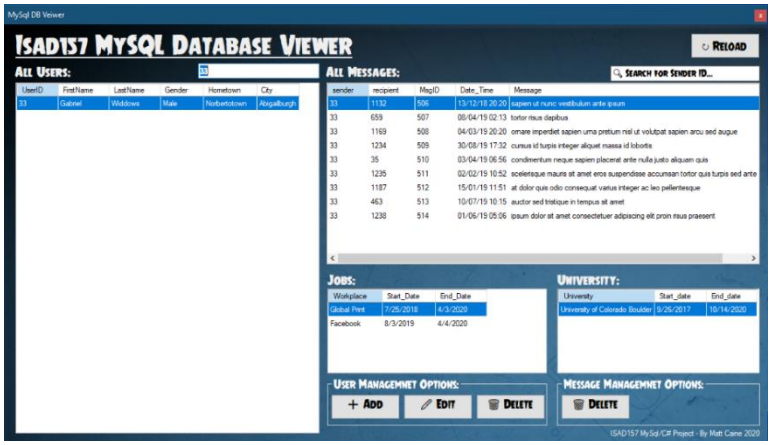


Final Design



Above is my final UI for my C# program. As it stands the final Program only displays the database, with the ability to search for specific user IDs and display all data connected to that user E.G All their messages, Jobs and all the places where that user has worked. Since this is only a showcase example of how a full program could look and work, I feel like the placeholder buttons for the functions are effective in showing this.

With additional time and necessity, I would implement a multiple form program, with each of the buttons leading to a new form where for example they can fill in the New users Details and connect to any existing users to form friendships.



Final Design (While Searching for UserID: 33 and displaying Information related to that User)

https://github.com/Matt-Caine/ISAD157

Evaluation & Conclusion

To conclude this report for the ISAD-157 module, I am very happy with my outcome. My prototype application shows the base concept but has potential to be grown and improved into a fully fletched program. My SQL database meets my requirements set out in my introduction, these include
The ability to store Personal User Data, store relationships between users and store Messages sent between users. Looking back at my requirements for the C# application I have only met the display function, but I feel like I have provided enough evidence to represent the missing functions that would obviously be present in a final build.

Looking back on the other aspects of this module, I feel like I struggled the most with the UML diagrams. I feel like this was down to a lot of them seeming more complex then they really are and once I had spent a bit of time reading up on each of them and watching a few YouTube videos, I was a lot more comfortable in creating them. I was in a similar boat with the SQL but once it clicked in my head that SQL is simple commands like ALTER and ADD, it was a lot easier and it was simple repetition when creating and organising the tables on MySQL workbench.

Overall this module was very rewarding, and I feel like I have created something professional, and something I feel like I can build upon, not only in my University life but also my potential career. SQL is a very powerful language and it is sort after in the industry, by going through this module it has made me want to gain more knowledge and practical expertise using SQL, in the aim of becoming more hireable in the future.

If I were to redo this assignment again or in the future, I would likely try focus more on the scenario of the task and going at the task in a more targeted and tailored way, as I feel my final application, even though it is unique, it feels mass marketed and in my mind that makes me think it is not fully designed for my Client in this case, working with the Facebook Database. A more tailored approach may include discussion with the client, more research into how it is intended to be used and making it in the most efficient way to suit the needs of those who will be using it the most, in the given scenario.

https://github.com/Matt-Caine/ISAD157