**Programming Project 1**

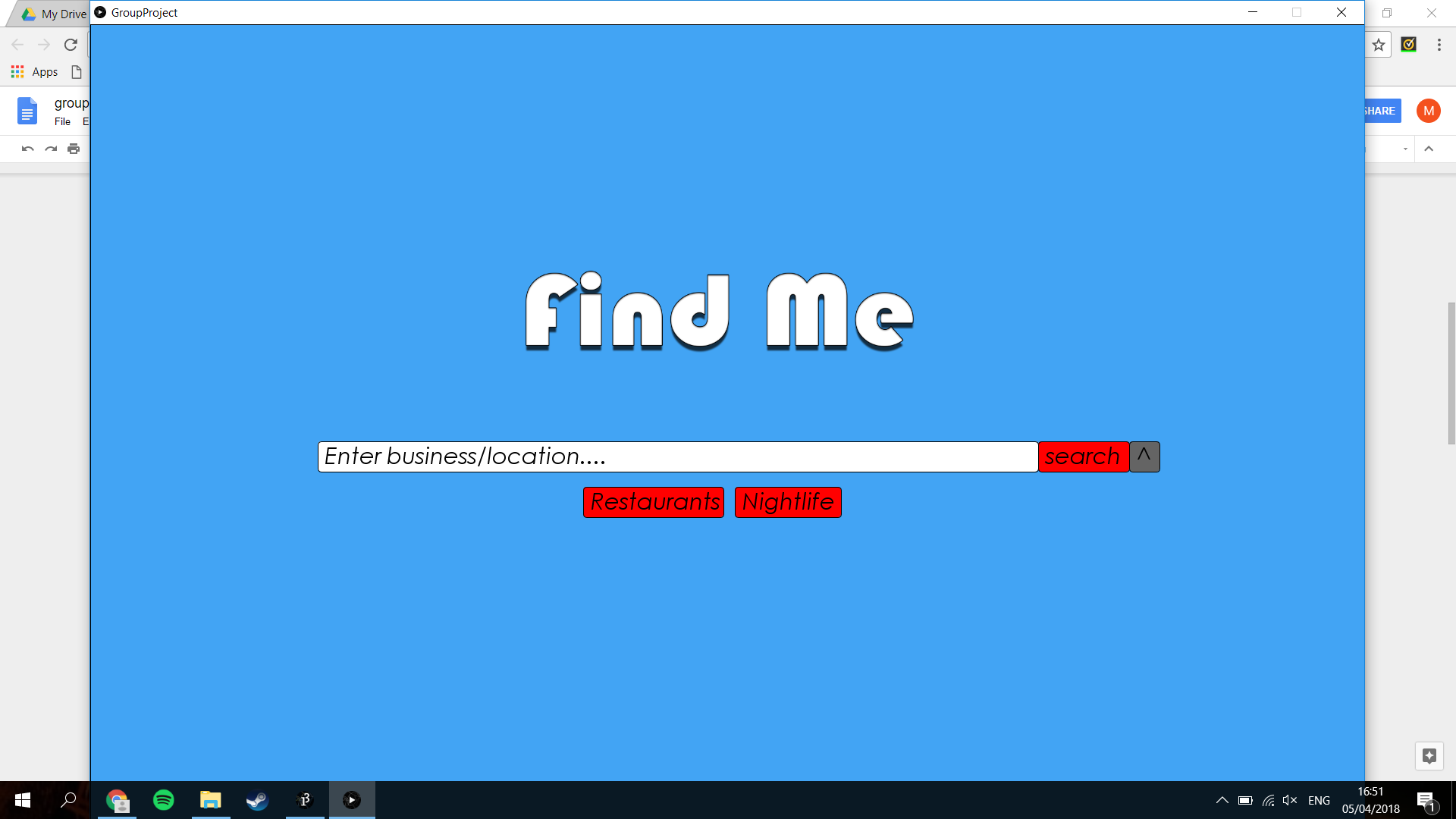
**CS1013:**

**Matthew Flynn: 17327199**

**Jamie Coffey: 17336373**

**Ciaran Cody: 17326951**

**Michael Black:17341914**



**Outline of Design**

The goal of our project was to create an application where a user can access and interact with a large amount of data. For us the key was to make an efficient and fast program which reacted quickly to the user, but also had enough functionality to make it user friendly and understandable. The data retains to a list of reviews for various businesses. The project design is based off 4 different screens. A search screen, results screen, business screen and user screen. The data is processed using SQLite and various sorting functions. Our project also contains the following additional features, a pie chart on each business and user screen which displays that business’s ratio of star ratings. It also contains an option to sort by category of business, night club, restaurant etc.

**How we split up the work**

When it came to dividing the work, we took a democratic approach where everyone decided what they would like to work on and also what they thought they were good at. Ciaran decided that he would be most suited to the backend functionality like data handling and was interested in learning about databases. Michael had an interest in the user interface and was comfortable working with widgets and screens. Matthew wanted to work with sorting and other algorithms and thus decided to work on the business class. Jamie wanted to implement a user feature where every user had its own unique statistics page displaying all the details of the user. But like with every group project these roles slowly faded and everyone seemed to overlap with each other, contributing ideas and helping fix bugs in each others code. This approach to the work allowed everyone to have a good understand of the code but also get work done faster and to a higher standard.

**Features implemented and project breakdown**

Section 1: The search and results screen:

The search screen uses a text widget to allow the user to search a business from the database. The search bar widget uses the query class to search the database and it returns the entire list of business that contain what the user typed in. Once it has been searched the businesses are placed in an arraylist and the search screen changes to the results screen. The results screen displays the business name and star rating of the list of the first 12 businesses and allows the user to go to the next page of results, select a business to view and return to the search page. The results screen also displays the message “No results found” is the users choice has no results.

Section 2: The business and user screen:

These pages display all data relating to the business and user’s. The business page displays the name, star rating, top review, address and contains a bar chart displaying the number of each possible star value that could be given of star ratings the business received. It also has a pie chart showing the ratio of each star rating for the business. By clicking the user name on the business page you can access that user’s page. The user page shows the name, user id and gives a bar chart of the distribution of the reviews by their ratings in funny, useful and cool. It also contains a widgets to allow the reviews to be sorted by latest, oldest, most useful, coolest and funniest.

Section 3: Data processing:

From the start we decided to use the large dataset, but to achieve this seemed difficult without the use of a database. Ciaran volunteered to do some research into possible solutions and decided to use SQLite. The reasons include its lightweight nature as it is used on low powered devices such as smartphones and is optimised for speed. It also ran locally without the need for any server as long as each of us had the database file locally on their machine. This allowed us to access the large dataset in a quick and efficient manner. Ciaran Implemented the query class which allowed our program to interface with the large dataset using the SQLite java library. Many of the queries use the full text search functionality provided by FTS5, to implement this required each table in the database to be added to its own virtual table. This caused the database to double in size but meant that our queries became nearly instant, which was one of the aims for our program. After a query has been called and returns a result it is usually stored in an arraylist. Matthew and Jamie then wrote various functions to sort the results based on different criteria e.g. most recent review. Michael then took these results and designed a graphical interface that would display results, in a meaningful and user friendly manner.

Section 4: additional features:

The additional features we added include:

1. A pie chart and key in the business page to help display the distribution of star ratings.
2. Sort by oldest, newest, highest rated and lowest rated reviewed options to the business page.
3. Sort by old oldest, newest, funniest, most useful and coolest review options to the user page.
4. A restaurants and nightlife button to the search page to allow the user to get a selection of businesses or nightlife options to view.
5. A scrollbar is added to the search results page to allow the user to scroll through the various options available.
6. There is also a search by location option that allows the user to enter an area and search by that location.

Problems encountered

The main problem we encountered was in the use of the SVN system. In the beginning especially we were very disorganized and sometimes encountered errors when two of us tried to edit the same class. We also sometimes couldn’t test the work we had done in a particular week due to not being ready to accept that piece yet. So sometimes functions were left untested for a week while we got up to the point where they’d be useful. In summary our major problem was organisation however we managed to pull ourselves together by week 4 of the project and worked smoothly for the last 3 weeks.