CS1013 Programming Project

Group 10

Documentation

How you split up the work and organised the team

We started by setting up a Facebook group to first contact each other. This allowed us to communicate whenever and decide the times that would most suit us to meet up. We arranged to meet up most Wednesdays and then on Thursdays a few hours before the lab to ensure all our work for the week was finished on time. On our first meeting we split up the workload, so everyone would have a job to do. Brian Lynch was responsible for maps and graphs, Matthew Henry did the database set-up, David Burke created the queries and Ciara O'Sullivan and Krishanu Dey were in charge of User-Interface. This proved to work successfully as each person set their own goals for the week while keeping the rest of us in the loop.

Features implemented

Search bar: this allows the user to search for any business they want to see if it is in the data set. They can press enter of search and it should work and there is even an autocomplete function for if they don't finish the name of the business.

Category List: a drop-down list, so the user can choose what type of business they are looking for

Home Buttons: quick access buttons that will show the user the most reviewed businesses, businesses near them and the highest rated businesses.

Results box: scroll box that shows the user the results of their search

Maps: takes your current location, or another one entered, plot this on a map and then show you businesses near you.

Graphs: show the star ratings for each business and the reviews per month. The user can switch between graphs by clicking a button

Queries: finds the business' star rating, whether the review was useful, funny or cool, business name, business address

Problems encountered

In the beginning we had some problems with our SVN. Additions to the code were being committed wrong and causing problems running the program after updating. However, we learned from our mistakes and ensured that we all knew how to commit any additions correctly from that point on. Another issue we had was not being able to use the larger data set on the college Wi-Fi. We soon resolved that problem as by each of us downloading a VPN. Any other issues were resolved by coding in pairs.

Outline of design

DataBase (Matthew)

SQL server

It was decided early on to implement a mySQL server, hosted online on an AWS relational database instance. This server holds the entire "big" dataset in separate tables. The server itself is accessed by a MySQLaccess object instance on the client side (java). Many columns in the SQL server tables are also indexed in order to improve and optimise the functionality they offer.

The server also hosts numerous SQL stored procedures in order to streamline the querying process, this in itself added to the SQL servers performance and reduced the load on the client when querying. It does however obfuscate some of the functionality as it is hosted on the server as opposed to being displayed on the codebase itself.

User Interface (Ciara and Krishanu)

Buttons, Search Bar, Drop Down list and Scroll List

We used ControlP5 library to create the buttons, the search bar, the dropdown button and the scroll List. Whenever a ControlP5 button is clicked, the function with the button's name is called. When an object in the list is called, the function with the list's name is called and the index number of the object is passed as parameter.

Loading Screen

For the loading Screen, there is a global variable run which is initialised 1. The draw function has an if statement, when run is equal to 1, it draws the loading screen and increments the run. When run is equal to 2, the database connection is made and the screens, the map and the graphs are initialised and run is incremented again. When run is equal to 3, all the elements are drawn. After this, whenever draw is called, only the code in the last if statement is called.

Changing Screens

For the changing screens there are four variables, home, resultPage, run and currentScreen. When run equals two the initialiseScreen function is called, which sets up the widgets for the home screen. Then the initialiseResultPage function is called, which sets up the widgets for the results page. When run equals three the background image is loaded and there is an if statement that checks if the currentScreen is equal to one, if yes the home screen is drawn. If it equals two the resultsPage is drawn. There is an exitButton for both screens and when this is pressed it calls the exit function which sets the value of the currentScreen.

Positioning of the Widgets

The x and y positions of the widgets have been done such that it's a relative value of the Screen Height and Screen Width. This has been done so that the position, and the size of the widgets will be always relative to the screen size, so the program will work work fine for all resolutions.

Map (Brian)

The map was created using a processing library called UnfoldingMaps, this showed the locations of both businesses and the user of the application. This proved quite tricky to do as after many failed attempts i discovered that UnfoldingMaps was not compatible with processing 3+, however a quick email to the developer solved this as he happily gave me the beta version for processing 3. This had its problems with some known bugs in the beta but i managed to overcome them, despite taking some time. Once this was done the actual usage of the library was quite straight forward and i collaborated with Matthew and david to get queries from the SQL server to take in longitude and latitude values. I also created my own marker using PGraphics, i felt this was better than an image because it was then customisable and clear. The map was able to be dragged using the mouse and zoomed in/out using the mouse wheel and the '+' and '-' keys.

Graph (Brian)

There were two graphs in this program, one was a histogram displaying the businesses star rating and another a line chart which shows star rating with respect to time. Both graphs were made using the processing library Grafica, this took some time to learn how to use but once I was familiar with the library it made making graphs very efficient. I chose this library because of the degree of customisation it offers and its aesthetic design.I added extra images of stars and opaque background to the graphs to improve their appearance. I collaborated again with david and matthew to pull in the data from the SQL server to load into both graphs. I also collaborated with Kris and Ciara on designing the colour scheme and position of the graphs so they did not look out of place.

Queries (David)

The Query class was created in order to take the raw data returned from the SQL queries to the database and parse them in a fashion that was more easily utilized by the UI team. In some cases this would involve receiving a collection of objects and performing logical operations with the variables within (dates, useful, funny, cool etc.), while in others the functions would simply parse the variables from the objects in order to more easily print lists of results for businesses and reviews.

ReviewsWithinDates() and ReviewsWithinStars()

The function ReviewsWithinDates() took a list of review objects and two dates expressed as strings. The strings were then converted into java Date objects to make logical operations easier and more intuitive. The list of reviews would then be filtered using the Date objects as parameters, returning the filtered list. The function ReviewsWithinStars() behaved in a similar fashion taking two int parameters instead. Although these methods were functional, it was found that using java to perform the logical operations on larger data sets wasn't time efficient and thus created a poor user experience. Due to this, functionality was transferred over to SQL, which quickened the result processing time significantly.

MostUsefulReviews() etc.

These functions take a list of Review objects and ranks the contents of that list in the amount of useful, cool, and funny votes acquired by the reviews. The sorting loop would switch two entries in the list if they were found to be out of order, and repeat this process for each entry until it was in the correct placing. Given more time this functionality could have been added to the review list. However, focusing on qualities that more closely pertain to user experience (Stars, History etc.) proved to produce a more effective and streamlined program.

ReviewDates(), BusinessIds() etc.

These functions take a list of Review or Business objects and parse a single variable from the objects, then return a list containing the selected variable for each object. These were created in order to avoid accessing the variables via the getter methods for the objects in Main(), which proved to be counter intuitive and overly cumbersome while programming.