**Technical document – Friendable**

Overview of the program:

Friendable is a program which aims to join and connect people of the same interests and qualities together in order to make friends. It takes data from users and matches it with other users in order to obtain a match that is calculated and relevant, in the hope that matching profiles with similar personal qualities, likes, hobbies and interests successfully creates a new friend.

The program is compatible with a PC, and eligible to use on the chrome and internet explorer browsers. In order to use the program, the user must have access to a trackpad, mouse and keyboard. It is currently not compatible and has not been tested on small handheld devices, iPhones and Androids, and has yet to been tested on a wide range of browsers.

Files:

The central file of the program is ‘index.html’ which is the building blocks of the program and puts all the foundational elements on the screen for the user to interact with and in order to display the matches made for users. The JavaScript file ‘majorProject.js’ contains all the functions and variables of the program, including the use of JQuery in order for the program to be functional. Styling for the project is created within the CSS page called ‘majorProject.css’, this specifies all the specific styling of each different element of the program. Images for the program are contained within the images folder, and the user’s documentation is attached as a PDF within the program.

Adding users to database:

There are two main functions that allow user details to be added to one database with two separate collections. The ‘addProfiles’ function posts the users profile to the ‘User’ collection passing the parameters item (the user), Url (Url of the database) and apikey. A second function, ‘add\_second\_response’ connects the user survey data to a separate collection, ‘answers’ through passing the parameters item 2, url2 and apikey.

When adding a user to the database, on clicking submit after signup the program will store the username, firstname, lastname, age, gender, birth month and password of the user under temp item and this will be passed into the database in order to create a new user with specific profile data.

When adding the second response to the ‘answers’ database, originally on sign up, the program will store tempItem2 which contains the username of the signing up user and set the values of question 1, 2, 3, 4 and 5 of the survey to be ‘blank’, as the user has not entered these values yet and will proceed to do this later in the program.

The login function allows users who are already stored in the database to access the website. It passes the parameters url, apikey, username and password. The function checks if the entered username exists in the database, and if this is true, then checking if the entered password matches the corresponding password of the username determined. If this is all true, then the function will allow the user to successfully login and proceed throughout the program.

Adding survey results

As outlined before, originally on signup the ‘add\_second\_response’ function causes the values of the survey questions to be set as blank, as they have not been filled in by the user yet. Once the user has access to the home page and has successfully logged in, the completion of the survey will update these values through the add\_survey\_response function. This uses the put method which allows the ‘answers’ collection to be continually updated every time the user completes the survey. Each field in the database corresponds to a value obtained by the function, the username field is already determined by the signed up user. The value of question\_one is obtained through receiving q1\_response\_v which will be the value of the button clicked, this is continued for each question and is received by the program when the user click ‘submit’ after completion of the survey.

Matching function:

There are a variety of matches throughout the program in which all follow the same logic however differ depending on the criteria of the match. There are three main sections of matches. This includes ‘Profile Matches’ which uses profile data submitted at sign up to match similar uses, the ‘Overall Match’ which matches users based on survey data and the amount of fields they have in common and finally the ‘Survey Matches’ which specifically targets one field of the data base and matches uses who have the same response to a survey questions.

The ‘matching\_ages’ function is an example of a profile match. The logged in user is defined as the global logged in user and from there it is determined that the logged in users age is the data we are trying to match to other users. The function sorts through all the ‘Users’ database and determines any users that have the same age data value as the logged in user, and when found, the matches are displayed through pushing the username of the matched individual onto the screen for the logged in user to see. The same logic is used for the ‘matching\_gender’, ‘matching\_OppositeGender’ and ‘matching\_BirthMonth’ functions which all allow the value of the specified field to determine if a profile is matched or not.

The ‘FirstMatch’ function is an example of an overall match. The logged\_in\_users\_answers is initially set to null and then once it is established that the logged in username is equal to the logged in user it fills the variable with data from the ‘Answers’ database. For example the data of question one is equal to the value of the logged in users question one field. The logic is then, if the logged in users values of question 1,2,3,4 and 5 are equal to another users value of question 1,2,3,4 and 5, then the users are deemed a match and the matched users username is displayed on the screen. The same logic is used for the functions ‘SecondMatch’, ‘ThirdMatch’ and ‘FourthMatch’, however the amount of data varies. For example, the FirstMatch is concerned with obtaining the most valuable match, a profile that has many of the same answers. Therefor, SecondMatch only requires 4 of the same survey answers, ThirdMatch only requires 3 of the same survey answers and FourthMatch only requires 2 of the same survey answers.

The ‘QuestionOneMatch’ function is an example of a specific survey match. This differs from the previous match as it only compares and sorts through one fields value. There are five different questions of the survey and thus five different survey match functions. However, it is constructed in the same logical way as the previous match. For ‘QuestionOneMatch’, the function determines the corresponding username in the survey database. It then sorts through the profiles through executing if the logged in users data from question one is equal to another users question one data, then they are a match. This occurs also for the ‘QuestionTwoMatch’, QuestionThreeMatch’, ‘QuestionFourMatch’ and ‘QuestionFiveMatch’. Allowing users who have the same field values for the specified field to be matched and appear on the screen.

Construction of the program:

The program also uses a series of jQuery selectors along with functions to hide and show different elements of the program at different necessary times. On original opening of the program the function ‘hide\_home\_page’ hides the elements of the ‘home\_page’ class and the function ‘hide\_home\_navigation’ hides the elements of the ‘homeNavigation’ class. The function ‘user\_home\_page\_function’ is used once the user has logged in and signed up in order to hide these elements of the page. It hides the ‘sign-up’ and ‘Login’ class while showing the ‘start\_up’, ‘home\_page’ and ‘homenavigation’ classes in order for the user to easily transfer between the sign in and the main functional page of the program.

Once the user is on this main home page, a series of jQuery selectors is used to help the navigation of the page and ease of switch between elements. This is achieved by clicking the various buttons then hiding and showing the corresponding classes that relate to that button.

Survey:

The survey is constructed through a series of jQuery selector functions that allow them to easily change to the next question after a response has been clicked. In order to obtain the survey the results there are a series of variables that contain what the user has clicked. The program consists of variables q1\_response, q2\_response, q3\_response, q4\_response and q5\_response, which are all initially set to ‘nothing’. A click handler is lodged on each separate button, this allows all the buttons within question one, when clicked, will record the value of which option was clicked to q1\_response. This continues to go on for each question, until all questions are answered, and the data is submitted into the database, under the corresponding fields.

Databases:

The program uses one database, therefor it only contains one apikey which is defined at the top of the program. However, it uses two collections ‘profiles’ and ‘Answers’ in order to hold user’s data retrieved from sign up and the user’s data retrieved from the survey results. They have two separate URL’s which are defined at the top of the program.

The profiles collection utilises URL 1 and involves saving all the data that the user enters on sign up, in order to be used later for when individuals log in or want to make a match based off their profile data. It contains the fields username, firstName, lastName, age, gender, birthMonth and password. These are all entered within the sign up page and submitted to the database in their corresponding fields.

The answers collection utilises URL 2 and involves saving the data that the users enter during the survey. As mentioned before, the profile collection retrieves the username of the user, when these user profiles are submitted, the corresponding username also submits to the survey collection, allowing the program to later on enter the survey results when completed and correspond them to the appropriate user. The fields of the collection are question\_one, question\_two, question\_three, question\_four and question\_five, these can be frequently updated whilst the username remains the same every new time the user completes the survey.