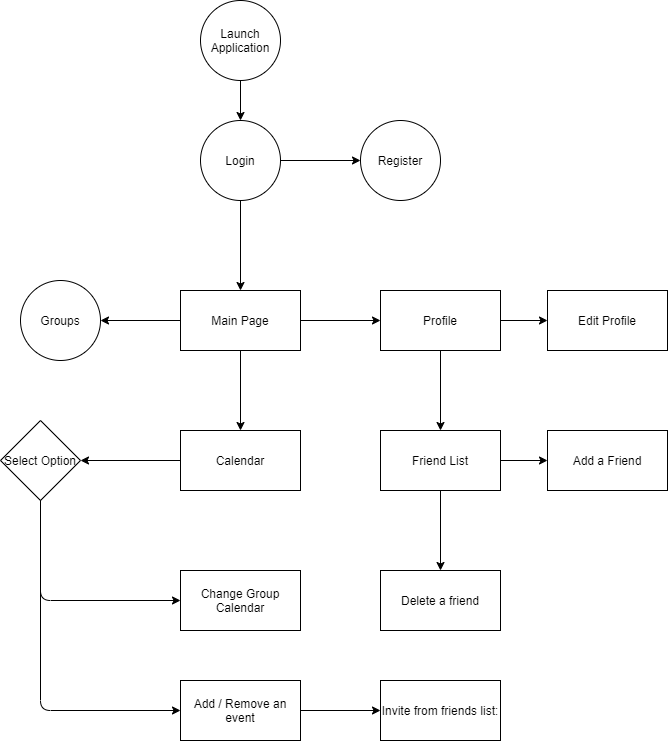
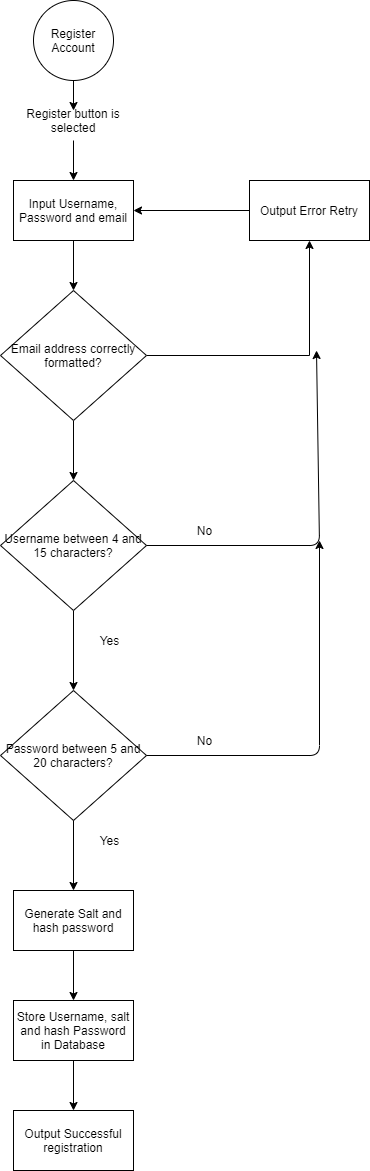
**Word –2**

**High level Overview – 2.1**

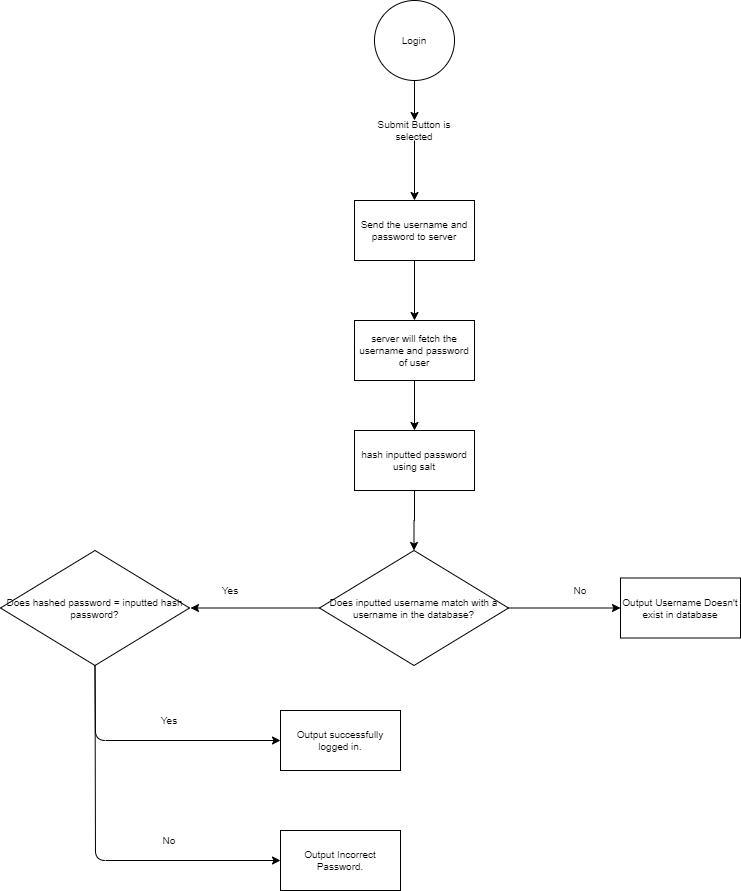
Main Menu – Flowchart



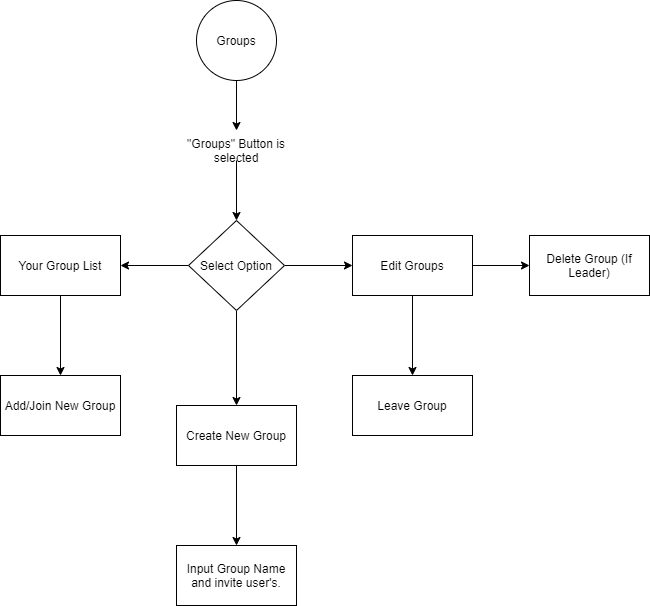
Register - Flowchart



Login – Flowchart



Groups – Flowchart



**Description of Modular Systems – 2.2**

When the application is launched you will be asked to either login or register an account then after successfully logging in the user will be shown the main menu. This diagram is the different pages that will be included.

Login

Register

Home Page

Calendar

Profile

Friends

Add new friend

Delete a Friend

Edit Profile

Groups

Add new group

Edit Groups

Settings

Log Out

The Home Page will be the hub for my application allowing the users to branch off to the three main pages, Profile, Groups and the calendar with each of these main pages having subpages such as the Friends tab for the Profile page.

I will be using a stack to represent each page so if the user goes onto the profiles page and then wants to return to the main menu they just need to press the back button and since the main menu is the last page opened it will be at the top of the stack. This makes it so I don’t have to code a separate back button for each page telling it what page to return to.

**Data Dictionary – 2.3**

User Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Field Type | Field Size | Purpose | Example Data | Validation |
| UserID | INT | 5 | Assigns a unique identifier to each user which will be used as a primary key. The number will just be automatically incremented each time a new account is registered. | 6 | Needs to be a whole number. |
| Username | VARCHAR | 15 | Stores the username of each account. | Luke | Starts with a letter, not empty, between 4 and 15 character |
| Email | VARCHAR |  | Stores the email of each account. | Xyz@email.com | Must be in the format xyz@email.com |
| Hashed\_password | CHAR |  | Stores each player’s password after its been salt and hashed. | d1e8a70b5ccab1dc2f56bbf7e | Unhashed password must be between 8 and 15 characters long. |
| salt | CHAR |  | Stores the randomly generated salt of the unique user. | ostrich |  |

Groups table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Field Type | Field Size | Purpose | Example Data | Validation |
| GroupID | INT | 5 | Assigns a unique identifier to each group which will be used as a primary key. The number will just be automatically incremented each time a new group is created. | 7 | Needs to be a whole number. |
| DateStarted | DATE |  | Stores the date that the group was first created | 2012- 05-16 | Needs to be in the correct format: YYYY – MM - DD |
| Group Name | VARCHAR | 15 | Stores the name of each Group. | Tesco Work. | Starts with a letter |

Member List table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Field Type | Field Size | Purpose | Example Data | Validation |
| GroupID | INT | 5 | Stores the ID of the group. Part of a composite key. | 6 | Needs to be a whole number. |
| UserID | INT | 5 | Stores the ID of the user who is in the group. Part of a composite key. | 12 | Needs to be a whole number. |

Data Structures – 2.4

**Validation – 2.5**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Validation Checks** | **Description** | **Error Message** |
| Username | 4 < Length <= 15  Matches the expression /^[A-Za-z][A-Za-z0-9]+$/ | Must be more than 4 characters and less than or equal to 15 characters.  Must start with a letter.  Can't have any spaces. | Usernames must be between 4 and 15 characters long.  Username must start with a letter and cannot contain any spaces. |
| Password | 8 <= Length <= 15  Matches the expression ^(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[^a-zA-Z0-9])(?!.\*\s).{8,15}$ | Must be between 8 and 15 characters  Must have a lowercase and uppercase letter.  Must have a number and a symbol. | Password must be between 8 and 15 characters longPassword must contain at least one lowercase letter, one uppercase letter, one numeric digit, and one special character |
| Email Address | Make sure email is in the format of [xyz@email.com](mailto:xyz@email.com) / .co.uk  Matches expression [\w-  ]+@([\w-]+\.)+[\w-]+ | Ensures email addresses are correctly formatted. | Emails must be in the format of [xyz@email.com](mailto:xyz@email.com) |

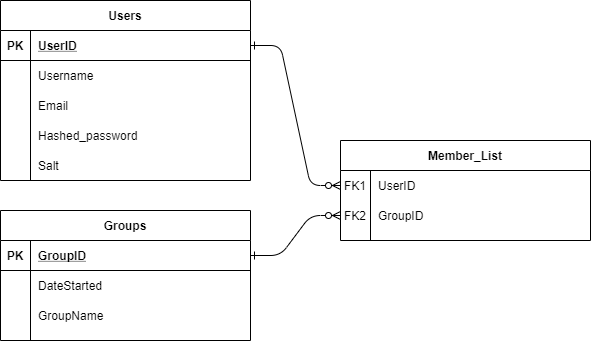
**Database Design**

The “User” table stores information on the users account, it stores their UserID, Username, Email, Hashed Password and Salt.

The “Group” table stores information about the user created groups, it stores their GroupID, DateStarted and GroupName.

Initially these were the only tables I had but there was a problem. A user could have many groups and a group could have many users, this meant that these tables had a many to many relationship which would bring about problems in integrity and problems when updating a user's or groups details.

In order to fix this problem, I had to create a new table to act as a linking table. I made the table “Member\_List”, this is table that consisted of UserID and GroupID. The primary key of Member\_List is a composite key consisting of UserID and GroupID



**Planned SQL Queries**

**Adding a new user into the database after they have registered:**

INSERT INTO users(UserID, Username, Email, Hashed\_password, Salt)

VALUES (NULL, “[Luke”,”Luke@email.com](mailto:Luke”,”Luke@email.com), “sags4wv2q54”, “afgva424x2xd”

The value for UserID is null as the user won’t be inputting that as it will be automatically incremented each time a new user is created.

**Adding a new Group into the database**

INSERT INTO Groups(GroupID, , DateStarted, GroupName)

VALUES (NULL, 2018-07-12, “Tesco Work”)

The value for GroupID is null as the user won’t be inputting that as it will be automatically incremented each time a new user is created.

**Adding a member to a group**

INSERT INTO Member\_list( UserID, GroupID)

VALUES (12, 13)

**User is updating their password**

UPDATE Users

SET Hashed\_password = “AAAAAAA”, SALT = “BBBBBBB”

WHERE UserID = 6

AAAAAAA is the new hashed password of the user and BBBBBBB is the new randomly generated salt.

**Searching for a user to add to the friends list:**

SELECT \*

FROM Users

WHERE Username = “Luke”

**Searching for a Group to join**

SELECT \*

FROM Groups

WHERE GroupName = “Waitrose Work”

**Deleting an account:**

DELETE FROM Users

WHERE UserID = 14

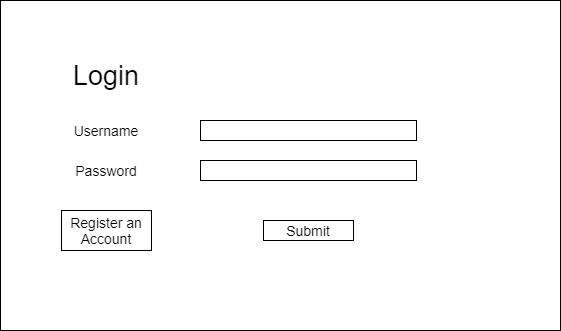
**Deleting a group:**

DELETE FROM Groups

WHERE GroupID = 4

**UI Design**

**1-Login**



The is the first screen that will be loaded up after the application is launched. One of the main features of this calendar is the groups and so in order for this each user needs to have a personal account so they have to either make an account or login to an account they’ve already created.

The input fields will be validated, they can’t be left empty and the username must start with a letter and be between 4 and 15 characters in length.

The password also must be between 8 and 15 characters long and contain a lowercase, uppercase, number and one special character. If there are any errors in this stage, then an error message will pop up explaining what the user has done wrong.

The “Register and Account” button will send you the register page so the user can make a new account.

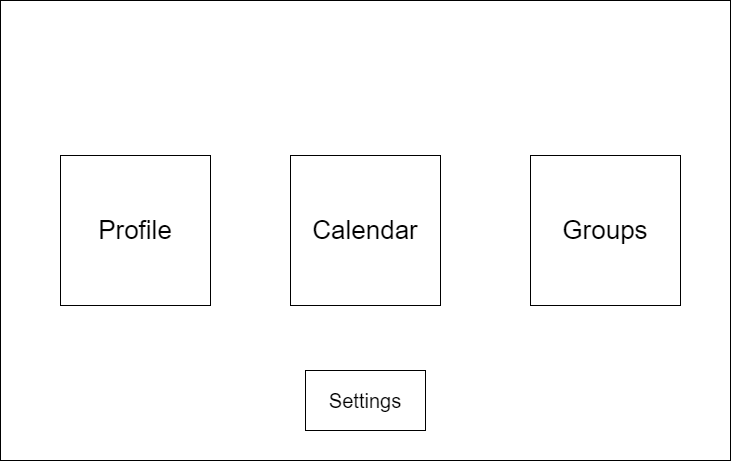
The “Submit” button will send the information to be checked on the database and if the username and password is the same as the data in the database they will be logged in and move to the main page.

**2- Register**



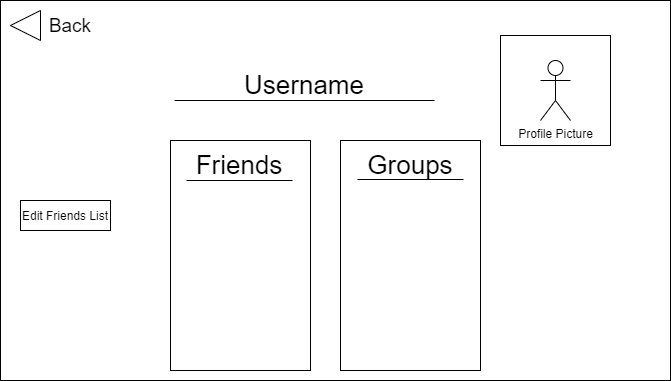
This is the screen the user will be shown if they click the register an account button. They will be shown four boxes, username, email, password, confirm password. The inputs will be validated in the same way as the login page. The program will check if the password and confirm password are the same string, this is to make sure the user didn’t make an error when typing their password the first time. When they have entered their username, email and password this information will be sent to the database. The password will be hashed with a salt for extra protection.

**3- Home Page**



After logging into their account, the user will be presented with the Home page as shown above. This page is like a hub for the main application as it allows the user to go to the 3 other main pages (Profile, Calendar and Groups) and to the Settings page. Each of the pages have a “back” button which will return the user to the Home Page when clicked by using stacks to ensure the code isn’t just repeated on each of the 3 separate pages. The “Profile”, “Calendar”, “Groups” and “Settings” buttons will send the user to their respective screens.

**4- Profile**



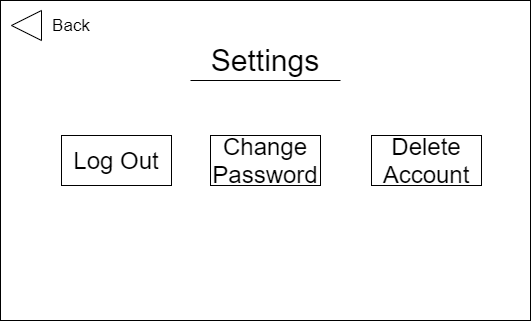
When the user clicks the “Profile” button they will load up this page. This is the user’s unique profile page for their account. It displays their username, profile picture, friend list and group list. Under the Friends will be a list of the user's friends sorted alphabetically by using a sort method. This is the same with the Groups, under the word will be the user’s groups displayed alphabetically using sort method.

The “Edit Friends list” will let the user Delete other users from their friend list or they can add a new user by using a search function.

**5- Groups**

**6- Calendar**

**7- Settings**



The settings page will show the three buttons, “Log Out”, “Change Password”, “Delete Account”. The “Delete Account” will start up the SQL Code to remove the User from the database. The “Change Password” will start the SQL Code to update the hashed password and (the randomly generated) salt in the database for the user’s account.

Finally, the “Log Out” button will return the user to the log in page and will force them to either log back in or make a new account before returning to the home page again. There is also the back button in the top left corner in order to return the user to the home page using stacks.