Design Document

Contents

Interface Design

Interface Descriptions

Desktop + Tablet + Phone – Index page

The main aspect of the index page will be the map interface and the venue toggle buttons. As the rest of the page will be static and the same on every other page on the desktop design. The map will be implemented using google maps and will have the google maps interface and features. It is this that we will build our functionality upon. Upon start up the map will center on the user and will display all events that are within the range of the user.

Then using the venue buttons the user can toggle the sight of the different types of venues. As multiple toggle buttons, can be selected the selected types of venues will be visual to the user on the map. The venue buttons will have a visual icon to tell the user what they represent but if the user hovers over the icon it will give a text name of what the icon represents. The user will also be able to scroll through the venue options as there will be multiple options to choose from.

Desktop + Tablet + Phone – Friends page

The friends page will have 2 sections the friends list, and the groups list, the friends list will display all the friends that the user has got, and will display them in order of who is online, and who is offline, displaying all members in alphabetical order in each section. Below the friends list will be 2 buttons that will be for adding and removing friends to the list. Th user will click on these buttons when they want to do one of these actions and a search bar will appear allowing the user to search for the username of the person they wish to add or remove.

The group section is the same as the friends list as it displays the groups in alphabetical order and has 2 buttons to add or remove yourself to and from groups using a search bar mechanism.

Desktop + Tablet + Phone – Events page

The events section will have a list of all the events that the user is a part of and will display the name of the event, a description of the event, and a small map cut out of where the event is. Also, there will be a create event button where the user will be able to create an event and set the parameters for the event such as location, date and time, description, event name, and sending event invitations to the user's friends. Also, there will be an option on each event for the user to delete the event and remove themselves from it.

Desktop+ Tablet + Phone – Settings page

In the settings page, there will be 2 main sections in the body of the page, the 2 parts are the account settings and the system settings. The account settings will allow the user to view and manipulate options such as; username for the account, password for the account, e-mail address for the account, and forename and surname for the account.

And the system settings will have options like colour-blind visual settings, font size settings, and resolution settings for the page. These are the types of options that we will try to implement for the user so give them the best usability.

Phone – Slide out menu

The phone has a slide out menu to compensate for the smaller workspace where the user can interact with, to solve this problem we are going to implement a slide out menu that will have a lot of the essential options that are available on the tablet and desktop versions the menu will incorporate the friends, groups, and events tabs that are normally on the side of the website. And will also include the menu options that are normally at the header.

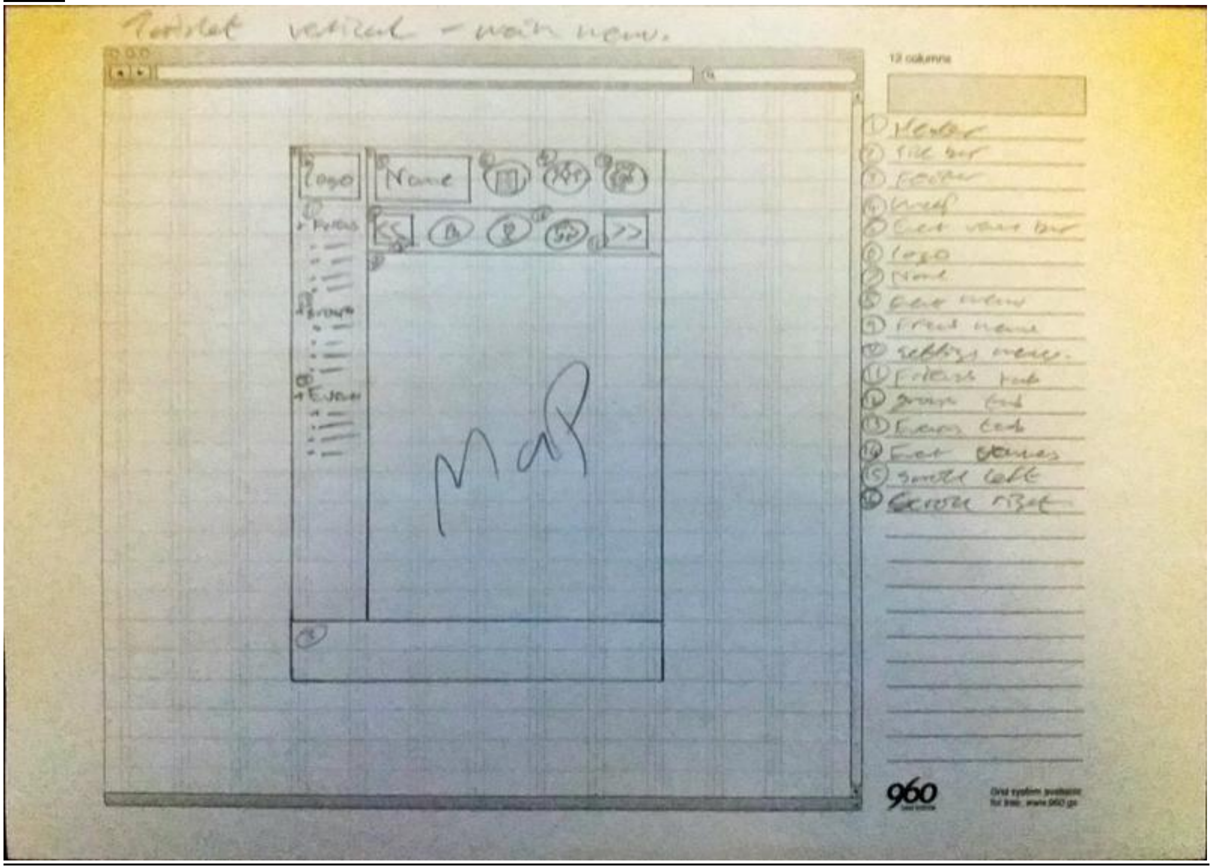
Design Illustrations

The following images are designs of the interface. They are created using the 960 method that will allow us to alter the interface dynamically according to screen size.

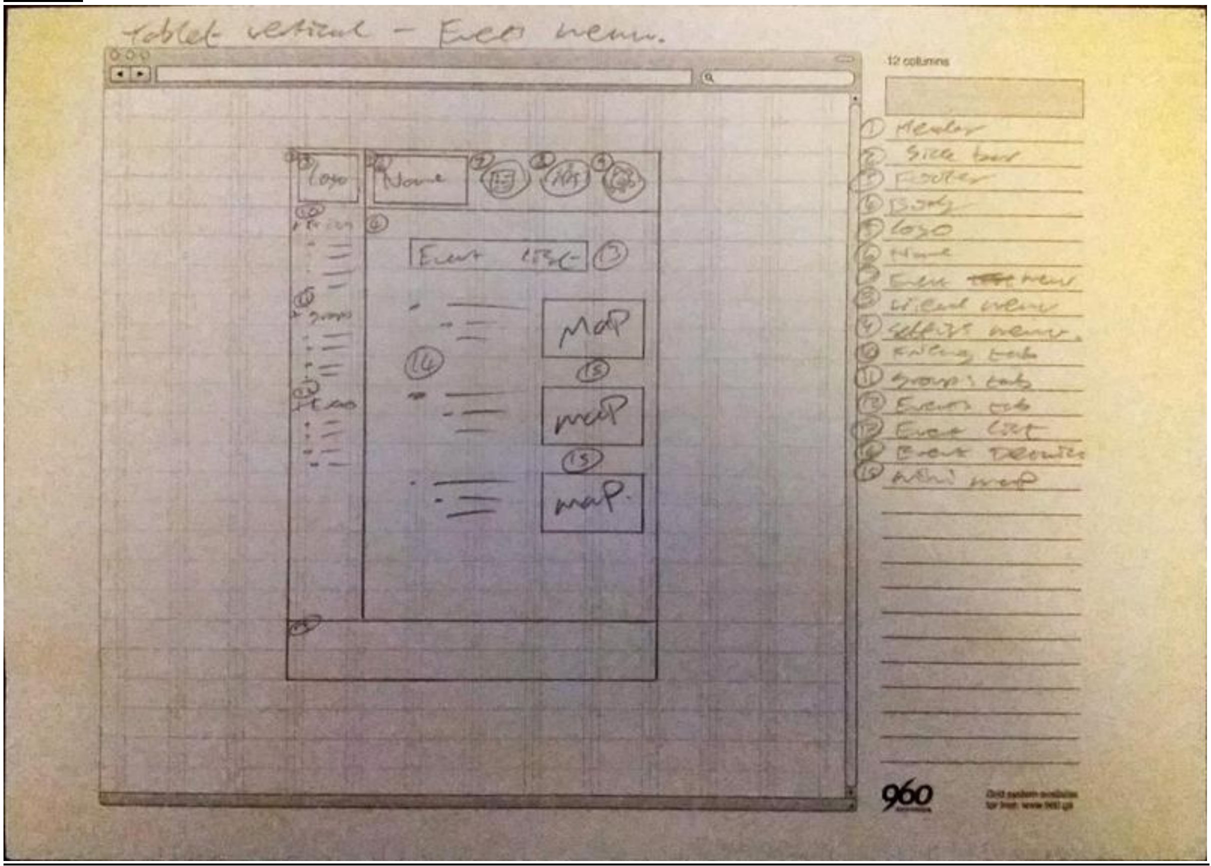
Vertical Tablet designs

As a tablet screen is big enough to support a map both horizontally and vertically there are two sets of designs. This set is for the tablet being held vertically, so that the longest edge is on the side, rather than the top and bottom of the interface.

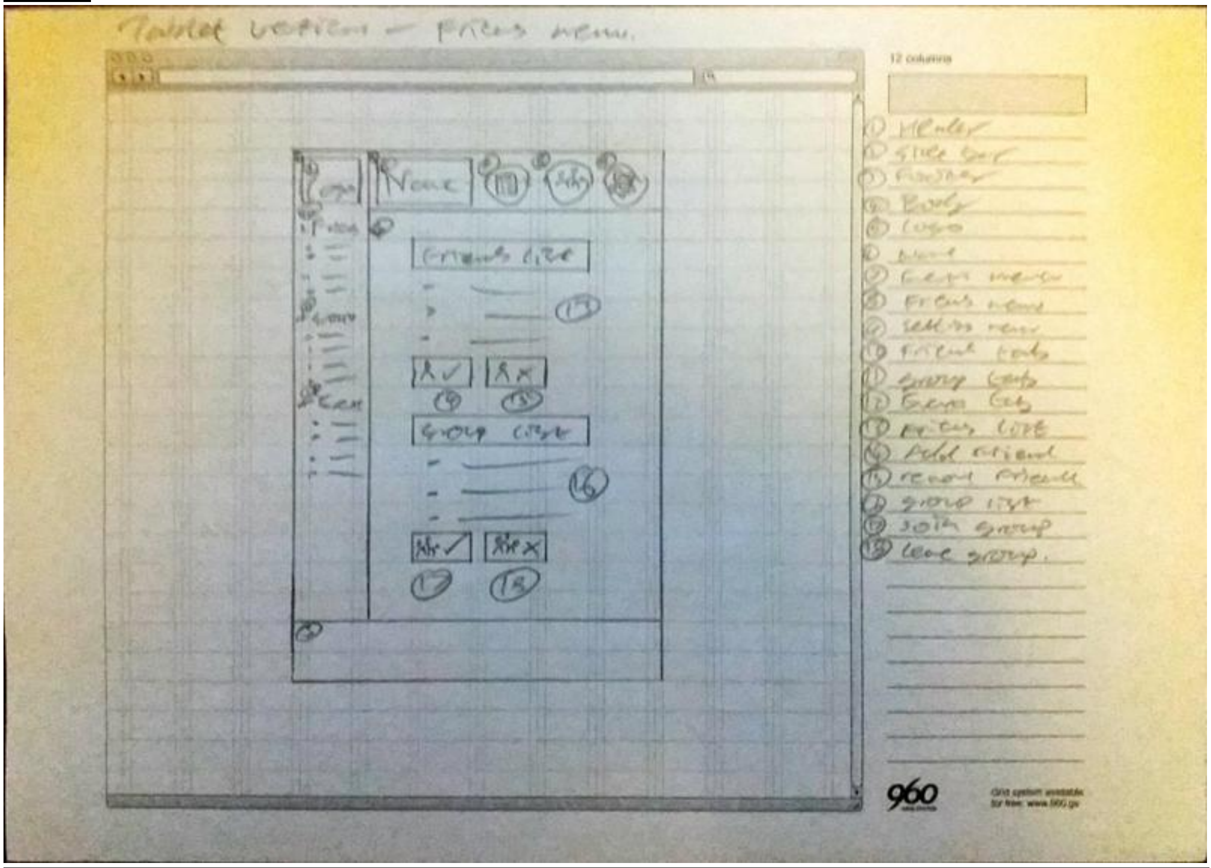
<u>Main</u>



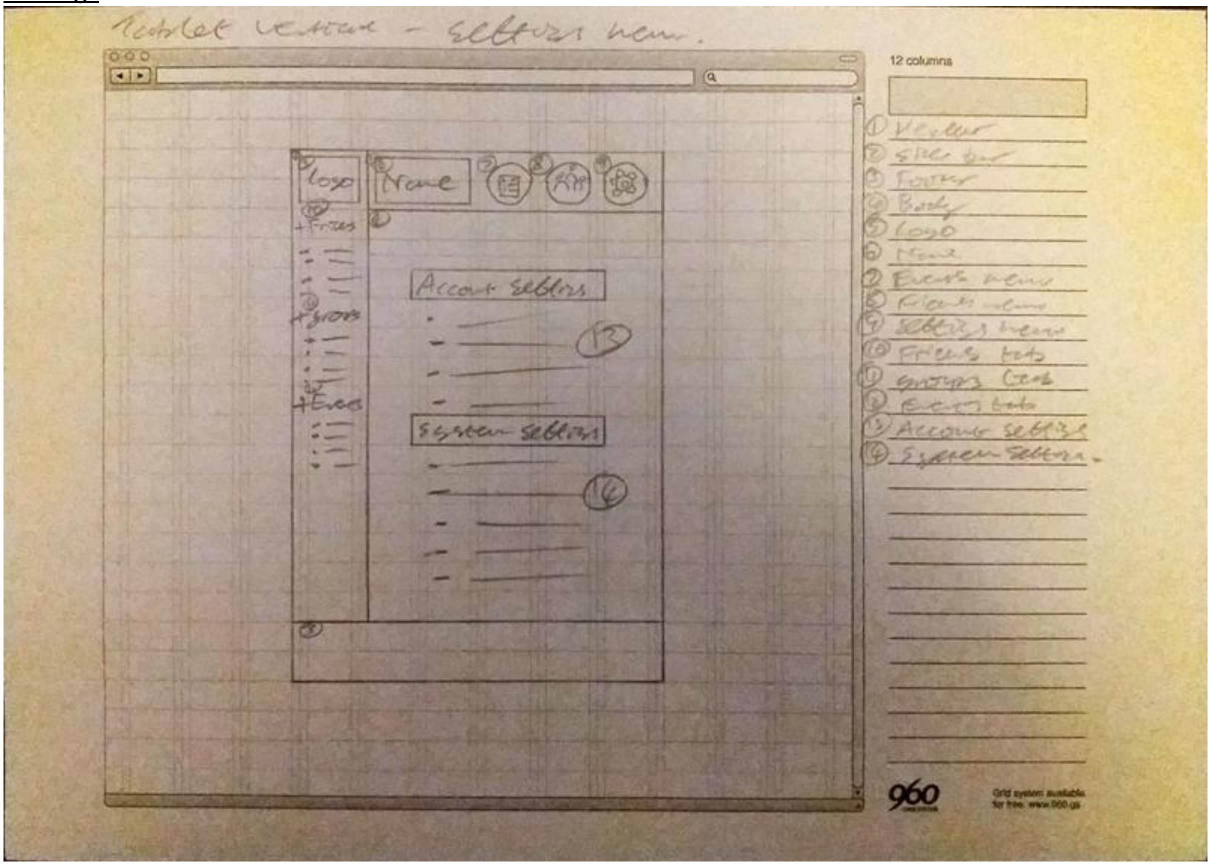
Events



Friends

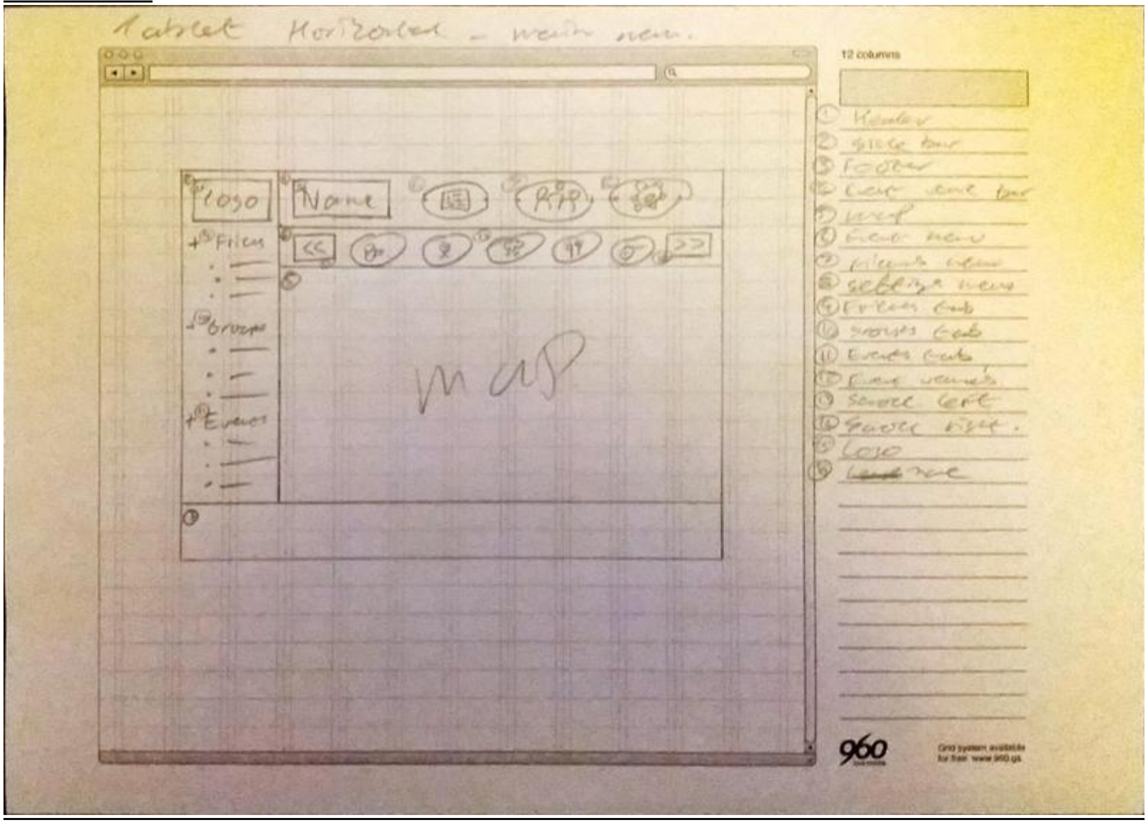


Settings

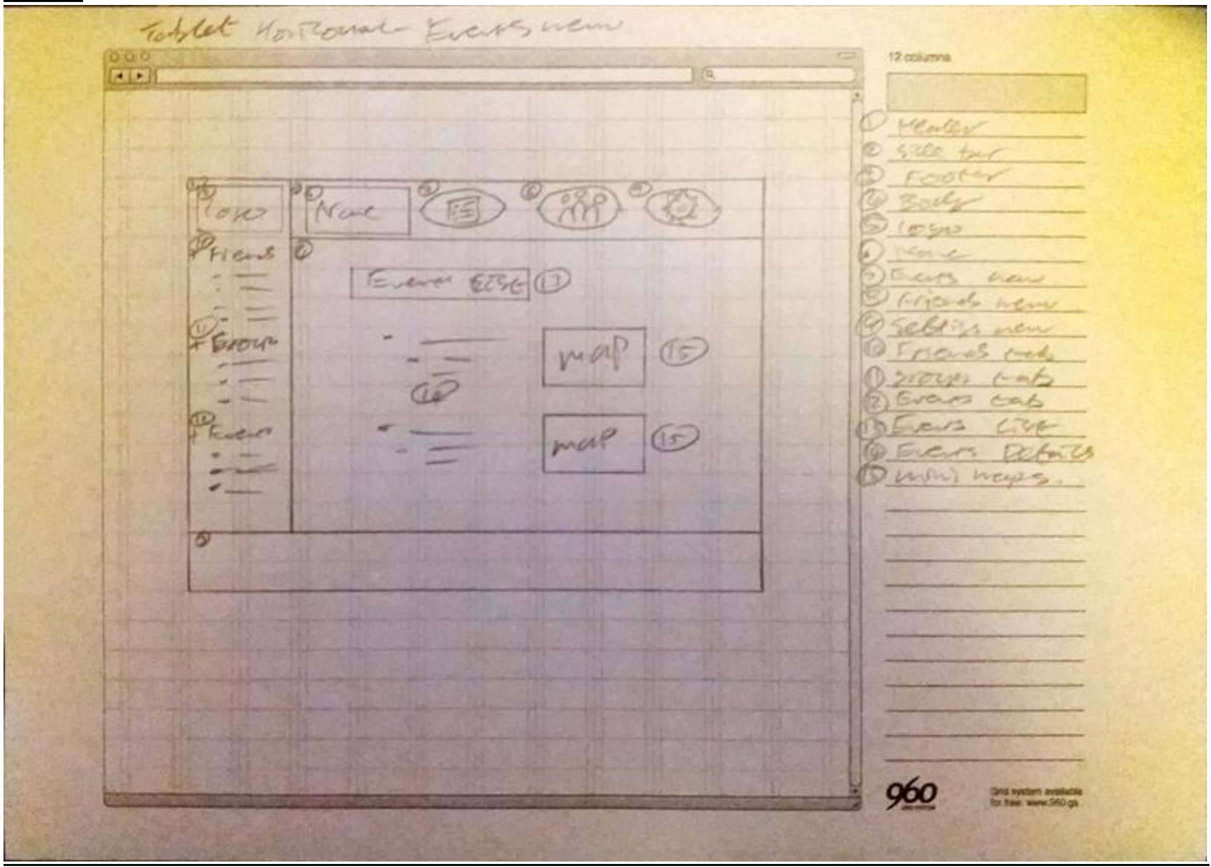


Tablet Horizontal

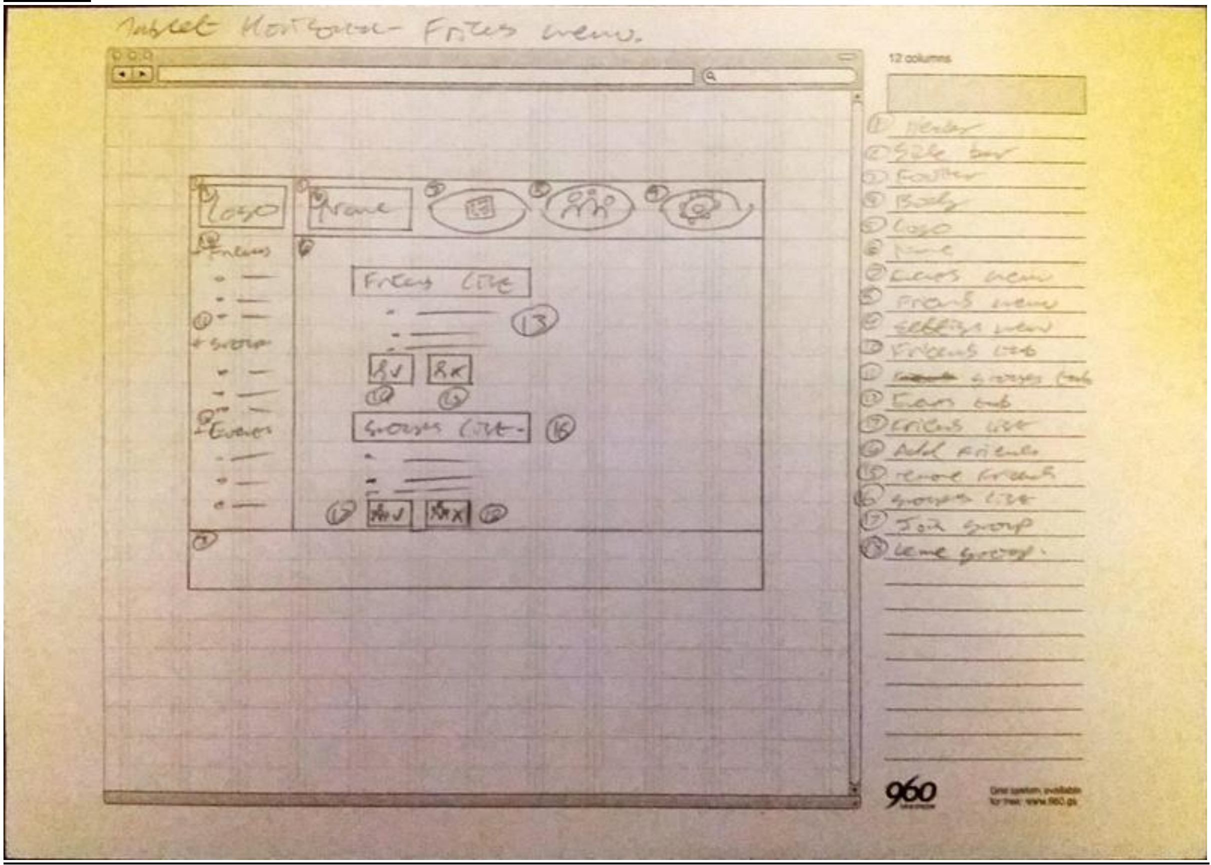
Main Screen



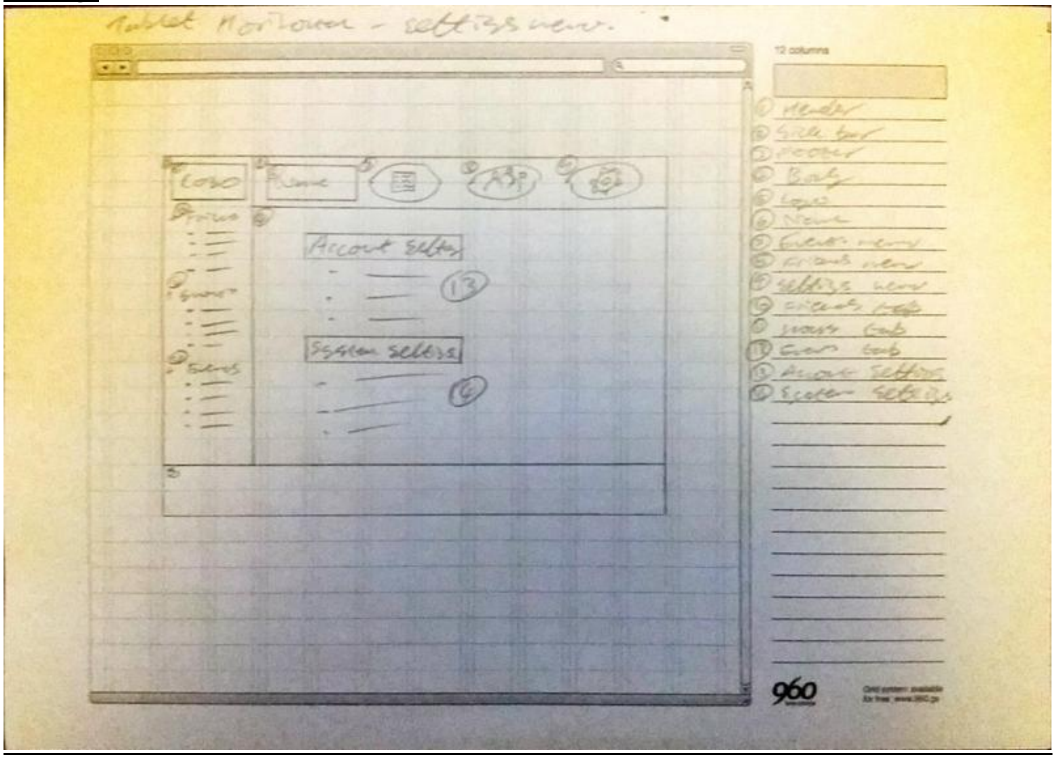
Events



Friends



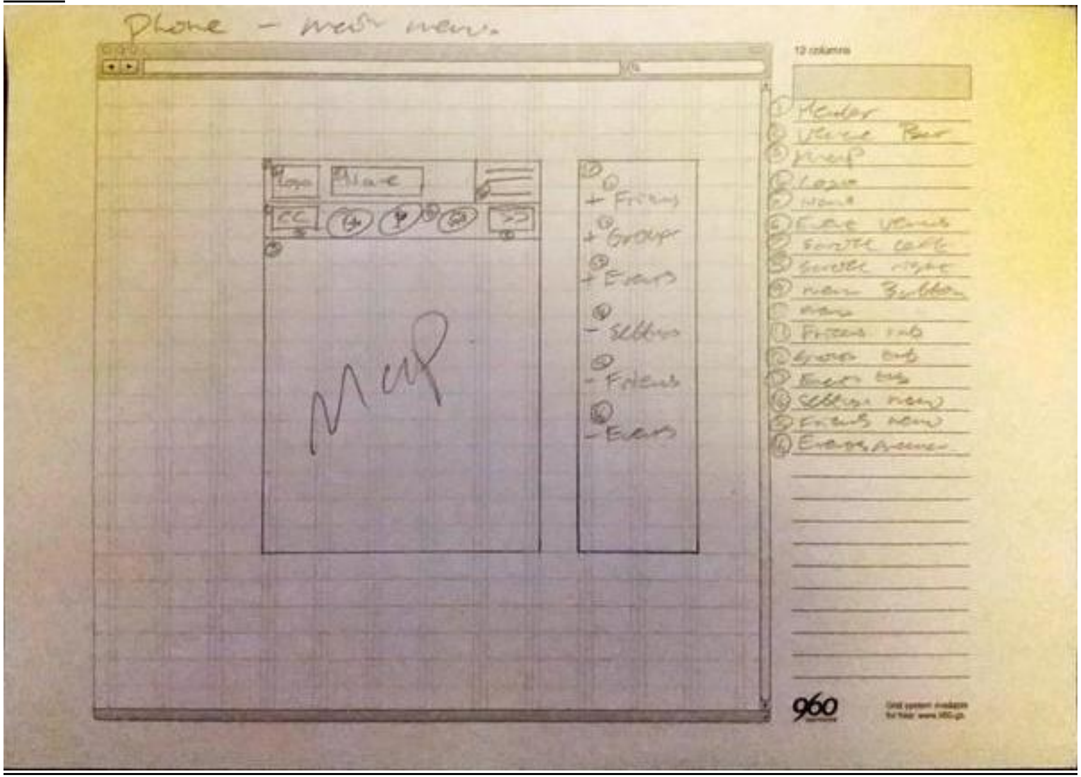
Settings



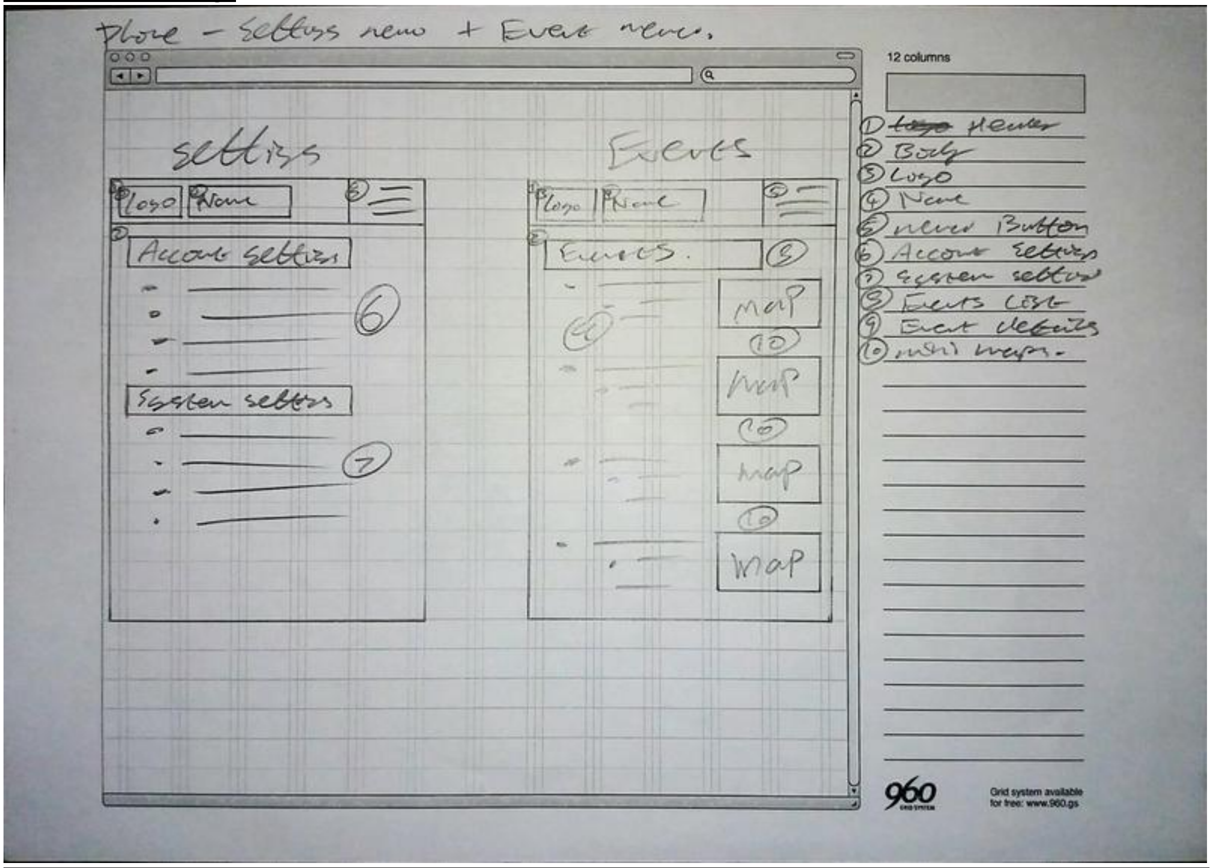
Phone Designs

Due to the size of phone screens the app wouldn't work with a phone held horizontally as there would not be enough space to fit all elements on the screen. The phone only has vertical screens created thus, each screen has a new button in the header that opens a side-bar to enable the user to change screens.

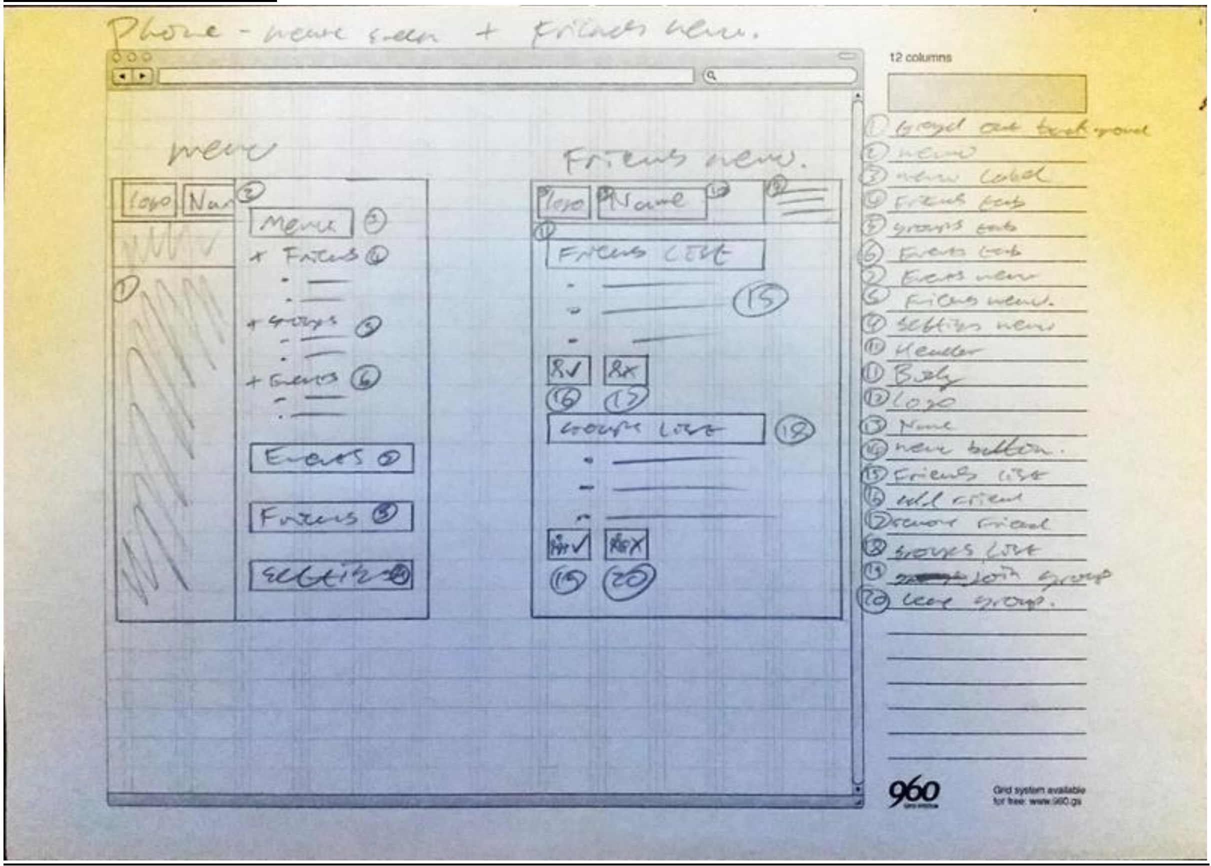
Main



Events and Settings



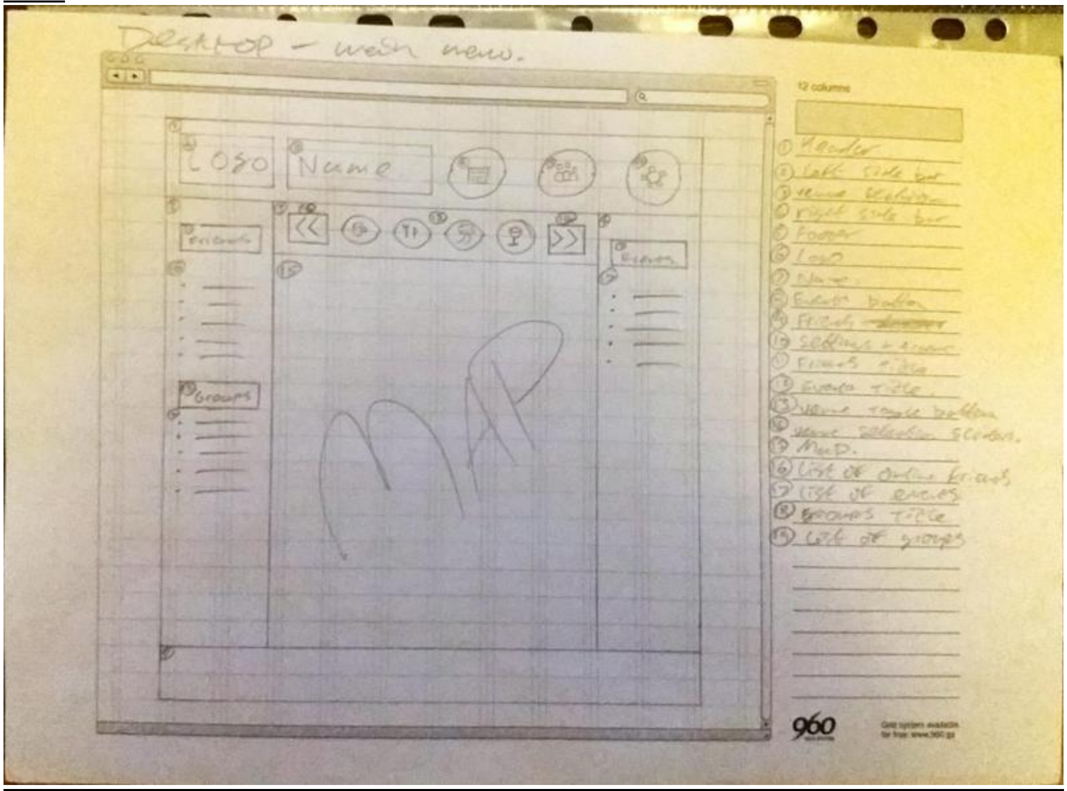
Friends and side-Menu



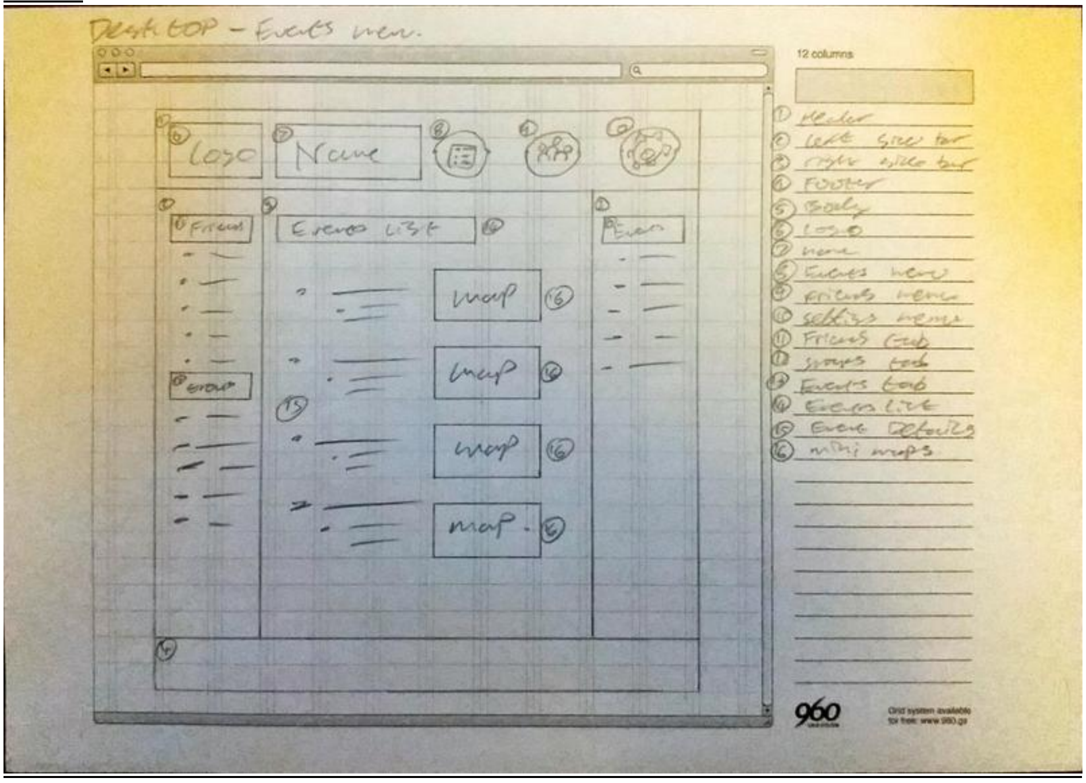
Desktop

The desktop version of the application is the only version that is mouse driven, however as the main features are based around a map the interface does not change much. As it is a mouse buttons can be smaller relative to the screen-size, and more information can be on the screen due to its larger size.

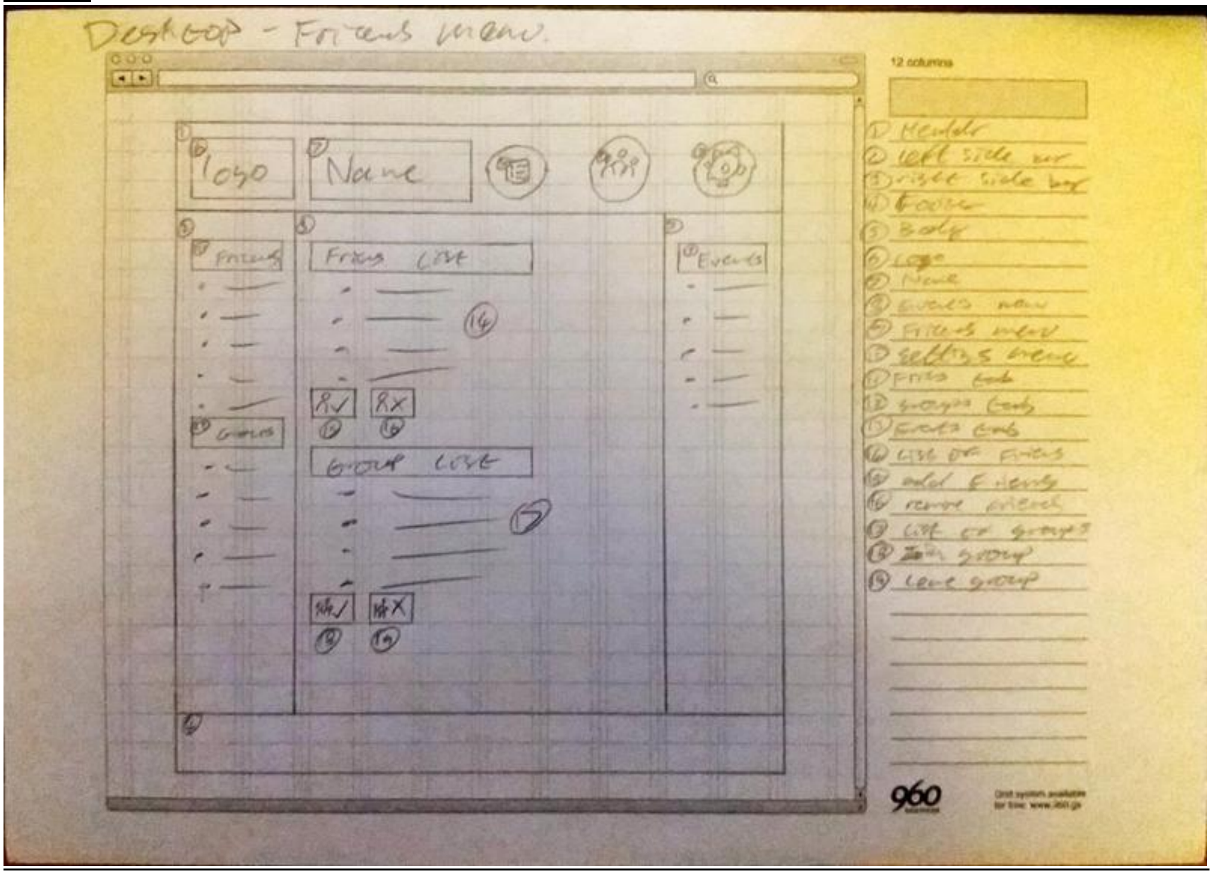
<u>Main</u>



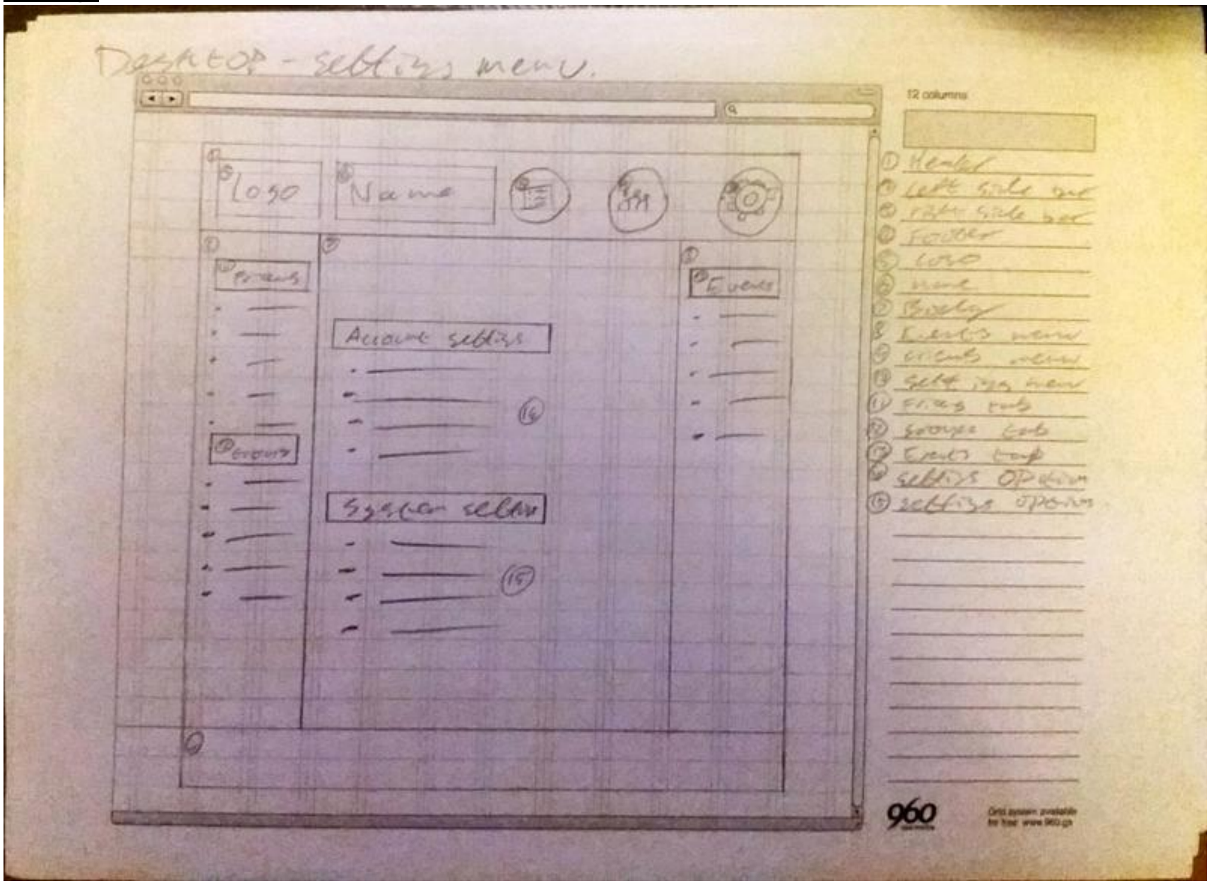
<u>Events</u>



Friends

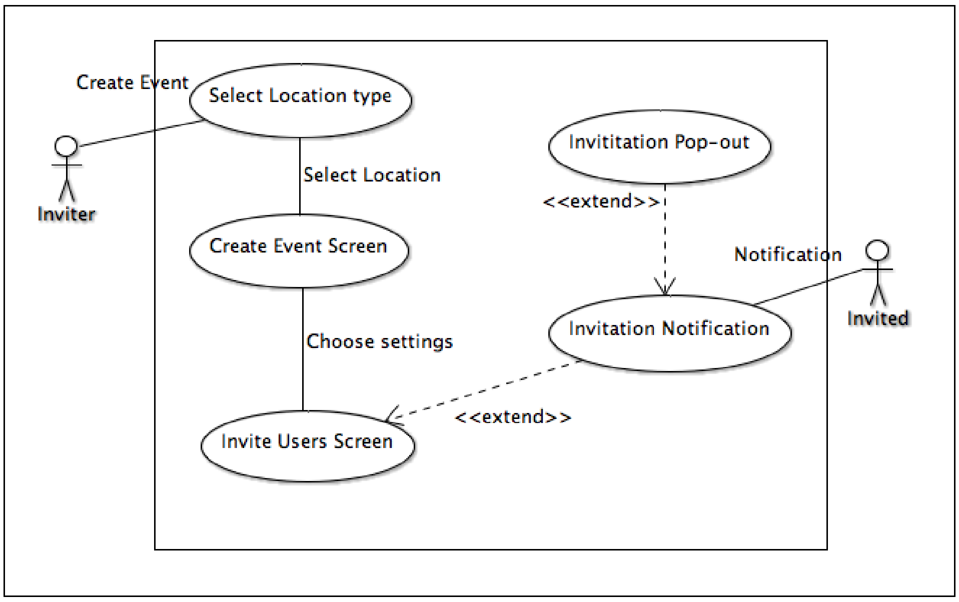


Settings



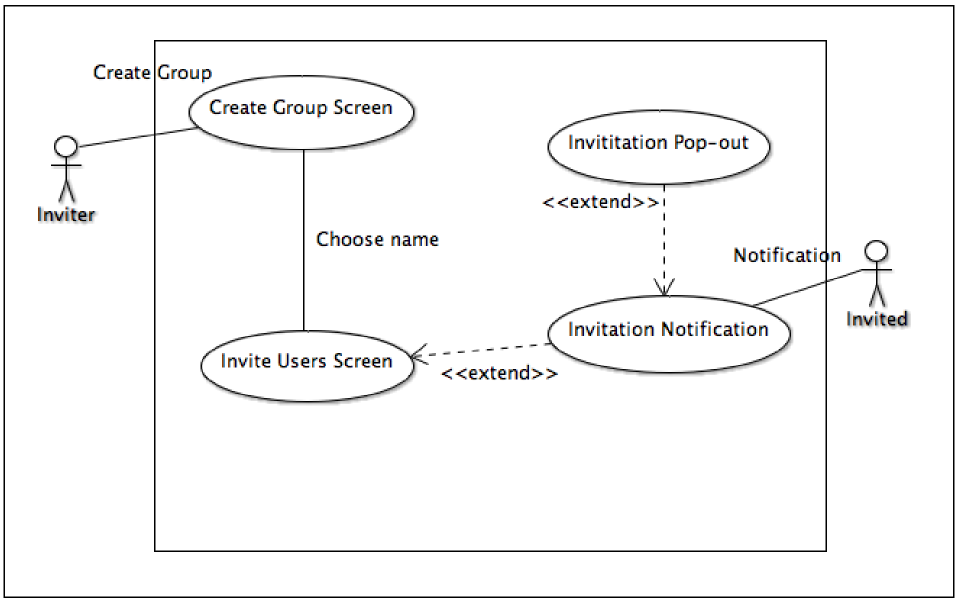
Use Case Diagrams

Inviting / getting invited to events



The User who creates the event has the option to invite users once they have set up the rest of the event. Users who are invited will receive a notification of their invitation, or if they have the app open then a pop-out window will appear informing them.

Inviting / getting invited to groups

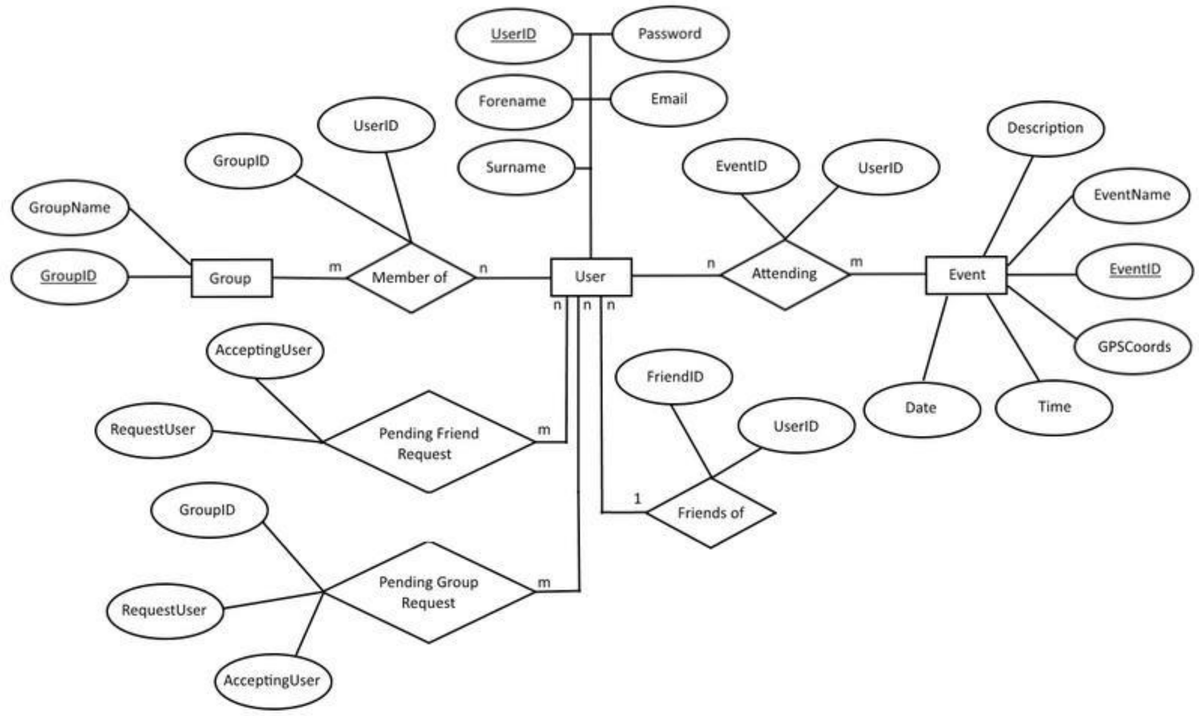


The User who creates a group can choose the name and then users to invite. Users who are invited will receive a notification of their invitation, or if they have the app open then a pop-out window will appear informing them.

More detail is provided in the Functional Descriptions.

Functional Descriptions of the System

StateChart of the System



Performing tasks on different Devices

Accessing Settings

<u>Desktop</u>

Using the link on the header

<u>Tablet</u>

Using the link on the header

<u>Mobile</u>

-an option on the side menu, accessed by tapping the button, or swiping from right to left

Accessing user settings

<u>Desktop</u>

Using the link on the header

<u>Tablet</u>

Using the link on the header

<u>Mobile</u>

-an option on the side menu, accessed by tapping the button, or swiping from right to left

Accessing Messages

<u>Desktop and tablet</u>

-Via the header on the main page

-via a friend/group page

Mobile

-Via the side menu tapping the button, or swiping from right to left
or via a friend/group page

Navigating Menus

Accessing Friends

<u>Desktop</u>
Selection Listed on the right hand bar
View all button at bottom of limited selection to take you to a new page of friends
<u>Tablet</u>
Expanding the "friends" tab on the left hand side will show a selection.
The selection will have "view all" at the bottom, taking you to a new page of firends.
<u>Mobile</u>
access the sidemenu via the top right button, or swiping from right to left
access friends via the menu in the sidebar

Accessing Groups

<u>Desktop</u>
Selection Listed on the right hand side-bar, below friends.
View all button at bottom of limited selection to take you to a new page of only groups you are in.
<u>Tablet</u>
Expanding the "groups" tab on the left hand side will show a selection.
The selection will have "view all" at the bottom, taking you to a new page of groups.
<u>Mobile</u>
access the sidemenu via the top right button, or swiping from right to left
access groups via the menu in the sidebar

Accessing Events

<u>Desktop</u>
Selection Listed on the left hand side-bar
View all button at bottom of limited selection to take you to a new page of only events you are in.
<u>Tablet</u>
Expanding the "events" tab on the left hand side will show a selection.
The selection will have "view all" at the bottom, taking you to a new page of all events.
<u>Mobile</u>
access the sidemenu via the top right button, or swiping from right to left
access events via the menu in the sidebar

Main Page

Finding all of 'x' in the area

<u>Desktop</u>
Select 'x' from the top menu using the picture corresponding to x
<u>Tablet</u>
Select 'x' from the top menu using the picture corresponding to x
<u>Mobile</u>
Select 'x' from the top menu using the picture corresponding to x

Selecting event location

<u>Desktop</u>
When all x have been displayed clicking on an x will cause a pop-out window on the screen
<u>Tablet</u>
Select 'x' from the top menu using the picture corresponding to x

Mobile
Select ‘x’ from the top menu using the picture corresponding to x

Specify custom event location

Desktop
Select “custom” from top menu using corresponding picture, then click location on map. It then prompts the user for the name of the location before taking to event creation page.
Tablet
Select “custom” from top menu using corresponding picture, then tap map, It then prompts the user for the name of the location before taking to event creation page.
Mobile
Select “custom” from top menu using corresponding picture, then tap map, It then prompts the user for the name of the location before taking to event creation page.

Create spontaneous event

Desktop
once location selected, choose spontaneous event from event creation options - pop out window on map - can choose which group/friends to alert - can choose radius of people who to alert
Tablet
once location selected, choose spontaneous event from event creation options - pop out window on map - can choose which group/friends to alert - can choose radius of people who to alert
Mobile
once location selected, choose spontaneous event from event creation options - takes user to new page - can select which group/friends to alert - can choose radius of people who to alert

Create planned event

Desktop
once location selected, choose planned event from event creation options - pop out window on map - can choose which group/friends to invite - can choose when event is, date or time, can choose message to be sent to invited users
Tablet
once location selected, choose spontaneous event from event creation options - pop out window on map - can choose which group/friends to alert - - can choose when event is, date or time, can choose message to be sent to invited users
Mobile
once location selected, choose spontaneous event from event creation options - takes user to new page - can select which group/friends to alert - can choose when event is, date or time, can choose message to be sent to invited users

Accepting an invitation to a planned event

-users have to be able to say whether they are taking a phone/tablet with the app to the event if we are tracking when they arrive

Mobile
-popup appears if active with “you have been invited to event” and goto or “later” options, later adding a (1) or something next to events and the menu button, also adding a new event under “invited” tab on events page. Goto, takes you to the event page for that event already. On that page you can accept or not.
Desktop
If active a pop out window with “you have been invited to an event appears” with details of the event in full, and the ability to accept, decline, or “later.” later moves it to the events page under

"invited" and under the events bar.
<u>Tablet</u>
If active a pop out window with "you have been invited to an event appears" with details of the event in full, and the ability to accept, decline, or "later." later moves it to the events page under "invited" and under the events tab on the sidebar.

Accepting an invitation to a spontaneous event
<u>Mobile</u>
-popup appears if active with "X is having a meet up at y, do you want to go? y is z miles away" and yes or no. If inactive a notification is sent with the same text. The options are yes or no and once dismissed will permanently disappear.
<u>Desktop</u>
If active a pop out window with "X is having a meet up at y, do you want to go? y is z miles away" with details of where it is happening in full. Options are yes or no. Once dismissed will permanently disappear.
<u>Tablet</u>
If active a pop out window with "X is having a meet up at y, do you want to go? y is z miles away" with details of where it is happening in full. If inactive a notification is sent with the same text. Options are yes or no. Once dismissed will permanently disappear.

Friends/groups Page

Sending an Invite
<u>Mobile</u>
tap the Add friend button. A pop-out window will appear and prompt for a userID, once entered it will send a request to that person.
<u>Desktop</u>
click the Add friend button. A pop-out window will appear and prompt for a userID, once entered it will send a request to that person.
<u>Tablet</u>

Accepting/denying a friend invite.
<u>Mobile</u>
Below the "friends list" tab will be a button with "invites" and tapping that will take you to a screen of invites and tapping on a person will bring up more options including accepting or denying that friend/
<u>Desktop</u>
At the top of the "friends list" tab will be "invitations" with a list of users and an x or a tick next to their name. Click the tick to accept, click the x to deny.
<u>Tablet</u>

Joining a group
<u>Mobile</u>
Below the "groups list" tab will be a button with "invites" and tapping that will take you to a screen of invites and tapping on a group will bring up more options including accepting or denying.
<u>Desktop</u>
At the top of the "groups list" tab will be "invitations" with a list of groups and an x or a tick next to their name. Click the tick to accept, click the x to deny.
<u>Tablet</u>
At the top of the "groups list" tab will be "invitations" with a list of groups and an x or a tick next to

their name. Tap the tick to accept or the x to deny.

Leaving a group

Mobile

tap the leave group button, a list of groups will appear. Tap a group and a popup will appear asking if you want to leave. Tap yes to leave the group.

Desktop

Select the leave group button. A pop-out window will appear with a list of all groups and an x next to them. Click the x to leave the group (after confirming.)

Tablet

Select the leave group button. A pop-out window will appear with a list of all groups and an x next to them. Tap the x to leave the group (after confirming.)

Create a Group

Mobile

Create Group Button from groups page. Choose name and then send invites.

Desktop

Create Group Button from groups page. Choose name and then send invites.

Tablet

Create Group Button from groups page. Choose name and then send invites.

Events Page

Accepting an Invite

Mobile

Tap on an event to bring up a larger window with more detail about it and an accept/deny button is on that screen.

Desktop

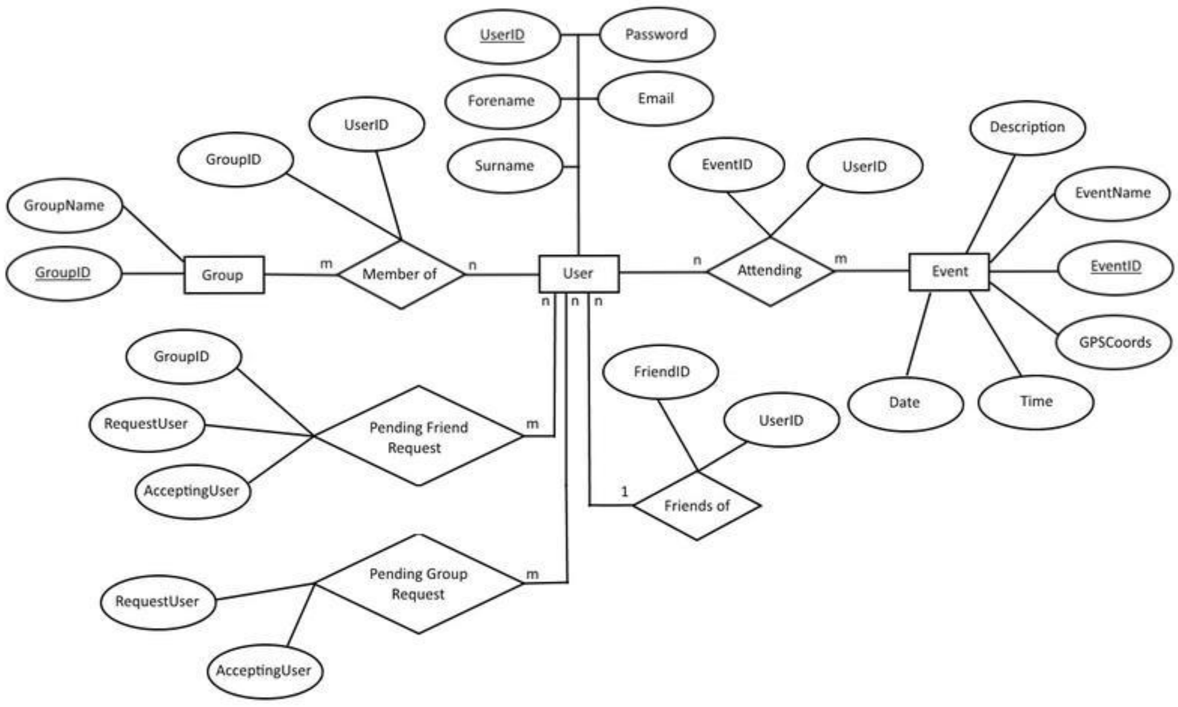
Under the Events list tab will be an “invited” list displaying events like in the attending section but with a tick or cross below to accept or deny the request.

Tablet

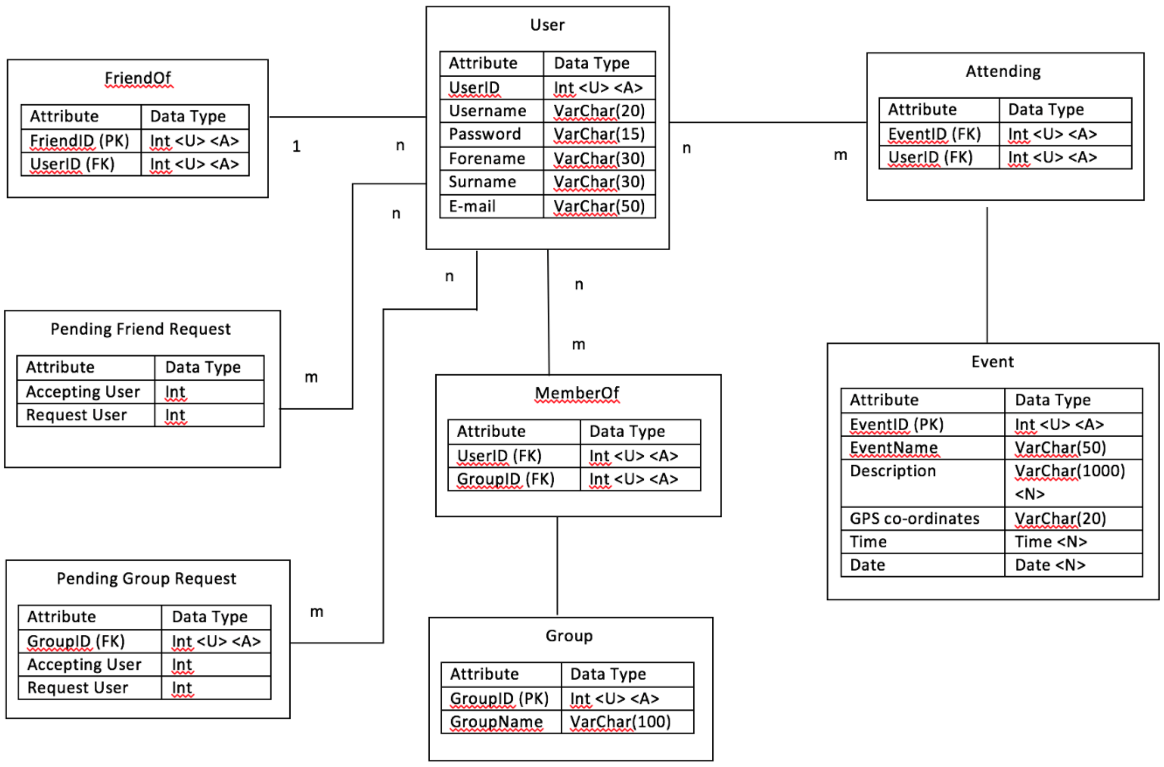
Under the Events list tab will be an “invited” list displaying events like in the attending section but with a tick or cross below to accept or deny the request.

Database Design

Conceptual Design



Physical Design



FriendOf

Attribute	Data Type
FriendID (PK)	Int <U> <A>
UserID (FK)	Int <U> <A>

User

Attribute	Data Type
UserID	Int <U> <A>
Username	VarChar(20)
Password	VarChar(15)
Forename	VarChar(30)
Surname	VarChar(30)
E-mail	VarChar(50)

Attending

Attribute	Data Type
EventID (FK)	Int <U> <A>
UserID (FK)	Int <U> <A>

Pending Friend Request

Attribute	Data Type
Accepting User	Int
Request User	Int

MemberOf

Attribute	Data Type
UserID (FK)	Int <U> <A>
GroupID (FK)	Int <U> <A>

Event

Attribute	Data Type
EventID (PK)	Int <U> <A>
EventName	VarChar(50)
Description	VarChar(1000) <N>
GPS co-ordinates	VarChar(20)
Time	Time <N>
Date	Date <N>

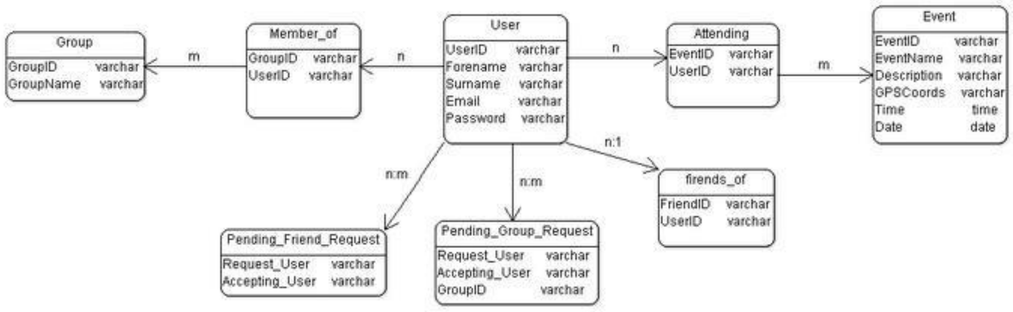
Pending Group Request

Attribute	Data Type
GroupID (FK)	Int <U> <A>
Accepting User	Int
Request User	Int

Group

Attribute	Data Type
GroupID (PK)	Int <U> <A>
GroupName	VarChar(100)

Logical Design



Data Dictionary.

User:

Entity Name	Caption Name	Field Type	Field Length	Can be null	Primary Key	Description
user_id	UserID	INT NOT NULL	10	False	True	For uniquely identifying someone in our database if they have the same name as someone else for example.
f_name	First Name	VARCHAR	20	False	False	For the profile of the user, this will be their show name to their friends, other group/event members etc.
s_name	Surname	VARCHAR	20	False	False	For the profile of the user, this will be their show name to their friends, other group/event members etc.
email	E-Mail	VARCHAR	50	True	False	This Is for contacting the user by email in case they want to change their password or there is an issue with their account.
password	Password	VARCHAR	50	False	False	This is so we can check the password the user entered to log in with the stored one then we know if it is them trying to access their account.

Group:

Entity Name	Caption Name	Field Type	Field Length	Can be null	Primary Key	Description
group_id	Group ID	INT NOT NULL	10	false	True	For uniquely identifying the group as there will be many.
group_name	Group Name	VARCHAR	20	false	False	For storing the name of the group, we will need this so we can keep a record and users can see the name.

Event:

Entity Name	Caption Name	Field Type	Field Length	Can be null	Primary Key	Description
event_id	Event ID	INT NOT NULL	10	False	True	For uniquely identifying the events as there will be many.
gps_coord	GPS Co-ordinates	FLOAT	10,6	True	False	To store the location of the user as temporary data so we can use the information to find people close to them.
event_name	Event Name	VARCHAR	20	False	False	For storing the name of the event, we will need this so we can keep a record and users can see the event name.
event_description	Event Members	VARCHAR	50	true	false	To store the description of the event.
event_time	Event Time	INT	30	false	false	To store the time of the event.
event_date	Event Date	Date	30	false	false	To store the date of the event.

Friends_of:

Entity Name	Caption Name	Field Type	Field Length	Can be null	Primary Key	Description
user_id	UserID	INT NOT NULL	10	False	True	For uniquely identifying the user.
friend_id	FriendID	INT NOT NULL	10	False	False	For uniquely identifying the friend of that user.

Attending

Entity Name	Caption Name	Field Type	Field Length	Can be null	Primary Key	Description
event_id	EventID	INT NOT NULL	10	False	True	For uniquely identifying the event.
user_id	UserID	INT NOT NULL	10	False	False	For uniquely identifying the user.

Member_of

Entity Name	Caption Name	Field Type	Field Length	Can be null	Primary Key	Description
Group_id	GroupID	INT NOT NULL	10	False	True	For uniquely identifying the group.
User_id	UserID	INT NOT NULL	10	False	False	For uniquely identifying the user.

Pending_group_request

Entity Name	Caption Name	Field Type	Field Length	Can be null	Primary Key	Description
Request_user	Request User	BOOL	N/A	true	True	To store when there is a request for the user from a group.
Accepting_user	Accepting User	BOOL	N/A	true	False	To store when the user accepts the request.
Group_id	GroupID	INT NOT NULL	10	False	False	For uniquely identifying the group.

Pending_friend_request

Entity Name	Caption Name	Field Type	Field Length	Can be null	Primary Key	Description
Request_user	Request User	BOOL	N/A	true	False	To store when there is a friend request for the user.
Accepting_user	Accepting User	BOOL	N/A	true	False	To store when the user accepts the friend request.

Creation process into 3NF

This segment of the Document shows how we got from the initial list of attributes we needed to the 3rd normal form tables that were used to create Entity-Relationship Diagrams and the Designs of the program.

All
User_ID
Group_ID
Event_ID
Name
Email
Password
GPS_coord
Time
Date
Description

(UNF to 1NF)

- Eliminate repeating groups in individual tables
- Create separate table for each set of related data
- Identify each set of related data with a key

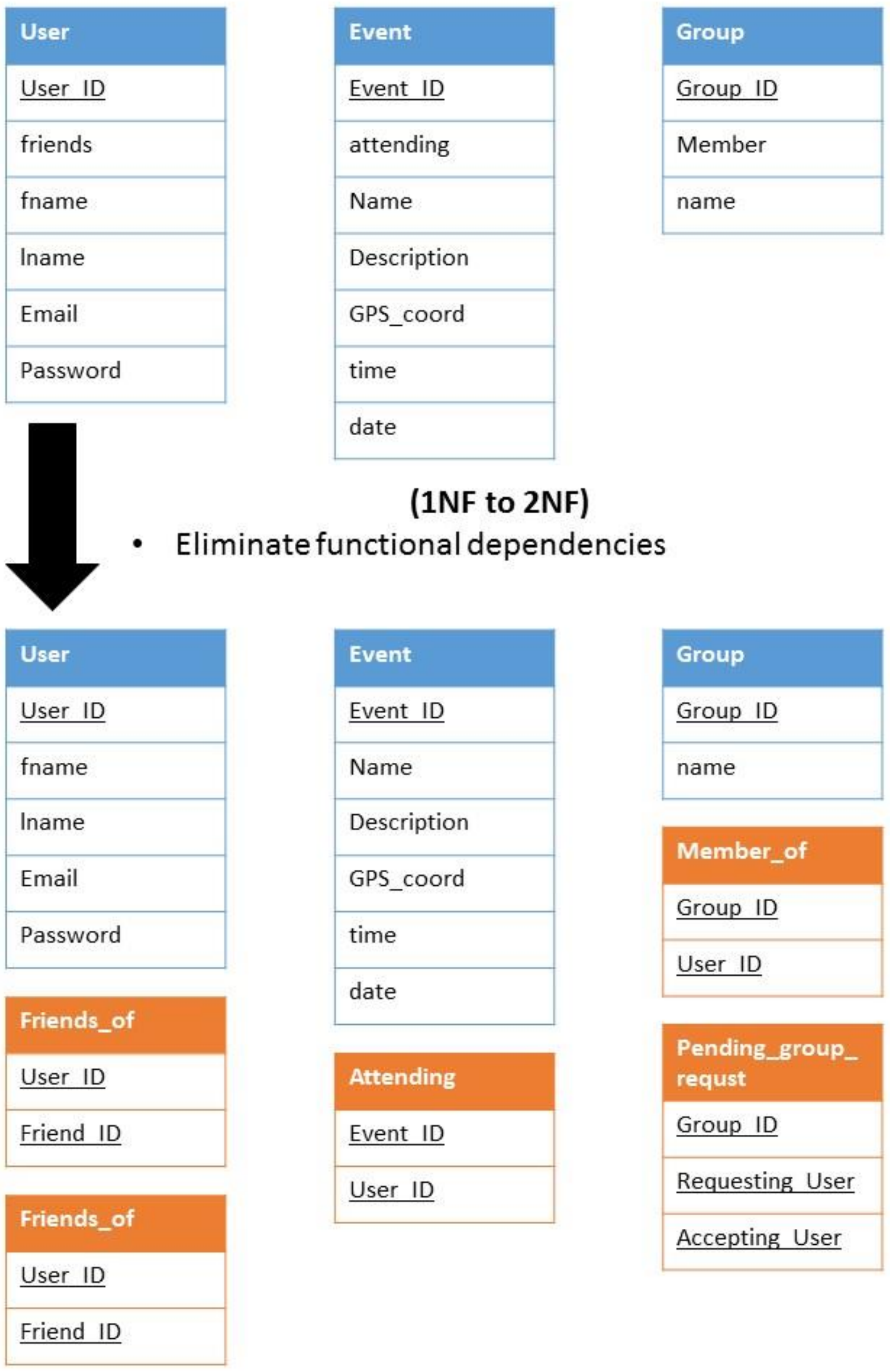
User
User_ID
friends
fname
lname
Email
Password

Event
Event_ID
attending
Name
Description
GPS_coord
time
date

Group
Group_ID
Member
name

(1NF to 2NF)

- Eliminate functional dependencies



User
User_ID
fname
lname
Email
Password

Friends_of
User_ID
Friend_ID

Friends_of
User_ID
Friend_ID

Event
Event_ID
Name
Description
GPS_coord
time
date

Attending
Event_ID
User_ID

Group
Group_ID
name

Member_of
Group_ID
User_ID

Pending_group_ requst
Group_ID
Requesting_User
Accepting_User

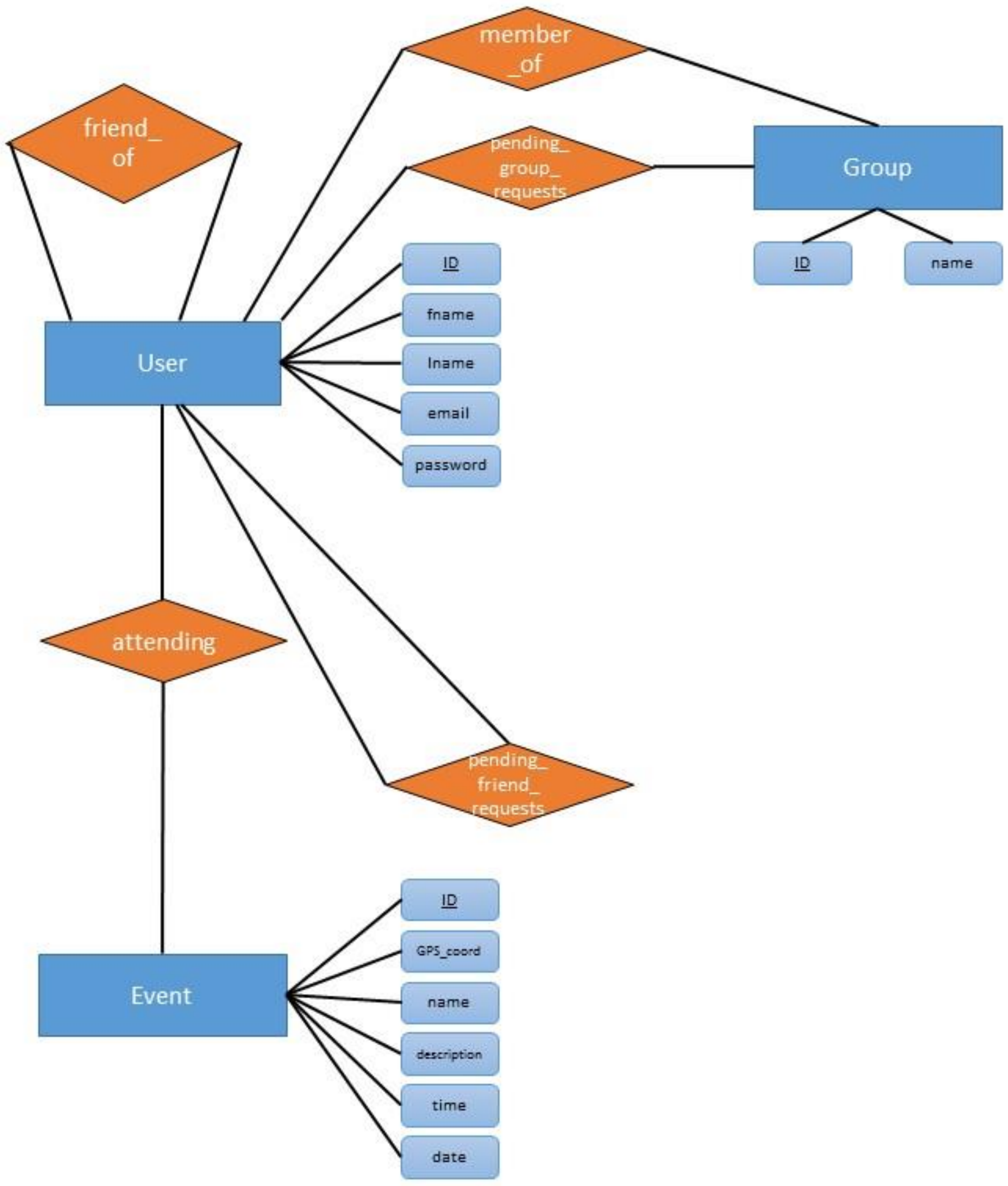
(2NF to 3NF)

- Every non-prime attribute of R is non-transitively dependent on every key of R

Tables are already in 3NF

Final Entity-Relationship Diagram

The entity relationship diagram created using the Third normal form tables created.



Queries

A selection of pseudocode queries have been made to plan out how to perform various tasks using the database.

Accept friend request:

```
INSERT INTO friend_of (userID, friendID) VALUES ($session.ID, $friendID);
INSERT INTO friend_of (friendID, userID) VALUES ($friendID, $session.ID);
```

Get password:

```
SELECT ‘password’ FROM user WHERE ‘$session.ID’ = ‘ID’
```

Update password:

```
UPDATE user SET ‘user.password’ = ‘$passowrd’
```

Decline friend request:

```
DELETE FROM pending_friend_request WHERE ‘$session.ID’ = ‘requesting_userID’ AND
‘$friendID’ = ‘accepting_userID’
```

Get list of friends:

```
SELECT name FROM user WHERE $userID IN
        SELECT * FROM friend_of WHERE ‘$session.ID’ = ‘userID’
```

Send friend request:

```
INSERT INTO pending_friend_request (requesting_userID, accepting_userID) VALUES
($session.ID, $friendID)
```

Create group:

```
INSERT INTO group (name) VALUES ($groupName);
```

Delete group:

```
DELETE FROM group WHERE ‘$groupName’ = ‘name’
```

Get list of group requests:

```
SELECT name FROM group WHERE groupID IN
        (SELECT groupID FROM pending_group_request WHERE ‘$session.ID’ =
‘accepting_userID’)
```

Send group invite:

```
INSERT INTO pending_group_request (groupID, requesting_userID, accepting_userID)
VALUES ($group, $session.ID, $friendID)
```

Get list of events:

```
SELECT name FROM event WHERE $session.ID IN
        (SELECT userID FROM attending)
```

Get list of groups:

```
SELECT * FROM member_of WHERE ‘$session.ID’ = ‘userID’
```

User Data Handling

This part of the document justifies why we need to have the data we are storing and how we are going to be using that data. We will need this information from the user for our web app to function properly. The user should know what data is being use and how.

Data needed from the user:

Unique ID:
For each user we will need to store a unique identifier in case two people have the same name or something like that so we will be using the ID to refer to people if someone has an issue ect. This will make it much simpler to find people in a database so it is the most important piece of information we will be storing, as all other information on the person will only be identifiable through this ID.
First Name:
We will need the user to input their name as this will be a profile, we need people to input real names as their friends need to know it is the correct person meeting up with them. We will store the first name as this is what we will need to refer to them as in the email if we need to contact them. It is easier to store this that look up the person on the profile every time when a record will already be there and stored for us.
Surname:
We don't have to store the surname as we can just refer to the person with their first name on the emails we send to them but we could still store this information for security reasons.
E-mail:
The user will use their email to log into the app so we need to store this information so we can check whether what the user is typing in matches what we have in the database. If it does, then they can log in (if they also get the password correct too). We can also send offers via email to the person and if they need to reset their password or anything for their account options we can pull up their email from the database and use it.
Password:
We will need to store the user’s password in the database of course because we need to check whether the user is typing the correct credentials into the log in page and it is indeed them and not someone malicious. We can encrypt this information in the database but that is for later in the development stage.

GPS co-ordinates:
We need to store the GPS co-ordinates of the user as when they log in the map will be focused on that user so we need the co-ordinates to point them out on the map. This information could also be encrypted to prevent other people finding out your location.

Pseudocode

Password

<u>Change password:</u>

```
var pw = value of password field
var pw_new = value of new
password field
var pw_new_repeat = value of new password re-enter field

if ( change password form submitted )

   var result = query db ( users, session.name, pw )

    if ( result == 0 )

        display "old password incorrect"

    elif ( result.pw == pw )

        display "old password incorrect"

    elif ( pw_new != pw_new_repeat )

        display "new passwords don't match"

     else

        pw_new = escape string ( pw_new )
         set 'pw_new', 'pw' for 'session.name' in db for 'users'
     display "password successfully reset"
```

<u>Forgot password:</u>

```
if ( forgot password button pressed )

   var addr = query db ( users, un, email )

    if ( addr == 0 )

        print "error: can't find email address"
```

```
else

        print "we've sent an email to you containing your password"
          construct email usingaddr
     send email
```

Login/out

<u>Login:</u>

```
var MAX_TRYS = 5
var un = value of username
field var pw = value of
password field var attempt

if ( login form submitted AND timer == 0 )

    if ( NOT valid ( un ) )

         increment attempt
         display "invalid username" message

    elif ( NOT valid ( pw ) )

         increment attempt
         display "invalid password" message

     else

          un = escape string (
un )             pw = escape
string ( pw )

         var result = query db ( users, un, pw )

          if ( result == 1 )

                var session
     session.name = un
print "successful login"
               navigate to home screen

          else

               increment attempt
               display "incorrect login" message

    if ( attempt >= MAX_TRYS )

         timer = 5 minutes
         attempt = 0

if ( timer > 0 )

     display "too many incorrect attempts, try again later"
message    detach login form from DOM
```

<u>Logout:</u>

```
 unset all session
varialbes delete
session cookie
destroy session
```

Friends

Display friends list:

```
... html stuff
<UL> start tag for friends list

var friends = query db for 'session.name' in table 'friends_with'

foreach ( friend in friends )

     print "<LI>" + friend +
"<BUTTON></LI>"
</UL> end tag for friends list
... html stuff
```

Display friend request list:

```
... html stuff
<UL> start tag for friend_requst list

var friend_rqsts = query db for 'session.name' in table
'friend_requests'

foreach ( rqst in friend_rqsts )

     print "<LI>" + rqst.sender + other info + "<BUTTON>" +
"<BUTTON>" + "</LI>"

</UL> end tag for friend_requst list
... html stuff
```

Add Friend:

```
var un = value of username field

if ( friend request form submitted )

   var request = query db for 'un' in table 'friend_requests'

     if ( request == 0 )
```

```
            insert 'un', 'session.name' into db for
'friend_requests'
```

Accept friend request:

```
var rqst = value of request accept button

if ( friend request accept button pressed )

   insert 'session.name', 'rqst' in db for 'friends_with'
```

Declinefriend request:

```
var rqst = value of request decline button

if ( friend request decline button pressed )

   delete 'rqst', 'session.name' in db for 'friend_requests'
```

Groups

<u>Displaygroup list:</u>

```
... html stuff
<UL> start tag for group_requst list

var groups = query db where 'session.name' = 'un' in
'group_requests'

foreach ( group in groups )

      display "<LI>" + group.name +
"<BUTTON><BUTTON></LI>"
</UL> end tag for group_requst list
... html stuff
```

<u>Displaygroup invite list:</u>

```
... html stuff
<UL> start tag for group_requst list

var groups = query db where 'session.name' = 'un' in
'group_requests'

foreach ( group in groups )

   display "<LI>" + group.name + "<BUTTON><BUTTON></LI>"

</UL> end tag for group_requst list
... html stuff
```

<u>Invite to group:</u>

```
 var un = value of
username field var group =
requesting group

if ( group invite request form submitted )

      var request = query db for 'un' and 'group' in table
'group_requests'
```

```
     if ( request == 0 )

         insert 'un' and 'group' into db for 'group_requests'
```

Create group:

```
var group_name = value of group name field

if ( create group form submitted )

    var result = query db (groups, group_name)

     if ( result == 0 )

         insert 'group_name', 'session.name' in db for 'groups'

else

          display "group already exists"
```

Delete group:

```
var group_name = $THIS.groupname

if ( delete group button pressed )

    delete 'group_name' in 'groups'
```

Events

Display list of events:

```
... html stuff
<UL> start tag for events list

var events = query db for 'session.name' in table 'attending'

foreach ( event in events )

      print "<LI>" + event + other info +
"</LI>"
</UL> end tag for events list
... html stuff
```

Development Stage Time-Plan

18/3	Skeleton Prototype Dev.	Testing	Documentation
…			
23/3	Skeleton Prototype Dev.	Testing	Documentation
24/3	Group Meeting: Prototype Iteration		
25/3	Interface Prototype Dev.	Testing	Documentation
…			
30/3	Interface Prototype Dev.	Testing	Documentation
31/3	Group Meeting: Prototype Iteration		
1/4	Back-end Prototype Dev.	Testing	Documentation
…			
6/4	Back-end Prototype Dev.	Testing	Documentation
7/4	Group Meeting: Prototype Iteration		
8/4	Database Dev.	Testing	Documentation
…			
13/4	Database Dev.	Testing	Documentation
14/4	Group Meeting: Prototype Iteration		
15/4	Front/back end integration	Back end db integration	Testing
…			
20/4	Front/back end integration	Back end db integration	Testing
21/4	Group Meeting: Prototype Iteration		
22/4	System Integration	Testing	Documentation
…			
25/4	System Integration	Testing	Documentation
26/4	Group Meeting: Final Iteration and System Review		
27/4	Demo Preparation		
28/4	Demo Submission		
29/4	Portfolio Dev.		
…			
10/5	Portfolio Dev.		
11/5	Group Meeting: Portfolio Review		
12/5	Portfolio Submission		