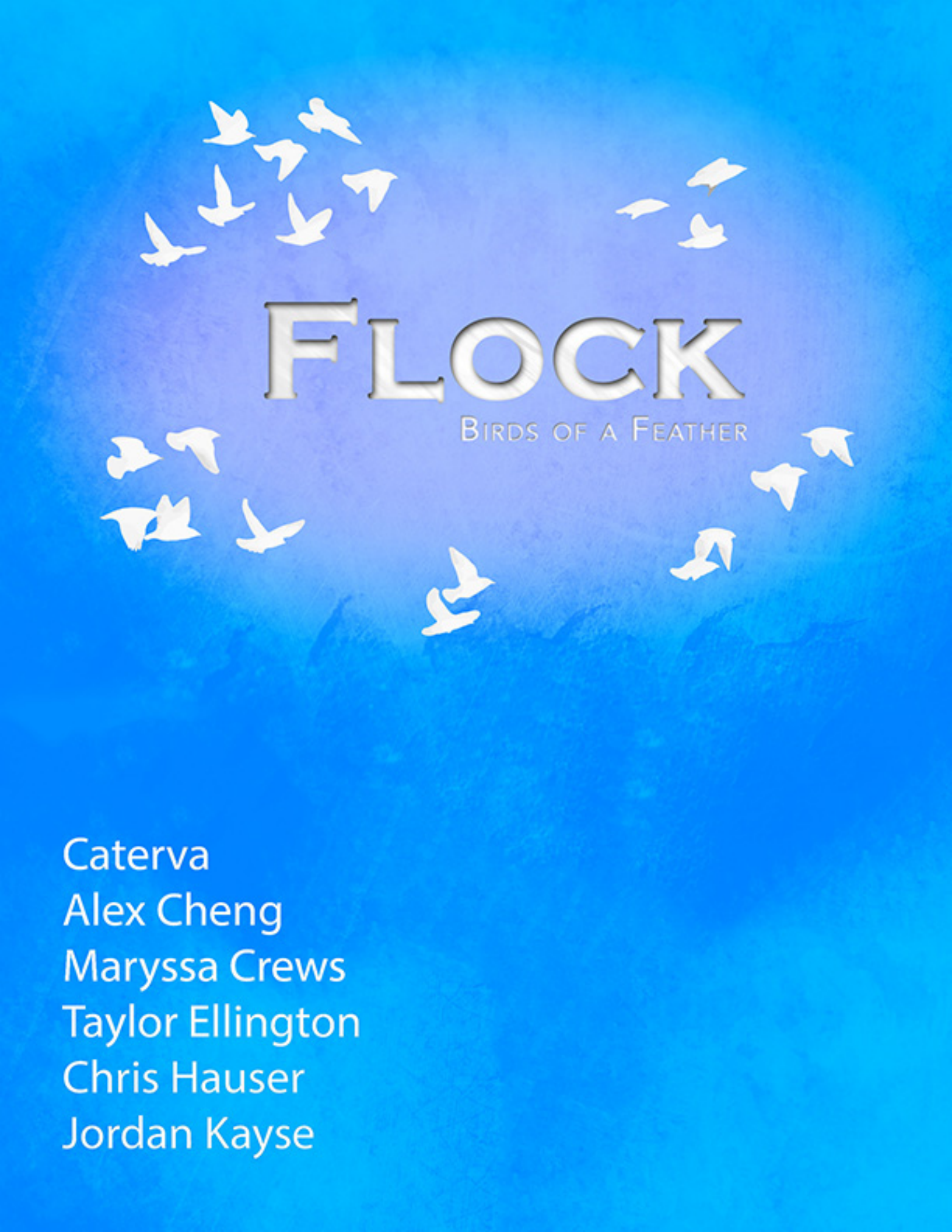

Caterva
Alex Cheng
Maryssa Crews
Taylor Ellington
Chris Hauser
Jordan Kayse

Introduction

Flock is an online event organizer available online as well as an Android application. With Flock, you can add friends, create groups, and create events to invite your friends to. Flock was made possible by the Caterva development team, who all brought varied skillsets to the team and contributed in different ways to the creation of the Flock product.

As a member of the backend, Alex Cheng was responsible for functions that allowed aspects of the product to be realized, including most of the profile, friend, and groups functionality. Chris Hauser also worked on the backend and helped build the backbone of Flock. He created much of the middleware that served as a connection from the database to the front end and back again as well as the event functionality.

Working on the GUI side, Maryssa Crews greatly contributed to both the design of the web user interface and its implementation.Taylor Ellington worked as lead Android developer, designing and implementing most of the app's functionality. He contributed to the products general look and feel. Finally, Jordan Kayse worked on both the Android and web applications. She is responsible for most of the fine tuning on user interfaces as well as much of the web application's functionality.

Software Features

Visitors

- Should be able to register to the website.
- Should be able to login to the website if they have an account.

Users

- Should be able to view and edit their profile.
- Should be able to see all the events they are attending along with the details. including name, description, time, and place.
- Should be able to view the event in a calendar.
- Should be able to create their own events.
- Should be able to invite friends to the events that they created.
- Should be able to invite friends to an event if the host says the event can be shared.
- Should be able to edit the name and description of the event and add more friends.
- Should be able to accept/decline event invitations.
- Should be able to add or remove friends.
- Should be able to create groups for friends.
- Should be able to add friends to groups or remove friends from groups.
- Should be able to remove groups.
- Should be able to log out.

Use Case Diagram

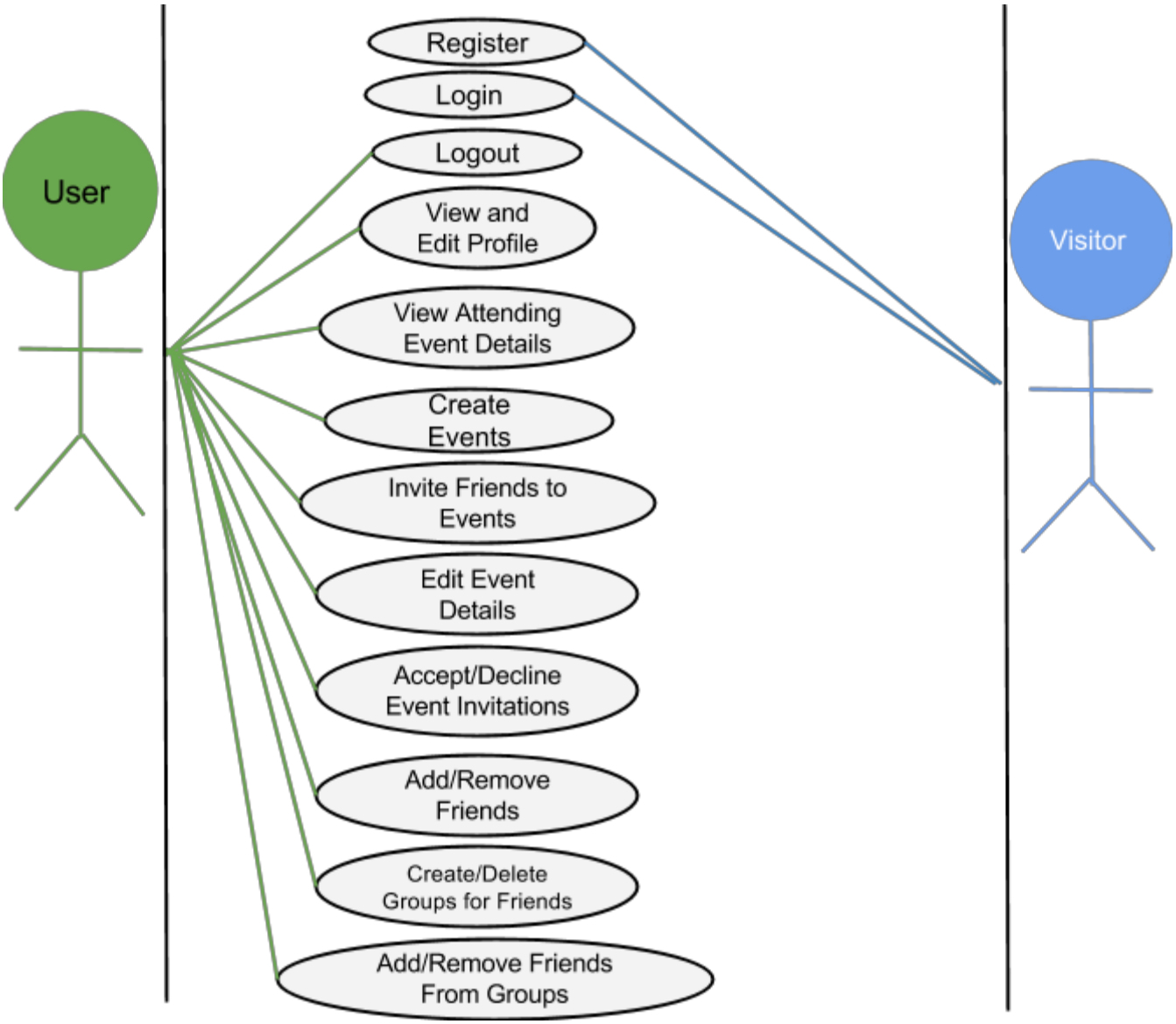

Figure 1: Use Case Diagram of our software.

The above *Figure 1* is a diagram of the different functionality a user and visitor should be able to accomplish. The functionality is shown in a visual way demonstrating the difference between a user and visitor.

Database Model

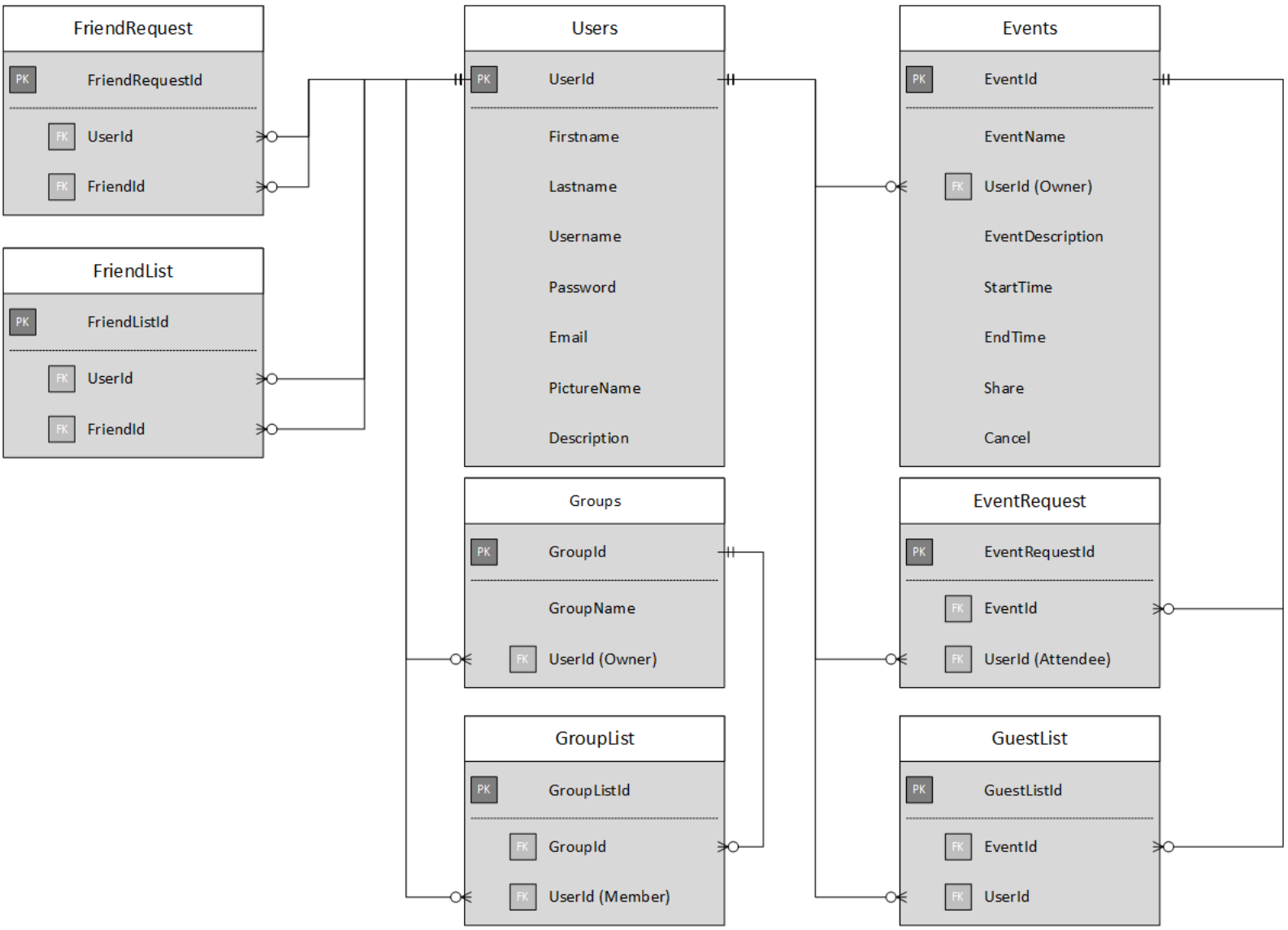

Figure 2: An ER Diagram of our database.

Figure 2 is a model of the database our team used. The diagram includes both the table names, the attributes, as well as the relationships between each table.

Users:

Table that stores all the User information including username, first and last name, email, etc.

Column Name	Type	Valid Range	Description
UserId	INT	1-...	The primary key for the Users table. Unique to each user. Auto-increments every time a user is created.
Username	varchar	1-100 characters	The name that users use to login to the website, and find users Each username is unique.
Firstname	varchar	1-100 characters	The first name of the user.
Lastname	varchar	1-100 characters	The last name of the user.
Email	varchar	1-100 characters	The email address of the user. User's receive notifications about their events through email. Each email is unique.
Password	varchar	1-100 characters	The password to the user's account. Encrypted on the database side.
Description	varchar	0-500	Text that user inputs, describing him or her. Can be null.
PictureName	varchar	0-100	File name to a profile picture that the user uploads. Can be null.

GuestList:

Table that contains a list of people that are attending events.

Column Name	Type	Valid Range	Description
GuestListId	INT	1-...	The primary key for the GuestList table. Uniquely identifies the Event and who is going to the Event. Auto-increments after a user invites another user to an event.
EventId	INT	1-...	The primary key from the Events table. Identifies the event.
UserId	INT	1-...	The primary key from the Users table. Identifies the users invited to the event with the EventId stated above.

Groups:

Table that contains the groups and the users who created the groups.

Column Name	Type	Valid Range	Description
GroupId	INT	1-...	The primary key for the Groups table. Unique. Auto-increments after each group is created.
GroupName	varchar	1-50 characters	The name of the group that the user created.
UserId	INT	1-...	The primary key from the Users table, that uniquely identifies the user who made the group.

GroupList:

Table that contains the users that are within each group.

Column Name	Type	Valid Range	Description
GroupListId	INT	1-...	The primary key to the GroupList Table. Unique. Auto-increments after a user is inserted into a group.
GroupId	INT	1-...	The primary key from the Groups table. Identifies the group tha the users are a part of.
.UserId	INT	1-...	The primary key from the Users table. Identifies the user within each group.

FriendsList:

Table that contains the list of friends that each user has.

Column Name	Type	Valid Range	Description
FriendListId	INT	1-...	The primary key to the FriendsList table. Unique. Auto-increments after users add friends.
UserId	INT	1-...	The primary key from the Users table. Uniquely identifies the user.
FriendId	INT	1-...	The primary key from the Users table. Uniquely identifies the friend of a user.

FriendRequest:

Table that contains the friend requests made by users to other users.

Column Name	Type	Valid Range	Description
FriendRequestId	INT	1-...	The primary key for the FriendRequest table. Unique. Auto-increments.
UserId	INT	1-...	The primary key from the Users table. Identifies the user sending the friend request.
FriendId	INT	1-...	The primary key from the Users table. Identifies the user receiving the friend request.

Events:

Table that contains the events that users have created and shared with their friends.

Column Name	Type	Valid Range	Description
EventId	INT	1-...	Primary key for the Events table. Unique. Auto-increments.
EventName	varchar	1-50 characters	The name of the event that the user has created.
UserId	INT	1-...	Primary key from the Users table. Identifies the host/creator of the event.
StartTime	datetime	yyyy-mm-dd hh:mm:ss	The starting time for the event.
EndTime	datetime	yyyy-mm-dd hh:mm:ss	The ending time for the event.
EventDescription	varchar	1-500	Description of the event. The user specifies what will be done at each event.
Share	INT	0 or 1	Gives user the option to allow user's friends to share event with their friends. 0 for cannot be shared, 1 for the opposite.
Cancel	INT	0 or 1	Cancels the user's event. 0 specifies the event is not cancelled. 1 specifies the event is cancelled. By default the Cancel column is set to 0.

EventRequest:

Table that contains the event invite requests given to users invited to an event.

Column Name	Type	Valid Range	Description
EventRequestId	INT	1-...	The primary key for the EventRequest table. Uniquely identifies the event and who is invited to the event. Auto-increments after an event request is sent.
EventId	INT	1-...	The primary key from the Events table. Identifies the event that the user is invited to.
UserId	INT	1-...	The primary key from the Users table. Identifies the user invited to the event.

Relationships

Users to Events:

The relationship is one user to many events. Users is not a weak entity, but Events is a weak entity in that events have a creator that is identified by the UserId from the Users table.

Users to EventRequests:

The relationship is many users to many event requests. EventRequests is a weak entity. It relies on the UserId from the Users table to identify who is being sent the event request.

Users to FriendRequests:

The relationship is one user to many friend requests. FriendRequests is a weak entities. It relies on the UserId from the Users table for both of its attributes UserId and FriendId.

Users to Groups:

The relationship is one user to many groups. Groups is a weak entities. It relies on the UserId from the Users table in order to identify which user owns the group.

Users to GroupList:

The relationship is one user to many group list entries. GroupsList is a weak entity. It relies on the UserId from the Users table to correctly identify which user is in the group.

Users to GuestList:

The relationship is one user to many guest list entries. GuestList is a weak entity. It relies on the UserId from the Users table in order to identify which user is going to the event.

Users to FriendList:

The relationship is one user to many friend list entries. FriendsList is a weak entity in this relationship. It relies on the UserId from the Users table to complete its two attributes UserId (user thats logged in to website) and FriendId (user's friends).

Events to EventRequests:

The relationship is one event to many event requests. EventRequests is a weak entity in that it relies on the EventId from the Events table in order to tell which event the user is invited to. Events in this relationship is not a weak entity.

Events to GuestList:

The relationship is one event to many guest list entries. GuestList is a weak entity in that it relies on the EventId from the Events table to identify which event the user is going to.

EventRequests to GuestList:

The relationship is one event request to one guest list entry. GuestList is a weak entity in this relationship. A guest list entry will not be created unless an event request has been accepted.

FriendRequest to FriendList:

The relationship is one friend request to two friend list entries. For every friend request that is accepted, a friend relationship is created between the sender and the receiver and vice versa. FriendList is a weak entity. A FriendList entry will not be created unless a FriendRequest is accepted.

Groups to GroupList:

The relationship is one group to many group list entries. GroupList is a weak entity in this relationship. It relies on the GroupId from the Groups table to identify which group the user belongs to.

Detailed Software Architecture Diagram

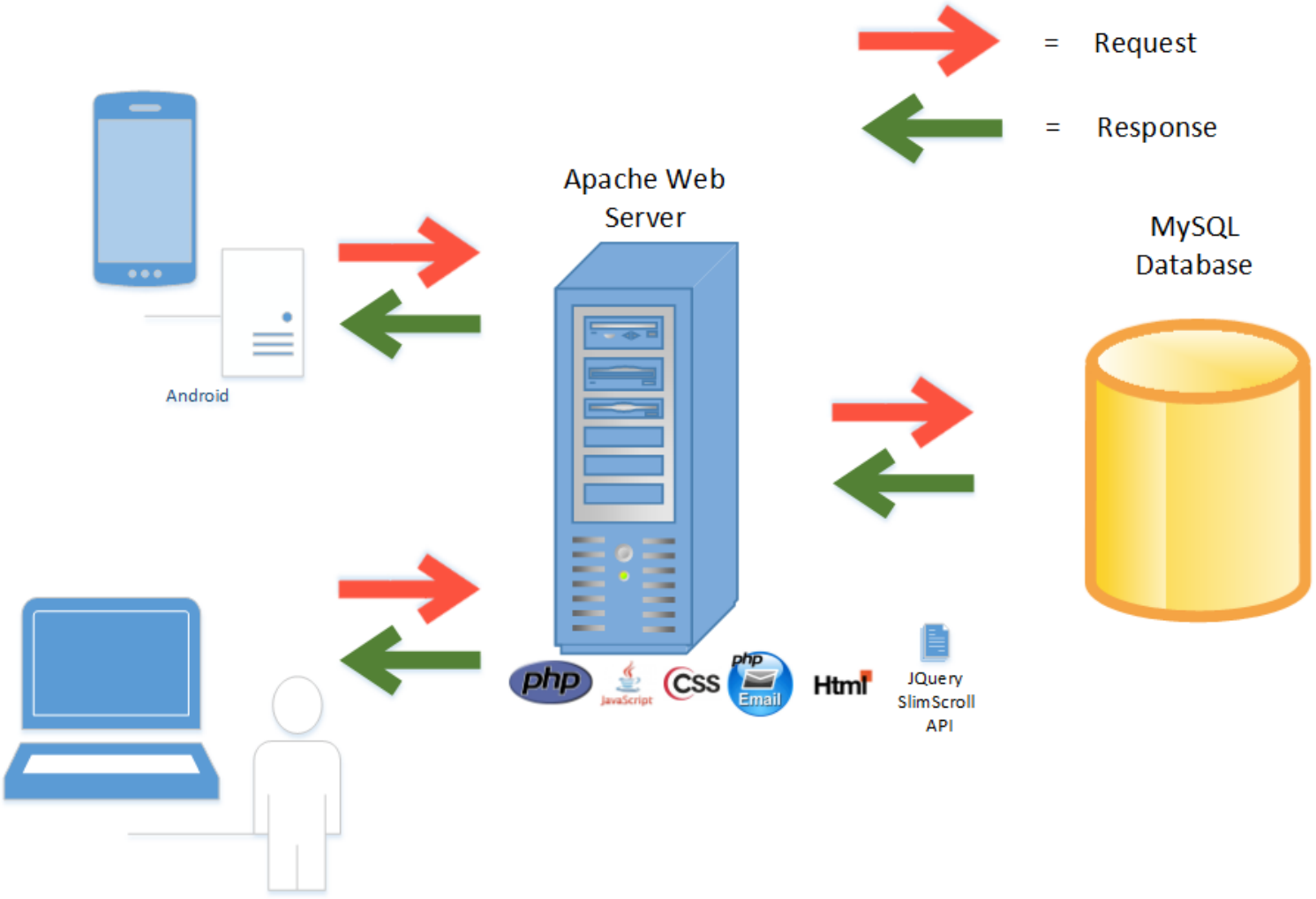

Figure 3: A Detailed Software Architecture Diagram.

The above *Figure 3* represents the structure used for our product Flock. The web client represents our website while the mobile client represents the created Android application. Flock uses both an Apache web server along with a MySQL database. The Apache web server is used to retrieve our web pages while the database stores all the information gathered throughout the website, including: creating an event, adding friends, and creating events. In addition, the Apache web server also uses email functionality to send notifications of events to users.

To create the web pages, HTML was used for layout, CSS for the styling, and Javascript and JQuery for the functionality. Additionally, our website made use of the Jquery Slim Scroll API. This was used to have scroll bars similar to ones used on Facebook. On the Android side, the functionality is created with java and the styling and layout with XML.

As discussed previously, our MySQL database holds all the information used. To communicate to the database and back, our team used PHP. This way we can get information out of the database and use it in both the web and mobile client.

Finally, our server has a cron job that runs every night. This script is used to clear the database of old events and to send the notification emails for events.

Web UI

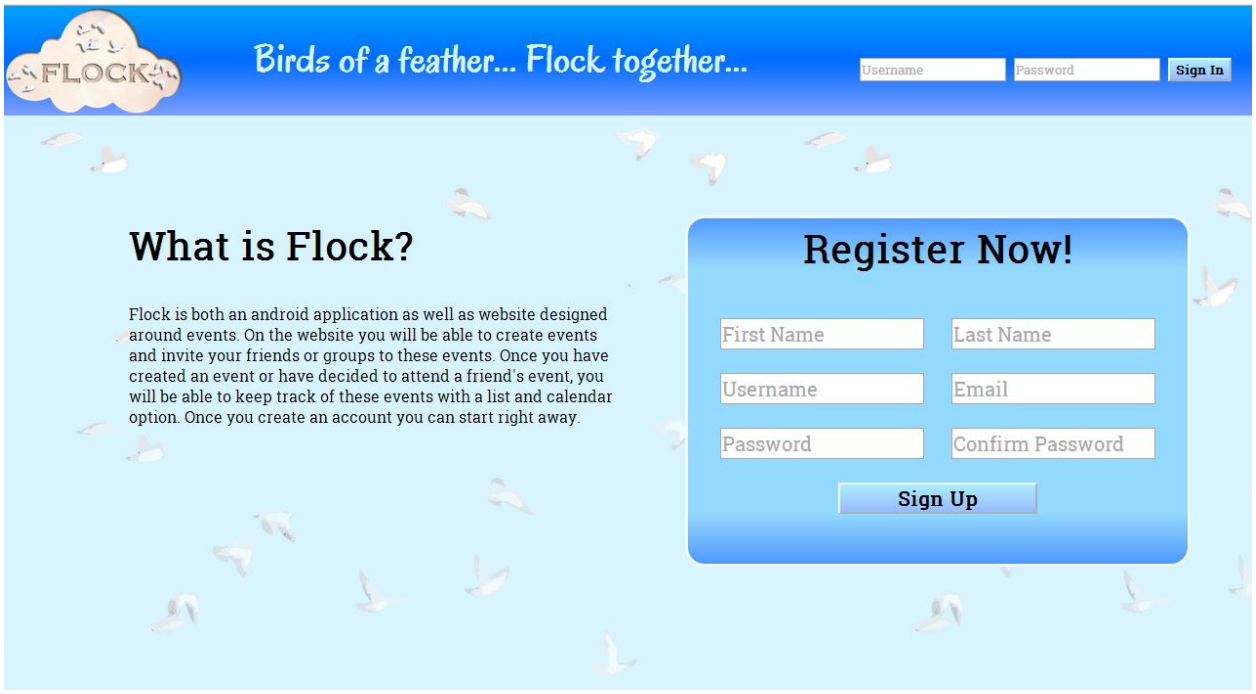

Figure 4.a: First Screen

When a user first visits a website this is the screen they see. A visitor cannot do anything but create an account or sign in.

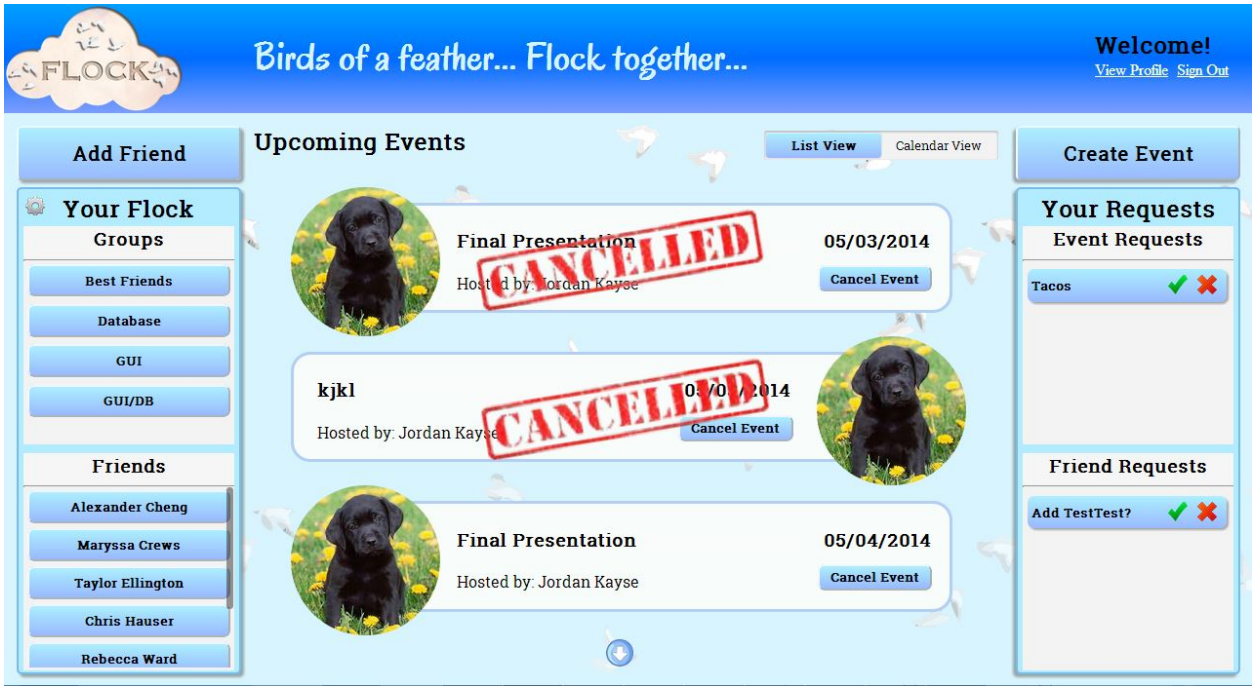

Figure 4.b: Main Screen

After a user signs in this is what they see. It shows events, groups, friends, and requests. This also displays the format of canceled events that the owner of the event has canceled.

Figure 4.c: Main Screen - Calendar
When a users clicks to view the calendar view. A calendar pops up showing the current month.

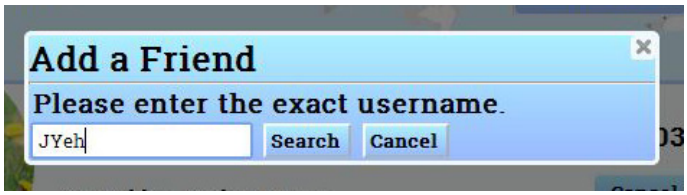

Figure 4.d: Add Friend - Search
When a users clicks Add Friend, a module shows asking for a username. if successful, another section is displayed, *Figure 4.e* with the user you searched for. You can then send a friend request for that user

to accept or deny.
Figure 4.e: Add Friend - Send Request

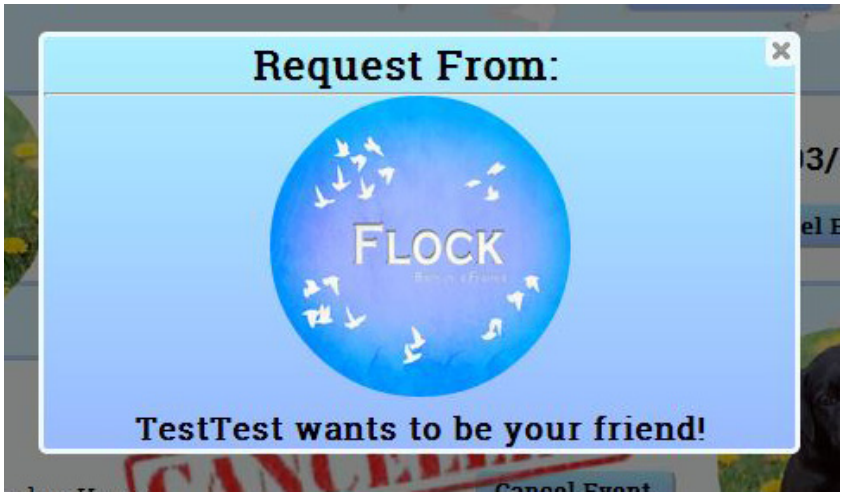

Figure 4.f: View friend request
When a user clicked on a friend request they have, a module pops up displaying the username and profile picture of the user sending the request.

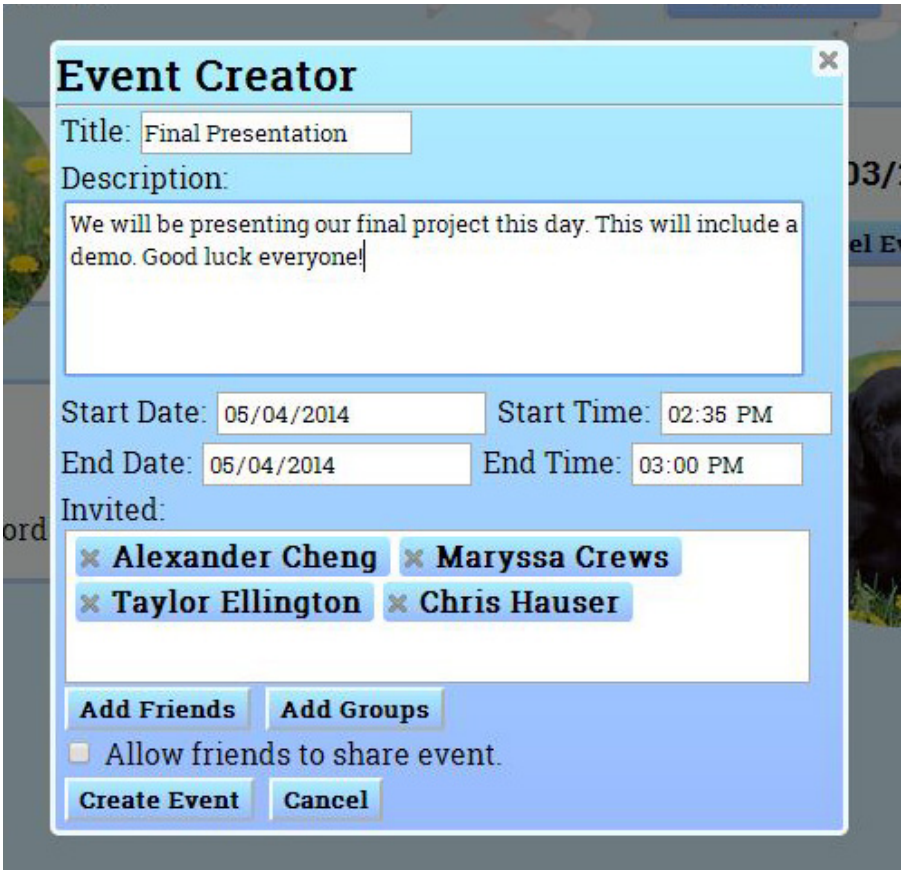

Figure 4.g: Create an event

When a users clicks create event, a module appears with a form of the information needed to create an account. A check is used to make sure the event isn't before the current day and that the end date does not start before the start date. By allowing friends to share an event, they can invite their friends to the event as well, spreading the event to multiple users.

Figure 4.h: Add Friends

When a creating, editing, or sharing an event you can add friends. This module will appear allowing you to select as many friends as you want in one sweep. The same is considered for adding groups as shown in *Figure 4.i*

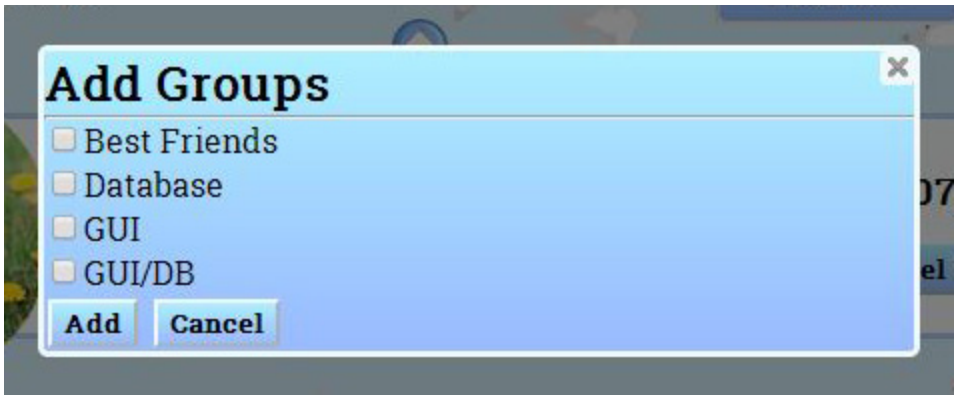

Figure 4.i: Add Groups

Figure 4.j: Calendar Event Information

In the calendar when you click on a day with events, the list of events will pop up like above.

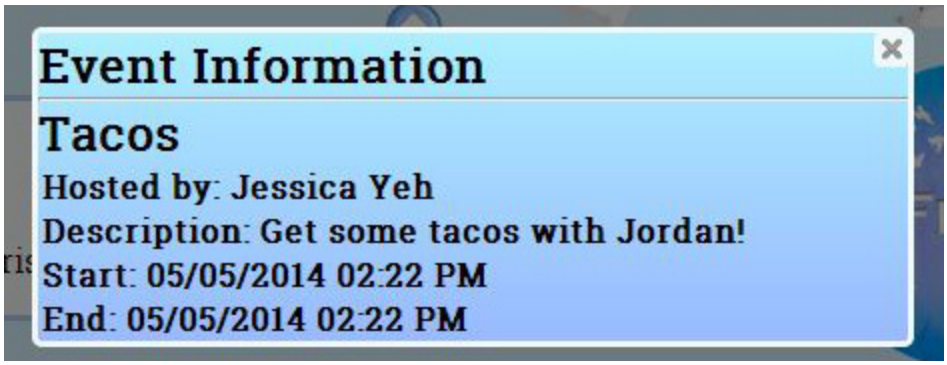

Figure 4.k: List View Event Information

In the list view of events, when clicking on an event information displays. If the event is not yours and you cannot share it, there is no icon displayed.

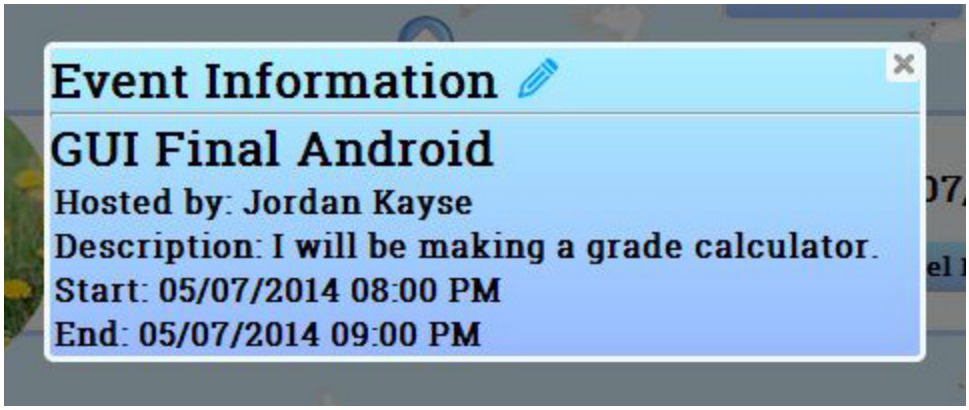

Figure 4.l: List View Event Information - Edit

This module displays information on an event. The event was created by the current user and that user can edit the event by pressing the pencil icon.

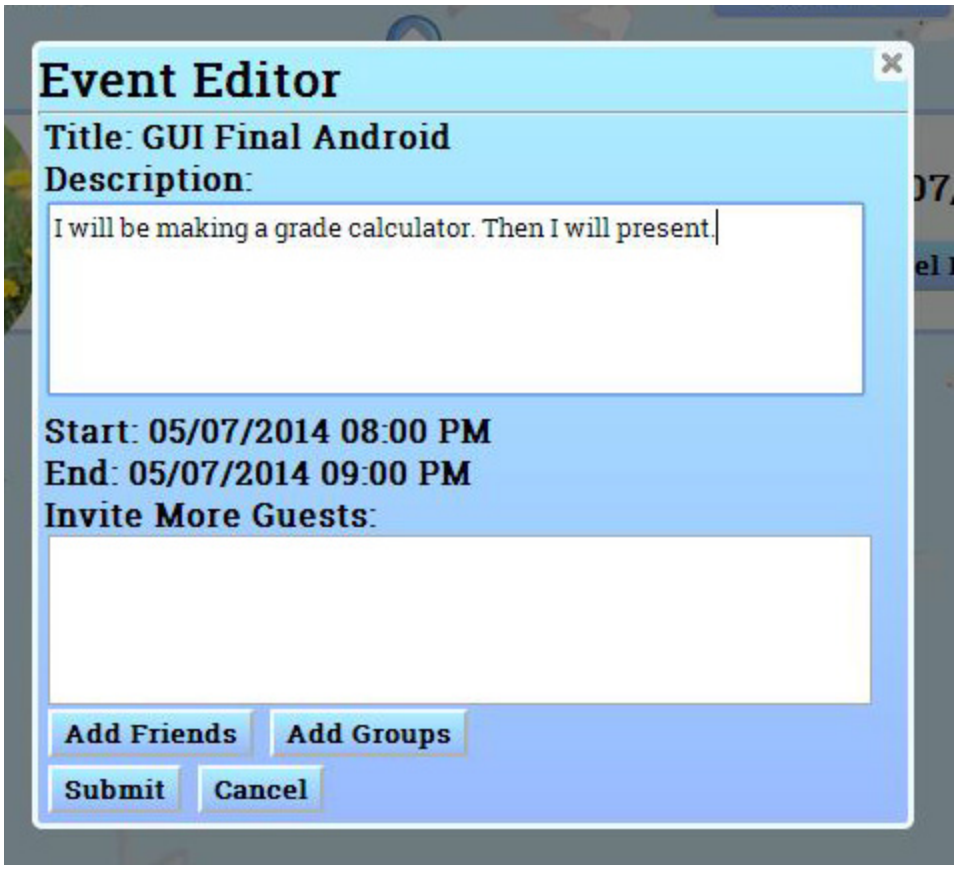

Figure 4.m: Edit an Event

When editing an event, you can only edit a description or add more friends to the event.

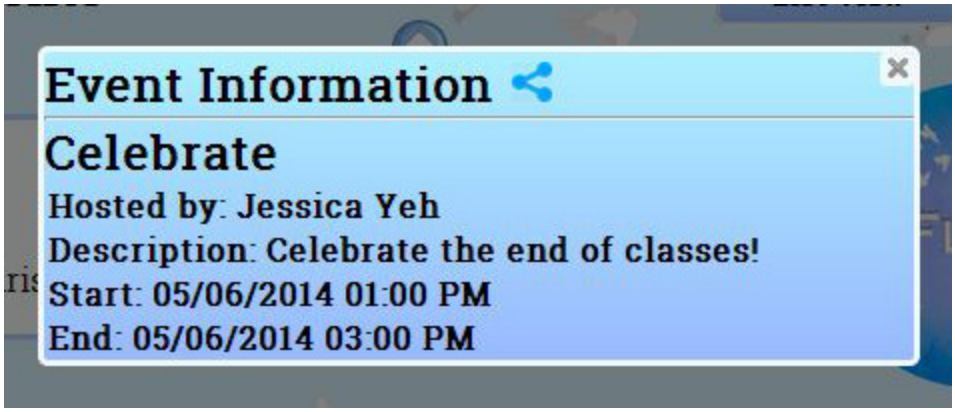

Figure 4.n: List View Event Information - Share

This module displays information on an event. The event was created by another user and that user allowed the event to be shared. The share icon accomplishes this.

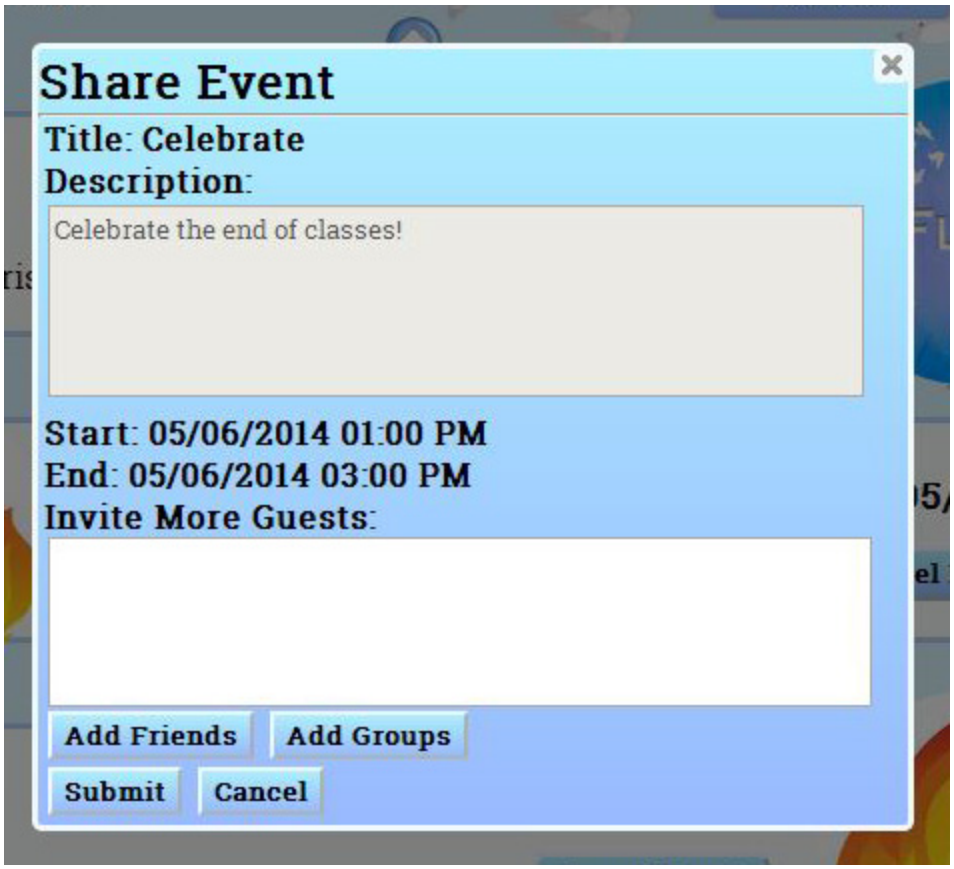

Figure 4.o: Share an Event

When sharing an event, you can only add more friends to the event. This is used when you want the event to be available to anyone and you want many people to show up.

Figure 4.p: Edit groups and friends.
This page can be reached after clicking the gear on the Flock section of the main page. Here you can delete friends or groups, create groups, and edit groups. Before deleting a friend or a group a pop up will appear confirming the action.

Figure 4.q: Create a group
After clicking the add sign on the groups section, you can create a group. At least one user must be added to the group for this to work.

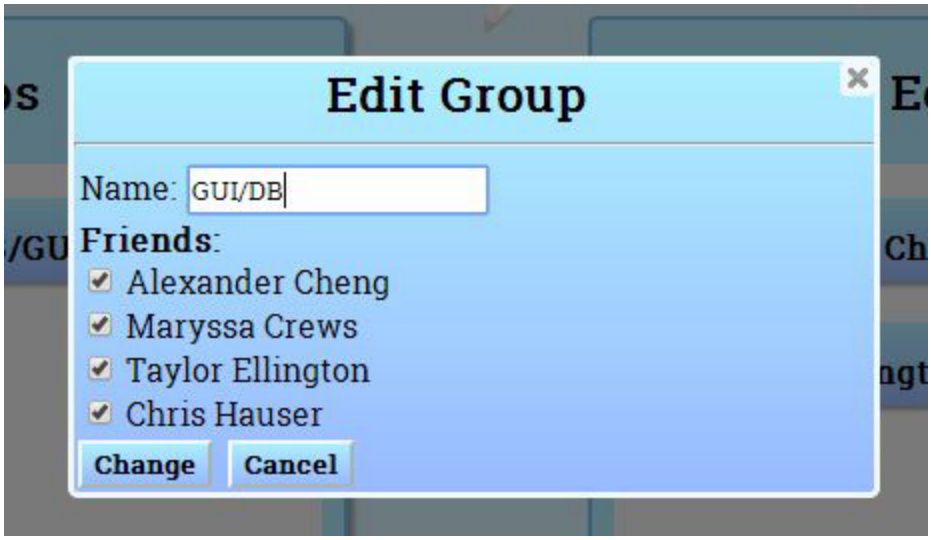

Figure 4.r: Edit a group
After clicking on a group, a pop up will appear with the information of the group. In this pop up you can delete friends from a group, add them, and change the title.

Figure 4.s: View your own profile
After clicking on view profile, you can view your own profile. With this you can edit the description and picture of your profile.

Figure 4.t: Edit your own profile

After clicking on edit profile, you can submit a new picture or change the description. Then you can submit the changes.

Figure 4.u: View a friend's profile

After clicking on friend in the main page, you are redirected to their profile. Here you can view their information. A check is used to make sure you are really friends.

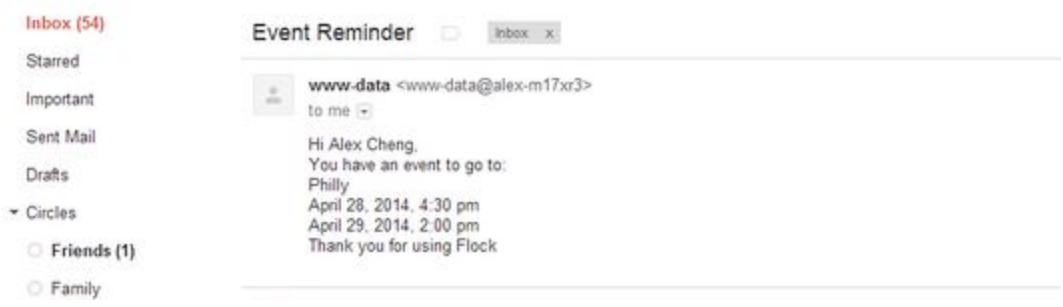

Figure 4.v: Picture of event notification

On the day before an event occurs, a cron job is run to find any events. The information is then formatted into an email and sent to anyone attending the event.

Android UI

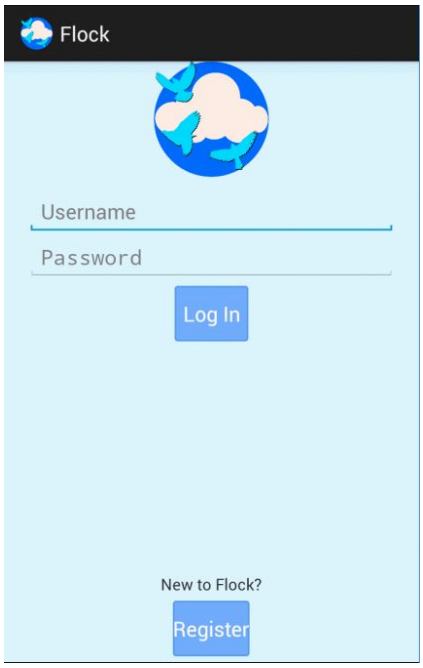

Figure 5.a: Application start page

When the application starts you can either sign in or create an account.

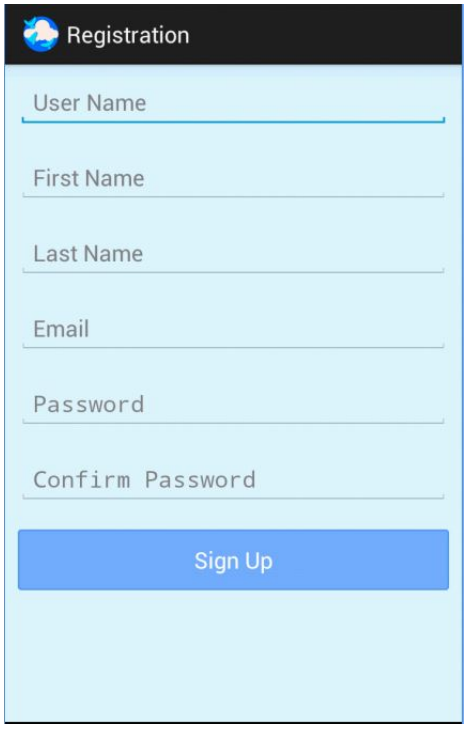

Figure 5.b: Registering an account
Here you can create an account with the proper fields.

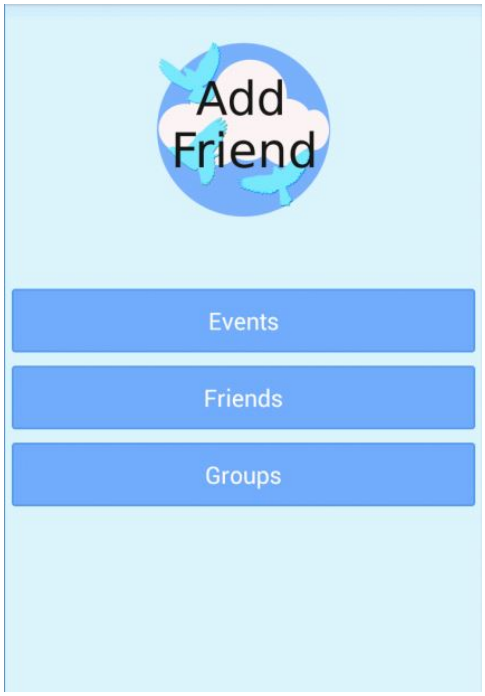

Figure 5.c: Main application page
After signing in this page will be displayed with the different options available.

Figure 5.d: View Events

A list view of all the events the user is attending.

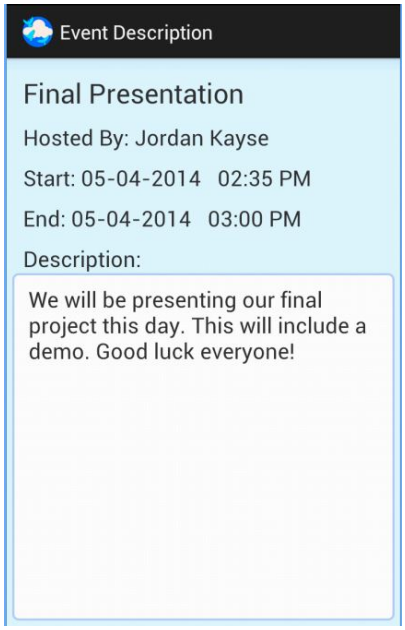

Figure 5.e: Event Item

A figure displaying the one event clicked.

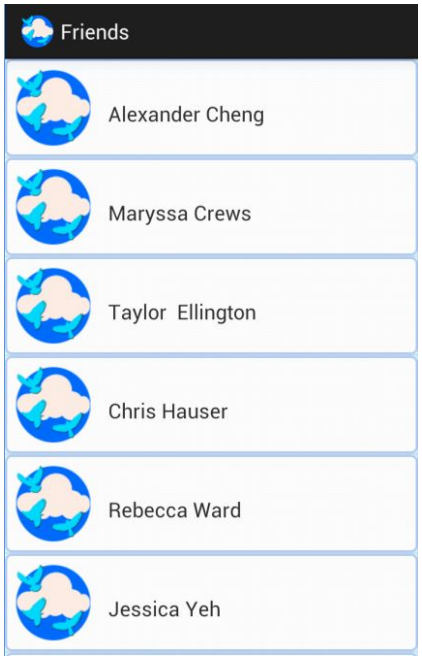

Figure 5.e: Friends View

A list view of all the friends of a user.

Figure 5.f: Groups View

A list view of all the groups a user created. Clicking on a group displays more information.

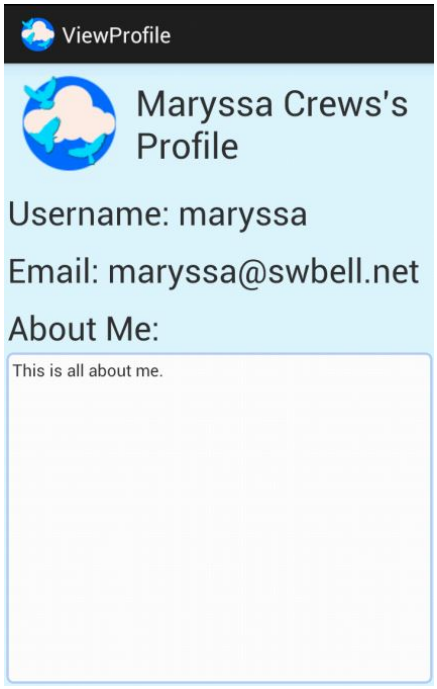

Figure 5.g: Profile Page
An activity to display the profile information for a user.

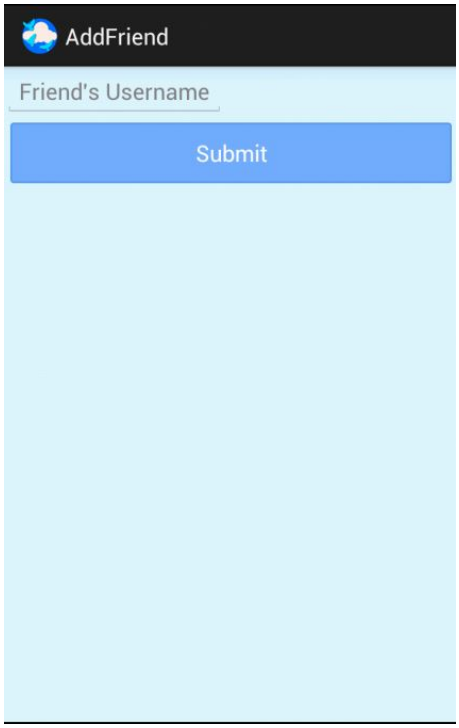

Figure 5.h: Add Friend
An activity to add a friend. If a user has a NFC enabled phone they can add a friend by bumping phones.

Testing Flock

With each element of functionality we added, we tested the backend functionality with Postman, the web client with Chrome Developer Tools, and the Android client with the Android Developer Tools. We would test the database functions separately to make sure they achieved the desired effect in the database and then would test the user interface to make sure it looked and acted as we desired. When it came to the Chrome Developer Tool, it was useful to be able to update the functionality and layout in elements and sources section. For the Android Client, LogCat was an extremely useful tool for debugging.

The use of a testing team turned out to be very helpful. Team One, known as Lab Labs, helped us to pinpoint annoyances and bugs that we overlooked since we were so involved in the development of the product. Caterva and Lab Labs were both very good at detailing an error by describing what exactly happened and what the user was doing when the error occurred. We were also good at communicating with each other by indicating what actions were being taken to fix the bugs or asking and answering questions to clarify the issue.

Based on comments from GitHub and the differences between iteration 1 and iteration 2, we believe our feedback was taken into consideration and directed the group to improving the usability of Help Me Out. As for Caterva, we tried to address every issue raised by Lab Labs to make sure the most glaring of bugs were taken care of first and then tackling the minor annoyances they found later. Along with the testing team, we were able to show our product to the TA. The TA was helpful with giving suggestions for our product. Since the TA had more experience with UI development, he was able to give useful comments that we took into consideration.

Caterva also had the help of a few testers outside of the database and graphical user interface classes. These included a 32-year-old administrative assistant with a good amount of experience using computer applications but no experience with coding or design. She commented that the look of the site was appealing and that it wasn't too difficult to understand or navigate. She was used more to make sure that the application was aesthetically pleasing rather than to search for bugs. This was made possible because we received design advice from an outside party.

Team Reflection

The biggest issue we expected to face as a team, was that of integrating so many different technologies into a singular product. We feared that leveraging so many tools and components would cause massive problems, so to compensate the team wrote very precise and complete documentation and ensured we built to our specifications. Additionally when a change was needed, the team excelled at communicating and ensuring everyone was together and up to date. This meant our biggest expected issue never truly caused any harm.

Another issue our team was faced with came in the form of an unexpected reduction in manpower. Initially our team had one more member, an individual assigned to work on the backend and middleware. For personal reasons this individual was unable to work on and help complete Flock, and as a result the remaining backend developers had an additional share of work to complete. Our devs rose to the circumstances, turning out a quality API in spite of the number of workers.

On a more technical note, the user interface team had some difficulty in regards to the calendar on the website. Initially the team planned on using a google api to create this functionality. This idea fell through as it was discovered that the products did not support the intended idea. This was a setback that resulted in a member of our user interface team building their own calendar solution that is now featured in the product.

In the beginning, it was difficult to integrate both the frontend and the backend. During the taco truck project, both sides did not know what they needed to send or what they would be receiving. Our team occasionally met as a group to discuss what we would need or be sending. This helped a lot with our project and creating the functionality necessary for our website and android.

Another technical issue faced by the team arose from the Android application's proposed NFC functionality. Compared to other Android features, the NFC and Android Beam tools are very under documented. This made implementing the proposed feature very difficult in the allotted time.

On the backend, the biggest technical issue was with the email functionality. It was easy to create a script to run every day with a cron job, but our team had trouble with setting up the email functionality. We were able to find a helpful resource that allowed us to fix this problem. In the beginning when working on the login functionality, one member had a lower php version. This caused the sign in and sign up functions to not work. This made testing the product difficult in the beginning until the member could update the php version to the current one.

Our ability to find solutions to any technical problems that occurred is what allowed our team to be successful during this project. It is inevitable that problem will happen and being able to think of

potential solutions is critical. Also, this is a team project and being able to work together to come up with the best idea is useful. To do this our team would meet up to work on the product. This was key for making sure our product was working the way we wanted it to. The only thing we would change would be to meet up even more if possible and make sure the whole team is on the right direction from the beginning.

Wrap Up

In the future, Caterva would like to expand Flock beyond its current features. The feature we would most like to implement would be the Android application in general. Today, everyone has a phone with them, if we can add more features from the web onto the android, more users would want to use our product because of the easy access. These features would include a calendar view of the events, and the ability to create events and groups. In addition, adding notifications for events would help notify users of events as well.

Security has become more essential for websites since users do not want their information stolen. Caterva would like to implement more security features onto the website, including encrypting the the data sent to the server. Also, having a reset password feature that would send the password to their email account is useful if users forget their password.

Caterva is a useful product for making events to either share with people you know, or keep as a to do list. Events can either be for fun or for assignments. We believe that our product is easy to use and is available for different uses, with this in mind, Caterva would in the future like to share our product with more users. First we would start with the Dallas area then continue to grow from there.