Team-Up!

(Design Document)

https://github.com/srhee91/Team-Up

Travis Coria, Trevor Coria, Boheon Jeong,
Yiyang Pan, Sang Rhee, Kartik Sawant

Design Document

Purpose:

We are designing a team building application for iOS/android devices which will allow users to find people who share the same interest either by age, location, or skill level. Users can join existing groups or make groups with the option to make it public or private. Each user will also have their own profile in which they can choose which information is shared to other users. Other features include chat and file sharing within groups.

Design Outline:

a. Design Decisions

We will be following the client-server model. The major components will be the server, clients, and GUI.

Server:

The server will hold all users' information as well as every category, sub-category, and group. The server listens to clients' requests, retrieves and process the database to respond to the clients.

Clients:

The client will run on each user's device and will request and receive information about teams, groups and chats from the server. Client component will guide GUI component to the pages that user wants.

GUI:

Android - The GUI will display groups and members in the form of buttons. Once a button is pressed, corresponding actions will be displayed on the device. When a chat message is received, it is the GUI's responsibility to have animation and sounds.

iOS - The GUI will display groups and members in the form of table view cells. Once a cell is pressed, corresponding actions will be displayed on the device. When a chat message is received, it is the GUI's responsibility to have animation and sounds.

b. Component Interactions:

On GUI, first time user will sign up for an account. Client will send the account information to the server. The server will validate the combination of username and password. Depending on the situation, the server will process accordingly and respond to client what to do.

Once logged on, user will see the five modules: MyGroup, Groups, Chat, Profile and Setting on GUI. Depending on how the user interacts with our GUI, the client will request whatever information is needed to the server and display the contents on GUI what the user would like to see.

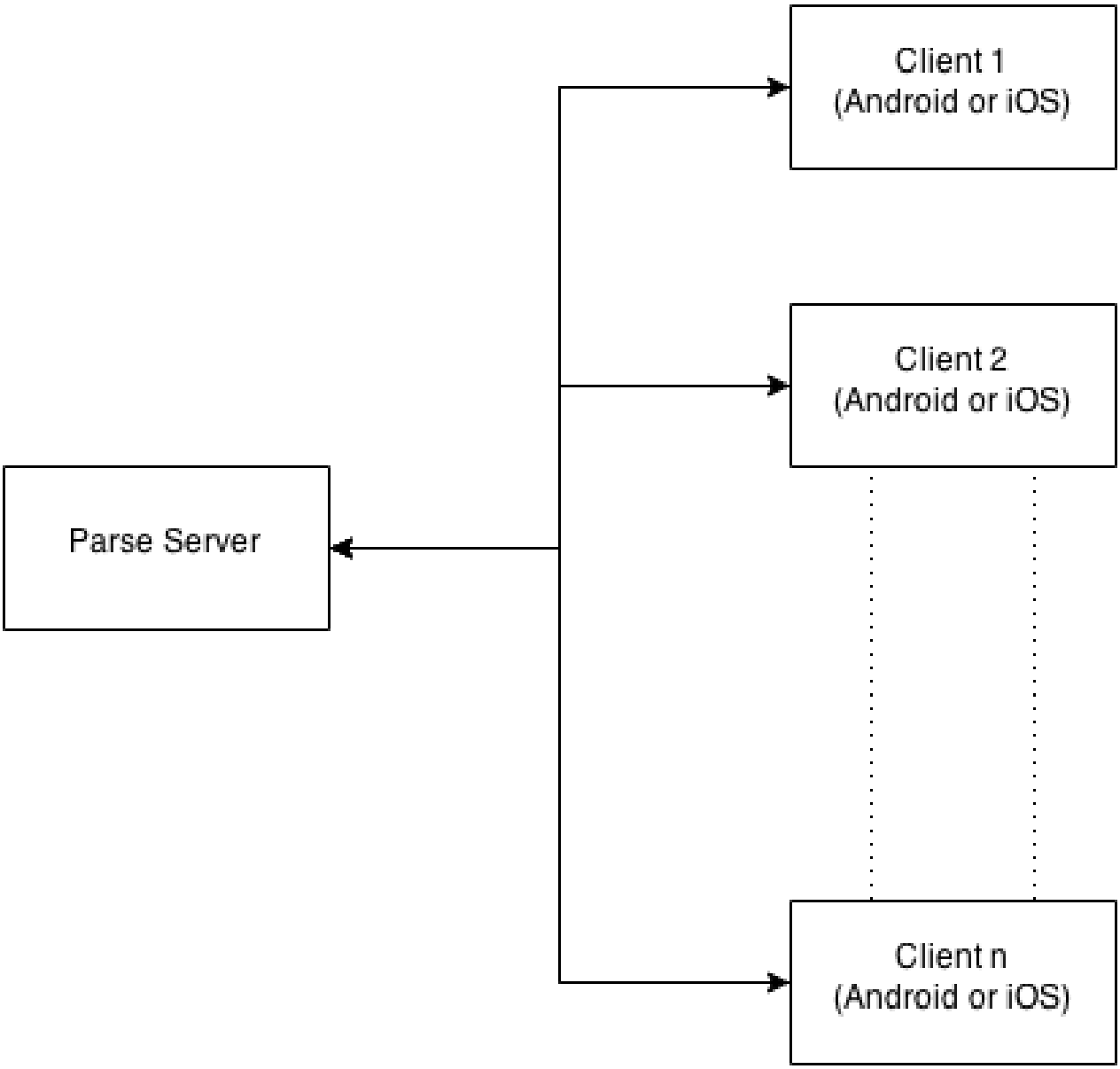


Figure1. Client-Server Diagram

Design Issues:

Issue 1: How do we construct the server back-end?

Option 1.1: Google App Engine.
Option 1.2: Purdue user web server.
Option 1.3: Parse.

Decision: We chose Option 1.3 because it's free, has more than enough resources, and is easy to use.

Issue 2: Which platform do we develop it on?

Option 2.1: Android
Option 2.2: iOS
Option 2.3: Both

Decision: We decided to go with Option 2.2, since iOS will be much easier to develop on and we will be able to make changes more rapidly down the road.

Issue 3: What if the admin of the group leaves the group and there are other people in the group?

Option 3.1: Admin privilege is passed to whomever previous admin selects
Option 3.2: Admin privilege is passed to the member who has been part of the group the longest.
Option 3.3: Disband the group

Decision: We chose Option 3.2. The problems with selecting an 'heir' is that it could make the interface seem too clunky. Disbanding a group also seems too extreme, and could be problematic in large groups with a single admin.

Issue 4: What if the admin of the group leaves the group and there is no one else in it?

Option 4.1: Delete the group
Option 4.2: Leave the group as it is, and switch the admin to the second earliest user joined in the group.

Decision: We chose Option 4.1. Simply because we believe as a group admin, he/she should be responsible for switching the title to someone else before leaving. We will definitely give a warning before he/she decides to leave and disband the group.

Issue 5: Should the client be in charge of storing chats or just transferring them to the correct user?

Option 5.1: Server stores the chat logs.
Option 5.2: Client stores the chat logs.

Decision: We chose Option 5.1. Storing chat logs on the client side will have many problems. When new users join a group or switches devices, they will not be able to obtain the entire chat history. This can be disorienting, and storing chat logs on the server will fix this problem.

Issue 6: Are users' chats synced across different devices?

Option 6.1: Yes, chats are synced across devices.
Option 6.2: No, chats are not synced across devices.

Decision: We chose to sync the chat across all devices because our backend(Parse) provides this feature. It also coincides with our decision in Issue 5.

Issue 7: Can users add new group categories?

Option 7.1: Yes, users can add new categories.
Option 7.2: No, users must use the preset categories.

Decision: We chose option 7.2. It would be too cumbersome to let users make their own categories.

Issue 8: If a user deletes his or her account, does it also delete any messages sent by him or her?

Option 8.1: Yes, the data will be erased after the deletion of the user account on the server.
Option 8.2: No, the messages are preserved, including their old username.
Option 8.3: No, the messages are preserved, and the user's username is replaced with alternative text.

Decision: Option 8.3. This is because once a user deletes their account, we will make their old username available to others.

Issue 9: Who should users be able to chat with?

Option 9.1: Directly with other users.
Option 9.2: With members of their group.

Decision: We chose Option 9.2, since storing individual chat logs may prove too cumbersome.

Issue 10: Can two groups have the same name?

Option 10.1: Yes, multiple groups can all have the same name.
Option 10.2: No, group names are unique.

Decision: We chose option 10.1, since it would be too inconvenient for the users to have to try to come up with a unique group name. Groups will be stored in the database based on their unique ID.

Design Details:

Class Descriptions and Interactions:

- The ***Login*** class is the class that sending the username and password to the server end, and valid the combination. If the combination is valid, it will bring the user to the *MyGroup* page.

- The ***SignUp*** class is the structure class that requests userID, password, userName, location, birthday and email address from user. By calling *sendUser(userID, userName, password, location, birthday, email)*, it creates new user account and store inputted information in database.
- The ***MyGroup*** class is the structure class that static methods *pullGroup()*, and *selectGroup()*, which are responsible for retrieving group information such as groupPicture, groupName and groupID and displaying groups pulled. *deleteGroup(groupID)* is responsible for deleting selected group.
- The ***Group*** class is the most important object in our program, it contains a set of Users with the same interests. It has a unique ID generated when it's being created, but its name might not be unique. *getAdmin()* method will retrieve the one and only one administrator of the group. But a single user can be the Admin for multiple groups.
- The ***Chat*** class displays the list of groups that the user is a part of. It uses the function *getGroups(userID)* to obtain the groups belonging to a user, as well as information that will be displayed about each group. When a user clicks a group, the group's respective chat will appear.
- The ***ChatPage*** class belongs to a particular group and displays the entire chat history of the group. The chat log will be obtained from the database using *getChat(groupID)*.
- The ***Settings*** class is a class that contains multiple methods to obtain and modify user's profile information. Preferences can be added and deleted, *changePassword()* is the methods to change the password stored in the server. It also has a *logout()* method to exit or switch the account.
- The ***MyProfile*** class is simple wrapper that contains all the user information and profile picture. User cannot change username, since it is being used for the login validation. But *modifyImage*() and *modifyLocation*() allow user to change those relevant fields.
- The ***ChangePassword*** class contains strings to store and compare the old and new passwords. It calls the function *setPassword(userName, oldPassword, newPassword)*, which changes the password stored on the database.
- The ***DeleteAccount*** class is a class that give the privilege to the user to deactivated his/her account in the server. Even it is named as delete, the user account will still be stored in the server and stay unused. This way, user will be given the ability of regain his/her account.
- The ***AddPreferences*** class allows the users to add/edit their preferences on their interests in group and category. Users can use *setPreferences()* to set their preferences.
- The ***UserProfile*** class contains and displays information about a user that is not the currently signed-in user. The class uses the methods *getProfile(userID)* and *getGroups(userID)* to obtain the relevant information.

- The ***Search*** class allows the user to type in order to search the database for groups and categories. The class will call two functions per search, *searchCategory(String)*, and *searchGroup(String)*. Both functions return an array of their respective type and will display the results in a list.
- The ***NewGroup*** class is a class that initiate a new Group object with one member, who will be the user who created the group itself. When this new group is created, he/she will be automatically set to be the Admin of this group. No changes will be allowed until at least two members are in the group.

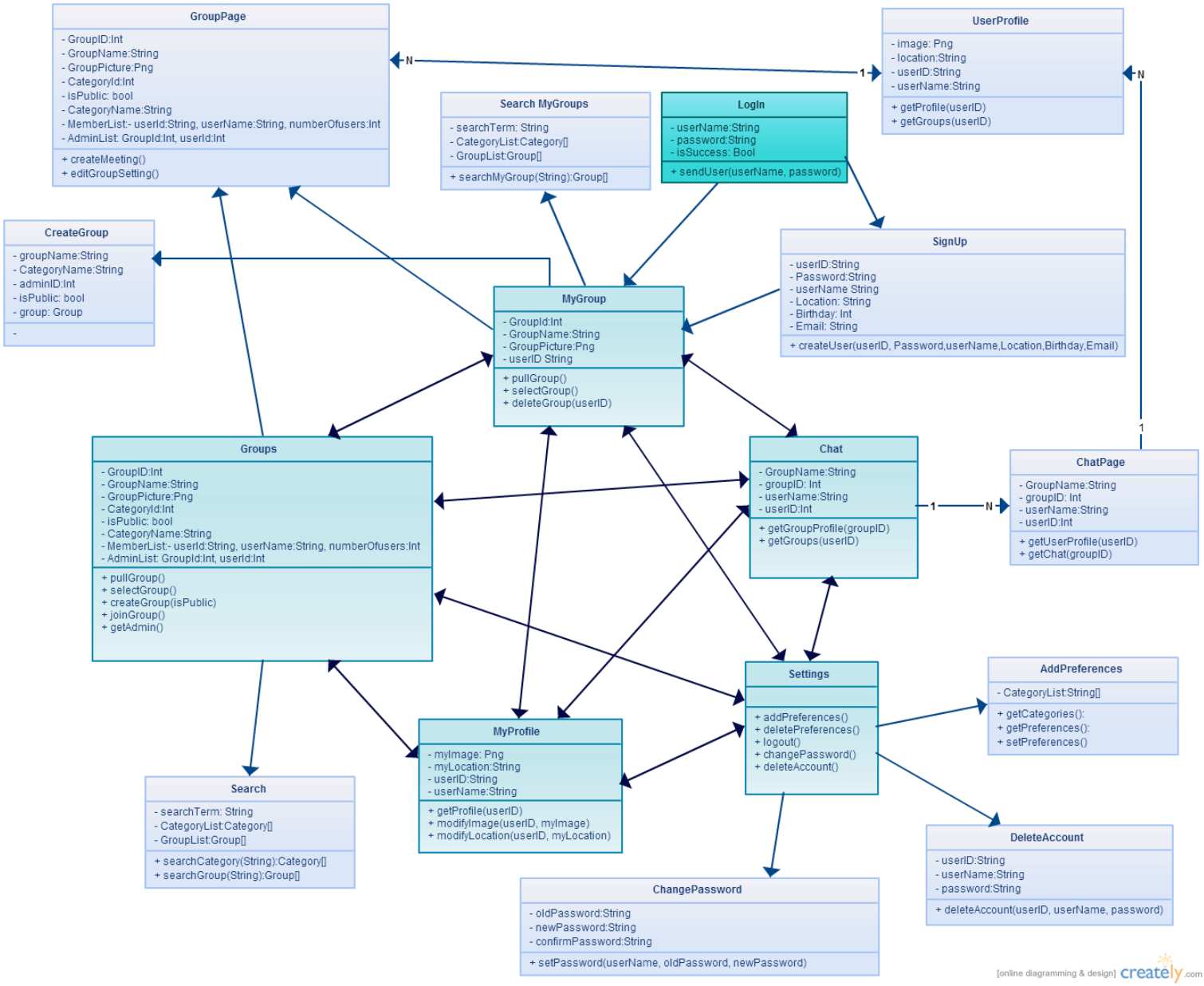


Figure 2. Frontend Class Diagram

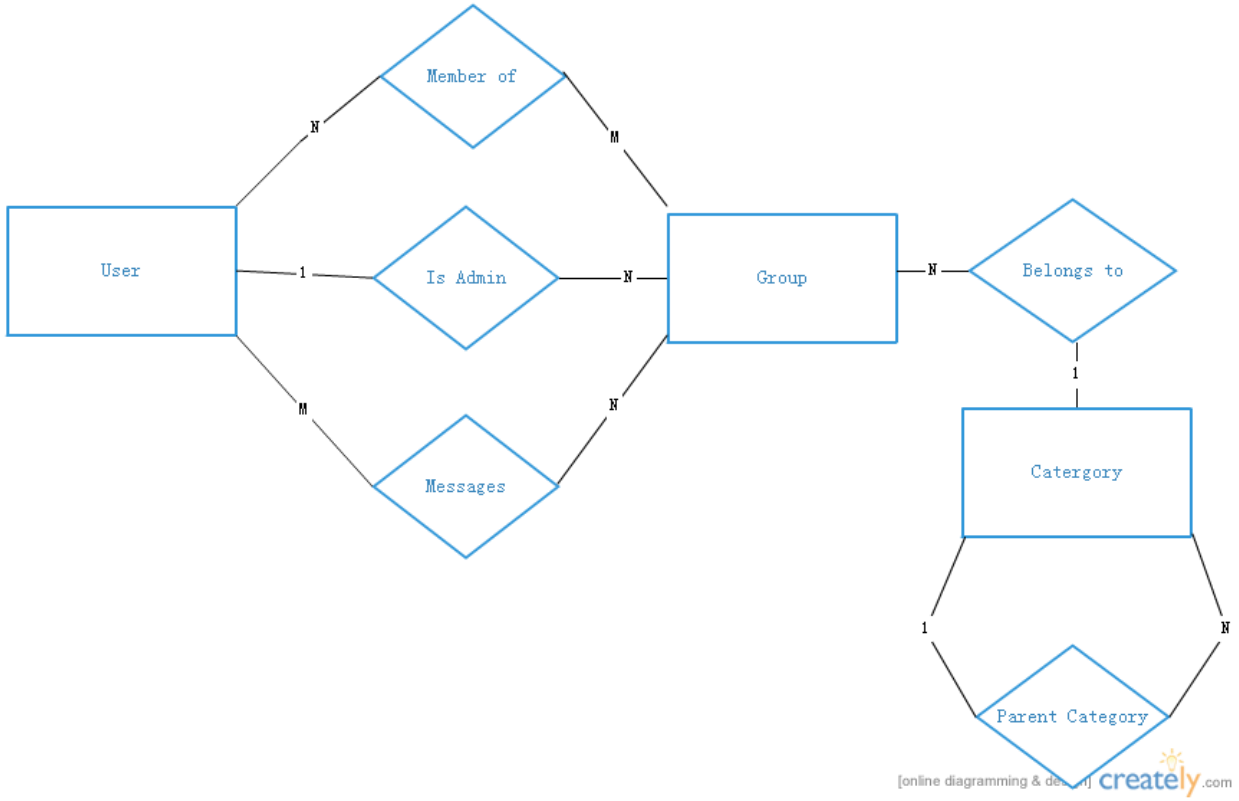


Figure 3. Backend ER Schema Diagram (w/o attributes)

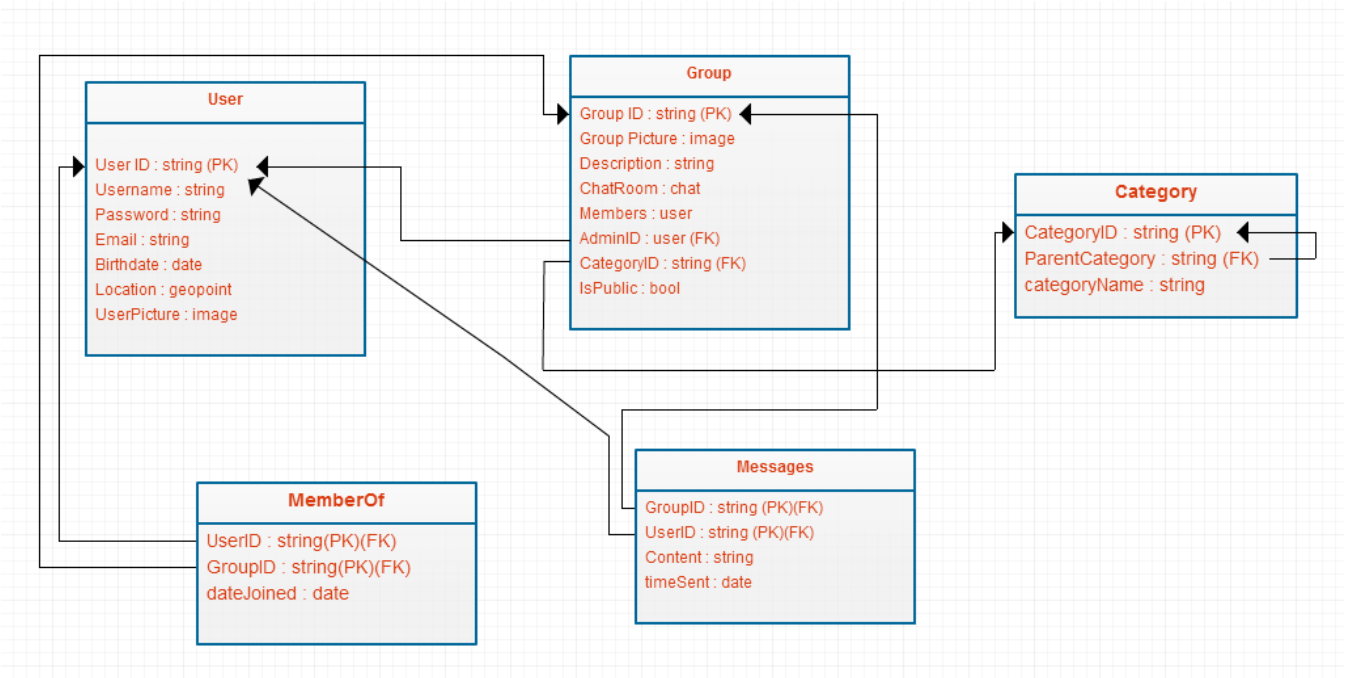
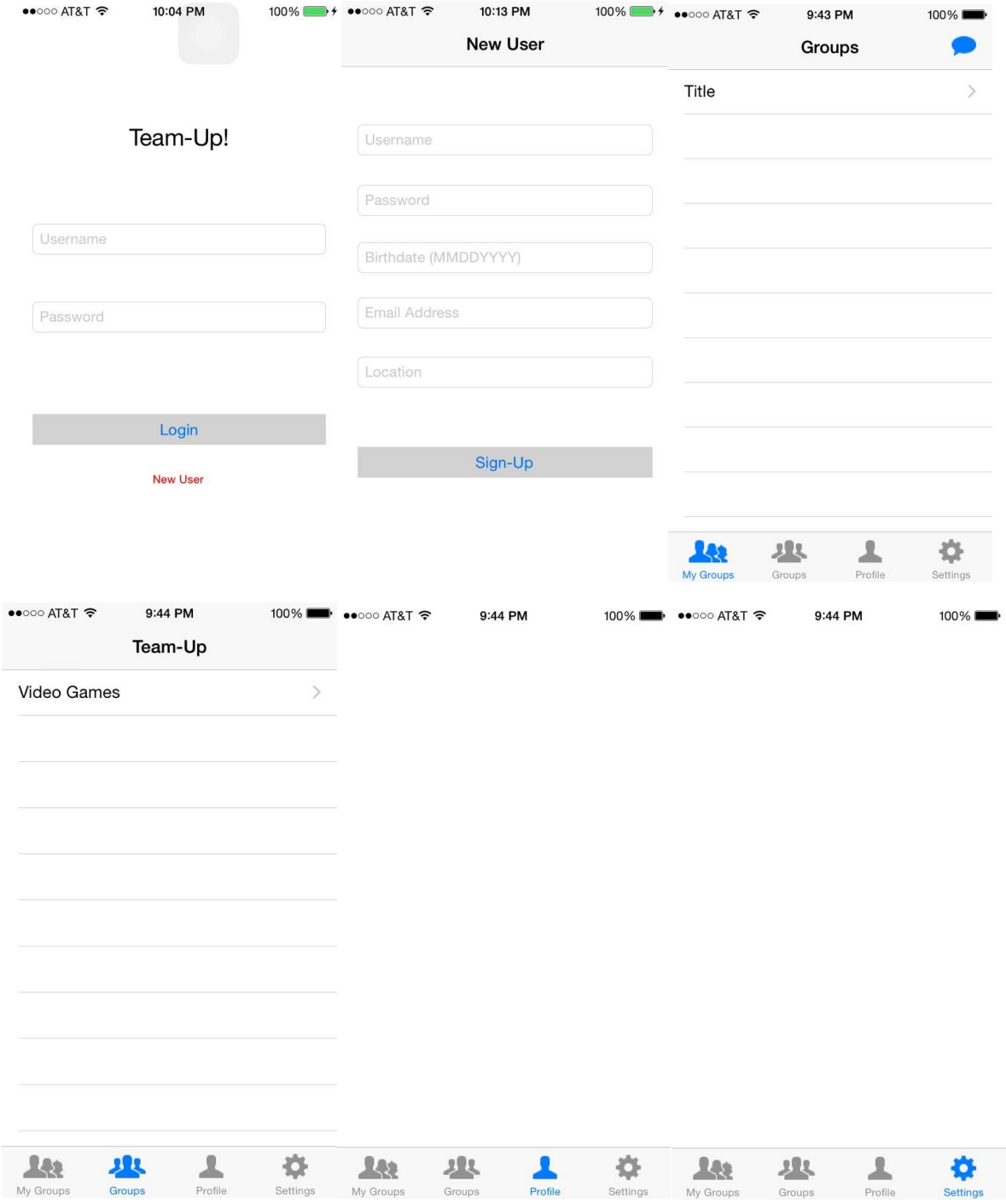
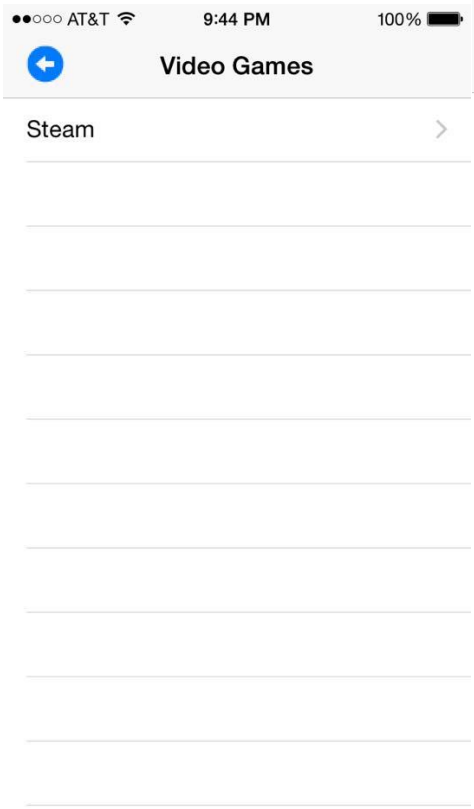
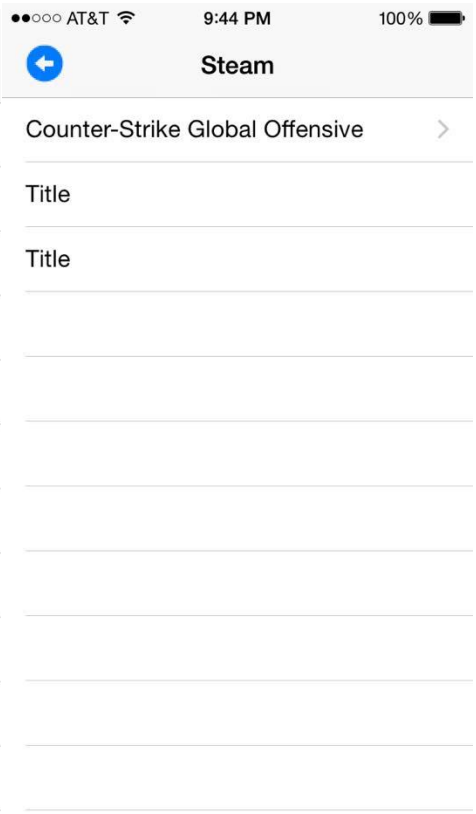


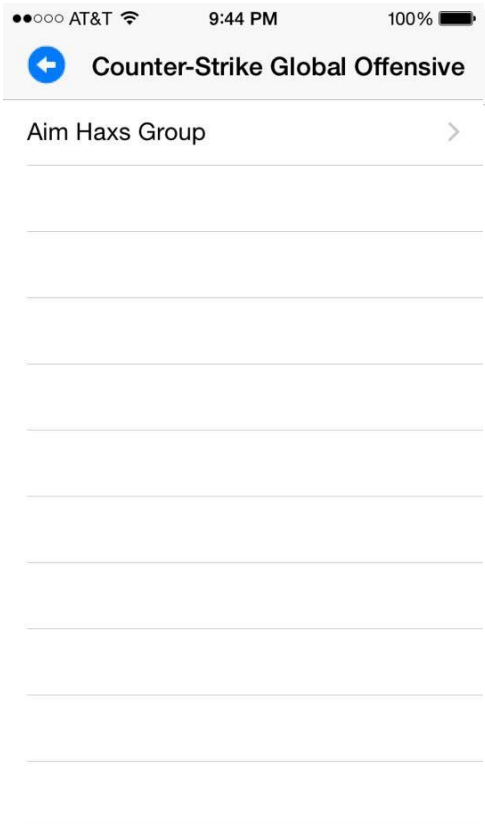
Figure 4. Backend Relational Database Schema

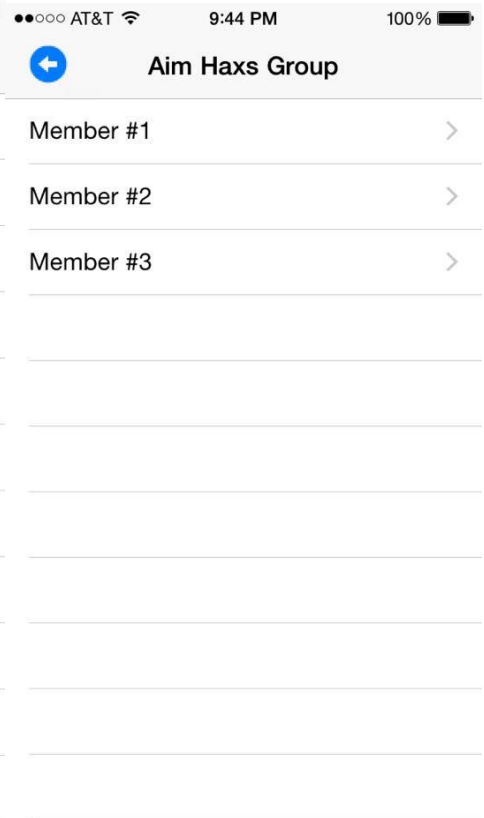
UI Mock-up:


●●○○○ AT&T 10:04 PM 100%
Team-Up!
Username
Password
Login
New User
●●○○○ AT&T 10:13 PM 100%
New User
Username
Password
Birthdate (MMDDYYYY)
Email Address
Location
Sign-Up
●●○○○ AT&T 9:43 PM 100%
Groups
Title
My Groups
Groups
Profile
Settings
●●○○○ AT&T 9:44 PM 100%
Team-Up
Video Games
My Groups
Groups
Profile
Settings
●●○○○ AT&T 9:44 PM 100%
My Groups
Groups
Profile
Settings
●●○○○ AT&T 9:44 PM 100%
My Groups
Groups
Profile
Settings


●●○○○ AT&T
9:44 PM
100%
Video Games
Steam


●●○○○ AT&T
9:44 PM
100%
Steam
Counter-Strike Global Offensive
Title
Title


●●○○○ AT&T
9:44 PM
100%
Counter-Strike Global Offensive
Aim Haxs Group


●●○○○ AT&T
9:44 PM
100%
Aim Haxs Group
Member #1
Member #2
Member #3