

GPS Friends

Final Report

Main word count: 1988
Apper context word count: 231

Introduction

GPS Friends is a location based microblogging service, allowing users to upload text and image posts to a dynamic content feed. Posts are tagged with the user's present location, and are viewable to users within the geographical vicinity of 500 meters of the post. Those users, who are seeing the post, may add their own comments to the post. The feed updates as the user moves through the sphere of other posts. Each post is assigned a category, and may also be designated as private, such that only their 'friends' may view the post. Users may request to be friends with other users, whom must approve those friend request.

GPS Friends was created out of a desire to make an application that would encourage people to explore their neighbourhoods. The app encourages wandering around and socializing, with the goal of turning neighbourhoods into communities - areas where citizens feel a sense of ownership through socialization and familiarity.

Overall Design

Our system consists of two main parts: the Android client application, acting as the front-end for user interaction, and a remote server machine, which consists of a database holding the application's data (such as user accounts and posts). An HTTP interface that allows the client to communicate with the server.

Figure 1 is a diagram of the overall system, which is followed by descriptions of each part.

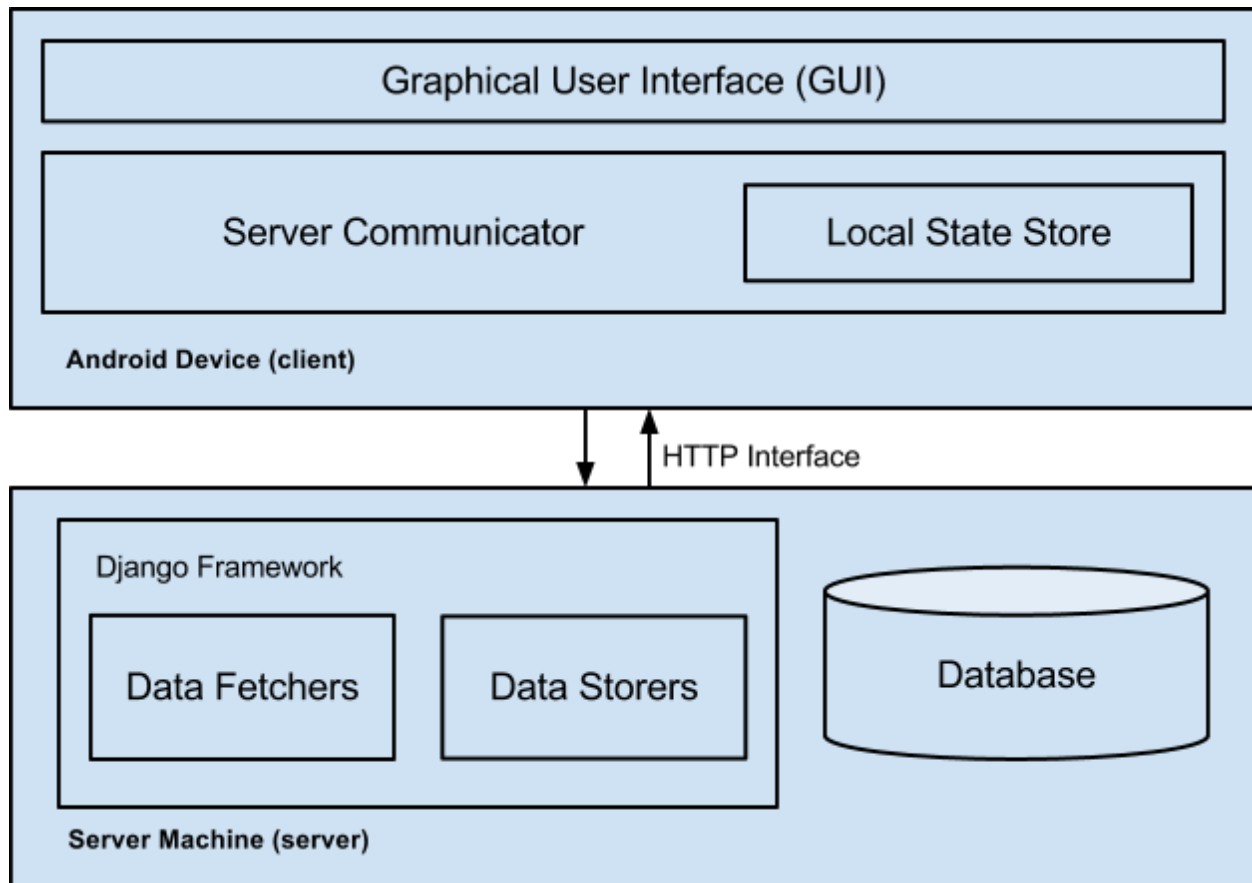


Figure 1: Block Diagram of the Overall System Design

Client Component Descriptions:

- **Graphical User Interface:** This consists of activities that allow the user to interact with the application on the Android device and perform actions such as registering a new user account, logging in, viewing posts made in the user's geographical vicinity, and submitting new posts. The details of this component are found in the Statement of Functionality & Screenshots section.
- **Server Communicator:** This component handles the details of communicating with the server via the HTTP protocol. It handles all submissions of data to the server, and parses/interprets the server's responses, which range in complexity from simple success/failure status indications to data sets retrieved from the database.

- **Local State Store:** This component, which is closely coupled with the Server Communicator, maintains a local set of data retrieved from the server. It contains information such as the session token required by the authentication backend, the current user's profile, and the posts in the user's vicinity. The store is updated whenever the Server Communicator receives new data from the server, and is used extensively by the GUI to display the contents to the user and allow the user to manipulate it.

Server Component Descriptions:

- **Django Framework:** This is a Python web framework used to rapidly develop dynamic web applications. It is used on the server to provide a simple communication interface for the client, access the database, and handle authentication.
 - **Data Fetchers:** These consist of the set of HTTP request handlers that are tasked with receiving information requests from the client, retrieving the relevant data from the database, and sending it to the client in XML format. The most important fetcher is responsible for returning a list of user posts to the client, given the client's geographical location (latitude/longitude). The server returns all posts within 500 meters of the client by converting coordinates between the latitude/longitude and Cartesian (x/y) formats, using a third-party library that computes a projection of the earth's surface into the Cartesian plane. Before returning the list of posts, the server also computes the distance and bearing (in degrees) of each post relative to the client's current location, which is later used by the GUI to indicate the location of each post relative to the client.
 - **Data Storers:** These consist of the set of HTTP request handlers that are tasked with receiving new information from the client and storing it in the database. All storers are simple, performing only minimal processing and validation on the received data, returning the success/failure status back to the client upon completion. This is by design - the more complex

computations are performed by the fetchers based on the information stored in the database by the storers.

- **Database:** This is a relational database system that is used to store all of the information required by our application (e.g. user accounts, posts, and friendships) in a series of interrelated tables. This project uses MySQL for this purpose.

Figure 2 is a visual representation of the project's relational schema:

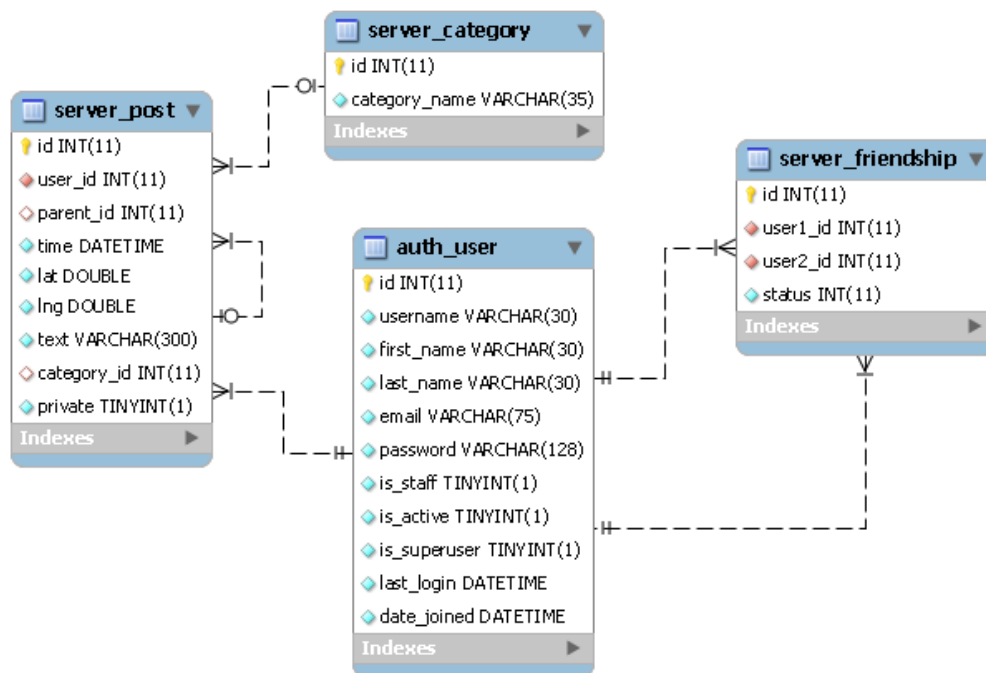


Figure 2: Database's Relational Schema

Statement of Functionality & Screenshots

The Android app consists of the following working functionality: login, registration, viewing the feed, viewing individual posts, creating posts, commenting on posts, requesting friends, approving friends, and updating user avatars (profile pic).

- **Login:** This activity (figure 3) is the start-up screen. It allows users to login to the server, using their username and password. Figure 4 shows the invalid username and/or password notification. New users may click on the register button, to start the registration activity.

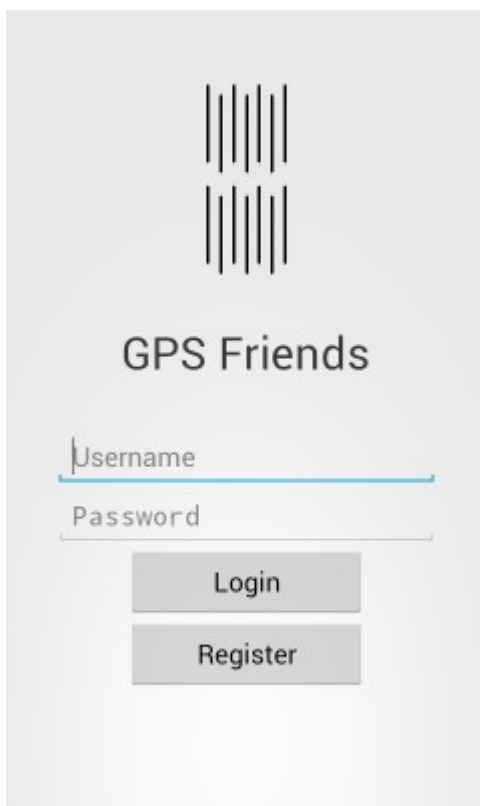


Figure 3: Login Screen

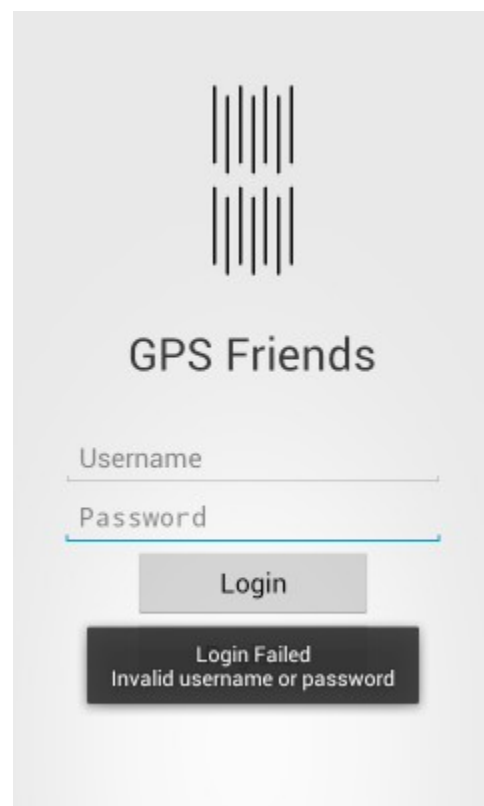


Figure 4: Login Failed

- **Registration:** This activity (figure 5) allows the user to register for GPS Friends. Registration requires a distinct email address, distinct username, first name, last name and password. The activity does some initial data verification, such as, do all fields have input, is the email a correctly formatted address, and do the two password fields match and are at least 6 characters long. If this initial data

verification fails, the user will be notified to which field(s) are incorrect (figure 6). Otherwise, the data will be sent to the server, which makes sure that neither the email address nor username are already registered. The user will be notified upon success and the app will return to the login screen (figure 7). If registration fails, the user is notified (figure 8).

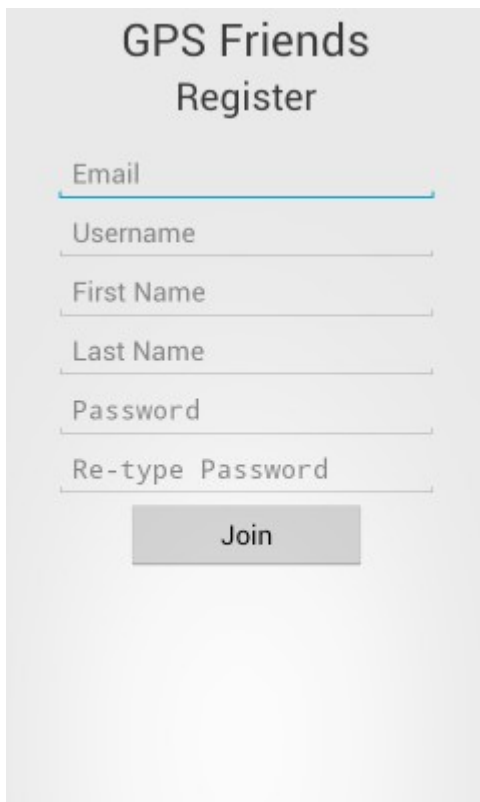
The image shows a mobile app registration screen titled "GPS Friends Register". It features six input fields stacked vertically: "Email", "Username", "First Name", "Last Name", "Password", and "Re-type Password". Each field has a light blue underline. Below the fields is a grey "Join" button.

Figure 5: Registration Screen

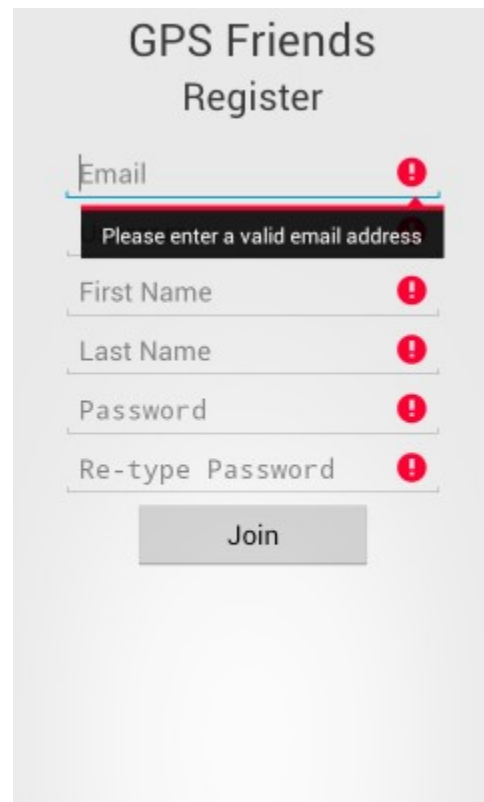
The image shows the same "GPS Friends Register" screen as Figure 5, but with validation errors. Red exclamation mark icons are placed to the right of each input field. A black error message box is displayed over the "Email" field, containing the text "Please enter a valid email address". The "Join" button remains at the bottom.

Figure 6: Incomplete Registration Data

- **Feed:** This activity (figure 9) displays posts in the user's vicinity. The list is populated by sending the current GPS coordinates to the server which returns all posts within 500 meters. Each post displays the username and avatar, the post's category, text, time, and distance and direction from the user current location and its number of comments. Also, each post, that contains images, will have a camera icon. As the user moves, the distances and the direction arrows are updated. The feed is automatically updated every minute or when the user moves five meters. The user may view an individual post by selecting it in the list.

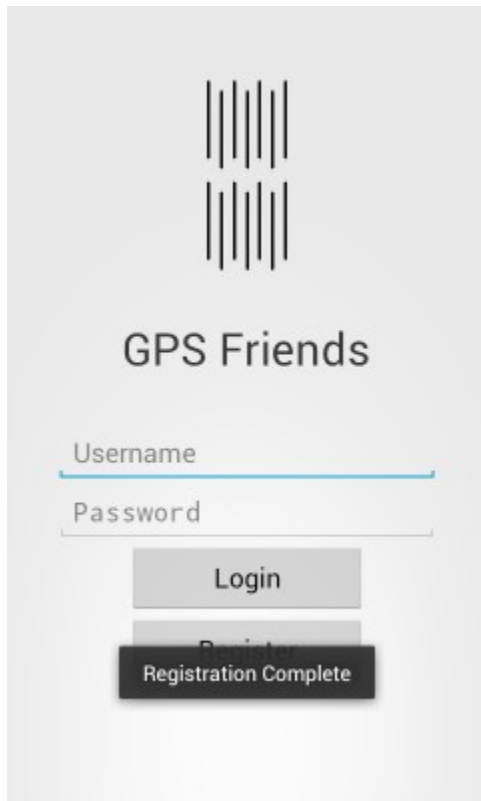


Figure 7: Registration Succeeded

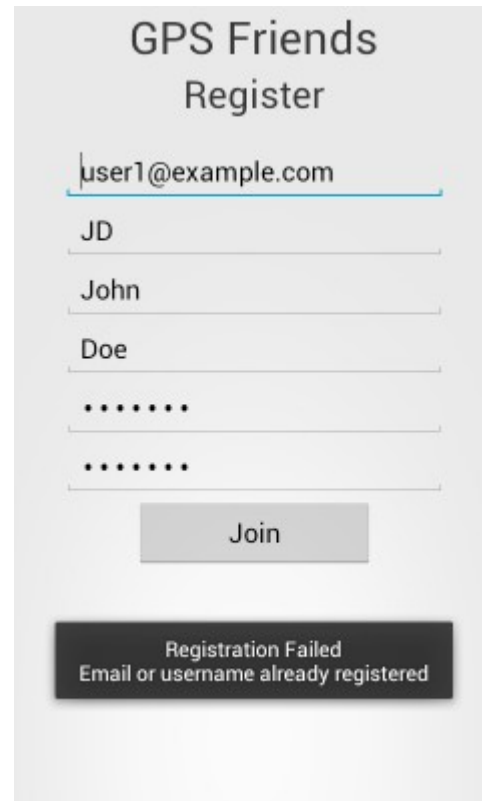


Figure 8: Registration Failed

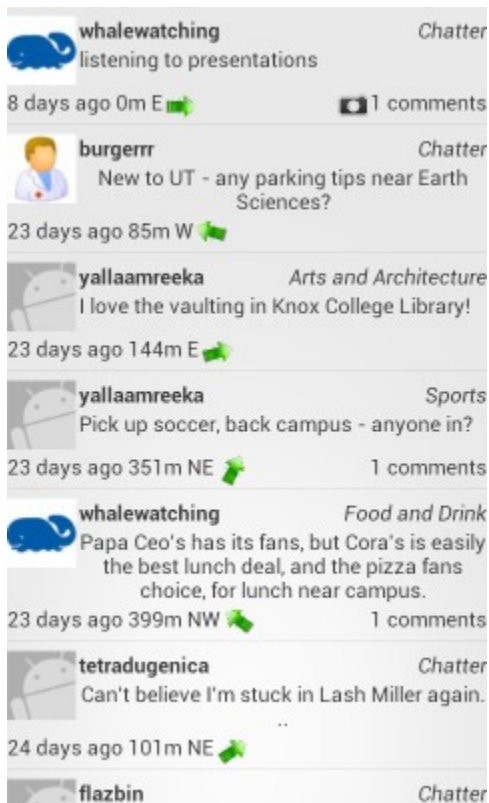


Figure 9: The Feed

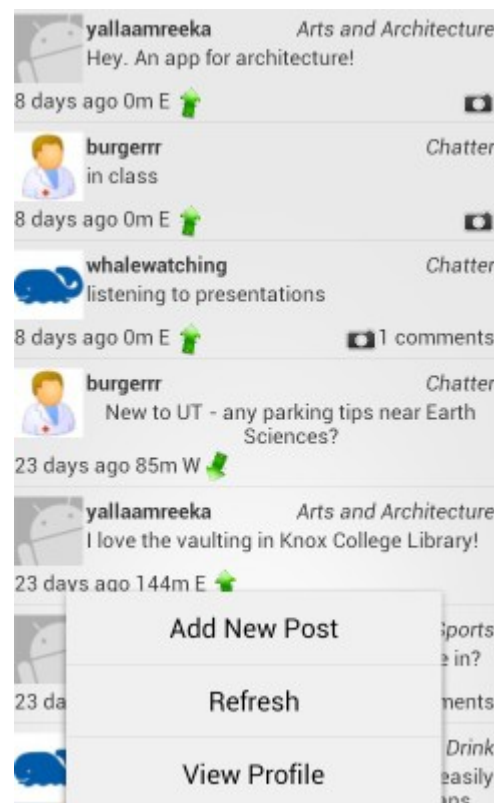


Figure 10: The Feed's Menu

This activity has a menu which allows the user to create a new post, refresh the feed, or view their profile (figure 10).

- **Posts:** This activity allows the user to see a post's comments (figure 11), images, make a comment (figure 12) and make a friend request (figure 13). If the post has images, it will be indicated by the camera icon (figure 14). Clicking the icon will display the images (figure 15).

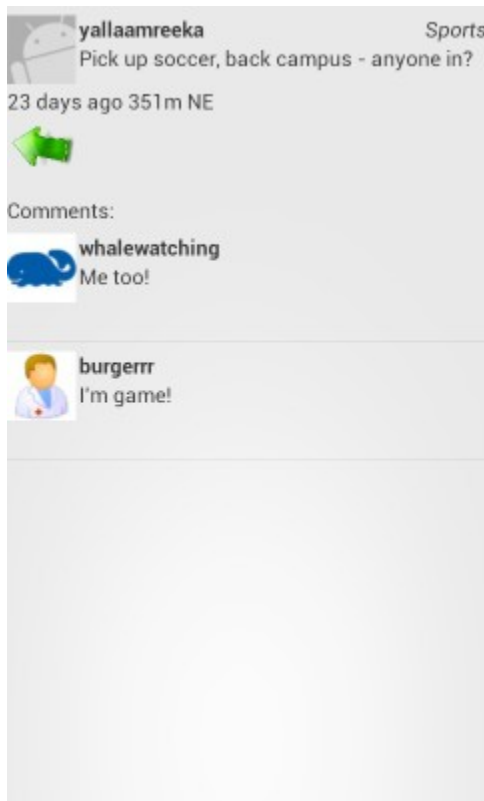


Figure 11: Post Screen

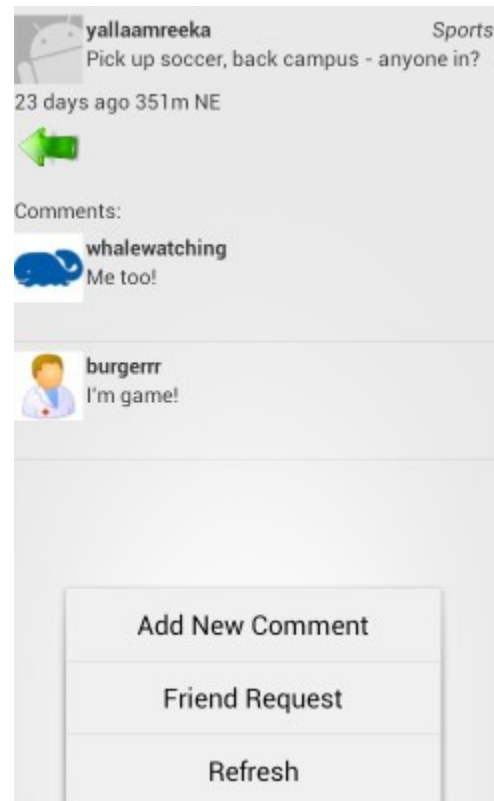


Figure 12: Post Screen's Menu

- **Creating a Post or Comment:** This activity allows a user to create a new post (figure 16), or comment on an existing post (figure 17). When creating a new post, the user must select a category (figure 18) and can optionally capture images to add to the post (figure 19). The user may view/delete images before posting.

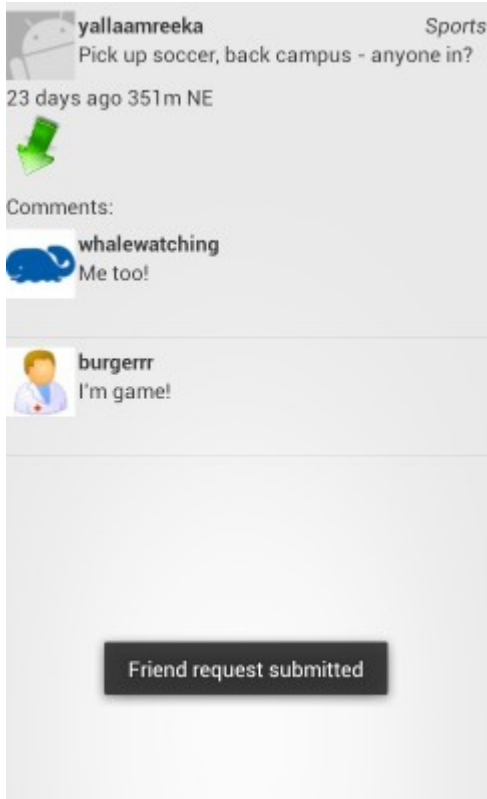


Figure 13: Friend Request

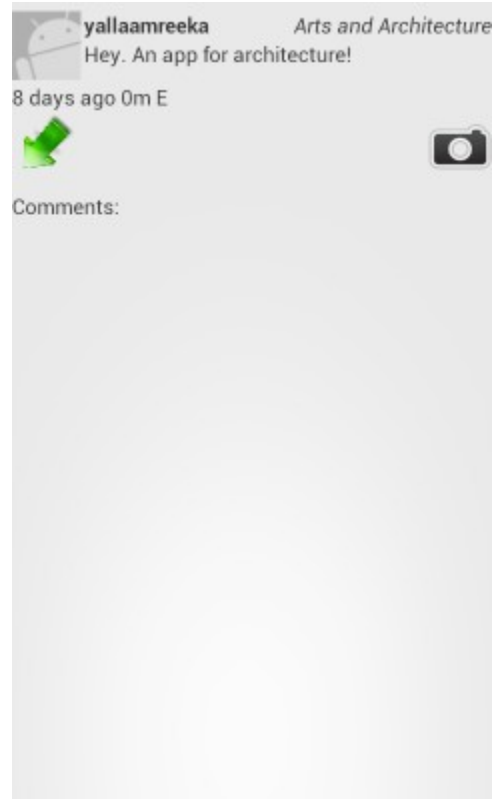


Figure 14: Post with Images



Figure 15: Image View

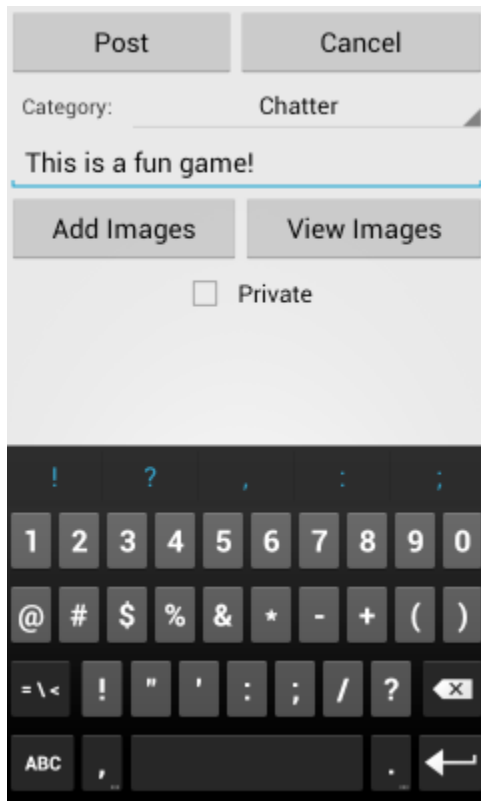


Figure 16: Create New Post

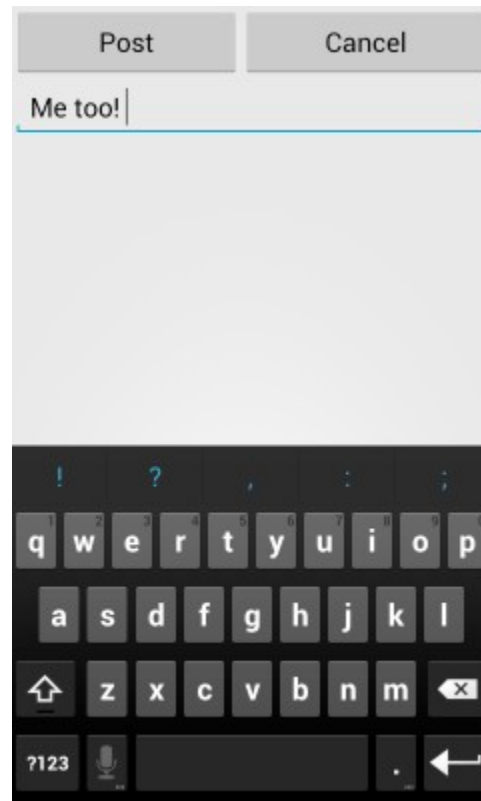


Figure 17: Add Comment

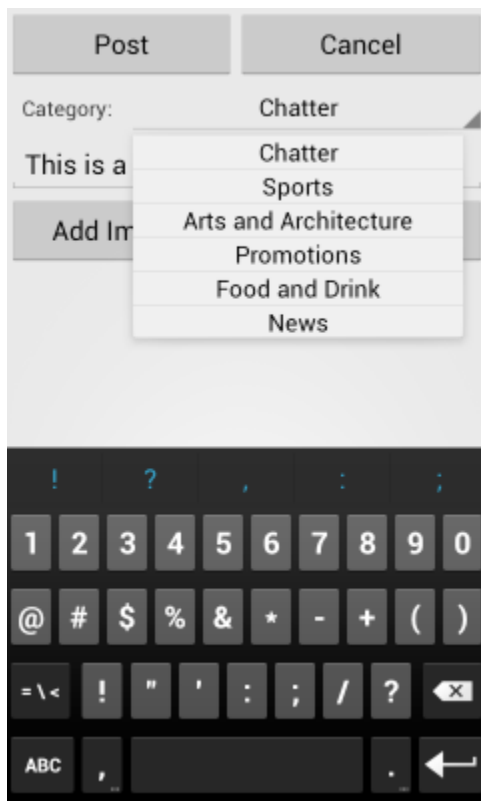


Figure 18: Choose Category



Figure 19: Capture Image

- **Friend requests:** A notification of pending friend requests are displayed at the top of the feed screen (figure 20). Clicking on the notification opens the friend request dialog (figure 21). If the request is accepted, both users will now be able to see the other user's private posts.

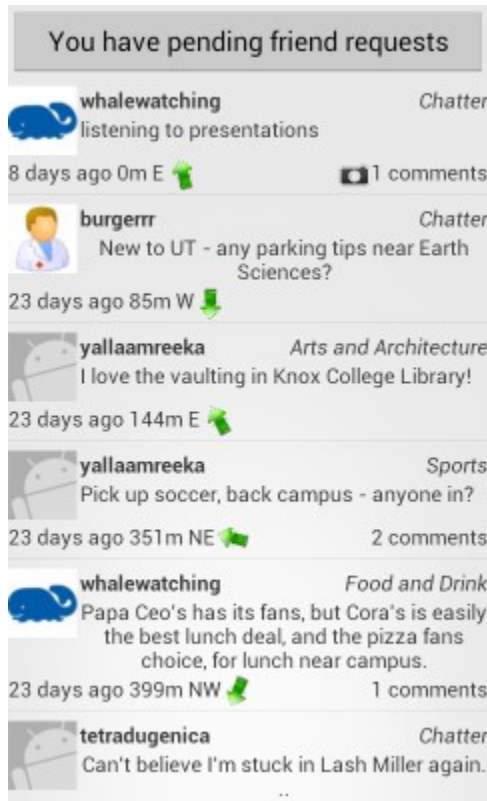


Figure 20: Pending Friend Requests

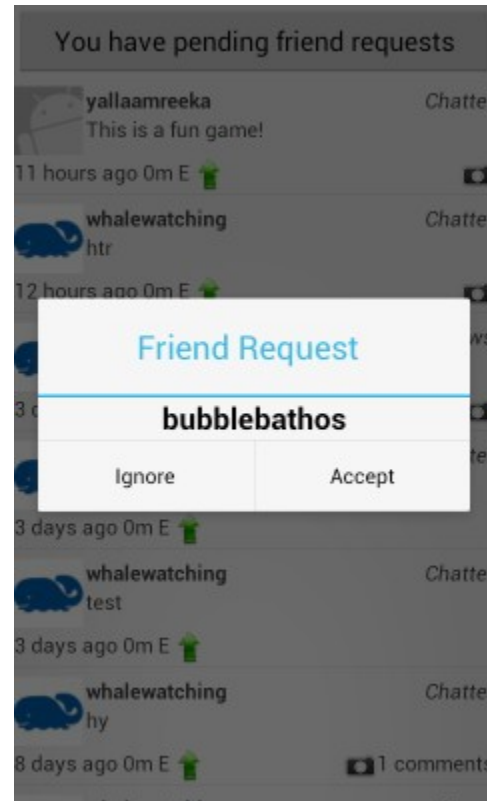


Figure 21: Friend Request Dialog

- **User Profile:** This activity allows the user to change their profile pic. A menu allows the user to change the profile pic and save those changes to the server (figure 22). Figure 23 shows the dialog for changing the profile pic. The dialog is populated with all images on the phone's SD card.

Key Learning

Programmers

The programmers' key lesson learnt was to spend time upfront gaining a thorough understanding of the Android debug environment and Android APIs. This

would have saved tremendous time later on in the project. An example is Android services. It would have been useful to understand how to use them to maintain a server connection across activity.

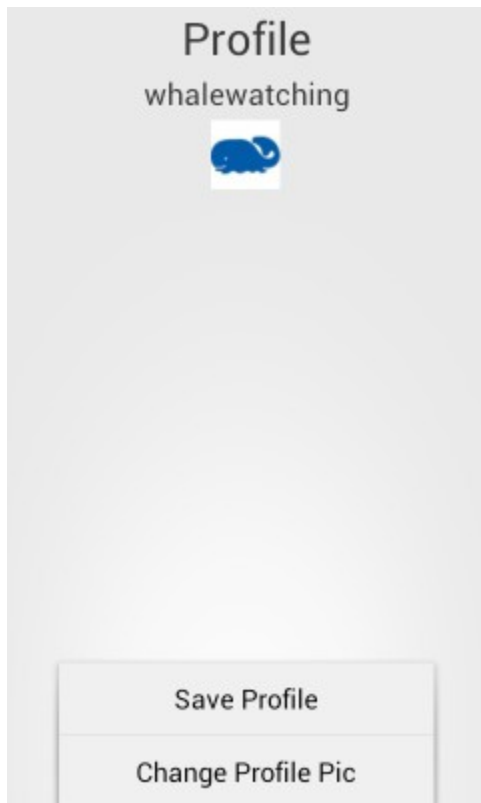


Figure 22: Profile Menu

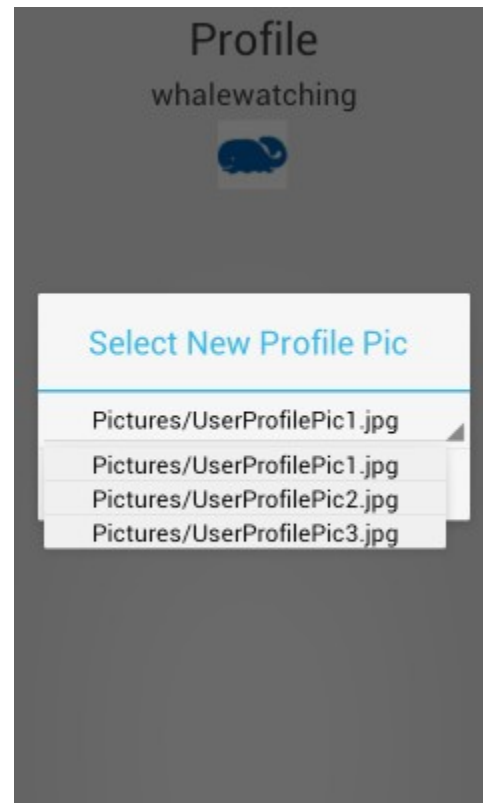


Figure 23: Profile Pic Selection

Apper

One of the key lessons that I took away from the process of designing GPS Friends is that it is extremely difficult to design a computer application without a strong technical knowledge of the code and systems underlying the app. I say this for two reasons. First, it makes communicating with those programming the application much easier - one can, for example, gauge the feasibility of a proposed feature before presenting it in a meeting. This streamlines the development process, and reduces distractions and scope creep. Second, it facilitates communication between the programmers and the designer by creating some degree of common ground, meaning that when features are discussed, the discussion is not bogged down with the cognitive burden of trying to understand one another.

A final note on communication and technical expertise. As someone from a very theoretical background, I grew to respect the difficulty in applying theory to the practice of making a technological object, while making it something viable, and that people would enjoy using. Being an effective architect or designer requires a vastly different skillset to being a talented theoretician, and the gap from one to the other may only be bridged by experience. With this in mind, I wonder if teaching the appers some basic technical skills would enable them to better bridge this gap.

Individual Contributions

Marc:

With the help of Blair and Felix, I formulated, and continued to develop, the initial concept for the app, up to and including its current form. I conceived of the UI, from an initial set of schematics, to its present form, iterating as the feature set evolved. I created test data, so that the app's full functionality could be displayed during demonstrations. Finally, I made the presentation slides, and contributed to the written components of the project.

Blair:

I wrote all the code for the app's GUI. This code includes all the activities, their associated layouts and menus, communication with the GPS, camera and compass, and the threads calling the client side communication function. Also, I was responsible to testing and debugging the app with server at every spiral. Finally, I did the final review and revision of all presentations and reports.

Felix:

I wrote the server-side application using Python and the Django framework, and developed the database schema. On the client side (i.e. Android device), I wrote the code responsible for directly communicating with the server, parsing the XML data returned by it, and maintaining state information such as user login sessions. I was also responsible for setting up the initial server environment and an SVN server to be used for collaboration among programmers.

Apper Context

My academic background is a scattered one. On one hand, I studied Theory & History at the Architectural Association School of Architecture. On the other, I am studying Knowledge Management at the iSchool. Being completely honest, the latter has not exactly captured my imagination, so I decided to emphasize the former when formulating ideas for an app. I have spilled much ink on how to form or create livable communities, how to make our cities more humane places, but I have never been given the opportunity to implement theory.

As I mentioned above, this process does have its challenges, but in many ways, I think that GPS Friends fulfils the mandate of making neighbourhoods and cities more like communities. This application is an engine both for the dissemination of information and memory, and the creation of new ones by way of exploration, and the expansion of one's social contacts. Perhaps most interestingly, this is all achieved by way of a virtual overlay - a web of posts accessible by way of a mobile device. The potential of a virtual architecture, particularly one in dialogue with a physical one, fascinated me.

Incidentally, my iSchool education did end up being of use too - thinking about metadata led me to create the post categories as a way to quickly and easily create new virtual maps, by rearranging and reorganizing pre-existing data.

Future Work

We would like to implement push notifications for replies and friend requests. In addition, we would like to implement some means of filtering posts by category, allowing a user to only see what interests them. A basic profile, where users could input their interests, and the ability to searching according to those interests. For example, someone might want to know where other people who like to speak Spanish hang out. Finally, we would be interested in learning more about what a marketing/entrepreneurship class could do for the app.