Scott McClellan, Joshua Foeh, Tushar Sharma, Trevor Edris, Chris Schlatter
Team 8 CS 307

Design Document
Project Name: “Xpertise”

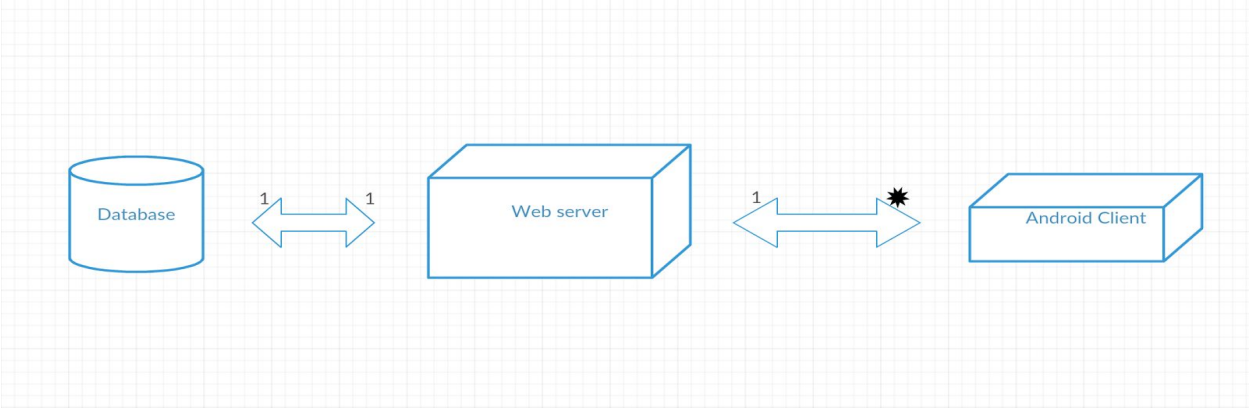
Purpose

This program’s intended purpose is to allow people with similar interests and hobbies to find each other and work together to accomplish tasks that would otherwise be very difficult if not impossible to complete alone.

Design outline

For our project, we have decided to implement a backend server that communicates with a mobile device application. The server will contain a database that will store user profiles for easy lookup as well as sensitive user information such as passwords and usernames. The mobile app will serve as a user friendly GUI to display information retrieved from the server.

The mobile app will establish a connection to the server and send requests for data as the user uses the app. The server will receive these requests for data, do the appropriate lookup in the database, and, depending on the search, sort the data before sending it to the app to be displayed.



Design Issues

Ensure you spend enough time thinking about the design issues. Only one or two design issues will not be sufficient to get full credit. Each design issue requires a descriptive title, solution options for the issue, and justification of your choice. You may divide this section into two subsections, Functional Issues and Non-Functional Issues :

Server-App Communication:
Problem: How will the server and the app communicate data over a network?

- Solution 1: Using JSON packets and a RESTful API we can send almost any kind of data across the internet.
- Solution 2: Simple HTTP get, post, and delete requests could be possible.

Chosen Solution: Using JSON packets with a RESTful API. We chose this solution due to its flexibility and ability to send varying data types such as pictures which is easier to use than HTTP requests.

Server Database Structure:

Problem: How will the server store user profiles for speedy lookup up?

- Solution 1: Using a hashmap solution based off of a unique numerical code generated at profile creation, we can store profiles in such a way that gives us the fastest possible lookup times but is not very memory efficient.
- Solution 2: Alphabetical sorting into some type of array or linked list could give fast lookup times given our searching algorithm is well made. However, entering new profiles could take a long time depending on how many profiles exist in the structure already. It could be possible to keep indices of where each letter starts/stops in the array so as to only search through entries with the same starting letter.
- Solution 3: Using a ranking algorithm to rank people depending on various parameters such as location, skill-level, and etc…
- Solution 4: Offload that decision to a pre-made database and use the API to communicate with it

Chosen Solution: Using a pre-made database. By using a premade database we don't have to worry about the small details of keeping and maintaining large data. It also saves us time by reusing an established working system.

User Login Storage

Problem: How will the server store user login information and verify against it at login?

- Solution 1: Using a separate data structure we could encrypt the passwords, usernames, or both into numerical codes which can then be hash mapped.
- Solution 2: A traditional approach of simply storing and retrieving structs containing login information.
- Solution 3: Storing login information into nodes and placing them in a dynamic, self-balancing tree structure. This could normalize both search and add times to decent levels even with large data.
- Solution 4: Store the user login information in the same pre-made database as all the other data.

Chosen Solution: We decided to store the user login information with all other data in the pre-made database. This reduces complexity but is not as secure. We aren't worried about security at this moment due to having a very small database and since we do not store any really damaging information.

Location

Problem: How will the user's location be calculated/determined?

- Solution 1: Wifi location
- Solution 2: Direct GPS usage

- Solution 3: Android location services

Chosen Solution: Android location services. We decided to use the android API to interface with the mobile device's GPS. This will make it so that we can get location data on any model of phone without having to worry about differing GPS technology

Design Details

Login, this is the first page the user will see where they will enter their username and password or select to create a new account.

- If the user selects to make a new account the SignUp will be called
- If the user enters a valid username and login then Home will be called

SignUp, this has users enter their username, password, and email. It then verifies with the server that this username and email are not already in use and if it is not then it creates a new profile object for the user.

- Once it is verified that the information is valid then the user will enter their personal information to fill out their profile.

Home, this is the user's home page where they can choose to search, view their own profile, or select popular tags, it also has a taskbar at the top for settings, viewing chats and other things.

- If searching is selected then the category to search on may be picked then SearchList is called.
- If chat is selected then the Chat class will be called along with the appropriate argument to open the correct chat.

Chat, this is the class that handles chatting between two or more clients, it will be a simple chat client that only supports text messages.

- The querier must be accessed so that the messages can be sent and received over the network.
- The chat can be started by selecting another user's profile and clicking a button to send a message.

SearchList, this will display the results of the user's search based on the criteria that they choose. Listed profiles may be clicked on so that that profile may be viewed in it's entirety.

- The searchlist will display a small snapshot of the other user's profile information such as their name and tags.

Profile, this is the class that houses a user's profile information that can be viewed through searches. It will display all of the user input information as well as have buttons to review the user and send a message to the user.

- You may review other user's, those reviews and ratings will then appear on their profile either anonymously or with your name on it depending on what you choose.

Review, this will be the data object that holds the reviews that user's can make about one another. This can be viewed on the profile of the user that was reviewed.

Querier, this will be the only class communicating over the network with the server. It will accept arguments from the other classes and return profile information that it gets from the server.

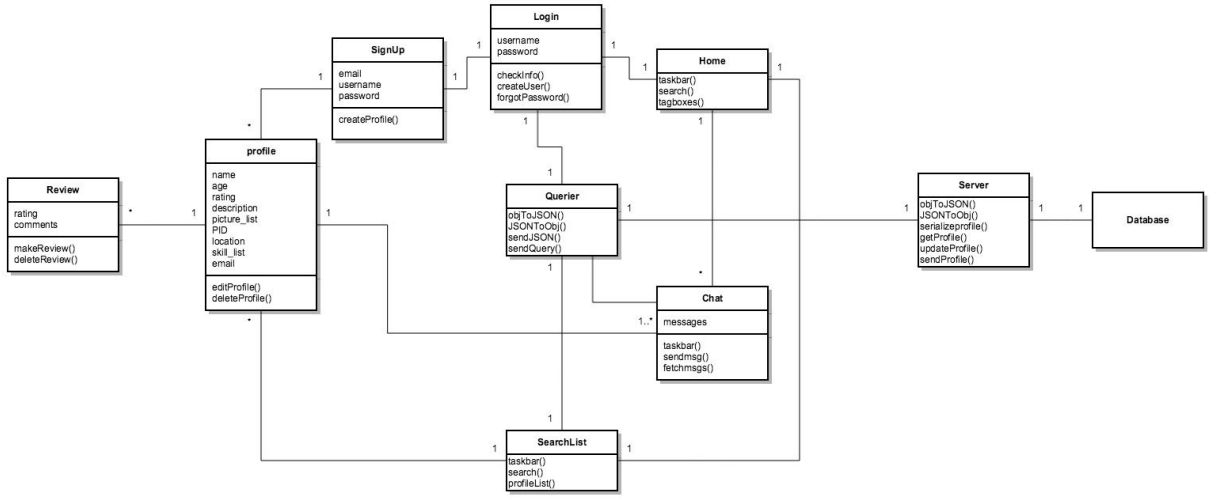
- The querier communicates with the server directly and handles the network communication aspect of the application.

Server, this will be part of the back end of the application and will not be run on the user's device. It will be run on a separate server.

- The server handles communication between the application and the database which stores the information.

Database, this holds all of the information for the application. It will hold chat information, reviews, and profile information. It will not communicate directly with the app and will instead communicate with the server only.

Class diagram:



Sequence diagram: