**Jump Start**

Saim Malik

1321796

CS 2XB3 – Lab Section 1

**Abstract:**

Jump Start is going to be a mobile application that would help its users quickly find help if their car’s battery is dead. Normally, the person would try to find someone with a working car and connect with their battery. With this app, that entire process will become a lot easier. The user would just have to open the app and press the help button. The app would then automatically track the closest app members to the user and ask them to help. If they tap yes, the user will then be connected with the helper’s phone in a chatting environment. All the app members would be stored in a database and their location would be used to track other members close to them (using graphing algorithms). A product like this could be very useful for emergency situations when there’s no one nearby to help.

**Objective and Scope of the Project**

The main problem Jump Start is going to solve is the difficulty people have to face when looking for urgent help. Most people unfortunately, don’t respond to their fellow human being’s need. For instance, my dad and I once found ourselves stranded in an unknown place because our van’s battery died. We tried several times to get the incoming car to stop and help us jump start our car. However, most of them probably did not know why or what we were trying to communicate and just drove by. Of course we eventually did find someone and got the car fixed later. However, if we had access to an application like Jump Start, a lot of time and effort could’ve have been saved.

Now that the motivation of the idea is clear, how am I going to implement it? The idea is to have a really simple user interface so most of the work will be done on the back-end. For instance, the application could just have a button to initiate ‘help’ which will then perform bunch of tasks in the background before giving the user a list of ‘helpers’ to choose from. Of course there would have to be some way to keep track of all the users that have the app and store them as ‘members’. This could be accomplished by prompting the user to create a unique username upon initial launch of the application and then storing that username into a database. The most difficult thing to implement would definitely be finding the closest members to the user that needs help. To do this, we could treat each user’s location as a node in a graph. This way, finding the closest app members would be as simple as finding N number of closest nodes to the user’s node (i.e. the edges would represent the distance b/w nodes). The next step would be to notify and give an option to help to all the app members that are close by. The ones that respond ‘yes’ would show up on the user’s screen to choose from.

Now if the app were to just reveal the user’s location to the app member they selected, it could be very unsafe. Thus, it will simply connect the user with that member in a chatting environment and from there they can arrange to meet. The user can also rate their experience with that specific member so that particular member would be given preference over other users next time. This would also make it a bit safer as the user will have some reason to trust whoever they’re chatting with.

**Input/output:**

The inputs for this application would be the user’s unique name and their location which will automatically be detected via their phone’s GPS. Since the entire point is to find the closest people to help, the location will be frequently updated (hence the graph will not be static). The output would be the result of searching for app members nearby. From the GUI perspective, it would be a list of people willing to help, from which the user will be able to pick one and chat with. The chatting environment will be maintained separately from the rest of the application for simplicity purposes.

**Algorithm Challenges of the Project:**

Search and sort algorithms will be used to maintain the database in a way that optimizes insertion and delete of users (e.g. clustered location wise or stored alphabetically). One of the graphing algorithms, shortest s-t path, will be used to figure out the shortest path b/w the nodes (from user’s node to other nodes). In order to keep track of all the users’ locations at all times, the app will need perform other operations in the background to update the location parameters (received through their phone’s GPS) every X amount of times per day. As described earlier, since every edge represents the distance between two nodes, an upper bound would have to be set to decide whether the given two nodes should be connected. This process might be slow but it will speed up the part when performing shortest s-t path algorithm b/c there could be a lot of separate connected components in the graph (depending on how small the upper bound is).

**Project Timetable:**

|  |  |
| --- | --- |
| Days | Task |
| 4 | Create and finalize the design |
| 1 | Setup GUI |
| 2-3 | Get user sign up process working   * Store usernames and locations in the database |
| 3-4 | Implement the Graph data structure   * Have find closest users functionality working |
| 2 | Create the chatting environment GUI and functionality |
| 2 | Finalize the application and its documentation |

**References:**

Sedgewick, R., & Wayne, K. (2011). Graphs. In *Algorithms* (4th ed., pp. 518-565). Upper Saddle River, NJ: Addison-Wesley.