CSE 570: Wireless and Mobile Networking
Project on Location and Trajectory: Location Wiz-

An Android App to learn your daily schedule and suggest friends based on your schedule

Submitted by:

Name	SBU ID	Contribution
Nittin Aggarwal	111401512	50% . Tasks: Friend Suggestion and Attendance Features. Server side scripts.
Vaibhav Mathur	111447903	50%. Tasks: Android App interface, My Schedule and Anomaly Notification features Integration of Apache Server and Android App client.

1. INTRODUCTION:

In today's fast paced world, everybody is in a rush and we frequently forget to be at certain places at designated times. It would be fantastic if someone provides us a reminder when we need to start travelling to reach the location of our next appointment without explicitly writing down our schedule! The Location Wiz does exactly this with minimum input from a user. The Schedule feature of Location Wiz keeps track of all places you visit daily and after collecting sufficient data, it constructs a weekly schedule for you, showing the locations you tend to be at every half an hour during a day of the week. Based on this schedule, the sends you push notifications whenever an anomaly is detected between your current location and what your schedule says. It also notifies you when it is time to travel to a certain location but you are stationary.

Another exciting feature of Location Wiz App is Friend Suggestions. Generally, individuals who are present at same locations at the same times have similar lifestyles and interests. Would it not be an absolute pleasure to make acquaintance with people who have congruent schedules as yours? The likelihood of such people developing great understanding and friendship is very high. The Location Wiz Server looks for such congruence in different people's trajectory in terms of both time and space. The Location Wiz Server constructs a score for every possible pair of users and sends friend suggestions to the user based on the score.

Taking attendance in every lecture is a time consuming exercise. With the data collected by the Location Wiz App, taking attendance of students in a lecture can be automated. A Professor just needs to define the time and location of a lecture and if the students use the Location Wiz App, their attendance will be determined for each lecture. The Professor will be able to get the attendance of each on a separate Attendance Android Application which we have developed. The entire process requires absolutely no effort from the students and the Professor once the location and timings of the class are defined via the Attendance App.

2. METHODOLOGY

Data Collection: The Location Wiz App collects location data from a user at every two-minute time interval using Google API for getting last known location. This data is locally stored in a file on the device and at end of each day is transferred to the server via HTTP POST REST call. This location data file is named as : 'Location_Data_<username>_<date>'. A sample data point from this file:
40.9073,-73.1079,15 Nov 2017 12:00:48 a.m.,Chapin Apartment L, Stony Brook, NY 11790, USA
The username is collected from the primary email ID the user is using on their android device.

Client Server Architecture:
The server is hosted on the Google Cloud Platform VM. The VM runs 4 GB RAM, 25 GB Hard Disk and has Ubuntu 17.10 operating System. An Apache HTTP Server 2.4.29 was hosted on this VM. At end of each day the Location Wiz App sends the location data file via a HTTP POST REST call. The server stores the daily location data files for all users in one location for further processing.

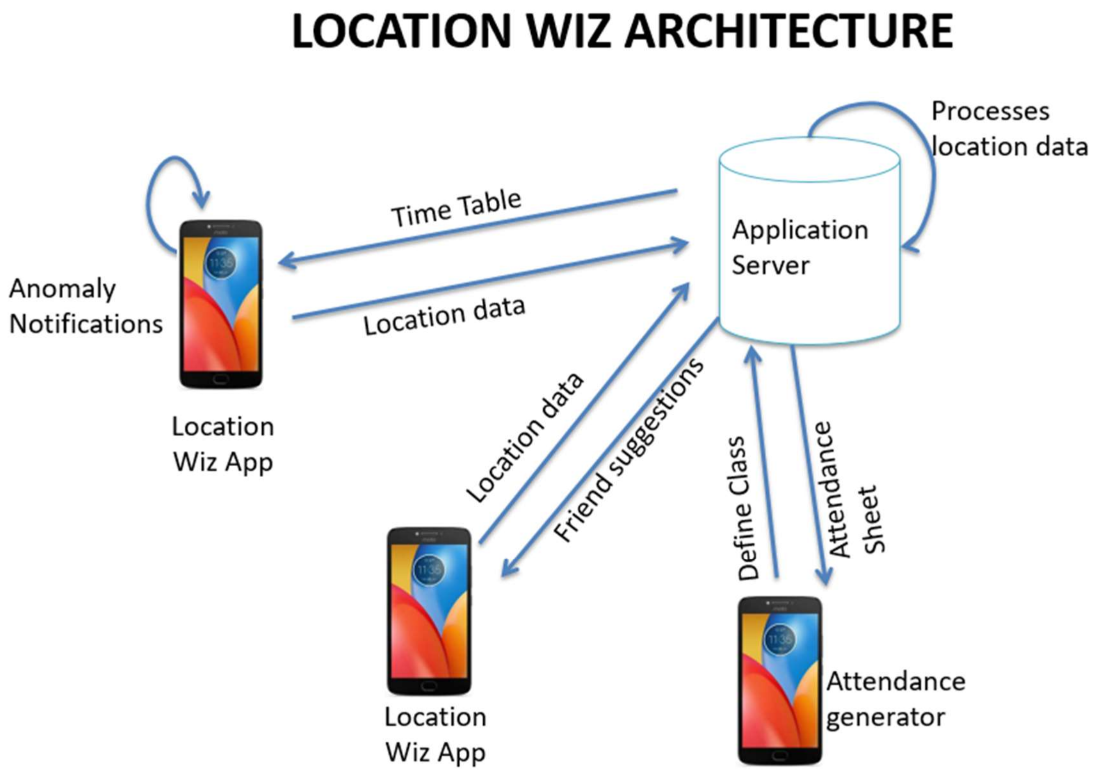


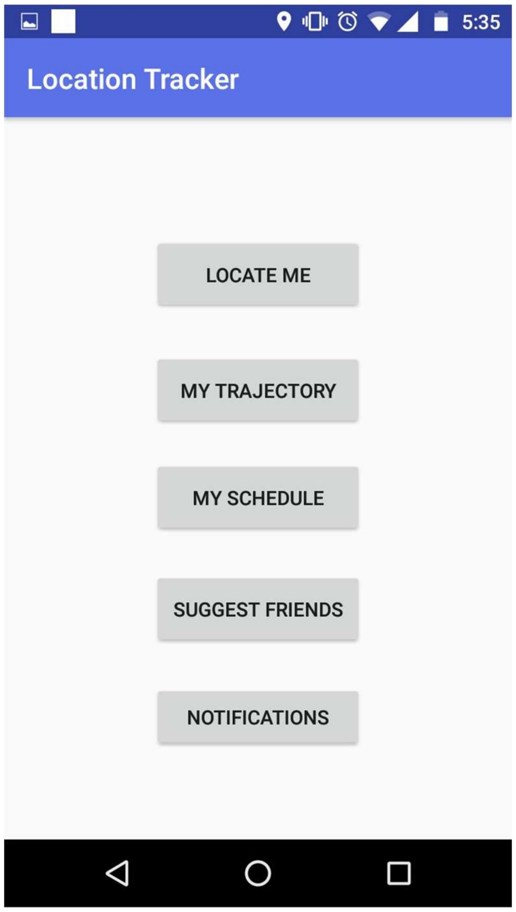
Figure showing Client Server Architecture of Location Wiz App

Using the location data files, the server performs several operations that translate into the many features of Location Wiz App. The server runs the following Python scripts on the location data:

1. Schedule.py: This Python script processes location data for a user for a particular day of the week and ascertains what the location of the user for every half an hour of that day. For example: The script will read all location data files for Mondays and ascertain the location of the user for intervals of half an hour. If the location of the user changes frequently during an interval, he is deduced to be travelling. If there is a conflict in location at an interval a majority vote is taken among all the locations to decide the location.
2. Friends.py: This Python script is used to determine potential friends for a user. The script compares the location data of one user across different users for all days and calculates a score that reflects the coexistence of current user with other users. (The score for a pair of users keeps increases when the two are at the same place at same time). The script returns the top four scores along with the usernames. These four users are suggested as potential friends for current user.
3. Attendence.py: This Python script takes in a location and timing of a class and returns the presence or absence of the users at that location in the time interval mentioned as input.

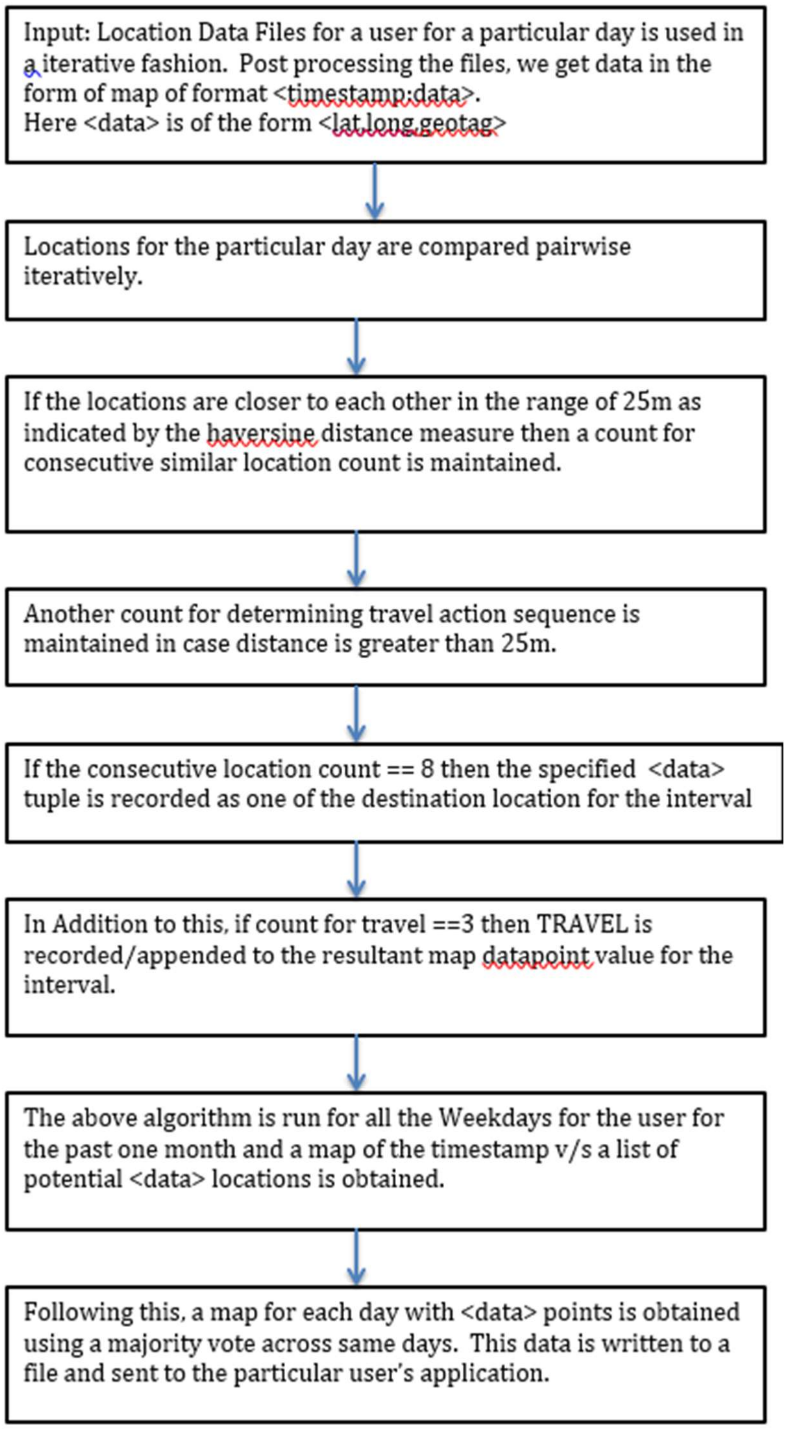
Features:

The home screen displays the variety of features offered by the app:



1. MY SCHEDULE

The location data which is collected from each user per day is appended to a .csv file which is pushed on to the server using a HTTP POST request at the end of the day. A Schedule.py script runs for every user input-file daily to generate a mapping of every 2-minute interval (starting at 12:00:00 am till 11:58:00 pm) to a latitude and longitude pair along with the geotag. Due to irregularities in data collection, data is not available when the user does not move, the location for such an interval has been safely assumed to be equal to that of the last location retrieved.



Input: Location Data Files for a user for a particular day is used in a iterative fashion. Post processing the files, we get data in the form of map of format <timestamp:data>.
Here <data> is of the form <lat,long,geotag>

Locations for the particular day are compared pairwise iteratively.

If the locations are closer to each other in the range of 25m as indicated by the haversine distance measure then a count for consecutive similar location count is maintained.

Another count for determining travel action sequence is maintained in case distance is greater than 25m.

If the consecutive location count == 8 then the specified <data> tuple is recorded as one of the destination location for the interval

In Addition to this, if count for travel ==3 then TRAVEL is recorded/appended to the resultant map datapoint value for the interval.

The above algorithm is run for all the Weekdays for the user for the past one month and a map of the timestamp v/s a list of potential <data> locations is obtained.

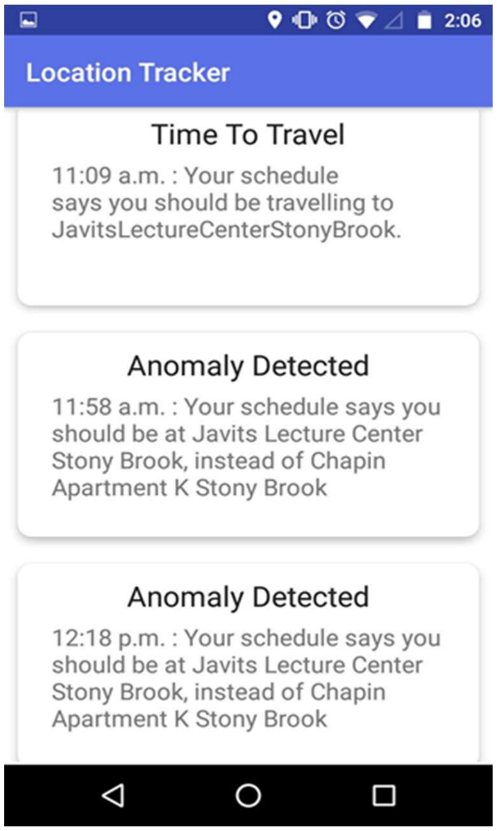
Following this, a map for each day with <data> points is obtained using a majority vote across same days. This data is written to a file and sent to the particular user's application.

The algorithm for computation of the schedule is illustrated above in form of a flowchart.

2. ANOMALY NOTIFICATIONS

An extension to the above feature has been added to make the application more interactive in form of anomaly notifications. This feature uses the generated schedule above to determine the proximity of the user device to the learned destination location for each interval. The algorithm can be summarized using the following procedure:

1) The Anomaly Notification feature is provided by a service run in background for every interval for the user.
2) If the user's current location is different from the desired location as prescribed by the schedule for that particular interval by haversine distance of 50m then the location difference is marked but no action is taken.
3) This procedure is repeated for each interval for a time interval of 20 minutes. If the user device does not reach the destination's proximity, then a notification explaining the anomaly is sent to the user as a push notification.
4) In special case for TRAVEL, if the user's distance does not change for more than 8 consecutive minutes by 25m, not even once for the duration of the interval then a notification explaining "TIME TO TRAVEL" is triggered.



Time To Travel

11:09 a.m. : Your schedule says you should be travelling to JavitsLectureCenterStonyBrook.

Anomaly Detected

11:58 a.m. : Your schedule says you should be at Javits Lecture Center Stony Brook, instead of Chapin Apartment K Stony Brook

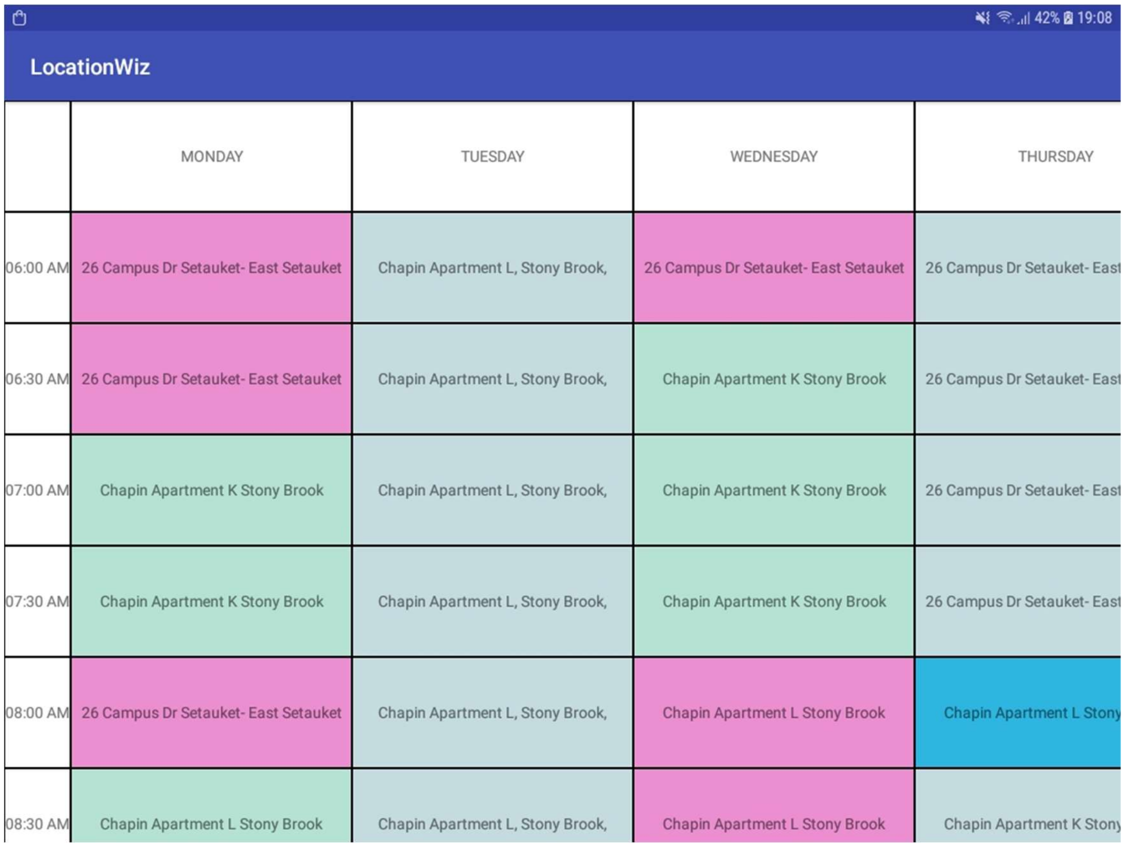
Anomaly Detected

12:18 p.m. : Your schedule says you should be at Javits Lecture Center Stony Brook, instead of Chapin Apartment K Stony Brook

Screenshot of Anomaly Notifications

Results and Limitations:

The schedule has been learnt using data of the user for the past 4 weeks to generate a schedule of all the potential locations using majority vote for all intervals of length half hour in the day. The notifications' feature is based on this schedule to alarm the user of any deviation from the desired scheduled location for each interval in a day. It has been observed that the notifications' feature is working with high accuracy based entirely on the schedule for days where data is not learnt i.e on test data.
The application is meant for target audience who have a similar schedule which repeats after a considerable time and hence possess patterns. In true sense, it is meant for users who do have a schedule and would like to follow one. Hence, for users who don't have a fixed trajectory over a period of time, the application does not cope well. Moreover, the assumption that a schedule would repeat after learning for a month long data for a user seems to be a fair assumption at-least for audience such as students, job goers etc.
A sample snapshot of the generated schedule is shown below for clarity.



42% 19:08

LocationWiz

	MONDAY	TUESDAY	WEDNESDAY	THURSDAY
06:00 AM	26 Campus Dr Setauket- East Setauket	Chapin Apartment L, Stony Brook,	26 Campus Dr Setauket- East Setauket	26 Campus Dr Setauket- East
06:30 AM	26 Campus Dr Setauket- East Setauket	Chapin Apartment L, Stony Brook,	Chapin Apartment K Stony Brook	26 Campus Dr Setauket- East
07:00 AM	Chapin Apartment K Stony Brook	Chapin Apartment L, Stony Brook,	Chapin Apartment K Stony Brook	26 Campus Dr Setauket- East
07:30 AM	Chapin Apartment K Stony Brook	Chapin Apartment L, Stony Brook,	Chapin Apartment K Stony Brook	26 Campus Dr Setauket- East
08:00 AM	26 Campus Dr Setauket- East Setauket	Chapin Apartment L, Stony Brook,	Chapin Apartment L Stony Brook	Chapin Apartment L Stony
08:30 AM	Chapin Apartment L Stony Brook	Chapin Apartment L, Stony Brook,	Chapin Apartment L Stony Brook	Chapin Apartment K Stony

Figure showing a schedule generated for a Stony Brook Graduate student.

3. SUGGEST FRIENDS

Individuals who are regularly at the same place at the same time usually have similar interests and lifestyles. It can be argued that being present at the same place for a short time may be a fluke and may not imply any similarity between individuals. This where the intelligent scoring function of Location Wiz comes into action. Only when two people are present at the same location (within 50 meters) for at least 10 minutes then this score gets increased. If two people are at the same location for more than 2 hours, then this score stops incrementing. This is done to prevent the score from bloating up for people live in the same apartment building.

The Location Wiz server runs the Friends.py script for each user. The script calculates the score for the user with all the other users. The way score is calculated for two users is as follows:

1. Read location data files of two users on the same date.
2. The locations of the two users are read from the files are stored in two maps. These maps have time intervals separated by 2 minutes as their keys and location (latitude and longitude) as the value.
3. If the haversine distance between the users is less than 50 meters then the score is incremented by 1. If the users are at the same location for 2 hours or more the scores stop incrementing.
4. Repeat the process for all the available dates and keep updating the scores.

The above steps are repeated for all users and then the top 4 for scores along with the usernames are written to a file and returned to the Location Wiz App when a user selects the Suggest Friends option.

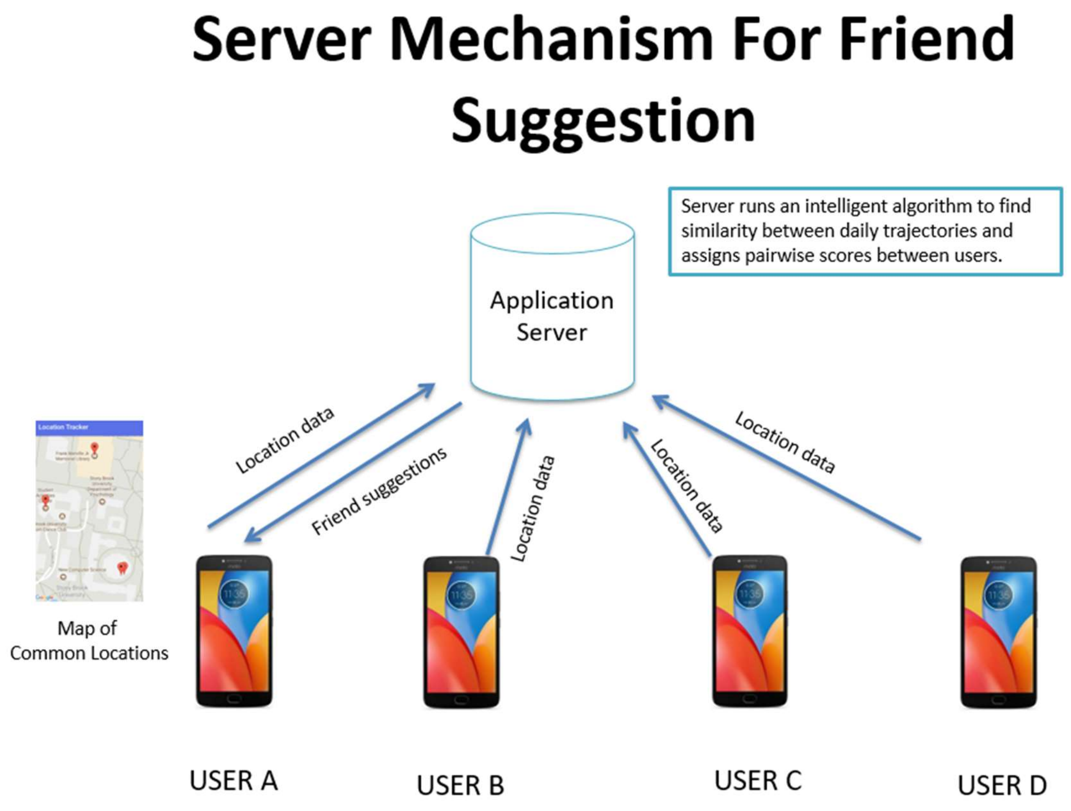

Server Mechanism For Friend Suggestion
Server runs an intelligent algorithm to find similarity between daily trajectories and assigns pairwise scores between users.
Application Server
Location data
Friend suggestions
Location data
Location data
Location data
Map of Common Locations
USER A
USER B
USER C
USER D

Figure showing Server Mechanism for Suggesting Friend

RESULTS:

When a user selects the Suggest Friends feature, they see a list of suggested friends along with the most common places between the users on a map:

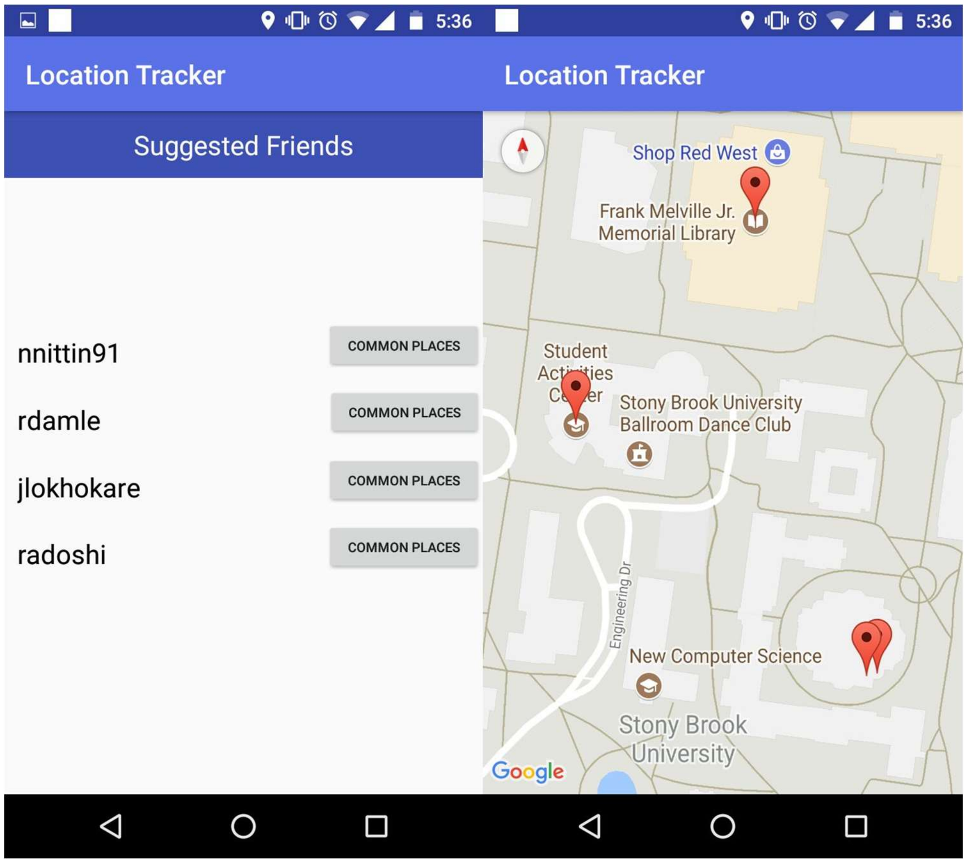


Figure showing suggested friends and the most common location between two users.

This data was for Stony Brook University students having two common lectures in Javitis Lecture Centre. The Location Wiz App was correctly able to determine this.

Limitations: Users having apartments close by default got a high score. To counter this, the Python script stops incrementing the score once two users are at the same location for more than two hours.

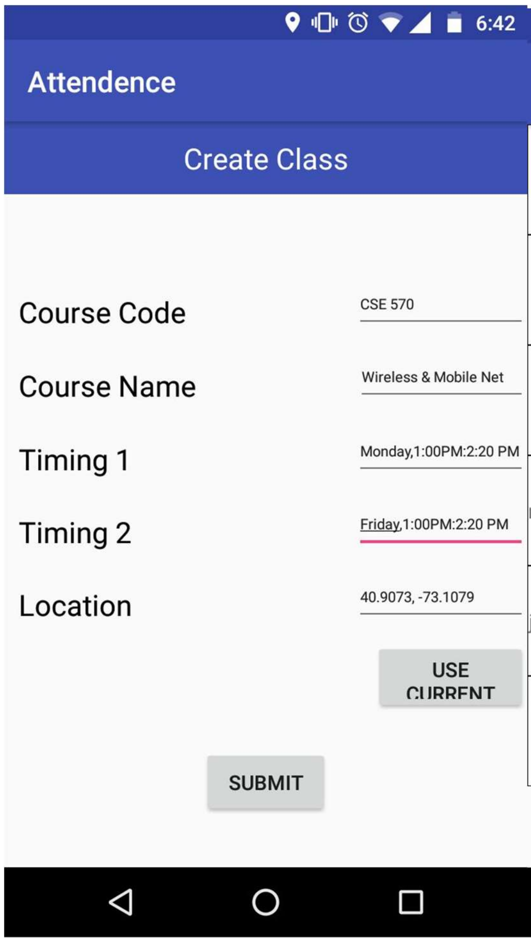
4. ATTENDANCE

Calling out attendance in every lecture is a gruesome task. With the data collected by the Location Wiz App, the location Wiz server can determine attendance of all the users in a lecture.
Once a Professor defines the location and timings of a lecture, the server can run the Attendance.py script to ascertain attendance without any action further action by a student or the Professor. The following steps are involved in determining attendance:

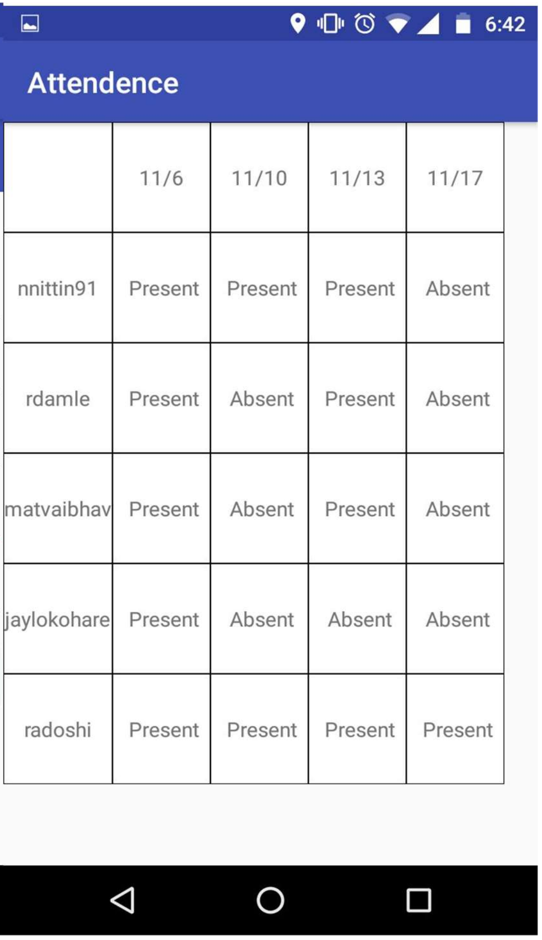
1. Take the location and timings of the class as input.
2. Iterate through the location data files of the mentioned days.
3. Check if the haversine distance between the location of the student at class start time and end time is within 25 m of the location of the lecture location.
4. A file with all the attendance data is created on the server and the sent to the attendance app on selecting the get attendance data option.

RESULTS:

A user has two options in the Attendance App. One is to create the class defining the location and timings of the class and the second is the option to get the attendance. These are illustrated below:



Course Code	CSE 570
Course Name	Wireless & Mobile Net
Timing 1	Monday,1:00PM:2:20 PM
Timing 2	Friday,1:00PM:2:20 PM
Location	40.9073, -73.1079



	11/6	11/10	11/13	11/17
nnittin91	Present	Present	Present	Absent
rdamle	Present	Absent	Present	Absent
matvaibhav	Present	Absent	Present	Absent
jaylokohare	Present	Absent	Absent	Absent
radoshi	Present	Present	Present	Present

Create class in Attendance app **Attendance data display screen**

Future possibilities for exploration

A possible extension to the idea of co-location used in attendance feature can be used to solve a problem of **Visitor Analytics**. Usually it is beneficial and required to estimate traction at important events such as exhibitions, career fairs, fests, hack-a-thons etc. to get an idea of the overall total footfall as well as generate the statistics for analysts who write cases for most popular stalls, least popular segments etc. Numbers are also required to be churned for individual persons to rank the locations visited at a particular event for later use by the user based on time spent. Suggestions for similar individuals at the event related statistics could be useful for the end users at a considerable large event.

To implement this idea, target location data needs to be input to the location wiz application just like the attendance target lecture data. All such data is to be added as part of an event added to the application by the organizers of the event.

Each target location data defines a particular stall of exhibition group location at the destination place of the event. The audience entering the destination place needs to use or avail the location wiz application with possible registration of personal data as if online registration of the user for the event. Then, the proximity query for each user to any particular stall can be used to generate the above numbers.

5. REFERENCES

1. https://developers.google.com/maps/documentation/android-api/
2. https://developer.android.com/training/location/index.html
3. https://developer.android.com/training/location/receive-location-updates.html
4. https://developers.google.com/android/reference/com/google/android/gms/maps/model/Polyline

6. ACKNOWLEDGEMENTS

This work would not have been possible without the consistent and kind support from Professor Jie Gao. We are indebted towards her valuable insights and prompt explanations. We would like to express our special thanks of gratitude to the TA of the course Hua Huang.