EDINBURGH NAPIER UNIVERSITY

HONOURS PROJECT REPORT(UG-4)

4th Year honours Project

BENG(Hons)Computing

WALK AND SHARING APP

2017

BY:SANYAM YADAV

SUPERVISOR: BEN PAECHTER

SECOND MARKER: NEIL URQUHART

CONTENTS

ABSTRACT

This project is about developing an app that allows people to find someone to walk with when they have a planned journey on foot. It is called a walk and sharing app.

Different route finding and matching algorithms have been presented with their drawbacks and advantages analysed.It presents how google maps directions API have been used to retrieve data such as route coordinates, time, distance and how to use this data is used for drawing the routes as it helps draw the most efficient routes.

It focuses on different matching algorithms used and their drawbacks and advantages.The evaluation of the matching algorithms and overall project has been presented too.Google Maps directions API indeed provide the most efficient routes and the matching algorithms used here show most adequate, correct results as we can decode the polyline. However, more matching algorithms can be used and they are described in further work.

For the database, MySQL rdbms has been used for designing schemas and retrieving and storing the user data. It was initially an android app but has been scaled down to a java server side application with all of the basic features implemented.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Ben Paechter for his immense support, expertise and guidance throughout the project. Without the support, I would have absolutely no clue on what to do.

I would also like to thank my parents, special friends and university staff for their unconditional support throughout my years as an undergraduate.

DECLARATION

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Sanyam Yadav

1.Project Description

1.1Description and aim of the project

This project is focussed on a walk sharing application which is related to subject areas like mobile computation and and software engineering. The principal goal of the project is to develop an app that allows people to find someone to walk with when they have a planned journey on foot.

Initial Specification of the project: The main aim of this project is to develop an android mobile phone app that allows users to specify a walking journey and a time window and then be matched with other users who share a significant portion of the route.

Revised Specification:

Initially the aim of the project was to create an android app.I changed from android studio to Java Application because :

- We cannot store all the routes in the users mobile phone, so it is better to store it in a central database which the app speaks to.
- Running the algorithms might be resource intensive on the mobile app, so the matching is better suited to be done on a server side program which the sense the matched result to the app.

Based on these reasons, I felt a server side application would be more suited for this project than a mobile application. In this project I created the user interface within the java application itself, but this could be used as a server side application that can be extended to be able to communicate with mobile applications.

Further more, the revised java application still implements all of the features that were intended in the mobile app. The features are as follows:

1.2Features implemented

Revised Specification features:

Features of the app(successfully implemented in this project):

1)Have a proper user interface that allows users to enter their journey details including start point, end point, time window, variable distance, date.

2)Show matched routes separately one by one to the user.

3)Show the shared journey of the two matched users.

4)Provide with the matched user's contact details e.g. phone number so that they can contact each other.

5)It should show different types of matching, like matching based on their start and end points, matching based on partial route matching.

6)It should be able to automatically zoom in when the user specifies the starting and ending locations.

7) Should show clearly the different routes, first users route in red colour, the second users route in blue colour and the matched routes in yellow colour,making it very distinct and clear for the user to see.

8) The time window should have a date, the hours and the minutes too for the correct matching to take place.

9)The app should be only accessible to users who have registered and logged in. For security concerns.

10)The start point and end point markers should be clearly marked with a dot or any marker symbol making it clear the start point, end point of the route.

11) Toggling: There is a button named "Toggle " on the app display page that lets you toggle between different routes that the user has been matched to.

It lets you see the matched routes one by one on the click of the button.It also displays in a pop up dialogue box the matched user's contact details.

Completion Criteria: A mobile phone app which helps two or more users to share a journey by matching their requirements and providing route information and maps.

Essential Skills: Good Programming skills, Interest in HCI and usability.

This project has no ethical issues involved in the project, no potential health and safety issues and no additional costs.

Resources required: Mobile phone and app development kit.

1.3 Background

There has been an increasing interest from academics and clinicians in harnessing a program or an application as a means of delivering and helping them be safe and improve their health. Despite the growing availability of a range of health-related and safety applications on the market, academic research on the development and evaluation of such applications is in the relative early stages. A few existing studies have explored the views of various populations on using mobile phones/gadgets for safety issues. Such apps has been proved to be very useful in today's world and hence is very much demanded.

In these modern times with abundant distractions it's harder than ever for people to stick with doing what they're supposed to be doing. During 'brainless' activities like walking, cycling, sports and driving people tend to daydream, make shopping lists, conduct telephone conversations, disregarding the activity at hand and becoming increasingly inattentive to their surroundings. It's that lack of alertness that can lead people into problematic situations. Trying to teach people to focus their attention to where it's supposed to be, helping to perceive dangerous situations in time to avoid them can be very beneficial.

It has been proved undoubtedly that if you are travelling by yourself and you know how to get home, using public transport is safer than walking. However, you should still use and try your best to keep yourself protected.

Using this app can help you with it.

There are several ride-sharing applications available ranging from "Uber" which is essentially private individuals acting as taxi drivers on an ad hoc basis to "hitch-a-ride" which describes itself as a social ride sharing and carpooling app. This makes sense because there are many journeys, particularly in urban areas, that many people want to make and sharing a car can decrease costs and pollution, whilst being faster than using public transport.

The need for "walk-sharing" may not be so immediate obvious but there are circumstances where people may feel more comfortable not to be going alone.

For example, at night in unfamiliar areas or simply as an incentive if the walking is voluntary exercise undertaken for health or fitness reasons.

The aim of this project will be to develop a mobile phone app that allows users to specify a walking journey and a time window, and then be match with other users who share a significant portion of the route.

2.Literature Review

2.1Other Apps(Past Work)

Other similar apps that are available and how they are different than our app:

1) LET'S WALK APP[9]

Lets Walk is an extremely accurate app to track your steps. Lets Walk uses Moves API to access your walking activity and track all your steps, distance, time and calories.

It also has a "Fitness & Health" feature.

It can help you improve your daily exercise and fitness goals. One is encouraged to setup goals and walk more frequently. Increase in exercise can lead to better health & lifestyle.

It also has a " Social Sharing" feature where one can invite their friends to share walking data, keep progress and weekly rankings. Easily post your walking activity in social networks like Facebook and Twitter.

However, in walk and sharing app you can specify your start and end points and choose if there is a matched user. You do not have to find a partner in advance to be able to walk and share a path with.

2)IFOOTPATH APP[4]

(How IFootPath App is different from my app)

iFootpath brings together the best of both worlds, old and new. Their walking guides include all the best features of traditional guide books with turn-by-turn directions, lovely photography and rich information about the history and the environment. They also provide all the benefits of modern technology. One can download the walks to the iFootpath iPhone or Android App and can follow their real-time progress around the live satellite route map, add your own comments and ratings and even create your own walking routes to share.

However, in walk and sharing app you do not need to download the walks. My app does not implement real-time progress around the satellite route map. It simply lets you specify your start and end points and lets you choose if there is a matched user.

3)MAP MY WALK [5]

MapMyWalk is a fitness tracking application that enables you to use the built-in GPS of your mobile device to track all of your fitness activities. Record your workout details, including duration, distance, pace, speed, elevation, calories burned, and route traveled on an interactive map. You can even effortlessly save and upload your workout data to MapMyWalk where you can view your route workout data, and comprehensive workout history.

However, walk and sharing app does not incorporate the recording of workout details and does not let you save and upload the workout data.It is more safety purpose than fitness purpose.

2.2 Different algorithms (Route finding and matching algorithms)

This section describes the already existing different route finding and route matching algorithms.

Route Matching

1.Practical Polyline Matching for GPS Data (Route Matching)

Here, two routes are being matched by taking coordinates of the routes (from the polyline) and matching them based on their proximity.This is called a partial route matching because no matter if the two routes are not completely matching to each other, even if the parts of the routes are matching it is displayed.

Rasterisation is the method or technique that has been used here. The proposed algorithm rsaterzies the routes into two sequences of grid fields. It is pretty simple, if two grid fields are close, there is a match. Furthermore, single character is used to match multiple users.

Then, these gird fields are used as strings of characters. So, string matching algorithm can be used. How this is done is interesting as the girls fields are seen as strings of characters and then matched accordingly. This method also has a cost function which is modified based on the length of route(not edit distance). The complexity is O(mn), m and n being the number of grid fields in the rasterized routes.

While these are dependent on the rasterisation, Lateral grid size is bounded by the average line segment length, then the complexity is bounded by O(st), S and T are the number of line segments in the routes.This algorithm has also been tested on a dataset made by GPS-tracking runners and cyclists and is found to be both feasible and capable of consistently producing the desired results.[12]

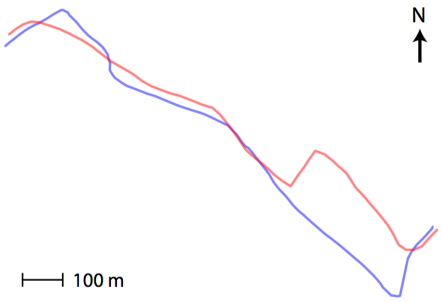
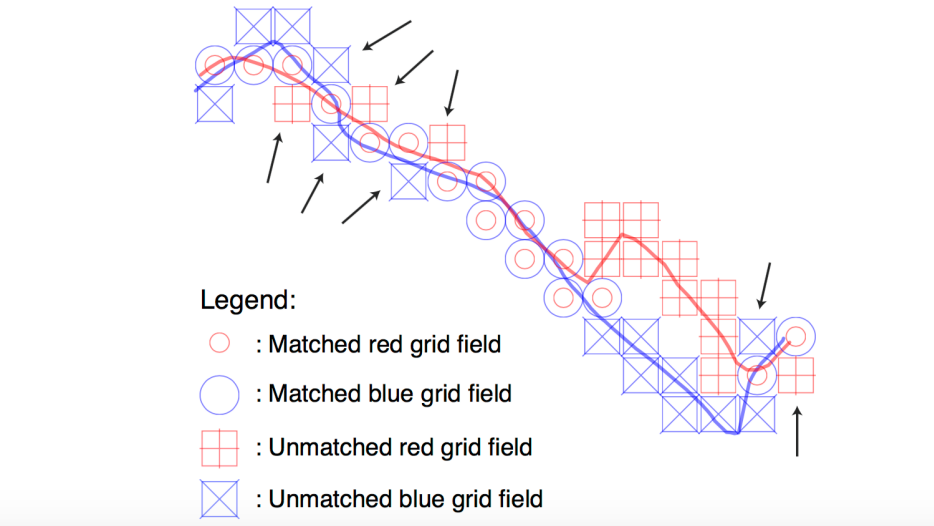


Figure 1 showing two routes.



The two example routes from figure 1 have been rasterized and matched with the modified algorithm developed in this section. The shape and color of grid fields are used to show which route they belong to and whether or not they have been matched as part of the optimal solution. The arrows point to undesired artifacts.

2.Matching Algorithm utilising road segment connectivity along with positions, speed and heading.

Here, the matching algorithm utilises road segment connectivity along with positions, speed and heading.
Here, they are considering the wright based shortest path. The vehicle trajectory aided map-matching (stMM) algorithm that enhances the map- matching of low frequency positioning data on a road map.
The well-known A$*$ search algorithm is employed to derive the shortest path between two points while taking into account both link connectivity and turn restrictions at junctions.

In this algorithm ,they take 2 additional weights related to shortest path and vehicle trajectory. For shortest path, the distance alongside the shortest path is taken onto account and the distance along vehicle trajectory, while the second one is about heading difference of the vehicle trajectory.

A high-accuracy integrated navigation system (a high-grade inertial navigation system and a carrier-phase GPS receiver) is used to measure the accuracy of the developed algorithm. The results suggest that the algorithm identifies 98.9% of the links correctly for every 30 s GPS data. Omitting the information from the shortest path and vehicle trajectory, the accuracy of the algorithm reduces to about 73% in terms of correct link identification. The algorithm can process on average 50 positioning fixes per second making it suitable for real-time ITS applications and services.[13]

Route Finding

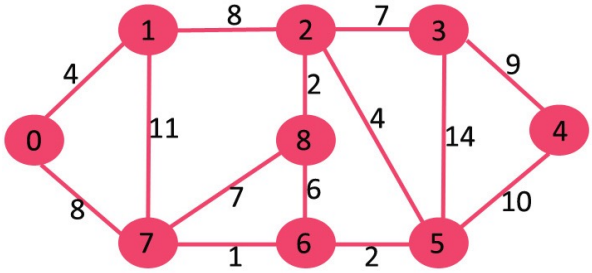
1. Dijkstra's algorithm: Finds the shortest path which can be used for route matching.

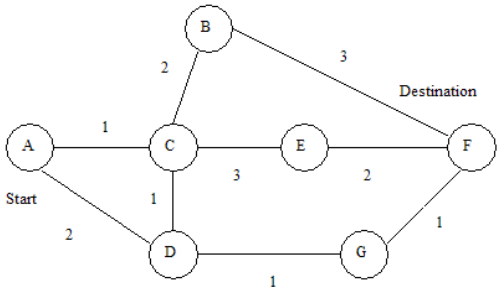
Most applications and people use Dijkstra's algorithm for this problem.(for finding the shortest paths between nodes in a graph)

It was invented by Dijkstra around 50 years ago and it calculates the best way of getting from point a to point b in a directed network.

- It is like each street is an edge in the network and where each edge is associated with a "cost" i.e. the time it takes to travel down the road, or the average speed.

The algorithm exists in many variants; Dijkstra's original variant found the shortest path between two nodes, but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a shortest-path tree.For a given source node in the graph, the algorithm finds the shortest path between that node and every other.In Artificial intelligence, Dijkstra's algorithm or a variant of it is known as uninformed search and formulated as an instance of the more general idea of best first search.[10]





In this algorithm,

we have an initial node set at "0". Then we have a distance y that computes the distance between this "0" node to another node.

We check the distance from this node to all of its neighbouring nodes. Supposed there are 10 neighbouring nodes to this particular node. We compute the distance y from "0" to all these neighbouring nodes. We take the smaller distance and adds that to our list. The other nodes that are not selected go into the Visited array/ removed from the unvisited array and is not visited again.Then from this selected node, we check its neighbouring nodes and selects the shortest distance one.

f the current node *A* is marked with a distance of 6, and the edge connecting it with a neighbour *B* has length 2, then the distance to *B* (through *A*) will be 6 + 2 = 8. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, keep the current value.Similarly we keep going,until our destination has reached and we have a route.

Combined Route Finding and Matching Algorithm.

1.As to this academic paper (Algorithms for Matching and Predicting Trajectories)[15]

Performance Metrics:

Here,the matching algorithm is based on Dijkstra-like graph traversals basically.

The runtime is determined by number of Dijkstra's operations(polls from the priority queue). Furthermore, graph representation, implementation language and the hardwares on which algorithms are run also influence our given running time.For easier platform and implementation-independent comparison we mostly state the running times of our algorithms in terms of Dijkstra operations.

How their approach is different :

They ensure the efficiency of their map-matching algorithm. They do this by a general technique to speed-up shortest path computations. Out of the many speeding up schemes, they choose the one that is in their implementation contraction hierarchies due to its simplicity and efficiency.

Here, there is a road network as G=(V,E,I) which connects it to a new road network G'=(V,E',I') by adding directed edges to G based on an ordering ω = v1,v2,...,v n of the vertices of G. The shortest path is the network is preserved.

Shortest path queries in G can then be answered more efficiently based on the fact that in G ′ there exists a shortest path for any pair of vertices (s, t) such that the indices of the vertices on this shortest path are monotonously increasing first and decreasing afterwards with respect to the ordering ω. Therefore, to identify a shortest path from s to t, a bidirectional Dijkstra algorithm can be adopted. More precisely, one can use one Dijkstra run (forward) starting in s and a second Dijkstra run (backward) starting in t. The forward run considers only outgoing edges which lead to a vertex with a larger index while the backward run considers only incoming edges which come from a vertex with larger index. The two runs have to meet in at least one vertex of the shortest path, thus computing a shortest path from s to t with a drastically reduced search space. Thus, one-to-one shortest path queries can be answered at a fraction (about 1/1000th) of the time of a full Dijkstra with the help of the augmented G′ = (V, E′, I′). [15]

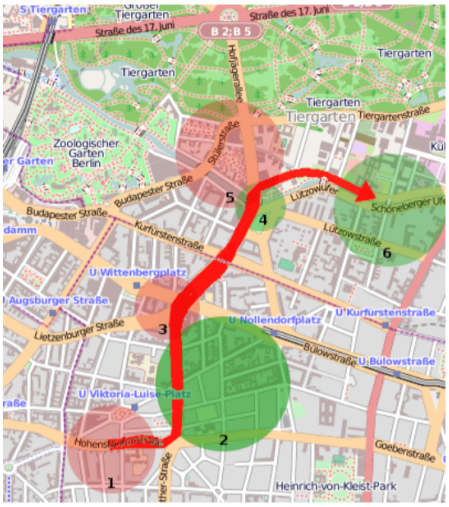


Figure 1: The Map Matching Problem: We are given a sequence of (here: 6) imprecise location measurements and want to reconstruct the original (red) path

Figure 2: The Path Prediction Problem: We are given the route a mobile user has travelled up to now and want to predict its route in the near future.

2.3 My approach towards the problem

(Route finding and matching algorithm used in this project)

The above 4 algorithms described are candidates for use within the system. They are all very suitable for finding/drawing the routes. Here is what has been implemented in this project:

Route finding

I have used Google maps directions APIs and used this to get coordinates for a route. To draw the routes using the coordinates points that we get from this API , they have been drawn by using drawPolygon() function that is provided by Open street maps.

Advantages of this approach

Google maps directions API not only search for directions for several modes of transportation, including transit, driving, walking or cycling, it also returns multi part directions using a series of waypoints.

It lets you specify origins, destinations and waypoints as text strings or as latitudes/ longitudes coordinates or as place IDs.

According to the research[reference to the page], it returns the most efficient routes when calculating directions. It takes distance, number of turns and many more when deciding which route is the most efficient.

The roads api takes up to 1000 GPS points collected along a route, and returns similar set of data, with the points snapped to the most likely roads the vehicle is traveling along.The points can be interpolated, resulting in a path that follows geometry of the road.

Drawbacks

One limitation of this approach is that this web service APIs include 2500 requests per day only, calculated as the sum of the client-side requests(made via the Google Maps JavaScript API Services) and the server side requests. [2]

However, for creating and testing this application, these many requests are not required.

We are dependent on the google maps directions api performance.

Route matching

I have three route matching algorithms. I have implemented the two of them and the third one in described in further work section below.

1) Matching based on proximity of start points and end points of two users.

2) Matching based on overlapping routes(The start points and end points of the two users may not be close but they share significant portion of their routes together)

3) Matching if the two users are travelling on parallel roads(Described in further work)

Advantages:

Since we are utilising google maps data for finding and matching routes, these algorithms are easier and quicker to fulfil the app's requirements. These are custom made to utilise google map features for route matching to meet the specifications and requirements of this project.

Drawbacks:

We are limited to matching based on information provided by google maps directions api. However, it provides distance, time, polylines and other data which is enough for these algorithms to work.

My initial approach towards the problem

CODING

- Get the basics done first. My first step was to have a map interface and get the routes working from one point to another.

-Get two routes working.

-Get two routes showing on the map only if their starting points and ending points are matching.

-Then get all the matches gradually.

-For different matching, different buttons are there.

-Toggle button to let you toggle between different matches while showing pop up dialog box with their contact details.

DESIGNING

-Eclipse IDE graphical user interface.

-Different buttons for showing different algorithms. On click methods seem more sensible.

3.Principles

3.1Software Engineering Principles

Things to keep in mind:

The system should be able to meet the known requirements(functional and non functional), it should be maintainable for the future work to progress (adapted to meet future requirements). It should be straightforward to explain to future coders or implementors and there should be appropriate use of existing and right technology.

Different design choices will make different future changes easy- a good design makes the most likely ones easiest.

The design of my application includes three pages, from login,registering to the main application page. Based on what are the responsibilities of each component, and what information is exchanged in what order, I have chosen the textfields, buttons and labels.

1)According to SE principles, High cohesion is desirable.

Cohesion is a measure of the strength of the relationship between pieces of functionality within a component.

I believe there is high cohesion in my code:

As these different functions are all interlinked with each other and one works because of other, I think there is high cohesion in my code. The explanation is as follows:

There is a method/function called distance(which calculates the distance between between two points based on their longitude, latitudes.)

There is a function called third which takes two coordinate points,

and if overlapping_paths is true and time of common meeting between two points is true for both the users, only then it draws the route using draw_route function.

The get_coordinates function gets the coordinates between two address points.

The partial method gives a list of coordinates that are the common coordinates between two lists of coordinates.

The function time_common_meeting checks whether the two users are meeting at a common time. It takes the users entered time and calculate the time it will for him to reach a certain point and add that to the initial time, hence giving the time he/she reaches at that time.

The function Converted_Distance returns the distance between two points and if that distance is less than 5 kms range, it returns true.

At certain points, The *draw_route* method can only run if this *converted_distance* function is true. That means, that the two users can only match if they are within certain distance. So, there is cohesion here.

Similarly, show_shared_route function draws the shared route but it implements *draw_route* and *get_coordinates* functions with them.

2) This also follows design principles called decomposition,modularisation:

Dividing a large system into smaller components with distinct responsibilities and well-defined interfaces.

As the code has been decomposed into smaller parts, and different functions have been coded, it follows the decomposition and modularisation principle of SE.

3.2 Neilson's 10 Heuristics(How this app adheres to these principles) [14]

1. **Visibility of system status**

The system always keeps users informed about what is going on, in our app case, it tells them about the active users, whether there are any active users or whether not. It is within reasonable time as it does not take neither too long nor too short.

2. Match between system and the real world

This heuristic talks about how the system should and does speak the users' language, with given labels (words, phrases and concepts familiar to the user) rather than system-oriented terms. My app does implement this feature.

3. User control and freedom

This heuristic says to support and undo.

It is about not choosing the system functions by mistake .It is about user being able to exit immediately without going through an extended dialogue.In my app, there is not a undo/ redo button. However, they can very easily leave the app by closing it.

In my further work, I have talked about how there could be a new table that would keep deleting the match in the active table once the user has found their matched user.

4. Consistency and standards

As consistency and standards are about not having the user wonder whether different words, situations and actions mean something else, in my report I have tried to make it very clear, what the buttons are meant to be doing by labelling them with appropriate names. I have also mentioned what inputs can be entered in the text fields. English has been used consistently.

5.Error Prevention

I believe here my app lacks this feature. There could be an extra feature where if there is an error, it would not be shown to the user directly or should show them a pop up dialog box stating the error.

However, according to this heuristic error prevention is better than good error message, this comes from a careful design which prevents a problem occurring in the first place. There can be more careful testing and debugging done here, which shall prevent the errors. In further work, the app should either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

6.Recognition rather than recall

According to this heuristic, one should be able to minimise the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another.

In the app, this is very well implemented. The app continuously takes you from one page to another based on your action or button clicked. The app is pretty straightforward but more instructions can be added to specify in more detail. Instructions for use of the system are made visible or easily retrievable whenever appropriate.

7.Flexibility and efficiency of use

According to this heuristic, the system should be able to cater to both inexperienced and experienced users. This app has been made so clear that some young kids or students should be able to understand it too. This allows users to tailor frequent actions.

8.Aesthetic and minimalist design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

9.Help users recognise, diagnose, and recover from errors

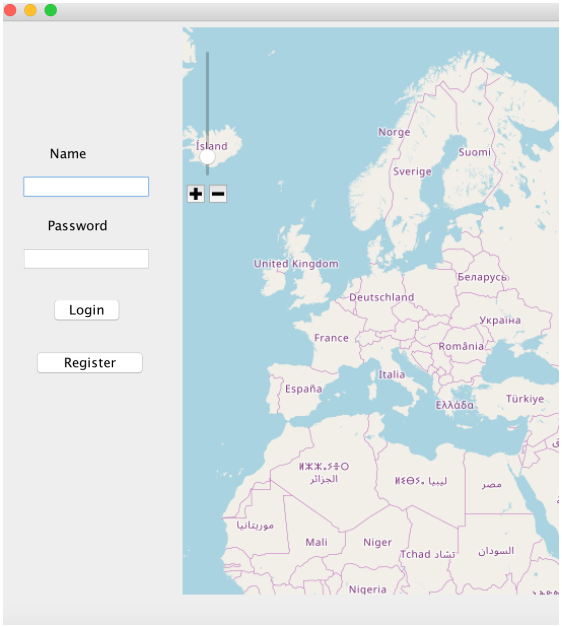
Error messages are indeed expressed in a plan language (english) and no code has been used. However, the solution to these problems have not been displayed yet. For further work, the solutions to the error problem should also be displayed.

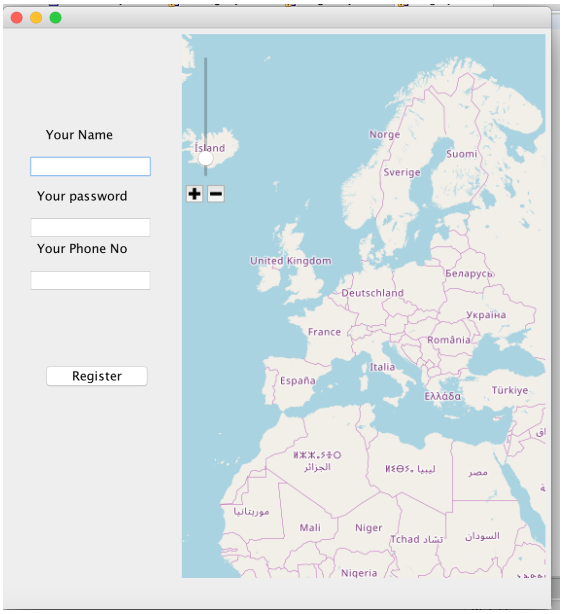
10. Help and documentation

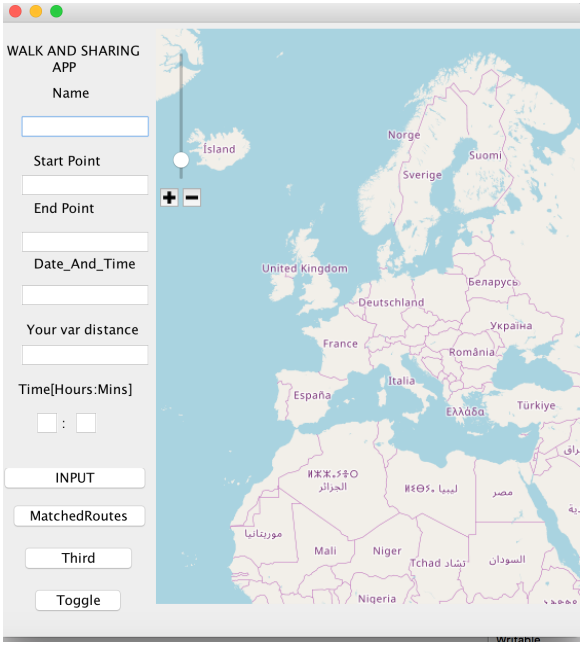
According to this heuristic, one should be able to use the system without documentation but it may be necessary to provide with some help and documentation.

This report does that. However, a simple google search might be able to solve the problem too.

3.3 User Interface Designs







4.INITIAL DECISIONS:

4.1 Using Android Studio

Initial work was based on android studio.

In android studio, I begun by making a simple app.I created my own custom AVD, using AVD manager. After cloning the device, i added some activity states to my project.

Running the changing states, design the user interface, create an interface with java, adding properties to widgets, and to more widgets. For viewing, converting DIP to Pixels. Then after getting a proper grid layout, and proper handling different events, the app was running.

I designed the user interface,created an interface with Java, adding properties to widgets, grid layouts,event handling, event listener and call back methods.

By selecting google maps activity, in the 'Add an activity on mobile' dialog. Android studio starts the Gradle and builds the project. When the build is finished, android studio opens the googe_maps_api.xml and the MapsActivity.java files in the editor.

After getting and setting a Google Maps API key to access the servers, the key received is the API key with restriction for Android Apps. The map is then shown.

The map was then implemented in the android studio app. There were multiple event listeners but getting the code running was difficult.

Why I did not choose to work further with Android Studio and used Eclipse IDE.

The most significant difference is the loading time of Android Studio.

The app simulator would take a long time to run and after a lot of research, i realised, this type of app needs a lot of users, data and a lot of routes to be processes so it is more suited to be done on servers. We cannot store all the routes in the users mobile phone, so it is better to store it in a central database which the app is connected to.Running the algorithms might be resource intensive on the mobile app, so the matching is better suited to be done on a server side program which the sense the matched result to the app.

4.2 Why I chose Eclipse IDE?

About Eclipse and its advantages

Eclipse offers desktop and cloud based development environments. Features include GitHub (version control repository).Eclipse has a large plugin library. There are excellent plugins available so make sure you are choosing reliable third-party plugins.

It is an open source, free and cross-platform.It is also compatible with Windows, Linux, OS X and Solaris Operating Systems.

Eclipse also supports for Java 8. The Eclipse IDE is optimised to leverage all of the great new constructs of the Java 8 language. So there is a lot of support for Java 8.Eclipse offer rapid User interface design with the drop and drag functionality. Eclipse IDE gives you a wide variety of options like Jbutton, JPanel, JTextField, JCheckBox and many more.You can choose any tools from the palette, drag and drop them on the right side and easily then make your graphical interface.

Eclipse IDE lets you do this from the user's point of view. You can see the interface from the user's perspective, how it would look when the application is actually running.

Because of the size of the project, the nature of the project and whether android development is a priority, I chose eclipse IDE.I have previous experience with Eclipse IDE and I am familiar with it too.

4.3Why I chose Java?

Java is a server side programming language that known for enterprise-level applications.It is also a core language for mobile application development.

Java is one of the most popular languages I the world. I have had a lot of experience in it and I am good at it. Java is estimated to be running on billions of devices all around the world.

For creating applications such as walk and sharing app, it is very prominent. It is an excellent language for developing enterprise applications in virtually any industry, including financial services, healthcare and manufacturing.

It also provides you with some great tools and libraries. There is actually a large active community that works together to grow and enhance this powerful programming language.

It is like one of the most common language in Computer science field, like we have English as our spoken language. This ubiquity can be extremely attractive to some companies. With java, it is quite easy to maintain code for years to come.

Encapsulation is one of the four fundamental OOP concepts. The other three are inheritance, polymorphism and abstraction.[8]

E- It is like wrapping the data and code acting on the data (methods) together as a single unit. It is also called data hiding. The variables of the class will be hidden from other classes and can also be accessed through their current class. (hence called) data hiding.

Abstraction is the quality of dealing with ideas rather than events. It is like hiding the implementation details from the user, only the functionality will be provided by the user.

Java is secure because it provides the encapsulation and data abstraction through which we can do data hiding. That is how we secure the data. Even an applet provides security. [7]

Why open Street Maps

It is a great source of using maps as it provides a free readable map of the world. The creation and growth of the OSM has been motivated by restrictions on use or its availability of map information across much of the world, and the advent of inexpensible portable satellite navigation devices.[17]When you start creating open map data,you get the following provided on the maps:

Points :used to mark places

Lines:Used to mark road and paths

Areas:used to mark outlines of buildings, fields, or any other large area).

These are very useful and important for this project as we need main routing and directions data. We need points, lines and areas to draw the routes and do matching.

Overview :

I am using open street maps in my application.

Once you have imported the map into your eclipse project and run the project ,you will see the map. To the right of the map are the map controls. You can zoom in, zoom out and view your current location on the map, and view the current location on the map(which has been to some coordinate).

(I have implemented and used mapViewer.ZoomIn and other functions that are present in this package) for viewing and zooming).

There are layers, map key share the map features also present.

There are openStreetMaps.osm file to download and import it in your eclipse project.Once it is set up in the Java Eclipse IDE, it is easier to create map data by adding points, lines and areas.

5.PROGRAM IMPLEMENTATION

5.1Initial matching strategy based on CSV files

Initial matching was done based on CSV files, taking the data from the CSV file, and matching the user's entered data to the CSV files data.

Using the CSV reader and importing

```
import au.com.bytecode.opencsv.CSVReader;
```

the parsing through the file was done.In the matching it would check the first 3 characters of the postcode.

For example 'AB101AA' and 'AB101AH' are the two postcodes which have the same three characters so they are within close perimeter.

Since they are within close premises, it is a potential match for two users if they have these starting points.

The two postcodes namely 'EH89JR' and 'EH74DG' wont be potential matches.

Drawbacks

In the CSV files, we only had Aberdeen's and Edinburgh's postcode.

This was a pretty big limitation as the app could not be used in areas other than Edinburgh and Aberdeen. However, the google maps API's lets you get the world's postcodes all together.

It also adds space in the project and needs to be imported, which is complicated, time consuming and needless.

5.2 New Implemented matching strategy based on google maps API

How to use API

The google maps directions API is a service that calculates directions between locations using an HTTP request.

It is very useful for someone who wants to compute direction data within maps provided by one of the Google Maps APIs as in this application.

Uncovering maps APIs:

Its is a big revelation,provides step by step direction from point A to point B. Being a web service, it is protected by an API key and that should not be consumed directly in the mobile app.The recommended procedure includes proxying it via our server as shown here.

It can give you walking ,cycling and driving directions as should expect. But it can also give great public transport direction.

We have used mode=walking in the http requests.

When a user is using this service and wants to go to a destination, it is best to show them the path on a map. Polylines are a great way to show path. But rather than have polyline appear, it is nicer to animate it in a place.

It provides clear distinction about the direction of travel.So, showing users where to go and how to get there,but once they are at their destination, our apps work is not done.There is interactivity in place. There is also an option to add a street view panorama into the app by including it in the layout (giving it also an initial location like so). But this an be done in future work.[16]

As it is very important to maintain user details and their time traveled in the journey, the API returns the most efficient data(routes) when calculating directions.

It outputs the travel time between two points and as it is an important factor to be taken into account.While laying out the route, it takes several factors like how many steps travelled, how may turns taken, time to give the most optimal route on the map.

This is the format of the google Maps Directions API

https://maps.googleapis.com/maps/api/directions/**outputFormat**?**parameters**

Output The output is returned in a JSON format.

Use of https instead of http for security concerns.

As *HTTP* stands for Hyper Text Transfer Protocol. It is a protocol that allows communication between different systems (transferring a data from a web server to a browser to view web pages)

But http data is not encrypted,which can be intercepted by other people and they can get access to all the data being passed between the two systems.

HTTPS , S however stands for Secure. It creates a secure encrypted connection between the web server and the browser.If there is sensitive data, that is being passed across the connection, such as e-commerce sites that accept online payments or login areas that require users to enter their credentials.

In the case of our app, the user writes their current location or their seat location and their destination, and this sensitive data requires it to be in https not http.

There are optional parameters like mode(we are using walking), way points, alternatives, avoid(tolls, highways, ferries,indoor), language, units, region, arrival_time, departure_time, traffic_model,transit_model, transit_routing_preference, that can be added to our link.If we give it the addresses in the origin and destination points, that is correct too. If we give it address, the direction service converts the address to a string and convert it to a latitude/longitude co-ordinate to calculus directions.

It does not accept spaces between latitude and longitude values. There should not be a space “ ” character in between latitudes and longitudes values.

Alternatives:

Place_id only works if the request includes an API key or a google maps APIs Premium Plan client ID.

Alternatively, you can also enter an encoded set of coordinates using the Encoded Polyline Algorithm.Its useful if the user has a large number of waypoints, because the URL is significantly shorter when using an encoded polyline. [16]For example,

The following URL initiates a Directions request for a route between Boston, MA and Concord, MA with stopovers in Charlestown and Lexington, in that order:

*https://maps.googleapis.com/maps/api/directions/json?origin=Boston,MA&destination=Concord,MA&waypoints=Charlestown,MA|Lexington,MA&key=**YOUR_API_KEY***

WAYPOINTS

For programming point of view:

For each waypoint in the request, the directions response includes an additional entry in the legs array to provide the corresponding details for that leg of the journey.

If you'd like to influence the route using waypoints without adding a stopover, prefix the waypoint with via:.
Waypoints prefixed with via: will not add an entry to the legs array, but will instead route the journey through the provided waypoint.[16]
The following URL modifies the previous request such that the journey is routed through Lexington without stopping:

*https://maps.googleapis.com/maps/api/directions/json?origin=Boston,MA&destination=Concord,MA&waypoints=Charlestown,MA|via:Lexington,MA&key=**YOUR_API_KEY***

The google maps directions API gives data in the following way :

```
{
  "status": "OK",
  "geocoded_waypoints" : [
     {
        "geocoder_status" : "OK",
        "place_id" : "ChIJ7cv00DwsDogRAMDACa2m4K8",
        "types" : [ "locality", "political" ]
     },
     {
        "geocoder_status" : "OK",
        "place_id" : "ChIJ69Pk6jdlyIcRDqM1KDY3Fpg",
        "types" : [ "locality", "political" ]
     },
     {
        "geocoder_status" : "OK",
        "place_id" : "ChIJgdL4flSKrYcRnTpP0XQSojM",
        "types" : [ "locality", "political" ]
     },
     {
        "geocoder_status" : "OK",
        "place_id" : "ChIJE9on3F3HwoAR9AhGJW_fL-I",
        "types" : [ "locality", "political" ]
     }
  ],
 "routes": [ {
   "summary": "I-40 W",
   "legs": [ {
     "steps": [ {
       "travel_mode": "DRIVING",
       "start_location": {
         "lat": 41.8507300,
         "lng": -87.6512600
       },
       "end_location": {
         "lat": 41.8525800,
         "lng": -87.6514100
       },
       "polyline": {
         "points": "a~l~Fjk~uOwHJy@P"
       },
       "duration": {
         "value": 19,
         "text": "1 min"
       },
       "html_instructions": "Head \u003cb\u003enorth\u003c/b\u003e on \u003cb\u003eS
       "distance": {
         "value": 207,
         "text": "0.1 mi"
       }
    ... additional legs of this route
     "duration": {
       "value": 74384,
       "text": "20 hours 40 mins"
     },
     "distance": {
       "value": 2137146,
       "text": "1,328 mi"
     },
     "start_location": {
       "lat": 35.4675602,
       "lng": -97.5164276
     },
     "end_location": {
       "lat": 34.0522342,
       "lng": -118.2436849
     },
     "start_address": "Oklahoma City, OK, USA",
     "end_address": "Los Angeles, CA, USA"
   } ],
   "copyrights": "Map data ©2010 Google, Sanborn",
   "overview_polyline": {
     "points": "a~l~Fjk~uOnzh@vlbBtc~@tsE`vnApw{A`dw@~w\\|tNtqf@l{Yd_Fblh@rxo@b}@xxS
   },
   "warnings": [ ],
   "waypoint_order": [ 0, 1 ],
   "bounds": {
```

```
    "bounds": {
      "southwest": {
        "lat": 34.0523600,
        "lng": -118.2435600
      },
      "northeast": {
        "lat": 41.8781100,
        "lng": -87.6297900
      }
    }
  } ]
}
```


5.2.2 JSON Output Format

It stand for JavaScript Object Notation.It is used in android, apis and web technologies.

It is a format, which helps in transferring data from client to server and from server to client. Initially, HTML was used to design websites, then came css. Now, the data which we get from server is dynamic, so to send requests to and from application to the server, in response the server sends some data, this data is written in JSON format.

It is a textual , readable data which is coming from server. Therefore, to retrieve the route and travel information from the web for this project, we can use google maps directions API which gives us JSON format output and we can easily parse it to fulfil our requirements.

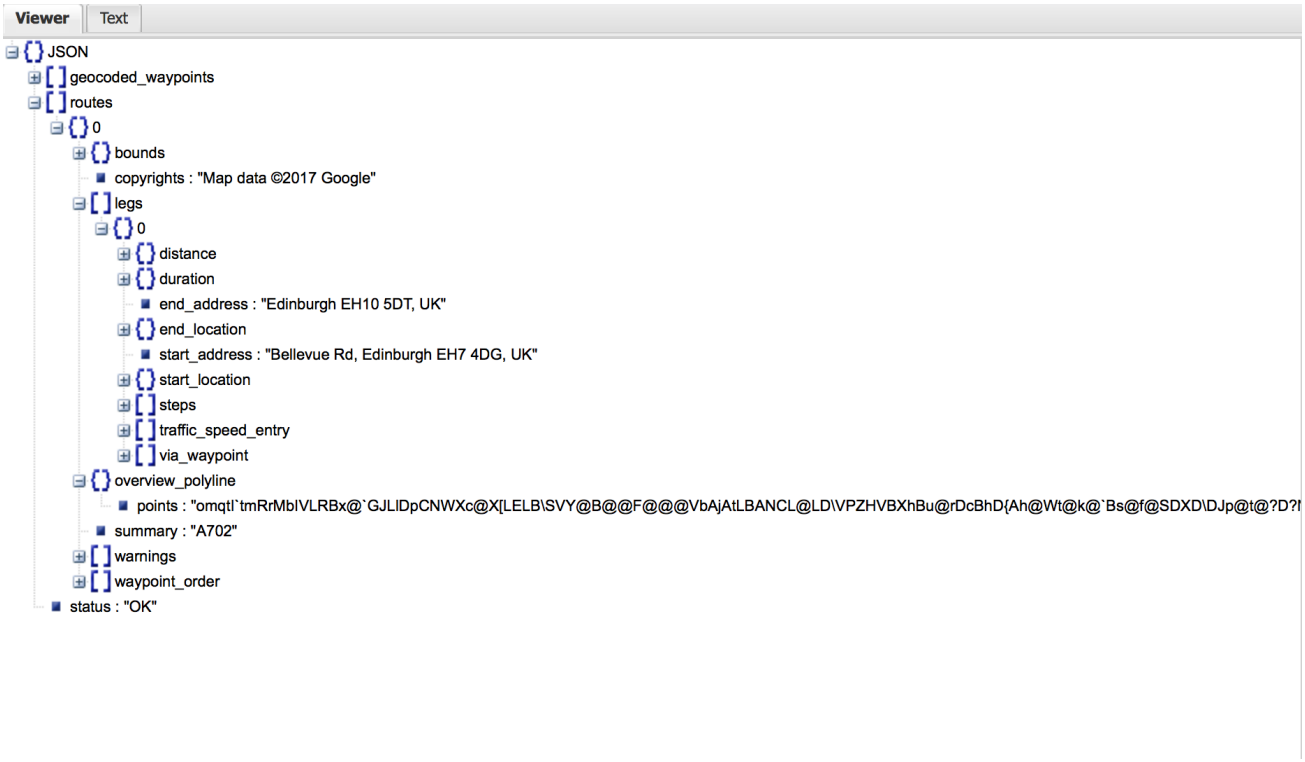
[1] is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format used for asynchronous browser–server communication, including as a replacement for XML in some AJAX-style systems.[1]

5.2.3PARSING JSON FORMAT OUTPUT

After giving a http request for the API, we get an output in JSON format. To get the data which will be useful for the app, we need to parse this output.

Copy the complete output and paste in any JSON viewer of our choice. I used this: jsonviewer.stack.hu/ and go to the viewer tab which gives us a tree like format, with parent nodes and child nodes that contain the data .

Parsing it, requires following steps:



STEPS

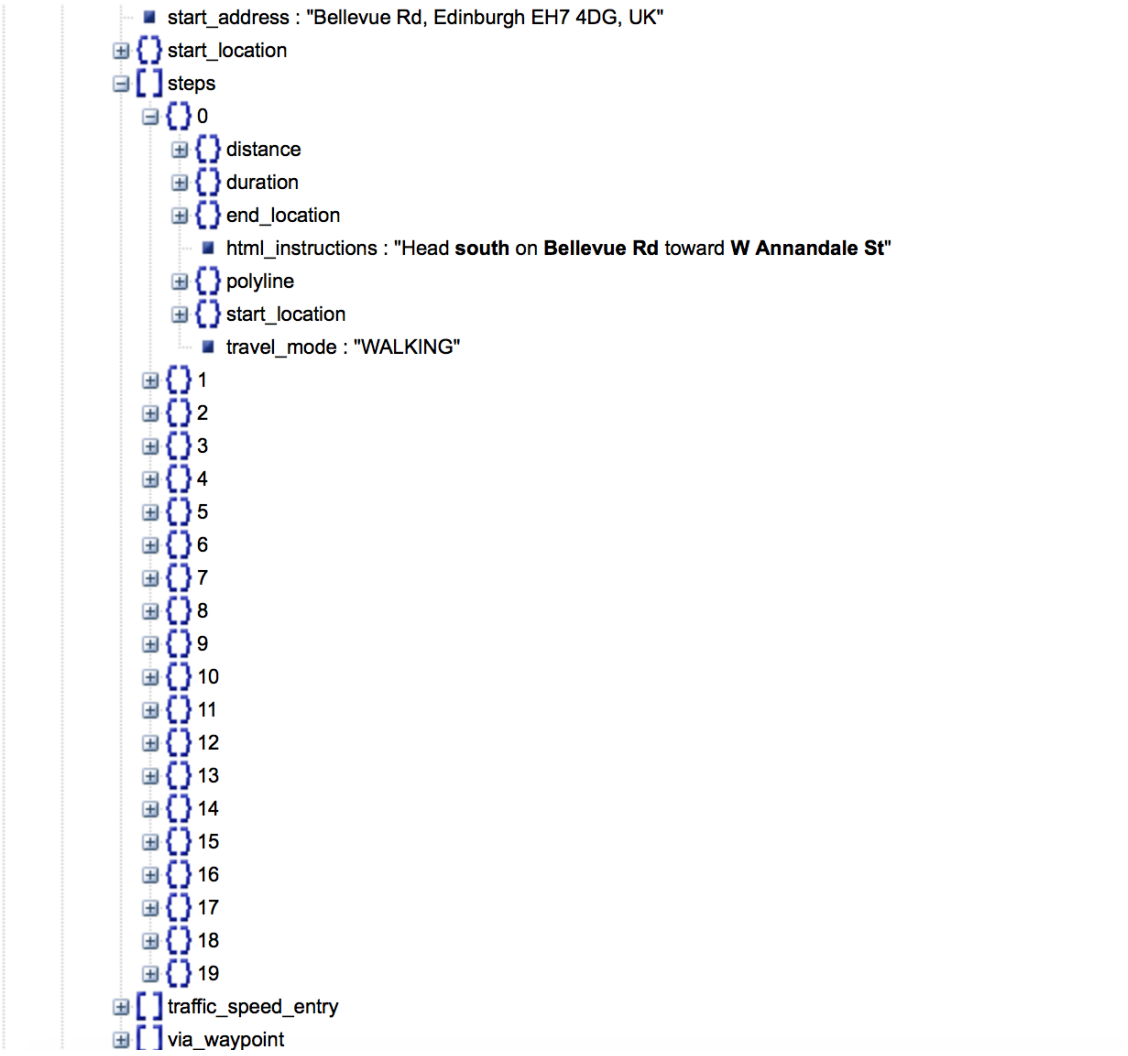
1) From JSON Viewer, you get routes array first.

2) It returns a JSON Object 0.

3) From that object, you get legs which has another JSON object that takes you to steps.

In the given route it is assumed that they have each been subdivided into a sequence of intervals.These intervals could be called Steps.

Steps gives you a list of n number of steps, where n demonstrates the number of steps there are between two points. Each step has distance, duration,end_location, polyline,start_location.



4) To get route from one point to another, in order to draw a proper route ,we need as many coordinates as possible so those coordinates together directs to a proper well-routed route.

5) Through a general api, only a limited number of points are given in a output.

Initially, i was getting very less points in between two points giving very inaccurate route. the route would pass through the buildings, parks etc

For example: distance of two miles, 10 coordinate points were not enough and resulted in my route going through the buildings, parks etc.

To get enough amount of coordinates for proper routes, I used another JSON Object called Polyline, which has a very different format.[3]

Polylines in Google Maps are formed as a set of latitude/longitude pairs. In addition, for each vertex (location) in an encoded polyline, a level can be specified indicating that the location should appear on that level and any level higher (i.e. any decrease in zoom). If a location does not appear on a given level, then the line will go from the last visible location to the next visible location. I kept in mind that the first and last locations must be Level 3 points, otherwise the polyline won't display on all levels.

5.2.4 Decoding the polyline and drawing the routes

A geographic coordinate, or geo-coordinate, consists of a latitude and longitude, uniquely describing a position on Earth. A route then is a sequence of such co- ordinates, R = (c0,c1,...,cs). These coordinates describe a polygonal curve, or polyline, consisting of s line segments, ((c0,c1),(c1,c2),…,(cs−1,cs)).**[12]**

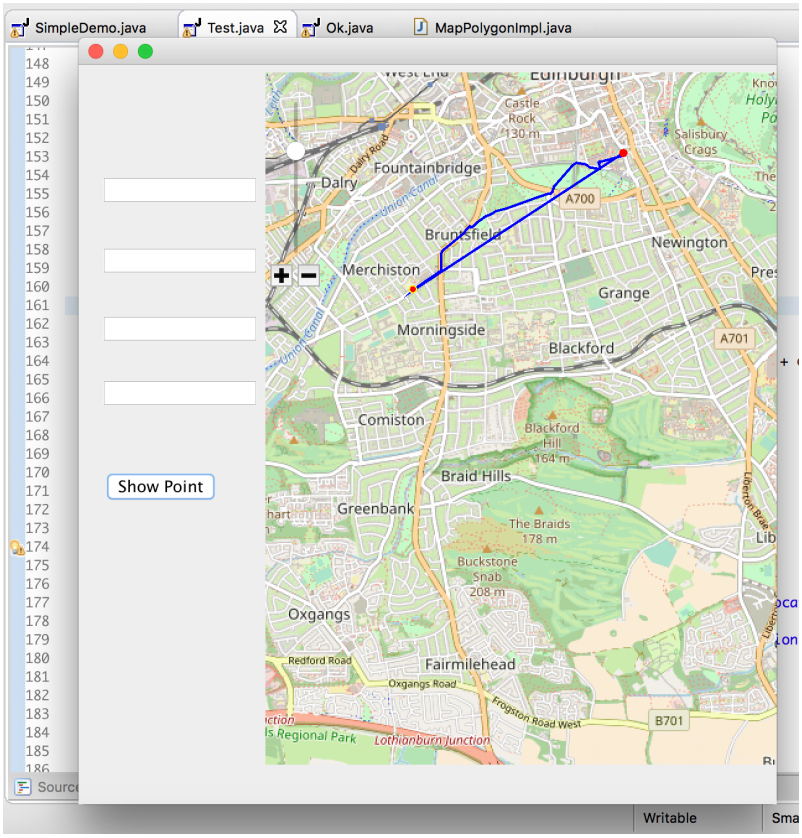
The route is a polygonal curve without the last line(because the polygon has the joint shape).It can also be thought of as a parametric representation of the movement of a point from the beginning to the end of the curve.

For Decoding the Polyline, i used a decoder algorithm which takes Polyline as an argument and returns an array of coordinates between the two points hence giving us enough points to draw the proper route.

This method returns an output of coordinates for the routes.To draw the route, open street maps have draw_polygon and _addmapPolygon and AddmapPolygonImpl methods which takes the coordinates.

These coordinates are from decoding the Polyline to a array list of coordinates and we put these coordinates in our polygon to draw the route.But as polygon is all side closed shape. We needed a function to get rid of the last line of the polygon.

Therefore, a function was implemented, that considers the last point and the second last point of the polygon the same so it draws the correct route.This way we can get distance, duration, end_location, start_location, steps, traffic_speed_entry, via_waypoint.



The first route

5.3 Matching algorithms

5.3.1 FOR MATCHING IF TWO USERS ARE WITHIN CLOSE START/END POINTS

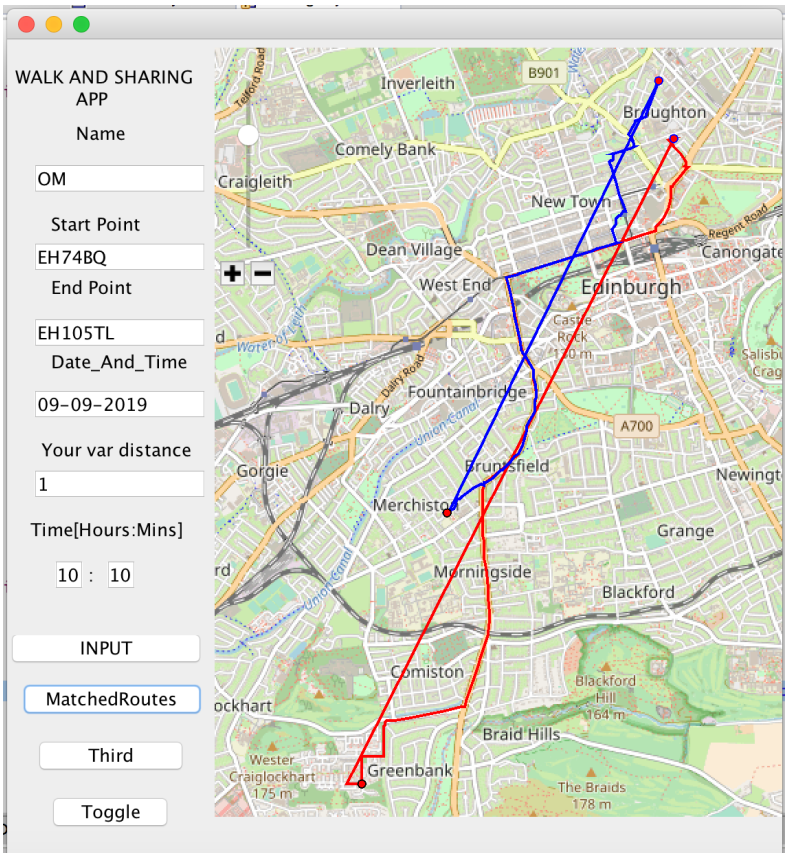
Initially the aim of the project was to get two routes to be shown based on the match, only if there are two people who wants to start from their own start locations and go to their own end locations but later it was changed to, only if their start locations are close enough to meet and start sharing the journey together and their end points are close enough to end their journeys together.

Each mapping between two intervals was annotated whether it represents a match or a mismatch according to a given proximity constraint, i.e. are they closer to each other than some distance, ε (variable distance).

So, we give a variable distance field to be filled by the user. the user inputs the value in this field between 1 to 5.

So, I have given the restriction of 5 kms that if they are within 5 kms of range, only then they would be shown a matching route(to share the journey together.)

If the start/end points are within this range, the draw_route function will draw the route.



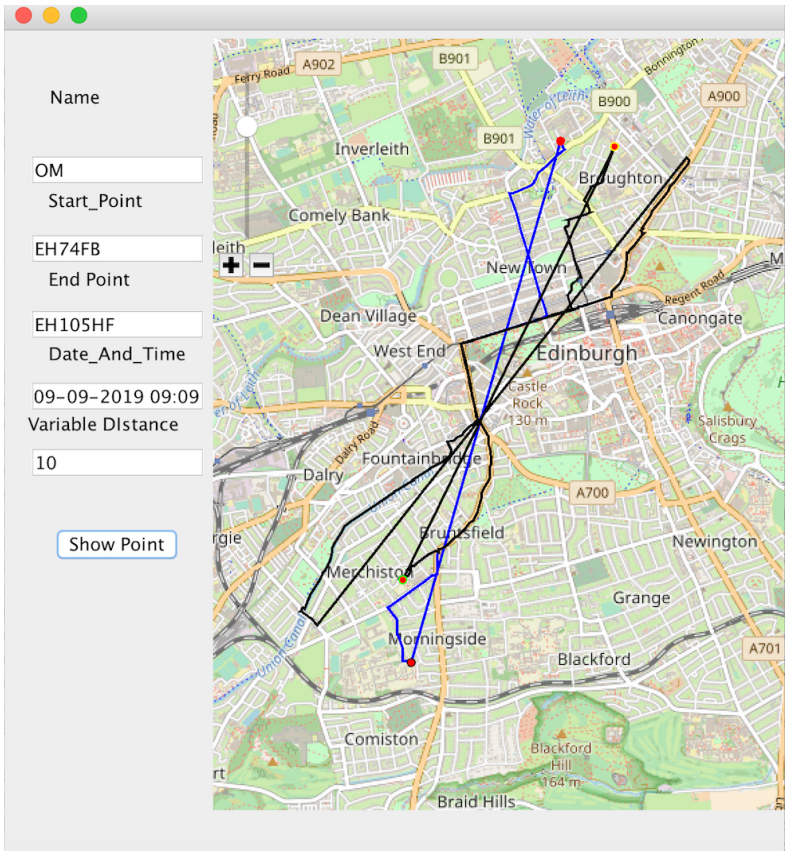
5.3.1.GETTING THE DISTANCE (To check close points)

First I used the Haversines formula to get the distance between two points but that gave the direct distance between two points without consideration of the streets or parks etc.

To get the proper walking distance between two points, the walking route between the two points was calculated and the number of steps.

One of the JSON Objects also returns the time taken between two points and also the walking distance in kms or meters.

Instead of using the direct(inaccurate) distance, this distance can be used as it is more correct.



5.3.1.2 FOR SHARED ROUTE, GETTING THE START POINT AND END POINT

In real world applications we need to compute the distance through proper roads.

So I calculated the time it takes to travel between the two start points of two users.

Distance is not too accurate as in case of steep roads or hills the same distance is travelled might be less, but the time it has taken for this user might be more. So to be fair to both the users, time is a better criteria which will decide the point where the two users meet.

There are several steps from one point to another.

Each step that the GMDA computes has its own time.

On adding up time to travel through all the steps, we get the overall whole time. When the time is half of the whole time (lets call this half_time), the very next coordinate after half_time will be the centre point for both the users.

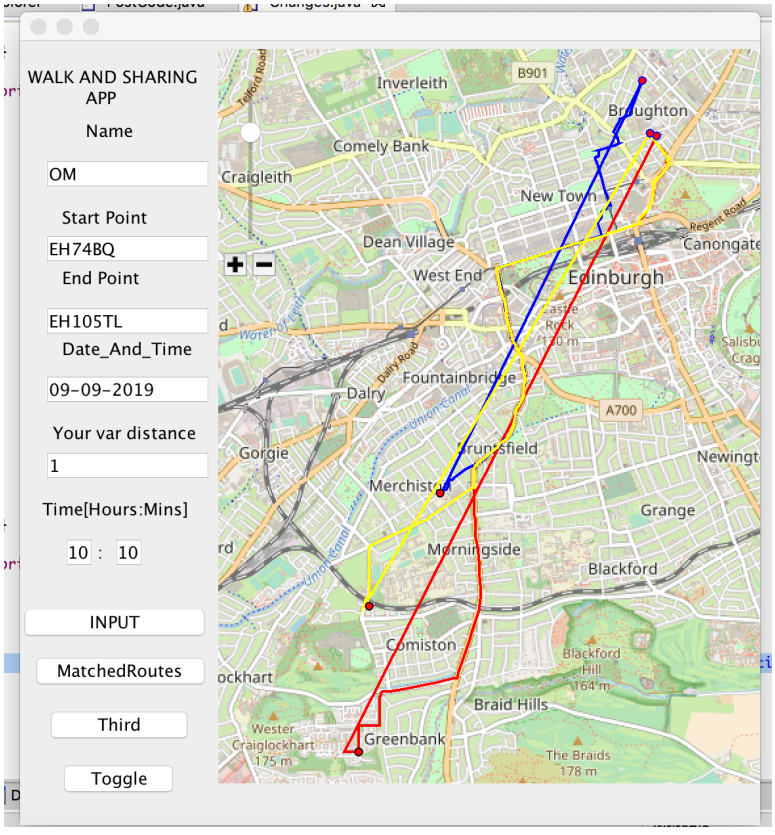
On retrieving the longitude and latitude of that point, that will be the point which is central to the start points of both users who are about to share the journey together.

They should both take almost equivalent amount of time (with a difference of a minute or two) to reach this centre point and start to share their journey together for their shared route.

Similarly, we find a centre point where the two users can split and go their own ways to their final destination.

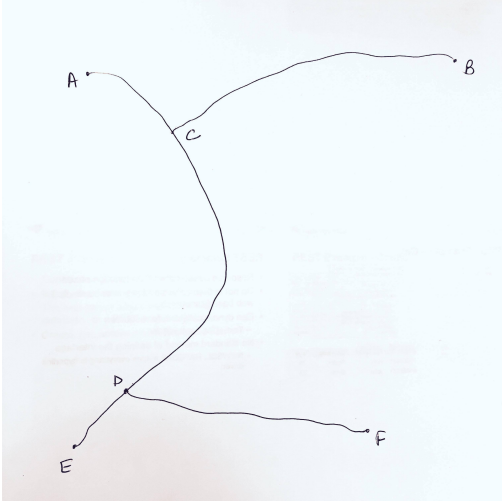
Once these two centre_start point and centre_end_point has been stored, we get the route between these two points. This route is shown in Yellow Colour in the app.

This is the shared route.



5.3.2 Matching based on overlapping routes

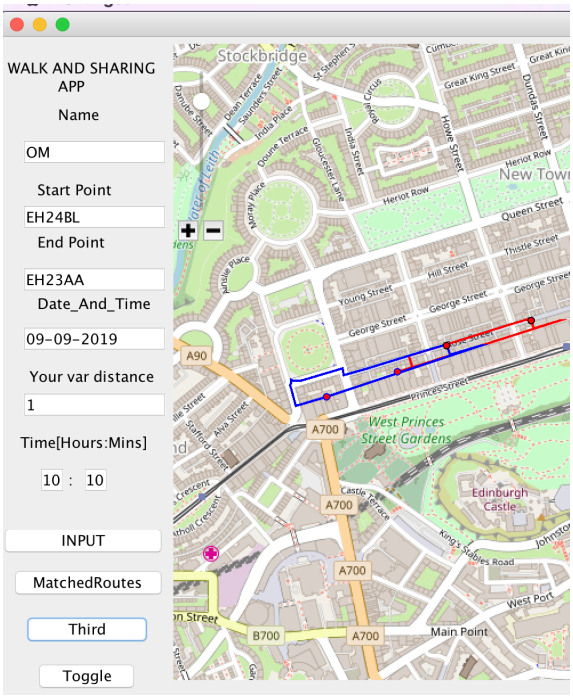
I understand that this matching is not good enough for this case:There are users who are not starting from the nearby start points and nearby end points, but they still might share a significant portion of the journey together.



The button "Third" implements this functionality: A partial matching which is a mapping of intervals on one route to intervals on another.

Checking that the coordinates on the partial matched routes, whether they are the same.At times, It is possible that the poly line returns co-ordinates on the same street with different points for different routes.

On clicking this button, the app shows those matched routes of two users whose start points and end points are not close but their routes still have significant portion that they can share.



The second condition for checking was to see if they are reaching that shared path at the same time. We also have to check that the shared route distance is significant.If the coordinates for both the routes, have some coordinates that are the same, that means there is a part of route for these two users that is the same.

So, then there is a need to check whether this path is significant enough to be travelled. That is, at least there is 15 minutes of sharing that the two users can be taking.

Condition 1: Whether the routes are overlapping (How it was implemented)

This is done by checking if the route 1's coordinates and route 2's coordinates have a same sub list. There is a method implementation which checks the two array lists. If one list is a sub list of other, then draw the route.

Condition 2:Whether they reach at the same time (How it was implemented)

The time is an important yet complicated procedure.

To see, if the two users reach their shared route at the same time, first we get the coordinates of the point where they will be meeting (the point where their shared journey starts, which is based on taking the centre point based on time)

For example, there are two start points for two users A and B, we get the route and time between two start points. Supposed the time is 36 minutes, so the half time would be 18 minutes.

That means the step next to when they have travelled for 18 minutes, should be their shared route's starting point(lets call this C). It is not the most accurate coordinate point but it is close enough.

Once we have this point.

We calculate the time it takes for user 1 to reach from their start point to C.

We already have the start time of the user, and once it has been calculated the time it takes for him to reach this C point, we know the time he reaches there. For example, he starts at 1:10 pm and it takes him 5 minutes to each the C point. He will reach there by 1:15 pm.

Same time frame or similarly, the time can be calculated for user 2.Then, we have to check whether they both reach there at the same time.

There is a time window of 5 minutes here. So, if one reaches 5 minutes before the other they can wait for 5 minutes. Once we have checked that they are reaching there at the same time(or within the same time window), there is a need to check if they are sharing a significant part of their journey together.

Condition 3:Whether the shared journey is significant enough(how it is implemented)

To check the distance and time the two users are sharing, we do that by taking the C point and corresponding End point and calculating the distance and time between these two points by parsing through JSON Objects.

Once all the conditions are checked, the application lets the user know and draws their routes along with their shared route.

On clicking the button, the matches are displayed with routes shown and they can see their journeys and notify each other.

6.BACKEND

6.1DATABASE

A MySQL database was created for storing the user's information and their routes' data. The setup was done on the personal laptop by installing it first and then connecting it to the eclipse project. I established a database connection from my Java program to a SQL database using JDBC.

```
    // time_common_meeting("EH74DG", "EH105DT");
    try {

        c = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/db?useSSL=false", "root",
                "Fatehjiom.!7");

    } catch (Exception e9) {
        e9.printStackTrace();
    }
```

```
   String a = "INSERT into main (user, start_point,end_point, date, hours,minutes) VALUES (?, ?, ?,?, ?, ?);";
   PreparedStatement ps;
   try {
       ps = c.prepareStatement(a);

       ps.setString(1, user);
       ps.setString(2, start_postcode);
       ps.setString(3, end_postcode);
       ps.setDate(4, sqlStartDate);
       ps.setInt(5, hour);
       ps.setInt(6, min);
       ps.execute();

   } catch (SQLException ez) {
       // TODO Auto-generated catch block
```

```
   String a = "INSERT into main (user, start_point,end_point, date, hours,minutes) VALUES (?, ?, ?,?, ?, ?);";
   PreparedStatement ps;
   try {
       ps = c.prepareStatement(a);

       ps.setString(1, user);
       ps.setString(2, start_postcode);
       ps.setString(3, end_postcode);
       ps.setDate(4, sqlStartDate);
       ps.setInt(5, hour);
       ps.setInt(6, min);
       ps.execute();

   } catch (SQLException ez) {
       // TODO Auto-generated catch block
       ez.printStackTrace();

   }
}
```

I have three tables namely:

LOGIN (For registration and login)

This table has four columns called name, password, phone Number and id. The first page of the app lets the user decide if they want to register or login.

If they do not already have the account, they register and input all the four above mentioned points. From the java application, it gets stored in the main database in the login table.

Once this login table has the details, the user can log in to the app because their details match the details that are already in the login.

On logging in, the user inputs their details using the "input" button. He enters his start point (the location he/she intends to start the journey from), end point (the point where he/she intends to end the journey), the date they will be travelling, the time and the variable distance. The variable distance (Kilo meters) is the distance that they are willing to travel to meet their matched user.

MAIN

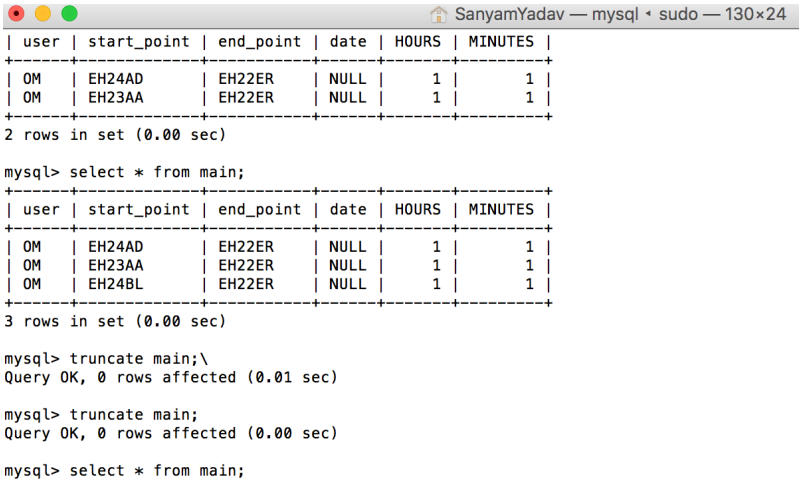
(For storing main information like start and points, time and variable distance, which is then used for matching)

Once the user inputs these details, they all go to the database in the table called main.

The main table has the following columns namely user, "start_point", "end_point", date, hours and minutes.What the user inputs is stored as start_point and end_point.The algorithm then matches the start_point and end_point in the whole database i.e., main table.

If in the table, there are already some active users, their information is stored in the database. They match the current user's inputs to the inputs in the table main.

The points from the database are called db_start_point and db_end_point.If these start points and end points are within 5 kilo meters, there is a match.



6.2Why choose MySQL?

While most relational databases were designed several decades ago for complex

ERP-type applications, MySQL was designed and optimized for Web applications. As new and different requirements emerged with the Internet, MySQL became the platform of choice for web developers, and the default database for web-based applications. [6]

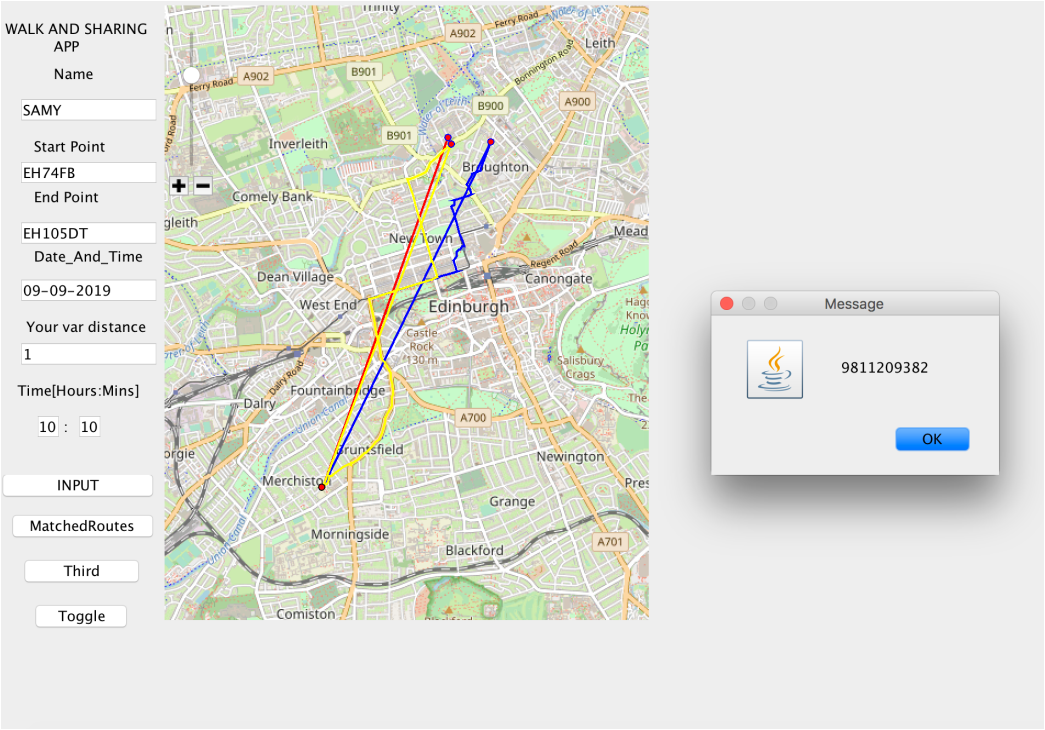
Having had a lot of experience in MySql since 6 years and done database systems courses on it, it seemed like a proper choice for my thesis.

6.3 Creating NOTIFICATION and Toggling

When the user creates or registers an account, the user enters their name, password, phone Number and ID.

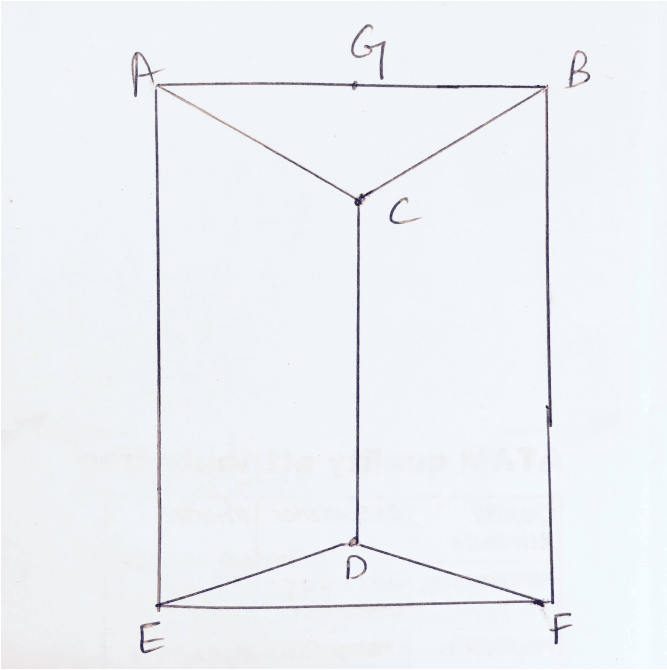
Their phone number has been stored in the database ‘db’ in a table named login. If that user matches with several users, they can then toggle (with the help of toggle button) to see their matches one by one.

While toggling with the matches, it also opens up a dialog box which displays the matched user’s phone Number. The contact information (phone number in this case) is then displayed in a pop up dialogue box. If they want to see other matches, then can close the dialog box, and keep toggling until they find the match that they are happy with.



7.Evaluation

In finding the shared route between two matched users,there are some drawbacks. suppose, in this case:



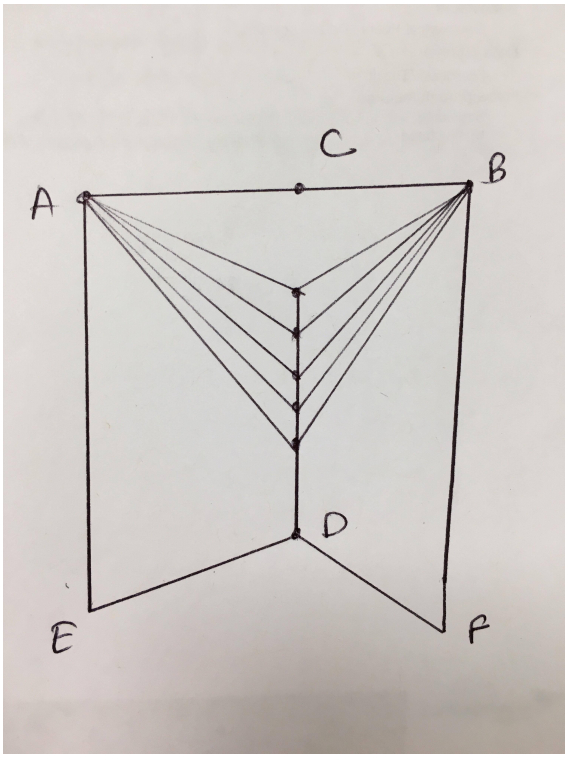
There are two users starting from points A and B respectively.

User 1 is going from Point A to E and User 2 is going from Point B to F.

According to my algorithm, they will meet at a centre point which is G in the image,their centre form their start points not based on distance but based on their time.

So they meet at G according to my theory that is the matching for both of them. But it would be better for both of them to meet at C. To decide on which of the points to go through is quite tough/complicated as it needs proper information to make the decision clear.

In fact they can meet on several points like:



It needs to decide which points to choose. “Second” decides the point based on time. Hence, it is not the perfect matching algorithm.

7.2 EXPERIMENTS/TESTING

For first matching algorithm based on proximity of start points and end points, when we test it for postcodes such as

User 1: EH30 9AR and EH30 9YJ

User 2: EH7 4DG and EH7 4BQ

It does not match because they start points and end points are not within 5 kilometres of range.

But if we give

User 1: EH7 4DG and EH105DT

User 2: EH7 4BQ and EH10 5TL, it draws the routes.

Similarly, many other test cases have been considered to make sure that the right result is provided.

For shared route,

The mid points between start points and end points have been calculated.

For overlapped routes,

User 1: EH7 4DG EH2 4AD

User 2: EH8 9JR EH2 2ER

It draws the routes, because the paths are overlapping and start points are not close. Similarly, a lot of tests have been run to make sure the algorithm gives the right results.

Testing the polyline

I have mentioned for future work, the last line of the polyline needs to be erased.

After testing several times, two methods for removing the last line of the polyline were implemented

1)That goes from one point to theist point, and on reaching the last point, it reverses back.

2)That we give it three points (A,B,B) , the second and third points remain the same.

So it goes like, (A,B,B) then (B,C,C) then (C,D,D) and so on.

Eventually, after testing several times it was figured that the drawing the polyline was not the problem, but instead the function draw_routes which draws the routes takes the coordinates three times because of three legs as seen in JSON output reader.

Fixing this, that the coordinates are not repetitive and only are output once will definitely draw the route with no last line. That will be more presentable and is indeed fixable.

7.3 SELF EVALUATION AND PROJECT EVALUATION

Initially the aim of the project was to create an android app.My supervisor advised that this project was too challenging but I remained adamant to not change the project.

Gradually, I started facing difficulties with the project and I realised that I need to start working and need to start showing the results.It is better to show some results than nothing to be able to do at all.I changed from android studio to the Eclipse as I have had experience in Eclipse in previous study years.

Further to this, I started building the GUI, and then started coding the matching algorithm. I realised this was getting me somewhere and was showing some results, so I started coding more and gradually developed the routes and so on.

I learnt it is much better to start early especially if you have taken a challenging problem. It is important to listen to others' advise/suggestions too.It is important not to stick to something if it hasn't worked for so long, so changing from Android studio to Eclipse IDE was a good idea.

Furthermore,I believe the project could have been much better.

The app has turned out pretty well but it would have been better if there had been more testing and more matching algorithms.Maybe Android studio would have been better too.The code could have been made better according to Software Engineering principles.The app could be implemented for multiple matching users so the journey can be shared by 2+ people.The option for the user to choose a starting point/ending point on maps by clicking on the map would have been better than type it. (More in Further Work)

7.4 FEEDBACK

According to some users, this would be a very helpful app that they are very likely to use. According to them, it can be a bit unsafe and it would be great if they can share the journey with someone but it will be better if there is a way to make sure that the other person is trustable.

There are a number of ways of doing this:

1) By connecting the app to Facebook so they can see their common friends.
2) Have a rating option, so once the user has shared the journey, they can rate them. If someone's rating is high, that means they are very safe.

7.5FURTHER WORK

1) In API, we can further have restrictions.You just have to pass an argument with the parameter indicator.Directions can adhere to some of them as follows:

- avoid=tolls
- avoid=highways
- avoid=ferries

It's possible to request a route that avoids any combination of tolls, highways and ferries by passing both restrictions to the avoid parameter. For example: `avoid=tolls|highways|ferries`.

Region Biasing

You can also set the Directions service to return results biased to a particular region by use of the `region` parameter. This parameter takes a ccTLD (country code top-level domain) argument specifying the region bias.

For example, the United Kingdom's ccTLD is "uk" (`.co.uk`) while its ISO 3166-1 code is "gb" (technically for the entity of "The United Kingdom of Great Britain and Northern Ireland").

You may utilise any domain in which the main Google Maps application has launched driving directions.
For example, a directions request from "Toledo" to "Madrid" returns a result when `region` is set to as "Toledo" is interpreted as the Spanish city:

https://maps.googleapis.com/maps/api/directions/json?origin=Toledo&destination=Madrid&region=es&key=YOUR_API_KEY

2)More algorithms can be used to match the routes of different users such as pairing up three or more people.

3)We can link the app to social media websites like Facebook, Instagram which will let the user know if they have common friends ultimately making it safer to choose a partner.

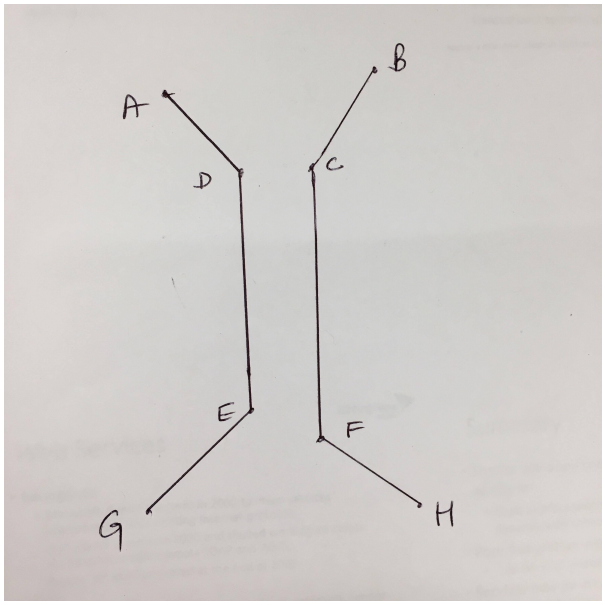
4)We can further change it into a mobile application using android or IOS applications. We can make user profiles to match based on preferences such as e.g., age group, common interests etc.

5) Remove the last line of the polyline drawing the route.

6) We can add a matching algorithm based on parallel roads.

7.5.1PARALLEL ROADS

Another implementation of the algorithm would be in this case.



Suppose the two users, are neither starting at similar points, not ending at similar points, they do not even have a significant shared route BUT

what if they have started from far apart start areas and their end points are not nearby either but a significant portion of their journey has been on parallel roads.

In this case too, they can share a journey together.

In this case there are three conditions to check

1) The roads are parallel

To check this, we get the coordinates of both the roads, we see the distance between coordinates of both the roads remain same for a significant number of points, that would proof that the two roads are parallel. Haversine's formula could be used at this situation.

Now that these roads are parallel, we can go ahead and to make them share the journey together.

2)Whether the two users are reaching the start points on these parallel roads at same time within a 5 minute time window.

In this case,we get a point where they can meet through the same above procedure ,

then they can either take parallel road 1 or parallel road 2.

Now, there can be two parallel roads but they might be very far apart. We need to make sure that the parallel roads are not too distant. We call this distance between two parallel roads <dis>. and we give it a restriction on 1 kilometre. If these two parallel roads are within 1 kilometre of distance, it makes a lot of sense for the two users to share the journey together.

Then we find the meeting point for the both the users between the two parallel roads.

This should be about the centre point for these roads, once these centre points are taken, the user can decide which parallel road to take that they can share together.

Any road they take, the centre meeting point and the centre separating point should be the same.

3) The shared portion is significant.

Check the distance of these parallel roads, the portion of the parallel roads and if that distance is significant.

——

7.6 REFERENCES

[1] (https://en.wikipedia.org/wiki/JSON)

[2] https://developers.google.com/maps/documentation/javascript/usage

[3] (https://developers.google.com/maps/documentation/utilities/polylineutility)

[4] https://www.ifootpath.com/

[5] http://www.mapmywalk.com/app/

[6] http://www.oracle.com/us/products/mysql/mysql-wp-top10-webbased-apps-461054.pdf

[7] https://blogs.oracle.com/oracleuniversity/10-reasons-why-you-should-consider-learning-java

[8] https://www.tutorialspoint.com/java/java_polymorphism.htm

[9] http://www.letswalkapp.com

[10] https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

[11] https://developers.google.com/maps/documentation/directions/intro#Waypoints

[12] http://etd.dtu.dk/thesis/268267/ep10_81.pdf

[13] https://www.sciencedirect.com/science/article/pii/S0968090X15000728

[14]https://www.nngroup.com/articles/ten-usability-heuristics/

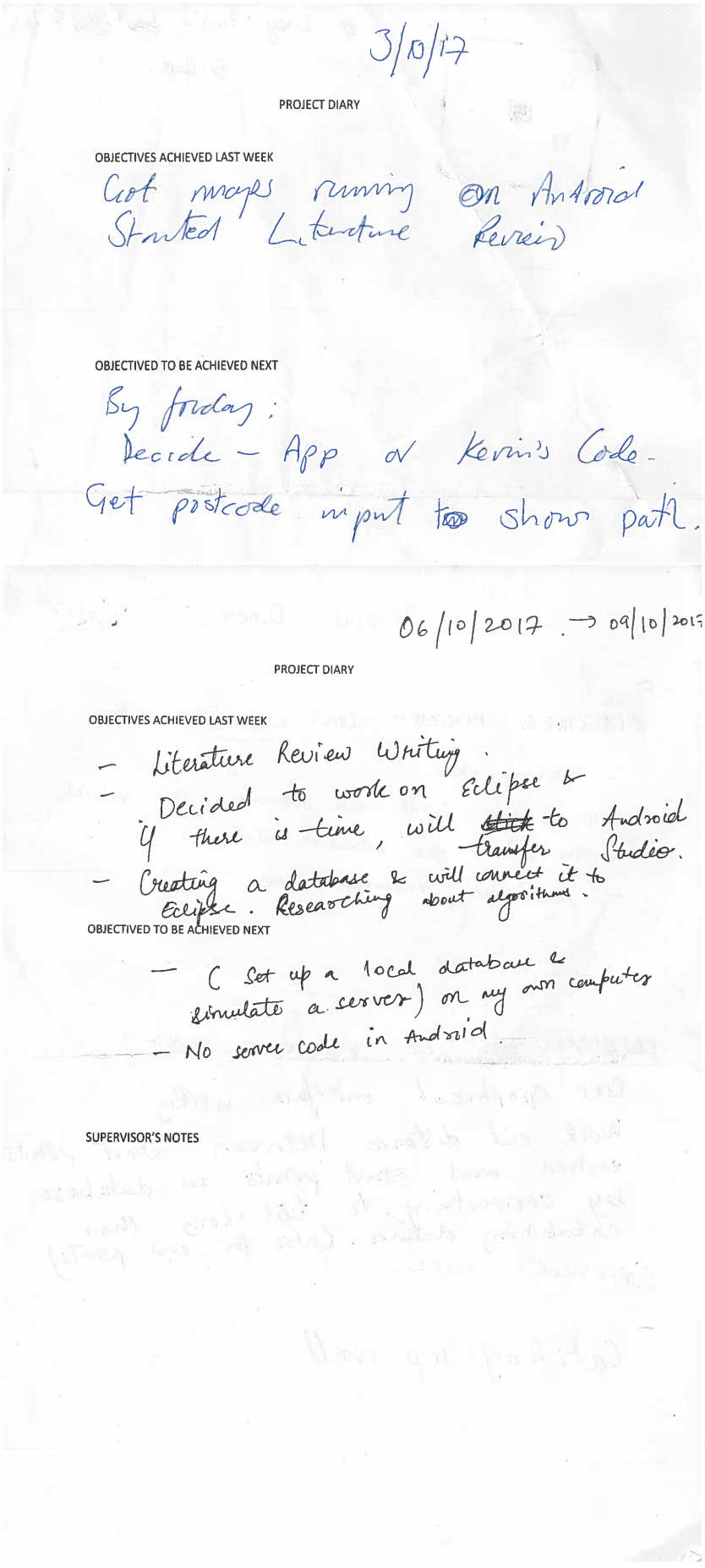
[15]http://ad-publications.informatik.uni-freiburg.de/ALENEX_matchpredict_EFHSS_2011.pdf

[16] https://developers.google.com/maps/documentation/directions/intro#Waypoints

[17] https://wiki.openstreetmap.org/wiki/Main_Page

APPENDIX

Here are a few of the project diaries scanned copies maintained during the course of the project.



3/10/17

PROJECT DIARY

OBJECTIVES ACHIEVED LAST WEEK

Got maps running on Android
Started Literature Review

OBJECTIVED TO BE ACHIEVED NEXT

By Friday:
Decide – App or Kevin's Code.
Get postcode input to show path.

06/10/2017 → 09/10/2017

PROJECT DIARY

OBJECTIVES ACHIEVED LAST WEEK

- Literature Review Writing.
- Decided to work on Eclipse & if there is time, will ~~stick~~ transfer to Android Studio.
- Creating a database & will connect it to Eclipse. Researching about algorithms.

OBJECTIVED TO BE ACHIEVED NEXT

- (Set up a local database & simulate a server) on my own computer
- No server code in Android

SUPERVISOR'S NOTES

PROJECT DIARY. 13/10/17

OBJECTIVES ACHIEVED LAST WEEK.

- Database done.
- Users can input into database their values.
- The values get stored in database.
- Algorithms research done.

OBJECTIVES TO BE ACHIEVED NEXT

Get graphical interface working
Work out distance between start points entred and stored points in database by converting to Lat + Long then calculating distance. (also for end points)

SUPERVISOR'S NOTE

- Catching up well.

16/10/2017

PROJECT DIARY

OBJECTIVES ACHIEVED LAST WEEK

- Algorithm
- Parsing csv file
- Coding & writing

OBJECTIVED TO BE ACHIEVED NEXT

- to be worked on buttons

SUPERVISOR'S NOTES

If you can't fix bug within 24 hours come back to see me.

20.10.17 Something runs but still not calculation correctly. Try to fix by Monday
Calculating distance via roads is too difficult – better not to try this.

0, 1, 2
Long Lati

23/10/2017

PROJECT DIARY

OBJECTIVES ACHIEVED LAST WEEK

- GUI working & matching.
- Designing

OBJECTIVED TO BE ACHIEVED NEXT

SUPERVISOR'S NOTES

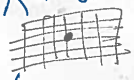
- Specify a walking journey (Routes) (Google Maps API)
- Time Window
- ~~the~~ Then be matched with ~~significant~~ other users who share a significant portion of ~~the~~ a the route. Different partners for sharing the app. (Different ~~algorithms~~ matching algorithms)
- Screenshots of errors.

27/10/2017

Next tasks

✓ (1) Variable distance for match

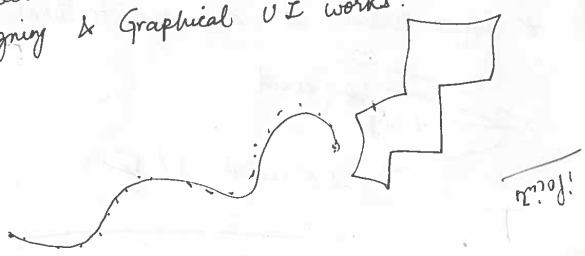
✓ (2) Calculate correct distance based on Lat Long (Grid Reference)



Maps
- (3) Show match graphically
- (4) Show own points on Map
- (5) Show match points on Map
- (6) Show joint route

Objective achieved last week

- Matching done.
- Objectives achieved
- ~~to~~ Designing & Graphical UI works.



9/11/17.

Try to write in dissertation about all the ways that two routes can be combined - taking into account both distance and safety. Then say what subset you are going to actually implement.

- shared (common meeting through time)
- "
- "

13/Nov/17

Objectives achieved last week:

- Do I need coordinates .contain all alg ?
- Tables ?
- Time Matching ?
- 3 diff routes. (3 buttons)
- var. dis
- Polyline

Tasks to be accomplished next:

Supervisor's Notes