PROJECT DIARY - *Group 9*

<u>Approaches:</u>

1. Map Component: For showing and computing the journey route
    a. Google Maps - https://developers.google.com/maps/documentation/android-sdk/intro
        i. Pros:
            - The simplest choice for maps on an Android phone.
            - Easiest to integrate
            - Offer many out of the box functionalities
        ii. Cons:
            - Cost - Incorporating Google Maps in the project would need a subscription for the SDK
            - Not Available Offline. As discussed with the professor, the offline aspect of the app is of utmost importance.
            - Using an available API would not be a good approach as recommended by the professor
    b. MapBox - https://docs.mapbox.com/android/maps/overview/
        i. Pros:
            - Provided us with offline functionality
            - Had extensive developer support will clear documentation and explanations
        ii. Cons:
            - The free tier had several limitations in terms of offline tiles available and number of users the supported
            - Cost - Incorporating Mapbox in the project would need a subscription for the SDK
    c. Here Maps - https://developer.here.com/
        i. Pros:
            - Offline maps SDK for Android
            - Provides routing functionality
        ii. Cons:
            - The services offered by the free tier were limiting
            - Several services required an internet connection to be available
    d. Graphhopper (Implemented in the project) - https://wiki.openstreetmap.org/wiki/GraphHopper
        i. Pros:
            - Free tool for implementing the offline map
            - Provides routing functionality offline
            - Uses the OpenStreetMap (OSM) Protocolbuffer Binary Format (PBF) Filed to create the nodes, edges and relations
            - Provided a routing engine to us for use and compute the route
        ii. Cons:

- Long queries will need lots of RAM, which is also not very handy on mobile devices. So you'll have to limit the length of the route, increase the heuristical nature of the algorithm
- Difficult to set up. Long process to extract the files and get them working with the app.
- Required files to be kept in a specific location to be accessed for the construction of the map.
- The tiles for offline maps are quite big and takes lot of space in the mobile device(~350Mb for Ireland).
- Offline maps do not provide features like live traffic which could be a problem if the chosen route is the optimal one

2. Peer to peer Communication:
   a. Bridgefy - https://www.bridgefy.me/developers.html
      i. Pros:
         - Provides Hybrid communication Functionality - Hybrid Offline and Online Networks supported
         - Provided many great functionalities out of the box
      ii. Cons:
         - Only worked on Bluetooth in our Tests
         - Would handle all of the p2p aspects and will not leave scope for us to develop functionalities
         - Would need a subscription for using certain functionalities
   b. Hypelabs - https://hypelabs.io/documentation/
      i. Pros:
         - Provides P2P mesh functionality to allow hops between peer networks
         - Would take care of the entire P2P component required by the app
      ii. Cons:
         - As the SDK provided a lot of functionality out of the box, the development tasks by the team would be trivial
         - The wifi - direct functionality was not working in our tests
   c. Wifi-Aware
      i. Pros:
         - This approach is provided by the android OS inherently.
      ii. Cons:
         - This is new technology which is documented to be supported android 8.0 and above, however the latest Google pixel 3 which has version 8.0+ was not found to have this capability.
   d. Wifi-Direct -
      Cons:
         - The problem in this approach was that all devices were listed after the discovery, i.e. all devices w/o Journear apps were included in the list to connect. There wasn't any way to filter the list to only display devices that had Journear App installed.
   e. DNS based Service Discovery -

This approach came out to be successful as only devices having Journear app are listed and the problem in using Wifi-Direct was solved.

3. Multiple Device Communication
    a. MQTT - Hive and Eclipse Paho
        i. Pros:
            - More efficient information distribution; increased scalability; a reduction in network bandwidth consumption dramatically
        ii. Cons:
            - Centralized broker can limit scale
    b. Rabbit MQ -
        i. Pros:
            - Fast and Simple management
        ii. Cons:
            - High latency, need for a broker to relay messages.
    c. DNS-SD based decentralized connectionless protocol
        i. Pros:
            - Works like a radio broadcast communication
        ii. Cons:
            - Limitations in message length. No implicit handshake for confirmation

<u>**Features (implemented and future scope):**</u>

Implemented features:

1. Security:
    a. AES algorithm is used for encryption the data while communicating with data among devices.
    b. User's profile password is hashed for security reasons using SHA256 algorithm.
2. Fault Tolerance:
    a. If the internet goes off: The app can work in offline mode using connection-less Wi-Fi mode.
    b. If the app crashes: The app saves the data on the server as well as on local mobile SQLite database.
3. Ethics:
    a. The user will be asked for the permissions for using location and device storage. If the user provides these permissions then only the app performs the functionalities. This has been mitigated by allowing the app to use these services only while it is in use by the user and not in the background.
    b. Another concern is the risk associated with sharing the ride details with unknown people, which could lead to criminal and malicious activities. We have used filters to limit this but it is another concern that must be carefully handled.
    c. As the app works on broadcasting P2P messages and uses user devices to hop the messages beyond the range, this could put immense load on the user devices leading to battery draining quickly and/or the phone overheating.
4. Scalability:
    a. Since most of the application's functionalities are based for offline based scenarios, we use DNS to communicate. We believe that, for this service, the mobile internally

uses Wifi to communicate and due to this reason itself, the limit to connect and communicate with other devices can be at most 8 devices. With respect to online devices, currently, we are using Heroku.

5. Flexibility:
    a. Every bit of constants was added in a common Constant file. For example, changing the database API can be done using the constants file instead of changing it everywhere in the code.

Other possible ways to increase scalability and flexibility:
Scalability:

1. If the number of users using the online functionality to schedule rides increases, we can switch to a better service provider like AWS. The same issue can be solved on AWS by load balancing but these are expensive solutions.
2. HaProxy with keepAlived can be used instead. This is TCP/HTTP based application software that provides features like which algorithm to use while transferring the route to other node servers if the master server goes down.
3. Other options like the number of requests a node can handle to serve the requests and if exceeded, the requests are shared among the nodes.

Flexibility:

1. XML based file systems can be used for taking the parameters like database information(database server address, port, password etc), server information(server address and APIs). This file can be edited directly by client/developers easily without needing to know the actual working of the code.

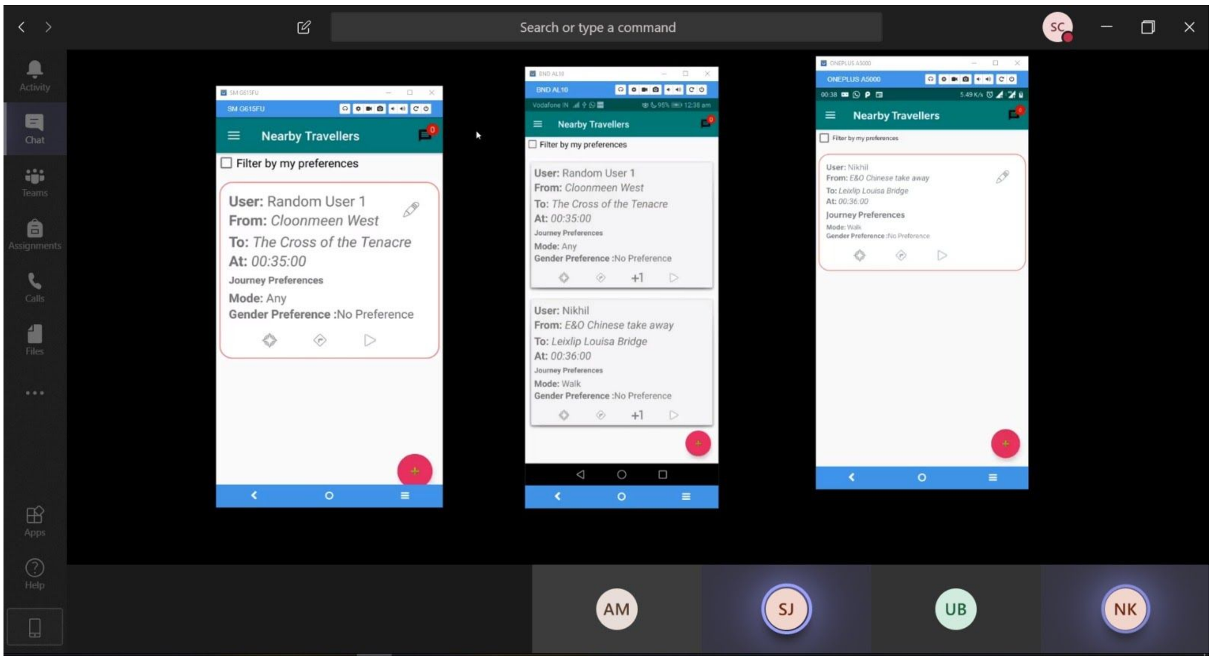
<u>**Pair Programming Schedule** *(Division of Labor):*</u>

Iteration	**Pair and Task**		**Pair and Task**		**Pair and Task**	
	Pair	*Task Component*	*Pair*	*Task Component*	*Pair*	*Task Component*
Semester 1 (Change pairs every week)						
1	Akshay, Nikhil	Research on use of Android and App development technology	Sarvani, Sujit	Research on P2P and feasibility of Encryption Techniques	Utkarsh, Taranvir	Research on server-side technology and cloud server research
2	Nikhil, Sarvani	Detailed Requirement Specifications for Deliverable 1	Sujit, Utkarsh	Functional and Technical Architectures	Taranvir, Akshay	Project Development Plan
3	Akshay, Sujit	P2P communication Proof of Concept	Sarvani, Taranvir	Android Application Proof of Concept	Utkarsh, Nikhil	Map API, Infrastructure Setup
4	Sarvani, Utkarsh	Preparing thin slice of Android App, Map API	Taranvir, Akshay	Database Design – Local and Cloud,	Nikhil, Akshay	Preparing thin slice of P2P Development

				Continuous Integration		
5	Utkarsh, Akshay	Finalizing thin slice of P2P Mesh	Taranvir, Nikhil	Finalizing thin slice of Map API, Make basic UI of the app	Sarvani, Sujit	Finalizing thin slice of Database Design
6	Taranvir, Sarvani	P2P Communication Core	Nikhil, Akshay	Maps – Routing and Matching	Sujit, Utkarsh	Authentication and Security and Mock Acceptance Test
Semester 2 (Change pairs every two weeks)						
7	Akshay, Nikhil	Isolating the devices on P2P since it is currently visible to anyone on Wifi Direct	Sarvani, Sujit	Communication of multiple devices using P2P	Utkarsh, Taranvir	Building an offline map with tiles and navigation using Graphhopper
8	Nikhil, Sarvani	Exploring the DNS - SD approach and build the P2P module	Sujit, Utkarsh	Research on MQTT & Mock Acceptance Tests	Taranvir, Akshay	Offline Route Calculation using Graphhopper
9*	Akshay, Sujit	Resolving the bugs in the offline map	Sarvani, Taranvir	Research and try tools to successfully follow the pair programming paradigm	Utkarsh, Nikhil	Set up integration methods to be used while working from home
10	Sarvani, Utkarsh	User Interface	Taranvir, Akshay	Integrating the output of Geocoding component into maps	Nikhil, Akshay	Integration of Peer communication module: convert Kotlin to Java
11	Utkarsh, Akshay	User Interface	Taranvir, Sujit	Geocoding in Maps	Sarvani, Nikhil	Integration of Peer communication module & Mock Acceptance Tests
12	Nikhil, Utkarsh	Integration of Geocoding on UI	Taranvir, Akshay	Maps & Mock Acceptance Tests	Sujit, Sarvani	P2P Communication
13	Akshay, Sarvani	Documentation and Integration	Taran, Sujit	Documentation and Server Communication	Utkarsh, Nikhil	Documentation and User Interface

***No significant work was possible as 4 of the members were travelling back home.**

Due to the COVID - 2019 breakout, the team was forced to work remotely and this gave rise to new challenges to be overcome. The issues of unreliable internet connection, issues with different time zones and the lack of devices to test the app made pair programming and development a challenging task. The team faced issues and experimented initially with Visual Studio Code Live Share, but finally settled on Microsoft Teams as the tool to be used for pair programming. It was a learning experience and the team managed to adapt to the unique situation and work effectively. Below is the snapshot of Microsoft Teams, where the team is testing the app.



Nearby Travellers

Filter by my preferences

User: Random User 1
From: Cloonmeen West
To: The Cross of the Tenacre
At: 00:35:00
Journey Preferences
Mode: Any
Gender Preference :No Preference

User: Nikhil
From: E&O Chinese take away
To: Leixlip Louisa Bridge
At: 00:36:00
Journey Preferences
Mode: Walk
Gender Preference :No Preference

<u>Time estimates and the impact of inaccurate estimation:</u>

1. Estimated time:
   10 hours a week for 6 weeks in the First Semester and 14 weeks in the Second Semester

2. Actual time spent:
   a. While it was attempted to strictly adhere to the allotted timelines, there were a few 1-2 hours skip especially when there were other deadlines coinciding with the ASE meetings. To cover up for this time lost, the team would meet for additional hours and compensate for the lost time.
   b. Inability to meet during the lock-down hindered the team's velocity due to unavailability of Android devices. No significant work was carried out for a week when four members of the team flew back to their homes and the project had to be worked on from five geographically different locations. However, to compensate this, the team had held meetings for an additional one hour every day for the next two weeks over the allotted 10 hours.
   c. The team also had to search and experiment with tools that could facilitate pair programming remotely.

3. Impact:
    a. Testing of peer to peer communication wasn't extensively done due to unavailability of devices to test on.
    b. Research on Multi-communication using Messaging Queue technique was carried out but could not be implemented and tested due to non-availability of mobile devices.
    c. Chat component could not be integrated
    d. Catering for travellers within a 500m radius could not be tested due to the lockdown situation
    e. Route optimization was not executed