***How to run program :***

1. python cluster.py [explained below]

2. python request.py [explained below]

Note : Allow cluster.py to run for (\_\_\_ min) so that it may learn the correct routes of buses

***Application Modules***:

1.cluster.py

a. Plot points listed in database and acts as server

b. Plot actual buses

c. Cluster the points based on clustering algorithm and classify clusters as either buses or pedestrians or any other vehicle

d. Plot the buses calculated based on clusters (and compare the difference between the actual and calculated buses graphically)

e. It also shows the bus frequency and average wait time of each station

2. request.py

a. Runs parallel with cluster.py

b. Takes origin and destination bus stops as input

c. Outputs the identity and current location of the optimum bus. The optimum bus is the bus which takes the least amount of time to reach source station and finally reach the destination.

Note : Allow cluster.py to run for (\_\_\_ min) so that it may learn the correct routes of buses

***Synthetic Data Creation:***

Note: 1. Synthetic data located in /database/

2. Database Attributes = Identity Number, Time Stamp, Speed, Location(x & y coordinate), Direction, Street

3. In reality the actual data would be obtained by extracting information from the users of the app

1. Synthetic data generated by program synthetic\_data.py

2. Created 4 bus routes which cover 12 bus stops

3. Initially all buses have 30 people and each bus stop has 20 people

4. Number of people boarding and deboarding a bus follows Poisson's distribution

5. Parameter of Poisson's distribution (λ) has been defined individually for each bus and bus stop

6. λ of a bus stop gives the average number of people who deboard there from any bus. Similarly the λ of a bus gives the average number of people who board the bus from any bus stop

7. The coordinates of the people in the bus is assigned within some small radius (2^0.5 pixel) and the differences of speed are tolerated up to 10 units going in the same direction as bus

8. The people on the bus stops have speeds up to 30 units and random directions

9. Random points on streets with speeds comparable to buses with random directions

***Working of Clustering Algorithm***:

1. In each iteration, points within 20 pixel radius of every bus stop are sent to a cluster function which returns the possible clusters which may form buses

2. Cluster filters out points whose speed is below cut-off (30 units). Then it creates a graph with edges between points whose distances are less than 7 units, difference in speed is less than 10 units and re in similar directions. Then the connected components created are returned as different sets.

3. The above process is done for each bus stop.

4. The obtained set of clusters is compared with existing buses (The set of points which we have identified as travelling in the same bus. Initially there are no buses. ) If the intersection of the bus with cluster is above a certain threshold we will replace the existing bus with that cluster.

5. All the remaining clusters are assigned as new buses

6. After each iteration of time stamp, average coordinates of points in the bus are written into the database which is used by request.py and also used for plotting purposes

7. Whenever the average speed of points in a bus falls below 30 and it is 5 pixel radius of a bus stop, the bus stop is added in the bus route of the particular bus. This bus route is also updated after each iteration and written into the database. This bus route is used by request.py to predict the optimum bus.

***Working of request.py:***

1. It takes input origin and destination through a GUI

2. Selects all the buses having origin and destination in its bus route.

3. For each bus identified in previous step, it calculates expected time taken by bus to reach origin and adds expected time taken to reach destination from source based on the route.

4. The bus having minimum total time is considered the optimum bus and it displays identity and current location of the optimum bus