**Habib University Map**

**Project:**

The project includes mapping the university including all the places that have engagement with the students with the exception of EHSAS where footsteps have been taken as unit of length. Then implementing the DIJKSTRA algorithm to find the shortest path between two places on the map. The project uses data structures including dictionary, list and tuple (as to the reason why, will be explained further on). As the GUI implementation was optional so we only implemented it on the lower ground floor and second floor and for this we used the turtle library (implementation is explained further in the report).

**Programming Language and interface:**

The programming language used was python and the code has been written in this language. Furthermore, the interface used was Python IDLE 3.7 and PyCharm and the libraries such as turtle and matplotlib and networkx were also from Python.

Now coming on to the type of data structures and the reason of usage:

**Data Structures:**

* **Dictionary** is a unique data type because it allows random access to any element in it as it does not maintain an ordered array like list. This is also the reason we chose this data type for storing our nodes and their corresponding names because we could easily change it as dictionary is mutable. The coordinates of the nodes for running the GUI have also been stored in the same manner. Its structure basically has a node and a key value assigned to that node which can easily be assigned by assignment call G[i]=’fall’. Also it saves time as any assignment or access is order O (1). We have used it for assigning codes to our nodes and so the key is the numerical code and then the value is the name of building corresponding to that node. Example shown below:
* N={'E-017':'Dukan',  
   'E-013':'Tapal Cafeteria',  
   'E-012':'Rangoonwala Classroom',  
   'E-011':'Dinshaw Seminar Room',  
   'E-010':'Visualization & Graphic Lab',  
   'E-003':'Linux & Networking Lab',  
   'E-002':'Media Mock-up Studio',  
   'C-025':'Engineering Workshop',  
   'C-023':'Head of Administration and Campus Services',  
   'C-022':'Administration and Safety Offices',  
   'C-018':'Female Gym',  
   'C-017':'Co-Ed Gym',  
   'C-015':'Classroom',  
   'C-007':'Project Lab',  
   'C-004':'Power Lab',  
   'C-001':'Circuits & Electronics Lab',  
   'W-007':'Plantroom',  
   'W-004':'Facilities Offices',  
   'LB-001':'EHSAS',  
   '001':'LEARN',  
   '002':'Indoor Games',  
   '003':'Fries',  
   '004':'Junction',  
   '005':'EE Labs',  
   '006':'Sports Court',  
   '007':'Dhaba',  
   '008':'Exit',  
   '009':'Washroom01',  
   '010':'Washroom02',  
   '011':'Pool',  
   '012':'Lift01',  
    
   'E-121':'Soorty Lecture Theater',  
   'E-105':'Faculty Pod',  
   'E-101':'Cassim Computing Lab',  
   'E-100':'Agha Multipurpose Hall',  
   'C-110':'Student Lounge',  
   'C-110A':'Student lounge second entrance',  
   'C-109':'Arif Habib Classroom',  
   'C-114':'Faculty Pod',  
   'C-100':'Faculty Pod',  
   'W-114':'Dig. Sys. & Instrumentation',  
   'W-111':'Communication Lab',  
   'W-110':'G.M. Adamjee Math Lab',  
   'W-121':'Ecology Lab',  
   'W-118':'Chemistry Lab',  
   'W-100':'Physics Lab',  
   'LB-100':'Library & Information Commons',  
   'SC-100':'Student Center',  
   '106':'Entrance',  
   '107':'CS1',  
   '108':'CS2',  
   '109':'CS3',  
   '110':'CS4',  
   '111':'CS5',  
   '112':'CS6',  
   '113':'CS7',  
   '114':'CS8',  
   '115':'CS9',  
   '116':'Water Courtyard',  
   '117':'Corn',  
   '118':'Library Door',  
   '119':'Cafe2Go',  
   '120':'Info Commons',  
   '121':'HS & SL',  
   '122':'R1',  
   '123':'R2',  
   '124':'R3',  
   '125':'DSA',  
   '126':'ATM',  
   '127':'Courts',  
   '128':'Lift11',  
   '129':'Bank',  
   '130':'Health Services',  
   '131':'Student Life',  
   '132':'Writing Center',  
   '133':'Reception',  
   '134':'Lift12',  
   '135':'Lift13',  
   '136':'Washroom11',  
   '137':'Washroom12',  
   '138':'Washroom13',  
   '139':'Washroom14',  
   '140':'Up1',  
   '141':'Down1',  
    
   'N-200':'Faculty Pod',  
   'N-219':'Gulamali Habib Classroom',  
   'N-220':'Standard Chartered Classroom',  
   'E-220':'Tariq Rafi Lecture Theater',  
   'E-226':'Film Studio',  
   'E-215':'Auditorium',  
   'C-203':'Faculty Pod',  
   'C-201':'Center for Pedagogical Excellence',  
   'C-200':'Kassim Parekh Classroom',  
   'W-244':'Design Studio',  
   'W-243':'Classroom',  
   'W-242':'Javat Seminar Room',  
   'W-221':'Faculty Pod',  
   'W-234':'Mahmood Nensey Classroom',  
   'LB-200':'Library & Information Commons',  
   '201':'To Baithak',  
   '202':'To Faculty',  
   '203':'slope 2',  
   '204':'Corridor 1',  
   '205':'Corridor 2',  
   '206':'Auditorium Stage',  
   '207':'Library Reception',  
   '208':'stairs1',  
   '209':'Tables',  
   '210':'Washroom21',  
   '211':'Washroom22',  
   '212':'Washroom23',  
   '213':'Yohsin',  
   '214':'Lift21',  
   '215':'Lift22',  
   '216':'Lift23',  
   '217':'Baithak',  
   '218':'stairs2',  
   '219':'Up2',  
   '220':'Down2',  
    
   'W-300':'Faculty Pod',  
   'W-311':'Projects Lab',  
   '302':'Lift31',  
    
   'N-408':'Prayer Room (Female)',  
   'N-407':'Prayer Room (Male)',  
   'N-400':'Staff Dining Hall',  
   '401':'Lift42',  
   '402':'Child-Care Center'  
   }
* **Tuple** is a different data type which we have used to store edges including the connected nodes and the weight of the connection which represents the number of footsteps between them. Tuple is an ordered array meaning each element can be accessed by their index and is also immutable and does not accept assignment calls like dictionary. Example:

E=[('005', '109', 35, 'Open'),  
 ('005', 'C-001', 16, 'Open'),  
 ('005', '001', 48, 'Open'),  
 ('001', '141', 30, 'Open'),  
 ('001', '012', 62, 'Open'),  
 ('012', 'W-004', 15, 'Open'),  
 ('W-004', 'C-025', 6, 'Open'),  
 ('C-025', '011', 57, 'Open'),  
 ('011', 'C-022', 8.5, 'Open'),  
 ('C-022', 'C-023', 13, 'Open'),  
 ('C-023', '002', 4, 'Open'),  
 ('002', 'C-018', 17, 'Open'),  
 ('002', 'C-017', 17, 'Open'),  
 ('002', 'C-015', 12, 'Open'),  
 ('C-015', '009', 26, 'Open'),  
 ('C-015', '003', 25, 'Open'),  
 ('009', 'E-017', 19, 'Open'),  
 ('E-017', '003', 21, 'Open'),  
 ('003', '006', 65, 'Open'),  
 ('003', '007', 21, 'Open'),  
 ('E-017', 'E-013', 8, 'Open'),  
 ('E-013', 'C-007', 44, 'Open'),  
 ('C-007', '004', 26, 'Open'),  
 ('C-007', 'C-004', 21, 'Open'),  
 ('C-004', 'C-001', 21, 'Open'),  
 ('C-001', 'E-002', 13, 'Open'),  
 ('E-002', 'E-003', 4, 'Open'),  
 ('E-002', '010', 4, 'Open'),  
 ('E-003', 'E-010', 33, 'Open'),  
 ('010', 'E-010', 33, 'Open'),  
 ('E-010', 'E-011', 4, 'Open'),  
 ('E-011', 'E-012', 4, 'Open'),  
 ('E-012', '008', 10, 'Open'),  
 ('E-012', '004', 11, 'Open'),  
 ('008', '004', 9, 'Open'),  
 ('008', 'E-226', 100, 'Open'),  
 ('004', 'E-105', 105, 'Open'),  
 ('124', '127', 205, 'Open'),  
 ('124', '123', 31, 'Open'),  
 ('124', '116', 31, 'Open'),  
 ('123', '122', 25, 'Open'),  
 ('123', '128', 14, 'Open'),  
 ('123', 'W-100', 14, 'Open'),  
 ('123', 'W-110', 14, 'Open'),  
 ('122', '121', 41, 'Open'),  
 ('122', '129', 75, 'Open'),  
 ('129', '106', 58, 'Open'),  
 ('W-114', '116', 79, 'Open'),  
 ('W-114', 'C-110A', 57, 'Open'),  
 ('W-111', '116', 79, 'Open'),  
 ('W-111', 'C-110A', 57, 'Open'),  
 ('W-121', '116', 79, 'Open'),  
 ('W-121', 'C-110A', 57, 'Open'),  
 ('W-118', '116', 79, 'Open'),  
 ('W-118', 'C-110A', 57, 'Open'),  
 ('116', '117', 16, 'Open'),  
 ('116', 'W-100', 26, 'Open'),  
 ('116', 'W-110', 26, 'Open'),  
 ('116', '128', 26, 'Open'),  
 ('128', '140', 7, 'Open'),  
 ('128', '138', 8, 'Open'),  
 ('128', '214', 0, 'Open'),  
 ('128', '012', 0, 'Open'),  
 ('W-100', '141', 7, 'Open'),  
 ('W-100', '138', 8, 'Open'),  
 ('W-110', '140', 7, 'Open'),  
 ('W-110', '138', 8, 'Open'),  
 ('130', '121', 24, 'Open'),  
 ('121', '131', 11, 'Open'),  
 ('140', '141', 4, 'Open'),  
 ('140', 'W-100', 7, 'Open'),  
 ('140', 'W-110', 7, 'Open'),  
 ('140', '128', 7, 'Open'),  
 ('141', '117', 20, 'Open'),  
 ('117', '141', 16, 'Open'),  
 ('117', 'C-109', 6, 'Open'),  
 ('C-109', '117', 6, 'Open'),  
 ('134', 'SC-100', 17, 'Open'),  
 ('134', '108', 26, 'Open'),  
 ('134', '215', 0, 'Open'),  
 ('C-110A', 'C-015', 33, 'Open'),  
 ('C-110A', '114', 17, 'Open'),  
 ('SC-100', '136', 14, 'Open'),  
 ('127', '115', 43, 'Open'),  
 ('115', 'C-114', 11, 'Open'),  
 ('115', 'C-110', 17, 'Open'),  
 ('115', '003', 50, 'Open'),  
 ('C-110', '139', 14, 'Open'),  
 ('139', 'C-110', 14, 'Open'),  
 ('114', 'C-110', 10, 'Open'),  
 ('114', '113', 22, 'Open'),  
 ('113', 'E-121', 21, 'Open'),  
 ('113', '126', 43, 'Open'),  
 ('E-121', '114', 21, 'Open'),  
 ('112', '126', 20, 'Open'),  
 ('112', 'C-109', 26, 'Open'),  
 ('112', '111', 11, 'Open'),  
 ('111', 'C-100', 10, 'Open'),  
 ('111', '110', 37, 'Open'),  
 ('110', '131', 18, 'Open'),  
 ('110', '118', 17, 'Open'),  
 ('110', '109', 15, 'Open'),  
 ('110', '136', 35, 'Close'),  
 ('109', '005', 35, 'Open'),  
 ('109', '133', 34, 'Open'),  
 ('133', 'SC-100', 18, 'Open'),  
 ('108', '133', 9, 'Open'),  
 ('107', '108', 23, 'Open'),  
 ('107', 'E-100', 20, 'Open'),  
 ('107', '106', 23, 'Open'),  
 ('120', 'LB-200', 50, 'Open'),  
 ('120', '118', 14, 'Open'),  
 ('120', '135', 28, 'Open'),  
 ('120', '137', 18, 'Open'),  
 ('132', '120', 31, 'Open'),  
 ('119', '118', 13, 'Open'),  
 ('125', '119', 20, 'Open'),  
 ('125', 'E-101', 5, 'Open'),  
 ('E-105', '125', 9, 'Open'),  
 ('E-105', '004', 105, 'Open'),  
 ('135', '132', 3, 'Open'),  
 ('135', '216', 0, 'Open'),  
 ('217', '201', 37, 'Open'),  
 ('201', 'W-243', 8.5, 'Open'),  
 ('W-243', 'W-244', 9, 'Open'),  
 ('201', 'W-242', 8.5, 'Open'),  
 ('W-242', '202', 21, 'Open'),  
 ('202', '203', 29, 'Open'),  
 ('202', '214', 22, 'Open'),  
 ('203', '117', 100, 'Open'),  
 ('203', '218', 19, 'Open'),  
 ('218', '126', 43, 'Open'),  
 ('218', 'E-220', 18, 'Open'),  
 ('E-220', 'E-226', 30, 'Open'),  
 ('E-226', '207', 62, 'Open'),  
 ('207', '211', 15, 'Open'),  
 ('207', '216', 15.5, 'Open'),  
 ('216', '213', 10, 'Open'),  
 ('207', 'LB-200', 19.5, 'Open'),  
 ('LB-200', '209', 17, 'Open'),  
 ('209', '206', 29, 'Open'),  
 ('209', 'C-201', 10, 'Open'),  
 ('C-200', '205', 10, 'Open'),  
 ('204', 'C-203', 33, 'Open'),  
 ('204', 'C-201', 9, 'Open'),  
 ('C-203', '219', 3, 'Open'),  
 ('219', '220', 4, 'Open'),  
 ('220', '140', 40, 'Open'),  
 ('220', '214', 12, 'Open'),  
 ('214', '212', 10, 'Open'),  
 ('214', 'W-234', 6, 'Open'),  
 ('214', '302', 0, 'Open'),  
 ('214', 'W-221', 4, 'Open'),  
 ('204', '205', 25.5, 'Open'),  
 ('205', 'N-200', 13, 'Open'),  
 ('205', 'N-219', 13, 'Open'),  
 ('205', 'N-220', 13, 'Open'),  
 ('N-200', '210', 25.5, 'Open'),  
 ('C-200', '206', 16, 'Open'),  
 ('206', 'E-215', 19.5, 'Open'),  
 ('E-215', '208', 17, 'Open'),  
 ('C-200', '215', 8.5, 'Open'),  
 ('208', '106', 75, 'Close'),  
 ('215', '208', 8.5, 'Open'),  
 ('215', '401', 0, 'Open'),  
 ('W-311', '302', 11, 'Open'),  
 ('302', 'W-300', 11, 'Open'),  
 ('W-311', '219', 85, 'Open'),  
 ('302', '219', 85, 'Open'),  
 ('401', '402', 10, 'Open'),  
 ('402', 'N-400', 17, 'Open'),  
 ('402', 'N-408', 16, 'Open'),  
 ('N-408', 'N-407', 7, 'Open')]

* **List** has been used to store all the tuples which are all the edges. It is also an ordered data type and uses index for accessing elements and is mutable meaning that its elements can be changed. The resulting un-directed adjacency list has also been displayed by this data type.

**DIJKSTRA:**

Now the basic objective was to find the shortest path and for this we used the DIJKSTRA algorithm to first find the distance of every node from the source node:

V = Nodes

E = Edges

G= adjacency list or map

d = list including distance from parent

p = list having parent from the specific shortest path

s= source

visited = list containing visited nodes in the path

w (u,v) = weight of edge between u and v

q= priority queue having distance of all nodes from source

Initialization(G,s):

In this step, we assign infinity as distance between source and all other nodes and the distance of source from itself as zero. Next we assign a nil parent to every node.

for each element v in V

d[v] = infinity

p[v] = Nil

d[s] = 0

RELAX(u,v,w):

In this part, we start traversing from source node and for each successive node we check whether the distance from source to this node is less than the distance already assigned and if it is less we update both the distance and parent of the node.

if d[v] > d[u] +w (u,v)

d[v] = d[u] + w(u,v)

p[v] = u

DIJKSTRA(G,w,s):

In the main function, we call initialization and then form a priority queue saving distances of nodes from source and then using that decide the next possible node so that we are at a minimum distance from the source

Initialization (G, s)

visited = []

q = nodes in priority series depending on the distance

while q not empty:

u = Extract minimum from q

visited. append (u)…... (add u to visited)

for each vertex v element of G[u]:

RELAX (u,v,w)

Now that we have the distances of every node from the source node we need to display the path from the source to the destination so for this we use the shortest path function:

Shortest Path:

What this does is that it starts from the destination node and then using the dictionary containing parents of nodes and distances of nodes from their parents, backtracks to the source and in the process stores all the edges and in the end then returns it thus, displaying the path from the source to destination.

path = []

parents = dictionary containing parents of nodes

distances = dictionary containing distances of nodes

while source!=destination

parent=parents[destination]

distance = distances[destination]

add (parent, destination, distance) to path

destination = parent

The python code of both DIJKSTRA and Shortest Path is attached in the code folder and sample outputs are given afterwards under results.

**Results or Sample Outputs:**

1) Going from Cassim Computing Lab(E-105) to male prayer room(N-407):

('E-105', '125', 9)

('125', '119', 29)

('119', '118', 42)

('118', '110', 59)

('110', '109', 74)

('109', '133', 108)

('133', '108', 117)

('108', '134', 143)

('134', '215', 143)

('215', '401', 143)

('401', '402', 153)

('402', 'N-408', 169)

('N-408', 'N-407', 176)

2) Going from Dhaba (007) to main entrance (006):

('007', '003', 21)

('003', 'E-017', 42)

('E-017', 'E-013', 50)

('E-013', 'C-007', 94)

('C-007', 'C-004', 115)

('C-004', 'C-001', 136)

('C-001', '005', 152)

('005', '109', 187)

('109', '133', 221)

('133', '108', 230)

('108', '107', 253)

('107', '106', 276)

**Challenges:**

Now there was a problem that how we were going to connect the different floors so what was done was that we identified the connections between the floors through the stairs and the lifts. The stairs distance was measured, and a node identified on the bottom floor was connected through the stairs to a node on the other floor. Likewise, we connected the lifts on different nodes keeping the distance between them zero.

('128', '214', 0, 'Open')

So in this edge, the Lift on ground floor is connected to the lift on first floor with zero weight.

Another problem was regarding the codes assigned to the intersection nodes. So, we formulated a scheme of forming the codes and so in any code we set the first digit according to the floor we were on and if the node was on any main street, we named it after that. The same was done for washrooms or lifts on different nodes. For example:

'007':'Dhaba’

In this node, it is shown that the code ‘007’ belongs to ‘Dhaba’ and the initial first zero is indication it is on the lower ground floor.

'110':'CS4'

In this node, the code shows it is on the ground floor and the name is also an acronym stating that it is the 4th node on central street.

'137':'Washroom12

This node is also on the ground floor and the name indicates that washroom numbered two on the ground floor.

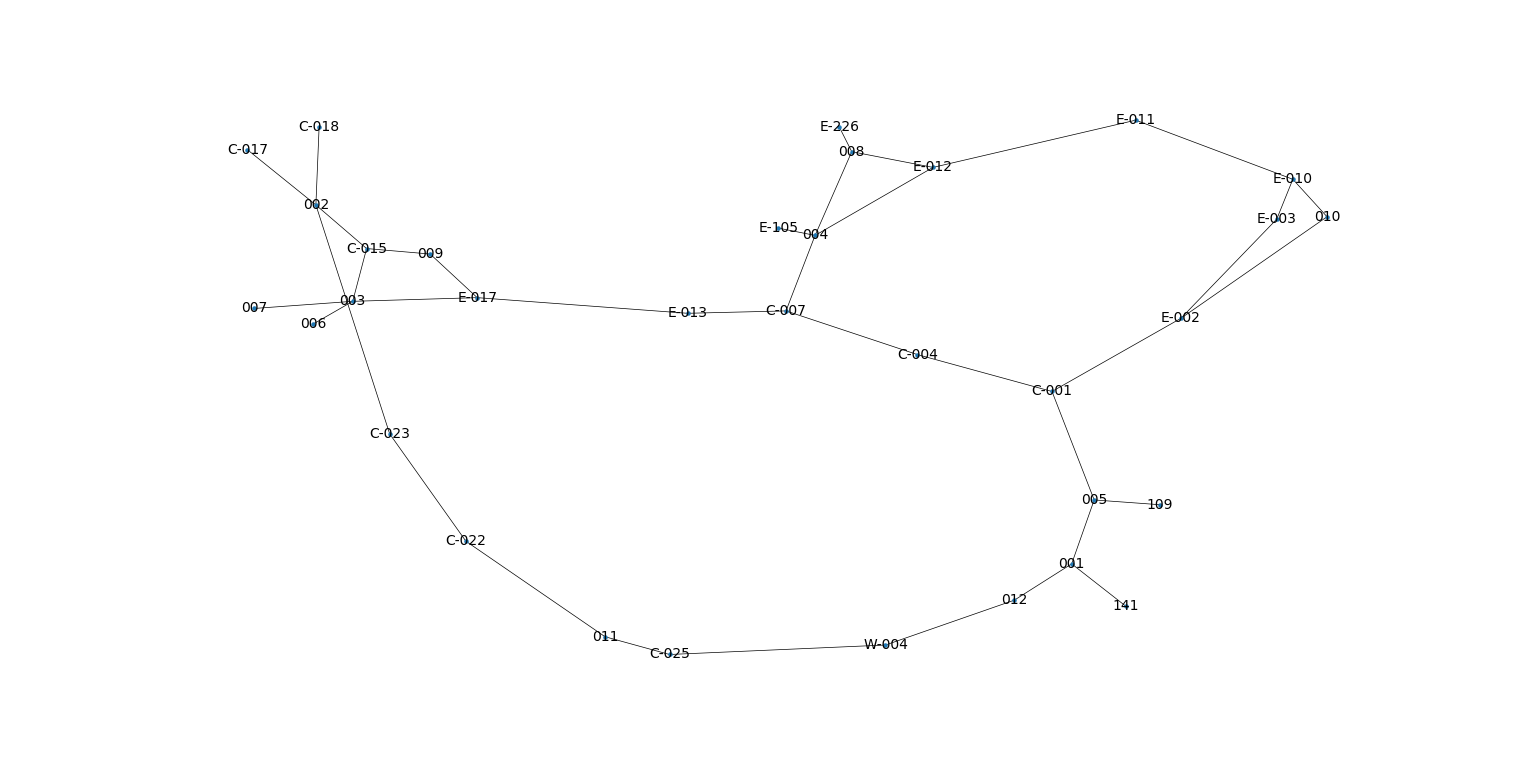
'215':'Lift22'

This node is on the first floor and in the name the numbers 22 indicate that it is on second floor and numbered second.

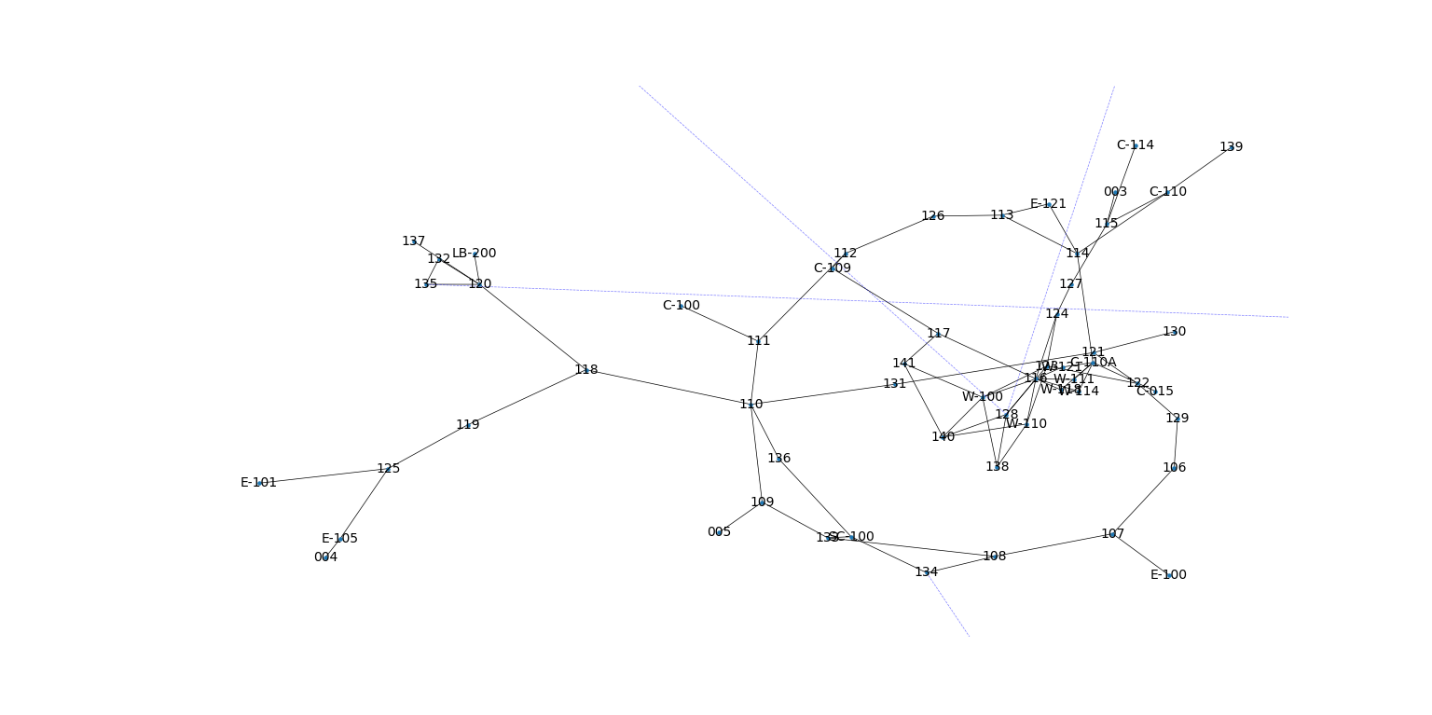
**GRAPH PLOT:**

Other than manual handmade graph, we used networkx and matplotlib to plot the computerized graphs. In this domain we used pip to install networkx and matplotlib. Pip is a python tool for installing and managing packages. Our source was github’s networkx weighted graphs. In this graph dotted blue lines represent zero distance between floors connected through lifts while black lines represent weighted connections. Below are the images of graphs of different floors:

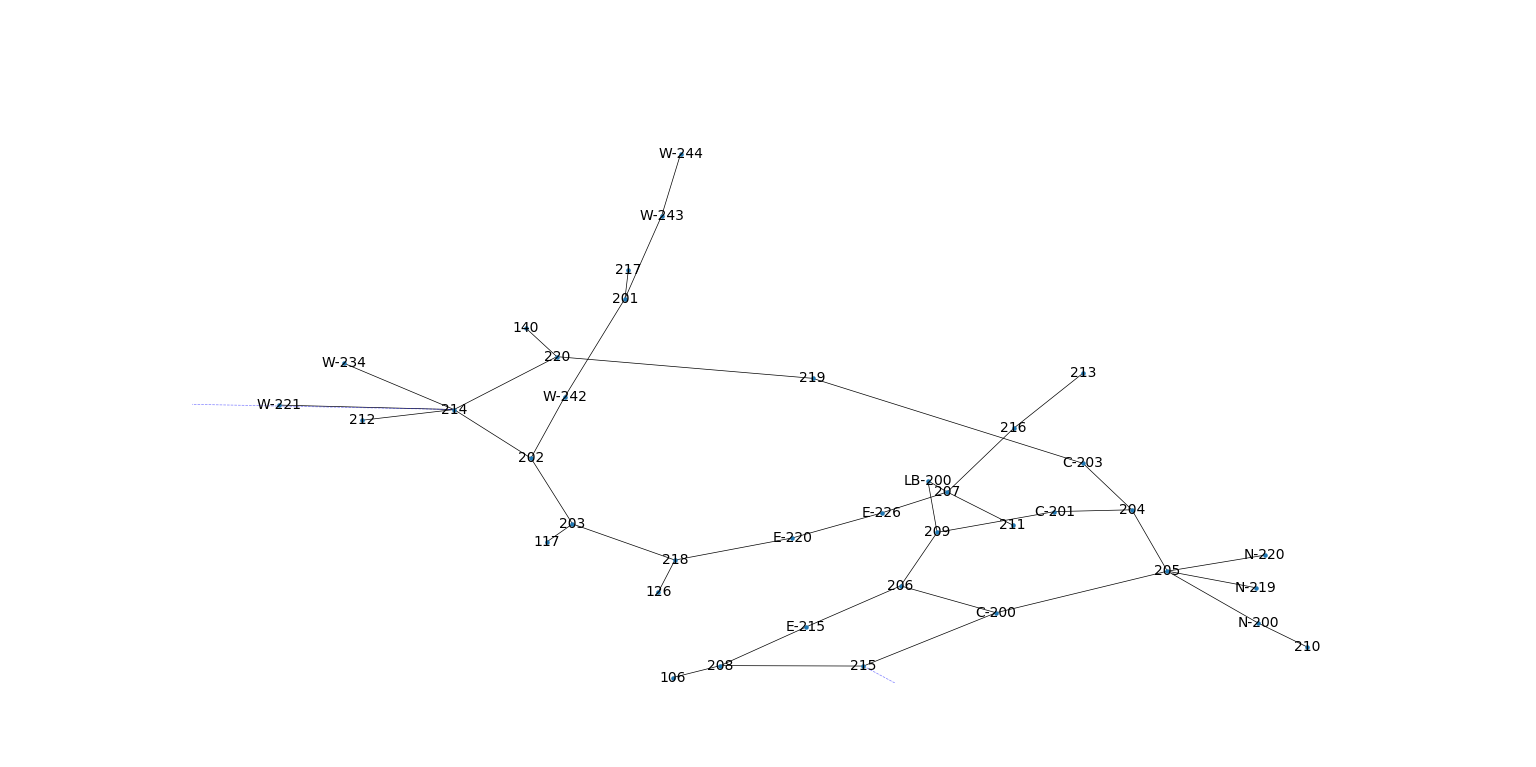
Lower Ground Floor:



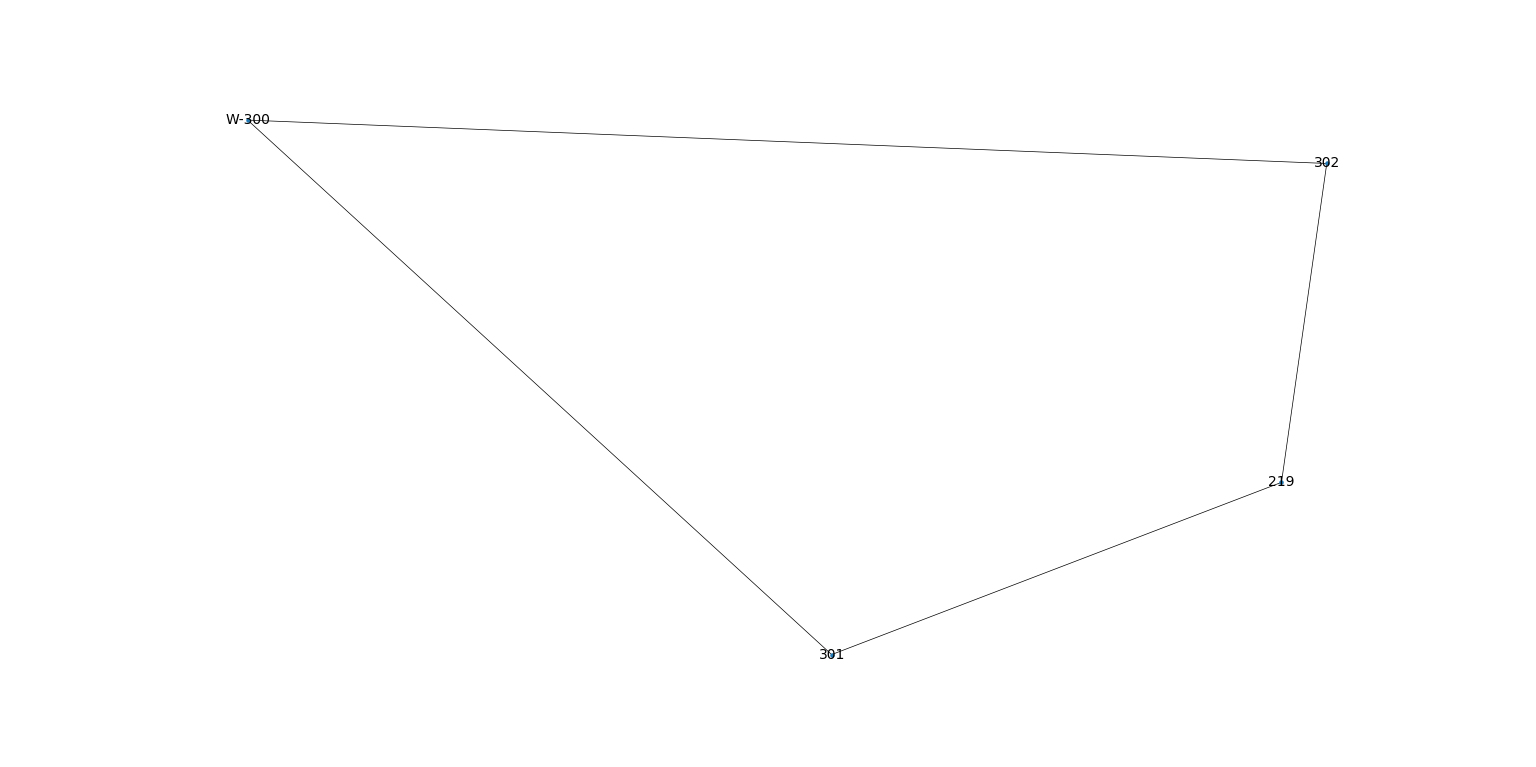
Ground Floor:



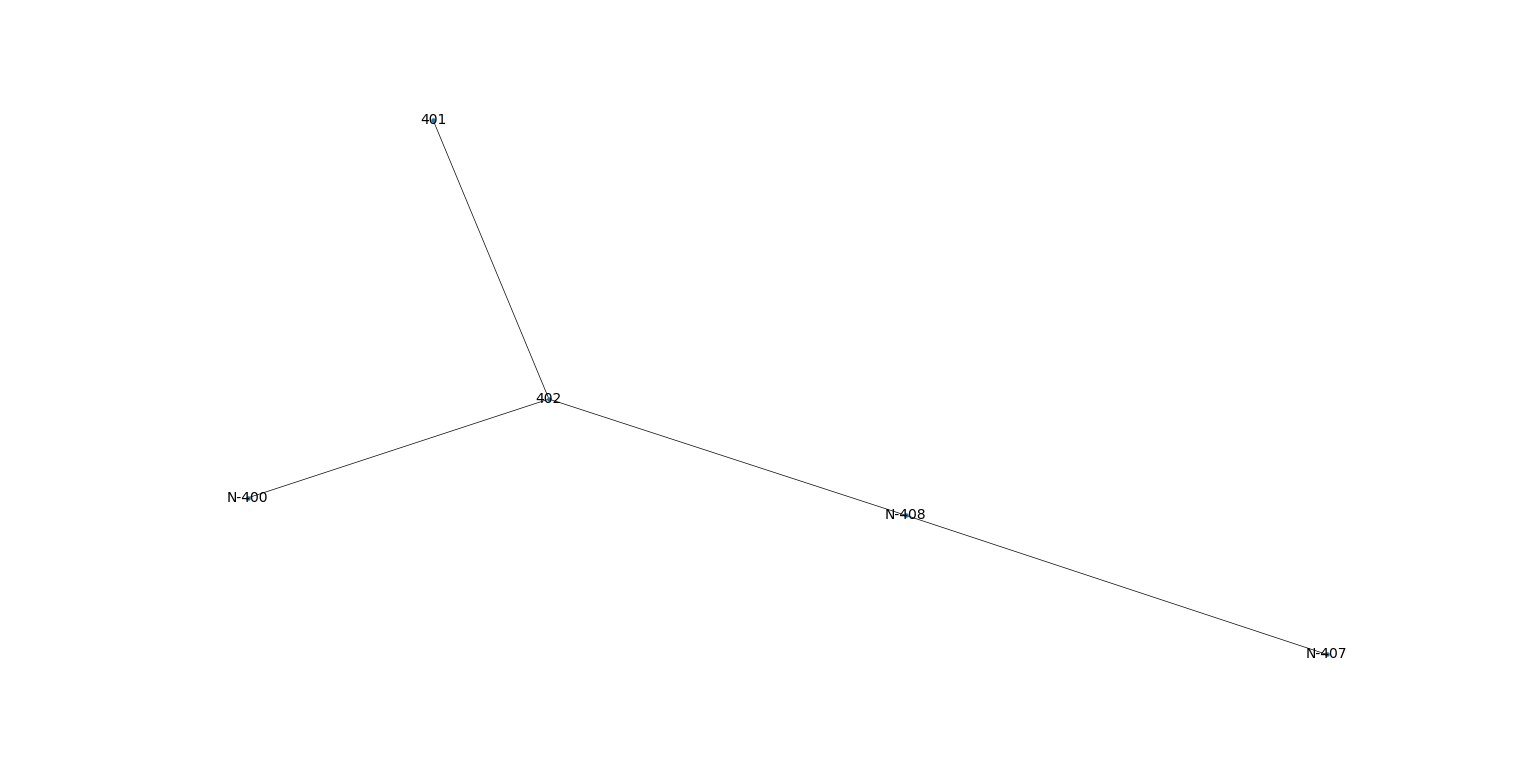
1st Floor:



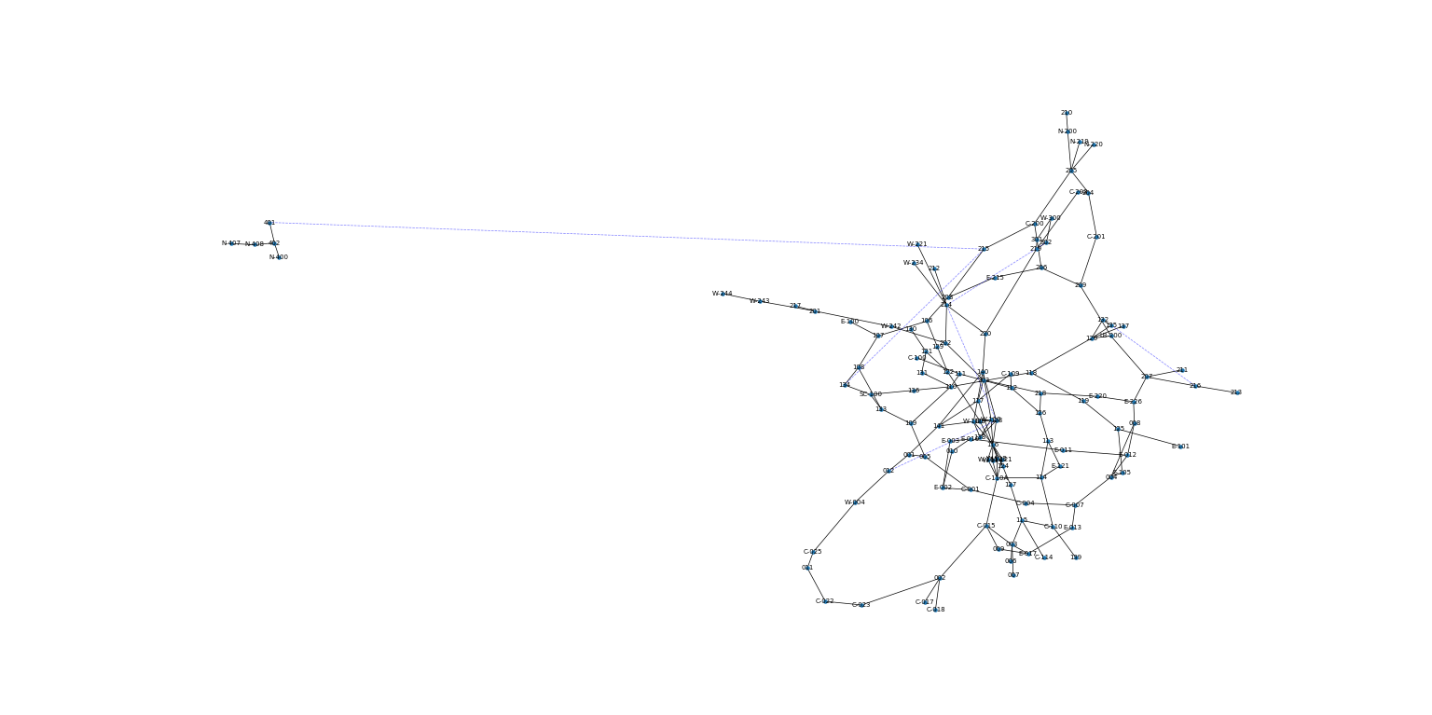
2nd Floor:



3rd Floor:



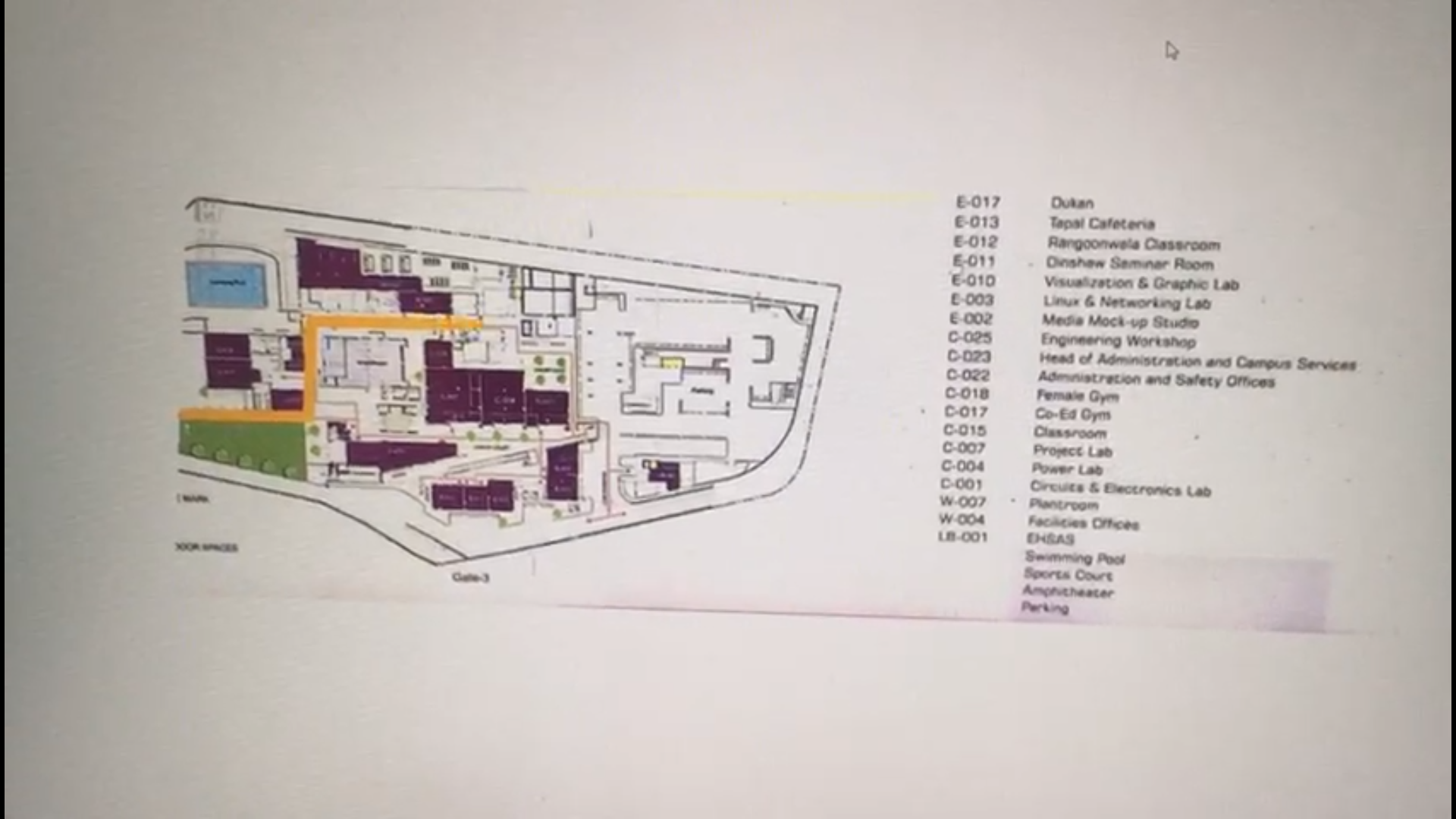
All Floors (complete graph):

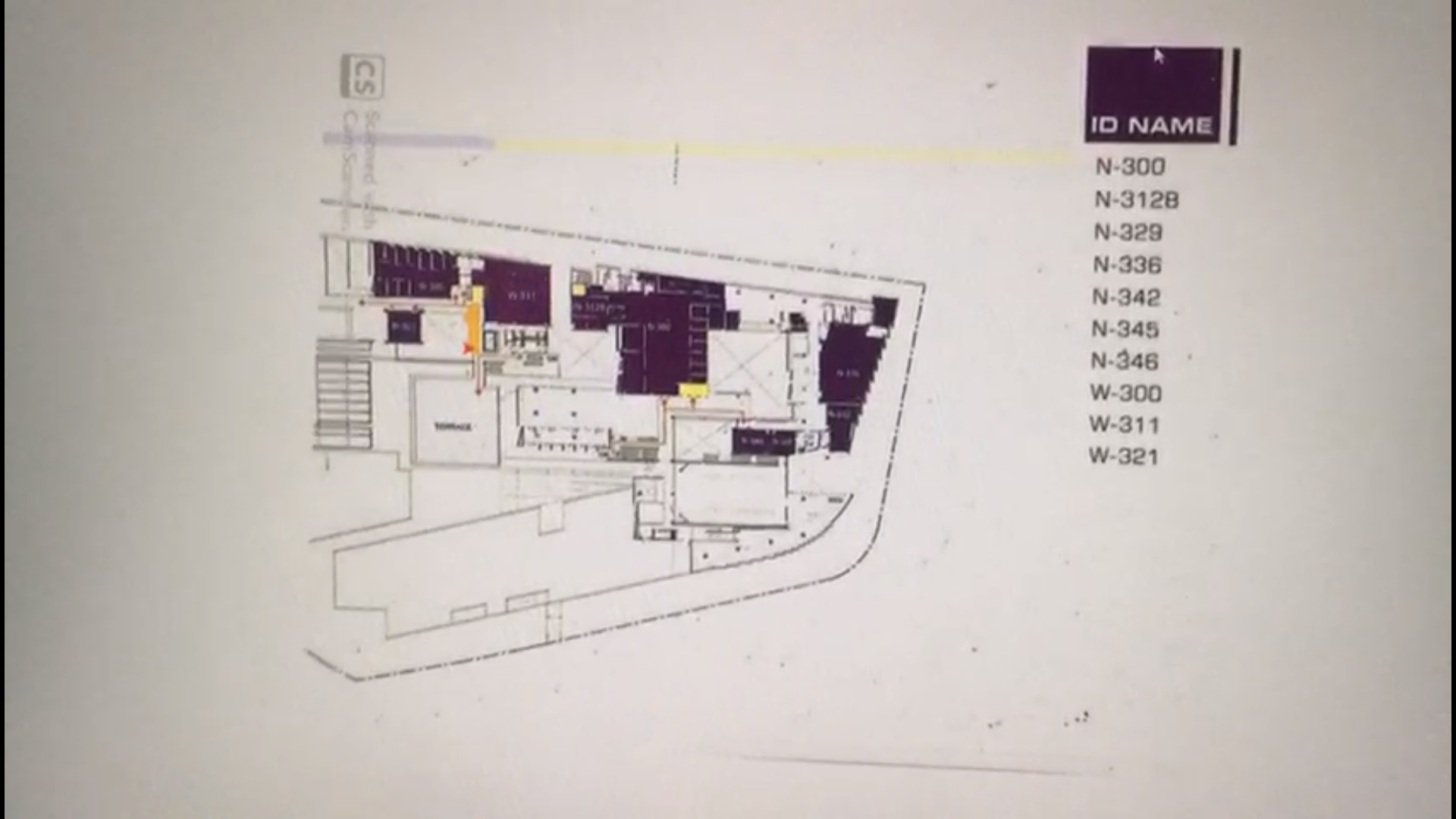


**TURTLE GUI:**

We used turtle library for the Graphical User Interface (GUI) of our project. The background of our GUI was nothing else but the university’s own map which makes it very easy to understand the path and the directions. We only covered lower ground floor and the 2nd floor for this purpose, other floors could also be accumulated but this would require far more data entry. The turtle setposition() function was used to get the coordinates. These coordinated were stored in the coordinates dictionary. Our dijkstra (shortest path) printed directions for the turtle to move. Initially we were confused that how could we manage the z-axis coordinates in order to link the floors but we were able to manage this and our floor background gets changed when we move from lower ground floor to the 2nd floor. In our case we hard coded the condition if we move from lower ground floor to the 2nd floor but we can set simple condition to check the floor level and add more floors because our nodes are made in way that they keep record of the floors. Lower ground floor nodes start from: 0, ground floor nodes start from: 1, 1st floor nodes start from: 2, 2nd floor nodes start from: 3, 3rd floor nodes start from: 4. The other nodes which were by default defined by the university also keep check of floors. We could simply split these nodes and get the floor level and our nodes starting digit showed the floor level. Therefore, it was easy to make conditions for the change of floors.

Following is an example of our turtle going from sport courts (lower ground floor) to the projects lab on 2nd floor.





**Conclusion:**

The aim of our project was to find the shortest path in terms of footsteps between any two points within the Habib University. We successfully completed this job and showed the results of our program to give the number of footsteps as well as the route between different locations of our university. We also included the opened and closed path option which was working correctly in a way that our program showed different path even for the directly adjacent node if the path was temporarily closed by the university for any purpose. Initially GUI was kept as optional but we successfully made it in a very clear manner because we used the university’s own map for this purpose. We made the computerized graphs and the turtle GUI.

The work done in moving from one point to other one and the directions to move left or right were not the initial scope of our project but obviously these things and other betterments can be added in our project as a further improvement in the future.