

University of Texas at Arlington
Department of Computer Science



**Designing University of Texas at Arlington's map on a GIS
and Spatial database systems**

MAP SECTION: J

TEAM NO: 10

Aditya Kore (1001523018)

Heera Sridhar (1001529609)

OUTLINE

In the first phase of the project, UTA campus map had to be edit/improved. This map included many of the buildings for colleges, departments, apartments, residential halls, different playgrounds and administrative offices. First the KML files were extracted from Google maps and converted them to shapefiles so that the shapefiles can be used in by the available GIs and Spatial Database tools such as QGIS, ArcGIS, PostGIS and SpatiaLite. To create a map, we created spatial objects like lines, points and polygons using google maps. As these maps were created previously, we had to improve and correct the data files that were previously created by students taking this course. In particular, the paths/roads from different sections on the map did not always meet at the same point in the current files, so this is one aspect where the KML files were fixed. Another aspect we fixed is to check that all building entrances are correctly entered, as well as building names and building types for each section of the map. After correcting the map, three types of queries were made:

- Choose a building (by choosing it from a list (pull-down menu) of building names), and display the building requested in a highlighted color on the map.
- Choose a building and a distance and return the names of all buildings or spaces of a particular type within that distance from the building (for example, find all "parking lot" type spaces within 2 miles from ERB building).
- Choose two points (building entrances) using pull-down menus (for example ERB entrance 1 and Davis Hall entrance 1) and calculate and return a path that is the shortest path between the two buildings.

INTRODUCTION

The purpose of this project was to create a small scale Geographic Information System on top of a custom built map of the UTA campus. The system would involve displaying the map on QGIS, performing spatial queries on it via python scripts and displaying the results onto the map. For other operations such as database fetch/insert and calculating shortest path, in-built plug-ins are used. The workflow is outlined below and in the subsequent sections, each step is described in detail.

The high-level description of the project:

Area J in UTA Map consists of Brazos house, Thermal Plant, UTA planetarium, Health Center, University Center, College Hall, Arlington Hall, University college. In this part of UTA Map the spatial data sets are Line, Points and Polygons. In this part of Map there are no Spaces, So we ignored the Spaces dataset.

IMPROVED KML:

Firstly we had to make changes to the co-ordinates of some Polygons, Since the co-ordinates were not properly aligned. We changed the co-ordinates of UTA Planetarium, University Center and Thermal plant. Apart from this, added connections between entrance and the Main Path. Added new lines between the entrance and Lines. For some polygons the coordinates were fixed and added New entrances.

Also added Spatial attributes for Lines, Polygons and Points.

Added Custom data to KML by using <ExtendedData> element :

This <ExtendedData> element provides:

- <Data> element - allows you to add untyped name/value pairs to the user data associated with a given Feature (NetworkLink, Placemark, GroundOverlay, PhotoOverlay, ScreenOverlay, Document, or Folder). These name/value pairs are displayed in the balloon by default. This information can also be used for entity replacement in the <text> element of <BalloonStyle>. (Using the BalloonStyle Element as a Template.)
- <Schema> and <SchemaData> elements - allow you to add typed data to the user data associated with a given Feature.
- Arbitrary XML data - allows you to preserve user data within a KML file. Google Earth passes this data along with the file and saves it, but does not use it.

<ExtendedData> Element for polygons:

```
<ExtendedData>
<Data name="B_ID">
<value>.... </value>
</Data>
<Data name="Name">
<value>.....</value>
</Data>
<Data name="Btype">
<value>.....</value>
</Data>
<Data name="Geometry">
<value>.....</value>
</Data>
</ExtendedData>
```

<ExtendedData> Element for Lines:

```
<ExtendedData>
<Data name="L-ID">
<value>.... </value>
```

```

</Data>
<Data name="Name">
<value>.....</value>
</Data>
<Data name="From">
<value>.....</value>
</Data>
<Data name="To">
<value>.....</value>
</Data>
<Data name="Geometry">
<value>.....</value>
</Data>
</ExtendedData>

```

<ExtendedData> Element for Points:

```

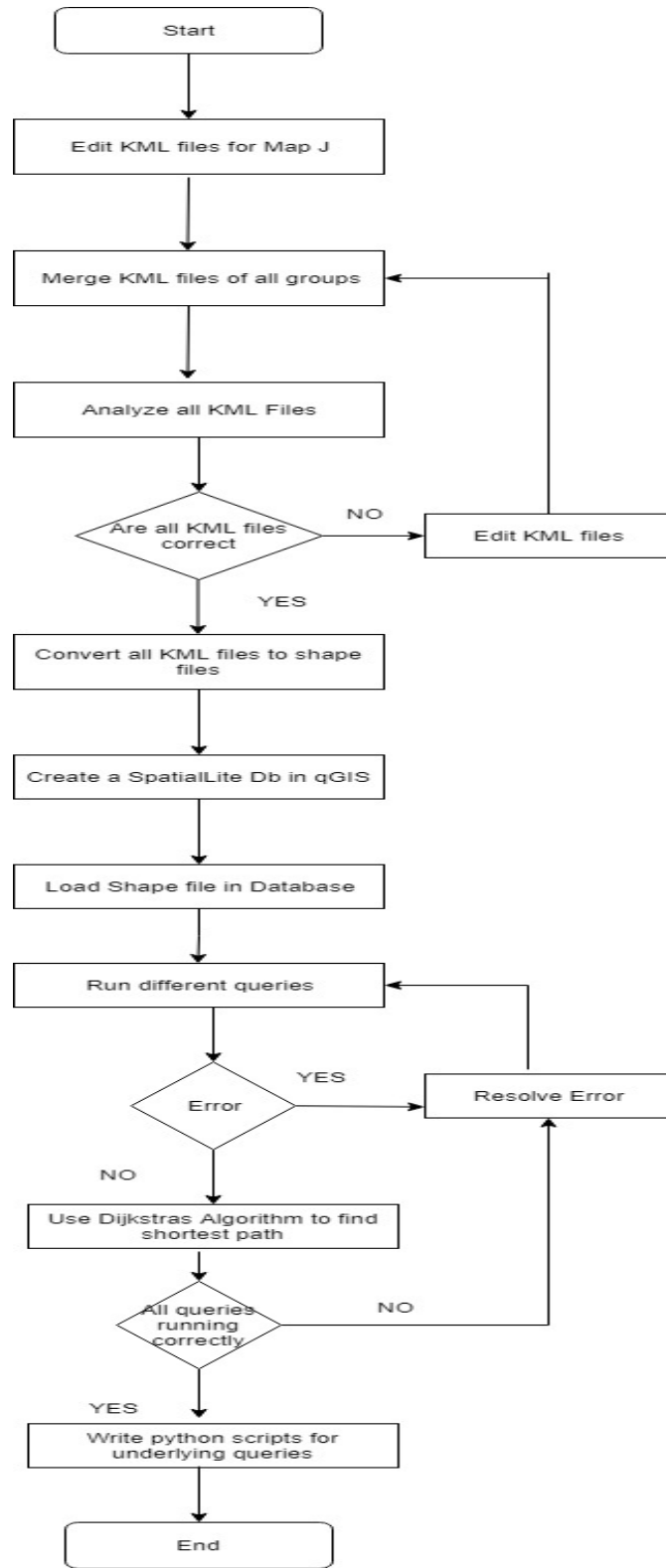
<ExtendedData>
<Data name="P_ID">
<value>....</value>
</Data>
<Data name="Name">
<value>.....</value>
</Data>
<Data name="Description">
<value>.....</value>
</Data>
<Data name="Type">
<value>.....</value>
</Data>
<Data name="Geometry">
<value>.....</value>
</Data>
</ExtendedData>

```

In this fashion we had added ExtendedData Element to all the points, Lines and Polygons in the KML files respectively.

- In the second phase of the project, all the KML files were converted to shape files and Stored the data in SpatialLite (or any spatial database), and built a Python GUI to query SpatialLite. After getting the result of the query, displayed the result using QGIS.

FLOWCHART



SYSTEM AND SOFTWARE USED

The tools used for the system are:

1. Google Earth [1]
2. QGIS [2]
3. Python programming Language
4. SpatiaLite database [3]
5. ArcGIS Pro

Google Earth [1]

Google Earth is a computer program that renders a 3D representation of Earth based primarily on satellite imagery. The program maps the Earth by superimposing satellite images, aerial photography, and GIS data onto a 3D globe, allowing users to see cities and landscapes from various angles. Users can explore the globe by entering addresses and coordinates, or by using a keyboard or mouse. Users may use the program to add their own data using Keyhole Markup Language and upload them through various sources, such as forums or blogs. Google Earth is able to show various kinds of images overlaid on the surface of the earth and is also a Web Map Service client.

QGIS [2]

QGIS functions as geographic information system (GIS) software, allowing users to analyze and edit spatial information, in addition to composing and exporting graphical maps. QGIS supports both raster and vector layers; vector data is stored as either point, line, or polygon features. Multiple formats of raster images are supported, and the software can georeference images. QGIS supports shapefiles, coverages, personal geodatabases, dxf, MapInfo, PostGIS, and other formats. Web services, including Web Map Service and Web Feature Service, are also supported to allow use of data from external sources. QGIS integrates with other open-source GIS packages, including PostGIS, GRASS GIS, and MapServer. Plugins written in Python or C++ extend QGIS's capabilities. Plugins can geocode using the Google Geocoding API, perform geoprocessing functions similar to those of the standard tools found in ArcGIS, and interface with PostgreSQL/PostGIS, SpatiaLite and MySQL databases.

SpatiaLite Database [3]

SpatiaLite is a spatial extension to SQLite, providing vector geodatabase functionality. It is similar to PostGIS, Oracle Spatial, and SQL Server with spatial extensions, although SQLite/SpatiaLite aren't based on client-server architecture: they adopt a simpler personal architecture. i.e. the whole SQL engine is directly embedded within the application itself: a complete database simply is an ordinary file which can be freely copied (or even deleted) and transferred from one computer/OS to a different one without any special precaution.

ArcGIS Pro

ArcGIS is a geographic information system (GIS) for working with maps and geographic information. It is used for creating and using maps, compiling geographic data, analyzing mapped information, sharing and discovering geographic information, using maps and geographic information in a range of applications, and managing geographic information in a database. The system provides an infrastructure for making maps and geographic information available throughout an organization, across a community, and openly on the Web.

DESCRIPTION OF ALGORITHM USED FOR ANSWERING QUERIES

1. Choose a building (by choosing it from a list (pull-down menu) of building names), and display the building requested in a highlighted color on the map.
 - a. First, we create view of a layer and then we use the select query to get the building. In the python code the query dynamically takes the layer and building name.
Query used in the code: **'create view '+layer_name+' as select * from UTA_final_polygon where Name="'+building_name+''**
 - b. We then insert the view in geometry_columns which convert the view in to in the geometry layer, which can be seen in qgis with the name as layer_name.
Query used in the code: **'INSERT INTO geometry_columns (f_table_name, f_geometry_column, geometry_type, coord_dimension, srid, spatial_index_enabled) VALUES (''+layer_name+'', "geometry", 3, 2, 0, 0)'**

2. Choose a building and a distance and return the names of all buildings or spaces of a particular type within that distance from the building (for example, find all "parking lot" type spaces within 500 meters from ERB building).

- a. In this query we use the spatialite function `PtDistWithin` with the arguments as geometry of buildings and the distance in meters. Here we select a building and then types of building (example parking lot) within a given distance.

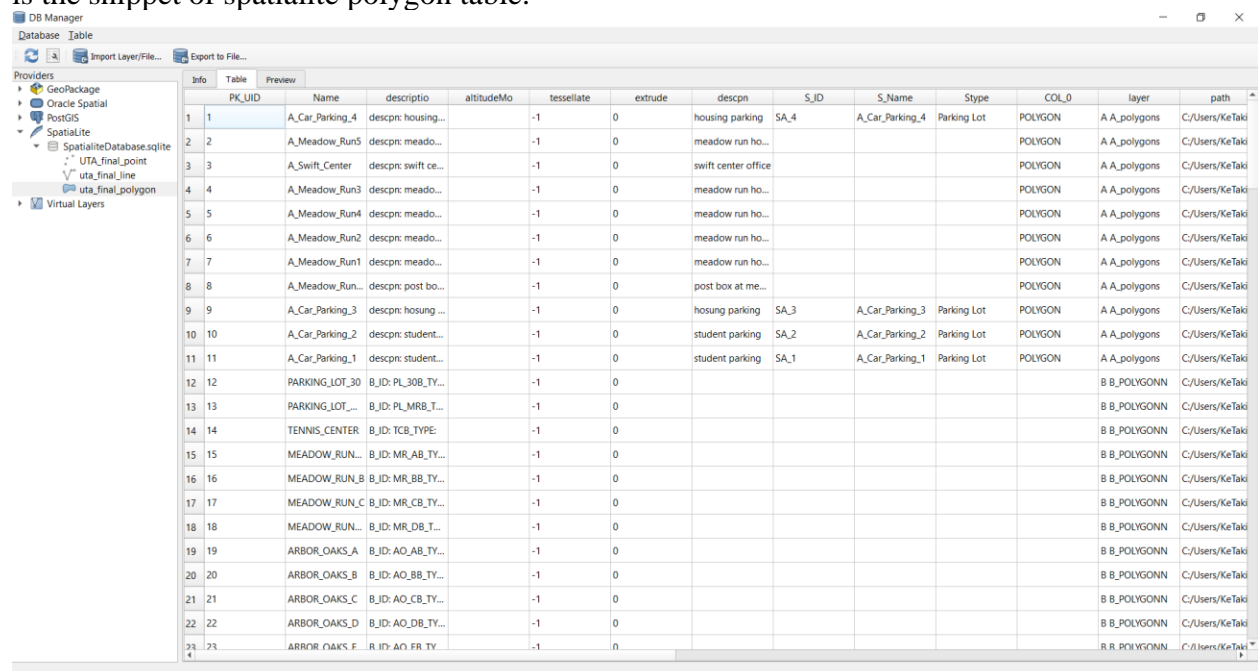
Query used in the code: `'create view '+layer_name+ ' as select n.name,n.geometry from uta_final_polygon as n, uta_final_polygon s where s.Name ='''+building_name+'''' and n.Btype='''+building_type+'''' and PtDistWithin(Centroid(n.geometry),Centroid(s.Geometry),' +distance+')`

- b. We then insert the view in `geometry_columns` which convert the view in to in the geometry layer, which can be seen in qgis with the name as `layer_name`.

Query used in the code: `'INSERT INTO geometry_columns (f_table_name, f_geometry_column, geometry_type, coord_dimension, srid, spatial_index_enabled) VALUES (''+layer_name+'', "geometry", 3, 2, 0, 0)'`

EXECUTION STEPS FOR EACH QUERY

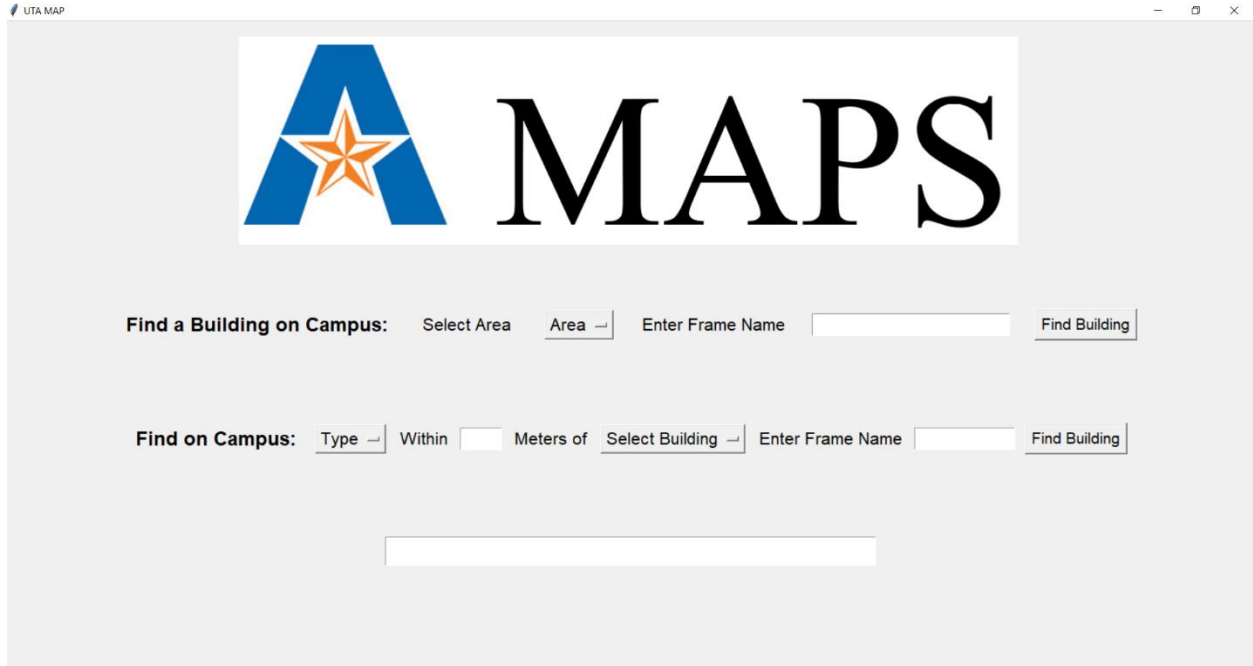
1. Loading the shape files into Spatialite database: Create a Spatialite database with the help of a plugin in QGIS and all three shape files were imported as 3 tables in the database. Following is the snippet of spatialite polygon table.



Info	Table	Preview
PK_UID	Name	descriptio
1	A_Car_Parking_4	descpn: housing...
2	A_Meadow_Run5	descpn: meado...
3	A_Swift_Center	descpn: swift ce...
4	A_Meadow_Run3	descpn: meado...
5	A_Meadow_Run4	descpn: meado...
6	A_Meadow_Run2	descpn: meado...
7	A_Meadow_Run1	descpn: meado...
8	A_Meadow_Run...	descpn: post bo...
9	A_Car_Parking_3	descpn: hosung ...
10	A_Car_Parking_2	descpn: student...
11	A_Car_Parking_1	descpn: student...
12	PARKING_LOT_30	B_ID: PL_30B_TY...
13	PARKING_LOT...	B_ID: PL_MRB_T...
14	TENNIS_CENTER	B_ID: TCB_TYPE...
15	MEADOW_RUN...	B_ID: MR_AB_TY...
16	MEADOW_RUN_B	B_ID: MR_BB_TY...
17	MEADOW_RUN_C	B_ID: MR_CB_TY...
18	MEADOW_RUN...	B_ID: MR_DB_T...
19	ARBOR_OAKS_A	B_ID: AO_AB_TY...
20	ARBOR_OAKS_B	B_ID: AO_BB_TY...
21	ARBOR_OAKS_C	B_ID: AO_CB_TY...
22	ARBOR_OAKS_D	B_ID: AO_DB_TY...
23	ARBOR_OAKS_F	B_ID: AO_FB_TY...

2. Running queries on SpatiaLite: Test the SpatiaLite database to check if all the queries run smoothly. One example query and its results:

a. User Interface



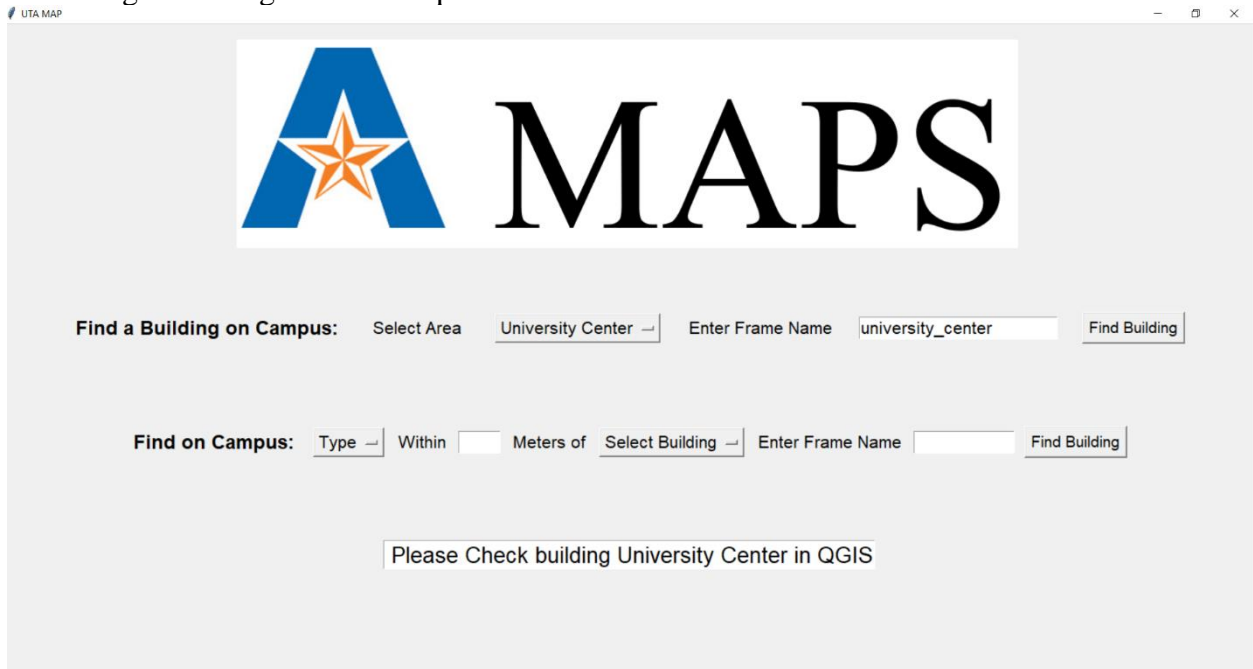
UTA MAP

A MAPS

Find a Building on Campus: Select Area Enter Frame Name

Find on Campus: Within Meters of Enter Frame Name

b. Finding a building on UTA map:



UTA MAP


A MAPS

Find a Building on Campus: Select Area Enter Frame Name

Find on Campus: Within Meters of Enter Frame Name

Please Check building University Center in QGIS

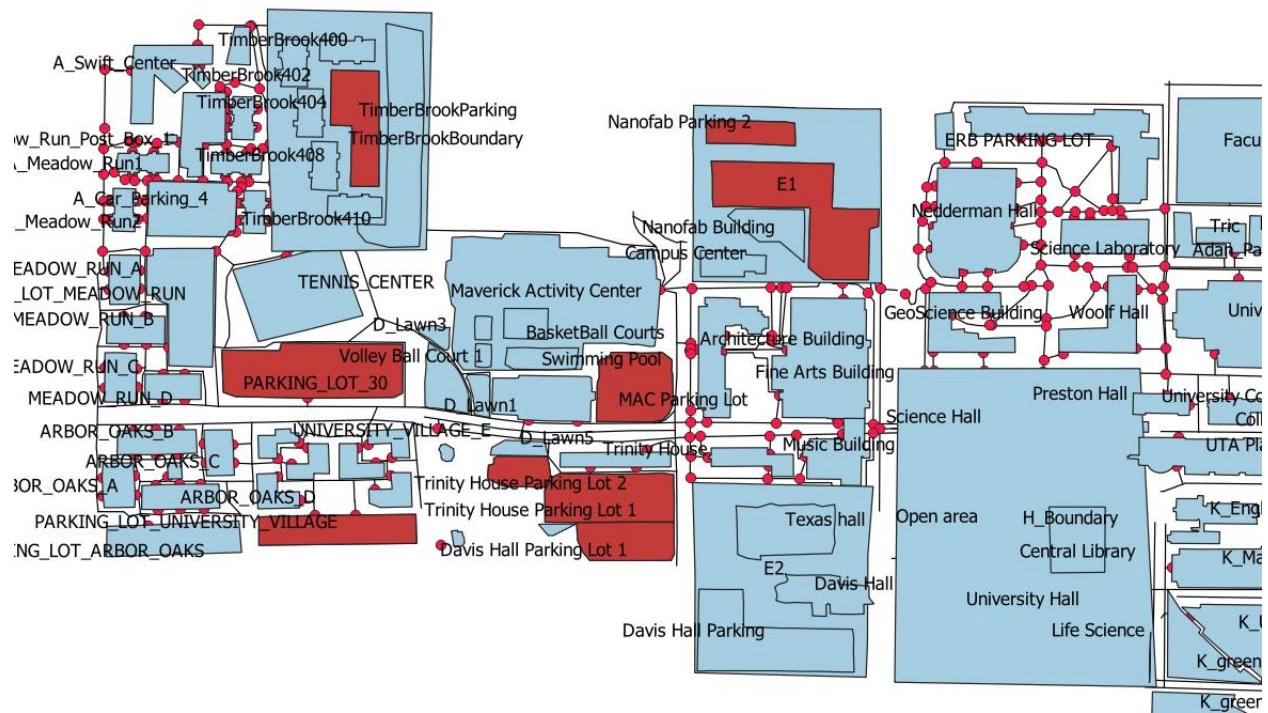
UTA MAP



Find a Building on Campus:

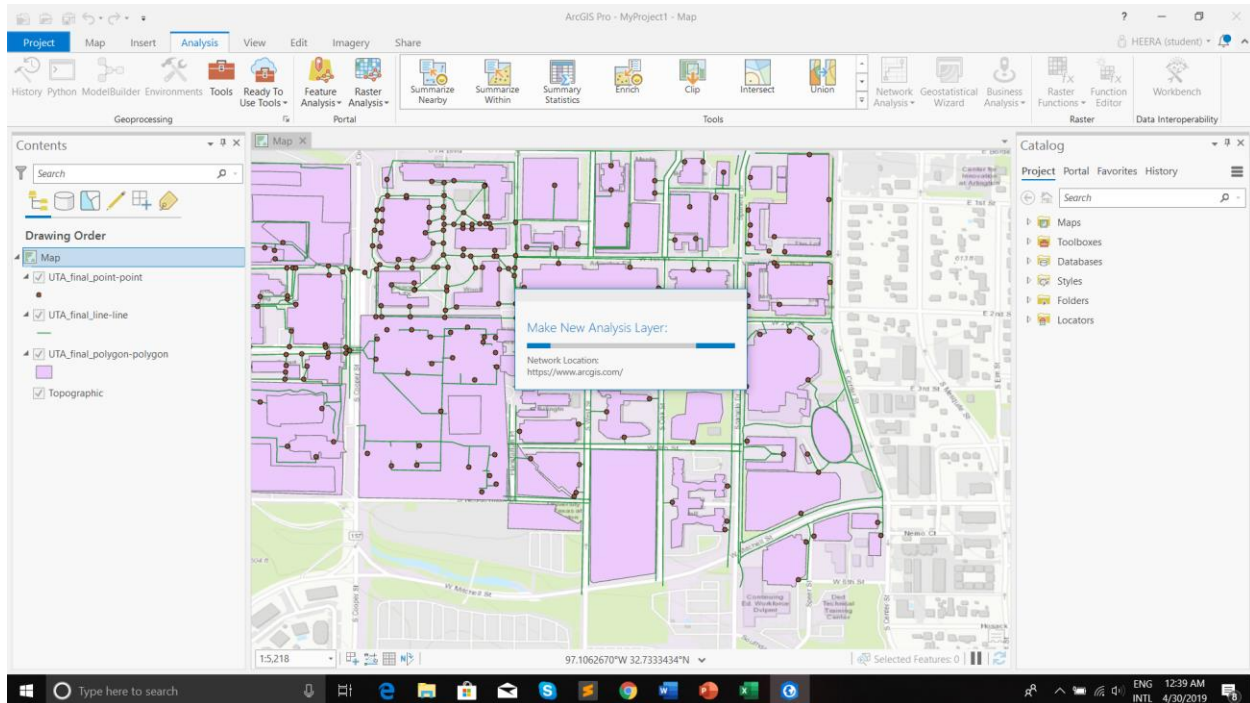
Find on Campus:

e) Result in QGIS:

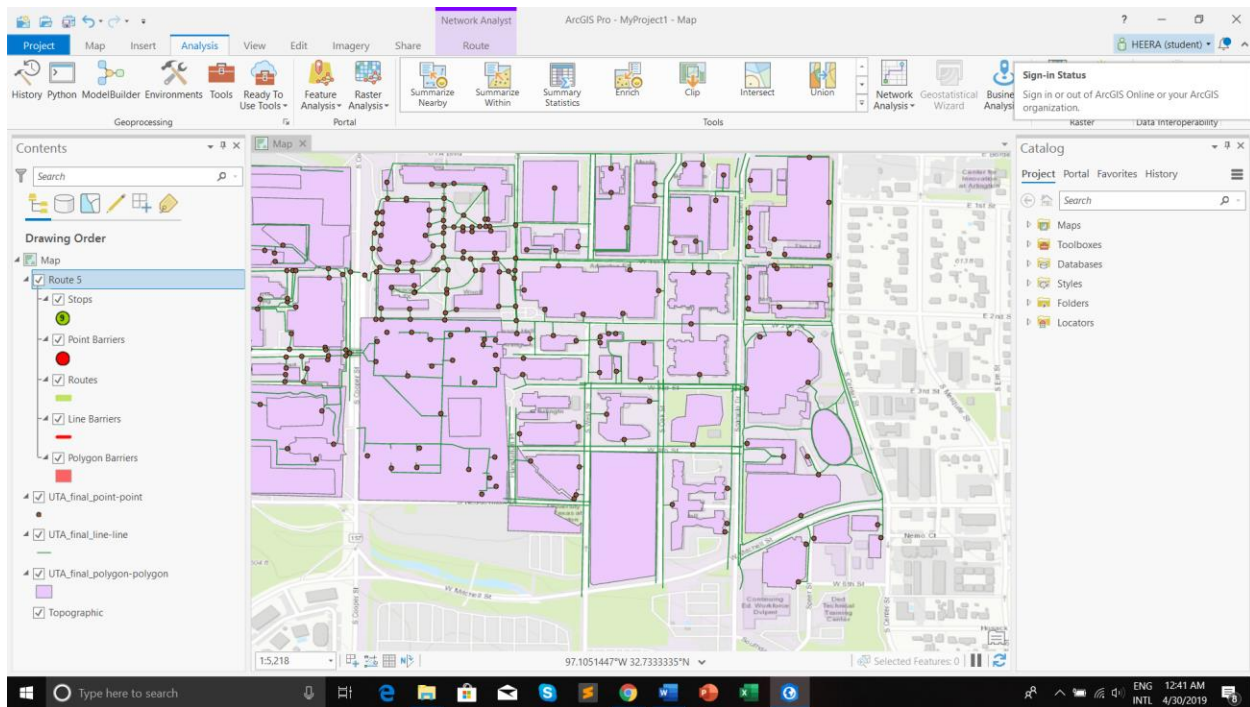


3. Choose two points (building entrances) using pull-down menus (for example ERB entrance 1 and Davis Hall entrance 1) and calculate and return a path that is the shortest path between the two buildings.

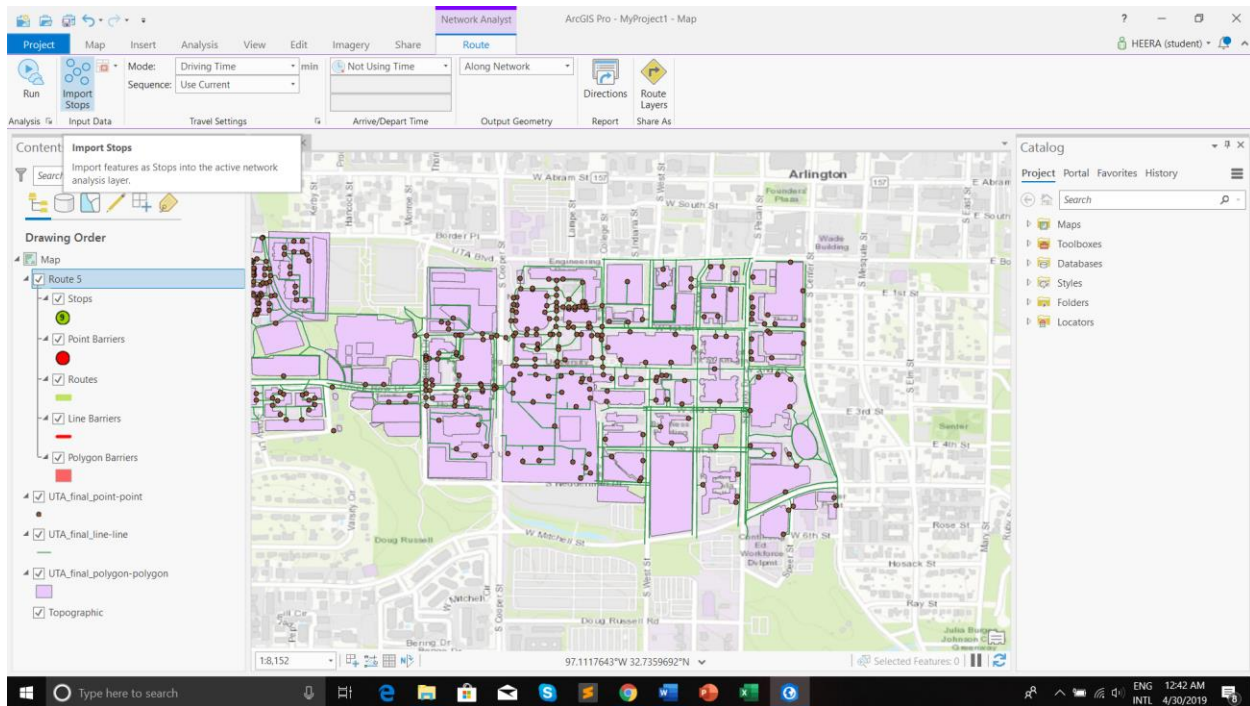
Click on network analysis tab



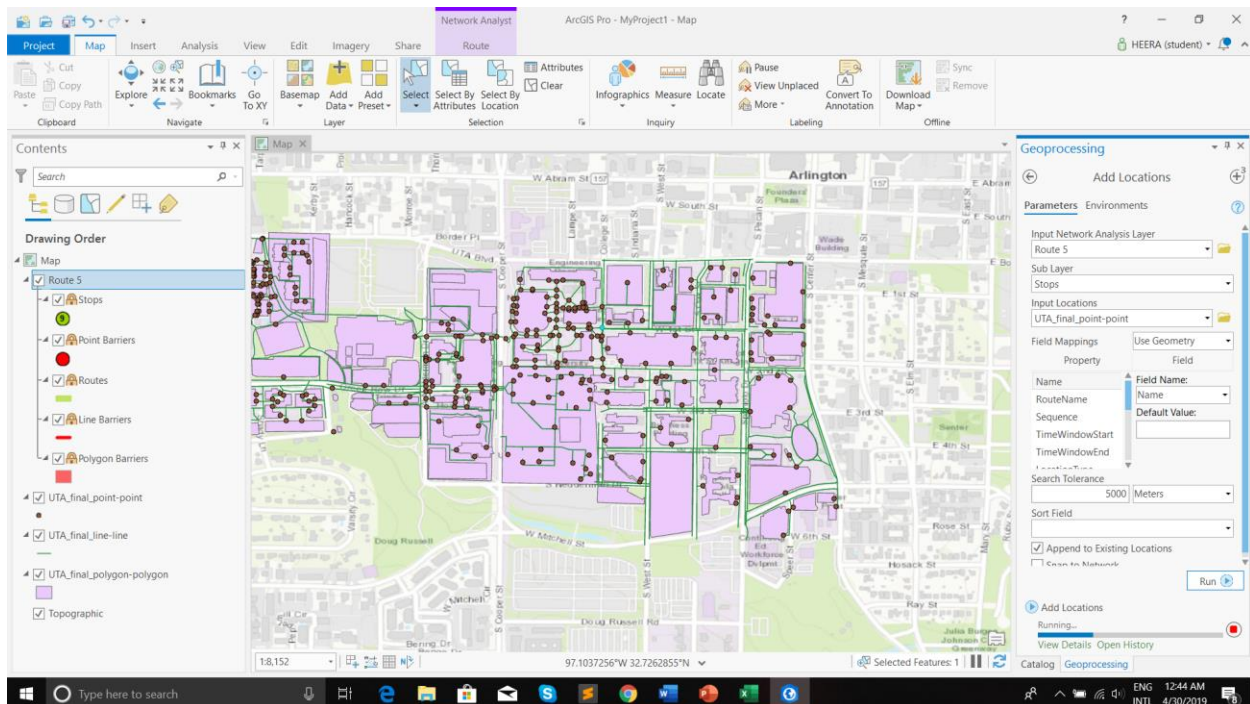
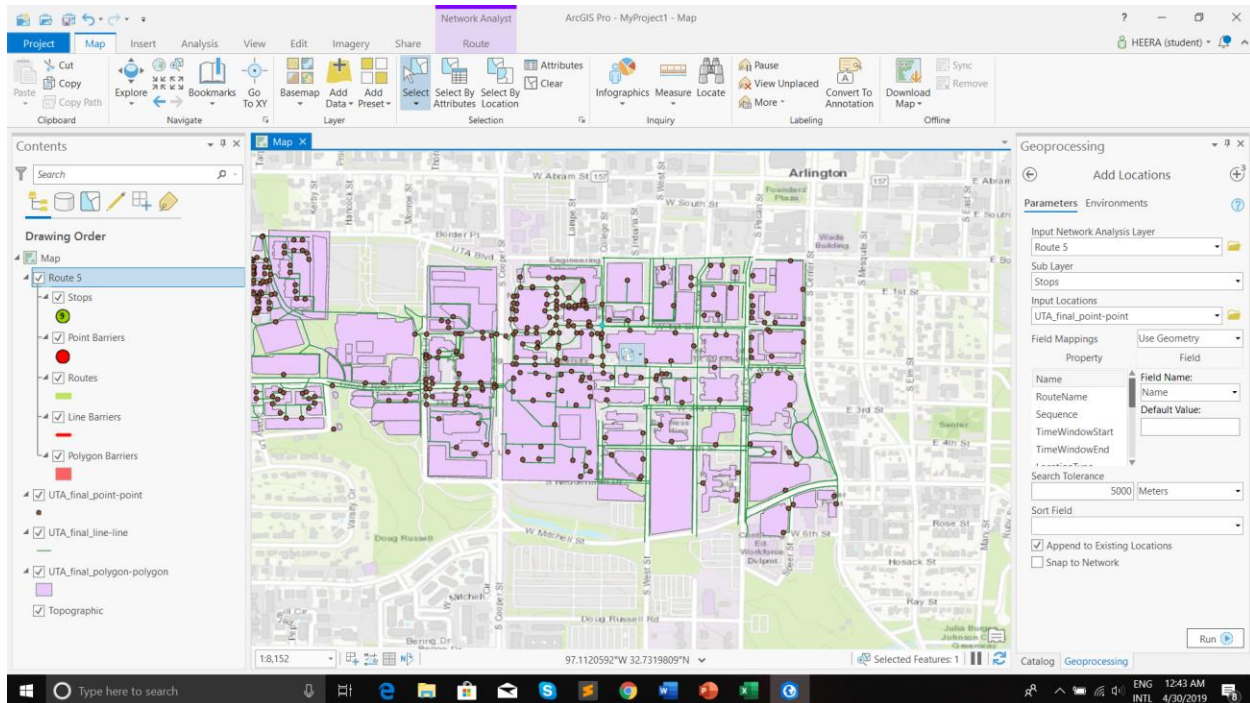
After network analysis you will see a route option in left hand side bar



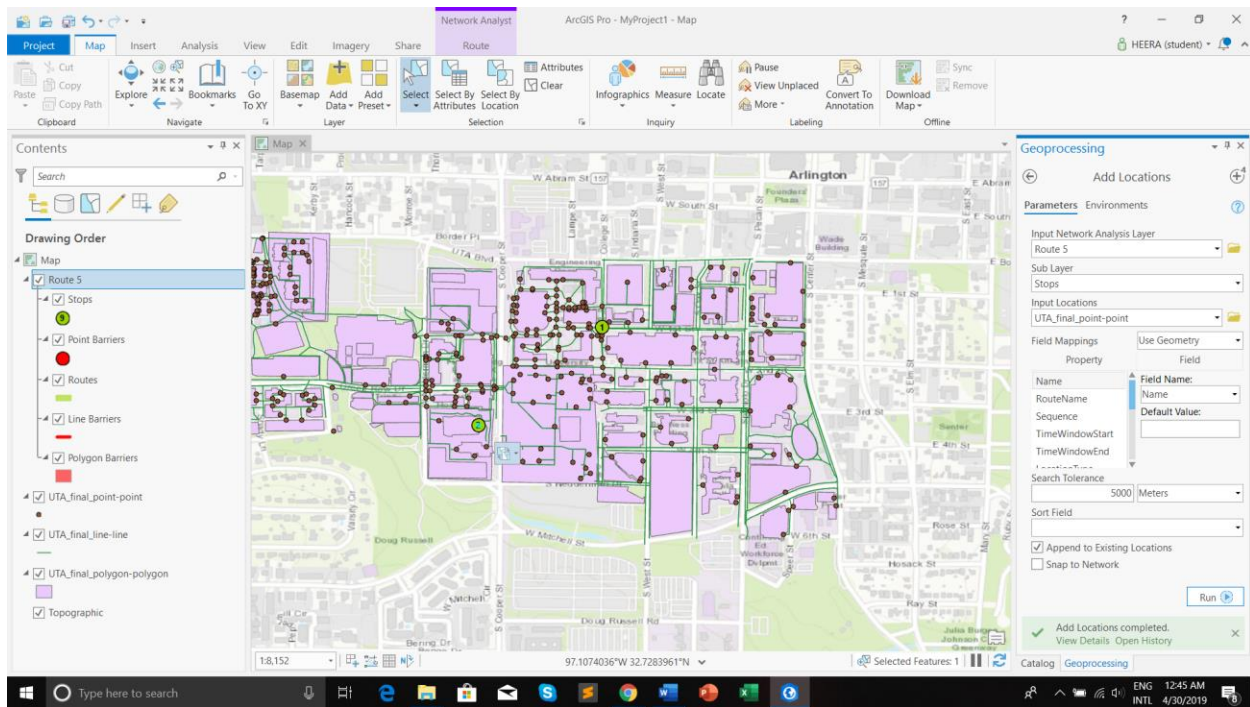
Now click on route tab and click on import stops



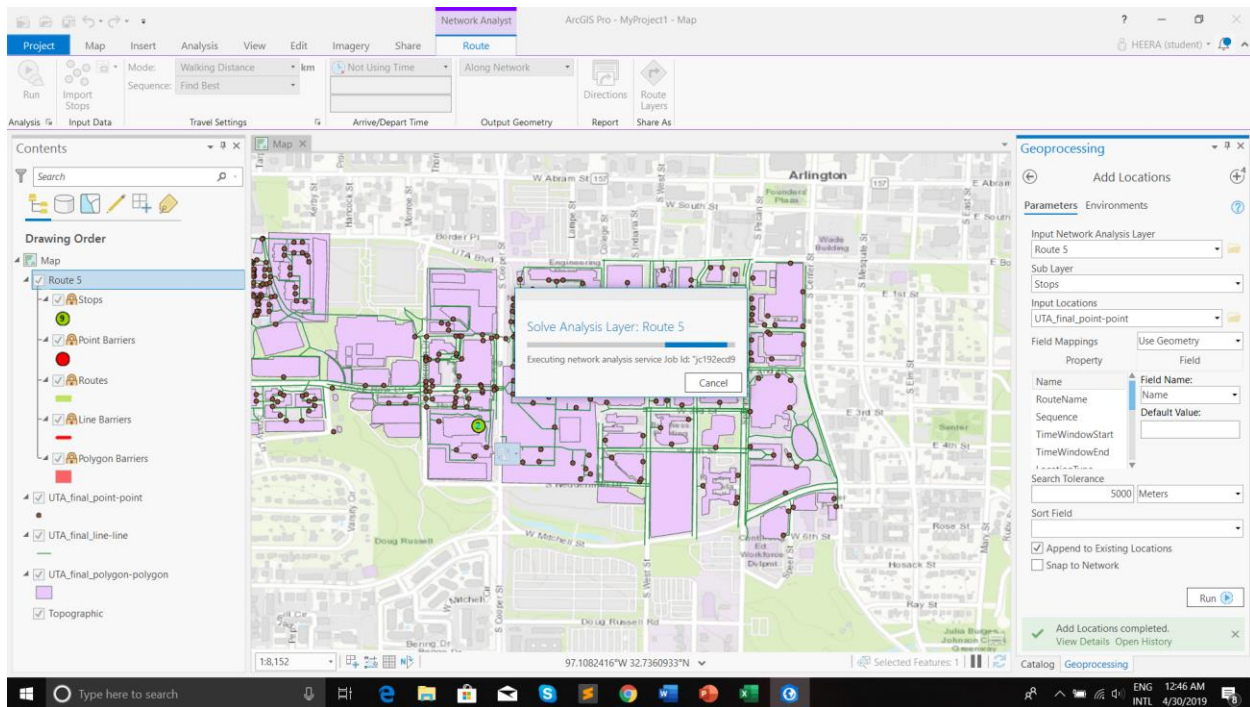
Now select the points and click on run



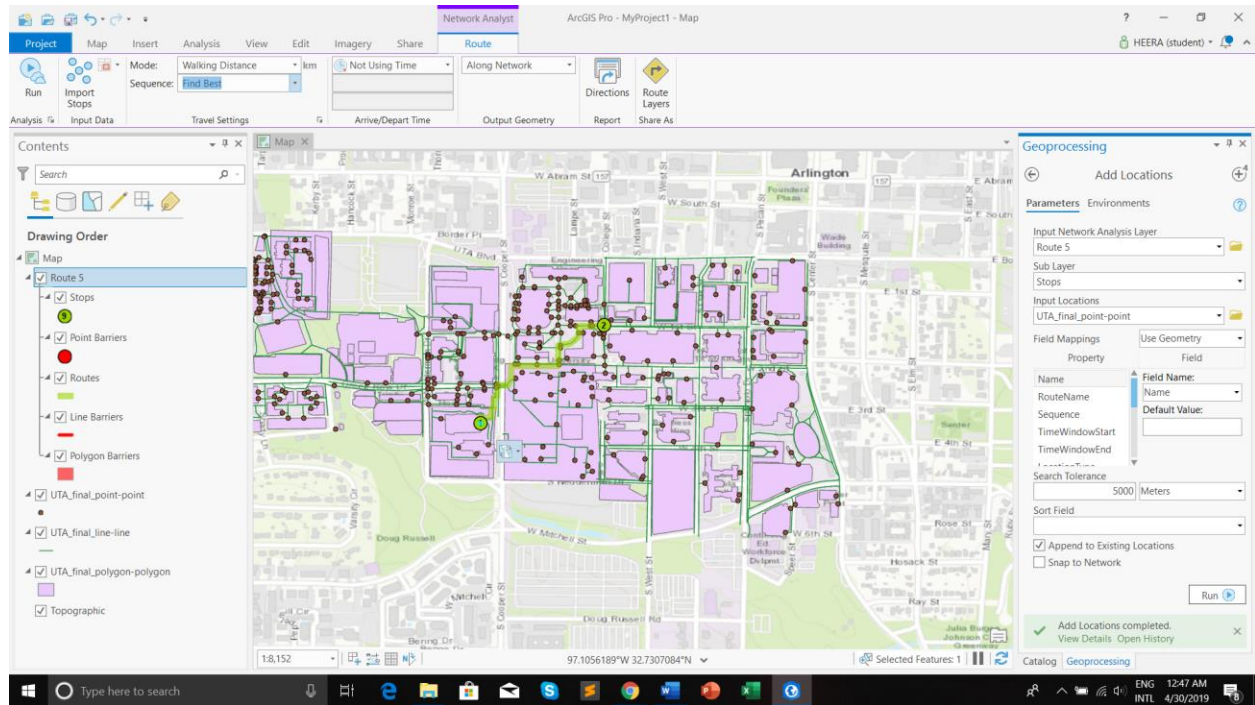
Now select another point after clicking on import stops and click run. Now that the two points are selected(highlighted in green)



Now select walking distance and find best under route and click on run



Shortest best path



REFERENCES

1. Google Earth, Wikipedia, "[https://en.wikipedia.org/wiki/Google Earth](https://en.wikipedia.org/wiki/Google_Earth)"
2. QGIS, Wikipedia, "<https://en.wikipedia.org/wiki/QGIS>"
3. SpatiaLite database, Wikipedia, "<https://en.wikipedia.org/wiki/SpatiaLite>"
4. QGIS Package reference, "<https://gis.stackexchange.com/questions/86812/how-to-draw-polygons-from-the-python-console>"
5. QGIS Network Analysis, "<http://gis-lab.info/qa/qgis-network-analysis-lib.html>"
6. <https://www.youtube.com/watch?v=8uAq1C7tJyU>
7. https://developers.google.com/kml/documentation/kml_tut

APPENDIX

The following files are attached:

file names(QGIS layer that has answer to the queries)

