University of Texas at Arlington

Department of Computer Science

Advanced Database Systems Project Report

Developing a GIS on the UTA Campus

Under the guidance of

Prof. Ramez Elmasri

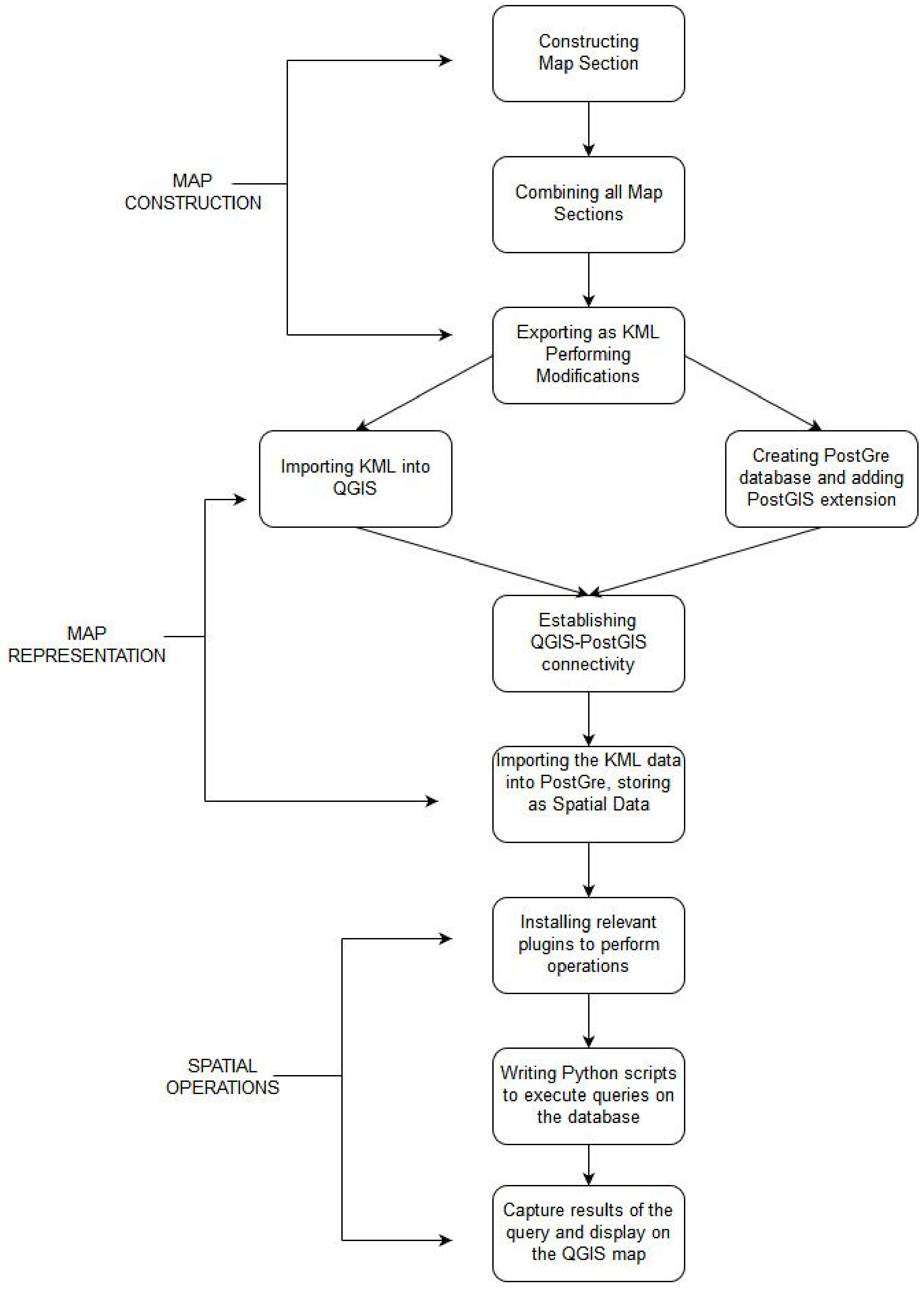
Project Group: Map Section G

Mansoor Abbas Ali **(1001453343)**

Chirag Hareshkumar Shah **(1001558824)**

Introduction

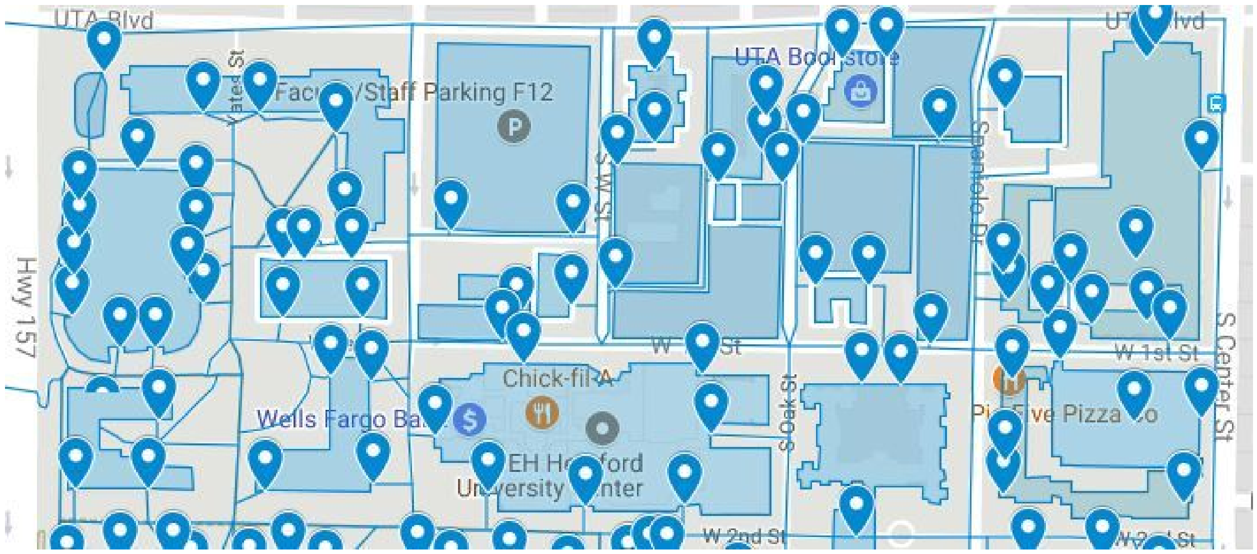
The purpose of this project was to create a small scale Geographic Information System on top of a custom built map of the UTA campus. The system would involve displaying the map on QGIS, performing spatial queries on it via python scripts and displaying the results onto the map. For other operations 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.



Constructing Map Section
Combining all Map Sections
Exporting as KML Performing Modifications
Importing KML into QGIS
Creating PostGre database and adding PostGIS extension
Establishing QGIS-PostGIS connectivity
Importing the KML data into PostGre, storing as Spatial Data
Installing relevant plugins to perform operations
Writing Python scripts to execute queries on the database
Capture results of the query and display on the QGIS map
MAP CONSTRUCTION
MAP REPRESENTATION
SPATIAL OPERATIONS

Map Construction

The entire UTA campus is considered as the base map for this project. The map was built in a collaborative manner with sections of the map being built by different teams. Softwares used for map construction include Google Earth and Google MyMaps.
Once the the different sections were completed, they were exported as KML/KMZ files, these files were then further modified for refinement and classification of different geographies. Below, is an image of the completed map on MyMaps. As can be seen, buildings, entrances and paths are depicted as polygons, points and line strings.



A portion of the UTA Campus map. This is exported as a KML file.

Map Representation

Once the map has been constructed, It is exported as a kml file and then imported into QGIS. Within QGIS it is represented as multiple vector layers. The different vector layers are based upon the classification done in the KML file. In our implementation(image below), we have 3 vector layers, namely: uta_entrances, uta_paths and uta_buildings.
At the same time, we create a database in PostGreSQL which will store the spatial information of the map. Furthermore, a PostGIS extension is added to the database which allows us to use various spatial queries on the data. The extension is added by the following query:

```
CREATE EXTENSION postgis;
```

Finally, when we have the vector layers and the database in place, we establish connectivity between QGIS and the PostGre database by using the PostGIS addon. Then we import data into the PostGre database by using a QGIS plugin called "DBManager". This plugin allows us to specify information about the server, the target database and the schema. It then allows us to import our selected vector layers into the database as spatial objects. Once the connectivity has been established, we can perform spatial queries on the different geometries.

Performing Spatial Operations

After the environment has been setup, we write Python scripts and add plugins which would allow us to perform certain spatial operations on the map and the spatial database. For our implementation we have considered the following operations:

1. Shortest Path: finding shortest path between two points
    a. Using Road Graph
2. Distance: find distance between two geometries
    a. Using st_distance()
3. Within: Finding geometries within a certain buffer
    a. Using st_within()
4. Selection: based on polygon type
    a. Using simple selection

Shortest Path

For finding shortest path between points, we make use of the “Road Graph” plugin in QGIS. The plugin requires a start and end point in the form of coordinates. It then calculates the shortest path between the two points and gives the distance. The plugin works on the following constraints:

- It operates on top of a line string layer and requires all lines in the line layer (road network) to be joined at common points.
- It also requires the line layer to be without a z-index as the plugin can only operate in two dimensions.

The operational summary of the plugin is as follows: It first converts the line layer into a weighted graph. It then adds the start and end points provided to the graph as vertices and then calculates the shortest distance between them by using Dijkstra’s algorithm. The path is then highlighted on the main map.

Distance, Within and Selection

The geometric operations are performed by combining the following:

1. Selection queries based on PostGIS
2. Python Scripts running on QGIS
3. QT window for user input
4. Result Display on QGIS map

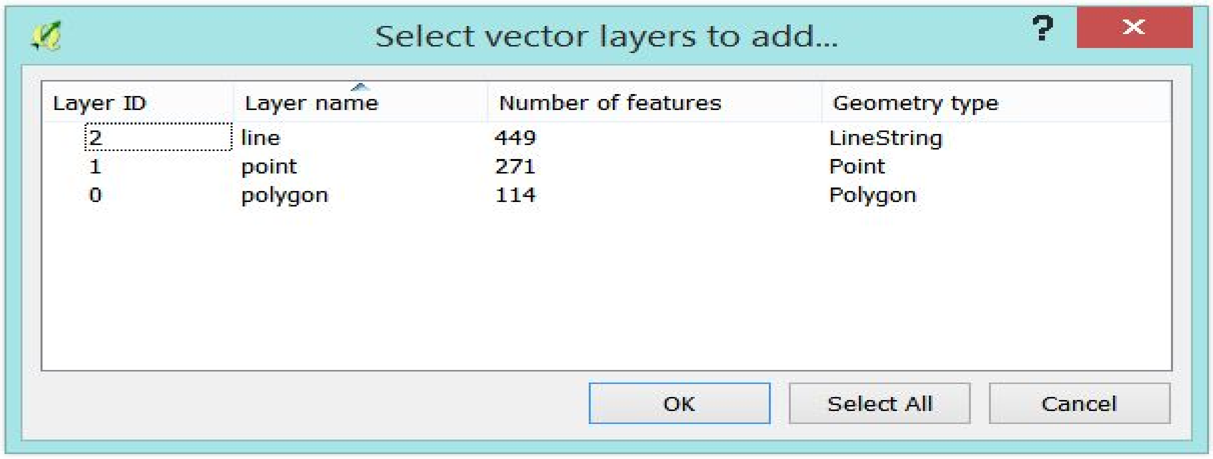
Given that the spatial queries run on PostGIS, our goal was to enable a user to run such queries without having to explicitly type them out, rather, they would enter data required by the queries on a friendly UI and get results onto the QGIS map without having to use the pgAdmin interface.

To that extent we have developed a Qt UI which takes data from the user and a script in python to establish connectivity with the spatial database, run the above mentioned queries, fetch results and display relevant selections on the map.

Pictorial Walkthrough

Here, we have a sequence of images by which we explain the implementation of the project.

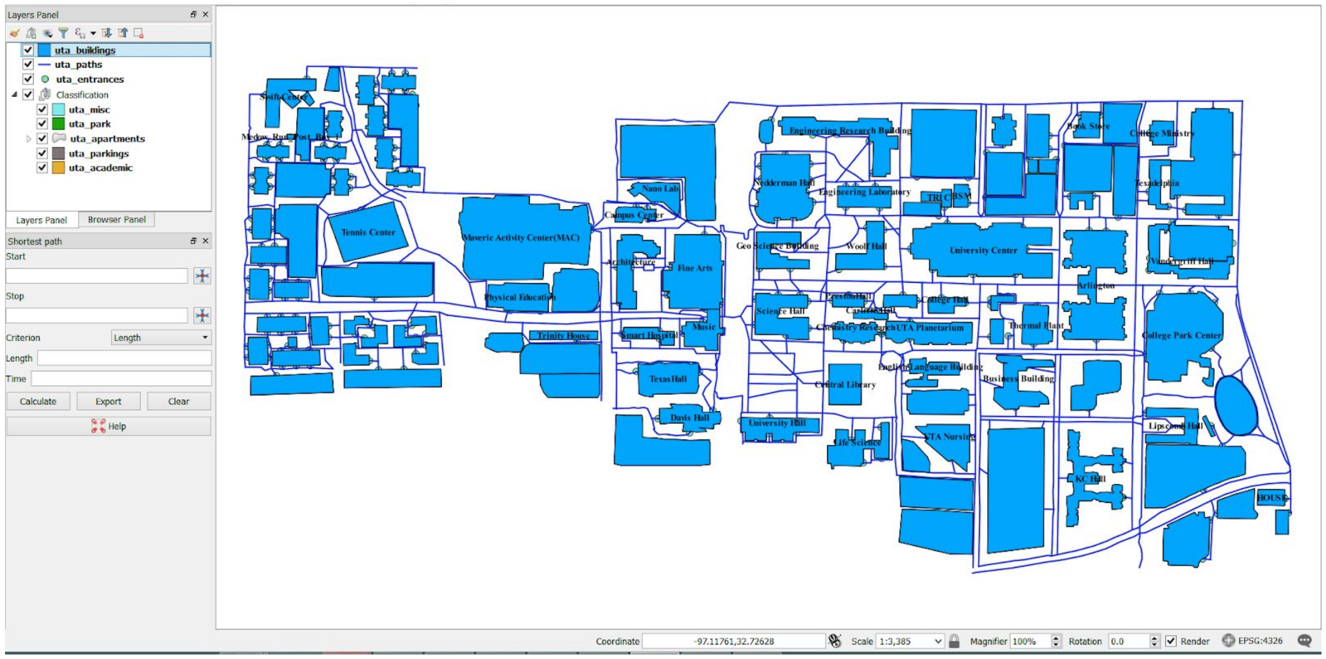
Image 1: Adding KML's to QGIS.



Select vector layers to add...

Layer ID	Layer name	Number of features	Geometry type
2	line	449	LineString
1	point	271	Point
0	polygon	114	Polygon

Image 2: The UTA map with layers for entrances(points), paths(lines) and buildings(polygons).

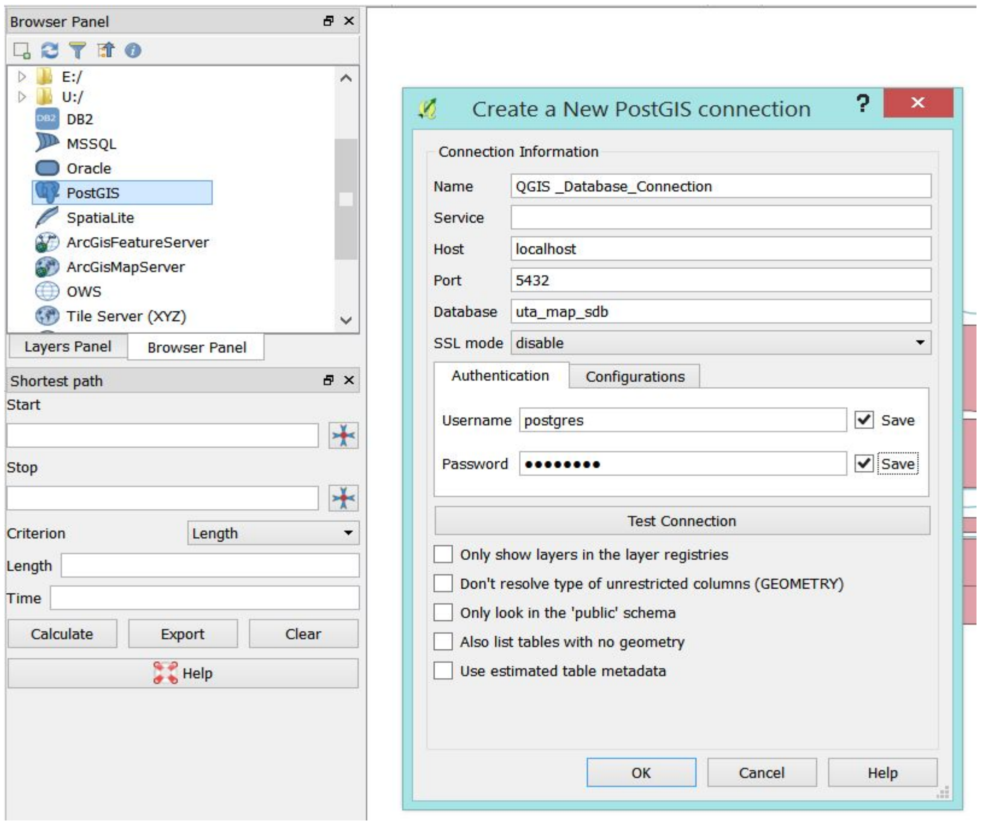


Layers Panel
- ✓ uta_buildings
- ✓ uta_paths
- ✓ uta_entrances
- ✓ Classification
  - ✓ uta_misc
  - ✓ uta_park
  - ✓ uta_apartments
  - ✓ uta_parkings
  - ✓ uta_academic

Shortest path
Start
Stop
Criterion: Length
Length
Time

Coordinate -97.11761,32.72628 Scale 1:3,385 Magnifier 100% Rotation 0.0 ✓ Render EPSG:4326

Images 3 & 4: pgAdmin extension and DBmanager used to establish PostGIS connectivity.



Create a New PostGIS connection

Connection Information

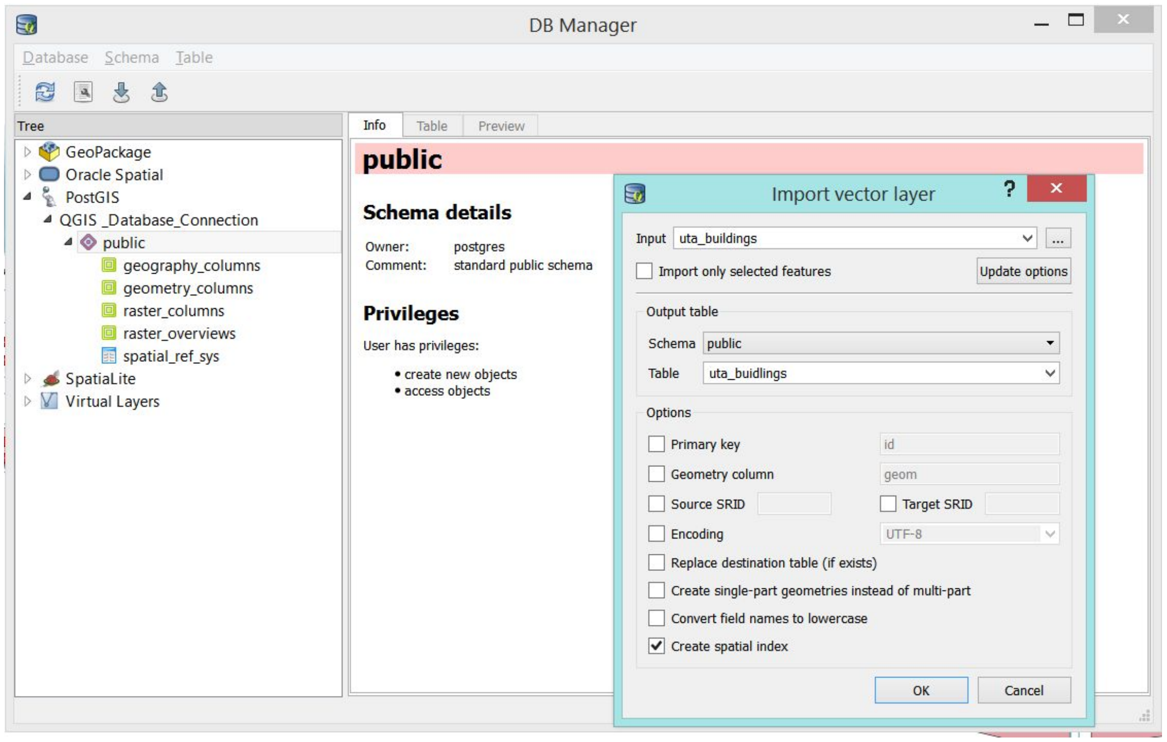
Name	QGIS _Database_Connection
Service	
Host	localhost
Port	5432
Database	uta_map_sdb
SSL mode	disable

Authentication | Configurations

Username	postgres	✓ Save
Password	••••••••	✓ Save

- ☐ Only show layers in the layer registries
- ☐ Don't resolve type of unrestricted columns (GEOMETRY)
- ☐ Only look in the 'public' schema
- ☐ Also list tables with no geometry
- ☐ Use estimated table metadata

Criterion: Length
Length:
Time:



public

Schema details

Owner: postgres
Comment: standard public schema

Privileges

User has privileges:

- create new objects
- access objects

Import vector layer

Input: uta_buildings

☐ Import only selected features

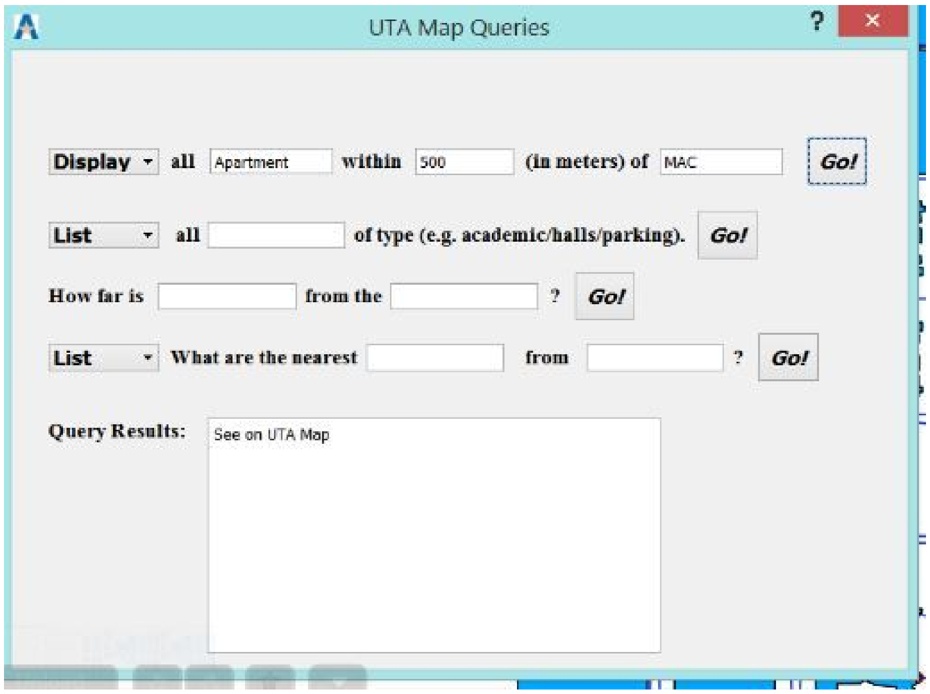
Output table

Schema	public
Table	uta_buidlings

Options

- ☐ Primary key — id
- ☐ Geometry column — geom
- ☐ Source SRID ☐ Target SRID
- ☐ Encoding — UTF-8
- ☐ Replace destination table (if exists)
- ☐ Create single-part geometries instead of multi-part
- ☐ Convert field names to lowercase
- ✓ Create spatial index

Image 5: Running "Within" query via our UI.


UTA Map Queries

Display all Apartment within 500 (in meters) of MAC Go!

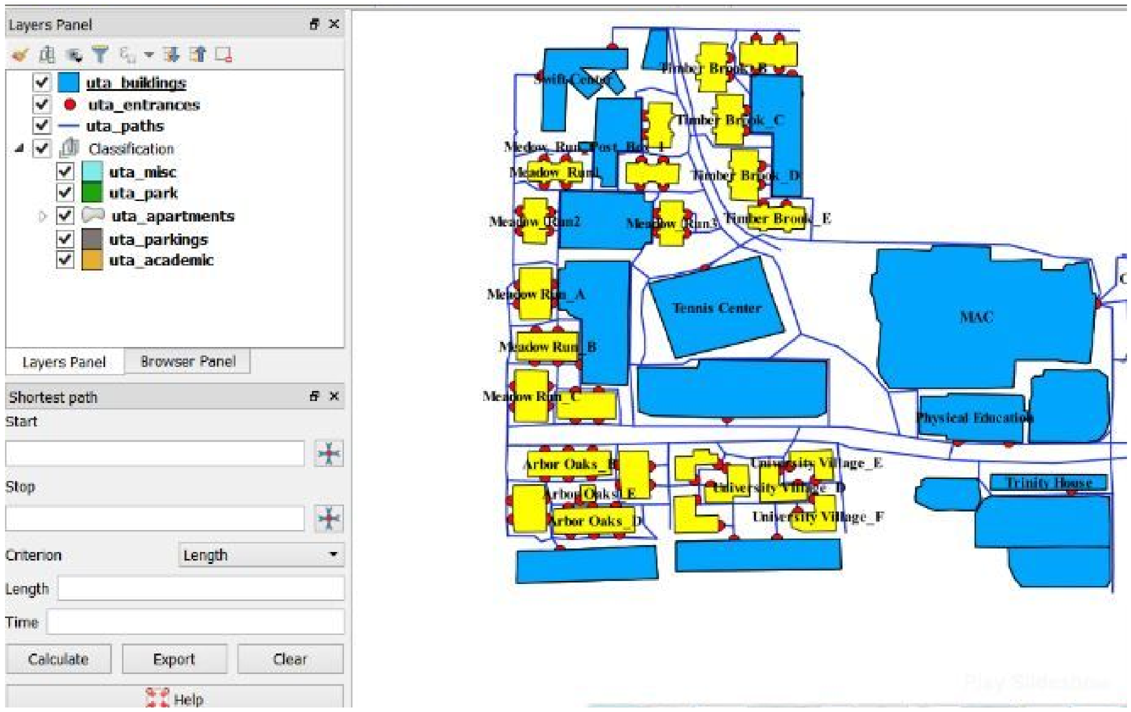
List all ____ of type (e.g. academic/halls/parking). Go!

How far is ____ from the ____ ? Go!

List What are the nearest ____ from ____ ? Go!

Query Results: See on UTA Map

Image 6: Displaying the result.


Layers Panel

- ✓ uta_buildings
- ✓ uta_entrances
- ✓ uta_paths
- ✓ Classification
  - ✓ uta_misc
  - ✓ uta_park
  - ✓ uta_apartments
  - ✓ uta_parkings
  - ✓ uta_academic

Shortest path

Start

Stop

Criterion: Length

Length

Time

Systems/Software Considerations

Our project implementation is based on the following systems and softwares:

- **QGIS -** to design a Visual representation of UTA's map. QGIS supports viewing, editing and analysis of spatial data. The main advantage of QGIS is that it is cross platform and has python support.
- **PostgreSQL -** PostgreSQL a object-relational database which stores spatial data. PostGIS is an open source software to support graphical objects. This is externally added as an extension to PostgreSQL.
- **Python -** Used to create UI and to run queries.