QGIS VRP Plugin guide

A QGIS plugin to determine routing and logistic solutions for disaster management scenarios

Contents

Introduction

This document contains the details of the Nodal Disaster Management for Vehicle Routing Problems (NDM-VRP) plugin .

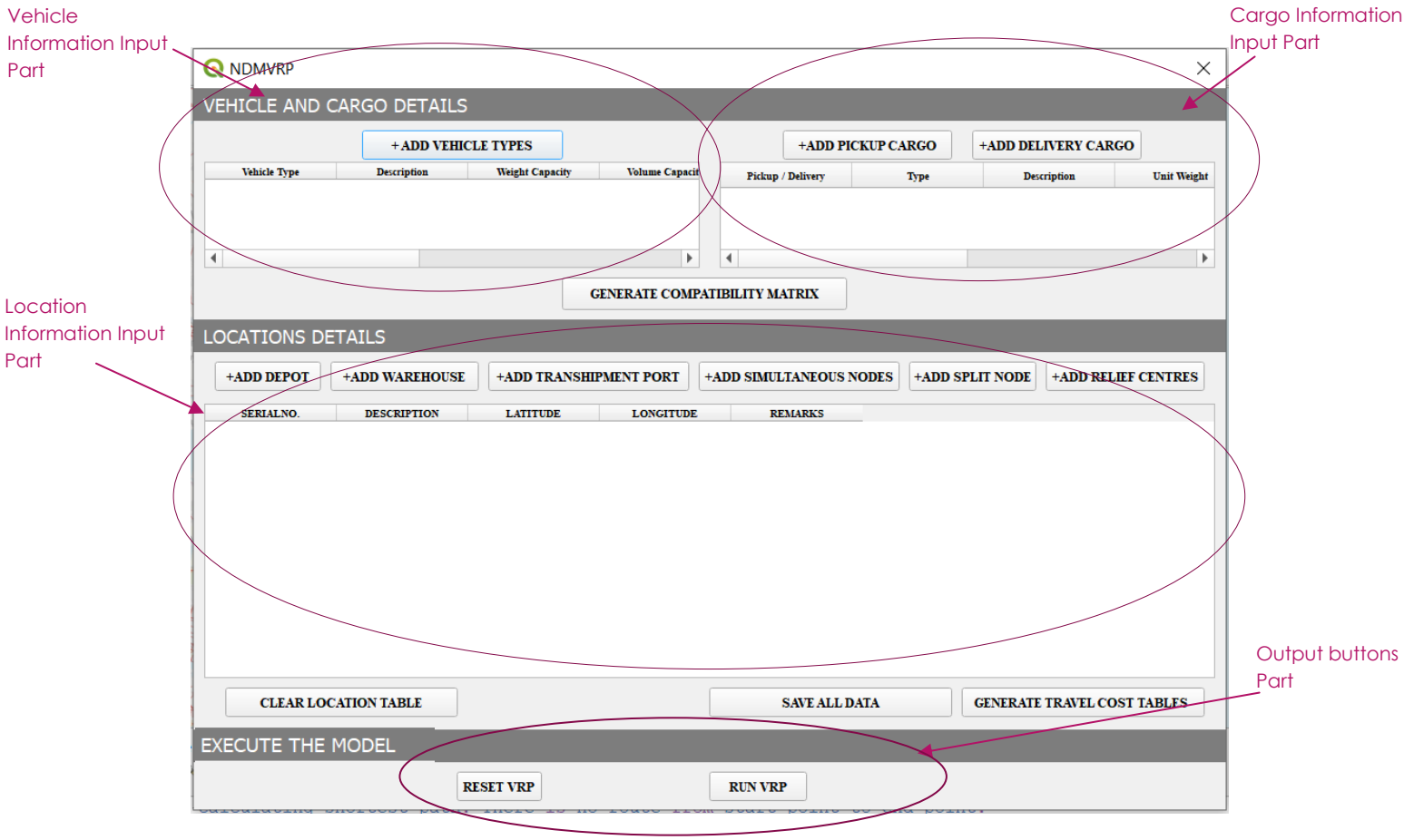
The plugin is used a Spatial Decision Support System (SDSS) tool for disaster management scenario. The potential users for such a tool include disaster management authorities and personnel, and researchers. The tool will help in providing logistical support in a disaster management scenario. The user may use the plugin to provide information such as available fleet of vehicles, items and equipment to be provided at user defined locations as well as to support relief and rescue operations.

The user is required to be familiar with the QGIS software and its uses to better utilise the tools. The user is expected to be familiar with how QGIS plugins work. And for better utilisation, the user may provide better datasets such as road network or disaster data which will help in using the plugin.

Parts of the plugin

• UI INTERFACE

The basic UI of the plugin is loaded upon pressing the plugin button on QGIS interface. The first look of the UI is as shown below:



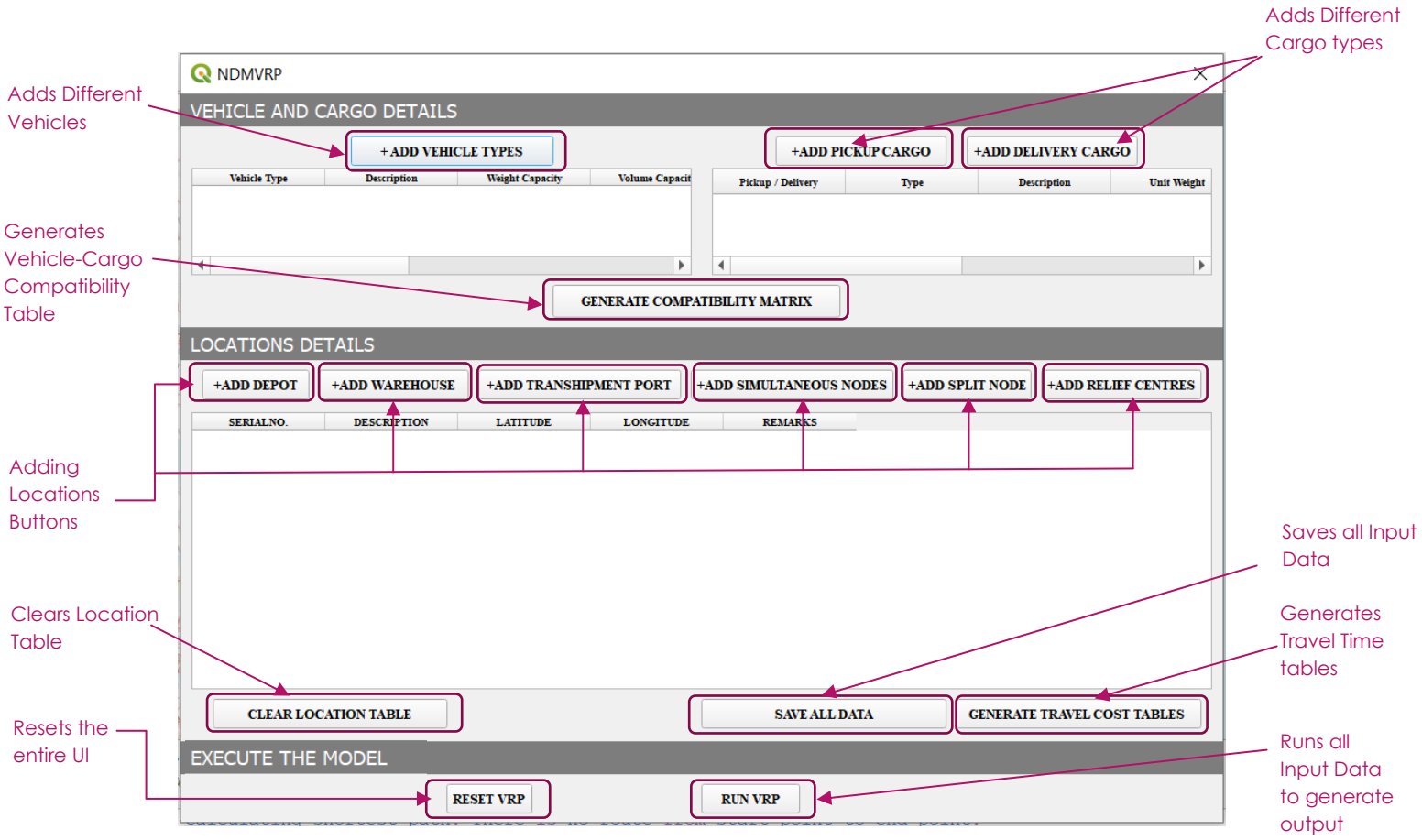
The UI contains three parts. The first two parts are for the user to give inputs to the system and the final part processes these inputs to generate the final results. The outputs are generated as layers in the QGIS pane and also, they are stored locally in the user's computer in a user defined location. The outputs are saved locally in .csv format.

Process flow:

1. Enter Vehicle Type details
2. Enter Pickup and/or Delivery Cargo details, as necessary
3. Click to Generate Compatibility Matrix and edit it. The Compatibility Matrix also indicates the Loading/Unloading of the Vehicle-Cargo pairs (if the value entered is -1, then that Vehicle-Cargo pair is deemed incompatible and is never used for the solution; for any value >= 0, the corresponding Vehicle-Cargo pair is deemed compatible with the value provided being the Loading as well as Unloading time for the specific pair).
4. Enter all the Vertex details (as necessary for the problem)
5. Save the Data
6. Generate Travel Time Matrices (is automatically generated for each Vehicle Type)
7. Run the VRP (you will be asked to browse to the directory for the Input/OutPut files)

• BUTTONS

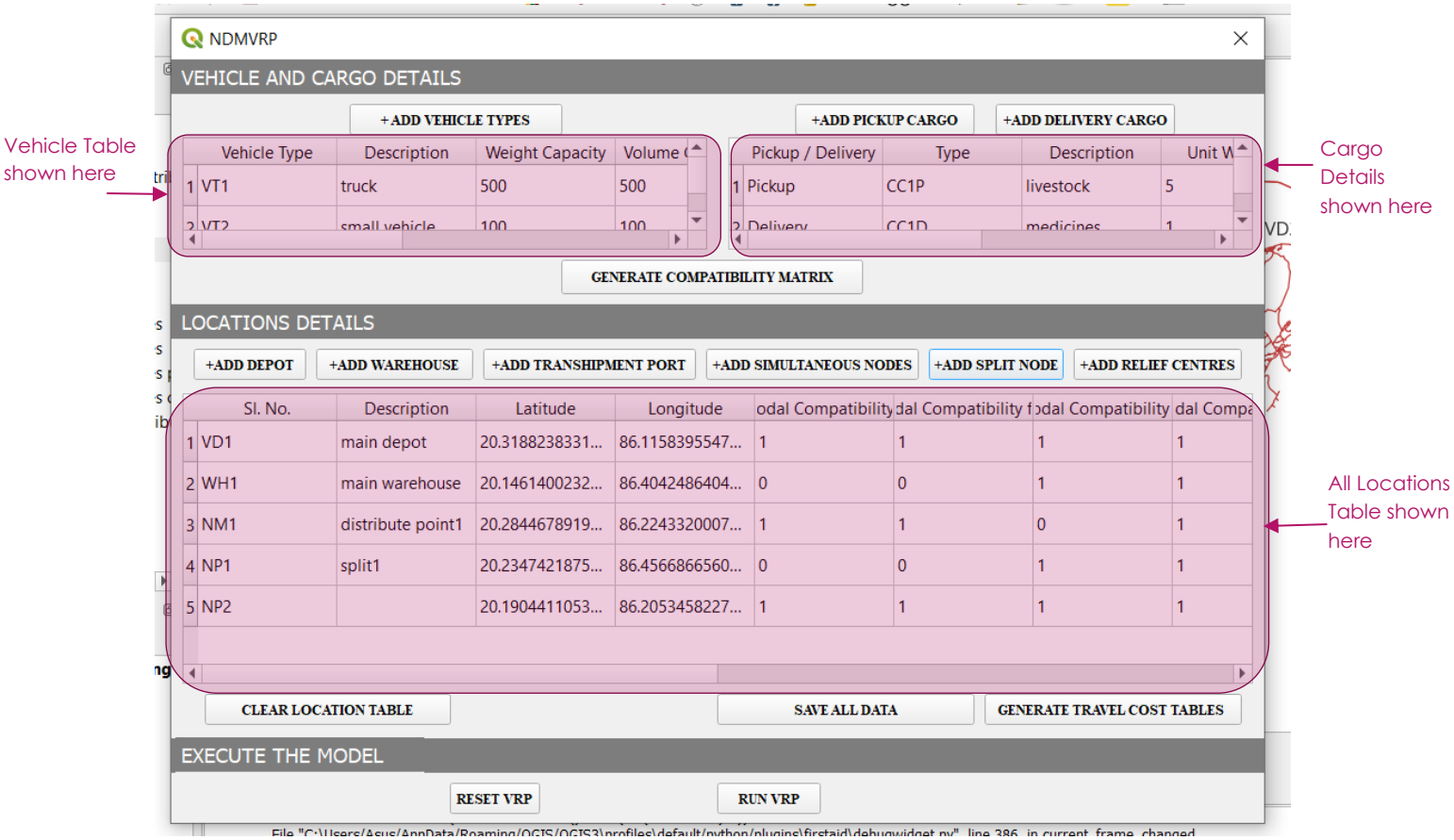
The UI contains buttons to trigger functions, dialogue boxes and add certain inputs to the algorithm such as generating travel cost for each vehicle type. On every button its functionalities are mentioned. One such button is shown below:



As seen above, the "GENERATE COMPATIBILITY MATRIX" button generates the vehicle-cargo compatibility matrix.

• VISIBLE TABLE DATASETS

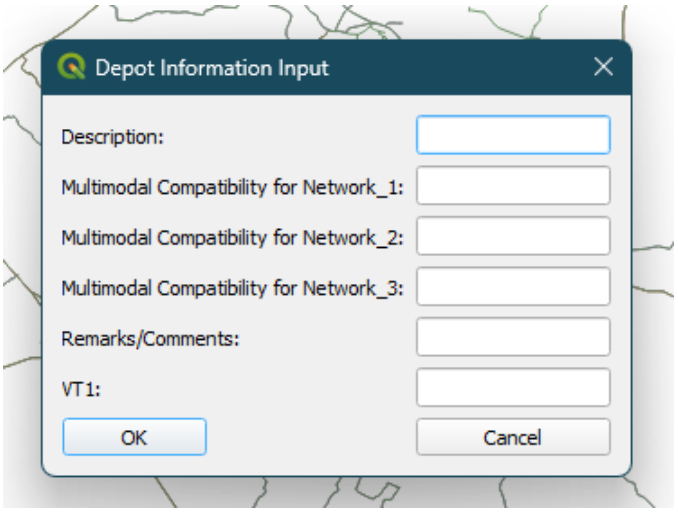
As the user is giving input, some of the inputs are shown on the UI itself in Tabular form. The user is able to change the data of a table, if needed. It is worth noting that changing any input may have effect on the entire output of the algorithm running in backend. Thus, users are advised to rerun the model every time the inputs are changed.

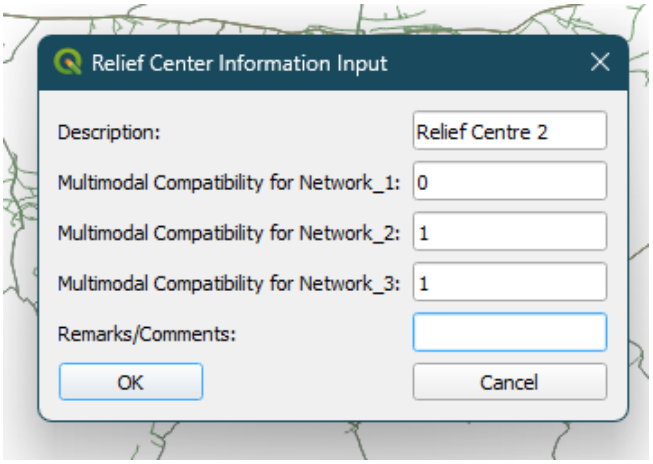


• DIALOGUE BOXES

Upon clicking certain buttons in the UI, a dialogue box will popup. The dialogue box has its own buttons and users are to provide inputs by filling up the dialogue boxes. Upon filling up all necessary information the dialogue box will close and the corresponding inputs will be saved on the UI table (mentioned before) where user will be capable to observing the outputs.

Two such dialogue boxes are shown below (in the current plugin's UI, the no. of Networks available for usage is limited to maximum 3):





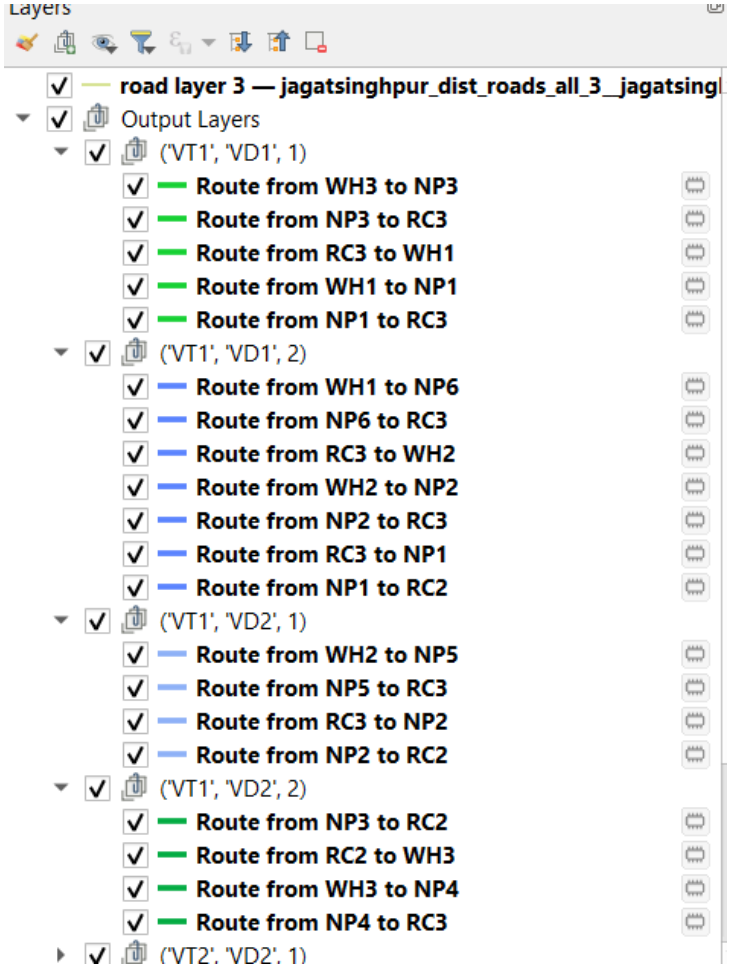
The inputs may be numeric (numbers only), alphanumeric (numbers, alphabets and symbols) or binary.

• OUTPUT VISUALISATION

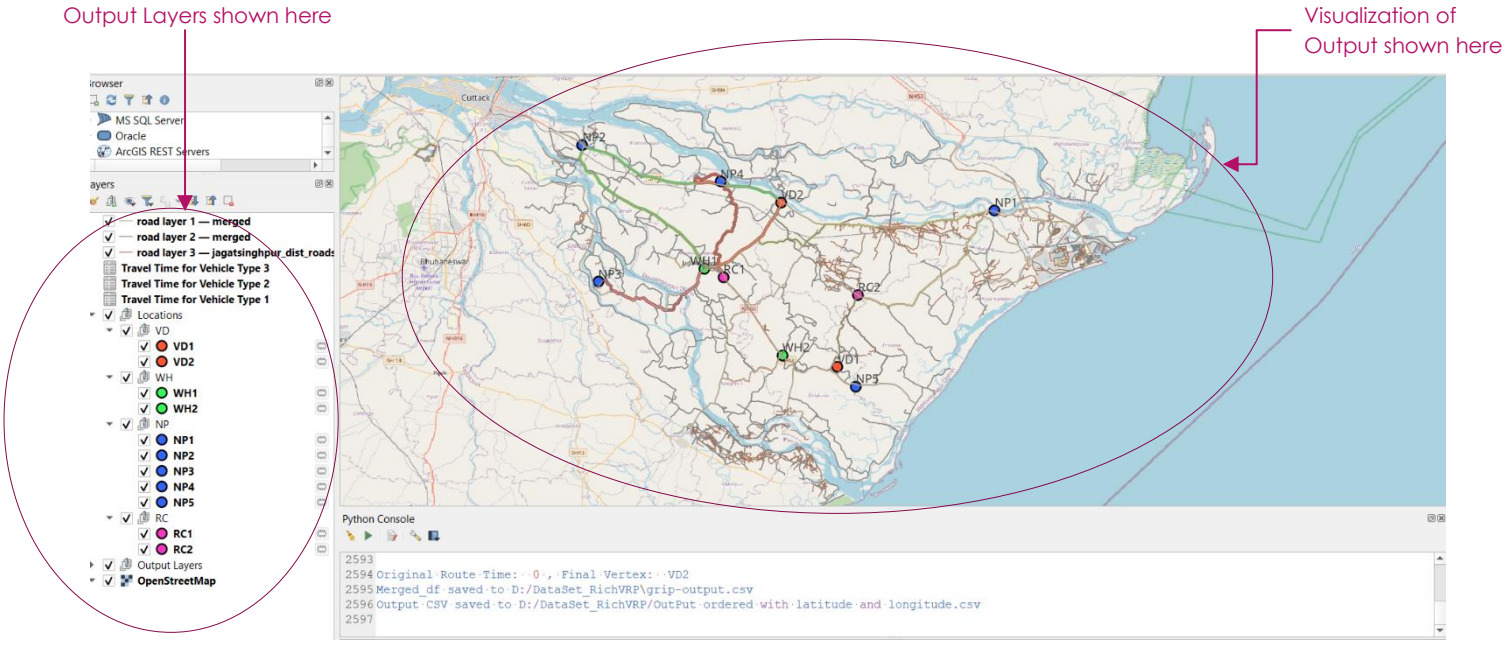
The outputs of the model is shown in two ways:

1. as layers in the layer pane
2. As visual outputs (routes and locations) on the QGIS canvas.

As layers, outputs are shown as collapsible groups that can be expanded to see individual outputs.



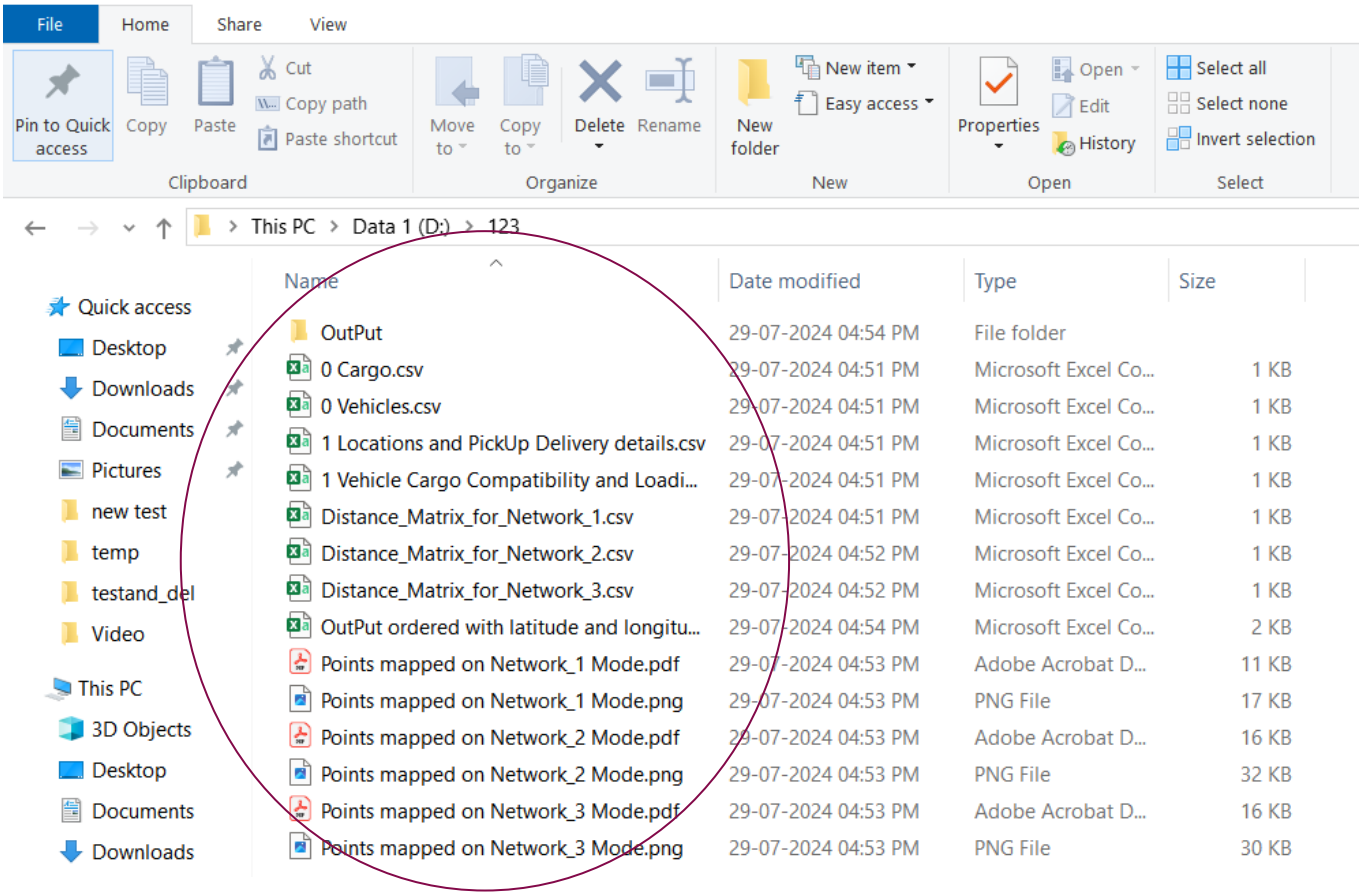
The above represents some vehicle route outputs that are shown for each individual vehicle.



The above represents a complete VRP summary run in a user defined area.

• INPUT AND OUTPUT REPOSITORY

The user is to save the input and outputs locally on their computers for handling and saving locally. It is critical that the user moves the files elsewhere if the user is to run successive runs of the model. After each run, a fresh batch of input and output CVSs will be generated and the old files are overwritten.



The output files are as follows:

- Cargo details
- Vehicle details
- Location details along with pickup and delivery data
- Vehicle-cargo compatibility table
- Model Output data with vehicles, vertices to visit and items to pickup/deliver information.
- Distance matrix between all the vertex through different road networks.
- Additional outputs for more elaborate explanation.

Design

- The plugin is written in python and is tested on **QGIS 3.28 (Firenze)**.
- The plugin uses **EPSG:4326** for its coordinates reference system.
- The plugin design has been done on QGIS QT designer
- The user can provide the Transportation networks for the analysis in **geopackage (.GPKG)** format.
- The outputs and inputs are available to the user for recording purpose in **.csv** form.
- Some internal outputs from the PSR-GIP Heuristic are available separately in an **/OutPut** folder automatically created as a subdirectory to the user's chosen input location; this folder will be overwritten each time the same input is run (i.e. deleted and created again with new files).

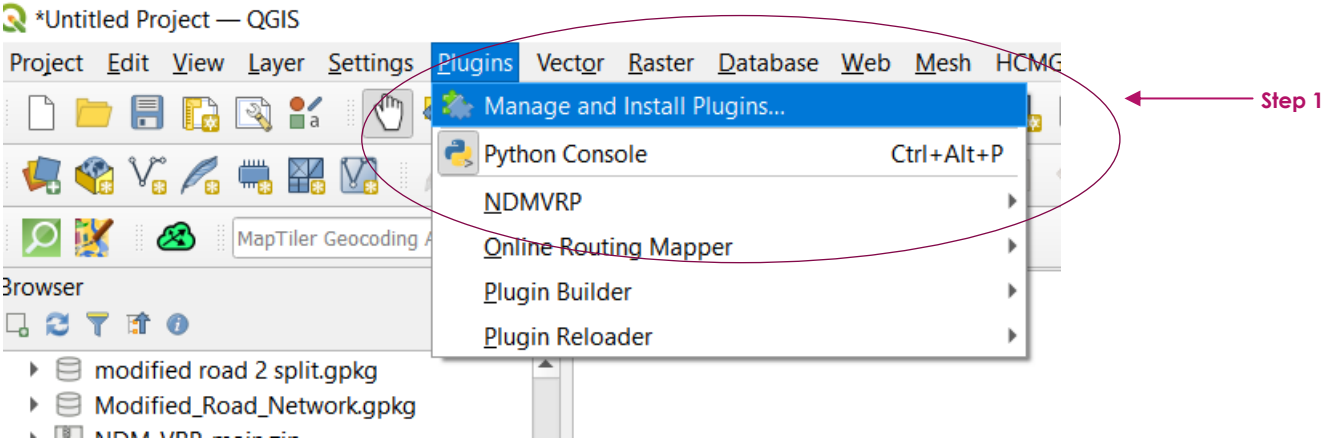
Downloading and Installation Guide

System Requirements:

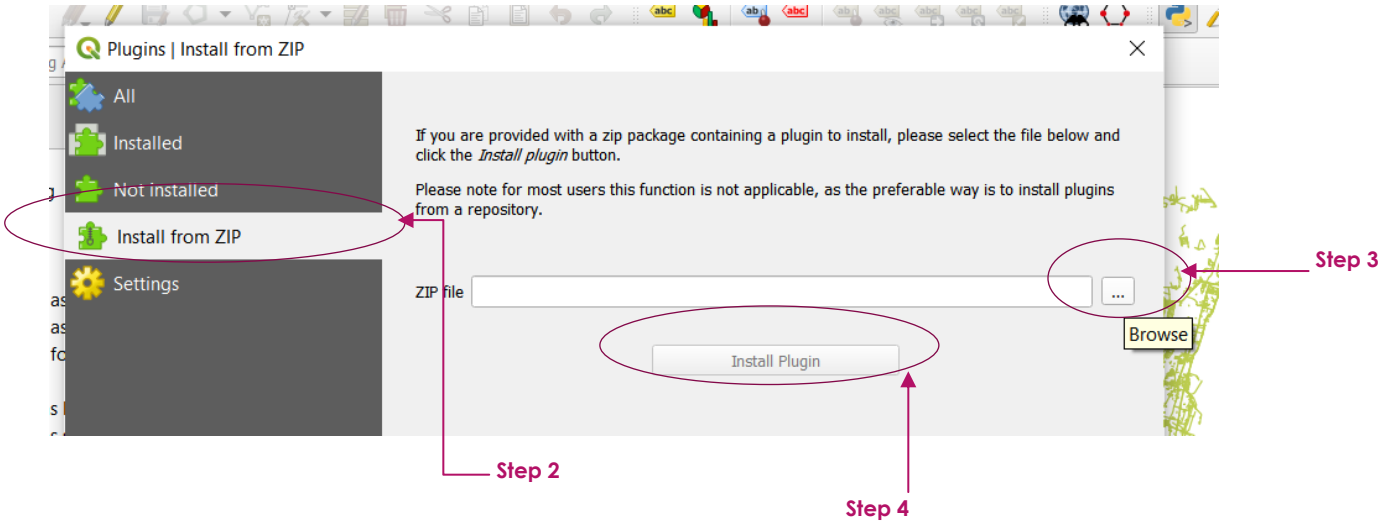
- A Modern PC running Windows
- 4 GB Ram (8 GB recommended)
- A decent processor (i5 or equivalent and above recommended)
- QGIS 3 or above installed (3.28 Firenze recommended)

Installation Guide:

- The Zip file for the Plugin can be downloaded from here.
- The user must then open QGIS and go to the top bar and click on the "Plugins" tab.



- Click on the "Manage and Install Plugin" popup button to get the plugin installation popup menu



- Click on "Install from ZIP" and navigate to the ZIP download in your computer and install. The plugin can now be viewed on the QGIS pane on top.

Instructions before you start

The software repository for the plugin is **HERE.**

The zip is to be downloaded and installed via QGIS

Some sample road network files are also provided in **FOLDER**

The plugin assumes there are three levels of road network such as level 1, 2 and 3. They correspond to large, medium and small vehicles respectively, allowing for heterogeneous fleet of vehicle.

The Transportation Networks are named "**Network_1**" (large vehicle), "**Network_2**" (medium vehicle), "**Network_3**" (small vehicle). If the user wishes to change the location where the model is run on, the user may build their own road layers. For running the model in a specific region, the region-specific road networks are required. It is worth noting that the size and complexity of the road network will have proportional effect on the runtime of the model for analysis.

If the user wishes to change the Network files, they must open the plugin repository and overwrite the existing road networks while keeping the names as instructed before. The default location of the plugin repository is

C:\Users**USER_account**\AppData\Roaming\QGIS\QGIS3\profiles\default\python\plugins\

where the **<USER_account>** is replaced by user's account name in the computer.

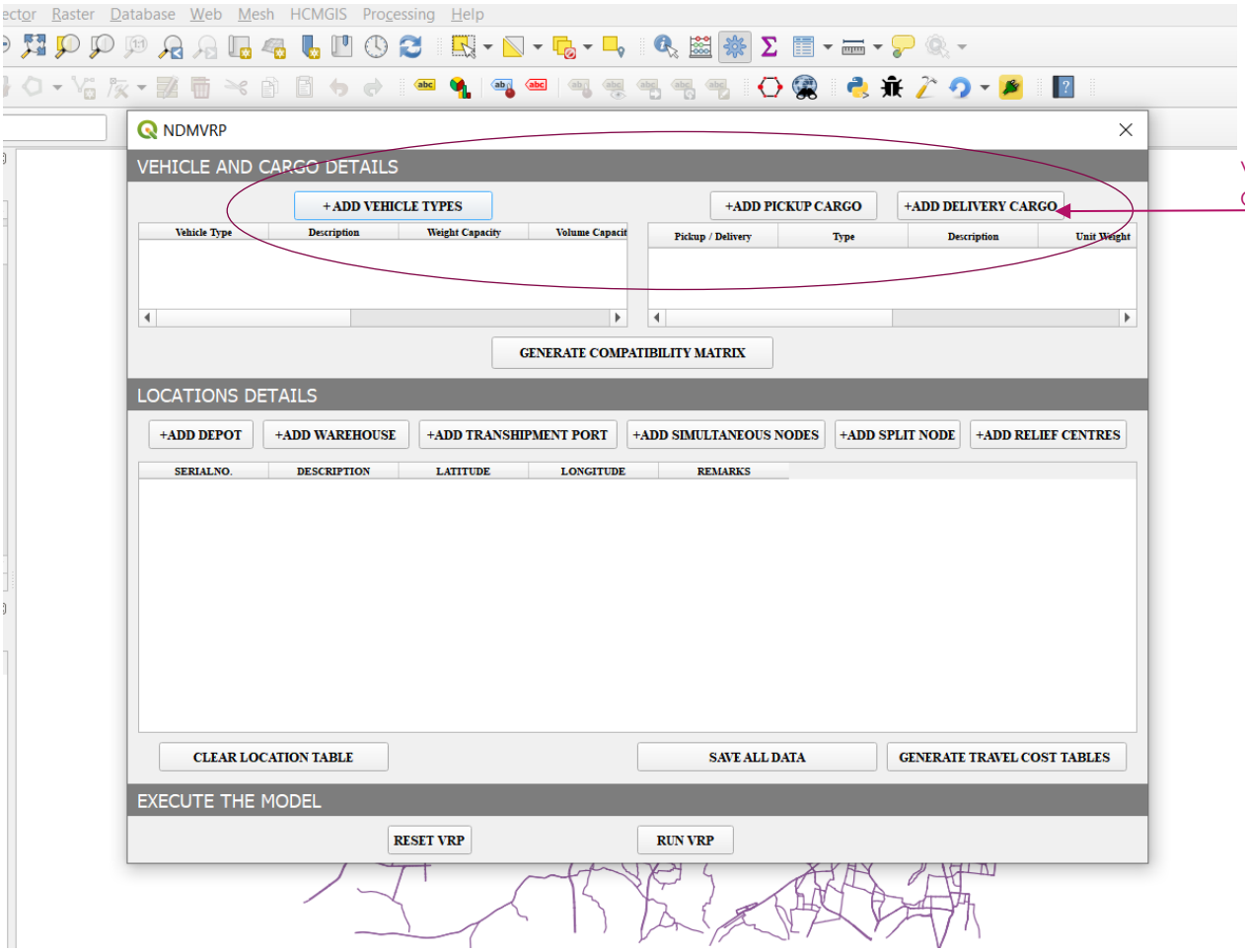
There is also an instruction pdf for using disaster scenario into the model. The plugin assumes that the road **Networks** provided are the ones to be used for analysis and the necessary changes for disaster scenario has been implemented into the road networks themselves.

The users may also request access to the private GitHub repository (https://github.com/sid0dodo1/NDMVRP) which is used to maintain the current code.

Step by step instructions

The developed **QGIS plugin** is to work as follows:

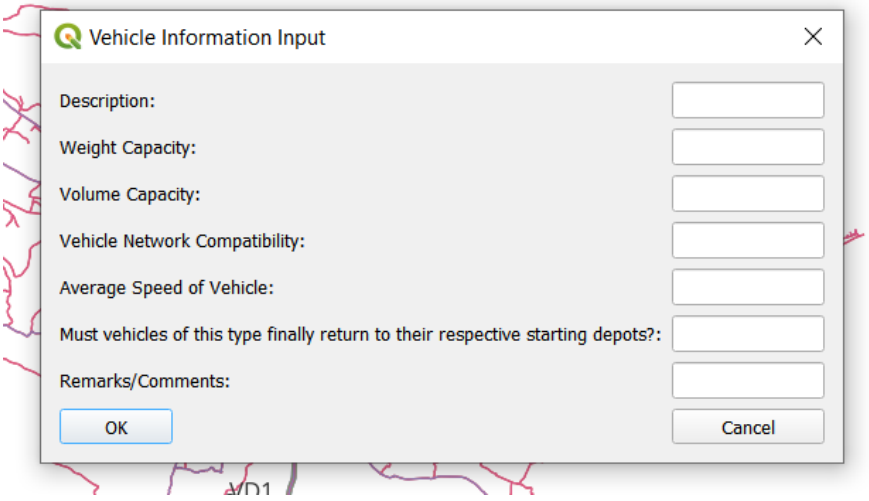
The input has several steps. The first is "Vehicle and Cargo details".



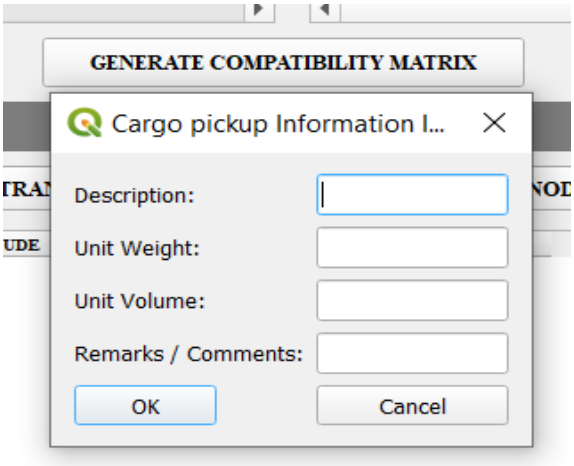
- The user will load the map into the canvas with EPSG: 4326 formats(google maps/satellite etc.

- The user will press the **"add vehicle types" button** to pop up a dialogue box with the following inputs:

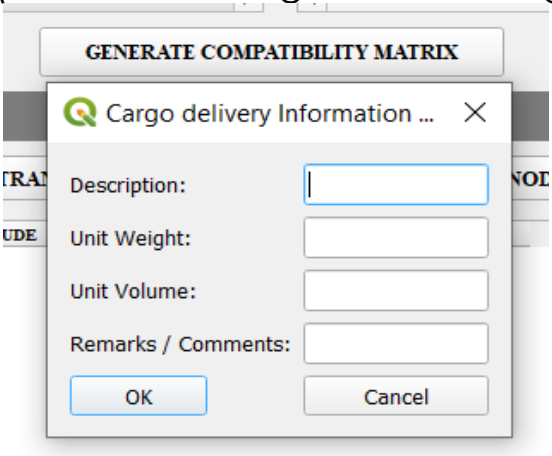
1. Description (user input, alphanumerical)
2. Weight Capacity (user input, numerical)
3. Volume Capacity (user input, numerical)
4. Vehicle Network Compatibility (user input, numerical: 1-3)
   The current plugin's UI supports upto a maximum of 3 road networks, and a Vehicle Type is compatible with any one of them, which must be indicated by the user through this input.
5. Average Speed of Vehicle (user input, numerical)
6. Must vehicles of this type finally return to their respective starting depot?(user input, Binary)
7. Remarks/Comments (user input, alphanumerical)
8. **Ok** button (to save input to the dataframe)
9. **Cancel** Button (to cancel and go back to Plugin UI)



- Next the user will press the **"add pickup cargo" button** to popup another dialogue box that would have the following inputs
  1. Description (user input)
  2. Unit weight (user input)
  3. Unit volume (user input)
  4. Remarks/Comments (user input)
  5. **Ok** button (to save input to the dataframe)
  6. **Cancel** Button (to cancel and go back to Plugin UI)



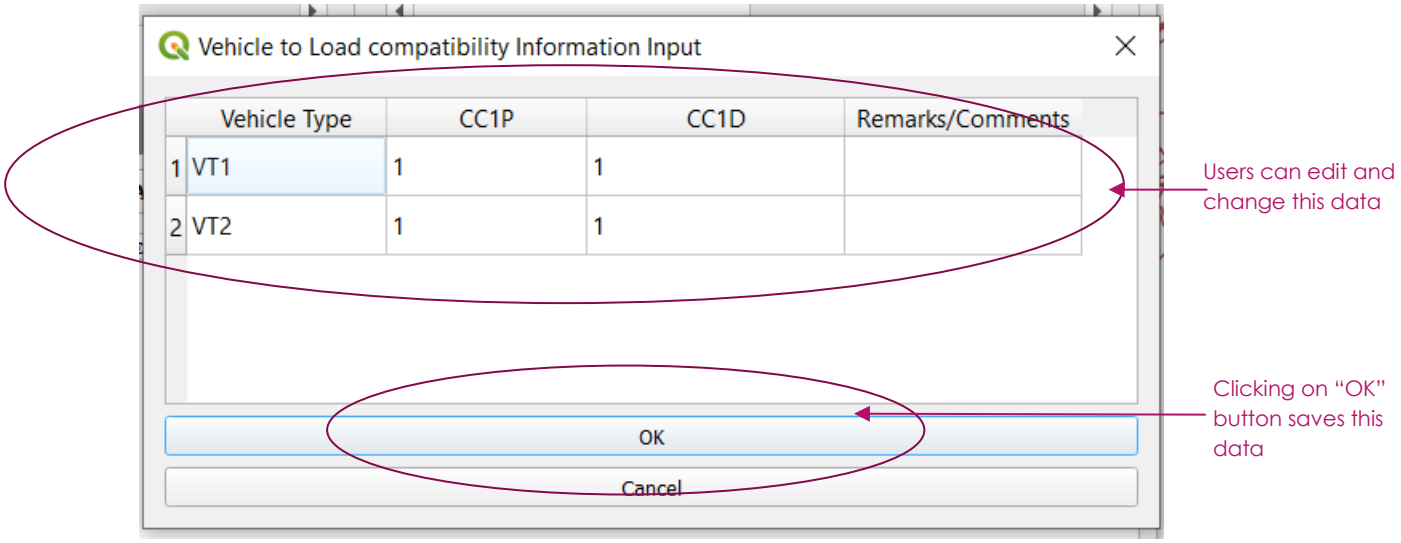
- Next the user will press the **"add delivery cargo" button** to popup another dialogue box that would have the following inputs
  1. Description (user input, alphanumeric)
  2. Unit weight (user input, numeric)
  3. Unit volume (user input, numeric)
  4. Remarks/Comments (user input, alphanumeric)
  5. **Ok** button (to save input to the dataframe)
  6. **Cancel** Button (to cancel and go back to Plugin UI)



- Next the user will press the **"generate compatibility matrix" button** to generate the default Vehicle to Cargo compatibility. User is free edit and change inputs in this (as well as in the other dataframes) as long as they are not saved by clicking the "save all data" button. If changes are made, user will have to save the data again before processing all inputs.

  For incompatible Vehicle-Cargo pairs, the user must enter -1 (negative one), and

  For compatible pairs, the user must enter the Loading/Unloading time (i.e. the time to load a single unit of the Cargo Type into the specific Vehicle) in the same time units as is generated via the Travel Time Matrix.



- The user must do these tasks of creating vehicle information and cargo information dataframes as well as generation of compatibility matrix. This is crucial before any further inputs are given, as this data is key to the next series of inputs.

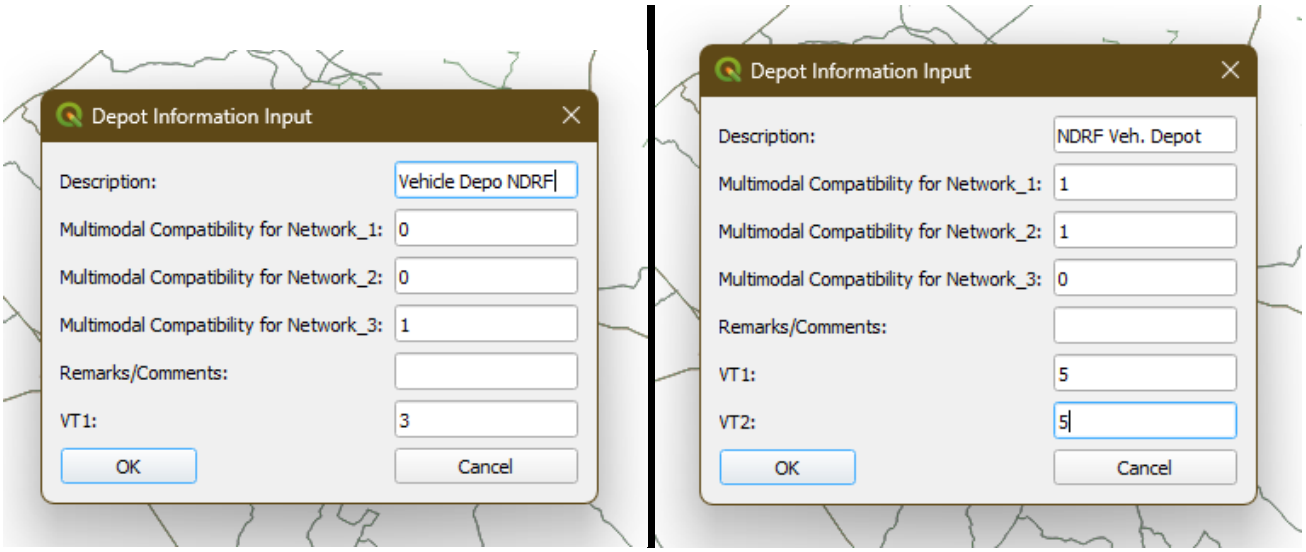
- The next set of inputs is location details.

  The user will be selecting locations on the map canvas by manually clicking. For example, pressing the **"add depot" button** will enable the

user to add depot(s) by adding a point layer to the canvas. In **each** click the following fixed input fields will be created in a pop up:

1. Description (user input, alphanumerical)
2. Multimodal Compatibility for Network_1 (user input: Binary)
3. Multimodal Compatibility for Network_2 (user input: Binary)
4. Multimodal Compatibility for Network_3 (user input: Binary)
   For all the Multimodal Compatibility inputs above, 0 indicates inaccessibility and 1 indicates accessibility of that vertex by the specific Network. If there are less than 3 Networks being used, 0 must be provided for the Multimodal Compatibility for the unused Network.
5. Remarks/Comments (user input, alphanumerical)
6. **Ok** button (to save input to the dataframe)
7. **Cancel** Button (to cancel and go back to Plugin UI)

Two separate pictures are provided below explaining this in details:



- In addition to these, there will be variable inputs depending upon the type of location.
  - For example, for the add **Depot** button the number of available vehicles of each type will be required as input.
  - For adding **Warehouses**, the variable inputs will be the available delivery cargo (no. of units of each Delivery Cargo-type available) at each warehouse.
  - PickUp Cargo Types are associated with **Relief Centres** indicating the capacity to accept each of the PickUp

types (no. of units of each PickUp Cargo-type acceptable).

- For **Transhipment Ports**, multiple Networks can take the value of one, indicating accessibility to all those Multimodal Networks.
  Additionally, each Transhipment Port require a compatibility information with each Load Type (PickUp or Delivery Cargos). This is a binary input for each Cargo Type and Transhipment Port paired combination:
  A value of 0 indicates that the corresponding Cargo Type is **not** allowed to be transferred (if necessary at any time during the solution making) through the Transhipment Port.
  A value of 1 indicates that the transhipment is allowed.
- Nodes are an essential part of the problem and there must be at least one Node (Split or Simultaneous) for the problem. A **Split Node** allows its requirement to be fulfilled using separate Vehicle Types and in multiple visits; whereas a **Simultaneous Node** requires exactly one Vehicle visit to fully satisfy its entire requirement, thus not allowing any load partitioning.
  The name of the Cargo Types, i.e. **PickUp** or **Delivery** is also understood by the workflow of any cargo at the Nodes. PickUp Load types would need to be transported from Nodes to Relief Centres and Delivery Load types would need to be transported from Warehouses to Nodes.

- There are total **six types of location data** (i.e. Vertices). Namely:

  1. Vehicle Depot
  2. Warehouse
  3. Transhipment Port
  4. Split Node
  5. Simultaneous Node
  6. Relief Center

The Plugin's Heuristic will not develop partial solutions for infeasible problems and therefore feasible problems must be carefully planned for and appropriate inputs provided for the solution search. Further no-unit conversion takes place inside the Heuristic as it assumes that all values of the same

fundamental quantities provided (like time, Volume, weight) are the same throughout.

- For each location, there is **dedicated button for their respective inputs** with some fixed inputs and some variable inputs, as defined above.

- For the vehicle data**. For all vehicle type a singular dataframe will be visible** to the user on the UI itself.

- **For PickUp cargo and Delivery cargo buttons, a combined dataframe** will be generated on the UI. **For all 6 location data adding buttons, a single combined dataframe** will be visible to the user in the UI.

- The User can observe all the inputs this way and make changes if necessary. Making any changes in the dataframes will be reflected in the generated csv file (discussed in next point) only after saving the dataset again.

- Upon clicking the **"save all data" button** the location data, vehicle data, cargo data willbe saved in **appropriate csv files** for further processing.

- The dataframes will contain passive data that is useful for processing and does not require user to put manually. For example, the vehicle types will be auto generated as **"VT1", "VT2"** etc. the cargo will automatically be saved as **"CC1D", "CC2D"** etc. (delivery cargo) and **"CC1P", "CC2P"** (pickup cargo) etc.

- Similarly for the locations, prefix followed by a number will be generated as serial number. For example, for Vehicle Depot

VD1, VD2 or for relief centers RC1, RC2 etc. also the **coordinates of the locations will be collected by the system** itself upon the clicks by the user.

- Next set of inputs is **the travel distance matrix**. For this, user will simply click on the "generate travel cost tables" button and the distance matrix **between all added locations** will be shown in **different tables for different road networks** and will be ready for further processing. The attribute table for such a table is as following:

	start_point_id	end_point_id	distance
1	VD1	WH1	0.06209521320...
2	WH1	VD1	0.06209521320...
3	VD1	NM1	0.31884908557...
4	NM1	VD1	0.31884908557...
5	VD1	NP1	0.64380955711...
6	NP1	VD1	0.64380955711...
7	VD1	NP2	0.23426752669...
8	NP2	VD1	0.23426752669...
9	VD1	RC1	0.32030608212...
10	RC1	VD1	0.32030608212...
11	WH1	NM1	0.28607981909...
12	NM1	WH1	0.28607981909...
13	WH1	NP1	0.61104029063...
14	NP1	WH1	0.61104029063...
15	WH1	NP2	0.23939744677...

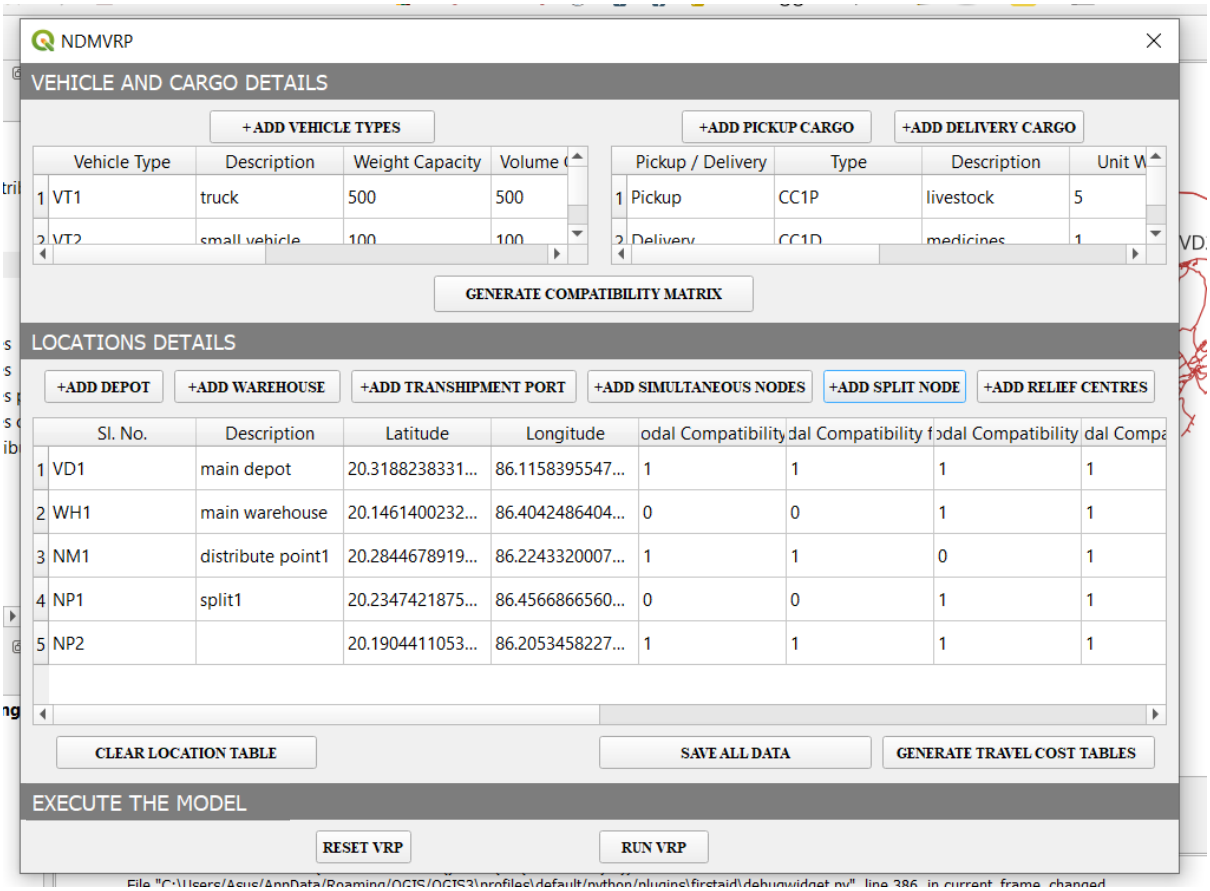
The final part has two buttons

- The **"run VRP" button** will run the heuristic and generate the result of the VRP and show the results in form of tables added as layers as well as visual representation of such results on the map canvas.

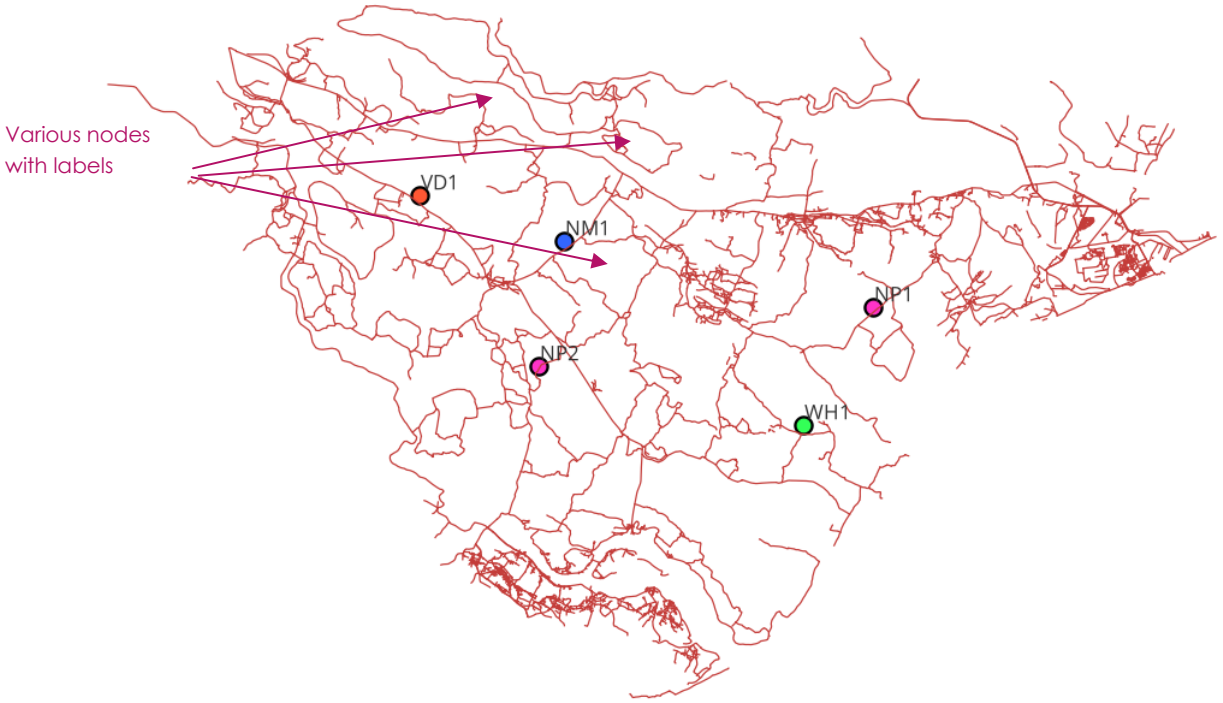
- The **“reset VRP” button** will reset all the dataframes and make the UI ready for a fresh set of inputs.

A representation of the results and its visualization is as follows:
First we see the **road networks being added to the map canvas**

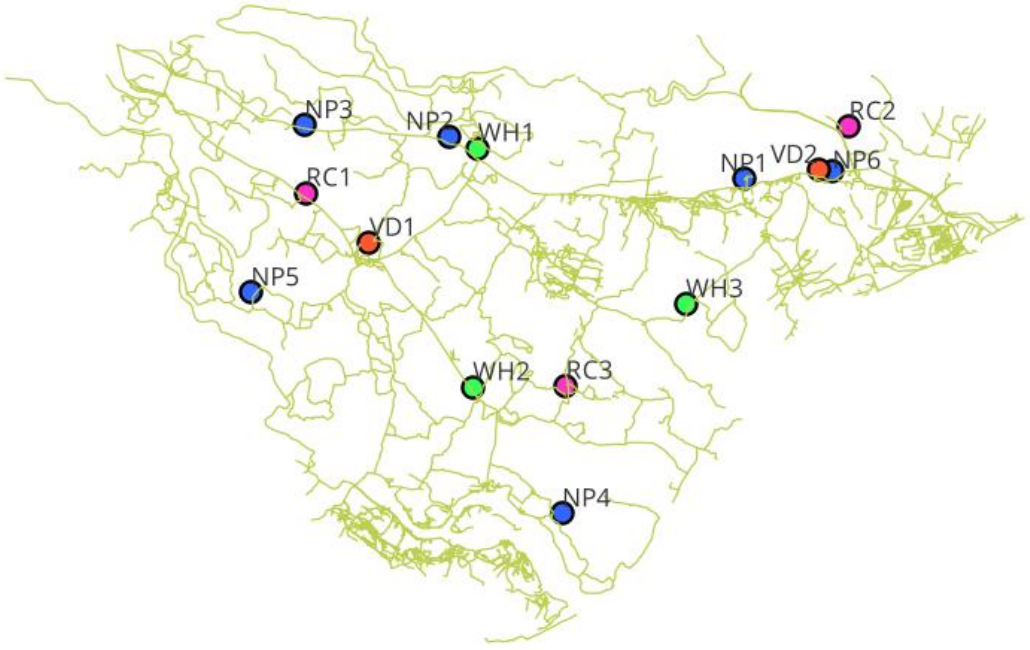
Next, we populate the plugin UI with our inputs. The filled-up UI looks as below.


NDMVRP
VEHICLE AND CARGO DETAILS
+ADD VEHICLE TYPES
+ADD PICKUP CARGO
+ADD DELIVERY CARGO
Vehicle Type
Description
Weight Capacity
Volume
1 VT1
truck
500
500
2 VT2
small vehicle
100
100
Pickup / Delivery
Type
Description
Unit W
1 Pickup
CC1P
livestock
5
2 Delivery
CC1D
medicines
1
GENERATE COMPATIBILITY MATRIX
LOCATIONS DETAILS
+ADD DEPOT
+ADD WAREHOUSE
+ADD TRANSHIPMENT PORT
+ADD SIMULTANEOUS NODES
+ADD SPLIT NODE
+ADD RELIEF CENTRES
Sl. No.
Description
Latitude
Longitude
odal Compatibility
dal Compatibility f
odal Compatibility
dal Compa
1 VD1 main depot 20.3188238331... 86.1158395547... 1 1 1 1
2 WH1 main warehouse 20.1461400232... 86.4042486404... 0 0 1 1
3 NM1 distribute point1 20.2844678919... 86.2243320007... 1 1 0 1
4 NP1 split1 20.2347421875... 86.4566866560... 0 0 1 1
5 NP2 20.1904411053... 86.2053458227... 1 1 1 1
CLEAR LOCATION TABLE
SAVE ALL DATA
GENERATE TRAVEL COST TABLES
EXECUTE THE MODEL
RESET VRP
RUN VRP

The locations added to the canvas looks like **coloured dots with labels**. They are **colour categorised** depending on the nature of the locations.



A more populated location input is shown here for reference:



The output is generated in a excel file in the local drive that looks like this:

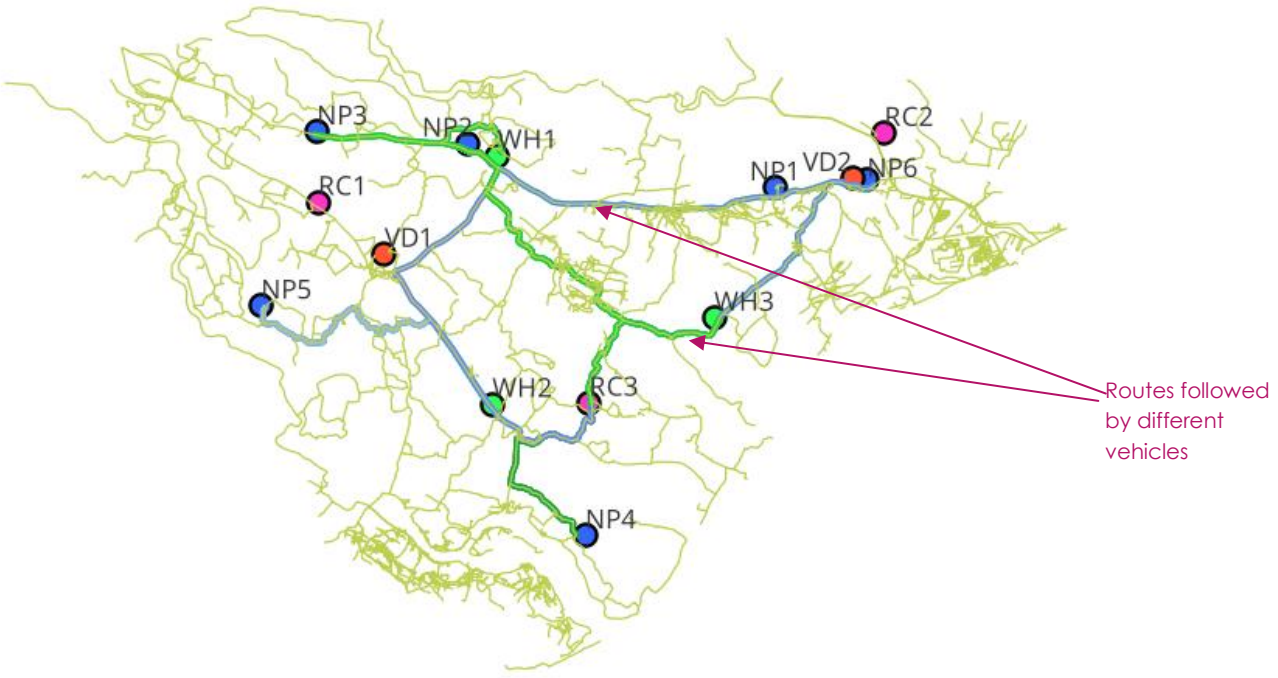
	A	B	C
1	Vehicle Unique Identity	Vertices Visited	LoadCodes at each Vertex
2	('VT1', 'VD1', 1)	['WH3', 'NP3', 'RC3', 'WH1', 'NP1', 'RC3']	[{'CC1D': 0, 'CC3D': 17, 'CC4D': 44, 'CC5D': 74, 'CC2D': 7, 'CC3P': 0, 'CC1P': 0, 'CC2P': 0, 'CC4P': 0}, {'CC1D': 0, 'CC3D': -17, 'CC4D': -44, 'CC5D': -74, 'CC2D': -7, 'CC3P': 0, 'CC1P': 1, 'CC2P': 0, 'CC4P': 97}, {'CC1D': 0, 'CC3D': 0, 'CC4D': 0, 'CC5D': 0, 'CC2D': 0, 'CC3P': 0, 'CC1P': -1, 'CC2P': 0, 'CC4P': -97}, {'CC3D': 80, 'CC2D': 0, 'CC5D': 0, 'CC4D': 0, 'CC1D': 0, 'CC3P': 0, 'CC4P': 0, 'CC2P': 0, 'CC1P': 0}, {'CC3D': -80, 'CC2D': 0, 'CC5D': 0, 'CC4D': 0, 'CC1D': 0, 'CC3P': 0, 'CC4P': 0, 'CC2P': 0, 'CC1P': 28.0}, {'CC3D': 0, 'CC2D': 0, 'CC5D': 0, 'CC4D': 0, 'CC1D': 0, 'CC3P': 0, 'CC4P': 0, 'CC2P': 0, 'CC1P': -28.0}]
3	('VT1', 'VD1', 2)	['WH1', 'NP6', 'RC3', 'WH2', 'NP2', 'RC3', 'NP1', 'RC2']	[{'CC3D': 34, 'CC5D': 0, 'CC1D': 0, 'CC4D': 0, 'CC2D': 0, 'CC1P': 0, 'CC4P': 0, 'CC2P': 0, 'CC3P': 0}, {'CC3D': -34, 'CC5D': 0, 'CC1D': 0, 'CC4D': 0, 'CC2D': 0, 'CC1P': 91, 'CC4P': 0, 'CC2P': 0, 'CC3P': 0}, {'CC3D': 0, 'CC5D': 0, 'CC1D': 0, 'CC4D': 0, 'CC2D': 0, 'CC1P': -91, 'CC4P': 0, 'CC2P': 0, 'CC3P': 0}, {'CC2D': 0, 'CC4D': 0, 'CC3D': 0, 'CC1D': 0, 'CC5D': 4.0, 'CC3P': 0, 'CC1P': 0, 'CC2P': 0, 'CC4P': 0}, {'CC2D': 0, 'CC4D': 0, 'CC3D': 0, 'CC1D': 0, 'CC5D': -4.0, 'CC3P': 0, 'CC1P': 25.0, 'CC2P': 0, 'CC4P': 0}, {'CC2D': 0, 'CC4D': 0, 'CC3D': 0, 'CC1D': 0, 'CC5D': 0, 'CC3P': 0, 'CC1P': -25.0, 'CC2P': 0, 'CC4P': 0}, {'CC3D': 0, 'CC2D': 0, 'CC5D': 0, 'CC4D': 0, 'CC1D': 0, 'CC4P': 0, 'CC1P': 47.0, 'CC2P': 0, 'CC3P': 79}, {'CC3D': 0, 'CC2D': 0, 'CC5D': 0, 'CC4D': 0, 'CC1D': 0, 'CC4P': 0, 'CC1P': -47.0, 'CC2P': 0, 'CC3P': -79}]
4	('VT1', 'VD2', 1)	['WH2', 'NP5', 'RC3', 'NP2', 'RC2']	[{'CC1D': 0, 'CC5D': 83, 'CC2D': 58, 'CC3D': 53, 'CC4D': 0, 'CC3P': 0, 'CC4P': 0, 'CC1P': 0, 'CC2P': 0}, {'CC1D': 0, 'CC5D': -83, 'CC2D': -58, 'CC3D': -53, 'CC4D': 0, 'CC3P': 0, 'CC4P': 100, 'CC1P': 79, 'CC2P': 0}, {'CC1D': 0, 'CC5D': 0, 'CC2D': 0, 'CC3D': 0, 'CC4D': 0, 'CC3P': 0, 'CC4P': -100, 'CC1P': -79, 'CC2P': 0}, {'CC2D': 0, 'CC4D': 0, 'CC3D': 0, 'CC1D': 0, 'CC5D': 0, 'CC4P': 0, 'CC3P': 63, 'CC2P': 0, 'CC1P': 0.0}, {'CC2D': 0, 'CC4D': 0, 'CC3D': 0, 'CC1D': 0, 'CC5D': 0, 'CC4P': 0, 'CC3P': -63, 'CC2P': 0, 'CC1P': -0.0}]

Vehicle ID

Nodes visited after coming from Depot

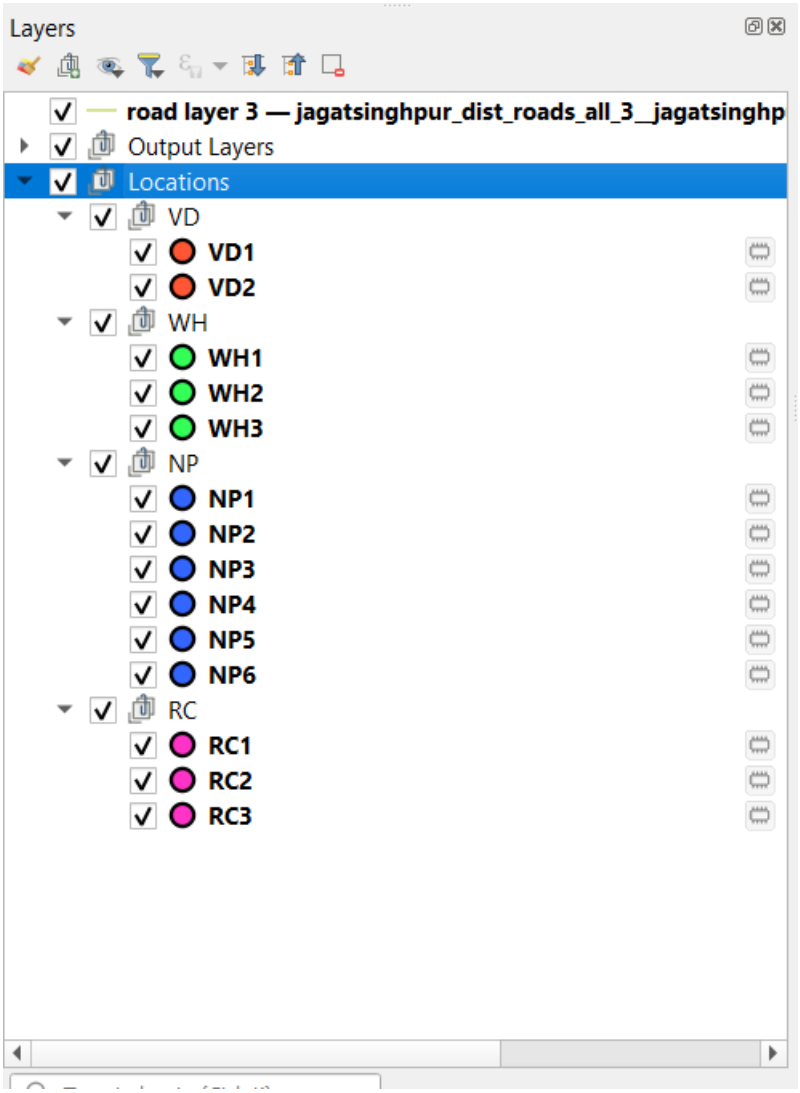
Pickup-Delivery load code at different nodes in order

The visualization of this output will be in the map canvas like below:

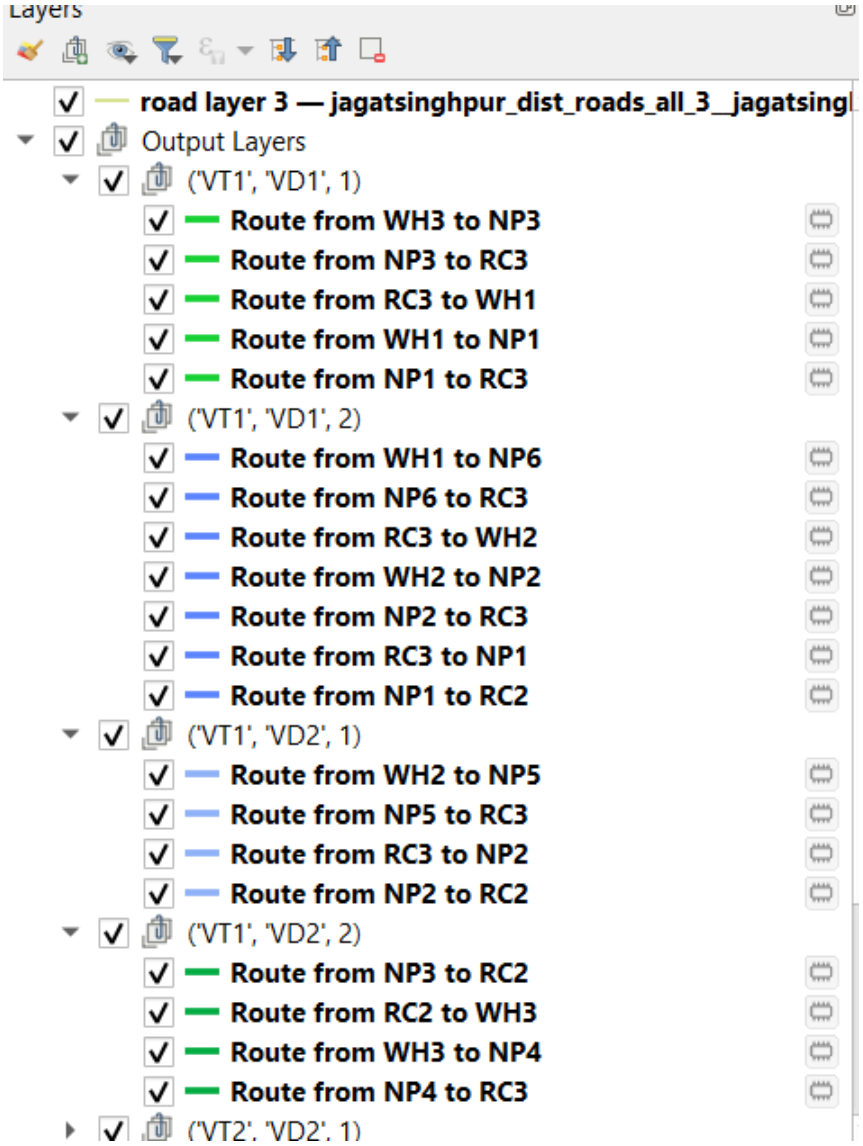


In the left layer pane, **all these locations, tables and routes will be categorized in groups** and shown as layers for the user to understand and interact with.

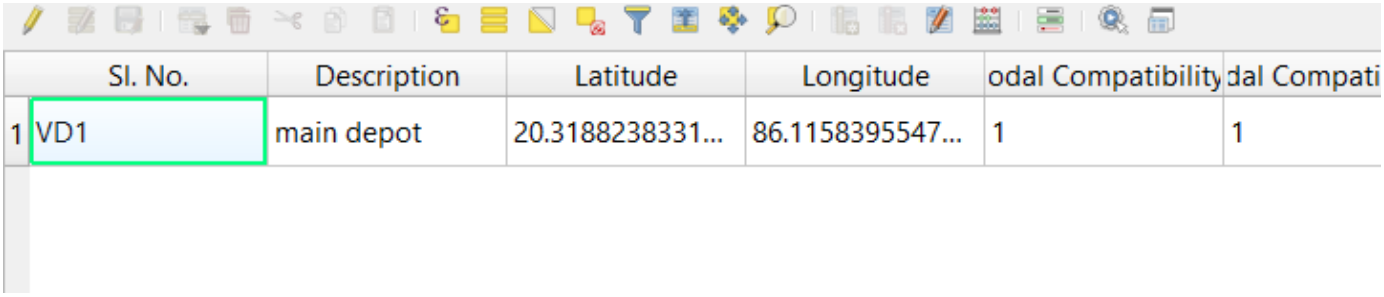
Below we can see only locations as layers



Below we can see the output routes as layers. User can check or uncheck their corresponding boxes to make them visible on the canvas as group or individually.

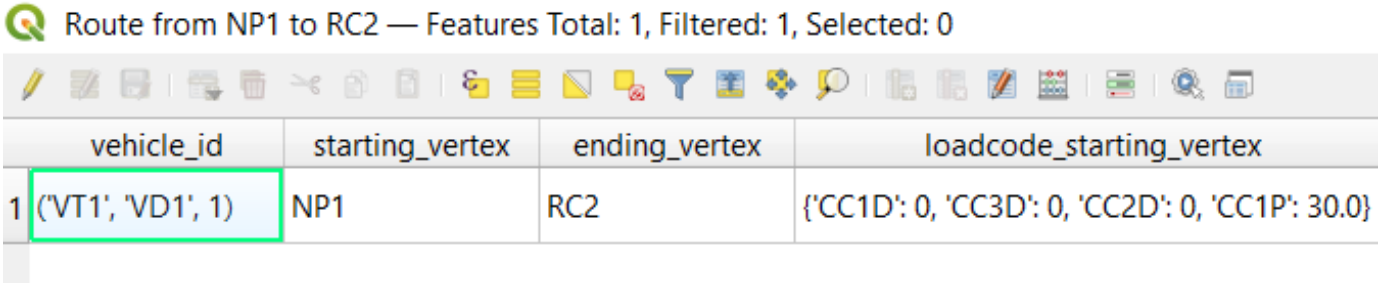


A location layer will have its attribute table that contains all its corresponding information associated as user defined earlier.



	Sl. No.	Description	Latitude	Longitude	odal Compatibility	dal Compati
1	VD1	main depot	20.3188238331...	86.1158395547...	1	1

Similarly, **an individual location layer will have its attribute table** that contains all its corresponding information associated as user defined earlier.



	vehicle_id	starting_vertex	ending_vertex	loadcode_starting_vertex
1	('VT1', 'VD1', 1)	NP1	RC2	{'CC1D': 0, 'CC3D': 0, 'CC2D': 0, 'CC1P': 30.0}

Case study

CASE STUDY 1: EFFECTS OF FANI CYCLONE IN JAGATSINGPUR ODISHA

1. INTRODUCTION-

Cyclone Fani was one of the strongest tropical cyclones to hit the Indian subcontinent in recent history. It made landfall near Puri, Odisha, on May 3, 2019, with wind speeds reaching up to 240 km/h (150 mph). Cyclone Fani affected 14 districts of Odisha, including Puri, Khurda, Cuttack, Kendrapara, Jagatsinghpur, Bhadrak, Balasore, Ganjam, Mayurbhanj, Jajpur, Nayagarh, Dhenkanal, Balangir, and Keonjhar.The cyclone caused widespread destruction, affecting millions of people and causing significant damage to infrastructure, agriculture, and livelihoods.

2. IMPACT-

- Cyclone Fani caused extensive damage to infrastructure, including housing, power lines, and roads.
- Coastal areas experienced significant storm surges, leading to flooding and destruction of crops.
- The cyclone disrupted livelihoods, particularly in the fishing and agriculture sectors, exacerbating socio-economic vulnerabilities in affected communities.
- Cyclone Fani affected over 16.5 million people across 14 districts of Odisha.
- The cyclone resulted in over 64 fatalities and caused injuries to thousands of individuals, necessitating urgent medical attention and relief assistance.
- In Jagatsinghpur district alone:
    - 8 blocks were affected
    - 256 villages were ravaged.
    - Approximately 500,000 individuals were affected.
    - 18,435 houses were reported as damaged.

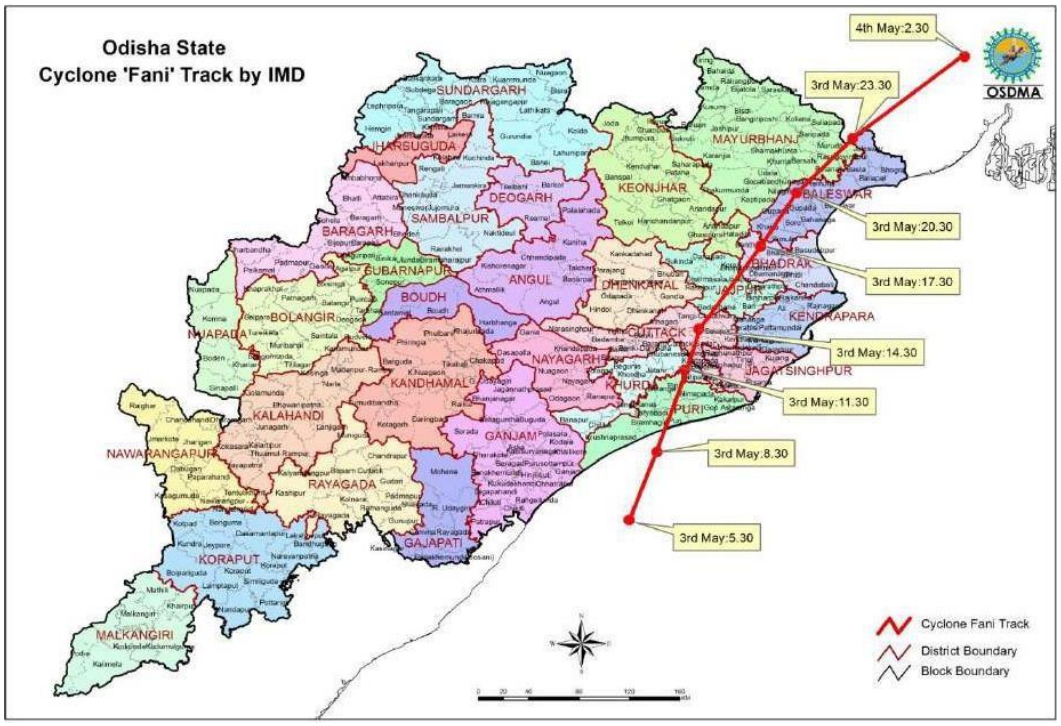
3. RELIEF EFFORTS-

- Government agencies, NGOs, and volunteers mobilized quickly to provide emergency relief, including food, water, and medical supplies, to affected populations.

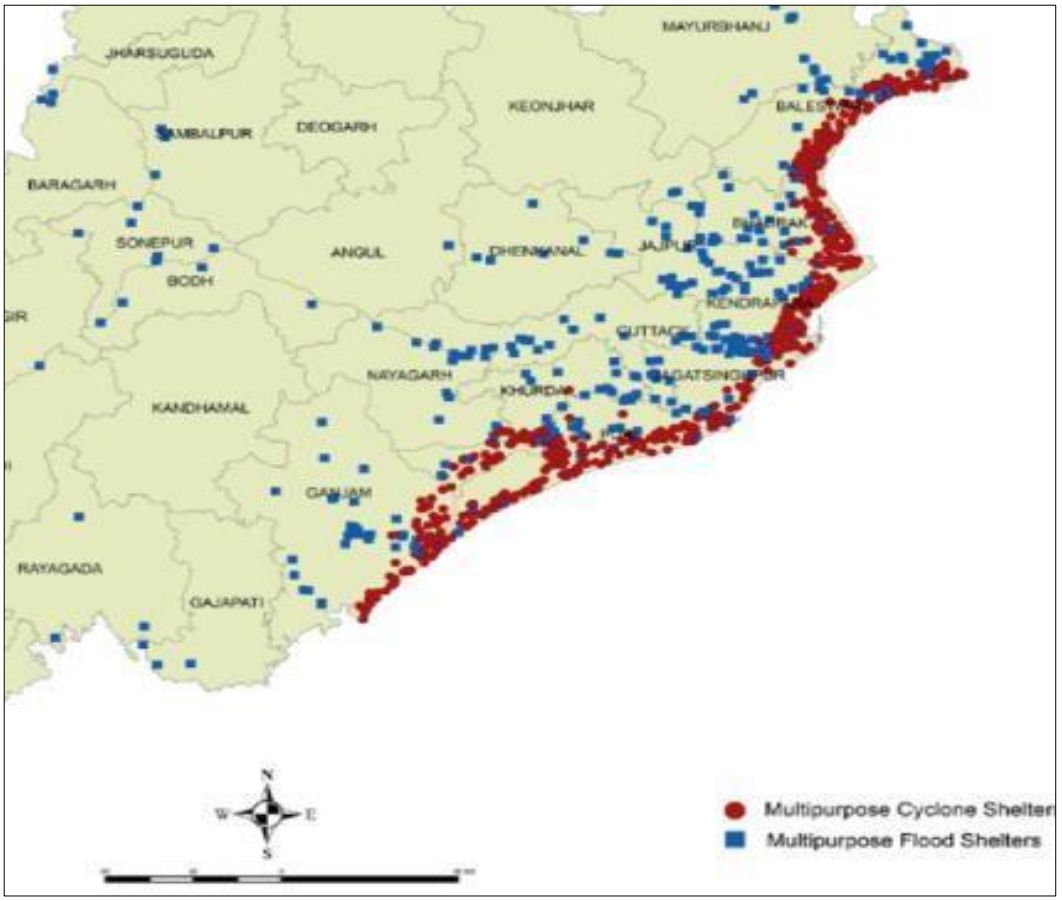
- Approximately 14 lakh people were evacuated to cyclone shelters and safe locations prior to Cyclone Fani's landfall.
- A total of 9,177 shelters, including multipurpose cyclone/flood shelters and public buildings, were utilized to accommodate evacuees.
- In Jagatsinghpur,
    - prior to Cyclone Fani's landfall, ODRAF team successfully evacuated and served over 92,326 individuals from vulnerable coastal areas to cyclone shelters and safe locations.
    - Over 1 lakh dry food packets were prepared and kept ready for distribution to affected populations.
    - Free kitchens were set up to provide cooked food to evacuees and affected communities.
    - 40 water tankers and 32 lakh water pouches were stocked to supply drinking water to affected people.

4. TIMELINE-

- April 30, 2019: Formal evacuation efforts began, with authorities issuing orders and mobilizing resources.
- May 1, 2019: Evacuation efforts commenced as district collectors were instructed to start evacuating vulnerable populations living near the coast or in low-lying areas across 17 districts.
- May 2, 2019: By 6:00 PM, approximately 5,46,586 people had been evacuated and accommodated in 3940 shelter or school buildings, with 3194 free kitchen centers opened.
- 0.45 million polythene sheets were prepositioned at district and sub-district levels, with an additional 1.5 lakh sheets kept in buffer stock at the state headquarters.
- A record 1.2 million people were evacuated in less than 48 hours, with almost 7,000 kitchens made functional overnight to cater to 9,000 shelters.



Path of cyclone Fani



Relief Centres in Orissa

5. SIMULATION DATA –

Assumptions-

- Avg. weight of person – 60 kg
- Weight of relief kit – 2kg
- Capacity of vehicles carrying relief kit and humans - 400kg to 2000 kg
- 8- 10% of the population needs to be evacuated
- 1 person would require 1 relief kit
- 1 person would require 1 food package
- 1 person would require 2 l water pouch

* The above assumption are made after referring to multiple research papers.

Vehicle –

Vehicle Type	Weight Capacity	Volume Capacity
Big-sized vehicle (Bus/Truck)	2000 kg	35000 litre
Mid-sized vehicle (Tempo)	1050 kg	15000 litre
Small-sized vehicle (Van,Ambulance)	400 kg	5000 litre

Cargo –

Cargo Type	Pickup/Delivery	Weight/unit	Volume/unit
People	Pickup	60 kg	80 litre
Food packets	Delivery	1 kg	2 litre
Water Pouch	Delivery	1 kg	1 litre
Relief kit	Delivery	2 kg	3 litre

Nodes –

(Shelters, Schools, College, Community Centre etc)

No.	Nodes (no. of people gathered)	Evacuated (8-10%)	Food packet	Water pouch	Relief kit
1	235	22	213	426	213
2	300	31	275	550	272
3	221	19	203	406	200

4	240	27	215	430	215
5	198	21	178	356	179
6	275	30	245	490	245
7	312	32	284	568	285

* There were 396 relief centres and around 92326 people were rescued, so around 233 people per node

Depot –

(Bus stations/ Ambulance stations etc)

No.	Description	Vehicle type 1	Vehicle type 2	Vehicle type 3
VD1	depot 1	5	6	6
VD2	depot 2	6	8	8

Warehouse –

(Indoor stadium/ Govt. Warehouse etc.)

No.	Description	Delivery Cargo type 1	Delivery Cargo type 2	Delivery Cargo type 3
WH1	warehouse 1	2500	3000	2000
WH2	warehouse 2	1500	1800	2000

Relief Centre/ Hospitals –

No.	Description	Relief Centre capacity
RC1	Relief centre 1	85
RC2	Relief centre 2	115

Map Input –

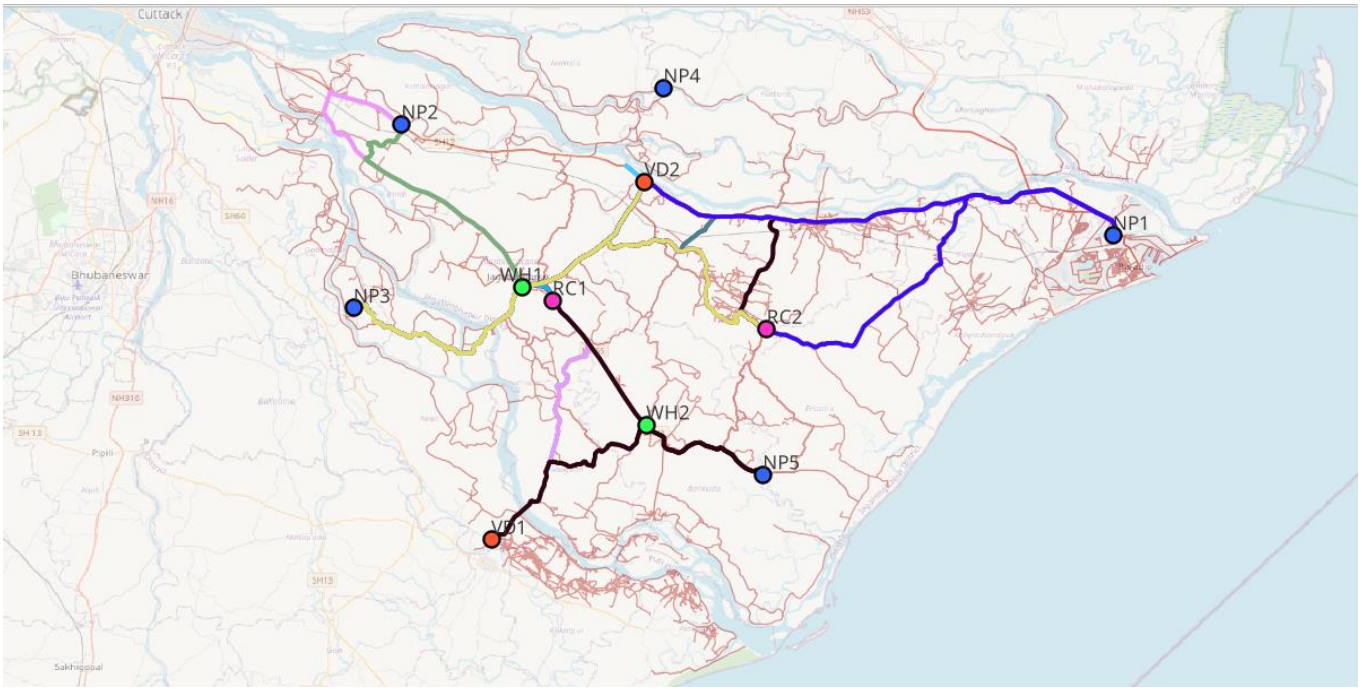
Road layer 1

Road layer 2

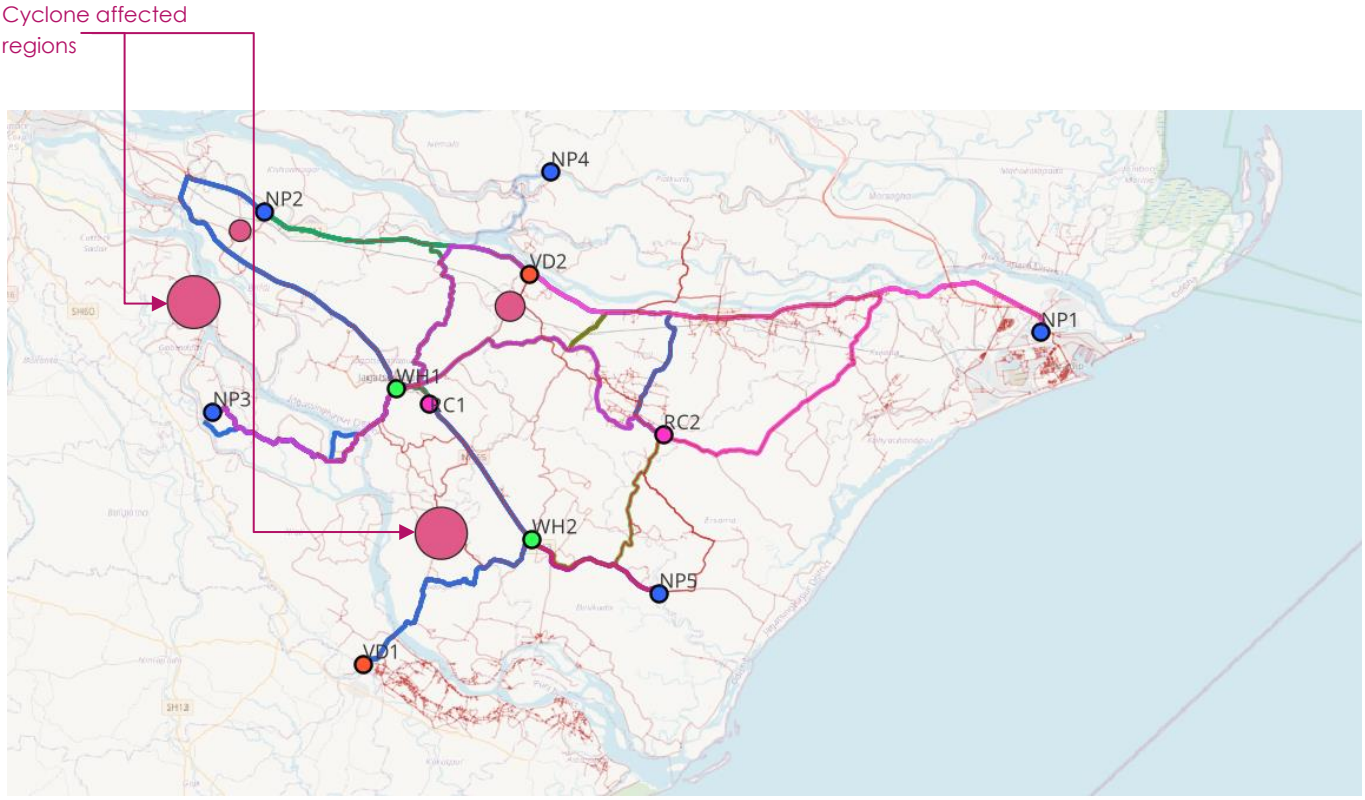
Road layer 3

6. RESULT –

We run the same logistics data for two cases (before and after cyclone) where only the road network changes but all other data including locations remain identical.



Canvas results in Jagatsinghpur (before disaster, original road network)



Canvas results in Jagatsinghpur (after disaster, modified road network)

The Pink circles represent regions where the disaster has resulted in closure of roads

CASE STUDY 2: CHENNAI FLOOD 2015

1. INTRODUCTION-

Chennai, the capital city of Tamil Nadu in India, faced one of its most devastating calamities in December 2015 when heavy rainfall inundated the region, leading to widespread flooding. The flood, caused by incessant rains and the overflowing of water bodies, severely affected millions of residents, causing loss of life, displacement, and extensive damage to property and infrastructure.

2. IMPACT-

- Over 4 million residents affected; death toll around 422 in Chennai alone.
- Economic damage estimated at around 3 billion USD, ranking it among the costliest disasters globally.
- Approximately 165 out of Chennai's 200 wards submerged; water levels exceeding 2 meters in some areas.
- Closure of Chennai International Airport, cancellation of train services, and suspension of school activities.
- 1.8 million people rendered homeless; over 3 million families suffered damage to their homes.
- More than 100,000 structures were damaged because of floods.

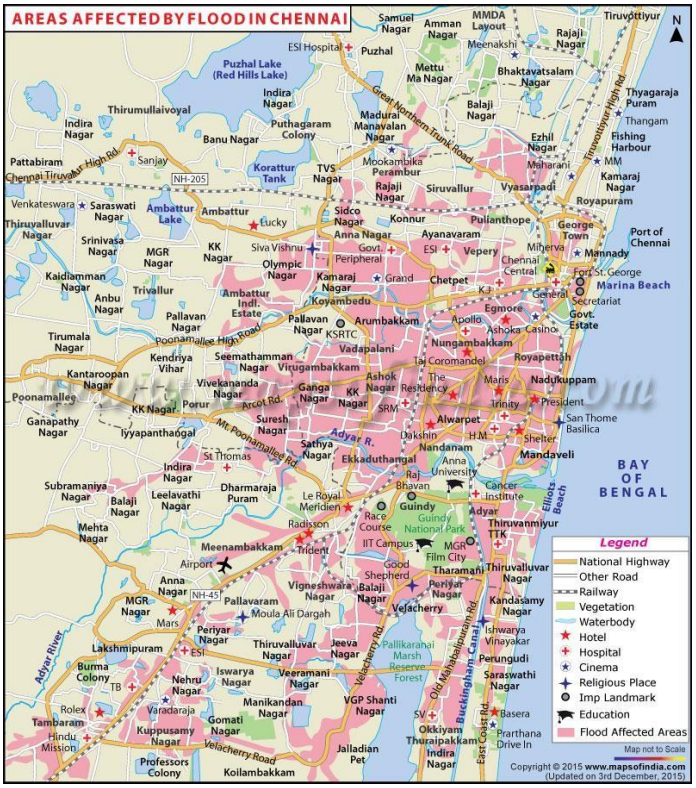
3. RELIEF EFFORTS-

- Greater Chennai Corporation (GCC) set up 176 relief centers across the city, including community halls, schools, and public buildings.
- Served 1.2 lakh individuals, providing temporary accommodation, food, and other essential services for more than 20 days.
- Relief material distribution center established at Jawaharlal Nehru Indoor Stadium.
- Relief materials distributed:
    - Food Packets: 1,99,244
    - Mats: 32,992
    - Bed Sheets: 2,07,433
    - Milk: 12,500 packets
    - Water Bottles: 54,448
    - Rice Bags: 2,122
    - Medicine Boxes: 3,004
    - Water: 2,38,568 liters
- Supplies crucial for meeting immediate needs of affected population, ensuring access to food, clean water, shelter, and

medical assistance.

4. TIMELINE-

- December 2-3, 2015: With the flooding reaching catastrophic levels, authorities step up preparations for evacuation and rescue operations. Emergency response teams, including the Indian Army, NDRF, and local police, are put on standby.
- December 4-5, 2015: As the floodwaters continue to rise, the pre-evacuation process intensifies. Vulnerable communities, particularly those living in low-lying areas and along riverbanks, are identified for priority evacuation.
- December 6, 2015: With the situation becoming increasingly critical, mass evacuation orders are issued for several flood-affected areas. Authorities use various communication channels, including radio, television, and social media, to alert residents about the evacuation orders and provide guidance on safety measures.
- December 7-8, 2015: The pre-evacuation process reaches its peak as thousands of residents in flood-prone areas are evacuated to safer locations, including relief camps and shelters set up by the government and NGOs. Efforts are made to ensure the orderly and safe evacuation of vulnerable populations, including the elderly, children, and people with disabilities.



5. SIMULATION DATA-

Assumptions-

- Avg. weight of person – 60 kg
- Weight of relief kit – 2kg
- Capacity of vehicles carrying relief kit and humans - 400kg to 2000 kg
- 8- 10% of the population needs to be evacuated
- 1 person would require 1 relief kit
- 1 person would require 1 food package
- 1 person would require 2 l water pouch

* The above assumption are made after referring to multiple research papers.

Vehicle –

Vehicle Type	Weight Capacity	Volume Capacity
Big-sized vehicle (Bus/Truck)	2000 kg	35000 litre
Mid-sized vehicle (Tempo)	1050 kg	15000 litre
Small-sized vehicle (Van, Ambulance)	400 kg	5000 litre

Cargo –

Cargo Type	Pickup/Delivery	Weight/unit	Volume/unit
People	Pickup	60 kg	80 litre
Food packets	Delivery	1 kg	2 litre
Water Pouch	Delivery	1 kg	1 litre
Relief kit	Delivery	2 kg	3 litre

Nodes –

(Shelters, Schools, College, Community Centre etc)

No.	Nodes	Evacuated (8-10%)	Food packet	Water pouch	Relief kit
1	695	56	639	1278	639
2	680	59	621	1242	621
3	720	62	658	1316	658
4	645	49	596	1192	596
5	710	60	650	1300	650
6	650	65	585	1170	585
7	670	69	601	1202	601
8	705	72	633	1266	633
9	685	51	634	1268	634
10	560	48	512	1024	512
11	653	65	588	1176	588
12	750	80	670	1340	670

* There were 176 relief centres and around 120000 people were rescued, so around 688 people per node

Depot –

(Bus stations/ Ambulance stations etc)

Sl. No.	Description	Vehicle Type 1	Vehicle Type 2	Vehicle Type 3
VD1	Depot 1	3	4	4
VD2	Depot 2	2	5	5
VD3	Depot 3	3	3	3

Warehouse –

(Indoor stadium/ Govt. Warehouse etc.)

No.	Description	Delivery Cargo type 1	Delivery Cargo type2	Delivery Cargo type 3
WH1	warehouse 1	6000	13000	5500
WH2	warehouse 2	6000	15000	5000

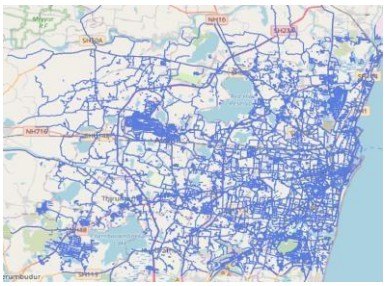
Relief Centre/ Hospitals –

No.	Description	Relief Centre capacity
RC1	Relief centre 1	600
RC2	Relief centre 2	300
RC3	Relief centre 3	300

Map Input-

Road layer 1

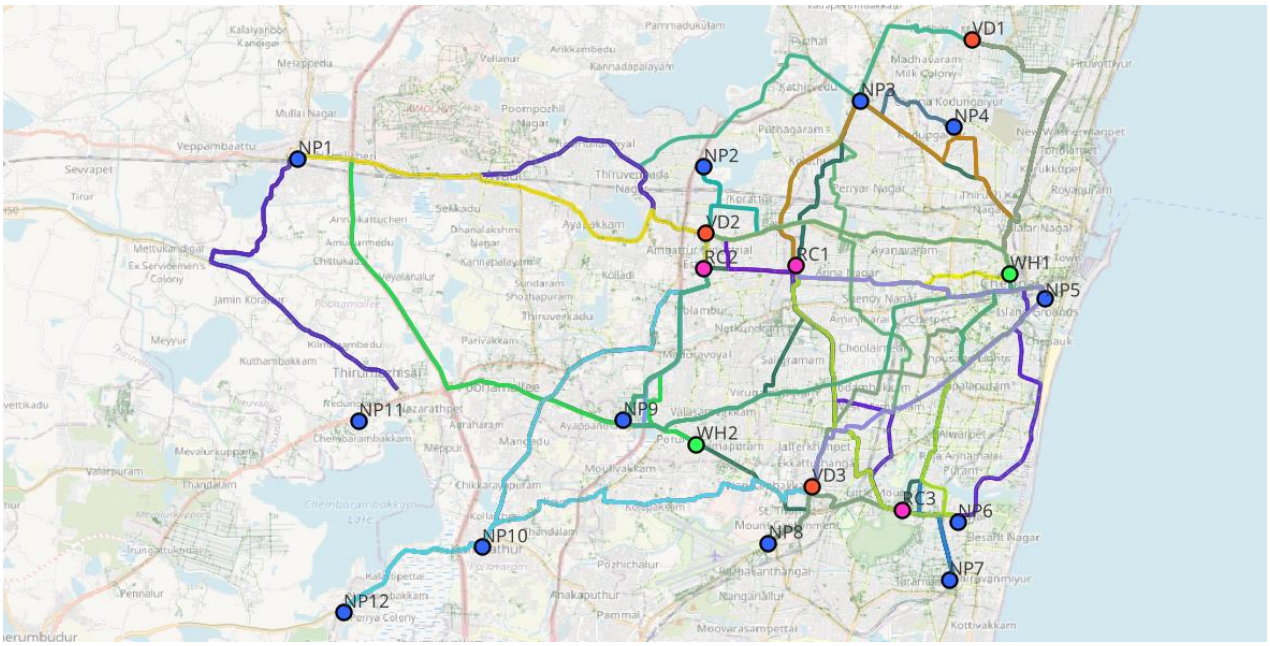
Road layer 2



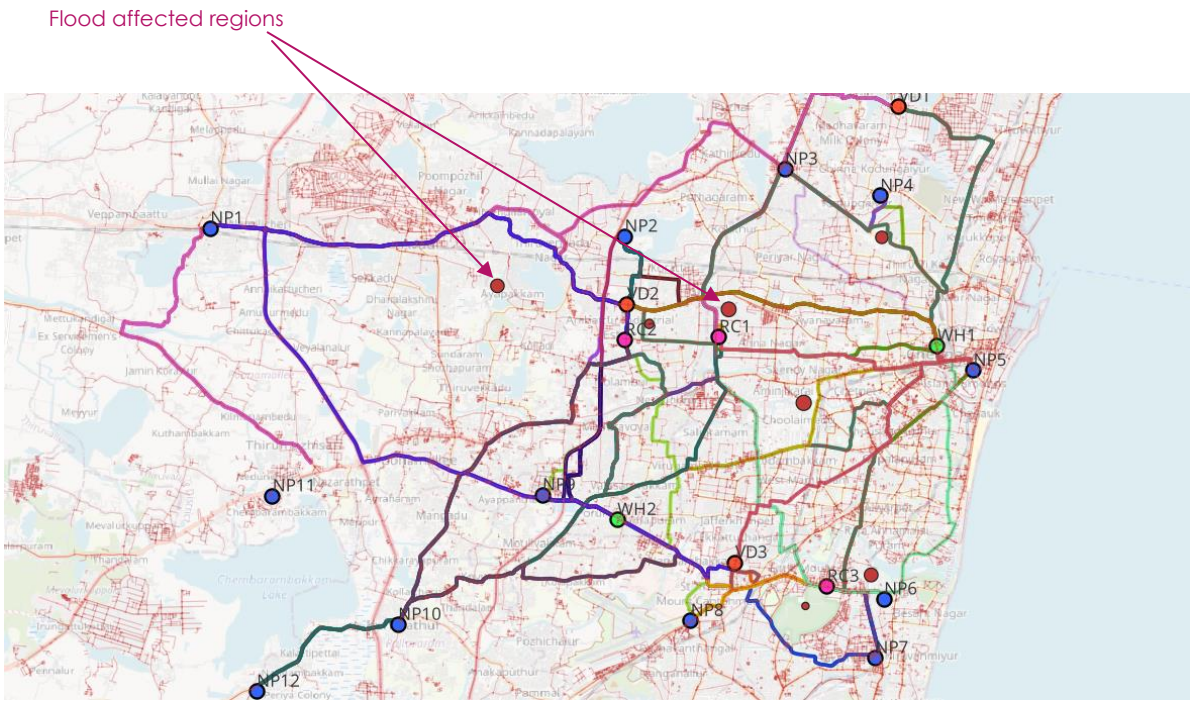
Road layer 3

6. RESULT-

We run the same logistics data for two cases (before and after flood) where only the road network changes but all other data including locations remain identical.



Chennai flood VRP with original road network



Chennai flood VRP with modified road network

Pink circles are flood affected segments of the city where roads have been closed/inoperable.

Thank You for using our

indigenously developed

NDMVRP

QGIS Plugin