Report On

Development of QGIS Plugin for Land Records Information System

Submitted by:
PRIYANSHI PAL
Roll No.-2142514
ABMCS21004

For the award of the degree of:
M.Sc (Computer Science)

Under the supervision of:
DR. GANESH KHADANGA
Sr. Technical Director (Scientist – F)
Land Resource Information System
National Informatics Centre, New Delhi

Department of Mathematics and Computing

Banasthali Vidyapith

Banasthali - 304022

ABSTRACT

The document intends to record the description, requirements and goals of the Plugin Creation and plot information, by defining the problem statement in detail at a functional level. The Ministry of Electronics and Information Technology (MeitY) is an executive body of the Union Government for India. It is responsible for IT policy, strategy and development and also for the updation and growth of India's strength and economy in technology. The Government of India needs to maintain land records in India. These records are necessary for the government as it helps the government in planning its welfare schemes. Not only the government, but keeping these records also helps individuals as now their plot's data will be exactly stored with its location, boundary, area, owner information, date of acquiring it etc. into the national record and this will prevent frauds. NIC, being the technology partner of the Government of India, is currently working on these kinds of shape files which will contain all this information.

This report presents a collection of QGIS plugins developed to enhance the land record system and streamline land-related operations. The plugins offer a range of essential functionalities for efficient land management and data access. The first plugin focuses on generating unique id for plot based on latitude and longitude coordinates, ensuring accurate identification and preventing land fraud. The second plugin enables users to split a plot into multiple parts, providing interactive features such as highlighting and zooming in on specific plots, selecting splitting points using vertices, and offering options for precise distance-based splitting. The third plugin facilitates the merging of two plots into a single plot, featuring a user-friendly interface for selecting plots, displaying a preview for verification, and ensuring that the merging operation is only performed on adjacent plots.

Furthermore, the report introduces a website designed for the land record system of Arunachal Pradesh. The website incorporates user authentication with username and password verification, employing a table in pgAdmin for storing admin information. The inclusion of a captcha ensures secure access to the website. The website offers functionalities such as searching for land records based on owner name, land type, or plot number. Users are presented with dropdown menus for selecting search criteria and are subsequently redirected to the respective pages displaying relevant plot information on interactive maps. The website provides a seamless experience,

allowing users to navigate through different functionalities while ensuring data integrity and transparency in land record management. We also have developed a small management system wherein we have a table that stores all information about a particular plot which can be saved into or retrieved from that database using a website. This is currently a test version and only has records for the state of Arunachal Pradesh.

These plugins and the website collectively contribute to an efficient and user-friendly land record system, providing tools for generating unique identifiers, splitting and merging plots, and facilitating easy access to land information. The incorporation of geospatial functionalities enhances visualizations and supports informed decision-making in land-related activities.

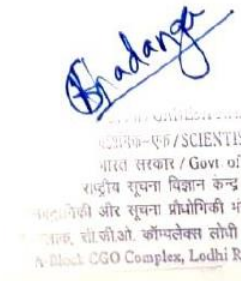
सत्यमेव जयते

भारत सरकार
इलेक्ट्रॉनिकी और सूचना प्रौद्योगिकी मंत्रालय
राष्ट्रीय सूचना-विज्ञान केन्द्र
ए-ब्लॉक, केन्द्रीय कार्यालय परिसर, लोधी रोड,
नई दिल्ली-110 003
Government of India
MINISTRY OF ELECTRONICS & INFORMATION TECHNOLOGY
NATIONAL INFORMATICS CENTRE
A-BLOCK, C.G.O. COMPLEX, LODHI ROAD
NEW DELHI-110 003
ग्राम्स / GRAMS : INCENT HQ
फैक्स / FAX :
ई-मेल / E-mail :

Date: 10/06/2023

Certificate

Certified that Priyanshi Pal has carried out the project work titled "Development of QGIS Plugins for Land Records Information System", from 26th December 2022 to 10th June 2023 at NIC, New Delhi under my supervision for the award of the M.Sc (Computer Science) from Banasthali Vidyapith. The thesis embodies result of original work and studies carried out by the student herself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else.



Ganesh Khadanga
(Name of Supervisior)
Designation: Scientist F
Place: New Delhi
Date:10/06/2023

ACKNOWLEDGEMENT

It is my privilege to express our sincerest regards to our project supervisor, **Dr Ganesh Khadanga**, for his valuable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of the internship. I also feel thankful and express my kind gratitude towards our placement cell for giving us the opportunity to work under the National Informatics Centre, New Delhi. I deeply express my sincere thanks to our Head of the Department, **Prof. C.K. Jha** for encouraging and allowing us to participate in the working of an organization and to present our work there on the topic "**Development of QGIS Plugin for Land Records Information System"** at our department for partial fulfilment of the requirement leading to the award of **M.Sc Computer Science.** I thank all teammates for their positive support and guidance. I pay my deep respect to my parents, other family members and friends for their love and encouragement throughout my career.

While making the project file I was able to amass knowledge about various topics. Throughout the duration of the project internship, I had immense learning experience and I also inculcated many personal qualities during this process like responsibility, punctuality, confidence, stewardship, integrity amongst others.

I would also like to thank my teachers who supported me all the time, cleared my doubts and also a big thanks to my parents who supported me and kept motivating me. I would take this opportunity to acknowledge their support and wish they keep supporting me in future.

A project is a bridge between theoretical and practical learning and keeping this in mind I worked on the project and made it successful due to timely support and efforts of all who helped us.

Once again, I would like to thank my classmates and friends for their encouragement and help in designing and making this report creative. I owe a tremendous amount of gratitude to all of them. Only because of all the above-mentioned people I was able to create this project and make the internship experience good and enjoyable.

TABLE OF CONTENTS

CHAPTER 1 - OBJECTIVE

1.1 INTRODUCTION

The National Informatics Centre (NIC) is a department of the Government of India under the Ministry of Electronics and Information Technology (MeitY). To deliver government services to citizens and facilitate Digital India initiatives. The Ministry of Electronics and Information Technology (MeitY) is an executive body of the Union Government for India. It is responsible for IT policy, strategy and development and also for the updation and growth of India's strength and economy in technology. The National Informatics Centre (NIC) is a premier government organization in India that focuses on providing e-governance solutions and implementing advanced technology initiatives. With its extensive experience and expertise in delivering digital services, NIC plays a pivotal role in modernizing administrative processes and enhancing government services across various sectors.

This report provides an overview of the QGIS plugins and website developed for the land record management system under the National Informatics Centre (NIC), Ministry of Electronics and Information Technology (MeiTy) in India. QGIS is a powerful geospatial software and the necessity of leveraging its capabilities for efficient land record management.

QGIS, an open-source Geographic Information System (GIS) software, is a key component in the arsenal of digital tools employed by NIC for land record management. It offers a wide range of features, allowing users to visualize, analyse, and manipulate geospatial data with precision. QGIS supports a variety of data formats, making it versatile and compatible with different data sources. The software's user-friendly interface and extensive plugin library enable customization and the development of specialized tools tailored to specific requirements.

Efficient land record management is crucial for a variety of reasons. It ensures accurate documentation of land ownership, facilitates land transactions, prevents disputes, and supports sustainable land use planning. However, traditional paper-based systems often suffer from inaccuracies, inefficiencies, and difficulties in

accessing and updating records. Therefore, there is a pressing need to digitize and streamline land record management processes.

The QGIS plugins and website developed by NIC address these challenges by harnessing the power of geospatial technology. These tools automate various tasks, such as generating unique identification numbers for land parcels, splitting and merging plots, and providing user-friendly interfaces for data retrieval. The website complements the plugins by offering a centralized platform for land record information, accessible to authorized users. By integrating QGIS capabilities with land record management, the NIC aims to enhance transparency, accuracy, and efficiency in maintaining and accessing land records.

In conclusion, this report highlights the importance of leveraging QGIS plugins and a dedicated website for effective land record management. With the expertise of NIC and the versatility of QGIS, these tools empower government authorities, land administrators, and citizens with streamlined processes, accurate data, and improved decision-making capabilities. The following sections of the report will delve into the detailed functionalities and features of the QGIS plugins and website, demonstrating their contributions to modernizing land record management in India.

The traditional method of maintaining land records suffers from several disadvantages. Firstly, the reliance on paper-based documentation makes the records vulnerable to damage, loss, and deterioration. This can lead to data loss and inaccessibility. Secondly, the manual process of retrieving specific land records is time-consuming and inefficient, resulting in delays and administrative bottlenecks. Moreover, the chances of human errors in data entry and record keeping are higher, leading to inaccuracies and potential legal disputes. Lastly, limited accessibility to physical records hinders stakeholders and citizens from easily accessing and verifying land information, impeding transparency and causing inconvenience.

The implementation of an online land record system and its manipulation offers significant benefits to government officials compared to traditional manual record handling. Firstly, the online system provides a centralized and easily accessible repository of land records, eliminating the need for officials to physically search through stacks of paper documents. This saves valuable time and effort, allowing officials to efficiently retrieve and verify land information with just a few clicks.

Moreover, the online system ensures data accuracy and integrity by minimizing the risk of human errors and document loss. Information is entered, stored, and updated electronically, reducing the chances of transcription mistakes and document misplacement. This enhanced accuracy helps officials make informed decisions based on reliable and up-to-date data, leading to more effective land management practices. The system also enables seamless collaboration among government officials involved in land administration. Multiple users can access and update land records simultaneously, promoting real-time data sharing and coordination. This facilitates interdepartmental communication, reduces administrative bottlenecks, and improves overall efficiency in decision-making processes.

Furthermore, the online system enhances transparency and accountability. The digital trail of record manipulation allows for tracking and auditing changes made to land records, reducing the potential for fraudulent activities. The system maintains an audit log, capturing details of each transaction and user activity, which can be used for accountability purposes and dispute resolution.

Additionally, the online system offers advanced search and reporting capabilities. Officials can perform targeted searches based on specific criteria, such as owner names, land types, or plot numbers, retrieving relevant information quickly and accurately. This expedites the process of retrieving specific land records, enabling officials to respond promptly to queries and requests from citizens and stakeholders.

Overall, the online land record system and its manipulation empower government officials with streamlined processes, enhanced data accuracy, improved collaboration, and increased transparency. By embracing digital technologies, officials can effectively address the challenges associated with manual record handling, ensuring efficient land administration, and contributing to sustainable development.

This document intends to record the description, requirements and goals of the Plugin creation and plot information, by defining the problem statement in detail at a functional level.

CHAPTER 2 – REQUIREMENT ANALYSIS(SRS)

A. Requirement Specifications

Some requirements for the project are specified below:

1. **Functional Requirements:** These are the features and functions that developers must implement to enable users to accomplish their tasks effectively. Functional requirements for this project are:
   a) Input for plugin should be a. shapefile.
   b) Input for the import function of the website should be a csv file.
   c) Input for the import function of the website should be a kml file.
   d) Path specified for output file in the plugin should be editable by the current user.
   e) When the plugin shows the line associated with the feature selected, it should be clearly highlighted, visible and selected, so that only that feature is taken into consideration.
   f) The user must know how to run a python code in an IDE.
   g) The user must know how to run a plugin in QGIS and also how to edit a vector file in QGIS.
   h) It is essential for users to adequate knowledge for using PGAdmin, an open-source administration and development platform for PostgreSQL databases.
   i) Users must be familiar with the XAMPP server to maintain the website database.
   j) Users should be familiar with the structure of the table to avoid any redundancy or inconsistency in data from being uploaded at the time of import.

2. **Non-functional Requirements**: Non-functional requirements are not related to the system functionality, but rather define how the system should perform. Non-functional requirements for this project are:
   a) Robust security measures and data protection.
   b) Strong authentication mechanisms to ensure only authorized individuals can access and modify land records.
   c) Ensures the integrity and confidentiality.

d) The User Interface for the website must be simple, understandable, and consistent.

e) It should be designed to handle a growing number of users and increasing database sizes without experiencing significant performance degradation.

3. **Business Requirements**: Business Requirements: Business requirements are overarching objectives and desired outcomes set by an organization. The business requirements for this project are:

a.) Plugin, which has a human intervention part should be tried to be reduced and focus should be more on making the automated part accurate and precise.

b.) The website and database for land record management systems must only be accessible by the government officials to protect the integrity of the system and prevent unnecessary theft or data tampering.

c.) The plugins and website must be merged so that when adding a record to the land record management system, its location and image can also be added from which the plugin automatically detects and saves the boundaries' information for later use. So that next time whenever someone searches for that land, not just its text information but also, it's map, location, boundary and other geographical information is also available.

B. Hardware and Software Requirements

1. Hardware requirements are:

I. *For Plugin*

a. At least a 32-bit computer is required.

b. A GPU is required for faster execution of code.

c. If using Anaconda, a minimum 3 GB disk storage is required to download and install and for visual or bracket 1GB disk storage is required.

II. *For QGIS*:

a. At least 4 Gb of RAM is required to run QGIS smoothly.

b. Windows 7 or above should be installed.

c. Intel i3 core 1.3 GHz processor or above is required or an equivalent of the same non-Intel processor.

d. At least 1Gb RAM graphic card and 500Gb SATA hard drive are required.

III. *For Land record management system:*

a. At least a 32-bit computer is required.

b. Intel Pentium 4 processor or later is required or an equivalent of the same non-Intel processor.

c. At Least 2 GB minimum RAM is required but 4 GB RAM is recommended

2. Software requirements are:

a. QGIS3 version 3.20 or higher is required.

b. Python 2.7, 3.4, 3.5 or 3.6 or above should be installed.

c. Any IDE that supports python is required. (E.g., Anaconda, Bracket, VSCode, Google Collab etc.) This is required only until code is not incorporated into a QGIS plugin.

d. To save the records, we have chosen SQL database. For that we need an XAMPP server.

e. Any IDE that supports PHP is required. (E.g., Visual studios, Bracket, etc.)

C. Feasibility Study

1.Operational Feasibility

The operational feasibility of the project is based on its ability to effectively integrate into the existing land record management system and improve overall operational efficiency. The plugins and website are designed with user-friendliness in mind, ensuring that government officials and users can easily access, manipulate, and retrieve land records. The automation of tasks, such as unique id generation, plot splitting, merging, and land record search, simplifies complex processes and reduces the time required for administrative tasks. This facilitates faster decision-making and enhances the overall efficiency of land record management. The availability of training and support resources further enhances the operational feasibility of the project. User training programs can be conducted to familiarize government officials and users with the functionalities of the plugins and website, ensuring a smooth transition from the traditional manual record system to the online platform. User-friendly interfaces, intuitive navigation, and comprehensive documentation can be provided to support users in efficiently utilizing the system. Additionally, ongoing technical support and troubleshooting services can be made available to address any

issues or concerns that may arise during system usage. The project aligns with the objectives of the land record management system, aiming to improve operational efficiency, data accuracy, and service delivery. The digital platform enables easier access to land records, reducing the time and effort required to retrieve information. It also enhances data accuracy by minimizing human errors and providing validation mechanisms. Furthermore, the system allows for seamless integration with existing processes, ensuring that government officials and users can perform their tasks effectively and efficiently. Overall, the operational feasibility of the project stems from its ability to enhance the speed, accuracy, and accessibility of land record management processes, resulting in improved service delivery and operational effectiveness.

2. Technical Feasibility

The technical feasibility of the project is supported by the capabilities of the QGIS platform and the advanced tools and technologies available for plugin development. QGIS provides a robust and extensible framework for geospatial data processing and analysis, making it an ideal platform for implementing the land record management system. The QGIS API (Application Programming Interface) offers a wide range of functionalities, such as spatial data manipulation, attribute querying, and cartographic rendering, which can be leveraged to develop the required plugins. The project's technical feasibility is further enhanced by the availability of a wide range of libraries, plugins, and development resources that can be utilized to enhance the functionality of the system. QGIS supports various programming languages, including Python, which is widely used for plugin development. This allows developers to leverage existing libraries and tools, making the development process more efficient and cost-effective. Furthermore, the project can take advantage of other technologies and databases to enhance its technical feasibility. For example, the integration with pgAdmin, a popular open-source database management system, enables efficient storage, retrieval, and querying of land records. This ensures that the system can handle large volumes of data effectively and provide fast and accurate responses to user queries. The scalability and interoperability of the QGIS platform also contribute to the technical feasibility of the project. QGIS supports various data formats and standards, allowing seamless integration with other geospatial systems

and data sources. This facilitates data exchange and collaboration with external stakeholders, such as government departments, survey agencies, and land management authorities.

3. Economic Feasibility

The economic feasibility of the project is an essential aspect to consider, as it determines the financial viability and sustainability of implementing the land record management system. Several factors contribute to the economic feasibility of the project. Firstly, the initial investment required for developing the plugins and website can be considered. This includes the cost of hiring skilled developers or allocating internal resources for development, acquiring necessary hardware and software infrastructure, and conducting user training programs. By carefully planning the project and leveraging open-source technologies, the initial investment can be optimized, reducing the overall cost of implementation. Additionally, the long-term cost savings achieved through the automation and digitization of land record management processes contribute to the economic feasibility of the project. The plugins and website streamline administrative tasks, reduce manual data entry errors, and improve operational efficiency. These improvements result in time and cost savings for government officials, reducing the resources required for managing land records. The availability of accurate and up-to-date land records also helps in preventing land disputes and fraudulent activities, further saving costs associated with legal disputes and investigations. Moreover, the online land record system provides an opportunity for revenue generation. By offering value-added services such as online land record searches, document downloads, and data analytics, the government can generate revenue through user fees or subscription models. This revenue can offset the ongoing maintenance and support costs of the system, making it economically sustainable in the long run. Furthermore, the economic benefits of the project extend beyond cost savings. The improved efficiency and accessibility of land records facilitate faster decision-making, faster land transactions, and increased investor confidence. This, in turn, can drive economic growth, attract investment, and boost revenue generation for the government.

In summary, the project is technically feasible due to the capabilities of the QGIS platform and plugin development tools. It also demonstrates economic feasibility by

reducing costs, increasing productivity, and enhancing revenue generation potential. Furthermore, the project is operationally feasible as it aligns with the objectives of the land record management system, improves operational efficiency, and provides user-friendly interfaces for seamless integration into existing processes. Thus, the project demonstrates all kinds of feasibility, viz. Operational, Technical and Economic feasibility.

D. PRODUCT FUNCTION

QGIS

QGIS, also known as the Quantum Geographic Information System, is a robust and widely utilized open-source software application designed for geospatial data analysis and visualization. As a sophisticated Geographic Information System (GIS), QGIS offers a comprehensive suite of tools and functionalities for processing, managing, and analyzing spatial data. It provides a user-friendly interface coupled with advanced geospatial capabilities, making it a valuable resource for professionals in fields such as geography, environmental science, urban planning, and natural resource management. QGIS supports various data formats, including vector and raster data, and enables seamless integration with other GIS platforms and databases. Its extensive plugin architecture further enhances its capabilities by allowing users to extend and customize the software to meet specific project requirements. With its broad range of features and its community-driven development, QGIS continues to empower users worldwide to explore, analyze, and visualize geospatial data with utmost precision and efficiency.

Plugin

Plugins are used to add or extend the functionalities to the websites. External plugins in the QGIS Plugins Repository need to be installed by users before they can be used. Plugin Manager tool is another way to browse and install these plugins. Users can create their own plugin also, QGIS plugins are built in python in python: the UI can be designed with QT designer. In QGIS, the plugin builder tool allows users to create a plugin template. This template can be customized by modifying the user interface (UI) and setting up the necessary parameters for the desired functionality. To streamline the testing process, it is recommended to have the plugin reloader installed. This tool eliminates the need to close and reopen QGIS after each change to the plugin's logic, allowing for efficient testing and debugging.

QT Designer

Qt Designer is a Qt tool used for designing and building GUIs with Qt Widgets. It offers a WYSIWYG interface for composing and customizing windows and dialogs. Users can test their designs with various styles and resolutions. Properties set in Qt Designer can be dynamically changed in the code.

1. UNIQUE ID generation

The Unique Land Parcel Identification Number (ULPIN) scheme was introduced in 2021 and is set to be implemented nationwide by March 2022, initially covering 10 states.This scheme is often referred to as the "Aadhaar for land" as it aims to provide a unique identification number to each surveyed parcel of land. The UNIQUE ID system is crucial in preventing land fraud, particularly in rural areas of India where land records are frequently outdated and subject to disputes. By assigning a unique identifier to each land parcel, UNIQUE ID facilitates accurate identification and verification, ensuring the integrity of land ownership records.

Unique id identification is based on the precise longitude and latitude coordinates of each land parcel, relying on detailed surveys and geo-referenced cadastral maps for accuracy and reliability.

BENEFITS:

1. Different states are using different methods for assigning unique IDs to the land parcel
2. Land parcel numbers are repeated within each village under the UNIQUE ID scheme, ensuring clarity and avoiding conflicts in identification.
3. Difficult to do data analytics and establish farmers- Land relationship
4. Need for sharing of land records data across departments, financial institutions and all stakeholders.
5. Standardization at data and application level would bring in effective integration and interoperability across departments.
6. The benefits of the UNIQUE ID scheme include ensuring uniqueness in transactions and maintaining up-to-date land records.

7. A link of all property transactions will get established.
8. Delivery of citizen services of land records through the single window.

Latitude Longitude

Every location on Earth is assigned a global address, which is represented by two numbers known as coordinates. These coordinates consist of a latitude number and a longitude number, commonly referred to as "Lat/Long." The latitude and longitude lines form a grid map system. Unlike straight lines on a flat surface, these lines encircle the Earth, forming either horizontal circles or vertical half circles. This grid system allows for precise location referencing on a global scale.

The UTM coordinate system partitions the Earth into 60 zones, each spanning 6 degrees of longitude. These zones serve as reference points for UTM grid coordinates within their respective boundaries. UTM zones encompass latitudes ranging from 80° S to 84° N.

EPSG/UTM zone

EPSG stands for European Petroleum Survey Group. They maintain a database of coordinate system information. The EPSG number is a 4 to 6 digit number which represents the parameters of a CRS (coordinate reference system). UTM stands for Universal Transverse Mercator. It is a plane coordinate grid system which is named for the map projection.

UTM zone for India is WGS 84 / UTM zone 44N. Most preferred EPSG for projection is 4326.

In regard to the generation of UNIQUE ID, we have developed two distinct plugins, each possessing unique functionalities. The initial plugin facilitates the conversion of latitude and longitude coordinates associated with a particular land parcel into its corresponding UNIQUE ID. On the other hand, the second plugin enables the reverse conversion, allowing for the retrieval of latitude and longitude data when provided with the UNIQUE ID. These complementary plugins contribute to a comprehensive UNIQUE ID generation system, offering enhanced flexibility and accuracy in managing geospatial information pertaining to land records.

Attribute table

An attribute table is a database or tabular file that stores information about a collection of geographic features. It is structured so that each row represents a feature and each column represents an attribute of that feature. In raster datasets, each row of the attribute table corresponds to a zone of cells with identical values. In a GIS system, attribute tables are linked or associated with spatial data layers. The attribute values within the table can be utilized for querying, symbolizing, and analysing features or raster cells.

2. GP

The proposed system is to design and create a plugin which can perform multiple required tasks on just a single click. The user shall only upload the shape file corresponding to the area and all the desired files must be generated accordingly.

The plugin must be able to perform following task:

a) Calculate the centroid points for each plot

b) Generate latitude longitude points

c) Generate a 14-digit unique id for every plot

d) Create a csv file corresponding for all the attributes

e) Create a point kml file for pointing the centroids of the plots

f) Create a polygon kml file for plotting the boundaries of the plot**s**

Shapefile (vector layer, raster layer)

A shapefile is a data format used to store both geometric and attribute information for spatial features. The geometric representation of a feature is stored as a shape, which consists of a set of vector coordinates. In the vector data model, geographic features are represented as points, lines, and polygons. Points are represented by a single pair of coordinates, while lines and polygons are represented by ordered lists of vertices. Shapefiles can accommodate various types of features, including points, lines, and polygons. Polygons, specifically, are represented as closed loop, double-digitized shapes.

Vector data and raster data are two different types of GIS data. The most common of GIS data is vector data. Most data loaded in any GIS software program is in the form of vector data. Geographic data is symbolically represented by vector data as points, lines, or polygons. Point, line, and polygon feature locations, geometry, and attribution are all stored in shapefiles. Adding shapefile data to a.zip file, uploading it, and publishing it as a hosted feature layer are the basic methods for making shapefile data accessible for others to see using a web browser. The .zip file must contain files with extension .shp, .shx, .dbf, and. prj. Users can create new data along with editing the existing one.

The primary file (.shp) of a shapefile consists of a fixed-length file header and variable-length records. Each variable-length record consists of a fixed-length record header and variable-length content. The file header provides essential information about the shapefile, while the record header contains specific details about each individual record within the file. The variable-length record contents hold the actual data and attributes associated with the spatial features stored in the shapefile.

Csv file

A CSV stands for comma-separated values. This file is a text file that allows data to be stored in a table structured format. Data tables appear in Comma Delimited, CSV text file format. although this file format enables the data table to be easily accessed into a variety of applications.

Kml file

KML is a type of file format used to display geographic data in Earth browsers such as Google Earth. user can create KML files to point out locations, add image overlays, and expose available data in new ways. KML is based on the XML (extensive mark-up language) standards and tag structure which may or may not contain nested elements and attributes.

3. MERGE Plugin

Within the QGIS environment, a plugin has been developed to provide users with enhanced functionality through a selection of three distinct options. These options are presented in the form of radio buttons, allowing users to choose between the

"Split Plugin", "Multipart Plugin" and "Merge Plugin." Each option corresponds to a specific plugin that is launched dynamically based on the user's selection. The "Split Plugin" enables the efficient splitting of a plot into two parts based on two points on the boundary of the plot, while the "Multipart Plugin" facilitates the splitting of a plot into ‘n’ number of parts of approximately equal area. Lastly, the "Merge Plugin" empowers users to merge two adjacent plots into a single plot. By incorporating these three plugins, users are afforded greater flexibility and efficiency in their geospatial data processing and analysis tasks within the QGIS environment.

3.1 SPLIT Plugin

The provided QGIS plugin serves the purpose of splitting a plot into two parts, offering users a user-friendly and interactive solution. After selecting an input shapefile, the plugin displays all plot IDs within a dropdown menu. Users can then choose a specific plot from the dropdown. Upon selection, the plugin highlights and zooms in on the chosen plot on the screen, ensuring clarity and precision during the splitting process.

To facilitate the splitting, the plugin identifies and displays the vertices derived from the geometric points along the plot boundary. These vertices are listed in two separate dropdown menus, with each dropdown corresponding to a vertex point. This feature allows users to select the desired starting and ending points for the splitting operation.

Users have the flexibility to either confirm the splitting directly from the displayed vertices by clicking the "Confirm Splitting" button or choose an additional point at a specified distance from the existing vertices. The desired distance is entered using an input line provided in the plugin, allowing users to define the separation precisely. The input line offers two editable fields for each vertex point. The first field determines the movement either to the left or right direction, where positive values represent movement to the right and negative values indicate movement to the left. Similarly, the second field determines the movement either in the upward or downward direction, with positive values indicating upward movement and negative values representing downward movement. These options provide users with granular control over the splitting process.

After the user has specified the splitting points and distances, clicking the "Confirm Splitting" button generates a preview of the splitting operation as a separate layer. This preview allows users to visualize the outcome before finalizing the operation. If the user is satisfied with the preview, they can proceed by clicking the "OK" button. At this stage, the plugin adds the newly created plots to the shapefile, while removing the original plot.

By incorporating these functionalities, the plugin simplifies the process of splitting plots, providing users with a comprehensive solution within the QGIS environment. It enhances efficiency and accuracy in land management, urban planning, and other geospatial applications.

Eg: Eg: if the plot no “64” is divided into 2 parts , then plot no. 64 will be deleted and two new plots will be added to the shapefile namely “117/1” and “117/2”. This will facilitate any new user to understand that these plots were created after splitting a plot having id “64”.

3.2 MULTIPART Plugin

The developed QGIS plugin provides users with a convenient interface for interacting with shapefiles and performing specific operations. Upon selecting a shapefile as input, the plugin dynamically generates a dropdown menu displaying the ID numbers associated with individual plots within the shapefile. This feature allows users to conveniently select a specific plot of interest from the available options. Upon selecting a plot number from the dropdown, the plugin performs highlighting and zooming functions to visually focus on the selected plot within the QGIS screen, thereby enhancing user convenience and navigation. By automatically adjusting the view and scale, users can easily examine and analyse the chosen plot with enhanced clarity and precision. To further enhance the flexibility of the plugin, a line input is provided, prompting the user to enter the desired number of parts into which the selected plot should be divided. Users can specify the exact number of parts required. Once the user specifies the number of parts and confirms the selection by clicking the "OK" button, the plugin automatically executes the splitting operation, dividing the plot into the designated number of parts. This automated process eliminates the need for manual and time-consuming partitioning of plots,

streamlining the workflow and ensuring consistency in data management. This feature streamlines the process of dividing plots within the QGIS environment, enabling users to efficiently manage and analyse spatial data. The new plots are added in the shapefile and the old plot is deleted.

Eg: if the plot no “117” is divided into 4 parts, then plot 117 will be deleted and four new plots will be added to the shapefile namely “117/1”, “117/2”, “117/3” and “117/4”. This will facilitate any new user to understand that these plots were created after splitting a plot having id “117”.

3.3 MERGE POLYGON Plugin

The developed QGIS plugin serves the purpose of merging two plots into a single plot, providing users with a seamless and efficient solution. When a shapefile is selected as input, the plugin initiates the retrieval of plot IDs associated with each individual plot contained within the shapefile. These plot IDs are then presented in two dropdown menus, enabling users to select the first plot from the first dropdown and the second plot from the second dropdown.

Upon clicking the "Confirm Merging" button, the plugin proceeds with the merging process. Before merging, the plugin performs a crucial validation step by checking if the selected plots are adjacent to each other. If the plots are not adjacent, a message is printed on the Python console, indicating that the plots are not adjacent and merging is not possible. In this case, the user is required to reselect the plots by choosing appropriate adjacent plots.

If the selected plots are found to be adjacent, the plugin generates a preview on the screen, visually representing the merged plot as a layer. This preview provides users with a visual understanding of how the merged plot will appear before confirming the operation. If the user is satisfied with the preview and wishes to proceed, they can click the "OK" button to initiate the merging process.

Once the merging is confirmed, the plugin executes the merging operation, combining the geometries and attributes of the selected plots into a single plot. The shape file is updated with the new plot, where the merged plot is represented as a distinct feature.

By providing this merging functionality, the plugin streamlines the process of combining plots, reducing manual effort and increasing efficiency in land

management, urban planning, and other geospatial applications within the QGIS environment.

Eg: if the plots to be merged have plot ids “45” and “62”. So, the merged plot will have the id “45_62”. This will facilitate any new user to understand that this plot was formed after merging two plots having id “45” and “62”.

PostgreSQL

PostgreSQL, a powerful open-source relational database management system, plays a crucial role in the land record management project. Its integration enables efficient storage, retrieval, and management of data related to land records. With its robust features, PostgreSQL ensures data integrity, reliability, and scalability, which are essential for handling large volumes of land records. The project utilizes PostgreSQL to create and manage the necessary database tables, storing information such as plot details, ownership records, and spatial data. The flexibility of PostgreSQL allows for easy customization and extensibility, accommodating the specific requirements of the land record management system. Furthermore, PostgreSQL's support for spatial data types and its integration with GIS tools like QGIS enhance the capabilities of the system, enabling spatial queries, analysis, and visualization of land records. With PostgreSQL as the underlying database technology, the project benefits from its stability, performance, and community support, providing a solid foundation for the efficient management of land records.

PostGIS

PostGIS, an extension of PostgreSQL, plays a vital role in the land record management project by providing advanced geospatial capabilities. With PostGIS, the project leverages powerful spatial data types and functions to store, query, analyze, and visualize geospatial data associated with land records. By enabling spatial indexing and optimized spatial queries, PostGIS ensures efficient retrieval of relevant spatial information, such as the location of plots, their boundaries, and spatial relationships. The integration of PostGIS with the project's database allows for seamless integration with GIS tools like QGIS, enabling users to perform spatial analysis, generate thematic maps, and make informed decisions based on the spatial attributes of land records. PostGIS also supports spatial data validation and integrity

checks, ensuring the accuracy and consistency of geospatial data in the system. Overall, the use of PostGIS enhances the functionality and performance of the land record management system, empowering users to effectively manage and utilize geospatial information for land-related operations and decision-making processes.

4. LAND RECORD SYSTEM

The website for the land record system of Arunachal Pradesh encompasses a secure login page as the initial point of entry. Users are required to input their username and password, which undergo verification against an admin table stored in the pgAdmin database. To ensure human access, a captcha feature is implemented. In case of invalid credentials, an alert message notifies the user about the invalid login attempt. Similarly, if the captcha response is incorrect, the user is informed of an invalid captcha. Upon successful authentication, the user is redirected to the menu page.

The menu page offers three options represented as buttons, enabling users to select their preferred method of accessing plot information. These options include searching by owner name, land type, or plot number. Upon selecting an option, the user is redirected to the corresponding page for further interaction.

If the user selects the "owner name" option, they are directed to the owner name page. This page establishes a connection with the pgAdmin database, extracting all owner names from the table and populating them in a dropdown menu. Once the user selects the desired owner name and clicks the "confirm" button, the page displays the selected owner name for verification. If the displayed owner name is correct, the user can proceed by clicking the "submit" button. Subsequently, the user is redirected to the map page, where all relevant plot information associated with the selected owner name is retrieved from the database. The map page displays all plots on an interactive map, with the plot owned by the selected individual being highlighted for easy identification.

In a similar fashion, if the user opts for the "land type" option, they are redirected to the land page. This page presents a dropdown menu containing all available land types. Upon selecting a land type, a connection with the pgAdmin database is established, allowing the highlighting of all plots associated with the selected land type on the map.

Likewise, if the user chooses the "plot number" option, they are redirected to the plot number page. Here, a dropdown menu displays all available plot numbers. Upon selecting a specific plot number, a database connection is made, enabling the highlighting of the corresponding plot on the map.

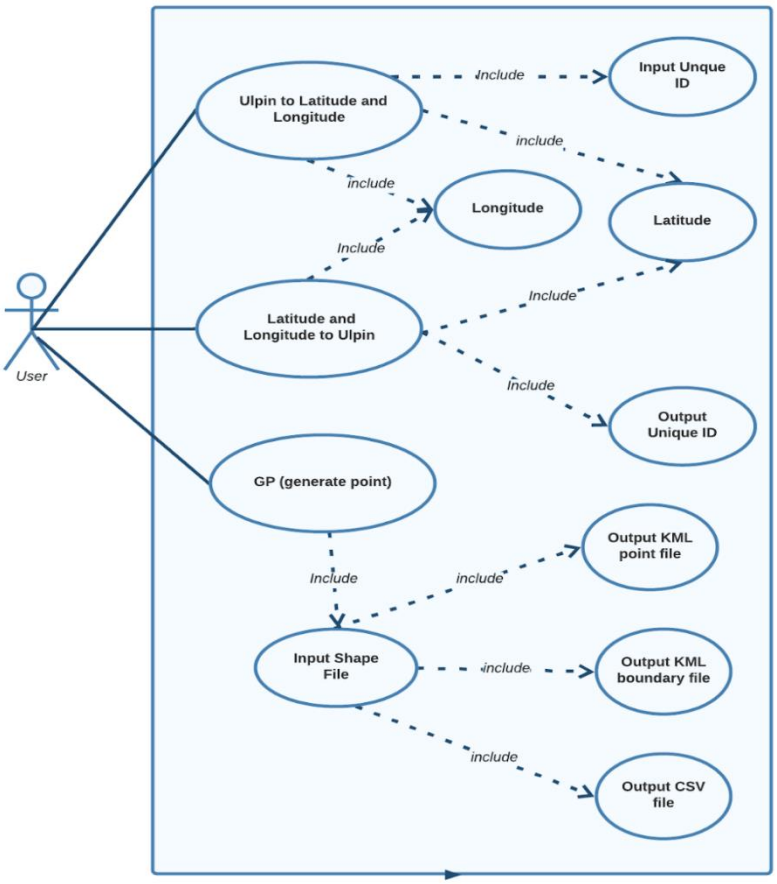
These functionalities enhance the user experience by providing a user-friendly interface and efficient access to plot information based on different search criteria. The integration of the pgAdmin database ensures accurate and up-to-date data retrieval, ultimately facilitating land record management and promoting transparency in the land record system of Arunachal Pradesh.

Future Scope

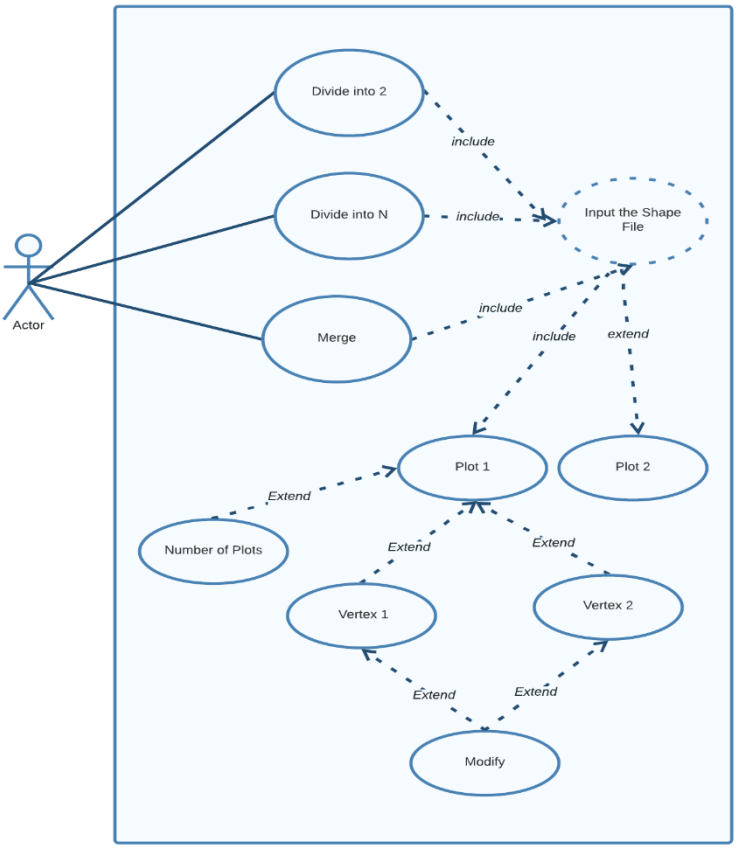
In addition to the current functionality, there are several future prospects for the plugins and website developed for land record management in India. Firstly, there is scope for enhancing the existing features by incorporating advanced search options, data analytics capabilities, and seamless integration with other land-related systems. This would enable users to have a more comprehensive and efficient land record management experience. Moreover, considering the growing popularity of mobile devices, optimizing the plugins and website for mobile platforms would enable users to access and manage land records on-the-go, providing greater convenience and accessibility. Another potential avenue for development is the integration of these tools with existing government systems, such as revenue departments and land administration authorities. This integration would facilitate real-time data exchange and updates, leading to improved coordination and efficiency in land record management. Additionally, incorporating data visualization tools and reporting features would enable users to analyze and present land-related data in a more visual and meaningful manner, aiding in decision-making processes. Overall, these future enhancements would contribute to the ongoing development and improvement of land record management systems in India.

E. USE CASE

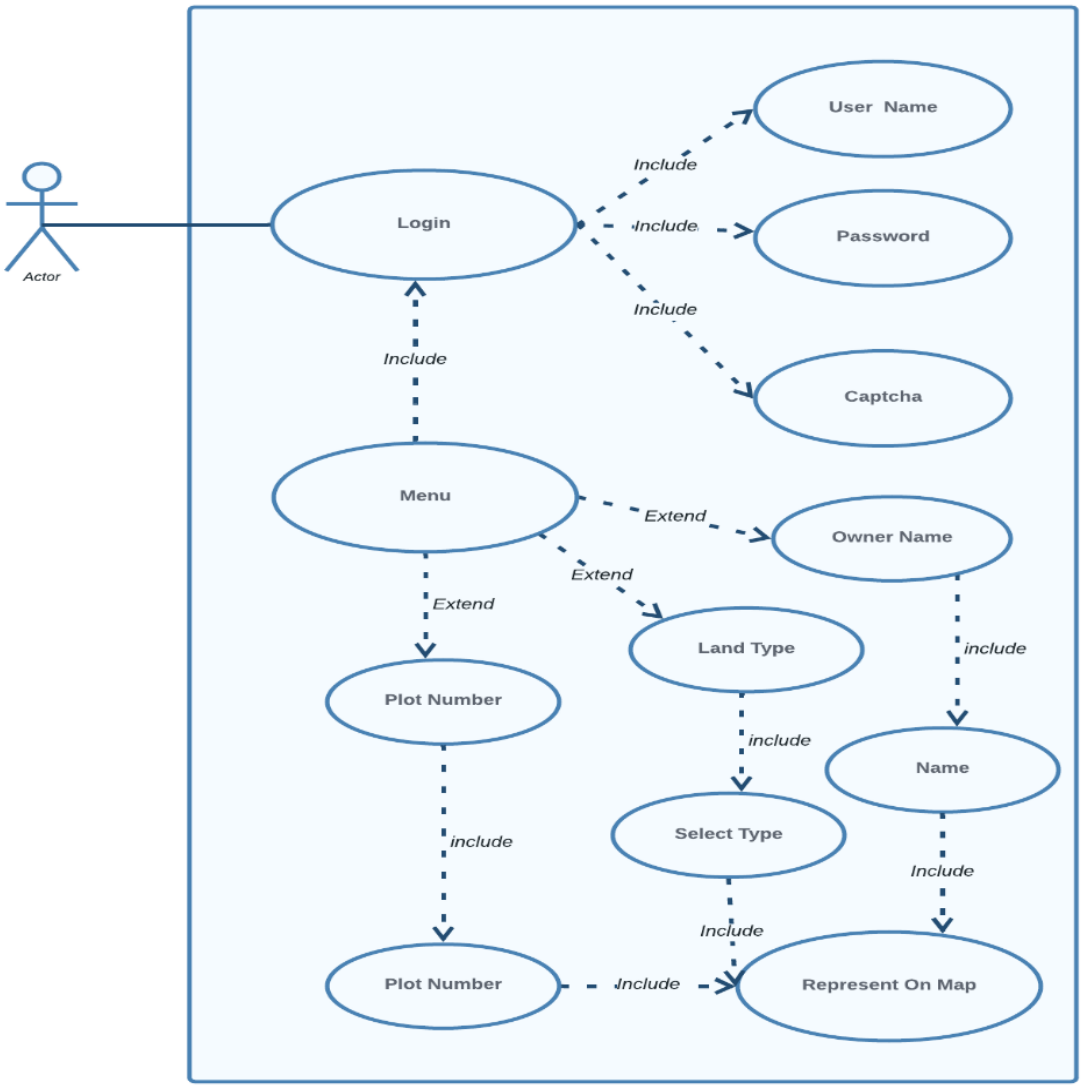
1. GP:



2. MERGE PLUGIN:



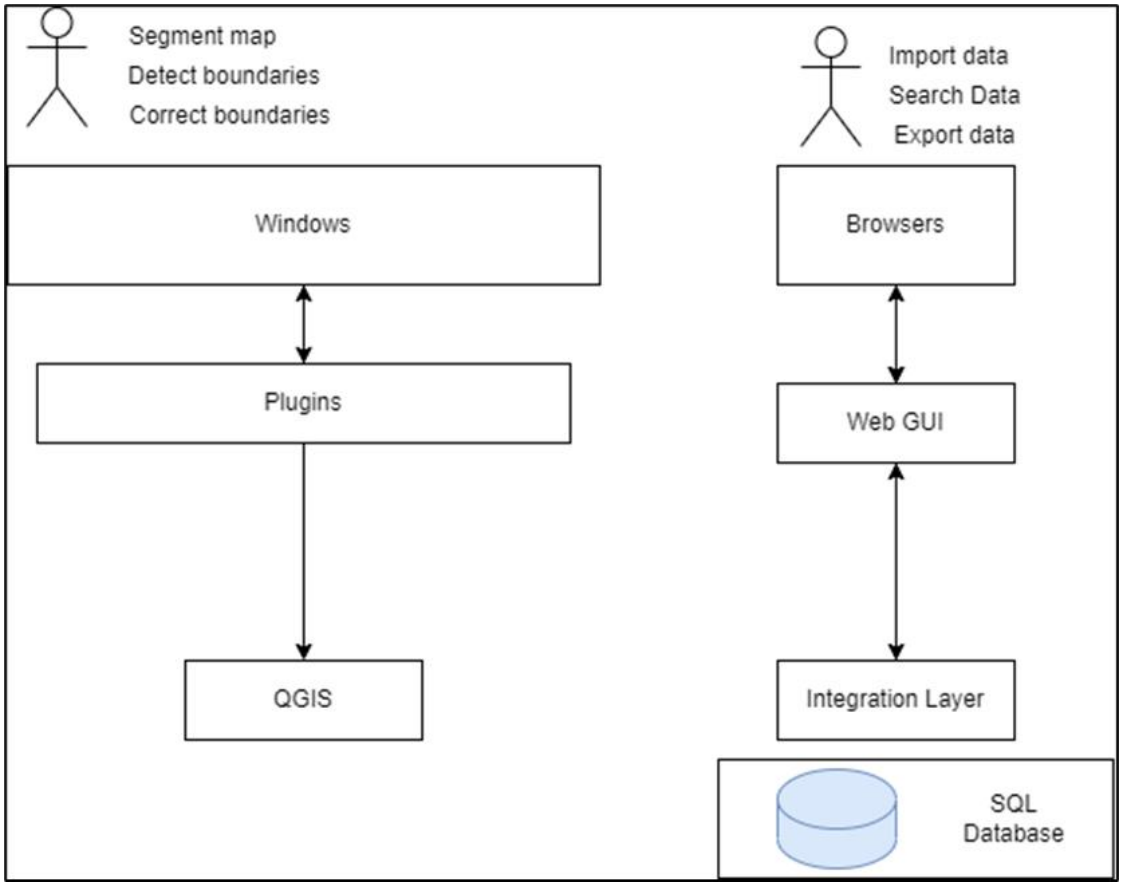
3. LAND RECORD SYSTEM:



CHAPTER 3- SYSTEM DESIGN (SDS)

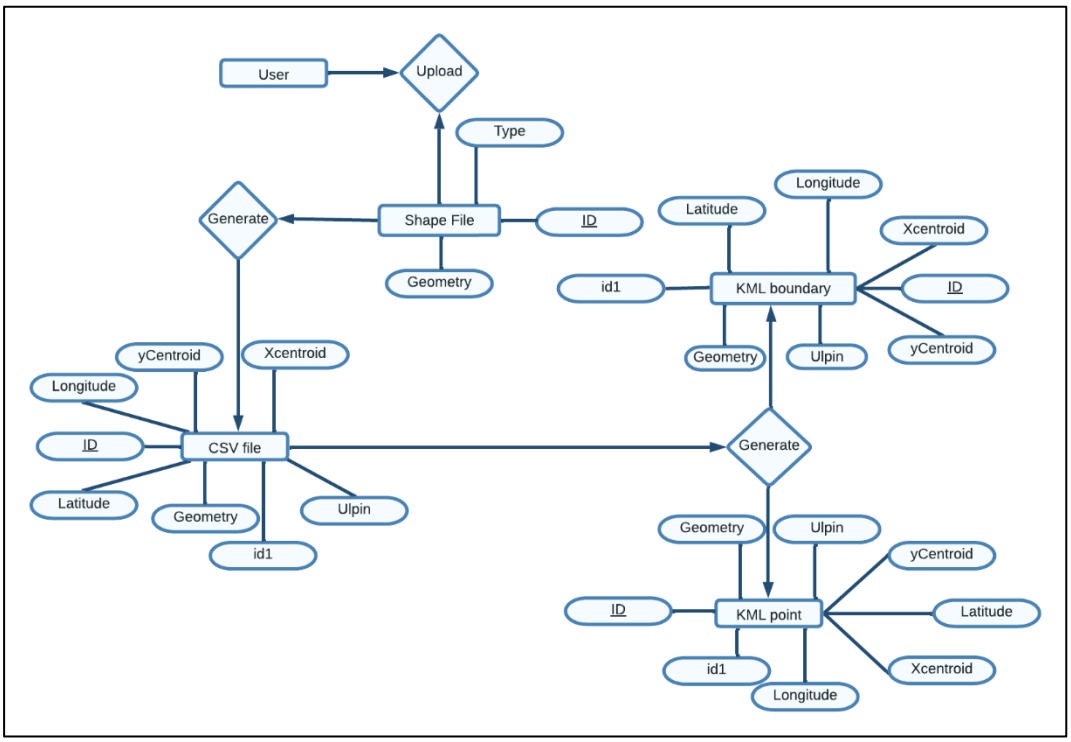
A. HIGH-LEVEL DESIGN

High-level design includes a high-level architecture diagram depicting the structure of the system, such as the hardware, database architecture, application architecture (layers), application flow (navigation), security architecture and technology architecture. A high-level design hypothesis is presented below, based on user needs and the business model. It provides a visual representation of the proposed system structure and layout, serving as a foundation for further development.

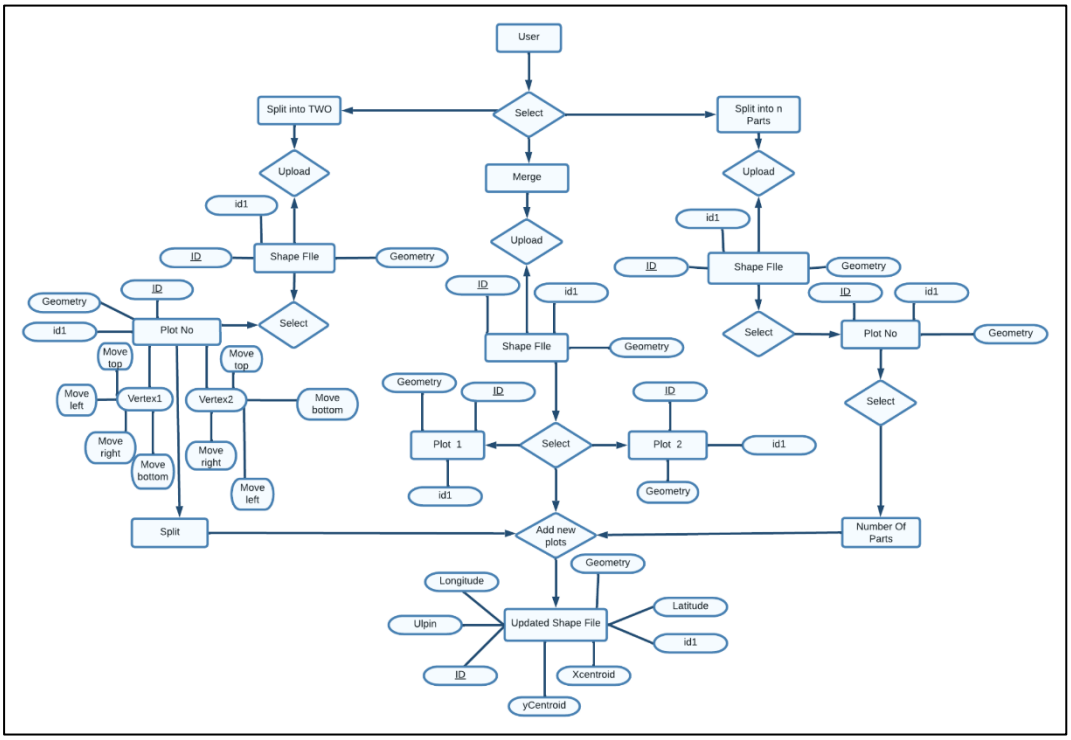


B. ER DIAGRAM

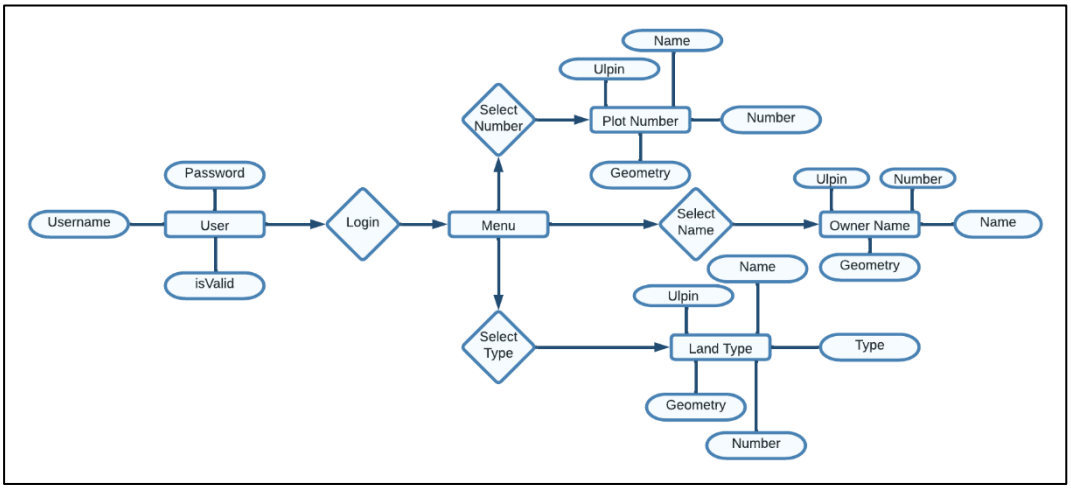
GP:



MERGE PLUGIN:

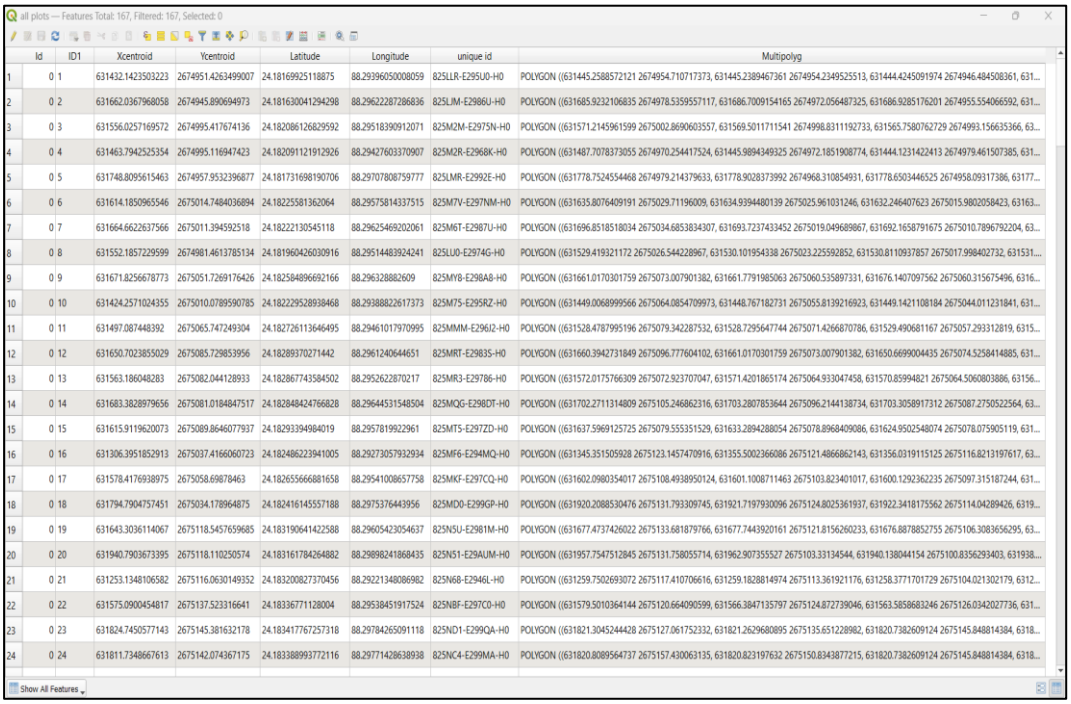


LAND RECORD SYSTEM:



C. DATABASE DESIGN

Attribute Table:

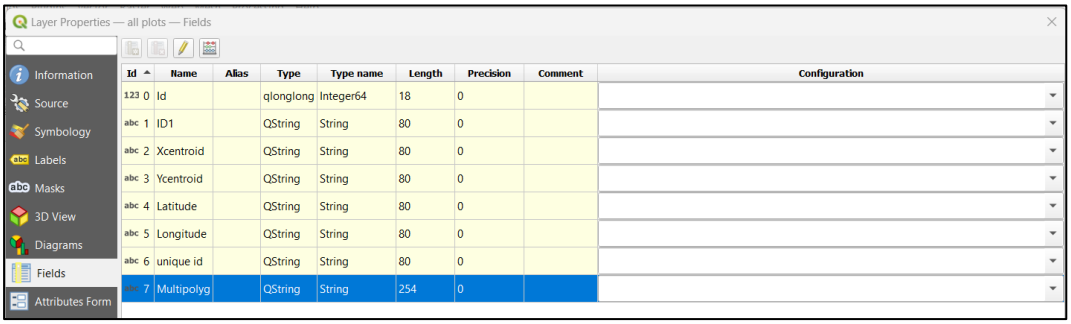


all plots — Features Total: 167, Filtered: 167, Selected: 0

	Id	ID1	Xcentroid	Ycentroid	Latitude	Longitude	unique id	Multipolyg
1	0	1	631432.1423503223	2674951.4263499007	24.18169925118875	88.29396050008059	825LLR-E295U0-H0	POLYGON ((631445.2588572121 2674954.710717373, 631445.2389467361 2674954.2349525513, 631444.4245091974 2674946.484508361, 631...
2	0	2	631662.0367968058	2674945.890694973	24.181630041294298	88.29622287286836	825LJM-E2986U-H0	POLYGON ((631685.9232106835 2674978.5359557117, 631686.7009154165 2674972.056487325, 631686.9285176201 2674955.554066592, 631...
3	0	3	631556.0257169572	2674995.417674136	24.182086126829592	88.29518390912071	825M2M-E2975N-H0	POLYGON ((631571.2145961599 2675002.8690603557, 631569.5011711541 2674998.8311192733, 631565.7580762729 2674993.156635366, 63...
4	0	4	631463.7942525354	2674995.116947423	24.182091121912926	88.29427603370907	825M2R-E2968K-H0	POLYGON ((631487.7078373055 2674970.254417524, 631445.9894349325 2674972.1851908774, 631444.1231422413 2674979.461507385, 631...
5	0	5	631748.8095615463	2674957.9532396877	24.181731698190706	88.29707808759777	825LMR-E2992E-H0	POLYGON ((631778.7524554468 2674979.214379633, 631778.9028373992 2674968.310854931, 631778.6503446525 2674958.09317386, 63177...
6	0	6	631614.1850965546	2675014.7484036894	24.18225581362064	88.29575814337515	825M7V-E297NM-H0	POLYGON ((631635.8076409191 2675029.71196009, 631634.9394480139 2675025.961031246, 631632.246407623 2675015.9802058423, 63163...
7	0	7	631664.6622637566	2675011.394592518	24.18222130545118	88.29625469202061	825M6T-E2987U-H0	POLYGON ((631696.8518518034 2675034.6853834307, 631693.7237433452 2675019.049689867, 631692.1658791675 2675010.7896792204, 63...
8	0	8	631552.1857229599	2674981.4613785134	24.181960426030916	88.29514483924241	825LU0-E2974G-H0	POLYGON ((631529.419321172 2675026.544228967, 631530.101954338 2675023.225592852, 631530.8110937857 2675017.998402732, 631531....
9	0	9	631671.8256678773	2675051.7269176426	24.18258489669216 6	88.296328882609	825MY8-E298A8-H0	POLYGON ((631661.0170301759 2675073.007901382, 631661.7791985063 2675060.535897331, 631676.1407097562 2675060.315675496, 6316...
10	0	10	631424.2571024355	2675010.0789590785	24.182229528938468	88.29388822617373	825M75-E295RZ-H0	POLYGON ((631449.0068999566 2675064.0854709973, 631448.767182731 2675055.8139216923, 631449.1421108184 2675044.011231841, 631...
11	0	11	631497.087448392	2675065.747249304	24.182726113646495	88.29461017970995	825MMM-E296J2-H0	POLYGON ((631528.4787995196 2675079.342287532, 631528.7295647744 2675071.4266870786, 631529.4906881167 2675057.293312819, 6315...
12	0	12	631650.7023855029	2675085.729853956	24.18289370271442	88.2961240644651	825MRT-E2983S-H0	POLYGON ((631660.3942731849 2675096.777604102, 631661.0170301759 2675073.007901382, 631650.6699004435 2675074.5258414885, 631...
13	0	13	631563.186048283	2675082.044128933	24.182867743584502	88.2952622870217	825MR3-E2978G-H0	POLYGON ((631572.0175766309 2675072.923707047, 631571.4201865174 2675064.933047458, 631570.85994821 2675064.5060803886, 63156...
14	0	14	631683.3828979656	2675081.0184847517	24.182848424766828	88.29644531548504	825MQG-E298DT-H0	POLYGON ((631702.2711314809 2675105.246862316, 631703.2807853644 2675096.2144138734, 631703.3058917312 2675087.2750522564, 63...
15	0	15	631615.9119620073	2675089.8646077937	24.18293394984019	88.2957819922961	825MT5-E297ZD-H0	POLYGON ((631637.5969125725 2675079.555351529, 631633.2894288054 2675078.8968409086, 631624.9502548074 2675078.075905119, 631...
16	0	16	631306.3951852913	2675037.4166060723	24.182486223941005	88.29273057932934	825MF6-E294MQ-H0	POLYGON ((631345.351505928 2675123.1457470916, 631355.5002366086 2675121.4866862143, 631356.0319115125 2675116.8213197617, 63...
17	0	17	631578.4176938975	2675058.69878463	24.182655666881658	88.29541008657758	825MKF-E297CQ-H0	POLYGON ((631602.0980354017 2675108.4938950124, 631601.1008711463 2675103.823401017, 631600.1292362235 2675097.315187244, 631...
18	0	18	631794.7904757451	2675034.178964875	24.182416145557188	88.2975376443956	825MD0-E299GP-H0	POLYGON ((631920.2088530476 2675131.793309745, 631921.7197930096 2675124.8025361937, 631922.3418175562 2675114.04289426, 6319...
19	0	19	631643.3036114067	2675118.5457659685	24.183190641422588	88.29605423054637	825N5U-E2981M-H0	POLYGON ((631677.4737426022 2675133.681879766, 631677.7443920161 2675121.8156260233, 631676.8878852755 2675106.3083656295, 63...
20	0	20	631940.7903673395	2675118.110250574	24.183161784264882	88.29898241868435	825N51-E29AUM-H0	POLYGON ((631957.7547512845 2675131.758055714, 631962.907355527 2675103.33134544, 631940.138044154 2675100.8356293403, 631938...
21	0	21	631253.1348106582	2675116.0630149352	24.183200827370456	88.29221348086982	825N68-E2946L-H0	POLYGON ((631259.7502693072 2675117.410706616, 631259.1828814974 2675113.361921176, 631258.3771701729 2675104.021302179, 6312...
22	0	22	631575.0900454817	2675137.523316641	24.18336771128004	88.29538451917524	825NBF-E297C0-H0	POLYGON ((631579.5010364144 2675120.664090599, 631566.3847135797 2675124.872739046, 631563.5858683246 2675126.0342027736, 631...
23	0	23	631824.7450577143	2675145.381632178	24.183417767257318	88.29784265091118	825ND1-E299QA-H0	POLYGON ((631821.3045244428 2675127.061752332, 631821.2629680895 2675135.651228982, 631820.7382609124 2675145.848814384, 6318...
24	0	24	631811.7348667613	2675142.074367175	24.183388993772116	88.2977142863893 8	825NC4-E299MA-H0	POLYGON ((631820.8089564737 2675157.430063135, 631820.823197632 2675150.8343877215, 631820.7382609124 2675145.848814384, 6318...

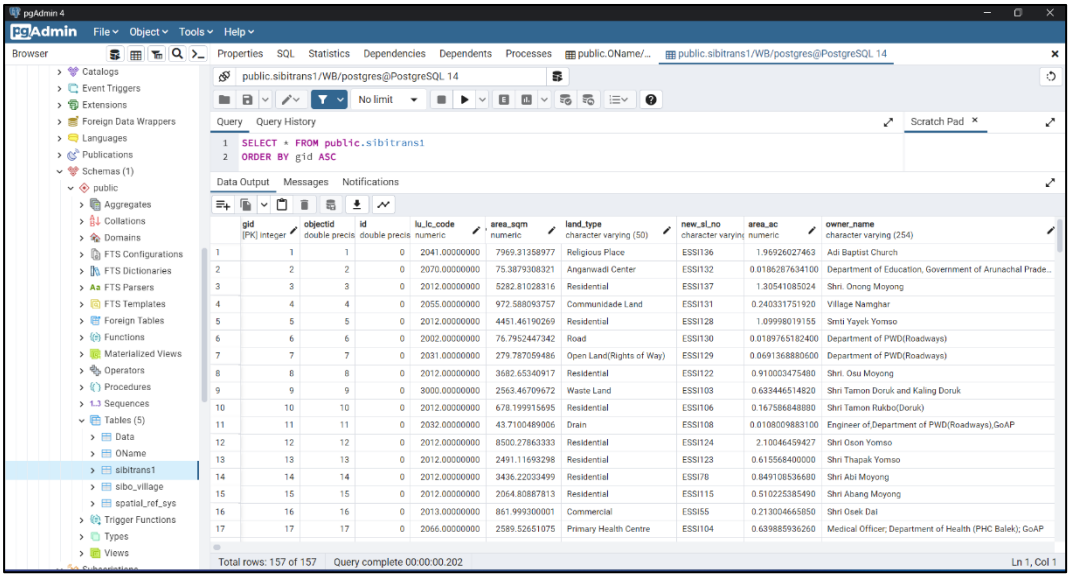
Show All Features

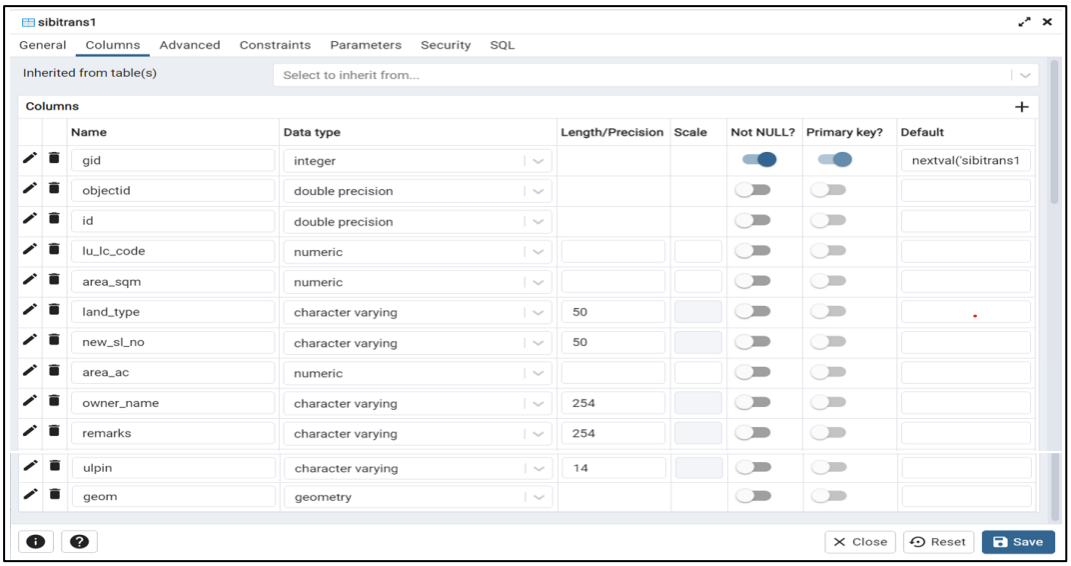
PLOT INFORMATION - SHAPEFILE:



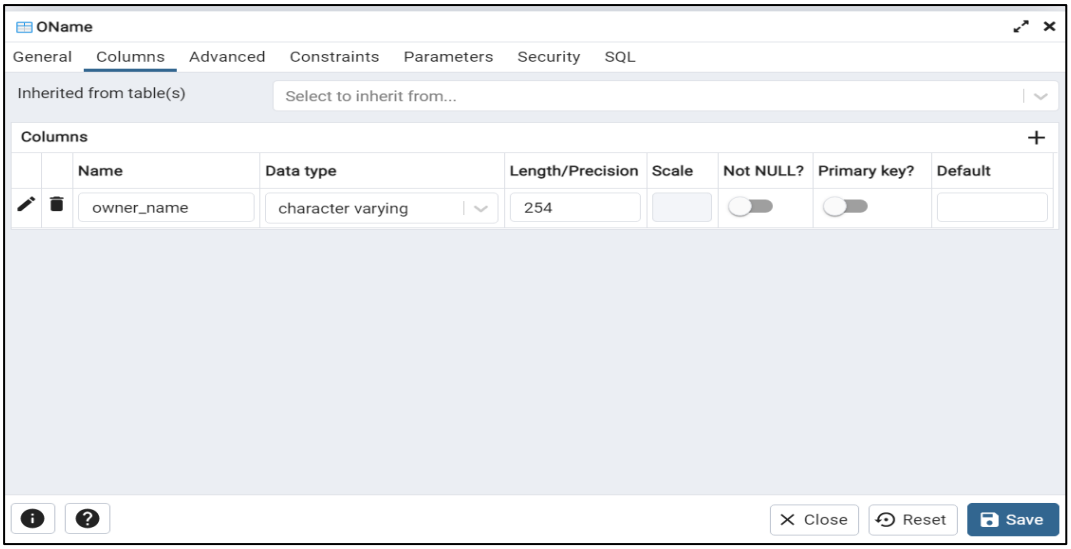
Id	Name	Alias	Type	Type name	Length	Precision	Comment	Configuration
123 0	Id		qlonglong	Integer64	18	0		
abc 1	ID1		QString	String	80	0		
abc 2	Xcentroid		QString	String	80	0		
abc 3	Ycentroid		QString	String	80	0		
abc 4	Latitude		QString	String	80	0		
abc 5	Longitude		QString	String	80	0		
abc 6	unique id		QString	String	80	0		
abc 7	Multipolyg		QString	String	254	0		

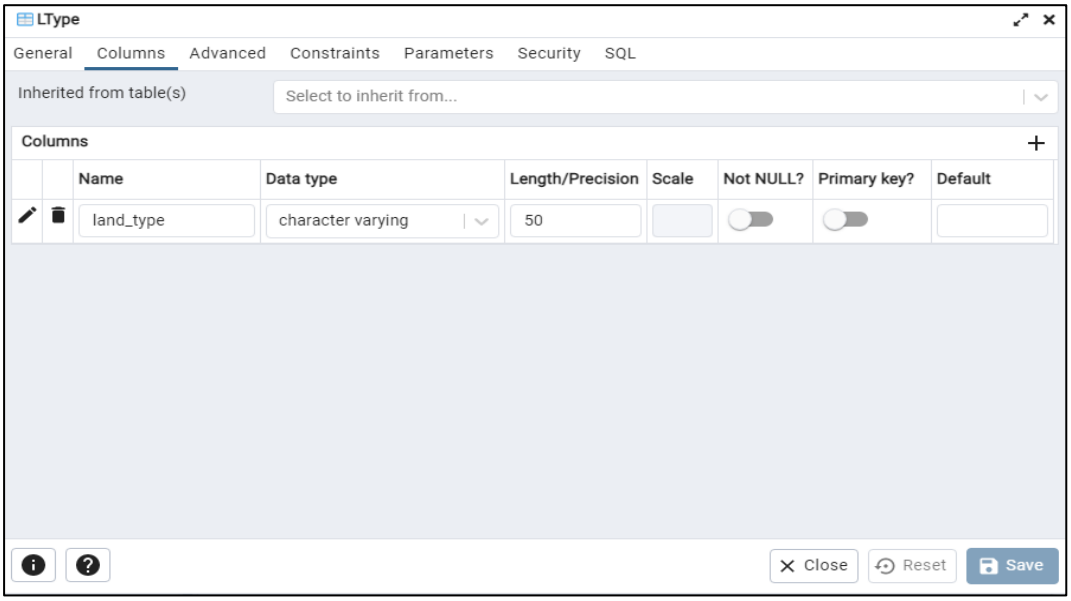
SIBITRANS1:

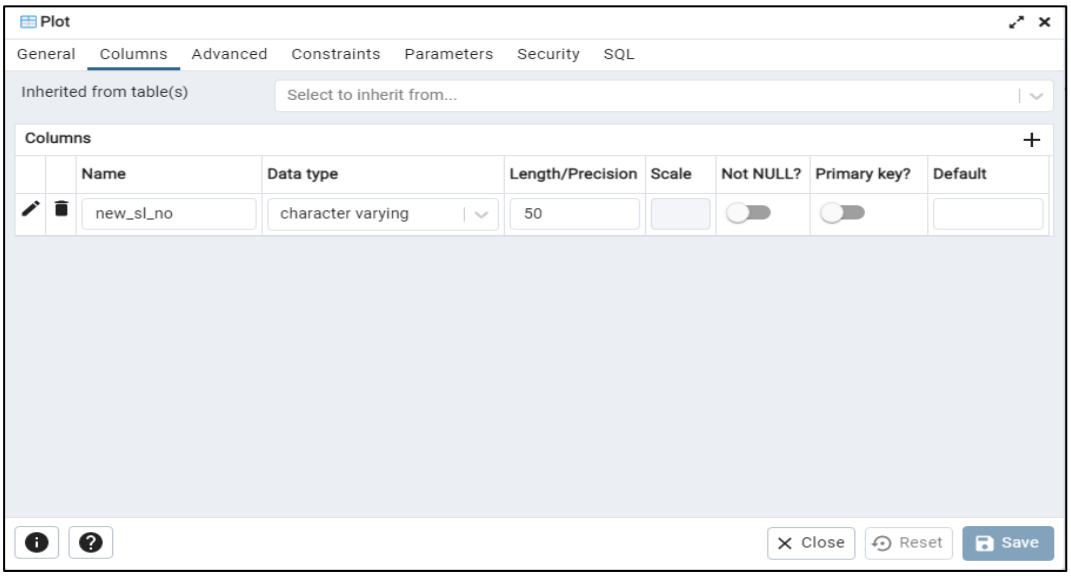




Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
gid	integer			✓	✓	nextval('sibitrans1
objectid	double precision					
id	double precision					
lu_lc_code	numeric					
area_sqm	numeric					
land_type	character varying	50				
new_sl_no	character varying	50				
area_ac	numeric					
owner_name	character varying	254				
remarks	character varying	254				
ulpin	character varying	14				
geom	geometry					

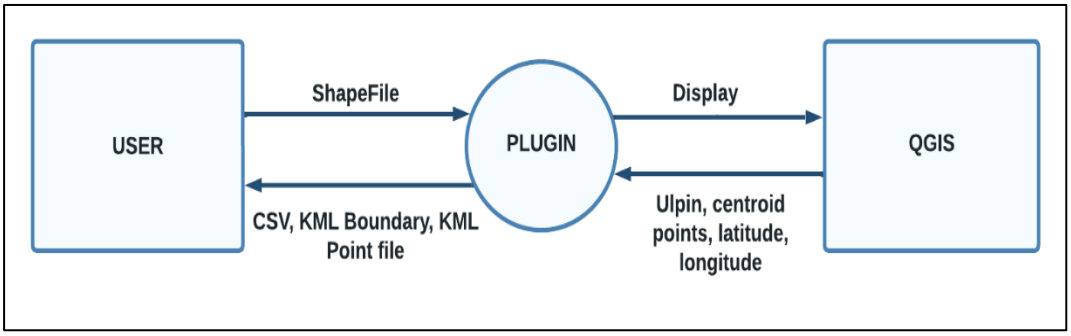

OName
General
Columns
Advanced
Constraints
Parameters
Security
SQL
Inherited from table(s)
Select to inherit from...
Columns
Name
Data type
Length/Precision
Scale
Not NULL?
Primary key?
Default
owner_name
character varying
254
Close
Reset
Save


LType
General
Columns
Advanced
Constraints
Parameters
Security
SQL
Inherited from table(s)
Select to inherit from...
Columns
Name
Data type
Length/Precision
Scale
Not NULL?
Primary key?
Default
land_type
character varying
50
Close
Reset
Save

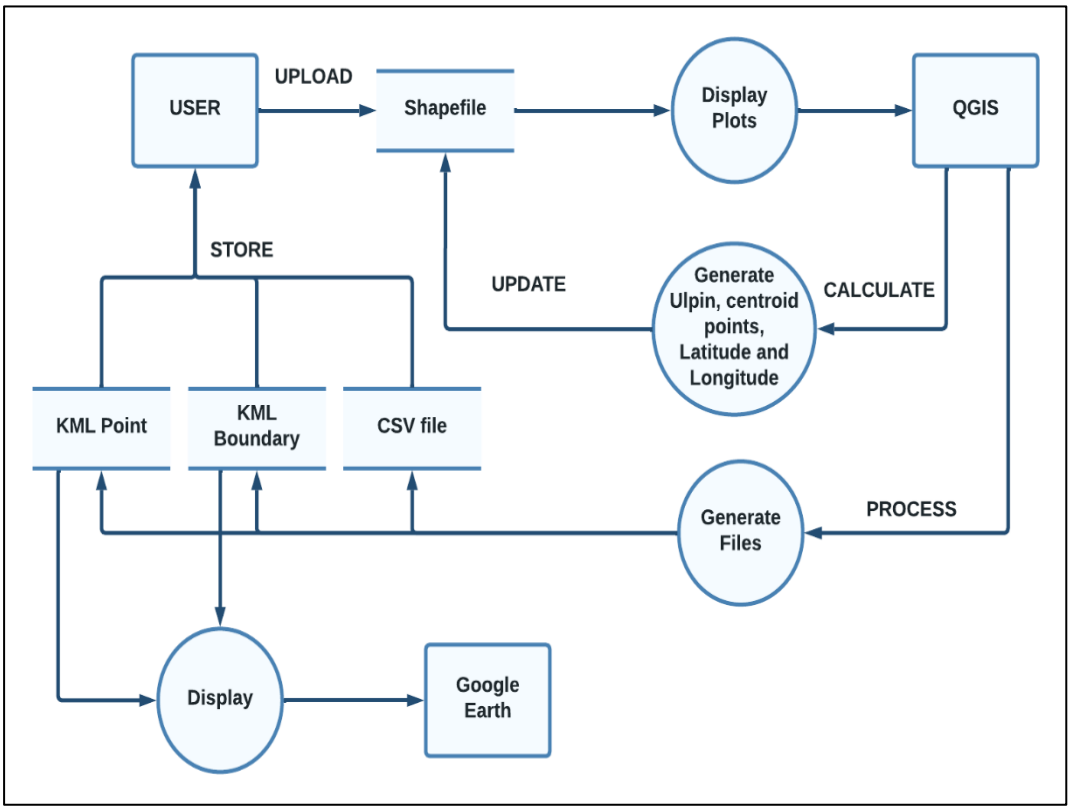

Plot
General
Columns
Advanced
Constraints
Parameters
Security
SQL
Inherited from table(s)
Select to inherit from...
Columns
Name
Data type
Length/Precision
Scale
Not NULL?
Primary key?
Default
new_sl_no
character varying
50
Close
Reset
Save

D. DATA FLOW DIAGRAMS

GP:

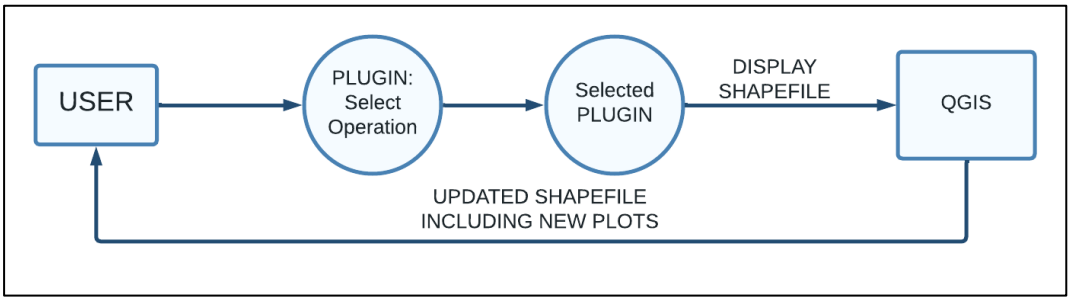


Level-0

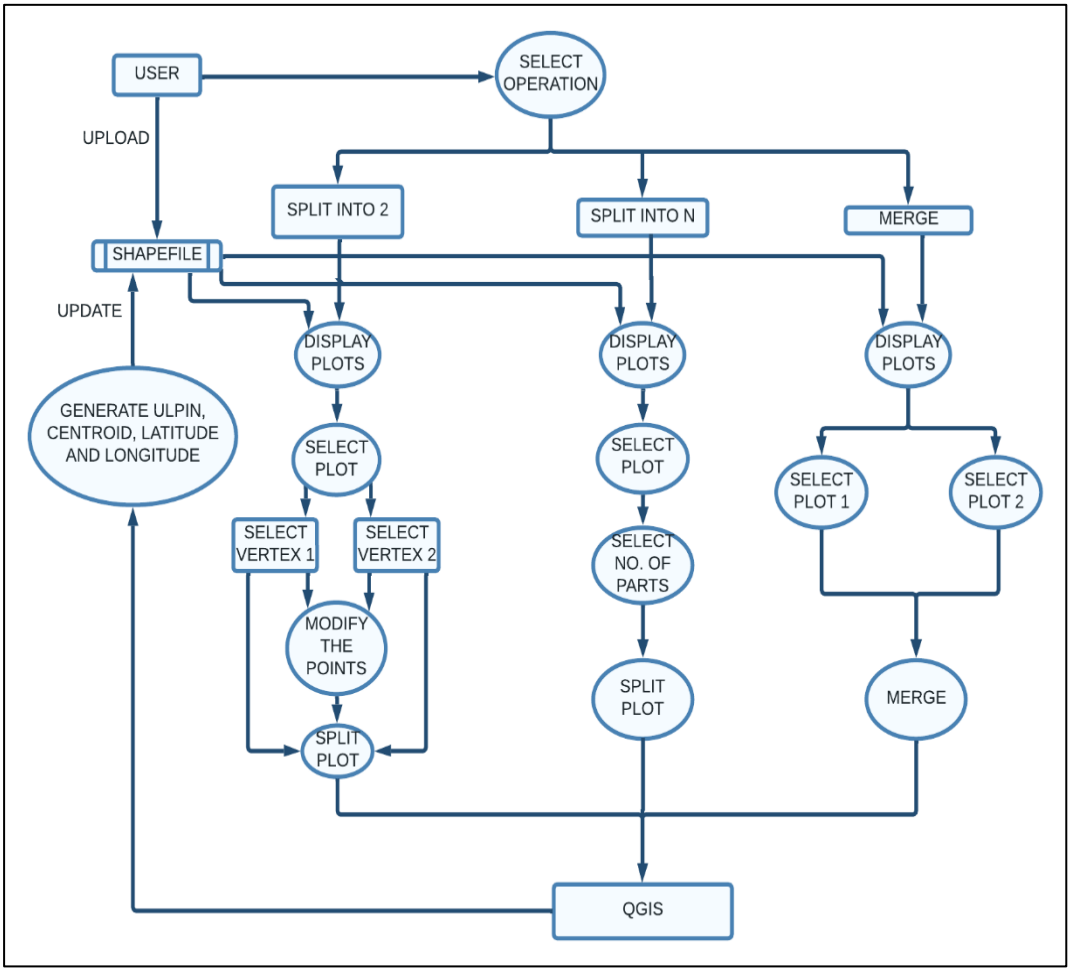


Level-1

MERGE PLUGIN:

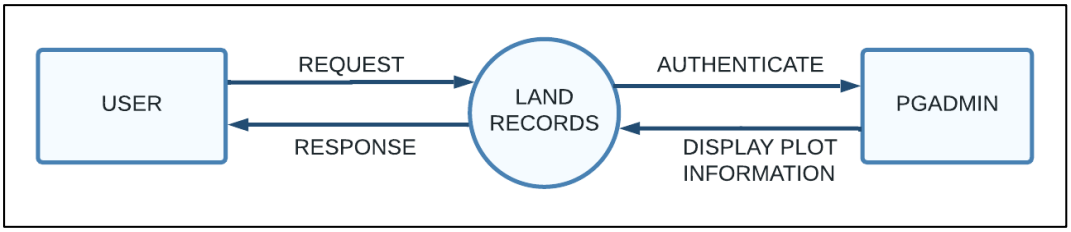


Level-0

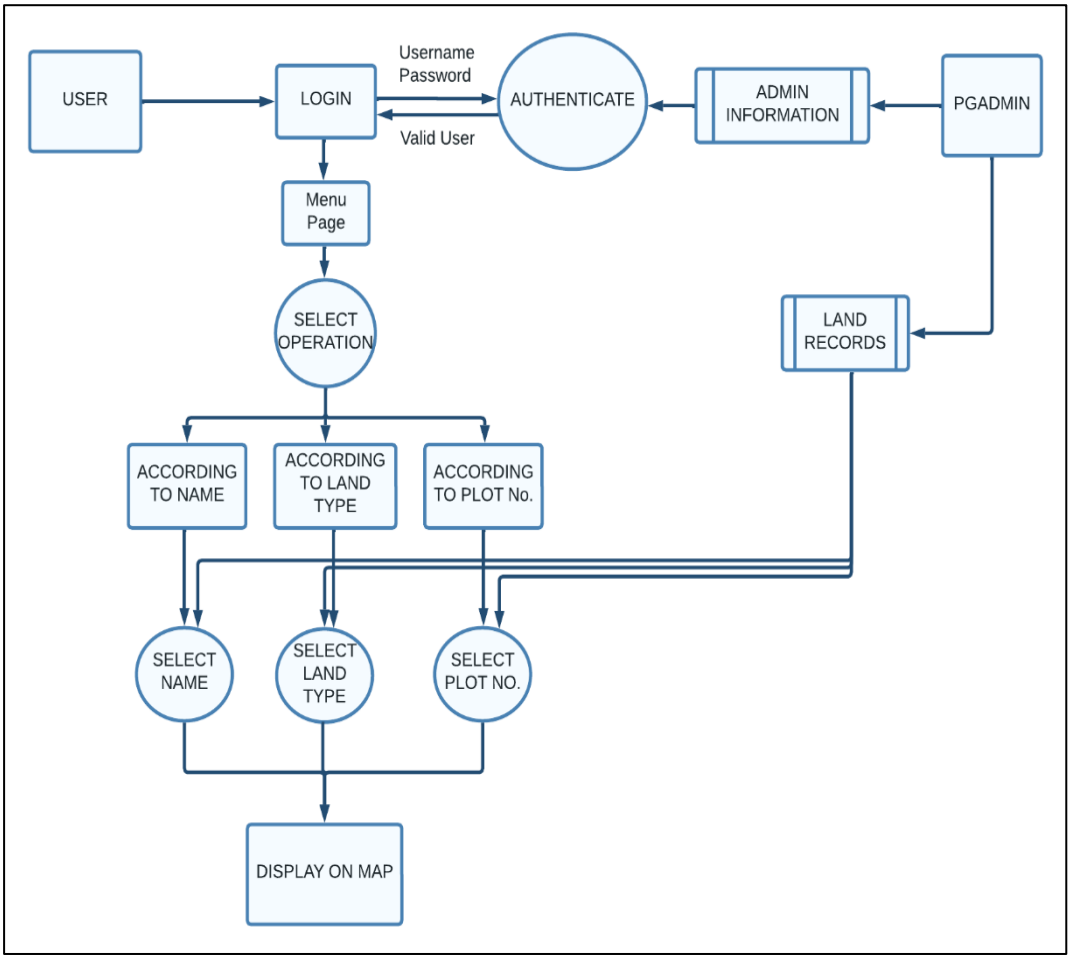


Level 1

LAND RECORD SYSTEM:



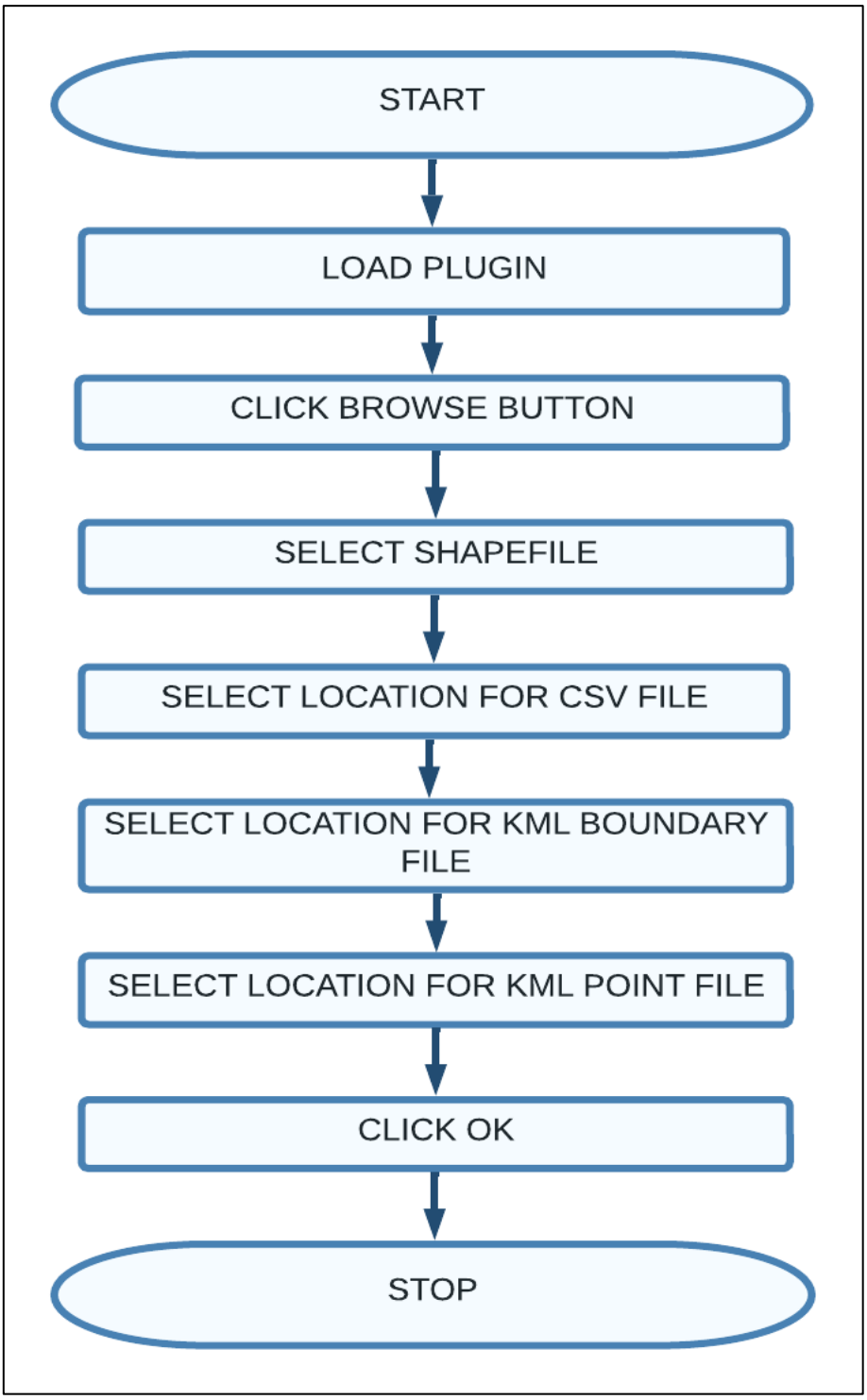
Level-0



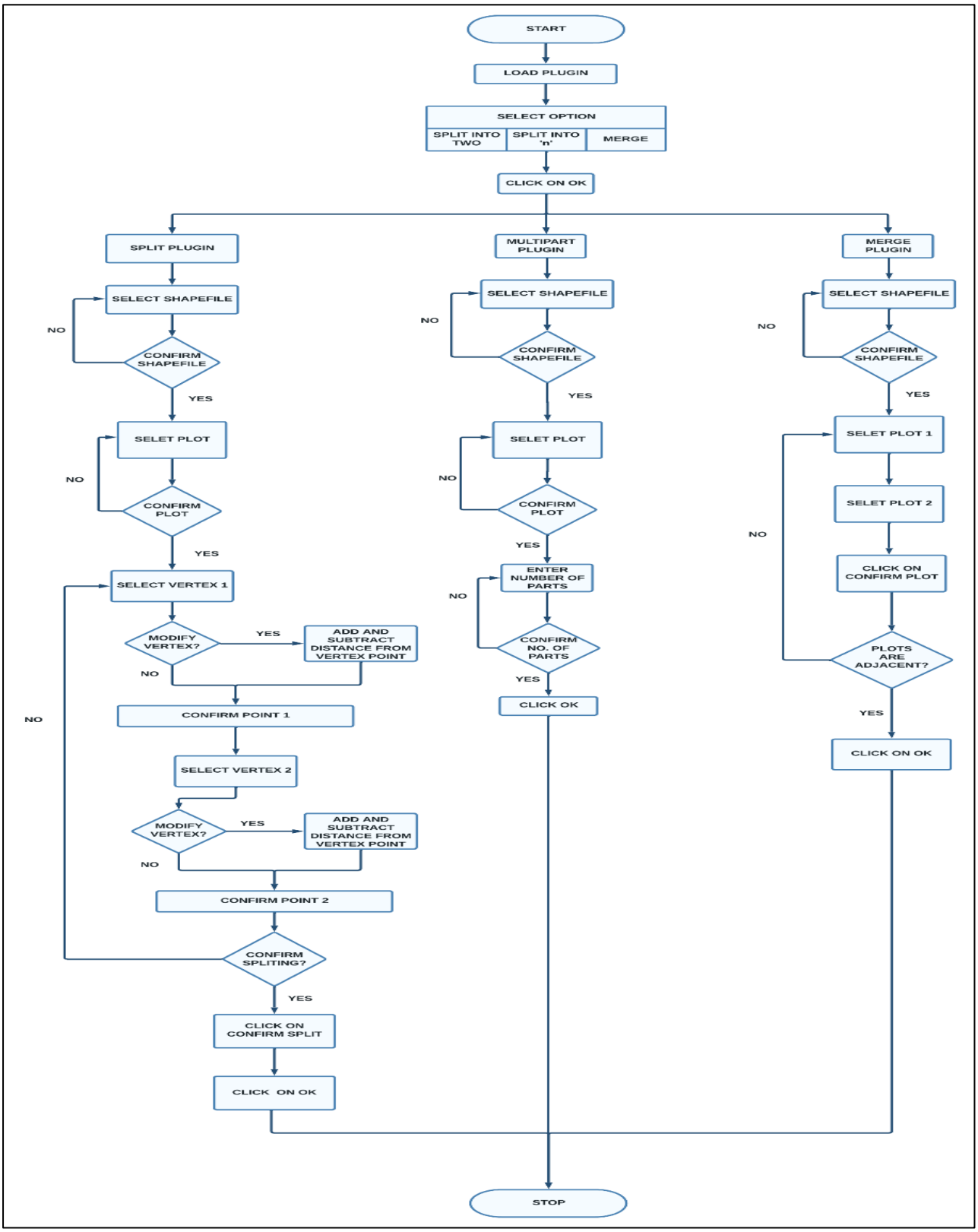
Level-1

E. **FLOWCHARTS**

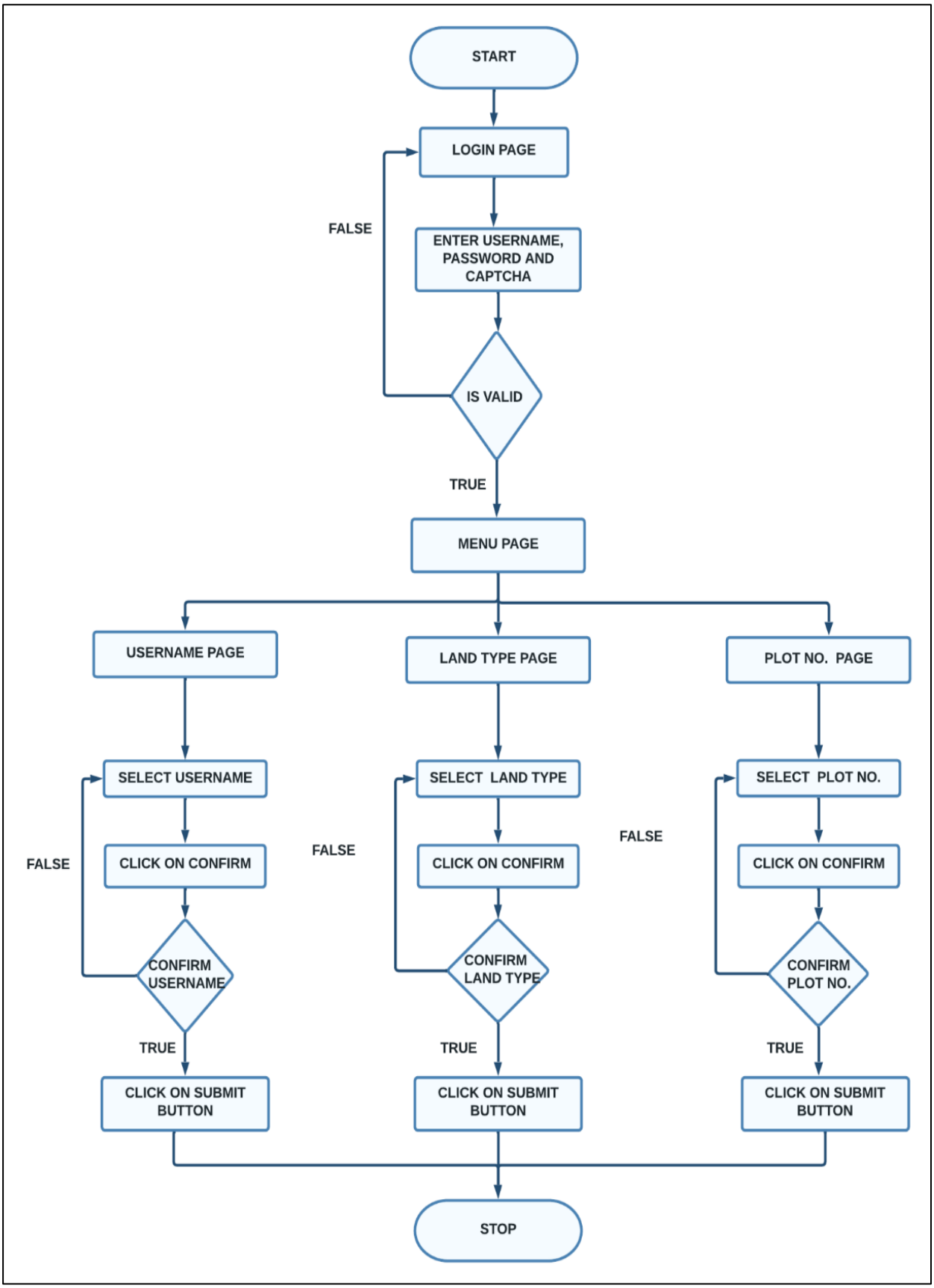
GP:



MERGE PLUGIN:



LAND RECORD SYSTEM:



CHAPTER 4- CODING

GP:

For plugin python code is divided into several different functionalities,each having separate task to perform.

1. Libraries to be imported and included in the file for supporting the inbuilt functions

2. To select location of input Shapefile,Output CSV file , OUTPUT KML Boundary File and Output.

```
// calling of functions
self.dlg.inputpushButton.clicked.connect(self.Upload_Point_Layer1)
self.dlg.csvpushButton.clicked.connect(self.Upload_Point_Layer2)
self.dlg.kmlpushButton.clicked.connect(self.Upload_Point_Layer3)
 self.dlg.pointpushButton.clicked.connect(self.Upload_Point_Layer4)
// functions
// shapefile location
def Upload_Point_Layer1(self):
      filename, _filter = QFileDialog.getOpenFileName(
      self.dlg, "Select Input text file ","", '*.shp')
      self.dlg.inputlineEdit.setText(filename)
// CSV file location
def Upload_Point_Layer2(self):
      filename, _filter = QFileDialog.getSaveFileName(
      self.dlg, "Select output text file ","", '*.csv')
      self.dlg.csvlineEdit.setText(filename)
// KML Boundary file location
def Upload_Point_Layer3(self):
      filename, _filter = QFileDialog.getSaveFileName(
      self.dlg, "Select output text file ","", '*.kml')
      self.dlg.kmllineEdit.setText(filename)
// KML point file location
```

```
def Upload_Point_Layer4(self):
    filename, _filter = QFileDialog.getSaveFileName(
    self.dlg, "Select output text file ","", '*.kml')
    self.dlg.pointlineEdit.setText(filename)
```

3. To display the shapefile using maps in QGIS

```
QgsProject.instance().clear()
filename = self.dlg.inputlineEdit.text()
orignalfile = QgsVectorLayer(filename, "polygon", "ogr")
QgsProject.instance().addMapLayer(orignalfile)
 layer = iface.activeLayer()
```

4. To create a CSV file using all calculated Centroid points, latitude, longitude and unique id for each plot in the shapefile.

```
# Writing into csv file
        header=layer.fields().names()
        k=abc.count('Latitude')
        if k==0:
           header.append('Latitude')
           header.append('Longitude')
           header.append('Unique point')
           header.append('Geometry')
        print(header)
        a=len(header)
        a=a-6
        with open(outputpath, 'w', encoding='UTF8', newline='') as f:
           writer = csv.writer(f)
           writer.writerow(header)
           features = layer.getFeatures()
           for feat in features:
```

```
id=feat.id()
h=utm.to_latlon(xc[id], yc[id], 45, 'N')
lat.append(h[0])
lon.append(h[1])
code=pniUgenerator(h[0],h[1],0)
c1.append(code)
l=[]
for I in range(a):
   p=gdf.loc[id,header[i]]
   l.append(p)
l.append(xc[id])
l.append(yc[id])
l.append(h[0])
l.append(h[1])
l.append(code)
l.append(g[id])
writer.writerow(l)
```

MERGE PLUGIN:

1. Libraries .

```
from qgis.PyQt.QtCore import QSettings, QTranslator, QCoreApplication
from qgis.PyQt.QtGui import QIcon
from qgis.PyQt.QtWidgets import QAction
# Initialize Qt resources from file resources.py
from .resources import *
# Import the code for the dialog
from .MergedPlugin_dialog import MergedPluginDialog
import os.path
import qgis
```

2. Selection of radio button and launching the corresponding plugin

```
# Assuming the radio buttons are called 'radioButton1', 'radioButton2', and 'radioButton3'
radio_buttons = [self.dlg.radioButton_1, self.dlg.radioButton_2, self.dlg.radioButton_3]
# Get the selected radio button
selected_radio_button = None
for radio_button in radio_buttons:
    if radio_button.isChecked():
        selected_radio_button = radio_button
        break
if selected_radio_button is not None:
    selected_radio_button_name = selected_radio_button.objectName()
print(selected_radio_button_name)
if(selected_radio_button_name=='radioButton_1'):
    secondPlugin = qgis.utils.plugins['split'] # plugin instance
    secondPlugin.run()
elif(selected_radio_button_name=='radioButton_2'):
    secondPlugin = qgis.utils.plugins['multipart'] # plugin instance
    secondPlugin.run()
elif(selected_radio_button_name=='radioButton_3'):
    secondPlugin = qgis.utils.plugins['mergepolygon'] # plugin instance
    secondPlugin.run()
```

3. SPLIT PLUGIN :

splitting of a plot into two parts .

```
def splitpoly(self):
      ########## variables ####################
    v=[]
    v1=[]
    v2=[]
    xend=[]
    yend=[]
    idend=[]
```

```
x=[]
y=[]
global id2
##### Extracting all vertices of polygon in a list - ver######
  id2=self.dlg.selectComboBox.currentText()
  filename = self.dlg.inputlineEdit.text()
layerList = QgsProject.instance().mapLayersByName("polygon")
layer=layerList[0]
gdf = gpd.read_file(filename)
features = layer.getFeatures()
global gid
for feat in features:
   if(id2==feat['ID1']):
      print(id2)
      print("feat['ID1']")
      print(feat['ID1'])
      gid=feat.id()
      mypolygon=gdf.loc[gid,'geometry']
  ver=list(mypolygon.exterior.coords)
#print("vertices of polygon")
#print(ver)
shapely_poly = shapely.geometry.Polygon(ver)
  ####### Extract vertex number from comboBox ##################
a=self.dlg.firstcomboBox.currentText()
b=self.dlg.secondcomboBox.currentText()
v.append(int(a[1:]))
v.append(int(b[1:]))
##### Values from input boxes###############
  vf1=self.dlg.forwardlineEdit1.text()    # horizontal of first vertex
vb1=self.dlg.backwardlineEdit1.text()   #vertical of first vertex
vf2=self.dlg.forwardlineEdit2.text()    #horizontal of second vertex
vb2=self.dlg.backwardlineEdit2.text()   #vertical of second vertex
  # print("v[0]")
```

```
# print(v[0])
# print("v[1]")
# print(v[1])
if(v[0]<v[1]):
    if(vf1==""):
        v1.append(float(0))
    else:
        v1.append(float(vf1))
    if(vb1==""):
        v1.append(float(0))
    else:
        v1.append(float(vb1))
    if(vf2==""):
        v2.append(float(0))
    else:
        v2.append(float(vf2))
    if(vb2==""):
        v2.append(float(0))
    else:
        v2.append(float(vb2))
   else:
    v.sort()
    if(vf2==""):
        v1.append(float(0))
    else:
        v1.append(float(vf2))
    if(vb2==""):
        v1.append(float(0))
    else:
        v1.append(float(vb2))
    if(vf1==""):
        v2.append(float(0))
    else:
```

```
        v2.append(float(vf1))
    if(vb1==""):
        v2.append(float(0))
    else:
        v2.append(float(vb1))
######### Getting points from user to draw line #############
########## Extract points of the selected vertices ##################
layerList = QgsProject.instance().mapLayersByName("point")
layer=layerList[0]
layer.select(v)
selection = layer.selectedFeatures()
for feat in selection:
    idend.append(feat['id2'])
    xend.append(feat['xCord'])
    yend.append(feat['yCord'])
   ##########  Calculating points for line creation ##############
if(idend[0]>idend[1]):
    x.append(float( float(xend[1]) + v1[0] ))
    y.append(float( float(yend[1]) + v1[1]))
    x.append(float( float(xend[0]) + v2[0] ))
    y.append(float( float(yend[0]) + v2[1]))
else:
    x.append(float( float(xend[0]) + v1[0] ))
    y.append(float( float(yend[0]) + v1[1]))
    x.append(float( float(xend[1]) + v2[0] ))
    y.append(float( float(yend[1]) + v2[1]))
########## Line for dividing polygon ############################
line = [(x[0],y[0]), (x[1],y[1])]
shapely_line = shapely.geometry.LineString(line)
########## intersection of line and polygon ##########
intersection_line = list(shapely_poly.intersection(shapely_line).coords)
print("intersection_line")
print(intersection_line)
```

```
   ########### create vector layer for line ########################
f.setGeometry(QgsGeometry.fromPolylineXY([QgsPointXY(float(intersection_l
ine[0][0]),float(intersection_line[0][1])),QgsPointXY(float(intersection_line[1][0
]), float(intersection_line[1][1]))]))
   # provider.addFeature(f)
   # layer.updateExtents()
   # QgsProject.instance().addMapLayer(layer)
     ############## plotting the new points in layer ###########
  layer=QgsVectorLayer('Point?crs=EPSG:32645','intersectionpoints','memory')
   provider=layer.dataProvider()
   f=QgsFeature()
   provider.addAttributes([QgsField('id',QVariant.Int)])
   provider.addAttributes([QgsField('x',QVariant.String)])
   provider.addAttributes([QgsField('y',QVariant.String)])
   layer.updateFields()
   p1=QgsPointXY(float(intersection_line[0][0]),float(intersection_line[0][1]))
   f.setGeometry(QgsGeometry.fromPointXY(p1))
   f.setAttributes([1,str(intersection_line[0][0]),str(intersection_line[0][1])])
   provider.addFeature(f)
   p1=QgsPointXY(float(intersection_line[1][0]),float(intersection_line[1][1]))
   f.setGeometry(QgsGeometry.fromPointXY(p1))
   f.setAttributes([2,str(intersection_line[1][0]),str(intersection_line[1][1])])
   provider.addFeature(f)
   layer.updateExtents()
   QgsProject.instance().addMapLayer(layer)
   ############finding id for new points ##############
   point1 = QgsPointXY(intersection_line[0][0],intersection_line[0][1])
   point2 = QgsPointXY(intersection_line[1][0],intersection_line[1][1])
   if(v[0]==1):
      checkid1=len(ver)-1
   else:
      checkid1=v[0]-1
     if(v[0]==len(ver)):
```

```
    checkid2=1
else:
    checkid2=v[0]+1
   if(v[1]==1):
    checkid3=len(ver)-1
else:
    checkid3=v[1]-1
   if(v[1]==len(ver)):
    checkid4=2
else:
    checkid4=v[1]+1
print("checkid1")
print(checkid1)
print("checkid2")
print(checkid2)
print("checkid3")
print(checkid3)
print("checkid4")
print(checkid4)
     ####### Calculating id for intersection point ################


########### vertex 1 ##############
pv1 = QgsPointXY(float(ver[checkid1-1][0]),float(ver[checkid1-1][1]))
pv3=QgsPointXY(float(ver[checkid2-1][0]),float(ver[checkid2-1][1]))
pv2=QgsPointXY(float(ver[v[0]-1][0]),float(ver[v[0]-1][1]))
point_feature = QgsFeature()
point_feature.setGeometry(QgsGeometry.fromPointXY(point1))
line_feature1 = QgsFeature()
line_feature1.setGeometry(QgsGeometry.fromPolylineXY([pv1,pv2]))
line_feature2 = QgsFeature()
line_feature2.setGeometry(QgsGeometry.fromPolylineXY([pv2,pv3]))
if(vf1=="" and vb1==""):
    myid1=v[0]
```

```
      else:
       # Check if the point is on the line between vertex1 and vertex1-1
       result=
str(line_feature1.geometry().closestSegmentWithContext(point_feature.geometr
y().asPoint()))
       r=result.split(",")
       r2=r[0]
       r1=float(r2[1:])
       if (r1== 0.0):
          myid1=v[0]
            # Check if the point is on the line between vertex1 and vertex1+1
       result=
str(line_feature2.geometry().closestSegmentWithContext(point_feature.geometr
y().asPoint()))
       r=result.split(",")
       r2=r[0]
       r1=float(r2[1:])
       if (r1== 0.0):
          myid1=v[0]+1
         print("id of first vertex ")
   print(myid1)
 ############ vertex 2 ###################
   pv4 = QgsPointXY(float(ver[checkid3-1][0]),float(ver[checkid3-1][1]))
   pv6=QgsPointXY(float(ver[checkid4-1][0]),float(ver[checkid4-1][1]))
   pv5=QgsPointXY(float(ver[v[1]-1][0]),float(ver[v[1]-1][1]))
   point_feature = QgsFeature()
   point_feature.setGeometry(QgsGeometry.fromPointXY(point2))
   line_feature3 = QgsFeature()
   line_feature3.setGeometry(QgsGeometry.fromPolylineXY([pv4,pv5]))
   line_feature4 = QgsFeature()
   line_feature4.setGeometry(QgsGeometry.fromPolylineXY([pv5,pv6]))
   if(vf2=="" and vb2==""):
      myid2=v[1]
```

```
    else:
        # Check if the point is on the line between vertex1 and vertex1-1
        result=
str(line_feature3.geometry().closestSegmentWithContext(point_feature.geometr
y().asPoint()))
        r=result.split(",")
        r2=r[0]
        r1=float(r2[1:])
        if (r1== 0.0):
            myid2=v[1]
              # Check if the point is on the line between vertex1 and vertex1+1
        result=
str(line_feature4.geometry().closestSegmentWithContext(point_feature.geometr
y().asPoint()))
        r=result.split(",")
        r2=r[0]
        r1=float(r2[1:])
        if (r1== 0.0):
            myid2=v[1]+1
           print("id of second vertex ")
    print(myid2)

    ######################### separating the polygon into two polygons using
the intersection vertices ########################
    xpoint=[]
    ypoint=[]

    layerList = QgsProject.instance().mapLayersByName("point")
    layer=layerList[0]
    features = layer.getFeatures()
    gdf = gpd.read_file(filename)
    j=0
```

```
    for feat in features:
        xpoint.append(feat['xCord'])
        ypoint.append(feat['yCord'])
        j+=1

    pointList1=[]
    pointList2=[]
    global poly1,poly2
    poly1=[]
    poly2=[]
    ################################### Adding points in list ####################
    xpoint.insert(myid1-1,intersection_line[0][0])
    ypoint.insert(myid1-1,intersection_line[0][1])
    xpoint.insert(myid2,intersection_line[1][0])
    ypoint.insert(myid2,intersection_line[1][1])
    j=j+2
    for i in range(j):
        print(str(i)+":"+str(xpoint[i])+","+str(ypoint[i]))



    ###################333distributing points into two polygon
    for i in range(j):

        if( i>=(myid1-1) and i<=(myid2)):

            temp=QgsPointXY(float(xpoint[i]),float(ypoint[i]))
            poly1.append(temp)

        if(i<=(myid1-1)):

            temp=QgsPointXY(float(xpoint[i]),float(ypoint[i]))
            poly2.append(temp)
```

```
    if(i>=(myid2)):

        temp=QgsPointXY(float(xpoint[i]),float(ypoint[i]))
        poly2.append(temp)

  end1=poly1[0]
  end2=poly2[0]
  poly1.append(end1)
  poly2.append(end2)

  pointList1 = geometry.Polygon(poly1)
  pointList2 = geometry.Polygon(poly2)

  layer = QgsVectorLayer('Polygon?crs=EPSG:32645', 'polydivider1' ,
"memory")
  pr = layer.dataProvider()

  pr.addAttributes([QgsField('ID1',QVariant.String)])
  pr.addAttributes([QgsField('xCentroid',QVariant.String)])
  pr.addAttributes([QgsField('yCentroid',QVariant.String)])
  pr.addAttributes([QgsField("Latitude",QVariant.String)])
  pr.addAttributes([QgsField("Longitude",QVariant.String)])
  pr.addAttributes([QgsField("unique id",QVariant.String)])
  pr.addAttributes([QgsField('polygon',QVariant.String)])
  layer.updateFields()
  poly = QgsFeature()
  global finalid1,finalid2
  finalid1=str(str(id2)+"/1")
  finalid2=str(str(id2)+"/2")
  # finalid1="159"
  # finalid2="160"
```

```
    c1=list(pointList1.centroid.coords)
    c2=list(pointList2.centroid.coords)

    h1=utm.to_latlon(c1[0][0], c1[0][1], 45, 'N')
    h2=utm.to_latlon(c2[0][0], c2[0][1], 45, 'N')

    code1=pniUgenerator(h1[0],h1[1],0)
    code2=pniUgenerator(h2[0],h2[1],0)
    poly.setAttributes([finalid1,str(c1[0][0]),str(c1[0][1]),str(h1[0]),str(h1[1]),str(c
ode1),str(pointList1)])

    poly.setGeometry(QgsGeometry.fromPolygonXY([poly1]))
    pr.addFeatures([poly])

    poly.setGeometry(QgsGeometry.fromPolygonXY([poly2]))
    poly.setAttributes([finalid2,str(c2[0][0]),str(c2[0][1]),str(h2[0]),str(h2[1]),str(c
ode2),str(pointList2)])
    pr.addFeatures([poly])

    layer.updateExtents()
    QgsProject.instance().addMapLayer(layer)

    ########################centroid                                    layer
####################################333
    print("creating centroid layer for polygons" )
    layer=QgsVectorLayer('Point?crs=EPSG:32645','polycentroid','memory')
    provider=layer.dataProvider()
    provider.addAttributes([QgsField('id2',QVariant.String)])
    provider.addAttributes([QgsField('xCentroid',QVariant.String)])
    provider.addAttributes([QgsField('yCentroid',QVariant.String)])
    layer.updateFields()
    f=QgsFeature()
```

```
p1=QgsPointXY(c1[0][0],c1[0][1])
f.setGeometry(QgsGeometry.fromPointXY(p1))
f.setAttributes([finalid1,str(c1[0][0]),str(c1[0][1])])
provider.addFeature(f)
p1=QgsPointXY(c2[0][0],c2[0][1])
f.setGeometry(QgsGeometry.fromPointXY(p1))
f.setAttributes([finalid2,str(c2[0][0]),str(c2[0][1])])
provider.addFeature(f)
layer.updateExtents()
QgsProject.instance().addMapLayer(layer)
```

4. MULTIPART PLUGIN

```
poly_qgs=iface.activeLayer()
# extent of the polygon
extentrect=poly_qgs.extent()
xmax = extentrect.xMaximum()
ymax = extentrect.yMaximum()
xmin = extentrect.xMinimum()
ymin = extentrect.yMinimum()
extentreg=str(xmin) + ',' + str(xmax) + ','+ str(ymin)+ ',' + str(ymax)
# extract the vertices of the polygon
paramextract = {'INPUT': poly_qgs, 'OUTPUT': 'memory:'}
polyvertices = processing.run("native:extractvertices", paramextract)
QgsProject.instance().addMapLayers([polyvertices['OUTPUT']])
# clustering
paramcluster = {'CLUSTERS': N_DIV, 'FIELD_NAME': 'CLUSTER_ID', 'INPUT':
polyvertices['OUTPUT'], 'OUTPUT': 'memory:', 'SIZE_FIELD_NAME':
'CLUSTER_SIZE'}
verticesclusters = processing.run("native:kmeansclustering", paramcluster)#temp
# mean coordinates
```

```
parammean = {'INPUT': verticesclusters['OUTPUT'], 'OUTPUT': 'memory:', 'UID': 'CLUSTER_ID', 'WEIGHT': ''}
verticesmean = processing.run("native:meancoordinates", parammean)
QgsProject.instance().addMapLayers([verticesmean['OUTPUT']])
# voronoi polygons
paramvoro = {'BUFFER': 100, 'INPUT': verticesmean['OUTPUT'], 'OUTPUT': 'memory:'}
voronoi_poly = processing.run("qgis:voronoipolygons", paramvoro)
QgsProject.instance().addMapLayers([voronoi_poly['OUTPUT']])
```

5.MERGE POLYGON PLUGIN

Merging two polygons

```
gdf = gpd.read_file(filename)
features = layer.getFeatures()
print(l)
for feat in features:
    if(int(id1)==feat.id()):
        mypolygon=gdf.loc[int(id1),'geometry']
        break
print(type(mypolygon))
print(merpolygon)
print(type(merpolygon))
multipolygon = MultiPolygon([merpolygon])
```

Delete a row in shapefile

```
layers=QgsProject.instance().mapLayersByName('polygon')
layer=layers[0]
```

```
caps=layer.dataProvider().capabilities()
dfeats=[]
selection = layer.selectedFeatures()
# creating point layer for selected plot
for feat in selection:
    id=feat.id()
    layer.deselect(id)
if caps & QgsVectorDataProvider.DeleteFeatures:
    dfeats.append(gid1)
    dfeats.append(gid2)
    res=layer.dataProvider().deleteFeatures(dfeats)
```

6.LAND RECORD SYSTEM:

Land Record System consists of 2 main functions and their codes are as follows:

1. Login

```
<div class="login_form">
    <div class="form">
        <form action="" method="POST">
            <h1 class="form_title"> Log In </h1>
            <div class="form_div">
                <input type="text" name="username" id="username"class="form_input"
placeholder=" ">
                <label class="form_label">User Name</label>
            </div>
            <div class="form_div">
                <input type="password" name="password" id="password"
class="form_input" placeholder=" ">
                <label class="form_label">Password</label>
```

```
        </div>
        <div id="captcha" class="form_div">
          <div class="preview"></div>
          <div class="captcha_form">
            <input type="text" id="captcha_form" class="form_input_captcha"
placeholder=" ">
            <label class="form_label_captcha">Enter Captcha</label>
            <button class="captcha_refersh"> <i class="fa fa-refresh"></i>
          </button>
          </div> </div>
</form>
  <form> <input class="form_button" value="Log In"> </form>
    </div>
  </div>
```

1.1 Database connection

```
$host = 'localhost';
$port = '5432';
$dbname = 'arjun';
$user = 'postgres';
$password = '123456';
$conn = pg_connect("host=$host port=$port dbname=$dbname user=$user
password=$password");
```

2.Menu

```
<div class="bg">
        <form action="AllName.php" >
    <button class="btn">
                    According to owner Name
     </button></form><br>
```

```
  <form action="LandType.php" >
    <button class="btn">
                          According to Land type
      </button></form><br>
  <form action="Plotno.php" >
    <button class="btn">
                          According to Plot Number
      </button></form><br>
 </div>
```

2.1 ALLNAME

```
<label>NAME:</label>
   <select class="input" name="mydropdown">
   <option value="">Select Name</option>
<?php
 //Query to retrieve all the name from the table.
  $sql ="SELECT owner_name FROM sibitrans1";
   global $selectedOption;
   if ($_SERVER["REQUEST_METHOD"] == "POST") {
   $selectedOption = $_POST["mydropdown"]; // assign the selected value to a
variable
      }
    echo "You selected: " . $selectedOption;
    ?>
   <input type="hidden" name="myVariable" value="<?php echo
$selectedOption; ?>">
```

Similar code is used for “plotno.php” and “landtype.php” pages by changing SQL query:

```
$sql ="SELECT new_sl_no FROM sibitrans1";
```

```
$sql ="SELECT DISTINCT land_type FROM sibitrans1";
```

Script for Displaying the map.

```
<script type="text/javascript">
var map = L.map('map');
var layer = new
L.TileLayer('http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png');

layer.addTo(map);
map.setView([27.90, 95.01], 9);
$.getJSON("http://localhost/arjun/formdata.php", function(data) {
addDataToMap(data,map); });
function addDataToMap(data, map) {
   var dataLayer = L.geoJson(data, {
     style: {
    fillColor: 'red'
   },
      onEachFeature: function(feature, layer) {
         var popupText = "New_SL_No: " + feature.properties.new_sl_no
            + "<br>Owner Name: " + feature.properties.owner_name;
            //+ "<br><a href='" + feature.properties.url + "'>More info</a>";
         layer.bindPopup(popupText); }
      });
   dataLayer.addTo(map);
}
</script>
```

Similar code is used for “plot.html” and “land.html” pages by changing feature property to plot information and land type information respectively for displaying on map.

To retrieve data from database(pgAdmin)

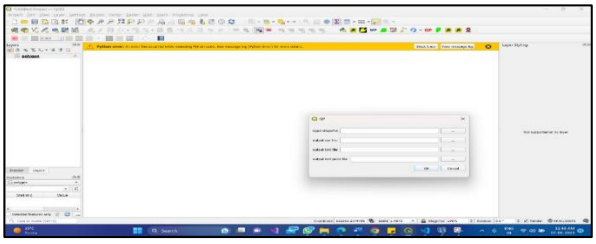
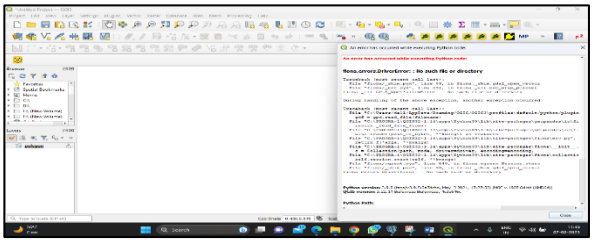
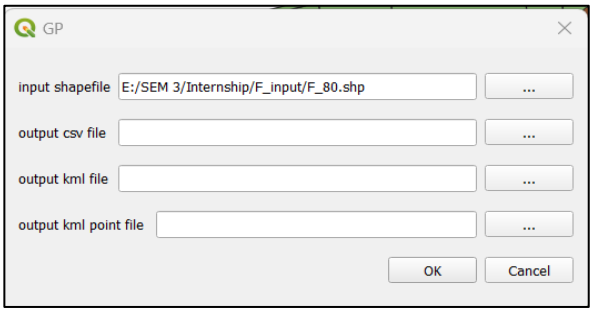
```
$conn = new PDO('pgsql:host=localhost;dbname=arjun','postgres','123456');
```

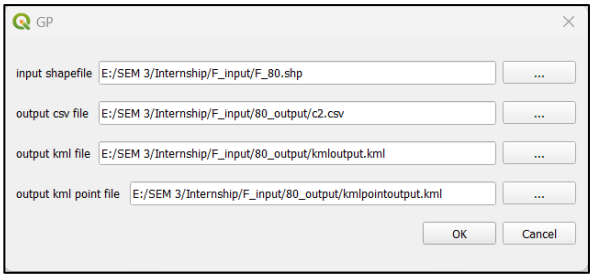
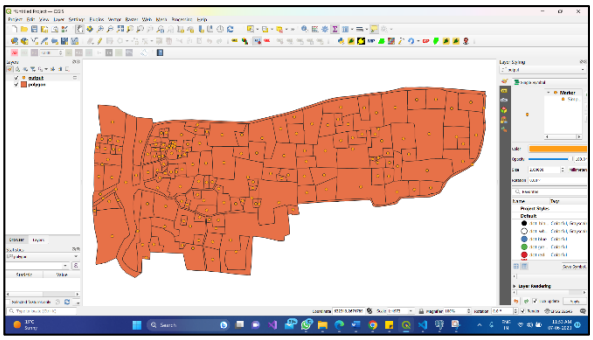
```
# Build SQL SELECT statement and return the geometry as a GeoJSON
element
$t1="'sibitrans1'";
$t2="'OName'";
$sql = "SELECT *,
public.ST_AsGeoJSON(public.ST_Transform((geom),4326),6) AS geojson
FROM ".$t1." INNER JOIN ".$t2." ON
".$t1.".owner_name=".'$t2.".owner_name";
```

Similar code is used for “plotdata.php” and “landdata.php” pages by changing $t2 value to “plot” and “Ltype” respectively for extracting plot information and land type information.

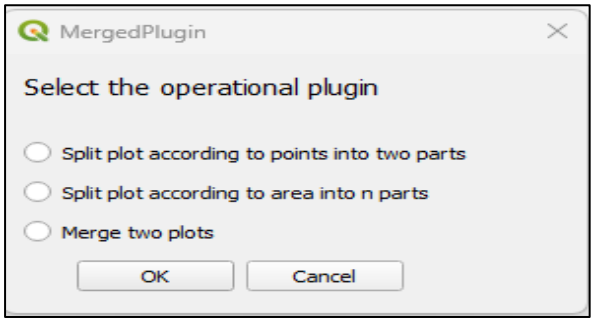
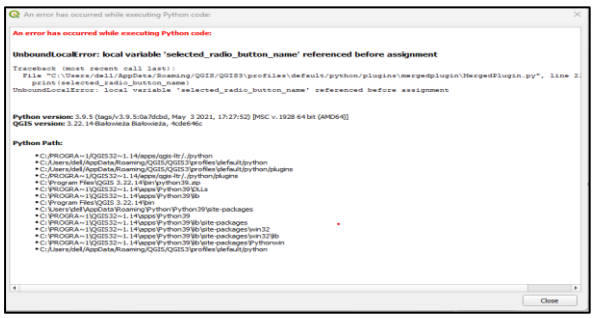
CHAPTER 5- TEST CASES

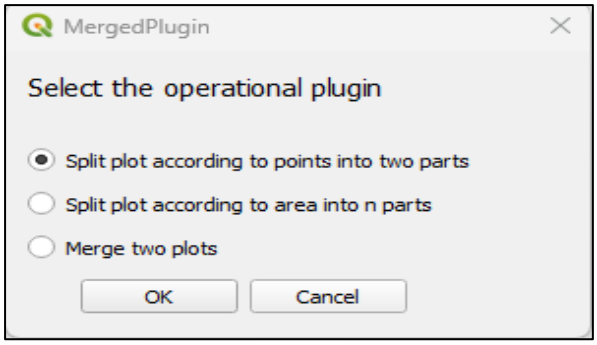
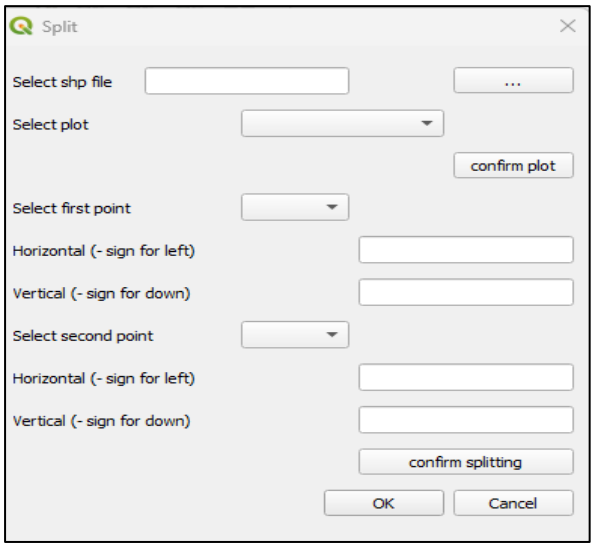
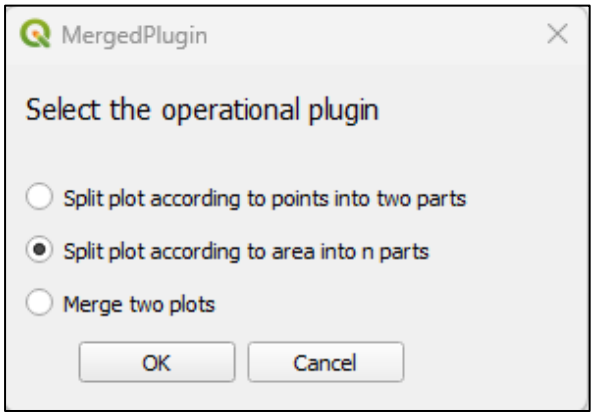
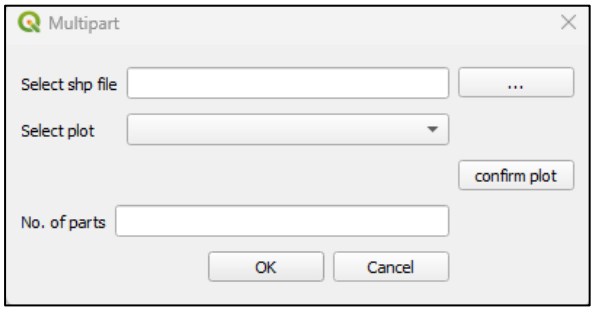
GP:

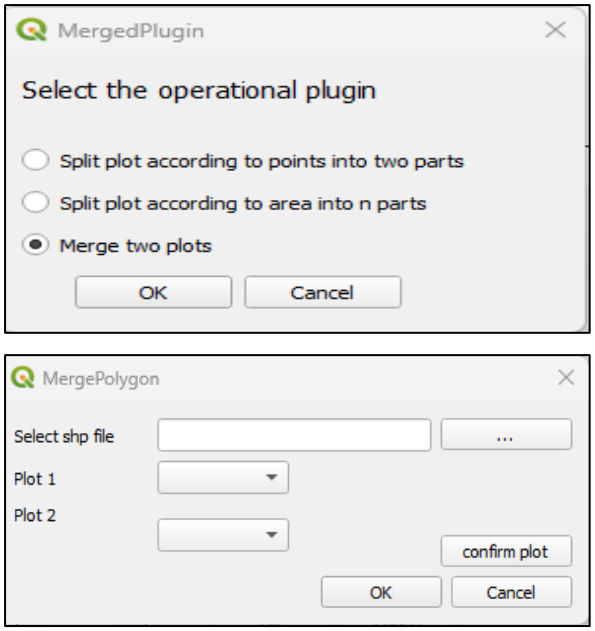
Sr. No	Test Case	Expected Output	Actual Result	PASS /FAIL
1.	When no file selected	A message should occur informing this to the user.	 	PASS
2.	When only the input file is selected. Path of output files are not provided	An error should occur: “no such file or directory”	 	PASS

3.	When all file locations are provided.	Plugin should run properly and create new files.	 	**PASS**

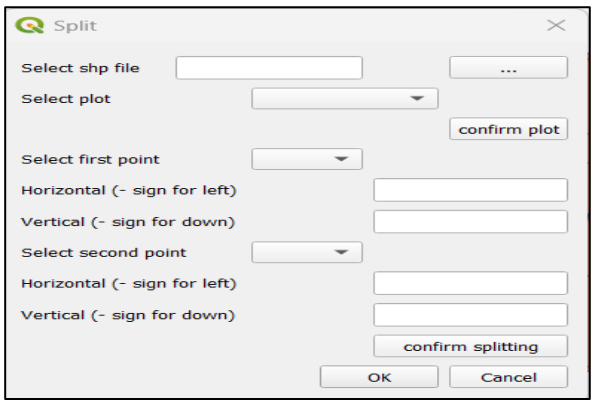
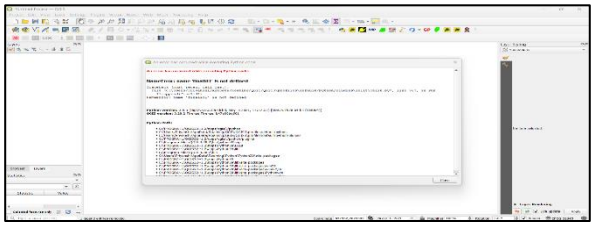
MERGE PLUGIN:

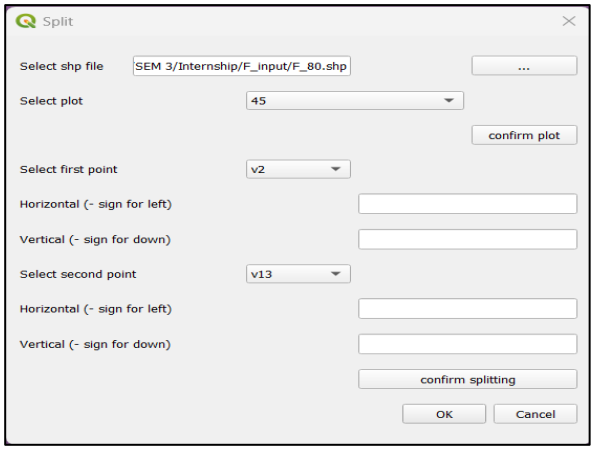
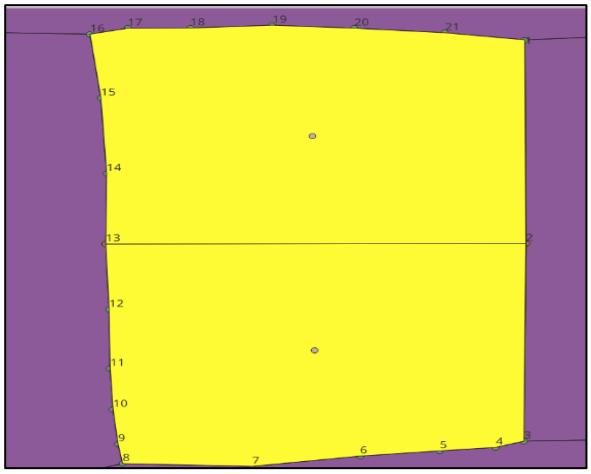
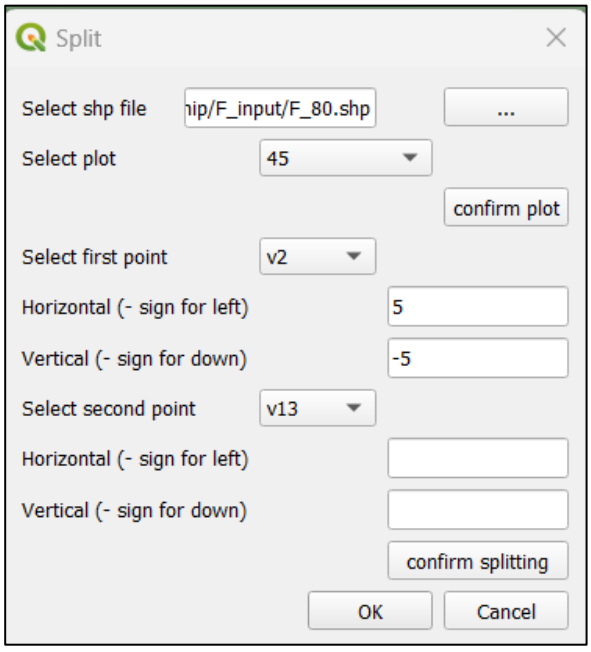
Sr. No	**Test Case**	**Expected Output**	**Actual Result**	**PASS /FAIL**
1.	When no option selected	A message should occur informing this to the user.	 	**PASS**

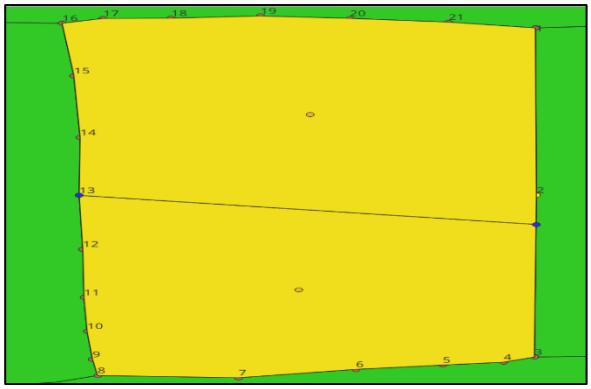
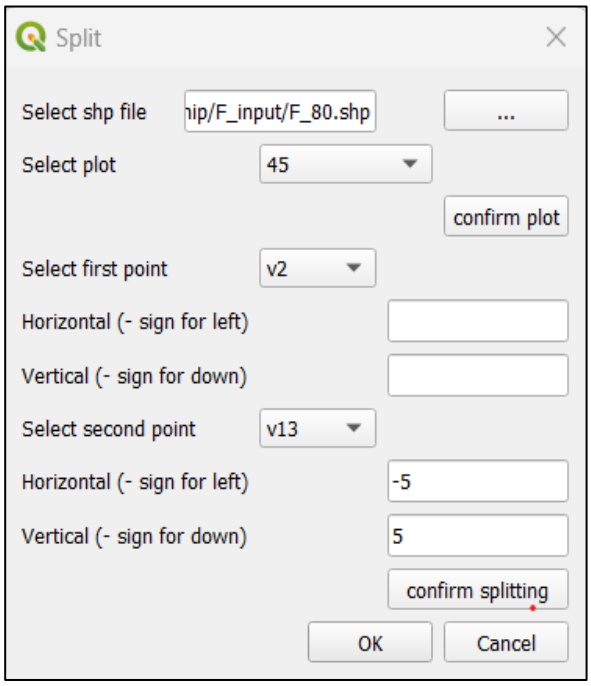
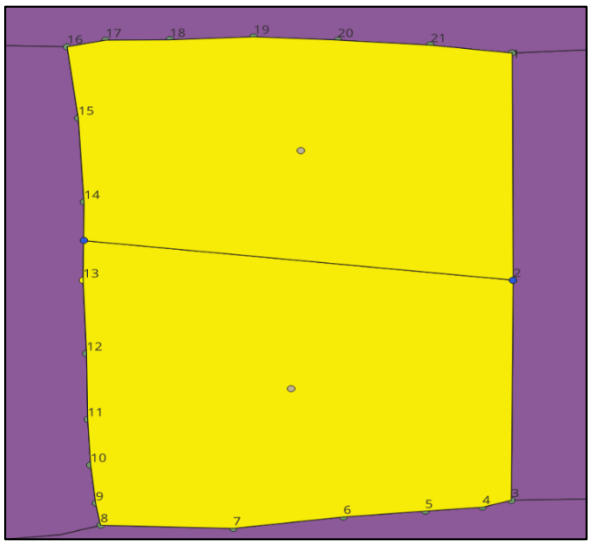
2.	When split plot according to points into two parts is Selected.	A split plugin open which is used to split plot into two.	 	**PASS**
3.	When split plot according to area into n parts selected.	A multipart plugin opens which is used to split the plot into n parts.	 	**PASS**

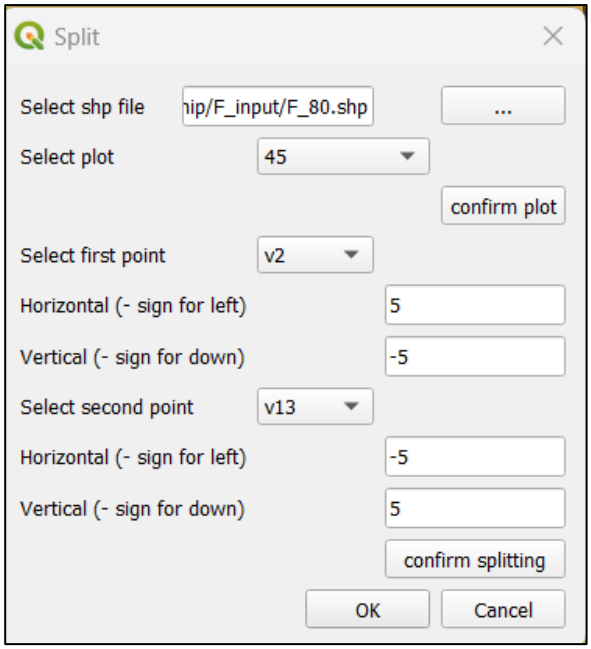
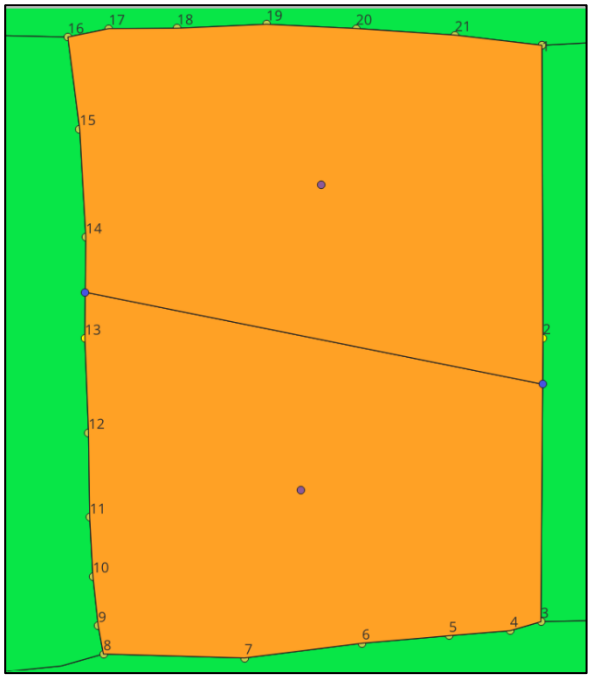
4.	When Merge two plots selected.	MergePolygon plugin opens which is used to merge two plots.		**PASS**

SPLIT:

Sr. No	**Test Case**	**Expected Output**	**Actual Result**	**PASS /FAIL**
1.	When no option selected	A message should occur informing this to the user.	 	**PASS**

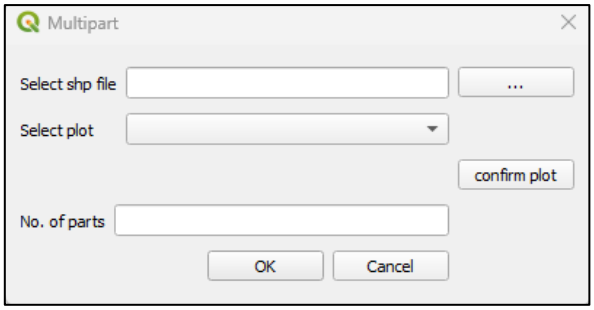
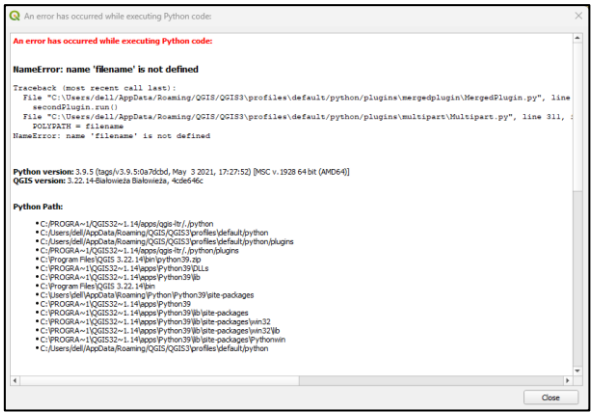
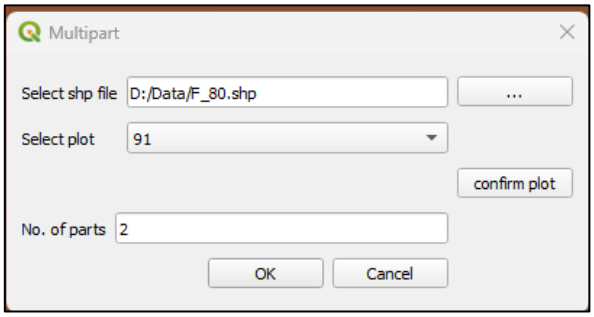
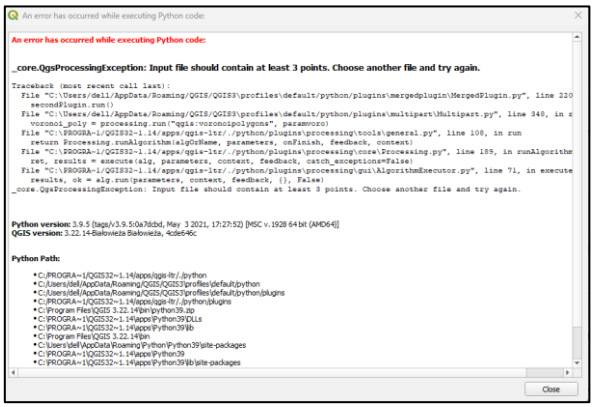
2.	When the user selects the vertex points for splitting the plot.	The splitting of the plot should occur from the two vertex points.	 	**PASS**
3.	When the point of intersection is at a particular distance from the first point .	The splitting of the plot should be done from the specified distance from the first point		**PASS**

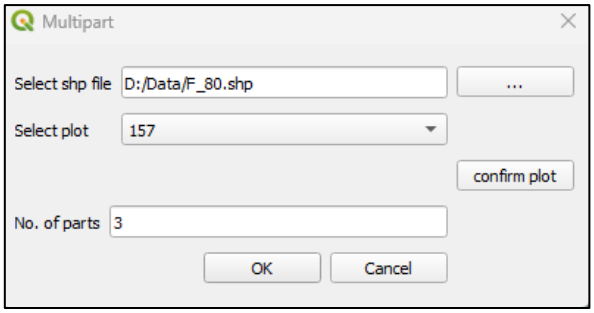
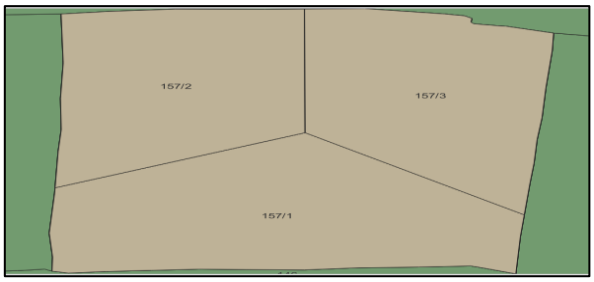
				
4.	When the point of intersection is at a particular distance from the second point .	The splitting of the plot should be done from the specified distance from the second point.	 	**PASS**

5.	When the user splits the plot into parts from the points which are at a distance from the vertex points.	The plot must split from the points which are at the given distance from the vertex points.	 	**PASS**

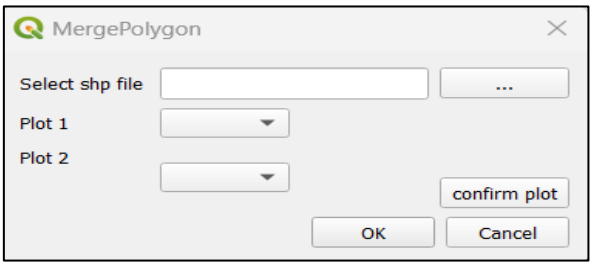
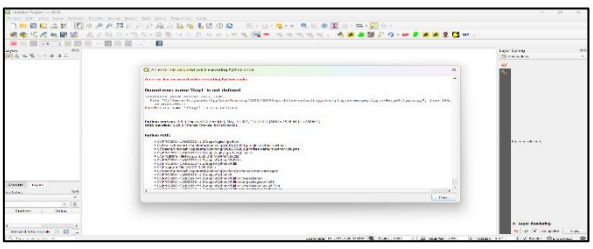
MULTI PART:

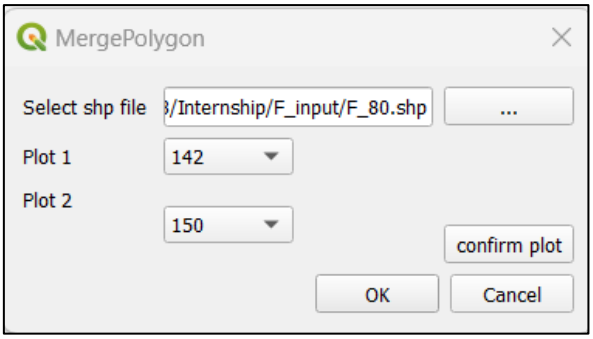
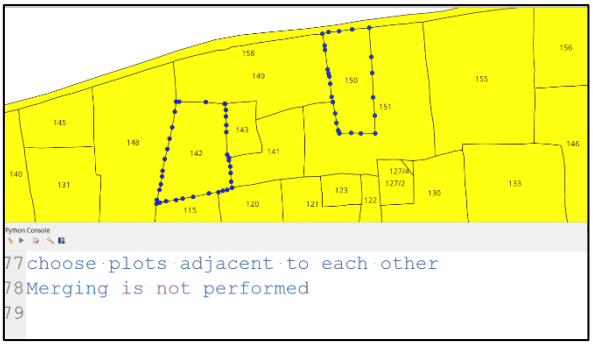
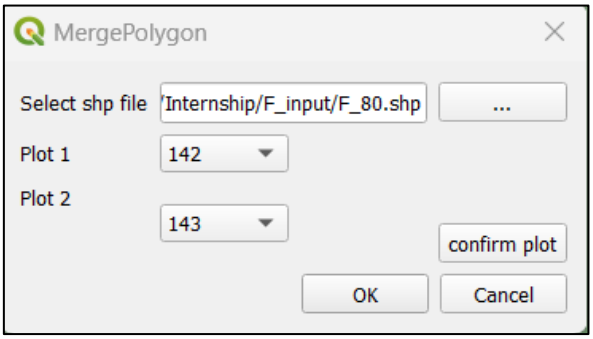
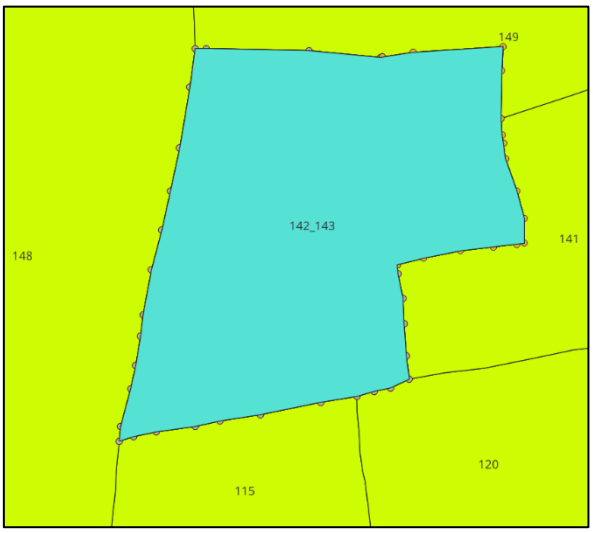
Sr. No	**Test Case**	**Expected Output**	**Actual Result**	**PASS /FAIL**

1.	When no option selected	A message should occur informing this to the user.	 	**PASS**
2.	When the user enters a number 2 or less than 2 for splitting the plot.	A message should occur informing this to the user	 	**PASS**

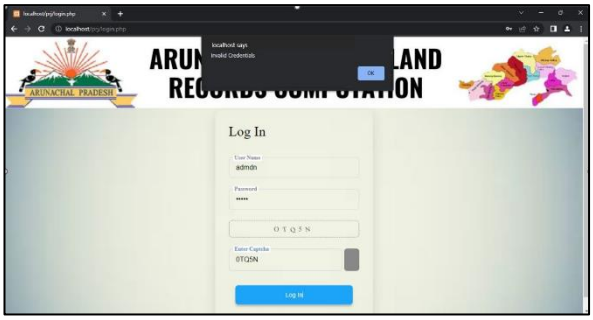
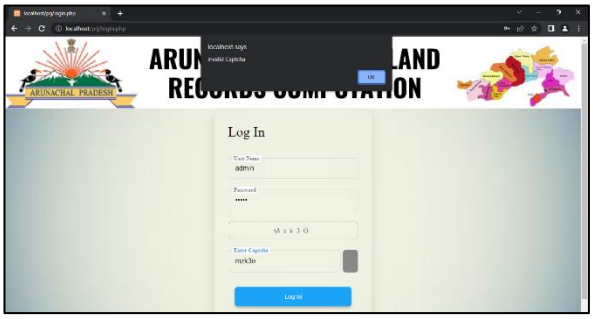
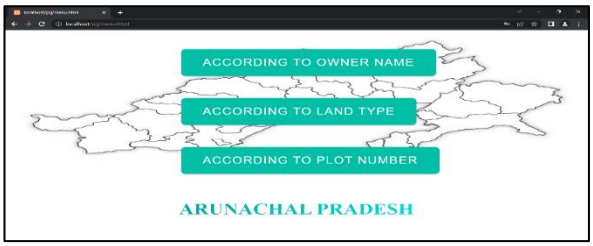
3.	When the user enters a number greater than 2 for splitting the plot.	The splitting of the plot into n parts occur	 	**PASS**

MERGE PLUGIN:

Sr. No	**Test Case**	**Expected Output**	**Actual Result**	**PASS /FAIL**
1.	When no option selected	A message should occur informing this to the user.	 	**PASS**

2.	When users choose plots which are not adjacent to each other.	User gets the message: “choose plots adjacent to each other Merging is not performed ”	 	**PASS**
3.	When users choose plots which are adjacent to each other.	Merging is performed.	 	**PASS**

LAND RECORD SYSTEM:

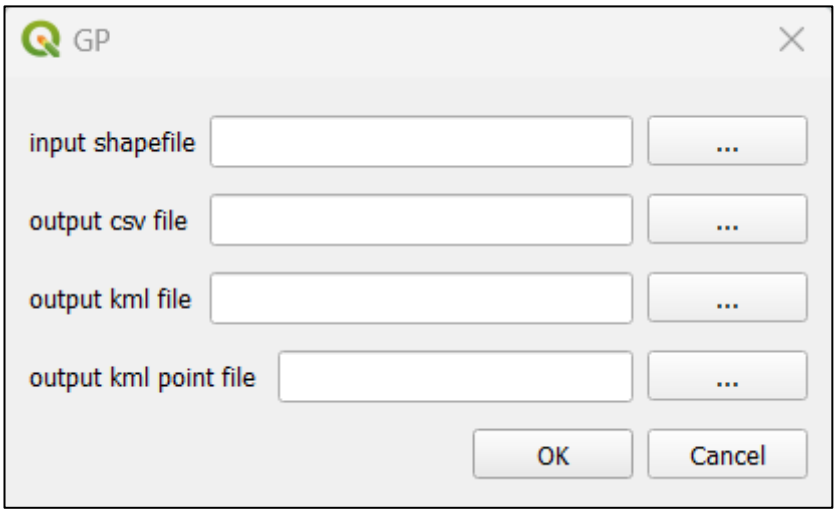
Sr. No	Test Case	Expected Output	Actual Result	PASS /FAIL
1.	If the user writes the wrong username or password.	A message should occur informing that invalid credentials.		PASS
2.	If the user writes wrong captcha.	A message occur informing that this is invalid captcha.		PASS
3.	If the user gives correct input and login.	User will be redirected to the menu page.	 	PASS

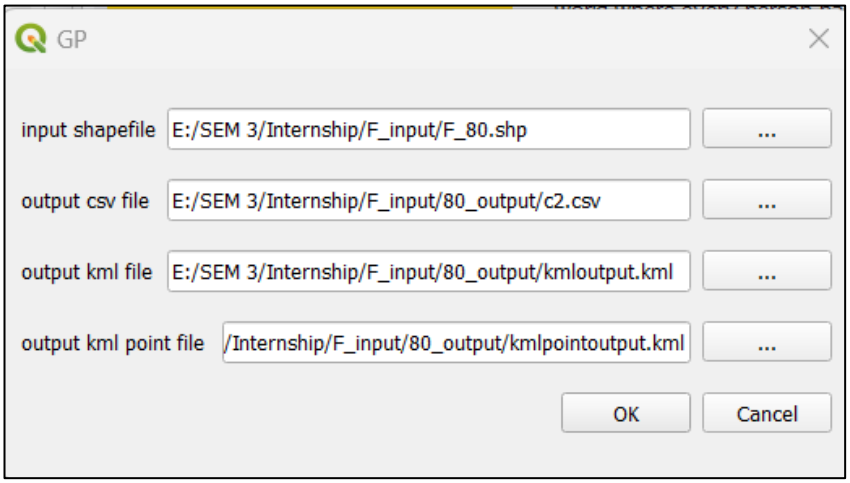
CHAPTER 6-USER INTERFACE

GP:

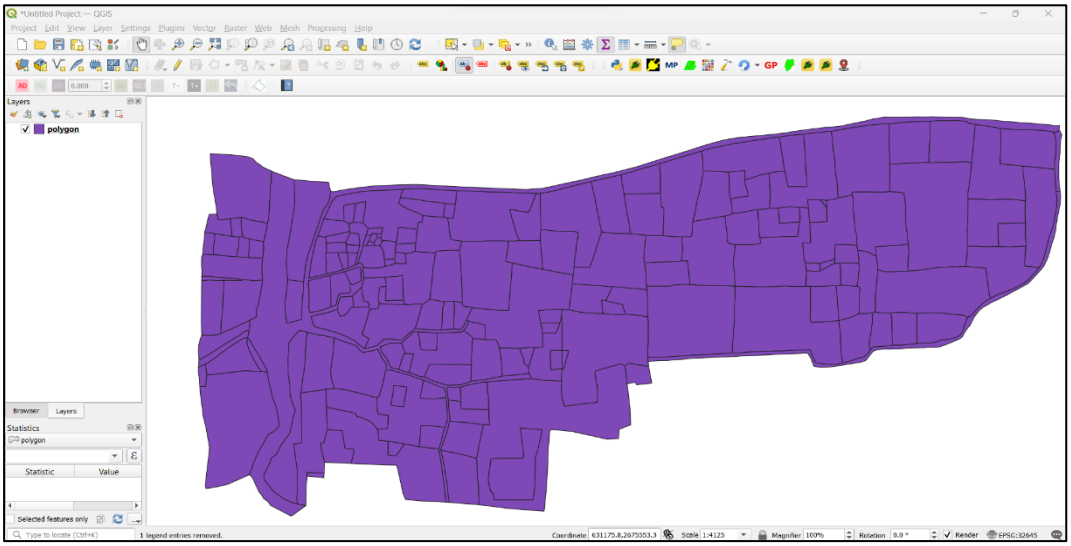
ICON:

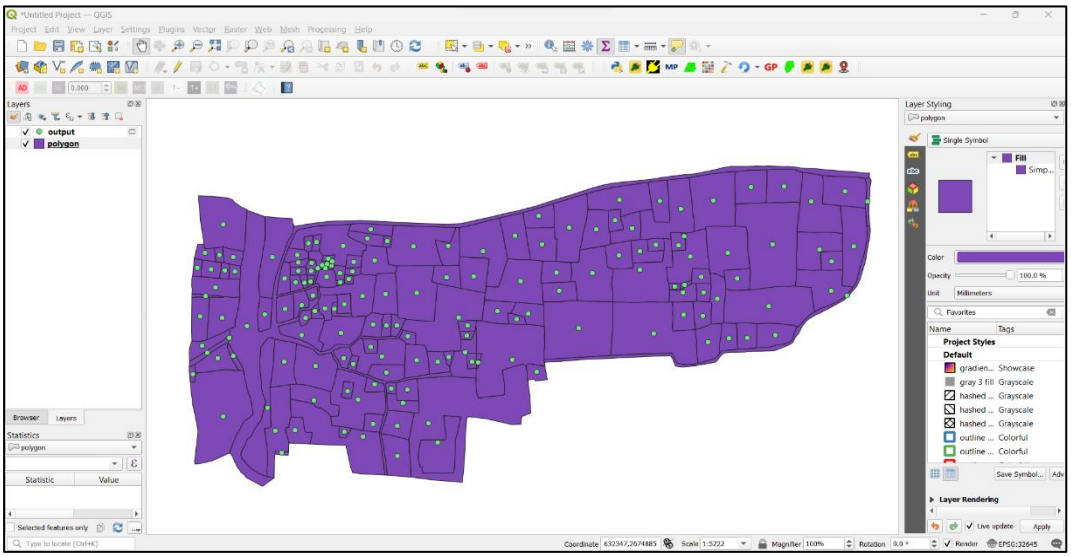
PLUGIN:

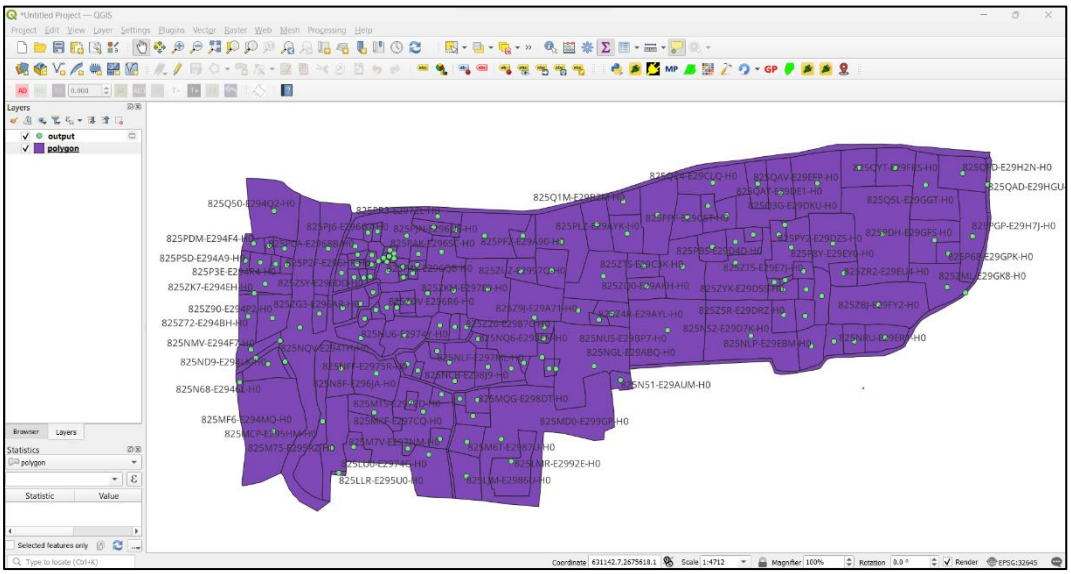


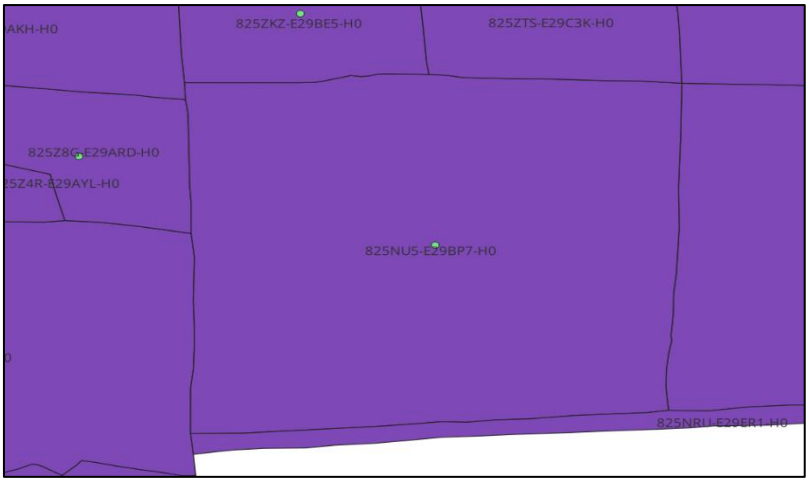


MAP:

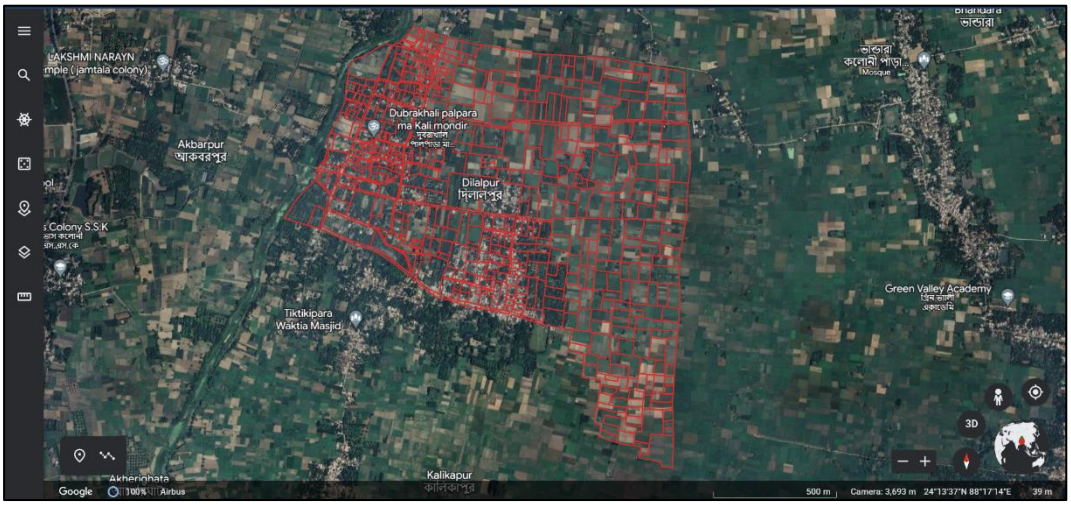

*Untitled Project — QGIS
Layers
polygon
Browser
Layers
Statistics
polygon
Statistic
Value
Selected features only

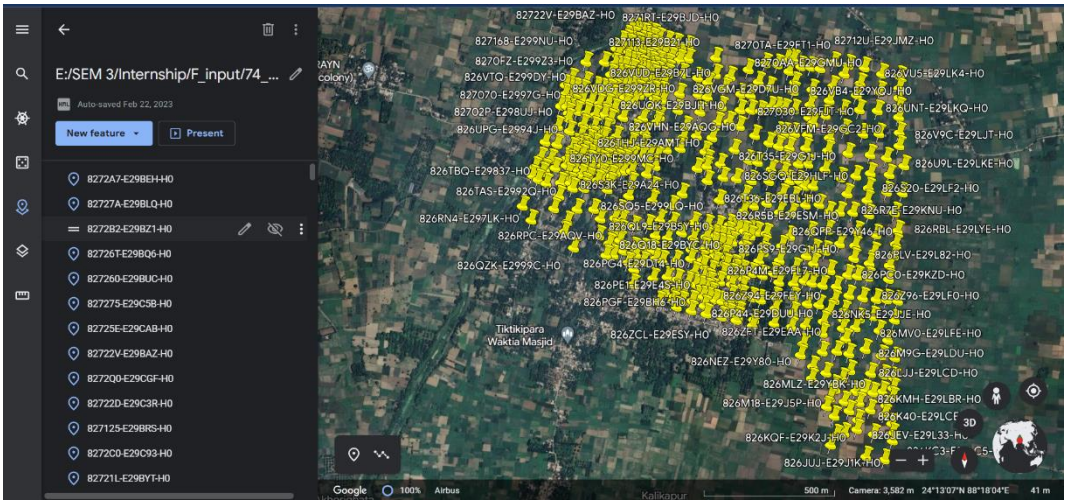

*Untitled Project — QGIS
Layers
output
polygon
Layer Styling
Single Symbol
Fill
Color
Opacity
Unit
Millimeters
Favorites
Name
Tags
Project Styles
Default
Layer Rendering
Browser
Layers
Statistics
polygon
Statistic
Value
Selected features only

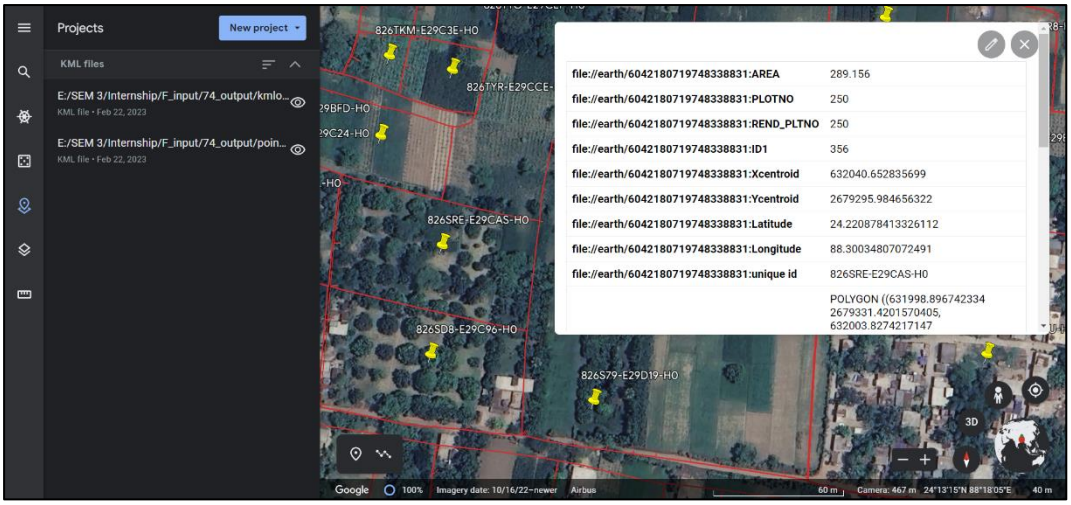

*Untitled Project — QGIS
Layers
output
polygon
Browser
Layers
Statistics
polygon
Statistic
Value
Selected features only



Google Earth:



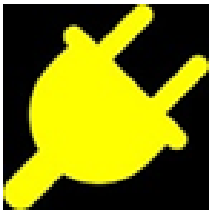




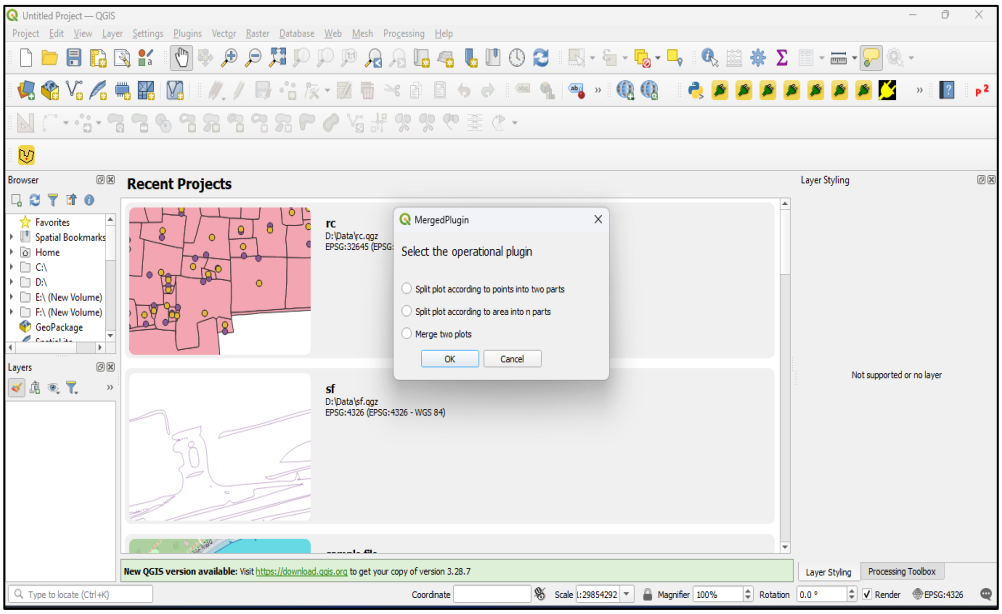
MERGE PLUGIN:

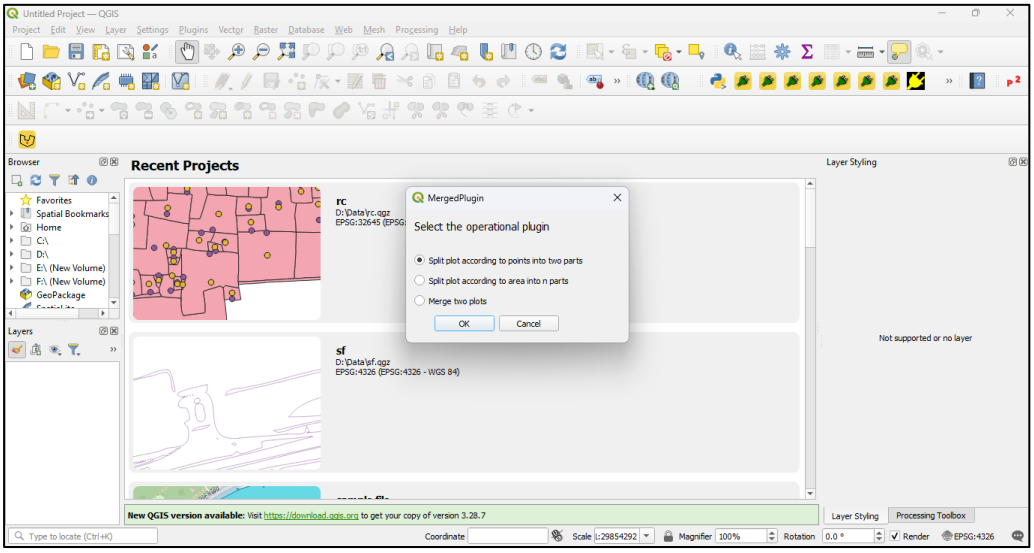
Split Into two

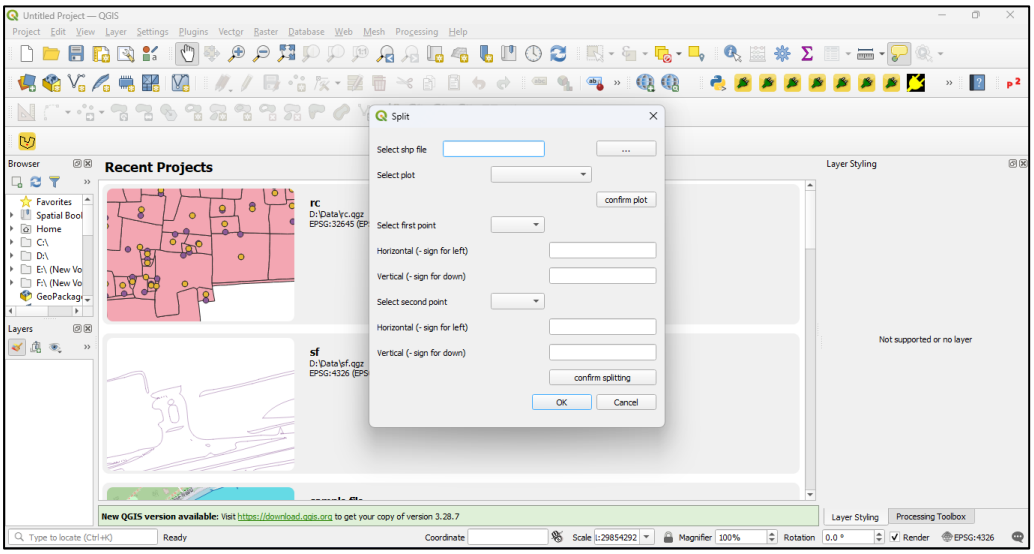
ICON:

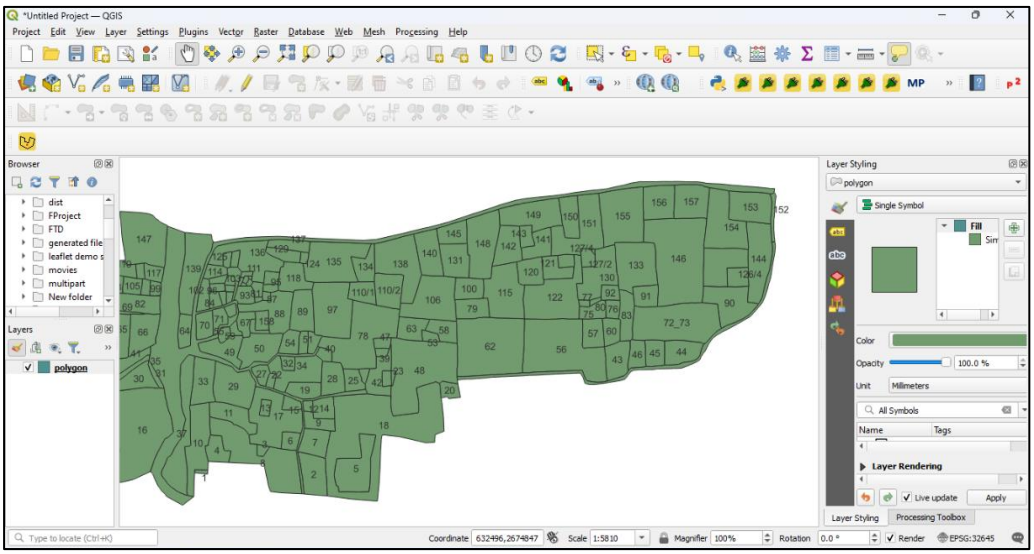


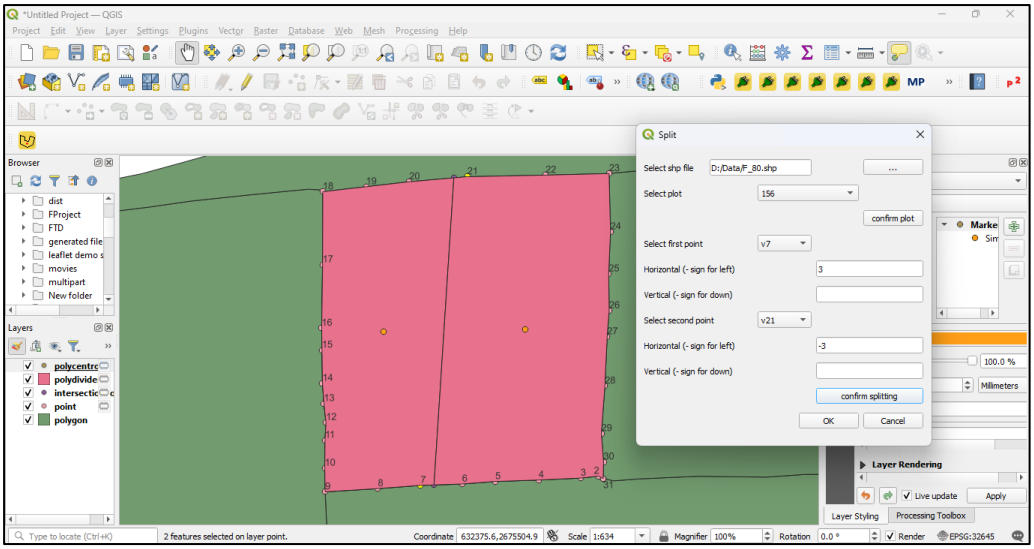
PLUGIN:

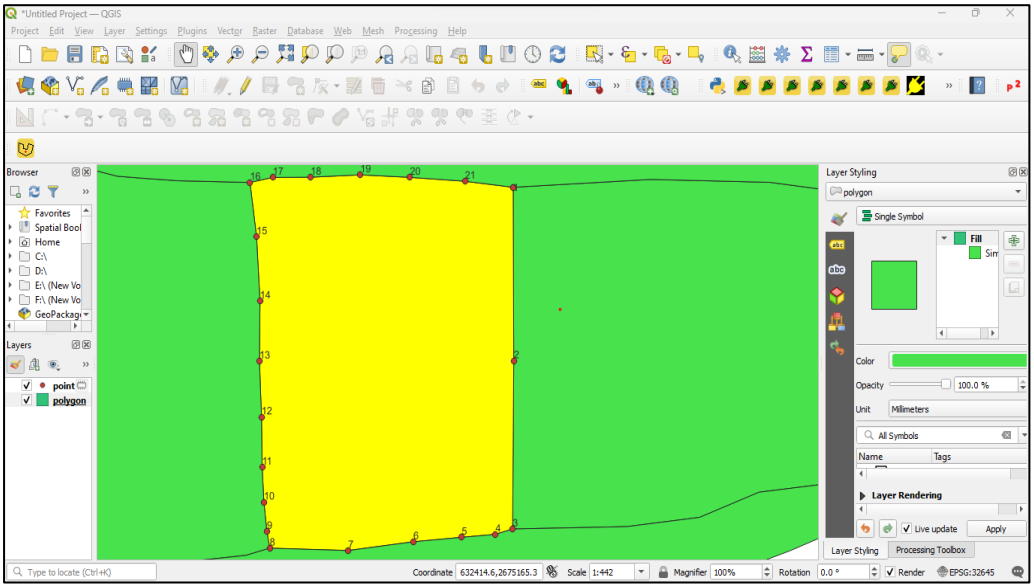


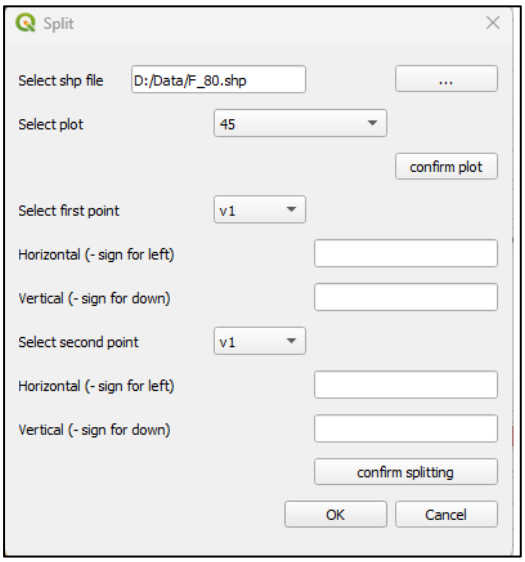

Recent Projects
MergedPlugin
Select the operational plugin
Split plot according to points into two parts
Split plot according to area into n parts
Merge two plots
OK
Cancel
rc
sf
Layer Styling
Not supported or no layer

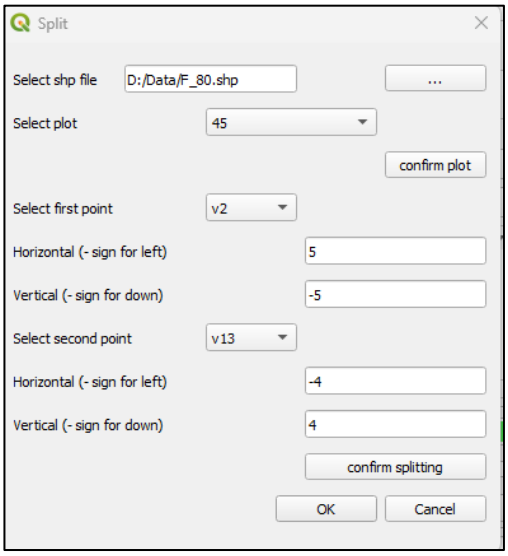

Recent Projects
Split
Select shp file
Select plot
confirm plot
Select first point
Horizontal (- sign for left)
Vertical (- sign for down)
Select second point
Horizontal (- sign for left)
Vertical (- sign for down)
confirm splitting
OK
Cancel
Layer Styling
Not supported or no layer

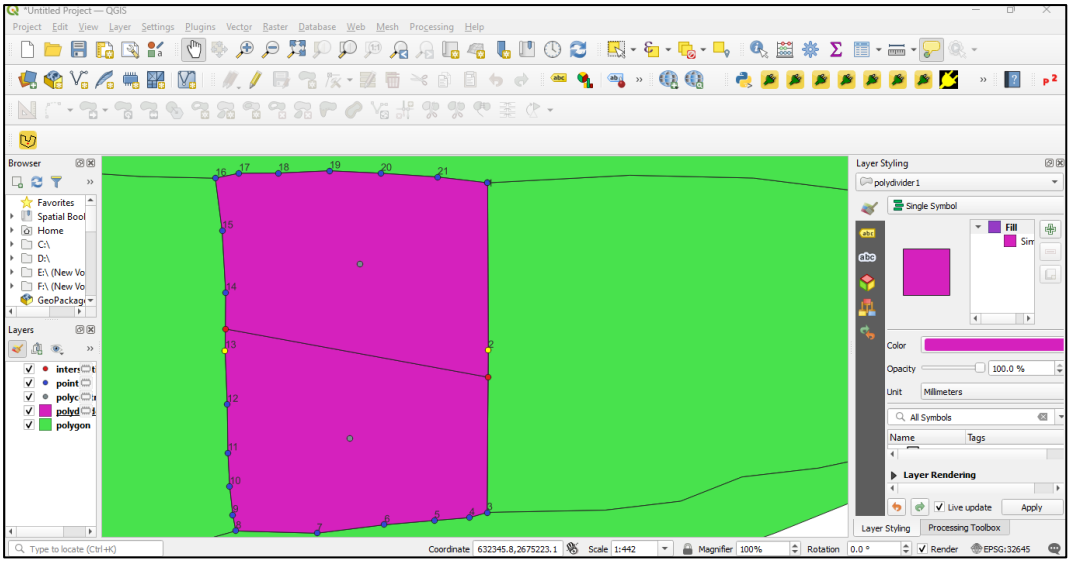

Layer Styling
polygon
Single Symbol

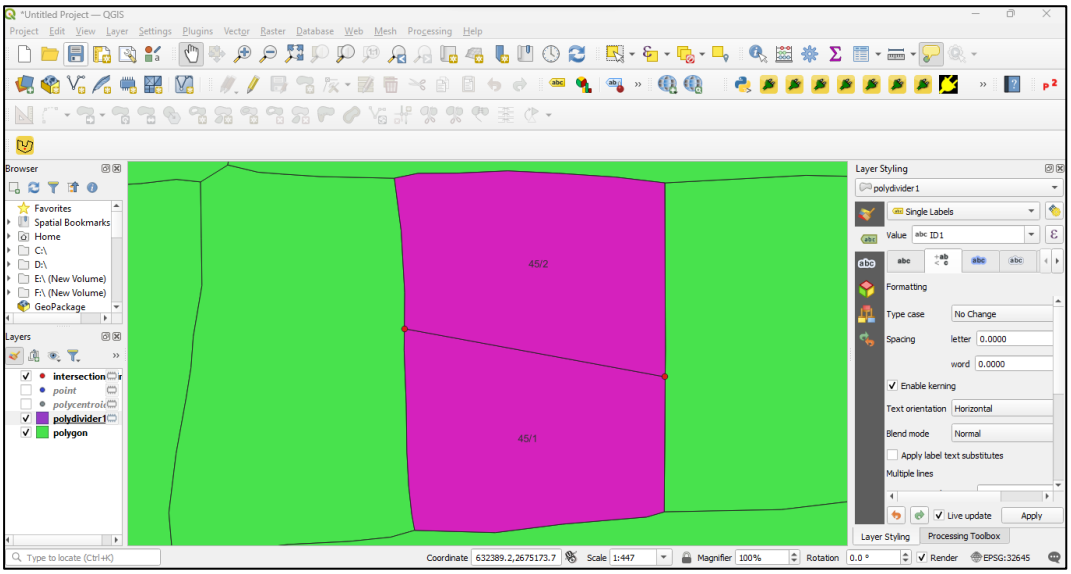

Split
Select shp file
D:/Data/F_80.shp
Select plot
156
confirm plot
Select first point
v7
Horizontal (- sign for left)
3
Vertical (- sign for down)
Select second point
v21
Horizontal (- sign for left)
-3
Vertical (- sign for down)
confirm splitting
OK
Cancel




Split
Select shp file
D:/Data/F_80.shp
Select plot
45
confirm plot
Select first point
v1
Horizontal (- sign for left)
Vertical (- sign for down)
Select second point
v1
Horizontal (- sign for left)
Vertical (- sign for down)
confirm splitting
OK
Cancel

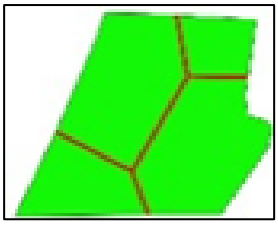

Split
Select shp file
D:/Data/F_80.shp
Select plot
45
confirm plot
Select first point
v2
Horizontal (- sign for left)
5
Vertical (- sign for down)
-5
Select second point
v13
Horizontal (- sign for left)
-4
Vertical (- sign for down)
4
confirm splitting
OK
Cancel



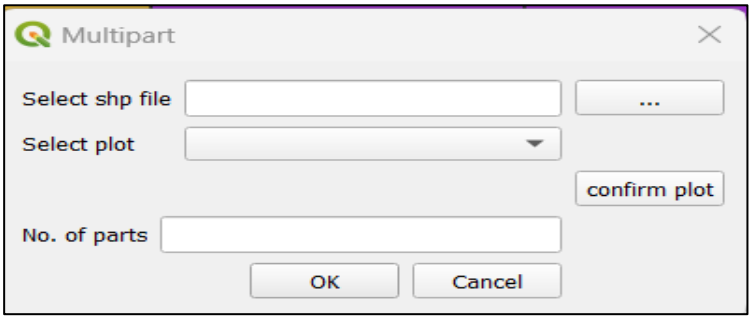


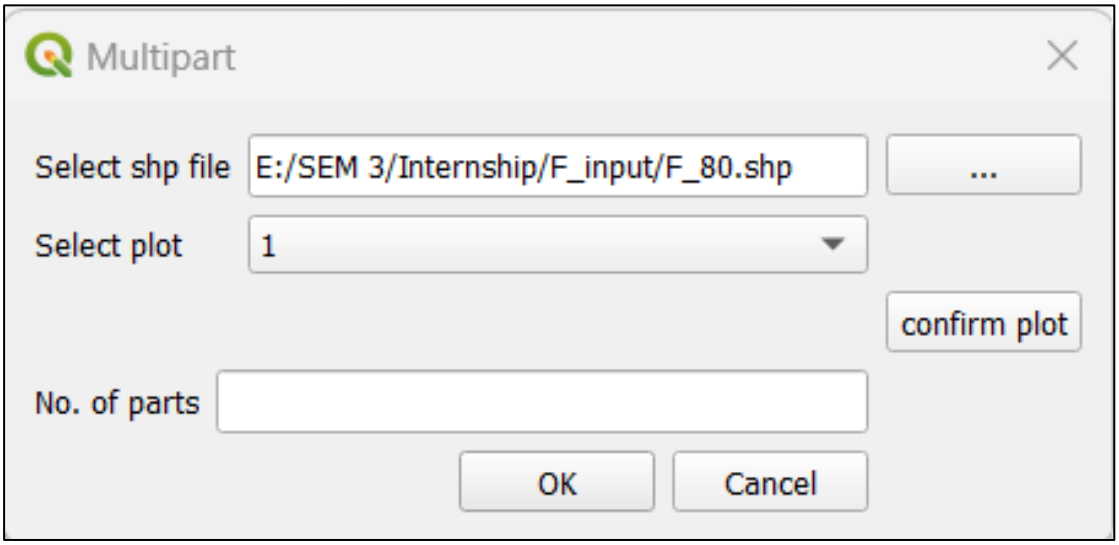
MULTIPART PLUGIN:

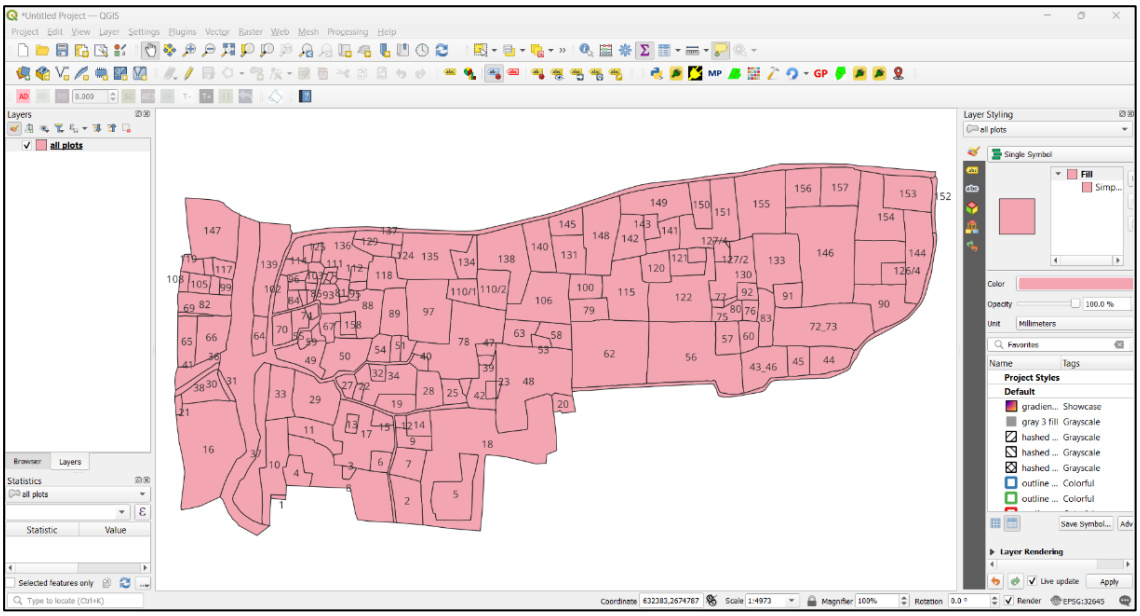
ICON:

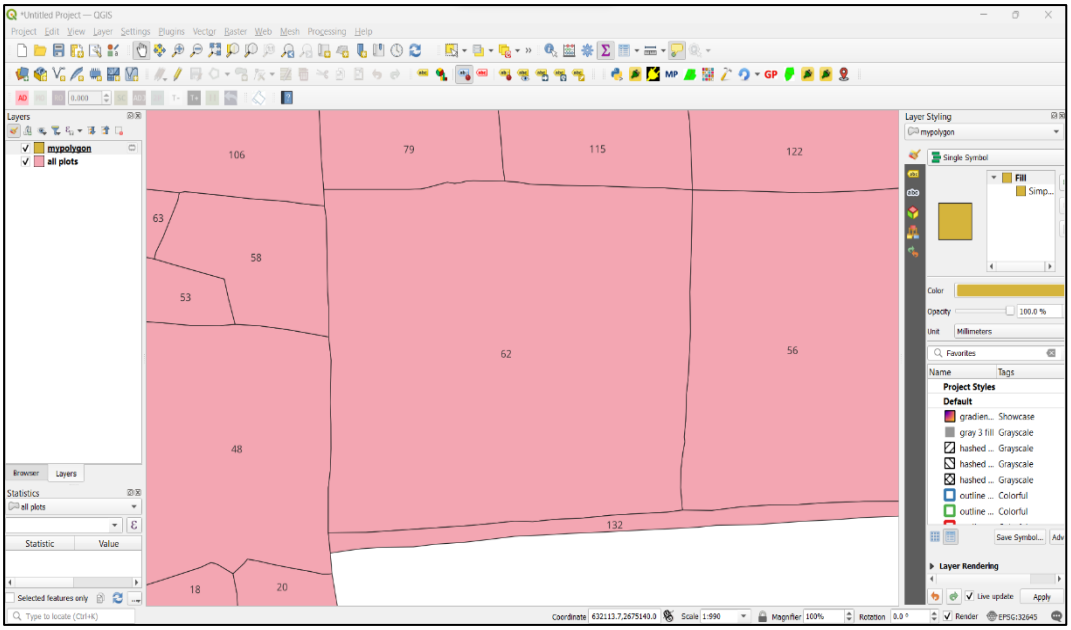


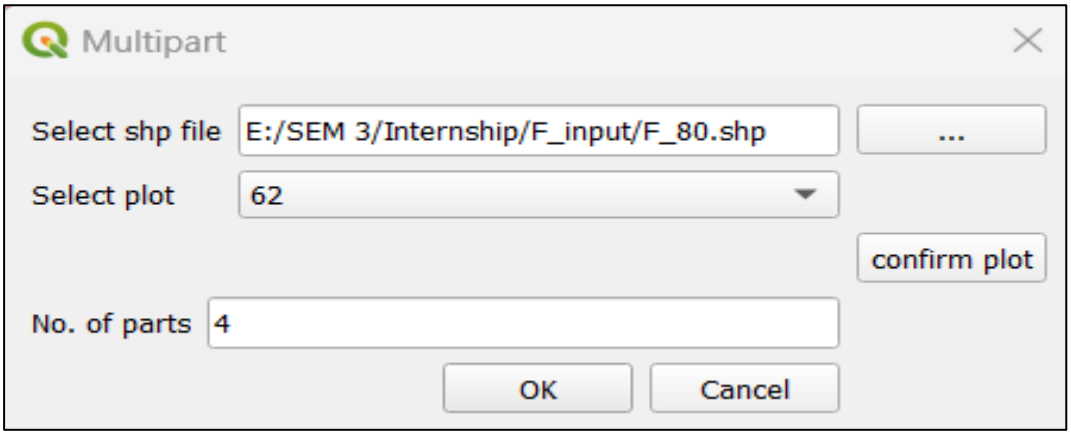
PLUGIN:

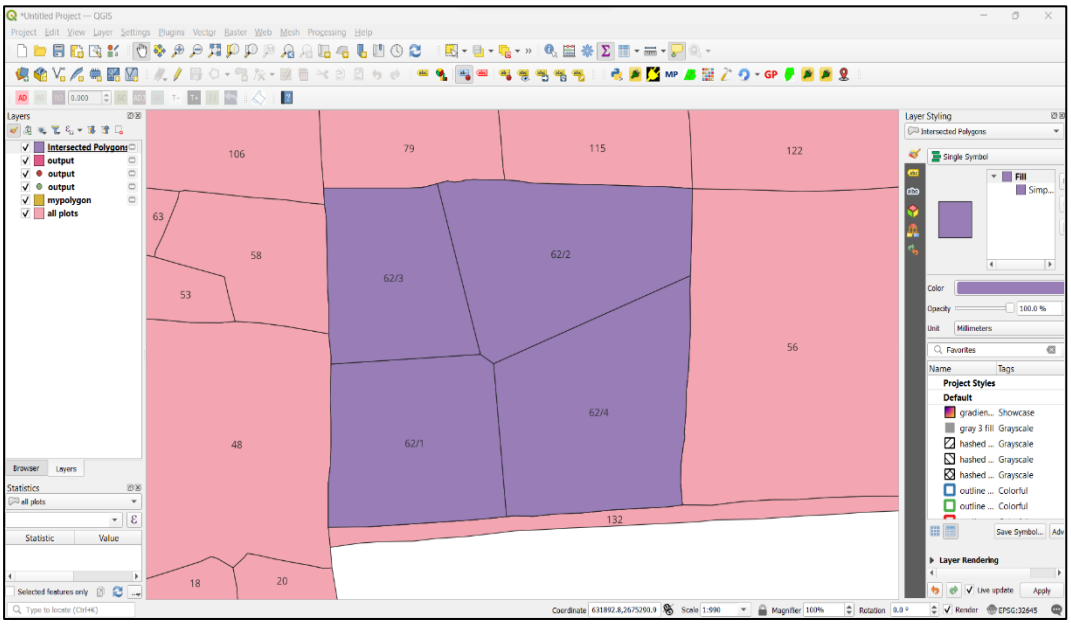

Multipart
Select shp file
...
Select plot
confirm plot
No. of parts
OK
Cancel


Multipart
Select shp file
E:/SEM 3/Internship/F_input/F_80.shp
...
Select plot
1
confirm plot
No. of parts
OK
Cancel




106
79
115
122
63
58
53
62
56
48
132
18
20

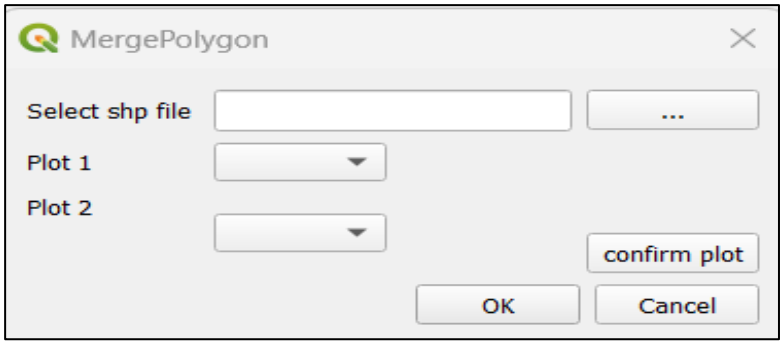

Multipart
Select shp file
E:/SEM 3/Internship/F_input/F_80.shp
...
Select plot
62
confirm plot
No. of parts
4
OK
Cancel

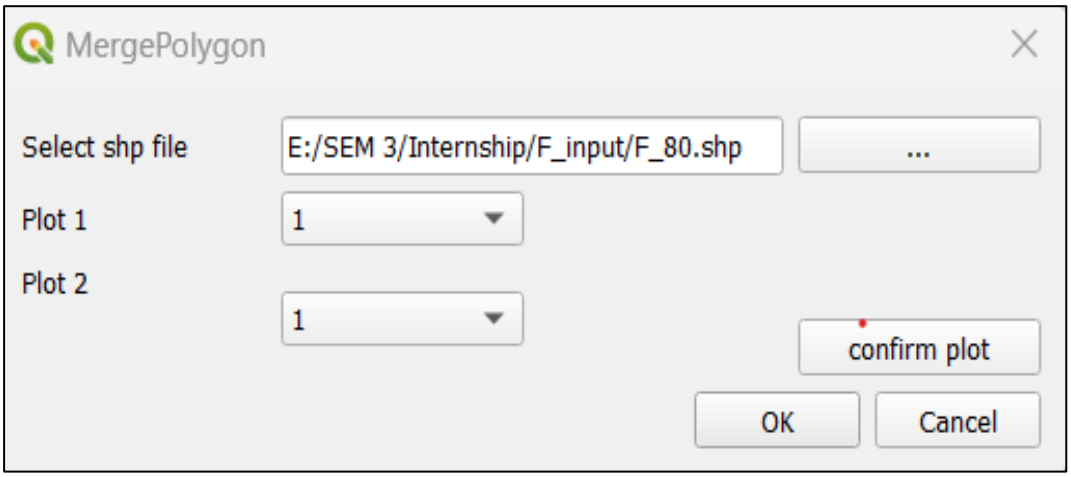

106
79
115
122
63
58
53
62/3
62/2
56
62/4
48
62/1
132
18
20

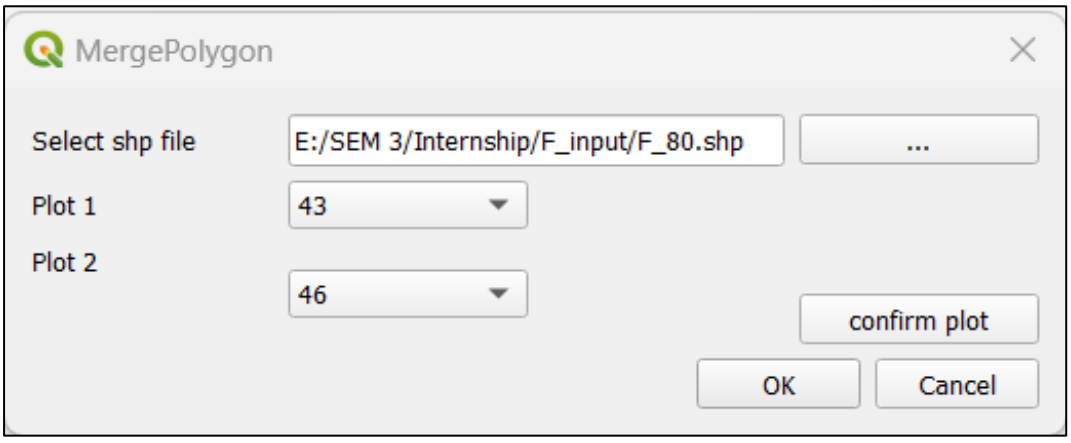
MERGE POLYGON:

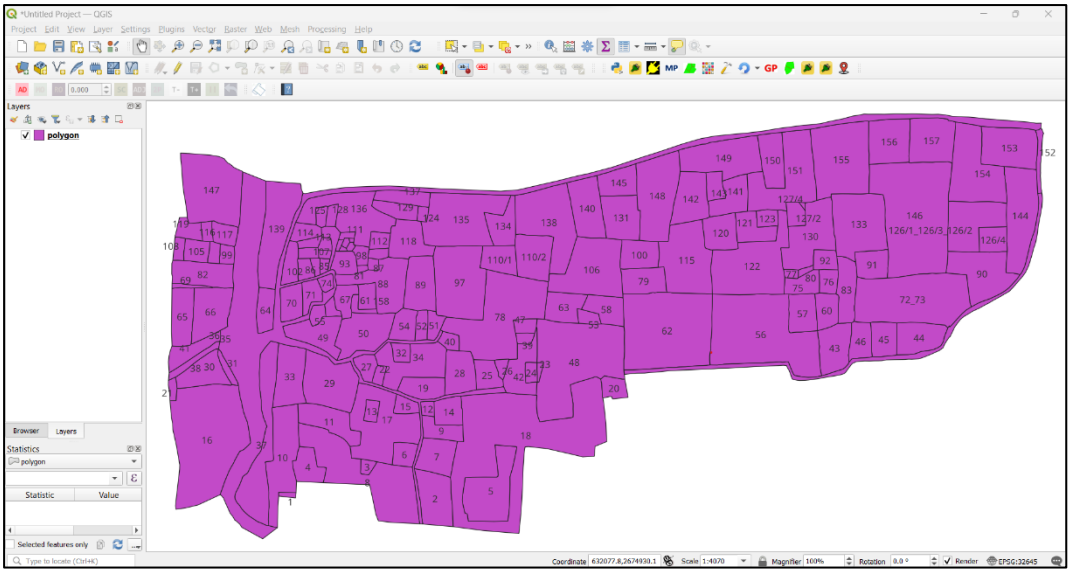
ICON:

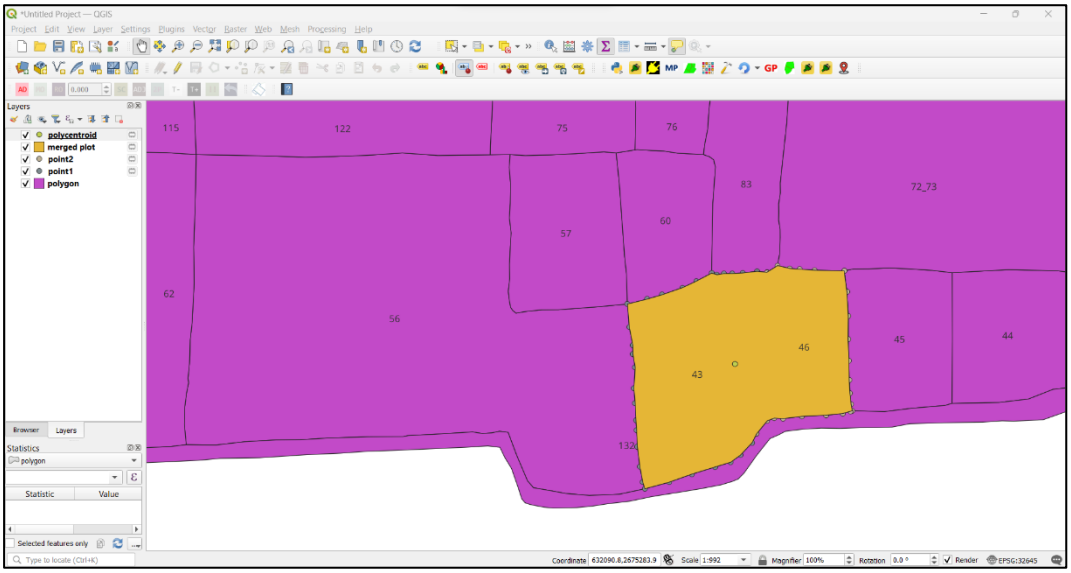
PLUGIN:

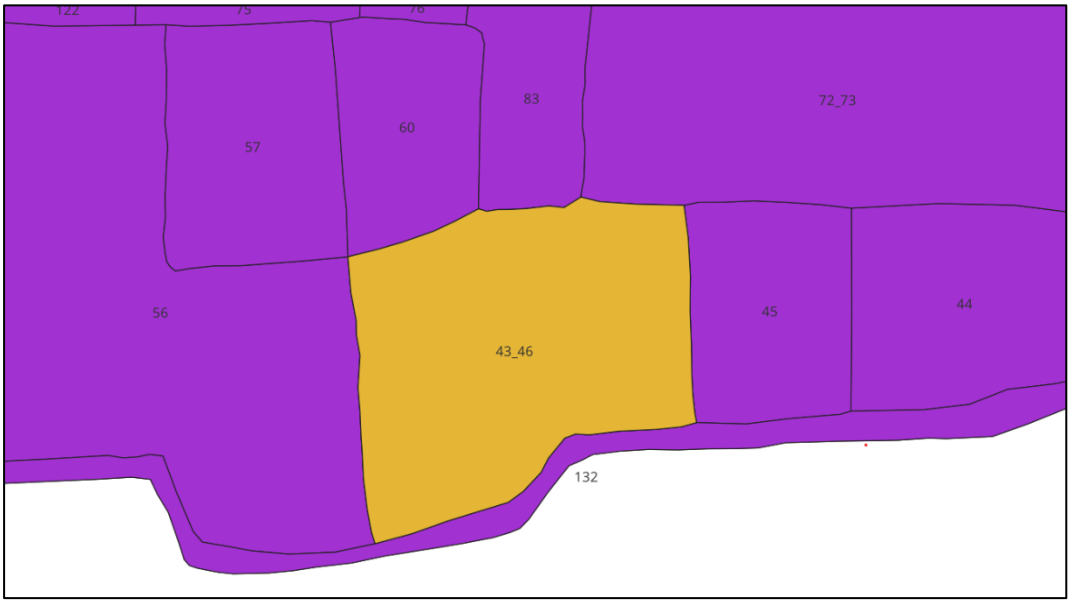







*Untitled Project — QGIS
Layers
polygon
Browser
Layers
Statistics
polygon
Statistic
Value
Selected features only


*Untitled Project — QGIS
Layers
polycentroid
merged plot
point2
point1
polygon
Browser
Layers
Statistics
polygon
Statistic
Value
Selected features only
115
122
75
76
83
72_73
57
60
62
56
46
45
44
43

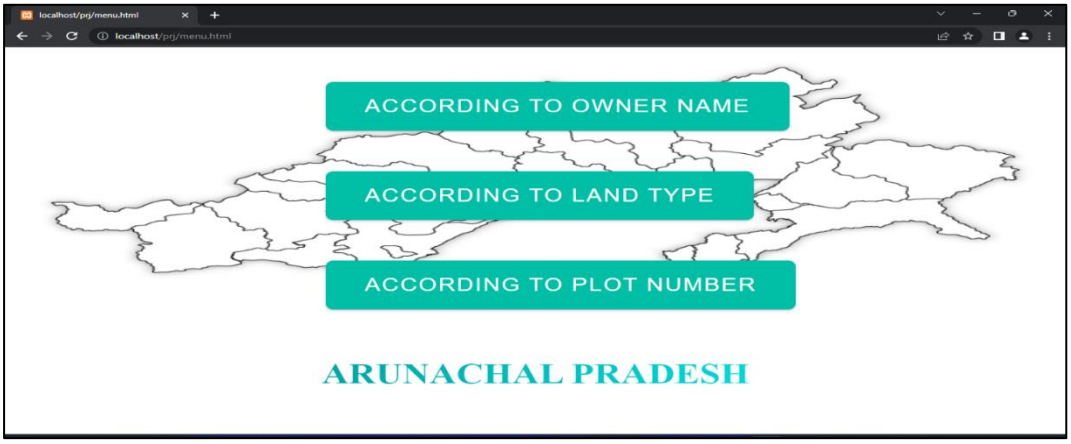

122
75
76
83
72_73
57
60
56
45
44
43_46
132

164	0	43_46	632299.077131...	2675184.31229...	24.1837294773...	88.3025151788...	825NMP-E29EG...	POLYGON ((632...

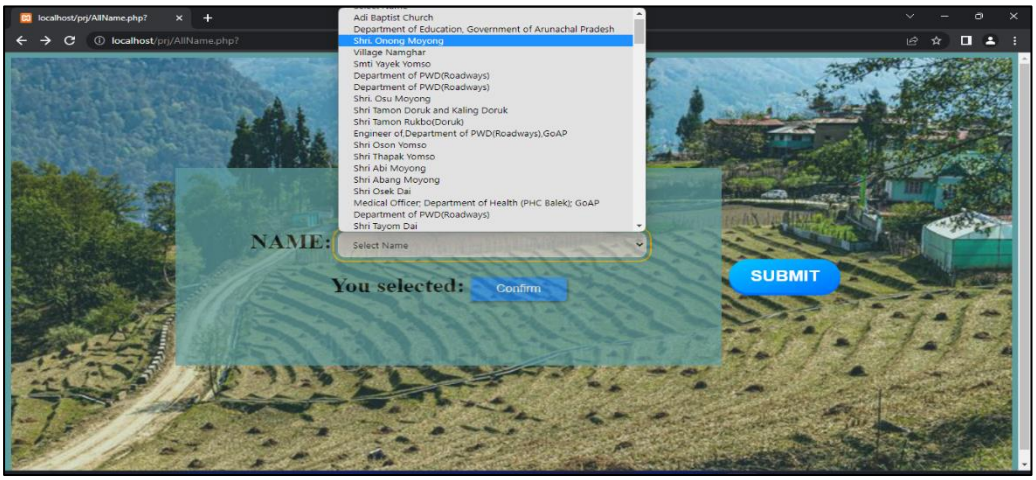
LAND RECORD SYSTEM:

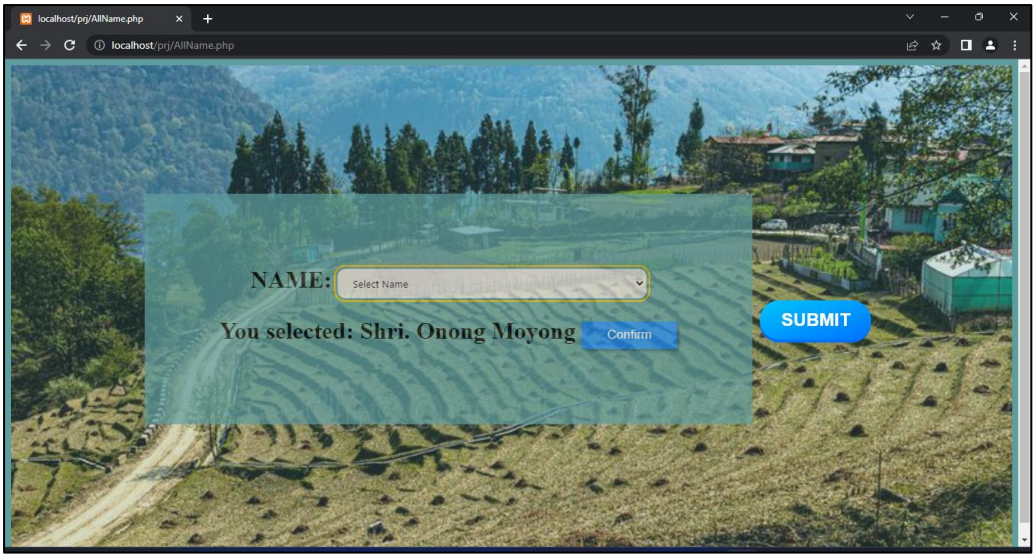
1.Login System

2. Menu

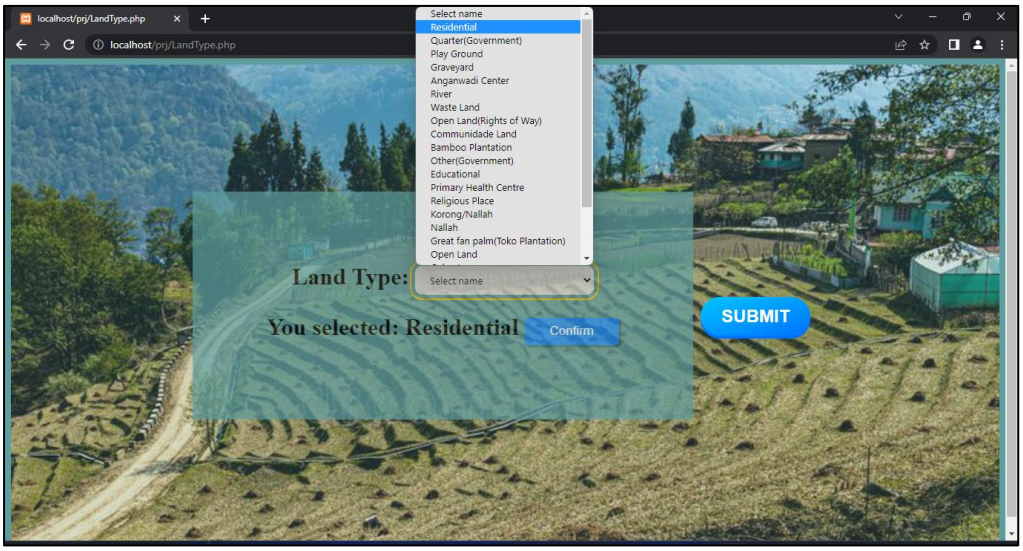


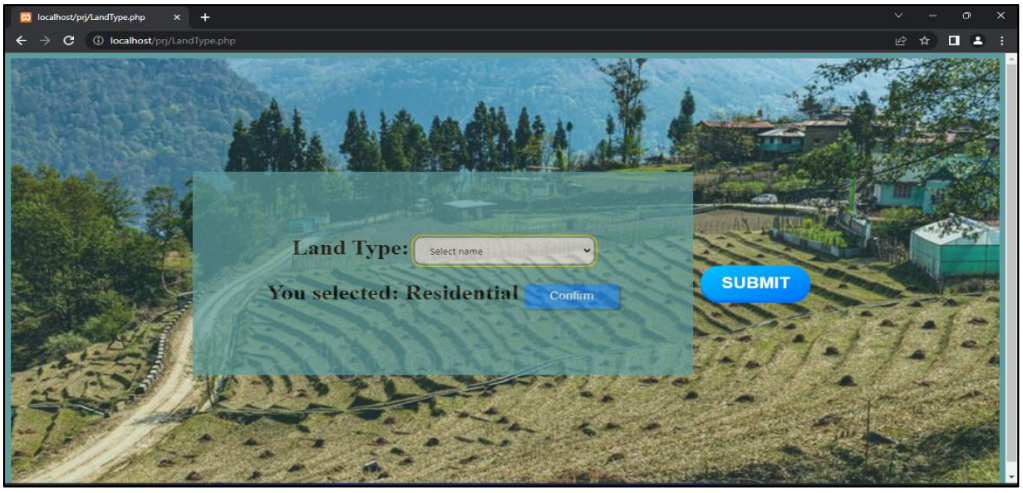
2.1 By owner Name



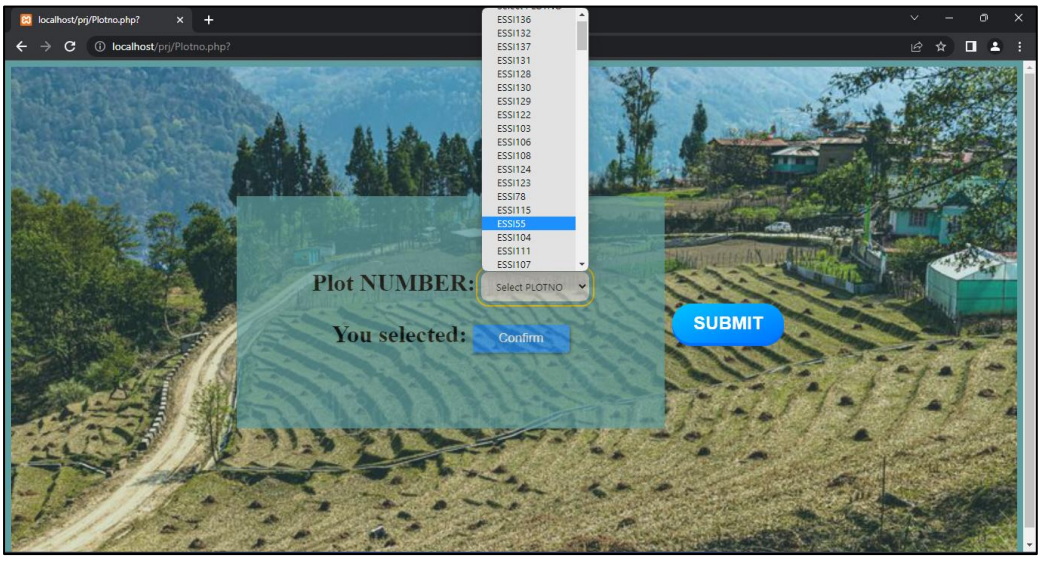


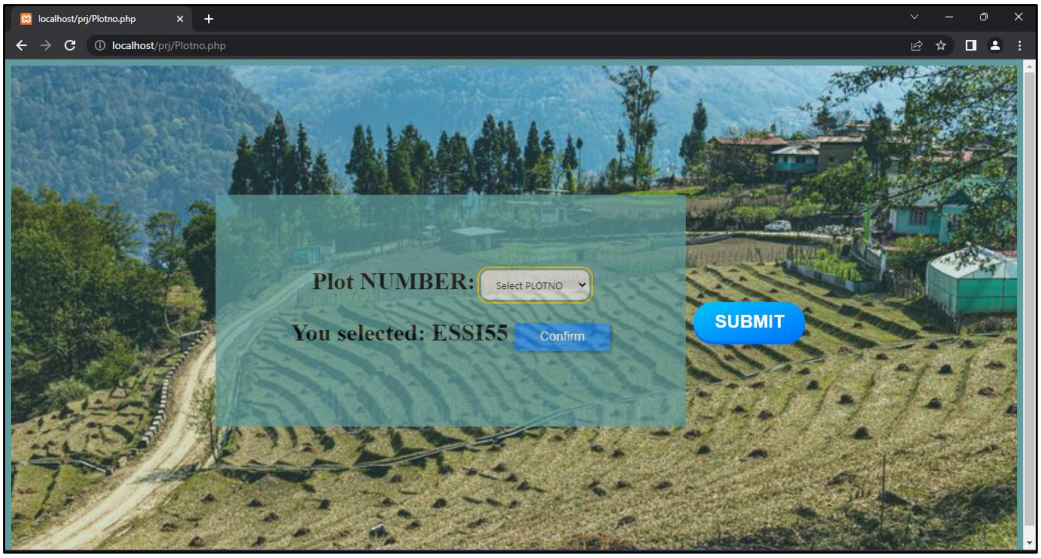
2.2 By Land type



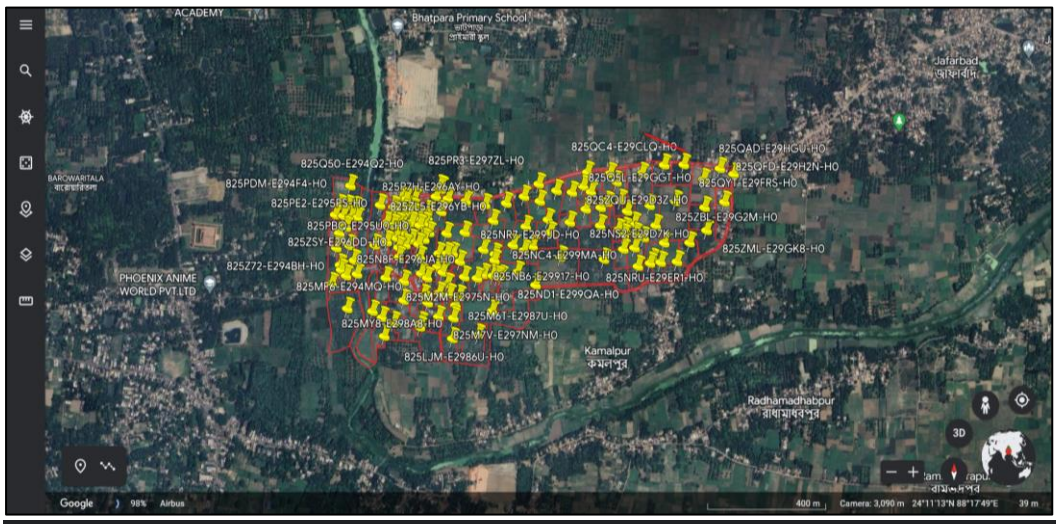


2.3 By Plot Number





Google Earth



CHAPTER 7- APPENDICES

7.1 Glossary

Sr. No.	Term	Description
1	QGIS	Quantum Geographic Information System
2	CSS	Cascading Style Sheets
3	CSV	Comma Separated Values
4	ER	Entity relation
5	ESRI	Environmental Systems Research Institute
6	Gb	Giga Bytes
7	GHz	Giga Hertz
8	GIS	Geographic information system
9	GPU	Graphics processing unit
10	HTML	Hypertext Markup Language
11	IDE	Integrated development environment

13	MySQL	My Structured Query Language
14	NIC	National Informatics Centre
15	PDF	Portable Document Format
16	PHP/.php	Hypertext Preprocessor
17	PyQGIS	Python environment within QGIS
18	RAM	Random Access Memory
21	SHP/.shp	Shape
22	TfLite/.tflite	TensorFlow Lite
24	VS Code	Visual Studio Code
25	QGIS	Quantum Geographic Information System
26	XAMPP	X-operating system, Apache, MySQL, Php, Perl.

CHAPTER 8. REFERENCES

1. Reference From the mentor (Dr. Ganesh Khadanga) research paper Open_30052017_SourceGISPlatformAtGrassRootsLlevelsforAdministrator sandPlanners

2. Sherman G, The PyQGIS Programmer's Guide Extending QGIS 2.x with Python

3. Official website of NIC
   https://www.nic.in/
4. PostGIS Extention of PostGreSQL, http://postgis.refractions.net.
5. https://northlandia.wordpress.com/2015/04/20/connecting-postgis-to-leaflet-using-php/
6. https://docs.qgis.org/testing/en/docs/pyqgis_developer_cookbook/vector.html
7. https://www.luisalucchese.com/post/split-polygons-pyqgis/
8. https://its4land.com/qgis-plugin-to-support-the-interactive-delineation-of-visible-cadastral-boundaries-from-uav-data-available-for-download
9. https://e-nli.org/
10. https://docs.qgis.org/3.22/en/docs/index.html
11. https://docs.qgis.org/3.22/en/docs/user_manual/plugins/plugins.html
12. https://github.com/m-vanshika/SHPFileGenerator
13. https://stackoverflow.com
14. https://www.quora.com
15. https://docs.qgis.org/3.22/en/docs/pyqgis_developer_cookbook/
16. Download XAMPP (apachefriends.org)