

# **Tamil Nadu Water Resources & Information Management System**

Submitted By

B.Venkata Pavani      R170431

Under the Guidance of

**Ch.Ratna Kumari**

Department of Computer Science and Engineering



Rajiv Gandhi University of Knowledge Technologies (RGUKT),  
R.K.Valley , Kadapa , Andhra pradesh.

as a part of  
Partial fulfillment of the degree of Bachelor of  
Technology in Computer Science and Engineering

**Date : 05-05-2023**

## **CERTIFICATE OF PROJECT COMPLETION**

This is to certify that the report entitled “**TamilNadu Water Resources & Information Management System**” submitted by **B.V.Pavani** bearing ID.No. **R170431** and in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out by them under my supervision and guidance.

The report has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

**Project Guide,**

Ch.Ratna Kumari,  
Dean of Academics,  
Dept of CSE,  
RGUKT, RK Valley

**Head of the Department ,**

N.Satyanandaram,  
Assistant Professor,  
Dept of CSE,  
RGUKT, RK Valley

## Declaration

We, hereby declare that this report entitled **“TamilNadu Water Resource & Information Management System”** submitted by us under the guidance and supervision of **Ch.Ratna Kumari** is a bonafied work. We also declare that it has not been submitted previously in part or in full to this university or other university or institution for the award of any degree or diploma.

We will be solenly responsible if any kind of plagiarism is found.

Date:19-09-2022  
Place:RK Valley

R170431,B.V.Pavani

## Acknowledgement

We would like to express our sincere gratitude to **Ch.Ratna Kumari**, Our project guide for valuable suggestions and keen interest throughout the period of project work.

We are grateful to **N.Satyanandaram, HOD OF CSE** for providing excellent computing facilities and a congenial atmosphere for progressing with our project.

At the outset, I would like to thank **Rajiv Gandhi University of Knowledge Technologies, RK Valley** for providing all the necessary resources for the successful completion of our project work.

We express our thanks to all those who contributed for the successful completion of our project work.

With gratitude,

B.V.Pavani    R170431

## **ABSTRACT**

In this work, the proposal is made to implement TNWRIMS(Tamil Nadu Water Resources & Information Management System).We use latest technologies to provide unified visibility into water resources in a single platform.We develop one authoritative system for all water supply,demand and environmental factors,with a vision of making water related data accessible transparently in real-time for different stakeholders through an online GIS/MIS web-portal as well as seamlessly available through mobile,tablets etc.By collecting information of the various water resources in Tamil Nadu,we developed a unified platform which tells details about the water supply and water demand in Tamil Nadu.In our dashboard we will be showing realtime and historical information related to rainfall, reservoirs, water demand , water quality of Tamil Nadu state which will be very useful for the government to take appropriate decisions in the critical situations.

# Table of Contents

<b>1.Introduction.....</b>	<b>7-8</b>
1.1 Introduction to the project	
1.2 Importance of the project	
1.3 Collecting Water Resources data	
<b>2.UML Diagrams.....</b>	<b>9-10</b>
2.1 Use case Diagram	
2.2 Sequence Diagram	
<b>3.Proposed Method and Flow of the Project.....</b>	<b>11-13</b>
3.1 Proposed Method	
3.2 Flow of the project	
3.3 Advantages and Disadvantages	
3.3.1 Advantages	
3.3.2 Disadvantages	
<b>4.Implementation.....</b>	<b>14-18</b>
4.1 Project Structure	
4.2 Aggregations	
4.3 Flow of API	
4.4 MITANK Module	
<b>5.Tools and Technologies.....</b>	<b>19-22</b>
5.1 Hardware Components	
5.2 Software Components	
5.3 Flask server	
5.4 Spring Framework	
5.5 Maven	
5.6 Kafka and Storm	
5.7 Databases	
<b>5.Output.....</b>	<b>23-25</b>
<b>6.Test Case Report.....</b>	<b>26</b>
<b>6. Conclusion.....</b>	<b>27</b>
<b>7.References.....</b>	<b>28</b>

# INTRODUCTION

## 1.1 Introduction to project

Water is precious natural resource for sustaining life and environment. Effective and sustainable management of water resources is vital for ensuring sustainable development. considering its increasing scarcity, the planning and management of water resource become most important. In our dashboard we will be showing realtime and historical information related to rainfall, reservoirs, MI tank, river gauge, ground water, water demand, water quality, evapotranspiration of Tamil Nadu state which will be very useful for the government to take appropriate decisions in the critical situations. It also contains content management and asset management where they can edit any information related to any component. It contains system monitoring which contains information about the working of api's in our dashboard and also user analytics which tells how many people are visiting our dashboard.

TNWRIMS means Tamil Nadu Water Resources & Information Management System. Tamil Nadu is a riverine state with major, minor and medium rivers. Adyar river, Amaravathi river, Bhavani river, Ambuliyar river are major rivers in Tamil Nadu. There are lot of reservoirs in Tamil Nadu, so Tamil Nadu government has aimed to make Tamil Nadu a drought proof state by providing assured water to all sectors in the state, towards this journey it is decided to provide near real time visibility of all water resources in the state and developed Tamil Nadu Water Resources Information and Management System. The TNWRIMS of the Andhra Pradesh Water Resources Department (APWRD) works in collaboration with National Remote Sensing Centre (NRSC) for Indian Space Research Organisation (ISRO) and Vassar Labs.

In TNWRIMS project mainly we have to develop a website for water management in Tamil Nadu. For Frontend mainly we use Angular Technology. For Backend we mainly use Java, Spring Boot, Play server, Maven, Kafka, Storm.

## 1.2 Importance of the Project

Water is having very important role in our life. But nowadays we are noticing scarcity of it. So we need to manage and use water resource effectively. In our project we will be showing the real-time information about water sources.

Here we will be having two things :-

**A. Data Integration and management system:** Here we will be collecting hydrological and meteorological data in different formats from different sources. Hydrological and Meteorological data: weather data, reservoir data, rivers, pump houses, ground water level, soil moisture, evaporation, surface water. Sources: web scrapers, external API's integration, mobile app data entry, manual file uploads, manual file uploads, automatic weather stations, automatic rain gauge, satellite data from external agencies(NRSC, IMD, ECMWF)

**B. Smart Decision System:** Helps or advices the authorities in their decision making process.

## 1.3 Collecting Water Resources Data

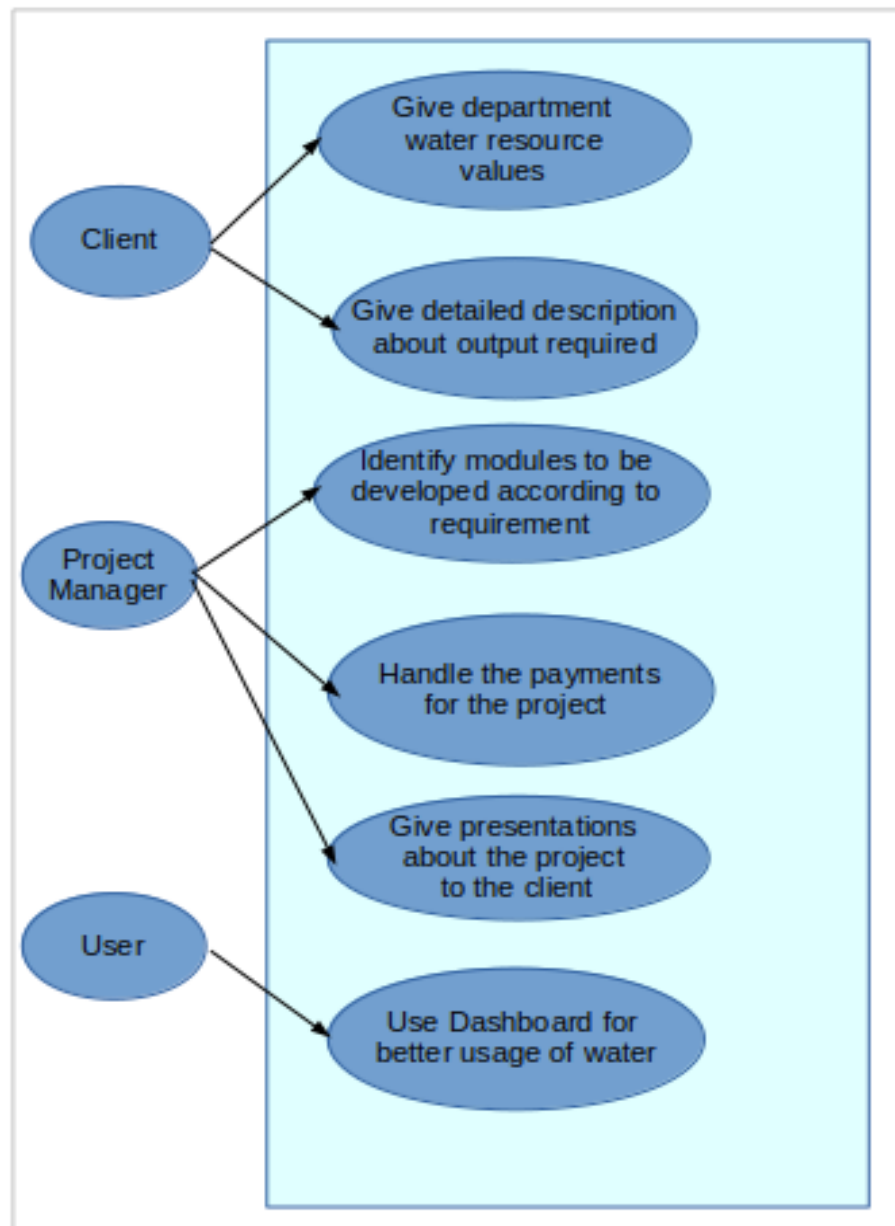
Mainly the water resources data is collected in the form of six ways:-

- 1) Reservoirs
- 2) MI Tanks
- 3) Rainfall
- 4) Soil Moisture
- 5) Lift Irrigation Schemes
- 6) Water Conservation Structures

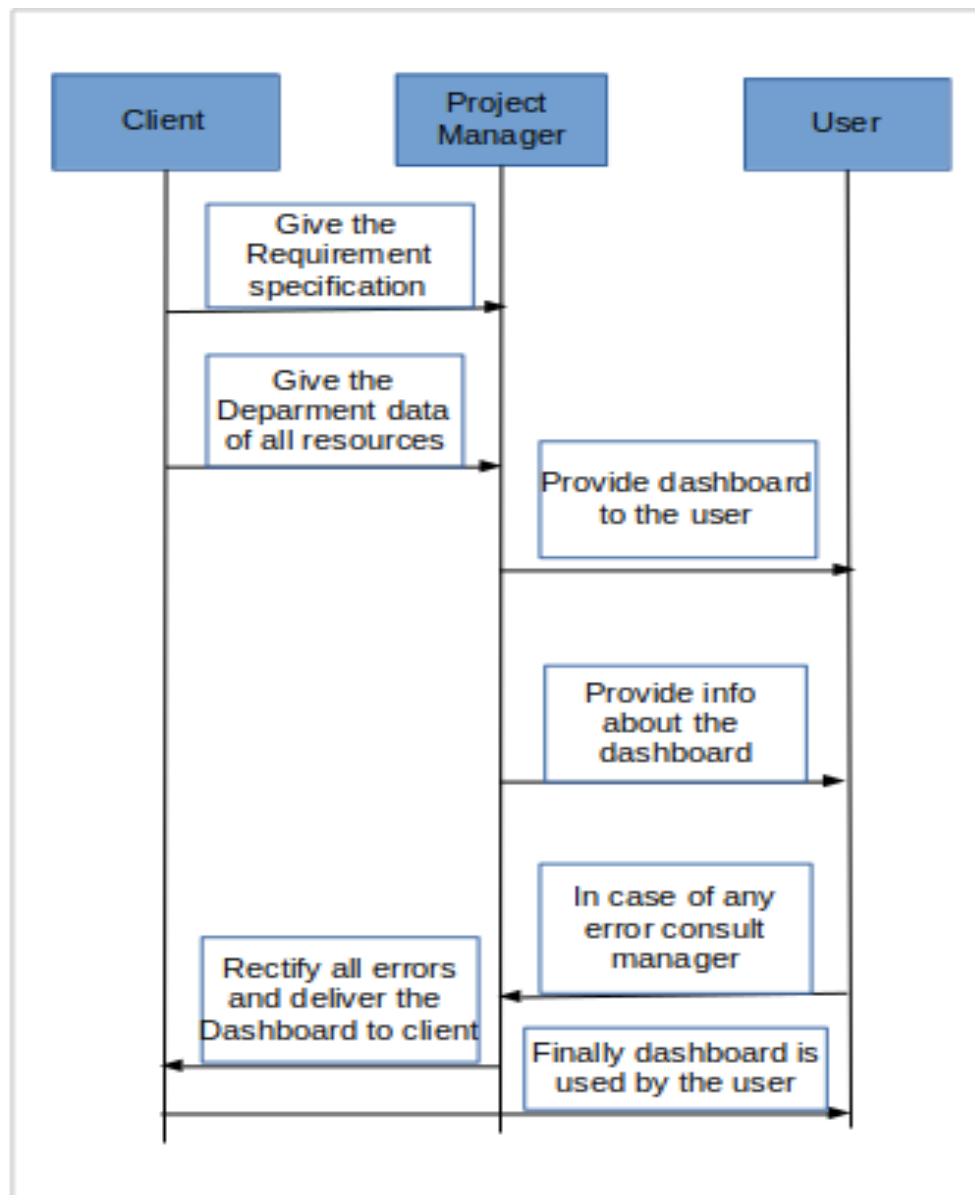


## UML DIAGRAMS :-

### 2.1 Use case Diagrams :-



## 2.2 Sequence Diagram :-



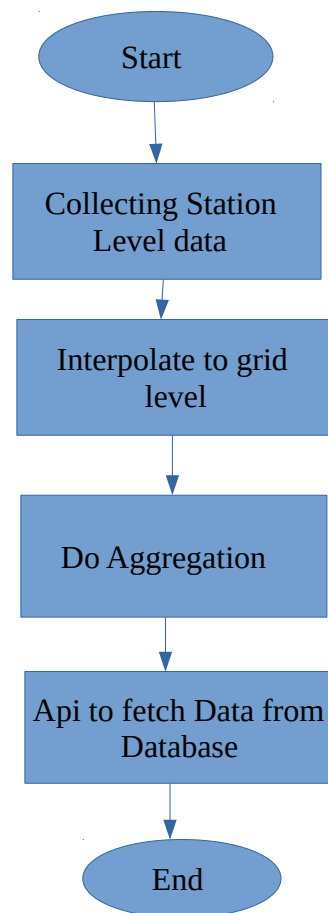
## **Proposed Method and Flow of the Project**

### **3.1 Proposed Method**

There are various stakeholders that utilize, track, and manage water based on their capabilities and requirements in the state. This in turn leads to the different authorities working independently which may sometimes end up in conflicting situations in the time of crisis. Moreover, effective collaboration and process alignment among different authorities is difficult to achieve due to the operational inefficiencies arising out of fragmented IT systems. Here we developed one authoritative system for all water supply, demand and environmental factors, with a vision of making water related data accessible transparently in real-time for different stakeholders through an online GIS/MIS web-portal as well as seamlessly available through mobile, tablets etc.

A single, unified and an authoritative platform for comprehensive, authoritative and consistent data and information of water resources for the state Tamil Nadu time visibility of water resources.

### 3.2 Flow of the Project



- First we will be collecting station level data from different sources
- Now the collected data is station level we will interpolate it to grid level
- Then grid level data is aggregated to village, block, district and state
- This all data will be stored in the database.
- According to the UI requirement we will write API's to access the data from the database and expose to the UI.

### **3.3 Advantages and disadvantages**

#### **3.3.1 Advantages**

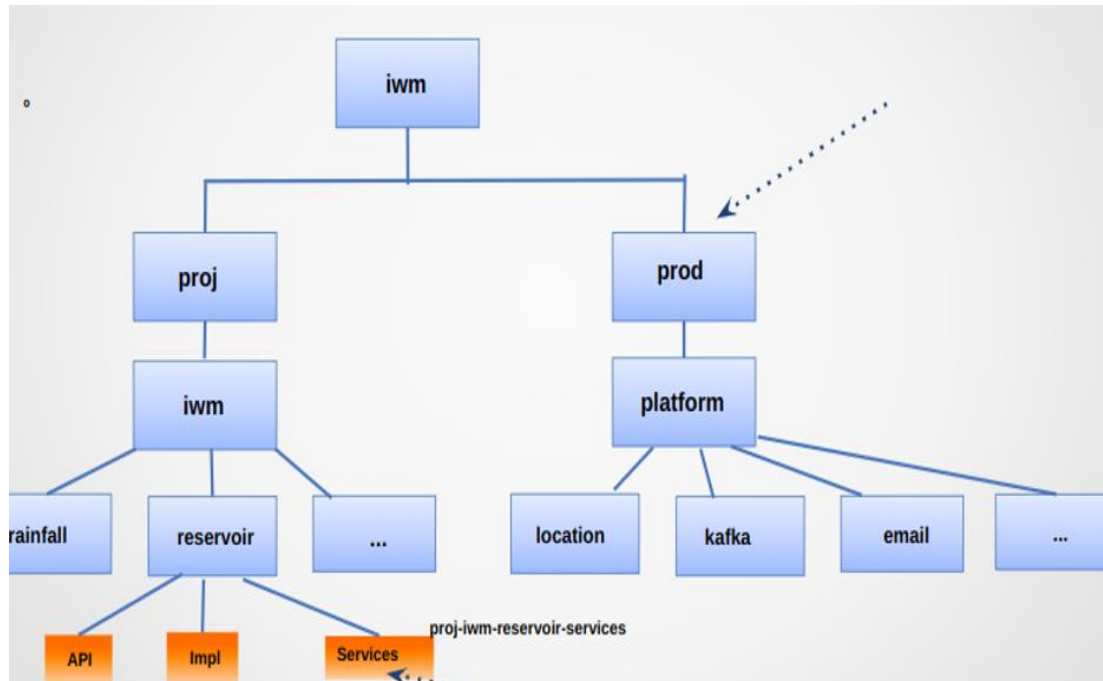
- Reduce the wastage of water.
- Helps in taking critical decisions.
- Officials can easily monitor the information of water resource for the state of Tamil Nadu.
- One authoritative system for all water resources related information
- Helps key decision makers for effective planning of water resource management

#### **3.3.2 Disadvantages**

- Sometimes server becomes slow due to heavy processing
- In User Analytics we are using GA4 free version, it has some limitations of getting user information. That is it can track only up to 10000 thousand page views
- For some modules in the dashboard we will be scraping from external sites. We can scrape and store the data successfully only when they following a consistent format.

# Implementation

## 4.1 Project Structure



- We will be having IWM repository.
- In IWM we will having proj and prod.
- Prod contains all the code required to connect with the different platforms.
- Proj contains the actual code.
- In proj we will have different modules like rainfall, reservoir, mitanks, ground water which contains all the code related to that particular module.
- Each internal module contains api, impl, services, scrappers.
- Api contains all the interfaces required for that particular module.
- Impl contains classes with getter and setter methods.
- Service contains actual logic of the api's, aggregation, interpolation.
- Some modules scrape data from external sites. In that case we will have scrappers where it has the code to connect to the external site, scrape the data and store it in the database.

## 4.2 Aggregations

There are two types of aggregations.

- Temporal aggregation: from realtime to hourly, daily, monthly, seasonally, yearly.
- Spatial aggregation: from grid level data to village, block, district, state.
- For these aggregations we will use simple aggregation or weighted aggregation based on our case.
- For example: if we are showing the no of rainfall stations in the specific block, then we just add the no of stations present in each village present in that particular block.
- If we want to show the rainfall of particular village, then we will be doing weighted average depending on the intersection area of the grid to the village.
- This aggregation logic will be present in the service module of its specific module.

## 4.3 Flow of API

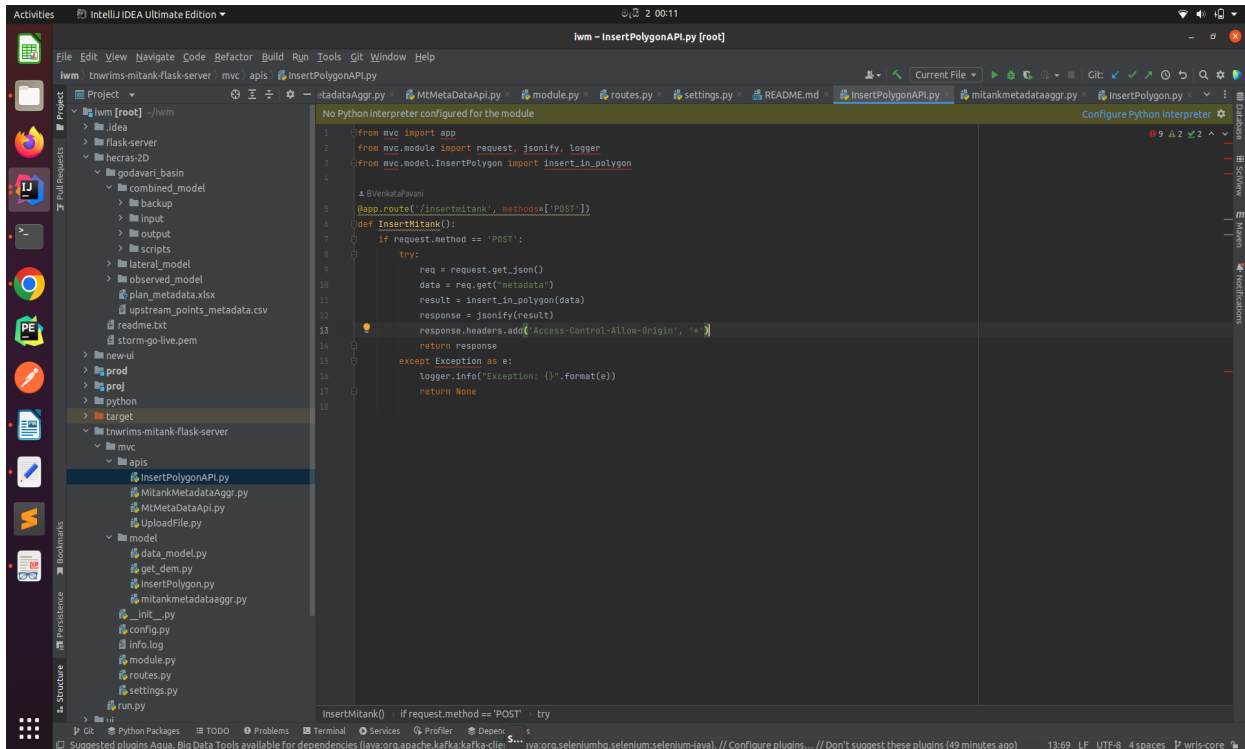
- All the api details will be present in the routes file which is present in conf folder.
- It also gives information about whether it is a get method or post method.
- the HTTP POST method is used to send data to a server to create or update a resource.
- the HTTP GET method is used to request data from a specified resource and should have no other effect.
- It also gives information about in which controller the implementation of the specific api is present.
- So when we hit that api it goes to that particular method in that particular controller.
- If it contains only simple logic then it will directly present its logic in that.
- If it has complex logic then its logic will be present in the service class of the specific module.

## 4.4 MITANK Module

- Mi tank module will tell about the no of mi tanks present in each block respectively and about the capacity,storage,water spread and fill% of each mitank at each level.
- Mitank level information is shown in this module because for present we have block level information.
- Parameters of the mitank like storage,capacity,fill%,water spread and no of tanks will be shown in dashboard.
- Storage,capcity,fill% and water spread are calculated from the satellite data every time,beacuse there will be no standard capacity for the mitank it will vary over time.
- Here we aggregate mitank level information to the block level,district level and state level respectively.
- We get mainly mi tank information from SENTINEL 1 and SENTINEL\_DEM sources for all districts.
- In Content Management module,we upload the file which is the bulk upload of the mitank level information.
- In Asset Management,we use existing mitank information for editing the fields of the mitanks in the production Database.
- For mitanks,we use postgres and cassandra for storing computed and aggregated data,
- We use flask server to run mitank module.
- We also use datamodels for effective connection with posgres Database.
- Mainly we use 2 tables in postgres:-
  - tnwrims\_mitank\_polygon\_layer
  - tnwrims\_mitank\_metadata\_aggr
- Mainly we have 4 apis in mitank module :-
  - InsertPolygonApi :- It is used to insert the new mitank information in the postgres and also to update the existing mitank information.
  - MitankMetadataAggrApi :- It is used for the aggregating mitank level information to block level,district level and state level.
  - MtMetadataApi :- It is used to fetch the mitank information from postgres.
  - UploadFileApi :-It is used to upload the new mitank file from frontend side.

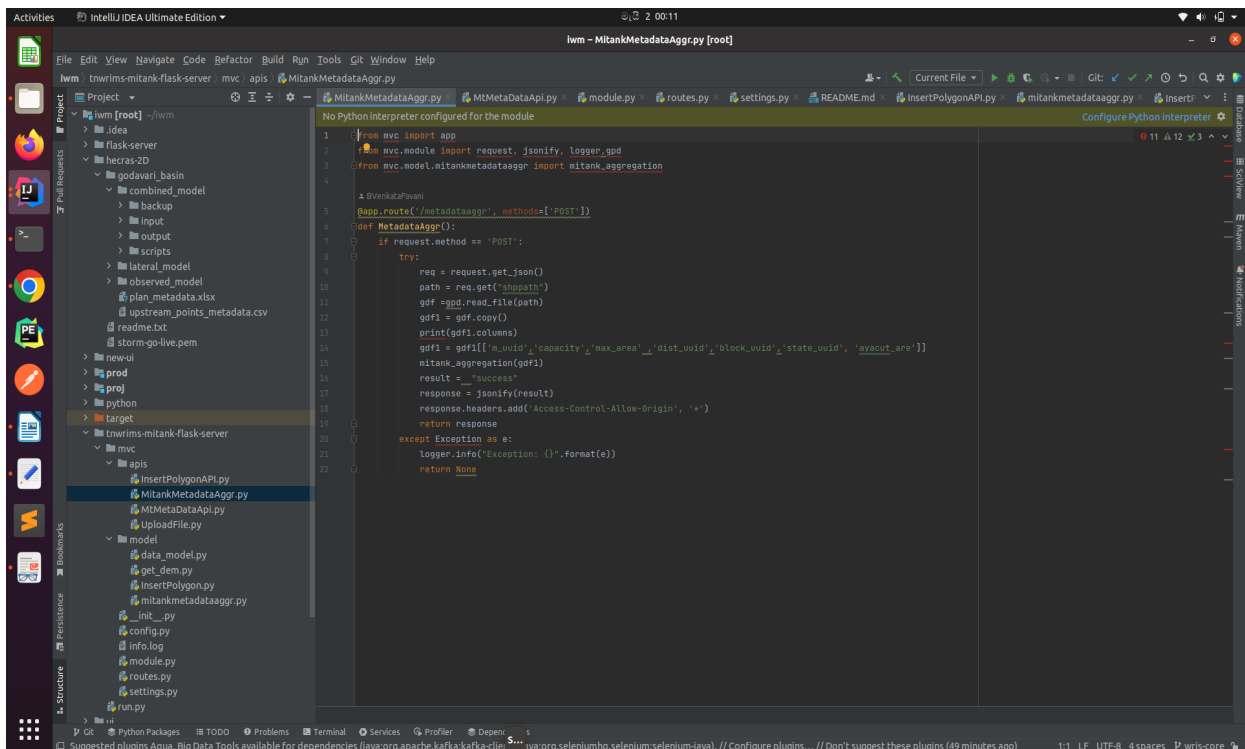


## InsertPolygonApi :-



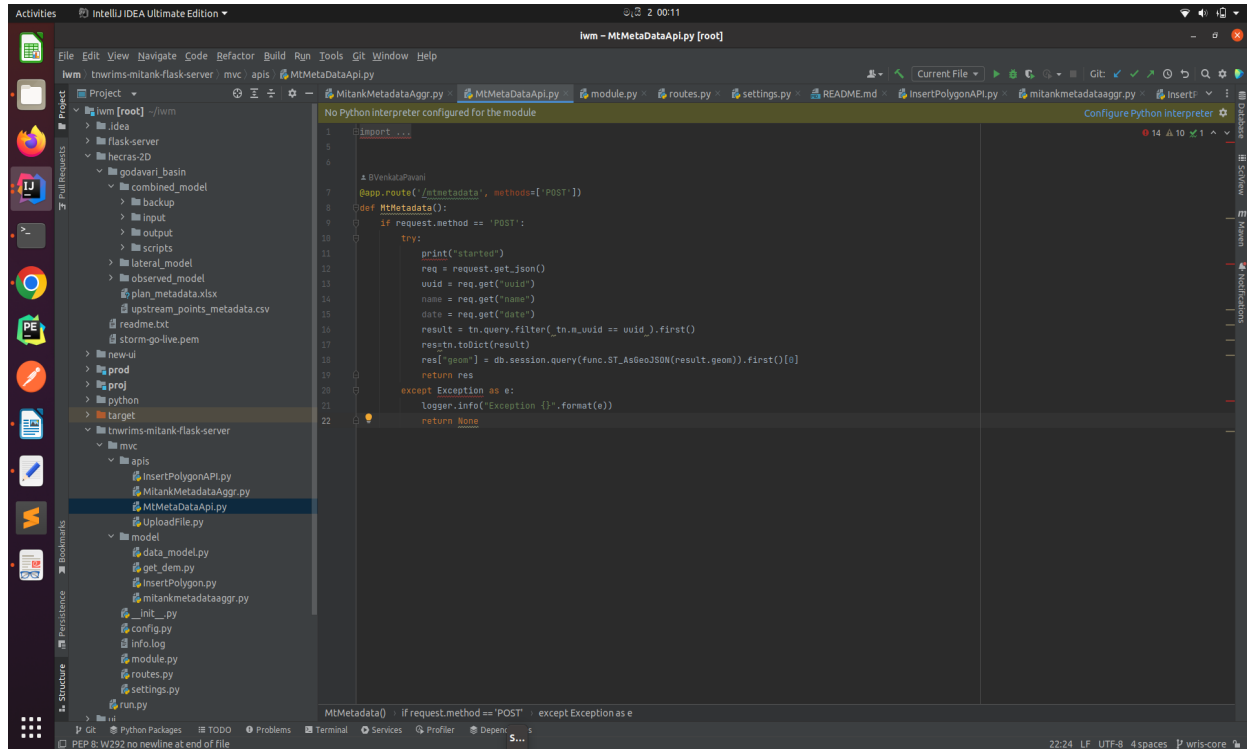
```
1 from mvc import app
2 from mvc.module import request, jsonify, logger
3 from mvc.model.InsertPolygon import insert_in_polygon
4
5 @app.route('/insertmitank', methods=['POST'])
6 def InsertMitank():
7     if request.method == 'POST':
8         try:
9             req = request.get_json()
10            data = req.get('metadata')
11            result = insert_in_polygon(data)
12            response = jsonify(result)
13            response.headers.add('Access-Control-Allow-Origin', '*')
14            return response
15        except Exception as e:
16            logger.info("Exception: {}".format(e))
17            return None
```

## MitankMetadataAggrApi :-



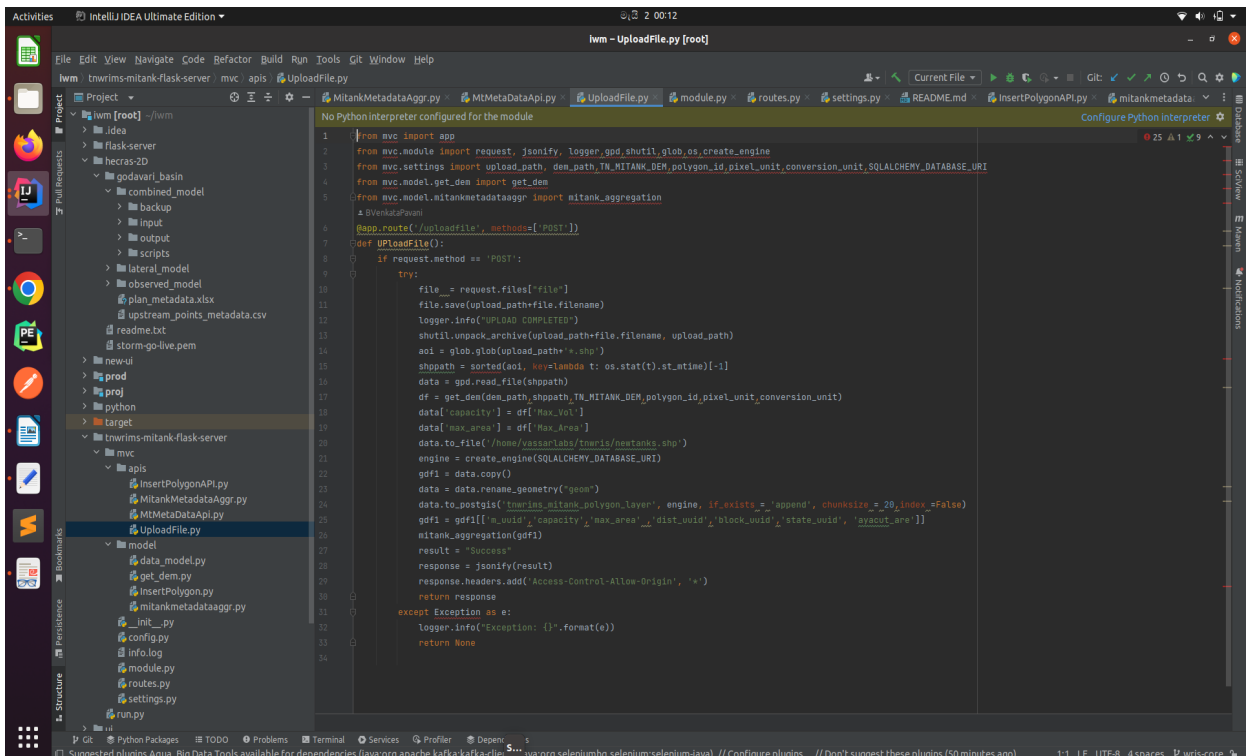
```
1 from mvc import app
2 from mvc.module import request, jsonify, logger, gpd
3 from mvc.model.mitankmetadataaggr import mitank_aggregation
4
5 @app.route('/metadataaggr', methods=['POST'])
6 def MetadataAggr():
7     if request.method == 'POST':
8         try:
9             req = request.get_json()
10            path = req.get('datapath')
11            gdf = gpd.read_file(path)
12            gdf1 = gdf.copy()
13            print(gdf1.columns)
14            gdf1 = gdf1[['x_ward', 'capacity', 'max_area', 'dist_ward', 'block_ward', 'state_ward', 'bycut_ward']]
15            mitank_aggregation(gdf1)
16            result = "success"
17            response = jsonify(result)
18            response.headers.add('Access-Control-Allow-Origin', '*')
19            return response
20        except Exception as e:
21            logger.info("Exception: {}".format(e))
22            return None
```

## MtMetadataApi :-



```
1 import ...
2
3
4
5
6
7 @app.route('/mtmetadata', methods=['POST'])
8 def MtMetadata():
9     if request.method == 'POST':
10         try:
11             print("started")
12             req = request.get_json()
13             uuid = req.get("uuid")
14             name = req.get("name")
15             date = req.get("date")
16             result = tn.query.filter(tn.a.uuid == uuid).first()
17             res = tn.toDict(result)
18             res["geom"] = db.session.query(func.ST_AsGeoJSON(result.geom)).first()[0]
19             return res
20         except Exception as e:
21             logger.info("Exception: {}".format(e))
22             return None
```

## UploadFileApi :-



```
1 from mvc import app
2 from mvc.module import request, jsonify, logger, gpd, shutil, glob, os, create_engine
3 from mvc.settings import upload_path, dem_path, TN_MITANK_DEM, polygon_id, pixel_unit, conversion_unit, SQLALCHEMY_DATABASE_URI
4 from mvc.model.get_dem import get_dem
5 from mvc.model.mitankmetadataagggr import mitank_aggregation
6
7 @app.route('/uploadfile', methods=['POST'])
8 def UploadFile():
9     if request.method == 'POST':
10         try:
11             file = request.files["file"]
12             file.save(upload_path+file.filename)
13             logger.info("UPLOAD COMPLETED")
14             shutil.unpack_archive(upload_path+file.filename, upload_path)
15             aoi = glob.glob(upload_path+"*.shp")
16             shppath = sorted(aoi, key=lambda t: os.stat(t).st_mtime)[-1]
17             data = gpd.read_file(shppath)
18             df = get_dem(dem_path, shppath, TN_MITANK_DEM, polygon_id, pixel_unit, conversion_unit)
19             data["capacity"] = df["max_vol"]
20             data["max_area"] = df["max_area"]
21             data.to_file('/home/gasser/Inria/Inria/mtank/mtank.shp')
22             engine = create_engine(SQLALCHEMY_DATABASE_URI)
23             gdf1 = data.copy()
24             data = data.rename_geometry("geom")
25             data.to_postgis('mtank_mitank_polygon_layer', engine, if_exists='append', chunksize=20, index=False)
26             gdf1 = gdf1[['a_uid', 'capacity', 'max_area', 'dist_uid', 'block_uid', 'state_uid', 'yout_are']]
27             mitank_aggregation(gdf1)
28             result = "Success"
29             response = jsonify(result)
30             response.headers.add('Access-Control-Allow-Origin', '*')
31             return response
32         except Exception as e:
33             logger.info("Exception: {}".format(e))
34             return None
```

## Tools and Technologies

### 5.1 Hardware Components

- Processor: 64-bit, quad-core, 2.5 GHz minimum per core
- RAM: 4 GB or more.
- HDD: 5 GB of available space or more.
- Display: Dual XGA (1024 x 768) or higher resolution monitors.
- Keyboard: A standard keyboard

### 5.2 Software Components

- **Java:** Java is a general-purpose, high-level programming language that is designed to be platform-independent, meaning that Java code can run on any platform that supports Java without the need for recompilation.
  - o Version:8 or 11
- **IntelliJ Idea:** IntelliJ IDEA is an integrated development environment (IDE) for Java and other programming languages, developed by JetBrains. It is designed to increase developer productivity by providing a powerful set of tools and features that simplify the coding process.
- Ubuntu OS 20.04
- **Cassandra:** Cassandra is a column-oriented database, which means that data is stored in tables as columns rather than rows.
  - o Version: 5.0.1
- **MySQL:** MySQL is a relational database management system(RDBMS) developed by oracle that is based on structured query language
  - o Version:5.7

### 5.3 Flask server

Flask is a lightweight Python web framework that provides useful tools and features for creating web applications in the Python Language. It gives developers flexibility and is an accessible framework for new developers because you can build a web application quickly.

Flask is a widely used micro web framework for creating APIs in python.

In mitank module we use the following libraries in python :-

- ➔ flask :-
- ➔ flask\_restful :-
- ➔ flask\_core
- ➔ rasterio
- ➔ json
- ➔ psycopyg2
- ➔ numpy
- ➔ pandas
- ➔ shapely.geometry
- ➔ sqlalchemy

### 5.4 Spring Framework

Spring is the most popular application development framework for enterprise Java. Millions of developers around the world use Spring Framework to create high performing, easily testable, and reusable code. Spring enables developers to develop enterprise-class applications using POJOs. The benefit of using only POJOs is that you do not need an EJB container product such as an application server but you have the option of using only a robust servlet container such as Tomcat or some commercial product. Testing an application written with Spring is simple because environment-dependent code is moved into this framework. Spring framework provides inversion of control and dependency injection. In dependency injection we will use `@component` for the class declaration and `@autowired` for the object. Normally for the object creation we will be using `new class name()` but by using spring framework we just need to write `@autowired class-name object-name` then object will be automatically created and also one more advantage is that only one object will be created.

## 5.5 Maven

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information. When we use maven we no need to take care of dependencies. In each module we will be having the POM file where we will be writing all the libraries that module requires when we build the project it will automatically all those libraries and store them jars in maven repository(.m2) directory .

## 5.6 Kafka and Storm

**Kafka :** Apache Kafka is a distributed publish-subscribe messaging system and a robust queue that can handle a high volume of data and enables you to pass messages from one end-point to another. Kafka is suitable for both offline and online message consumption. Kafka messages are persisted on the disk and replicated within the cluster to prevent data loss. Kafka is built on top of the ZooKeeper synchronization service. It integrates very well with Apache Storm and Spark for real-time streaming data analysis.

**Storm :** It is an open-source and real-time stream processing system. Apache Storm was mainly used for fastening the traditional processes. It reliably processes the unbounded streams. It has spouts and bolts for designing the storm applications in the form of topology. Any programming language can use it. Thus, it is simple to use. It can process millions of messages within a second.

## 5.7 Databases

### **MySQL :**

MySQL is a relational database management system(RDBMS) developed by oracle that is based on structured query language. A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or a place to hold the vast amounts of information in a corporate network. In particular, a relational database is a digital store collecting data and organizing it according to the relational model. In this model, tables consist of rows and columns, and relationships between data elements all follow a strict logical structure. An RDBMS is simply the set of software tools used to actually implement, manage, and query such a database. It contains static data in our project.

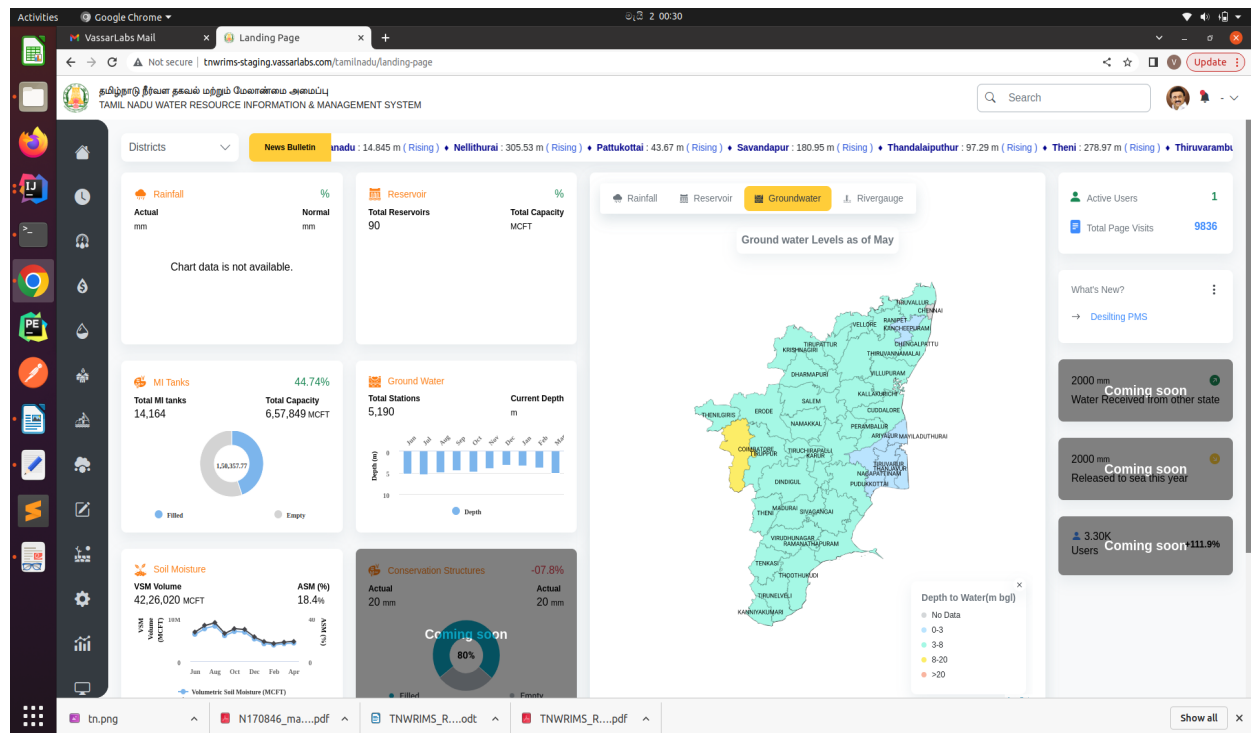
### **Cassandra:**

Cassandra is a column-oriented database, which means that data is stored in tables as columns rather than rows. This makes it highly efficient for querying large datasets because it only reads the columns that are required for a query, rather than reading entire rows. Cassandra is also designed to handle high write-throughput, making it well-suited for applications that require real-time data processing. In our project we will be storing business data in Cassandra.

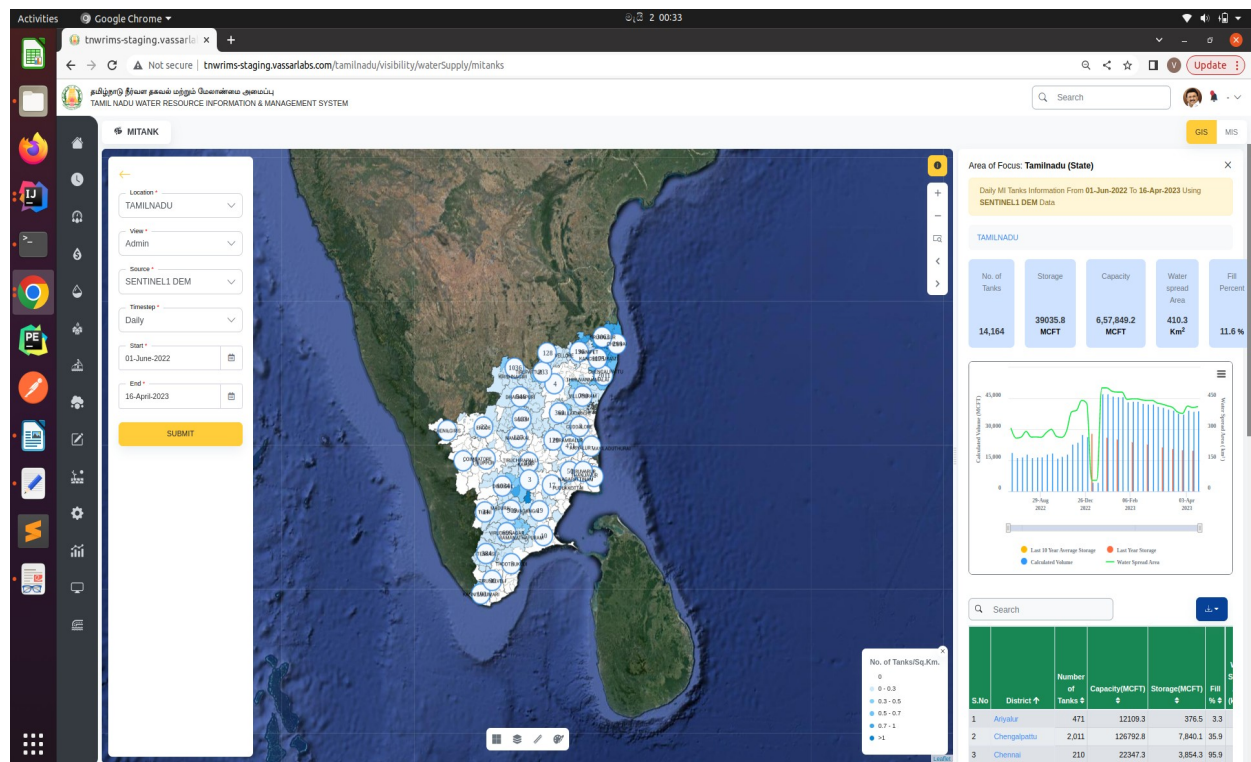
### **PostgresSQL :**

Postgres is a traditional RDBMS(relational database management system) SQL database. It is highly stable database management system. PostgreSQL is used as the primary data store or data warehouse for many web, mobile, geospatial and analytical applications.

## Output :-



Home Page



Mitank GIS Page

Activities Google Chrome 2 00:33

MIS MI Tank

Not secure | tnwrims-staging.vassarlabs.com/tamilnadu/mis/mismitankspType=STATE;cType=DISTRICT;location=TAMILNADU;uid=e98cd5b7-6556-4c0f-a778-3429e1c14a6b;src=SENTINEL1\_DEM

தமிழ்நாடு நீர்வளப் பராமிதிகளின் மேலாண்மை அமைப்பு  
TAMIL NADU WATER RESOURCE INFORMATION & MANAGEMENT SYSTEM

MITANK

Select Source  
SENTINEL1\_DEM

TAMILNADU

Search

S.No	District	Number of Tanks	Capacity(MCFT)	Storage(MCFT)	Fill %	Water Spread Area (km²)	No. of Tanks with Storage				
							75-100 (%)	50-75 (%)	25-50 (%)	0-25 (%)	0 (%)
1	Aranyapur	471	12.1	0.4	3.3	5.7	0	0	0	232	0
2	Chengalpattu	2,011	126.8	7.8	35.9	76.1	40	66	162	546	0
3	Chennai	210	22.3	3.9	95.9	20.4	0	2	4	17	0
4	Coimbatore	-	0	0	-	-	-	-	-	-	-
5	Cuddalore	3	0.1	0.0	2.9	0.0	0	0	0	2	0
6	Dharmapuri	546	24.2	0.3	1.3	4.7	0	0	1	208	0
7	Dindigul	1,034	21.1	0.4	2.0	6.4	0	0	1	409	0
8	Erode	22	1.5	0.2	11.4	1.5	0	0	0	20	0
9	Kallakurichi	360	22.7	1.2	5.3	14.6	0	0	1	260	0
10	Kancheepuram	1,175	104.7	9.9	45.2	102.7	32	55	130	368	0
11	Kanniyakumari	193	0.5	0.0	5.1	0.1	0	0	0	8	0
12	Karur	106	2.0	0.1	4.7	1.5	0	0	0	69	0
13	Koduvayur	1,036	21.9	1.2	5.6	14.2	0	0	4	570	0
14	Madurai	909	45.3	0.4	0.9	7.1	0	0	2	362	0
15	Mayiladuthurai	-	0	0	-	-	-	-	-	-	-
16	Nagapattinam	-	0	0	-	-	-	-	-	-	-
17	Namakkal	127	3.7	0.1	2.9	1.0	0	0	1	43	0
18	Perambalur	129	5.0	0.1	2.6	1.4	0	0	0	60	0
19	Pudukkottai	17	1.4	0.0	2.9	0.7	0	0	0	14	0
20	Ramanathapuram	10	0.9	0.0	4.5	0.6	0	0	0	7	0
21	Ranipet	196	8.3	0.3	4.2	4.4	0	0	0	125	0
22	Salem	183	6.4	0.3	4.1	3.2	0	0	0	105	0
23	Shivagangai	19	2.3	0.1	3.7	0.9	0	0	0	12	0
24	Tiruvallur	384	20.0	0.6	3.1	9.5	0	0	4	219	0
25	Thiruvannamalai	50	0.7	0.0	3.8	0.5	0	0	0	31	0

Mitank MIS Page

Activities Google Chrome 2 00:33

Content Mgmt - Data Entry

Not secure | tnwrims-staging.vassarlabs.com/tamilnadu/content-management/data-entry-forms/dashboardType=data-entry-forms

தமிழ்நாடு நீர்வளப் பராமிதிகளின் மேலாண்மை அமைப்பு  
TAMIL NADU WATER RESOURCE INFORMATION & MANAGEMENT SYSTEM

Data Entry

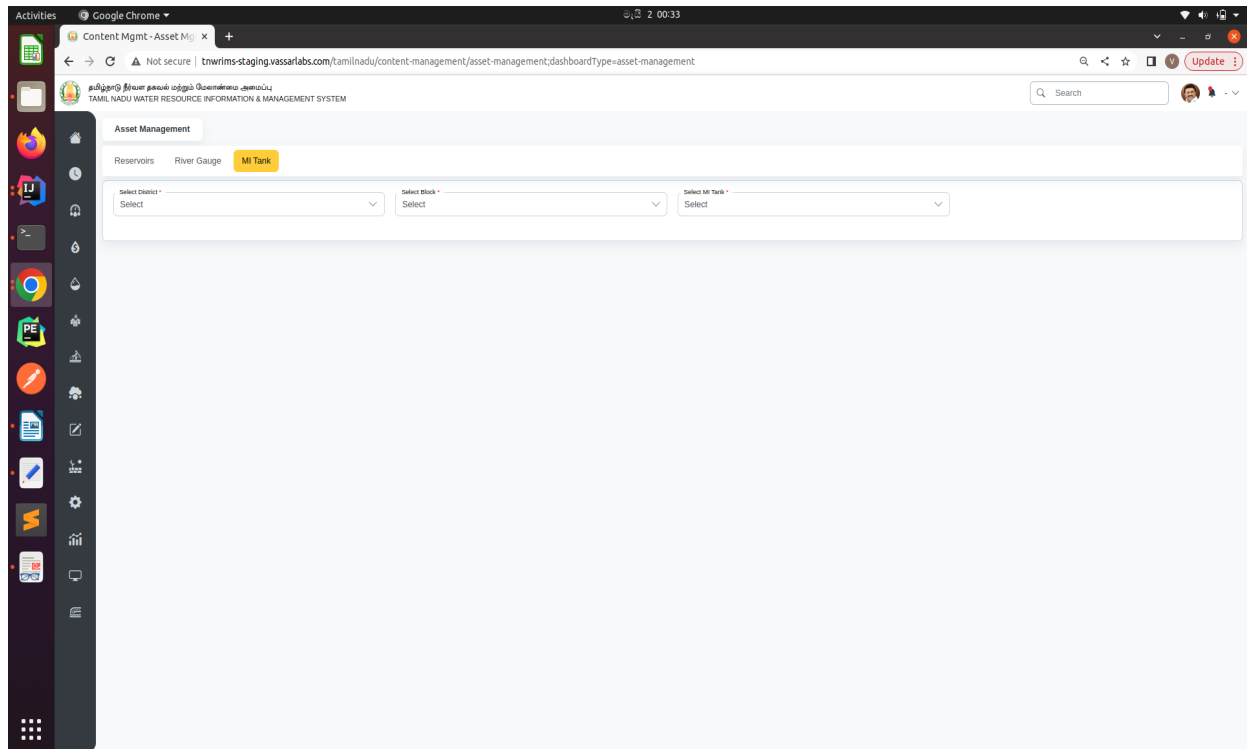
Reservoirs River Gauge Rainfall Ground Water **MI Tank**

MI Tank Meta Data Bulk Upload

Upload Data

Mitank Metadata Bulk Upload





## Asset Management

The screenshot shows the 'Last Level Mitank page to edit Mitank Information'. The page displays a form for editing the 'Andarkuppam Tank'. The form includes the following fields:

- Select District: Chengalpattu
- Select Block: Chitlamur
- Select MI Tank: Andarkuppam Tank
- Andarkuppam Tank (Title)
- District Name: Chengalpattu
- Block Name: Chitlamur
- Panchayat Name: Andar Kuppam
- Habitation Name:
- Name of the MI Tank: Andarkuppam Tank
- Local Name:
- Tank State: in progress
- Nature of Tank: Non System
- Total No. of Survey no:
- Area of the tank: 0.22287474
- Ayacut area: 113.15
- Whether Karuvel Plantation exist in the water storage area:
- Water Spread area (in Hectares)at FTL: 49
- Water Spread area (in Hectares)at MWL:
- Capacity of the Tank: 21.87413024902344
- Total No. of Inlet channels:
- Inlet Channels Type:
- Inlet Channels Condition:

## Last Level Mitank page to edit Mitank Information

**Test Case Report:-**

S.No.	Test Cases	Input	Expected Result	Actual Result	Status	Remarks
1	Test whether the graph is displaying or not	Enter todays date and any source	Graph displayed in gis page	Graph displayed	Pass	No remarks
2	Checking whether map is correctly displaying	Enter todays date and any source	Map is to be displayed for the source	Map is displayed	Pass	No remarks
3	Validate the count and capacity of mitank	Enter todays date and select any source	Correct count and capacity values to be displayed	Count and capacity is displayed correctly	Pass	No remarks
4	Update any field in mitank fields in asset management and try to update	Change the name of the mitank	Name of the mitank should to changed and updated successfull y	Name of mitank updated succesful ly	Pass	No remarks
5	Try to upload shape file in content management	Shape file containin g mitanks	Display uploaded successfull y	Uploade d successfu lly displayed	Pass	No remarks

## **Conclusion**

Our dashboard provides information on water resources, providing near real-time visibility of water available from rainfall, reservoirs, canals, minor irrigation tanks, groundwater etc which helps the key decision makers for effective planning of water resource management .

A single, unified and an authoritative platform for comprehensive, authoritative and consistent data and information of water resources for the state of Andhra Pradesh rendering real-time visibility of water resources.

## References

1. <https://apwrims.ap.gov.in/>
2. <http://kaleshwaram.vassarlabs.com/>