Report for Research Teaser – IoT'25 (Software Task 2)

IoT Sensor Data Visualization Dashboard using FastAPI, PostgreSQL, and Apache ECharts

Name: Varshashri
College: ACE Engineering College, JNTUH
Branch: Artificial Intelligence & Data Science
Task: Software Task 2 – Data Visualization
Year: 2025

---

1. Introduction

The objective of this project is to design and develop a complete IoT Sensor Data Visualization Dashboard that retrieves data from a database, exposes it through a FastAPI backend, and displays it interactively using a React-based frontend integrated with Apache ECharts.

This project aims to visualize real-time IoT data for three verticals:

- **Air Quality (AQ)**
- **Water Flow (WF)**
- **Smart Lighting (SL)**

Each vertical represents sensor data captured from different IoT devices and is processed to provide meaningful visual insights.

---

2. Objectives

- To clean and preprocess IoT sensor data for accuracy and consistency.
- To store cleaned data into a PostgreSQL database.
- To create REST APIs using FastAPI for each vertical.
- To visualize data interactively using React and Apache ECharts.
- To provide a simple, user-friendly dashboard for analyzing sensor data trends.

---

3. Tools and Technologies Used

Layer	Tools / Frameworks
Programming Languages	Python, JavaScript
Backend Framework	FastAPI
Database	PostgreSQL
Frontend	React.js
Visualization	Apache ECharts
Libraries	Pandas, Axios, SQLAlchemy
Environment	VS Code, pgAdmin, Node.js

---

4. Step 1 – Data Preparation and Cleaning (Python)

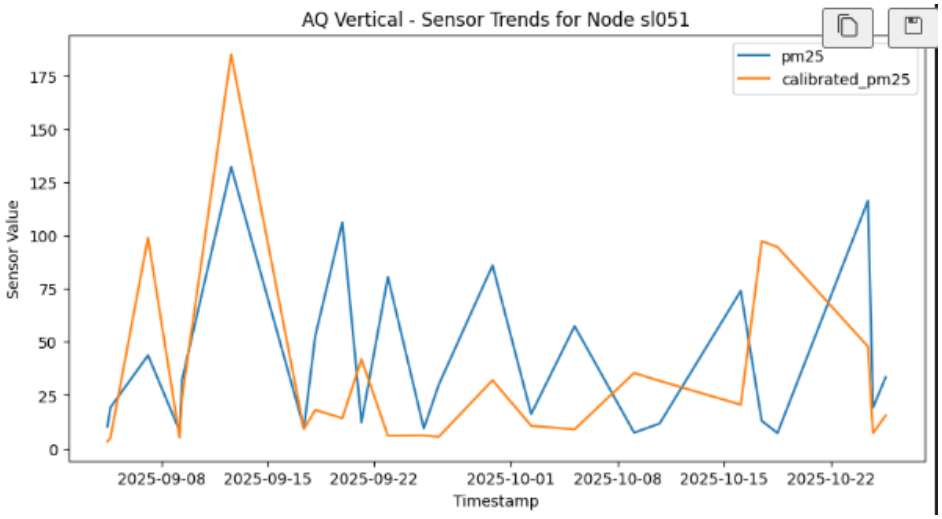
The preprocessing phase ensures that raw IoT data is clean and well-structured before storage.

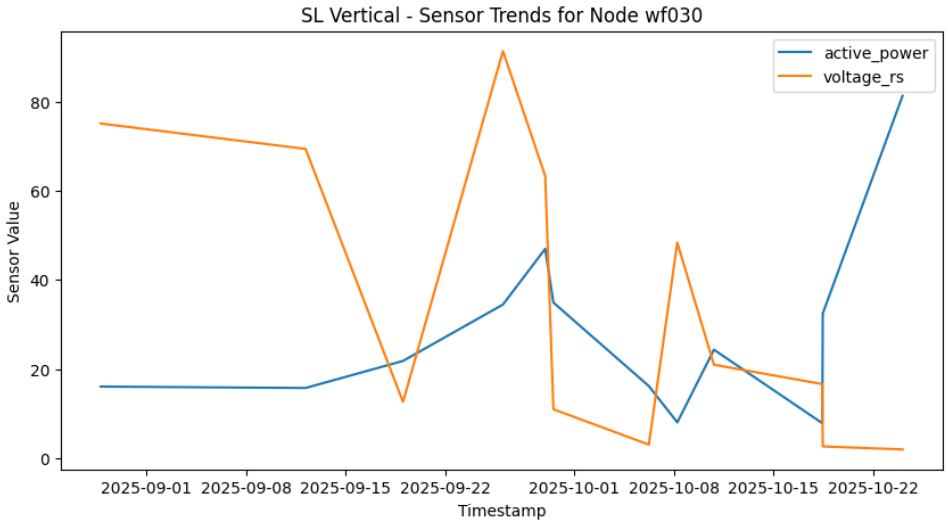
Steps Performed:

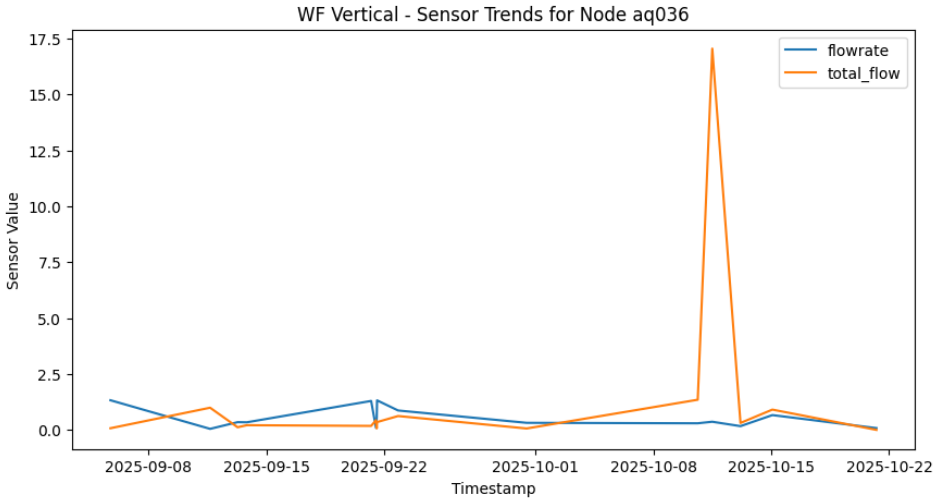
1. Imported datasets iot_dataset.csv and iot_dataset_mapping.csv using Pandas.
2. Checked dataset structure, datatypes, and missing values.
3. Converted timestamps to standard datetime format.
4. Standardized vertical names and removed duplicates.
5. Replaced missing numeric values with median values.
6. Parsed the mapping file to rename sensor columns correctly.
7. Split the dataset by verticals (AQ, WF, SL).
8. Visualized trends using Matplotlib for exploratory analysis.

Output:

- Cleaned datasets for each vertical.
- Line plots showing sensor trends over time (Temperature, Humidity, PM2.5, etc.)







---

5. Step 2 – Backend API Development (FastAPI + PostgreSQL)

The backend provides RESTful API endpoints for each IoT vertical using FastAPI.

Key Steps:

- Connected FastAPI with PostgreSQL database (Task1_IOT_project).
- Uploaded datasets into PostgreSQL using ingest.py.
- Created tables: aq_data, wf_data, sl_data.
- Implemented endpoints in main.py for each vertical.
- Enabled CORS to allow React frontend access.
- Verified successful responses from endpoints in /docs.

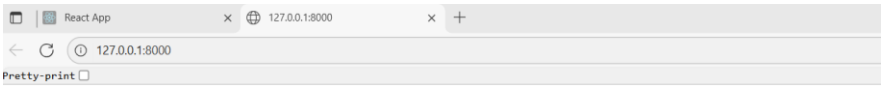
Endpoints:

Endpoint	Description
/	Health check (verifies DB connection)
/api/verticals/AQ	Returns Air Quality data
/api/verticals/WF	Returns Water Flow data
/api/verticals/SL	Returns Smart Lighting data

Backend Run Command:

uvicorn main:app --reload

```
PS C:\Users\Srinivas\Desktop\Varshashri_ACE_SWTask2> cd Backend
PS C:\Users\Srinivas\Desktop\Varshashri_ACE_SWTask2\Backend> python ingest.py
✅ Uploaded AQ → aq_data (4582 rows)
✅ Uploaded SL → sl_data (2413 rows)
✅ Uploaded WF → wf_data (3005 rows)
PS C:\Users\Srinivas\Desktop\Varshashri_ACE_SWTask2\Backend>
```



---

6. Step 3 – Frontend Visualization (React + Apache ECharts)

The frontend displays interactive visualizations fetched from the FastAPI API using Axios.

Key Features:

- Interactive charts built with **Apache ECharts**.
- Real-time data fetched from /api/verticals/{vertical}.
- Navigation buttons for switching between AQ, WF, SL dashboards.
- Tooltips display sensor values when hovering over points.
- Responsive layout and smooth transitions.

Frontend Run Commands:

cd Frontend

npm install

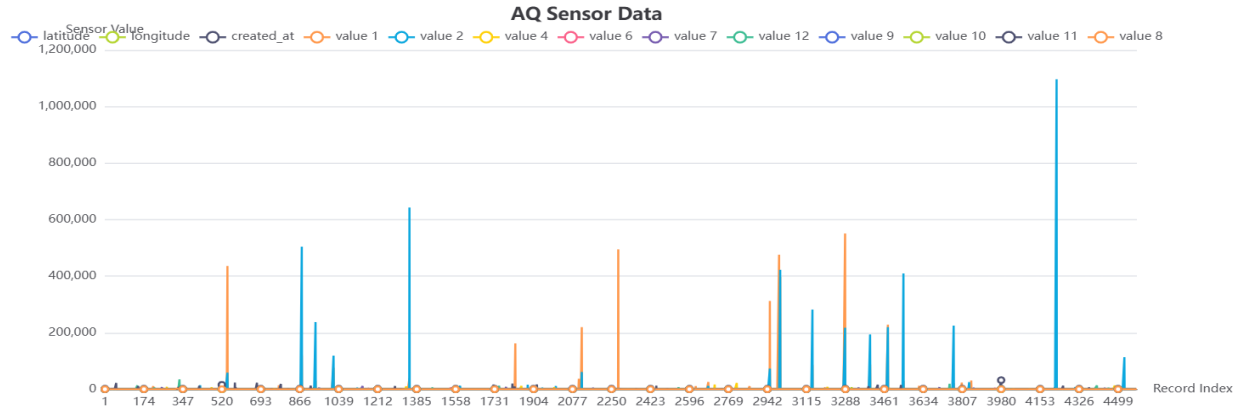
npm start

Frontend Components:

- ChartComponent.js → Renders charts using ECharts.
- Dashboard.js → Fetches and displays data for each vertical.
- App.js → Handles navigation and layout.

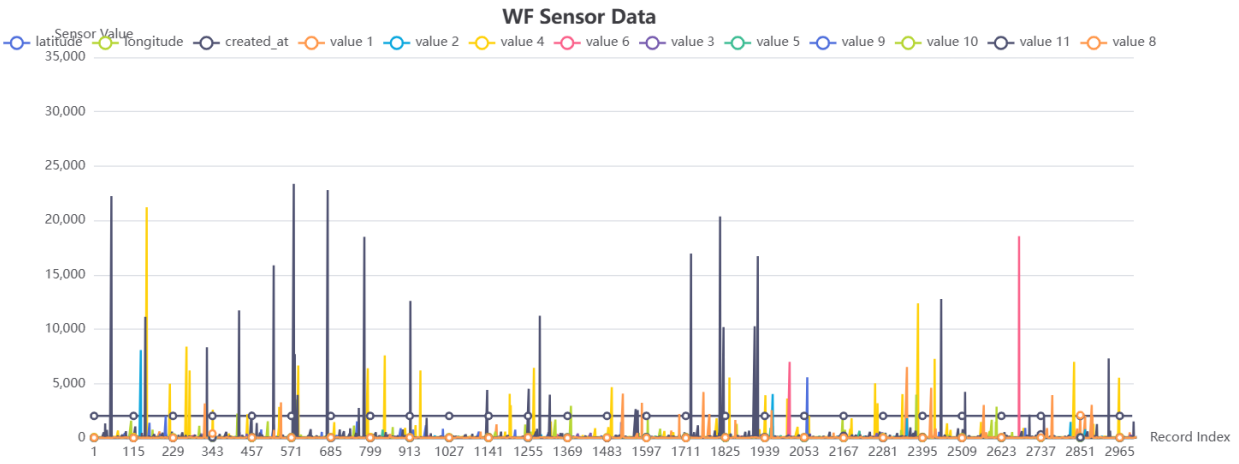
IoT Data Visualization Dashboard

AQ Dashboard | WF Dashboard | SL Dashboard



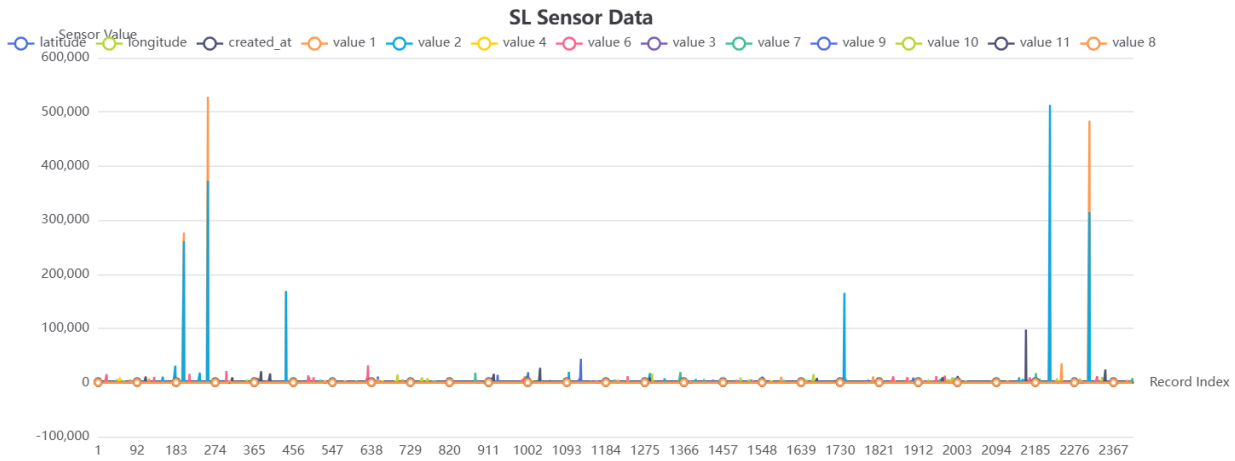
IoT Data Visualization Dashboard

AQ Dashboard | WF Dashboard | SL Dashboard



IoT Data Visualization Dashboard

AQ Dashboard | WF Dashboard | SL Dashboard



7. Results and Observations

- IoT datasets successfully processed and visualized.
- FastAPI connected seamlessly to PostgreSQL.
- React frontend rendered dynamic ECharts with real-time data.
- AQ, WF, SL dashboards provided clear data trends and interactive insights.

Key Takeaways:

- Data flow established:
  PostgreSQL → FastAPI → React → Apache ECharts
- Enhanced understanding of full-stack IoT data pipelines.
- Visualization enables faster and more intuitive data analysis.

---

8. Challenges Faced

- Handling database connection issues with PostgreSQL URL encoding.
- Mapping column names dynamically across verticals.
- Ensuring CORS compatibility between FastAPI and React.
- Adjusting ECharts configuration for dynamic datasets.

---

9. Conclusion

The project successfully implemented an **IoT Data Visualization Dashboard** integrating:

- Data preprocessing in Python
- FastAPI backend with PostgreSQL
- Interactive frontend using React and ECharts

This full-stack approach demonstrates end-to-end IoT data handling — from collection and cleaning to visualization — providing a scalable solution for real-world sensor monitoring applications.

---

10. Appendix

- Python Notebook: Varshashri_ACE_recommender.ipynb
- Backend Files: ingest.py, main.py
- Frontend: React + ECharts source files
- Database: PostgreSQL (Task1_IOT_project)

✅ **End of Report**

Prepared by: *Varshashri Nagapuri*
**B.Tech – AI & Data Science, ACE Engineering College (JNTUH)*