



24-25J-224

Intelligent Eco-Urban Monitoring System (IEMS)

Sri Lanka Institute of
InformationTechnology



OUR TEAM

24-25J-224



Dr. Samantha Rajapaksha
Head | Department of IT
Supervisor



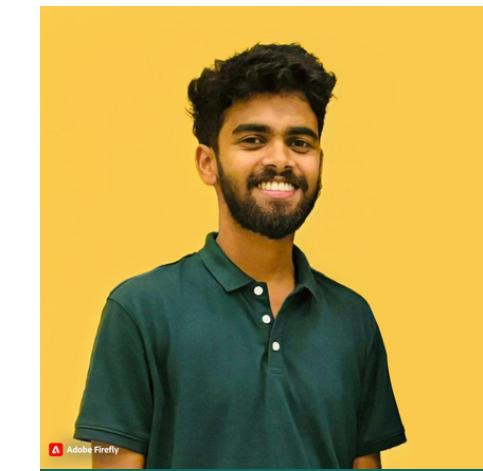
Ms. Kaushika Kavindi
Assistant Lecturer
Co-Supervisor



Thuduwage I.M.H.G
IT21169380
Software Engineering



Arandara S. D.
IT21164330
Software Engineering



Karunarathne R. Y. D.
IT21169144
Software Engineering



Kodithuwakku C.K.
IT21156960
Software Engineering

INTRODUCTION



Urban Environmental Challenges

- Rapid urbanization leads to issues like poor air quality, green space loss, noise pollution, and high vehicle emissions.

Why Address These Issues?

- Sustainable urban living is vital for residents' well-being, requiring effective monitoring and solutions.

Introduction to IEMS

- IEMS addresses these challenges with integrated monitoring and prediction, featuring EcoSensor AI, GreenVision AI, NoiseGuard AI, and Eco Go.

RESEARCH PROBLEM



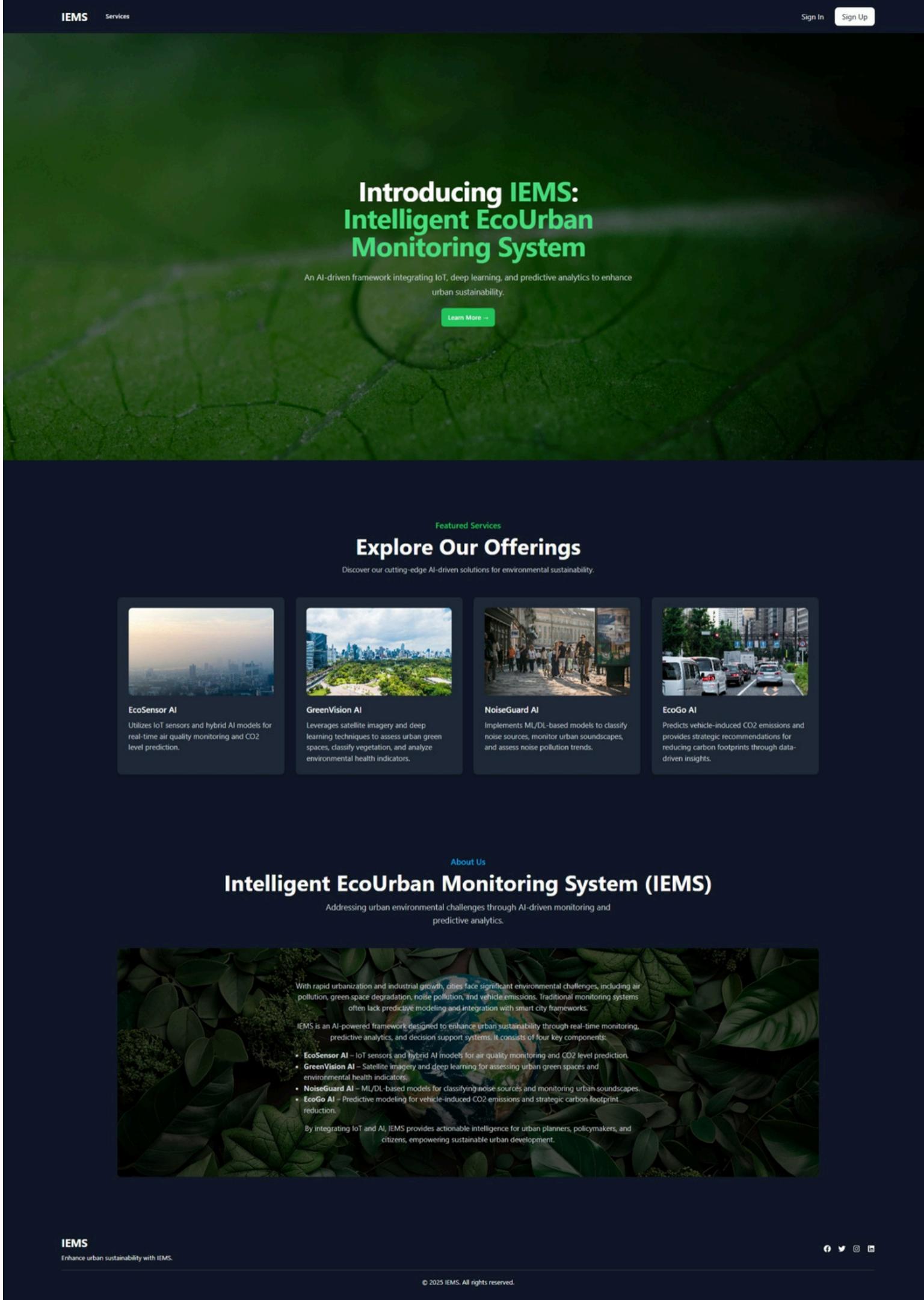
Environmental Challenges in Urban Areas

- **Air Quality Concerns**
 - Urban pollution from transport and industry includes harmful pollutants like PM2.5, NO₂, and O₃, posing health risks.
- **Green Space Degradation**
 - Urbanization reduces and fragments green spaces, impacting carbon capture and biodiversity.
- **Noise Pollution**
 - Noise from transport and construction causes stress and health issues like sleep disturbances.
- **Vehicle Emissions**
 - Increasing traffic contributes to air pollution and greenhouse gases, requiring emission reduction strategies.

IMPLEMENTED SOLUTION

Integrated Environmental Monitoring System (IEMS)

- **EcoSensor AI**
 - Real-time air quality monitoring and predictions.
- **NoiseGuard AI**
 - Smart noise monitoring and source identification.
- **Eco Go**
 - Predicts and reduces vehicle CO2 emissions.
- **GreenVision AI**
 - Manages green spaces using satellite imagery.



Introducing IEMS:
Intelligent EcoUrban Monitoring System

An AI-driven framework integrating IoT, deep learning, and predictive analytics to enhance urban sustainability.

[Learn More >](#)

Featured Services

Explore Our Offerings
Discover our cutting-edge AI-driven solutions for environmental sustainability.



EcoSensor AI
Utilizes IoT sensors and hybrid AI models for real-time air quality monitoring and CO2 level prediction.



GreenVision AI
Leverages satellite imagery and deep learning techniques to assess urban green spaces, classify vegetation, and analyze environmental health indicators.



NoiseGuard AI
Implements ML/DL-based models to classify noise sources, monitor urban soundscapes, and assess noise pollution trends.



EcoGo AI
Predicts vehicle-induced CO2 emissions and provides strategic recommendations for reducing carbon footprints through data-driven insights.

About Us

Intelligent EcoUrban Monitoring System (IEMS)
Addressing urban environmental challenges through AI-driven monitoring and predictive analytics.

With rapid urbanization and industrial growth, cities face significant environmental challenges, including air pollution, green space degradation, noise pollution, and vehicle emissions. Traditional monitoring systems often lack predictive modeling and integration with smart city frameworks.

IEMS is an AI-powered framework designed to enhance urban sustainability through real-time monitoring, predictive analytics, and decision support systems. It consists of four key components:

- **EcoSensor AI** – IoT sensors and hybrid AI models for air quality monitoring and CO2 level prediction.
- **GreenVision AI** – Satellite imagery and deep learning for assessing urban green spaces and environmental health indicators.
- **NoiseGuard AI** – ML/DL-based models for classifying noise sources and monitoring urban soundscapes.
- **EcoGo AI** – Predictive modeling for vehicle-induced CO2 emissions and strategic carbon footprint reduction.

By integrating IoT and AI, IEMS provides actionable intelligence for urban planners, policymakers, and citizens, empowering sustainable urban development.

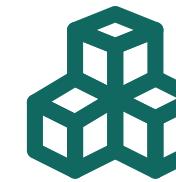
IEMS
Enhance urban sustainability with IEMS.

© 2025 IEMS. All rights reserved.

Facebook Twitter Instagram LinkedIn

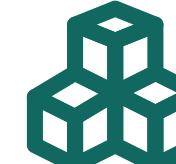
Main Objective

Enhancing Urban Sustainability



Goal

To enhance urban sustainability through effective monitoring, prediction, and mitigation of environmental challenges.



Objectives

- Optimize green spaces.
- Improve air quality.
- Reduce noise pollution.
- Lower vehicle emissions by leveraging advanced AI and IoT technologies.

Sub Objectives

Specific Goals and Tasks

.....• **Arandara.S.D:**

- Deploy IoT sensors for CO₂ level monitoring.
- Train models for air quality prediction and recommendations.
- Implement adaptive sensor calibration using AI for dynamic accuracy adjustments.

.....• **Thuduwage I.M.H.G:**

- Analyze satellite imagery for green space management.
- Train deep learning models for vegetation assessment.
- Provide visualization tools for data analysis and decision making.

Sub Objectives

Specific Goals and Tasks

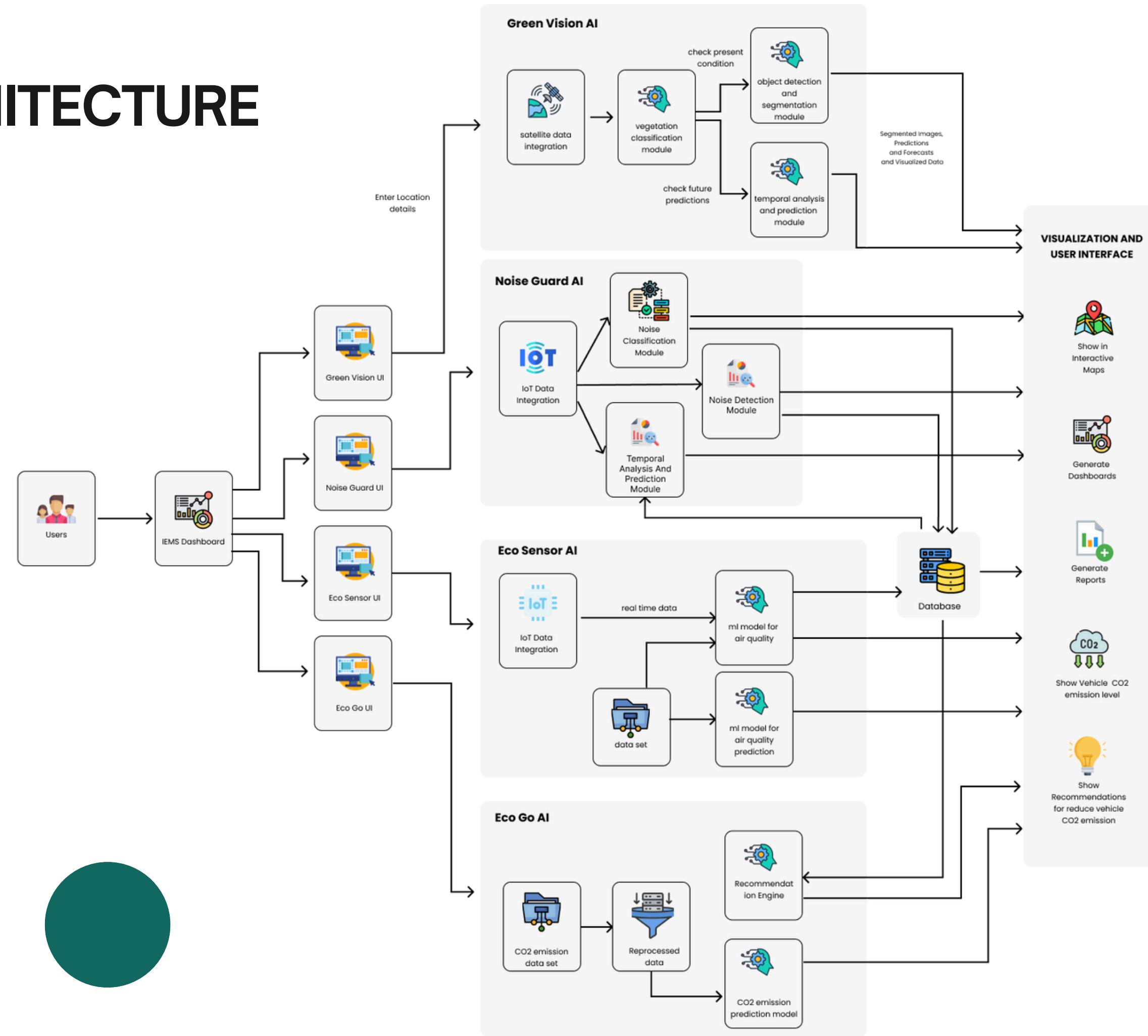
.....• **Karunarathne R.Y.D:**

- Implement smart noise monitoring with IoT sensors.
- Classify noise sources and predict future noise levels.
- Develop alert mechanisms for noise thresholds.

.....• **Kodithuwakku C.K**

- Develop a personalized CO2 emissions prediction and reduction engine.
- Train machine learning models and implement user feedback systems.
- Continuously retrain models to adapt to new data and feedback.

SYSTEM ARCHITECTURE



Team Member 1



GreenVision AI:
Managing Urban Green Spaces

I.M.H.G. Thuduwage
IT21169380

Background

Urban green spaces are essential for reducing heat, improving air quality, and supporting biodiversity, but urban density and pollution make maintenance challenging.

- **Deep Learning Analysis**
 - AI analyzes satellite images to manage green spaces.
- **Vegetation Indices and Object Detection**
 - Assesses health and distribution of greenery.
- **Informed Decision-Making**
 - Identifies areas needing more green space.
- **Environmental Benefits**
 - Enhances biodiversity, reduces heat, and improves environmental quality.





Research Problem

Urban areas struggle to assess and manage green spaces due to fragmented data and lack of effective analysis tools. An advanced system is needed to analyze and enhance green spaces.

Research Gap

Features	[1]	[2]	[3]	[4]	IEMS
Multi-Layered Vegetation Indices	Partial	✓	✗	Partial	✓
Vegetation Classification	✓	Partial	✗	Partial	✓
Object Detection and Segmentation	✗	Partial	✗	✗	✓
Temporal Analysis and Prediction	✗	✗	✗	Partial	✓
Real-Time Data Integration	✗	✗	✗	✗	✓
Visualization and User Interface	Partial	✓	Partial	Partial	✓

Objectives

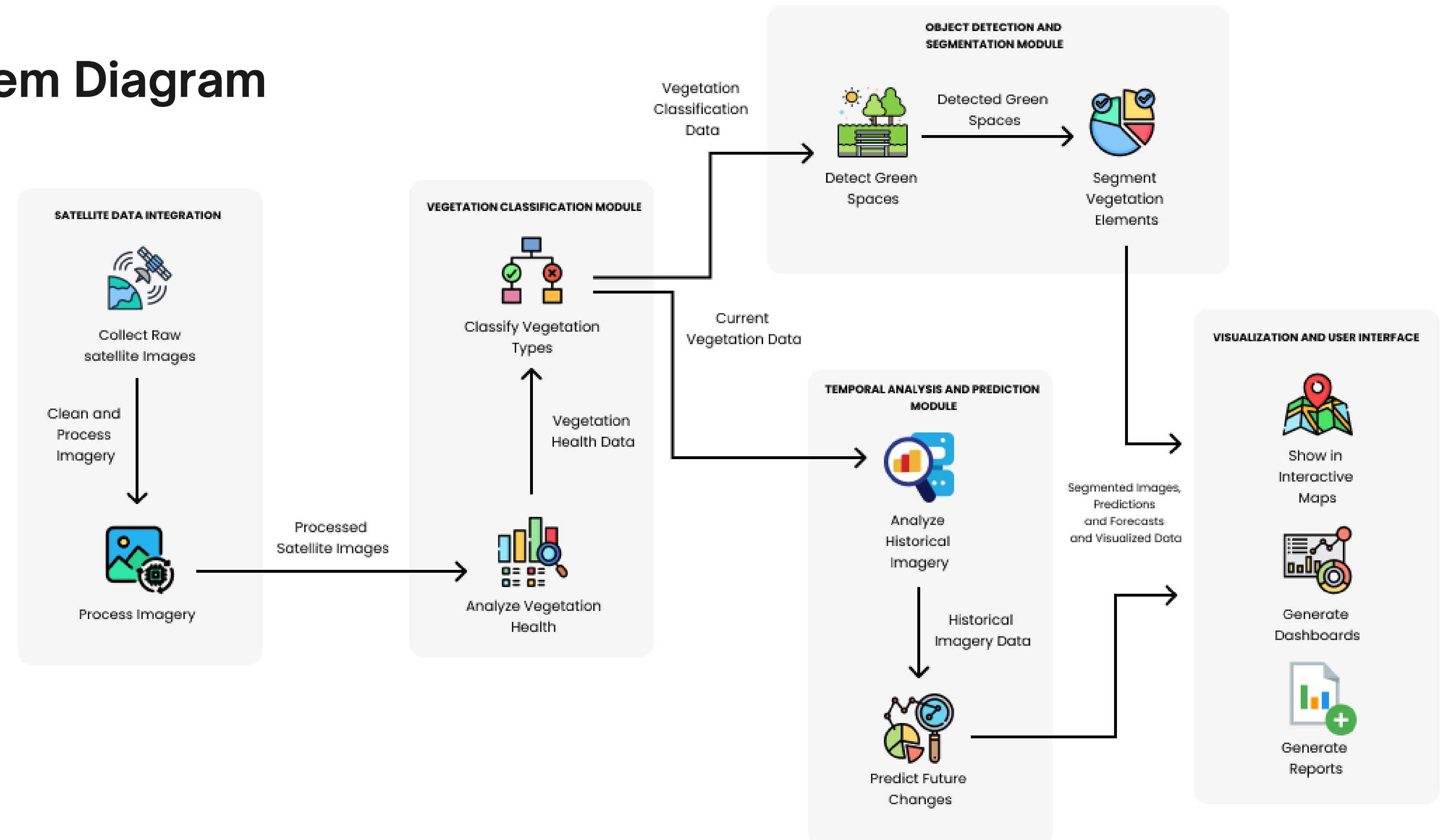
• Main Objectives

- Improve urban sustainability by monitoring, predicting, and managing green spaces.
- Provide cities with advanced tools for better urban development and environmental management

• Sub-Objectives

- Use satellite images and AI to manage green spaces.
- Identify areas needing more greenery based on green coverage and Temporal Analysis and Prediction.
- Guide decisions on creating new green spaces to improve biodiversity and reduce heat.

System Diagram



Technologies Used

Deep Learning

- CNN, U-Net – Tree segmentation and vegetation classification.
- LSTM – Predicting future vegetation coverage and CO2 levels.

Remote Sensing

- NDVI Calculation – Vegetation density classification.
- Satellite Imagery Processing – Green area analysis.

Backend

- Flask API, TensorFlow/Keras – Image processing and AI model inference.
- OpenCV – Image preprocessing and visualization.

Frontend & Integration

- React.js, Google Maps API – UI for selecting locations and viewing results.
- Axios – Frontend-backend communication.

Current Progress



- Developed the core backend models and Flask API for GreenVision AI.
- Created a tree segmentation model using a CNN architecture to classify vegetation areas from satellite images.
- Built a forecasting model to predict future vegetation coverage based on historical data
- Implemented image preprocessing techniques such as normalization, resizing, and mask processing for accurate segmentation.
- Developed vegetation density analysis using NDVI calculations to categorize high, medium, and low vegetation coverage (app.py).
- Created a Flask API to handle image uploads, process satellite images, and return predictions (app.py).



- Developed the frontend interface using React for an interactive user experience.
- Integrated the React app with the Flask backend, allowing users to interact with the AI model.
- Implemented a map selection feature where users can choose a location from a satellite map.
- Added functionality to capture and crop a snippet of the selected location.
- Enabled image and location data upload to the backend for analysis.
- Developed an interactive dashboard to visualize green area distribution and provide recommendations based on AI predictions.

Project Evidence

IT21169380 | Thuduvage I.M.H.G. | 24-25J-224

IEMS Services Sign In Sign Up

GreenVision AI

GreenVision AI leverages deep learning techniques and satellite imagery to assess urban green spaces, classify vegetation, and analyze environmental health indicators. This AI-powered system helps urban planners monitor and preserve green areas, contributing to a sustainable future.

By utilizing high-resolution satellite images and various vegetation indices, GreenVision AI provides real-time insights into the health of urban green spaces, helping address challenges related to urban heat islands, air quality, and biodiversity.

How to Use

1. Search for or locate the desired location on the map.
2. Take a cropped screenshot of the selected area by pressing the "PrtSc" key on your laptop.
3. Use the selection method (Rectangle or Pin) to mark the location on the map.
4. Upload the cropped screenshot of the selected area.
5. Click the "Submit" button to send the selected location and image snippet.

Rectangle Selection Pin Selection

NSBM Green University, Dampe Road, Mahenwatta, Dampe, Colombo District, Western Province, 10200, Sri Lanka

Search

Next Steps

You have selected a location. Please follow these steps:

1. Take a screenshot of the selected area on the map.
2. Upload the screenshot using the file input below.

Choose File Screenshot 2025-03-17 120228.png

Uploaded Image

Submit

IEMS Enhance urban sustainability with IEMS. © 2025 IEMS. All rights reserved.

IEMS Services Sign In Sign Up

Analysis Results

Vegetation Density Coverage

High Vegetation Density
Medium Vegetation Density
Low Vegetation Density

Green Percentage

Green Percentage 27.64%

NDVI Score

-0.4472000000000004
Normalized Difference Vegetation Index
Date: 19/03/2025
Area Covered: SUJIT
Health Status: Low Vegetation

The NDVI score ranges from -1 to 1, where values closer to 1 indicate healthier vegetation. This score helps in assessing the health and density of vegetation in the specified area.

Forecast

Forecast

Segmentation Results

contour original overlay predict

Recommendations

- Improving Vegetation Health Trend Detected: The forecast indicates an increase in vegetation health over time, which is a positive sign! This could be due to favorable growing conditions, effective land management, or recent environmental conservation efforts. To maintain and accelerate this growth:
 - Expand vegetation coverage by planting additional trees, shrubs, or crops
 - Optimize fertilization practices to ensure plants receive adequate nutrients
 - Encourage biodiversity by introducing native plant species and maintaining a balanced ecosystemYour efforts are yielding great results! Keep up the good work to sustain long-term vegetation health.
- Low NDVI Score: Your vegetation health appears poor. To improve plant vitality, consider enhancing soil quality with compost or fertilizers, optimizing irrigation schedules, and reducing soil compaction.
- Low Green Cover: The area has limited greenery. You may want to plant more trees, shrubs, or ground cover to enhance biodiversity and improve air quality.
- Low Vegetation Density: Over half of the area has sparse vegetation. Consider reforestation, afforestation, or introducing more drought-resistant plant species.

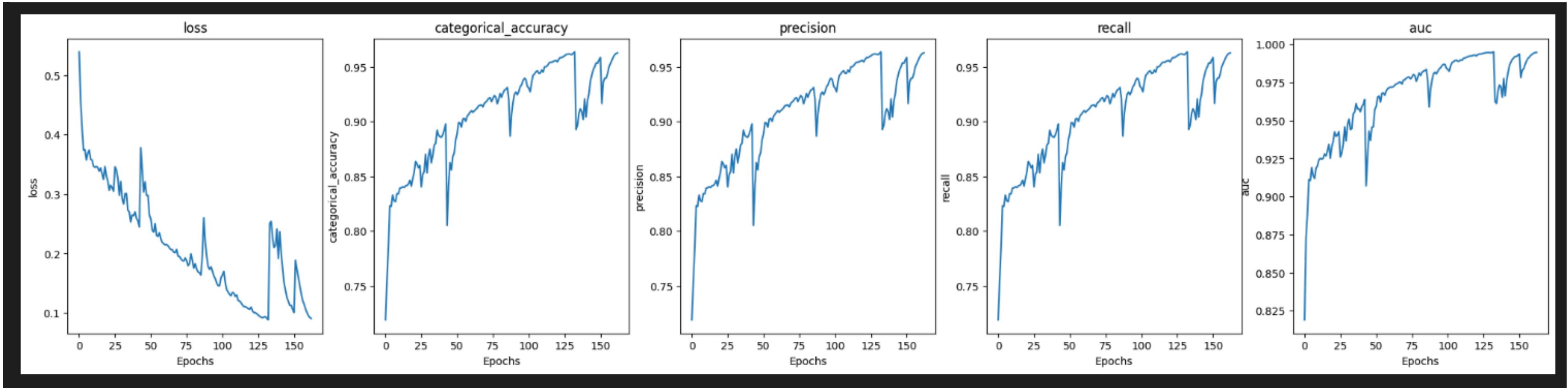
IEMS Enhance urban sustainability with IEMS. © 2025 IEMS. All rights reserved.

Image Masking



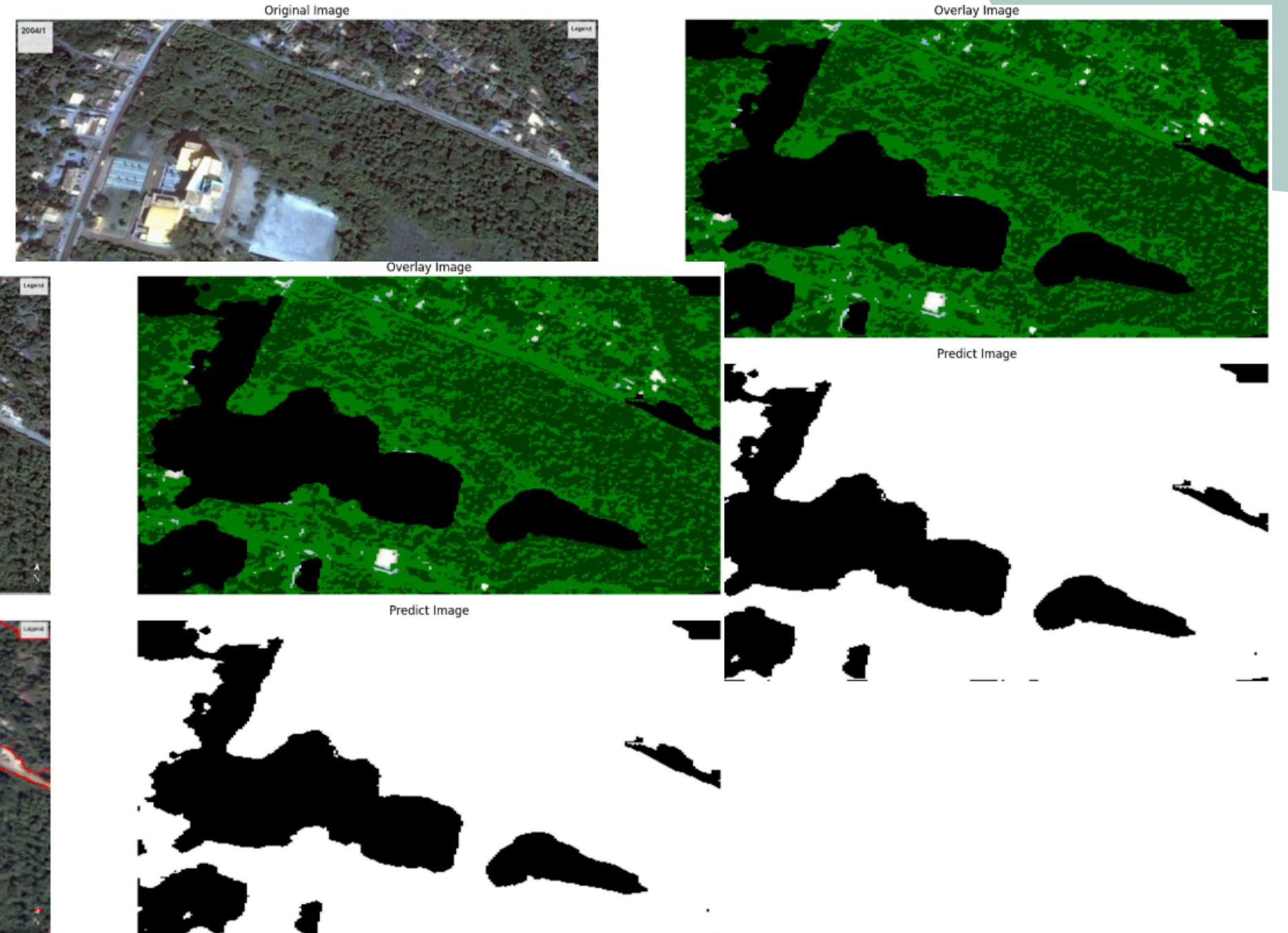
Accuracy

```
...
Epoch 162/200
17/17 [=====] - 4s 214ms/step - loss: 0.0922 - categorical_accuracy: 0.9621 - precision: 0.9621 - recall: 0.9621 - auc: 0.9946
Epoch 163/200
17/17 [=====] - 4s 215ms/step - loss: 0.0906 - categorical_accuracy: 0.9626 - precision: 0.9626 - recall: 0.9626 - auc: 0.9948
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```



Predictions

```
1/1 [=====]
1/1 [=====
Location ID : 0
1/1 [=====
{'forecast': {'month 1': '20.530',
  'month 2': '20.530',
  'month 3': '20.540'},
 'segmentation_results': {'contour': '2004/1',
  'original': '2004/1',
  'overlay': '2004/1',
  'predicted': '2004/1'},
 'segmentation_stats': {'Green P
  'High Vegetation Density Coverage': '3.35 %',
  'Low Vegetation Density Coverage': '55.73 %',
  'Medium Vegetation Density Coverage': '40.92 %',
  'NDVI Score': 0.02}}
```



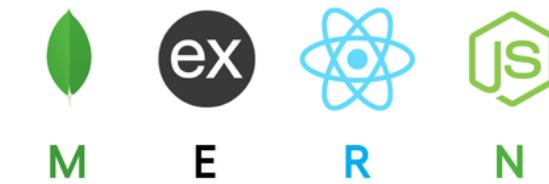
TOOLS & TECHNOLOGIES



TensorFlow



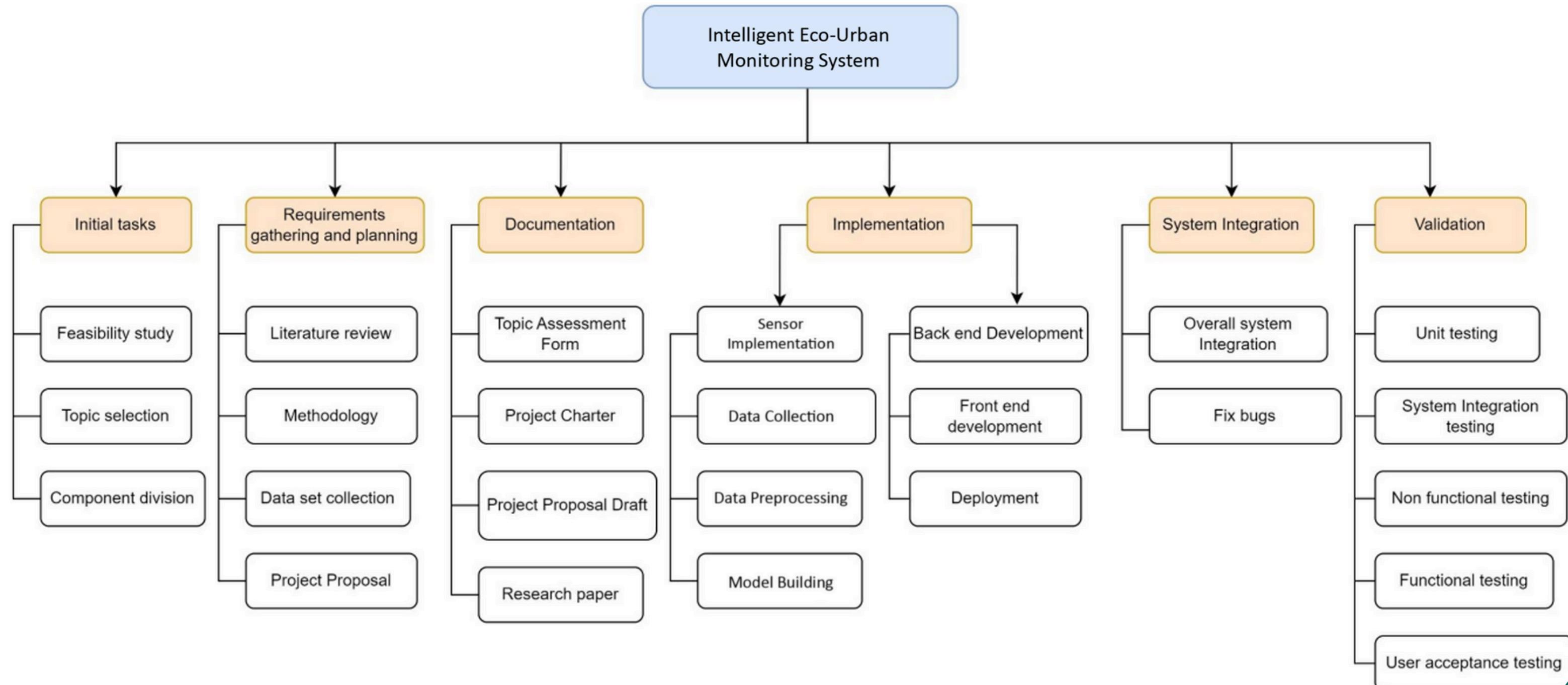
matplotlib



Requirements

-• **System Requirements**
 - High-resolution satellite imagery, High-Performance Computing
-• **Software Requirements**
 - **Functional Requirements**
 - Data Acquisition and Preprocessing
 - Vegetation Analysis and Classification
 - Temporal Analysis and Forecasting
 - Visualization and Reporting
 - **Non-Functional Requirements**
 - Performance
 - Scalability
 - Reliability
 - Usability

WORK BREAKDOWN CHART



PROJECT TIMELINE - GANTT CHART



References

- [1] Urban green spaces analysis for development planning in Colombo, Sri Lanka utilizing THEOS satellite imagery - A remote sensing and GIS approach, ResearchGate, 2013. [Online]. Available: [link](#).
 - [2] Application of satellite images and GIS in evaluation of green space destruction in urban area: Case study Boukan City, ResearchGate, 2012. [Online]. Available: [link](#).
 - [3] Green spaces and cognitive development in primary school children, National Center for Biotechnology Information (NCBI), 2022. [Online]. Available: [link](#).
 - [4] Urban green spaces, heat island effects and sustainable development: A review, IOPscience, 2021. [Online]. Available: [link](#).
- Urban green spaces and their potential to improve air quality in cities, Francis Press, 2022. [Online]. Available: [link](#).

Team Member 2



Arandara S. D.
IT21164330

EcoSensor AI
Air Quality Monitoring ,
Prediction and Recommondation Management

Background

Urbanization challenges air quality, vital for health and sustainability. EcoSensor AI uses IoT sensors for real-time pollutant monitoring and AI for analysis and predictions, offering recommendations to maintain optimal CO₂ levels.

Air Quality Monitoring and IoT Sensor Deployment

- Strategically place IoT sensors in urban areas to collect data on air pollutants like CO₂

Data Analysis and Prediction

- Use advanced machine learning models, such as Random Forest, Gradient Boosting, and LSTM networks, to analyze data and predict future air quality trends..

Recommendations for Air Quality Management

- EcoSensor AI includes predictive models that provide recommendations to maintain optimal CO₂ levels and overall air quality.





Research Problem

Urbanization challenges air quality and green space management, impacting public health and the environment. Traditional systems lack real-time data, future air quality predictions, and actionable recommendations.

Research Gap

Features	[1]	[2]	[3]	[4]	IEMS
Air Quality Monitoring	✓	✓	✗	✓	✓
IoT Integrated	✓	✓	✗	✓	✓
Recommendations	✗	✗	✗	✗	✓
Air Quality Prediction	✗	✗	✓	✗	✓
All in one Implement Web Application	✗	✗	✗	✗	✓
Report Generation	✗	✗	✗	✗	✓

[1] An IoT Based Air Pollution Monitoring System for Smart Cities

[2] Real-Time Air Quality Monitoring System using IoT

[3] Machine learning-based artificial intelligence method for predicting the air pollution index PM2.5

[4] An Integrated IoT and Machine Learning Approach for Environmental Monitoring and Management

Objectives

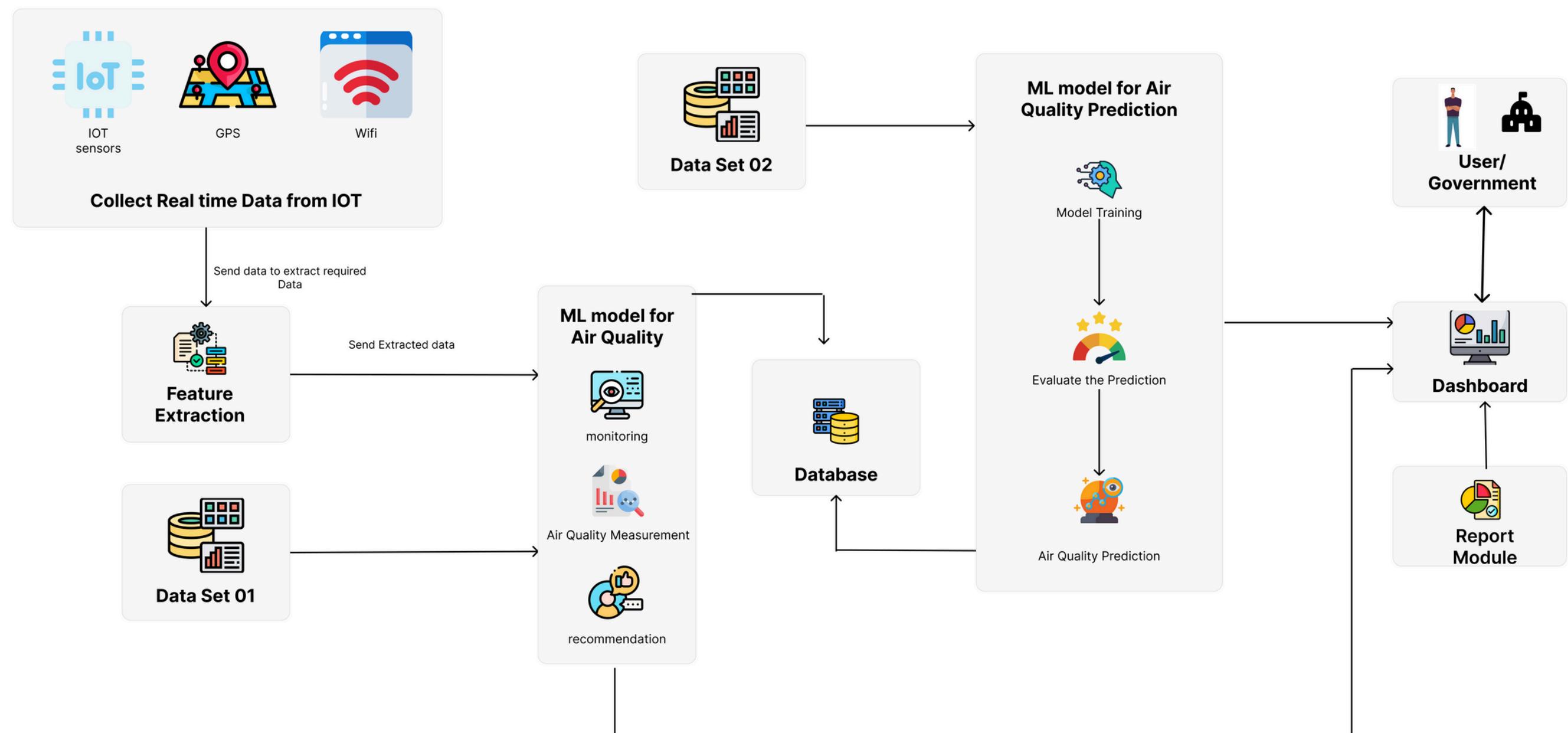
• Main Objectives

- Enhance urban sustainability through effective monitoring, prediction, and management of air quality using IoT sensors and AI.
- Generate actionable recommendations for air quality management based on predictive models.

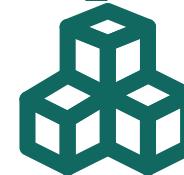
• Sub-Objectives

- Deploy IoT Sensors
- Develop Predictive Models
- Provide Recommendations
- Develop User Interface

System Diagram



Technologies Used



Machine Learning:

- Supervised Models such as Random Forest, Gradient Boosting

IoT Sensors:

- ESP-32 , MQ-138 , e.t.c

Data Visualization:

- D3.js, Chart.js

Programming languages :

- python , MERN stack , Flask

Current Progress



- Developed a Machine Learning Model for Air Quality Impact Classification
- Fine-Tuned Model Parameters using Optimization Techniques
- Implemented Backend API using Flask



- Upgraded to Deep Learning Approach for Impact Classification
- Designed and Built IoT Device for Real-time Air Quality Monitoring
- Collected & Preprocessed Data for Air Quality Prediction Model
- Developed Air Quality Prediction Model for Future Forecasting

Project Evidence

The screenshot shows the Postman API client interface. At the top, there are tabs for Home, Workspaces, and Explore. Below that, a search bar says "Search Postman". The main area displays a collection named "Welcome to Lightweight API Client". A POST request is selected with the URL "http://127.0.0.1:5000/predict_air_impact". The "Body" tab is active, showing a JSON payload:

```
1 "AQI": 87.2,
2   "date": "2024-12-20",
3   "latitude": 6.9731,
4   "longitude": 80.125
```

Below the body, the "Body" tab is again highlighted, showing the response in "Pretty" format:

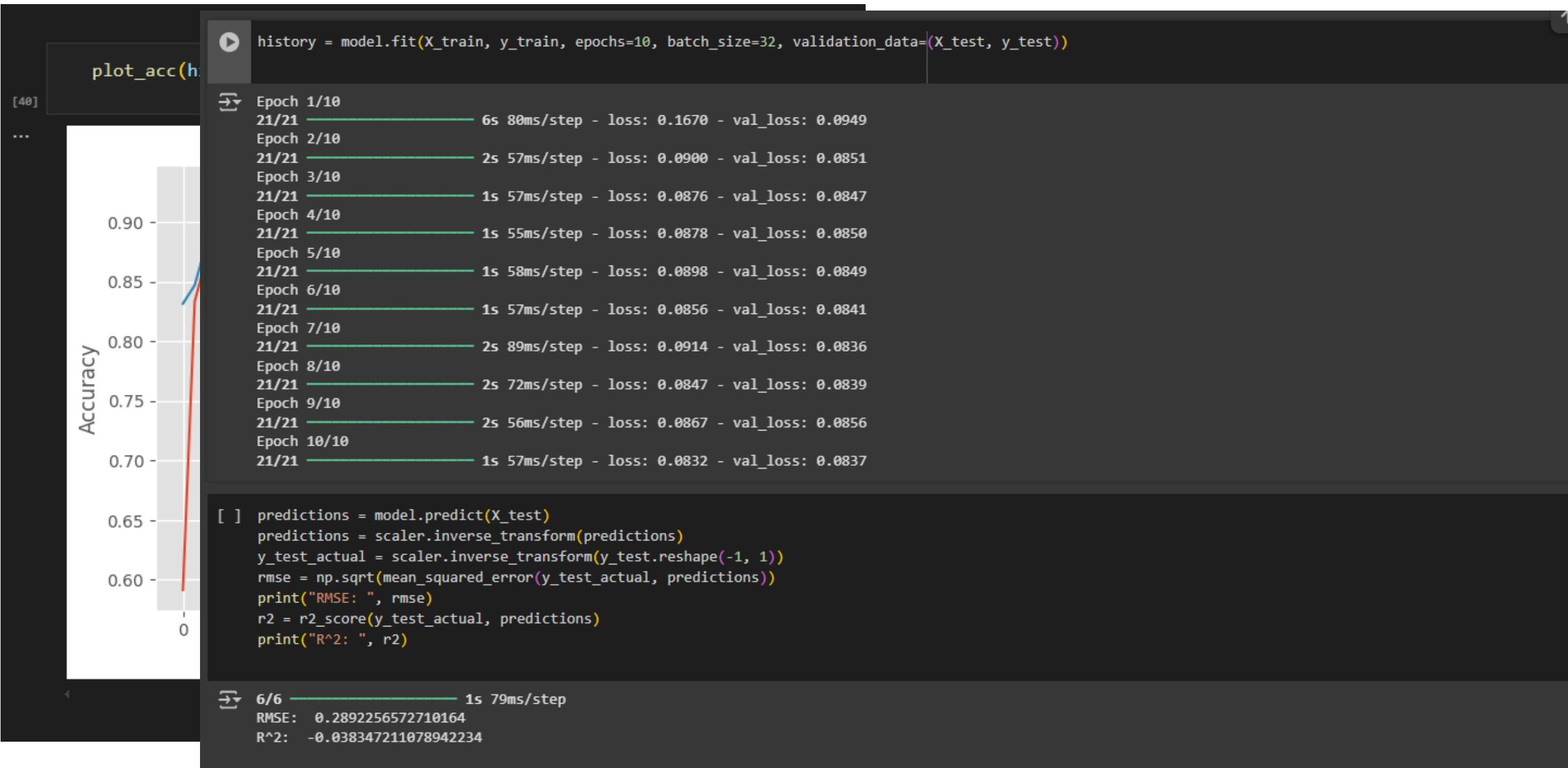
```
1   "predicted_aqi": 2,
2     "confidence": 0.502947211265564,
3     "healthImpactClass": "Moderate",
4     "prediction": 2
```

Data Distribution

df.head()

	RecordID	AQI	PM10	PM2_5	NO2	SO2	O3	Temperature	Humidity	WindSpeed	RespiratoryCases	CardiovascularCases	HospitalAdmissions	MortalityRate
0	1	187.270059	295.853039	13.038560	6.639263	66.161150	54.624280	5.150335	84.424344	6.137755	7	5	1	0.000000
1	2	475.357153	246.254703	9.984497	16.318326	90.499523	169.621728	1.543378	46.851415	4.521422	10	2	0	0.000000
2	3	365.996971	84.443191	23.111340	96.317811	17.875850	9.006794	1.169483	17.806977	11.157384	13	3	0	0.000000
3	4	299.329242	21.020609	14.273403	81.234403	48.323616	93.161033	21.925276	99.473373	15.302500	8	8	1	0.000000
4	5	78.009320	16.987667	152.111623	121.235461	90.866167	241.795138	9.217517	24.906837	14.534733	9	0	1	0.000000

Accuracy



A screenshot of a Jupyter Notebook cell. The cell contains Python code for training a model and plotting its accuracy. The output shows the training history with 10 epochs and a final accuracy plot.

```
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))

plot_acc(history)

[40]: ...
```

The plot shows Accuracy on the y-axis (ranging from 0 to 0.90) and Epochs on the x-axis (ranging from 1 to 10). A blue line represents training accuracy, which starts at approximately 0.60 and quickly rises to about 0.85 by epoch 10. A red line represents validation accuracy, which follows a similar initial rise but remains slightly lower than the training accuracy, ending around 0.83.

```
[ ] predictions = model.predict(X_test)
predictions = scaler.inverse_transform(predictions)
y_test_actual = scaler.inverse_transform(y_test.reshape(-1, 1))
rmse = np.sqrt(mean_squared_error(y_test_actual, predictions))
print("RMSE: ", rmse)
r2 = r2_score(y_test_actual, predictions)
print("R^2: ", r2)
```

```
6/6 1s 79ms/step
RMSE:  0.289225657210164
R^2:  -0.038347211078942234
```

Predictions

```
# Extract features from the date
day_of_week = date.weekday()
month = date.month
day = date.day
year = date.year

# Create a feature vector
features = np.array([day_of_week, month, day, year])

# Normalize the features
features_scaled = features / features.max()

return features_scaled

# Updated function
def make_prediction(date, latitude, longitude):
    # Preprocess the input
    features_scaled = preprocess_input(date, latitude, longitude)

    # Predict the AQI
    prediction = model.predict(features_scaled)
    predicted_aqi = prediction[0][0]

    # Inverse transform the prediction
    predicted_aqi = np.exp(predicted_aqi)

    return predicted_aqi

# Example usage with TensorFlow
date = "2024-12-20"
latitude = 6.9731
longitude = 80.125

# Make prediction
predicted_aqi = make_prediction(date, latitude, longitude)

# Output the prediction
print(f"Predicted AQI value for {date} at Latitude {latitude}, Longitude {longitude}: {predicted_aqi*1000}")

WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until
WARNING:tensorflow:5 out of the last 9 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distrib
1/1 ━━━━━━━━ 0s 40ms/step
Predicted AQI value for 2024-12-20 14:30:00 at Latitude 6.9731, Longitude 80.1256: 163.73797607421875
```

TOOLS & TECHNOLOGIES



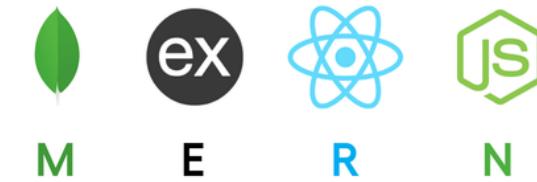
TensorFlow



python™



matplotlib



Requirements

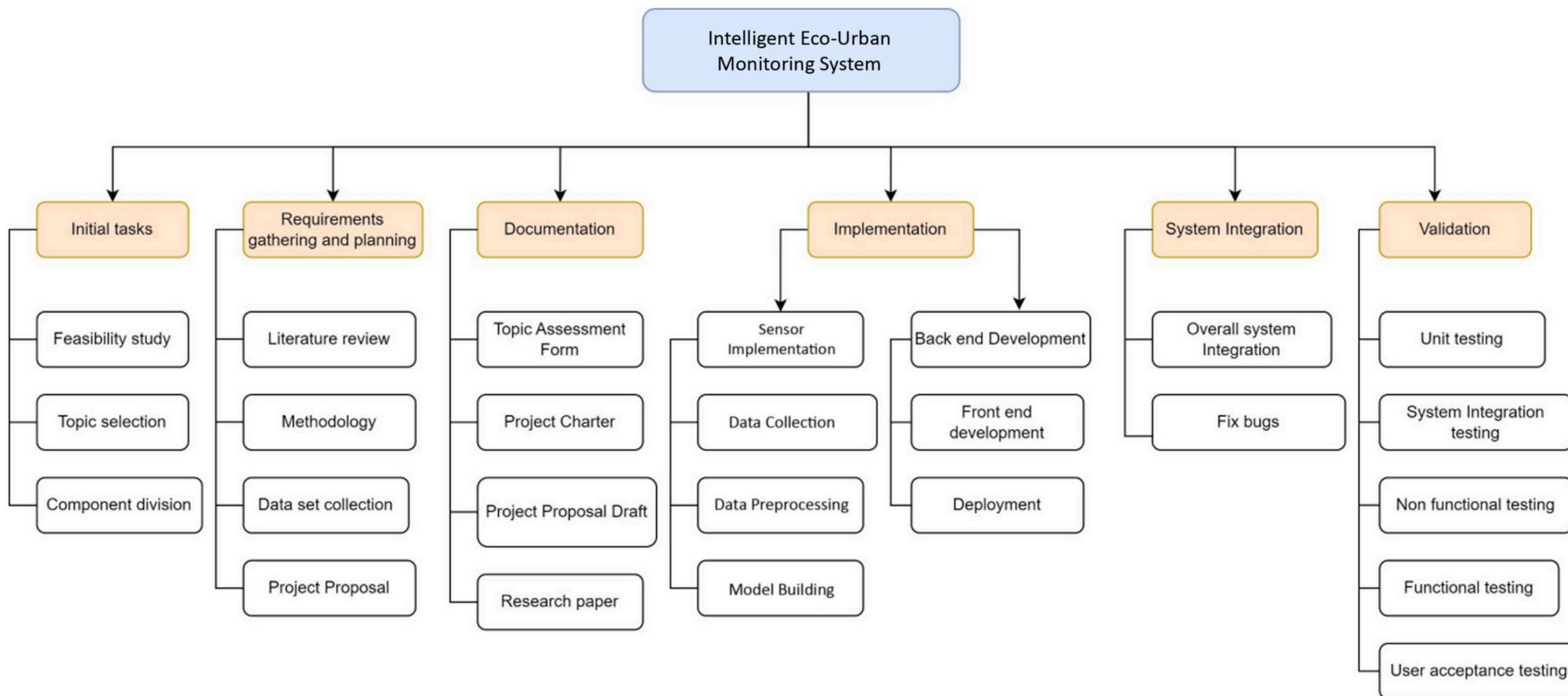
• **System Requirements**

- Hardware: IoT sensors for CO2 and PM2.5 data collection, high-performance computing for model training.
- Software: Machine learning frameworks (TensorFlow, PyTorch), data visualization tools.

• **Software Requirements**

- **Functional Requirements**
 - Air quality monitoring
 - Data analysis and predictions
 - Recommondations for good air quality
- **Non-Functional Requirements**
 - Performance
 - Scalability
 - Reliability
 - Usability

WORK BREAKDOWN CHART



PROJECT TIMELINE - GANTT CHART



References

- [1] V. Tikkwal et al., "An IoT Based Air Pollution Monitoring System for Smart Cities," ResearchGate, 2019. [Online]. [Link](#)
- [2] A. Gunathilaka et al., "Real-Time Air Quality Monitoring System using IoT," Journal of Physics: Conference Series, 2021. [Online]. [Link](#)
- [3] P. Wang et al., "Machine learning-based artificial intelligence method for predicting the air pollution index PM2.5" Journal of Cleaner Production, 2024. [Online]. [Link](#)
- [4] S. Kumar et al., "An Integrated IoT and Machine Learning Approach for Environmental Monitoring and Management," Environmental Research, 2021. [Online] [Link](#)



Noise Guard
AI

Team Member 3



Karunarathne R. Y. D.
IT21169144

Noise Guard AI

Smart Noise monitoring with IoT sensors,
Classification and Prediction

Background

Urban noise pollution harms public health and quality of life. Effective noise management is crucial for livable cities, yet current methods are inadequate due to fragmented data and limited analysis tools.



Uses AI to Analyze Noise Data for Better Management

- Leverages machine learning to analyze real-time noise levels using IoT data.

Classification and Prediction

- Uses advanced ML and DL models to classify noise sources and predict future trends based on historical data.

Informed Decision-Making

- Identifies high-noise areas and provides real time insights.



Research Problem

Urban areas struggle with managing noise pollution.

Need for a system to analyze and improve noise monitoring using advanced AI and IoT technologies.

Research Gap

IT21169144 | Karunarathne R. Y. D. | 24-25J-224

Features	[1]	[2]	[3]	[4]	IEMS
Future Predictions	✗	✗	✗	✓	✓
Noise Classification	✓	✓	✓	✗	✓
IOT Integrated	✓	✗	✓	✓	✓
Report Generation	✗	✗	✗	✗	✓
Implement Web Application	✗	✗	✗	✗	✓
Shown in Interactive Maps	✗	✗	✗	✗	✓

[1] Urban Sound Classification using CNN

[2] A Machine Learning Driven IoT Solution for Noise Classification in Smart Cities

[3] Internet of Things for Noise Mapping in Smart Cities: State-of-the-Art and Future Directions

[4] Noise Prediction Using Machine Learning with Measurement

Objectives

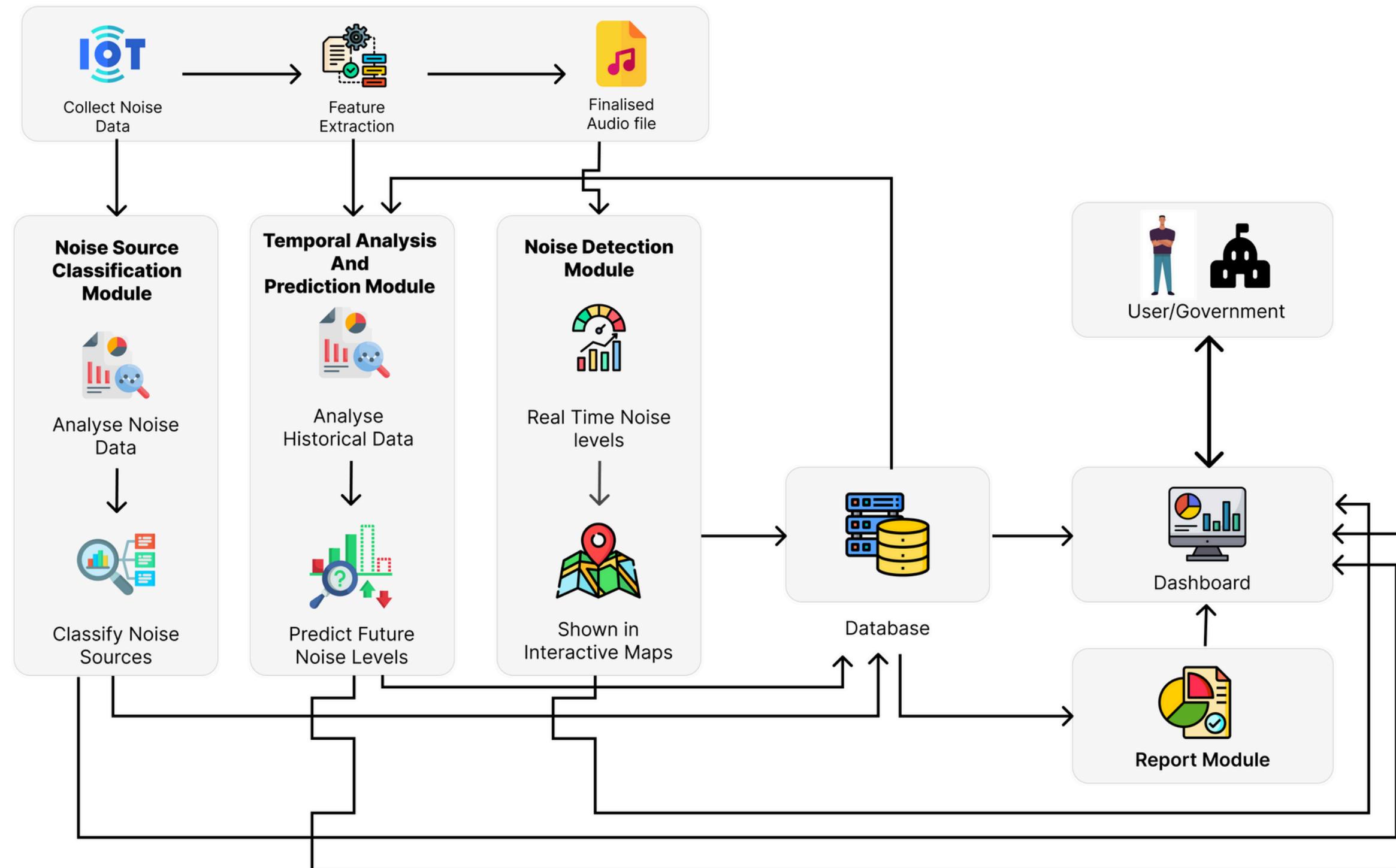
• Main Objectives

- Develop a comprehensive system for real-time
 - Noise Level Monitoring
 - Noise Classification
 - Prediction.

• Sub-Objectives

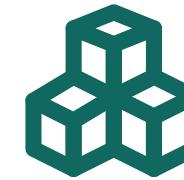
- Collect noise data using IoT and mobile devices.
- Classify noise sources using ML and DL models.
- Predict future noise levels based on historical data.

System Diagram



Technologies Used

- **IoT, Mobile phone Microphone**
 - For real-time noise data collection.
 - GPS Sensor, MAX4466 Audio Sensor, ESP-32
- **ML/DL Models**
 - CNNs for classification and prediction, EfficientNetB0



1. **Data Collection:** Deploy IoT sensors in various locations and Pre Data-sets
2. **Pre-processing:** Clean and preprocess collected data.
3. **Model Training:** Train ML/DL models for classification and prediction.
4. **System Integration:** Integrate models with real-time data for monitoring.

Current Progress

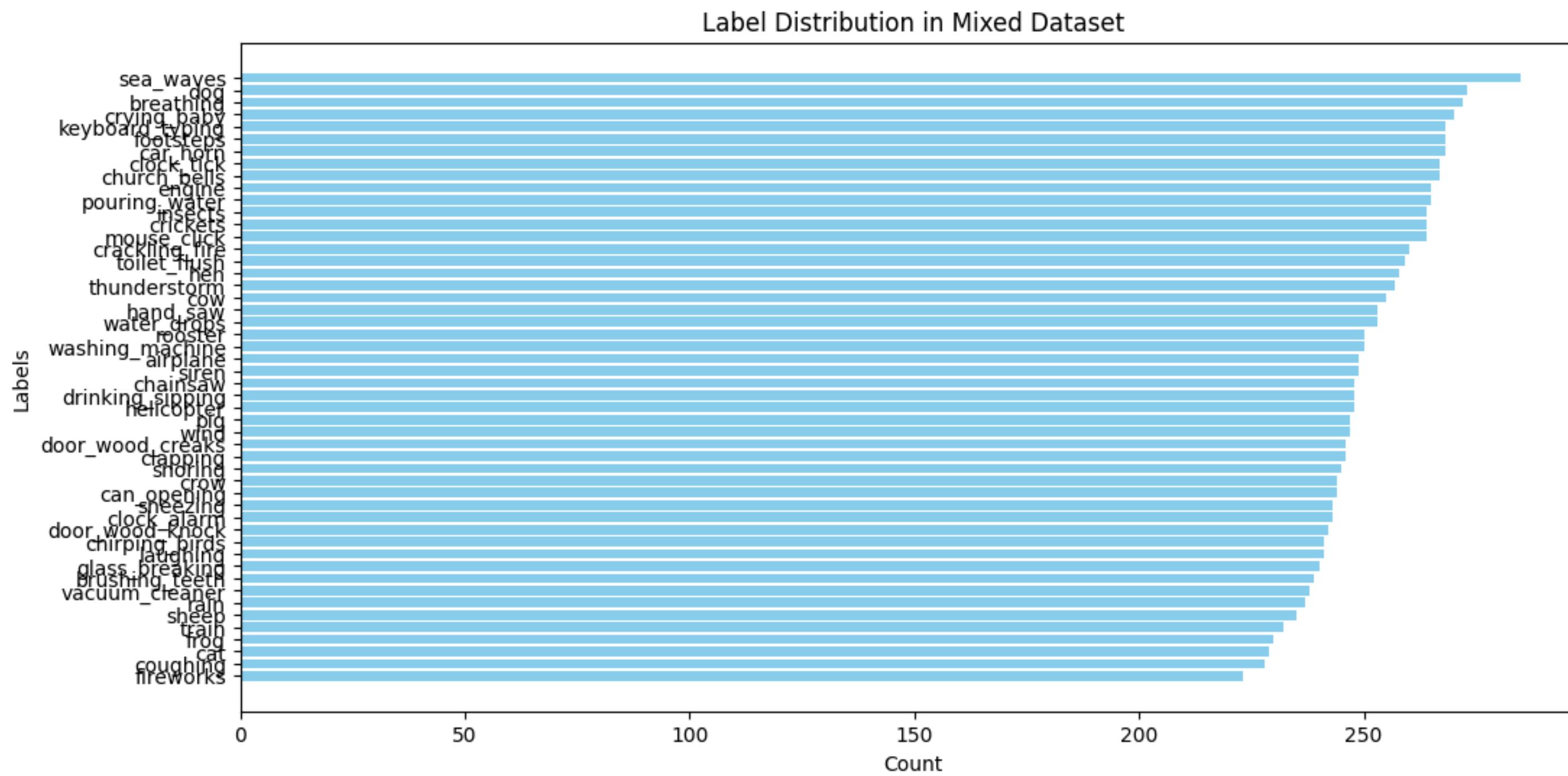


- Trained an initial model to predict single classification.
- Used a CNN model for audio classification.
- Converted audio into Mel spectrograms for feature extraction.
- Began fine-tuning hyperparameters to improve the accuracy and reliability of feasibility predictions.

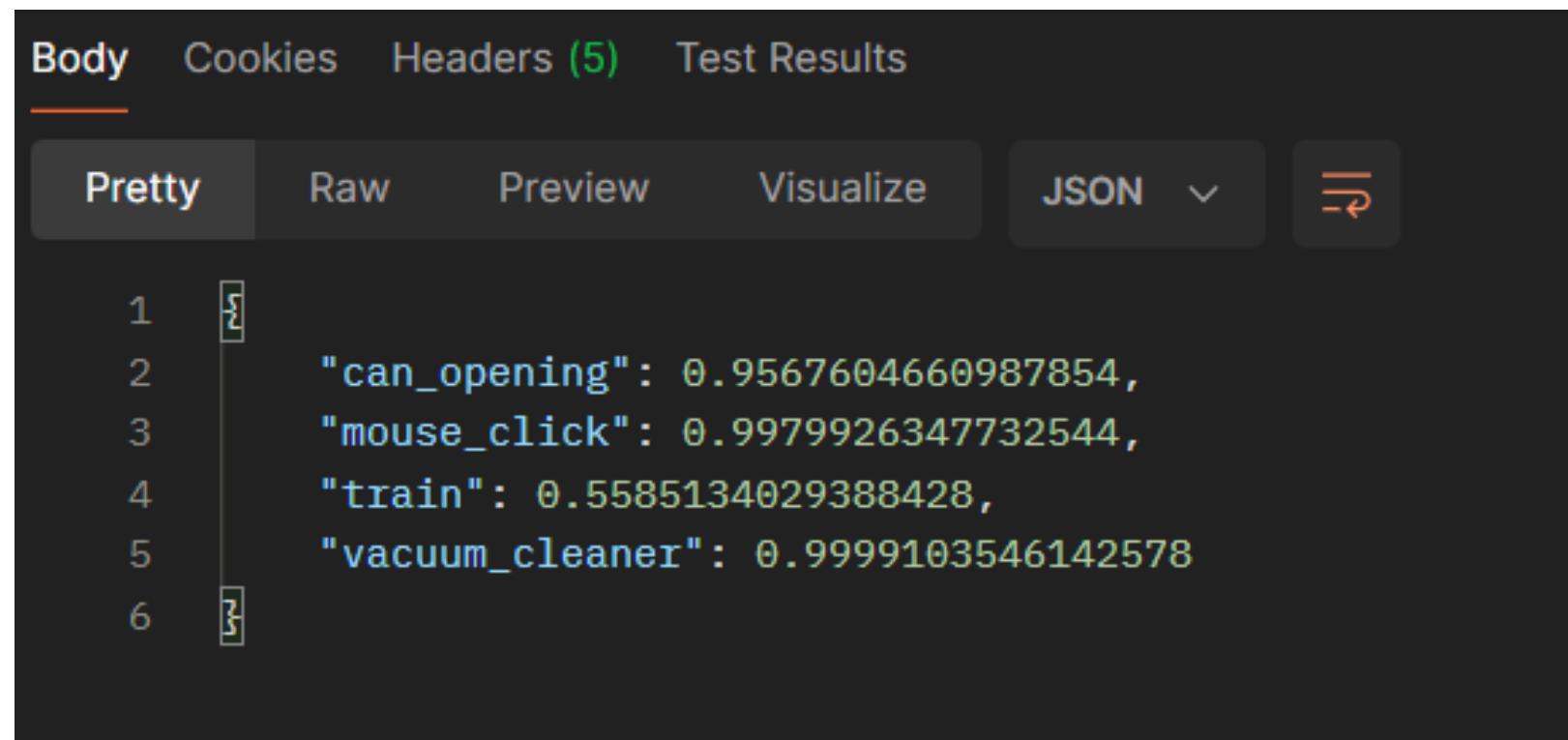


- Implemented IoT sensors to collect real-world audio data along with geo-location metadata.
- Collected real-world audio samples and mixed them with an existing dataset to create a new dataset.
- Trained a multi-label classification model using CNN.
- Trained the model with 50 different classes for improved classification performance.
- Began fine-tuning hyperparameters to improve the accuracy and reliability of feasibility predictions.

Data Distribution



Project Evidence



A screenshot of a browser developer tools interface showing a JSON response. The top navigation bar includes tabs for 'Body' (which is selected), 'Cookies', 'Headers (5)', and 'Test Results'. Below the tabs are buttons for 'Pretty' (selected), 'Raw', 'Preview', 'Visualize', and a 'JSON' dropdown. The main content area displays a JSON object with numbered lines 1 through 6. Line 1 starts with '{', line 2 contains 'can_opening': 0.9567604660987854, line 3 contains 'mouse_click': 0.9979926347732544, line 4 contains 'train': 0.5585134029388428, line 5 contains 'vacuum_cleaner': 0.9999103546142578, and line 6 ends with '}'.

```
1  {
2      "can_opening": 0.9567604660987854,
3      "mouse_click": 0.9979926347732544,
4      "train": 0.5585134029388428,
5      "vacuum_cleaner": 0.9999103546142578
6 }
```

Accuracy

		precision	recall	f1-score	support
	car_horn	0.84	0.81	0.83	47
	engine	0.56	0.75	0.64	44
	brushing_teeth	0.71	0.63	0.67	57
	breathing	0.69	0.47	0.56	47
	glass_breaking	0.90	0.51	0.65	53
	door_wood_creaks	0.72	0.65	0.68	43
	clock_tick	0.34	0.38	0.36	50
	laughing	0.39	0.70	0.50	40
	frog	0.77	0.75	0.76	44
	chirping_birds	0.94	0.74	0.83	42
	thunderstorm	0.76	0.71	0.74	59
	hand_saw	0.60	0.67	0.63	54
	can_opening	0.43	0.47	0.44	43
	crow	0.90	0.85	0.87	52
	train	0.72	0.59	0.65	49
	water_drops	0.33	0.18	0.23	51
	siren	0.93	0.91	0.92	47
	helicopter	0.60	0.49	0.54	51
	chainsaw	0.91	0.90	0.91	48
	sneezing	0.70	0.50	0.58	52
	snoring	0.51	0.46	0.48	46
	pig	0.81	0.61	0.70	36
	dog	0.79	0.52	0.63	52
	rooster	0.76	0.81	0.78	47
	coughing	0.77	0.47	0.58	49
	keyboard_typing	0.49	0.42	0.45	59
	vacuum_cleaner	0.92	0.98	0.95	58
	footsteps	0.59	0.48	0.53	46
	insects	0.78	0.66	0.71	47
	crickets	0.87	0.81	0.84	42

Predictions

```
▶ with open("labels.json", "r") as f:  
    loaded_labels = json.load(f)  
  
    # Load model  
    loaded_model = load_model("multi_label_sound_classification_model.keras", compile=False)  
    loaded_model.compile(optimizer=Adam(learning_rate=0.0005), loss="binary_crossentropy", metrics=["accuracy"])  
  
    def predict_audio(audio_path, model, labels, threshold=0.1):  
        mel_spec = audio_to_mel_spectrogram(audio_path, target_shape=(128, 128))  
        mel_spec = np.expand_dims(mel_spec, axis=-1)  
        mel_spec = np.expand_dims(mel_spec, axis=0)  
        y_pred = model.predict(mel_spec)[0]  
        predicted_labels = {label: float(prob) for label, prob in zip(labels, y_pred) if prob > threshold}  
        return dict(sorted(predicted_labels.items(), key=lambda item: item[1], reverse=True))  
  
    # Test prediction  
    test_audio_path = "/content/drive/MyDrive/mixed_dataset/mixed_1023.wav"  
    predictions = predict_audio(test_audio_path, loaded_model, loaded_labels)  
    print(predictions)  
  
→ 1/1 ━━━━━━━━ 7s 7s/step  
{'brushing_teeth': 0.9999773502349854, 'hand_saw': 0.9985563158988953, 'mouse_click': 0.9757606983184814}
```

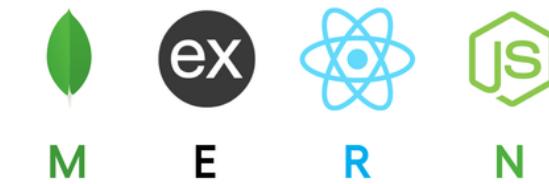
TOOLS & TECHNOLOGIES



TensorFlow



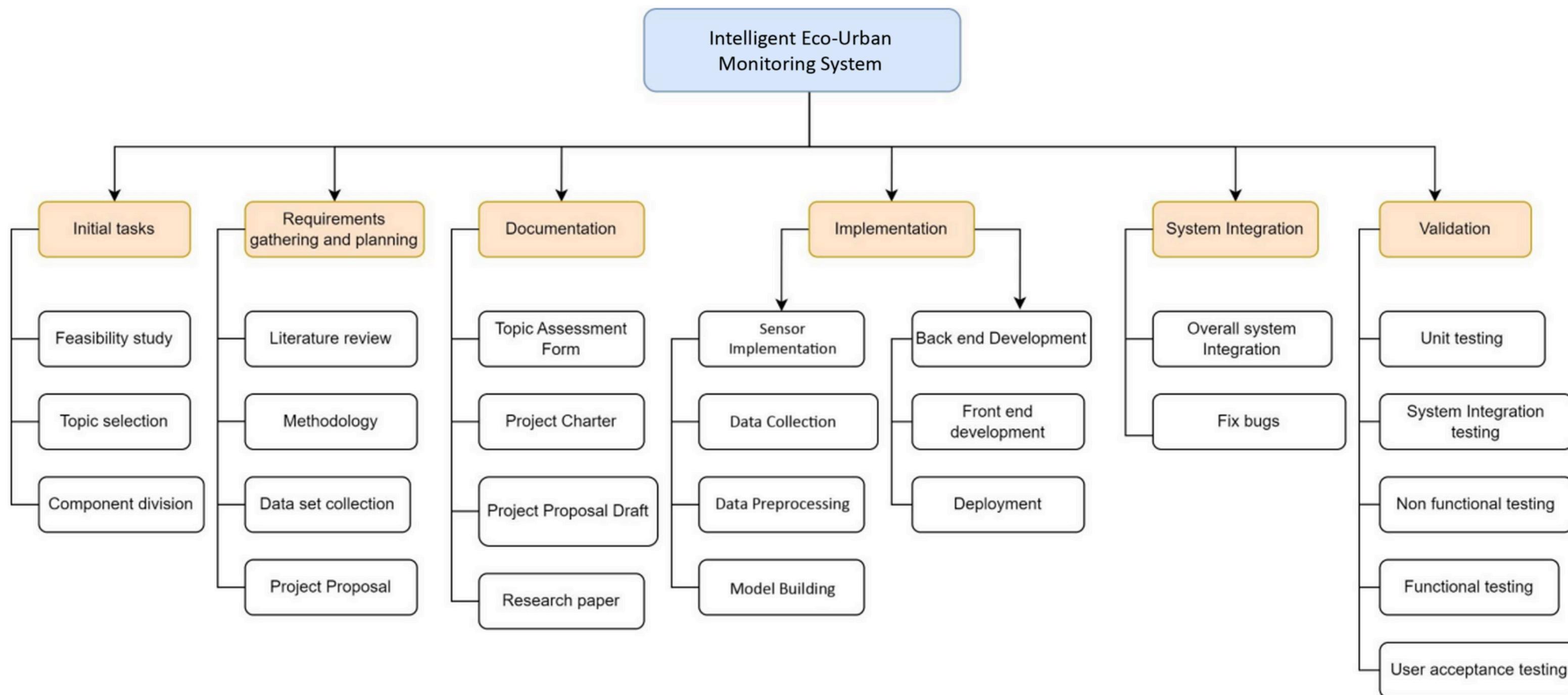
matplotlib



System Requirements

-• **System Requirements**
 - **Hardware:** IoT sensors, mobile phone's microphone for noise data collection, high-performance computing for model training.
 - **Software:** TensorFlow, PyTorch, data visualization tools.
-• **Software Requirements**
 - **Functional Requirements**
 - Noise Data Collection
 - Noise Source Classification
 - Temporal Analysis And Prediction Module
 - **Non-Functional Requirements**
 - Performance
 - Scalability
 - Reliability
 - Usability

WORK BREAKDOWN CHART



PROJECT TIMELINE - GANTT CHART



References

- [1] M. A. Basuni and S. H. Alzahrani, "Urban Sound Classification using CNN," ResearchGate, 2021. [Online]. Available: [Link](#)
- [2] M. F. Khan, M. Al-Kahtani, S. H. Islam, N. Kumar, and P. W. C. Prasad, "A Machine Learning Driven IoT Solution for Noise Classification in Smart Cities," ResearchGate, 2018. [Online]. Available: [link](#)
- [3] D. Gómez-Gutiérrez, J. Pascual-Gaspar, J. M. Lanza-Gutiérrez, and E. Sanchristobal, "Internet of Things for Noise Mapping in Smart Cities: State-of-the-Art and Future Directions," ResearchGate, 2020. [Online]. Available: [link](#)
- [4] S. A. Kumar, K. R. Sudheesh, and S. M. S. Islam, "Noise Prediction Using Machine Learning with Measurement," ResearchGate, 2020. [Online]. Available: [link](#)



Team Member 4



Kodithuwakku C.K.
IT21156960

Eco go
Predicting and Reducing
CO2 Emissions from Vehicles.

Background

Eco Go, a component of the Integrated Environmental Monitoring System (IEMS), leverages advanced machine learning to predict and reduce vehicle CO₂ emissions, offering personalized strategies and insights for urban emission challenges.

Predictive Emission Analysis

- Uses machine learning to predict vehicle CO₂ emissions by analyzing vehicle attributes.
- Use two models to predict CO₂ emissions accurately.

Personalized Recommendation Engine

- Provides tailored CO₂ reduction strategies based on individual driving habits and vehicle characteristics.
- Use TOPSIS algorithm to recommend best vehicle





Research Problem

Urban areas experience significant CO2 emissions from vehicles, contributing to climate change and air pollution.

A system is needed that uses advanced machine learning to predict CO2 emissions and offer customized reduction strategies for each driver.

Research Gap

Features	[1]	[2]	[3]	[4]	IEMS
Use two ML models	✗	✗	✗	✗	✓
Adaptive Learning and Strategy Optimization	✗	✗	✓	✗	✓
Personalization	✓	✗	✗	✗	✓
Broader Vehicle and Condition Coverage	✗	✗	✗	✗	✓
Comprehensive System Integration	✓	✗	✗	✗	✓

[1] Modelling of CO2 Emission Prediction for Dynamic Vehicle Travel Behavior

[2] From lab-to-road & vice-versa

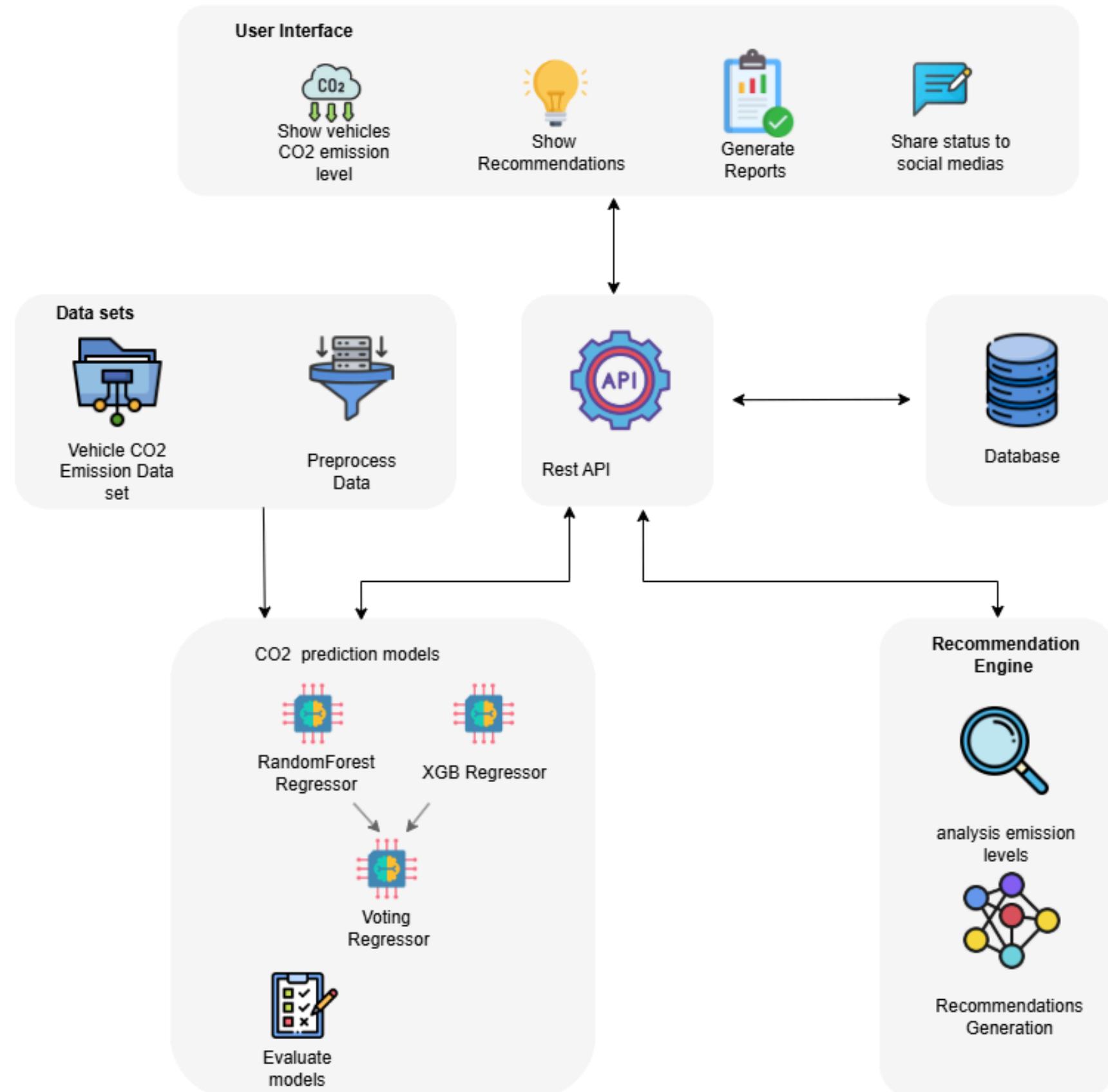
[3] Estimation of transport CO2 emissions using machine learning algorithm

[4] Models for predicting vehicle emissions

Objectives

-• **Main Objective**
 - Improve urban sustainability by predicting vehicle CO2 emissions and delivering personalized recommendations for effective emission reduction.
-• **Sub-Objectives**
 - Predict Vehicle CO2 emission.
 - Develop personalized strategies.
 - Develop a recommendation engine that suggest best vehicle

System Diagram



Technologies Used

Machine Learning

- Ensemble learning models (Random Forest, Gradient Boosting)
- Metrics such as RMSE,R2
- LLM such as llama 3.3

API

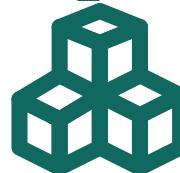
- Here map

Data Visualization

- D3.js, Chart.js

Programming languages

- python , MERN stack , Flask



Current Progress



- Trained ML model to predict CO₂ emissions using ensemble techniques.
- Implemented Eco Vehicle Suggestion using a Multi-Criteria Decision Algorithm.
- Developed Flask backend to expose APIs for CO₂ prediction and eco vehicle suggestion.



- Implemented CO₂ emission reduction recommendations using Retrieval-Augmented Generation (RAG) with LLaMA 3.3 LLM.
- Designed and developed UIs for the EcoGo component and integrated them with the Flask backend.
- Implemented Eco Navigator Map, which provides the best routes to reduce CO₂ emissions.

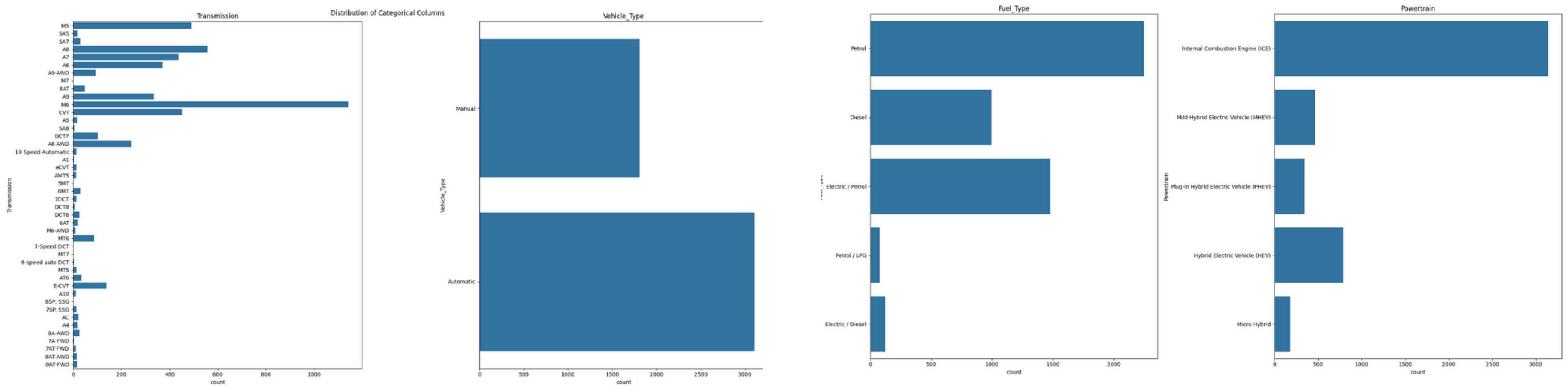
Project Evidence

The screenshot shows a dark-themed web application interface for managing a fleet of vehicles. At the top, there is a navigation bar with the 'IEMS' logo, a 'Services' dropdown, a user profile for 'kavinda', and a 'Logout' button. Below the navigation, a message says 'Showing 4 vehicles'. A green button labeled 'Show Best Eco Vehicle' is visible. The main content area displays four vehicle cards:

- Land cruiser** (2000):
 - Vehicle Type: SUV
 - CO2 Emission: 139 g/km
 - Emission Level: Medium (yellow bar)
 - Actions: Edit, Delete, View More, Predict CO2 Emission
- Vizel** (2018):
 - Vehicle Type: SUV
 - CO2 Emission: 120 g/km
 - Emission Level: Low (green bar)
 - Actions: Edit, Delete, View More, Predict CO2 Emission
- Vitz** (2016):
 - Vehicle Type: Hatchback
 - CO2 Emission: 114 g/km
 - Emission Level: Low (green bar)
 - Actions: Edit, Delete, View More, Predict CO2 Emission
- fit** (2017):
 - Vehicle Type: Sedan
 - CO2 Emission: 115 g/km
 - Emission Level: Low (green bar)
 - Actions: Edit, Delete, View More, Predict CO2 Emission

At the bottom of the page, there is a footer with the 'IEMS' logo, the tagline 'Enhance urban sustainability with IEMS.', social media links (Facebook, Twitter, Instagram, LinkedIn), and a copyright notice: '© 2025 IEMS. All rights reserved.'

Data Distribution



Accuracy



Predictions

```
def inference_co2(  
    sample_json,  
    cat_cols = [  
        'Transmission',  
        'Vehicle_Type',  
        'Fuel_Type',  
        'Powertrain'  
    ]):  
    df = pd.DataFrame([sample_json])  
    df = df.astype(str)  
    for col in df.columns:  
        if col in cat_cols:  
            df[col] = df[col].str.strip()  
            df[col] = encoder_co2[col].transform(df[col])  
    x = df.values  
    Ypred = model_co2.predict(x)  
    return int(Ypred[0])
```

```
inference_co2(sample_json)
```

688

TOOLS & TECHNOLOGIES



TensorFlow



matplotlib

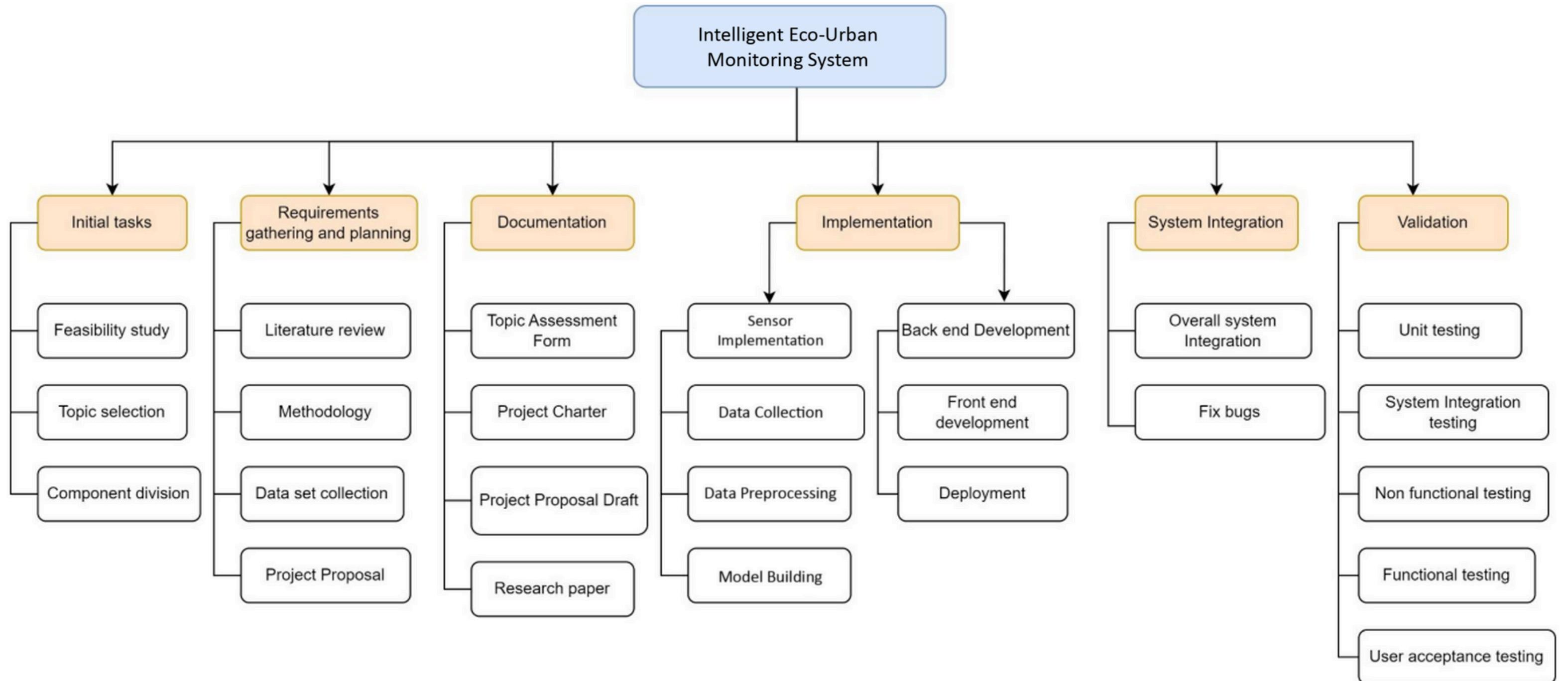


Requirements



-• **System Requirements**
 - High-Performance Computing
-• **Software Requirements**
 - Functional Requirements
 - CO2 Prediction Model
 - Recommendation Engine
 - Visualization and Reporting
 - Non-Functional Requirements
 - Performance
 - Scalability
 - Reliability
 - Usability

WORK BREAKDOWN CHART



PROJECT TIMELINE - GANTT CHART



References

- [1] Navarajan Subramaniam, Norhakim Yusof, "Modelling of CO2 Emission Prediction for Dynamic Vehicle Travel Behavior Using Ensemble Machine Learning Technique," IEEE, 25 November 2021. [Online]. Available: [link](#). (Accessed: Jul. 22, 2024.)
- [2] S. Tsiakkas, G. Fontaras, J. Dornoff, and V. Valverde, "From lab-to-road & vice-versa: Using a simulation-based approach for predicting real-world CO2 emissions," Energy, vol. 177, pp. 495–508, 15 February 2019. [Online]. Available: [link](#). (Accessed: Jul. 22, 2024.)
- [3] S. Li, Z. Tong, and M. Haroon, "Estimation of transport CO2 emissions using machine learning algorithm," Transportation Research Part D: Transport and Environment, vol. 122, August 2024. [Online]. Available: [link](#). (Accessed: Jul. 23, 2024.)
- [4] H. Zhong, K. Chen, C. Liu, and M. Zhu, "Models for predicting vehicle emissions: A comprehensive review," Science of The Total Environment, vol. 791, 1 May 2024. [Online]. Available: [link](#). (Accessed: Jul. 23, 2024.)

Commercialization

Market Demand

- Market demand analysis involves understanding the need for the Intelligent EcoUrban Monitoring System (IEMS) in the market. This includes identifying the target audience, market size, growth trends, and potential market opportunities.

Target Audience

- Municipalities and city planners ,Environmental agencies , Goverment , Citizens

Market Opportunities

- Integration with existing smart city infrastructure. ,Partnership with governments for large-scale implementation , Custom solutions for different urban environments.



24-25J-224

THANK YOU !





24-25J-224

Q & A

