CERTIFICATE

This is to certify that this project entitled **"EcoPulse : Predicting Renewable and Non-renewable energy"** is done by **Ms. Kashish Malik (07817002023)** is an authentic work carried out for the partial fulfillment of the requirements for the award of degree of Bachelor of Computer Applications under the guidance of **Dr.Sushma Bahuguna and Dr.Neha**. The matter embodied in this project has not been submitted earlier for the award of any degree or diploma to the best of my knowledge and belief.

Signature of the student **Signature of the Guide**

SELF CERTIFICATE

This is to certify that the dissertation/project report entitled "EcoPulse : Predicting Renewable and Non-renewable energy " is done by me is an authentic work carried out for the partial fulfilment of the requirements for the award of the degree of Bachelor of Computer Applications under the guidance of Dr.Sushma Bahuguna and Dr. Neha. The matter embodied in this project work has not been submitted earlier for award of any degree or diploma to the best of my knowledge and belief.

Signature of the Student

Name of the Student: KASHISH MALIK

Enrollment No: 07817002023

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to Dr.Sushma Bahuguna (Assistant Professor) and Dr. Neha (Associate Professor) for their invaluable guidance and support throughout the course of this project Data Driven Analysis of Solid Waste Generation and its Environmental effects_ . Their unwavering dedication to fostering a nurturing learning environment and her commitment to excellence have been instrumental in shaping my academic and professional journey. I am truly grateful and fortunate to have had the privileged of working under the mentorship of Mr Suresh Panchal (Assistant Professor) and Dr Ratandeep Kaur (Associate Professor) . their leadership and encouragement have not only facilitated the development of my technical skills but have also inspired to strive for excellence in all my endeavors.

Kashish Malik
(07817002023)
BCA 5th Sem
1st Shift

SYNOPSIS

Title of the Project : EcoPulse : Predicting Renewable and Non - renewable energy

Growing Energy Needs: Renewable and Non-Renewable Energy Resources

Abstract:

The world's energy demand is increasing rapidly due to population growth, industrialization, urbanization, and technology-driven lifestyles. To meet this growing demand, most countries still rely heavily on **non-renewable energy resources** such as coal, oil, and natural gas. While these sources are highly effective in power generation, they are finite and cause severe environmental impacts like air pollution, greenhouse gas emissions, global warming, and ecological degradation.

At the same time, **renewable energy resources** such as solar, wind, hydro, geothermal, and biomass are emerging as sustainable alternatives. However, awareness about their long-term benefits, usage potential, and environmental advantages is still limited among common users, students, and even small-scale decision makers.

This project aims to develop a web-based **Energy Awareness and Analytics System** that provides an interactive platform to study **growing energy needs** and compare **renewable and non-renewable energy resources**. The system analyzes sample or real-world energy consumption data, visualizes it using interactive charts, and explains the environmental and ethical implications of different energy choices.

The application uses **HTML, CSS, and JavaScript** for the frontend and **Python (Flask/Django)** for backend data processing. Energy data is stored in **JSON/CSV or MySQL** for easy retrieval, filtering, and visualization. The system includes:

- Source-wise and sector-wise energy usage charts
- Comparison between renewable and non-renewable energy shares
- Time-based trend analysis
- Awareness content on energy conservation and sustainability

The project demonstrates how **IT-based tools** and **data visualization** can be used to promote sustainable energy usage, educate users, and support decision-making towards cleaner and more responsible energy consumptio

Overview:

With rising global energy demand driven by industrialization, population growth, and expanding technology-driven lifestyles, dependence on non-renewable energy sources remains dominant. While fossil fuels such as coal, petroleum, and natural gas provide significant energy supply, they contribute to environmental degradation, greenhouse gas emissions, climate change, and long-term ecological impact. In contrast, renewable energy sources—solar, wind, hydro, geothermal, and biomass—emerge as environmentally sustainable alternatives but still lack widespread understanding and adoption.

Project Description:

This project proposes the development of **EcoPulse**, a web-based Energy Analytics and Awareness System that enables users to analyze, compare, and visualize renewable vs non-renewable energy consumption using interactive dashboards. Using technologies such as **HTML, CSS, JavaScript, Python, and Chart.js**, the system displays energy patterns through graphical representations and facilitates better understanding of long-term sustainability implications.

The platform provides:

- Source-wise and sector-wise energy consumption analytics
- Visualization of energy mix using pie/bar/line charts

- Time-based trend analysis
- Educational content regarding environmental ethics and sustainability
- Practical energy-saving tips for users

Objectives:

1. To visualize and analyze energy consumption data across various sources.
2. To compare renewable and non-renewable resources in terms of usage and environmental impact.
3. To present time-based trends for energy demand.
4. To provide awareness and educational insights regarding sustainability and energy ethics.
5. To support informed decision-making and environmental responsibility through data visualization.

Scope:

The system uses simulated or real datasets and can be extended to incorporate:

- Real-time API-based energy data
- Machine learning-based demand forecasting
- Mobile application integration
- Multi-country energy comparison
- Carbon footprint analysis

Conclusion:

EcoPulse demonstrates how data visualization and IT-based tools can be leveraged to inform and educate users about sustainable energy consumption. By presenting complex energy data in an intuitive digital format, this platform encourages responsible energy behavior, promotes renewable energy awareness, and supports global efforts toward environmental sustainability.

LIST OF FIGURES

LIST OF ABBREVIATIONS

Sr. No.	Abbreviation name	Description
1	FR	Functional Requirements
2	NFR	Non-Functional Requirements
3	UI	User Interface
4	HTML	HyperText Markup Language
5	CSS	Cascading Style Sheets
6	JS	JavaScript
7	DOM	Document Object Model
8	SVG	Scalable Vector Graphics
9	CPU	Central Processing Unit
10	RAM	Random Access Memory
11	OS	Operating System
14	UX	User Experience

INDEX

CHAPTER 1 – INTRODUCTION

1.1 Brief Description of Organization

Tecnia Institute of Advanced Studies (TIAS), affiliated with Guru Gobind Singh Indraprastha University (GGSIPU), is committed to delivering quality education in Information Technology, Computer Applications, and Management. As part of the **BCA curriculum**, students are required to undertake a minor project to bridge theoretical concepts with practical implementation.

This project, titled **"Growing Energy Needs: Renewable and Non-Renewable Energy Resources"**, has been designed and implemented under the guidance of the faculty members of TIAS. It aims to demonstrate how IT tools can be effectively utilized to study energy trends and promote sustainable energy practices.

1.2 Introduction to Energy and Environmental Ethics

Energy is a fundamental requirement for all modern activities, including transportation, industry, communication, healthcare, and domestic life. The rising standard of living and technological growth has led to a massive increase in **energy demand**.

Energy resources are broadly classified into:

- **Non-Renewable Energy Resources**: Coal, petroleum, natural gas, nuclear fuels – finite in nature and associated with pollution and greenhouse gas emissions.

- **Renewable Energy Resources**: Solar, wind, hydro, geothermal, tidal, biomass – naturally replenished and comparatively cleaner and more sustainable.

From the perspective of **environmental ethics**, the way we produce and consume energy directly influences climate change, resource depletion, intergenerational justice, and the rights of other living beings. Responsible energy consumption is not only a technical or economic issue but also a **moral obligation** towards future generations and the planet.

1.3 Problem Description

The world is facing a dual challenge:

1. **Rapidly growing energy needs** due to urbanization, industrialization, and digital lifestyles.

2. **Over-dependence on non-renewable resources**, leading to:
    - Greenhouse gas emissions
    - Global warming and climate change
    - Resource depletion
    - Environmental degradation

Most existing platforms either:

- Provide **raw energy statistics** that are difficult for common users to interpret, or
- Focus only on technical or economic aspects without addressing **environmental ethics**, sustainability, or user awareness.

There is **no single, simple, student-friendly system** that:

- Compares renewable vs non-renewable energy usage in an interactive way
- Shows time-based energy demand growth visually
- Links energy choices to environmental and ethical implications
- Provides practical energy conservation tips

Hence, there is a need for an **interactive, web-based Energy Awareness and Analytics System** that helps users understand how growing energy needs can be met responsibly, with increased adoption of renewable energy resources.

1.4 Need for the System

The proposed system is required because:

- Energy consumption is increasing, but **awareness about sustainable energy** is not growing at the same pace.
- People often **do not know** the difference between renewable and non-renewable resources in terms of impact and longevity.
- Policymakers, students, and citizens need a simple **visual tool** to understand:
    - How much energy is coming from fossil fuels vs renewables
    - How energy demand is changing over time
    - What actions can be taken to save energy

The system:

- Presents complex data in an easy-to-understand form (charts, graphs, dashboards).
- Connects technical parameters with **ethical and environmental aspects**.

- Encourages users to **reduce wastage**, adopt **clean energy**, and develop **responsible energy habits**.

1.5 Objectives of the Project

The main objectives of **"Growing Energy Needs: Renewable and Non-Renewable Energy Resources"** are:

1. To design and develop a **web-based dashboard** to visualize growing energy needs.
2. To **compare** renewable and non-renewable energy resources with respect to usage and impact.
3. To provide **time-based analytics** (e.g., monthly or yearly trends) of energy consumption.
4. To demonstrate the environmental consequences of **over-reliance on fossil fuels**.
5. To create an **awareness module** containing energy conservation tips and ethical guidelines.
6. To use **IT tools (HTML, CSS, JavaScript, Python, Chart.js)** for data visualization and analysis.

7. To prepare a **student-friendly educational platform** that can be used for teaching and research.

1.6 Scope of the Project

The scope of the project includes:

- Analysis of energy usage for a **country, state, or simulated region** using sample datasets.
- Visualization of energy consumption by:
    - **Source** (solar, wind, hydro, coal, oil, gas, etc.)
    - **Sector** (residential, industrial, transport, commercial).
- Comparison of **percentage share** of renewable vs non-renewable energy
- Time-series analysis (e.g., **12-month or multi-year consumption trends**).
- Static/dynamic content on **renewable energy benefits** and **energy conservation**.

Possible extensions:

- Real time integration with **government or open data APIs**.
- Prediction and forecasting of **future energy demand** using machine learning.

1.7 Methodology

The project follows a structured methodology:

1. **Data Collection**
    - Collect or simulate data on energy consumption from different sources and sectors.
2. **Data Processing**
    - Clean and format the data (e.g., CSV/JSON).
    - Classify energy sources into renewable and non-renewable categories.
3. **System Design**
    - Design architecture and data flow.
    - Plan user interface components and dashboards.
4. **Frontend Development**
    - Develop UI using **HTML** for structure and **CSS** for styling
5. **JavaScript Logic & Visualization**
    - Implement functions to load data, filter by region/time/source.
    - Use **Chart.js** to generate bar, line, and pie charts.
6. **Backend Development (Optional)**
    - Use **Python (Flask/Django)** to serve data from files or a database via REST APIs.

7. **Testing**
    - Test performance, correctness, and usability.
8. **Deployment**
    - Deploy on localhost or a cloud server for demonstration.

1.8 Data Required

- Source-wise energy consumption values, such as:
    - Solar, wind, hydro, biomass, geothermal (renewable)
    - Coal, oil, natural gas, nuclear (non-renewable)
- Sector-wise data (transport, industry, residential, commercial).
- Time-series datasets (e.g., monthly consumption values).
- CO_2 emission factors for different energy sources (for optional impact analysis).
- Textual data for:
    - Energy conservation tips
    - Advantages of renewable energy
    - Environmental ethics and sustainability principles

CHAPTER 2 – SYSTEM ANALYSIS OF EXISTING SYSTEM

System analysis involves studying current systems, identifying their limitations, and defining the requirements for an improved solution. For this project, system analysis focuses on how energy information is currently presented to users and how it can be made more **interactive, educational, and ethics-oriented**.

2.1 Existing System

2.1.1 Current Energy Information and Monitoring Systems

Currently, energy-related data is available through:

- Government energy departments and ministries
- Electricity boards and power corporations
- International agencies (IEA, IRENA, etc.)
- Research reports and academic publications

These systems generally provide:

- Statistical reports in PDF or spreadsheet form
- Static charts and graphs

- Technical dashboards for policymakers and experts

However, from the perspective of a **student or common citizen**, the existing systems have certain drawbacks:

- Interfaces are often **complex and technical**.
- Information is highly detailed but **not simplified** for basic understanding.
- They focus mainly on **numbers and figures**, not on ethics, sustainability, or user-friendly awareness.

2.1.2 Limitations of Existing System

Major limitations include:

1. **Lack of Simplified Visualization**
    - Data is often presented in large tables, making it difficult to interpret trends.

2. **No Direct Comparison Dashboard**
    - Users cannot easily see the proportion of renewable vs non-renewable energy in a single, clear view.

3. **Minimal Focus on Environmental Ethics**
    - Most systems treat energy as a technical or economic subject only.
    - There is little to no discussion on **moral responsibility, intergenerational equity, or environmental ethics**.

4. **Limited Awareness Content**
    - Very few systems offer **tips for energy conservation** or explain how everyday actions affect energy demand.

5. **Not Student-Friendly**
    - Interfaces are designed mainly for researchers or professionals, not for learners at the undergraduate level.

6. **No Custom Visualization on Demand**
    - Users often cannot choose time range, source, or sector and instantly see a custom chart.

2.2 Proposed System

The proposed **Energy Awareness and Analytics System** is designed to overcome these limitations by providing a **simple, interactive, and educational** platform.

2.2.1 Features of Proposed System

The system provides:

- ✓ Source-wise energy consumption visualization
- ✓ Comparison between renewable and non-renewable resources
- ✓ Time-based graphs (e.g., last 12 months) showing energy demand trends
- ✓ Pie charts showing energy mix (renewable vs non-renewable)
- ✓ Textual insights on environmental impacts of different resources
- ✓ Module on **energy conservation and environmental ethics**
- ✓ User-friendly interface suitable for students and general users

2.2.2 Advantages of Proposed System

Existing System	Proposed System (Energy Analytics)	Improvement
Raw statistics and tables	Interactive graphs and dashboards	Better understanding at a glance
Technical and expert-focused	Student-friendly and easy to navigate	Wider accessibility
No explicit ethics/environment module	Embedded content on sustainability and environmental ethics	Awareness + responsibility
Poor renewable vs non-renewable focus	Clear visual comparison between energy resource types	Direct understanding of energy mix
Limited user interaction	User-driven filters (time, source, sector)	Custom views & engagement

2.3 System Analysis Methods Used

To analyze the existing scenario and define requirements, the following fact-finding techniques were used:

- **Observation**
    - Studied existing government and international energy dashboards and reports.

- **Internet Research**
    - Checked official energy statistics and renewable energy portals.

- **Informal User Feedback**
    - Asked students and teachers what kind of energy information they find useful and easy to understand.

- **Document Analysis**
    - Reviewed articles, reports, and books on renewable energy, sustainability, and environmental ethics.

These techniques helped identify the need for a **visual, simplified, and ethics-focused energy information system.**

2.4 Feasibility Study

Before implementation, a feasibility study was carried out in three key areas.

2.4.1 Technical Feasibility

The project uses widely available and well-supported technologies:

- **Frontend:** HTML, CSS, JavaScript
- **Visualization:** Chart.js
- **Backend (optional):** Python with Flask/Django
- **Data Storage:** JSON/CSV/MySQL

All required tools are:

- Free or open-source
- Compatible with standard operating systems
- Easy to install and use for a student-level project

Thus, the system is **technically feasible**.

2.4.2 Operational Feasibility

- The system is intuitive and easy to work with.
- Users only need a web browser to access the dashboard.
- Students, teachers, and general users can quickly learn how to:
    - Select time periods or types of energy
    - Read and interpret charts
    - Understand the basic messages on energy conservation

Hence, the system is **operationally feasible**.

2.4.3 Economic Feasibility

- The project uses **free tools and libraries**
- No license fees or proprietary software costs are involved.
- It runs on standard hardware without additional infrastructure.

So, the cost of development is minimal, making it **economically feasible**.

CHAPTER 3 – SYSTEM REQUIREMENT ANALYSIS

System requirement analysis defines what the system must do and what resources are needed for its implementation.

3.1 Objective of the System

The main objective is to create an **interactive and educational platform** that:

- Visualizes **growing energy needs** over time.
- Compares **renewable and non-renewable energy resources**.
- Provides insights and awareness about **sustainable energy usage**.

3.2 Hardware Requirements

3.2.1 Minimum Hardware Requirements

- **Processor:** Intel i3 or equivalent
- **RAM:** 4 GB or higher
- **Storage:** At least 500 MB free space
- **Display:** 720p or higher resolution

- **Input Devices:** Keyboard and Mouse

These specifications are sufficient for development and running the web-based system locally.

3.3 Software Requirements

3.3.1 Minimum Software Requirements

Software	Purpose
Windows / macOS / Linux	Operating System
Web Browser (Chrome, etc.)	Running the web interface
Python 3.x	Backend processing and scripting
Flask / Django	Web backend framework (optional)
Chart.js	Data visualization (charts & graphs)
HTML, CSS, JavaScript	Frontend development

3.4 Functional Requirements

Functional requirements define **what** the system must do.

3.4.1 Resource Selection Function

- Users must be able to select filters such as:
    - Energy source type (renewable / non-renewable / specific sources)
    - Time period (months/years)
    - Sector (optional: residential, industrial, transport).

- System must update charts based on selected filters.

3.4.2 Energy Mix Display

- The system must display:
    - Percentage share of renewable vs non-renewable energy.
    - Graphical representation (pie chart / bar graph).

3.4.3 Time-Based Trend Analytics

- The system must show a **line or bar chart** representing energy consumption trends over a time period (e.g., 12 months).

- The chart should automatically update when the user changes the filter (source/sector).

3.4.4 Environmental Impact & Ethics Content

- System must display basic **environmental impacts** of non-renewable resources.
- Provide **ethics and awareness content**, such as:
    - Need to protect resources for future generations
    - Responsibility to reduce pollution and wastage

3.4.5 Energy Conservation Tips

- Based on the analysis, general **energy saving tips** must be displayed, such as:
    - Use LED lights
    - Use public transport
    - Install solar panels where possible

3.5 Non-Functional Requirements

Non-functional requirements specify **how** the system should behave.

3.5.1 Performance Requirements

- Charts and data must load within **1–2 seconds** for typical dataset sizes
- Filtering and chart redraw operations must feel smooth.

3.5.2 User Interface Requirements

- Clean, modern, and responsive UI.
- Clear labels, legends, and color coding for renewable vs non-renewable.
- Should work properly on both desktop and laptop screens.

3.5.3 Reliability & Availability

- The system should run reliably as long as the browser is open.
- Local datasets allow the system to run even **without continuous internet**, if the data is bundled.

3.5.4 Security Requirements

- If a backend is used, it should validate user inputs (e.g., selected filters).
- Protect against script injection in any user-entered fields (if forms are present).

3.5.5 Scalability

- System should support:
    - Addition of more energy sources or sectors in the dataset.
    - Extension to larger datasets (e.g., multi-year data).

3.6 User Requirements

3.6.1 End User Requirements

End users (students, teachers, general public) should be able to:

- Open the dashboard in a web browser.
- Select energy sources, time ranges, or sectors.
- Easily understand charts representing energy mix and trends.
- Read energy conservation tips and ethics-related information.

3.6.2 Administrator Requirements

Admins (project developers or future maintainers) should be able to:

- Update or replace the energy dataset.
- Add new sources and sectors.
- Modify or extend awareness content.
- Change configuration related to colors, chart types, or labels.

CHAPTER 4 – SYSTEM DESIGN

System design translates the requirements into a **blueprint** for implementation. It includes architecture, data flow, diagrams, database design, and user interface layout.

4.1 System Architecture

The system follows a **client-server web architecture**:

- **Client (Browser):**
    - Displays HTML pages
    - Runs JavaScript to render charts using Chart.js
    - Receives data (locally or from backend APIs)

- **Server (Optional Backend – Python/Flask/Django):**
    - Reads datasets from JSON/CSV/MySQL
    - Provides data to frontend as JSON responses

4.2 Architecture Overview

Process Flow:

1. User opens the web application in a browser.
2. User selects energy source type, time period, or sector.
3. JavaScript fetches the corresponding data (from local JSON or backend API).
4. Data is processed (aggregated, grouped by source/time).
5. Charts (bar/line/pie) are rendered using Chart.js.
6. Awareness and conservation tips are displayed along with analysis.

4.3 Data Flow Diagrams (DFD)

4.3.1 DFD Level 0 (Context Diagram)

- **External Entity:** User
- **System:** Energy Awareness and Analytics System

Data Flow:

- User → (Input: filter selection) → System
- System → (Output: charts, summaries, tips) → User

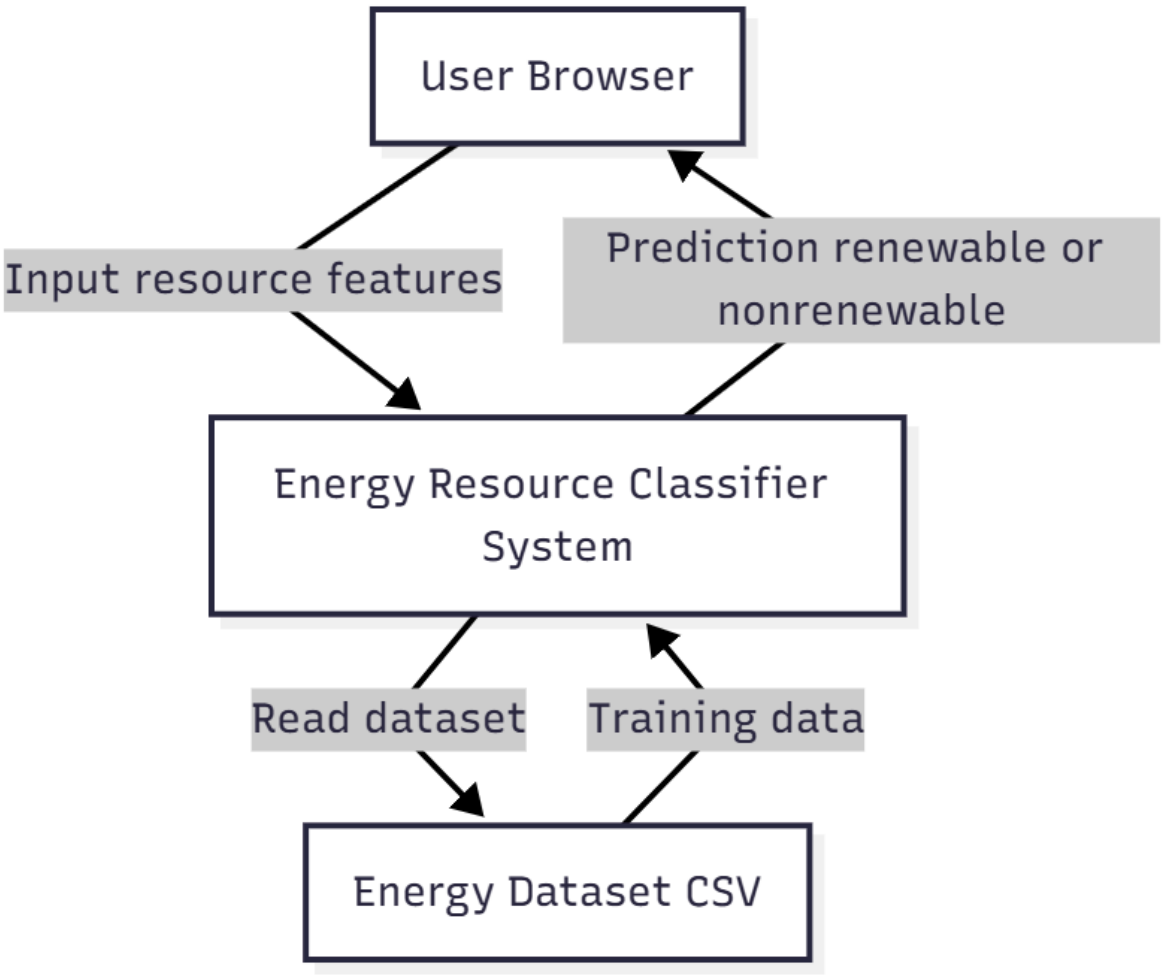
User Browser
Input resource features
Prediction renewable or nonrenewable
Energy Resource Classifier System
Read dataset
Training data
Energy Dataset CSV

fig: 1 (DFD-0)

4.3.2 DFD Level 1 (Detailed View)

Main processes:

1. **P1: Receive Filter Input**
    - User selects source/time/sector.

2. **P2: Fetch Dataset**
    - Data retrieved from JSON/CSV/Database.

3. **P3: Process Data**
    - Classify energy as renewable/non-renewable.
    - Aggregate for selected time range.

4. **P4: Generate Charts**
    - Prepare chart datasets.
    - Render using Chart.js.

5. **P5: Display Awareness Content**

- Load relevant conservation tips and ethics information.

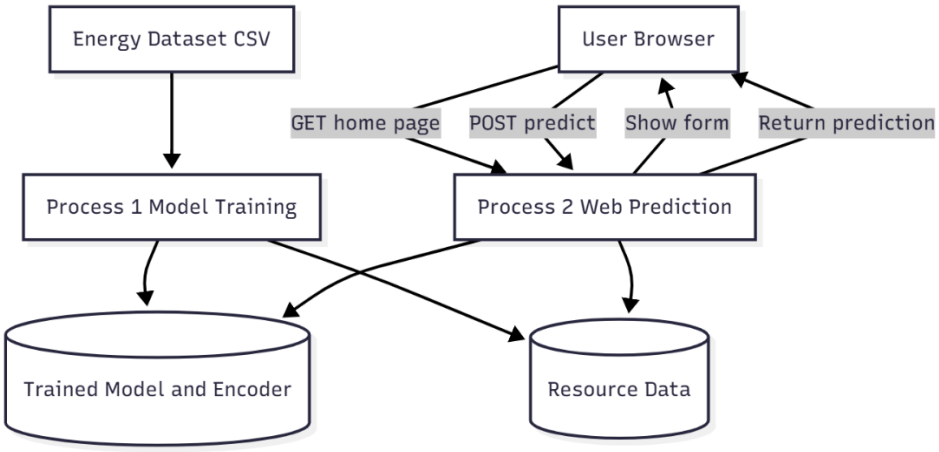

Fig : 2 (DFD-1)

4.4 Use Case Diagram

Actors:

- **User** – Interacts with the dashboard.
- **Admin** – Manages datasets and content (optional).

User Use Cases:

- Select energy source / time / sector
- View energy mix
- View trend charts
- Read conservation tips

- Learn about renewable vs non-renewable resources

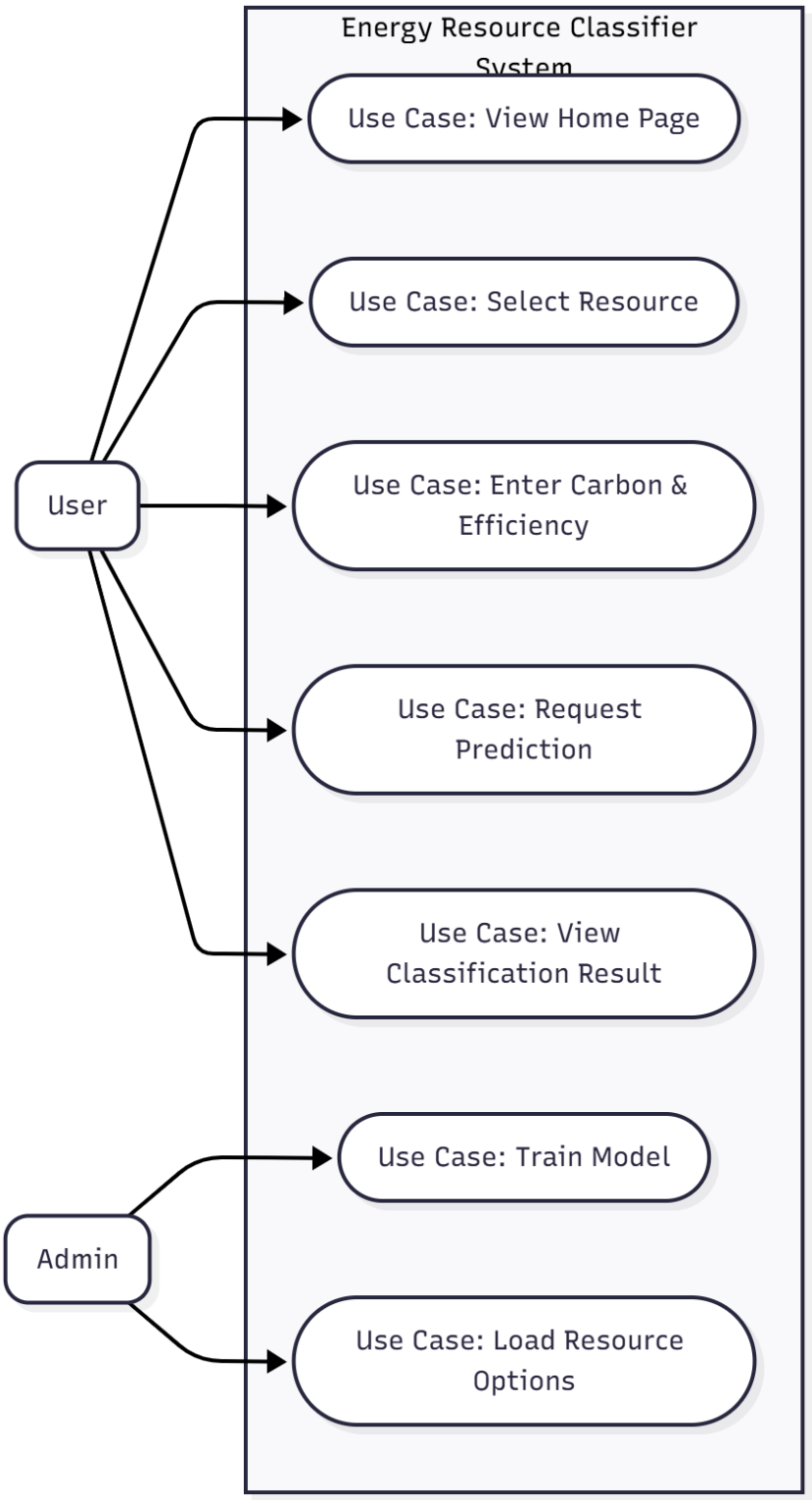

fig : 3 (Use-case)

4.5 Entity Relationship Diagram (ERD)

Main Entities:

1. **EnergySource**
    - source_id (PK)
    - source_name (e.g., Solar, Coal)
    - type (Renewable/Non-Renewable)

2. **EnergyData**
    - data_id (PK)
    - source_id (FK)
    - time_period (month/year)
    - sector (e.g., Residential, Industrial)
    - consumption_value (e.g., kWh, GWh)

3. **Tips**
    - tip_id (PK)
    - category (General / Renewable / Non-Renewable)
    - tip_text

Relationships:

- One **EnergySource** can have many **EnergyData** entries.
- Tips are displayed based on category or context (no strict FK required).

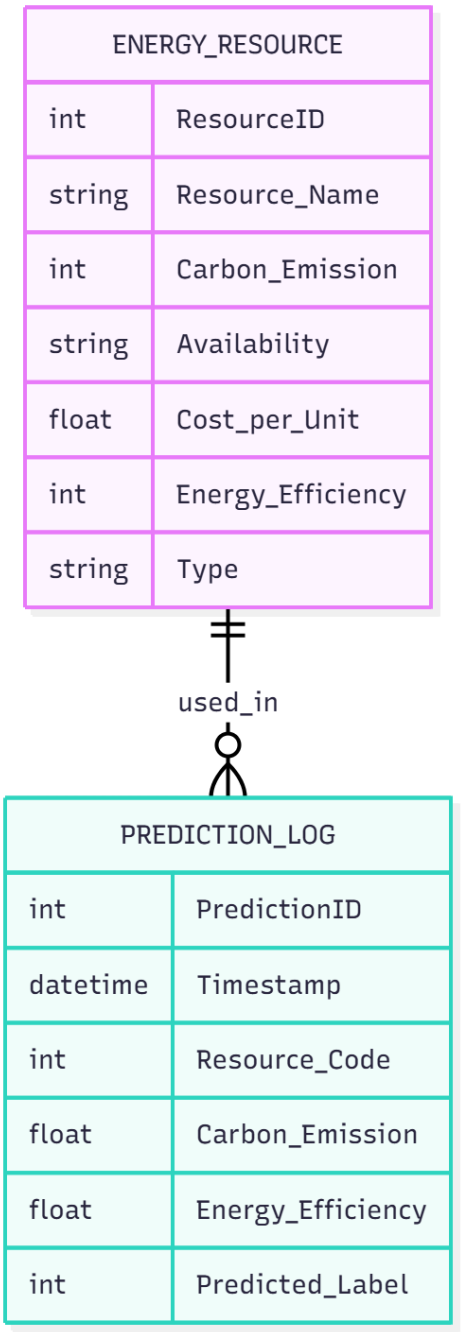

Fig : 4 (ER Diagram)

4.6 Class Diagram

Logical classes for implementation:

Class: EnergySource

- Attributes: sourceName, type
- Methods: getType(), isRenewable()

Class: EnergyRecord

- Attributes: source, timePeriod, sector, consumption
- Methods: getConsumption(), getTimePeriod()

Class: DataProcessor

- Methods:
    - loadData()
    - filterBySource()
    - aggregateByTime()
    - computeEnergyMix()

Class: ChartManager

- Methods:
    - renderBarChart()
    - renderPieChart()
    - renderLineChart()

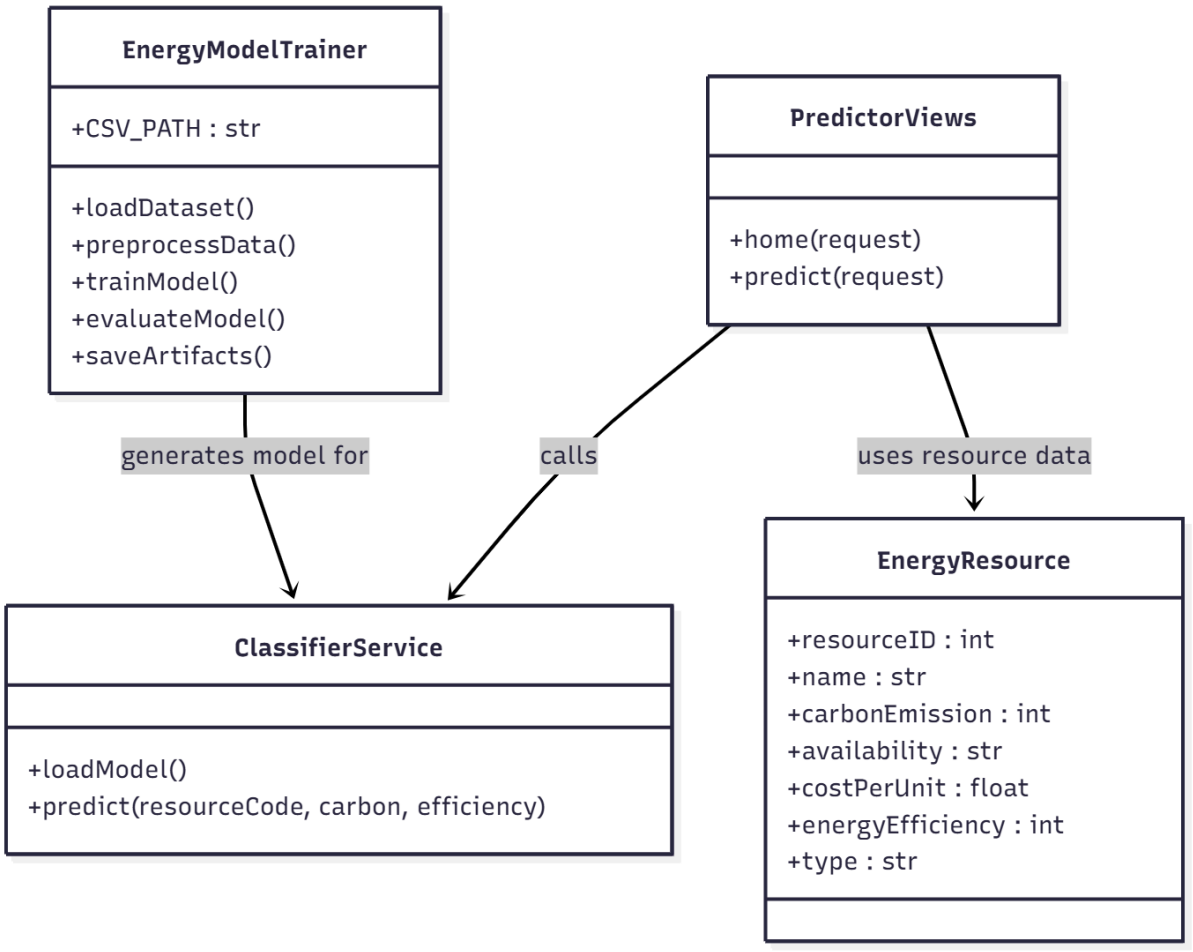

Fig : 5 (Class Diagram)

CHAPTER 5 – SYSTEM DEVELOPMENT

This chapter explains the tools, technologies, and implementation steps followed to build the system.

5.1 Programming Platform

- **Python 3.x** – For data preprocessing and optional backend.
- **Flask / Django** – To create REST APIs for energy data (optional).
- **JavaScript (ES6)** – For frontend logic and interactivity.
- **Chart.js** – For data visualization.
- **HTML5, CSS3** – For building a responsive web interface.
- **VS Code / PyCharm** – As IDE for development.

5.2 Dataset Preprocessing (Python Code – Conceptual)

Data preprocessing includes:

1. Reading raw CSV/Excel files.
2. Mapping each energy source to its type (Renewable/Non-Renewable).
3. Aggregating data per month/year for each source.
4. Exporting cleaned data to JSON for easy consumption in JavaScript.

Pseudo-steps:

- Load raw dataset using Python (pandas).
- Group data by time and source
- Calculate totals and percentages.
- Save output to a structured JSON file.

5.3 Frontend Interface Development (HTML)

The HTML structure:

- `<header>` – Project title and navigation menu.
- `<section id="filters">` – Dropdowns and buttons for filters.
- `<section id="summary">` – Cards for showing renewable %, non-renewable %.
- `<canvas>` elements – For Chart.js graphs.
- `<section id="awareness">` – Text blocks for tips and ethics content.

5.4 JavaScript Logic and Backend Integration

JavaScript responsibilities:

- Load JSON dataset using `fetch()`.
- Filter data based on user selection.
- Prepare dataset for Chart.js charts.
- Render or update charts dynamically.
- Show context-relevant tips (e.g., more tips about renewables when their share is low).

If a backend (Flask/Django) is used:

- JavaScript calls API endpoints (e.g., `/api/energy?year=2024`) to fetch data.
- Python backend reads from database/JSON and returns JSON responses.

5.5 User Interface Styling (CSS)

- Use **CSS3** for:
    - Responsive layout (flexbox/grid).
    - Color schemes:
        - Green/blue for renewables
        - Grey/red for non-renewable
    - Clear typography and spacing.

5.6 Summary of System Development

- Data preprocessing prepared **clean, structured datasets**.
- Frontend design focused on **simplicity and clarity**.
- JavaScript with Chart.js enabled **interactive visualizations**.
- Optional backend support (Python/Flask) enhances **scalability and maintainability**.

CHAPTER 6 – SYSTEM TESTING

System testing ensures that the application functions correctly, efficiently, and as per user expectations.

6.1 Testing Objectives

- To verify that energy data loads correctly.
- To confirm that charts and graphs display accurate information.
- To ensure that filtering by source/time updates charts properly.
- To check that awareness content is displayed correctly
- To validate cross-browser compatibility and responsiveness.

6.2 Testing Levels

6.2.1 Unit Testing

Tested individual functions:

- Data loading functions (JSON fetch).
- Filter functions (e.g., filter by year, by renewable type).
- Chart generation functions.

6.2.2 Integration Testing

Tested:

- JavaScript data processing + Chart.js rendering.
- Frontend filter controls + chart updates.
- (If backend used) API endpoints + JavaScript fetch calls.

6.2.3 System Testing

End-to-end tests:

- User opens the app, selects a time period and source.
- System loads data, generates charts, and displays awareness messages.

6.2.4 Functional Testing

Checked each feature against expected behavior:

- Resource selection
- Time-based trend display
- Energy mix chart
- Awareness content section

6.2.5 Usability Testing

Evaluated:

- Ease of navigation
- Clarity of charts and legends
- Readability of tips and descriptions
- Overall user experience

6.3 Test Cases (Examples)

Test Case 1 – Source Filter

- Action: Select “Renewable Only
- Expected: Charts show only renewable sources and correct totals
- Result: Passed

Test Case 2 – Time Range Filter

- Action: Select Year 2023
- Expected: Trend chart shows data for 2023 only
- Result: Passed

Test Case 3 – Energy Mix Chart

- Action: Load dashboard
- Expected: Pie chart showing renewable vs non-renewable percentage
- Result: Passed

Test Case 4 – Awareness Section

- Action: Load page & scroll to awareness section
- Expected: Energy conservation tips and ethical messages visible and readable
- Result: Passed

CHAPTER 7 – SYSTEM IMPLEMENTATION

7.1 Implementation Overview

The system has been implemented as a **web-based dashboard** that:

- Loads energy data from JSON/CSV.
- Allows user to apply filters
- Displays interactive charts with Chart.js.
- Shows awareness content related to energy conservation and sustainability.

7.2 Implementation Tools and Technologies

- **Frontend:** HTML, CSS, JavaScript, Chart.js
- **Backend (optional):** Python 3.x with Flask/Django
- **Data Storage:** JSON or MySQL
- **IDE:** VS Code / PyCharm
- **Browser:** Chrome / Edge / Firefox for testing

7.3 Implementation Steps

1. **Set Up Project Structure**
    - Create folders: `css`, `js`, `data`, `img`.

2. **Prepare Dataset**
    - Place preprocessed JSON/CSV files in `data/`.

3. **Build HTML Structure**
    - Create main dashboard with sections for filters, charts, and awareness.

4. **Write JavaScript Logic**
    - Implement data loading and chart rendering.

5. **Apply CSS Styling**
    - Ensure responsive and clean design.

6. **Configure Backend (Optional)**
    - Set up Flask/Django routes to serve datasets.

7. **Testing and Debugging**
    - Test on multiple browsers and screen sizes.

8. **Deployment**
    - Host on localhost or deploy to a web server.

7.4 Screenshots (Descriptions)

(In your final project report you can replace these descriptions with actual screenshots.)

- **Figure-1:** Home dashboard showing total energy demand and share of renewable vs non-renewable resources.
- **Figure-2:** Filter panel where user selects year and energy type to update analytics.
- **Figure-3:** Line chart displaying monthly energy consumption trend for a given year.
- **Figure-4:** Pie chart showing percentage contribution of different energy sources.

- **Figure-5:** Awareness section listing energy conservation tips and renewable energy benefits.

Chapter 9 CODING and OUTPUT:

9.1) Code:-

Energy_model.py

```
import os
import pickle


import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier


# ------------------------------------------------
# Paths
# ------------------------------------------------
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
CSV_PATH = os.path.join(BASE_DIR, "energy_dataset_with_resources.csv")


print("Using CSV:", CSV_PATH)
```

```
# ---------------------------------------------
# Load dataset
# Expected columns (case-sensitive):
#   - Resource_Name
#   - Carbon_Emission
#   - Energy_Efficiency
#   - Type    (values like 'Renewable' / 'Non-Renewable')
# ---------------------------------------------
df = pd.read_csv(CSV_PATH)

needed_cols = ["Resource_Name", "Carbon_Emission", "Energy_Efficiency",
"Type"]
missing = [c for c in needed_cols if c not in df.columns]
if missing:
    raise ValueError(f"Missing columns in CSV: {missing}")

df = df[needed_cols].dropna()

# ---------------------------------------------
# Encode categorical columns
# ---------------------------------------------
le_resource = LabelEncoder()
```

```
df["Resource_Code"] = le_resource.fit_transform(df["Resource_Name"])


# Make label 1 = Renewable, 0 = Non-Renewable (adjust text to match your
CSV)
y_text = df["Type"].str.strip().str.lower()
y = np.where(y_text == "renewable", 1, 0)


X = df[["Resource_Code", "Carbon_Emission", "Energy_Efficiency"]]


# ------------------------------------------------
# Train / test split
# ------------------------------------------------
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)


# ------------------------------------------------
# Model
# ------------------------------------------------
model = RandomForestClassifier(
    n_estimators=200,
    max_depth=None,
```

```
    random_state=42,
)
model.fit(X_train, y_train)

acc = model.score(X_test, y_test)
print(f"Validation accuracy: {acc:.3f}")

# ------------------------------------------------
# Save model + encoder  (IN PROJECT ROOT)
# ------------------------------------------------
model_path = os.path.join(BASE_DIR, "energy_classifier.pkl")
encoder_path = os.path.join(BASE_DIR, "resource_encoder.pkl")

with open(model_path, "wb") as f:
    pickle.dump(model, f)

with open(encoder_path, "wb") as f:
    pickle.dump(le_resource, f)

print("Saved:")
print("  ", model_path)
print("  ", encoder_path)
```

MYSITE : asgi.py

```
"""
ASGI config for mysite project.
It exposes the ASGI callable as a module-level variable named ``application``.
For more information on this file, see
https://docs.djangoproject.com/en/5.2/howto/deployment/asgi/
"""

import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'mysite.settings')

application = get_asgi_application()
```

settings.py

```
"""
Django settings for mysite project.

Generated by 'django-admin startproject' using Django 5.2.7.

For more information on this file, see
https://docs.djangoproject.com/en/5.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/5.2/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.2/howto/deployment/checklist/
```

```
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-aqmj&!hkqcx@-
ugheqp*vmc#7(s!jnd$s87@0t)21!ko4$f*wq'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'predictor',
]
```

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'mysite.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
```

```
            ],
        },
    },
]

WSGI_APPLICATION = 'mysite.wsgi.application'



# Database
# https://docs.djangoproject.com/en/5.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}



# Password validation
# https://docs.djangoproject.com/en/5.2/ref/settings/#auth-password-validators
```

```
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
```

```
# Internationalization
# https://docs.djangoproject.com/en/5.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.2/howto/static-files/

STATIC_URL = 'static/'

# Default primary key field type
# https://docs.djangoproject.com/en/5.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

urls.py

```
"""
URL configuration for mysite project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/5.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
```

```
    path('', include('predictor.urls')),
]
```

PREDICTOR

Templates

<>home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Energy Classifier</title>

    <!-- Bootstrap 5 -->
    <link

href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
        rel="stylesheet"
    >

    <style>
        body {
```

```
            background: #f5f5f7;
            font-family: system-ui, -apple-system, BlinkMacSystemFont, "Segoe
UI", sans-serif;
        }
        .page-title {
            font-weight: 700;
            letter-spacing: 0.03em;
        }
        .card {
            max-width: 520px;
            margin: 40px auto;
            border-radius: 16px;
            box-shadow: 0 10px 30px rgba(0,0,0,0.08);
        }
        .helper-text {
            font-size: 0.85rem;
            color: #6c757d;
        }
    </style>
</head>
<body>
<nav class="navbar bg-light mb-4 shadow-sm">
```

```
    <div class="container">
      <span class="navbar-brand mb-0 h1">Energy Resource Classifier</span>
    </div>
</nav>

<div class="container">
  <div class="card">
    <div class="card-body">
      <h2 class="page-title mb-3">Make a Prediction</h2>
      <p class="helper-text mb-4">
        Choose an energy resource from the list or enter its numeric code,
        then provide carbon emission and efficiency to classify it as
        <strong>Renewable</strong> or <strong>Non-
Renewable</strong>.<br>
        <em>This list is built from your dataset ({{ resource_count }}
resources).</em>
      </p>

      <form method="post" action="{% url 'predict' %}">
        {% csrf_token %}

        <!-- Resource select -->
```

```
            <div class="mb-3">
                <label for="resource_name" class="form-label">Resource
Name</label>
                <select class="form-select" id="resource_name"
name="resource_name">
                    <option value="">-- Choose resource or enter code --</option>
                    {% for name, code in options %}
                        <option value="{{ name }}" data-code="{{ code }}">
                            {{ name }} (code: {{ code }})
                        </option>
                    {% endfor %}
                </select>
                <div class="helper-text">
                    Selecting a resource will auto-fill the numeric code field below.
                </div>
            </div>

            <!-- Resource code (optional manual override) -->
            <div class="mb-3">
                <label for="resource_code" class="form-label">
                    Or enter numeric code directly
                </label>
```

```
    <input
      type="number"
      class="form-control"
      id="resource_code"
      name="resource_code"
      placeholder="Optional: override with numeric code"
    >
    <div class="helper-text">
      Leave blank if you prefer to use the dropdown above.
    </div>
  </div>

  <!-- Carbon emission -->
  <div class="mb-3">
    <label for="carbon" class="form-label">Carbon Emission</label>
    <input
      type="number"
      step="any"
      class="form-control"
      id="carbon"
      name="carbon"
      required
```

```
            >
          </div>

          <!-- Energy efficiency -->
          <div class="mb-4">
            <label for="efficiency" class="form-label">Energy
Efficiency</label>
            <input
              type="number"
              step="any"
              class="form-control"
              id="efficiency"
              name="efficiency"
              required
            >
          </div>

          <button type="submit" class="btn btn-primary w-100">
            Predict
          </button>
        </form>
      </div>
```

```
    </div>
</div>

<script>
    // When resource is selected, auto-fill its code into the numeric field
    const selectEl = document.getElementById('resource_name');
    const codeInput = document.getElementById('resource_code');

    if (selectEl && codeInput) {
        selectEl.addEventListener('change', function () {
            const opt = this.options[this.selectedIndex];
            const code = opt.getAttribute('data-code');
            if (code) {
                codeInput.value = code;
            }
        });
    }
</script>
</body>
</html>
```

<>result.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Prediction Result</title>

    <!-- Bootstrap -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet">

    <style>
        body {
            background: #f5f5f7;
        }
        .result-card {
            max-width: 520px;
            margin: 60px auto;
            border-radius: 16px;
            box-shadow: 0 12px 35px rgba(0,0,0,0.10);
```

```
        }
        .result-title {
            font-weight: 700;
            letter-spacing: 0.04em;
        }
        .emoji {
            font-size: 2.5rem;
            margin-left: 8px;
        }
        .subtitle {
            color: #6c757d;
            font-size: 0.9rem;
        }
    </style>
</head>

<body>
    <div class="container">
        <div class="card result-card p-4 text-center">

            <h1 class="result-title mb-3">Prediction Result</h1>
```

```
{% if prediction == "Renewable" %}
  <h2 class="text-success">
    Renewable
    <span class="emoji">🌱</span>
  </h2>
  <p class="subtitle mt-2">
    This resource is classified as <strong>renewable</strong>, which
generally means
    lower long-term environmental impact and better sustainability.
  </p>
{% else %}
  <h2 class="text-danger">
    Non-Renewable
    <span class="emoji">🔥</span>
  </h2>
  <p class="subtitle mt-2">
    This resource is classified as <strong>non-renewable</strong>,
which usually
    means limited availability and higher environmental impact.
  </p>
{% endif %}
```

```
            <div class="mt-4">
                <a href="{% url 'home' %}" class="btn btn-outline-primary">
                    Predict Again
                </a>
            </div>


        </div>
    </div>
</body>
</html>
```

apps.py

```
from django.apps import AppConfig
class PredictorConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'predictor'
```

urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.home, name='home'),
    path('predict/', views.predict, name='predict'),
]
```

views.py

```
# predictor/views.py

import os
import pickle

import numpy as np
import pandas as pd
from django.shortcuts import render
from django.http import HttpResponse
```

```
# ------------------------------------------------------
# Paths
# ------------------------------------------------------
# This file lives in: Energy_Classifier/mysite/predictor/views.py
THIS_DIR = os.path.dirname(os.path.abspath(__file__))        #
.../mysite/predictor
MYSITE_DIR = os.path.dirname(THIS_DIR)                       # .../mysite
PROJECT_ROOT = os.path.dirname(MYSITE_DIR)                   #
.../Energy_Classifier

MODEL_PATH = os.path.join(PROJECT_ROOT, "energy_classifier.pkl")
CSV_PATH   = os.path.join(PROJECT_ROOT,
"energy_dataset_with_resources.csv")
ENCODER_PATH = os.path.join(PROJECT_ROOT, "resource_encoder.pkl")



# ------------------------------------------------------
# Globals: model + dropdown options
# ------------------------------------------------------
model = None              # will hold the trained classifier
RESOURCE_OPTIONS = []     # list of (name, code) tuples
```

```
def _load_model_and_options():
    """
    Called once when Django imports this module.
    Loads the trained model and builds the dropdown options
    from the CSV file.
    """
    global model, RESOURCE_OPTIONS

    # ---------- Load model ----------
    try:
        with open(MODEL_PATH, "rb") as f:
            model = pickle.load(f)
        print(f"[OK] Loaded model from {MODEL_PATH}")
    except Exception as e:
        model = None
        print(f"[!] Error loading model from {MODEL_PATH}: {e}")

    # ---------- Build RESOURCE_OPTIONS from CSV ----------
    RESOURCE_OPTIONS = []
    try:
        df = pd.read_csv(CSV_PATH)
```

```
        # Expect a column named 'Resource_Name'
        if "Resource_Name" not in df.columns:
            raise ValueError("CSV is missing 'Resource_Name' column")

        # Unique names, sorted for deterministic codes
        names = sorted(df["Resource_Name"].dropna().unique())

        # Create mapping name -> code (0,1,2,...) – must match energy_model.py
        name_to_code = {name: idx for idx, name in enumerate(names)}

        # Store as list of (name, code) for the template
        RESOURCE_OPTIONS = [(name, name_to_code[name]) for name in
names]

        print(f"[OK] Loaded {len(RESOURCE_OPTIONS)} resource options
from CSV.")
    except Exception as e:
        print(f"[!] Error loading resource options from {CSV_PATH}: {e}")
        # Fallback so the app still works
        RESOURCE_OPTIONS = [
            ("Solar", 0),
```

```
            ("Wind", 1),
            ("Petroleum", 2),
        ]
        print("[!] Using fallback RESOURCE_OPTIONS:",
RESOURCE_OPTIONS)


# Load everything when Django imports this file
_load_model_and_options()


# ----------------------------------------------------------
# Views
# ----------------------------------------------------------


def home(request):
    """
    Show the main form.
    """
    context = {
        "options": RESOURCE_OPTIONS,
    }
```

```
    return render(request, "home.html", context)



def predict(request):
    """
    Handle POST from the form, run the model, and send result to result.html.
    """
    if request.method != "POST":
        # If someone visits /predict directly, just show the form again
        return render(request, "home.html", {"options": RESOURCE_OPTIONS})

    if model is None:
        return HttpResponse(
            "Model not loaded on server. Check server logs for details.",
            status=500,
        )

    # --------- Read form values ---------
    # from dropdown (numeric code as string)
    resource_from_select = (request.POST.get("resource_name") or "").strip()
    # manual numeric code input
    resource_from_input = (request.POST.get("resource_code") or "").strip()
```

```
carbon_str = (request.POST.get("carbon") or "").strip()
efficiency_str = (request.POST.get("efficiency") or "").strip()

# --------- Decide which resource code to use ---------
try:
    if resource_from_input:
        # user typed a code manually
        resource_code = int(resource_from_input)
    elif resource_from_select:
        # dropdown value is also a numeric code
        resource_code = int(resource_from_select)
    else:
        return HttpResponse(
            "Please select a resource OR enter its numeric code.",
            status=400,
        )
except ValueError:
    return HttpResponse("Resource code must be an integer.", status=400)

# --------- Parse numeric features ---------
try:
```

```
        carbon = float(carbon_str)
        efficiency = float(efficiency_str)
    except ValueError:
        return HttpResponse(
            "Carbon Emission and Energy Efficiency must be numeric values.",
            status=400,
        )

    # --------- Run model ---------
    features = np.array([[resource_code, carbon, efficiency]])

    try:
        pred_raw = model.predict(features)[0]
    except Exception as e:
        return HttpResponse(f"Prediction error: {e}", status=500)

    # Most sklearn classifiers return a numeric class (e.g., 0 or 1)
    try:
        pred_int = int(pred_raw)
    except Exception:
        # If your model returns a label string directly, fall back to 1 for
"Renewable" style templates
```

```
        # Adjust this if needed.
        pred_int = 1 if str(pred_raw).lower().startswith("ren") else 0


    # Pass the numeric prediction to result.html
    # Your existing result.html probably has:
    #   {% if prediction == 1 %} Renewable {% else %} Non-Renewable {%
endif %}
    context = {"prediction": pred_int}
    return render(request, "result.html", context)
```

manage.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'mysite.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)
```

```
if __name__ == '__main__':
    main()
```

Minor.ipynb

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split , cross_val_score
from sklearn.preprocessing import LabelEncoder , StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import(accuracy_score , confusion_matrix ,
classification_report , roc_auc_score , roc_curve)
import matplotlib as plt
import seaborn as sns

# load dataset
df =
pd.read_csv(r"c:\Users\DELL\OneDrive\Documents\energy_dataset_with_reso
urces.csv")

# check dataset
print(df.head())
```

Resource_Name	Carbon_Emission	...	Energy_Efficiency	Type
0 Solar	170	...	90	Renewable
1 Petroleum	796	...	48	Non-Renewable
2 Wind	124	...	74	Renewable
3 Wind	65	...	87	Renewable
4 Solar	127	...	70	Renewable

[5 rows x 6 columns]

print(df.info())

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1000 entries, 0 to 999

Data columns (total 6 columns):

#	Column	Non-Null Count	Dtype
0	Resource_Name	1000 non-null	object
1	Carbon_Emission	1000 non-null	int64
2	Availability	1000 non-null	object
3	Cost_per_Unit	1000 non-null	float64

4 Energy_Efficiency 1000 non-null int64

5 Type 1000 non-null object

dtypes: float64(1), int64(2), object(3)

memory usage: 47.0+ KB

None

print(df.describe())

Carbon_Emission	Cost_per_Unit	Energy_Efficiency
count 1000.000000	1000.000000	1000.000000
mean 390.757000	0.082537	64.072000
std 297.987504	0.029935	19.057334
min 10.000000	0.030132	35.000000
25% 116.000000	0.059293	46.000000
50% 410.500000	0.079863	59.000000
75% 674.000000	0.099408	82.000000
max 899.000000	0.149935	94.000000

 # DATA PREPROCESSING

```
#ENCODE 'AVAULABILITY'
avail_encoder = LabelEncoder()
df['Availability_encoded'] = avail_encoder.fit_transform(df['Availability'])

#ENCODE 'TYPE'(RENEWABLE/NON-RENEWABLE)
type_encoder = LabelEncoder()
df['type_encoded'] = type_encoder.fit_transform(df['Type'])

# ENCODE RESOURCE NAME
resource_name_encoder = LabelEncoder()
df['Resource_Name_Encoded'] =
resource_name_encoder.fit_transform(df['Resource_Name'])

# DEFINE FEATURES AND TARGETS
feature_cols = ['Carbon_Emission' , 'Availability_encoded' , 'Cost_per_Unit' ,
'Energy_Efficiency']
X = df[feature_cols]
Y = df['type_encoded']

print(df.columns)

Index(['Resource_Name', 'Carbon_Emission', 'Availability', 'Cost_per_Unit',
```

'Energy_Efficiency', 'Type', 'Availability_encoder', 'type_encoded',

'Resource_Name_Encoded', 'Availability_encoded'],

dtype='object')

X_train , X_test , Y_train , Y_test = train_test_split(X,Y,test_size=0.2,

random_state=42)

TRAIN THE MODEL

model = RandomForestClassifier(n_estimators=42, random_state=42)

model.fit(X_train, Y_train)

RandomForestClassifier

?

i

Parameters

Accuracy: 1.0

n\Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	105
1	1.00	1.00	1.00	95
accuracy			1.00	200
macro avg	1.00	1.00	1.00	200
weighted avg	1.00	1.00	1.00	200

<>:2: SyntaxWarning: "\C" is an invalid escape sequence. Such sequences will not work in the future. Did you mean "\\C"? A raw string is also an option.

<>:2: SyntaxWarning: "\C" is an invalid escape sequence. Such sequences will not work in the future. Did you mean "\\C"? A raw string is also an option.

C:\Users\DELL\AppData\Local\Temp\ipykernel_17000\806865310.py:2: SyntaxWarning: "\C" is an invalid escape sequence. Such sequences will not work in the future. Did you mean "\\C"? A raw string is also an option.

```
print("n\Classification Report:\n" , classification_report(Y_test, Y_pred))
```

```
Y_pred = model.predict(X_test)

print("Accuracy:" , accuracy_score(Y_test, Y_pred))
print("n\Classification Report:\n" , classification_report(Y_test, Y_pred))
```

Accuracy: 1.0

n\Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	105
1	1.00	1.00	1.00	95
accuracy			1.00	200
macro avg	1.00	1.00	1.00	200
weighted avg	1.00	1.00	1.00	200

<>:2: SyntaxWarning: "\C" is an invalid escape sequence. Such sequences will not work in the future. Did you mean "\\C"? A raw string is also an option.

<>:2: SyntaxWarning: "\C" is an invalid escape sequence. Such sequences will not work in the future. Did you mean "\\C"? A raw string is also an option.

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_17000\806865310.py:2:
SyntaxWarning: "\C" is an invalid escape sequence. Such sequences will not
work in the future. Did you mean "\\C"? A raw string is also an option.
  print("n\Classification Report:\n" , classification_report(Y_test, Y_pred))


# CONFUSION MATRIX
cm = confusion_matrix(Y_test, Y_pred)
sns.heatmap(cm , annot= True, fmt='d' , cmap='Blues' ,
            xticklabels=['Non-Renewable', 'Renewable'],
            yticklabels=['Non-Renewable', 'Renewable'])
plt.xlabel('Predictable')
plt.ylabel('Actual')
plt.title('confusion_matrix')
plt.show()
```

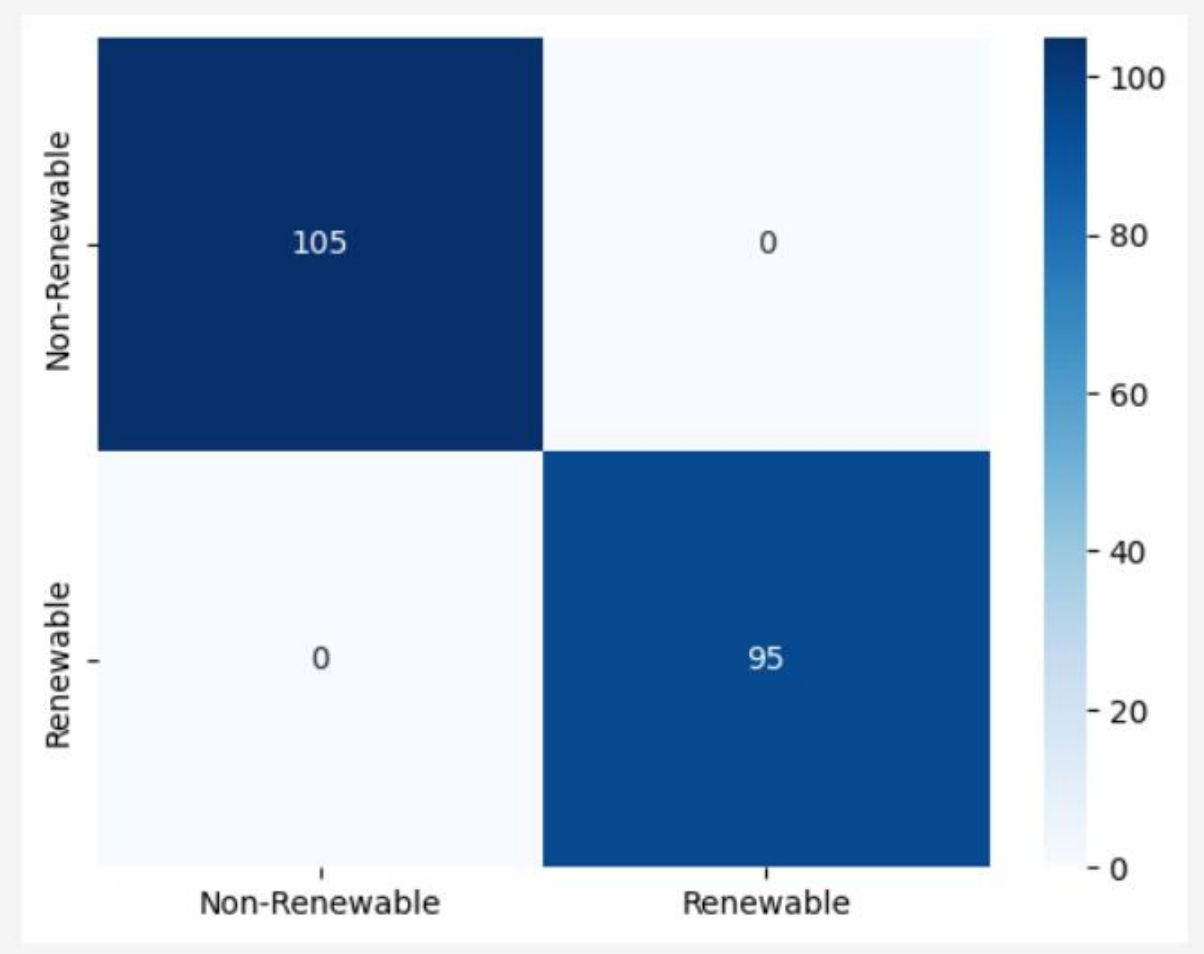

Fig : 6 Train Model

9.2) OUTPUT:

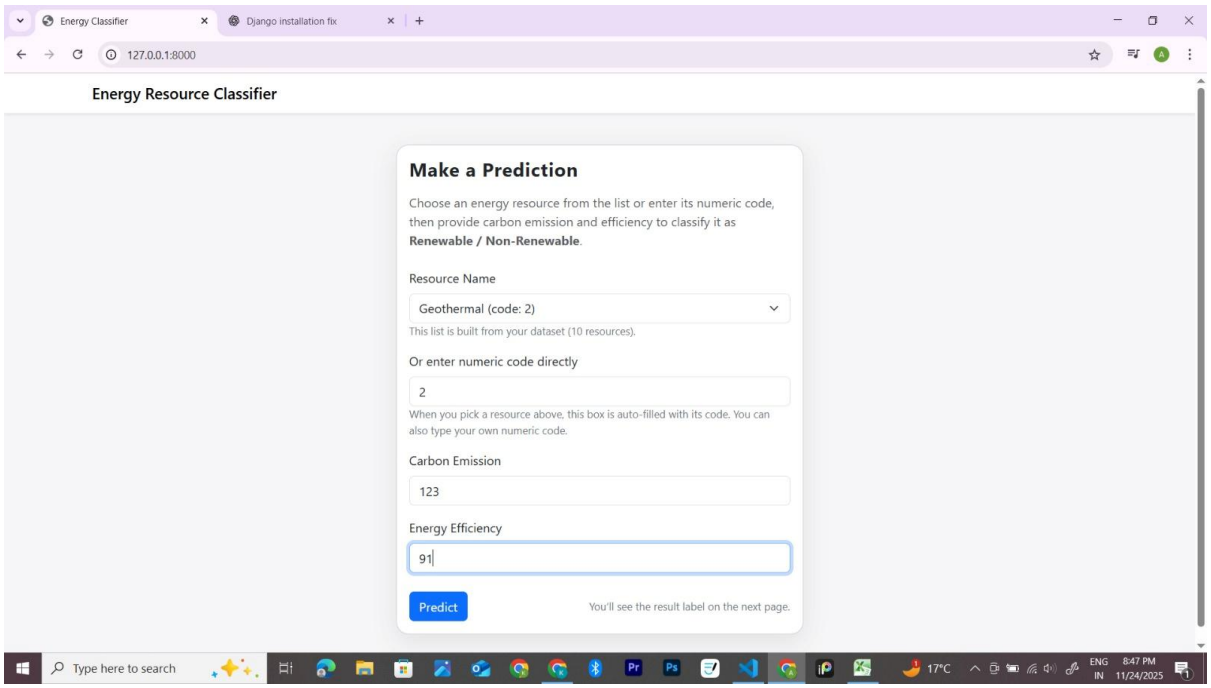

Fig : 7

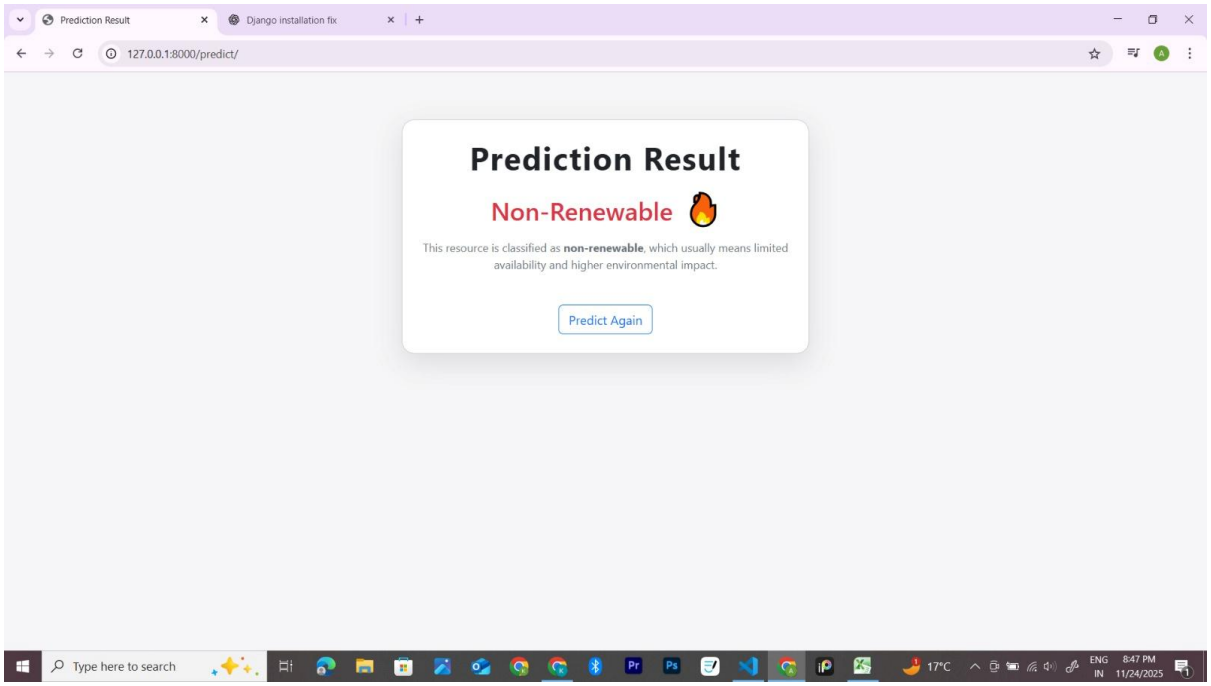

Fig : 8

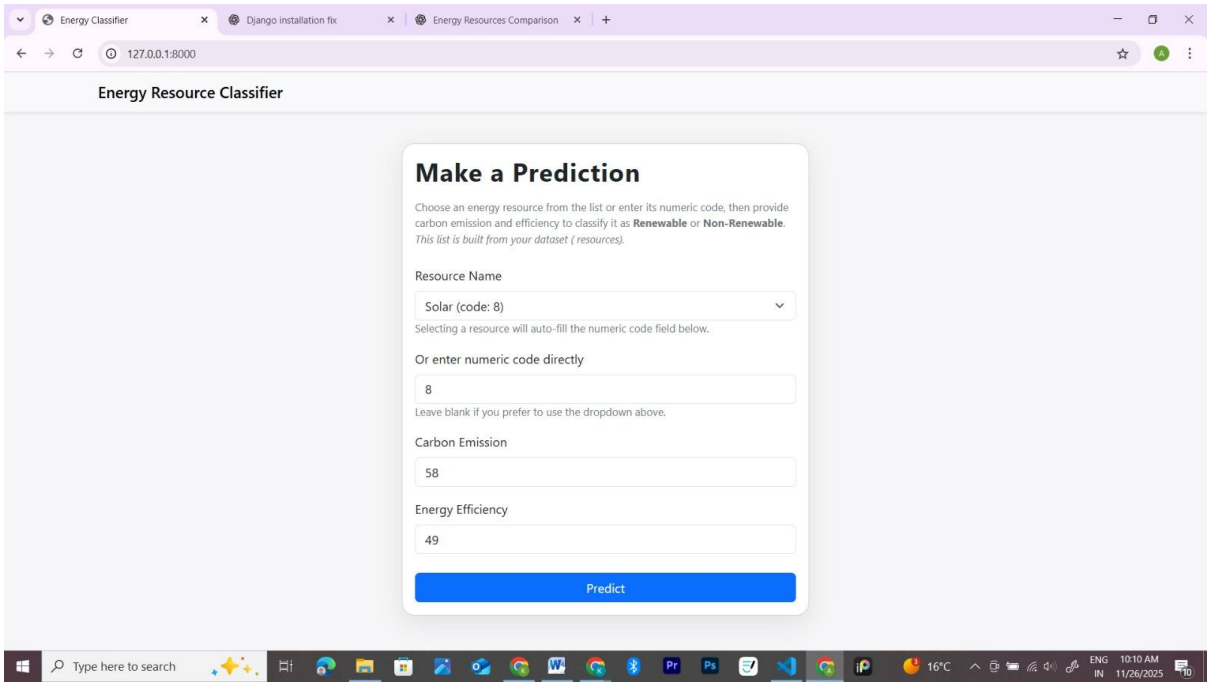
Energy Classifier
Django installation fix
Energy Resources Comparison
127.0.0.1:8000
Energy Resource Classifier
Make a Prediction
Choose an energy resource from the list or enter its numeric code, then provide carbon emission and efficiency to classify it as Renewable or Non-Renewable.
This list is built from your dataset (resources).
Resource Name
Solar (code: 8)
Selecting a resource will auto-fill the numeric code field below.
Or enter numeric code directly
8
Leave blank if you prefer to use the dropdown above.
Carbon Emission
58
Energy Efficiency
49
Predict
Type here to search
16°C
ENG
IN
10:10 AM
11/26/2025

Fig : 9

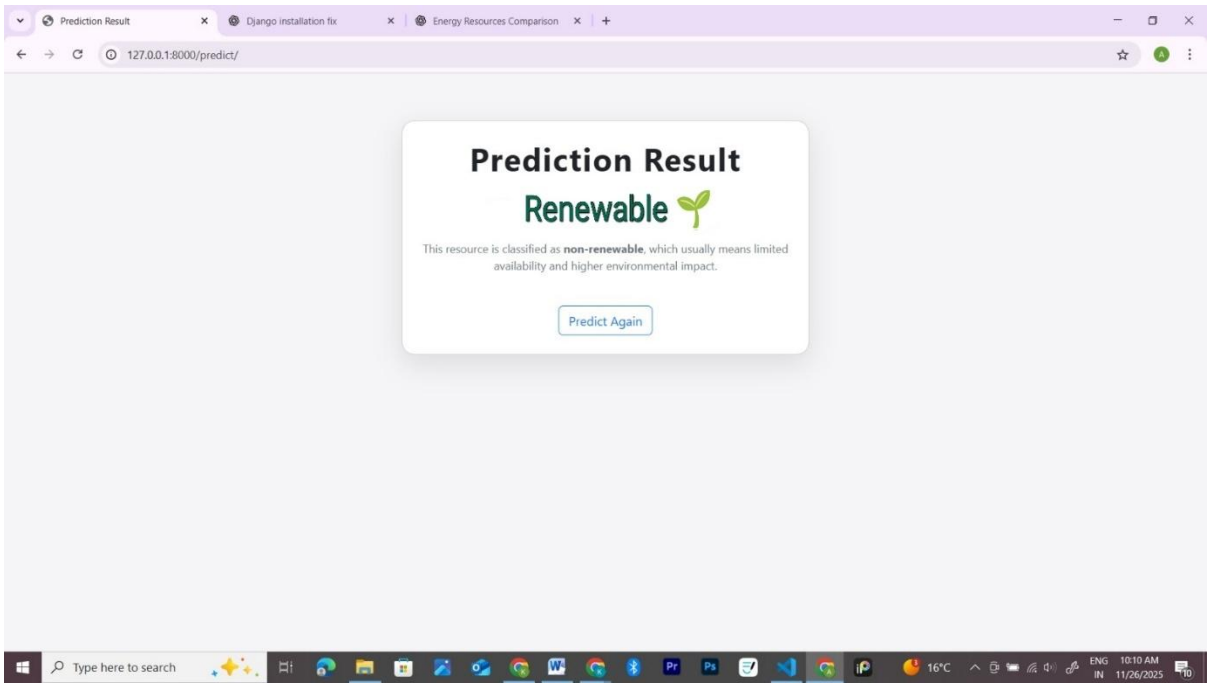
Prediction Result
Django installation fix
Energy Resources Comparison
127.0.0.1:8000/predict/
Prediction Result
Renewable
This resource is classified as non-renewable, which usually means limited availability and higher environmental impact.
Predict Again
Type here to search
16°C
ENG
IN
10:10 AM
11/26/2025

Fig : 10

CHAPTER 8 – SUMMARY & CONCLUSION

8.1 Summary

The project **"Growing Energy Needs: Renewable and Non-Renewable Energy Resources"** was developed to analyze and visualize the rising demand for energy and to compare renewable and non-renewable energy usage using an interactive web-based platform.

The system provides:

- Source-wise and time-based **energy analytics**
- Visual comparison of **renewable vs non-renewable energy resources**
- **Graphs and charts** that make complex data easy to understand
- Awareness content on **energy conservation and environmental ethics**

Using **HTML, CSS, JavaScript, Chart.js, and Python**, the project demonstrates the integration of IT tools with environmental and energy studies. It serves as an educational platform that can be used by students, teachers, and individuals to understand the importance of sustainable energy choices.

8.2 Conclusion

From this project, the following conclusions can be drawn:

1. **Growing Energy Demand:** Energy usage is increasing continuously, and if this demand continues to be met primarily through non-renewable resources, it will lead to severe environmental and resource challenges.

2. **Need for Renewable Energy:** Shifting towards **renewable energy resources** is essential to ensure long-term energy security, reduce pollution, and mitigate climate change.

3. **Role of Awareness and Ethics:** Technical solutions alone are not enough. People must be aware of the **ethical responsibility** to use energy wisely and conserve resources for future generations.

4. **Power of Visualization:** Interactive charts and dashboards make it much easier for users to understand patterns, trends, and comparisons in energy data.

5. **IT as an Enabler of Sustainability:** The project shows that IT tools can support **policy, education, and awareness** by presenting environmental and energy data in a meaningful and accessible way.

Overall, this system stands as both a **technical demonstration** and an **educational tool**, promoting sustainable energy practices and responsible environmental behavior.

CHAPTER 9 – LIMITATIONS OF THE PROJECT

Despite its usefulness, the project has certain limitations:

1. **Static or Sample Data**
    - The current system may rely on sample or static datasets rather than real-time data.

2. **Limited Geographic Scope**
    - The system may focus on a single country/region or simulated data, not global coverage.

3. **No Real-Time Integration**
    - Live integration with official energy APIs is not implemented in the basic version.

4. **No Predictive Modeling**
    - The system does not currently predict future energy demand or simulate scenarios.

5. **No Mobile Application**
    - The platform is browser-based and does not have a dedicated mobile app.

6. **Limited Behavioral Metrics**
    - It does not directly measure specific user behavior (like household energy usage) due to data limitations.

7. **Manual Data Updates**
    - Adding new data requires manual updates, which may be time-consuming.

CHAPTER 10 – FUTURE DIRECTIONS

The project can be extended and enhanced in several ways:

1. **Real-Time Data Integration**
    - Connect to live energy data APIs from government or international agencies.

2. **Predictive Analytics**
    - Use machine learning algorithms to forecast **future energy demand** and renewable penetration.

3. **Mobile Application Development**
    - Create Android/iOS apps for easy and frequent user access.

4. **Advanced Analytics**
    - Add features such as:
        - CO_2 emissions estimation
        - Cost analysis of different energy mixes
        - Scenario modeling (e.g., “what if 50% energy comes from renewables?”)

5. **Expanded Geographic Coverage**
    - Include multiple countries or regions for comparative analysis.

6. **Interactive Learning Module**
    - Add quizzes, interactive tutorials, and case studies for students.

7. **Alert and Notification System**
    - Notify users about important changes, such as major increases in fossil fuel usage or milestones in renewable energy adoption.

REFERENCES

You can adapt these to your chosen citation style:

1. **International Energy Agency (IEA)** – Global Energy Statistics

   Website: https://www.iea.org/

2. **Ministry of Power, Government of India** – Energy Statistics and Reports

   Website: https://powermin.gov.in/

3. **Ministry of New and Renewable Energy (MNRE), Government of India**

   Website: https://mnre.gov.in/

4. **International Renewable Energy Agency (IRENA)** – Renewable Energy Data and Publications

   Website: https://www.irena.org/

5. IPCC. *Climate Change Reports* – Intergovernmental Panel on Climate Change.
   Website: https://www.ipcc.ch/

6. Boyle, G. (Ed.). (2012). *Renewable Energy: Power for a Sustainable Future*. Oxford University Press.

7. DesJardins, J. R. (2013). *Environmental Ethics: An Introduction to Environmental Philosophy*. Wadsworth Publishing.

8. Brundtland Commission. (1987). *Our Common Future*. United Nations.

9. Tiwari, G. N., & Ghosal, M. K. (2005). *Fundamentals of Renewable Energy Sources*. Narosa Publishing.