**Weather-Based Prediction of Wind Turbine Energy Output**

**1. Introduction**

* **Project Title:** WindCore (A Weather-Based Predictor of Wind Turbine Energy Output)
* **Team Leader:** Anudeep Reddy Bandi
* **Team Members:** 
  + Giri Kumar Pandaram
  + Nagaraju Pasupula

**2. Project Overview**

* **Purpose:**The purpose of this project is to build a machine learning–based system that predicts wind turbine energy output using historical turbine data and live weather inputs.
* **Features:**
  + Data preprocessing and cleaning.
  + Random Forest regression model for prediction.
  + Flask-based web dashboard for user interaction.
  + Integration with OpenWeather API for real-time weather data.
  + Visualization of actual vs predicted power outputs.

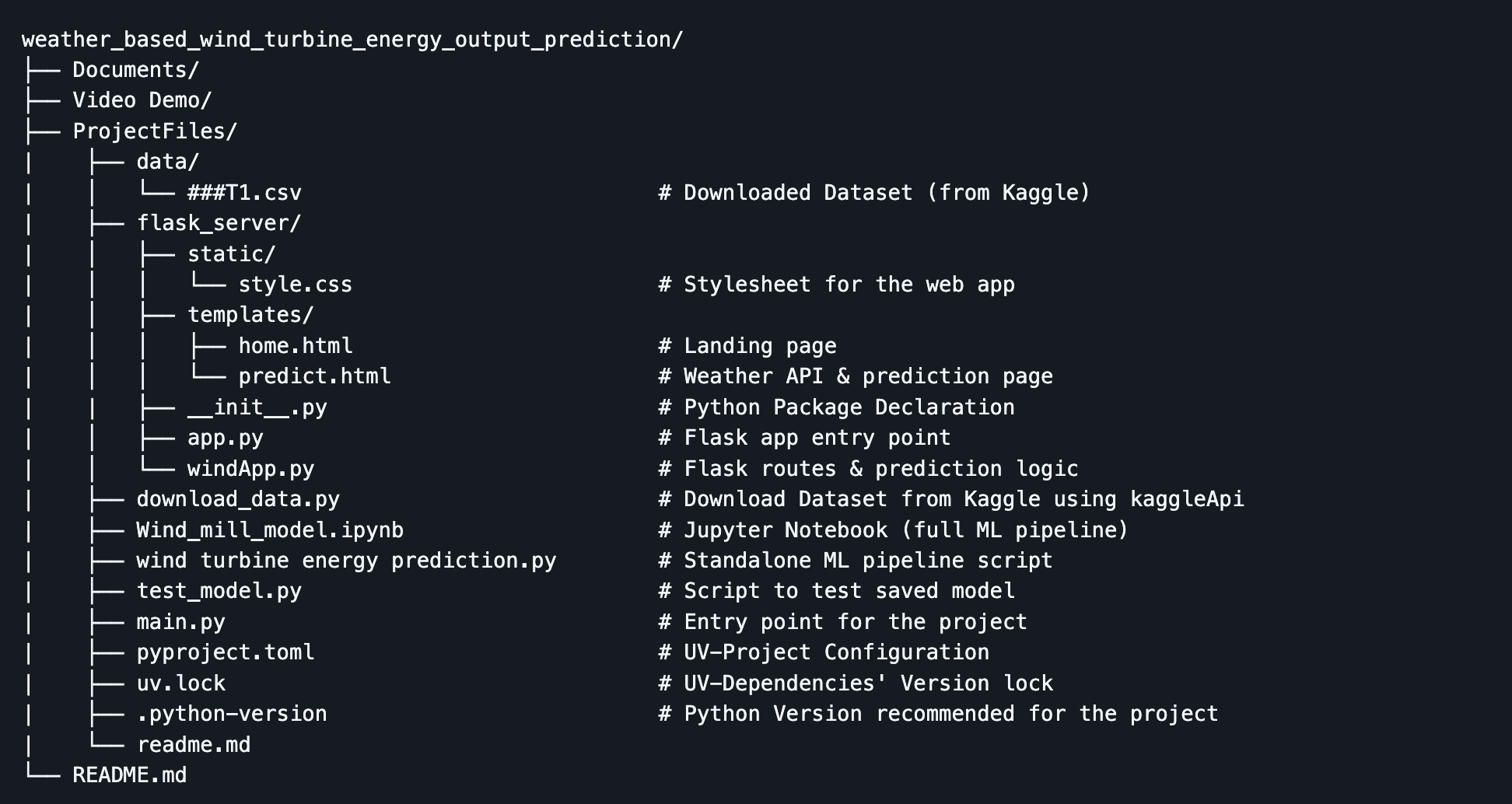
**3. Architecture**

* **Frontend:**Flask templates (HTML, CSS, JavaScript) used for UI design and dashboard visualization.
* **Backend:**Python Flask application handling API requests, ML model predictions, and weather data integration.
* **Database:**Local CSV dataset (T1.csv) for training and testing. Model stored as .sav file using Joblib. Future scope includes cloud database integration (MongoDB Atlas / AWS RDS).

**4. Setup Instructions**

* **Prerequisites:**
  + Python 3.10.x
  + Flask
  + Pandas, NumPy, Scikit-learn, Matplotlib
  + Joblib
  + OpenWeather API key
  + Kaggle API
* **Installation:**
  + Clone the repository.
  + Install required python version using “uv python install”
  + Install dependencies using “uv sync”
  + Set up .env file (API key for OpenWeather, Kaggle Username& key).
  + Run the Flask server with “uv run python main.py full”

**5. Folder Structure**

****

**6. Running the Application**

* **Frontend:** Rendered automatically by Flask Backend
* **Backend:** Start with “uv run python main.py serve”.
* **Access at** [**http://127.0.0.1:5000/**](http://127.0.0.1:5000/)**.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**7. API Documentation**

* **Endpoint 1: ‘/’ (Landing Page)**
  + Rendered Page: **home.html**
  + Method: GET
* **Endpoint 2: ‘/predict’**
  + Rendered Page: **predict.html**
  + Method: GET
  + Possible Rendering options(one of these):
    - No parameters
    - temp, humid, pressure and speed
    - prediction\_text – Energy Prediction result
* **Endpoint 3: ‘/windapi’**
  + Rendered Page: **predict.html**
  + Method: POST
  + Renders predict page with weather data attached.
* **Endpoint 4: ‘/y\_predict’**
  + Rendered Page: **predict.html**
  + Method: POST
  + Renders predict page with prediction result.

**8. Authentication**

* Currently open access.
* API key required for OpenWeather API integration and Kaggle API for dataset.
* Future scope: OAuth2 + SuperTokens to manage Rotating Tokens and Sessions.

**9. User Interface**

* Intro page with project overview.
* Dashboard with:
  + Weather data display.
  + Prediction module.
  + Visualization graphs.

**A screenshot of a computer

AI-generated content may be incorrect.**

**10. Testing**

* Unit testing for ML model predictions.
* API testing for weather data retrieval.
* Functional testing for input validation and dashboard navigation.

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**11. Screenshots or Demo**

* Scatter plot of actual vs predicted power.
* Dashboard screenshot showing weather + prediction results.
* Line chart trends of wind speed vs power output.

**12. Known Issues**

* Limited dataset size may affect generalization.
* API errors if invalid city names are entered.
* No authentication for dashboard access (future enhancement needed).

**13. Future Enhancements**

* Deploy on cloud (AWS/GCP/Azure).
* Add OAuth2 + SuperTokens authentication for secure access.
* Expand dataset for improved accuracy.
* Add more visualizations (heatmaps, time-series forecasting).
* Integrate grid demand APIs for real-time energy balancing.

**14. Conclusion**

The **Wind Turbine Energy Prediction project** successfully demonstrates the application of machine learning and data-driven techniques to solve a real-world renewable energy challenge. By leveraging historical turbine data, preprocessing methods, and a **Random Forest regression model**, the project achieves strong predictive accuracy, enabling stakeholders to forecast energy output with confidence.

The integration of a **Flask-based dashboard** and **OpenWeather API** ensures that predictions are not only technically sound but also accessible and user-friendly. Visualizations such as scatter plots, line charts, and correlation heatmaps further enhance interpretability, making the solution practical for energy companies, wind farm operators, and grid managers.

Through structured **Agile sprint planning, backlog management, and performance testing**, the project was executed with clarity and measurable progress. Testing confirmed the robustness of the model, while defect analysis and bug tracking highlighted areas for improvement.

Ultimately, this project provides a scalable foundation for future enhancements, including cloud deployment, advanced authentication, expanded datasets, and integration with grid demand APIs. It stands as a strong example of combining **data science, software engineering, and agile methodology** to deliver a solution that addresses both technical and customer-centric needs in the renewable energy sector.