Date	1 oct 2025
Team ID	LTVIP2025TMIDS40838
Project Name	Weather-Based Prediction of Wind Turbine Energy Output: A Next-Generation Approach to Renewable Energy
Maximum Marks	3 Marks

Phase 5: Project Executable Files (Expanded)

1. Overview

This phase documents all executable components of the Wind Turbine Energy Prediction project, including:

- **Flask Web Application** for user interaction
- **Jupyter Notebooks** for data preprocessing, model training, testing, and deployment

The goal is to provide a fully functional system that predicts wind energy output based on real-time or historical weather data.

---

2. Flask Web Application

Purpose:

- To allow end-users to input weather parameters and receive predicted wind turbine energy output in real-time.

Features:

- **Input:** Wind speed, temperature, humidity, air pressure, and optional date/time for historical prediction
- **Output:** Estimated energy generation (kWh or MW)
- **Interface:** Clean and user-friendly web interface with responsive design
- **Integration:** Connects directly to the trained machine learning model

Workflow:

1. User enters weather parameters in the web form.
2. Flask backend receives the input and preprocesses it.
3. Preprocessed data is fed into the trained ML model.
4. Model predicts energy output and returns the result on the web page.

Technology Stack:

- Python (Flask framework)
- HTML/CSS for interface
- Pickle/Joblib for loading trained model
- Bootstrap or simple CSS for layout

---

3. Jupyter Notebooks

a) model_training.ipynb

- **Purpose:** Preprocess the data and train machine learning models.
- **Contents:**
    - Importing libraries (Pandas, NumPy, Scikit-learn)
    - Data cleaning, normalization, feature engineering
    - Model selection (Linear Regression, Random Forest, Gradient Boosting, XGBoost)
    - Evaluation using RMSE, MAE, R^2
    - Saving the best-performing model using Pickle/Joblib

b) model_testing.ipynb

- **Purpose:** Test the trained model on unseen data to validate predictions.

- **Contents:**
    - Loading the saved model
    - Input test data for prediction
    - Compare predicted vs actual energy output
    - Generate plots for visual comparison (scatter plots, line charts)
    - Performance metrics calculation for model evaluation

c) Flask_App.ipynb (or app.py)

- **Purpose:** Integrate the trained model into a Flask application for deployment.
- **Contents:**
    - Flask app setup (Flask(__name__))
    - Routes for home page and prediction endpoint
    - Input form processing and validation
    - Model prediction and output rendering
    - Launching the app locally for testing (app.run(debug=True))

4. File Structure (Example)

```
Turbine_Project/
|
├– notebooks/
|   ├– model_training.ipynb
|   ├– model_testing.ipynb
|   └ Flask_App.ipynb
|
├– templates/
|   └ index.html
├– static/
|   └ style.css
```

```
├─ saved_model/
|   └─ wind_model.pkl
└─ README.md
```

---

5. Expected Outcome

- Fully functional **web application** capable of predicting wind turbine energy output.
- **Reproducible machine learning workflow** from data preprocessing to model deployment.
- **Interactive interface** for stakeholders to input weather data and get predictions.
- Clear documentation of model performance, preprocessing steps, and deployment setup.

---

6. Additional Notes

- The Flask app can be **deployed on local machines or cloud platforms** (e.g., Heroku).
- The notebooks allow for **continuous model improvement** by retraining with new data.
- The project is designed to be **scalable**, allowing integration with multiple turbines or farms.