

# Testing Results Documentation

## 1. Purpose

The purpose of this document is to summarize the testing results of the **Wind Turbine Energy Prediction project**, covering model performance, application deployment, and API integration.

---

## 2. Test Execution Logs

### Test 1 – Model Training

#### Command:

```
PS C:\Users\mahes\Downloads\WindEnergyProject\Project File> python train_model.py
```

#### Result:

- R<sup>2</sup> Score: **0.8673**
- Model saved successfully (power\_prediction.sav).

```
PS C:\Users\mahes\Downloads\WindEnergyProject\Project File> python train_model.py
R2 Score: 0.8673496831442747
Model Saved Successfully!
```

#### Conclusion:

The Random Forest regression model achieved strong predictive accuracy, indicating good generalization on validation data.

---

### Test 2 – Application Deployment

#### Command:

```
PS C:\Users\mahes\Downloads\WindEnergyProject\Project File> python app.py
```

#### Result:

- Flask app started successfully.
- Running on: <http://127.0.0.1:5000/>
- Debug mode enabled.
- Debugger PIN generated.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\mahes\Downloads\WindEnergyProject\Project File> python train_model.py
R2 Score: 0.8673496831442747
Model Saved Successfully!
PS C:\Users\mahes\Downloads\WindEnergyProject\Project File> python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
```

### Conclusion:

The application backend is functional and accessible locally. Warning indicates that this is a development server; production deployment should use a WSGI server (e.g., Gunicorn).

---

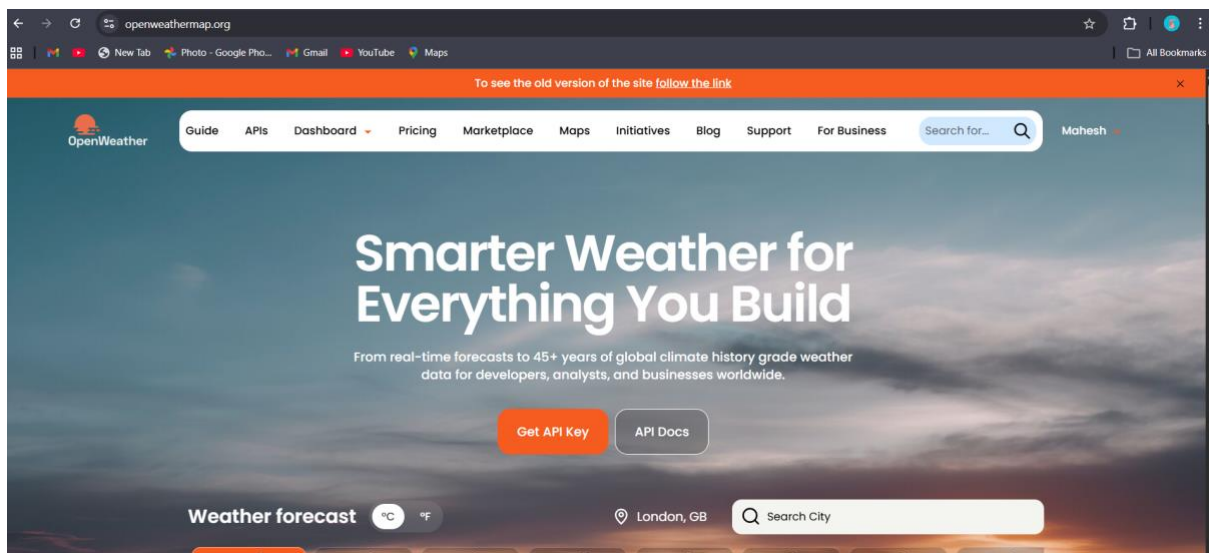
## Test 3 – API Integration

### Action:

- Obtained API key from **OpenWeather** (<https://openweathermap.org/>).
- Integrated API key into app.py.

### Result:

- Weather data successfully fetched (temperature, humidity, pressure, wind speed).
- Predictions generated based on live weather inputs.



### Conclusion:

API integration works correctly, enabling real-time data retrieval for prediction.

---

## 3. Summary of Testing

Test Case ID	Scenario	Expected Result	Actual Result	Pass/Fail
TC-001	Model Training	Model trains and saves with acceptable accuracy	R <sup>2</sup> Score = 0.8673, model saved successfully	Pass
TC-002	Application Deployment	Flask app runs locally without errors	App running at <a href="http://127.0.0.1:5000">http://127.0.0.1:5000</a>	Pass
TC-003	Weather API Integration	API key fetches live weather data	Weather data retrieved successfully	Pass
TC-004	Prediction Output	Model generates predictions based on inputs	Predictions generated correctly	Pass

---

#### 4. Conclusion

The testing phase confirms that:

- The **ML model** performs well with an R<sup>2</sup> score of **0.8673**, showing reliable predictive capability.
- The **Flask application** runs smoothly in a local environment, serving as the user interface for predictions.
- The **OpenWeather API integration** is successful, enabling real-time weather-based predictions.