## Deployment Documentation

### 1. Architecture Overview

This project is a **Phishing URL Detection System** using a trained **Random Forest classifier** deployed through a **Flask web application**. Here's how it works:

**User → Web Interface → Flask Backend (API) → ML Model → Response**

* Users input a URL through a user-friendly web interface (HTML form).
* The Flask backend receives the URL and extracts relevant features.
* These features are passed to the pre-trained machine learning model.
* The model returns a prediction: 1 (Phishing) or 0 (Legitimate).
* The result is displayed back to the user.

### 2. Setup Instructions

#### **📦 Requirements**

Make sure you have Python installed (preferably Python 3.9+), then install all dependencies with:

pip install -r requirements.txt

#### **📁 Project Structure**

ML\_Flask\_App/

├── Dockerfile # dockerfile

├── app.py # Main Flask backend

├── test\_prediction.py # Optional script to test API

├──final\_model.pkl # Trained ML model

├── requirements.txt # Dependencies

├── deployment\_documentation.docx

├── templates/

│ └── index.html # Web UI template

#### **▶️ How to Run the Application**

1. Open a terminal and navigate to the my\_flask\_app/ directory.
2. Activate your Python environment if needed.
3. Start the Flask app:

python app.py

1. Open your browser and go to:

http://127.0.0.1:5000/

1. Enter a URL in the form and press "Check URL" to receive a prediction.

### 3. Monitoring and Maintenance Plan

To ensure the model continues to work effectively:

* **Monitor Model Accuracy:**
  + Periodically evaluate the model with fresh phishing datasets.
  + Update the model if accuracy drops due to changing phishing techniques.
* **Log API Activity:**
  + Add logging to track requests, errors, and predictions for debugging and analytics.
* **Handle Model Updates:**
  + When retraining, update final\_model.pkl.
  + Ensure the new model is compatible with the existing feature extraction logic.
* **Performance Checks:**
  + Monitor response times and user load if deployed online.
* **Security:**
  + Sanitize user inputs to avoid injection attacks.
  + Use HTTPS if deploying beyond local environment.