Phishing Domain Detection

Overview

This document provides a detailed explanation of the architectural design of the **Phishing Domain Detection** system. It explains the interaction between various components such as the REST API, Machine Learning model, and the Web Interface, along with diagrams and descriptions of their functionality.

---

1. Software Architecture

The system is designed to combine the power of Machine Learning (ML) for phishing detection with user-friendly accessibility through a REST API and a Web Interface.

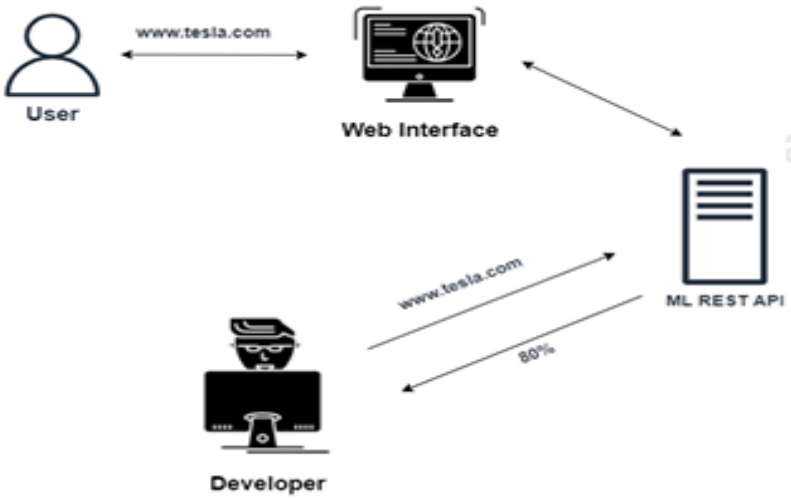
Key Components

1. **Data Input**:
    - URLs are provided by users or developers via the Web Interface or directly through the REST API.
    - Input validation ensures proper URL format and security.
2. **Feature Extraction**:
    - Extracts essential features from the input URL, such as:
        - URL length.
        - Number of special characters (e.g., `@`, `//`).
        - Domain age and WHOIS information.
        - Use of HTTPS.
        - Blacklist status from known phishing databases.
3. **Machine Learning Model**:
    - A supervised learning model trained to classify URLs as phishing or legitimate.
    - Models used: Logistic Regression, Random Forest, Gradient Boosting (e.g., XGBoost).
    - Output: A probability score indicating the likelihood of a URL being phishing.
4. **REST API**:
    - Facilitates communication between the Web Interface and the ML model.
    - Built using Python frameworks such as Flask or FastAPI.
5. **Web Interface**:
    - A flask-based interface for user interaction.
    - Designed to ensure non-technical users can access and utilize the system seamlessly.
6. **Database**:
    - Stores analyzed URLs and their classifications for future reference.
    - Technologies used: SQLite or MongoDB.

7. **Deployment**:
    - The system is hosted on cloud platforms like Heroku, AWS, or Google Cloud for accessibility and scalability.

System Architecture Diagram

(Include a diagram showing data flow from user input to Web Interface to REST API to ML Model and back to the user.)



---

2. REST API Architecture

The REST API acts as the core communication layer between the Machine Learning model and external clients.

Key Features

- **Endpoint**: `/predict`

Input Format: JSON payload

```
{
  "url": "http://example.com"
```

- }

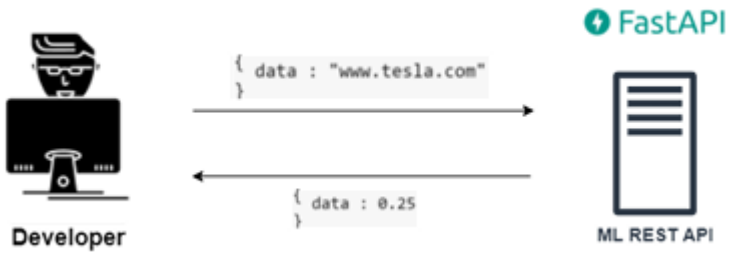
Output Format: JSON response

```
{
  "url": "http://example.com",
  "probability": 0.85,
  "classification": "Phishing"
```

- }

API Workflow

1. User sends a URL in JSON format.
2. REST API validates the input.
3. Features are extracted from the URL.
4. ML model predicts the likelihood of phishing.
5. Response is sent back as a JSON object.



REST API Flow Diagram

(Include a diagram illustrating the request/response cycle: Client -> REST API -> Feature Extraction -> ML Model -> Response.)

---

3. Web Interface Wireframe

The Web Interface is designed to ensure ease of use for non-technical users.

Key Features

1. **Home Page**:
    - **Input Field**: Allows users to input a URL.
    - **Submit Button**: Sends the URL to the REST API for analysis.
2. **Results Page**:
    - Displays the classification result (e.g., Safe/Phishing).
    - Visualizes the probability score using a progress bar.
3. **History Page** (Optional):
    - Shows previously analyzed URLs and their classifications.

Wireframe Diagram

(Include a wireframe depicting the interface: Input Field -> Submit Button -> Result Display.)

---

4. Security Considerations

1. **Data Protection**:
    - Enforces HTTPS for secure communication.
    - Ensures sensitive data like URLs are not stored unnecessarily.
2. **Input Validation**:
    - Validates all inputs to prevent SQL injection or cross-site scripting (XSS).

3. **Model Security**:
    - Protects the ML model from adversarial attacks by sanitizing input features.

---

5. Scalability and Extensibility

1. **Scalability**:
    - Implements load balancing to handle concurrent requests.
    - Uses containerization (e.g., Docker) for deployment on multiple servers.
2. **Extensibility**:
    - Adds capabilities like email phishing detection.
    - Integrates Natural Language Processing (NLP) for analyzing webpage content.

---

6. Future Enhancements

1. **Real-Time Detection**:
    - Analyzing URLs in emails or messages in real time.
2. **Browser Extension**:
    - Developing a Chrome/Firefox extension for phishing warnings.
3. **Mobile App**:
    - Creating an Android/iOS app for on-the-go phishing detection.
    -