📌Phishing Detection System — Complete Final Year Project Guide

This project explains how to build a Phishing Detection System using machine learning to detect phishing websites by analyzing URLs and related metadata. It covers data collection, feature extraction, model training, testing, building an API, and optionally a Chrome extension.

📌 🗂️OVERVIEW: Major Stages\ ✂️ 📑Project Setup\ ✒️ 🗃️Dataset Collection\ 🛏️ 🧰Feature Extraction Module\ 🌂 🧹Data Preprocessing\ 🌡️ 🖍️Model Building & Training\ 🎒 🧪Testing & Evaluation\ 🎓 🩲 Backend API with Flask\ 🎩🧰(Optional) Chrome Extension or Web UI\ 🎽 🗜️Deployment & Demo

🔗**Stage 1 — Project Setup\ Requirements:**

- Python 3.10+
- VS Code or PyCharm
- Git & GitHub
- (Optional) Chrome browser

Python Libraries:

```
pip install pandas numpy scikit-learn beautifulsoup4 requests whois flask
```

Add later if needed:

```
pip install streamlit matplotlib seaborn
```

Folder Structure:

```
phishing_detection_system/
  ├── data/
  ├── notebooks/
  ├── feature_extractor/
  ├── model/
  ├── api/
  ├── extension/
  ├── requirements.txt
  ├── README.md
```

🔗**Deliverables for Stage 1:** Python environment, folders, Git repo.

🔗**Stage 2 — Dataset Collection**\ Sources: PhishTank, OpenPhish, Kaggle for phishing; Alexa/Tranco for legit URLs.

```
URL, Label
http://malicioussite.com, phishing
http://google.com, legit
```

🔗**Deliverables for Stage 2:** data/raw/phishing.csv, data/raw/legit.csv.

🔗**Stage 3 — Feature Extraction Module**\ Extract: URL length, symbols, IP presence, HTTPS, subdomains, keywords, WHOIS domain age.

```python
from urllib.parse import urlparse
import whois

def extract_features(url):
    features = {}
    features['url_length'] = len(url)
    features['has_at'] = '@' in url
    parsed = urlparse(url)
    features['has_https'] = parsed.scheme == 'https'
    features['has_ip'] = parsed.netloc.replace('.', '').isdigit()
    features['num_subdomains'] = len(parsed.netloc.split('.')) - 2
    features['has_dash'] = '-' in parsed.netloc
    try:
        domain_info = whois.whois(parsed.netloc)
        if domain_info.creation_date:
            age = (domain_info.expiration_date - domain_info.creation_date).days
            features['domain_age'] = age
        else:
            features['domain_age'] = 0
    except:
        features['domain_age'] = 0
    return features
```

🔗**Deliverables for Stage 3:** feature_extractor/extract_features.py, tested.

🔗**Stage 4 — Data Preprocessing**\ Run extractor, save as features.csv, encode booleans, normalize.

```
url_length, has_at, has_ip, has_https, num_subdomains, domain_age, label
45, 0, 1, 0, 3, 0, phishing
```

🔗**Deliverables for Stage 4:** data/features.csv, notebooks/EDA.ipynb.

🔗**Stage 5 — Model Building & Training**\ Split, train (Logistic Regression, Random Forest, XGBoost), tune, save.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import joblib

# Load data
df = pd.read_csv('data/features.csv')
X = df.drop('label', axis=1)
y = df['label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = RandomForestClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(accuracy_score(y_test, y_pred))
joblib.dump(model, 'model/phishing_model.pkl')
```

🔗**Deliverables for Stage 5:** model/train_model.py, model/phishing_model.pkl.

🔗**Stage 6 — Testing & Evaluation**\ Check accuracy, precision, recall, confusion matrix. 🔗**Deliverables for Stage 6:** notebooks/evaluation.ipynb, confusion matrix.

🔗**Stage 7 — Backend API with Flask**\ Serve predictions with Flask.

```
from flask import Flask, request, jsonify
import joblib
from feature_extractor import extract_features

app = Flask(__name__)
model = joblib.load('model/phishing_model.pkl')

@app.route('/predict', methods=['POST'])
def predict():
    url = request.json['url']
    features = extract_features(url)
    input_data = [[features[k] for k in sorted(features)]]
    prediction = model.predict(input_data)
    return jsonify({'prediction': prediction[0]})

if __name__ == '__main__':
    app.run(debug=True)
```

Test:

```
curl -X POST http://127.0.0.1:5000/predict -H "Content-Type: application/json" -
d '{"url":"http://suspicious.site"}'
```

🔗**Deliverables for Stage 7:** api/app.py, test POST.

🔗**Stage 8 — Optional Chrome Extension**\ Use JavaScript to grab URL, call API. 🔗**Deliverables for Stage 8:** extension/manifest.json, popup.html, background.js.

🔗 **Stage 9 — Deployment & Demo**\ Deploy Flask API (Heroku/Render). Write README.md with usage, screenshots.

👙**Final Deliverables:** Classifier, extractor, API, optional extension, demo, repo.