EX.NO.: 03
DATE: 30.06.2025

PHISHING DETECTION WEB APPLICATION

AIM

To develop a machine learning-based web application that detects phishing attempts across Email, SMS, and URL inputs by leveraging separately trained classifiers, providing real-time predictions through an interactive Flask-based user interface.

ALGORITHM

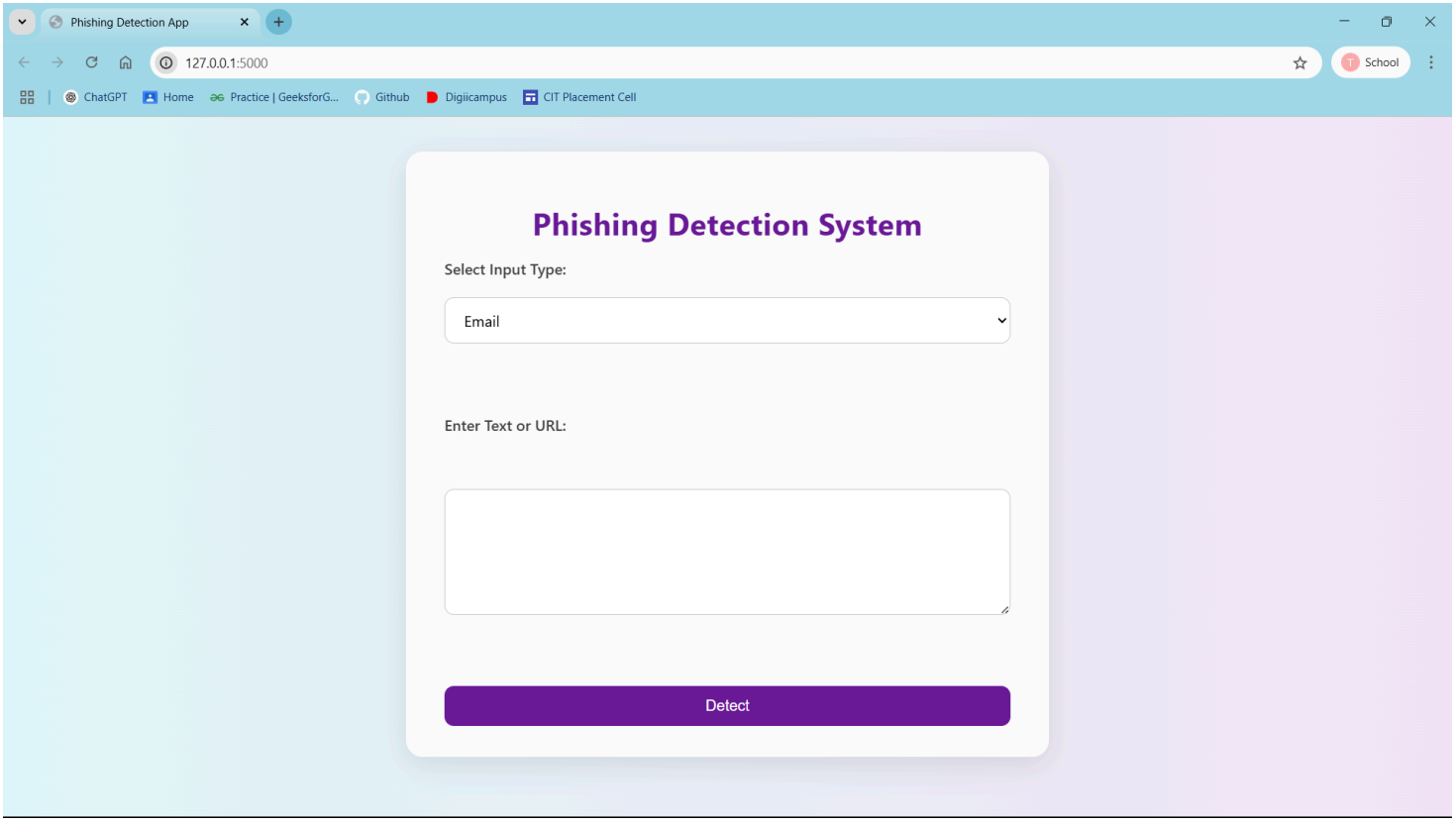
1. **Data Collection**: Gather labeled datasets for emails, SMS messages, and URLs from public sources like Kaggle.
2. **Data Preprocessing**:
    a. For emails: Combine subject and body fields.
    b. For SMS: Convert labels from "ham/spam" to binary (0/1).
    c. For URLs: Clean and tokenize raw URLs or extract features.
3. **Text Vectorization**: Use `CountVectorizer` to convert textual data into numerical format.
4. **Model Training**:
    a. Train individual models: Naive Bayes (for email/SMS), Random Forest (for URLs).
    b. Split data into training and test sets using `train_test_split`.
5. **Model Saving**: Persist trained models and vectorizers using `pickle`.
6. **Web Integration**:
    a. Build a Flask app to accept user input via a form.
    b. Load the correct model and vectorizer based on user input type.
    c. Predict and display whether the input is "Phishing" or "Legitimate".
7. **UI Presentation**: Present the prediction with a clean and styled web dashboard.

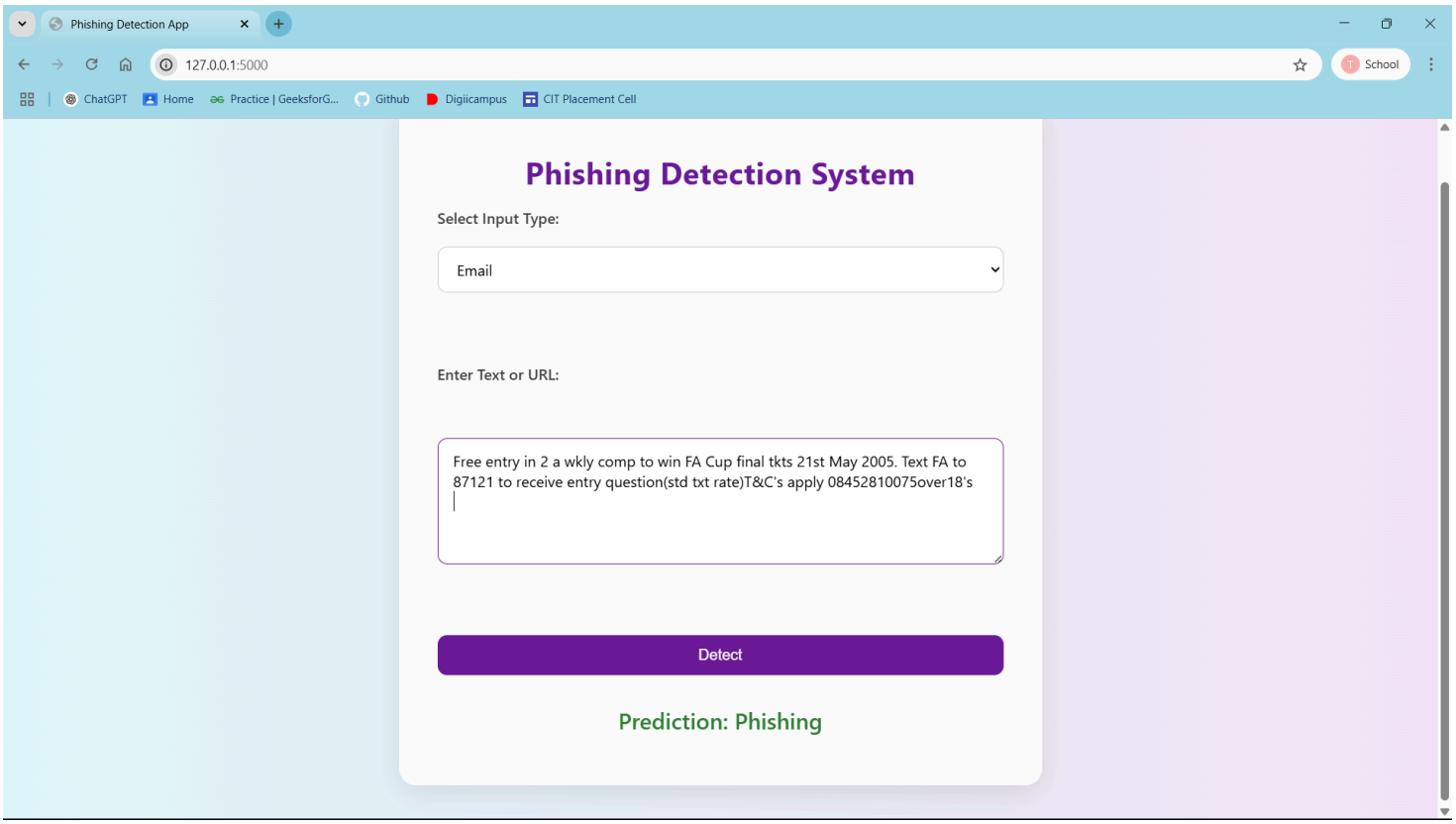
CODE AND OUTPUT

```
from flask import Flask, render_template, request
import pickle

app = Flask(__name__)

# Load pickled models/vectorizers
email_model = pickle.load(open('models/email_model.pkl', 'rb'))
email_vectorizer = pickle.load(open('models/email_vectorizer.pkl', 'rb'))
sms_model = pickle.load(open('models/sms_model.pkl', 'rb'))
sms_vectorizer = pickle.load(open('models/sms_vectorizer.pkl', 'rb'))
url_model = pickle.load(open('models/url_model.pkl', 'rb'))
url_vectorizer = pickle.load(open('models/url_vectorizer.pkl', 'rb'))

@app.route('/', methods=['GET', 'POST'])
def index():
    result = None
    if request.method == 'POST':
        data_type = request.form['data_type']
        user_input = request.form['user_input']
        if data_type == 'email':
            vect = email_vectorizer.transform([user_input])
            pred = email_model.predict(vect)[0]
        elif data_type == 'sms':
```

```
            vect = sms_vectorizer.transform([user_input])
            pred = sms_model.predict(vect)[0]
        elif data_type == 'url':
            vect = url_vectorizer.transform([user_input])
            pred = url_model.predict(vect)[0]
        else:
            pred = None
        if pred is not None:
            result = 'Phishing' if pred == 1 else 'Legitimate'
    return render_template('index.html', result=result)

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

```
import os, pickle
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

def train_save(text_col, label_col, dataset_path, model_path, vec_path,
model_type="nb"):
    df = pd.read_csv(dataset_path)
    X, y = df[text_col], df[label_col]
    vec = CountVectorizer()
    X_vec = vec.fit_transform(X)
    X_train, X_test, y_train, y_test = train_test_split(X_vec, y, test_size=0.2,
random_state=42)
    if model_type == "rf":
        model = RandomForestClassifier(n_estimators=100, random_state=42)
    else:
        model = MultinomialNB()
    model.fit(X_train, y_train)
    print(f"Saved {model_type} model to {model_path}")
    pickle.dump(model, open(model_path, 'wb'))
    pickle.dump(vec, open(vec_path, 'wb'))

os.makedirs('models', exist_ok=True)

train_save('body', 'label', 'datasets/email_dataset_1.csv', 'models/email_model.pkl',
'models/email_vectorizer.pkl', 'nb')
train_save('text', 'label', 'datasets/sms_dataset_1.csv', 'models/sms_model.pkl',
'models/sms_vectorizer.pkl', 'nb')
train_save('URL', 'Label', 'datasets/url_dataset_1.csv', 'models/url_model.pkl',
'models/url_vectorizer.pkl', 'rf')
```





INFERENCE

The phishing detection system effectively distinguishes between phishing and legitimate inputs in real-time across three formats by using specialized machine learning models, improving usability and cyber safety through a unified and accessible web interface.