**🛡️ AI-Powered Phishing Email Detector (React + FastAPI + Vercel + Firebase/Google Cloud Run)**

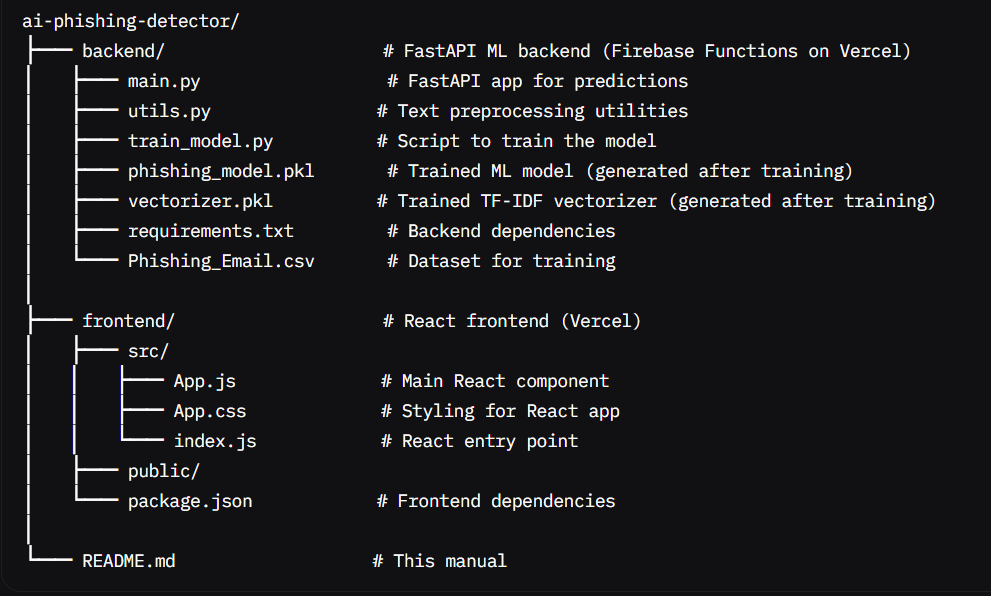
**🚀 Overview**

This project implements an **AI-based phishing email detection system** using a machine learning model trained on the Phishing\_Email.csv dataset. Key features include:

* ✅ A Python **FastAPI** backend to serve predictions, deployed on **Google Cloud Run** (integrated with Firebase)
* ✅ A **React** frontend for user interaction, deployed via **Vercel**
* ✅ A training pipeline using scikit-learn and nltk for text processing
* ✅ Deployment-ready setup with Firebase/Google Cloud Run (backend) and Vercel (frontend)

The system requires retraining the model using the provided Phishing\_Email.csv dataset to generate phishing\_model.pkl and vectorizer.pkl before deployment.

**📁 Project Structure**

****

**📊 Dataset Format (CSV)**

The dataset, Phishing\_Email.csv, must be placed in the backend/ folder and retrained to generate the model and vectorizer files. It contains email texts and their corresponding labels.

**✅ Format:**

|  |  |
| --- | --- |
| **Email Text** | **Email Type Level** |
| "Click here to verify your account immediately!" | Phishing Email High/Low/Medium |
| "Team, please find the report attached." | Safe Email - |

* **Email Text**: The combined subject and body of the email (string).
* **Email Type**: Phishing Email (malicious) or Safe Email (legitimate).

The dataset includes professional correspondence (Safe) and suspicious emails with links or offers (Phishing). **You must retrain the model using** Phishing\_Email.csv **to ensure the model and vectorizer are up-to-date.**

**🧠 Training the Model**

1. **Install dependencies**:

pip install pandas scikit-learn nltk joblib

1. **Train the model**: Place the following script in backend/train\_model.py to train the model using Phishing\_Email.csv. **Run this script to retrain the model and generate** phishing\_model.pkl **and** vectorizer.pkl**.**

import pandas as pd from sklearn.model\_selection import train\_test\_split from sklearn.ensemble import RandomForestClassifier from sklearn.feature\_extraction.text import TfidfVectorizer import joblib from utils import clean\_text

**Load and preprocess the dataset**

df = pd.read\_csv("Phishing\_Email.csv") df["cleaned"] = df["Email Text"].apply(clean\_text) df["label"] = df["Email Type"].map({"Phishing Email": 1, "Safe Email": 0})

**Split data**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df["cleaned"], df["label"], test\_size=0.2, random\_state=42)

**Vectorize text**

vectorizer = TfidfVectorizer(max\_features=5000) X\_train\_vect = vectorizer.fit\_transform(X\_train)

**Train model**

model = RandomForestClassifier(n\_estimators=100, random\_state=42) model.fit(X\_train\_vect, y\_train)

**Save model and vectorizer**

joblib.dump(model, "phishing\_model.pkl") joblib.dump(vectorizer, "vectorizer.pkl")

**Evaluate model**

X\_test\_vect = vectorizer.transform(X\_test) accuracy = model.score(X\_test\_vect, y\_test) print(f"Model accuracy: {accuracy:.2f}")