Site Web de Détection de Contenus Malveillants sur Réseaux Sociaux : PhishBlocker

1. Contexte

Les réseaux sociaux sont devenus des plateformes incontournables pour la communication, que ce soit pour les particuliers ou les entreprises. Cependant, ils sont aussi des terrains fertiles pour des **activités malveillantes**, dont l'**escroquerie** est une des plus répandues.

Une **étude de marché** a permis de mettre en lumière des chiffres inquiétants :

- **63%** des jeunes de moins de 13 ans possèdent déjà un compte sur un réseau social, ce qui les expose davantage aux risques d’escroquerie.
- Les **16-25 ans** passent en moyenne **3 à 5 heures par jour** sur les réseaux sociaux, augmentant leur vulnérabilité aux arnaques en ligne.
- **1 jeune sur 5** a déjà été victime de harcèlement sur les réseaux sociaux, un phénomène qui s’accompagne souvent d’usurpations d’identité et d’autres formes de fraude

2. Les missions :

1. Veille Technologique IA

Exploration et évaluation des meilleures solutions en traitement du langage naturel (NLP) et analyse d'images pour la détection de phishing.

2. Développement Frontend et Backend

Frontend : Angular est un framework front-end qui permet de créer des applications web dynamiques et évolutives. Voici les étapes à suivre pour créer une interface conforme aux normes d'accessibilité :

- **Création de Composants Angular** : Divise l'interface utilisateur en plusieurs composants réutilisables, chacun avec sa logique et son template HTML.
    - Exemples : `navbar`, `footer`, `main-content`, `formulaire`, etc.
- **Gestion de la Navigation** : Utilise le module `Angular Router` pour gérer la navigation entre les différentes pages ou sections de l'application.
    - Exemples : des routes comme `/dashboard`, `/settings`, `/profile`.
- **Accessibilité (WCAG)** :
    - Ajoute des attributs ARIA pour aider les utilisateurs avec des handicaps à mieux interagir avec l'interface (e.g., `aria-label`, `role`).
    - Assure une bonne navigation au clavier.
    - Garantir un contraste suffisant pour les utilisateurs malvoyants.
    -

3. Intégration de Bootstrap

Bootstrap est une bibliothèque les plus connus parmi les composants UI qui permet de créer des interfaces responsives facilement. Tu peux l'intégrer dans Angular pour améliorer rapidement le style visuel :

- **Responsive Design** :
  - Utilise la grille de Bootstrap (`row`, `col`) pour rendre l'interface responsive sur différents types d'écrans (mobile, tablette, desktop).

- **Composants Bootstrap** : Utilise des composants prêts à l'emploi comme les boutons, les formulaires, les cartes, les modals, etc.
  - Exemples : `<button class="btn btn-primary">`, `<div class="card">`.

- **Styles personnalisés** : Si besoin, tu peux personnaliser le style en utilisant des classes CSS supplémentaires, ou en surchargeant les styles de Bootstrap via un fichier `styles.css` dans Angular.

4. Conformité aux Normes d'Accessibilité

Veille à ce que ton interface soit conforme aux normes d'accessibilité (WCAG) :

- **Navigation au clavier** : Assure-toi que tous les éléments interactifs sont accessibles avec la touche `Tab`.

- **Contrastes** : Utilise un outil pour vérifier les contrastes de couleurs.
- **Alt Text pour les Images** : Ajoute des descriptions appropriées pour les images via l'attribut `alt`.

Backup : Pour le backup création d'une API en python (Flask) pour faire la liaison entre le modèle d'apprentissage et le projet angular qui communiquera avec l'API python.

Backend : Le **backup** dans ce contexte fait référence à la création d'un service de sauvegarde qui permet à l'application front-end Angular de stocker, sécuriser, et récupérer des données, ainsi que de communiquer avec un modèle d'apprentissage automatique. Cela se fait via une **API REST** développée avec Flask, un micro-framework en Python. Cette API joue un rôle crucial en tant qu'interface entre l'application front-end et le back-end, où se déroulent la sauvegarde des données et d'éventuels traitements de machine learning.

5. Fonctionnement de l'API Flask pour le Backup

L'API Flask se charge de recevoir des données envoyées par l'interface Angular, telles que les informations à sauvegarder ou des requêtes d'analyse. Les principales tâches de cette API incluent :

- **Réception et validation des données** : Flask reçoit les requêtes de sauvegarde de l'interface utilisateur, qui peuvent contenir des fichiers, des messages, ou tout autre type de contenu pertinent. L'API valide ensuite ces données avant de les traiter.
- **Gestion des Sauvegardes** : L'API est responsable de la gestion des sauvegardes, notamment l'enregistrement des données

dans une base de données (par exemple, PostgreSQL, MongoDB) ou dans un service de stockage local ou distant. Cette étape peut inclure l'horodatage des sauvegardes, la gestion des versions et l'organisation des données pour faciliter leur récupération future.

- **Sécurisation des Sauvegardes** : La sécurité est une priorité dans tout processus de sauvegarde. L'API peut inclure des mécanismes d'authentification (comme les tokens JWT) pour garantir que seules les requêtes authentifiées peuvent déclencher une sauvegarde ou récupérer des données. Des protocoles de cryptage peuvent également être mis en place pour protéger les données stockées.
- **Sauvegardes incrémentales ou complètes** : Selon les besoins, l'API peut offrir plusieurs stratégies de sauvegarde. Les sauvegardes complètes copient toutes les données à chaque fois, tandis que les sauvegardes incrémentales ne stockent que les nouvelles informations ou les modifications depuis la dernière sauvegarde, optimisant ainsi le temps et l'espace de stockage.

6. Communication avec le Modèle d'Apprentissage Automatique

En plus de la gestion des sauvegardes, l'API Flask peut interagir avec un modèle d'apprentissage automatique, en particulier pour des analyses en temps réel. Voici comment l'API peut gérer cette interaction :

- **Traitement des Requêtes** : L'API permet au front-end d'envoyer des données, comme des messages ou des images à

analyser. Ces données sont ensuite transmises au modèle d'apprentissage automatique pour un traitement.

- **Retour de Résultats** : Après l'analyse, le modèle renvoie les résultats (par exemple, une prédiction, un score de fiabilité, ou une classification) via l'API Flask à l'application Angular. Ces résultats sont ensuite affichés à l'utilisateur sous forme d'alertes ou de rapports.

7. Aspects de Performance et de Scalabilité

Pour garantir la performance et la scalabilité, des techniques comme le **caching** (mise en cache) peuvent être mises en place pour éviter de répéter des traitements coûteux. De plus, l'API peut être déployée dans un environnement cloud afin de gérer de grandes quantités de données de manière efficace et évolutive.

8. Sécurisation et Gestion des Droits d'Accès

La sécurité joue un rôle essentiel dans la sauvegarde des données et dans la communication avec le modèle d'apprentissage. L'API doit implémenter des contrôles d'accès stricts, basés sur des jetons d'authentification (comme JWT), et éventuellement des certificats SSL pour assurer une communication sécurisée entre le front-end et le back-end.

Utilisation de l'API Google Cloud Vision

Nous avons choisi d'intégrer l'API **Google Cloud Vision** dans notre solution en raison de ses capacités avancées en **extraction de texte** (OCR - Optical Character Recognition) et en **analyse d'images**.

Cette API nous permet de récupérer avec précision le texte présent dans les captures d'écran issues des réseaux sociaux, qui est ensuite traité par les modèles de détection de phishing

9. Intégration et Tests des Modèles

Tests de Modèles NLP :

Lors de la phase d'expérimentation, plusieurs modèles de **Hugging Face** ont été évalués pour la détection de phishing Chaque modèle présente des caractéristiques spécifiques, et en raison du manque de données, nous avons opté pour des modèles pré-entraînés afin d'optimiser les performances.

Voici les modèles testés et leurs objectifs respectifs :

- ***ealvaradob/bert-finetuned-phishing***

Ce modèle est une version **finement ajustée de BERT-large-uncased**, entraînée sur un jeu de données spécifique au phishing.

Il est capable de détecter le phishing dans ses quatre formes les plus courantes : **URLs**, **Emails**, **SMS**, et **Sites web**.

Performances du modèle sur le jeu de validation :

- **Loss :** 0.1953
- **Accuracy :** 97.17%
- **Precision :** 96.58%
- **Recall :** 96.70%
- **False Positive Rate :** 2.49%

Description du Modèle :
BERT (Bidirectional Encoder Representations from Transformers) est un modèle basé sur les transformateurs, préentraîné sur un large corpus de données **en anglais,**

en utilisant une approche d'auto-supervision.

Cela signifie qu'il a été préentraîné uniquement sur des textes bruts, sans intervention humaine, ce qui lui permet de tirer parti d'une grande quantité de données publiques.

Le modèle a ensuite été ajusté sur un jeu de données de phishing pour en améliorer les capacités de détection.

Configuration du modèle :

- **24 couches**
- **1024 dimensions cachées**
- **16 têtes d'attention**
- **336 millions de paramètres**

Ce modèle vise à prévenir de manière efficace et précise les attaques de phishing contre les individus et les organisations.

Pour atteindre cet objectif, BERT a été entraîné sur un jeu de données diversifié et robuste comprenant des **URLs**, des **Messages SMS**, des **Emails** et des **Sites web**, ce qui permet au modèle d'étendre ses capacités de détection à divers contextes.

Hyperparamètres d'entraînement :

- **Learning rate :** 2e-05
- **Batch size pour l'entraînement :** 16
- **Batch size pour l'évaluation :** 16
- **Seed :** 42
- **Optimiseur :** Adam (betas=(0.9, 0.999), epsilon=1e-08)
- **Scheduler de taux d'apprentissage :** linéaire
- **Nombre d'époques :** 4

- ***dima806/phishing-email-detection***

Ce modèle est spécifiquement conçu pour la détection des **emails de phishing**. Il utilise également la base de **BERT** pour analyser et classifier des emails suspects en phishing ou non-phishing.

Objectif du modèle :
Ce modèle est entraîné pour reconnaître les modèles linguistiques et les indices dans les emails utilisés pour tromper les utilisateurs,

comme des invitations à cliquer sur des liens, des offres frauduleuses ou des demandes d'informations personnelles. Il est capable d'analyser le contenu textuel et de fournir une évaluation précise des emails suspects.

Utilisation dans le projet :

Nous avons utilisé ce modèle pour analyser spécifiquement les captures d'écran de messages email afin d'identifier les menaces de phishing. Bien que ce modèle soit principalement ajusté pour les emails, il a également montré de bonnes performances lorsqu'il est appliqué à d'autres types de messages textuels sur les réseaux sociaux

Nous avons également testé les modèles **bert-base-uncased**, **roberta-base**, et **unitary/unbiased-toxic-roberta**, mais sans succès.

Ces modèles ne sont pas spécifiquement entraînés sur des données liées à notre sujet, car ils sont davantage orientés vers l'analyse de sentiments ou la détection de contenus toxiques, et non sur la détection de phishing

4. Analyse SWOT

Forces	Faiblesses
- Utilisation de technologies IA	- Peu de données pour entraîner un modèle NLP propre

- Intégration d’outils puissants (Google Vision, BERT) - Expérience utilisateur fluide avec Angular et Bootstrap	- Limitation de temps pour le développement complet
Opportunités	**Menaces**
- Croissance du marché de la cybersécurité - Besoin de solutions spécifiques pour les réseaux sociaux	- Forte concurrence des solutions existantes - Évolution rapide des techniques de fraude et phishing

5. Méthodologie et Outils Utilisés

Technologies :

- **Frontend :** Angular, Bootstrap
- **Backend :** API Google Vision pour la reconnaissance d’image
- **Modèle NLP :** BERT fine-tuned pour le phishing
- **Organisation** : Trello , Document word partagé, Github
- **Marketing** : Canva

6. Processus d’analyse :

1. **Chargement de l'image** : L'utilisateur télécharge une capture d'écran

2. **Extraction du texte** : Google Vision API extrait le texte de l'image.
3. **Analyse NLP** : Le texte extrait est analysé par le modèle BERT finement ajusté pour identifier les signaux malveillants (escroquerie).
4. **Résultat d'analyse** : Retour immédiat à l'utilisateur avec un diagnostic visuel (contenu malveillant détecté ou non).

7. Livrables

- **Site web fonctionnel** avec téléversement d'images, analyse automatisée via Google Vision et BERT.
- **Vidéo de démonstration** : Vidéo de 7 minutes présentant à la fois le côté marketing et les aspects techniques du projet.

8. Planning

Tâches	Délai
Conception de l'interface utilisateur (Angular, Bootstrap)	2 jours
Intégration de Google Vision API	1 jour
Intégration du modèle NLP BERT	1 jour
Tests et optimisation (accessibilité, faux positifs)	1 jour
Finalisation et préparation de la soutenance	1 jour