Université Paris Nanterre

Licence CMI seconde année

Rapport de projet informatique

Application de chat en réseau avec système de cryptage

Projet réalisé du 27 octobre 2023 au 15 janvier 2024

Membres du groupe

Assal Nermin
Hafeez Feza
Zaman Tasnime

Remerciements

Merci à Mr. Delbot d'avoir été notre professeur ce semestre et nous avoir permis de progresser en informatique.
Nous sommes reconnaissantes envers notre superviseur pour ses conseils et son soutien tout au long du processus.

Table des matières

1 Introduction

2 Environnement de travail

Nous avons travaillé sur le projet lors des cours de TD, de chez nous et ainsi qu'à la bibliothèque. Nous travaillions sur nos parties individuellement et lorsqu'en groupe, nous partagions nos difficultés et avancés afin de voir où nous nous étions. Nous avons travaillé sur nos ordinateurs personnels, soit Asus et Mac.

3 Description du projet et objectifs

L'objectif principal de ce projet est de fournir un moyen sécurisé et confidentiel pour les utilisateurs de communiquer à travers un réseau, que ce soit localement (sur un réseau local) ou à distance (via Internet). En intégrant un système de cryptage, l'application vise à garantir la confidentialité des échanges entre les utilisateurs, en empêchant toute interception non autorisée des messages.

Les principaux objectifs du projet incluent :

Confidentialité des Communications : protéger les messages échangés entre utilisateurs contre toute interception non autorisée en utilisant des techniques de cryptage robustes.

Sécurité : fournir un environnement sécurisé pour la communication, minimisant les risques de violation de la vie privée ou d'usurpation d'identité.

Facilité d'utilisation : offrir une interface conviviale pour permettre aux utilisateurs de profiter des fonctionnalités de chat sans compromettre la sécurité.

Compatibilité Réseau : permettre la communication à travers différents types de réseaux, que ce soit en local ou à distance.

Gestion des Clés : mettre en place un mécanisme sécurisé pour la gestion des clés de cryptage, garantissant qu'elles ne tombent pas entre de mauvaises mains.

En résumé, l'objectif global est de créer une application de chat en réseau qui assure une communication sécurisée et confidentielle, tout en restant accessible et conviviale pour les utilisateurs.

4 Bibliothèques, Outils et technologies

La réalisation d'une application de chat en réseau avec un système de cryptage implique l'utilisation de plusieurs langages de programmation, selon les différentes couches de l'application.

Nous avons choisi Python est choisi comme langage principal pour le projet d'application de chat en réseau avec système de cryptage en raison de sa lisibilité, de sa

rapidité de développement, de ses frameworks web comme Flask, de ses bibliothèques de cryptographie bien établies, de sa compatibilité réseau, de sa large communauté de développeurs, de sa portabilité et de son intégration facile avec d'autres technologies. Ces caractéristiques rendent Python adapté pour créer une application efficace, sécurisée et maintenable, répondant aux besoins spécifiques du projet.

Nous avons choisi Flask pour ce projet de chat sécurisé en raison de sa légèreté, de sa simplicité, et de sa facilité d'apprentissage. Sa modularité permet aux développeurs de construire des applications sur mesure en choisissant les composants nécessaires. La documentation complète, l'adaptabilité et l'intégration aisée dans l'écosystème Python sont des avantages supplémentaires. En fin de compte, le choix de Flask est motivé par sa flexibilité, sa facilité d'utilisation, et sa capacité à répondre aux besoins spécifiques du projet sans imposer une structure rigide.

Nous avons choisi JavaScript car c'est est un choix pertinent pour notre projet d'application de chat en réseau avec un système de cryptage en raison de sa capacité à faciliter la communication en temps réel, à rendre l'interface utilisateur interactive, à offrir une réactivité accrue, et à intégrer facilement des technologies web modernes. L'inclusion de JavaScript améliorerait l'expérience utilisateur et renforcerait les fonctionnalités de sécurité de votre application.

Nous avons choisi HTML pour sa capacité à fournir une structure de base pour organiser le contenu de manière universellement compatible avec les navigateurs. HTML s'intègre efficacement avec JavaScript et CSS, offrant ainsi une flexibilité accrue pour créer des applications web interactives. Son interopérabilité, ses fonctionnalités d'accessibilité, son soutien robuste dans l'écosystème de développement, son impact positif sur le référencement, et sa facilité de maintenance font de HTML un choix standard et pratique pour le développement d'applications web.

Nous avons choisi la bibliothèque cryptography.fernet pour crypter les messages dans votre projet en raison de sa simplicité d'utilisation, de sa gestion automatique des clés, de sa réputation en matière de sécurité, de ses bonnes performances, et de son format de token standard qui favorise l'interopérabilité. Le choix de cette bibliothèque est orienté vers une solution robuste, efficace et facile à mettre en œuvre dans le contexte d'une application de chat sécurisée en temps réel.

Jsonify est une fonction de Flask qui convertit un objet Python en JSON. Cela est souvent utilisé pour renvoyer des réponses JSON depuis une route Flask.

Flash est utilisé pour envoyer des messages flash qui peuvent être affiché à l'utilisateur. Cela est souvent utilisé pour afficher des messages d'erreur ou de succès après une action.

G (contexte global) est une structure de données qui peut être utilisée pour stocker des données qui doivent être accessibles tout au long de la durée de vie de la requête. Elle est utile pour stocker des données temporaires.

Gestion de la base de données (SQLite) : des fonctions telles que get db(), init db(), et close connection() sont utilisées pour établir, initialiser et fermer la connexion à la

base de données SQLite.

Nous avons choisi CSS dans notre projet d'application de chat en réseau avec système de cryptage est essentiel pour définir la présentation visuelle, assurer la séparation des responsabilités entre le contenu et le style, garantir l'adaptabilité à différents écrans, maintenir la consistance visuelle, intégrer des animations et des effets visuels, optimiser la performance, améliorer l'accessibilité, et favoriser l'évolutivité de l'application. En résumé, CSS joue un rôle crucial dans la création d'une interface utilisateur attrayante, fonctionnelle et adaptée à divers dispositifs.

5 Travail réalisé

5.1 Fonctionnalités du projet

Pour la réalisation de ce projet, les différentes tâches à réaliser étaient les suivantes : - Serveur et client - Interface visuelle de l'application - Création des comptes utilisateurs - Visualisation des utilisateurs en ligne - Chiffrement et déchiffrement des messages - Historique des messages - GUI réactive - Création d'une base de données

5.2 Répartition du travail

Nermin a fournit le code pour le serveur et le client. Feza a fournit le code pour le chiffrement et déchiffrement des messages. Tasnime s'est chargée de rassembler les codes ainsi que de créer l'interface et toutes les fonctionnalités qui lui sont associées. Enfin, le rapport en latex a été fait par Feza.

5.3 Évaluation de Feza

En ce qui me concerne, j'ai trouvé très intéressant de travailler sur ce projet, car c'était un peu une suite logique de mon projet de l'année dernière. En effet, j'avais travaillé sur "l'envoi et la réception de données via un réseau local en architecture en C. Dans les perspectives d'amélioration le chiffrage des données avait été mentionné, c'est pour cela que cette année, j'ai voulu travaillé dessus. J'ai aimé me renseigner sur la sécurité des données. Obtenir le code n'a pas été très compliqué une fois que j'ai compris ce qui était attendu. Toutefois, l'intégrer dans le code principal a été une tâche beaucoup plus compliquée.
Selon moi, le travail fournit par mes camarades était satisfaisant. La plus grosse partie a clairement était faite par Tasnime, mais j'ai bien aimé le fait qu'elle nous fasse part de ses avancées et demande de l'aide lorsqu'elle se sentait débordée.

Voici le code que j'ai fourni :

```python
def ecouter_messages(socket_client):
    while True:
        try:
            # Recevoir et déchiffrer le message
            encrypted_message = socket_client.recv(128)
            decrypted_message = cipher_suite.decrypt(encrypted_message).decode("utf-8")

            print(decrypted_message)
        except Exception as e:
            print(f"Erreur lors de la réception du message: {e}")
            break

def envoyer_messages(socket_client, username):
    while True:
        try:
            # Saisir le message à envoyer
            message = input("Votre message: ")

            # Ajouter le nom d'utilisateur au message
            message = f"{username}: {message}"

            # Chiffrer et envoyer le message
            encrypted_message = cipher_suite.encrypt(message.encode("utf-8"))
            socket_client.send(encrypted_message)
        except Exception as e:
            print(f"Erreur lors de l'envoi du message: {e}")
            break
```

5.4 Évaluation de Nermin

En partageant ma perspective sur ma contribution au projet, je souligne l'importance de ma responsabilité dans l'envoi et la réception des messages notamment en assumant la responsabilité d'assurer l'envoi et la réception des messages, j'ai trouvé cette expérience particulièrement enrichissante, surtout en considérant que l'année précédente, j'avais déjà pris en charge la mise en place de la partie serveur et client pour faciliter la communication. Retravailler sur ces aspects m'a permis de consolider mes compétences et de constater les progrès réalisés depuis. L'intégration du système de cryptage par Feza a ajouté une dimension essentielle à notre application, renforçant la sécurité des échanges. Malgré les défis rencontrés lors de l'assemblage des codes, notre équipe a réussi à surmonter ces obstacles, témoignant de notre collaboration résiliente. Il est important de souligner que Tasnime a pris en charge la partie la plus conséquente du projet, revenant vers nous pour solliciter de l'aide et collaborer à la prise de décisions. Cette expérience a non seulement consolidé mes compétences techniques, mais a également renforcé ma compréhension de l'importance du travail d'équipe dans le développement logiciel.

Voici les codes que j'ai fourni :

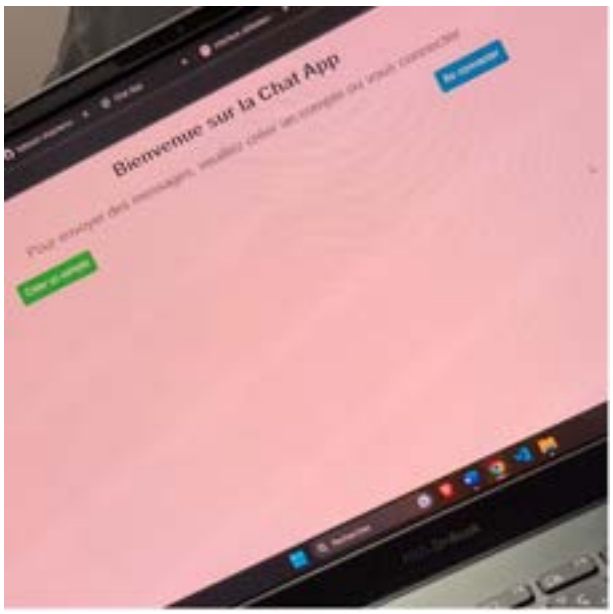
```
import select

serveur = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host, port = "127.0.0.1", 4000
serveur.bind((host, port))
serveur.listen(4)
client_connecte = True
socket_objs = {serveur}

print("Bienvenue dans le chat!")

while client_connecte:
    liste_lu, liste_acce_ecrit, exception = select.select(socket_objs, [], socket_objs)

    for socket_obj in liste_lu:
        if socket_obj == serveur:
            client, adresse = serveur.accept()
            socket_objs.add(client)
            print(f"Nouvelle connexion de {adresse}")
        else:
            donnees_recues = socket_obj.recv(128).decode("utf-8")
            if donnees_recues:
                print(donnees_recues)
            else:
                print("Un participant s'est déconnecté")
                socket_objs.remove(socket_obj)

    print(f"{len(socket_objs) - 1} participants restants")
```

```
import socket
import threading
def ecouter_messages(socket_client):
    while True:
        try:
            message = socket_client.recv(128).decode("utf-8")
            print(message)
        except:
            print("Erreur lors de la réception du message.")
            break

def envoyer_messages(socket_client):
    while True:
        message = input()
        socket_client.send(message.encode("utf-8"))
# Création d'un socket client
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host, port = "127.0.0.1", 4000
# Connexion au serveur
client.connect((host, port))
print("Connecté au serveur!")

# Lancer des threads pour écouter et envoyer des messages
thread_ecoute = threading.Thread(target=ecouter_messages, args=(client,))
thread_envoi = threading.Thread(target=envoyer_messages, args=(client,))

# Démarrer les threads
thread_ecoute.start()
thread_envoi.start()
# Attendre que les threads se terminent
thread_ecoute.join()
thread_envoi.join()
# Fermer la connexion
client.close()
```

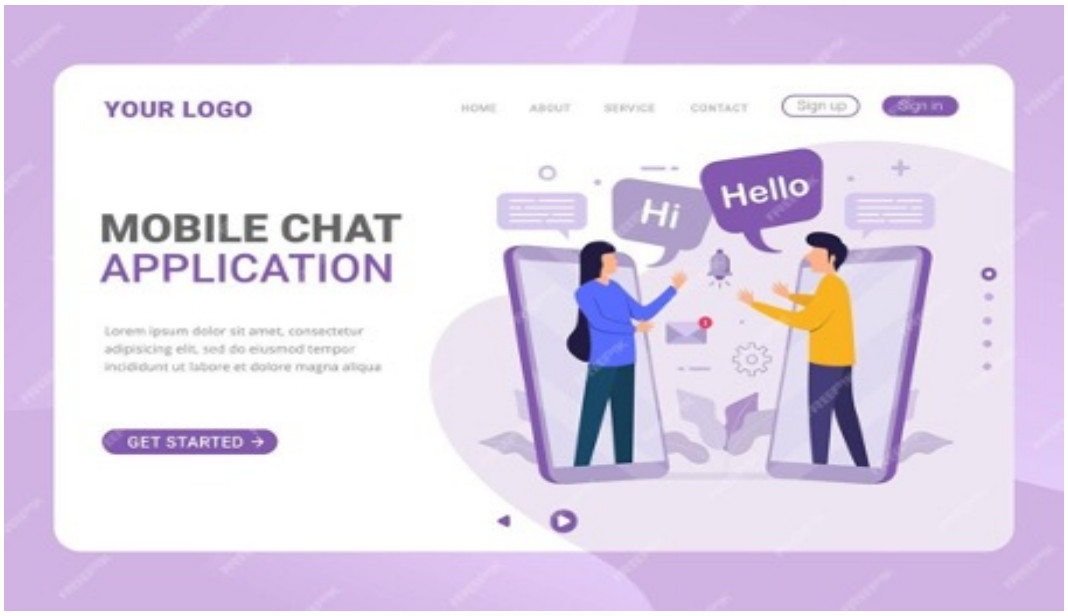
5.5 Évaluation de Tasnime

Pour créer l'interface utilisateur j'ai utilisé flask qui est un framework web que j'ai modifié petit à petit, elle permet la gestion des sessions, de mettre ma base de donne SQLite, pour stocker les utilisateurs et les messages et contient les différentes routes qui permet d'afficher les différentes pages. En résumé, cette application web de chat utilise Flask pour la gestion des routes et Flask-SocketIO pour permettre une communication en temps réel entre les utilisateurs via des sockets WebSocket. Elle intègre également la gestion des sessions, la base de données SQLite pour stocker les utilisateurs et les messages, ainsi que le chiffrement des messages.

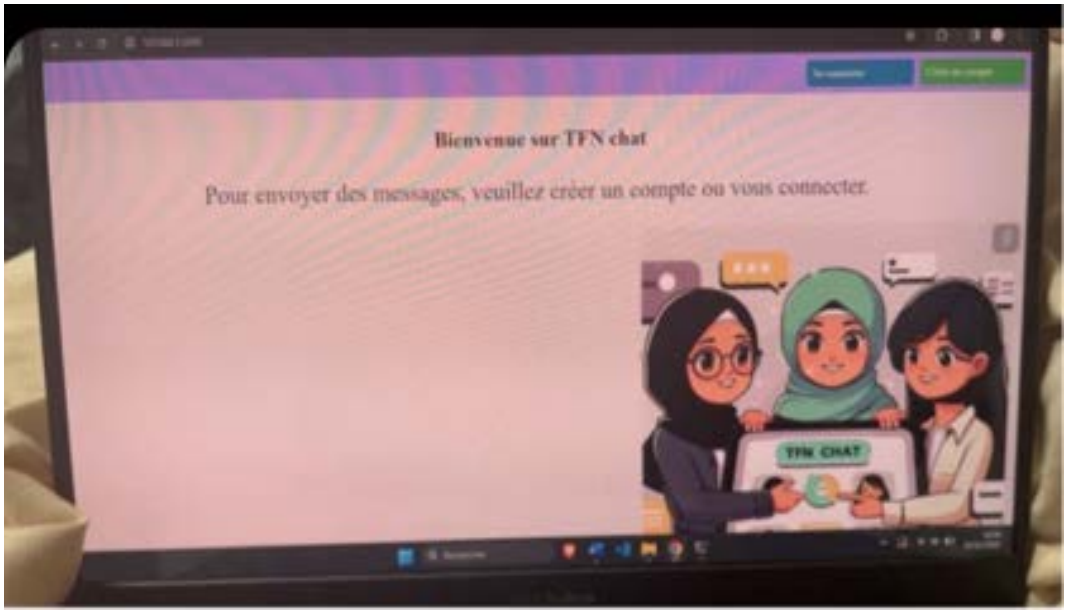
Pour ma part, dès le début, j'ai rencontré des difficultés pour exécuter le script Flask, on m'affichait plusieurs messages d'erreur comme "can't open file" qui indique que le terminal ne peut pas ouvrir le fichier spécifié ou "No such file or directory". Après plusieurs essaies, j'ai réussi à exécuter mon fichier et obtenue une page blanche avec mon texte d'essai.



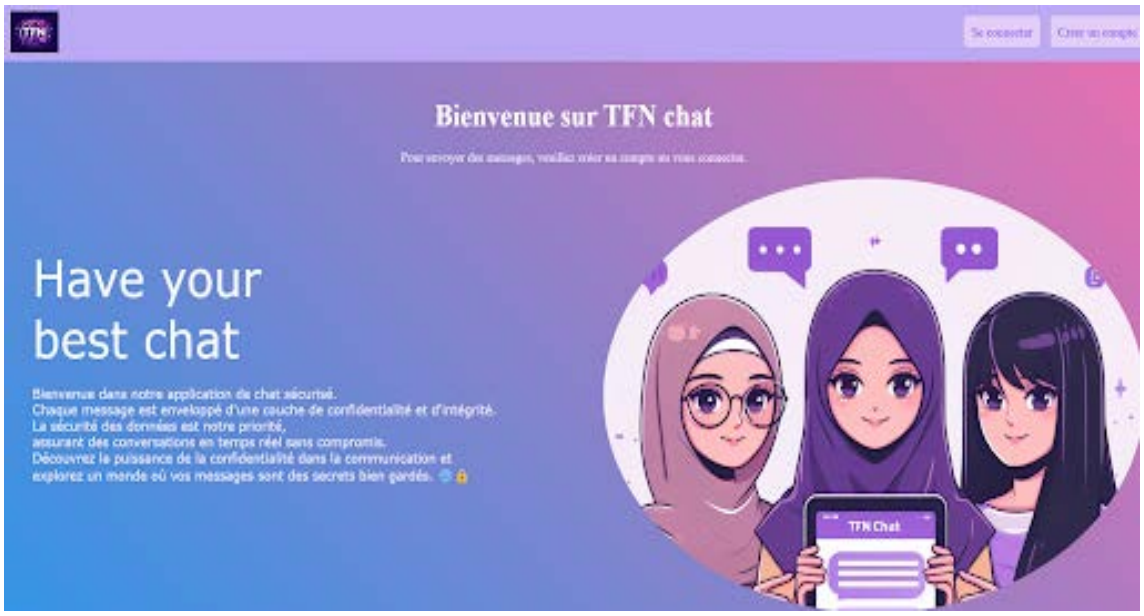
J'ai commencé par développer la page d'accueil. Pour ce faire, je me suis inspirée de cette image :



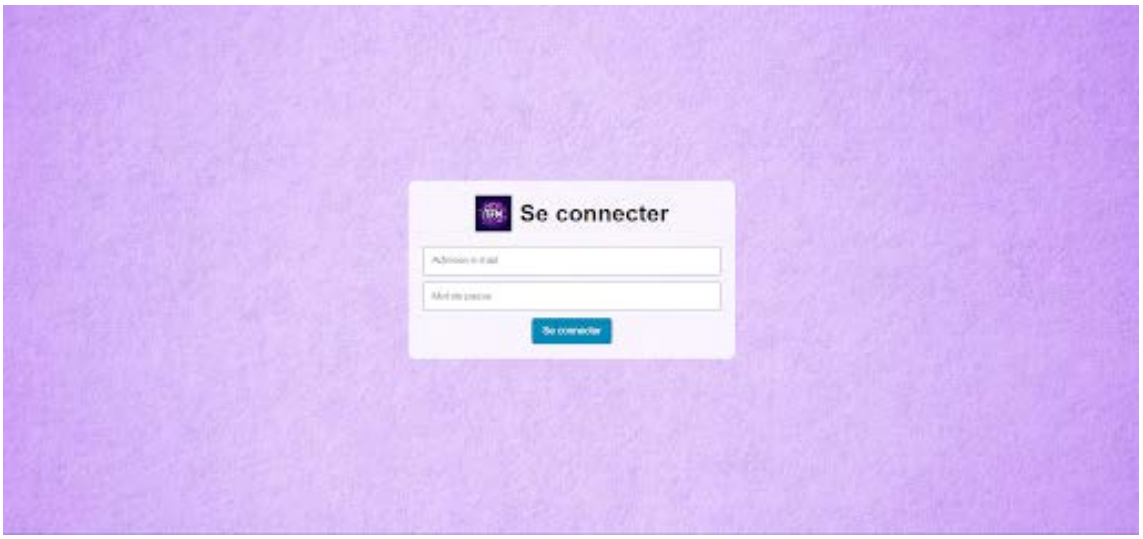
Au début, j'avais réalisé cette page grâce à des vidéos YouTube, des sites internet qui expliquaient le fonctionnement de HTML (changer la position d'un texte, changer la couleur, ajouter un bouton) et ChatGpt m'a aidé pour trouver mes erreurs et les corriger, lorsque j'avais un souci :



Pour au final, obtenir ce résultat :



J'ai utilisé CSS pour avoir des effets en fond, et une IA pour créer le logo TFN situé en haut à gauche ainsi que l'image qui représente les 3 membres du groupe.
Après avoir réalisé la page d'accueil qui m'a pris énormément de temps, car j'étais toujours insatisfaite du résultat, pour des petites futilités tel que le choix des couleurs ou l'emplacement du texte. J'ai ensuite réalisé les pages de connexion et création de compte, je pense que la modification du visuel a pris plus de temps que l'écriture du code en lui-même. Cela a été une erreur de ma part, de m'être autant focalisé sur des détails, ce qui m'a fait énormément perdre de temps. Je pense que si j'avais réalisé une page beaucoup plus simple, j'aurais pu peut-être améliorer ma GUI pour qu'elle soit plus propre et interactive. Pour la page création de compte et connexion, j'ai créé une base de données SQLite pour sauvegarder les informations des utilisateurs. Lorsqu'un utilisateur essaye de se connecter, mais n'a pas de compte, un message flash va apparaître pour lui demander de se connecter.
Voici les pages d'accueil et de création de compte :



Une fois connecté, on arrive sur la page du chat home où on voit les utilisateurs en ligne. Pour voir ces utilisateurs en ligne, je me suis basée sur la mise à jour manuelle du statut dans la base de données. Mais je pense qu'une approche plus robuste en utilisation d'un socket spéciale pour mettre à jour les utilisateurs connectés en temps réel aurait été mieux, j'ai essayé, mais en vain. Malgré les efforts déployés, l'implémentation de l'historique de chat sécurisé n'a pas été réalisée. Actuellement, les messages ne sont pas sauvegardés de manière sécurisée pour une consultation ultérieure.

6 Difficultés rencontrées

Le plus difficile a été l'assemblage de codes. L'affichage des utilisateurs en ligne n'a pas été chose aisée. De plus, deux choses n'ont pas pu être réalisées. En effet, après de maints essais, la sauvegarde des chats n'a pas été réussie. En outre, un problème inattendu est apparu, les noms d'utilisateurs ne sont pas visibles, le SID est affiché à la place.
Enfin, le dernier problème rencontré et qui n'a pas pu être résolu est la mise en ligne du site. En effet, nous avions réussi à héberger le site, mais uniquement la page "créer un compte" s'affiche, le reste devenant une page blanche. On pense que c'est dû à un problème de base de données, mais rien n'est sûr et ChatGPT n'a pas su nous aiguiller. Nous avons même essayé de changer plusieurs fois d'hébergeur, mais en vain, le même problème persistait. Le site fonctionne sans problème localement.

7 Bilan

7.1 Conclusion

En conclusion de notre projet sur l'application de chat en réseau avec système de cryptage, nous sommes ravis d'annoncer le succès de notre mission. Notre objectif principal, garantir la confidentialité des échanges, a été atteint grâce à l'implémentation réussie de techniques de cryptage.

L'application offre une expérience utilisateur fluide avec des fonctionnalités telles que la messagerie instantanée, la gestion des utilisateurs et la création de salons de discussion. Au-delà de la sécurité, ce projet a renforcé nos compétences en programmation réseau, en sécurité informatique et en conception logicielle, tout en favorisant une collaboration efficace au sein de notre équipe.

7.2 Perspectives

Notre application permet la communication entre plusieurs utilisateurs comme demandé, il serait ainsi pertinent de pouvoir améliorer l'application afin de créer des canaux privés et des groupes entre deux ou plusieurs personnes.
Notre application est "sécurisée" mais elle pourrait l'être encore plus, ainsi, c'est un point qui se doit d'être amélioré. On devrait aussi permettre à l'utilisateur de modifier les coordonnées de son compte telles que son pseudo ou mot de passe.
On peut aussi améliorer la sécurité des mots de passe en utilisant une fonction de hachage de sécurité.

8 Webographie

Références

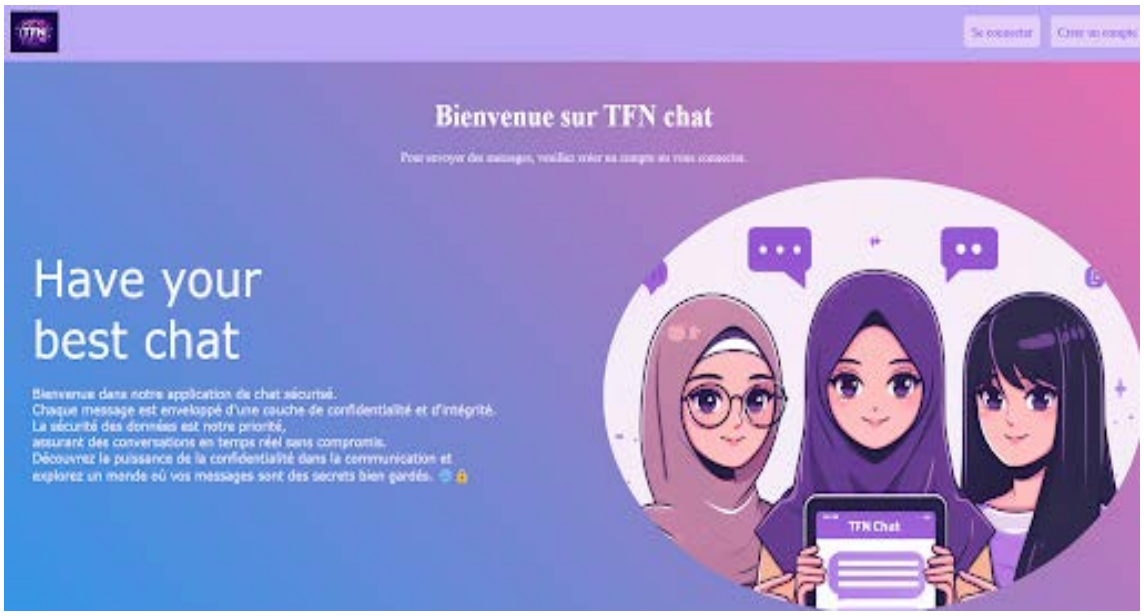
[vidéo] https://www.youtube.com/watch?v=bM1kg1Wdz3A&list=LL&index=10&t=281s

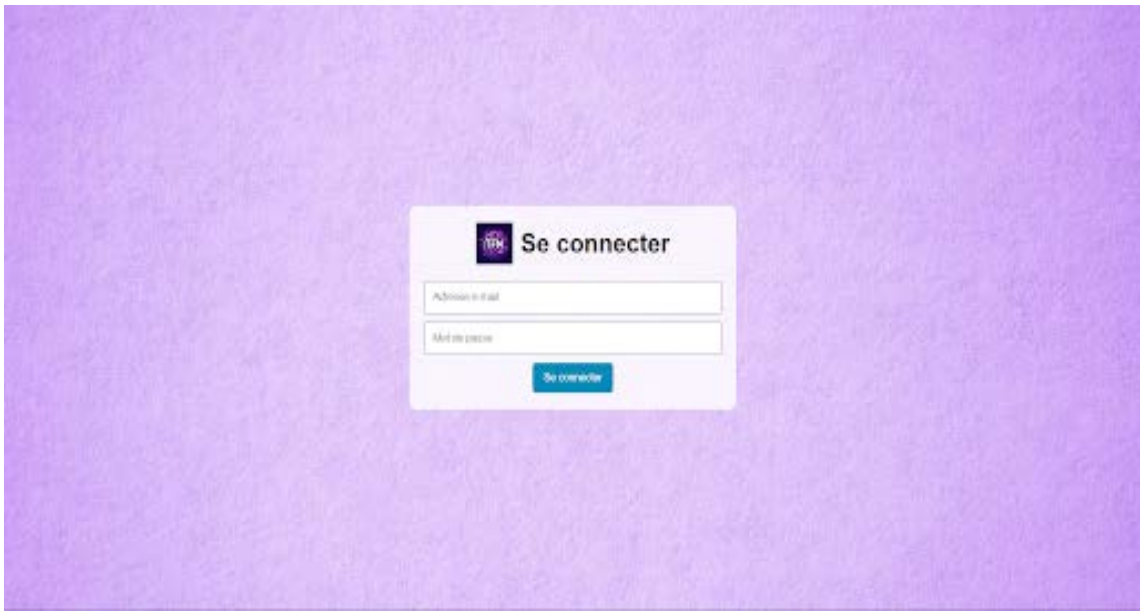
[ChatGPT] https://chat.openai.com/

[couleur] https://html-color-codes.info/Codes-couleur-HTML/

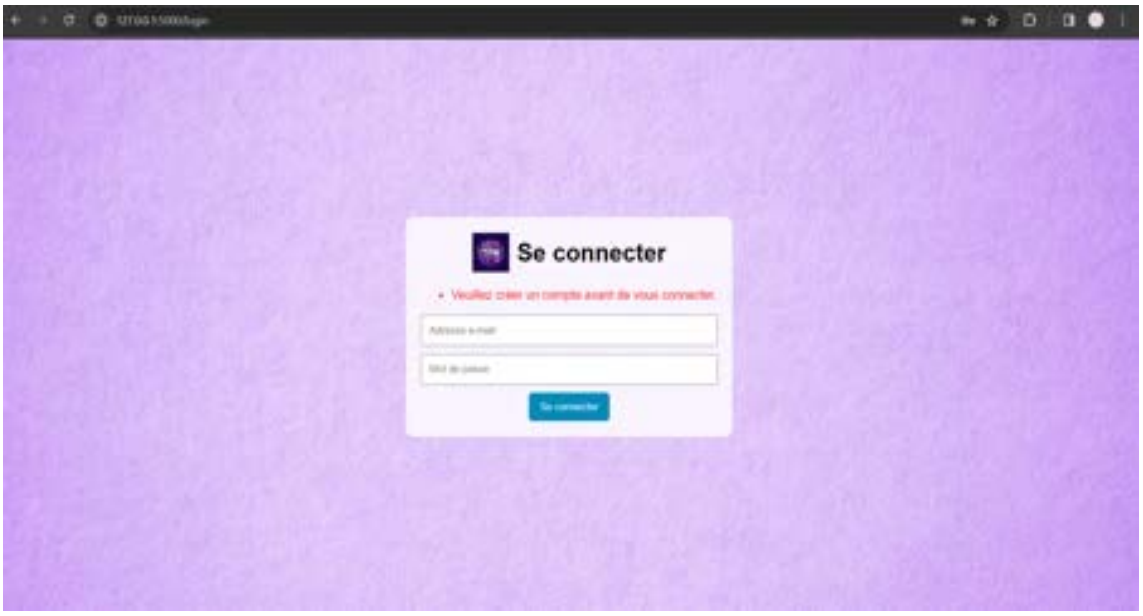
9 Annexes

Annexe A : Exemple d'exécution du projet

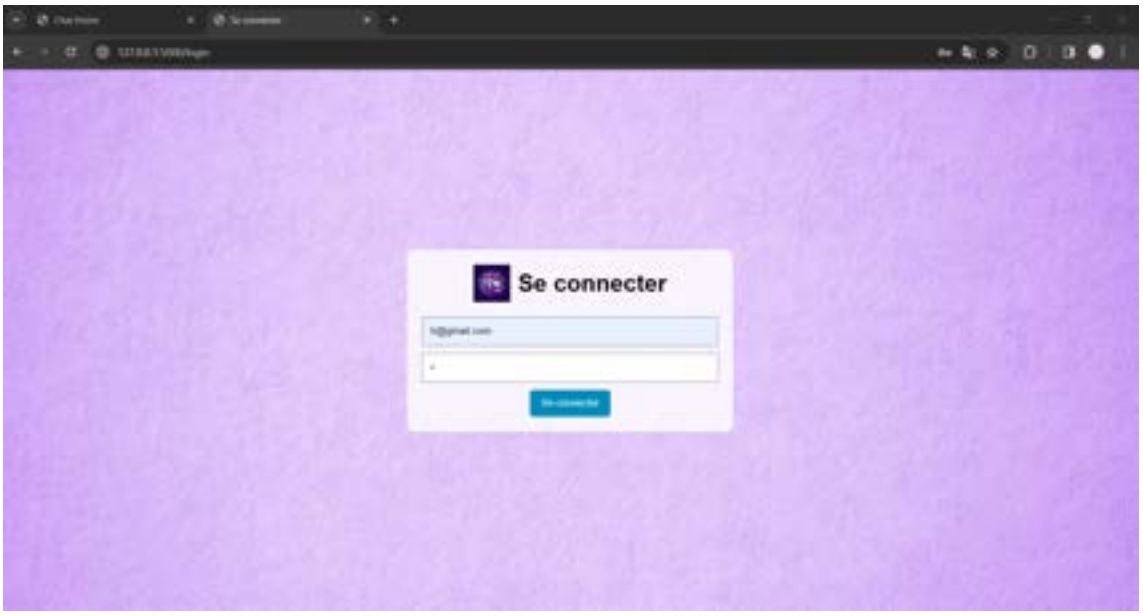


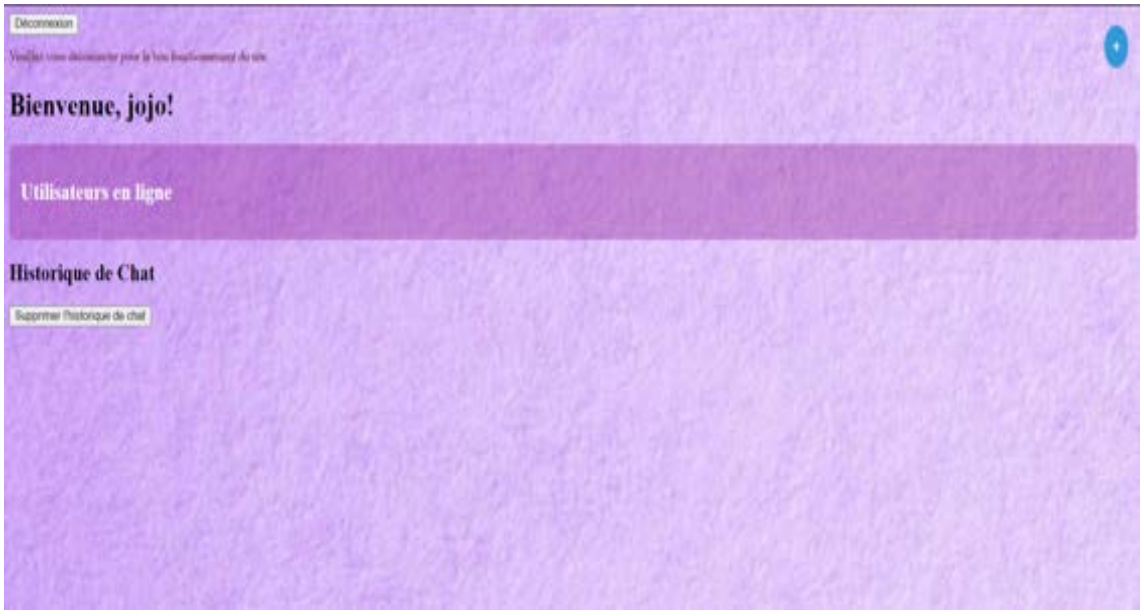


Lorsqu'un utilisateur veut se connecter, mais n'a pas de compte, le message suivant apparaît :



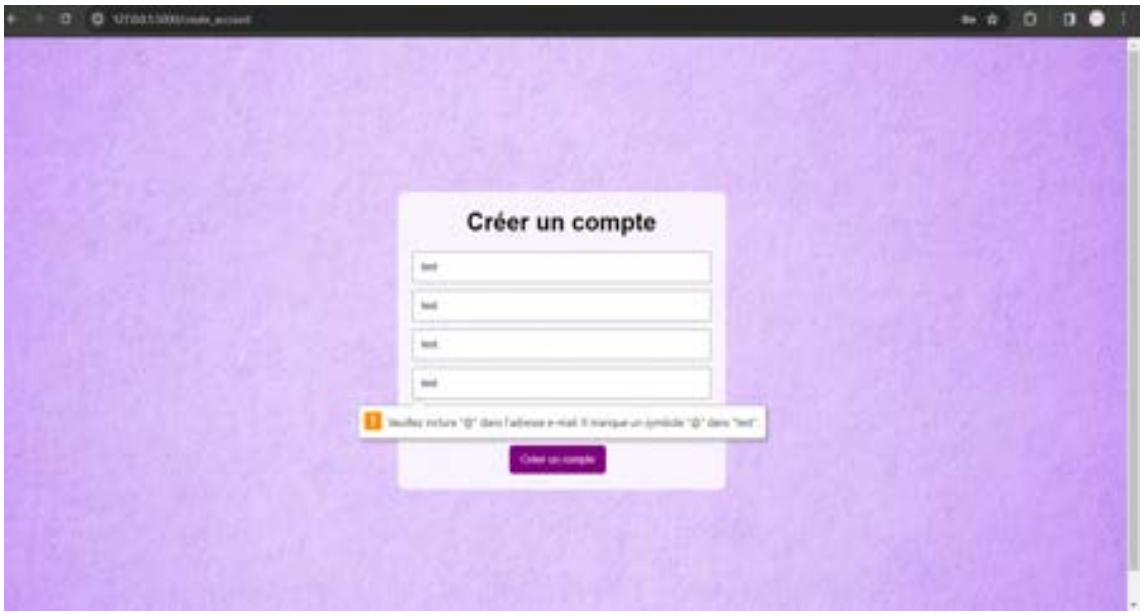
Lorsqu'un utilisateur se connecte, un message de bienvenue suivi de son nom d'utilisateur est affiché :



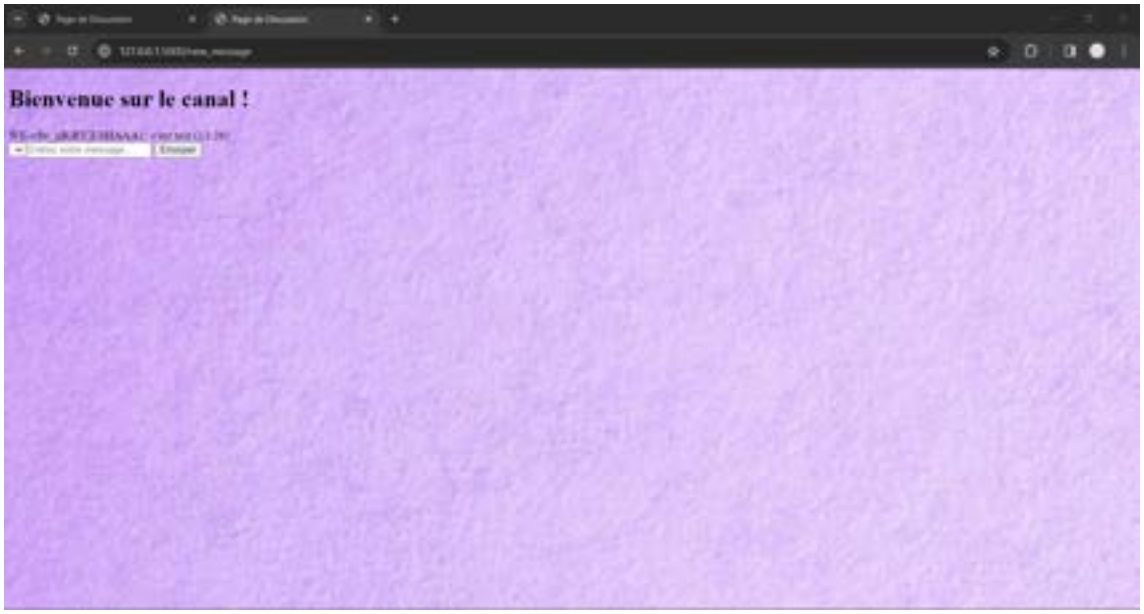


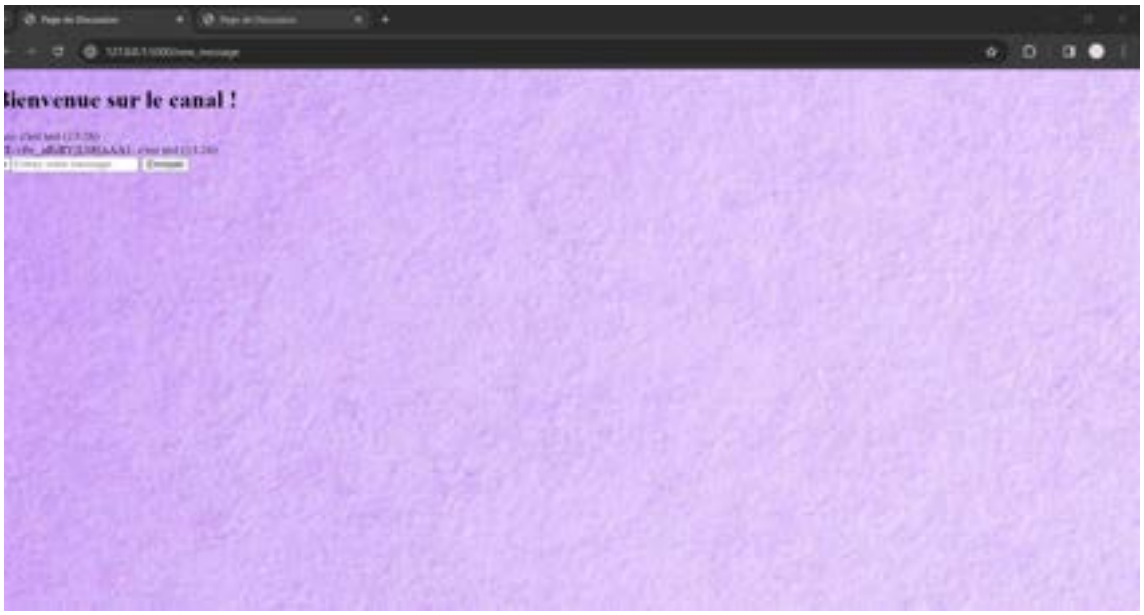
Lorsque 2 utilisateurs sont connectés simultanément, ils peuvent se voir en ligne.

Il suffit de rafraîchir la page afin de voir les utilisateurs connectés en temps réel. Le petit "+" ,en haut à droite de l'écran, permet de créer de nouvelles conversations. Lors de la création du compte, certains caractères sont obligatoires. Par exemple, le '@' dans le mail.



Voici un exemple d'envoie de message entre les 2 utilisateurs connectés :





Malheureusement, le SID s'affiche au lieu du nom d'utilisateur.

Voici le lien du site : https ://tfn-chat-app.onrender.com/