

Mini-project-topics-in-network-security

Yosef Shawah, 322727116. Ofir Zcharya, 315112300.

Pages 2-3 : Explanation and the key concepts about the project

Pages 4-5: How to run the application and the login

Pages 6-14: Explanation about the website, all the features

Pages 15-16: The pusher and the db

Pages 17-18: Strengths and Weaknesses and the Acknowledgements

Explanation

The goal: Make a secure chat app by using a few methods of security such as encryption, authentication, and middleware file between client and server.

Introduction:

The first layer of security in our application is authentication, for which we have implemented Google Authentication. Through the Google Cloud Console, we manually specify which users are authorized to access the application. The second layer of security is a messaging system, where we employ the Advanced Encryption Standard (AES) algorithm to ensure the confidentiality of communications. We saved the data on DB. We will explain more about each one.

Design and implementation:

We used Next.js for backend and frontend, next.js is react framework that is recommended by react docs, for db we used upstash-redis, for css we chose tailwindcss.

Endpoints:

<http://localhost:3000/login> the login page (with your Google authentication)

<http://localhost:3000/dashboard/> after login you will be redirected here

<http://localhost:3000/dashboard/add> here you can send post req to add friends

<http://localhost:3000/dashboard/requests> here you can accept/deny friend req.

http://localhost:3000/dashboard/chat/* here you can talk with your friend.

About our encryption method: AES: (Advanced Encryption Standard) is a symmetric key encryption algorithm, meaning the same key is used for both encryption and decryption. It was established by the U.S. National Institute of Standards and Technology (NIST) in 2001 and is widely used for secure data transmission.

Key Features:

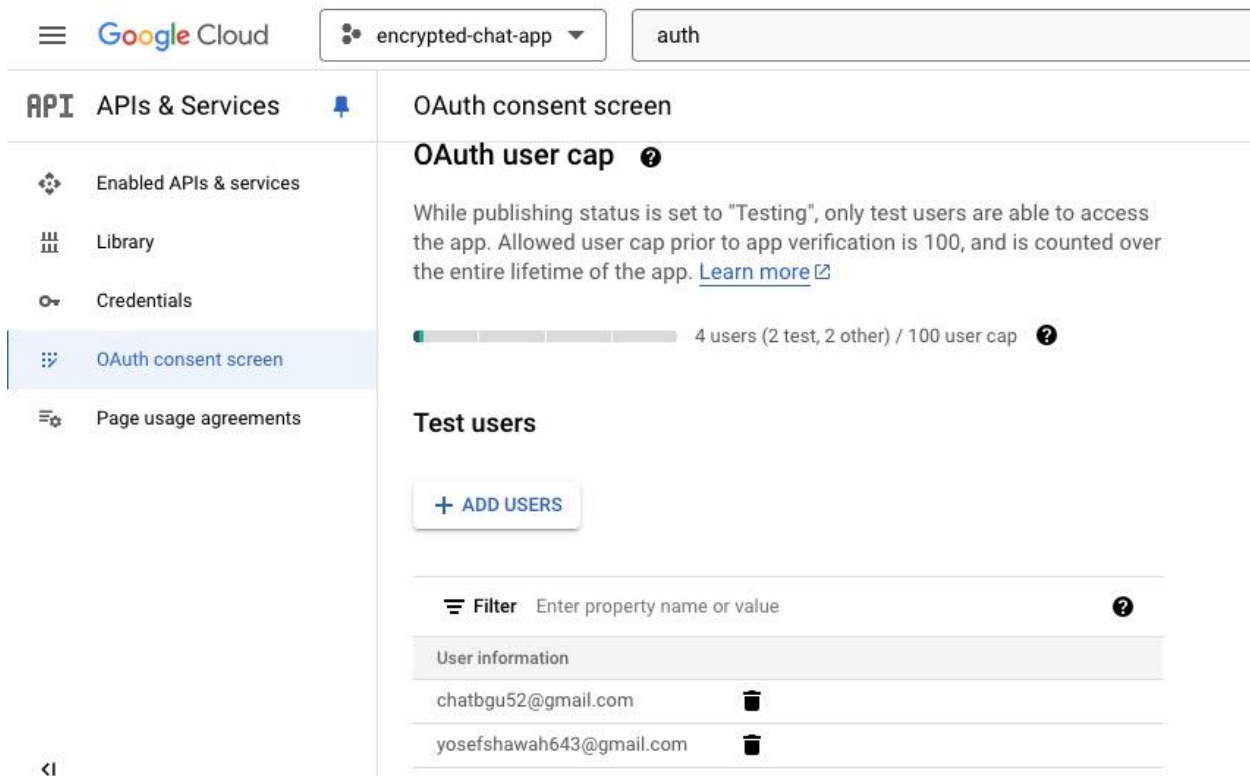
1. **Block Cipher:** AES operates on fixed-size blocks of data (128 bits or 16 bytes).
2. **Key Sizes:** It supports key sizes of 128, 192, or 256 bits, offering different levels of security.
3. **Rounds:** Depending on the key size, AES performs multiple transformation rounds (10, 12, or 14 rounds) to encrypt the data.
4. **Security:** AES is highly secure and resistant to most known attacks, making it a standard for protecting sensitive information worldwide.
5. **Efficiency:** It is fast and efficient in both hardware and software implementations.

AES is commonly used in applications like HTTPS, file encryption, and secure messaging.

Walkthrough:

After you clone the repo and install all the necessary dependences and start the app with ‘**yarn dev**’, you can find the app at <http://localhost:3000/login>, to enter you have to use Gmail account after that you can start to add people that are already registered and send them messages note that both of the parties has to have the same encryption & decryption key so they can communicate with each other here how it goes with pictures:

Notice: only 2 accounts have access to the app



The screenshot shows the Google Cloud console interface. At the top, there's a navigation bar with the Google Cloud logo, a dropdown menu showing 'encrypted-chat-app', and a search bar with 'auth'. On the left, a sidebar lists 'APIs & Services' with sub-items: 'Enabled APIs & services', 'Library', 'Credentials', 'OAuth consent screen' (which is highlighted), and 'Page usage agreements'. The main content area is titled 'OAuth consent screen' and 'OAuth user cap'. It explains that while publishing status is 'Testing', only test users can access the app, with a cap of 100 users. A progress bar shows '4 users (2 test, 2 other) / 100 user cap'. Below this, there's a section for 'Test users' with a '+ ADD USERS' button. A table lists the current test users:

Filter Enter property name or value	
User information	
chatbgu52@gmail.com	[trash icon]
yosefshawah643@gmail.com	[trash icon]

We write it in our project in a file (called *it .env*) there put the values of the authentications :
GOOGLE_CLIENT_ID
GOOGLE_SECRET_ID
(you can use this manual; <https://www.balbooa.com/help/gridbox-documentation/integrations/other/google-client-id>)

How Our Application looks like:

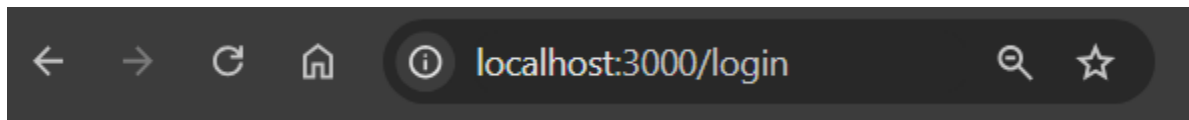
First, Operate the application:

- After you create the secret keys in a file.(*env.local for example*) (with all the authentication)
- After you install all the dependencies
- We run it in Visual Studio.

The output should be like this:

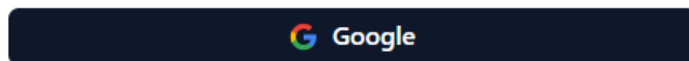
```
PS C:\Users\ofirz\nextjs-realtime-chat-master> yarn dev
yarn run v1.22.22
$ next dev
ready - started server on 0.0.0.0:3000, url: http://localhost:3000
info - Loaded env from C:\Users\ofirz\nextjs-realtime-chat-master\.env.local
warn - You have enabled experimental feature (appDir) in next.config.js.
info - Thank you for testing `appDir` please leave your feedback at https://nextjs.link/app-feedback
warn - Experimental features are not covered by semver, and may cause unexpected or broken application behavior. Use at your own risk.
```

After you run, open a browser and login: *http://localhost:3000/login*:

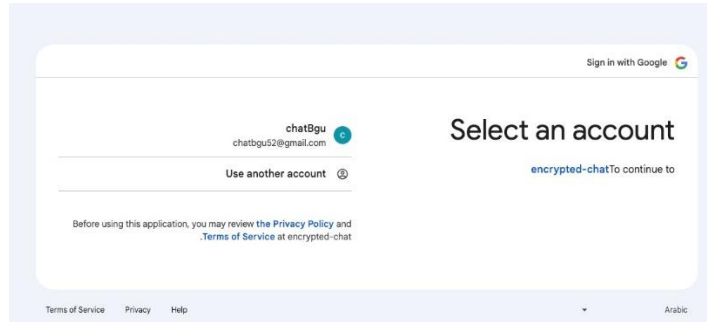


Chat Application

Sign in to your account

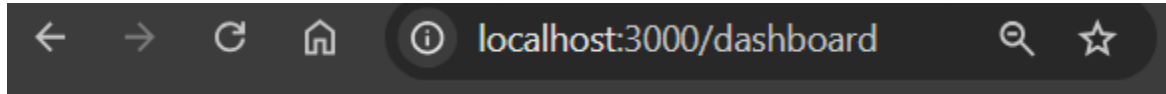


Run with Google:



After it you will go to the main page.

The main page:



Looks like this:

Without friends:

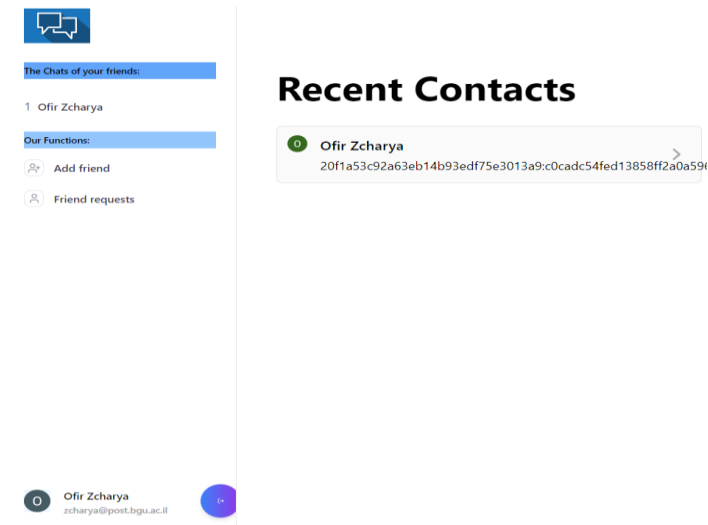


Our Functions:



Add friend

With friends that you added:



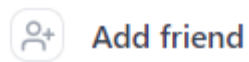
Explanation about the left Side:

Left Side:

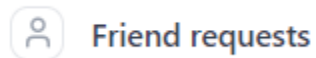
The Chats of your friends: all the friends that you added before in a list, counted from 1.

Our Functions:

- Add friend: If you want to add a new friend



- Friend Requests: If you receive a friend request and you want to add.



- LogOut: In a button in the bottom right corner.



The Chats of your friends:

Put on the name of “The Chats of your friends”

If you don’t have any friends, it will be like this:



Our Functions:



Add friend



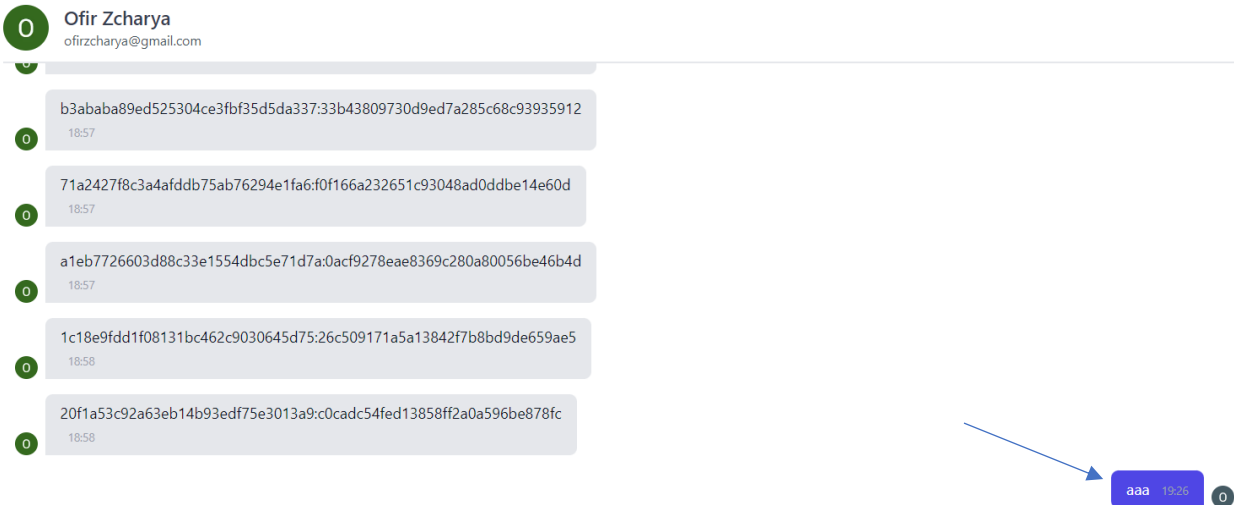
Friend requests

But, if you have at least one, it will be like this:

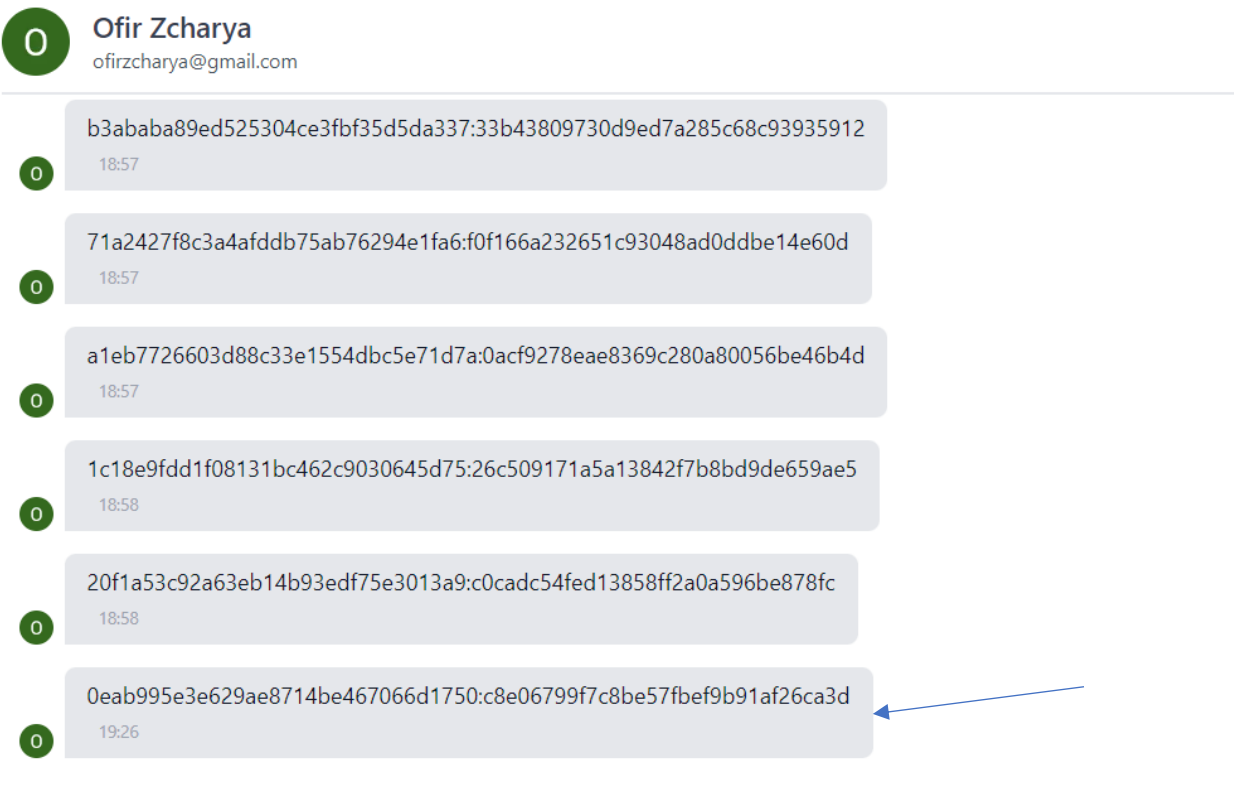
The Chats of your friends:

1 Ofir Zcharya

You will see all the messages are encrypted, unless you are in the chat now, and there you can see the messages without encryption.



When you go back to the “Our Functions” and return to the chat, you will see the “aaa” encrypted as well: (If you want to write a message, just typing in the box and press “ENTER”)



If you are not in the chat, you will see a pop-up message, like this: (encrypted)



Ofir Zcharya

3b3e59cd122b0f90c50ed8d4cb4aab55:5a1e794b17809f5cf531636217ad06bb

[Close](#)

The Chats of your friends:

1 Ofir Zcharya

Our Functions:



Add friend



Friend requests

Friend Requests

Send for a Friend

Add friend by your Email (he must to register first!)

Add

Add friend:

Friend Requests

Send for a Friend

Add friend by your Email (he must to register first!)

You should type a friend email (only if he only registered)

When you add a friend: The sender

Friend Requests



Send for a Friend

Add friend by your Email (he must to register first!)

Friend request sent!

When you add a friend: The receiver

Add a friend

 zcharya@post.bgu.ac.il  

Press V to accept, Press X to deny.

Friend Requests:

If you don't receive any requests: you will see like this:

Add a friend

There are no new friends requests

If you have requests: you will see like this:



Our Functions:



Add friend



Friend requests

1

Logout:

If you press logout, you will go to the login.

The DB:

URL: <https://console.upstash.com/login>

First, when you logged in, Go to Redis -> DataBases -> Create database (put name and Type, we used Regional type). After it go to the REST API section in the website, and you will see UPSTASH_REDIS_REST_URL and UPSTASH_REDIS_REST_TOKEN, put them also in the secret keys file (we called it `./env/local`)

Second, the db and the data browser: Notice here we save the users:

The screenshot shows the Upstash Data Browser interface. At the top, there's a header with the Upstash logo, a star icon, and a menu icon. Below the header, there are tabs for 'Free', 'AWS', 'Frankfurt, Germany', 'eu-central-1', and 'Global'. To the right, there are links for 'Docs' and 'SDK'. Below the header, there are tabs for 'Details', 'Usage', 'CLI', 'Data Browser' (which is active), 'Monitor', 'NEW', and 'Backups'. The main content area is divided into two panels. The left panel shows a list of keys with a search bar and a dropdown menu. The keys listed are: 'user:025dad16-d5a8-44...', 'user:account:by-user-id:...', 'user:account:google:111...', and 'user:email:chatbgu52@...'. The right panel shows a detailed view of the key 'user:account:google:111704620703960376929'. The value is a JSON object with the following fields: 'access_token', 'expires_at', 'id', and 'id_token'. The 'id' field is 'google:111704620703960376929'. The 'id_token' field is a long string. At the bottom of the right panel, there are tabs for 'TTL: Forever' and 'Memory: ~1683 bytes'.

encrypted-chat-db ☆ ...

Free AWS Frankfurt, Germany eu-central-1 Global Docs SDK

Details Usage CLI Data Browser Monitor NEW Backups

Search All Types +

- user:025dad16-d5a8-44...
- user:account:by-user-id:...
- user:account:google:111... >
- user:email:chatbgu52@...

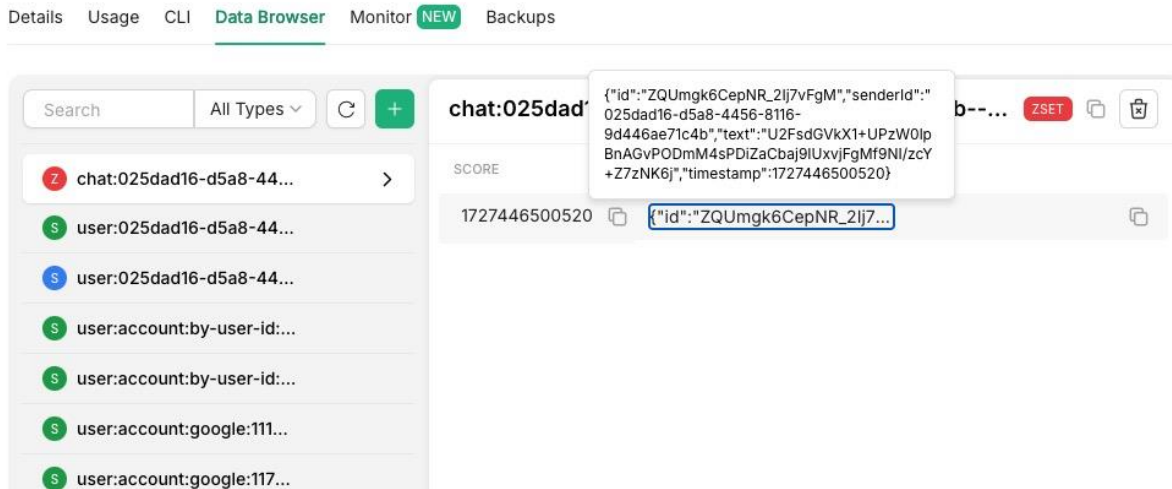
Total: 4 < >

user:account:google:111704620703960376929 STRING

```
1 {
2   "access_token": "ya29.
   a0AcM612wcSfz80mDU3DgiWNVKvDZ3o8-_bLT2jra9uURq1oH DU8qeujuw72L-Jc6T
   hL3SXehQs0uBITCF5s15TTXseIvJtP7pvH9NLawVYgaNV4ayYqdvwe4s45ha0Lv2H
   M0MagfwggJXa1XZA2Y_syYYJUtnE1kXy0QaCgYKAeISARISFQHGx2Mi9UiRuYATZs
   CupjCkZ30TYg0169",
3   "expires_at": 1727449690,
4   "id": "google:111704620703960376929",
5   "id_token":
   "eyJhbGciOiJIUzI1NiIsImtpZCI6IjVhYXZmNDdjMjFkMDZlMjY2Y2NmZk1YjIjX
   NDVjN2M2ZDQ3MzBlYTUuIiwiaXNjaXNjaXNjaXNjaXNjaXNjaXNjaXNjaXNjaXNj
   eyJpc3MiOiJodHRwczovL2FjY291bnRzLmdvb2dsZS5jb20iLCJhenAiOiI50Tc3M
   jA0MjYzNzgtbmJtYTF0bWEwdm1qNTVvcDhqDQ5N2lhanIzZTVkZWouYXBwcy5nb2
   9nbGV1c2VyY29udGVudC5jb20iLCJhdWQ10iI50Tc3MjA0MjYzNzgtbmJtYTF0bWE
   wdm1qNTVvcDhqDQ5N2lhanIzZTVkZWouYXBwcy5nb29nbGV1c2VyY29udGVudC5j
   b20iLCJzdWIiOiI50Tc3MjA0MjYzNzgtbmJtYTF0bWEwdm1qNTVvcDhqDQ5N2lhan
   I3U1MkNbnWFBpC5jb20iLCJlbWFBpF92ZXJpZmllZCI6dHJ1ZSwiYXRfaGZaCi6Ik
   9DSU5hamJ5YWVhd2lHR3hMd2lmdUEiLCJyZW1lIjo1Y2hhdEJndSIsInBpY3R1cmU
   iOiJodHRwczovL2x0My5nb29nbGV1c2VyY29udGVudC5jb20vYS9BQ2c4b2NKUFR2
   VUxyRTJWVTRJZmFrTUtiN1F4djVGT1VkX1FhXzk3VWVGM3NMUGlWYzRzZ0E9czk2L
```

TTL: Forever Memory: ~1683 bytes

We can see the message text here:



Pay Attention! Because it is a secure chat application, we write the encrypted chat in the DB. That's mean the only when you in chat you can see the messages. If you look the messages outside the chat, there are encrypted messages.

Pusher:

We use it because we want a realtime chat.

Enter also in the secret keys file (*we called it ./env/local*) the pusher keys:

PUSHER_APP_ID, NEXT_PUBLIC_PUSHER_APP_KEY and PUSHER_APP_SECRET.

Finally, the `./env/local` file is like this:

```
NEXTAUTH_SECRET=
UPSTASH_REDIS_REST_URL=
UPSTASH_REDIS_REST_TOKEN=
GOOGLE_CLIENT_ID=
GOOGLE_CLIENT_SECRET=
PUSHER_APP_ID=
NEXT_PUBLIC_PUSHER_APP_KEY
PUSHER_APP_SECRET=
SECRET_KEY=
```


Social engineering:

In the context of chat messaging, encryption plays a critical role in securing communication between users. We use the **AES (Advanced Encryption Standard)** algorithm to encrypt and decrypt messages, ensuring that only authorized parties with the correct decryption key can access the content of a message. AES is widely regarded for its robust security, efficiency, and adaptability across various platforms, making it an ideal choice for protecting sensitive information in a chat environment.

Despite the strong encryption used in our system, social engineering threats can compromise security by exploiting human vulnerabilities rather than technical ones. Here are some potential issues we've encountered that could expose our app to such attacks:

- 1. Domain Spoofing (Apparent Domain Name):**

A seemingly legitimate but deceptive domain name may trick users into believing they are accessing our secure application. Hackers can create phishing websites with similar-looking domain names to steal login credentials or other sensitive information, which can later be used to breach the system.

What we thought: We must ensure that our official domain is clearly communicated to users and implement security measures like SSL certificates and HTTPS to establish a secure connection, reassuring users they are on the authentic platform.

- 2. Leaked Encryption and Decryption Keys:**

If the encryption or decryption keys used in our AES-based messaging system are exposed, the security of the entire chat system is compromised. Once a hacker gains access to these keys, they can decrypt any message, bypassing all other protective layers.

What we thought: We encrypted the data all the way excepted when both users are in chat, but all the messages that store in the dbs are encrypted. That's why, hackers can still figure out when both users are in the chats and read the messages also.

Strengths and Weaknesses

- We are aware that we cannot ensure the complete safety of our app because there will always be hackers that are very competent in breaching our app.
- The site is very intuitive to use; how to add people and send messages and how to accept people it's all apparent and minimalistic to the use of the client.
- The site isn't deployed for us to communicate remotely we need to host it somewhere.
- We use middleware.ts to protect our sensitive routes such that no one without authorization can access chat routes without permission.

Summary

This project has enriched our knowledge on web Sockets and http requests a lot, and how to use encryption method and how to create schemas for the chat and Users and how to communicate between them.

Acknowledgements:

YouTube videos that helps in authentication and chat functionality without the encryption part.

<https://shadcn.com> for the buttons and tailwindcss for styling.

<https://upstash.com/docs/introduction> for DB, chat, and users.

<https://pusher.com/> for Realtime message sending.

<https://iconscout.com/icons> for icons used in the app

Chatgpt & stackoverflow - with finding bugs and suggesting stuff.