A SECURE WAY OF MESSAGING

INSTANT MESSAGING APPLICATION TO SHARE CONFIDENTIAL MESSAGES USING IMAGE STEGANOGRAPHY.

GROUP 8

Prudhvinath Reddy Katha Madan Kumar Sutapalli Vijay Durgesam

Supervised By:
Maha Ali Allouzi, PhD

CONTENTS

ABSTRACT

Secure Chat is an innovative approach to communication, focusing heavily on storage, security and integration with image scanning. Our team has focused on developing a product that will stand out in the market, rather than creating another typical chat application. We've started by prioritizing the creation of a Clean and User-friendly Interface. We have effectively used Angular, JS, and Firebase to achieve this objective, drawing inspiration from a variety of existing applications. They all work on their own, and we've been trying to combine them to get a Web Secure Chat application.

The image steganography function is one of the best features in this project which stands out from other features, it provides a new way to hide text from easy and plain pictures. It is a best way to secretly send messages to your friend or colleague in an image.

This project aims to develop a comprehensive end-to-end encryption framework for messaging web applications. It will incorporate features such as End-to-End encryption, and AES cryptography algorithm to ensure security and efficiency. Furthermore, the project will enhance security by integrating Image Steganography to share confidential messages securely.

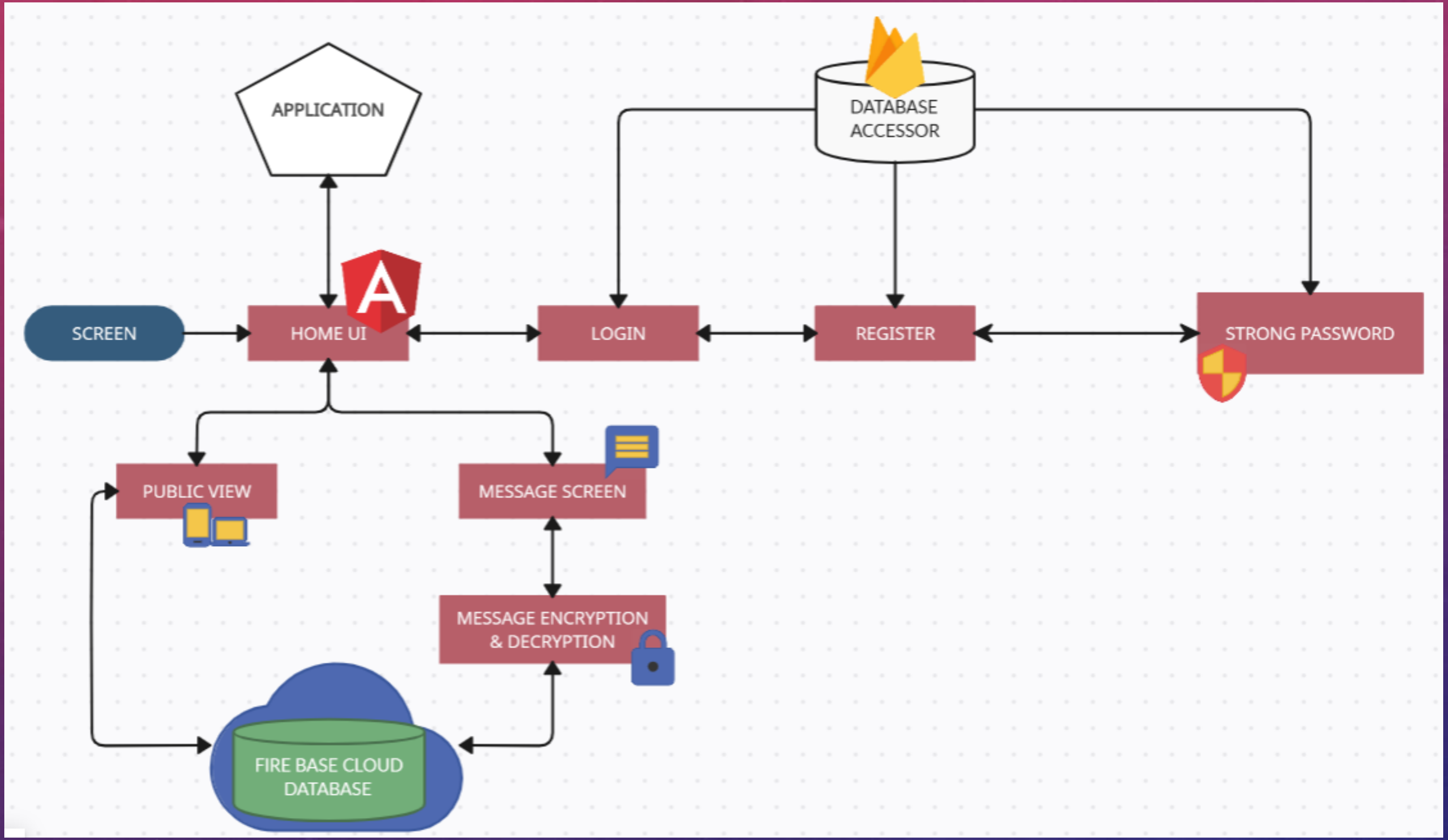
LITERATURE REVIEW

A significant focus has been given to improving digital communications security in recent years. Nurhayati, Kastari and F. Fahrianto (2022) explored the implementation of end-to-end encryption (E2EE) on Android-based instant messaging applications using the AES cryptography algorithm, stating that "their study demonstrated the effectiveness of E2EE in securing text messages" (Nurhayati et al., 2022).

Similarly, Neogi's (2022) investigation of the E2EE system in WhatsApp revealed details on its functioning and implications for security. Beyond encryption, Williamson (2021) discussed the importance of multi-factor authentication (MFA) in modern security systems, highlighting its role in enhancing protection against unauthorized access. Shirvanian and Agrawal (2021), the creators of 2D-2FA, a novel approach to two factor authentication, have renewed this concept.

Meanwhile, Subramanian et al. (2021) reviewed recent advances in image steganography, emphasizing its role as an additional layer of security. Finally, Chandani and Sharma (2023) proposed cryptographic solutions for ensuring data integrity and confidentiality which addressed the need to secure transmission of information in Internet of Things (IoT) or sensor networks. These studies are a collective contribution to the efforts being made in order to enhance safety of communication and data transmission.

ARCHITECTURE



Sample Chat Application Architecture

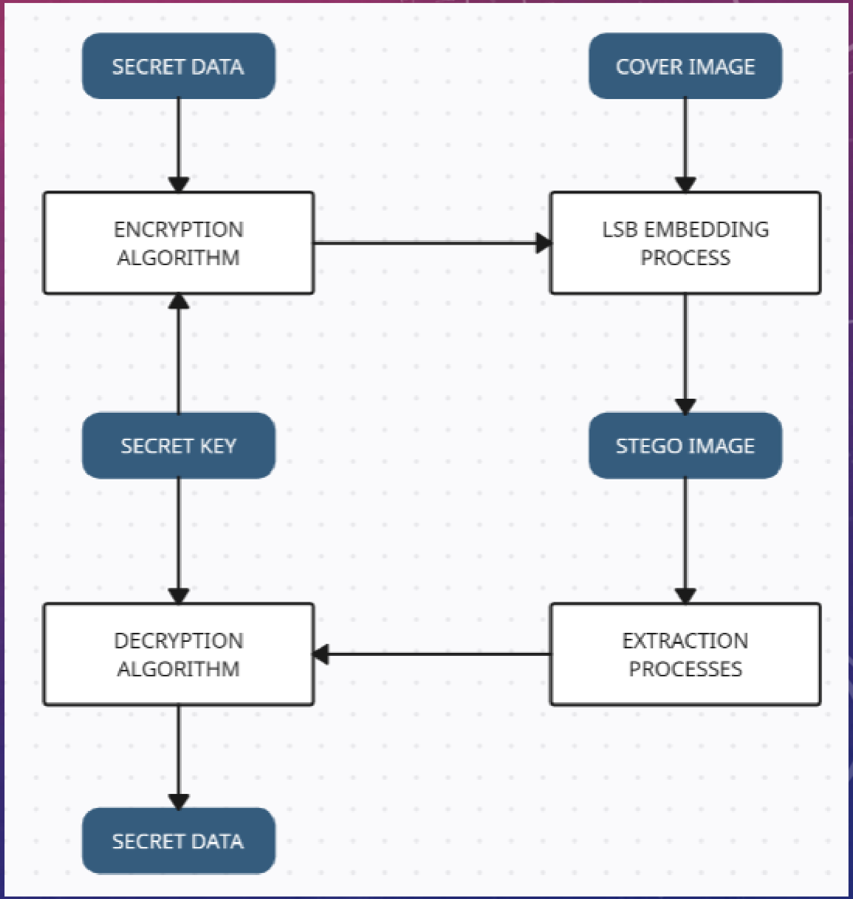
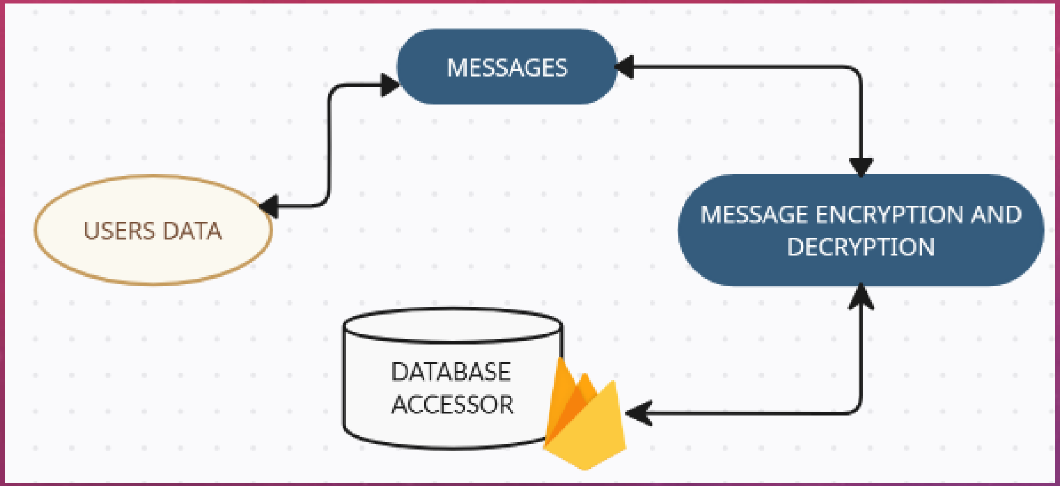
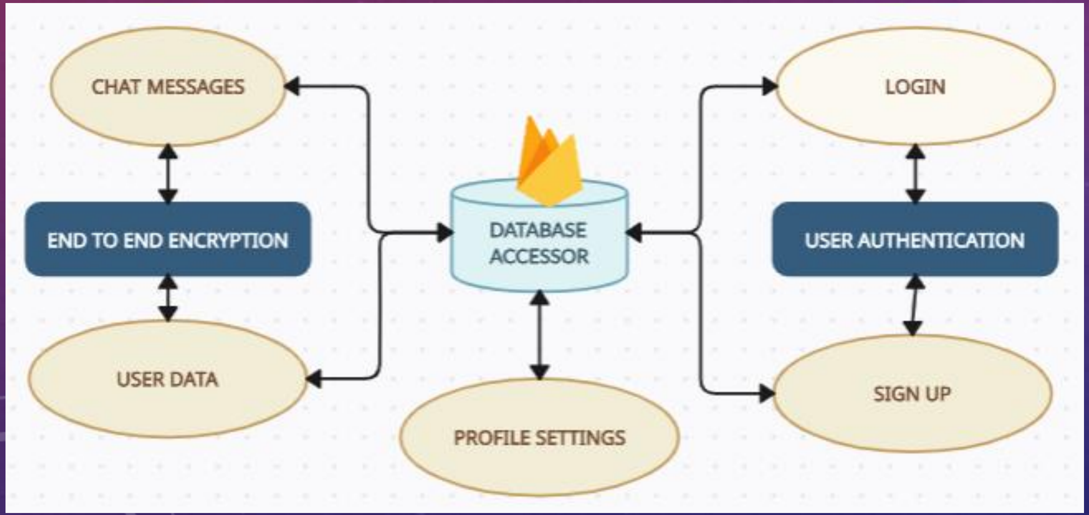


Image Steganography Architecture

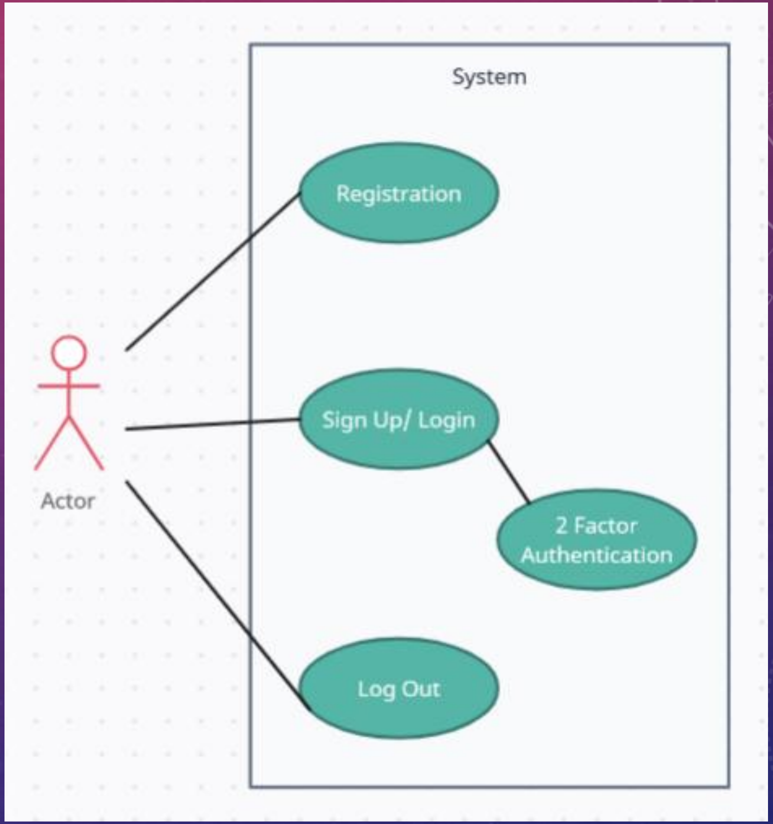
USE CASE DIAGRAMS



End to End Encryption



Different component connection to database



Authentication flow

IMPLEMENTATION – A FOCUS ON SECURITY & STORAGE

A Sample **Implementation** steps include:

- Install Node.js for building & developing angular project and manage angular code dependencies
- Setting up the firebase (login, creating project and getting environment details)
- Constructing all the angular components and start connecting to firebase by using the environment details received from firebase after project creation.
- Implement AES using CryptoJS.
- End-to-End encryption of messages implementation.
- Encode and Decode function for hiding text under an image using LSB algorithm
- For access control, use Firebase Security Rules.

Frontend – GUI - Angular

Developed the user interface by using angular where we had multiple components like home-page component, sign-up component, profile-page component, landing-page component, environments etc.

Each component does their work when called using type scripts.

Backend – Database - Firebase

Firebase's the back end support. To use some of the services such as authentication, storage, database.

Database is organized based on owner requirements and has the ability to query among the data.

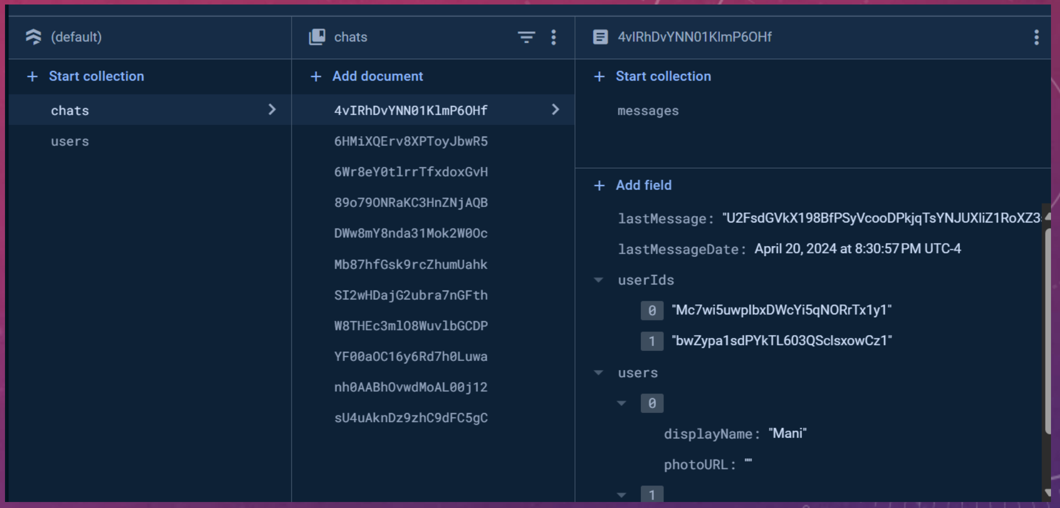
END TO END ENCRYPTION

How does it work in our case?

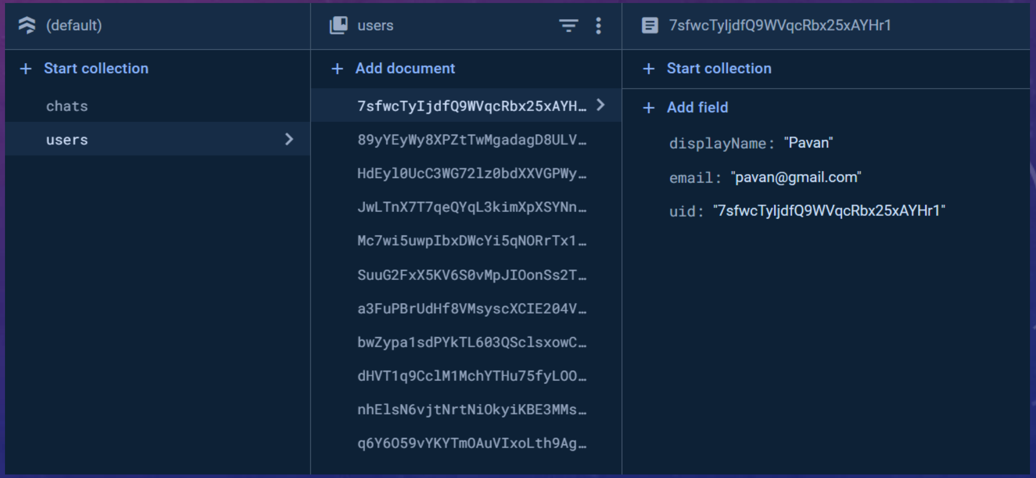
Being a chatting application all the messages should be encrypted only those who are intended to see it will see it. Both of them share an encryption key without even involvement of the third party.

Here sender encrypts using recipients public key, that message can be decrypted using recipients private key. The message is transferred over network, since its encrypted meaning it is converted to cipher text which is unreadable.

Here we have successfully implemented it in Angular components where the service provider firebase can not see the messages we share. All the data is encrypted.

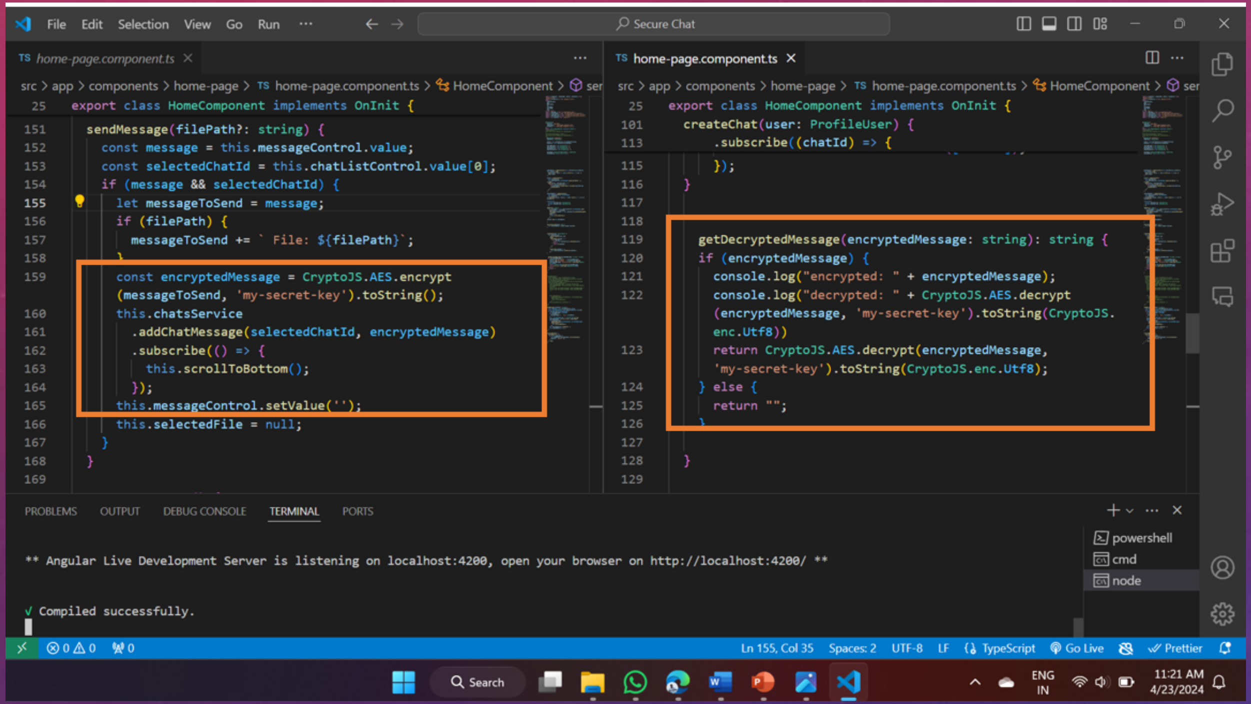


```
(default)                 chats                       4vIRhDvYNN01KlmP6OHf
+ Start collection        + Add document              + Start collection
chats                 >   4vIRhDvYNN01KlmP6OHf    >   messages
users                     6HMiXQErv8XPToyJbwR5
                          6Wr8eY0tlrrTfxdoxGvH        + Add field
                          89o79ONRaKC3HnZNjAQB        lastMessage: "U2FsdGVkX198BfPSyVcooDPkjqTsYNJUXIiZ1RoXZ3
                          DWw8mY8nda31Mok2W0Oc        lastMessageDate: April 20, 2024 at 8:30:57 PM UTC-4
                          Mb87hfGsk9rcZhumUahk        userIds
                          SI2wHDajG2ubra7nGFth            0  "Mc7wi5uwplbxDWcYi5qNORrTx1y1"
                          W8THEc3ml08WuvlbGCDP            1  "bwZypa1sdPYkTL603QSclsxowCz1"
                          YF00aOC16y6Rd7h0Luwa        users
                          nh0AABhOvwdMoAL00j12            0
                          sU4uAknDz9zhC9dFC5gC              displayName: "Mani"
                                                            photoURL: ""
                                                          1
```



```
(default)                 users                          7sfwcTyljdfQ9WVqcRbx25xAYHr1
+ Start collection        + Add document                 + Start collection
chats                     7sfwcTyIjdfQ9WVqcRbx25xAYH… >  + Add field
users                 >   89yYEyWy8XPZtTwMgadagD8ULV…    displayName: "Pavan"
                          HdEyl0UcC3WG72lz0bdXXVGPWy…    email: "pavan@gmail.com"
                          JwLTnX7T7qeQYqL3kimXpXSYNn…    uid: "7sfwcTyljdfQ9WVqcRbx25xAYHr1"
                          Mc7wi5uwpIbxDWcYi5qNORrTx1…
                          SuuG2FxX5KV6S0vMpJIOonSs2T…
                          a3FuPBrUdHf8VMsyscXCIE204V…
                          bwZypa1sdPYkTL603QSclsxowC…
                          dHVT1q9CclM1MchYTHu75fyLOO…
                          nhElsN6vjtNrtNiOkyiKBE3MMs…
                          q6Y6059vYKYTmOAuVIxoLth9Ag…
```

AES – ALGORITHM



```
export class HomeComponent implements OnInit {
  sendMessage(filePath?: string) {
    const message = this.messageControl.value;
    const selectedChatId = this.chatListControl.value[0];
    if (message && selectedChatId) {
      let messageToSend = message;
      if (filePath) {
        messageToSend += ` File: ${filePath}`;
      }
      const encryptedMessage = CryptoJS.AES.encrypt
      (messageToSend, 'my-secret-key').toString();
      this.chatsService
        .addChatMessage(selectedChatId, encryptedMessage)
        .subscribe(() => {
          this.scrollToBottom();
        });
      this.messageControl.setValue('');
      this.selectedFile = null;
    }
  }
```

```
export class HomeComponent implements OnInit {
  createChat(user: ProfileUser) {
      .subscribe((chatId) => {
      });
  }

    getDecryptedMessage(encryptedMessage: string): string {
    if (encryptedMessage) {
      console.log("encrypted: " + encryptedMessage);
      console.log("decrypted: " + CryptoJS.AES.decrypt
      (encryptedMessage, 'my-secret-key').toString(CryptoJS.
      enc.Utf8))
      return CryptoJS.AES.decrypt(encryptedMessage,
      'my-secret-key').toString(CryptoJS.enc.Utf8);
    } else {
      return "";
    }

  }
```

```
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

√ Compiled successfully.
```

How have we used?

We used AES Block cipher techniques where the data is divided in to fixed size blocks for encryption usually each block of 128 bits. Each block is encrypted independently. We use same secret key used for encryption and decryption.. Since our project is a chat application, messages should be encrypted and transferred accordingly. Here we have used End to End Encryption, which intern helps us to encrypt at one end and decrypt only at the target.

CryptoJS is a js library used for cryptographic works like encryption, decryption, hashing etc. CryptoJS enables easy implementation of AES in script based applications. Since ours is script based application, we could use it easily. It provides enhanced security & protection to data.

IMAGE STEGANOGRAPHY

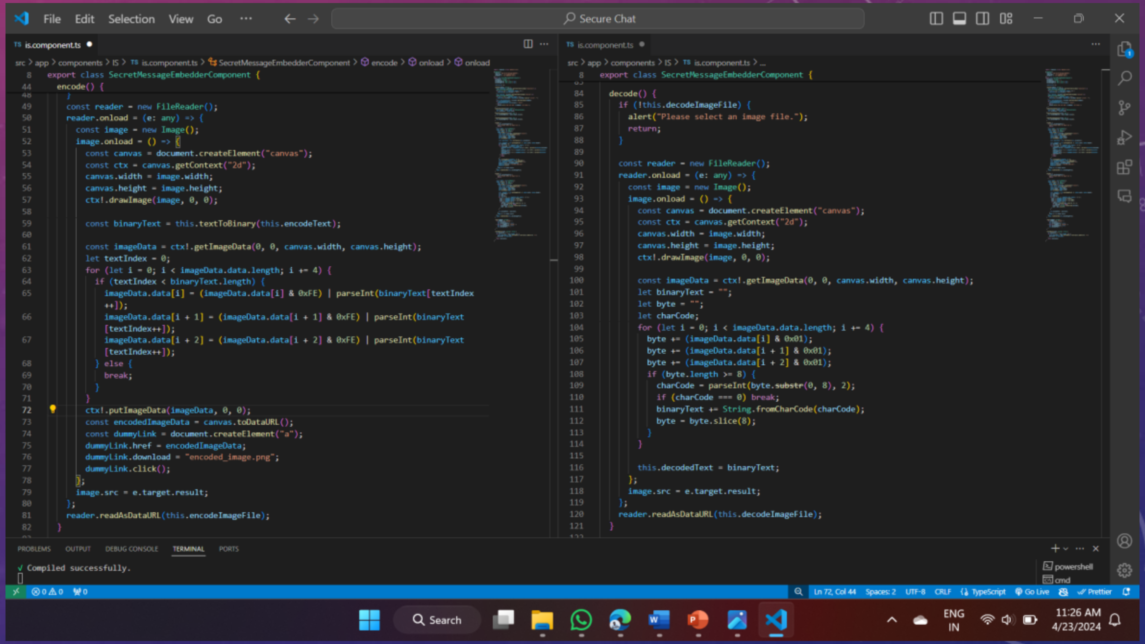
Here in this project we have used LSB algorithm to encoding and decoding the text from image.

LSB (Least Significant bit) algorithm is a technique used to hide data in the least significant nit of pixel values with out changing the image structure/appearance.

Each pixel is color coded into RGB values, here in LSB the color component can be changed with out making major changes to pixel's color.

We then replace the LSB pixels with secret message bits, thus message can be encoded in to image.

On the other end it looks for a message in the same LSB location which is implemented in decode function.


File
Edit
Selection
View
Go
Secure Chat
is.component.ts
src > app > components > IS > is.component.ts > SecretMessageEmbedderComponent > encode > onload > onload
src > app > components > IS > is.component.ts > ...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
powershell
cmd
Ln 72, Col 44
Spaces: 2
UTF-8
CRLF
TypeScript
Go Live
Search

```
export class SecretMessageEmbedderComponent {
  encode() {
    }
    const reader = new FileReader();
    reader.onload = (e: any) => {
      const image = new Image();
      image.onload = () => {
        const canvas = document.createElement("canvas");
        const ctx = canvas.getContext("2d");
        canvas.width = image.width;
        canvas.height = image.height;
        ctx!.drawImage(image, 0, 0);

        const binaryText = this.textToBinary(this.encodeText);

        const imageData = ctx!.getImageData(0, 0, canvas.width, canvas.height);
        let textIndex = 0;
        for (let i = 0; i < imageData.data.length; i += 4) {
          if (textIndex < binaryText.length) {
            imageData.data[i] = (imageData.data[i] & 0xFE) | parseInt(binaryText[textIndex
            ++]);
            imageData.data[i + 1] = (imageData.data[i + 1] & 0xFE) | parseInt(binaryText
            [textIndex++]);
            imageData.data[i + 2] = (imageData.data[i + 2] & 0xFE) | parseInt(binaryText
            [textIndex++]);
          } else {
            break;
          }
        }
        ctx!.putImageData(imageData, 0, 0);
        const encodedImageData = canvas.toDataURL();
        const dummyLink = document.createElement("a");
        dummyLink.href = encodedImageData;
        dummyLink.download = "encoded_image.png";
        dummyLink.click();
      };
      image.src = e.target.result;
    };
    reader.readAsDataURL(this.encodeImageFile);
  }
```

```
export class SecretMessageEmbedderComponent {

  decode() {
    if (!this.decodeImageFile) {
      alert("Please select an image file.");
      return;
    }

    const reader = new FileReader();
    reader.onload = (e: any) => {
      const image = new Image();
      image.onload = () => {
        const canvas = document.createElement("canvas");
        const ctx = canvas.getContext("2d");
        canvas.width = image.width;
        canvas.height = image.height;
        ctx!.drawImage(image, 0, 0);

        const imageData = ctx!.getImageData(0, 0, canvas.width, canvas.height);
        let binaryText = "";
        let byte = "";
        let charCode;
        for (let i = 0; i < imageData.data.length; i += 4) {
          byte += (imageData.data[i] & 0x01);
          byte += (imageData.data[i + 1] & 0x01);
          byte += (imageData.data[i + 2] & 0x01);
          if (byte.length >= 8) {
            charCode = parseInt(byte.substr(0, 8), 2);
            if (charCode === 0) break;
            binaryText += String.fromCharCode(charCode);
            byte = byte.slice(8);
          }
        }

        this.decodedText = binaryText;
      };
      image.src = e.target.result;
    };
    reader.readAsDataURL(this.decodeImageFile);
  }
```

```
✓ Compiled successfully.
```

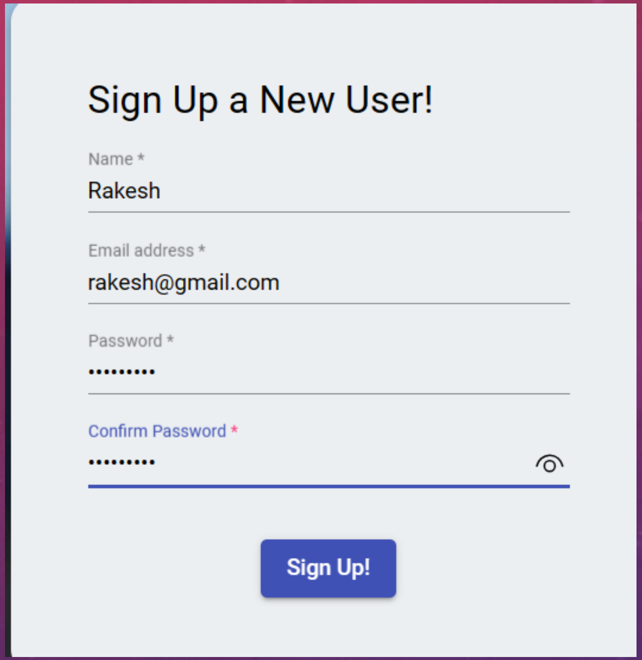
CHALLENGES & LIMITATIONS

- Connectivity issues with angular and Firebase
- Working with large amount of code often brings confusion, even simple things are changed it effects application running.

This application is limited to some features only, these are some of the limitations of this secure chat application

- The application needs internet to run
- Even provided with most security features, there is always room for some security concerns like having weak password, unauthorized login, defected systems etc.
- Secure Chat application lacks emotional indicators, such as voice tone, facial expressions, and body language, that are present in face-to-face interactions. This might make understanding the message difficult and lead to misunderstanding.

OUTPUTS


Sign Up a New User!

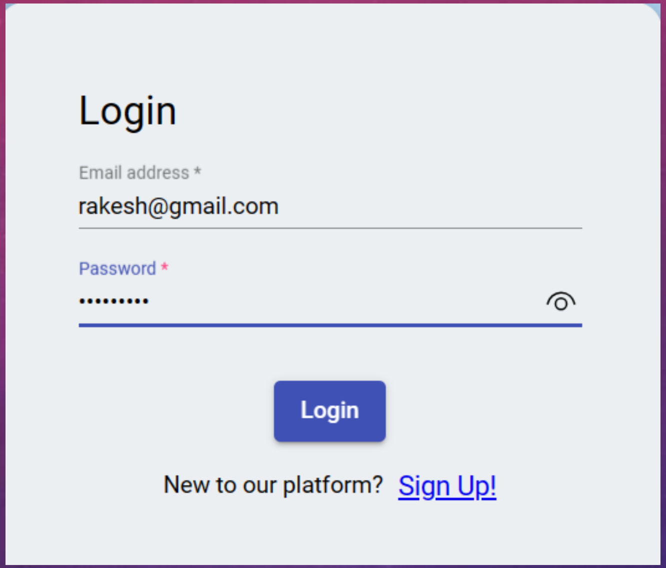
Name *
Rakesh

Email address *
rakesh@gmail.com

Password *
•••••••••

Confirm Password *
•••••••••

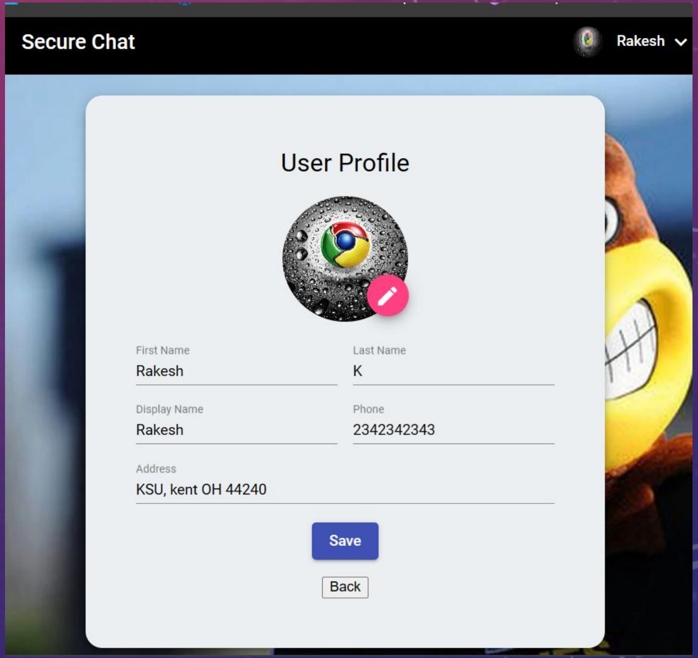
Sign-up Screen


Login

Email address *
rakesh@gmail.com

Password *
•••••••••

Login Screen

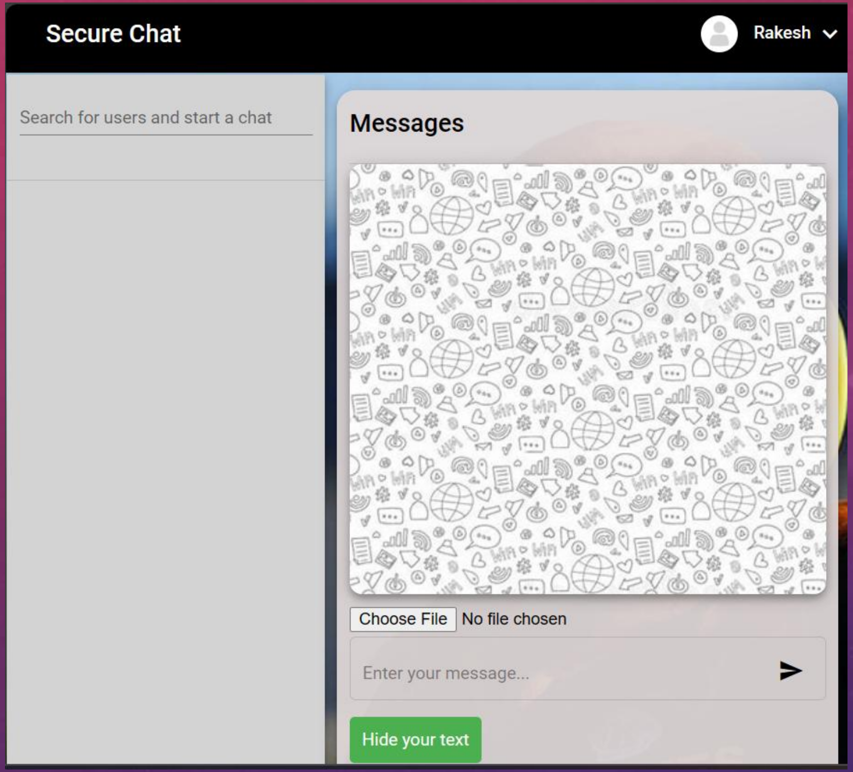


User Profile

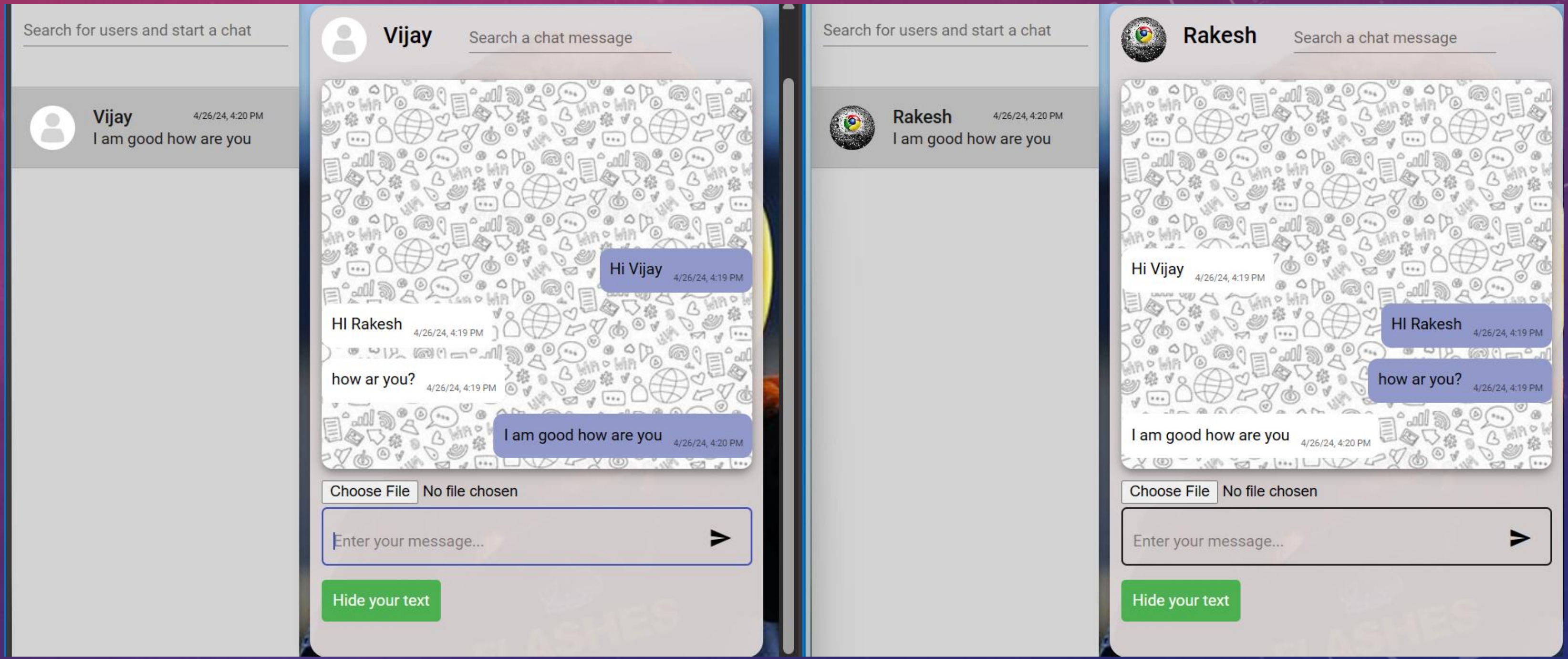
First Name: Rakesh	Last Name: K
Display Name: Rakesh	Phone: 2342342343
Address: KSU, kent OH 44240	

Profile Screen

OUTPUTS



Sample Message Screen



Vijay 4/26/24, 4:20 PM
I am good how are you

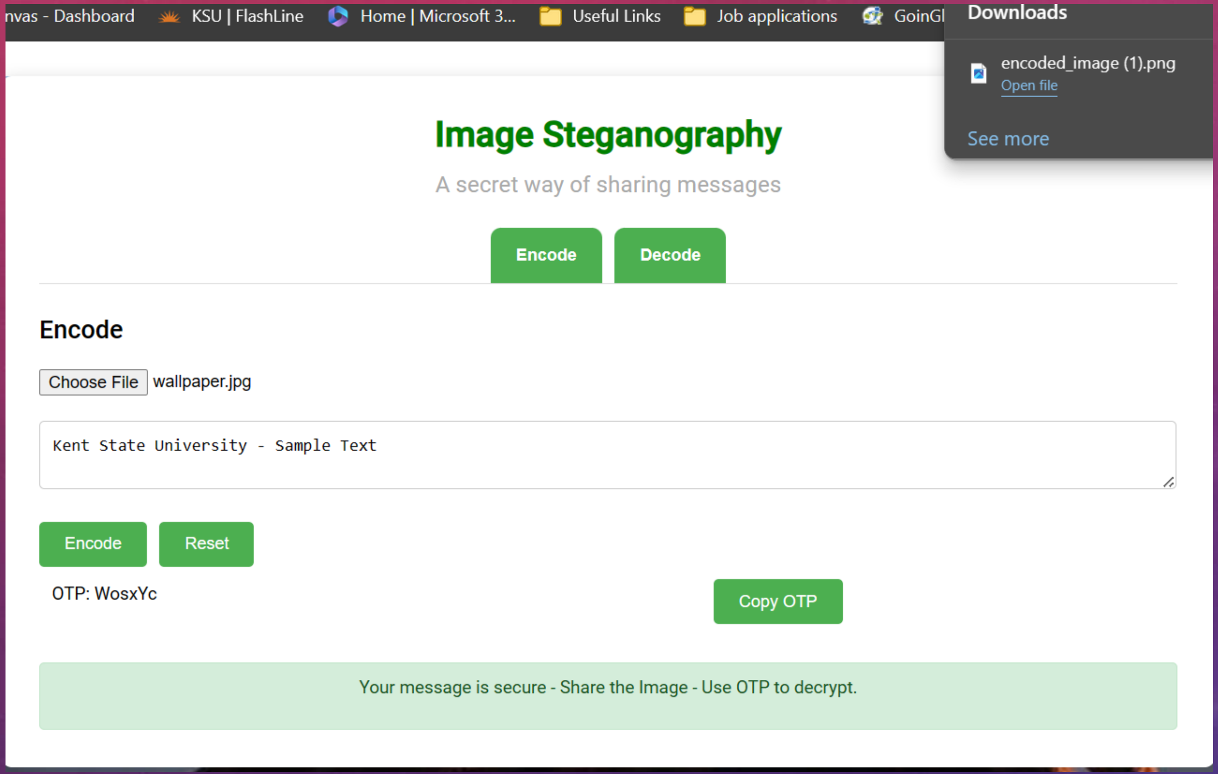
Hi Vijay 4/26/24, 4:19 PM
HI Rakesh 4/26/24, 4:19 PM
how ar you? 4/26/24, 4:19 PM
I am good how are you 4/26/24, 4:20 PM

Rakesh 4/26/24, 4:20 PM
I am good how are you

Hi Vijay 4/26/24, 4:19 PM
HI Rakesh 4/26/24, 4:19 PM
how ar you? 4/26/24, 4:19 PM
I am good how are you 4/26/24, 4:20 PM

Two way message sharing – chat screen

IMAGE STEGANOGRAPHY - OUTPUTS



nvas - Dashboard | KSU | FlashLine | Home | Microsoft 3... | Useful Links | Job applications | GoinGl

Downloads

encoded_image (1).png
Open file

See more

Image Steganography

A secret way of sharing messages

Encode | Decode

Encode

Choose File wallpaper.jpg

```
Kent State University - Sample Text
```

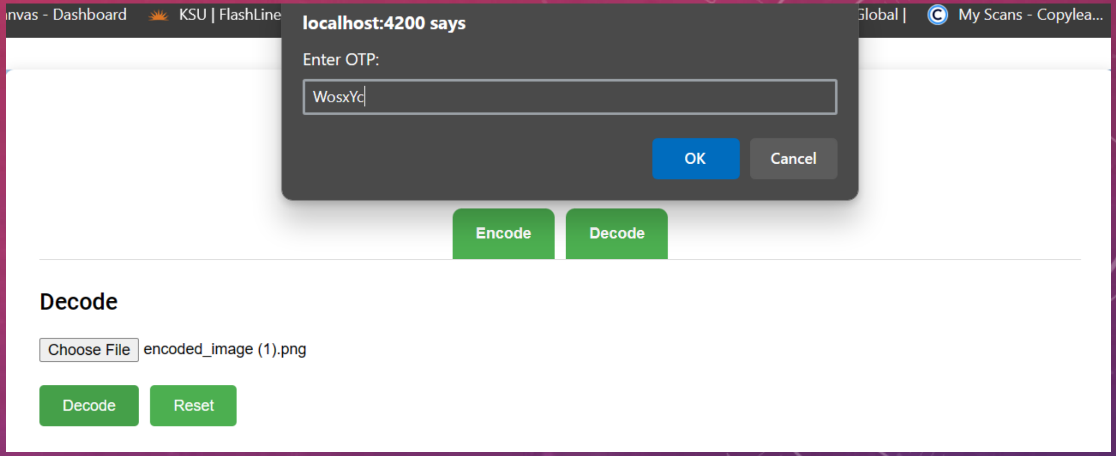
Encode | Reset

OTP: WosxYc

Copy OTP

Your message is secure - Share the Image - Use OTP to decrypt.

Text encoding screen



nvas - Dashboard | KSU | FlashLine | Global | | My Scans - Copylea...

localhost:4200 says

Enter OTP:

WosxYc

OK | Cancel

Encode | Decode

Decode

Choose File encoded_image (1).png

Decode | Reset

OTP request to decode text

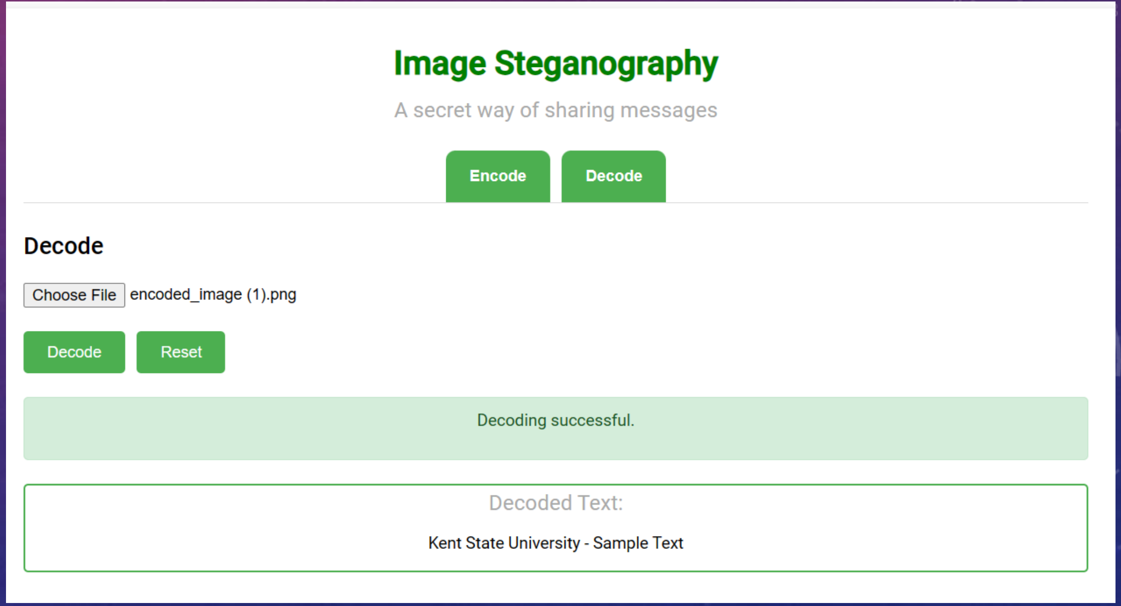


Image Steganography

A secret way of sharing messages

Encode | Decode

Decode

Choose File encoded_image (1).png

Decode | Reset

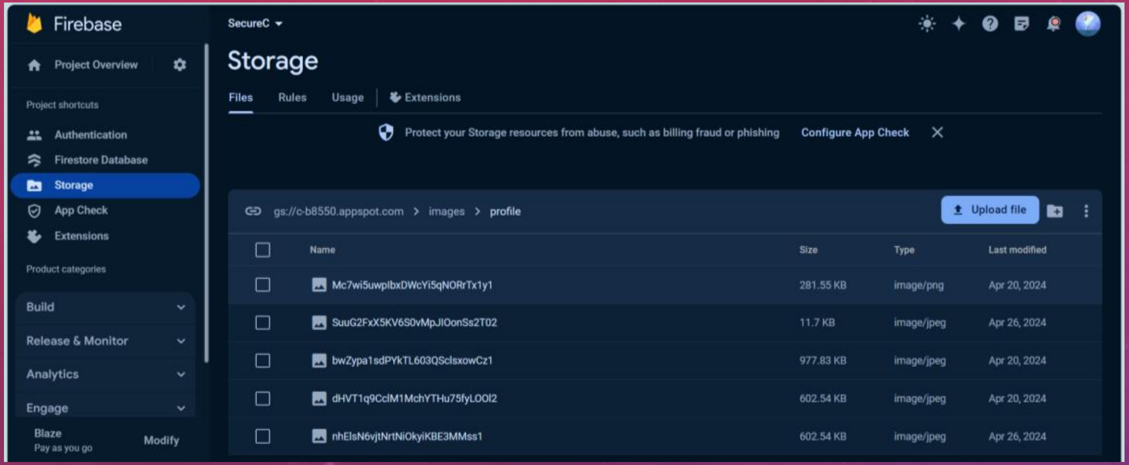
Decoding successful.

Decoded Text:

Kent State University - Sample Text

Successful decoding Screen

IMAGE STEGANOGRAPHY - OUTPUTS

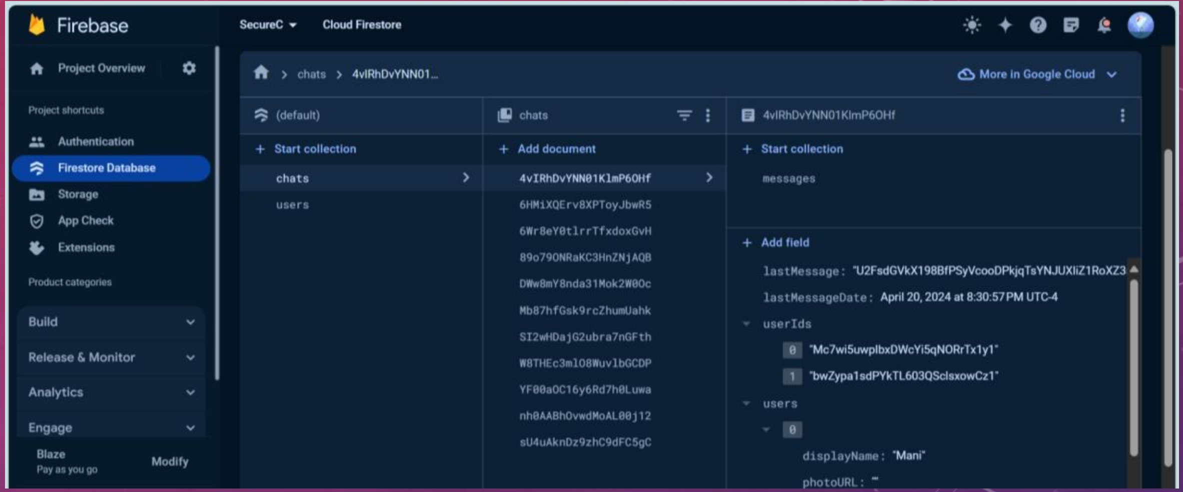


Protect your Storage resources from abuse, such as billing fraud or phishing

gs://c-b8550.appspot.com > images > profile

Name	Size	Type	Last modified
Mc7wi5uwplbxDWcYi5qNORrTx1y1	281.55 KB	image/png	Apr 20, 2024
SuuG2FxX5KV6S0vMpJIOonSs2T02	11.7 KB	image/jpeg	Apr 26, 2024
bwZypa1sdPYkTL603QSclsxowCz1	977.83 KB	image/jpeg	Apr 20, 2024
dHVT1q9CclM1MchYTHu75fyLOOl2	602.54 KB	image/jpeg	Apr 20, 2024
nhElsN6vjtNrtNiOkyiKBE3MMss1	602.54 KB	image/jpeg	Apr 26, 2024

Image Storage – profile and chat images



chats > 4vIRhDvYNN01...

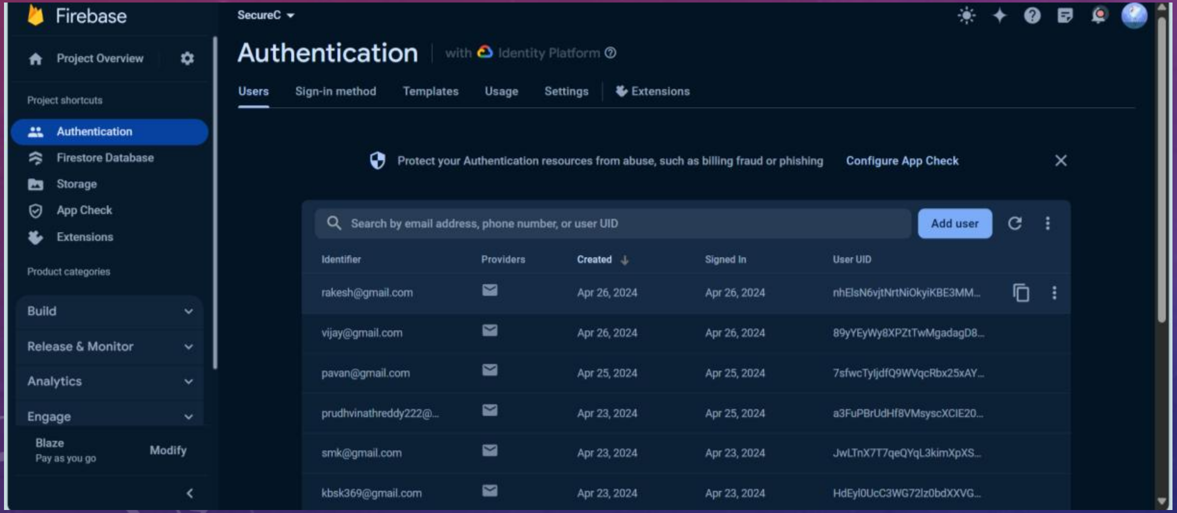
(default)
chats
users

chats
4vIRhDvYNN01KlmP6OHf
6HMiXQErv8XPToyJbwR5
6Wr8eY0tlrrTfxdoxGvH
89o790NRaKC3HnZNjAQB
DWw8mY8nda31Mok2W0Oc
Mb87hfGsk9rcZhumUahk
SI2wHDajG2ubra7nGFth
W8THEc3mlO8WuvlbGCDP
YF00a0C16y6Rd7h0Luwa
nh0AABhOvwdMoAL00j12
sU4uAknDz9zhC9dFC5gC

4vIRhDvYNN01KlmP6OHf
messages

```
lastMessage: "U2FsdGVkX198BfPSyVcooDPkjqTsYNJUXliZ1RoXZ3
lastMessageDate: April 20, 2024 at 8:30:57 PM UTC-4
userIds
    0 "Mc7wi5uwplbxDWcYi5qNORrTx1y1"
    1 "bwZypa1sdPYkTL603QSclsxowCz1"
users
    0
        displayName: "Mani"
        photoURL: ""
```

Chat data stored in database after encryption

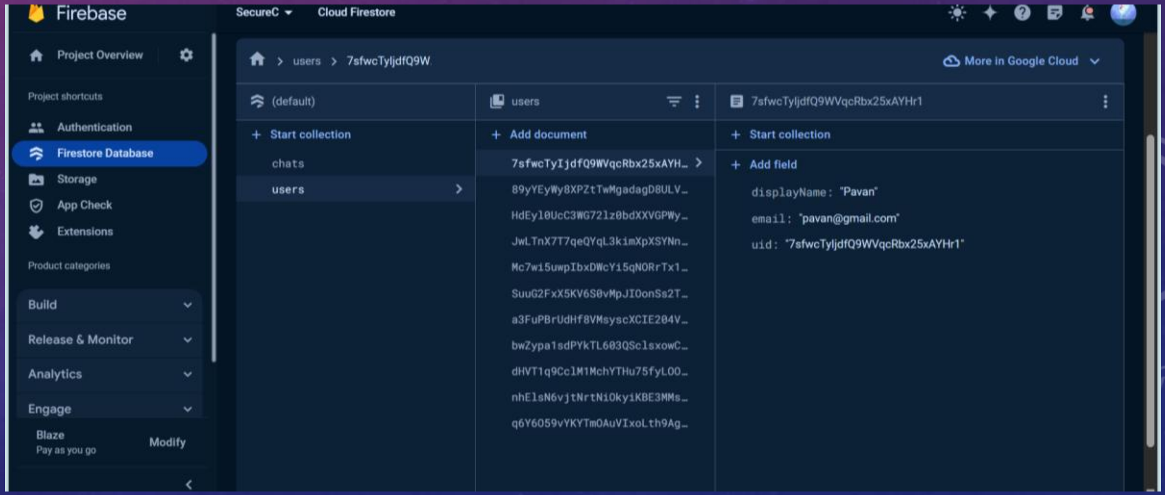


Protect your Authentication resources from abuse, such as billing fraud or phishing

Search by email address, phone number, or user UID

Identifier	Providers	Created	Signed In	User UID
rakesh@gmail.com	✉	Apr 26, 2024	Apr 26, 2024	nhElsN6vjtNrtNiOkyiKBE3MM...
vijay@gmail.com	✉	Apr 26, 2024	Apr 26, 2024	89yYEyWy8XPZtTwMgadagD8...
pavan@gmail.com	✉	Apr 25, 2024	Apr 25, 2024	7sfwcTyljdfQ9WVqcRbx25xAY...
prudhvinathreddy222@...	✉	Apr 23, 2024	Apr 25, 2024	a3FuPBrUdHf8VMsyscXCIE20...
smk@gmail.com	✉	Apr 23, 2024	Apr 23, 2024	JwLTnX7T7qeQYqL3kimXpXS...
kbsk369@gmail.com	✉	Apr 23, 2024	Apr 23, 2024	HdEyl0UcC3WG72lz0bdXXVG...

Authentication – signed up users



users > 7sfwcTyljdfQ9W

(default)
chats
users

users
7sfwcTyIjdfQ9WVqcRbx25xAYH...
89yYEyWy8XPZtTwMgadagD8ULV...
HdEyl0UcC3WG72lz0bdXXVGPWy...
JwLTnX7T7qeQYqL3kimXpXSYNn...
Mc7wi5uwpIbxDWcYi5qNORrTx1...
SuuG2FxX5KV6S0vMpJIOonSs2T...
a3FuPBrUdHf8VMsyscXCIE204V...
bwZypa1sdPYkTL603QSclsxowC...
dHVT1q9CclM1MchYTHu75fyLOO...
nhElsN6vjtNrtNiOkyiKBE3MMs...
q6Y6059vYKYTmOAuVIxoLth9Ag...

7sfwcTyljdfQ9WVqcRbx25xAYHr1

```
displayName: "Pavan"
email: "pavan@gmail.com"
uid: "7sfwcTyljdfQ9WVqcRbx25xAYHr1"
```

Profile details stored in database after encryption

FUTURE WORK

We can make additional functionalities in future for this Secure chat web application. Some of them are:

- Adding voice/ video calling functionality, image editing function, sending messages in different font styles, emojis etc.
- Animation to GUI, allow users to share large size data files with ease.
- Creating groups, Work on Image steganography like adding extra layer of protection to decrypt a message (like giving a QR code to scan the image verify the user and then decrypt the image to get the hidden text).
- Allow access to share multiple images at once.
- Camera access to take instant pictures and send in chat, news feed option to get more access to the world.

CONCLUSION

During the design phase, the project has successfully fulfilled all user requirements and prioritised data integrity and avoidance of redundancies. Team successfully developed a chat application along with an added feature of Image steganography. The interface is user friendly, with technical details and interactions aimed at making users feel comfortable using the system. A DEMO is provided, although this may not be necessary in view of the user oriented approach. The system is also flexible, which allows for changes to be made without having an impact on the functionality. The app is compatible with any version of the android operating system, making it available to users that have a different level of device capability. Our main focus was on storage & security. We achieved it by using firebase for our storage, authentication and AES for security of messages on both ends.

REFERENCES

1. *Nurhayati, Kastari and F. Fahrianto, "**End-To-End Encryption on the Instant Messaging Application Based Android using AES Cryptography Algorithm to a Text Message**," 2022 IEEE, 10th International Conference on Cyber and IT Service Management (CITSM), Yogyakarta, Indonesia, 2022, pp. 01-06, doi: 10.1109/CITSM56380.2022.9935963. e2ee_documentation.pdf*
2. *Neogi, Pinaki Prasad Guha. "**A Dive into WhatsApp's End-to-End Encryption.**" arXiv preprint arXiv:2209.11198 (2022). WhatsApp's E2EE.pdf (arxiv.org)*
3. *Williamson, J. a. (2021). **The Role of Multi-factor Authentication for Modern Day Security.** Semiconductor Science and Information Devices, 03(01). Retrieved from Multifactor_authentication.pdf (researchgate.net)*
4. *Maliheh Shirvanian and Shashank Agrawal. 2021. **2D-2FA: A New Dimension in Two-Factor Authentication**. In Proceedings of the 37th Annual Computer Security Applications Conference (ACSAC '21). Association for Computing Machinery, New York, NY, USA, 482–496. Doi: https://doi.org/10.1145/3485832.3485910 2FA_ACM_documentation.pdf*
5. *N. Subramanian, O. Elharrouss, S. Al-Maadeed and A. Bouridane, "**Image Steganography: A Review of the Recent Advances**," in IEEE Access, vol. 9, pp. 23409-23423, 2021, doi: 10.1109/ACCESS.2021.3053998. Image_steganography_IEE*
6. *P. Chandani and M. Sharma, "**Secure Data Transmission using Cryptography for Internet of Things and Sensor Networks Applications**," 2023 2nd International Conference for Innovation in Technology (INOCON), Bangalore, India, 2023, pp. 1-7, doi: 10.1109/INOCON57975.2023.10101069. Secure Data transmission*

THANK YOU..!

GROUP – 8

pkatha@kent.edu

msutapal@kent.edu

vdurgesa@kent.edu