SECURE COMMUNICATIONS AND COLLABORATION DESIGN, PROGRAMMING AND ANALYSIS

CHATBOOK

COURSE TITLE

Computer Systems Security

MODULE CODE: B9IS103

LECTURER NAME

Paul Laird

GROUP MEMBERS

Aishwarya Bafna (10632385)

Niphy Anto(10637063)

Sneha Chaudhary(10631640)

Divya Ghodke(10634832)

1. INTRODUCTION

The Chatbook web app marks a momentous stride in the digital communication landscape, fundamentally reshaping the way users interact in the online realm. In alignment with the principles upheld by all prominent social networking platforms, safeguarding data privacy and ensuring top-tier security are pivotal in fostering a safe and trustworthy user experience. This report delves into the foundational aspect of Chatbook's security architecture – its advanced encryption and decryption techniques.

Central to Chatbook's allure is its seamless and secure exchange of messages and multimedia content. To shield user data from potential threats and prevent unauthorized access, the application employs cutting-edge encryption and decryption mechanisms. By adopting these robust security measures, Chatbook instils confidence in users as they engage and communicate within the platform.

Through a comprehensive analysis of Chatbook's encryption and decryption practices, this report aims to shed light on the meticulous measures taken to uphold user privacy and data security. Our exploration will encompass an in-depth examination of the encryption algorithms and keys employed.

The core objective of this report is to illuminate Chatbook's unwavering commitment to safeguarding user privacy, cultivating a digital environment that encourages connection and shared experiences without compromising sensitive information. As we embark on this endeavour, we shall explore how Chatbook's formidable security measures contribute to the app's burgeoning popularity and, in turn, how they shape the ever-evolving landscape of digital communication.

By unearthing the intricacies of Chatbook's encryption and decryption practices, we seek to provide a comprehensive understanding of the measures undertaken to protect user data, earning the trust and loyalty of its users. In traversing this investigative path, we aim to reveal the essence of Chatbook's dedication to data protection and its pivotal role in defining the future of secure and reliable digital communication.

1.1 FUNCTIONAL REQUIREMENTS:

i. User Registration: Users must be able to create accounts and register their identities with the system

ii. End-to-End Encryption: In order to ensure that only the intended recipients can decrypt and read the messages sent between two parties, end-to-end encryption should be used.

iii. Key Exchange: The program should make it possible for strangers to exchange keys securely. Public key cryptography or other safe key exchange protocols could be used to accomplish this.

iv. Message Delivery: Delivering encrypted messages between users should be timely and reliable thanks to the system.

1.2 NONFUNCTIONAL REQUIREMENTS:

i. Security: The system gives priority first security, and it is confidential. It will be designed to refused attacks such as eavesdropping, man-in-the-middle attacks, and data breaches.

ii. Performance: The web application will be Responsive, and it provides low level suspension in order to deliver message.

iii. Scalability: The system will be allowed to accommodate an increasing number of users and messages without compromising performance.

iv. Usability: The Interface of the application will be intuitive and easy to use and ensures a positive impact experience.

v. Reliability: The application will be highly available and relies, minimizes downtime and data loss.

1.3 SECURITY REQUIREMENTS:

i. End-to-End Encryption: Keys must be securely exchanged and managed, and messages must be encrypted using powerful cryptographic algorithms.

ii. Secure Key Exchange: The application should employ secure key-exchange protocols, such as public-key cryptography or Diffie-Hellman key exchange.

iii. Authentication: To prevent unauthorized access, users must be verified using the identity verification method of choice.

iv. Data Protection: Both in transit and at rest, user data must be protected, including encrypted messages.

v. Secure Server: To prevent unauthorized access, the server handling message relay and user authentication needs to be securely configured.

2. TECHNOLOGY USED FOR CHATBOOK

This report delves into the underlying technology stack of a web application, incorporating Angular for the frontend and Flask for the backend. The application's architecture and design choices are critically evaluated, considering the benefits and drawbacks of each technology in achieving the desired user experience and performance.

1. **Angular: Empowering the Frontend**

Angular, a powerful frontend framework developed by Google, serves as the foundation for creating dynamic single-page applications (SPAs). Its component-based architecture promotes modularity and code reusability, allowing developers to build complex user interfaces efficiently. The two-way data binding feature ensures seamless synchronization between data and UI elements, enhancing real-time user interactions.

By employing Agular’s dependency injection mechanism, developers can efficiently manage and share dependencies between components, leading to a maintainable and flexible codebase. The templating engine and directives further empower developers to create expressive and interactive UI elements, fostering a compelling user experience.

2. Flask: Enabling Robust Backend Operations

On the backend side, Flask, a lightweight Python micro-framework, plays a pivotal role in handling server-side operations. Its minimalist architecture enables developers to build a tailored backend environment based on the specific project requirements. Flask excels in URL mapping and routing, simplifying the handling of HTTP requests and responses.

Through the integration of the Jinja templating engine, Flask facilitates dynamic content rendering on the server-side, enhancing the application's rendering efficiency. Additionally, Flask-SQLAlchemy streamlines the interaction with databases through object-relational mapping, optimizing data management and retrieval.

The technology stack of Angular for the frontend and Flask for the backend offers a well-rounded approach to web application development. Angular empowers developers to create responsive and interactive SPAs, while Flask provides the necessary flexibility and efficiency to handle backend operations. By carefully assessing the strengths and limitations of both technologies, developers can ensure the creation of a robust and user-friendly web application.

The following sections of this report will delve deeper into each technology, exploring their specific functionalities and implementation within the web application. Through a comprehensive evaluation, we aim to gain valuable insights into the effectiveness of the chosen technology stack and its impact on the overall performance and user experience of the web application.

3. LIBRARY USED

The increasing importance of data security has prompted developers to seek robust encryption solutions to safeguard sensitive information. In this report, we delve into two prominent encryption libraries, RSA and JSON Encrypt, known for their efficacy in the realm of software development. The RSA library, based on asymmetric encryption, and the JSON Encrypt library, designed specifically for JSON data, offer crucial tools for developers to fortify data integrity and confidentiality across various applications.

1. RSA Library: Strengthening Data Protection with Asymmetric Encryption

The RSA encryption algorithm, named after its creators Rivest, Shamir, and Adleman, is widely adopted for its asymmetric encryption properties. It employs public and private key pairs, facilitating secure data exchange between parties without requiring the prior sharing of secret keys. Key aspects of the RSA library include:

a. Key Generation: The RSA library empowers developers to generate public and private key pairs. The public key encrypts data, while the private key, known only to the recipient, decrypts the information, ensuring confidentiality.

b. Encryption and Decryption: Using RSA encryption, developers can encrypt sensitive data with the recipient's public key, rendering it indecipherable to unauthorized entities. Upon receipt, the recipient employs their private key for decryption, ensuring data integrity and confidentiality.

c. Digital Signatures: RSA serves a vital role in generating digital signatures, verifying data authenticity and integrity for both the sender and the recipient.

2. JSON ENCRYPT LIBRARY: INCREASING THE SECURITY OF JSON DATA.

Applications using JSON objects are catered to by the JSON Encrypt library, which provides JSON data with specialized security. Its main objective is to selectively encrypt individual JSON attributes or whole objects, enhancing data security during transmission and storage. The JSON Encrypt library's salient characteristics are as follows.

a. **Granular JSON Encryption:** The library enables a meticulous approach to data security by letting developers select JSON attributes for encryption.

b. **Key Management:** To ensure smooth encryption and decryption processes without sacrificing performance, JSON Encrypt streamlines key management for JSON attributes.

c. **JSON encryption:** It allows developers to secure data during transmission across networks, reducing the likelihood of data interception and tampering.

The RSA and JSON Encrypt libraries stand out as crucial players in the field of data security, in the end. While the JSON Encrypt library offers specialized security for JSON data, RSA's asymmetric encryption capabilities enable secure data exchange and digital signatures.

These encryption libraries have the potential to help developers implement strong security controls in their applications, protecting sensitive data from unauthorized access and maintaining data integrity over the course of the application's lifecycle. Developers can make well-informed decisions when choosing encryption techniques suited to their particular use cases by having a thorough understanding of the advantages and functionalities of RSA and JSON Encrypt.

Developers can strengthen data security, build user trust, and make data breaches and unauthorized access difficult for bad actors by deftly integrating RSA and JSON Encrypt libraries into their applications. The importance of encryption libraries like RSA and JSON Encrypt in protecting sensitive data from outside threats is becoming more and more apparent as the landscape of data security continues to change (www.w3schools.com).

4. RSA ALGORITHM

The RSA algorithm stands as a widely embraced method of asymmetric encryption within the domain of cryptography. Its development credits Ron Rivest, Adi Shamir, and Leonard Adleman, the initials of whom inspired its name. Asymmetric encryption revolves around the use of two distinct yet interconnected keys: the public key and the private key.

During the process of key generation, each user fashions a unique pair of keys, comprising the public key, openly disseminated and shared among users, and the private key, known exclusively to the key holder, safeguarded from unauthorized eyes.

To secure communication, when User A aims to transmit a message to User B, User A employs User B's public key for encryption. This entails conducting specific mathematical operations on the plaintext message, yielding ciphertext, rendering the original message indecipherable. Ciphertext can traverse unsecured channels without fear of unauthorized interception.

For decryption, upon receipt of the ciphertext, User B employs their private key to unveil the original message. Applying precise mathematical operations on the ciphertext using the private key reveals the plaintext message. Solely User B, equipped with the private key, possesses the capacity to comprehend and interpret the message.

The strength of the RSA algorithm emanates from the complexity of factoring large numbers into their prime factors. Encryption's potency corresponds to the key size, measured in bits. Larger key sizes heighten security by introducing intricacy into the factoring process. As of present, no swift algorithm exists to efficiently factorize large numbers, cementing RSA's prominence as an ideal selection for secure data encryption. Applications of RSA span diverse areas, encompassing secure communication, digital signatures, and key exchange in SSL/TLS protocols. It assumes a pivotal role as a cornerstone of modern data security, furnishing a reliable means of protecting sensitive information and assuring privacy across numerous digital applications (Educative: Interactive Courses for Software Developers.).

5. WORKFLOW

Chatbook is a chat-based web application that helps you in connecting with your friends, families & colleagues. While creating this application we kept security & privacy of the users in our mind and for this we implemented security algorithms like RSA.

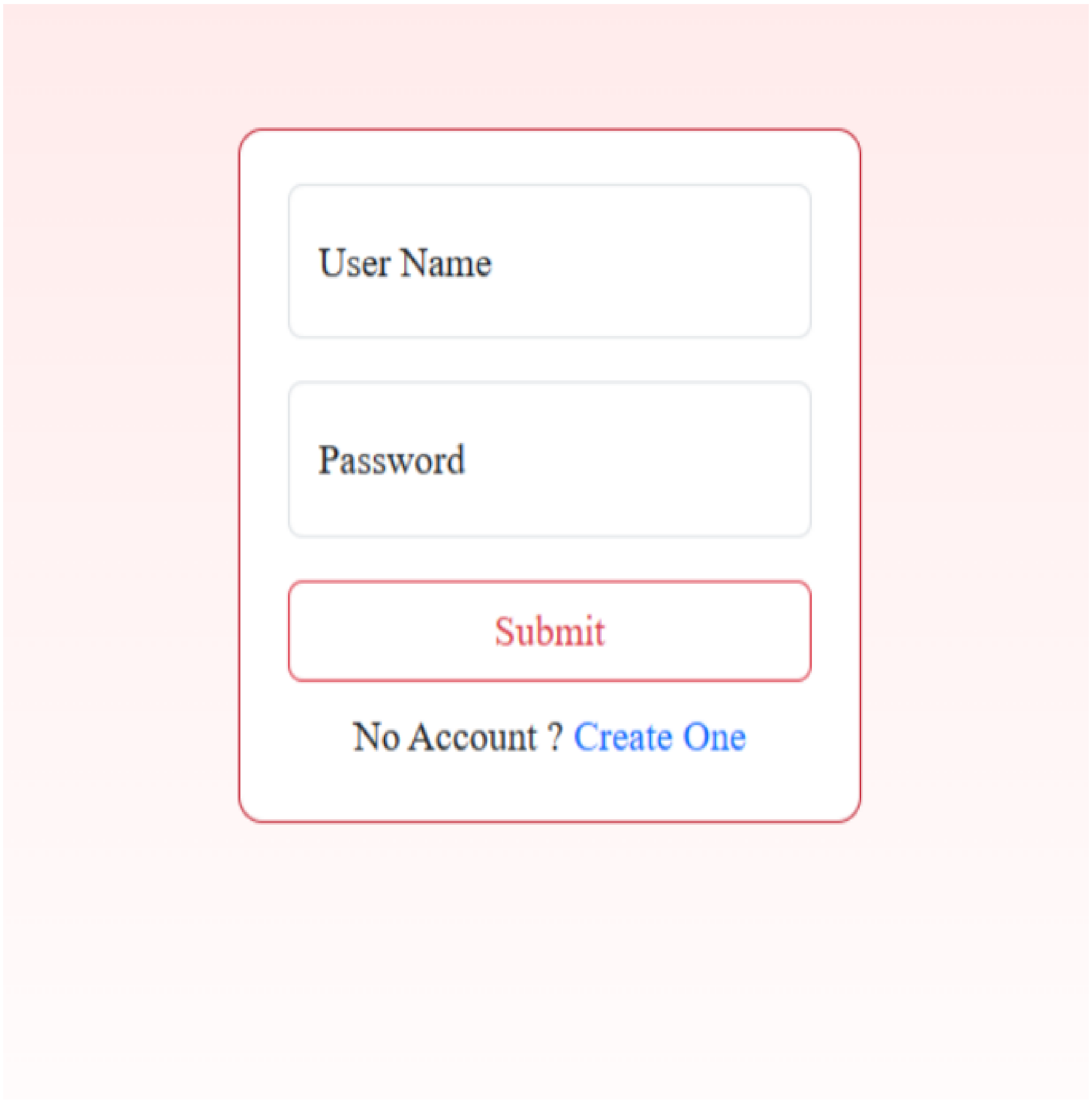


Fig 1 : Login. Screen for ChatBook

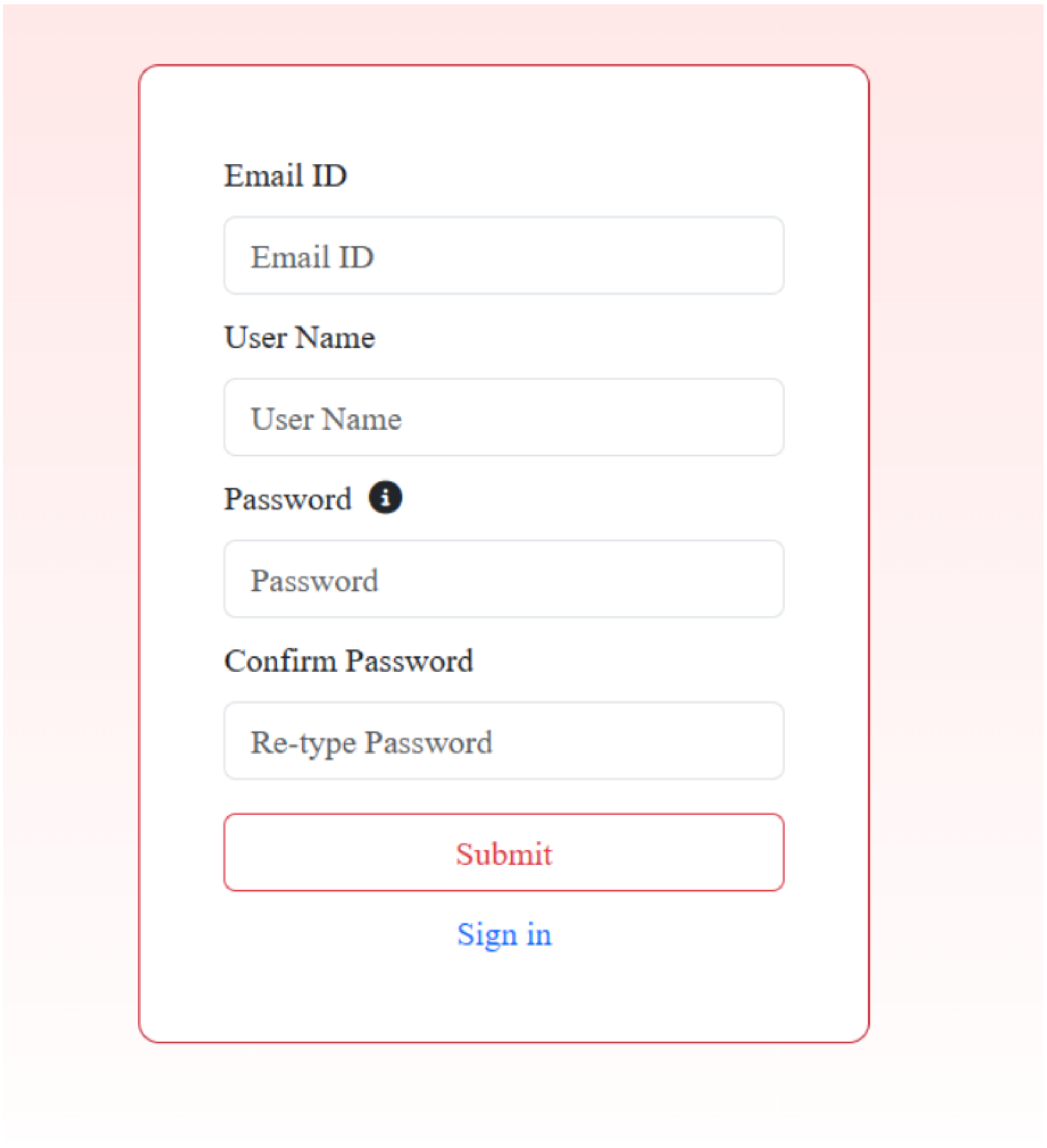


Fig 2 : Register Page of ChatBook

a. When a user accesses the chat application 'CHATBOOK', the front-end built with Angular welcomes them with a user-friendly interface.
b. Now guest user can be able to register in our application via using Create One option and registered user can directly login in our application.

Now there are two methods for sending messages:

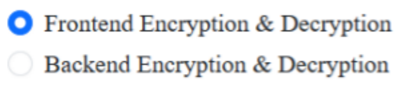
I. Frontend Encryption & Decryption –

i. If user wants to send the message it can be done via send section.

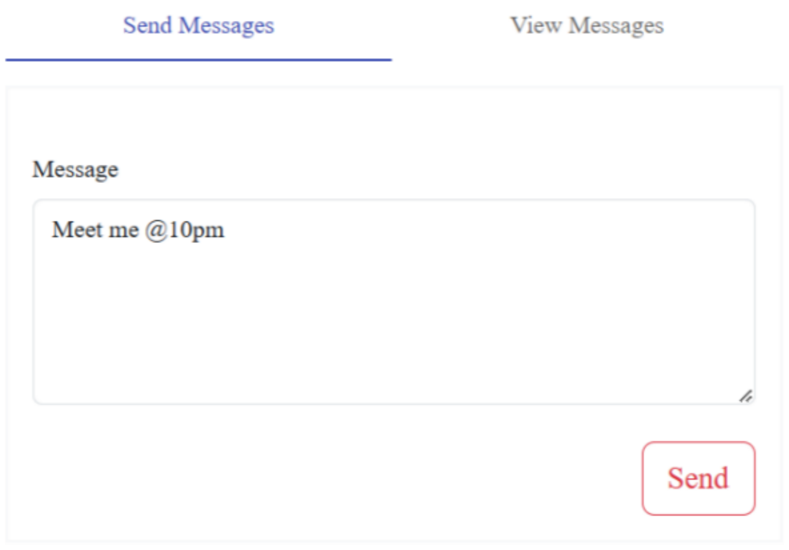
ii. As users begin typing their messages, the front-end initiates the encryption process. This process involves converting the plaintext messages into unreadable ciphertext using the RSA encryption algorithm.

iii. If user wants to view the message it can be seen under view section.

iv. Now in view section the message will be first decrypted and then actual message is visible.

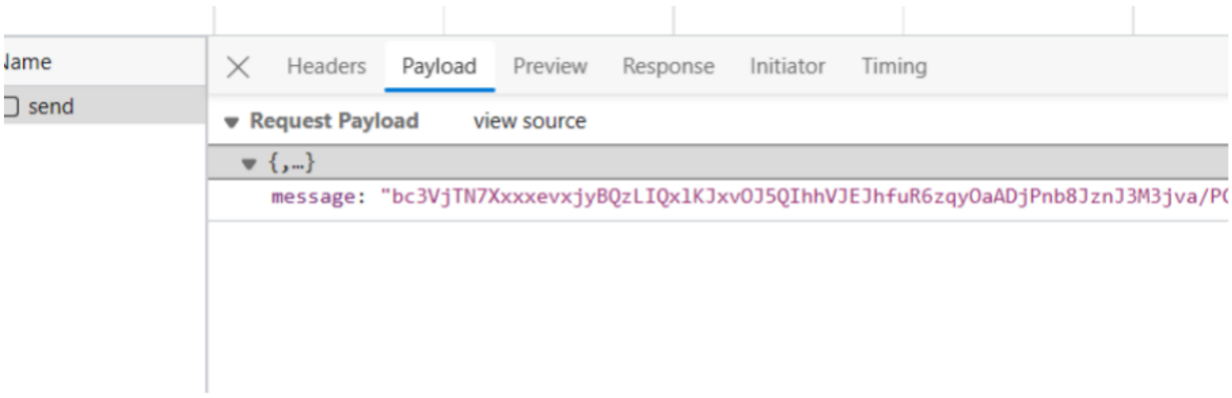


● Frontend Encryption & Decryption
○ Backend Encryption & Decryption



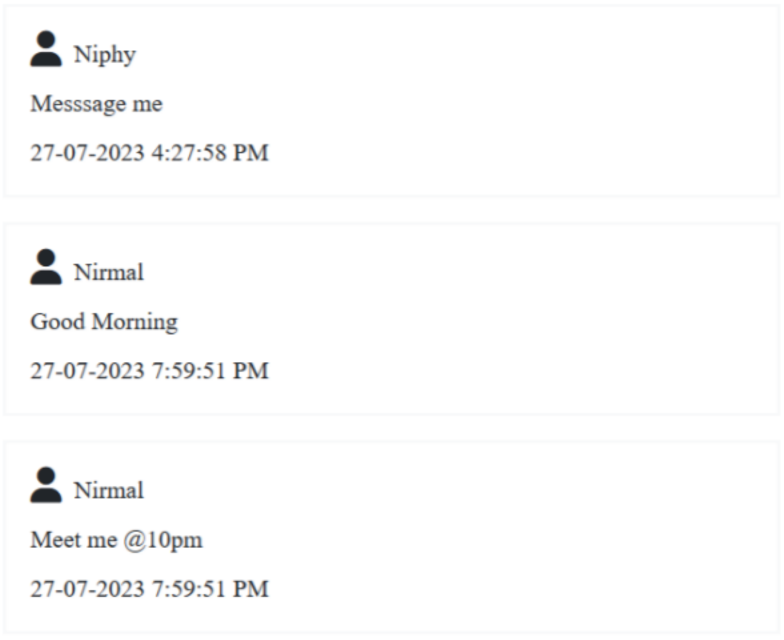
Message: Meet me @10pm

Fig 3 : Sending message to backend



message: "bc3VjTN7XxxxevxjyBQzLIQx1KJxvOJ5QIhhVJEJhfuR6zqyOaADjPnb8JznJ3M3jva/P

Fig 4 : Receiving encrypted messages from backend



Niphy

Messsage me

27-07-2023 4:27:58 PM

Nirmal

Good Morning

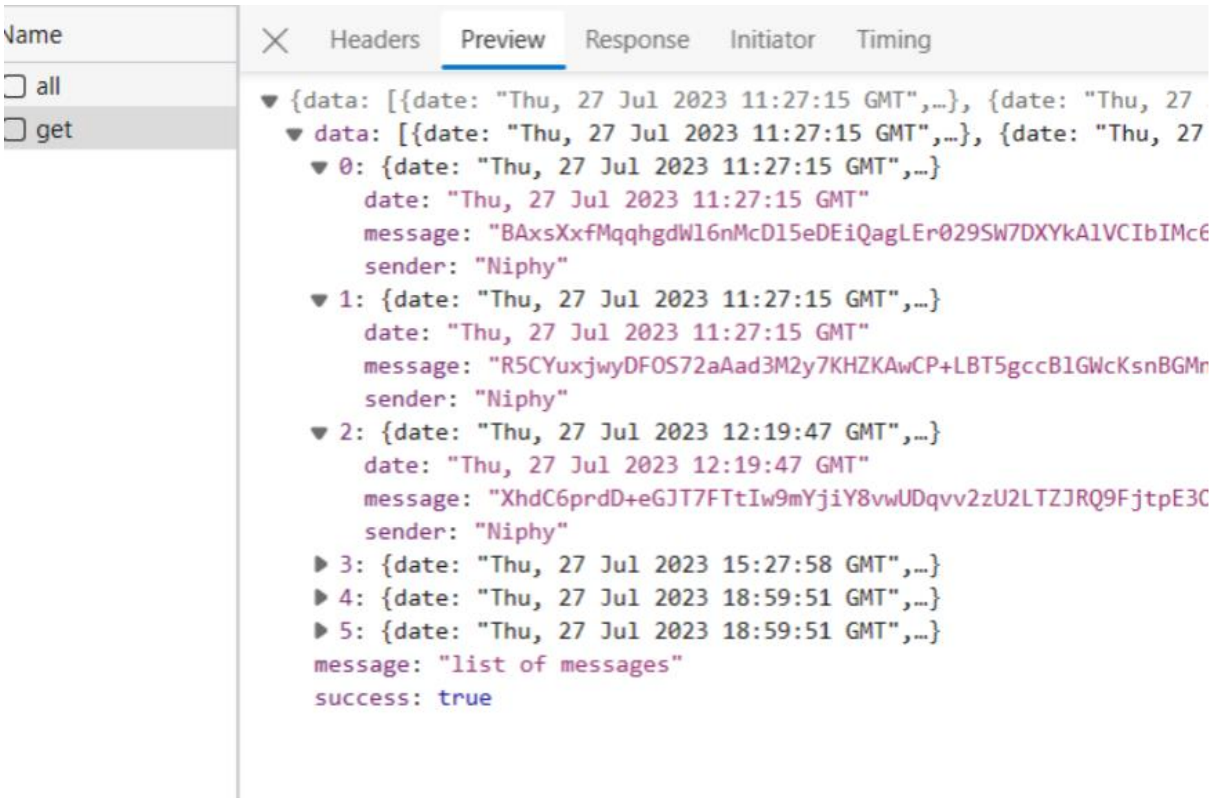
27-07-2023 7:59:51 PM

Nirmal

Meet me @10pm

27-07-2023 7:59:51 PM

Fig 4.1 : Receiving encrypted messages from backend



```
▼ {data: [{date: "Thu, 27 Jul 2023 11:27:15 GMT",…}, {date: "Thu, 27
  ▼ data: [{date: "Thu, 27 Jul 2023 11:27:15 GMT",…}, {date: "Thu, 27
    ▼ 0: {date: "Thu, 27 Jul 2023 11:27:15 GMT",…}
        date: "Thu, 27 Jul 2023 11:27:15 GMT"
        message: "BAxsXxfMqqhgdWl6nMcDl5eDEiQagLEr029SW7DXYkAlVCIbIMc6
        sender: "Niphy"
    ▼ 1: {date: "Thu, 27 Jul 2023 11:27:15 GMT",…}
        date: "Thu, 27 Jul 2023 11:27:15 GMT"
        message: "R5CYuxjwyDFOS72aAad3M2y7KHZKAwCP+LBT5gccBlGWcKsnBGMn
        sender: "Niphy"
    ▼ 2: {date: "Thu, 27 Jul 2023 12:19:47 GMT",…}
        date: "Thu, 27 Jul 2023 12:19:47 GMT"
        message: "XhdC6prdD+eGJT7FTtIw9mYjiY8vwUDqvv2zU2LTZJRQ9FjtpE3C
        sender: "Niphy"
    ▶ 3: {date: "Thu, 27 Jul 2023 15:27:58 GMT",…}
    ▶ 4: {date: "Thu, 27 Jul 2023 18:59:51 GMT",…}
    ▶ 5: {date: "Thu, 27 Jul 2023 18:59:51 GMT",…}
    message: "list of messages"
    success: true
```

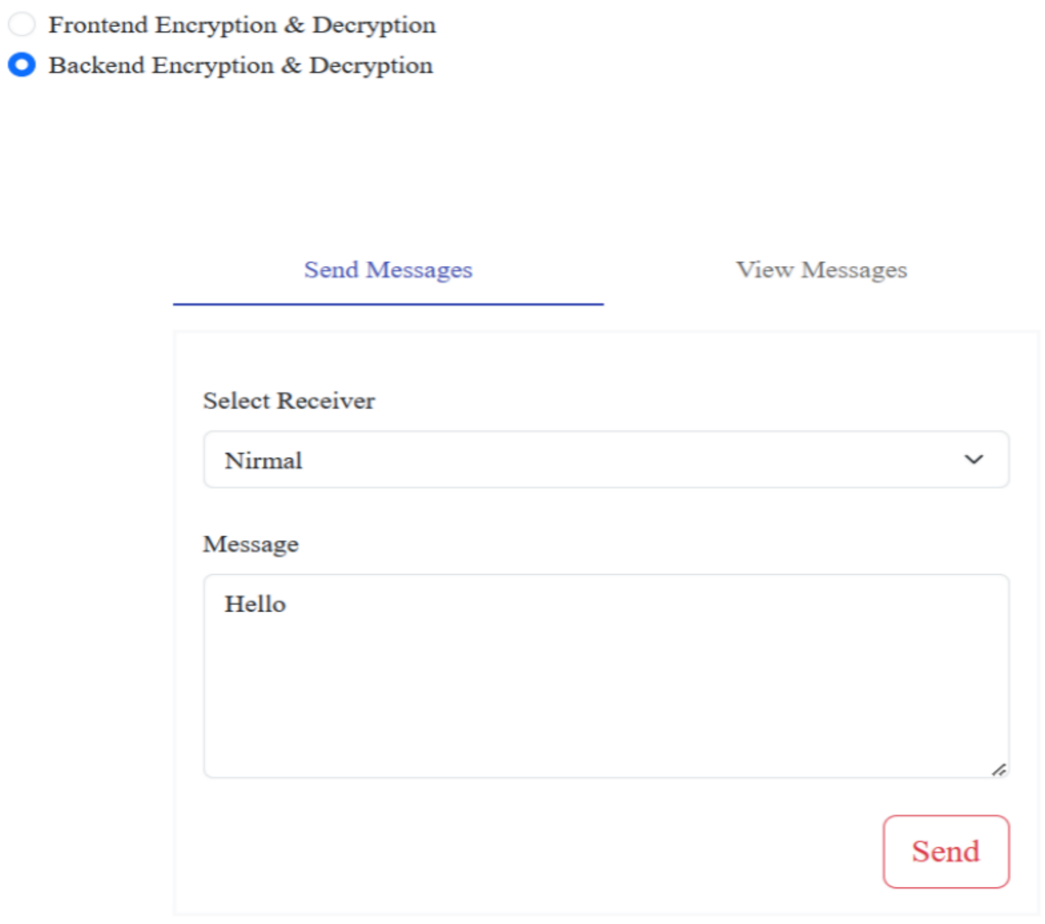
Fig 4.2 : Receiving encrypted messages from backend

II. Backend Encryption & Decryption.

In this part, two users are required to send & receive the message.

i. User 1 will send a message to user 2.

ii. Now the message will be encrypted in backend and while receiving the message, it will be decrypted first and then user 2 will get actual message.



( ) Frontend Encryption & Decryption
(●) Backend Encryption & Decryption

Send Messages | View Messages

Select Receiver

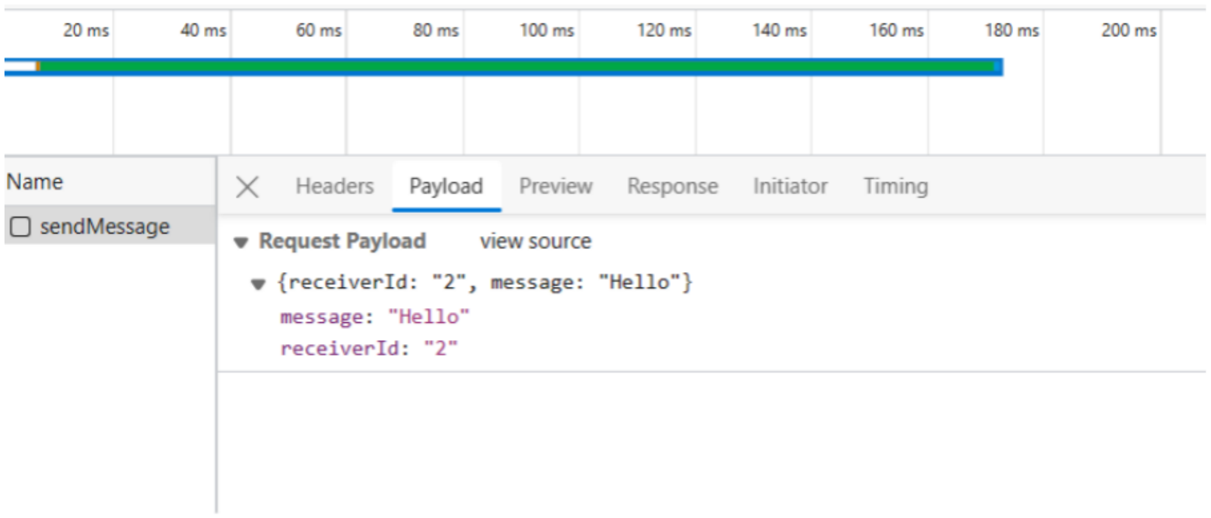
Nirmal

Message

Hello

Send

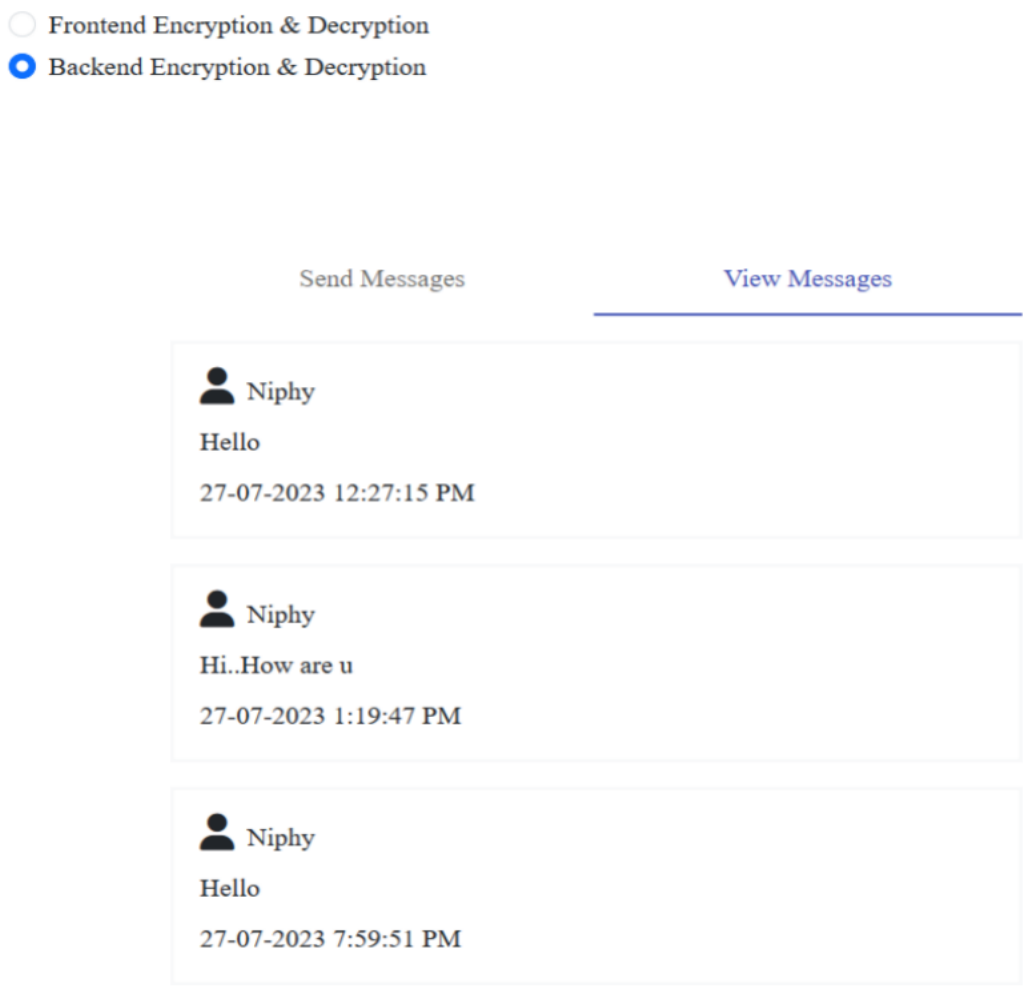
Fig 5 : Sending normal message to backend and decryption will be done in backend.



sendMessage

```
▼ Request Payload      view source
  ▼ {receiverId: "2", message: "Hello"}
      message: "Hello"
      receiverId: "2"
```

Fig 6 : Receiving decrypted messages from backend.


Frontend Encryption & Decryption
Backend Encryption & Decryption
Send Messages
View Messages

Niphy

Hello

27-07-2023 12:27:15 PM

Niphy

Hi..How are u

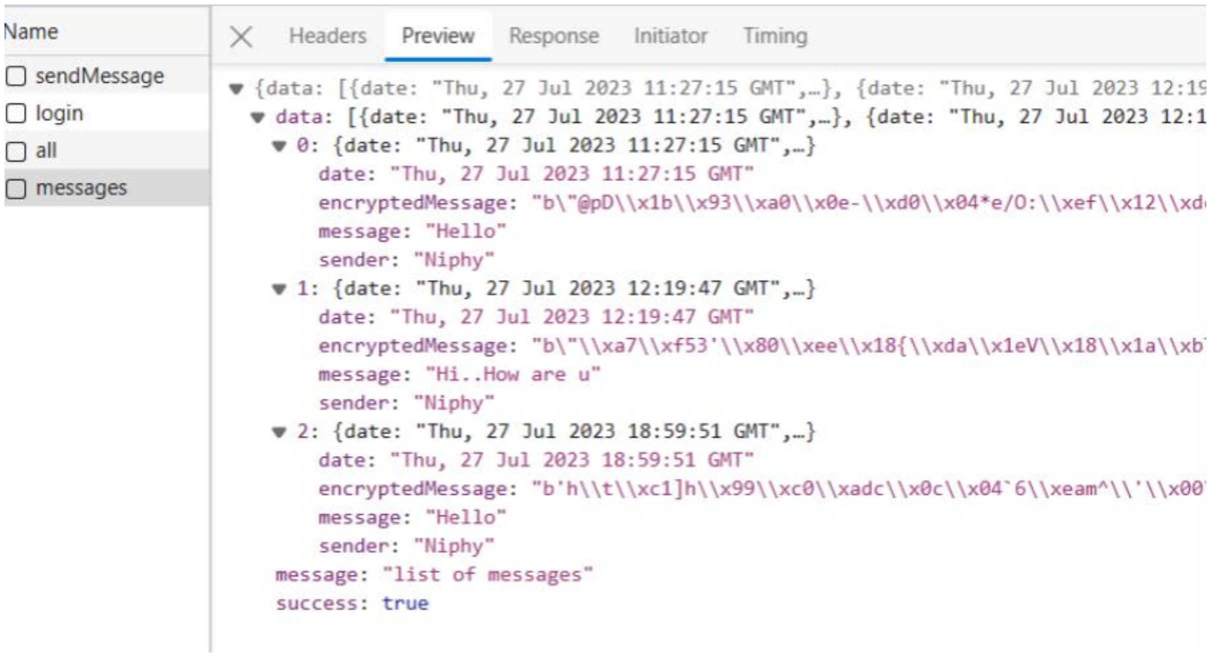
27-07-2023 1:19:47 PM

Niphy

Hello

27-07-2023 7:59:51 PM

Fig 6.1 : Receiving decrypted messages from backend.



```
▼ {data: [{date: "Thu, 27 Jul 2023 11:27:15 GMT",…}, {date: "Thu, 27 Jul 2023 12:19
  ▼ data: [{date: "Thu, 27 Jul 2023 11:27:15 GMT",…}, {date: "Thu, 27 Jul 2023 12:1
    ▼ 0: {date: "Thu, 27 Jul 2023 11:27:15 GMT",…}
        date: "Thu, 27 Jul 2023 11:27:15 GMT"
        encryptedMessage: "b\"@pD\\x1b\\x93\\xa0\\x0e-\\xd0\\x04*e/O:\\xef\\x12\\xd
        message: "Hello"
        sender: "Niphy"
    ▼ 1: {date: "Thu, 27 Jul 2023 12:19:47 GMT",…}
        date: "Thu, 27 Jul 2023 12:19:47 GMT"
        encryptedMessage: "b\"\\xa7\\xf53'\\x80\\xee\\x18{\\xda\\x1eV\\x18\\x1a\\xb
        message: "Hi..How are u"
        sender: "Niphy"
    ▼ 2: {date: "Thu, 27 Jul 2023 18:59:51 GMT",…}
        date: "Thu, 27 Jul 2023 18:59:51 GMT"
        encryptedMessage: "b'h\\t\\xc1]h\\x99\\xc0\\xadc\\x0c\\x04`6\\xeam^\\'\\x00
        message: "Hello"
        sender: "Niphy"
    message: "list of messages"
    success: true
```

Fig 6.2 : Receiving decrypted messages from backend.

6. SECURITY MEASURES

I. USER AUTHENTICATION

JSON Web Token (JWT) is a popular method for authentication and authorization in web applications. It allows for stateless authentication, meaning the server doesn't need to store session data, making it scalable and suitable for modern API-based applications. In Flask, you can implement JWT token authentication using various libraries. In our application, we are using **Flask-JWT-Extended** extension, which provides easy integration of JWT authentication. A JWT token will be generated, when a user login to the application.

II. BACKEND ENCRYPTION AND DECRYPTION

Encryption and decryption on the backend side involve the use of public and private keys. When users register in the system, a unique set of public and private keys is generated for each user. The public key is employed for encrypting data, while the private key is used for decryption. Subsequently, these key pairs are securely stored in the database. In this method, the backend system is responsible for both encrypting and decrypting the data.

III. FRONTEND ENCRYPTION & DECRYPTION

In frontend encryption and decryption, a single pair of public and private keys is utilized, and this key pair remains consistent for all users. These keys are stored in the **config.ts** file together with the source code. In this approach, the responsibility for both encrypting and decrypting data lies with the frontend system.

7. VULNERABILITES

I. SQL INJECTION

Users are the intruders attempting to access your app's valuable data, which is like a fortress that needs to be protected. Now, SQL injection resembles a cunning burglar who manages to sneak past your fortress walls unnoticed.

Users' inputs should be verified and treated appropriately when they interact with your app. In contrast, malicious content is inserted into the input fields using SQL injection, a sophisticated hacking technique. And guess what? Your app processes that garbage as if it were regular data, not realizing it was a trap. As a result, the hackers may tamper with your database, take information, corrupt your data, or otherwise cause mayhem.

Consider the possibility that your app might accidentally delete your priceless "users" table if someone entered their username as "; DROP TABLE users; --". SQL Injection. Users are the intruders attempting to access your app's valuable data, which is like a fortress that needs to be protected. Now, SQL injection resembles a cunning burglar who manages to sneak past your fortress walls unnoticed.

Users' inputs should be verified and treated appropriately when they interact with your app. In contrast, malicious content is inserted into the input fields using SQL injection, a sophisticated hacking technique. And guess what? Your app processes that garbage as if it were regular data, not realizing it was a trap. As a result, the hackers may tamper with your database, take information, corrupt your data, or otherwise cause mayhem.

Consider the possibility that your app might accidentally delete your priceless "users" table if someone entered their username as "; DROP TABLE users;--". The deletion of all user data is equivalent to a digital catastrophe!

II. ATTACK BY THE MAN-IN-THE-MIDDLE (MITM)

Consider how your app connects the user (using their web browser) and the server where it is stored, like a phone line. A MITM attack is comparable to a cunning eavesdropper listening in on their conversation.

Users typically have confidence that their data is secure as it is transferred between them and your app. The sly spy, however, places themselves in between the user and your server during a MITM attack. Without anyone knowing, they secretly intercept and tamper with the data they exchange. It's like an online spy listening in on private conversations.

As an illustration, when a user logs in, they send your server their credentials (username and password). However, the MITM hacker steals that information, reads it, and is then able to pose as the user and wreak havoc on their account.

You must step up the security of your app if you want to address these vulnerabilities. SQL injection can be avoided by using parameterized queries or prepared statements. And don't forget to use HTTPS (TLS) to encrypt the data flow between your users and the server. Treat user input as plain data, not some executable code. The information will be more difficult for the snoops to interpret as a result.

8. CONCLUSION

Our chat application shows our efforts in ensuring secure and private communication for users. The combination of Angular UI and Python for the backend reflects a thoughtful choice, offering a friendly interface and a solid server infrastructure. With the use of RSA algorithm, we have implemented security measures in our application. Encryption & Decryption is implemented in front-end as well as in back-end which gives us additional benefits to data security. Maintaining security of any application is a never-ending part.

9. GITHUB LINK

https://github.com/Niphy-10637063/Chatbook

10. REFERENCES

i. Educative: Interactive Courses for Software Developers. (n.d.). *What is the RSA algorithm?* [online] Available at: https://www.educative.io/answers/what-is-the-rsa-algorithm.

ii. flask-jwt-extended.readthedocs.io. (n.d.). *Flask-JWT-Extended's Documentation — flask-jwt-extended 4.4.4 documentation*. [online] Available at: https://flask-jwt-extended.readthedocs.io/en/stable/.

iii. npm. (2023). *jsencrypt*. [online] Available at: https://www.npmjs.com/package/jsencrypt [Accessed 27 Jul. 2023].

iv. www.w3schools.com. (n.d.). *json - Google Search*. [online] Available at: https://www.google.com/search?q=json&oq=json&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTINCAEQABiDARixAxiABDINCAIQABiDARixAxiABDIHCAMQABiABDINCAQQABiDARixAxiABDINCAUQABiDARixAxiABDIHCAYQABiABDIHCAcQABiABDIHCAgQABiABDIHCAkQABiABNIBCDM3NTlqMGo3qAIAsAIA&sourceid=chrome&ie=UTF-8 [Accessed 27 Jul. 2023].