

**REPORT**  
**ON**  
**INFORMATION AND NETWORK SECURITY MINI**  
**PROJECT**

*Submitted to*

**NMAM INSTITUTE OF TECHNOLOGY, NITTE**

*(Deemed to be University)*

*In partial fulfilment of the requirements for the award of the*

**Degree of Bachelor of Engineering**

**In**

**Information Science & Engineering**

*by*

**PRANATHI G R**

**USN NNM22IS111**

**Under the guidance of**

Dr. Jason Elroy Martis

Associate Professor

# CERTIFICATE

*This is to certify that the “INS Mini Project Report” submitted by Miss. Pranathi G R bearing USN NNM22IS111 of 6th semester B-Tech., a bonafide student of NMAM Institute of Technology, Nitte, fulfilling the partial requirements for the award of degree of Bachelor of Engineering in Information Science & Engineering at NMAM Institute of Technology, Nitte.*

---

**Name and Signature of Professor**

---

**Signature of HOD**

# Abstract

In the era of rapidly expanding digital communication, ensuring secure message exchange has become a fundamental necessity. The *Secure Message Capsule* project aims to safeguard sensitive communication by employing a hybrid encryption model that combines symmetric and asymmetric cryptographic techniques. AES-256 is used for encrypting the message content to ensure data confidentiality, while RSA-based digital signatures provide integrity and authenticity verification. A robust key management system underpins the encryption and decryption processes, leveraging password-derived AES keys with salt and IV for enhanced security. Additionally, the RSA key pair is generated once and used consistently for signing and verification. This system mitigates common security threats such as data tampering, interception, and unauthorized access by ensuring secure storage, transmission, and validation of message data. The report provides an in-depth overview of the system's architecture, the cryptographic workflow, implementation details, and the layered security approach adopted to handle encryption, digital signatures, and secure key handling. This project delivers a practical and scalable solution for secure digital communication.

# Introduction

In the current digital age, where cyber threats such as data interception, tampering, and unauthorized access are increasingly prevalent, securing sensitive information is of utmost importance. Traditional systems often lack a comprehensive security mechanism that ensures data protection during both storage and transmission. A modern security solution must integrate confidentiality, integrity, and authenticity to defend against such evolving threats.

The **Secure Message Capsule** project addresses these challenges by combining symmetric encryption (AES-256) for data confidentiality with asymmetric cryptography (RSA) for digital signatures and integrity verification. It provides a secure environment for message encryption, storage, and verification while utilizing password-based key derivation with dynamic salt and initialization vectors (IV) for enhanced protection. This ensures that sensitive messages are not only encrypted but also authenticated using a digital signature, allowing verification of the sender and the content's integrity.

The system is designed to enable secure message handling by encrypting the message content, digitally signing it, and verifying the signature before decryption. This layered approach ensures that any tampering or unauthorized access attempts are effectively detected and mitigated. By incorporating modern cryptographic practices, the Secure Message Capsule offers a practical, reliable, and scalable solution for secure digital communication.

# Problem Statement

Sensitive messages are often vulnerable to unauthorized access and tampering due to weak encryption and lack of authentication. The Secure Message Capsule project addresses this by implementing AES encryption for secure storage, RSA-based digital signatures for integrity, and password-derived keys with salt and IV to ensure confidentiality, authenticity, and protection during transmission.

## Objectives

The primary objectives of this project are:

1. To implement AES-256 encryption with password-derived keys for secure message storage.
2. To ensure message integrity and authenticity using RSA digital signatures.
3. To securely transmit encrypted data between sender and receiver using HTTPS.
4. To prevent unauthorized access and tampering by enforcing one-time decryption and secure key derivation.
5. To verify the authenticity of messages before decryption through digital signature validation.

# Methodology

The project follows a secure client-server architecture where Alice (the sender) encrypts and signs the message, while Bob (the receiver) decrypts and verifies it. The system ensures confidentiality, integrity, and authenticity through encryption, digital signatures, and secure transmission.

---

## 1. Environment Setup

- **Node.js & Express.js:** For building the REST API.
  - **Crypto module:** Node.js module for AES encryption and RSA signature generation/verification.
  - **fs module:** For reading RSA key files.
  - **Both Postman and Frontend UI:** For user interaction with the endpoints.
  - **HTTPS:** Enabled via TLS 1.2 to secure message transmission.
- 

## 2. Message Encryption and Storage (Sender - Alice)

- Alice enters a message and a shared secret password on the Encrypt Page.
  - AES-256-GCM is used for encrypting the message using a key derived from the password via PBKDF2.
  - A digital signature is generated using Alice's **RSA private key** to ensure authenticity.
  - The encrypted message, IV, salt, auth tag, and digital signature are stored in a JSON file on the server.
  - The storage is **one-time only** — only one message can exist at a time.
- 

## 3. Message Decryption and Verification (Receiver - Bob)

- Bob opens the Decrypt Page and enters the shared password.

- The AES key is derived from the password and used to decrypt the message.
  - The digital signature is verified using Alice's **RSA public key**.
  - If the signature is valid, the decrypted message is displayed.
  - The stored message is immediately deleted after successful decryption, enforcing one-time access.
- 

#### 4. Digital Signature Validation

- Ensures that the decrypted message was indeed sent by Alice and not altered.
  - Signature is generated during encryption using RSA private key.
  - Signature is verified during decryption using RSA public key.
  - If verification fails, the message is **not decrypted**, preventing tampered or forged content.
- 

#### 5. Secure Transmission

- All communication between the client (Alice/Bob) and the server is secured using **HTTPS (TLS 1.2)**.
  - This protects the shared password, encrypted message, and signature from interception or tampering during transmission.
- 

#### 6. Workflow

This workflow outlines a one-way encrypted message system where Alice sends a secure message to Bob using AES encryption and RSA signatures.

##### **Shared Secret**

- Alice and Bob share a secret password beforehand.
- This password is used to derive the encryption key.

##### **Message Encryption (Alice's Side)**

1. Alice enters her message and password on the Encrypt Page.
2. The frontend sends this data to the backend via the /send API.

### **Encryption Process:**

- The system derives an AES key from the password using PBKDF2.
  - The message is encrypted using AES-256-GCM.
  - The encrypted message is signed with Alice's RSA private key.
  - The server stores the encrypted data, initialization vector (IV), salt, tag, and signature in a JSON object.
3. The message is securely stored on the server.
  4. Only one message is allowed to exist at any time.

### **Message Decryption (Bob's Side)**

1. Bob accesses the Decrypt Page and enters the shared password.
2. The frontend sends the password to the backend via the /receive API.

### **Decryption Process:**

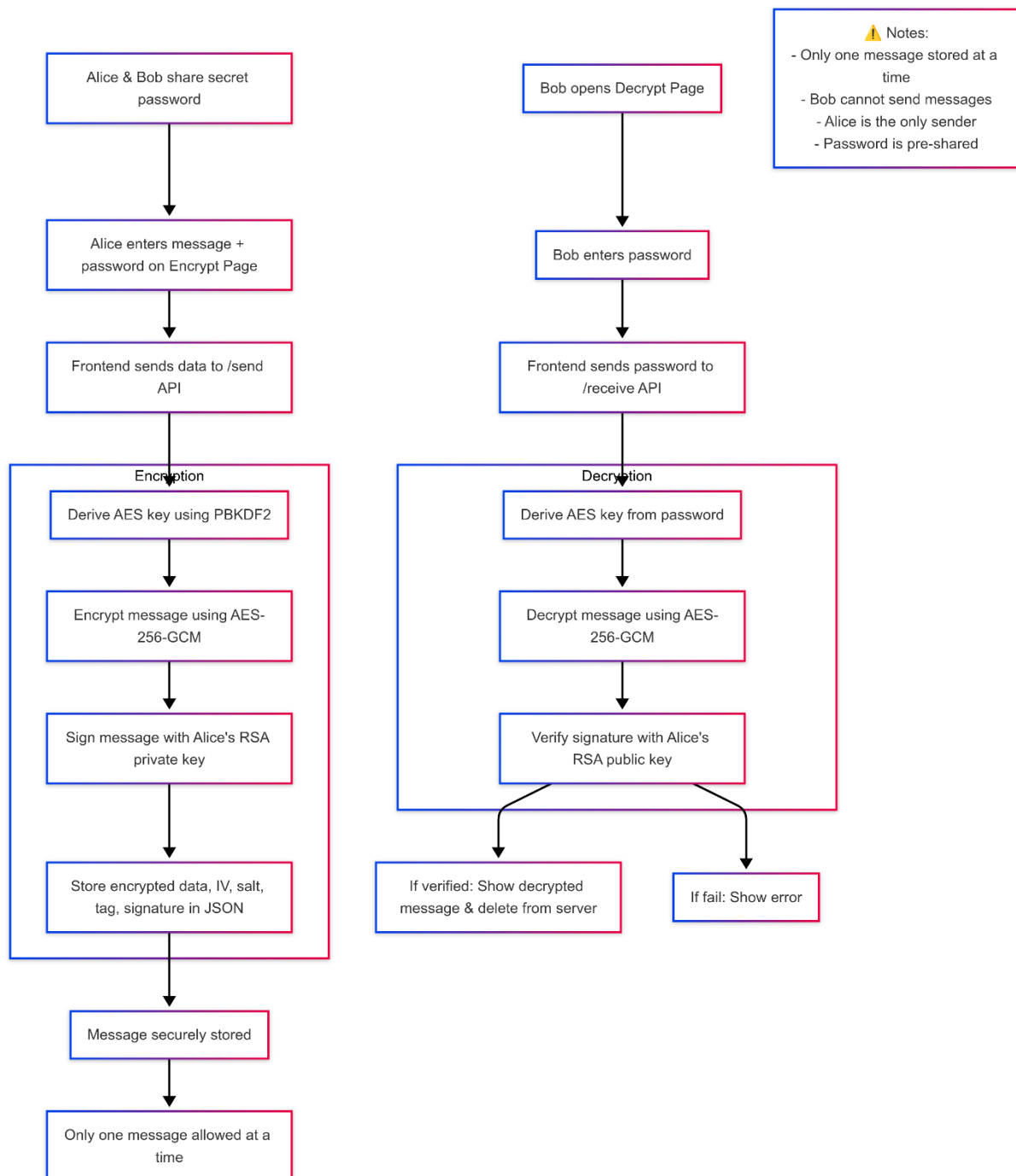
- The AES key is derived from the password.
  - The message is decrypted using AES-256-GCM.
  - The signature is verified using Alice's RSA public key.
3. If the verification succeeds:
    - The decrypted message is shown to Bob.
    - The message is deleted from the server after being read.
  4. If the verification fails:
    - An error is displayed.

### **Additional Notes**

- Only one message is stored at a time.
- Bob cannot send messages.
- Alice is the only sender.

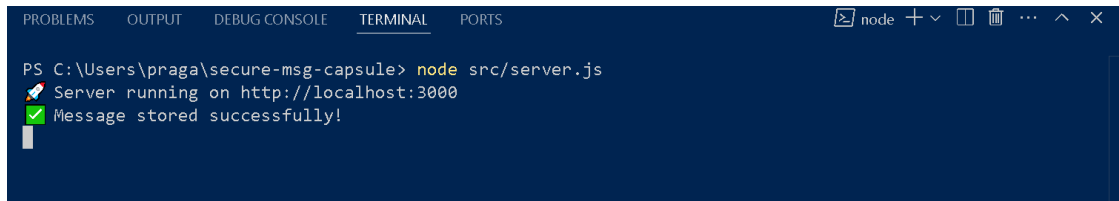


- The password is pre-shared and not transmitted openly.



**fig 1: flowchart of entire workflow**

# Results



```
PS C:\Users\praga\secure-msg-capsule> node src/server.js
Server running on http://localhost:3000
Message stored successfully!
```

Fig 2: Initialization of project

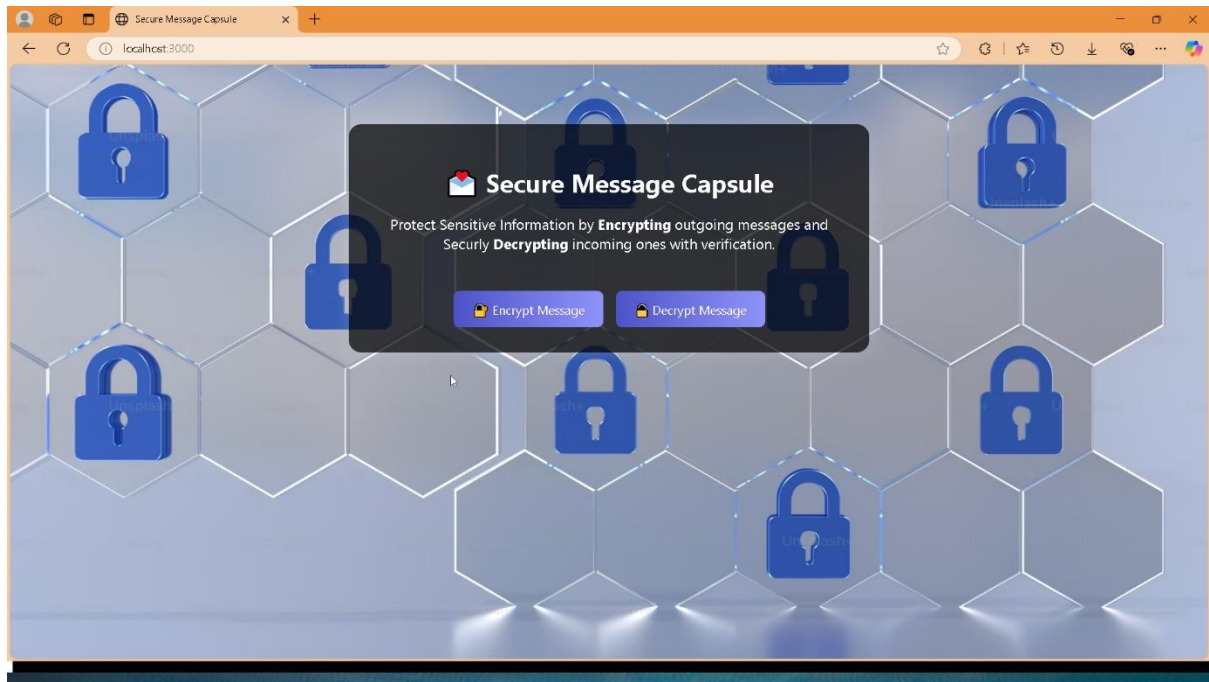
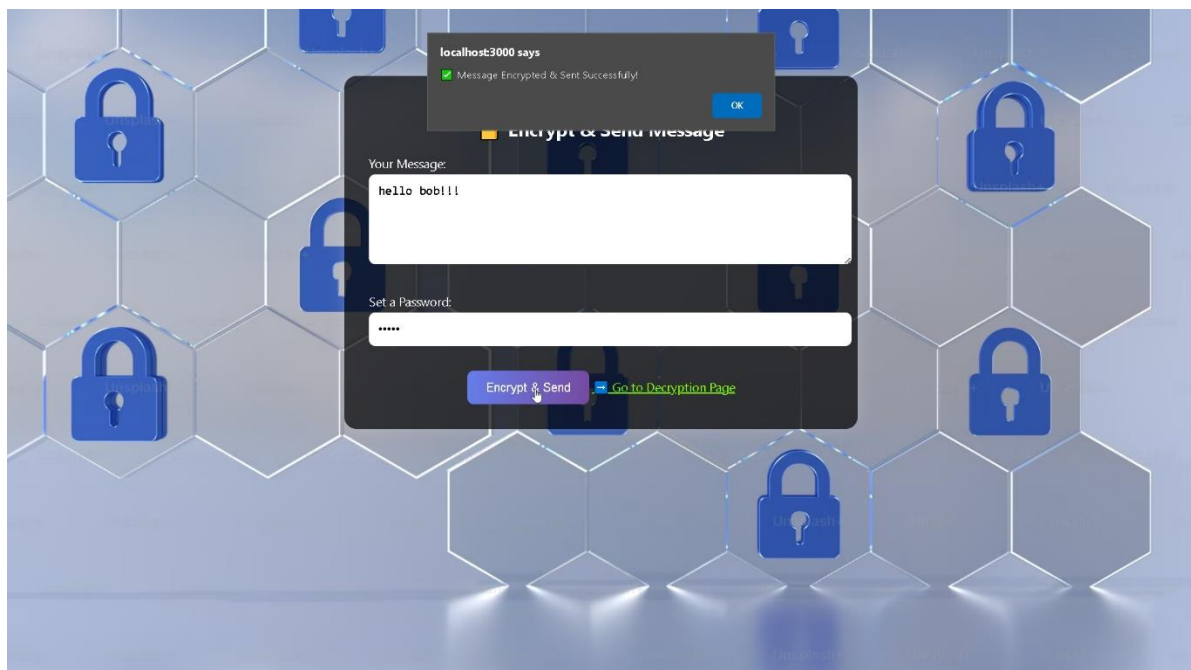
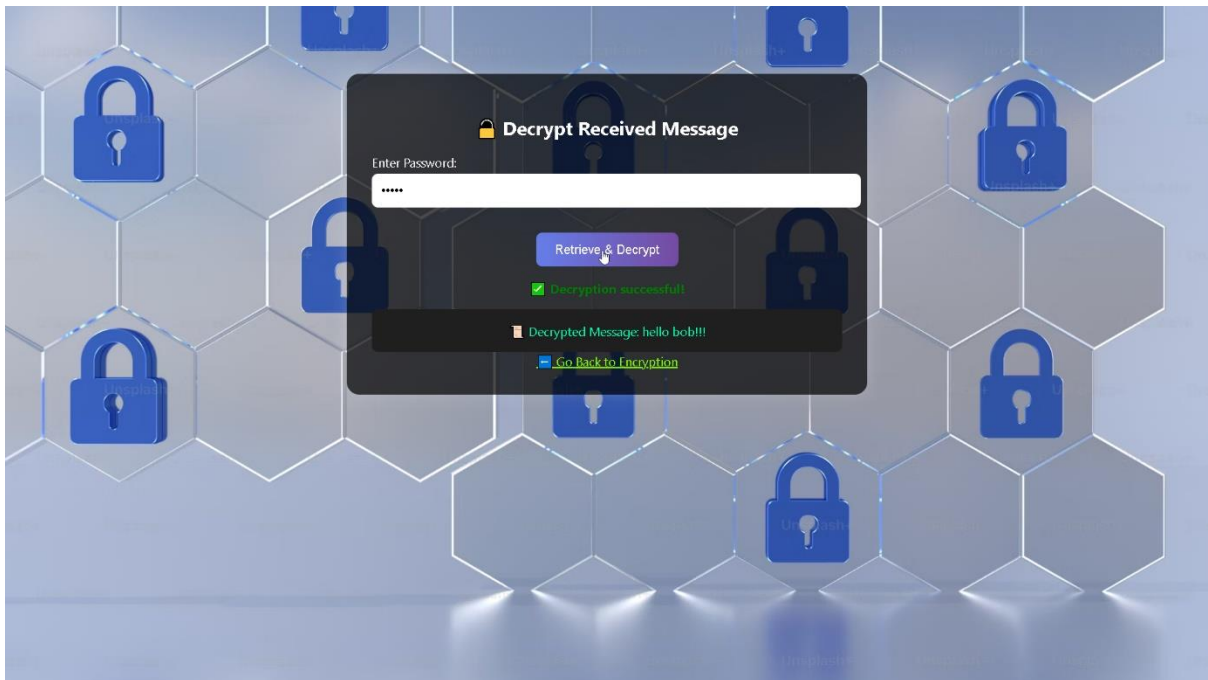


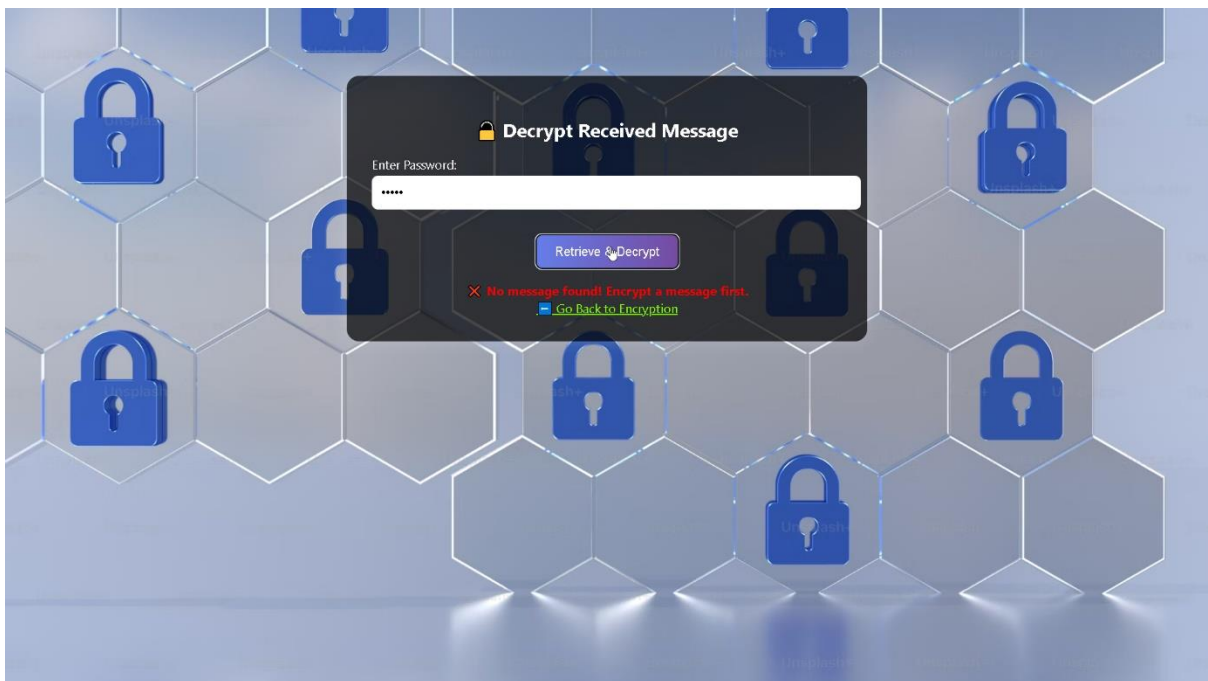
Fig 3: User Interface



**Fig 4: User Interface for Alice to send message securely.**

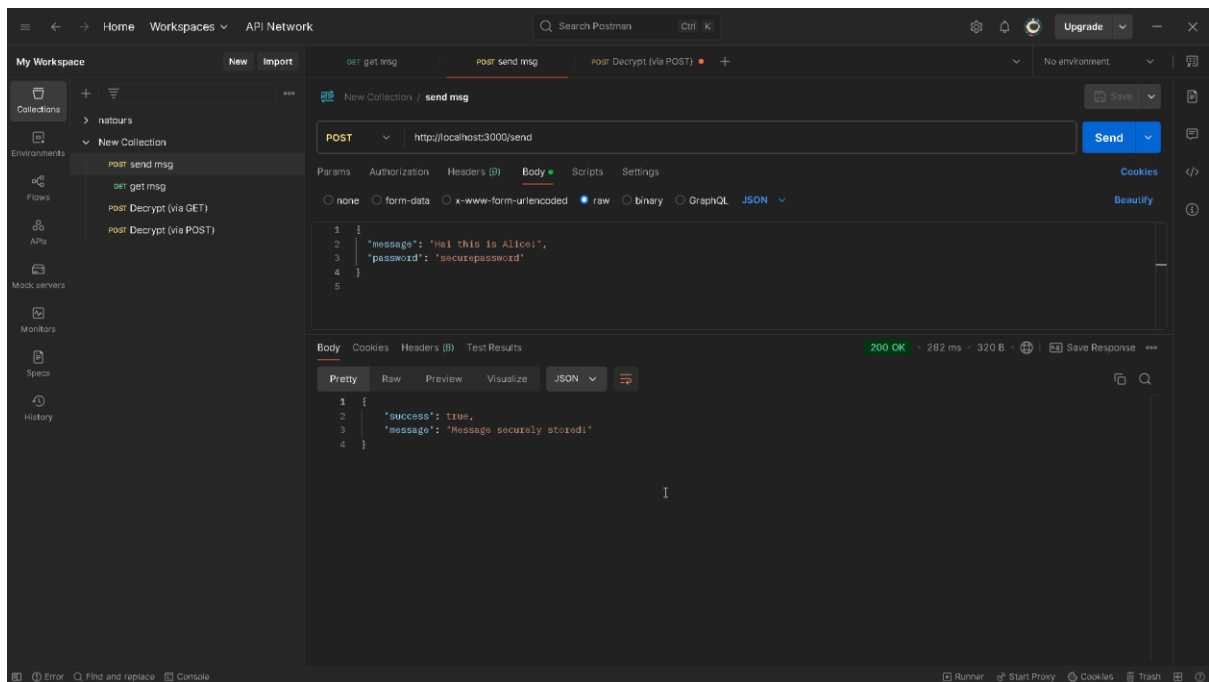


**Fig 5: User Interface where bob can decrypt the message**

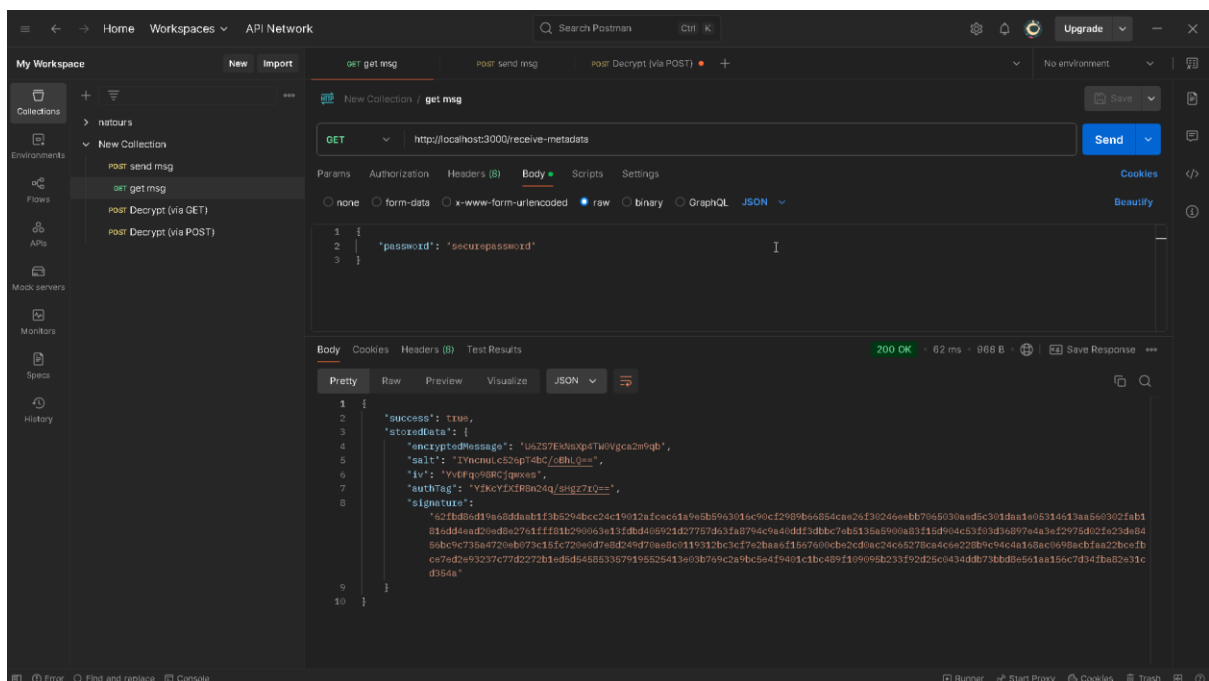


**Fig 6: Bob cannot decrypt the message again he can read the message only once to prevent reply attacks.**

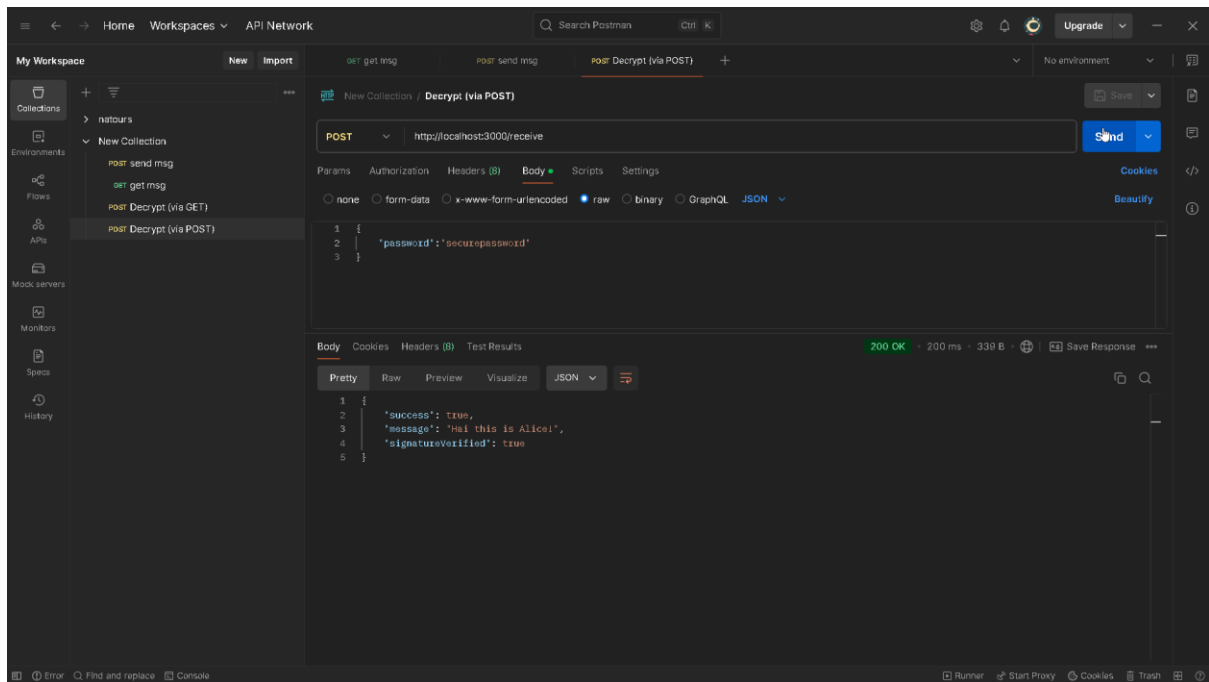
## Verification using Postman:



**FIG 7:** The image shows a successful POST request in Postman storing an encrypted message securely using the /send endpoint of the Secure Message Capsule system.



**Fig 8:** The image shows a successful GET request in Postman retrieving encrypted message metadata, including salt, IV, authTag, and signature, from the /receive-metadata endpoint.



**Fig 9:** The image shows a successful POST request in Postman to /receive that decrypts the message and verifies the digital signature, confirming both confidentiality and authenticity.

## LINKS

**Github :** <https://github.com/pranathigr11/inslab.git>

**Recording :** [ins\\_project.mp4](#)

# Conclusion

The Secure Message Capsule project successfully delivers a lightweight yet robust framework for secure message exchange between a trusted sender and receiver. By integrating **AES-256-GCM encryption** for confidentiality, **RSA digital signatures** for authenticity, and **HTTPS (TLS)** for secure transmission, the system effectively protects sensitive data against unauthorized access, tampering, and interception.

The project leverages the built-in **Node.js crypto module** for secure key derivation, encryption, and digital signature handling, ensuring end-to-end protection during both storage and transmission. The use of PBKDF2 for AES key derivation from a pre-shared password further enhances resistance against brute-force attacks.

The design enforces strict message integrity through signature verification and enables **one-time message retrieval**, eliminating data persistence risks. The Express.js-based API supports a clear separation of encryption, storage, and decryption logic, making the system modular and scalable for real-world secure communication scenarios.

Future enhancements could include implementing a secure database for structured storage, adding support for multiple users or messages, automating key pair generation and management, and introducing role-based access control to expand the system's capabilities while maintaining high security standards.

