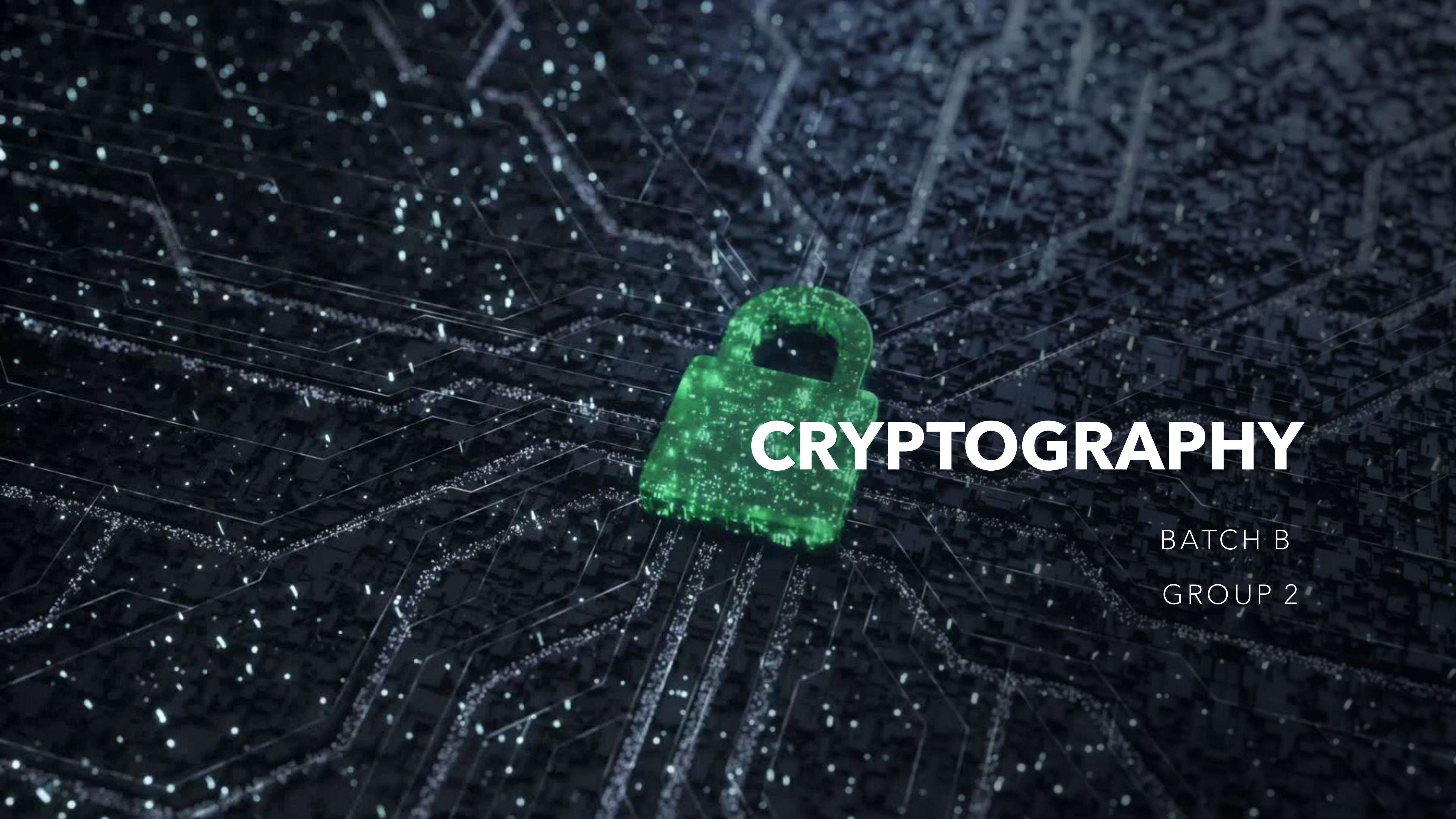


CRYPTOGRAPHY

BATCH B

GROUP 2

TEAM - MEMBERS

1.JAIDEV . K – CB.EN.U4AIE21117

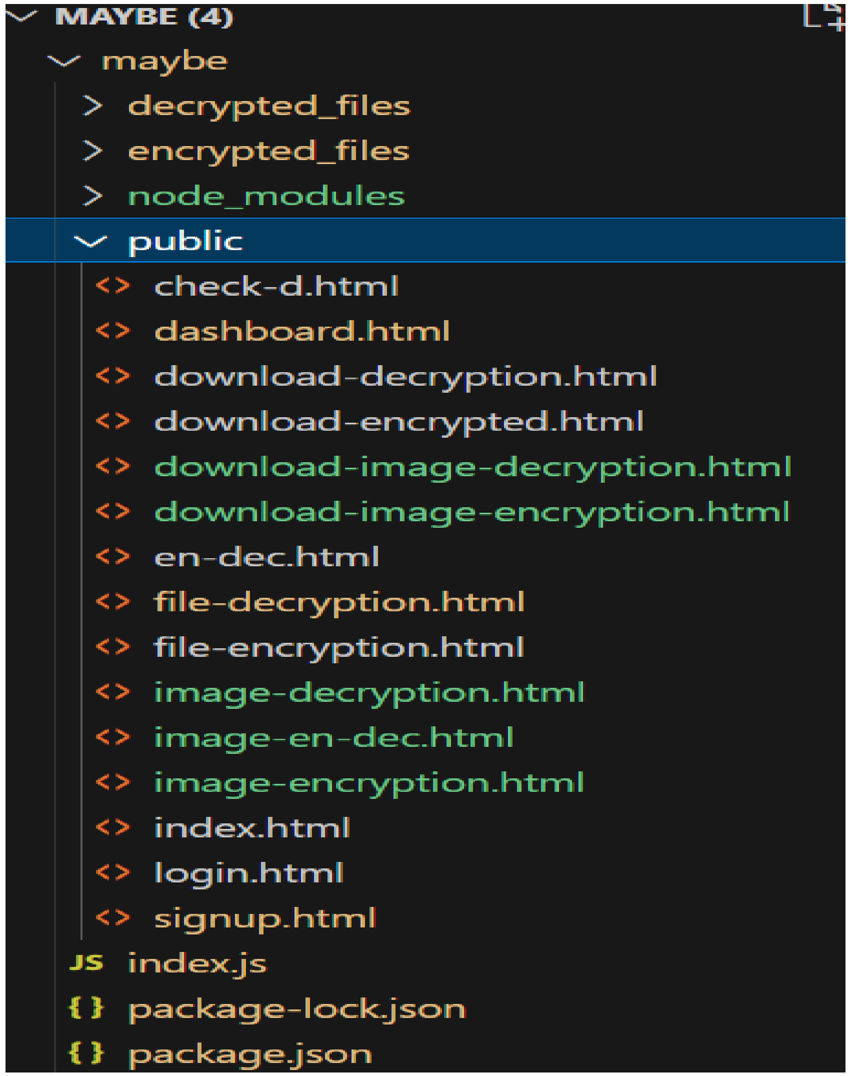
2. SAI CHANDANA . J – CB.EN.U4AIE21118

3.PRANISH KUMAR. M – CB.EN.U4AIE21137

4. CHARISHMA CHOWDARY .T - CB.EN.U4AIE21169

User Based File & image Encryption/Decryption

FILES IN OUR PROJECT

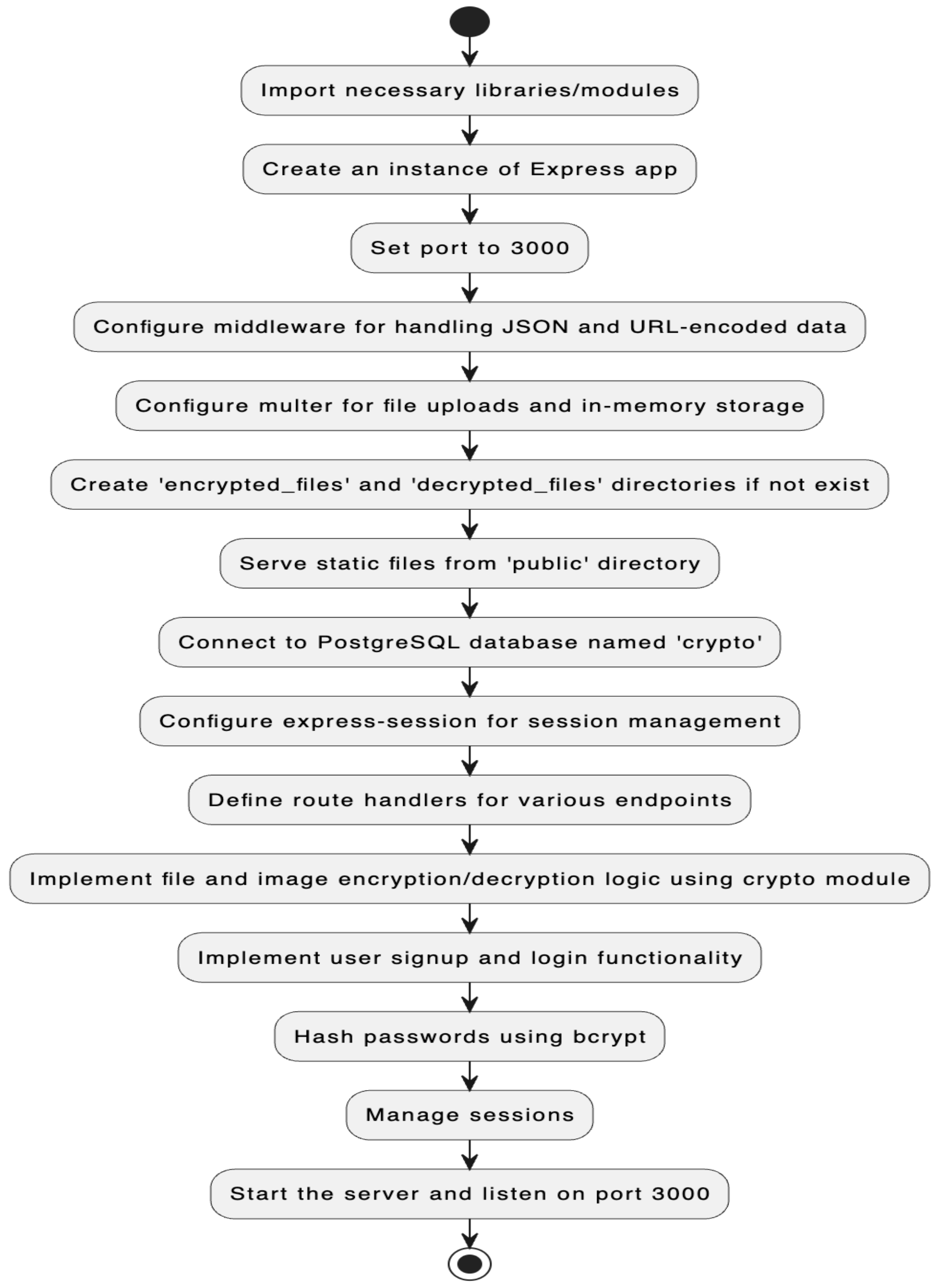


```
MAYBE (4)
  maybe
    > decrypted_files
    > encrypted_files
    > node_modules
    public
      check-d.html
      dashboard.html
      download-decryption.html
      download-encrypted.html
      download-image-decryption.html
      download-image-encryption.html
      en-dec.html
      file-decryption.html
      file-encryption.html
      image-decryption.html
      image-en-dec.html
      image-encryption.html
      index.html
      login.html
      signup.html
    index.js
    package-lock.json
    package.json
```

INTRODUCTION

- The project at hand is a focused endeavor to create a robust File and Image Encryption/Decryption Web Application. The primary goals are to empower users with secure encryption and decryption capabilities for their files and images while ensuring a personalized experience through effective user authentication. Leveraging technologies such as Node.js, Express, PostgreSQL, Multer, Crypto, and Bcrypt, the application combines a user-friendly interface with advanced security measures. The significance lies in addressing the contemporary need for safeguarding sensitive information, where encryption plays a pivotal role in maintaining confidentiality during data transit. Overall, this project aims to strike a balance between accessibility and robust data security, providing users with a seamless and protected digital experience.

BLOCK DIAGRAM


Import necessary libraries/modules
Create an instance of Express app
Set port to 3000
Configure middleware for handling JSON and URL-encoded data
Configure multer for file uploads and in-memory storage
Create 'encrypted_files' and 'decrypted_files' directories if not exist
Serve static files from 'public' directory
Connect to PostgreSQL database named 'crypto'
Configure express-session for session management
Define route handlers for various endpoints
Implement file and image encryption/decryption logic using crypto module
Implement user signup and login functionality
Hash passwords using bcrypt
Manage sessions
Start the server and listen on port 3000

TECHNOLOGIES USED

1. Node.js::Utilized as the runtime environment for server-side scripting.
2. Express::Acts as the web application framework, ensuring streamlined development.
3. Multer:Efficiently handles file uploads, a crucial aspect of the application's functionality.
4. bcrypt:Implements secure password hashing for robust user authentication.
5. PostgreSQL:Chosen as the relational database to provide reliable and scalable data storage.
6. Crypto:Employs the Crypto library for advanced encryption and decryption processes, ensuring data security.

Encryption Workflow: Steps and Security Measures

User Input: User initiates the encryption process by providing a file and a secure encryption key or password.

File Upload: The selected file is uploaded to the server using Multer, ensuring a smooth and secure transfer.

Key Generation: A cryptographic key is generated, typically using the scrypt function, from the user-provided password. This adds a layer of security to the encryption process.

Initialization Vector (IV) Creation: A random Initialization Vector (IV) is generated. The IV, along with the key, is crucial for the security of the encryption algorithm.

Encryption Algorithm (AES-256-CFB): The file content is encrypted using a robust algorithm, such as AES-256-CFB, with the generated key and IV. This ensures a high level of data security.

Contd…

Concatenation: The IV is concatenated with the encrypted content to create the final encrypted data package.

Storage: The encrypted data is stored securely, ensuring that unauthorized access is prevented.

File Naming and Saving: The encrypted file is saved with an appropriate name in the designated encrypted_files directory.

Response to User: A response is sent to the user, confirming the successful encryption and providing a download link to retrieve the encrypted file.

Logging: Log entries are made to record the encryption event, capturing relevant details for auditing and debugging purposes.

Decryption Workflow: Steps and Security Measures

User Input:User initiates the decryption process by uploading the encrypted file and providing the decryption key or password.

File Upload:The encrypted file is securely uploaded to the server.

Key Generation:A cryptographic key is generated using the scrypt function from the user-provided decryption password.

Initialization Vector (IV) Extraction:The IV is extracted from the beginning of the encrypted file. It is a critical component for the decryption process.

Decryption Algorithm (AES-256-CFB): The file content is decrypted using the AES-256-CFB algorithm, utilizing the generated key and extracted IV.

Contd...

Content Extraction: The decrypted content is extracted from the decryption process.

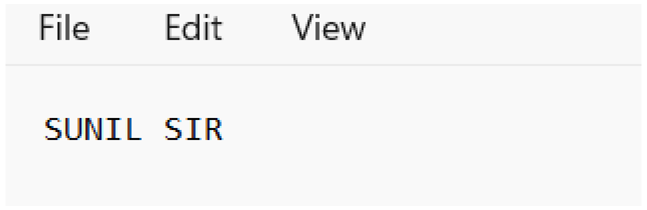
File Saving: The decrypted content is saved as a new file in the designated decrypted_files directory.

Response to User: A response is sent to the user, confirming the successful decryption and providing a link to download the decrypted file.

Logging: Log entries are made to record the decryption event, capturing relevant details for auditing and debugging.

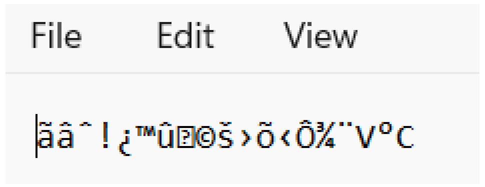
Security Measures: Emphasize the secure generation and handling of cryptographic keys and IVs.

RESULTS : FILE ENCRYPTION AND DECRYPTION



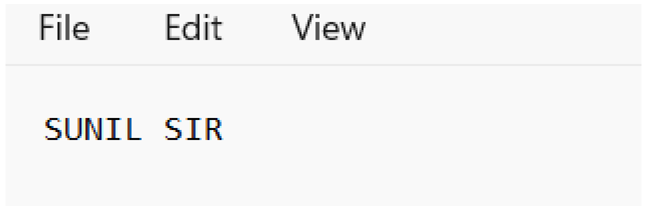
```
SUNIL SIR
```

Our text file



```
ãâˆ!¿™û⍰©š›õ‹Ô¾¨V°C
```

Our encrypted file

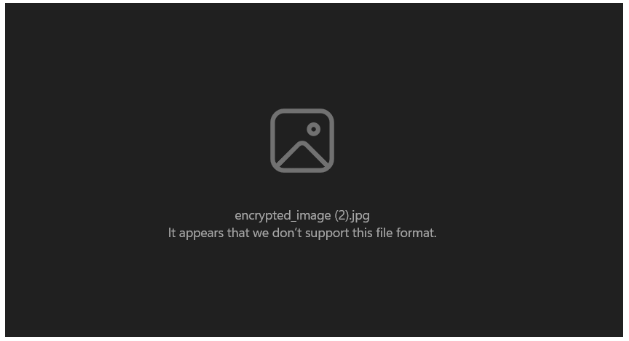


```
SUNIL SIR
```

Our decrypted file

RESULTS : IMAGE ENCRYPTION AND DECRYPTION

Input image


encrypted_image (2).jpg
It appears that we don't support this file format.

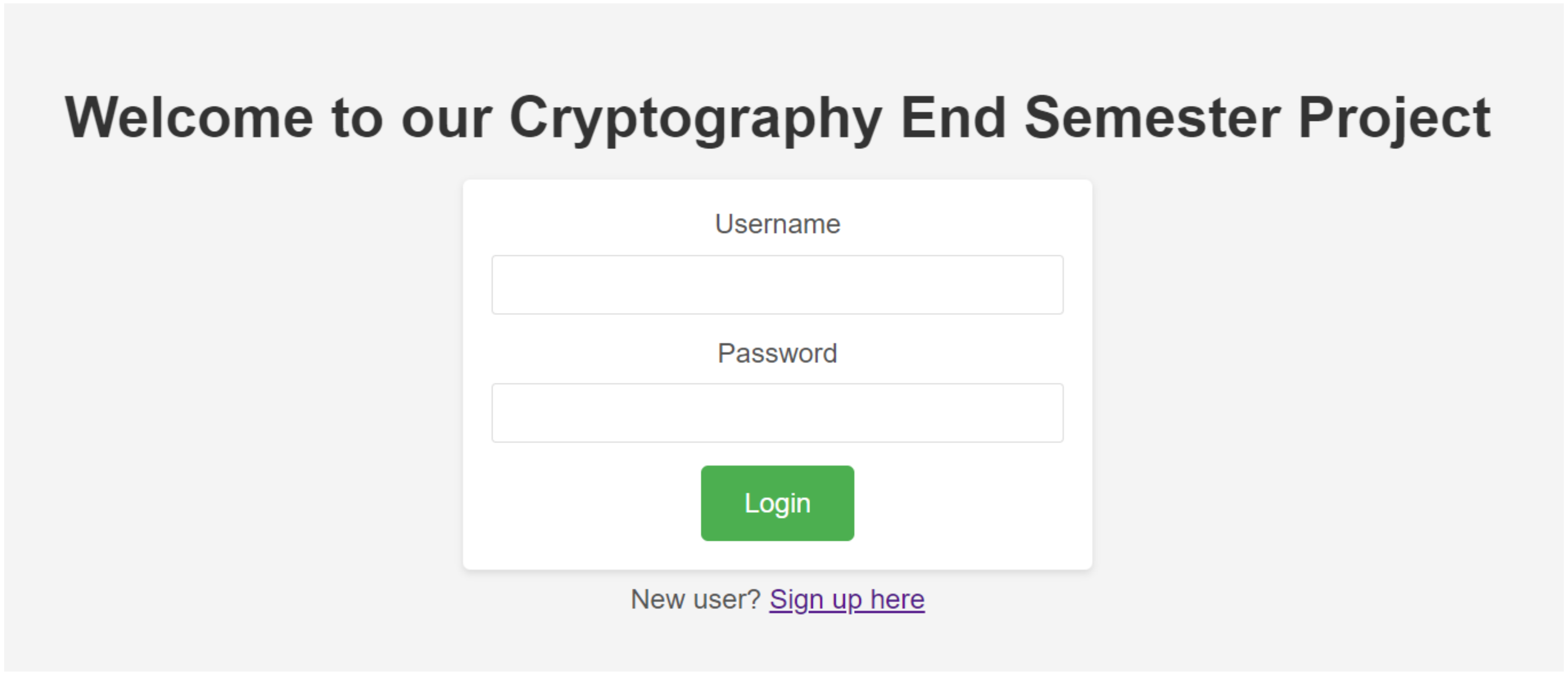
Encrypted image

Decrypted image

User Authentication

- User Input: Users provide their credentials (username and password) for authentication.
- Database Query: A query is made to the PostgreSQL database to retrieve the stored hashed password associated with the provided username.
- Bcrypt Comparison: Bcrypt is used to compare the hashed version of the entered password with the stored hashed password in the database.
- Authentication Result: If the comparison is successful, indicating a match, the user is authenticated.
- Session Management: A session is established, commonly using Express sessions, to persist the user's authentication status.
- Secure Access: Authenticated users gain access to specific functionalities and secure areas of the application.
- Security Measures: Emphasize the use of bcrypt for secure password hashing, preventing unauthorized access even if the database is compromised.

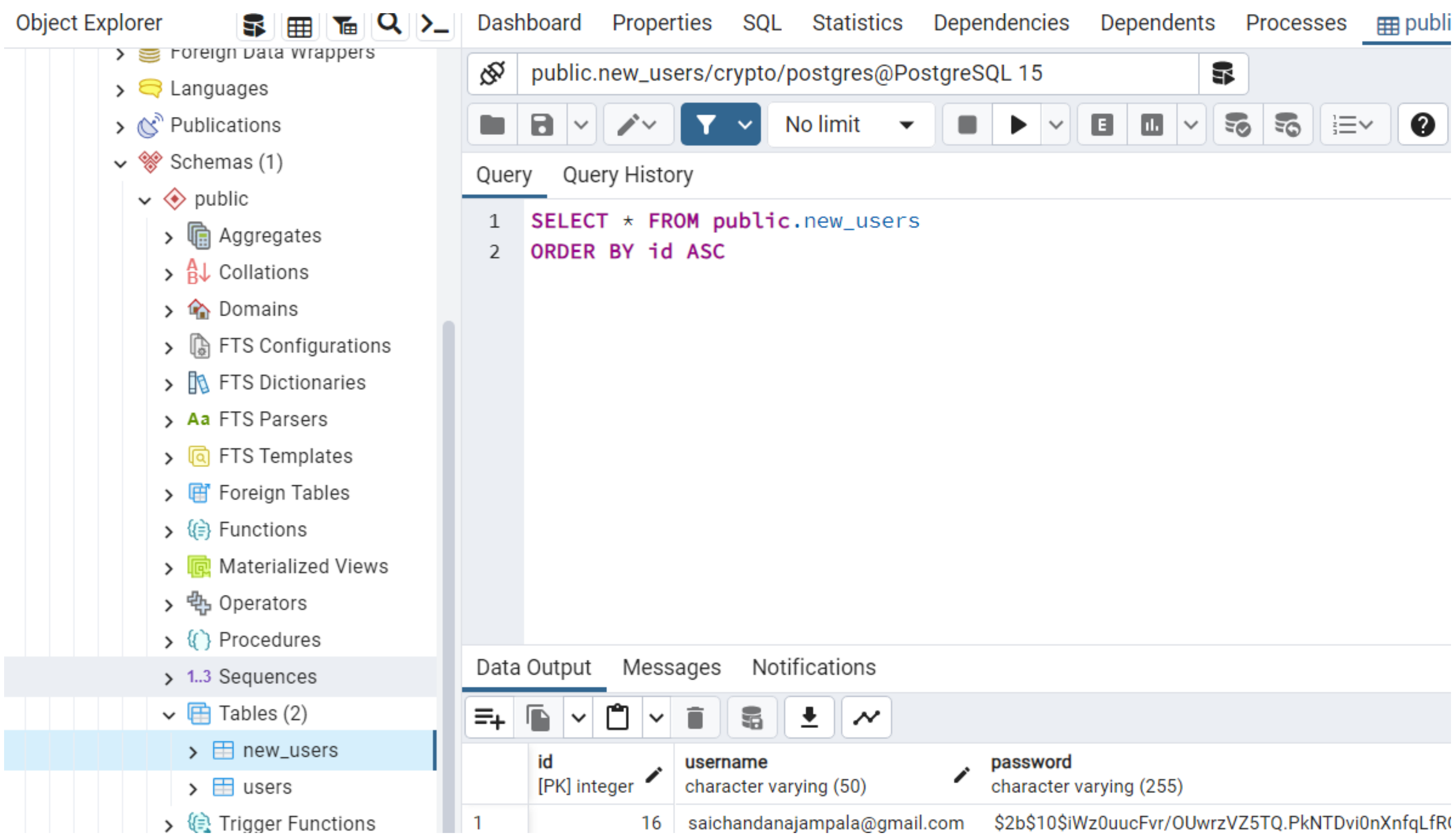
User Authentication


Welcome to our Cryptography End Semester Project

Database Integration:

- User Table:Contains user information, including username and securely hashed passwords.Used for user authentication during login.
- File Metadata Table:May include information such as file names, types, and upload timestamps.Supports efficient file management and retrieval.
- Encryption History Table:Records encryption events, including file names, timestamps, and user IDs.Aids in auditing and tracking encrypted files.
- Database Queries:Utilizes SQL queries to interact with the database for user authentication, file management, and logging purposes.Ensures secure and structured data access.
- Connection Pooling:Implements connection pooling to efficiently manage database connections and optimize performance.

- POSTGRE -SQL



public.new_users/crypto/postgres@PostgreSQL 15

```
SELECT * FROM public.new_users
ORDER BY id ASC
```

	id [PK] integer	username character varying (50)	password character varying (255)
1	16	saichandanajampala@gmail.com	$2b$10$iWz0uucFvr/OUwrzVZ5TQ.PkNTDvi0nXnfqLfR

CHANGES WE MADE: AFTER THE PRESENTAION

1. EXPLAINING ABOUT THE BYCRYPT FUNCTION
2. MODIFYING THE WEBSITE TO TAKE ANY FILE AND DECRYPT AND ECNCRYPT

Bcrypt Function

Hashing :

When a user creates or updates their password, the bcrypt function hashes the password. Hashing is a one-way process that transforms the password into a fixed-length string of characters.

Salting:

bcrypt uses a technique called salting to enhance security. A unique salt is generated for each password. This salt is then combined with the user's password before hashing. Salting is crucial because it prevents attackers from using precomputed tables (rainbow tables) to quickly look up the hash value for a given password.

Work Factor (Cost Parameter):

bcrypt allows the specification of a cost parameter (also known as the work factor). This parameter determines the computational cost of the hashing algorithm. A higher cost makes it more computationally intensive, thereby making brute-force and rainbow table attacks more difficult.

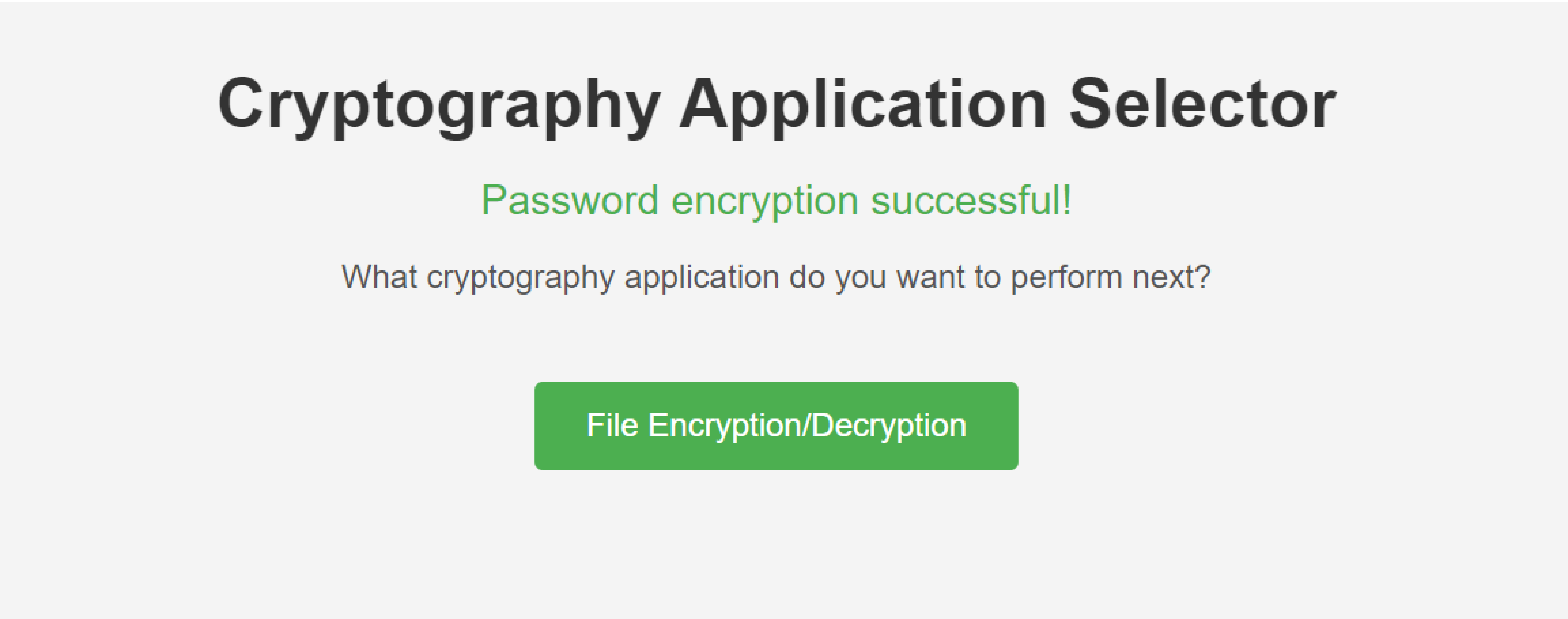
Storage:

The hashed password, along with the salt and cost parameter, is then stored in the database. The typical format is a string t hat includes the hash, salt, and cost factor. For example, a bcrypt hash might look like this:

$2a$10$SALTSTRINGHASHSTRING

- “$2a$” indicates the use of the bcrypt algorithm.
- “10” is the cost factor.
- “SALTSTRING” is the salt.

MODIFIED TO TAKE ANY FILE AND ENCRYPT AND DECRYPT



Cryptography Application Selector

Password encryption successful!

What cryptography application do you want to perform next?

Future Enhancements:

- Algorithm Flexibility:

  Introduce support for different encryption algorithms, providing users with options based on their security requirements.

- Improved UI/UX:

  Enhance the user interface for a more intuitive and visually appealing experience.

  Streamline user interactions and navigation.

- Additional File Types:

  Expand file handling capabilities to support a broader range of file types.

  Cater to diverse user needs for encryption and decryption.

- Multi-User Collaboration:

  Implement features that enable secure collaboration between multiple users, allowing shared access to encrypted content.

Conclusion:

- In conclusion, this File and Image Encryption/Decryption Web Application combines cutting-edge technologies with robust security measures. From secure password handling and encryption to database integration, the application ensures the confidentiality and integrity of user data. As we look to the future, potential enhancements aim to provide users with greater flexibility, an improved user experience, and expanded functionality. This project stands as a testament to the commitment to both user-friendliness and stringent security standards in the realm of file and image handling.

Thank you :)