Queen's Gambit

Where your data plays by its own rules

Project Synopsis Submitted

to

MANIPAL ACADEMY OF HIGHER EDUCATION

For Partial Fulfillment of the Requirement for the

Award of the Degree

Of

Bachelor of Technology

in

Information Technology

by

Aniruddh Ballal

Reg. No. 220911256

Under the guidance of

Mr. Akshay K.C
Assistant Professor - Senior Scale
Department of I&CT
Manipal Institute of Technology
Manipal, Karnataka, India

Dr. Raviraj Holla
Assistant Professor – Senior Scale
Department of I&CT
Manipal Institute of Technology
Manipal, Karnataka, India

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
A Constituent Unit of MAHE, Manipal

September 2024

1. Objective

The primary objective of *Queen's Gambit* is to create a secure and innovative system for data encoding, encryption, and transmission by combining the strategic elements of chess with conventional cryptography. This project explores an unconventional approach to data security, wherein data is encoded into chess moves, represented in PGN format, which makes it challenging for unauthorized parties to interpret. By leveraging the structure of chess, the application hides encrypted information within familiar gameplay formats, introducing a level of cryptographic novelty and complexity beyond traditional file encryption.

In addition to the unique encoding method, the project aims to provide a secure, user-friendly platform with robust layers of encryption and controlled access. Advanced encryption techniques like RSA and AES are integrated to create multi-layer security around user data, while OTP-based user authentication provides an extra level of user verification. These measures collectively address common security concerns, such as unauthorized access, data tampering, and pattern detection, making *Queen's Gambit* a viable solution for modern data security needs.

Furthermore, the application aspires to serve as a proof of concept for new cryptographic approaches that blend steganography, data obfuscation, and encryption. The platform is designed not only to protect data but also to showcase how creative elements (like a game) can enhance encryption methods. This objective holds potential for practical applications in areas requiring high confidentiality.

Key Points:

- **Unique Data Encoding**: Encodes data as chess moves in PGN format, making encrypted data look like ordinary chess games.
- **Advanced Security Layers**: Integrates RSA (for key encryption) and AES (for data encryption) to secure files, ensuring data integrity.
- **High Confidentiality and Accessibility**: Balances ease of use with a high level of security, suitable for users with diverse technical backgrounds.
- **Innovative Use of Chess for Obfuscation**: Utilizes the complexity of chess for obfuscating data, making the encryption method more difficult to detect and reverse-engineer.
- **Proof of Concept for New Cryptography**: Demonstrates potential for combining creative elements with cryptographic techniques, expanding the traditional approach to data security.

2. Problem Statement

Traditional encryption methods, while reliable, often exhibit identifiable patterns that can be detected and exploited by sophisticated cyber attackers. These patterns may be observed in the metadata, file headers, or structural regularities within the encrypted data. As cyber threats continue to advance, there is an increasing need for encryption techniques that do not follow conventional formats and are inherently more resistant to detection. The challenge lies in creating an encryption method that not only secures the data but also disguises its encrypted nature effectively.

Queen's Gambit addresses this challenge by encoding encrypted data into sequences of chess moves stored in PGN files, an approach that makes the encrypted content look like ordinary chess game records. This unique method reduces the likelihood of successful attacks by blending cryptography with steganography, making it difficult for attackers to identify the data as encrypted in the first place. The integration of AES and RSA encryption adds further layers of security, ensuring that even if the data is detected, it cannot be easily decrypted without the appropriate keys.

The problem is further compounded by the increasing frequency and sophistication of cyberattacks, such as brute force attempts and pattern recognition techniques used in steganalysis. By adopting a multi-layered approach that combines standard encryption with innovative encoding strategies, this project aims to create a more robust solution for data protection that can withstand modern cybersecurity threats.

Key Points:

- **Limitations of Traditional Encryption**:
    - Identifiable patterns and metadata in standard encrypted files can make them vulnerable to attacks.
    - Sophisticated cyberattacks, including brute force and steganalysis, exploit these patterns to decrypt data.
- **Innovative Solution with Chess-Based Encoding**:
    - Encodes encrypted data as chess moves in PGN files, disguising the data and reducing the risk of detection.
    - Blends cryptographic methods with steganography to enhance security and make the encryption method less predictable.
- **Mitigating Modern Cyber Threats**:
    - Addresses advanced threats by combining AES and RSA encryption with an unconventional approach to data encoding.
    - Offers a unique solution to enhance data security and reduce vulnerability to cyberattacks.

3. Need for the Application

In the current digital era, securing sensitive data is more crucial than ever. Traditional encryption methods, while effective, are increasingly vulnerable to sophisticated attacks, such as brute force, pattern recognition, and steganalysis. As cyber threats evolve, so must the mechanisms we use to protect information. *Queen's Gambit* addresses this need by introducing a novel encryption approach that combines the strategic complexity of chess with robust cryptographic techniques like AES and RSA encryption.

The unique aspect of encoding data as chess moves in PGN files makes it difficult for unauthorized users to identify and decipher the hidden information. Unlike standard encrypted files, which often have identifiable patterns or metadata indicating their encrypted nature, PGN files appear as ordinary chess game records. This form of data obfuscation adds an additional layer of security by

making the encrypted content look harmless and unrelated to its true nature, thus reducing the risk of detection.

Beyond personal and organizational data security, this application could be beneficial in environments where discreet communication is necessary. For example, in scenarios involving sensitive negotiations or covert operations, the ability to encode messages in chess game files adds an extra level of camouflage. Moreover, with the increasing use of online communication and data sharing, an application like *Queen's Gambit* can be invaluable for users who require a reliable method to ensure their data remains confidential.

Key Points:

- **Enhanced Security**: Provides an innovative encryption method using chess moves, reducing predictability and making it harder for attackers to decrypt the data.
- **Data Obfuscation**: Encodes encrypted data in PGN files, disguising the information and preventing unauthorized detection or analysis.
- **Applicability Across Sectors**: Suitable for individuals, organizations, and scenarios requiring discreet, secure data transmission, such as sensitive business communications or covert messaging.
- **Adaptability to Evolving Threats**: Combines traditional encryption with unique encoding, making it robust against a variety of cyber threats.
- **User Demand**: Meets the growing need for more advanced and less detectable methods of data encryption in an increasingly digital and interconnected world.

4. Scope

The scope of *Queen's Gambit* encompasses a wide range of applications in secure data storage, encrypted communication, and steganography. This project's design leverages both traditional encryption techniques and innovative encoding strategies to cater to contexts where data confidentiality is paramount. By combining AES and RSA encryption with a chess-based encoding approach, the project introduces a robust solution to secure data in both file storage and file transfer scenarios. This makes it suitable for various sectors, from academic research in cryptography to secure personal and enterprise data handling.

One notable aspect of this application's scope is its potential to function in domains that value covert data transmission and obfuscation. Educational institutions, for instance, may find it useful for teaching cryptography, as it provides a tangible example of combining encryption with creative encoding. Moreover, individuals and organizations with privacy concerns can use it to securely transmit data disguised as a chess game. Since the encoded data appears as a standard PGN file, it is less likely to attract unwanted scrutiny, making it well-suited for secure environments.

Additionally, the project scope extends into future iterations, which could bring web and mobile platform compatibility, cloud-based storage solutions, and more advanced cryptographic features. With these developments, *Queen's Gambit* could evolve into a powerful multi-platform tool for a broader audience, from end users who prioritize privacy to professionals handling sensitive files in industries like law, finance, and healthcare.

Key Points:

- **Wide Range of Applications**: Suitable for secure data storage, encrypted file transfer, and covert communication in sensitive sectors.
- **Educational Relevance**: Can serve as a cryptographic teaching tool, illustrating a unique approach to encryption through gamified data encoding.
- **Use of Familiar Formats for Obfuscation**: Encodes encrypted data as PGN files, making it inconspicuous and ideal for environments where covert data transmission is essential.
- **Future Development Potential**:
    - **Platform Expansion**: Scope to develop as a web or mobile application, enhancing accessibility and user experience.
    - **Cloud Storage Integration**: Could enable secure, cloud-based storage of encrypted data disguised as chess games, adding a layer of security and accessibility.
    - **Enhanced Cryptography**: Room for further security upgrades, such as advanced cryptographic protocols to meet evolving data protection standards.
- **Broader Target Audience**: Potentially useful for privacy-conscious individuals, students in cryptography, and industries requiring high confidentiality.

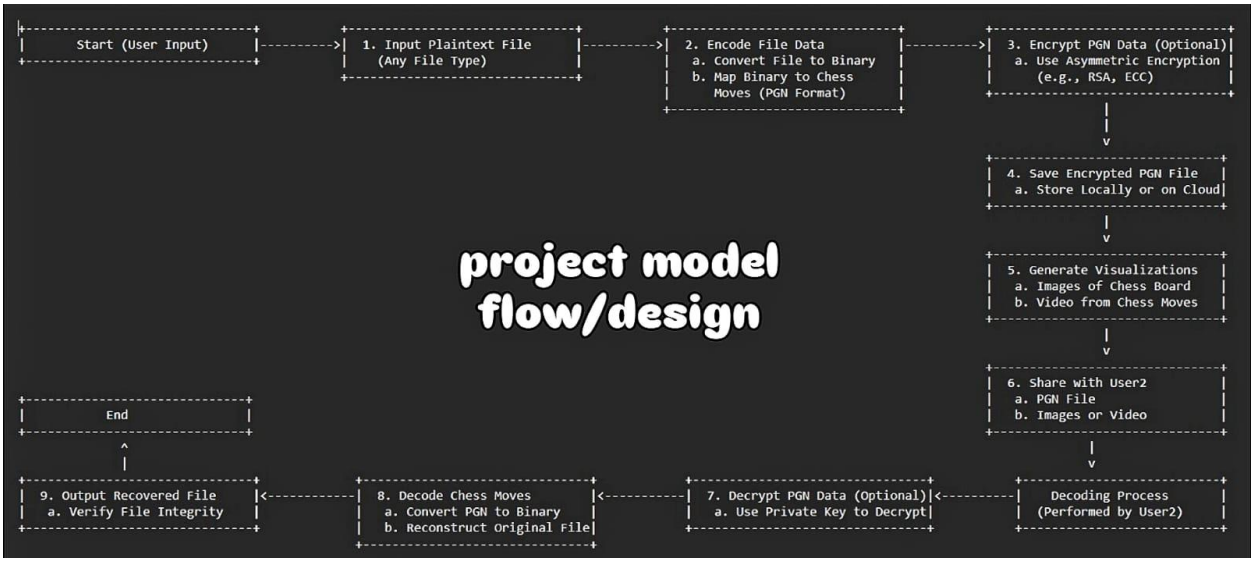
5. Project Description

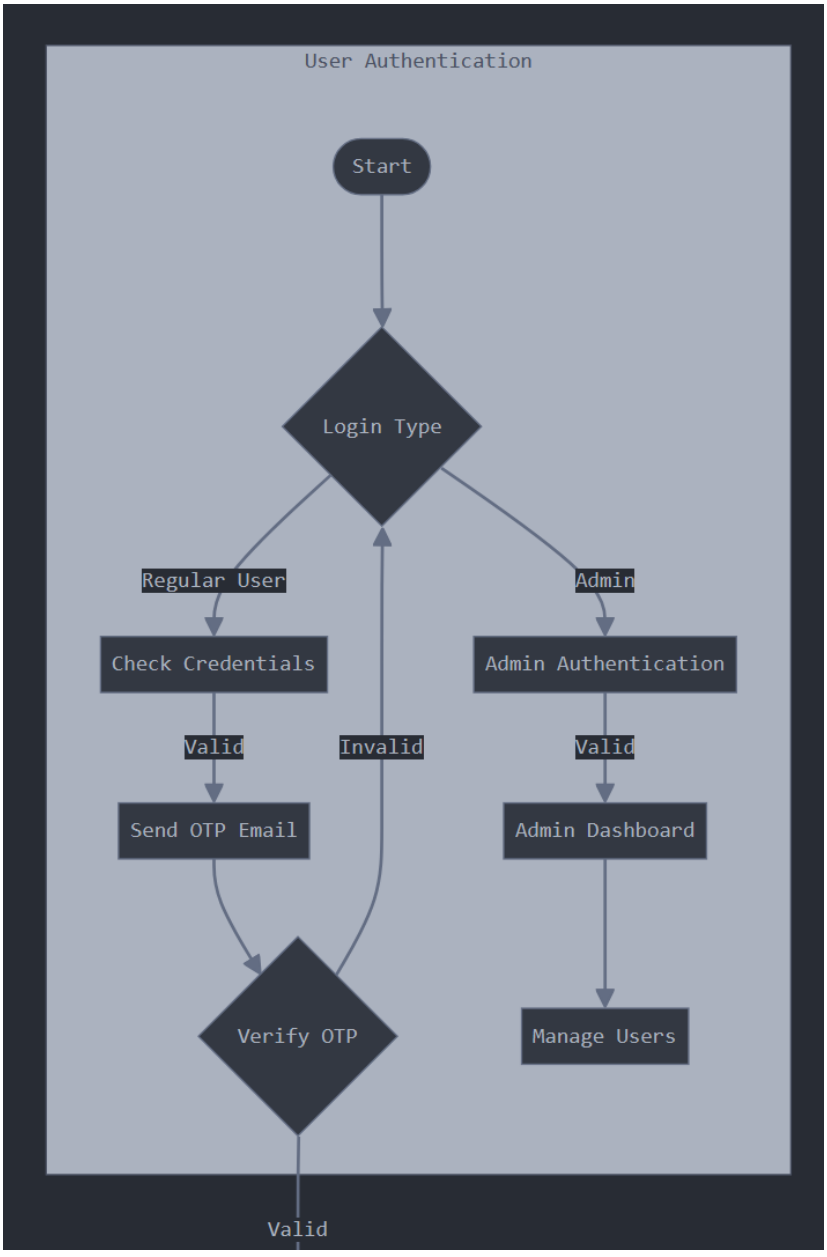
Queen's Gambit is a comprehensive software system designed to encode and encrypt various types of files using a blend of traditional cryptographic methods and a novel approach inspired by the game of chess. The application is built using Python and the Flask web framework, providing a user-friendly interface for secure file encryption, OTP-based user authentication, and file management. The core functionality involves transforming binary data into sequences of chess moves and storing these moves in a Portable Game Notation (PGN) file, a common format for recording chess games.

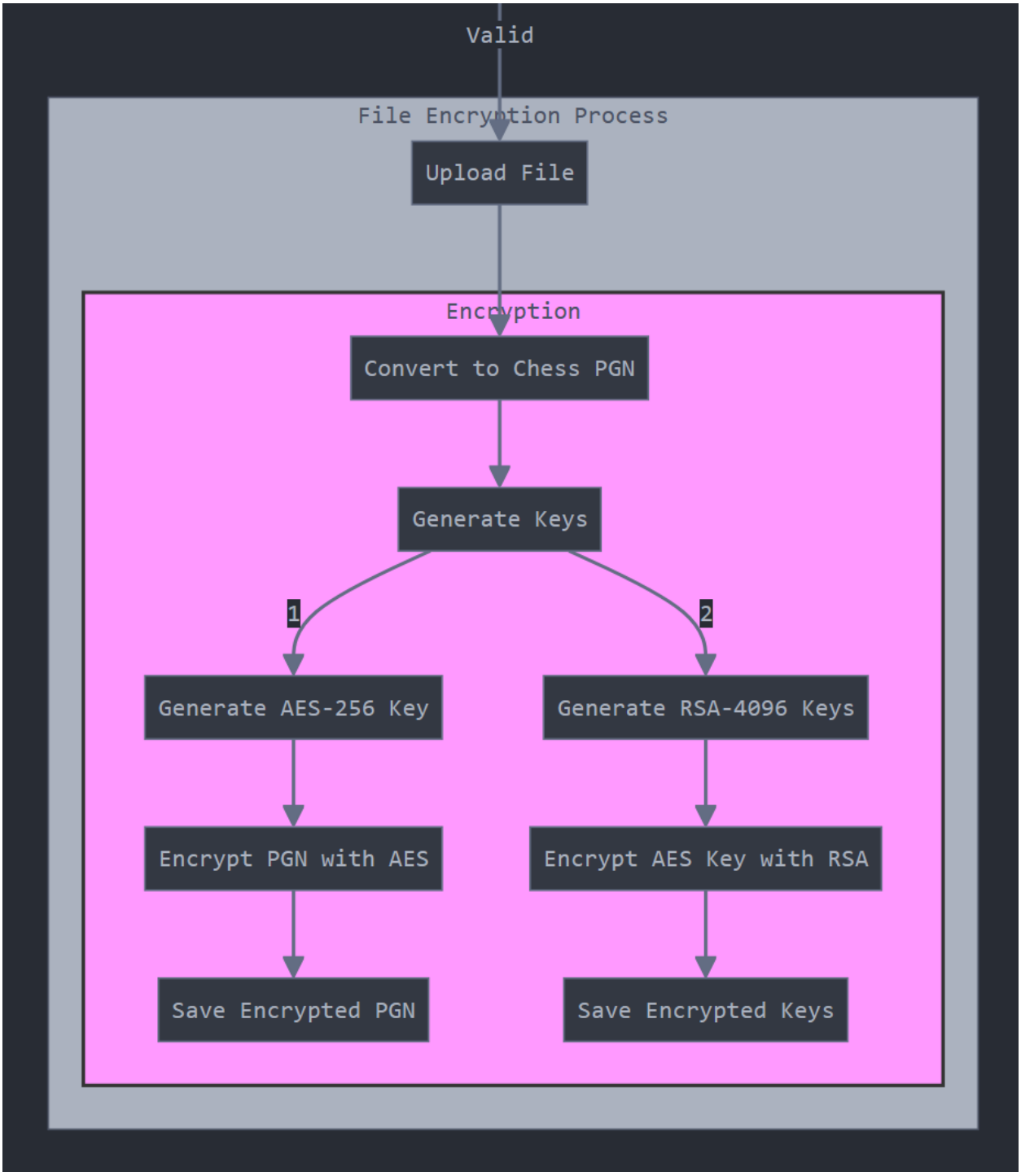
The encryption process begins with the user uploading a file, which is then encrypted using AES (Advanced Encryption Standard). The AES key itself is encrypted using RSA (Rivest–Shamir–Adleman) encryption, ensuring that even if the encrypted data is intercepted, it cannot be decrypted without access to the RSA private key. The encoded data is mapped onto legal chess moves, creating a series of moves that collectively represent the entire file. The resulting PGN file is indistinguishable from a standard chess game record, enhancing the stealth and security of the encrypted data.

For decryption, the application reads the PGN file, deciphers the RSA-encrypted AES key, and uses it to decrypt the encoded data back into its original binary form. This method not only ensures robust data protection but also offers a unique way of disguising sensitive information. The project includes a web-based interface, allowing users to register, authenticate using OTP, and manage files securely. The backend system is capable of handling user sessions, file uploads, and encrypted file storage, making it a complete solution for secure data handling.

Model Design/Flowchart:


project model
flow/design
Start (User Input)
1. Input Plaintext File
(Any File Type)
2. Encode File Data
a. Convert File to Binary
b. Map Binary to Chess
Moves (PGN Format)
3. Encrypt PGN Data (Optional)
a. Use Asymmetric Encryption
(e.g., RSA, ECC)
4. Save Encrypted PGN File
a. Store Locally or on Cloud
5. Generate Visualizations
a. Images of Chess Board
b. Video from Chess Moves
6. Share with User2
a. PGN File
b. Images or Video
Decoding Process
(Performed by User2)
7. Decrypt PGN Data (Optional)
a. Use Private Key to Decrypt
8. Decode Chess Moves
a. Convert PGN to Binary
b. Reconstruct Original File
9. Output Recovered File
a. Verify File Integrity
End


User Authentication
Start
Login Type
Regular User
Admin
Check Credentials
Admin Authentication
Valid
Invalid
Valid
Send OTP Email
Admin Dashboard
Verify OTP
Manage Users
Valid



Above is the flowchart shown for the encryption process, the decryption process is similar to this.

As of yet, this project is a CLI (Command Line Interface) App, and it could be improvised into a fully functional app/website with better UI/UX. This would be considered as an additional feature and is not the CORE idea of this project.

Key Points:

- **Data Encryption and Encoding**:
    - Uses AES for data encryption and RSA for key encryption, combining symmetric and asymmetric encryption for enhanced security.

    - Encodes encrypted data as chess moves, stored in a PGN file format to disguise the encrypted content.

- **Secure User Authentication**:
    - Implements OTP-based login to verify user identity, adding a layer of security before file encryption/decryption operations.
- **Web-Based Interface**:
    - Provides a user-friendly platform for uploading, encrypting, and decrypting files with ease, accessible from any device with a web browser.
- **End-to-End Security**:
    - Includes multi-layer encryption (AES-RSA), file encoding as chess games, and OTP authentication to ensure data remains protected throughout its lifecycle.
- **Innovative Use of Chess**:
    - Utilizes chess game notation as a unique medium for encoding data, combining cryptographic techniques with an unconventional approach to data obfuscation.

6. Related Works / Existing State of Art Architectures

In the field of cryptography and data security, numerous applications and architectures focus on using conventional methods like AES (Advanced Encryption Standard) and RSA (Rivest–Shamir–Adleman) encryption for securing sensitive information. These techniques are widely recognized for their strength and reliability in protecting data, both in transit and at rest. However, the rise of sophisticated cyberattacks has necessitated the exploration of innovative approaches that go beyond traditional encryption. This is where *Queen's Gambit* introduces a unique twist by blending standard cryptography with elements of steganography through the game of chess.

Existing Encryption Solutions

Most modern encryption systems rely heavily on symmetric (e.g., AES) and asymmetric (e.g., RSA) cryptography. Symmetric encryption like AES is highly efficient for encrypting large data volumes, while RSA provides secure key exchange, making them a powerful combination for securing data in various applications, including online banking, secure messaging, and file storage services. However, these methods often exhibit identifiable patterns in the encrypted data, which can be exploited by advanced techniques like steganalysis and pattern recognition.

Steganography and Obfuscation Techniques

Steganography involves hiding data within other seemingly innocent files, such as images, audio files, or even text documents. Popular tools like Steghide and OpenPuff use steganography to embed hidden messages within multimedia files. However, these methods can sometimes be vulnerable to detection through statistical analysis and pattern recognition. *Queen's Gambit* differentiates itself by using the complex, unpredictable nature of chess moves, encoded in PGN files, as a medium for steganography, making the encoded data difficult to distinguish from regular chess game records.

Innovative Use of Chess in Cryptography

While there have been some experimental projects using games as a medium for cryptography, few have explored the potential of chess as deeply as *Queen's Gambit*. The unique use of chess moves encoded in PGN format provides an additional layer of obfuscation, leveraging the vast complexity of chess to disguise data effectively. The project combines this innovative encoding with strong cryptographic practices (AES and RSA), resulting in a novel approach that merges steganography with encryption.

Key Points:

- **Conventional Encryption**: Standard AES and RSA encryption techniques form the core of secure data handling in existing solutions.
- **Steganographic Methods**: Popular steganography tools hide data within multimedia files but may be vulnerable to advanced detection methods.
- **Unique Approach of Queen's Gambit**: Introduces a novel blend of chess-based encoding and cryptographic techniques, enhancing data obfuscation and security.
- **Comparison with Existing Works**: Provides a new avenue in cryptography by using chess as a medium, a concept not widely explored in existing state-of-the-art architectures.

7. Method

The method implemented in *Queen's Gambit* combines several layers of security to ensure the confidentiality, integrity, and obfuscation of user data. The application follows a structured approach, beginning with user authentication and extending through data encryption, encoding, and secure transmission. Here's a detailed breakdown of the process:

User Authentication via OTP

The process starts with a secure user login system that requires OTP (One-Time Password) verification. When users attempt to log in, an OTP is sent to their registered email address. The application uses Python's smtplib to send the OTP, ensuring that only verified users gain access to sensitive features like file encryption and decryption.

- **Registration and OTP Verification**: Users register with their details and must verify their email using a six-digit OTP, preventing unauthorized access.
- **Session Management**: Secure sessions are created for logged-in users, reducing the risk of session hijacking.

File Encryption and AES-RSA Layering

Once authenticated, users can upload files for encryption. The application uses AES-256 for data encryption, providing strong symmetric encryption. The AES key itself is encrypted using RSA-4096, an asymmetric encryption technique, ensuring that the AES key cannot be decrypted without the corresponding RSA private key. This dual-layer encryption method combines the speed of AES with the security of RSA.

- **AES Encryption**: Encrypts the file content using a symmetric key, ensuring fast and secure data encryption.

- **RSA Key Encryption**: Encrypts the AES key with an RSA public key, making it secure even if the encrypted file is intercepted.
- **Key Storage**: The RSA-encrypted AES key is stored separately, further enhancing security.

Data Encoding as Chess Moves

The encrypted data is then encoded into a series of chess moves. The application uses the Python chess library to simulate a chessboard and map binary data onto legal chess moves. Each byte of encrypted data is translated into a corresponding move, and the sequence is recorded in a Portable Game Notation (PGN) file.

- **Binary-to-Move Mapping**: Translates binary data into legal chess moves using the current board state.
- **PGN File Creation**: The sequence of chess moves is saved in a PGN file, which appears as a regular chess game record.

Decryption and Decoding

To retrieve the original file, the user uploads the PGN file for decryption. The application first decrypts the AES key using the RSA private key, then uses this AES key to decrypt the file contents. The encoded chess moves are interpreted back into binary data, reconstructing the original file.

- **RSA Decryption**: Decrypts the AES key using the RSA private key.
- **AES Decryption**: Decrypts the file content with the decrypted AES key.
- **Move Decoding**: Interprets chess moves back into binary data, reconstructing the original file.

Key Points:

- **Multi-Layered Security**: Combines OTP authentication, AES data encryption, RSA key encryption, and chess-based encoding.
- **Innovative Encoding**: Uses chess moves as a medium for encoding data, enhancing obfuscation.
- **Efficient Decryption**: Provides a structured approach to decrypt and decode data, ensuring accurate file recovery.

8. Functionalities

The *Queen's Gambit* application offers a range of robust and innovative functionalities designed to provide a secure and user-friendly experience. These features collectively enhance data security and make the application versatile for various use cases, from personal data protection to secure file handling in professional settings. Each functionality has been carefully implemented to align with the project's objective of creating a unique and efficient encryption system.

User Registration and OTP Authentication

One of the standout features of this application is its secure user registration process, which includes OTP-based email verification. When a new user signs up, an OTP (One-Time Password) is sent to their registered email address for identity verification. This adds an extra layer of security by ensuring that only authorized users can access the platform. The OTP mechanism also reduces the risk of unauthorized account creation or access, enhancing the overall security of the application.

- **User Registration**: New users can create accounts, with a mandatory OTP verification step to ensure valid registration.
- **OTP Authentication**: OTPs are sent via email to verify user identity during login, enhancing security against unauthorized access.
- **Session Management**: The application securely manages user sessions, ensuring that authenticated users maintain access during their interactions.

Admin Dashboard

The application includes a dedicated admin dashboard, providing the administrator with a centralized interface to manage user accounts. The admin can view a list of registered users, monitor their activity, and delete accounts if necessary. This feature allows for effective user management and helps maintain the security and integrity of the platform.

- **User Monitoring**: Admins can view all registered users and their details.
- **Account Deletion**: Admins have the capability to delete user accounts, enhancing control over the user base.
- **Security Oversight**: The dashboard allows admins to monitor user activity, ensuring the platform is used securely and appropriately.

File Encryption and Decryption

The core functionality of *Queen's Gambit* revolves around its encryption and decryption capabilities. Users can upload various file types, which are then encrypted using a combination of AES and RSA encryption. The encrypted data is encoded as a series of chess moves in PGN format. For decryption, users provide the necessary keys, and the application decodes the chess moves back into the original file format.

- **File Upload and Encryption**: Users can upload files, which are encrypted using AES (for data) and RSA (for key encryption).
- **Data Encoding**: Encrypted data is mapped onto legal chess moves, stored in a PGN file to disguise its true content.
- **File Decryption**: Users can decrypt the PGN file back into its original format, using the RSA-decrypted AES key.

Account Management

The application offers comprehensive account management features, allowing users to delete their accounts if they no longer wish to use the service. This ensures user control over their data and accounts, aligning with best practices for user privacy and data protection.

- **User Account Deletion**: Users can delete their own accounts, ensuring they have control over their personal data.
- **Admin Control**: Admins can manage and delete user accounts as needed, enhancing platform security.

Key Points:

- **Secure Authentication**: User registration and login require OTP verification to prevent unauthorized access.
- **Robust Encryption**: Combines AES and RSA encryption to secure files, providing a high level of data protection.
- **User and Admin Features**: Offers functionalities for both regular users (file encryption/decryption) and admins (user management).
- **Flexible File Handling**: Supports various file types for encryption, ensuring versatility and broad applicability.

9. Results and Demonstration

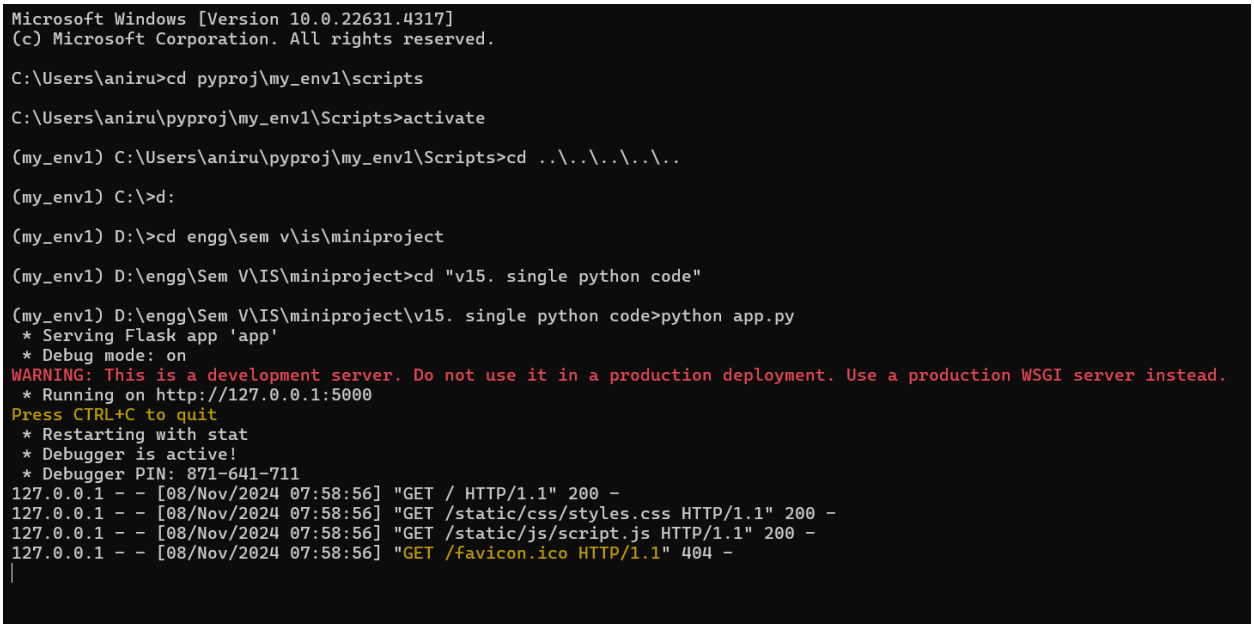


Fig 1. Loading and running of the python flask application on the command line interface.

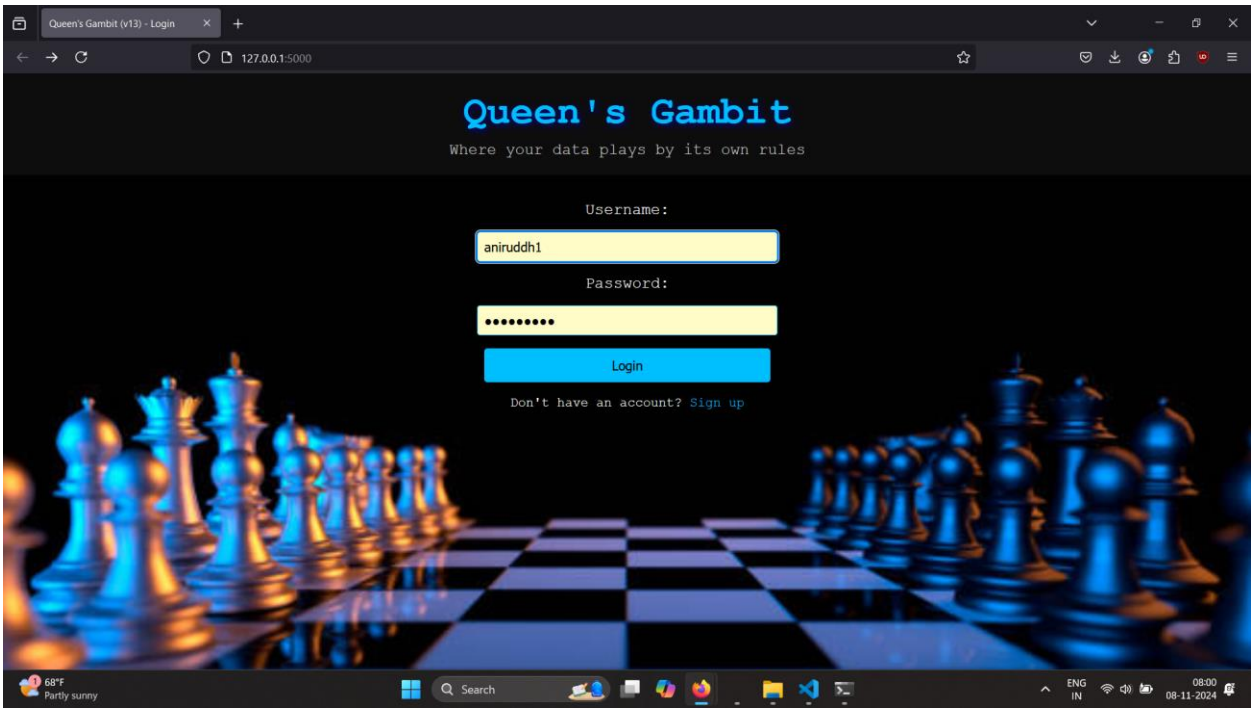


Fig 2. Login screen for a sample user.

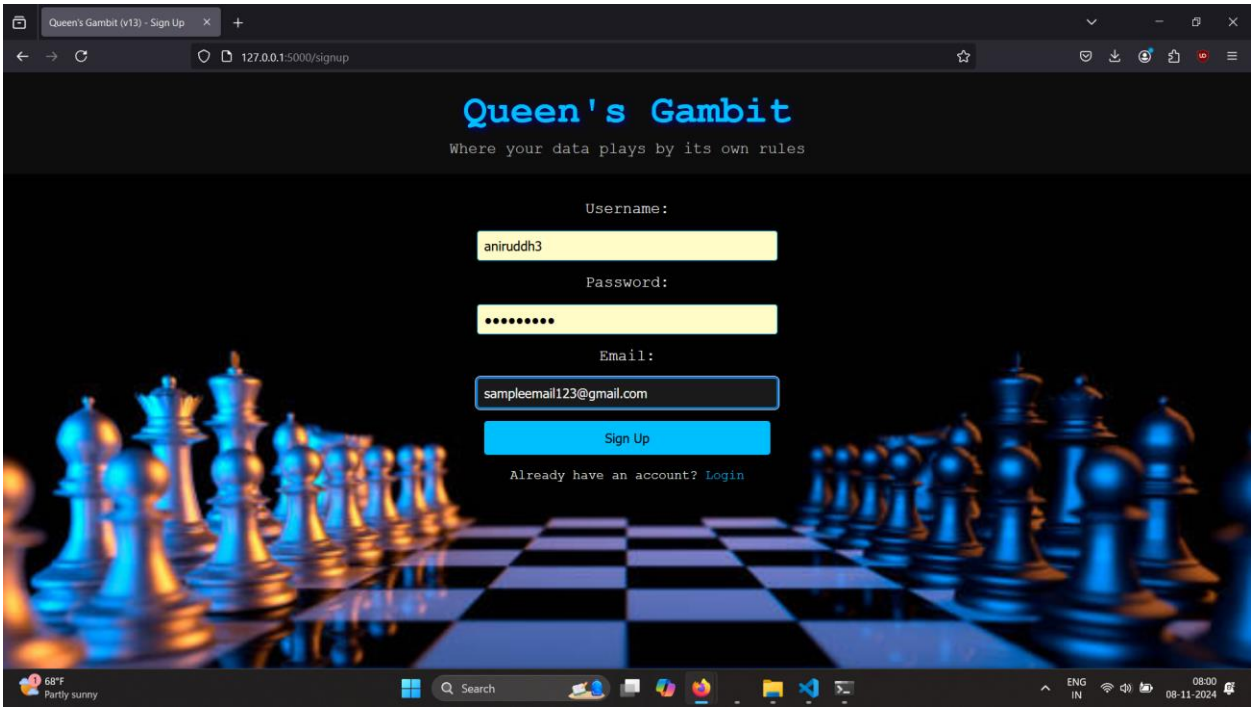


Fig 3. Signup screen for a sample user.

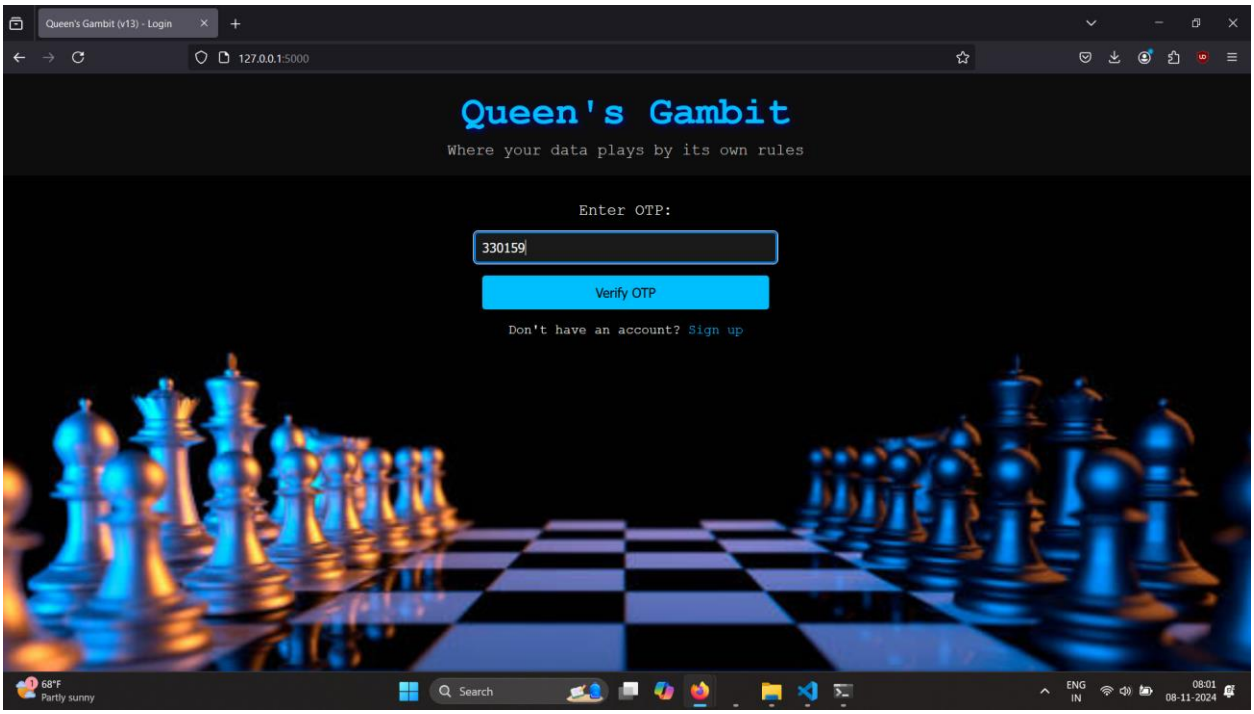


Fig 4. OTP Verification for a sample user.

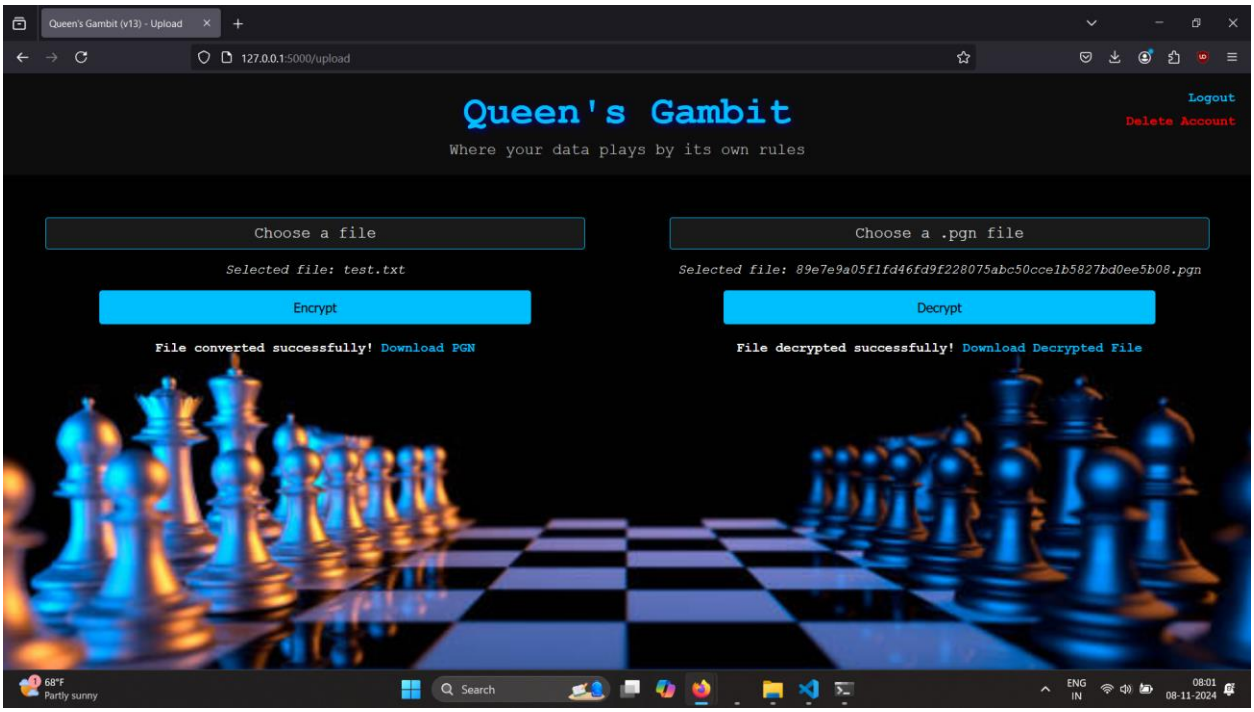


Fig 5. Main page to upload any file to get its encrypted PGN file, and later to upload the PGN file to get the original file back.

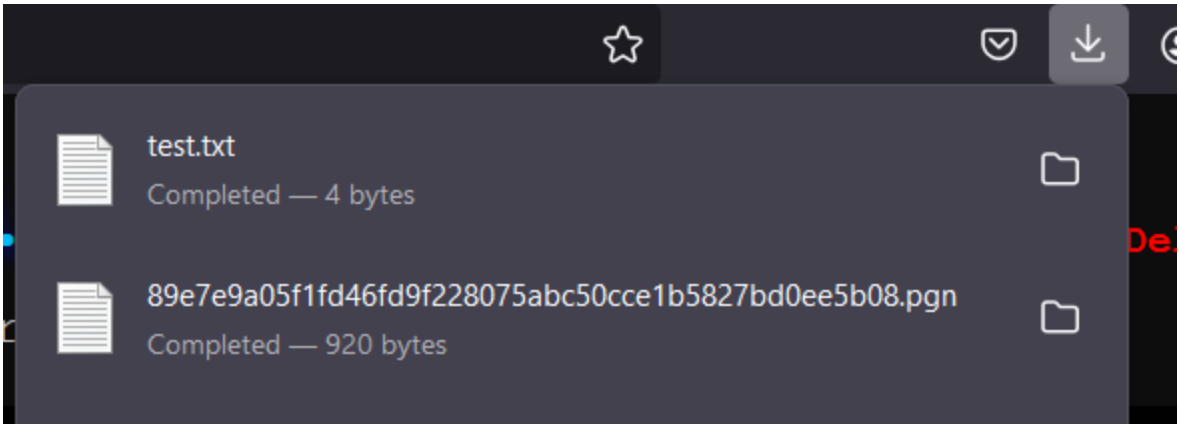


Fig 6. Downloaded the encrypted PGN file and the decrypted file (same as original file uploaded).

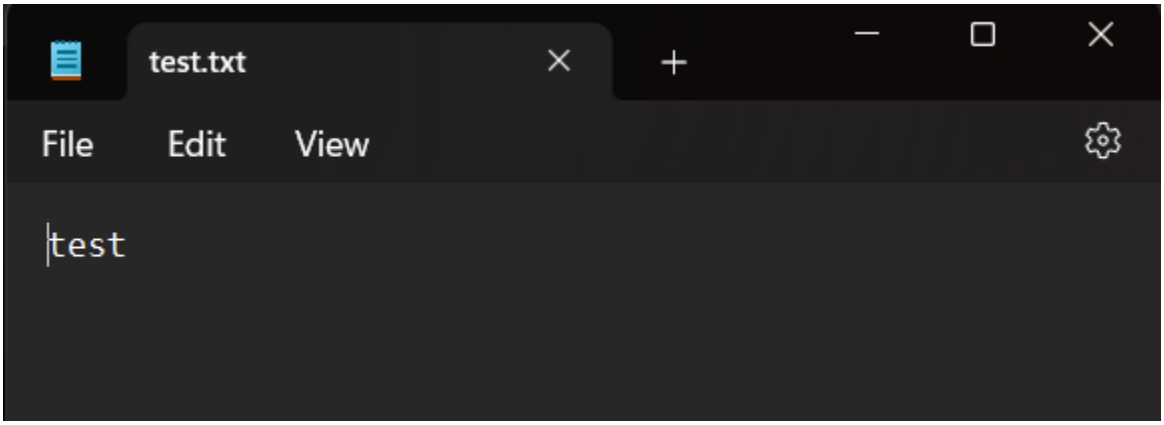


Fig 7. Content of the original sample file (test.txt).

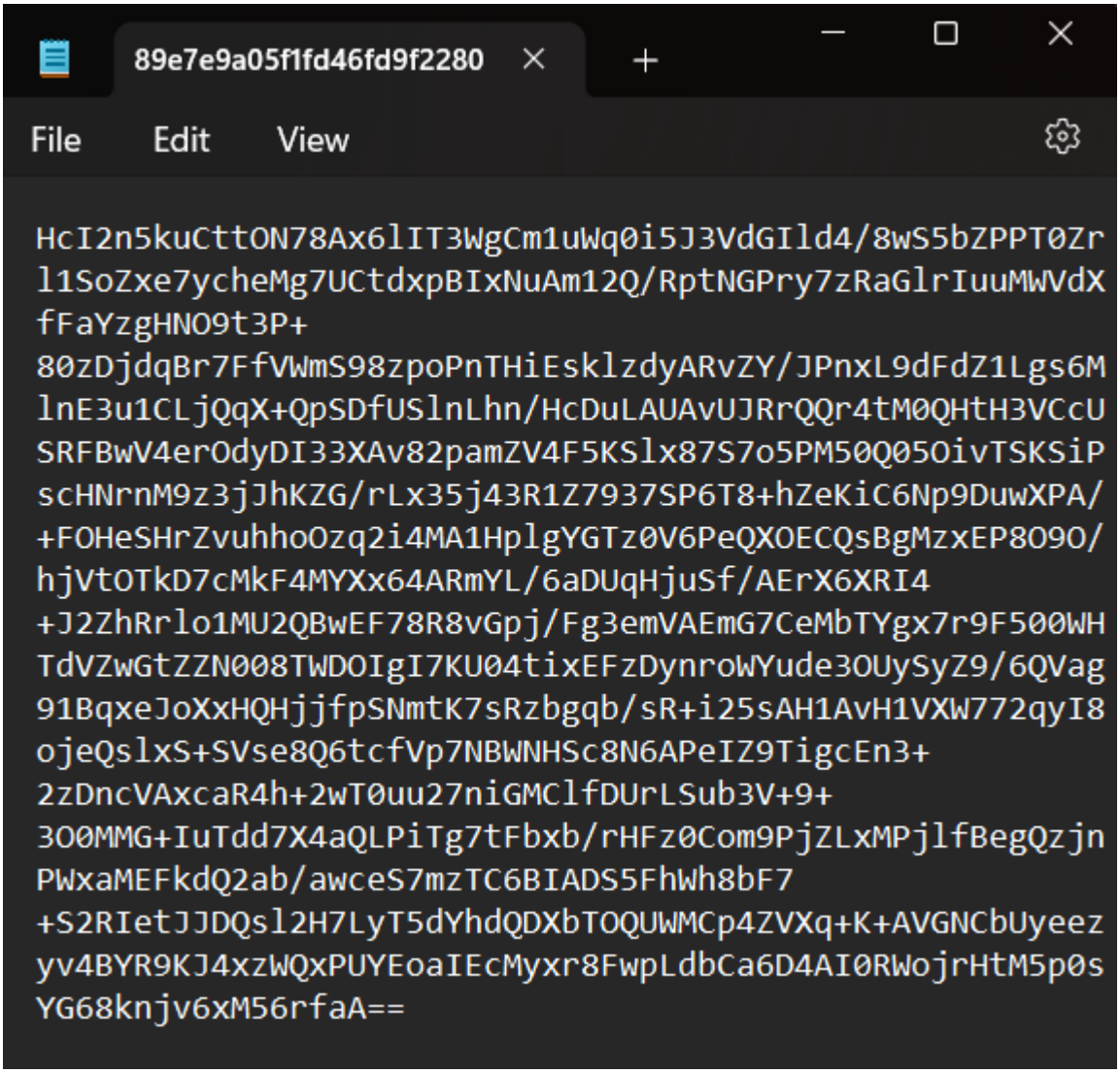


Fig 8. Content of the encrypted PGN file.

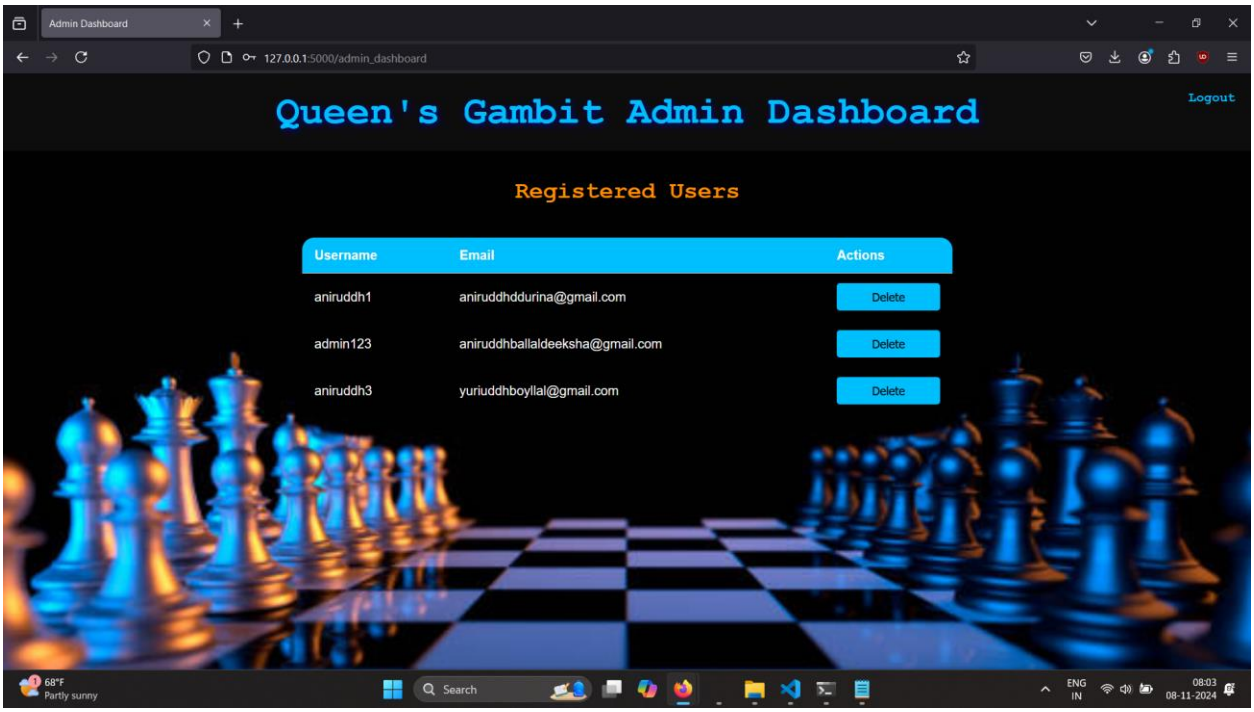


Fig 9. Admin login to view and delete any user of the app.

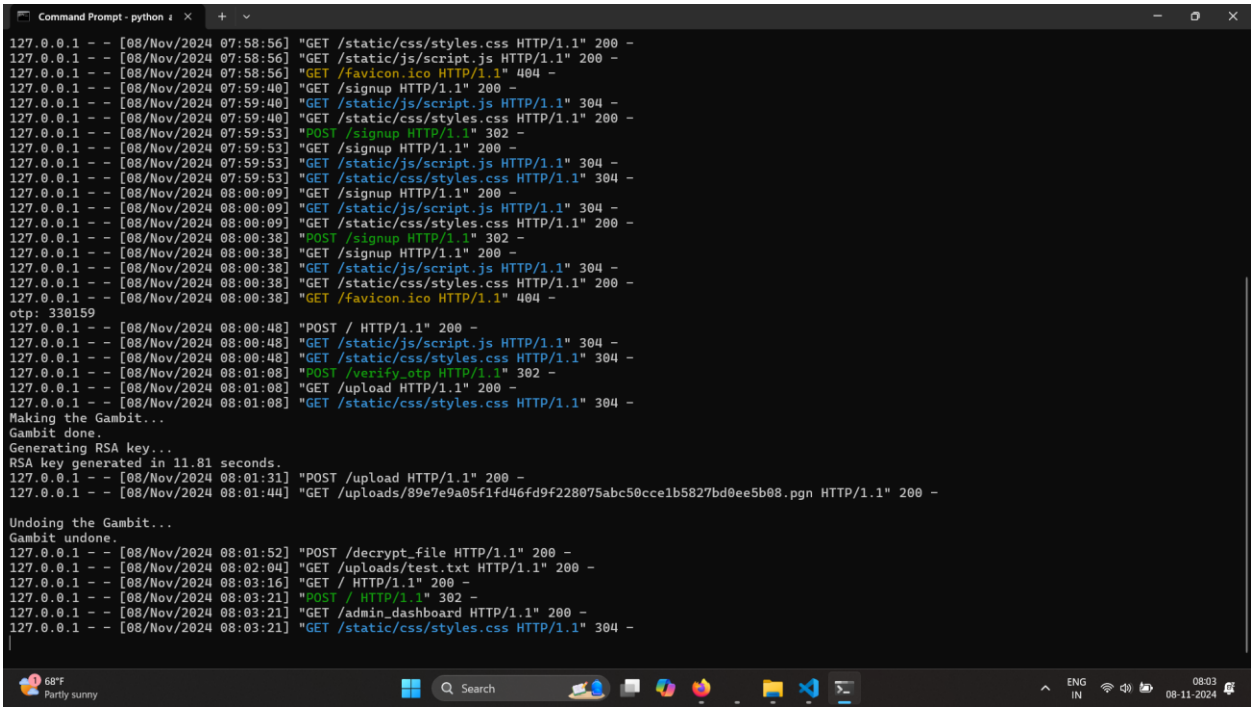


Fig 10. Command Line debug logs while running the app.

10. Hardware Requirements

To run *Queen's Gambit* effectively, certain hardware specifications are recommended to ensure smooth performance, especially during encryption and decryption processes. Given the computational demands of handling cryptographic operations and managing user sessions in a web environment, the application requires a moderate level of hardware capability.

Minimum Requirements:

- **Processor**: Intel Core i3 or equivalent – Suitable for basic encryption and web server operations, handling small to medium-sized files efficiently.
- **RAM**: 4 GB – Adequate for basic use cases but may limit performance during simultaneous user activities or when processing larger files.
- **Storage**: 500 MB of free disk space – Enough to store the application files and temporary encrypted/decrypted files.
- **Operating System**: Windows 10, macOS, or any Linux distribution – The application is designed to be cross-platform, ensuring compatibility with major operating systems.

Recommended Requirements:

- **Processor**: Intel Core i5 or higher – Provides better performance for handling complex encryption tasks and supports a larger number of concurrent users.
- **RAM**: 8 GB or more – Ensures smooth performance during simultaneous encryption, decryption, and user interactions.
- **Storage**: 1 GB or more – Accommodates additional files and larger data sets, especially when working with multimedia or large text files.
- **Network**: Stable internet connection – Required for sending OTPs via email and for accessing the web application.

Key Points:

- **Scalable Performance**: Higher specifications improve performance, especially for handling multiple users or larger files.
- **Cross-Platform Compatibility**: Works on Windows, macOS, and Linux, offering flexibility in deployment.
- **Internet Access**: Necessary for email-based OTP verification and secure user authentication.

11. Software Requirements

The software requirements for *Queen's Gambit* focus on providing a reliable and secure environment for both development and deployment. The application is built using Python, with several additional libraries and frameworks to handle web functionalities, encryption processes, and chess encoding.

Development Environment:

- **Python 3.x**: The primary programming language used, chosen for its extensive library support and ease of use in web development and cryptographic functions.
- **IDE/Text Editor**: Any Python-compatible IDE, such as PyCharm, Visual Studio Code, or Jupyter Notebook, can be used for development.

Libraries and Packages:

- Here's a brief summary of how each imported library or package is used in your *Queen's Gambit* application:
- **Flask and Related Imports**
    - **Flask**: Used to create the web application, handle HTTP requests, and manage routing between different web pages.
    - **render_template**: Renders HTML templates, enabling dynamic content display on web pages.
    - **request**: Accesses form data submitted by users (e.g., during login or file upload).
    - **redirect, url_for**: Handles page redirections within the application based on specific routes or actions.
    - **flash**: Displays one-time messages (e.g., error or success messages) to users on the front-end.
    - **send_from_directory**: Serves files from a specified directory, used for downloading encrypted or decrypted files.
    - **session**: Manages user sessions, storing temporary data like OTPs and user authentication status.
- **Python Standard Libraries**
    - **os**: Handles file operations, such as creating directories, saving uploaded files, and managing file paths.
    - **base64**: Encodes and decodes data in Base64 format, used for encrypting and transmitting binary data securely as text.
    - **json**: Loads and saves user data to JSON files, enabling easy data storage and retrieval.
    - **time**: Measures execution time for certain processes (e.g., RSA key generation timing).
    - **smtplib**: Sends emails via SMTP, specifically used for sending OTPs during the user authentication process.
    - **random**: Generates random numbers for OTPs and user IDs, enhancing security.
    - **math (log2)**: Calculates the binary length required for representing indices in move selection, used in chess data encoding.
- **Cryptography (Crypto) Library**
    - **Crypto.PublicKey.RSA**: Generates RSA key pairs (public and private keys) for encrypting and decrypting AES keys, providing asymmetric encryption.
    - **Crypto.Random.get_random_bytes**: Generates random bytes, used for creating AES keys and other random data securely.
    - **Crypto.Cipher.AES**: Implements AES encryption for securing file content.

    - **Crypto.Cipher.PKCS1_OAEP**: Provides an RSA encryption/decryption scheme, specifically used for encrypting the AES key with the RSA public key and decrypting it with the private key.
    - **Crypto.Util.Padding (pad, unpad)**: Pads plaintext data before AES encryption to fit block size requirements and removes padding after decryption.
- **Email Handling**
    - **MIMEText**: Creates email messages in plain text format, used to compose OTP email messages sent to users during authentication.
- **Chess Library**
    - **chess.Board**: Manages the chessboard state, legal moves, and move execution, facilitating the encoding of binary data as chess moves.
    - **chess.pgn**: Handles the creation and parsing of Portable Game Notation (PGN) files, which store sequences of chess moves representing the encoded data.
- **Other Libraries**
    - **io.StringIO**: Creates in-memory file-like objects for handling text data, used to parse PGN strings without requiring actual file I/O operations.

Additional Tools:

- **Version Control (Git)**: For managing code versions and collaborating during development.
- **Operating System Compatibility**: The application can run on Windows, macOS, and Linux, as long as Python 3.x is installed.
- **Database (optional)**: Could be integrated for storing user data and encryption keys, such as SQLite or PostgreSQL for lightweight, scalable storage solutions.

Key Points:

- **Python-Based**: Built on Python for flexibility and ease of integration with various libraries.
- **Core Libraries**: Uses Flask for web development and PyCrypto/Crypto for robust encryption functionalities.
- **Flexible Development Tools**: Compatible with various IDEs and version control systems, enhancing development and deployment efficiency.
- **Cross-Platform Support**: Ensures the software can be easily run on different operating systems, provided Python is installed.

12. Future Scope

Queen's Gambit has significant potential for enhancement, with a range of future developments that could make it more versatile, secure, and accessible across multiple platforms. Expanding the application beyond its current capabilities could open new possibilities, such as integrating cloud storage, adding advanced cryptographic features, and increasing user accessibility through mobile

and web platforms. Each of these improvements would strengthen the application's position as a robust tool for secure data management.

One promising avenue for future development is **cloud integration**, allowing users to securely store encrypted data remotely. By encoding sensitive information in a PGN file format that resembles regular chess games, *Queen's Gambit* could offer secure, cloud-based storage that doesn't draw attention or expose the data's true nature. This would be especially beneficial for users requiring accessible and secure storage options without sacrificing confidentiality.

Additionally, **advanced cryptographic enhancements** could be incorporated, including multi-layer encryption methods and asymmetric encryption algorithms. While the current design includes AES and RSA encryption, further upgrades could involve incorporating state-of-the-art encryption standards or even hybrid encryption methods. This would allow *Queen's Gambit* to remain adaptable and relevant in a fast-evolving field, ensuring it provides a high level of protection against sophisticated cyber threats.

Another area of potential growth is **expanded platform compatibility**. By developing both web and mobile versions, *Queen's Gambit* could reach a broader audience and provide users with the flexibility to encrypt, upload, and decode files on various devices. For instance, a mobile version would enable users to manage encrypted data on the go, while a web-based interface could offer easier access for non-technical users, making the application user-friendly and highly accessible.

Another point to be noted is that the chess encryption depends on the number of legal moves that are possible at a given state of the chess pieces on the chess board. And it needs a minimum of two possible legal moves for it to be able to assign each move to a binary value and then encrypt the data – hence, the encryption stops the moment there is only one possible legal move for either of the two sides (black or white). Hence, the game never really reaches an end. Because for the chess game to reach an end, there is atleast one consecutive move where the player has no option but to move a particular chess piece to a particular square on the board. And this wouldn't allow for data to be encrypted as it needs a minimum of two moves – to assign a move to a certain binary number and another to the next binary number. With only one possible legal move, there cannot be any data further encoded in the same game. Hence, every game that the chess encryption function (make_gambit) generates in the code, is incomplete/ongoing. This is an issue that has to be further looked into, as if unwanted parties decipher the AES-RSA encrypted data into the chess data, then it would be very evident that every chess game in the PGN file is incomplete and could raise suspicions. A method to play dummy moves or somehow finish the game once it reaches such a predictable state could be included but that further complicates the decryption process as there must be a way to identify and separate actual data encoded moves from the dummy moves that are played just to bring the chess game to an actual end. One minor point to also be noted, is that the metadata generated for every chess game is dummy and random and it is possible for the perpetrator to analyse the chess game and realise that some if not all of the chess game's metadata is not matching the actual chess moves being played in the game. Hence, the chess metadata should also be looked into – for further improvisation of this app.

Key Points:

- **Cloud Integration**:
    - Enables secure, remote storage of encoded data in the cloud, disguising data within typical chess PGN files.

    - Increases accessibility and provides a scalable storage solution for users with large or complex data needs.

- **Advanced Cryptographic Enhancements**:
    - Potential to incorporate cutting-edge encryption methods, hybrid encryption, or adaptive cryptographic approaches to maintain data security.
    - Multi-layered encryption methods could be added to increase protection against advanced cyber threats.
- **Expanded Platform Compatibility**:
    - Development of web and mobile applications to increase accessibility and convenience.
    - Mobile app compatibility would enable users to encrypt and manage data securely from their devices.
    - Web-based access would make the application more user-friendly, especially for those unfamiliar with command-line interfaces.
- **Enhanced Steganography and Obfuscation**:
    - Additional techniques could be added to further disguise encoded data and resist steganalysis.
    - Dynamic, randomized chess moves or obfuscation patterns could make the encrypted data even harder to detect and analyze.
- **User-Centric Features**:
    - Potential for adding user-friendly features like GUI improvements, integration with Lichess.org bots for dynamic chess game simulations, or automated key-sharing functionalities.
    - Key-sharing protocols could allow secure communication and encrypted file sharing between verified users.

13. Study Limitations

While *Queen's Gambit* presents a unique and innovative approach to data encryption by combining cryptographic methods with chess-based encoding, several limitations need to be addressed to enhance its practicality and scalability. These limitations offer valuable insights for future iterations and highlight areas where improvements could significantly boost the application's performance and usability.

Performance and Scalability Issues

One of the primary limitations of the current system is its performance when handling large files. The process of encrypting data using AES, encoding it as chess moves, and then storing it in a PGN file can become computationally intensive, especially for files of substantial size. This can lead to longer processing times, which may affect the user experience. Additionally, the system's

reliance on the Python chess library for encoding and decoding data introduces computational overhead, especially during the generation and interpretation of legal chess moves.

- **File Size Limitations**: The encoding process can become slow and resource-intensive for large files, affecting usability.
- **Processing Overhead**: The reliance on Python's chess library for move generation adds computational complexity, impacting performance.
- **Scalability Challenges**: As the number of users or file sizes increases, the application may require more processing power and memory, making it less efficient without further optimization.

Security Concerns and Threat Mitigation

Although the application combines AES and RSA encryption with OTP-based authentication to provide a robust security framework, it is not immune to potential vulnerabilities. For instance, the strength of the RSA encryption relies heavily on secure key management. If the private RSA key is compromised, the AES key can be decrypted, exposing the encrypted data. Moreover, while using PGN files for obfuscation adds a layer of disguise, determined attackers could still perform pattern analysis on the moves to identify encoded content, especially if they suspect that the PGN files are being used for data encoding rather than genuine chess games.

- **Key Management Risks**: The security of the RSA-encrypted AES key is dependent on the proper safeguarding of the RSA private key.
- **Potential for Pattern Detection**: If attackers suspect the use of chess-based encoding, they could analyze move patterns to identify anomalies and detect hidden data.
- **Vulnerability to Social Engineering**: OTP authentication, while secure, could be vulnerable to social engineering attacks where users might be tricked into revealing their OTPs.

Usability and Accessibility Limitations

The current implementation of *Queen's Gambit* is primarily designed as a web-based application accessible through a standard browser interface. However, users who are unfamiliar with cryptographic concepts or chess may find the system intimidating or difficult to use. Additionally, the need for email-based OTP verification requires users to have consistent internet access, which could be a barrier in environments with limited connectivity.

- **User Learning Curve**: Users without a background in cryptography or chess may find the application challenging to navigate and understand.
- **Dependency on Internet Access**: The reliance on email-based OTP for authentication requires a stable internet connection, limiting accessibility in offline or low-connectivity scenarios.
- **Limited Interface Flexibility**: The web-based interface, while user-friendly, may not fully accommodate mobile users or those requiring more advanced features, such as batch encryption of multiple files.

Key Points:

- **Performance Limitations**:
    - The application may struggle with encrypting large files due to computational overhead.
    - Scalability issues may arise as user numbers or file sizes increase without further optimization.
- **Security Limitations**:
    - Vulnerabilities in key management and potential pattern detection could expose encrypted data to risks.
    - Social engineering attacks could potentially bypass OTP-based security measures.
- **Usability Challenges**:
    - The user interface may not be intuitive for those unfamiliar with chess or cryptography.
    - Internet dependency for OTP verification could limit accessibility in offline environments.
    - The current design may not fully support mobile or advanced user scenarios, such as batch processing of files.

Opportunities for Improvement

Addressing these limitations in future updates could enhance the performance, security, and user experience of *Queen's Gambit*. Potential improvements include optimizing the encoding process for better performance, implementing stronger key management practices, and expanding the interface to support mobile devices and offline authentication options. Additionally, integrating advanced steganographic techniques or further randomizing the chess move generation could mitigate the risk of pattern detection.

- **Optimization**: Enhancing the encoding and encryption process to reduce computational overhead and improve performance for large files.
- **Enhanced Key Management**: Implementing more secure methods for storing and managing RSA keys to prevent unauthorized access.
- **User Experience Improvements**: Expanding the interface to include tutorials, mobile support, and offline features to make the application more accessible to a wider audience.
- **Advanced Obfuscation**: Incorporating additional steganographic techniques to further obscure encoded data and resist pattern analysis.

14. Conclusion

Queen's Gambit successfully combines traditional cryptographic methods with an innovative approach to data encoding, providing a unique solution for secure file handling. By utilizing the strategic complexity of chess, the application introduces a new dimension to steganography and encryption, making it a formidable tool for data protection. The multi-layered security framework, which includes AES and RSA encryption as well as OTP-based authentication, ensures that user

data is not only encrypted but also disguised effectively, reducing the likelihood of detection and decryption by unauthorized parties.

The project showcases the potential of merging creative elements like chess with cryptographic techniques to enhance data security. This approach not only offers robust encryption but also obfuscates the nature of the encrypted data, providing an additional layer of protection. The use of PGN files as the medium for encoded data disguises sensitive information as standard chess game records, making it less likely to attract attention or be flagged by security systems.

Overall, *Queen's Gambit* represents a promising step forward in the field of cryptography, introducing innovative methods to address modern cybersecurity challenges. With future enhancements like cloud integration, mobile platform support, and more advanced cryptographic protocols, the application could evolve into a comprehensive tool for secure data encryption and transmission, appealing to a wide range of users and use cases.

Key Points:

- **Innovative Approach**: Combines chess-based encoding with AES-RSA encryption, creating a unique and secure data protection method.
- **Effective Obfuscation**: Encodes data in PGN files, disguising it as chess games and reducing the risk of detection.
- **Multi-Layer Security**: Utilizes OTP authentication and dual-layer encryption for robust protection against unauthorized access.
- **Future Potential**: With enhancements like cloud integration and platform expansion, the application has the potential to serve broader audiences and use cases.

Submitted by

Name	Registration number	Roll Number	Semester & Branch	Section
Aniruddh Ballal	220911256	17	V (IT)	IT - C