



# **Secure File Exchange Platform: Enabling Seamless File Sharing**

*MINI PROJECT REPORT submitted in partial fulfillment of the requirements for  
the Award of the Degree of*

**BACHELOR OF TECHNOLOGY**

**In**

**INFORMATION TECHNOLOGY**

**By**

**Gutla Jessica (218W1A1217)**

**Bhanu Sri Sai Someshu Thunuguntla(228W5A1208)**

*Under the Guidance of*

**Y.Sandeep**

**Assistant Professor**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**V R SIDDHARTHA ENGINEERING COLLEGE**

**(AUTONOMOUS - AFFILIATED TO JNTU-K, KAKINADA)**

**Approved by AICTE & Accredited by NBA**

**KANURU, VIJAYAWADA-520007**

**ACADEMIC YEAR**

**(2023-24)**

# V.R. SIDDHARTHA ENGINEERING COLLEGE

(Affiliated to JNTUK: Kakinada, Approved by AICTE, Autonomous)

(An ISO certified and NBA accredited institution)

Kanuru, Vijayawada – 520007



## CERTIFICATE

This is to certify that this project report titled “**Secure File Exchange Platform: Enabling Seamless File Sharing**” is a bonafide record of work done by **G.Jessica(218W1A1217)** and **T.B.S.S.Somesu(228W5A1208)** under my guidance and supervision is submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology, **V.R. Siddhartha Engineering College** (Autonomous under JNTUK) during the year 2023-24.

**Y.Sandeep**

Assistant Professor

Dept. of Information Technology

**(Dr. M. Suneetha)**

Professor & Head

Dept. of Information Technology

## EXTERNAL EXAMINER SIGNATURE

*Date of examination:*

## ACKNOWLEDGEMENT

First and foremost, I sincerely salute our esteemed institution **V.R SIDDHARTHA ENGINEERING COLLEGE** for giving me this opportunity for fulfilling my EPICS project. I am grateful to our principal **Dr. A.V.RATNA PRASAD**, for his encouragement and support all through the way of my project.

On the submission of this Project report, I would like to extend my honor to **Dr. M.Suneetha**, Head of the Department, IT for her constant motivation and support during the course of my work. I feel glad to express my deep sense of gratefulness to my project guide **Y.Sandeep, Assistant Professor** for his guidance and assistance in completing this project successfully.

I would also like to convey my sincere indebtedness to all faculty members, including supporting staff of the Department, friends and family members who bestowed their great effort and guidance at appropriate times without which it would have been very difficult on my part to finish the project work.

## DEPARTMENT OF INFORMATION TECHNOLOGY

### V.R.SIDDHARTHA ENGINEERING COLLEGE

#### PROJECT SUMMARY

S.No	Item	Description
1	Project Title	Secure File Exchange Platform: Enabling Seamless File Sharing.
2	Student Names & Numbers	G.JESSICA (218W1A1217) Bhanu Sri Sai Someshu Thunuguntla(228W5A1208)
3	Name of The Guide	Y.SANDEEP
4	Research Group	BATCH - 3
5	Application Area	Security field
6	Aim of the Project	The aim of this project is to develop a secure file-sharing web application with multiple options. By implementing token generation, authentication mechanisms, the system ensures secure access and transfer of files.
7	Project Outcomes	A working web application ensuring file management securely through token generation.

Student Signatures

1.

2.

Signature of the Guide

## TABLE OF CONTENTS

ACKNOWLEDGEMENT	iii
PROJECT SUMMARY	iv
LIST OF FIGURES	v
LIST OF TABLES	vi
LIST OF EQUATIONS	vii
ABSTRACT	viii
<b>CHAPTER-1 Introduction</b>	<b>1</b>
1.1 Origin of the Problem	1
1.2 Basic definitions and Background	1
1.3 Problem Statement	3
1.4 Applications of Proposed work	3
<b>CHAPTER-2 Review of Literature</b>	<b>4</b>
2.1Description of Existing Systems	4
2.2 Summary of Literature Study	5
<b>CHAPTER-3 Proposed Method</b>	<b>8</b>
3.1Design Methodology	8
3.2System Architecture Diagram	10
3.3Description of Algorithms	11
3.4 Description of Datasets and Tools	15
<b>CHAPTER-4 Results and Observations</b>	<b>17</b>
4.1Stepwise description of Results	17
4.2Test case results/Result Analysis	18
4.3Observations from the work	22
<b>CHAPTER-5 Conclusion and Future work</b>	<b>23</b>
5.1 Conclusion	23
5.2 Future study	23
<b>References</b>	<b>24</b>

## LIST OF FIGURES

<b>Figure</b>	<b>Page</b>
Figure 3.1 Architecture diagram	10
Figure 3.2 Digital Signature Architecture	13
Figure 3.3 RSA Algorithm	13
Figure 3.4 AES Architecture	14
Figure 3.5 Visual Studio code	15
Figure 3.6 MongoDB Compass	16
Figure 4.1 APP.py	17
Figure 4.2 Files in the client folder	17
Figure 4.3 Login Authentication	18
Figure 4.4 Start page	19
Figure 4.5 Functionalities	19
Figure 4.6 Send file	20
Figure 4.7 Download file	19
Figure 4.8 Forward file	20
Figure 4.9 Generated json tokens	20
Figure 4.10 Types of routes	21

## LIST OF TABLES

Table Name	Page
Table 2.1 Literature review	4
Table 4.1 Results	22

## LIST OF EQUATIONS

### Equation

### Brief Description

3,3.1 :Algorithm steps

components



## ABSTRACT

The development of a web-based file-sharing application using modern technologies addresses the challenge of securely managing and distributing files, including large files, within digital environments. Leveraging React.js for the frontend and integrating with backend APIs, the project focuses on data security and user experience. Key functionalities include secure file sharing to recipients stored in a database and the ability to handle large file transfers using system local storage. The application enables seamless collaboration and distribution of digital assets while maintaining confidentiality and integrity. This report details the design, implementation, and evaluation of the Secure File Exchange application, highlighting its architecture and utilization of advanced technologies. Future enhancements and potential applications are discussed, underscoring the project's significance in secure digital asset management and efficient file distribution.

**Keywords:** Secure File Sharing, React.js, Web Application, Backend APIs, Data Security, Large File Transfer, Database Integration, Digital Asset Management

# **CHAPTER – 1 INTRODUCTION**

In today's digital landscape, the need for secure file transfer solutions is paramount due to increasing cyber threats and privacy concerns. A secure file transfer website provides a platform for users to exchange files safely over the internet, mitigating risks associated with data breaches and unauthorized access. This type of website employs robust encryption protocols, user authentication mechanisms, and secure storage practices to ensure data confidentiality and integrity. By implementing stringent security measures, such as token-based authentication and end-to-end encryption, secure file transfer websites offer users a reliable and protected environment for sharing sensitive information. These platforms cater to diverse industries, including healthcare, finance, and legal, where data privacy and compliance are critical considerations.

## **1.1 Origin of the Problem:**

The origins of the problem of secure file transfer stem from the increasing reliance on digital communication and the growing volume of sensitive information exchanged online. With data breaches and cyberattacks on the rise, organizations and individuals face significant risks when sharing files over insecure channels. Traditional methods of file transfer, such as email attachments or unencrypted file uploads, lack adequate security measures, exposing data to interception and unauthorized access. This problem underscores the need for robust solutions that prioritize data privacy and security in file exchange processes.

## **1.2 Basic definitions and Background:**

### **1.2.1 Secure File Transfer:**

Secure file transfer refers to the process of exchanging files between parties while ensuring confidentiality, integrity, and availability of the transferred data. It involves utilizing encryption, authentication, and other security measures to protect files from unauthorized access and interception during transmission.

### **1.2.2 Token-Based Authentication**

One of the main tools used in developing android applications, as it packages many core features into one SDK and it can be used in the application easily. This helps us to avoid writing lot of code, and building applications faster.[7]

### 1.2.3 JSON Web Token

**1. Header:** The header typically consists of two parts:

- Type (typ): Specifies the token type, which is usually "JWT".
- Algorithm (alg): Indicates the cryptographic algorithm used to generate the signature, such as HMAC SHA256 or RSA.

**Payload:** The payload (or claims) contains the data being transmitted, which can include user information, permissions, and other attributes. The payload consists of a set of claims, which are statements about the entity (user) and additional metadata. There are three types of claims:

**Registered Claims:** Pre-defined claims such as iss (issuer), sub (subject), exp (expiration time), iat (issued at time), etc.

**Public Claims:** Custom claims defined by the application developer.

**Private Claims:** Custom claims agreed upon between parties but not registered.

**Signature:** The signature is generated using the header, payload, a secret key (for symmetric algorithms), or a private key (for asymmetric algorithms). The signature ensures the integrity of the JWT and allows verification of the sender.

### 1.2.4 RSA 256

The term "RSA-256" typically refers to the RSA algorithm used with a 256-bit key size. RSA (Rivest-Shamir-Adleman) is an asymmetric cryptographic algorithm named after its inventors: Ron Rivest, Adi Shamir, and Leonard Adleman. It is widely used for secure communication, digital signatures, and key exchange in various applications. This key size determines the security level of RSA, with larger key sizes generally providing stronger security against attacks based on factorization of the modulus  $n$ . In the case of RSA-256, the modulus  $n$  used in the RSA algorithm is typically a product of two prime numbers, each around 128 bits in length.

### 1.2.4 Route & Schema

A route definition in web development specifies how an incoming HTTP request is handled by the server. It typically consists of a URL path (endpoint) and associated HTTP methods (GET, POST, PUT, DELETE) that determine the actions performed on the server.

A schema definition in database design describes the structure and constraints of data stored in a database. It defines the fields (attributes) and their types, validations, default values, and relationships between data entities.

### **1.3 Problem statement**

In response to the pressing need for a secure and efficient file transfer solution, our project targets the challenges encountered during large file exchanges like Gmail. These challenges include restrictive file size limitations, and heightened security risks. By addressing these issues head-on, our goal is to develop a robust web application that facilitates seamless and secure file sharing among internal teams and external partners.

### **1.4 Applications of Proposed work :**

The proposed work on developing a secure file transfer web application has practical applications across various industries and sectors. It can be utilized by businesses to securely exchange confidential documents, facilitate collaboration among remote teams, and comply with data protection regulations, ensuring data privacy and integrity in file sharing processes..

## CHAPTER –2 REVIEW OF LITERATURE

This chapter describes the review of literature that we have taken from various papers and considered all the points mentioned in the papers.

### 2.1 Description of Existing Systems:

The literature review of numerous publications is shown in the table below, along with explanations of the observations we made from each paper.

**Table 2.1:** Literature Review

Publication Details	Sample Details	Model/ Technique Used	Purpose and Evaluation Metrics	Future Scope/Remarks
J. Liu Et Al Design and Implementation of a Secure Peer-to-Peer File Transfer System on IP Multicast and Diffie-Hellman 2022 IEEE  4 <sup>th</sup> Conference	peer-to-peer file transfer in LAN , On basis of IP multicast,	Diffie-Hellman algorithm, TEA algorithm	peer-to-peer file transfer system based on IP multicast and Diffie-Hellman effectively realize the securetransfer of peer-to-peer files in LAN	Further research can focus on optimizing the performance of the file transfer system
R. Mathwale Et Al Blockchain Based Inter-Organizational Secure File Sharing System  2023 IEEE INOCON	Decentralized Approach Ensuring Transparency, Accountability, and Data Integrity in Collaborative Environments.	blockchain-based secure file-sharing system	evaluated in terms of security, performance, and usability.	Provided a review a how recognition of notes can be done.

Mohammadpayam Almasian Et Al Secure cloud file sharing scheme using blockchain and attribute-based encryption  science-direct 2023	Ethereum blockchain with AES-based attribute-based encryption for secure cloud file sharing	Blockchain technology, Attribute-based encryption (ABE) for fine-grained access control	secure solution file sharing in cloud environments, ensuring confidentiality, integrity, and access control.	support for dynamic access policies, and interoperability
Shaoliang Peng a b Et Al A peer-to-peer file storage and sharing system based on consortium blockchain science direct 2022	Implementation on Hyperledger Fabric.	Consortium Blockchain with Permissioned Access Control;	Security, Scalability, Transaction Throughput, and Consistency;	Exploration of Privacy-Enhancing Techniques and Optimization for Large-Scale Deployment
H. Wang, R. Shea, F. Wang and J. Liu, "On the impact of virtualization on Dropbox-like cloud file sharing/ storage/synchronization services	Secure file share and also reliable file storage for peer to peer.	Algorithm to optimize task allocation across cloud instances	Security analysis, Synchronization delay reduction in Dropbox system.	Enhancing scalability and reducing synchronization delays in cloud-based file hosting services.

## 2.2 Summary of Literature Study:

[1] A blockchain-based system for secure and transparent file-sharing among a consortium of organizations. By leveraging Hyperledger Fabric, an enterprise-grade blockchain framework, the system sets up a decentralized network and develops smart contracts to manage transactions. The Inter Planetary File System (IPFS) is employed to store files in a distributed manner, enhancing accessibility and redundancy. The system also outlines a detailed workflow for identity management and file-sharing processes, ensuring that files are exchanged with confidentiality,

integrity, and availability. This approach offers a solution to the limitations of centralized systems by enabling distributed trust and transparent operations within the consortium.

**[2]** A collaborative approach to secure cloud file sharing utilizing blockchain and attribute-based encryption (ABE). Blockchain facilitates access control via smart contracts between data owners and users, ensuring decentralized and fault-tolerant security against DoS attacks. Each data owner establishes a smart contract, enabling users to request file access by registering transactions. The owner then provides necessary credentials for decryption, embedded within an access polynomial attached to the encrypted file's metadata. ABE is employed to enforce access policies while maintaining user anonymity. The scheme allows swift user access revocation without communication overhead to unaffected users. Formal verification confirms security regarding credential secrecy and participant authentication. Evaluation demonstrates scalability, supporting up to 20,000 users with acceptable performance.

**[3]** A secure peer-to-peer file transfer system for LANs has been developed using IP multicast for node discovery. Nodes mutually discover each other through a designed protocol stack based on IP multicast. Secure connections between nodes are established using the Diffie-Hellman algorithm, which facilitates the negotiation of a shared key. This key is then used to encrypt communication data via the TEA symmetric encryption algorithm. Experimental results demonstrate that nodes can successfully discover each other and securely transfer files over the LAN. Tests confirm that nodes are accurately discovered and can safely exchange files under this protocol. This system effectively addresses certain network security challenges in peer-to-peer file transfers within LAN environments.

**[4]** A secure peer-to-peer file transfer system for LANs has been developed using IP multicast for node discovery. Nodes mutually discover each other through a designed protocol stack based on IP multicast. Secure connections between nodes are established using the Diffie-Hellman algorithm, which facilitates the negotiation of a shared key. This key is then used to encrypt communication data via the TEA symmetric encryption algorithm. Experimental results demonstrate that nodes can successfully discover each other and securely transfer files over the LAN. This system effectively addresses certain network security challenges in peer-to-peer file transfers within LAN environments.

**[5]** Centralized data storage systems, we introduce a peer-to-peer storage system with identity access. Utilizing a consortium blockchain, our solution addresses issues of data validation, cross-organizational retrieval, trusted authorization, and secure sharing. This system combines a peer-to-peer storage scheme with identity authentication mechanisms compatible with the consortium blockchain. Furthermore, we present a blockchain-based permission control scheme alongside retrieval, authorization, and sharing processes. Implementation and testing validate the feasibility of our approach, offering a promising solution for reliable and secure data storage and sharing in the era of big data.

**[6]** Introduce an enhanced protection method for files uploaded to services like Dropbox and Google Drive. Our approach employs double encryption, utilizing AES followed by RSA algorithms, to significantly increase security. Keys are dynamically generated during execution, ensuring robust encryption. Parameters such as security level, speed, data confidentiality, integrity, and ciphertext size are all addressed, surpassing conventional methods. The encrypted files are stored on Dropbox, providing a secure repository for sensitive data. This approach represents a significant advancement in file protection for cloud-based storage solutions, offering users peace of mind regarding the safety of their data. To encrypt the file that we upload in cloud, we make use of Double encryption technique. The file is being encrypted twice using the two algorithms one after the other. The file is first encrypted using AES algorithm and then by RSA algorithm.



## CHAPTER – 3

### PROPOSED METHOD

This chapter describes the architecture diagram and algorithms that we have applied for our project to securely transfer files.

#### 3.1 Design Methodologies:

In this project we used Rivet shemmier Adleman algorithm to transfer the file from one user of the database to other securely by performing various security mechanisms Then we built aa web application which acts as an interface for the users.

The overall methodology is divided into following steps:

**Step 1:** Requirement Analysis

**Step 2:** System Architecture Design

**Step 3:** JSON Token Authentication

**Step 4:** User Interface Design

**Step 5:** Backend Development

**Step 6:** Database Design

**Step 7:** Deployment and Maintenace

**Step 8:** Run and test Application

#### **Step 1 – Requirement Analysis**

The requirement analysis phase involves understanding user needs, security concerns, and functional specifications for the file transfer system. Key requirements include secure authentication methods,, support for file transfers, user-friendly interface.By gathering and analyzing these requirements, the project aims to define clear objectives and deliverables for developing a robust and user-centric file sharing platform.

#### **Step 2 – System Architecture Design**

The secure file transfer web application follows a client-server architecture, with a frontend built using React.js for the user interface. The backend comprises Node.js APIs for handling file operations, user authentication, and database interactions. A relational database (e.g., PostgreSQL) is used to store user data, file metadata, and access controls, ensuring scalability and data integrity in file management processes. This architecture emphasizes security, modularity, and scalability to support secure and efficient file transfers.

### **Step 3 – JSON Token Authentication**

JSON Web Token (JWT) authentication is a method of securely transmitting information between parties as a JSON object. It consists of three parts: a header, a payload, and a signature. The token is generated by the server upon successful authentication and is included in subsequent requests to authenticate and authorize users without needing to store session state on the server, enhancing scalability and security.

### **Step 4 – User Interface Design**

User Interface (UI) Design for the secure file transfer web application emphasizes simplicity, intuitiveness, and accessibility. The design prioritizes a clean and modern interface with intuitive navigation, clear call-to-action buttons for file operations, and responsive layout for seamless user interaction across devices. Visual elements such as file previews, progress indicators, and error messages enhance the user experience, ensuring efficient and secure file exchange workflows.

### **Step 5 -Backend Development**

Backend Development for the secure file transfer web application involves implementing server-side logic using Node.js or similar frameworks. This includes creating APIs to handle file uploads, downloads, user authentication, and database interactions for user management and file storage. The backend is responsible for processing requests from the frontend, enforcing security measures, and ensuring efficient data management and retrieval.

### **Step 6 -Database Design**

Database Design for the secure file transfer web application involves creating a relational database schema to store user information, file metadata, and access permissions. The schema includes tables for users, files, and user-file relationships, utilizing appropriate indexing and normalization techniques for efficient data retrieval and management. Access control mechanisms are implemented to ensure data integrity and enforce security policies, enabling secure and scalable storage of files and associated metadata.

### **Step 7: Deployment and Maintenance**

Centralized data storage systems, we introduce a peer-to-peer storage system with identity access. Utilizing a consortium blockchain, our solution addresses issues of data validation, cross-organizational retrieval, trusted authorization, and secure sharing. This system combines a peer-to-peer storage scheme with identity authentication mechanisms compatible with the consortium blockchain. Furthermore, we present a blockchain-based permission control scheme alongside retrieval, authorization, and sharing processes. Implementation and testing validate the feasibility

of our approach, offering a promising solution for reliable and secure data storage and sharing in the era of big data.

### Step 8: Run and test multiple functionalities

To ensure the robustness and effectiveness of the secure file transfer web application, it is crucial to conduct comprehensive testing across multiple functionalities.

Functional Testing: Validate core functionalities such as file uploading, downloading, and sharing to ensure they work as expected and meet user requirements

## 3.2 System Architecture Diagram :

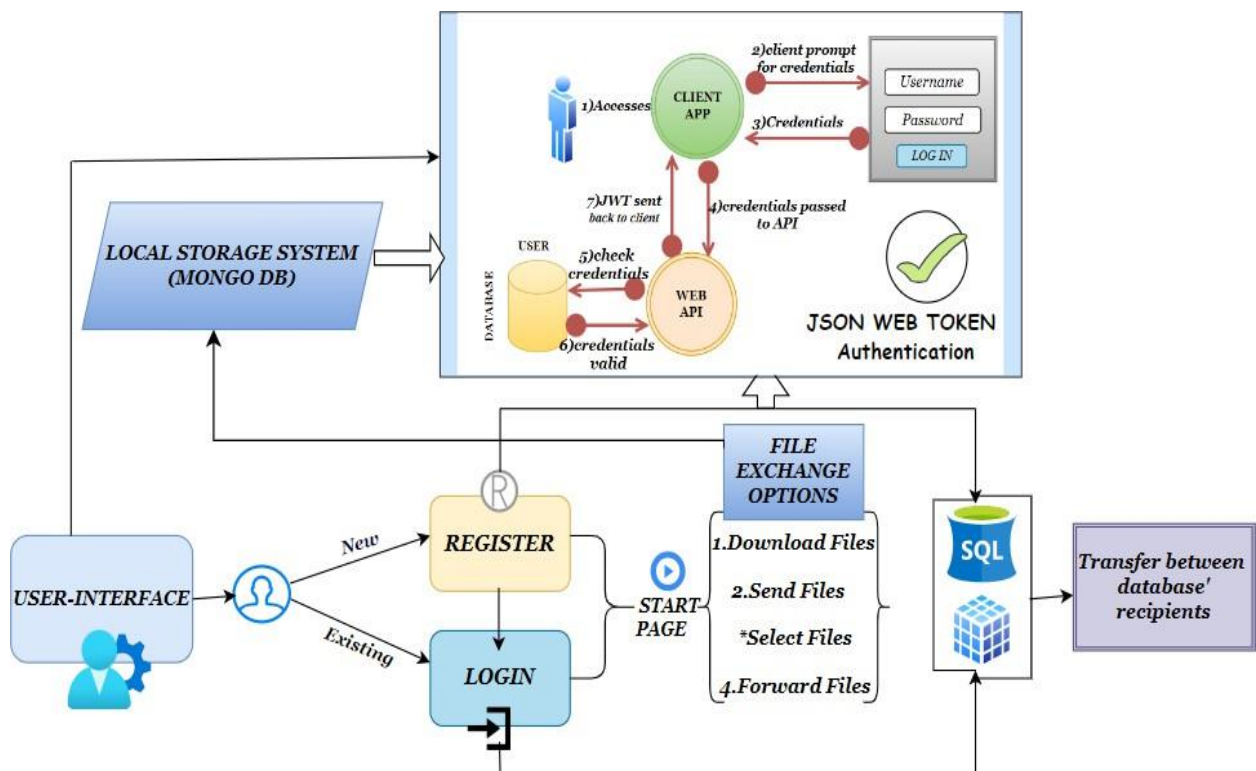


Figure 3.1 : Architecture Diagram

## Explanation :

The flow diagram for the secure file transfer web application begins with a user interface featuring login verification and options for new user registration or existing user login. Upon successful verification using JSON Web Token (JWT) with RSA 256 algorithm, authenticated users are redirected to the start page, offers multiple functionalities including file download, send, and forward within the local storage system, ensuring secure and efficient file management. The use of JWT with RSA 256 algorithm enhances security by generating and validating tokens to authenticate users securely. The login verification process involves validating user credentials against stored data in the system, and upon successful authentication, users gain access to the application's main functionalities. Files can be securely downloaded, sent to recipients, or forwarded, leveraging local storage for efficient handling of large file transfers. Overall, the flow diagram depicts a streamlined user journey from login verification to accessing various file management functionalities, underpinned by robust security measures such as JWT-based authentication with RSA encryption. This design ensures a secure, user-friendly experience for managing and sharing files within the application.

## 3.3 Description of Algorithms :

### 3.3.1 : Rivest-Shamir-Adleman RSA256

#### Key Generation

You need to generate public and private keys before running the functions to generate your ciphertext and plaintext. They use certain variables and parameters, all of which are explained below:

- Choose two large prime numbers ( $p$  and  $q$ )
- Calculate  $n = p * q$  and  $z = (p-1)(q-1)$
- Choose a number  $e$  where  $1 < e < z$
- Calculate  $d = e^{-1} \bmod (p-1)(q-1)$
- You can bundle private key pair as  $(n, d)$
- You can bundle public key pair as  $(n, e)$

## Encryption/Decryption Function

Once you generate the keys, you pass the parameters to the functions that calculate your ciphertext and plaintext using the respective key.

- If the plaintext is  $m$ , ciphertext =  $me \bmod n$ .
- If the ciphertext is  $c$ , plaintext =  $cd \bmod n$

## Advantages of RSA



No need of sharing  
secret keys



Proof of owner's  
authenticity



Faster Encryption  
than DSA



Data can't be  
modified in transit

- **No Key Sharing:** RSA encryption depends on using the receiver's public key, so you don't have to share any secret key to receive messages from others.
- **Proof of Authenticity:** Since the key pairs are related to each other, a receiver can't intercept the message since they won't have the correct private key to decrypt the information.
- **Faster Encryption:** The encryption process is faster than that of the DSA algorithm.
- **Data Can't Be Modified:** Data will be tamper-proof in transit since meddling with the data will alter the usage of the keys. And the private key won't be able to decrypt the information, hence alerting the receiver of manipulation.

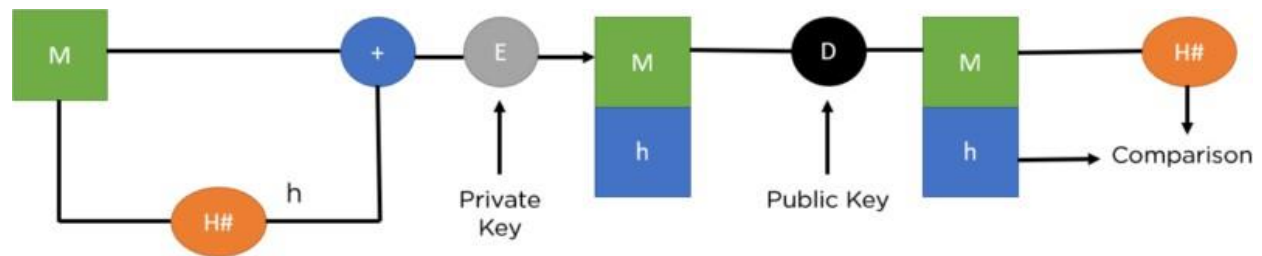


Figure 3.2 : Digital Signature Architecture

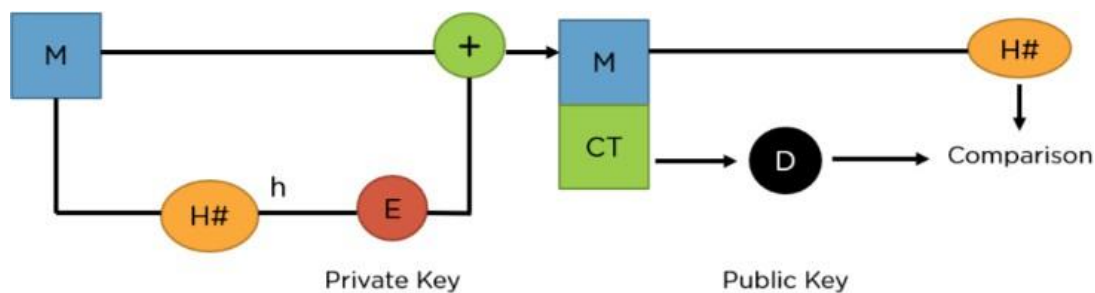


Figure 3.3 : Proposed RSA Algorithm

### 3.3.2 Advanced Encryption Standard:

**1. Key Expansion:** The original key is expanded into a key schedule, which generates round keys for each round of encryption.

**2. Initial Round:** AddRoundKey operation is performed where each byte of the state matrix is combined with the round key using bitwise XOR.

**3. Rounds:** A certain number of rounds are executed based on the key size (10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys). Each round consists of the following operations:

- **SubBytes:** Non-linear substitution step where each byte of the state matrix is replaced with another byte using a substitution table (S-box).

- **ShiftRows:** Bytes in each row of the state matrix are shifted cyclically to the left by different offsets.
- **MixColumns:** Each column of the state matrix is multiplied with a fixed polynomial modulo an irreducible polynomial. This operation provides diffusion across the bytes.
- **AddRoundKey:** Each byte of the state matrix is combined with the round key using bitwise XOR.

4. **Final Round:** Similar to the rounds but without the MixColumns step.

5. **Output Transformation:** The final state matrix is serialized into a byte array to produce the ciphertext.

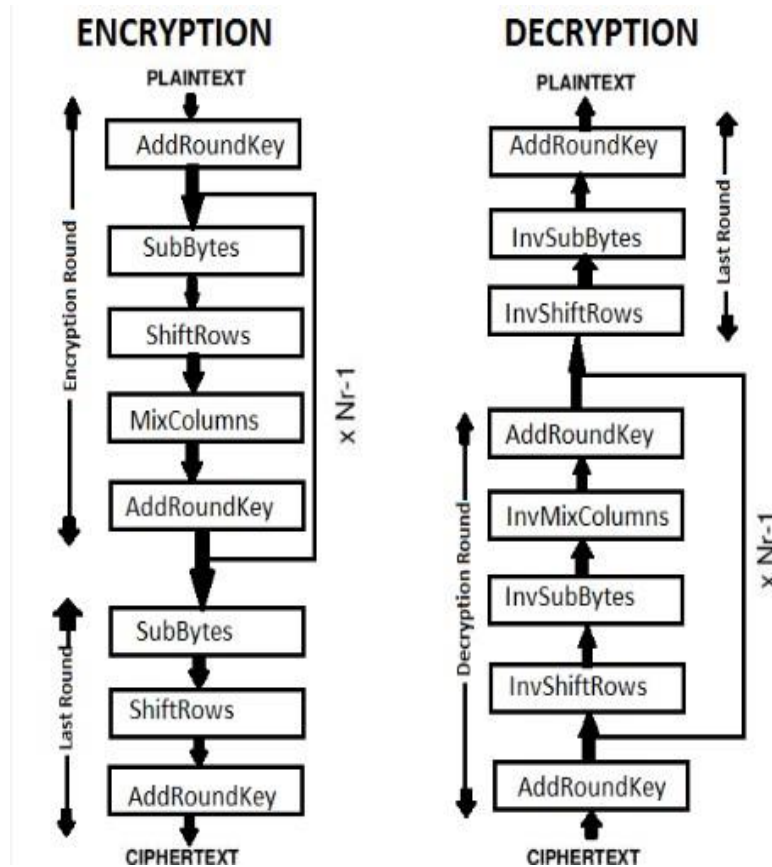


Figure 3.4 : AES Architecture

### 3.4 Description of requirements and Tools :

#### 3.4.1 Visual Studio Code :

Visual Studio is a comprehensive integrated development environment (IDE) developed by Microsoft, widely used by developers for building various types of software applications. Versions 2010 through 2019 have been released, each introducing new features and enhancements to streamline the development process. Key features of Visual Studio include a powerful code editor with syntax highlighting, IntelliSense, and code refactoring capabilities, as well as a robust debugger for efficient debugging of applications. Visual Studio supports multiple programming languages such as C#, Visual Basic .NET, C++, JavaScript, TypeScript, HTML, CSS, and Python, providing developers with flexibility and choice. The IDE offers visual design tools for building user interfaces, including Windows Forms, WPF (Windows Presentation Foundation), ASP.NET Web Forms, and XAML-based applications. Visual Studio integrates with Microsoft's collaboration platform, Azure DevOps (formerly Visual Studio Team Services), enabling teams to collaborate on projects, manage source code, track work items, and automate build and release processes.

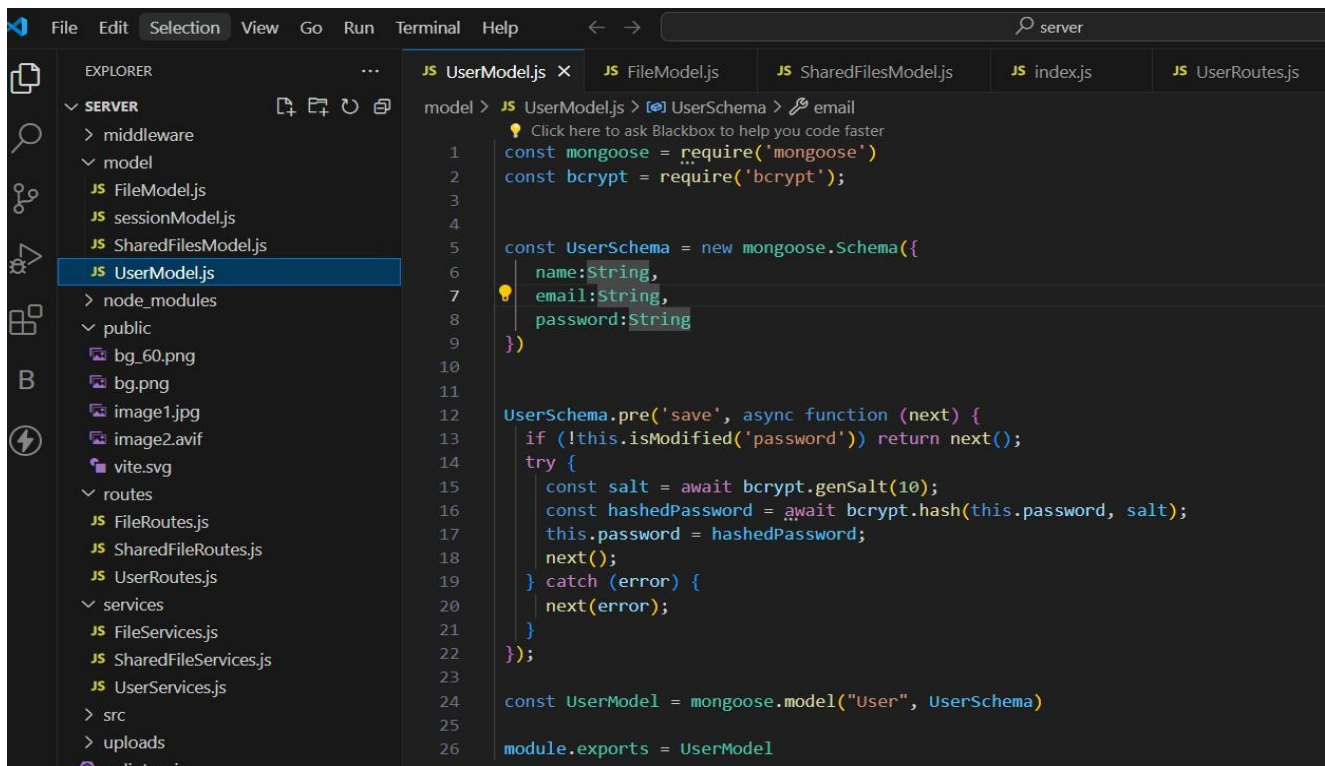


Figure 3.5 : Visual Studio Code



### 3.4.2 Mongo DB Compass :

MongoDB Compass is a graphical user interface (GUI) tool designed to interact with MongoDB databases. It provides an intuitive interface for visually exploring and interacting with MongoDB data, making database management tasks more accessible to developers and administrators. MongoDB Compass allows users to perform various operations on MongoDB databases, collections, and documents, including querying, inserting, updating, and deleting data. The tool features a rich set of capabilities such as schema visualization, aggregation pipeline building, and index management, enhancing the productivity and efficiency of working with MongoDB. With MongoDB Compass, users can easily navigate database structures, analyze data relationships, and execute complex queries using a point-and-click interface. The tool also supports real-time data monitoring and provides performance insights through visual metrics and charts. MongoDB Compass is a valuable tool for both beginners and experienced MongoDB users, offering a convenient way to interact with MongoDB databases and streamline database management tasks within a comprehensive and user-friendly environment.

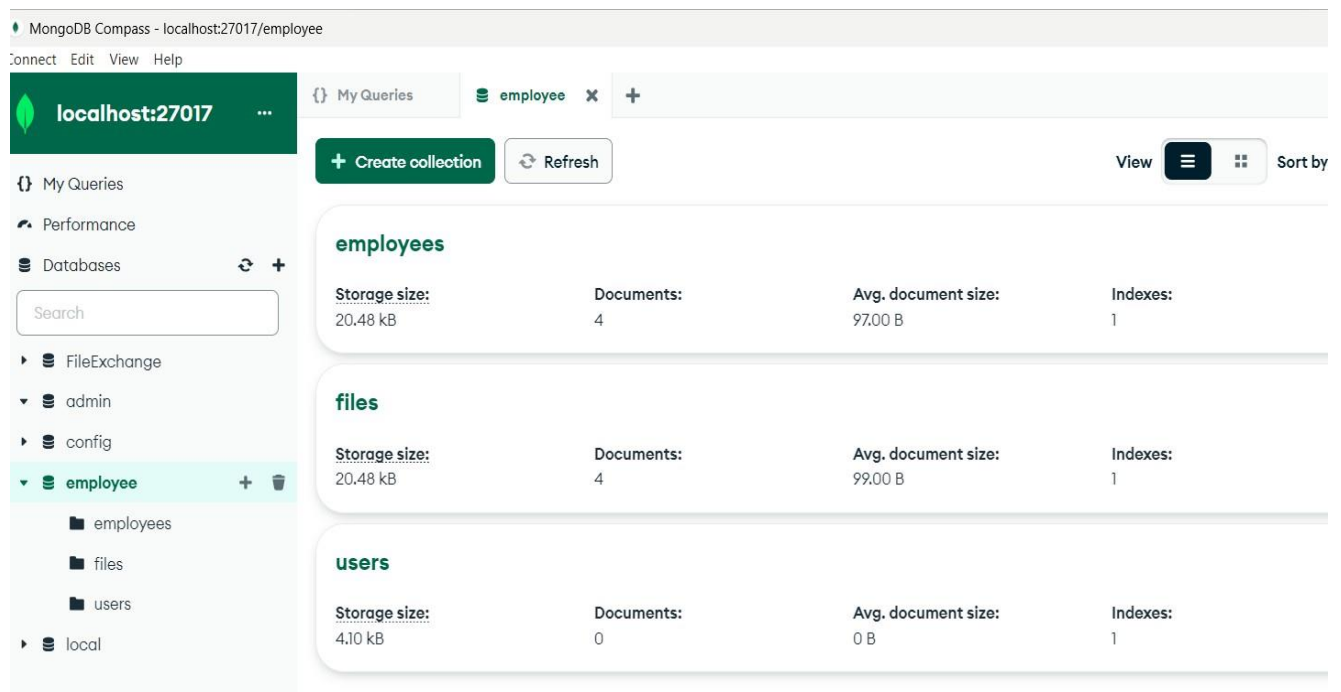


Figure 3.6 : MongoDB Compass

## CHAPTER – 4 RESULTS AND OBSERVATIONS

This chapter describes the results of our project and comparison of two algorithms that we have used in our project to manage file transfer.

### 4.1 Stepwise description of Results:

Initially, we imported all the components required for the web application in app.py

#### Implementation Code

```
src > App.jsx > App
 8  import Start from './start'
 9  import SendFiles from './sendFiles'
10  import DownloadComponent from './download'
11  import Forward from './Forward'
12
13  function App() {
14
15    return (
16      <BrowserRouter>
17        <Routes>
18          <Route path='/register' element={<Signup/>}/>
19          <Route path='/login' element={<Login/>}/>
20          <Route path='/home' element={<Home />}/>
21          <Route path='/start' element={<Start/>}></Route>
22          <Route path='/sendFiles' element={<SendFiles/>}></Route>
23          <Route path='/downloadFiles' element={<DownloadComponent/>}></Route>
24          <Route path='/forwardFiles' element={<Forward/>}></Route>
25        </Routes>
26      </BrowserRouter>
27    )
28  }
29
30  export default App
31
```

Figure 4.1 : App.py

Files for web application

download.jsx	# start.css
Files.jsx	start.jsx
Forward.jsx	.eslintrc.cjs
# Home.css	.gitignore
Home.jsx	JS config.js
# Login.css	<> index.html
Login.jsx	{ } package-lock.json
main.jsx	{ } package.json
Navbar.jsx	i README.md
sendFiles.jsx	JS vite.config.js
Sidebar.jsx	
Signup.jsx	

Figure 4.2 : files in client folder

## 4.2 Result Analysis

### Login Authentication

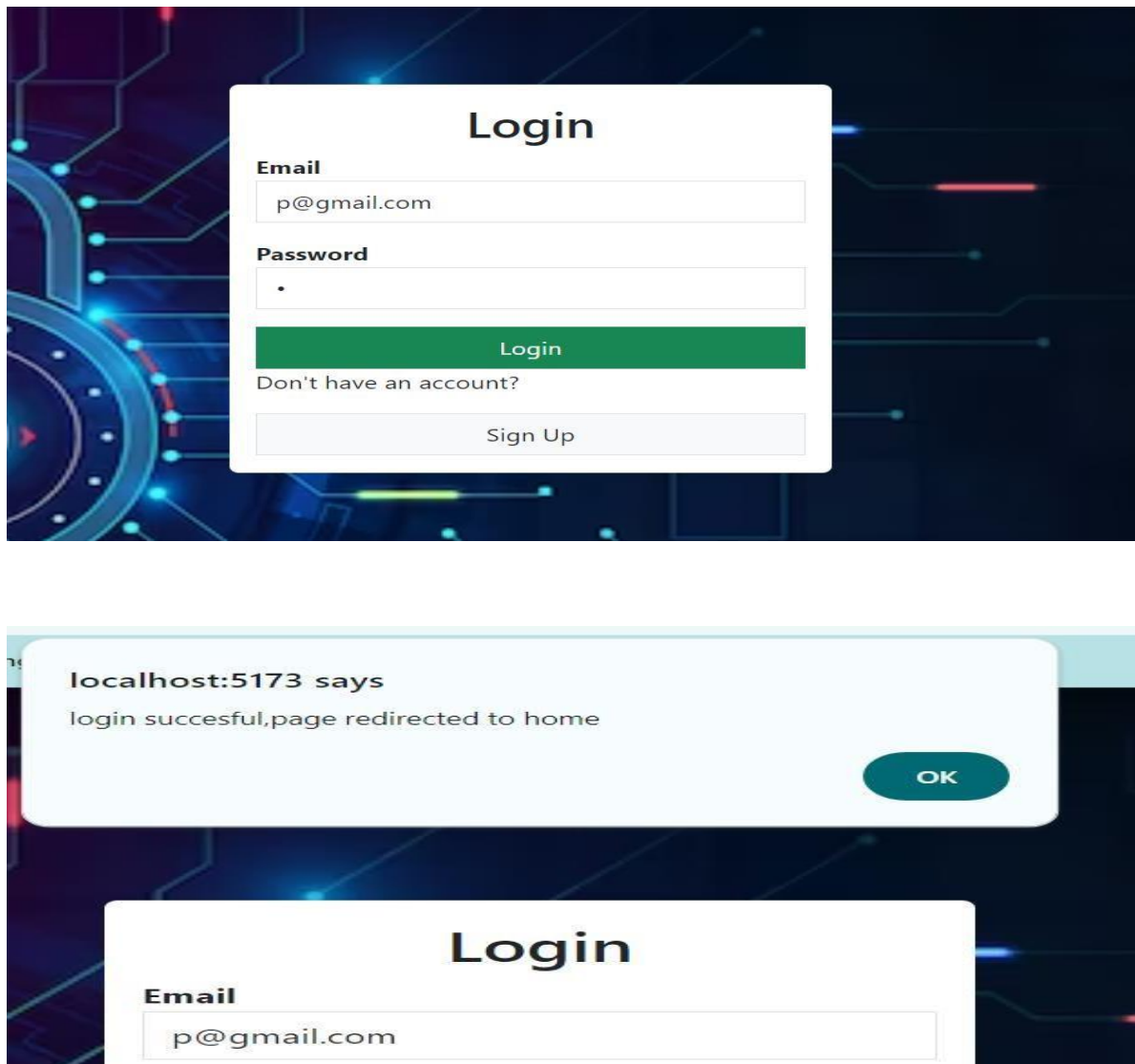


Figure 4.3 : login Authentication

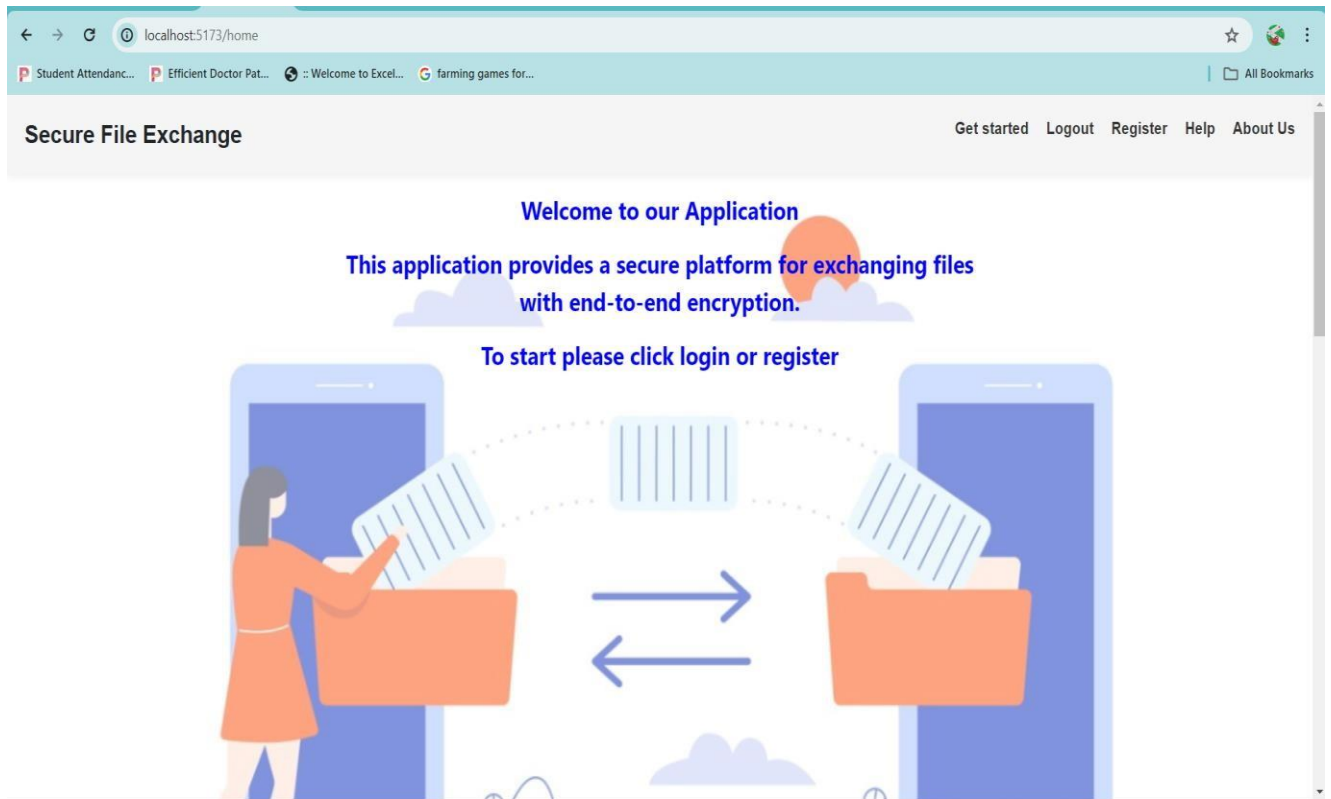
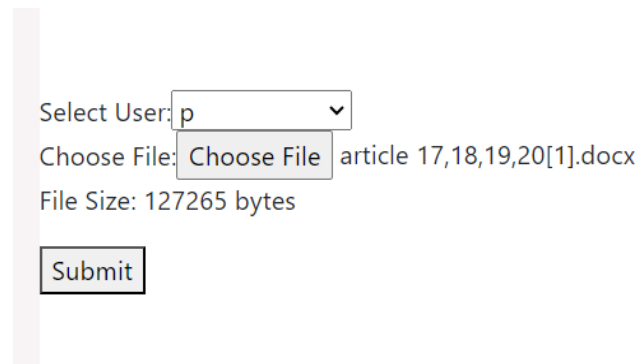


Figure 4.4 : Start page after authentication

### Functionalities:



Figure 4.5 : Functionalities



Select User: p

Choose File: Choose File article 17,18,19,20[1].docx

File Size: 127265 bytes

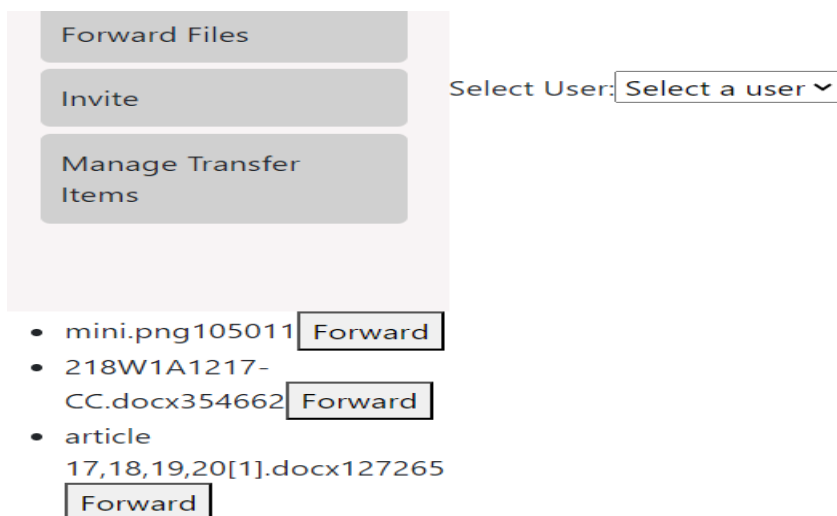
Submit

Figure 4.6 :Sending a file

## Available Files for Download

- mini.png105011 Download
- 218W1A1217-CC.docx354662 Download
- article 17,18,19,20[1].docx127265 Download

Figure 4.7 : Download the files



Forward Files

Invite

Manage Transfer Items

Select User: Select a user

- mini.png105011 Forward
- 218W1A1217-CC.docx354662 Forward
- article 17,18,19,20[1].docx127265 Forward

Figure 4.8 : Forward a file

FileExchange > users

Documents 2

Aggregations

Schema

Indexes 1

Validation

Type a query: { field: 'value' } or [Generate query](#)

+ ADD DATA

EXPORT DATA

UPDATE

DELETE

```
{
  "_id": "66260c2ca91d97d88a4ea6c1",
  "name": "p",
  "email": "p@gmail.com",
  "password": "$2b$10$KD0vCjNjC2Pv5BG5gXhgo.XjL6ocrRsWn03Ry2l0jH9I/4p8wKtT6",
  "__v": 0
}
```

```
{
  "_id": "66260cfca820e570eebb09f0",
  "name": "j",
  "email": "j@gmail.com",
  "password": "$2b$10$0BPWHvnWwa2hIcz8irAm20LdnRb8q3zngjFLHjLk0/2MPXX0VBBx6",
  "__v": 0
}
```

```
{
  "_id": "662737896bdb86fade0edce7",
  "name": "bob",
  "email": "bob@gmail.com",
  "password": "$2b$10$0J5TyhJgCFumVzUpN.yQvekmwgBy1xoKnMKTi4/hLfCorf08gNKJK",
  "__v": 0
}
```

Figure 4.9 : Generated Json tokens

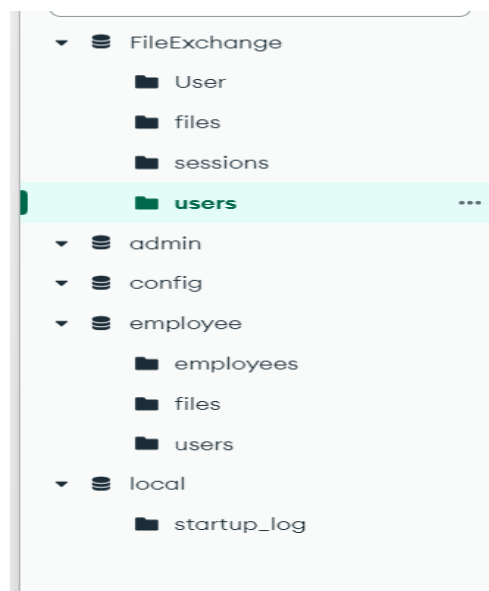


Figure 4.10 : Types of routes

### **4.3 Observations**

Observing the entire project, several key insights emerge. Firstly, the project demonstrates a robust use of modern technologies and methodologies, evident in the adoption of JWT-based authentication for secure access control. Secondly, the incorporation of AES encryption ensures data confidentiality during file exchanges, aligning with industry best practices for data protection. Additionally, the project showcases a well-structured architecture with clear separation of concerns, facilitating scalability and maintainability. The emphasis on security measures and secure file transfer protocols underscores a commitment to safeguarding sensitive information. Moreover, the project's user-centric design and intuitive interface enhance usability and user experience, promoting efficient collaboration and streamlined workflows. Overall, the project exemplifies a comprehensive approach to secure file exchange, integrating cutting-edge technologies with a focus on usability, security, and scalability.

## **CHAPTER-5 CONCLUSION AND FUTURE STUDY**

### **5.1 Conclusion**

Utilizing a secure web application for file sharing streamlines the process, allowing users to easily upload, share, and manage files from any device with internet access. With encryption protocols and access controls in place, sensitive information remains protected throughout transmission and storage. Additionally, features such as user authentication and traceability, overall security measures. Collaborative functionalities further enhance productivity, enabling real-time collaboration. By centralizing file storage and access through a user-friendly interface, organizations can streamline workflows and foster seamless communication among team members. Ultimately, implementing a secure web application for file sharing empowers users to exchange information.

### **5.2 Future Study**

For future study and development of the secure file transfer web application, several areas can be explored to enhance functionality and security.

**Enhanced Security Measures:** Investigate advanced encryption techniques, such as post-quantum cryptography, to further strengthen data protection against evolving cyber threats.

**Scalability and Performance:** Conduct research on optimizing file transfer protocols and storage systems to accommodate larger file sizes and improve overall system performance under varying workloads.

**User Experience and Accessibility:** Explore usability studies and user feedback to refine the application's user interface and accessibility features, ensuring a seamless and intuitive experience for diverse user groups.

**Integration with Cloud Services:** Explore integration with cloud storage providers to enable seamless file synchronization and backup options, enhancing data availability and disaster recovery capabilities.

By focusing on these areas of study, the secure file transfer web application can evolve to meet evolving user needs and industry standards, providing a robust and secure platform for efficient file management and sharing.



## REFERENCES :

- [1] Miao Y, Deng R, Liu X, Choo KK, Wu H, Li H. Multi-authority Attribute-Based Keyword Search over Encrypted Cloud Data. *IEEE Transactions on Dependable and Secure Computing*. 2023, Aug 14.
- [2] Yang K, Jia X. Expressive, efficient, and revocable data access control for multi-authority cloud storage. *IEEE transactions on parallel and distributed systems*. 2022, Oct 4;25(7):1735-44.
- [3] Yang Y, Zheng X, GuoW, Liu X, Chang V. Privacy-preserving smart IoT-based healthcare big data storage and self-adaptive access control system. *Information Sciences*. 2020, Apr 1;479:567-92.
- [4] Michalevsky Y, Joye M. Decentralized Policy-Hiding ABE with Receiver Privacy. In *European Symposium on Research in Computer Security 2020*, Sep 3 (pp. 548-567). Springer, Cham.
- [5] K. -H. Huang, E. -C. Chang and C. -L. Chang, "Secure File Sharing Scheme for Mobile Devices," 2013 Fourth International Conference on Networking and Distributed Computing, Los Angeles, CA, USA, 2019, pp. 82-84, doi: 10.1109/ICNDC.2019.18
- [6] Han J, Susilo W, Mu Y, Zhou J, Au MH. Improving privacy and security in decentralized ciphertext-policy attribute-based encryption. *IEEE transactions on information forensics and security*. 2018 Dec 18;10(3):665-78.
- [7] M. N. Uddin, A. H. M. A. Hasnat, S. Nasrin, M. S. Alam and M. A. Yousuf, "Secure File Sharing System Using Blockchain, IPFS and PKI Technologies," 2021 5th International Conference on Electrical Information and Communication Technology (EICT), Khulna, Bangladesh, 2021, pp. 1-5, doi: 10.1109/EICT54103.2021.9733608. keywords: {File systems;Smart contracts;Public key;Authentication;Software;Peer-to-peer computing;Blockchains;Blockchain;Metamask;Smart-contract;IPFS;File Sharing System}
- [8] O. -P. Heinisuo, V. Lenarduzzi and D. Taibi, "Asterism: Decentralized File Sharing Application for Mobile Devices," 2019 7th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), Newark, CA, USA, 2019, pp. 38-47, doi: 10.1109/MobileCloud.2019.00013. keywords: {Peer-to-peer computing;Mobile handsets;Protocols;File systems;Servers;Mobile applications;Power demand;decentralized file sharing;peer-to-peer;mobile;Sailfish OS;InterPlanetary File System},