Spring 2024 ENGR-E516

INDIANA UNIVERSITY
BLOOMINGTON

ENGR-E516 Engineering Cloud Computing

Final Project Report

Mirva Dudhagara (middudha@iu.edu)
Shubham Halvadia (shalvadi@iu.edu)
Malhar Dhopate (mdhopate@iu.edu)

Indiana University Bloomington
Engineering Cloud Computing

April 26, 2024

A Cloud Computing Approach for Secure File Transfer

(GitHub Link: https://github.com/MDhopate/ECC_Final_Project)

Introduction

Cloud computing has revolutionized how organizations manage and access resources, offering a scalable and cost-effective solution for various computing needs. This approach provides an infrastructure that can easily scale to meet fluctuating demands, thereby reducing operational costs. Due to this flexibility, cloud computing has gained widespread acceptance across various industries.[6]

In the context of secure file transfer, cloud computing introduces new opportunities and challenges. The ability to share, store, and manage files in the cloud requires robust security measures to ensure data integrity and confidentiality. As more companies transition to cloud-based solutions, addressing security risks is crucial for maintaining trust and complying with data protection standards.

The objective of a cloud computing approach for secure file transfer is to harness the advantages of cloud infrastructure while implementing strategies to protect sensitive information. This includes encryption, secure protocols, and access control mechanisms to prevent unauthorized access or data breaches.

Major technology companies like Google and Microsoft have developed their own solutions to safeguard user data and files, employing a variety of access controls and encryption protocols. These protocols aim to provide their users with a sense of security regarding their data integrity and confidentiality. Nowadays, protocols like end-to-end encryption, authenticated encryption, block ciphers, etc. are utilized to encrypt data while uploading, or in transit.

This project aims to create a file-sharing application using modern security protocols, deployed on the cloud through Amazon Web Services (AWS). Additionally, it attempts to introduces a user-driven encryption feature, allowing users to choose their preferred encryption method before sending files to the intended recipient, which include but not limited to sending password protected files, temporary links, and only one time view files.

Background and Related Work

Data encryption has a rich history, dating back to 500 BC, when the ancient Egyptians used it for religious purposes, and the Greeks and Romans applied it in military contexts. As technology advanced, so did encryption techniques. Specialized devices, such as the Enigma machines, were developed to encode and decode messages for banking, diplomatic, and military communications. These machines relied on pre-shared keys and needed precise configuration to ensure the same encryption pattern, allowing them to decode the encrypted messages. However, with the advent of new technologies and powerful computers, these traditional methods have become largely obsolete.[8]

Modern computers can execute a vast number of mathematical operations in a short time, making it easier to decrypt messages based on pre-shared keys through brute-force attacks. Computers can attempt millions of key combinations, increasing the likelihood of finding the correct one. There have been many studies and research being conducted to advance the field of cryptography.[7][8]

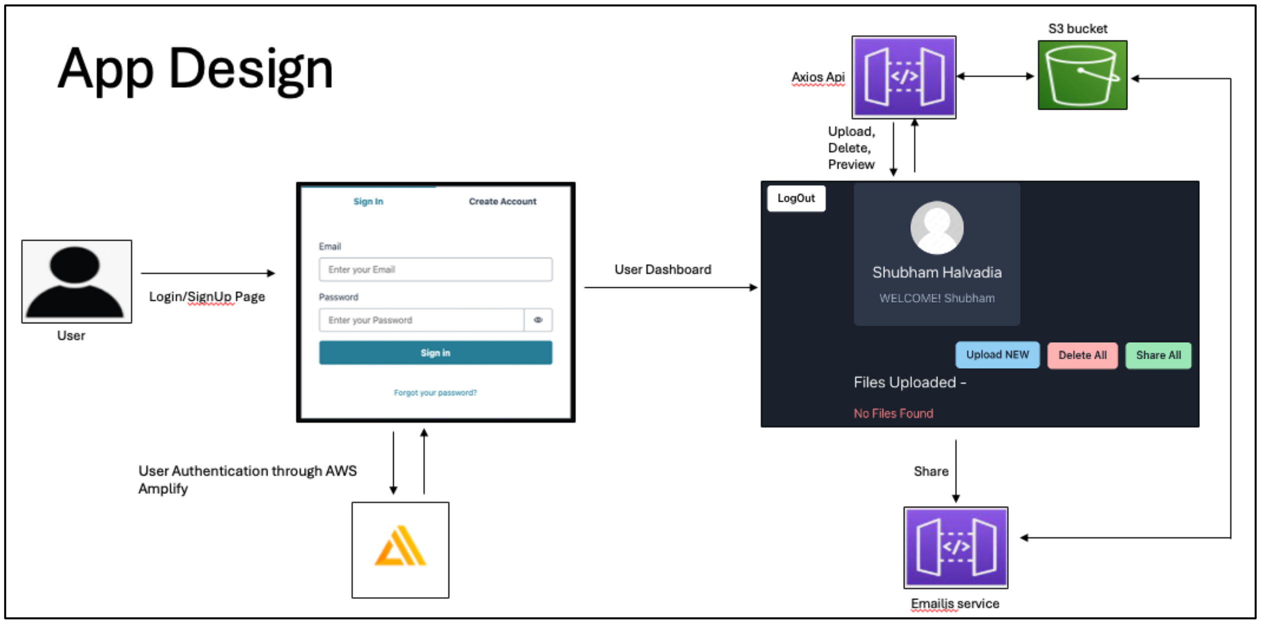
Today, the Advanced Encryption Standard (AES) 256-bit encryption is considered the most secure and unbroken standard. Modern encryption protocols often combine various techniques to create complex cryptographic systems. AES-based block cipher codes are among the most reliable and are widely used by the U.S. government and security agencies to ensure robust data protection.[4]

AES is a symmetric encryption algorithm, using the same key for both encoding and decoding data. It employs a block cipher approach, dividing the input into small blocks and encrypting each individually. This process involves multiple rounds of encryption, transforming the data into an unintelligible format, which is exceptionally secure.[7]

Stenography is a technique that involves hiding data in visual files like images, videos, etc. Stenography technique can be implemented in addition to other encryption protocols to store and send secure data in a non-secure file. Initially this project aimed to attempt and implement the file share application with a AES based protocol, in addition to using stenography to send the secure key to the end recipient.[5]

Application Architecture

The following application architecture outlines the structure used to develop a React-based application and deploy it on the cloud using AWS services. User authentication for login and registration was handled with AWS Amplify, while the user dashboard was built with React. Various APIs were implemented to facilitate communication between the application and the S3 bucket, as well as to enable file sharing with end recipients.



1. AWS Amplify –

AWS Amplify is a fully managed continuous integration and continuous delivery (CI/CD) and hosting service that has been developed for fast, secure and reliable static and servers-side rendered applications that are able to scale easily with a developer’s requirement. [1][2]

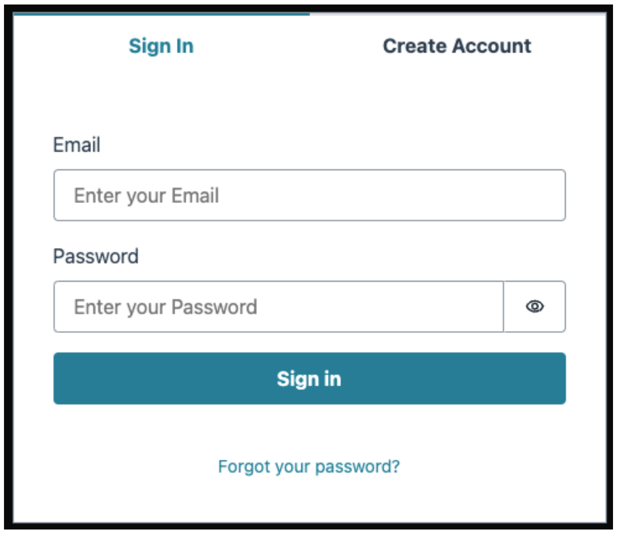
a. **User Registration:** New users can register by providing a valid email address. Upon registration, AWS Amplify triggers an authentication workflow where a verification code is sent to the provided email. This code must be entered by the user to complete the registration process, ensuring that email addresses are verified before allowing access.

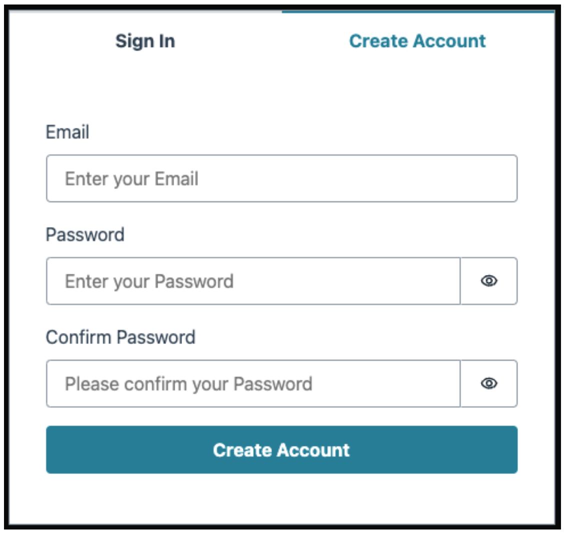
b. **User Login:** For returning users, the login process is straightforward. Users enter their credentials, and AWS Amplify manages the authentication, providing a secure session token upon successful login. This token is then used for session management and to secure communication with backend services. This is the initial page a user interacts with when using this application.

c. **Security Features:** AWS Amplify incorporates several security features that are vital for protecting user data and ensuring secure access:
    i. **Multi-Factor Authentication (MFA):** An optional layer of security that can be enabled to require a second form of authentication, which provides an additional barrier against unauthorized access.
    ii. Automated Security Checks: Amplify regularly scans for vulnerabilities and applies security patches and updates automatically, reducing the risk of security breaches.

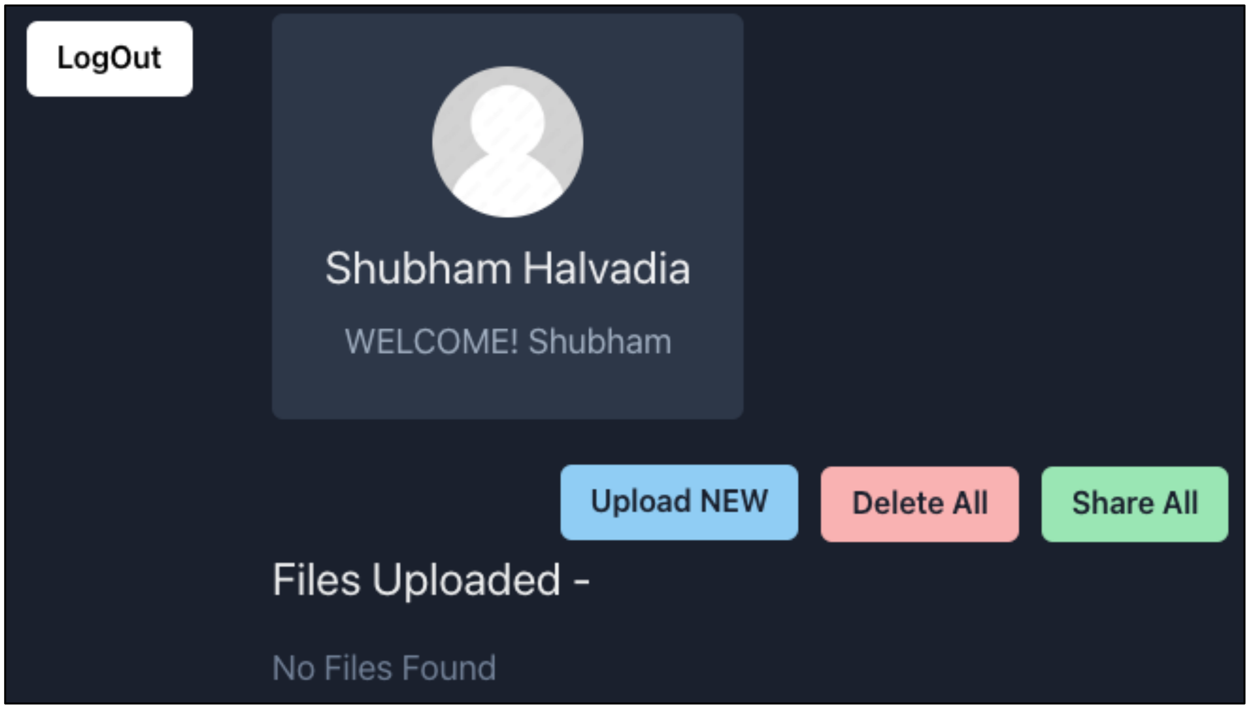
d. **Scalability and Reliability:** AWS Amplify offers a managed environment that scales automatically according to the application's needs. This is crucial for handling varying loads, ensuring that the application remains responsive and available during peak usage.

e. Developer Experience: Amplify provides a unified toolchain with integration to various AWS services, which simplifies the process of developing, deploying, and managing the application. It supports continuous integration and continuous deployment (CI/CD) workflows, which enhances development productivity and helps in maintaining high code quality throughout the project lifecycle.





2. React Application Dashboard –

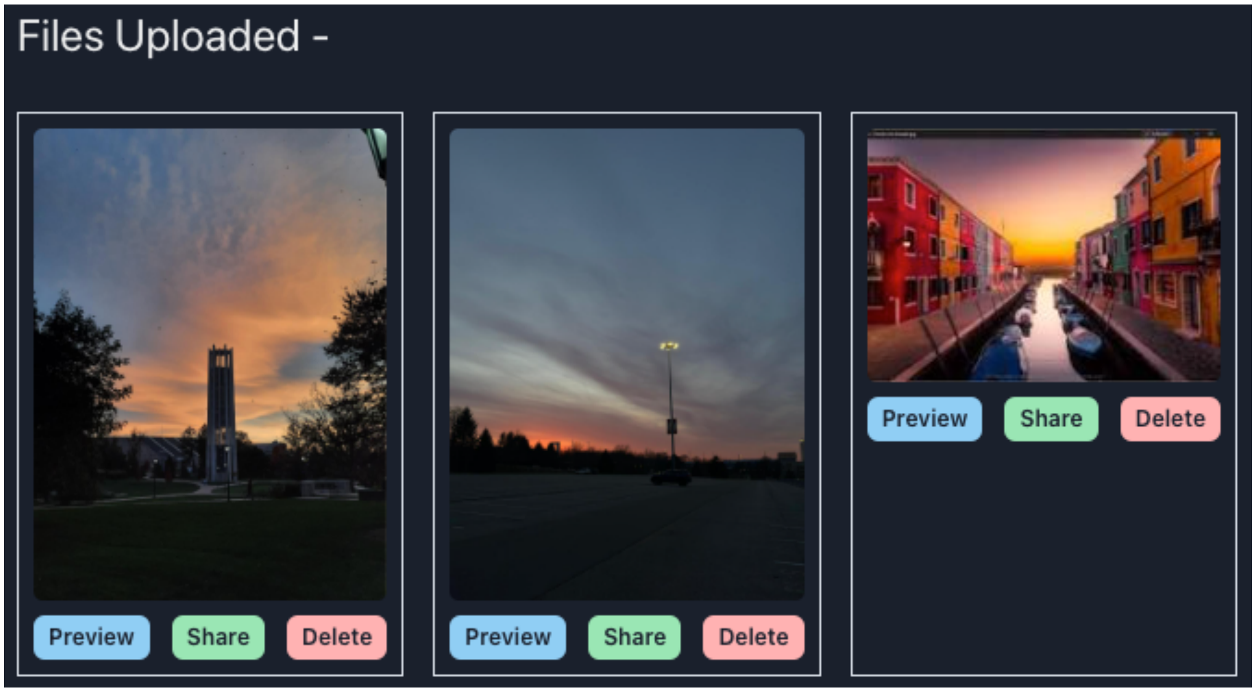


The React Application Dashboard serves as the central interface for users after they log in. Designed with usability and functionality in mind, the dashboard provides a user-friendly environment for managing file transfers securely. Below are the key features and functionalities of the dashboard:

a) **User Interface (UI) Design:** The dashboard is designed to be intuitive and easy to navigate, ensuring that users can effortlessly manage their tasks. The layout is clean and organized, with clear labels and buttons for various actions, such as uploading files, viewing the file list, and accessing shared files. The design adheres to modern UI/UX practices to enhance user engagement and satisfaction.

b) **File Upload Functionality:** A prominent feature of the dashboard is the file upload functionality. Users can select files from their device and upload them directly to AWS S3 through the application. The upload process is secured with AWS Amplify, ensuring that files are transmitted securely to the cloud storage.

c) **File Management:** Once uploaded, files are listed in an organized manner on the dashboard. Each file entry shows a preview of the file, in a sorted manner with the recent most uploaded file displayed first. Users can perform actions like preview, share, or delete directly from this interface.



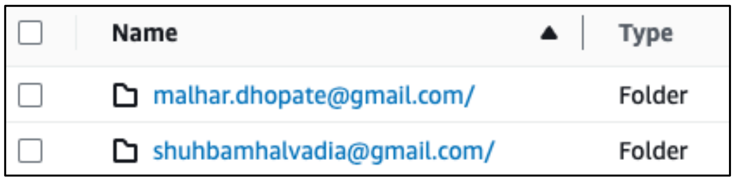
d) **Responsive Design:** The dashboard is built using responsive design principles, making it accessible and functional across different devices and screen sizes. This responsiveness ensures that users can manage their files on-the-go, using smartphones, tablets, or desktops without any compromise in functionality or security.

e) **Visual Feedback and Notifications:** To enhance user interaction, the dashboard provides visual feedback for user actions such as successful file upload or errors during file processing. Notifications are also used to alert users about important security updates or sharing activities.

The dashboard is the hub for all user interactions within the application, providing a secure and efficient way to manage file transfers in the cloud environment. The dashboard components were developed by utilizing a ‘Chakra UI’ library, which is a component library that provides developers with the Its development in React showcases the adaptability and robustness of modern web frameworks in creating dynamic and responsive applications.

3. AWS S3 –

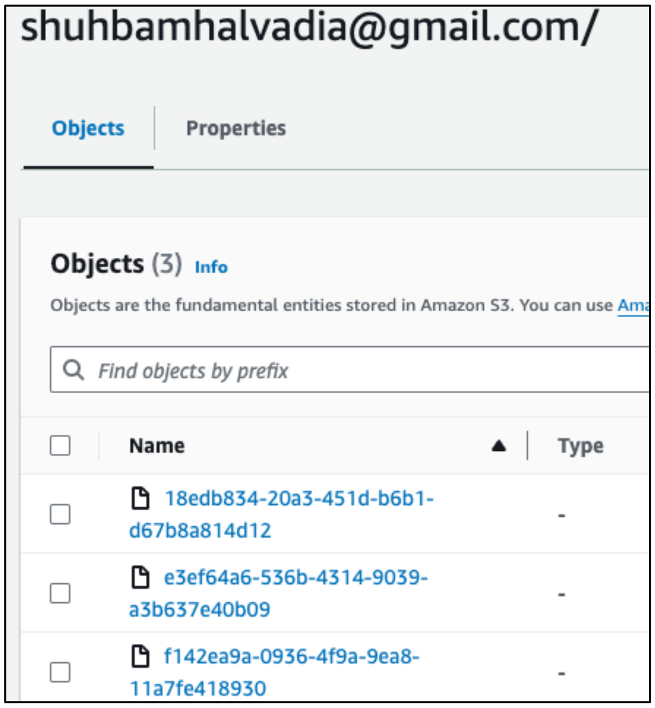
Amazon offers a variety of cloud-based storage classes designed for various use cases. This application leverages Amazon S3 to store all the files users upload. Each user's files are saved in a dedicated folder on S3, providing easy access for sharing or deleting files. This organization structure ensures that all files for a particular user are grouped together.[9]



Name	Type
malhar.dhopate@gmail.com/	Folder
shuhbamhalvadia@gmail.com/	Folder

The screenshot above shows the folder structure with user IDs used to separate folders. For added security, each uploaded file is saved with a random filename, making it difficult for unauthorized users to view or identify specific files. The screenshot below illustrates the files uploaded by 'shubhamhalvadia@gmail.com', displaying the randomly generated filenames to ensure privacy and prevent unauthorized identification.

To ensure data integrity and privacy, security protocols for the S3 bucket have been implemented through AWS-provided policies. Data encryption has also been configured to safeguard the uploaded files.



shuhbamhalvadia@gmail.com/

Objects (3) Info

Objects are the fundamental entities stored in Amazon S3. You can use Ama

Name	Type
18edb834-20a3-451d-b6b1-d67b8a814d12	-
e3ef64a6-536b-4314-9039-a3b637e40b09	-
f142ea9a-0936-4f9a-9ea8-11a7fe418930	-

4. **Axios APIs –**

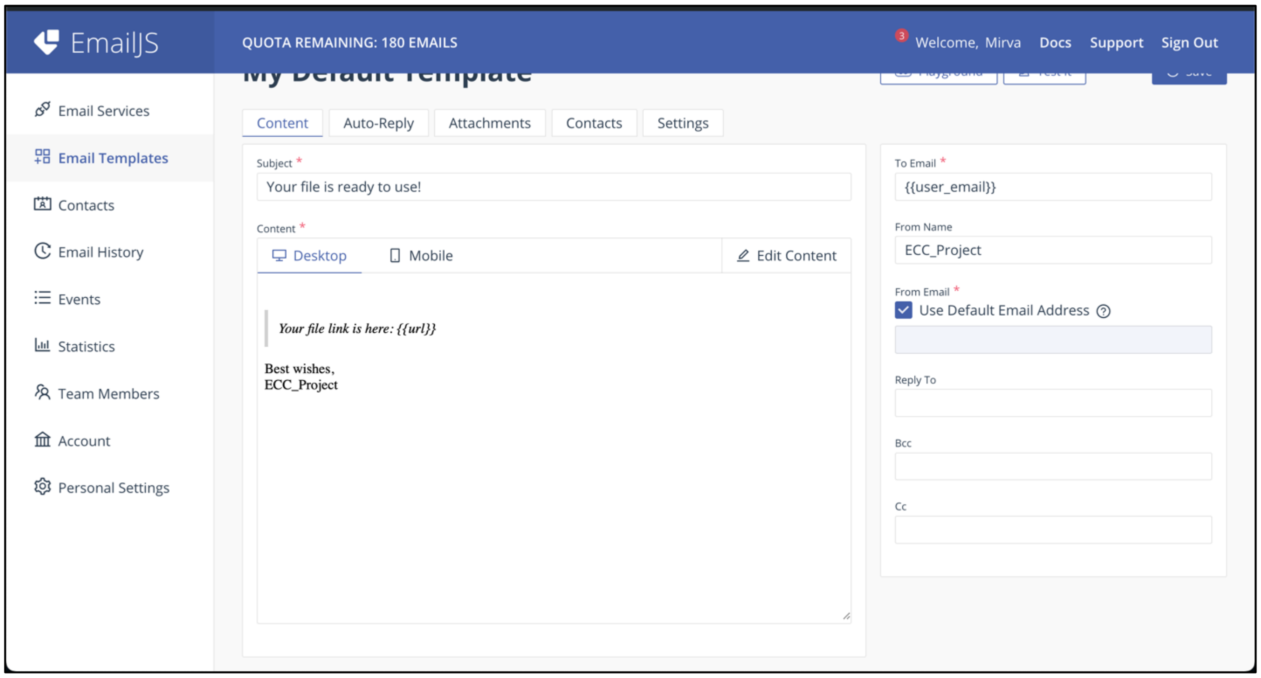
Axios is an application programming interface utilized in this application to make HTTP requests in the web application. Since this is application utilized S3 in a React application Axios, played a very important role in the application development.[11]

a. **Efficient HTTP Requests and Promise-Base Approach –** Axios provided a very straightforward wat to make HTTP requests like GET, POST, and DELETE to the S3 bucket to store or retrieve files. Axios has a Promise-based approach that aligns perfectly with React's asynchronous nature. This made it easier to handle upload operations.
b. **Simple Configuration –** Axios had some premade default configurations, which enabled us to integrate out application with AWS S3 in a somewhat straightforward manner.
c. **Error Handling –** Inbuilt error handling error handling capabilities allowed us to set-up interceptors for requests and responses shared between our application and AWS backend. These interceptors allow us to display errors in case of a network connection, or a file upload error, allowing for a quicker debug and error resolution.

5. File Sharing, EmailJS API –

In our project, the EmailJS service was integral to the functionality of secure file sharing by enabling the transmission of email notifications to users. EmailJS is a cloud-based email service that is a great option for projects that need streamlined email integration because it makes sending emails easier without requiring server-side coding. Emails may be automatically sent from the front end of a variety of online apps thanks to its API, which makes integration easy. This was very helpful for our project since it allowed for direct communication with our React-based application, which made sure users could get alerts in real time about what they were sharing. [12]

Rather than attaching the files directly, we used the EmailJS service to send emails with links to the chosen files. This technique gets over the restrictions of EmailJS's free plan, which excludes file attachments in emails, and also complies with security best practices by avoiding the direct transmission of data. Using pre-made templates from EmailJS's web platform that we tailored to our application's unique requirements was part of the installation. Using EmailJS's powerful features to improve user experience in our secure file transfer system, this strategy offered a dependable and effective way to enable file sharing without compromising the ease of our email correspondence.



Subject *
Your file is ready to use!

Content *

Your file link is here: {{url}}

Best wishes,
ECC_Project

To Email *
{{user_email}}

From Name
ECC_Project

From Email *
✓ Use Default Email Address

Reply To

Bcc

Cc

Evaluation

The project was primarily evaluated by running the code to test the functionality of the application. The testing involved verifying the integration of various components such as AWS S3, AWS Amplify, and the React-based frontend. The code execution focused on ensuring that the application could handle secure file transfers efficiently and that the user interface facilitated easy interaction for registration, login, and file handling. The evaluation confirmed that the application meets the specified requirements for secure file transfer using modern encryption protocols.

The application was developed to share a limited time access link to the designated end recipient. Additionally, we were able to confirm that a variety of file types can be stored, with no size limits. The application is able to provide a proof of concept as of now, by demonstrating the basic functions of a file sharing application.

Conclusion and Future Work

We propose to implement the following enhancements in the future, to make the project more streamlined and practical in day-to-day user,

1. **Additional Encryption Layers** – An additional encryption layer can be implemented to encrypt the files while uploading them to the AWS S3 bucket. This will enhance the security of files at rest, ensuring that even if unauthorized access to the storage is gained, the data remains protected. Additionally, exploring more advanced features such as automated threat detection and integration with additional AWS security services can further enhance the application's security posture.
2. **Enhanced Functions** – We propose to enhance the dashboard functions to incorporates several additional features. We want to incorporate a function where a user is able to set limits to the number of times the recipient views a shared file or keep a track of the number of times a file has been viewed. Further the user will have the option to un-share/recall the shared files.
3. **User Driven Encryption Protocol –** An additional function that we propose to add is to let the user determine how to send the file – by password protection,

more complex cryptography protocols, non-encrypted, limited access files, etc. This approach will allow us to provide the users with a sense of security as they will have direct control over the encryption protocols for their files, while being share.

We believe that the above features ensure that sensitive information remains protected throughout a file’s lifecycle. The project faced several challenges, primarily due to the team's initial unfamiliarity with the technologies involved, such as React, Node.js, and AWS. Setting up the environment required substantial effort, with each team member learning and applying these technologies from scratch. Despite these hurdles, the project successfully created a functional secure file transfer application deployed on the cloud. This accomplishment not only demonstrates the team's ability to overcome steep learning curves but also enhances their practical understanding of cloud-based application development and secure computing practices.

References

[1] https://aws.amazon.com/getting-started/hands-on/build-react-app-amplify-graphql/

[2] Amazon AWS Documentation: https://aws.amazon.com/amplify/?gclid=Cj0KCQjw2PSvBhDjARIsAKc2cgNkwbD5FteJhneWuEnVHMnnIxjCuZRzoIa22YLsrtRZNfD2RKS_V3QaAs7CEALw_wcB&trk=66d9071f-eec2-471d-9fc0-c374dbda114d&sc_channel=ps&ef_id=Cj0KCQjw2PSvBhDjARIsAKc2cgNkwbD5FteJhneWuEnVHMnnIxjCuZRzoIa22YLsrtRZNfD2RKS_V3QaAs7CEALw_wcB:G:s&s_kwcid=AL!4422!3!646025317188!e!!g!!amazon%20amplify!19610918335!148058249160

[3] Saha, P. (2021). *Everything You Need To Know About Diffie-Hellman Key Exchange Vs. PSA.* Encryption Consulting. https://www.encryptionconsulting.com/diffie-hellman-key-exchange-vs-rsa/

[4] Oswald, Ed. (2022). *What is the Advanced Encryption Standard (AES)?* U.S. \ News.https://www.usnews.com/360-reviews/privacy/what-is-advanced-encryption-standard#:~:text=The%20Advanced%20Encryption%20Standard%20(AES)%20is%20an%20algorithm%20that%20uses,make%20it%20harder%20to%20compromise.

[5] Semilof, M. *Steganography*. TechTarget. https://www.techtarget.com/searchsecurity/definition/steganography

[6] https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-cloud-computing#:~:text=Simply%20put%2C%20cloud%20computing%20is,resources%2C%20and%20economies%20of%20scale.

[7] https://www.cia.gov/legacy/museum/artifact/enigma-machine/

[8] https://tresorit.com/blog/the-history-of-encryption-the-roots-of-modern-day-cyber-security/

[9] https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html

[10] https://v2.chakra-ui.com/

[11] https://axios-http.com/docs/api_intro

[12] https://www.emailjs.com/

[13] https://legacy.reactjs.org/