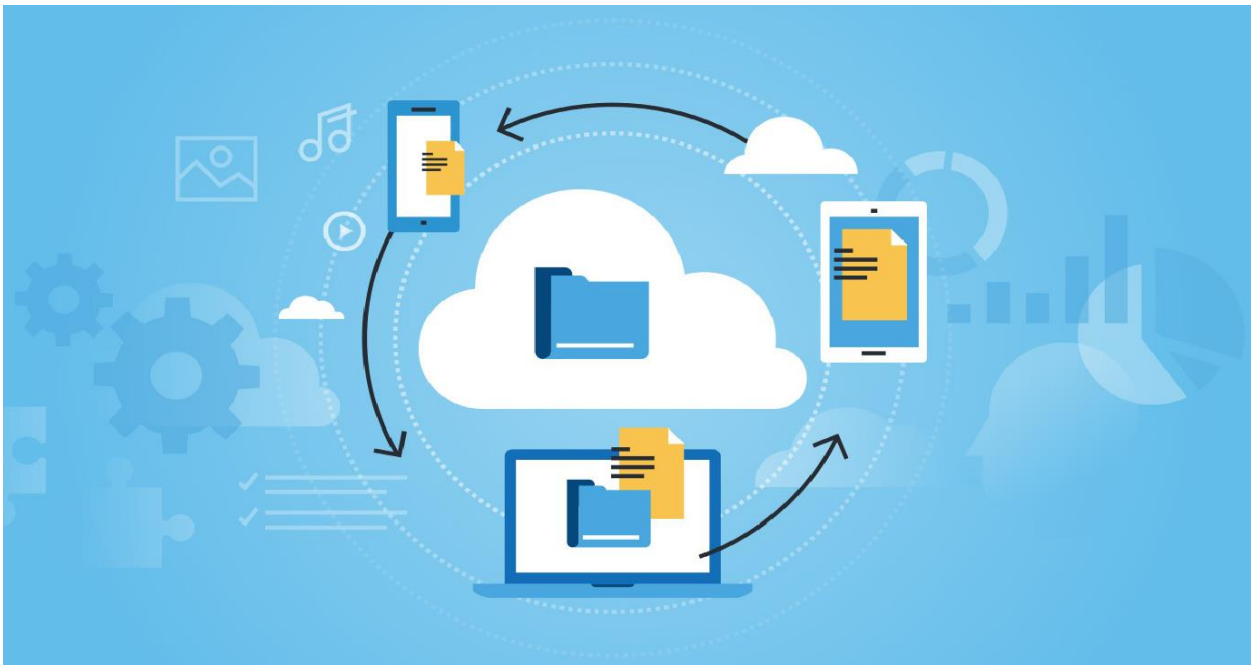


Secure File Exchange System

Created by:

Stephen Boren [SB]

Gabriel Cincov [GC]

Timofey Lykov [TL]

Lionel Song [LS]

Date: Saturday, March 18, 2023

For SFU CMPT 372

Term Project

Table of Contents

Introduction

SB

The purpose of this document is to define the scope of the software project for the development of a Node.js website using Angular, with a database and a login system for sharing encrypted files.

The system uses 256 bit Advanced Encryption Standard (AES) encryption, which is one of the most secure encryption algorithms currently available. AES is a symmetric key encryption algorithm that uses the same key for both encryption and decryption, making it fast and efficient. The system generates a unique session key for each file that is being transferred, and this key is used to encrypt the file data. The session key is also encrypted in the header of the encrypted file using the user's key, making it virtually impossible for anyone to decrypt the file without the user's permission.

Project Overview

SB

The project will involve the development of a web application that allows users to securely upload, share and download encrypted files. The application will have the following features:

- User registration and authentication
- Secure file upload and download functionality
- Encrypted file creation and transmission
- User management and file access control

The file structure on the server will have a USERS folder and each user will have their own folder in there. This is where all the encrypted files sent to them will be stored. Each user will have an UPLOADS folder in their own folder to temporarily store the files they are sending before being converted to the recipients key and deposited in the recipients folder.

Technical Requirements

SB,TL,LS,GC

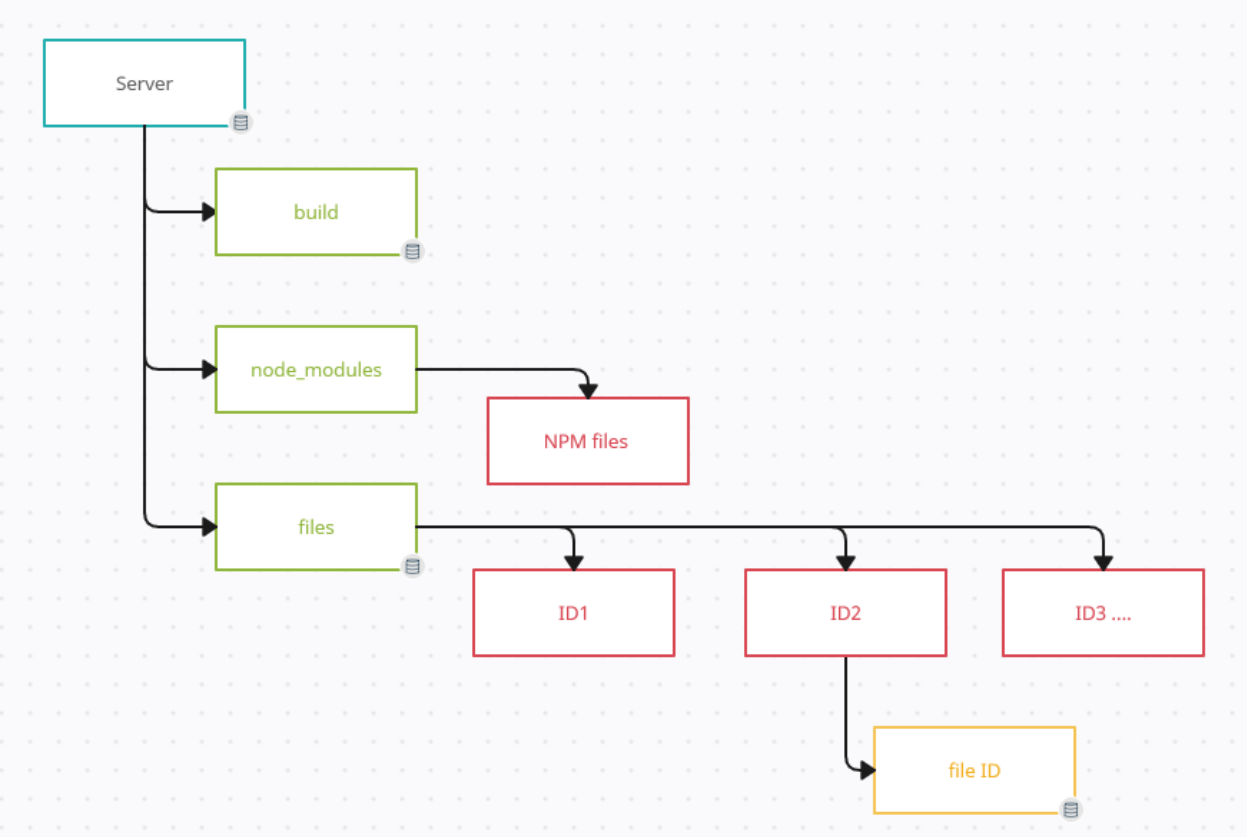
The web application will be built using the following technologies:

- Node.js as the backend server
- Angular as the frontend framework
- MongoDB as the database using Mongoose
- Encryption utilities for secure file storage and sharing

File System

LS,TL,GC,SB

The web application will have the following file system:

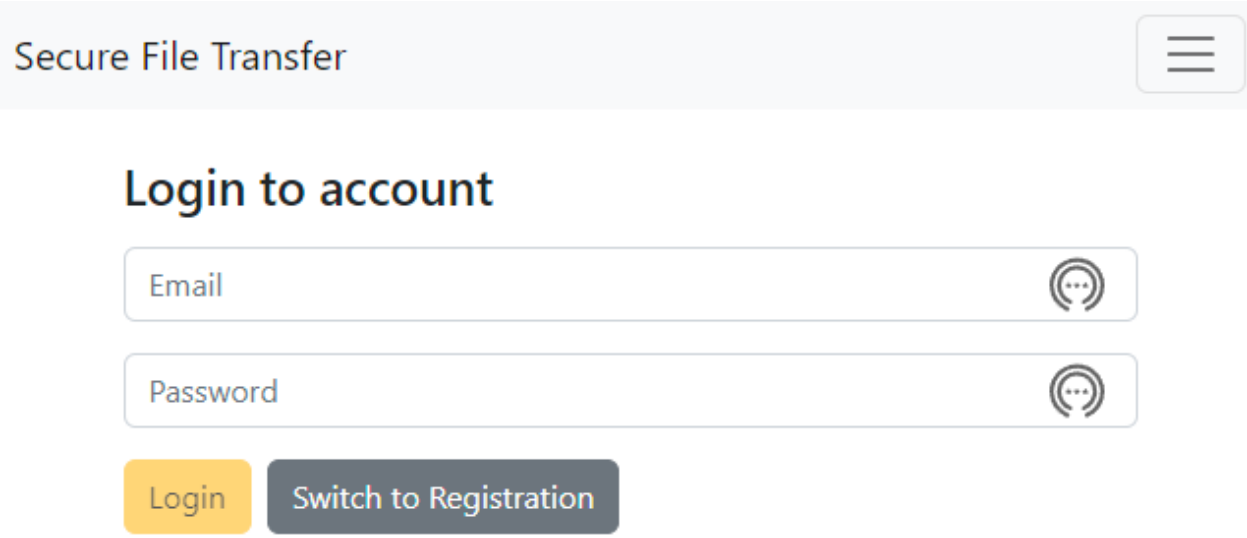


- Regular User: can upload and download encrypted files that they own or have access to
- User can send to any other user their encrypted file

User Registration and Authentication

TL

Users will be required to register with the application using their email address and password. Upon registration, users will receive an email with a verification link to verify their account. Once their account is verified, users will be able to login to the application using their email and password.



Database design and schema

LS,SB

The database is made up of several Tables

- Users
- Files
- Groups [optional]

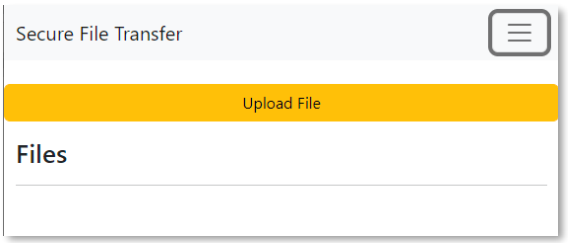
```
var Schema = mongoose.Schema;
var userSchema = new Schema({
    email: {type: String},
    fname: {type: String},
    lname: {type: String},
    encryption_key: {type: String},
    password: {type: String,
minlength: 5},
    createdAt: {type: Date, default:
Date.now},
    updatedAt: {type: Date, default:
Date.now}
});
```

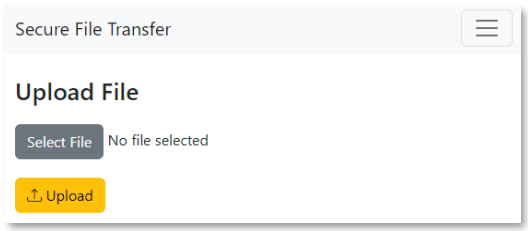
```
var fileSchema = new Schema({
    user_id: {type: String, required:
true},
    shared_by: {type: String,
default: null},
    name: {type: String},
    path: {type: String},
    downloaded: {type: Boolean,
default: false},
    createdAt: {type: Date, default:
Date.now},
    updatedAt: {type: Date, default:
Date.now}
});
```

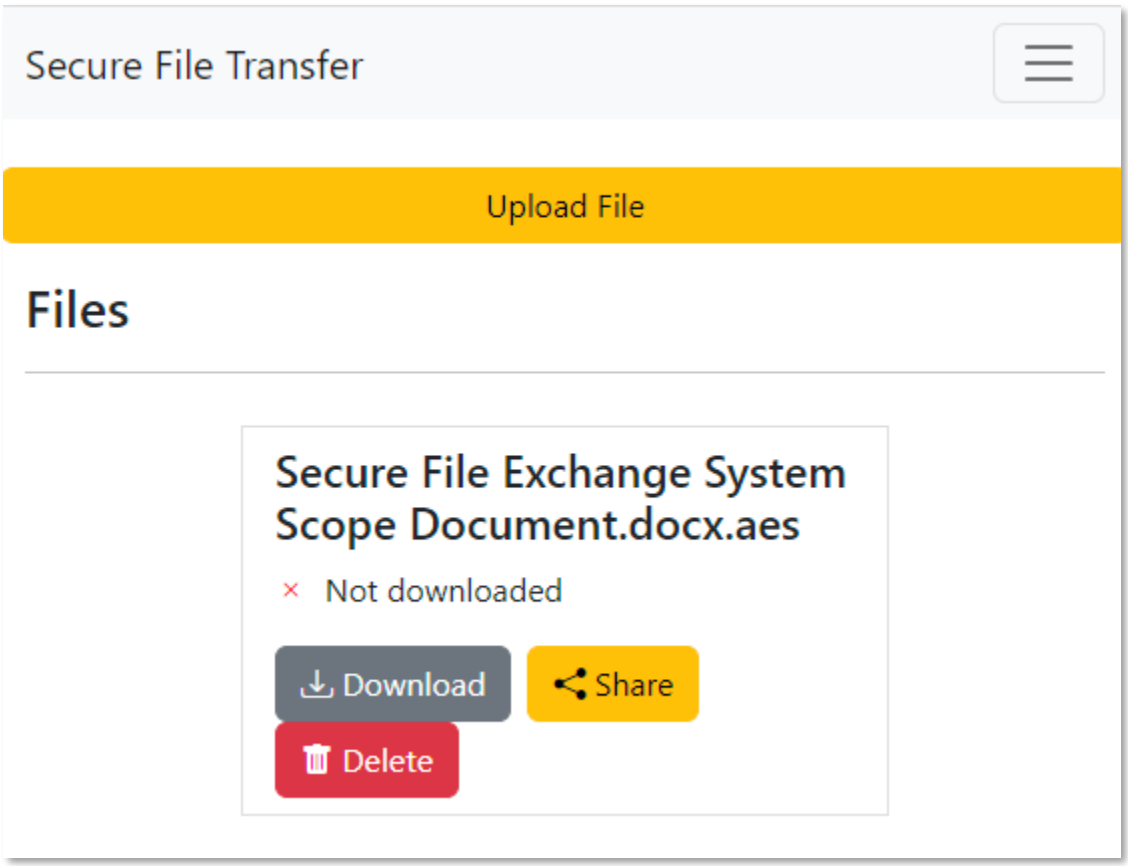
File Upload and Download Functionality

LS,GC

Users will be able to upload files to the application, and the files will be encrypted and stored in the database. Users will be able to download their own files, as well as files that have been shared with them by other users. Files can only be accessed by users who have been granted access by the file owner or an admin.







Encrypted File Utilities

SB

Files uploaded to the application will be encrypted using a secure encryption algorithm before they are stored in the database. The encryption key will be generated randomly for each file, and the key will be stored in a separate table in the database. Files will only be accessible to users who have been granted access by the file owner or an admin.

The command line functions that do all the encryption functions. makekey, createkeyfile, encrypt, decrypt, convertkey. These are all written in gcc and allow the user to encrypt the file that they wish to send. The individual functions are explained below.

By using gcc, utilities can be compiled on any system and OS.

makekey : Outputs a random 64 character hex key (2^{256} bit key)

```
PS D:\Personal\STEVE\CMPT372\Project\sfx\utils> .\makekey.exe
4A432341ECD9CE587570964DB27ADCC58741232B2AAF46BBF6A45118BA84B437
PS D:\Personal\STEVE\CMPT372\Project\sfx\utils>
```

createkeyfile : Create a 'default.key' file in the same folder as the createkeyfile executable that is loaded by default if encrypt or decrypt do not have a key specified using -k flag.

```
PS D:\Personal\STEVE\CMPT372\Project\sfx\utils> .\createkey.exe "4A432341ECD9CE587570964DB27ADCC58741232B2AAF46BBF6A45118BA84B4
Key successfully written to file 'D:\Personal\STEVE\CMPT372\Project\sfx\utils/default.key'.
PS D:\Personal\STEVE\CMPT372\Project\sfx\utils>
```

encrypt : Encrypts the input file(which is required) and if no output file is specified, the output file just appends '.aes' to the input file name. If no key is specified, 'default.key' is attempted to be loaded.

```
PS D:\Personal\STEVE\CMPT372\Project\sfx\utils> .\encrypt.exe -h
Usage: D:\Personal\STEVE\CMPT372\Project\sfx\utils\encrypt.exe [-i,--input <filename>] [-o,--output <filename>] [-k,--key <hex_key>]
PS D:\Personal\STEVE\CMPT372\Project\sfx\utils>
```

decrypt : Decrypts the input file to the output file(which is just the input file name minus the '.aes, unless specified). 'default key' is used unless it is specified.

```
PS D:\Personal\STEVE\CMPT372\Project\sfx\utils> .\decrypt.exe -h
Usage: D:\Personal\STEVE\CMPT372\Project\sfx\utils\decrypt.exe [-i,--input <filename>] [-o,--output <filename>] [-k,--key <hex_key>]
PS D:\Personal\STEVE\CMPT372\Project\sfx\utils>
```

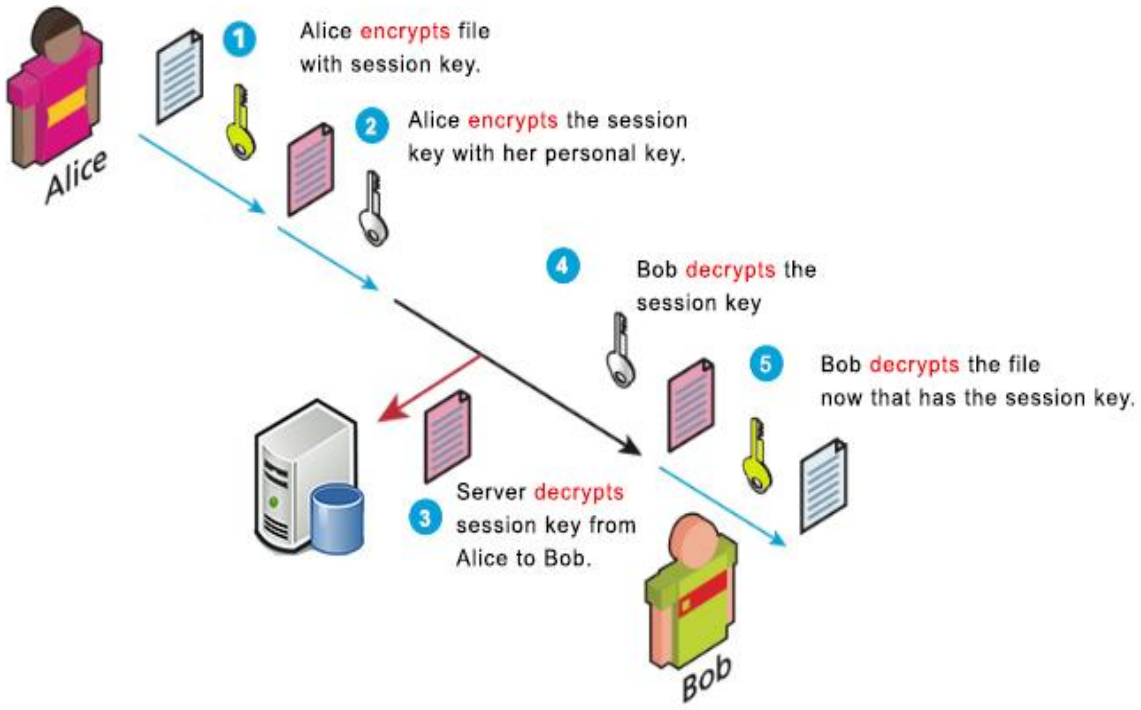
convertkey : Convertkey decrypts the session key in the input file using the inkey and then encrypts the session key using the outkey and outputs the encrypted file to the output file name.

```
S D:\> .\convertkey.exe -h
sage: D:\convertkey.exe [-i,--input <filename>] [-o,--output <filename>] [-ik,--inkey <hex_key>] [-ok,--outkey <hex_key>]
S D:\>
```

Encrypted File Transfer Control

SB

Files are encrypted with a random session key, and then the session key is encrypted with the users key. This allows the user to send to another user by decrypting the session key and then have the system encrypt the session key with the users key. This way the file is never decrypted on the server..



User Management and File Access Control

[Under Construction]

Admin users will have access to a user management system that allows them to add, edit and delete users, as well as assign roles and grant access to files. Regular users will only be able to manage their own files and access files that have been shared with them by other users or by an admin.

Conclusion

The scope of this software project includes the development of a Node.js website using Angular, with a database and a login system for sharing encrypted files. The application will have user registration and authentication, file upload and download functionality, encrypted file storage, and user management and file access control features. The project will be built using Node.js, Angular, MongoDB, and encryption libraries for secure file storage and sharing.