

Final Submission Project Report

Envolution: Secure File Sharing

Rolland Goodenough, Tyler Huckeba, Donald Tran, Spencer Wirth

Capstone Project Spring, 2023

The University of West Florida

19 April 2023

CIS 4595 Capstone Project

Dr. Bernd Owsnicki-Klewe

Table of Contents

Table of Contents.....	2
List of Figures	3
1 Project Description	4
2 Comparison of Final and Initial Requirements	4
2.1 Create an Account	5
2.2 Create a Group.....	6
2.3 Join a Group	6
2.4 Invite Group Member(s)	7
2.5 Remove Group Member(s)	7
2.6 Upload Document(s).....	8
2.7 Delete Document(s)	8
3 Comparison of Initial and Final Timeline	9
4 Project Results Compared with Expectations	10
4.1 Functional Initial Use Cases	10
4.2 Removed/Added/Modified Use Cases.....	10
5 Software Evaluation: Functionality	10
5.1. Functionality Testing	10
5.1.1. Tools	11
5.1.2 Steps	11
5.2 Jest Unit Testing.....	12
5.2.1 Tools	12
5.2.2 Steps	12
5.3 Testing Timeline and Testing Plan.....	13
5.4 Testing Documentation.....	13
5.5 Functionalities Issues That Are Still Open	17
6 Software Evaluation: Security	17
7 Work to Be Done	21

List of Figures

1.	Test Case for Use Case 1	13
2.	Test Case for Use Case 2	14
3.	Test Case for Use Case 3	14
4.	Test Case for Use Case 4	15
5.	Test Case for Use Case 5	15
6.	Test Case for Use Case 6	16
7.	Test Case for Use Case 7	16
8.	Test Case for Use Case 8	17
9.	Vulnerability Spreadsheet Sprint 1	18
10.	Vulnerability Spreadsheet Sprint 2	19
11.	Vulnerability Spreadsheet Sprint 3	20
12.	Vulnerability Spreadsheet Sprint 4	20

1 Project Description

The idea for this project is to create a way to share files among users within a group in a secure manner. The overall concept of the project is to allow users to upload files to their own personal drive for their own personal viewing, or a group drive to be viewed by all users within the respective group. When files are uploaded, it is secured and encrypted through the usage of user and group key management and AES encryption. Overall, what we find is a web application relatively inspired from existing applications such as Google Drive and DropBox.

In terms of the functions of the project, to reiterate, the project exists in order to allow file sharing among users in a group. Groups may consist of colleagues, friends, project members, family, etc. The group manager, also known as the group admin, has the ability to send out invites to other users registered with the web application. The group manager also has the ability to remove members if needed. An account would have to be created using an email address in order to join a group. Once in the group, a member could upload files and send those to the group or download files sent within the group. The user may also upload their own personal files. If uploaded to a personal drive, the user may delete the file, however, a file uploaded to a group drive is only able to be deleted by the group admin.

2 Comparison of Final and Initial Requirements

The requirements for the File Sharing web application are outlined in our initial Project Plan. The requirements for the project are indicated in the form of Use Cases. The initial Use Cases of our project are: Create an Account, Create a Group, Join a Group, Invite Group Member(s), Remove Group Member(s), Upload Document(s), Delete Group, Delete Document(s).

The major requirements for our project were all met except for the implementation of group deletion. Therefore, the major components and requirements of the web application that has been developed are:

2.1 Create an Account

Summary: Users will be able to create an account on the home screen of the web application. Allowing them to register an email address, account username and a password for their personal account on the web application.

Rationale: Users will need to be able to create their own account for authentication of identity and for further access into the web application to access features.

Users: Any individual who wishes to use the web application to either receive or send documents.

Preconditions: The user must have the web application loaded up in the browser and must not be already logged into it.

Basic Course of Events:

1. The user will click on the button listed as “Create Account”
2. The application will prompt the user to enter their desired Username, email, and password for the account.
3. The web application will then check to see if an account already exists within the system with the previously entered Username and Email.
 - . If the username and email already exist within the system, then the web application will prompt the user to reenter the account information and alert them that there is an already existing account with those credentials.
 - a. If the username and email does not exist, then the web application will then proceed with the process of storing the credentials on the Database with proper security measures being applied to the password.
2. Once the account is created on the system, the web application will back the user out to the main screen and ask them to login with their credentials for the new account.

Postconditions: The affected user will effectively be logged into their user account, resulting in the user being redirected to the account dashboard or account-bound area of the web application that they are now able to access.

2.2 Create a Group

Summary: Users will be able to create a group in the Envolution web application to be able to invite members to and upload documents to.

Rationale: The users will need to be able to create a group if they choose to so that they can invite members to in the future if they desire.

Users: Any individual who wishes to use the web application to either receive or send documents.

Preconditions: The user must have an Envolution account, be logged into the web application and navigate to the “Create a group screen”.

Basic Course of Events:

1. The user will click on the button listed as “Create Group”
2. The application will prompt the user to enter group related information such as
 1. The web application will prompt the user to enter the group name.
 2. The web application will prompt the user to enter the group description.
 3. The web application will prompt the user if they would like to create a specific join code.
 1. If no specific code is entered, then it will generate a random one for the group.
2. Once the join code has been created, the creator of the group will be marked as its owner and be given the “Admin” role and add the user to the list of members for the group. In addition, it will store the group information on the database, including the name of the group, the group description and the group join identifier, along with the member list.
3. Once the database has properly stored the information the server will update the client to show the newly created group on the user’s dashboard.

Postconditions: All invited users will have access to the group dashboard and the group owner will be able to manage the group and files.

2.3 Join a Group

Summary: Users will be able to join a pre-existing group on the Envolution web application with a provided join code.

Rationale: The users will need to be able to join a group to share files with if they decide they do not want to make their own group.

Users: Any individual who wishes to use the web application to either receive or send documents.

Preconditions: The user must have an Envolution account, be logged into the web application and navigate to the “Join a group screen”.

Basic Course of Events:

1. The user will click on the button labeled “Join a group”.
2. The web application will then prompt the user for a join code.
3. Once entered the web application will communicate with the database to see if the join code exists.
 - a. If the join code exists then the user will be added to the list of members for the group and will refresh the currently logged in user’s dashboard to display the group on their client.
 - b. If the join code does not exist then the web application will re-prompt the user to enter a valid join code and will continue this process until the user enters a valid one or leaves the “Join a group” screen.

Postconditions: The user is placed into the group and is able to view the group dashboard.

2.4 Invite Group Member(s)

Summary: Owner of the group will be able to invite users to their group

Rationale: The purpose of a group is to allow multiple users to share documents, therefore, there needs to be multiple users within a group, so there needs to be functionality to invite users to a group.

Users: Group Administrators

Preconditions: The user must have an Envolution account, be logged into the web application and navigate to the “Join a group screen”.

Basic Course of Events:

1. The user will click on a button labeled “Invite”
2. The web application will generate a randomized join code.
3. The user will copy the join code and share it with users they want to invite.

Postconditions: Join code is generated and the user who received valid join code can use that code to join the group.

2.5 Remove Group Member(s)

Summary: Group Administrators will be able to remove Group Members.

Rationale: User access to files is generally time constrained, as employees leave companies or a consultant's contract comes to an end.

Users: Group Administrators

Preconditions: The user must have an Envolution account, be logged into the web application and navigate to the group dashboard.

Basic Course of Events:

1. The user will click on a button labeled "Remove Member".
2. The user will select the users they want to remove.
3. The user will click a second button to confirm changes.

Postconditions: All selected members are removed from the group.

2.6 Upload Document(s)

Summary: Users will be able to upload files to be stored and accessed at any time remotely.

Rationale: A file storage system requires that files are first uploaded to be accessed later. Without the ability to upload files this system hardly has a function.

Users: All Users of the Envoluton web application.

Preconditions: Given a User has created an account and been authenticated.

Basic Course of Events:

1. The User navigates to the file upload page
2. The User clicks the upload file button
3. The Program displays a filesystem pop up
4. The User selects a file or multiple files to upload
5. The Program will send the file to the server
6. The server will then properly encrypt the file before sending it to the database.

Postconditions: The uploaded document is viewable within the dashboard.

2.7 Delete Document(s)

Summary: Group owners will be able to delete files from their respective groups

Rationale: Being able to delete documents from groups will be able to fix any problems of users uploading the incorrect documents.

Users: All group owners in the Envolution web application.

Preconditions: Given a User has created an account and been authenticated and has created a group.

Basic Course of Events:

1. The User navigates to their respective group(s).
2. The User clicks the remove file button.
3. The Program displays a list of existing files in the group.
4. The User selects a file or multiple files to delete.
5. The Program will prompt the user for verification that they would like to continue with deleting the file.
6. The server will then communicate with the database to remove the file from the group.

Postconditions: The uploaded document is viewable within the dashboard.

3 Comparison of Initial and Final Timeline

Our project completed on time, with most of the original planned features. This was possible due to the fact the project was given a massive boost towards the start of our second sprint which initially was set to be done from Feb 13 - Feb 26, however, with our choice of switching from MongoDB to Firebase for handling users and storing data which made implementing CRUD operations in the beginning phases of the project much simpler, which resulted in the second sprint requirements being completed before its due date. Therefore, we were able to spend the rest of the second sprint fine tuning some functionalities.

Our project was split up into several two-week sprints, with the tasks for the sprint being decided at its start of each. Our sprints followed the original schedule; with the exception of the switch to Firebase in the beginning causing some small confusion for the first sprint. However, since each sprint is planned independently, they adapt quickly to changes in the project.

4 Project Results Compared with Expectations

4.1 Functional Initial Use Cases

The Use Cases that are relevant to our project were previously listed in section 2 on this Project Report. The initial use cases included: Create an Account, Create a Group, Join a Group, Invite Group Member(s), Remove Group Member(s), Upload Document(s), Delete Group, Delete Document(s). Throughout the group's four planned sprints, the team have been able to implement and complete all use cases listed with the exception of the ability to delete groups. Furthermore, tests were developed, specifically, unit and functionality testing to ensure that each use case is functional as a whole. Therefore, it may be concluded that all initial use cases with the exception of the Group Deletion use case are functional within the web application.

4.2 Removed/Added/Modified Use Cases

The use cases that have been removed is Group Deletion. Besides this, no other use cases as described from the initial use cases from section 2 have been removed, added, or modified.

5 Software Evaluation: Functionality

The two main testing methods utilized for testing our project requirements were functionality testing and Jest Unit Testing.

5.1 Functionality Testing

Functionality testing involved manually performing actions upon the web application to ensure that all functionalities and features worked as expected. This includes testing the user interface, navigation, and overall usability, which was one of the project's targeted quality criteria.

5.1.1. Tools

Due to the nature of functionality testing being manual, there is no need for any testing tools, except for the utilization of Microsoft Excel or Google Sheets in order to record and document results.

5.1.2 Steps

The steps we took in order to perform the Functionality testing method is as follows:

1. Design test cases: Design a detailed test case for specific scenarios to be tested, specifying a summary of the test case, any preconditions, the test inputs, expected results, and relevant comments. These test cases are documented within a spreadsheet.
2. Set up test environment: Configure and prepare the test environment by installing necessary software, packages, etc. Making sure that the project being tested upon is the latest version.
3. Execute test case: Manually go through each test case, following the steps outlined within the test case documentation. Record the results and compare them with expected results.
4. Report bugs and errors: If any discrepancies were found between the expected and actual results, then the issue is to be reported as a defect. A comment is included with the results in order to give detailed information about the issue, such as steps to replicate it, as well as its severity.
5. Document results: Finally, the results are all documented into an organized spreadsheet and shared with the rest of the team. The documentation serves as a reference for future testing activities and serves as valuable insights for the development team.

5.2 Jest Unit Testing

Jest Unit tests is a JavaScript testing framework that allows the team to write and execute unit tests for our back-end functionalities. The unit tests are focused on individual functions and components ensuring that they are able to perform and work properly in isolation.

5.2.1 Tools

Since Jest is a JavaScript testing framework, there are a couple tools needed in order to perform such tests, most of them being dependencies. First, Jest is built on top of Node.js, so in order to perform Jest Unit tests, the testing system will need to have Node.js installed. Next, the testing system would need to install Jest as a dependency in the project. Since Jest includes built-in utilities for testing, such as mock modules/functions, matchers, and snapshots, there is no need to install any other additional testing libraries.

5.2.2 Steps

The steps we took in order to perform the Functionality testing method is as follows:

1. Set up testing environment: Install Node.js and install Jest as a development dependency to the project. Configure Jest by adding a script for running Jest within the project's package.json file.
2. Write the tests: Create the test files with the .test.js extension and place them inside a '__test__' folder. Jest automatically identifies these files as test files. Then, write the test cases using the it() function and use built-in matchers such as toBe() and toEqual() in order to write assertions to compare actual and expected results.
3. Run Tests: Execute the tests using the configured script done during the first step. Usually, the script is just to make it such that if 'npm test' is run, then it runs the Jest tests.

4. Analyze and document results: Review the test results printed to terminal, keying in on failed tests. If a test fails, fix the issue and then rerun the tests. Record all test results in an organized spreadsheet.

5.3 Testing Timeline and Testing Plan

Since our team utilizes the Agile Software development method, the testing timeline for our project is an ongoing process that occurs through the entire project development lifecycle. In each iteration/sprint of our project, which occurs over a duration of 2 weeks. We integrate the functionality and unit testing in order to obtain continuous feedback. As such, the testing plan for our project went as follows:

1. Sprint Planning: Create test scenarios and test cases for the user stories for the current iteration.
2. Sprint Execution: As new features are being developed, unit tests are written and executed. Testing lead provides feedback to the team members.
3. Spring Review: Test results are reviewed with the whole team, and we discuss outstanding issues and then plan the solution for it for the next sprint.

5.4 Testing Documentation

Test Case ID	Req	Summary	Precondition	Steps	Expected result	Results	Comments
Use Case 1: Create an account							
GENERAL PRECONDITION http://localhost:3000							
Go to http://localhost:3000 Log in (credentials: Google account)							
UC1TC-1		Access Google Accounts popup		1. Navigate to http://localhost:3000 2. Click Login to Envolution	1. Login page with button to login 2. Choose an Account through popup	passed	
UC1TC-2		Login with Google Accounts with accounts already saved		1. Navigate to http://localhost:3000 2. Click Login to Envolution 3. Click on an account to login with 4. Enter account password	1. Login page with button to login 2. Choose an Account through popup 3. Enter password for account chosen 4. Access dashboard	passed	
UC1TC-3		Login with Google Accounts with no accounts saved		1. Navigate to http://localhost:3000 2. Click Login to Envolution 3. Enter an email or phone 4. Enter password	1. Login page with button to login 2. Choose an Account through popup 3. email saved page navigates to password enter screen 4. Access dashboard	passed	

Figure 1: Test Case for Use Case 1

The testing of Use Case 1: Create an Account was divided into three requirements: The ability to access the Google Accounts popup, the ability to login with the Google account with accounts already saved, and the ability to login with Google accounts with no accounts saved. As observed, all test requirements passed and there were no observed defects or failures.

Test Case ID	Req	Summary	Precondition	Steps	Expected result	Results	Comments
Use Case 2: Create a group GENERAL PRECONDITION: http://localhost:3000/share-with-me Access to dashboard Access to share with me dashboard							
UC2TC-1		Access group creation popup		1. Click Shared with me navlink 2. Click Choose Group drop down menu 3. Click Create Group	1. Access shared drive with option to create group 2. Choose Group drop down menu opens 3. Create a group popup	passed	
UC2TC-2		Create a group with a group name		1. Click Shared with me navlink 2. Click Choose Group drop down menu 3. Click Create Group 4. Enter a group name in the group name text field 5. Click OK	1. Access shared drive with option to create group 2. Choose Group drop down menu opens 3. Create a group popup 4. Group name entered within group name text field 5. Group created	passed	
UC2TC-5		Group name can not be blank		1. Click Shared with me navlink 2. Click Choose Group drop down menu 3. Click Create Group 4. Enter nothing for the group name 5. Click OK	1. Access shared drive with option to create group 2. Choose Group drop down menu opens 3. Create a group popup 4. The length of group name is 0 characters 5. Error, Group name must be between 3-73 characters	passed	Need to display error message

Figure 2: Test Case for Use Case 2

The testing of Use Case 2: Create A Group was divided into three requirements: The ability to access the group creation popup, the ability to create a group with a group name, and for group names to not be blank. All test requirements passed as there were no observed defects.

Test Case ID	Req	Summary	Precondition	Steps	Expected result	Results	Comments
Use Case 3: Join a Group GENERAL PRECONDITION http://localhost:3000 Go to http://localhost:3000 Log in (credentials: Google account)							
UC3TC-1		Join a Group		1. Login and access the web application. 2. Click the alerts button on the header. 3. View the group invitation. 4. Accept the Group Invitation.	1. Alerts badge appears on alerts button 2. Group Invitation appears in menu. 3. User is added to the group	passed	
UC1TC-2		Decline a Group		1. Login and access the web application. 2. Click the alerts button on the header. 3. View the group invitation. 4. Decline the Group Invitation.	1. Alerts badge appears on alerts button 2. Group Invitation appears in menu. 3. User is not added to group	passed	

Figure 3: Test Case for Use Case 3

The testing of Use Case 3: Join A Group was divided into two requirements: The ability to join a group and the ability to decline a group. All test requirements passed as there were no observed defects.

Test Case ID	Req	Summary	Precondition	Steps	Expected result	Results	Comments
Use Case 4: Invite Group Member(s)							
GENERAL PRECONDITION: Access to dashboard							
UC4TC-1		Add members to group		1. Click Shared with me navlink 2. Click Choose Group drop down menu 3. Click on a group you created 4. Click Add Member 5. Enter user's email address 6. Click OK	1. Access shared drive with option to create group 2. Choose Group drop down menu opens 3. Access to group dashboard 4. Update group popup 5. User's email entered into User email input field 6. Invite Sent	passed	

Figure 4: Test Case for Use Case 4

The testing of Use Case 4: Join A Group was divided into a single requirement: The ability to add members to a group. The test requirements passed as there were no observed defects.

Use Case 5: Remove Group Member(s)							
GENERAL PRECONDITION http://localhost:3000 Go to http://localhost:3000 Log in (credentials: Google account)							
UC3TC-1		Remove group members as an admin		1. Login as the admin of the group 2. Click the View Groups button on the sidebar 3. Click the "eye" icon next to the group 4. Click on the kick button next to the name of the user to kick	1. All groups for logged in user is loaded 2. All groups are displayed from View Groups modal 3. All users within the group are displayed 4. User is kicked from group	passed	

Figure 5: Test Case for Use Case 5

The testing of Use Case 5: Remove Group Member(s) was divided into a single requirement: The ability to remove groups members as an admin. The test requirements passed as there were no observed defects.

Test Case ID	Req	Summary	Precondition	Steps	Expected result	Results	Comments
Use Case 6: Upload Document(s)							
GENERAL PRECONDITION: Access to dashboard							
UC6TC-1		Upload files to your drive		1. Click on My Drive 2. Click Upload File 3. Select files to upload 4. Click submit to upload files	1. Access to your drive 2. Upload files popup 3. Files are saved to upload 4. Files are uploaded to your drive	passed	
UC6TC-2		Contribute to group		1. Click Shared with me navlink 2. Click Choose Group drop down menu 3. Click on a group you have access to 4. Click on Contribute 5. Select files to upload 6. Submit files	1. Access shared drive with option to create group 2. Choose Group drop down menu opens 3. Access to group dashboard 4. Select files to upload popup 5. Files are saved to upload 6. Files are uploaded to group dashboard	passed	
UC6TC-3		Your drive is unique to each user (if a user uploads a file to their drive, it should not be viewable by other users)		1. Click on My Drive 2. Click Upload File 3. Select files to upload 4. Click submit to upload files	1. Access to your drive 2. Upload files popup 3. Files are saved to upload 4. Files are uploaded to your drive	passed	
UC6TC-4		Able to select multiple files to upload		1. Click on My Drive 2. Click Upload File 3. Select more than 1 files to upload 4. Click submit to upload files	1. Access to your drive 2. Upload files popup 3. Files are saved to upload are >= 2 4. Files are uploaded to your drive	failed	Need to implemented functionality to select multiple file to upload

Figure 6: Test Case for Use Case 6

The testing of Use Case 6: Upload Document(s) was divided into four requirements: Upload files to personal drive, upload files to group, have personal drives be unique to each user, and the ability to select multiple files to upload. As of right now, a user can only select one file to upload at a time due to complications dealing with the encryption of a single file during upload compared to the encryption of each of a group of files during upload. So, requirement ID UC6TC-4 fails for Use Case 6. Other than that, the rest of the requirements passed as there were no observed defects.

Test Case ID	Req	Summary	Precondition	Steps	Expected result	Results	Comments
Use Case 5: Delete Group							
GENERAL PRECONDITION http://localhost:3000 Go to http://localhost:3000 Log in (credentials: Google account)							
UC5TC-1		Delete Group as an admin		1. Login as the admin of the group 2. Click the View Groups button on the sidebar 3. Click the "eye" icon next to the group 4. Click on the kick button next to the name of the admin	1. All groups for logged in user is loaded 2. All groups are displayed from View Groups modal 3. All users within the group are displayed 4. Admin is kicked from the group and group is deleted	passed	

Figure 7: Test Case for Use Case 7

The testing of Use Case 7: Delete Group was divided into a single requirement: The ability to delete groups as an admin. The test requirements passed as there were no observed defects.

Test Case ID	Req	Summary	Precondition	Steps	Expected result	Results	Comments
Use Case 8: Delete Document(s) GENERAL PRECONDITION http://localhost:3000 Go to http://localhost:3000 Log in (credentials: Google account)							
UC8TC-1		Delete group documents as an admin		1. Login as the admin of the group 2. Access the Shared With Me drive 3. Right click on a group file 4. Choose "Delete" from drop down menu	1. Shared With Me Drive is accessed displaying all group files 2. "Delete" option is displayed from selection menu 3. File is deleted from group	passed	
UC8TC-2		Delete group documents as an non-admin		1. Login as the non-admin of the group 2. Access the Shared With Me drive 3. Right click on a group file 4. Choose "Delete" from drop down menu	1. Shared With Me Drive is accessed displaying all group files 2. "Delete" option is displayed from selection menu 3. File is not deleted from group	passed	

Figure 8: Test Case for Use Case 8

The testing of Use Case 8: DeleteDocument(s) was divided into two requirements: The ability to delete group documents as an admin, and the ability for non-admins to not be able to delete group documents. The test requirements passed as there were no observed defects.

5.5 Functionalities Issues That Are Still Open

The most prominent issue that is still open is that the web application performs rather slowly for the Shared Drives due to the front-end making too many unnecessary requests to the back end, as well as our web application not caching any data. Other than this, the only other functionality issue that is present from the testing is not being able to select multiple files at once during file upload. Right now, a user can only upload one file at a time.

6 Software Evaluation: Security

VSCode extensions Snyk, Deepscan, and Codesight were used to test the source code security. Codesight was used on the first sprint vulnerability check but not beyond that because the Security Lead was hit with a paywall to continue using it. These ran through the code that we created on VSCode and checked for any vulnerabilities within and vulnerabilities on any

dependencies. Also, the Snyk database, the MITRE Corporation, and CVE Details, among others, were used to analyze and evaluate applications that we were planning on using. The security of the code was tested after it was created. For example, the code that was created for sprint 2 was tested over sprint 3. It is impossible to test the code before it is created so this is the way that the testing had to be done. Once the code was tested and all of the vulnerabilities were identified the group was alerted and the code was altered by the person who worked on that section of code. For the final sprint, penetration testing was deployed. The Security Lead acted as a User and saw if there was any way that a User was able to access something they shouldn't. He also checked for attacks such as Cross-Site Scripting and SQL Attacks, among others. In total, there were only one or two high risk vulnerabilities across all of the sprints. A majority of them were low risk like variables not being used, unreachable code, and no catch methods within the code. As of right now, the security issues that are still open are 1 with High risk, a few medium risks, and just a couple low risks that we were unable to get to within the allotted time and we deemed less critical than getting the encryption for the file sharing done.

	A	B	C	D	E	F	G	H	I
1	Vulnerability	Impact	Risk	Mitigations					
2	Buffer overrun triggered in the X.509 certificate verification. Attacker crafts and email address to overflow 4 bytes on the stack	Denial of Service and Remote Code Execution	High	Application doesn't continue certificate verification after a failure to construct a path to a trusted user					
3	Buffer overrun triggered in the X.509 certificate verification. Attacker crafts and email address to overflow any number of bytes on the stack	Denial of Service and Remote Code Execution	High	Application doesn't continue certificate verification after a failure to construct a path to a trusted user					
4	Certain browsers still attempt to resolve invalid octal IP addresses via DNS. Combined with an active --inspect session	Code Execution and DNS rebinding	Low	Don't allow browsers that still attempt to resolve octal IP addressesPotentially put into code that Google Chrome is only browser allowed. Also put in that browsers aren't allowed to run at the same time as --inspect is running. Escape and validate user input. Encode data on output. Non-					

Figure 9: Vulnerability Spreadsheet Sprint 1

Here you can see that all of the applications and APIs were researched for any vulnerabilities, bugs, or weak points that existed within their system. Once that information was gathered, I

ranked them based on their risk impact using the scale Low, Medium, High, or Critical. Each application was on another spreadsheet as can be shown in the picture. Since there was no code written yet for this sprint I couldn't check it for vulnerabilities so the applications we planned on using had to be checked instead.

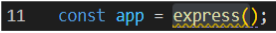
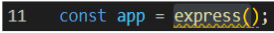
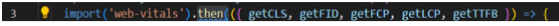
	A	B	C	D
1	Vulnerability	Location	Risk	Picture
2	CSRF protection is disabled for your Express app. This allows attackers to execute requests on a user's behalf	Line 11 of index.js	Medium	
3	Disable X-Powered-By header for your Express app. Consider using Helmet middleware, because it exposes information about the used framework to potential attackers.	Line 11 of index.js	Medium	
4	No catch method for promise. This may result in an unhandled promise rejection.	Line 3 of webReportVitals.js	Low	
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				

Figure 10: Vulnerability Spreadsheet Sprint 2

Here we can see the vulnerability spreadsheet for Sprint 2. This was the first sprint that I could check the code our group created. I used the Snyk and Code Sight scanners for this sprint. There weren't a lot of vulnerabilities for this sprint just because there wasn't enough high-level code created yet.

	A	B	C	D
1	Vulnerability	Location	Risk	Picture
2	Imported binding 'firebase' is not used	Line 1 of DocumentTable.js	Low	<pre>import firebase from 'firebase/compat/app';</pre>
3	<p> should not be used as a child of <px>. Not that only phrasing content is allowed inside <px>.	Line 126 of DocumentTables.js	Medium	<pre>if (user?.displayName) { p.appendChild(document.createTextNode(user.displayName)); }</pre>
4	Imported binding 'makeStyles' is not used.	Line 13 of Header.js	Low	<pre>makeStyles,</pre>
5	Imported binding 'useEffect' is not used.	Line 17 of Header.js	Low	<pre>import { useEffect, useState } from "react";</pre>
6	Imported binding 'Settings' is not used.	Line 18 of Header.js	Low	<pre>import { PersonAdd, Settings } from "@material-ui/icons";</pre>
7	Imported binding 'firebase' is not used.	Line 12 of MyDrive.js	Low	<pre>import firebase from 'firebase/compat/app';</pre>
8	Imported binding 'firebase' is not used.	Line 1 of ShareWithMe.js	Low	<pre>import firebase from 'firebase/compat/app';</pre>
9	Required module component 'responseData' is not used.	Line 1 of checkAuth.js	Low	<pre>const { respondUnauthorized, responseData } = require('../util/responses');</pre>
10	Required module component 'Log' is not used.	Line 1 of file.repo.js	Low	<pre>const { log } = require('../logging/logging');</pre>
11	Private field '#groupsRef' is not used.	Line 8 of file.repo.js	Low	<pre>#groupsRef;</pre>
12	Required module component 'respondUnauthorized' is not used.	Line 4 of file.routes.js	Low	<pre>const { responseData, respondSuccess, respondUnauthorized, respondBadRequest } = require('../util/responses');</pre>
13	Required module 'Log' is not used.	Line 2 of groups.routes.js	Low	<pre>const { log } = require('../logging/logging');</pre>

Figure 11: Vulnerability Spreadsheet Sprint 3

For sprint 3 I had to switch over from using Code Sight to using Deepscan. Code Sight had to be paid for beyond my free trial subscription. I found that Deepscan actually highlights and shows more vulnerabilities than Code Sight does which is nice. However, most of the vulnerabilities it found were low level and more about imports not being used as shown.

	A	B	C	D
1	Vulnerability	Location	Risk	Picture
2	Local variable 'getFileIcon' is not used	Line 77 of DocumentTable.js	Low	<pre>77 ~ const getFileIcon = (fileType) => {</pre>
3	Imported binding 'makeStyles' is not used	Line 13 of Header.js	Medium	<pre>13 makeStyles,</pre>
4	Imported binding 'useEffect' is not used	Line 17 of Header.js	Medium	<pre>17 import { useEffect, useState } from "react";</pre>
5	Imported binding 'Settings' is not used	Line 18 of Header.js	Medium	<pre>18 import { PersonAdd, Settings } from "@material-ui/icons";</pre>
6	Imported binding 'firebase' is not used	Line 8 of Sidebar.js	Medium	<pre>8 import firebase from 'firebase/compat/app';</pre>
7	Imported binding 'setAuthHeader' is not used	Line 12 of Sidebar.js	Medium	<pre>12 import { BackendRequest, setAuthHeader } from "../requests/client";</pre>
8	Imported binding 'auth' is not used	Line 14 of Sidebar.js	Medium	<pre>14 import { auth } from "../firebase/firebase";</pre>
9	Imported binding 'client' is not used	Line 16 of Sidebar.js	Medium	<pre>16 import { client } from "../requests/client";</pre>
10	Imported binding 'firebase' is not used	Line 1 of ShareWithMe.js	Medium	<pre>1 import firebase from 'firebase/compat/app';</pre>
11	Variable 'user' is null checked here, but its property is accessed without null check afterwards at line 18	Line 9 of login.js	High	<pre>9 .doc(user?.user.uid)</pre>
12	Variable 'user' is null checked here, but its property is accessed without null check afterwards at line 18	Line 11 of login.js	High	<pre>11 uid: user?.user.uid,</pre>

Figure 12: Vulnerability Spreadsheet Sprint 4

For sprint 4 there were more vulnerabilities as this was the most complete code with the most high-level skill level. Some of these vulnerabilities were deemed not as important to fix as implementing the encryption on the files so they are still open in our code.

Steps Taken to Complete Security Assessments

1. Extensions researched and downloaded.
2. Extensions given permission to use the workspace before each sprint since the workspace was changed to reflect the new code created.
3. Extensions scanned each file on the workspace for vulnerabilities.
4. Each file was clicked on and examined so that the output was shown on the screen.
5. Security Lead checked output from extensions and put the vulnerability, its risk level, its location, and a picture of the line of code on a spreadsheet that was divided by what extension found what.
6. Security Lead checked the code for any vulnerabilities using his own knowledge.
7. Spreadsheet was altered to reflect Security Lead's opinion on risk level.
8. Spreadsheet for each sprint was sent to group so they know what to look out for and fix for next sprint.

7 Work to Be Done

Scalability of this software is a concern; encryption can be a heavy operation and with more users this could become unreasonably slow. This could be solved by splitting the encryption service into a separate container to allow for horizontal scaling.

In the same vein as scalability, performance improvements need to be added. Currently, loading files is slow because the program makes more requests than it needs to and doesn't have any caching.