Secure File Vault
Functional Specification

Author: Jack Hooton Byrne
Student ID: C00230173
Project Supervisor: James Egan
Recipient: Institute of Technology Carlow
Date: Friday 27th November 2020

Abstract

Secure File Vault protects individual files or data by encrypting them using different encryption methods. When the user wants to view the files, they login into the cloud to view the files which have been decrypted for the user. The cloud is a Raspberry Pi that stores valuable data.

Have you ever run out of storage space on your laptop or computer and had to invest in external memory? Have you ever thought about any other options?

Unfortunately, this has happened to me and has led me to create my project. Many people each day run out of storage space on there on devices. But not many people think about using cloud storage to store their data. People are afraid of storing their data in a cloud because they don't know where the cloud is. A cloud is just a set of servers that are stored on an offsite location. Users are also afraid of how their data is being stored. Recently there have been many cyber-attacks on cloud systems. According to the 2020 Trustwave Global Security Report, attacks on cloud services have doubled from 2019 and have accounted for 20% of investigation incidents. Cloud systems are now the third most targeted environment for cyber-attacks. The purpose of my project is to create the most secure cloud system on the market.

Table of Contents

Introduction

This Functional Specification document is a document that provides detailed information on how the Secure File Vault application will function and the requested behaviour. This document provides a clear idea of the actors interacting with the system and what they are expected to interact with the system, what they should expect from the system, and what they cannot do while interacting with the system.

Project Scope

This document focuses on all scopes of the project, considering every aspect of the application, ranging from the Raspberry Pi database to the frontend and backend of the Android Application and the operation of how each function is carried out in the application.

General Description

Secure file Vault is an android application that allows a user to upload any file which is encrypted and then stored securely on a separate database. The database which stores the encrypted files will be stored on a Raspberry Pi. When the user wants to view the files, they select the file and will be given two options download or delete. At this stage, the file will be decrypted. When a user wants to register an account, it is done by using XAMP. The email and password are encrypted to the highest security standard. The application will also be secure against many common android Application vulnerabilities.

User Characteristics and objectives

Target market

This system is targeted at individuals who need to transfer files to a secure cloud, and the users are expected to be comfortable with the application. The users can upload any sort of file to the cloud and can access the file at any time. As explained in the Research document for Secure File Vault, many solutions already exist in the android app market. However, the unique aspect of Secure File Vault is that it is for personal use, and the user owns the server. This is a significant advantage against Secure File Vaults competitors.

Objectives

Desirable characteristics of the application

- User-friendly interface
- Application is easily navigable
- Secured with modern cryptographic suites
- Responsive
- Secured against common web vulnerabilities

System components

Secure file vault consists of two significant components hardware and software. The hardware components are the raspberry Pi and the hard drives for the cloud.

Hardware Components

Raspberry Pi [3]: The Raspberry Pi will be used for creating the cloud server. To create the cloud server, the hard drives will be connected through the USB slots. The image below shows a Raspberry Pi and how it can connect to the hard drives.

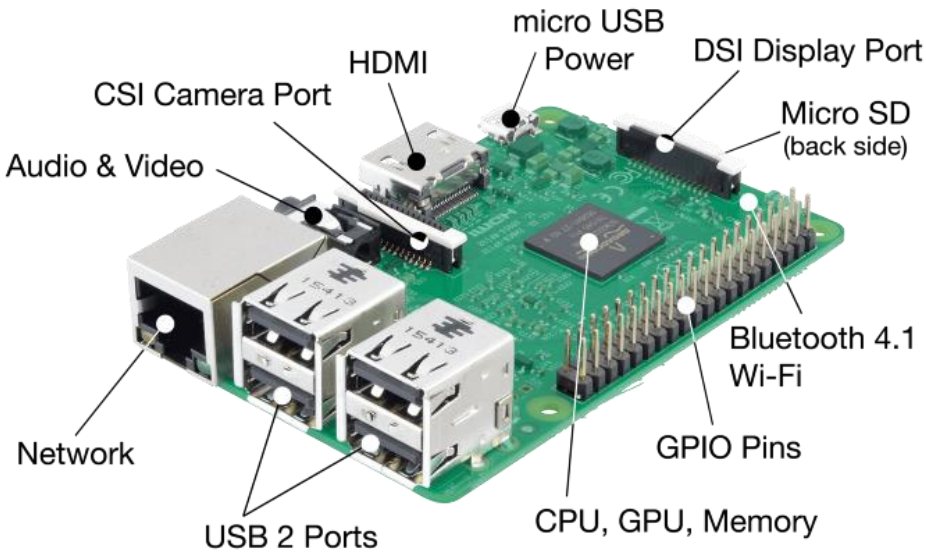

Figure 1: Raspberry Pi

(Mulonda, 2020)

Hard Drives: Multiple hard drives will be connected to the Raspberry Pi to form a server for the cloud.

Software components

Java Application development will be used for making my application for the cloud. I will be using many APIs to form the security aspects of the application. I will also use APIs for the cryptography part of the application as they provide outstanding security and integrity for the user's files and credentials. I will also use strong security techniques to prevent against common vulnerabilities.

Operational scenarios

There are two steps for the users to use this application: initial setup and general usage.

Initial setup

1. The user visits the google play store and downloads the app.
2. The user registers with the correct details and logs in.

Now the users are ready to start uploading files to the cloud.

General usage

1. The user opens the application and logs in.
2. The user can upload any file to the database where it will be secured properly.
3. The user can download the files they have uploaded previously to view them.
4. The user can change their password if they wish to.

Project Plan

Sprint #	Plan	Due Date	Deliverable
0	Complete research material	13/11/2020	Research to show what cloud storage is and how to set it up
1	Java app development	18/12/2020	Learn Java app development

3	Start application development	10/1/2021	Create a secure registration and login page for users.
4	Develop backend technologies	/2/2021	Connecting the app to the database.
5	Encryption and Decryption	4/4/2021	Encrypting and decrypting user's files to be uploaded.
6	Testing	17/3/2021	Test application note any flaws that may occur
7	Debugging	20/3/2021	Fix any problem that has occurred from testing
8	Final application	30/4/2021	Making sure the app is of exceptional quality

Functional Requirements

#	Function	Description	Criticality	Tech Issues	Dependencies
1	Registration	Allows a new user to make an account	High	N/A	The database must be live
2	Login	Allows any users to log in	High	N/A	Must be registered

3	Upload and Download files	Allows a user to upload a file and download a file they previously uploaded	High	N/A	Must be logged in, and the database must be live
5	Change password	Allows any users to change their password	High	N/A	User must be logged in and will also have to provide their current password
6	Logout	Allows the users to logout securely	High	N/A	The user must be logged in

Use case Diagram

Registration

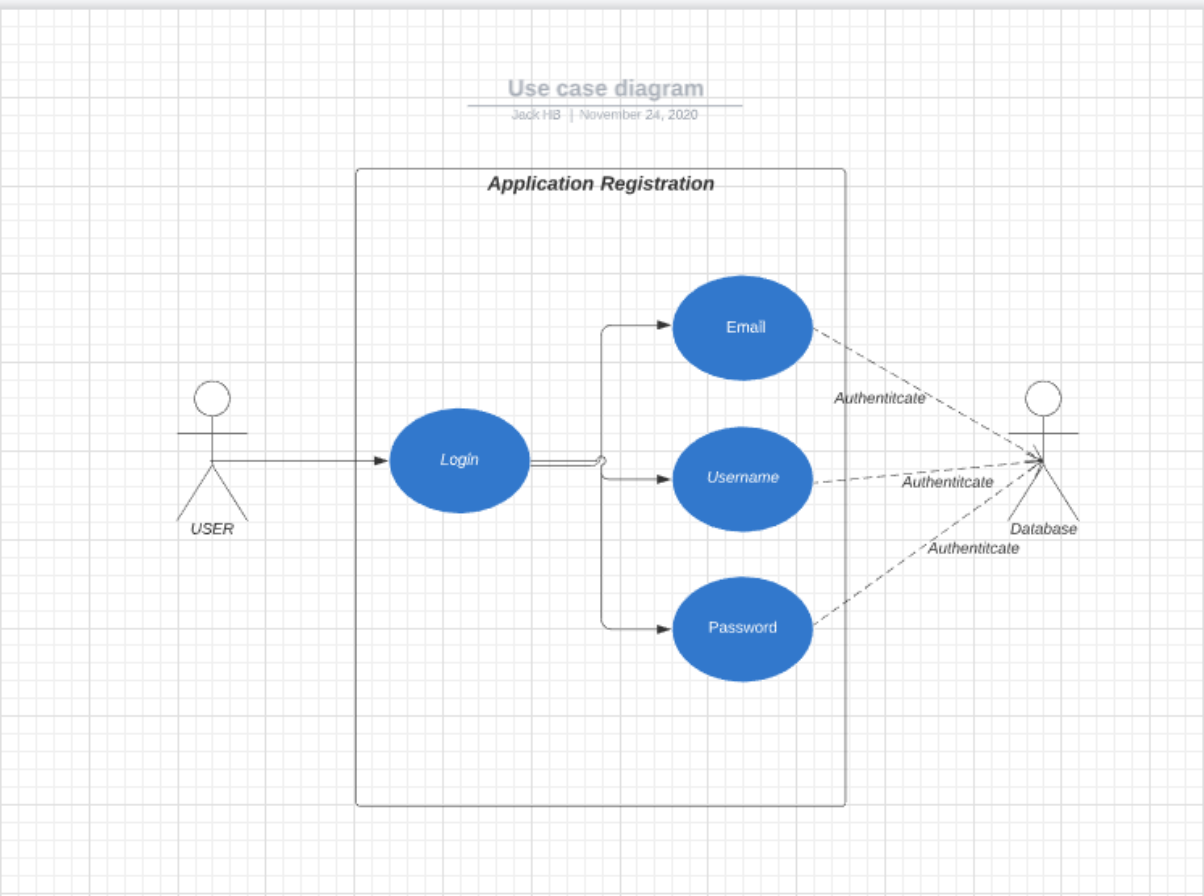

Figure 2: Registration Use Case

Primary Actor

User

Preconditions

None

Success Guarantee

The user is now able to register on the app. Their Username and password are saved to the database.

Main Success scenario

1. The user downloads the app from the google play store
2. The user clicks the button to register
3. The user enters a valid username and password
4. If the Username is available and the password is strong enough, the user's account will be set up. The user's credentials will be saved to the database

Alternative flow

- The Username or password is not available, the user is prompted to try again with different credentials.

Application Login

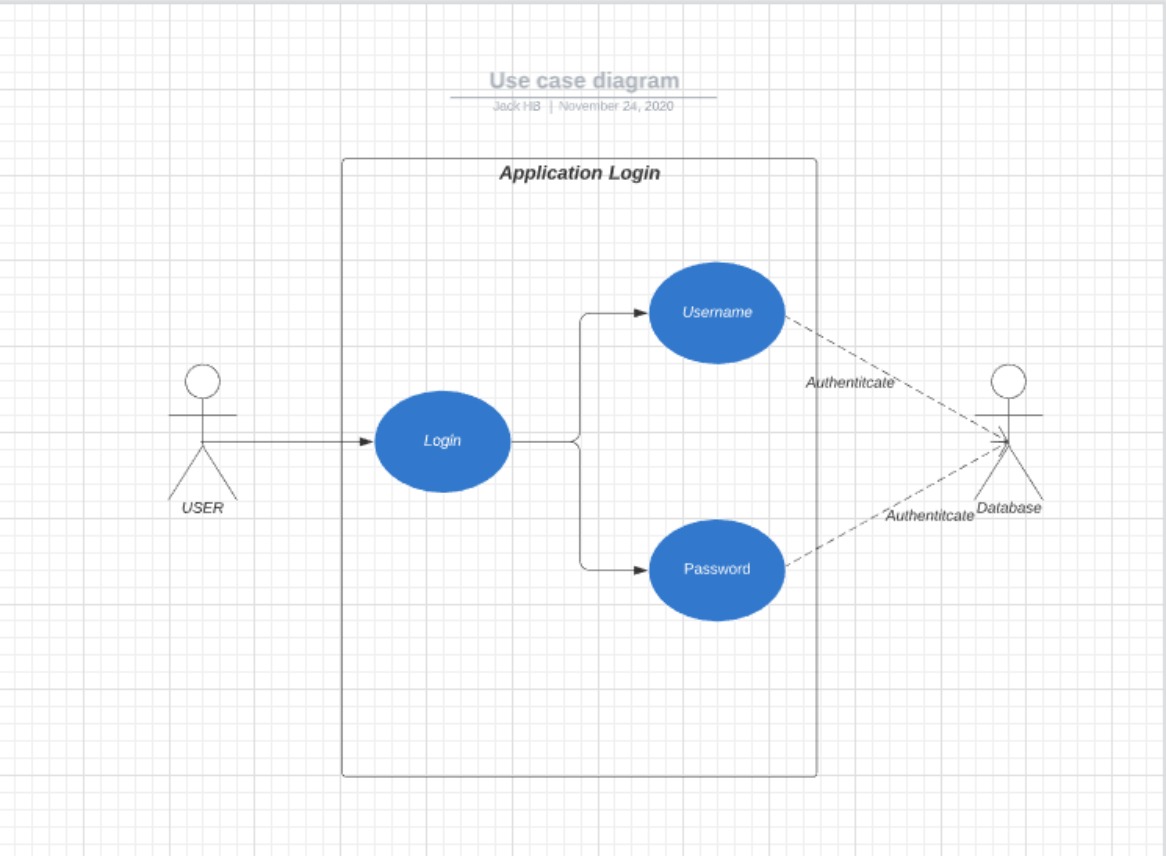

Figure 3: Login Use Case

Primary Actor

User

Preconditions

The user must have made an account by registering.

Success Guarantee

The user successfully authenticates with the app and can now login.

Main Success scenario

1. The user visits the app.
2. The user clicks the button to login.
3. The user enters a valid username and password.
4. If the Username and password are valid, the user can login successfully.

Alternative flow

- The Username or password is not valid, the user is prompted to try again with different credentials.

Authenticated user File upload and download

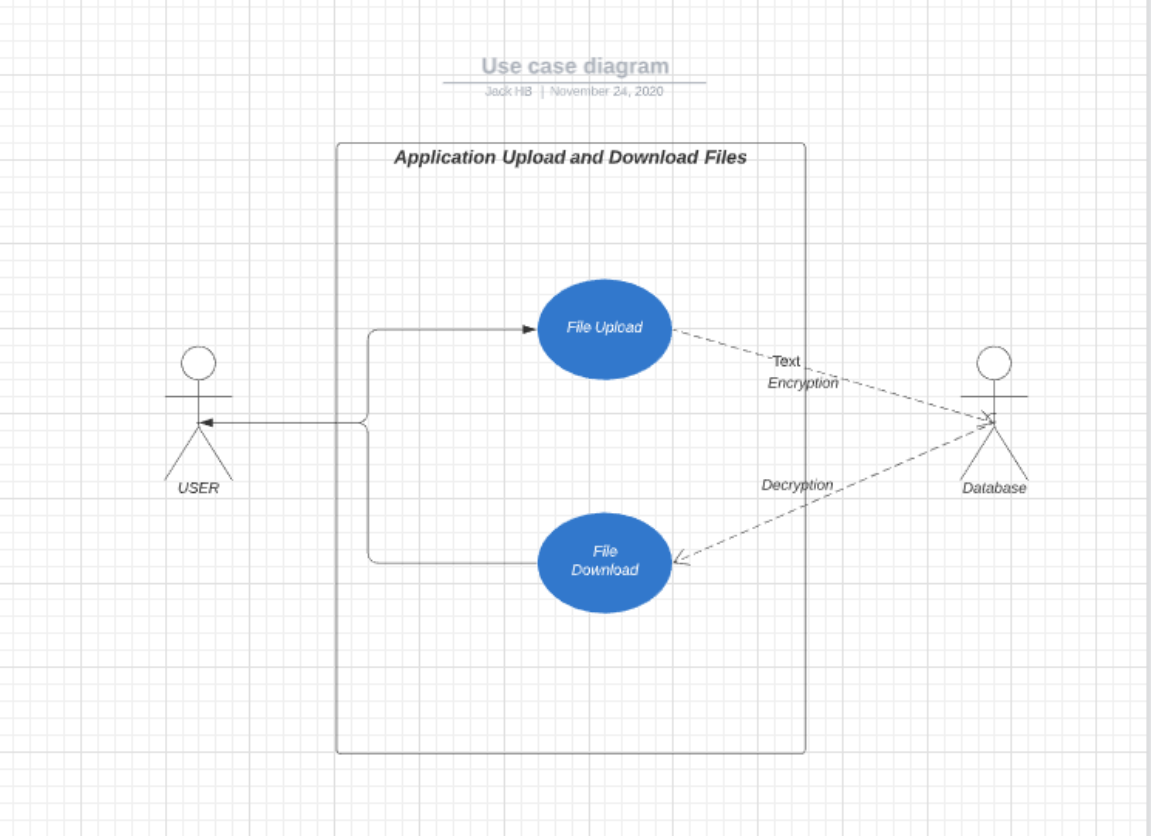

Figure 4:File Upload and Download Use case

Primary Actor

User

Preconditions

The user has successfully logged in.

Success Guarantee

The user is now able to upload files and can also download the files they have uploaded to the cloud.

Main Success scenario

1. The user visits the app and logs in.

2. The user can upload any file to the cloud.
3. The user can download any file from the cloud to view it.

Alternative flow

- The user uploaded the same file twice and asked if they want to overwrite the old file.

Change Password

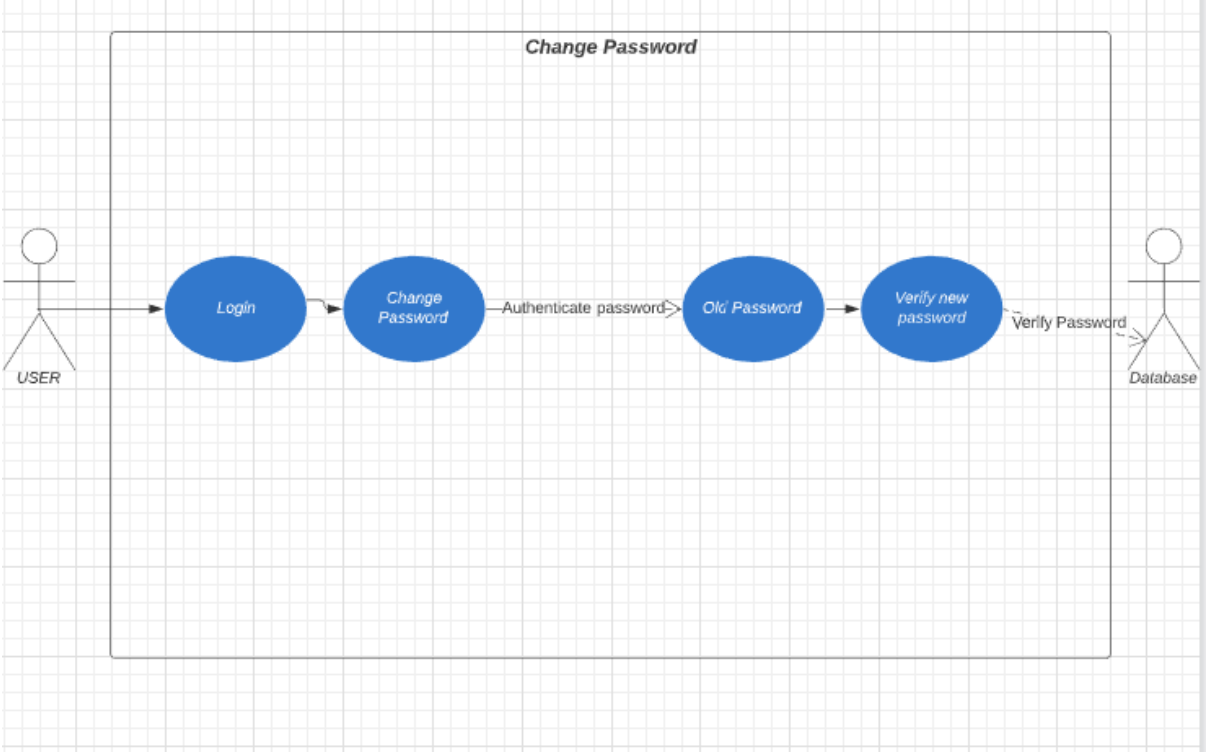

Figure 5:Change Password Use case

Primary Actor

User

Preconditions

The user has successfully logged in

Success Guarantee

The user is now able to change their password by entering their current password and entering the new password twice

Main Success scenario

1. The user visits the app and logs in.
2. The user clicks on change password.
3. The user enters their current password and their new password twice
4. The user will be prompted that their password has been changed successfully

Alternative flow

- The user will be prompted to try again if their current password isn't correct or if the new password is too weak.

Supplementary Specification

To define the supplementary specification for this application, the FURPS+ model will be used. FURPS+ is an acronym that stands for Functionality, Usability, Reliability, Performance, Support, and the plus is used to specify constraints around the design. Interface and implementation, among a few other things. In this section, I will discuss each part of the FURPS+ model.

Functionality

For this application, the following functionality required is:

- The application should let user's login incorrectly
- The application should allow the users to upload any file to the cloud. One of the users has logged incorrectly.
- The application will allow the users to download and view any files they have uploaded.

Usability

This part should describe how easy it is for the users to use the application

- The application should be user-friendly.
- The application should be easy to upload files and download files
- The application should be easy to navigate around

Reliability

This part will show the possibility of the application failing.

- The application must always be connected to the cloud.
- The Raspberry Pi should always be on for the cloud to stay alive.

Performance

The performance describes how the application behaves when the users interact with it in various scenarios.

- The uploading of files will vary in time as each file may be of different sizes.
- The downloading of the file will also vary in time as each file may be different sizes

Supportability

Supportability will ask questions about the application, such as whether it is testable, serviceable, installable, and monitored.

- The application will be tested to see whether it is user-friendly.
- The application is easy to service as the admin can add and take away hard drives to add more space for the cloud.
- The application is easy to install the user must look it up on the google play store and download it.
- The admin can monitor the application at any time.

+ (Security)

- All the user's credentials will be stored securely as they will be encrypted and stored in the database.
- The user's files they have upload will also be encrypted and stored in the database.
- The application will secure against many vulnerability attacks.

Testing

Testing is a significant part of my application as I want it to be the most secure cloud on the market and the most accessible cloud storage application for users. To test the application, I will ask an individual to use my application for a day and note any issues. The application will be tested against common vulnerabilities and see if the proper prevention has been put in place. If the prevention has not worked, I will try to develop the application further and test it again against common vulnerabilities.

Bibliography

Mulonda, Y., 2020. Raspberry Pi 3 — Shell Scripting — Door Monitor (An IoT Device). [online] Medium. Available at: <https://medium.com/coinmonks/raspberry-pi-3-model-b-shell-scripting-door-monitor-b44944f82d87> [Accessed 26 November 2020].

Table of Figures