* 1. **Introduction.**

Cloud Winterstore is to let software developers keep their clients’ data more secure than ever, by bringing more robust access control to cloud storage. Instead of securing files by naming them with an n long character string which makes it harder to guess the name of the file, Cloud Winterstore will allow a client to set the access control measures for a file upon upload. For a person to view this file they will have to be either authenticated as the owner of the file, or the key to that file exists in the current authenticated user’s key set.

This will reduce the possibility of a person gaining access to that file, cause all a person had to do previously was obtain the URL of a file, which would give them access to that file.

Removing the long string naming conventions for files will make it easier for a developer to manage their storage, cause now they can make use of names that make sense.

* 1. **Background of The Study.**

One thing to note is that I am a very big fan of google firebase (https://firebase.google.com/), my favorite part being the fact that it is free to use:) However there are other factors like it is very easy to use, hence you do not need to know a lot about backend to build an entire application. Which I have done so by building 2 of my applications on it, Solve it (https://github.com/rbryanben/Solve-It) and ChatSock (https://github.com/rbryanben/ChatSock-v1.0.2). Bad choice to use Firebase with a C# application by the way!

I mostly use Firebase Authentication, RTD and Storage for my applications, and one thing I have noticed is that they offer the least security possible when it comes to files. Files are created with a n char long string so that a person may not easily guess the path of the file, but if the person is authenticated, they can view any file they want, they just have to obtain the URL of that file. No checks are done to verify if the person can access the files.

This problem I came to realize when I figured that Midlands State University keeps their student’s e-learning profile pictures in 1 folder, which is hosted by NGINX. The person who created the system did try to restrict access to these profile pictures by restricting access to the folder as whole. But injecting a student’s registration number to the end of the absolute URL of the images folder will allow you to view the person’s profile image. This gets way worse when you take note that all students emails can be synced to your android device, because MSU uses Google Mail for their students. Hence it allows us to get data on every student that is on campus, cause we already now all registration numbers.

* 1. **Problem Definition.**

I have categorized all problems into 3, because a developer either hosts their files on a cloud service like Amazon S3 Buckets, or on a webserver like Apache Tomcat. The 3rd category arises on a bases that some developers may not host their files on the webserver but pull the files from a SQL database.

|  |  |
| --- | --- |
| **Cloud Storage Problems** | * The current cloud storage services do not offer robust access control. * Some services like Firebase are limited to google based software i.e., Android. |
| **Webserver Problems** | * Webservers (NGINX, APACHE Tomcat) do not offer a strong logic layer to handle multiple files with access control. |
| Pulling from A SQL Database | * SQL database cannot keep very large files. * Querying BLOB data on a SQL database will take a lot of time, hence we will have latency issues. * A SQL database upon querying the data, will load the data into main memory, hence a 12GB file will require 12GB+ Ram on the host machine. |

* 1. **Aims and Objectives.**

The system will be divided into 4 parts. Hence why I have created objectives as per each part. **[High], [Med]** and **[Low]** represent priority levels.

|  |  |
| --- | --- |
| Backend (API Calls only) | * Allow folder creation with access control **[High]** * Allow file upload with access control **[High]** * Deletion of a file **[High]** * Creation and deletion of client accounts **[High]** |
| Web Application (Landing) | * Allow a developer to create an account. **[High]** * Allow a developer to recover an account. **[High]** * Show user documentation. **[Med]** * Show how a person can contribute to the service. **[Low]** * Allow a developer to purchase a package. **[Med]** |
| Web Application (Console) | * Allow a developer to manage files from a superuser perspective (delete, add, access control) **[High]** * Allow a developer to monitor their account. **[Med]** * Allow a developer to manage client accounts. **[High]** * Allow a developer to view statics on upload and retrieval of files by clients **[Low]** |
| Libraries (.NET Core C# and Java Android) | * Allow a client to utilize API call s with ease **[High]** * Allow a developer to build and develop apps easily **[High]** |

* 1. **Methodologies and Instruments**

On the frontend side I will have my libraries and the web application which will be communicating with the backend via API and Channels. Meaning my Web Application will not just be following an MVC model, but MVVM as well.

The backend will have Django as my webserver, dealing with all the logic handling. Hosted on a UNIX socket by NGINX and GUNICORN as my gateway. Daphne will be used to handle all socket connections to the webserver.

Files will be store in Mongo GridFS, which allows me to stream a file instead of loading it into main memory. Lastly, I will be using MySQL to store all data that needs constraining.

Libraries are to be built natively, .Net using Visual Studio and Android using Android studio.

Diagram

Description automatically generated

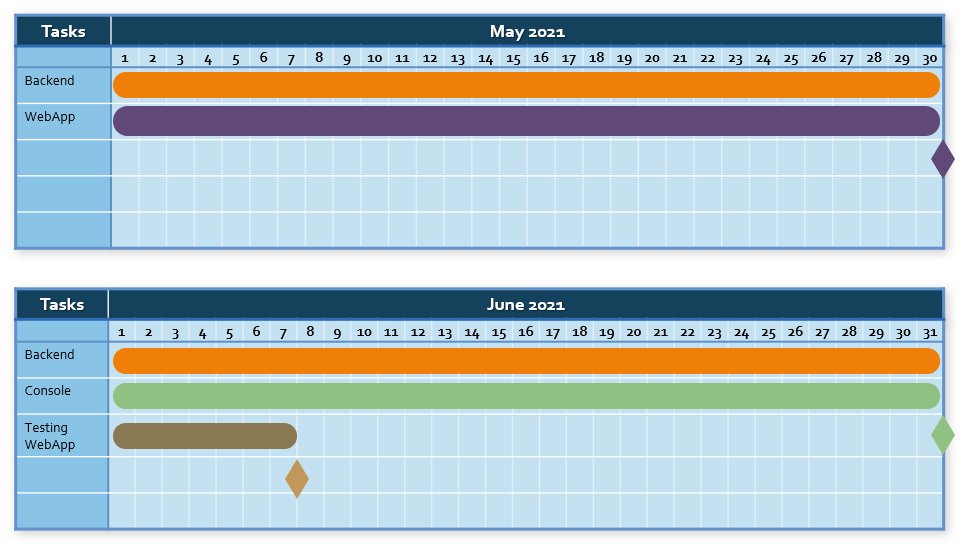
I believe the Agile development methodology will best suit the system, given the project is quite big to oversee. There needs to be room for iteration, cause some parts are depended on each other, hence I may need to revisit a part to make another work.

* 1. **Expected Results and Significance.**

The expected result is a service that allows a developer to register and subscribe to a package and setup their storage. Which they will be able to build applications using the storage service. A client being able to upload a file, and add access control to the file, hence keeping it secure.

The significance of the project is to provide security for developer client’s data, cause in this era their data is very valuable.

* 1. **Timelines**

****

**Timeline

Description automatically generated**

I expect the project to be in beta mode by the 8th of August 2021, with all components working coherently as a whole, all 7 milestones reached.

Milestone

* 1. **Conclusion**

The project is hosted on the repository <https://github.com/rbryanben/Cloud-Winterstore>, issues can be raised here.