INFO 4110: S50

Project Report- Cloud USB

Team 3 Members:

Harshpreet Singh Saini -100345054 - harshpreet.saini@email.kpu.ca

Steven Saito -100296655 - steven.saito@email.kpu.ca

Thomas Federau- 100335326 - thomas.federau@email.kpu.ca

Table of Content

Introduction

Cloud storage is quickly becoming a very popular option with bonafide applications like Google Drive. An application that allows a user to access their files and data whenever and wherever they want. For our project, we wanted to create something similar and see how this kind of tool works. Our team wanted to create a website where authorized users can upload files to a storage on the cloud. The files will be uploaded to the cloud and managed separately from the website. The website will link the user to a google cloud bucket and its content which is hosted on the Google Cloud Platform. The user will be able to upload a file and have access to that file from any other device that is connected to the internet.

Background

Cloud computing or more commonly called “the cloud” is the delivery of on-demand computing resources which ranges from applications to data centers. This is usually setup on a pay-for-use basis. There are different types of clouds which bring different benefits to their users but also can have different requirements for their usage. To name the popular models, there is the public, private and hybrid cloud.
Cloud Computing is a very powerful tool that offers many advantages. Such as its availability, mobility, flexibility, integration and cost efficiency. There are still disadvantages to Cloud Computing such as its dependence on a constant internet connection, its security, the vulnerability of not having complete control over the system and not knowing the underlying infrastructure. So the cloud is not a perfect solution for everything, but is proving to be very valuable for a lot of different needs..

Our Objective

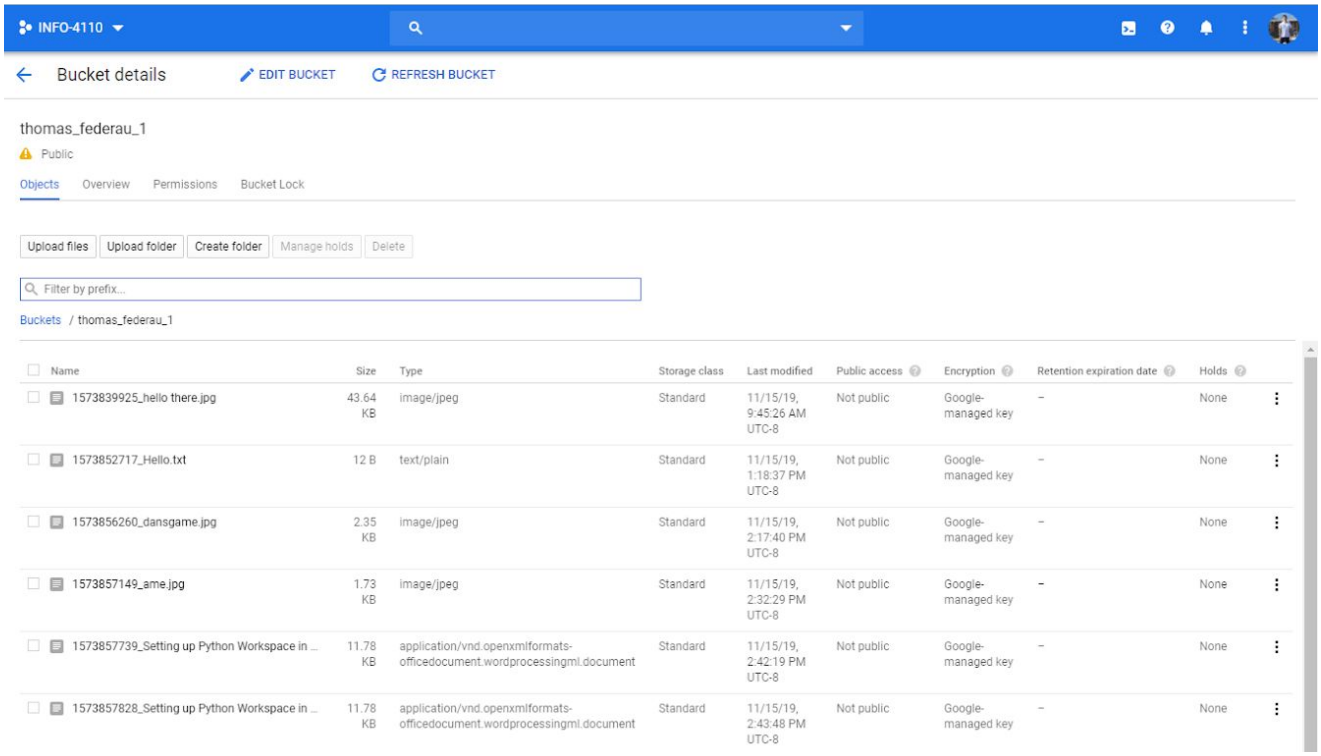
To create a storage application that the user can use to access files from anywhere on any device with an internet connection.

Why do we need Cloud Technology

Our project needs cloud technology because the else the user would not be able to access their data from truly anywhere. One of the biggest upsides to cloud computing is the ability to access an app or data from anywhere. Our cloud storage tool allows the user to upload content from one computer and then can access it on another.
A user could have a USB with them, but if they forget their USB, then cannot access their data. Our cloud solution can be more reliable in terms of transporting and making the data easier to access. Since it only requires the user to have a device that can connect to the internet and it removes the need for a USB port.

Our Design

Our project is a website that utilizes Google Cloud Platform and a couple of its tools, specifically Google App Engine and Google Storage. We will be creating a Google Cloud Platform project that will host and hold our entire project. In this project we will use Google App Engine to run a LAMP instance, which will be our website. This instance can be accessed via an IP address associated to it and we will remote desktop into the VM to make changes. We will use Google Cloud Storage to hold the buckets that will store all of our content. Our website will connect to a single bucket which will hold all of the user's content.



a. A peek into how the google cloud interface looks

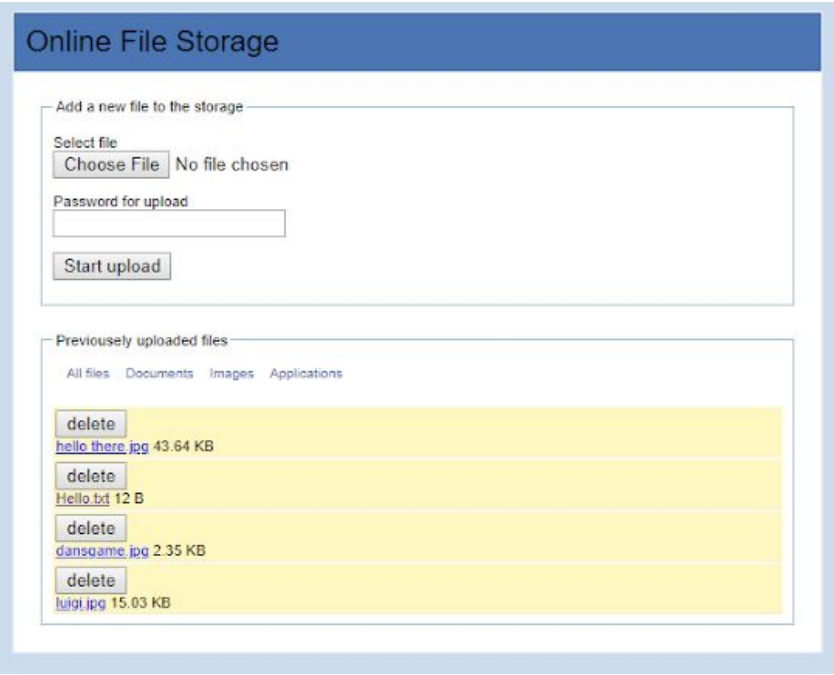
Our website will have a simple design that allows a user to store content on the cloud. The base of our website is it will have an upload section, password section, and section to list all of the cloud storage's content and the ability to delete it.

There will be an option to select a file for the user to upload. The user can drag an item over top of the field or can choose a file by clicking the button and using file explorer to find their desired item. The user will be required to have a password in order to upload a file. This is our solution to have some security at the moment. And then there will be a button that executes the upload process for the user's selected file.

The user can also delete files, but must enter the password as well to do so.

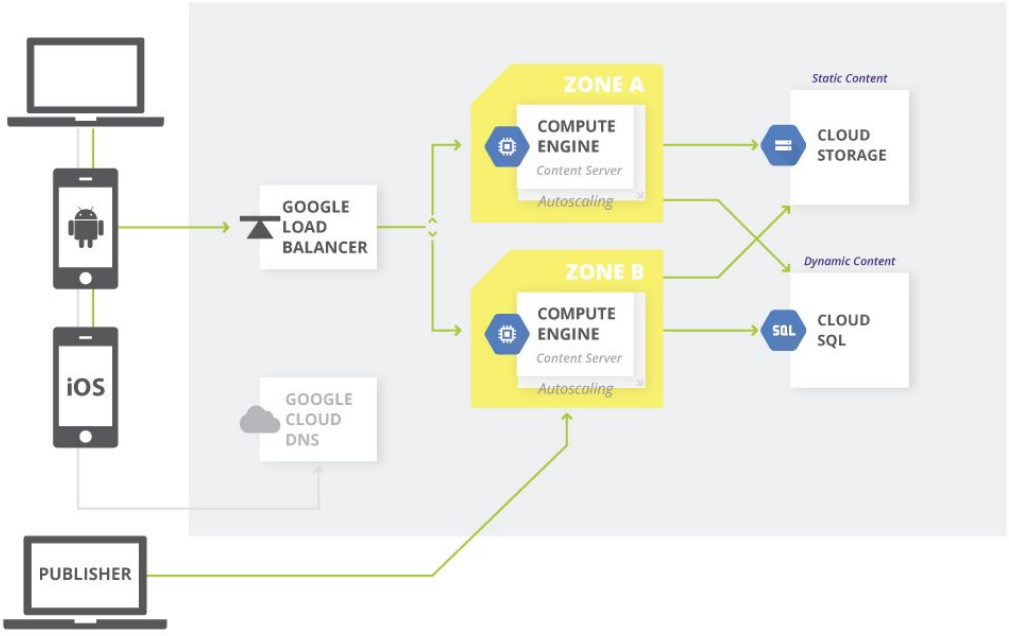
In another section of the website, we will list out all the contents that are in the Google Storage bucket. Thomas imagined an additional feature which allows the user to sort the files by file

type. So if you press on those different headers, each of them would list out files with specific file extensions.
We will add constraints to some of the user's actions adding if-else statements and exceptions to catch not recommended actions.



b. How the project looks like

The map below quickly illustrates how our system is mapped. The left side being the devices that will use our website. They connect to Computer Engine, which is what runs our website on a VM instance. And then the far right side is Google Storage, which is all the buckets holding the users files.



c. A map showing how our system works

Implementation

We will be creating our project using Google Cloud Platform. Our plan is to create a Google Cloud Platform Project and then learn about cloud buckets to store all of the user's content. We would then create an instance to host our website. Our instance will be the connector for the user to view the bucket's content and to utilize it. We would connect to our VM instance through a remote desktop connection.
The plan

Why Google Cloud Platform

We chose Google Cloud Platform because of the documentation that it had to support our project idea. We found numerous documents that guided us through the development of our project. Google Cloud Platform also gave us 300 free credits to use on whatever we wanted. When we tried AWS, we had been charged for a couple of tools. We were not entirely sure why as we thought we would be given free trial stuff as well. But we ultimately went with Google Cloud Platform because we knew that we could try it out for free.
We were also influenced by Google Drive. We figured by building our project on the same platform as Google Drive, chances of it working and finding documentation on building such a tool would be high.

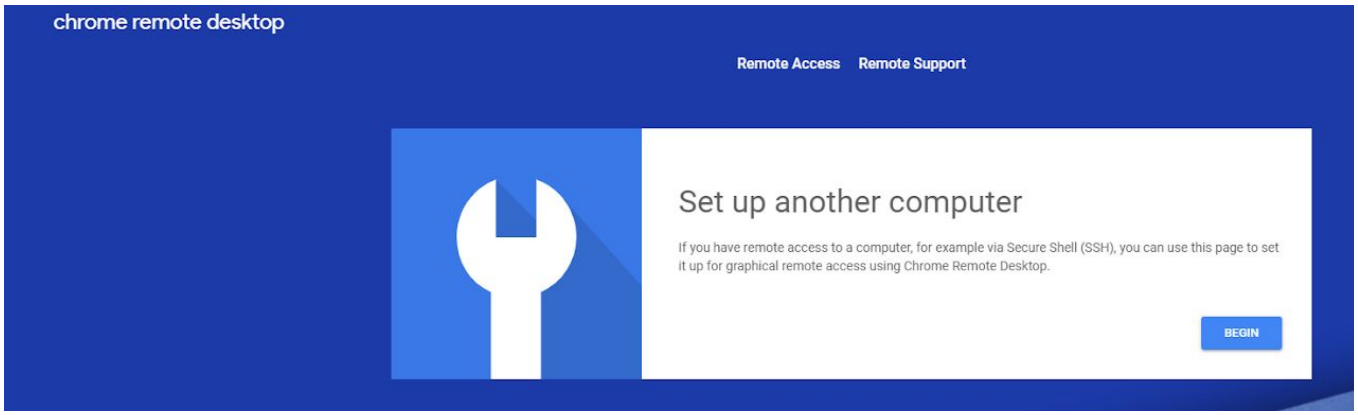
How to Access our Project

Fast Notes

Website IP Address: **http://34.83.145.187/**
Password: **123456**

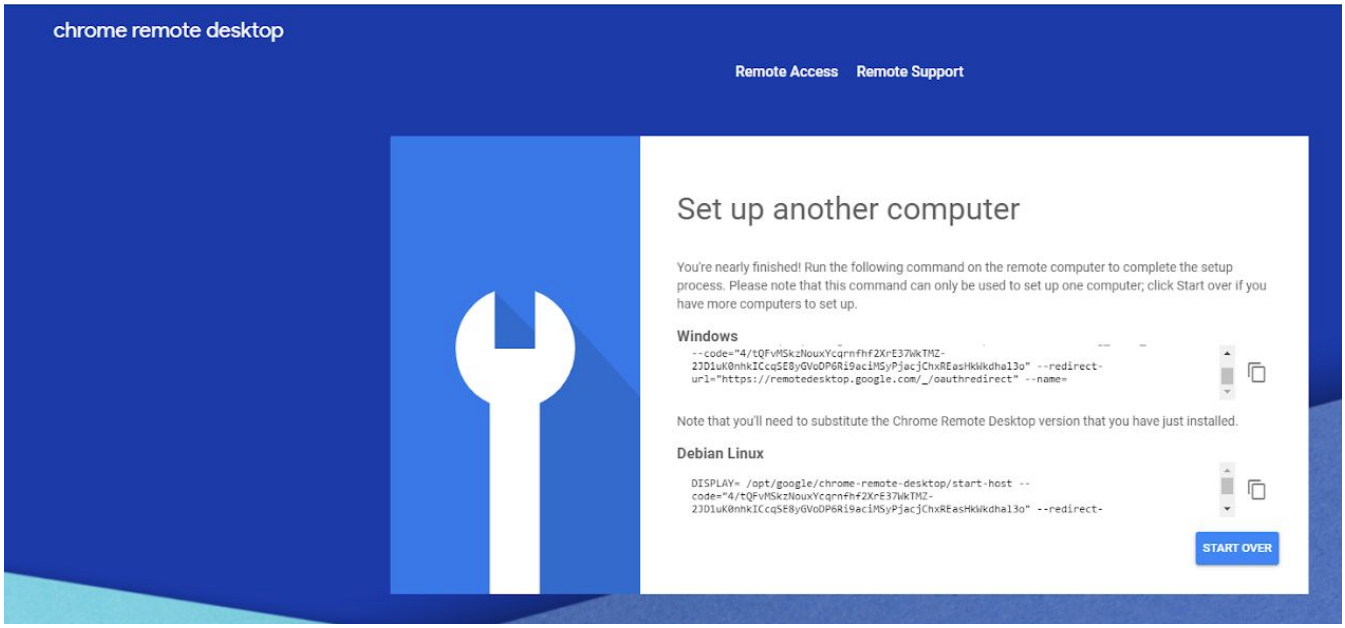
Walkthrough

You can access the Google Cloud Platform project area via invitation from one of the admins. So an admin of the project would need to invite you and then you would click on a link that would give you access to the project.
To access the application you only need the IP address which is
http://34.83.145.187/.
To access the VM instance you can access it via Google Remote Desktop. But you can only access it once an admin has given you access. It is not through an invitation system. So first you would go to https://remotedesktop.google.com/headless.



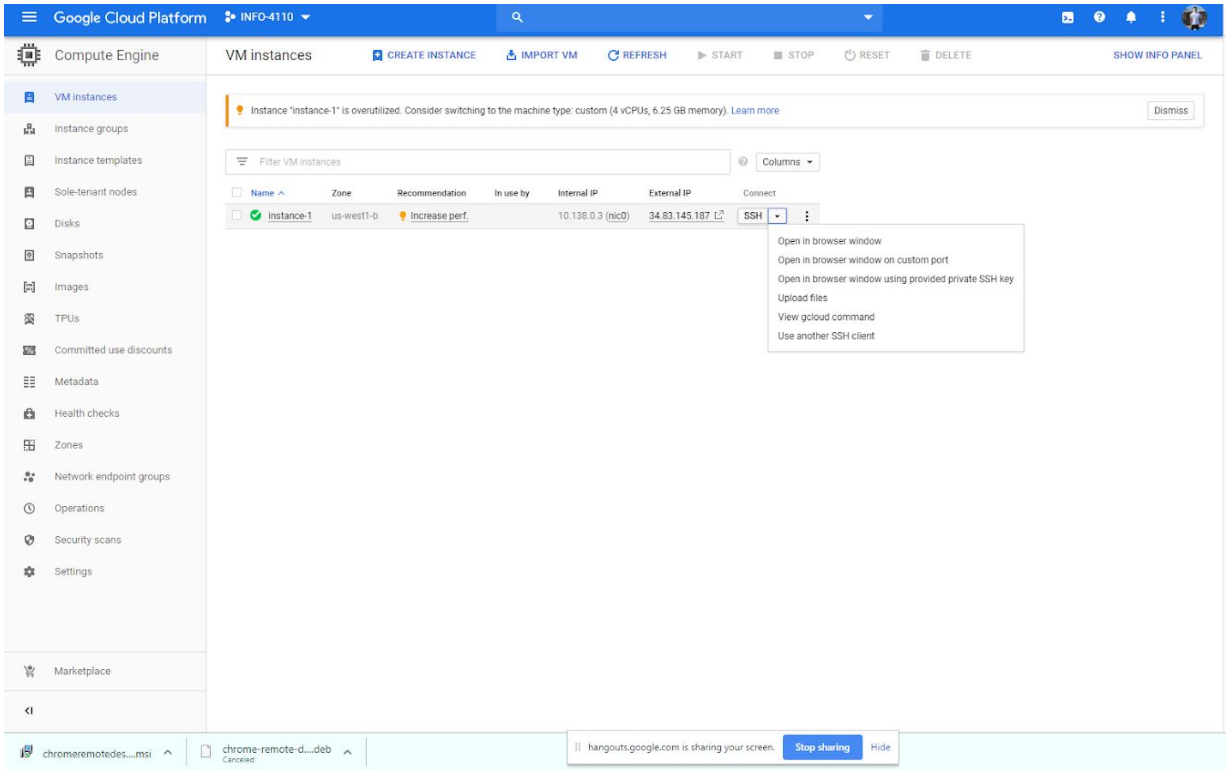
d. Setting up remote desktop using google chrome desktop

Click Begin and then Next a bunch of times. Eventually you will be prompted to log into your gmail. This is of course under the assumption the admin of the website has provided the user privileges. Once logged into your gmail, you should see a page like the image below.



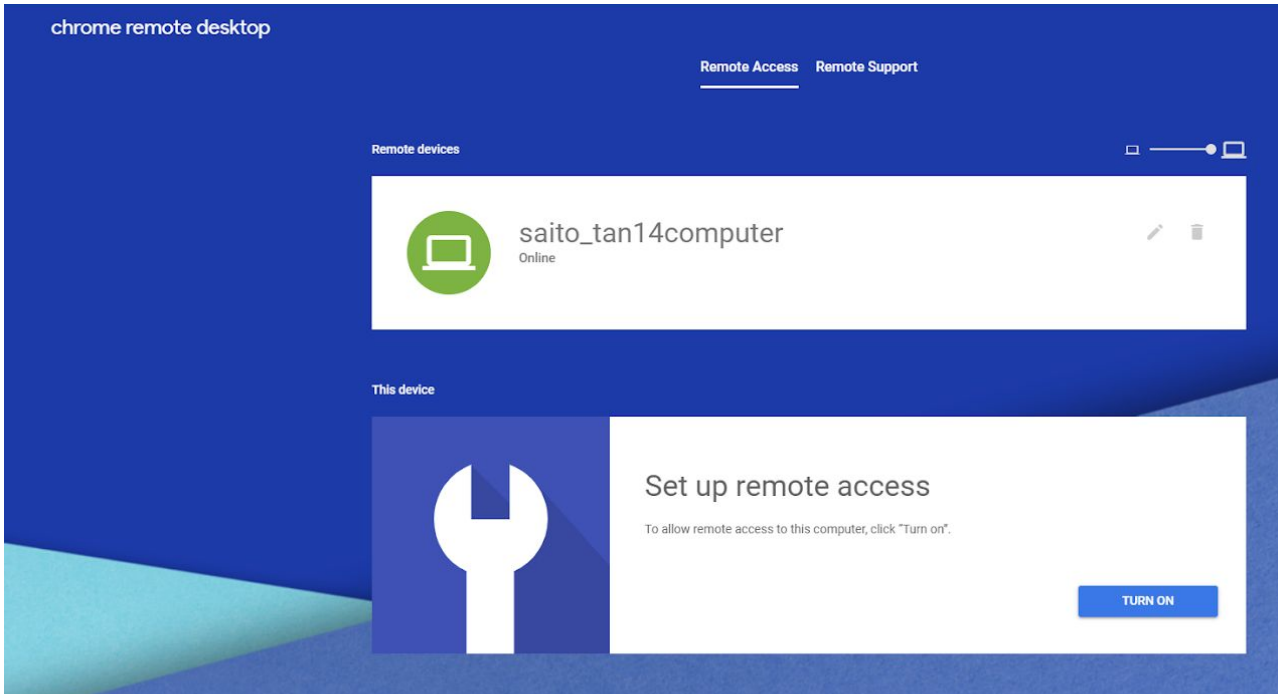
e. Once login done from email, you can see the screenshot as above

You will copy the Debian Linux text.
Now you will go to the Google Cloud Platform Compute Engine tool page of the project that you were invited to. Select VM instances and the page should look like the image below.



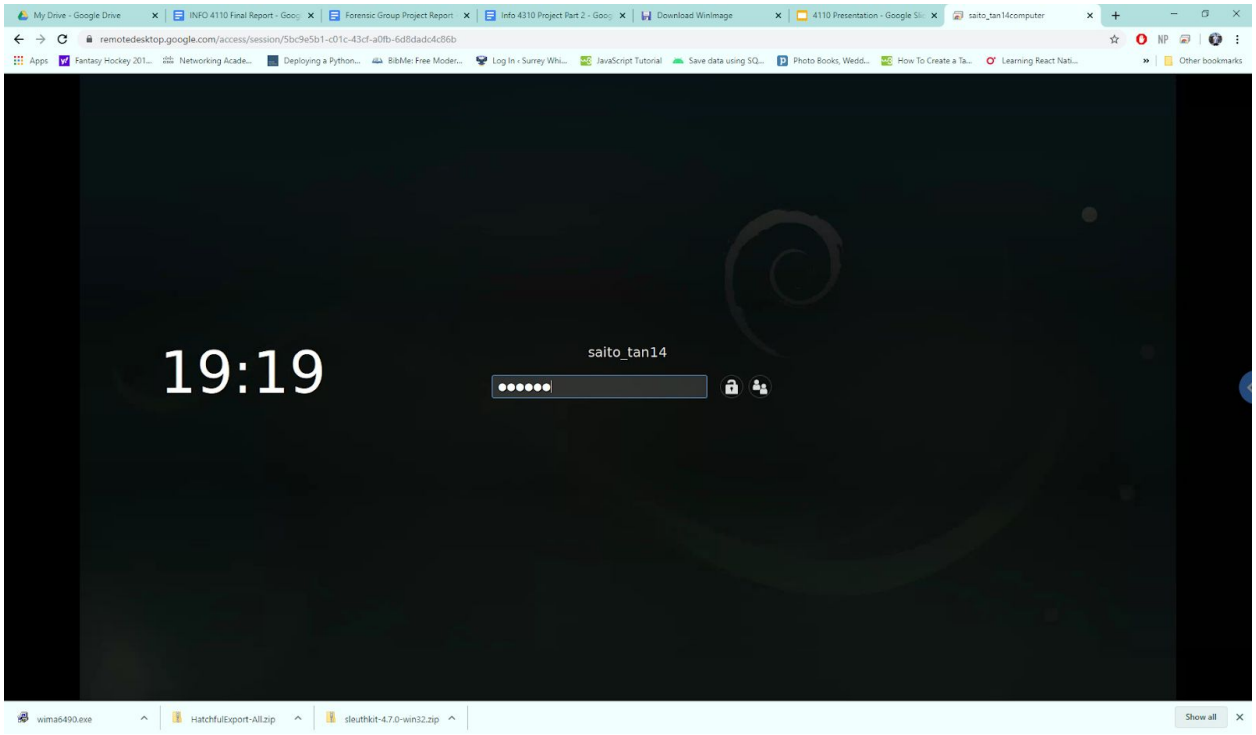
f. Instances page- This is how the instances look like

Click on the SSH button that is inline with the VM instance. And then copy and paste the text that you had copied from Chrome Remote Desktop. That text will allow your browser to access the VM instance.
Now when you return to Chrome Remote Desktop, click on Remote Access near the top and now you will see a new remote connection.



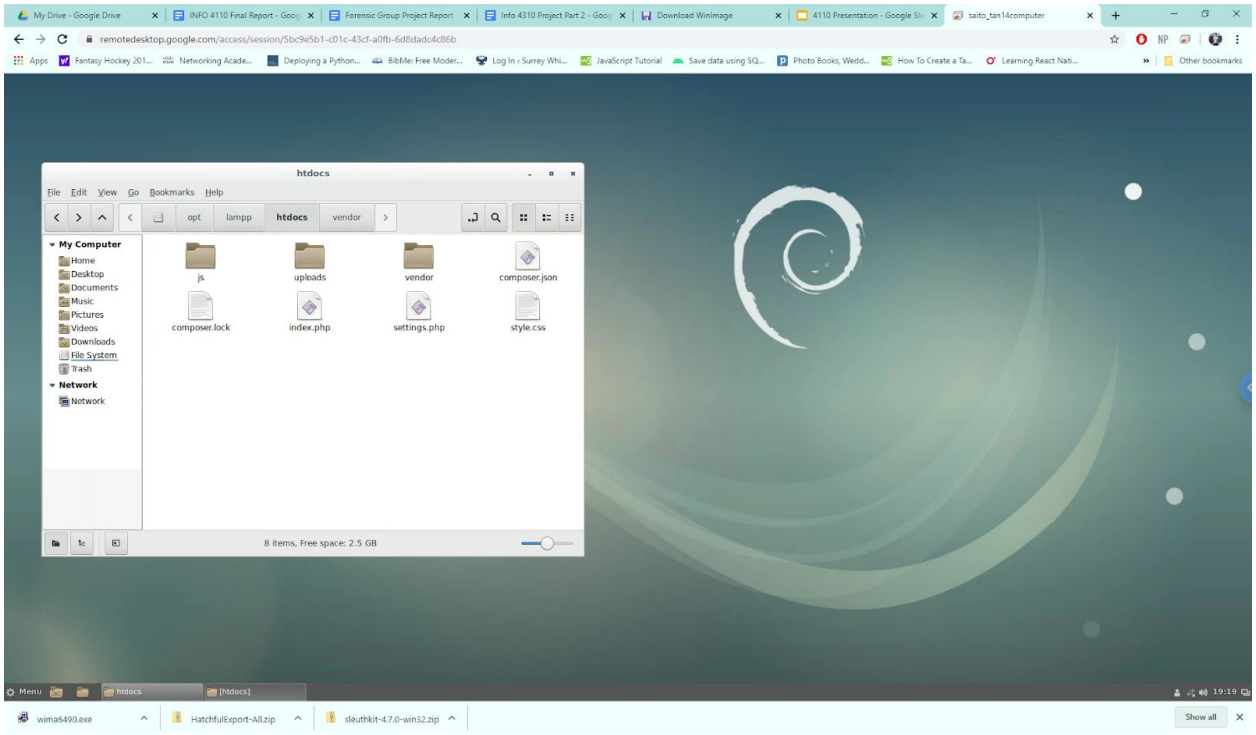
g. A new remote connection can now be seen for the newly set VM

You can now click on your new remote device and connect to the VM instance that contains our website. You will be prompted twice to enter your password.



h. The password prompt will ask you to insert your password as above

The first time you log in you will create a password. We found that multiple people can log in at the same time and make changes, but changes would only be made if the other user saved and then refresh the file.



i. How the VM looks once the login has been completed

Project Schedule

September 23	- Research project ideas
September 30	- Decide on a project idea and create a project proposal - Decide on what features we hope to implement
October 11	- Perform research on how to implement project idea
October 18	- Redesign project based on new research - Familiarize ourselves with the Google Cloud platform interface - Create a bucket
October 25	- Create our first instance (A website) - Evaluate our progression and alter our goals as necessary
November 1	- Create an interface
November 8	- Connect our instance to our Google Cloud Platform Project - Have a basic working model of our project
November 15	- Implement one of our more advanced features to our project

	or keep on working on the basic functionality of our project. - Improve the interface - Repair any found bugs
November 22	- Complete a draft of our project's final report - Fix any more bugs - Speculate if we will have time to add to our project - Create a presentation and present our project.
November 29	- Complete report and submit

Project Resource Planning

We plan to share the workload equally. In the beginning we worked together in the initial steps of developing our project. To help each other out and give us all a chance to learn how to develop a cloud application. Though as the project has progressed, we have divided the work to speed up the development of our project.

- We all contributed the same efforts for the research, designing the project and setting up the cloud project.
- Steven and Thomas took the lead when creating our instance and getting a website up and running.
- Harsh and Thomas were responsible with the development of the base model of our project and coding its basic functionality and usage.
- Steven and Harsh worked to write up the majority of the final project report while Thomas touched up on some of the code for the project.

Limitations

We did discover limitations to our project. We may have been able to overcome some of these. But with our limited time and our novice skill levels, we were satisfied with our basic, working and completed project.

- We found out that the maximum upload file size could be no more than 128MB.
- In order to delete or upload an item you need a password to secure the cloud storage a bit. But right now if the password is entered once, the browser holds onto it. Meaning the user does not have to enter the password again to say delete a file or upload again.
- The settings for the website cannot be configured on the website. They can only be configured on the VM instance by altering the php code or in the admin console on the Google Cloud Platform interface.
- The application can only sort files by their extensions which we hard coded into the code.
- You can only upload one file at a time.

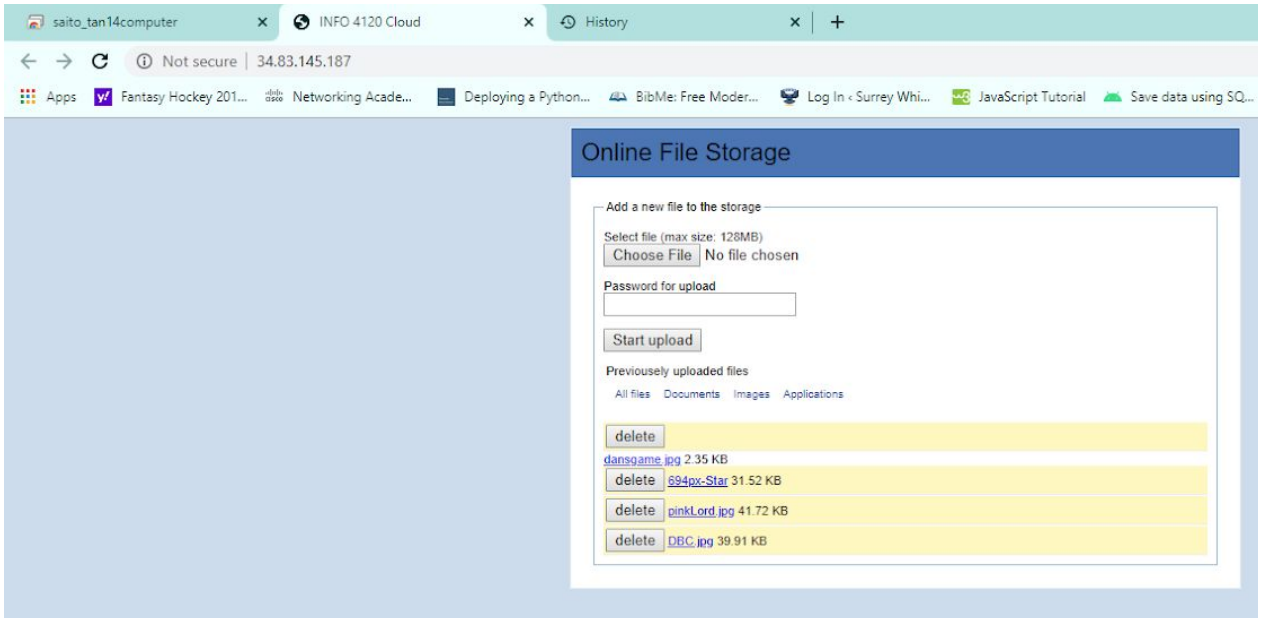
- There is no archiving for the system. So files will never be deleted from the storage.

Assumptions

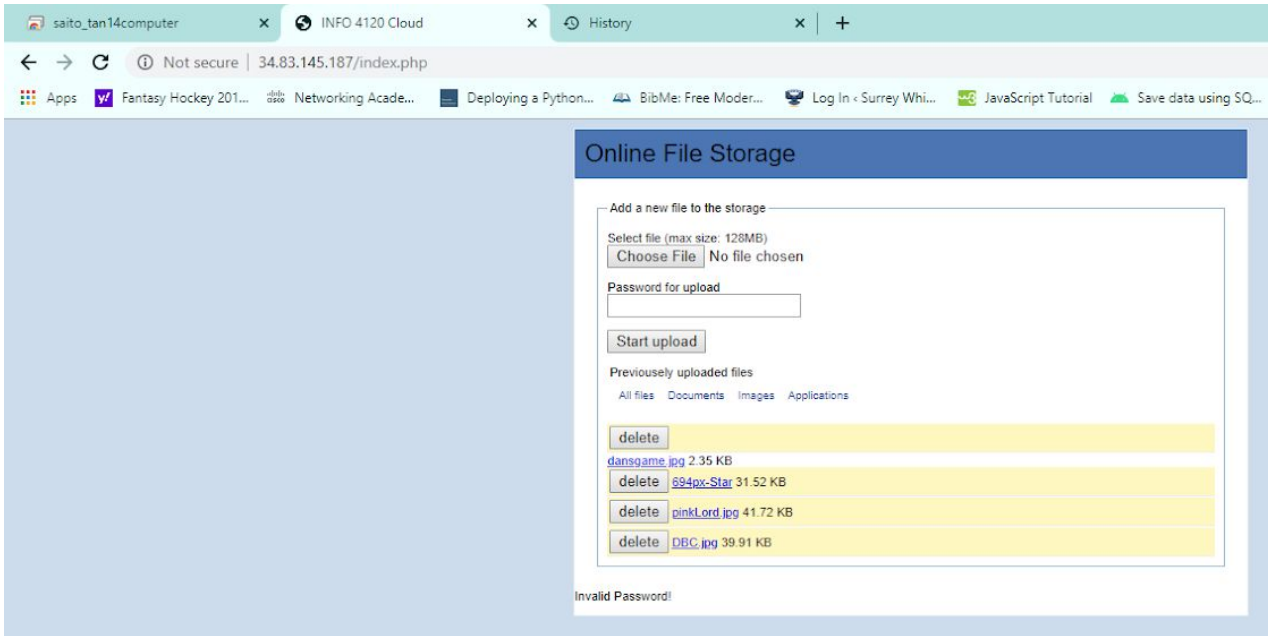
Our project is basic and can only execute simple functions. So we do have a few assumptions about the user and how they will use it.

- We assume that the user's who have access to our cloud storage were enabled by the cloud project admin. Right now there is no functionality to make changes to the storage's settings from the web interface. All changes need to be made in the Google Cloud Platform project console or the php files.
- We assume that the only people who want to access this drive have the IP address to the website. There is no name associated to our website so it is not something that you could find from a simple Google search.
- We assume that the user will read all of the information on the website and follow its wording.

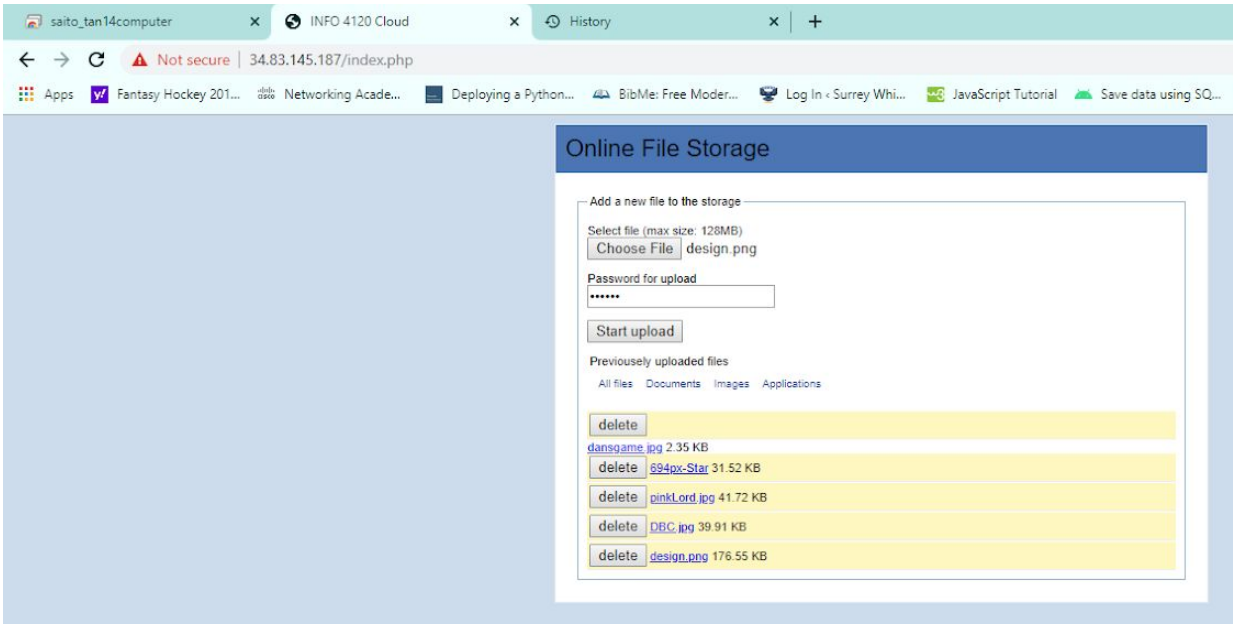
Screenshots of application



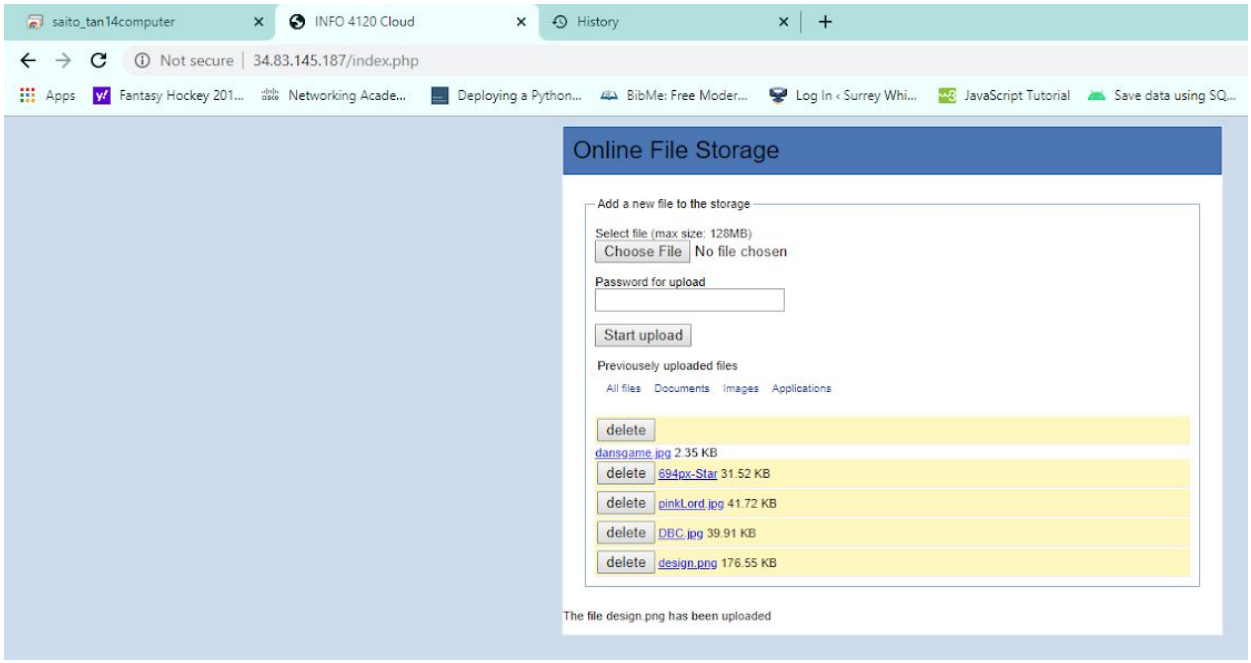
j. How the application looks to the end user



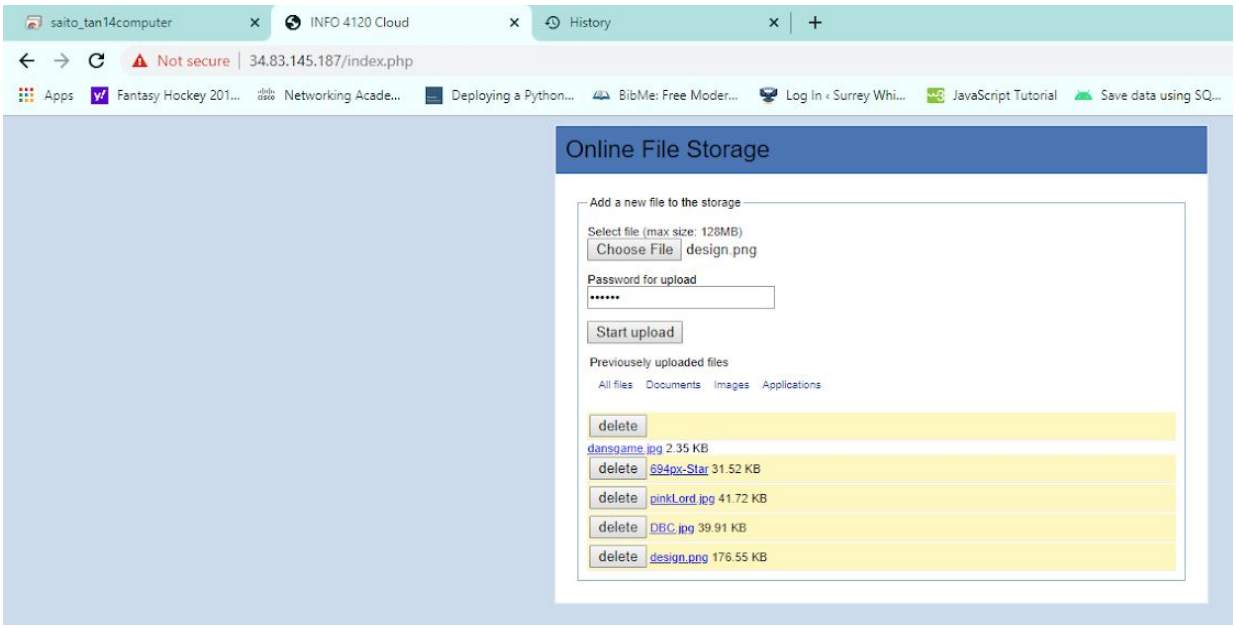
k. A file is selected in this process



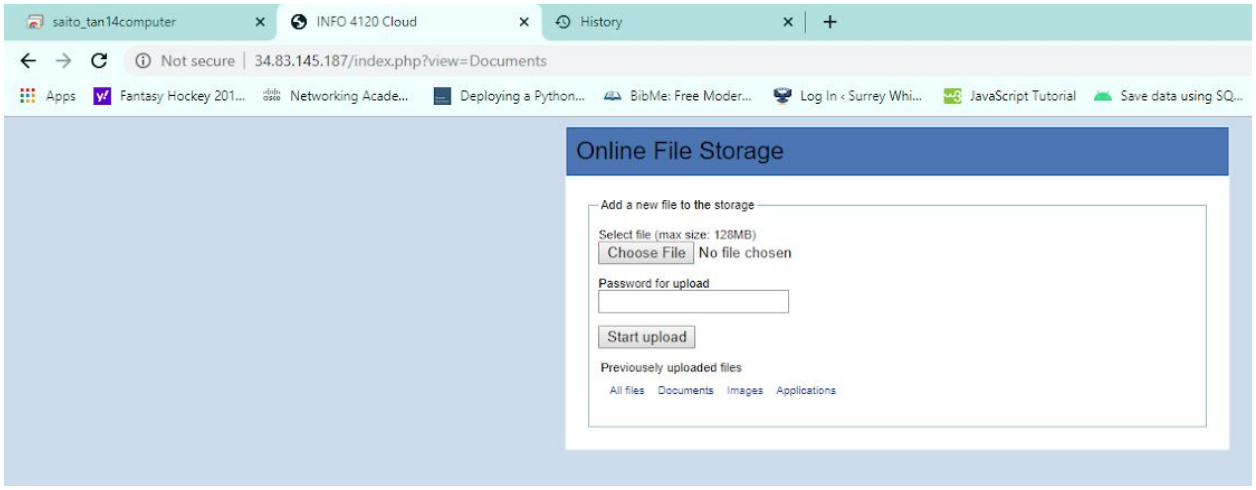
l. Password is entered



The file is uploaded

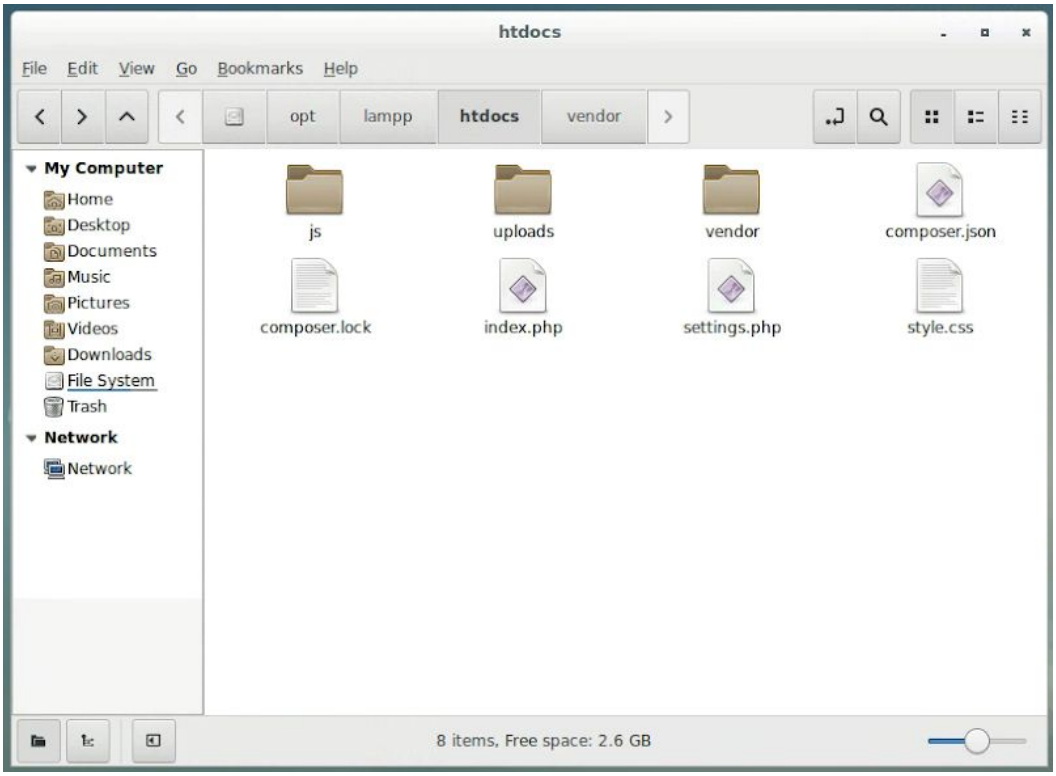


m. To erase a file, enter the password again and then you can see the option to delete the file



n. The files you selected will be deleted then

Screenshots of code:



Index.php

```
<?php

//Load the settings

require_once("settings.php");


require 'vendor/autoload.php';

use Google\Cloud\Storage\StorageClient;
```

```
$message = "";

//Has the user uploaded something?

$storage = new StorageClient();

$bucket = $storage->bucket(Settings::$bucketName);


if(isset( $_GET['view'])){

        $viewMode = $_GET['view'];

} else {

        $viewMode = "all";

}

if(isset($_POST['action'])){

        if($_POST['action'] == 'delete'){

                if(isset($_POST['deleteFile'])){

                        deleteFile($_POST['deleteFile'], $bucket);

                }

        } else if($_POST['action'] == 'Start upload'){

                if(isset($_FILES['file']))

{


        $target_path = Settings::$uploadFolder;
```

```
    $target_path = $target_path . time() . '_' . basename( $_FILES['file']['name']);
    //Check the password to verify legal upload
    if($_POST['password'] != Settings::$password)
    {
    $message = "Invalid Password!";
    }
    else
    {
            //Try to move the uploaded file into the designated folder
  if(move_uploaded_file($_FILES['file']['tmp_name'], $target_path)) {
    $message = "The file ".  basename( $_FILES['file']['name']).
    " has been uploaded";
                    $file = fopen($target_path, 'r');
                    $object = $bucket->upload($file, ['name']);
                    $object->update(['acl' => []], ['predefinedAcl' => 'PUBLICREAD']);
                    unlink($target_path);

  } else{
    $message = "There was an error uploading the file, please try again!";
  }

}
```

```
}
        }
}

//echo $viewMode;

if(strlen($message) > 0)
{
    $message = '<p class="error">' . $message . '</p>';
}
/** LIST UPLOADED FILES **/

$files_in_cloud = "";
        foreach ($bucket->objects() as $object) {
                //printf('Object: %s' . PHP_EOL, $object->name());
                $filename = $object->name();
                $filepath = Settings::$folderpath;
                $parts = explode("_", $object->name());
                $info = $object->info();
                $size = formatBytes($info['size']);
```

```
                $origName = $parts[1];
                $filenameparts = explode('.', $filename);
                $fileextention = end($filenameparts);
                $filetype = getFileType($fileextention);
                if($viewMode == "all" || $filetype == $viewMode){
                        $files_in_cloud .= "
                                <li class=\"$filetype\">
                                        <input type=\"hidden\" name=\"deleteFile\"
value=\"$filename\">
                                        <input type=\"submit\" name=\"action\" value=\"delete\">
                                </form>
                                <a href=\"$filepath$filename\">$origName</a>
                                $size
                                </li>
                        \n";
                }
        }


if(strlen($files_in_cloud) == 0)
{
```

```
    $$files_in_cloud = "<li><em>No files found</em></li>";
}

function getFileType($extension)
{
    $images = array('jpg', 'gif', 'png', 'bmp');
    $docs   = array('txt', 'rtf', 'doc', 'docx');
    $apps   = array('zip', 'rar', 'exe');

    if(in_array($extension, $images)) return "Images";
    if(in_array($extension, $docs)) return "Documents";
    if(in_array($extension, $apps)) return "Applications";
    return "";
}
function formatBytes($bytes, $precision = 2) {
    $units = array('B', 'KB', 'MB', 'GB', 'TB');

    $bytes = max($bytes, 0);
    $pow = floor(($bytes ? log($bytes) : 0) / log(1024));
    $pow = min($pow, count($units) - 1);

    $bytes /= pow(1024, $pow);
```

```
    return round($bytes, $precision) . ' ' . $units[$pow];
}

function deleteFile($filename, $bucket){
        $object = $bucket->object($filename);
        if(isset($_POST['password'])){
                if($_POST['password'] != Settings::$password){
                $message = "Invalid Password!";
                } else {
                        if($object->exists()){
                                $object->delete();
                        }
                }
        }

}

?>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<link rel="stylesheet" type="text/css" href="style.css">

<style>



</style>



<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />

<title>INFO 4120 Cloud</title>

</head>



<body>

<div id="container">

   <h1>Online File Storage</h1>



   <fieldset>



      <legend>Add a new file to the storage</legend>

      <form method="post" action="index.php" enctype="multipart/form-data">
```

```
    <input type="hidden" name="MAX_FILE_SIZE" value="12800000000" />
    <p><label for="name">Select file (max size: 128MB)</label><br />
    <input type="file" name="file" /></p>
    <p><label for="password">Password for upload</label><br />
    <input type="password" name="password" /></p>
    <p><input type="submit" name="action" value="Start upload" /></p>

<legend>Previousely uploaded files</legend>
<ul id="menu">
    <li><a href="?view=all">All files</a></li>
    <li><a href="?view=Documents">Documents</a></li>
    <li><a href="?view=Images">Images</a></li>
    <li><a href="?view=Applications">Applications</a></li>
</ul>

<ul id="files">
                <?php
                        //echo $uploaded_files;
                        echo $files_in_cloud;
                ?>
</ul>
        </form>
```

```
</fieldset>
        <?php echo $message ?>
</div>
<script src="js/filestorage.js" />
</body>
</html>
```

Settings.php

```
<?php



/**
 * Class Settings holds the upload settings
 *
 */
class Settings
{
    static $password = "123456";
    static $uploadFolder = "uploads/";
        static $bucketName = "thomas_federau_1";
        //static $folderpath = "https://storage.cloud.google.com/thomas_federau_1/";
        static $folderpath = "https://storage.googleapis.com/thomas_federau_1/";
```

```
        //static $uploadFolder =
"console.cloud.google.com/storage/browser/thomas_federau_1/"

}

?>
```

Style.css

```
@CHARSET "ISO-8859-1";


body

{

   background-color: #cddcec;

   font-family: "Arial";

   font-size: 11px;

}


div#container

{
```

```
    width: 600px;
    margin: 0px auto;
    border: 1px solid #e6eef6;
    background-color: #ffffff;
}

div#container h1
{
    background-color: #4b75b3;
    margin: 0px;
    padding: 8px;
    font-family: "Arial";
    font-weight: normal;
    border: 1px solid #3564a9;
}

div#container fieldset
{
    margin: 20px;
    border: 1px solid #98b9d0;
}
```

```
ul#menu
{
    list-style-type: none;
    margin: 4px;
    padding: 0px;
}

ul#menu li
{
    float: left;
    margin: 4px;
}

ul#menu li.active
{
    background-color: #98b9d0;
    border-left: 1px solid #3564a9;
    border-top: 1px solid #3564a9;
    border-bottom: 1px solid #e6eef6;
    border-right: 1px solid #e6eef6;
}
```

```
ul#menu li a
{
    text-decoration: none;
    font-size: 10px;
    padding: 2px;
    color: #3564a9;
}

ul#files
{
    list-style-type: none;
    margin: 40px 0px 0px 0px;
    padding: 0px;
}

ul#files li
{
    background-color: #fff7c0;
    border-bottom: 1px solid #efefef;
    padding: 2px;
    margin-bottom: 1px;
```

```
}
```

References

https://cloud.google.com/php/getting-started/using-cloud-storage
https://cloud.google.com/storage/docs/downloading-objects#storage-download-object-code-sample
https://cloud.google.com/storage/docs/uploading-objects#storage-upload-object-code-sample
https://cloud.google.com/storage/docs/quickstart-console
https://cloud.google.com/storage/
https://www.google.com/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjJ_ubDxPnlAhUZIjQIHUSIAxkQjRx6BAgBEAQ&url=%2Furl%3Fsa%3Di%26source%3Dimages%26cd%3D%26ved%3D%26url%3Dhttps%253A%252F%252Fcloud.google.com%252Fsolutions%252Fweb-serving-overview%26psig%3DAOvVaw1w9Q7rDIyYHmp2pHdTUCBJ%26ust%3D1574364914321512&psig=AOvVaw1w9Q7rDIyYHmp2pHdTUCBJ&ust=1574364914321512