**TASK 2 :CERTIFICATE REGISTRATION PROJECT IN GCP (Google Cloud Platform):**

Team Members :

Saadana S(8081980)

Mythreyi Ravi(8082040)

Ishwarya Selvam(8079896)

**Task 1: Sign in to the Google Cloud Platform (GCP) Console**

**Task 2: Deploy a web server VM instance**

1. In the GCP Console, on the **Navigation menu** (), click **Compute Engine** > **VM instances**.
2. Click **Create**.On the **Create an Instance** page, for **Name**, type gcp-task1-instance1
3. For **Region** and **Zone**, select the region and zone us-central1-a
4. For **Machine type**, accept the default.
5. For **Boot disk**, if the **Image** shown is not **Debian GNU/Linux 9 (stretch)**, click **Change** and select **Debian GNU/Linux 9 (stretch)**.
6. Leave the defaults for **Identity and API access** unmodified.
7. For **Firewall**, click **Allow HTTP traffic**.
8. Click **Management, security, disks, networking, sole tenancy** to open that section of the dialog.
9. Enter the following script as the value for **Startup script**:

*apt-get update*

*apt-get install apache2 php php-mysql -y*

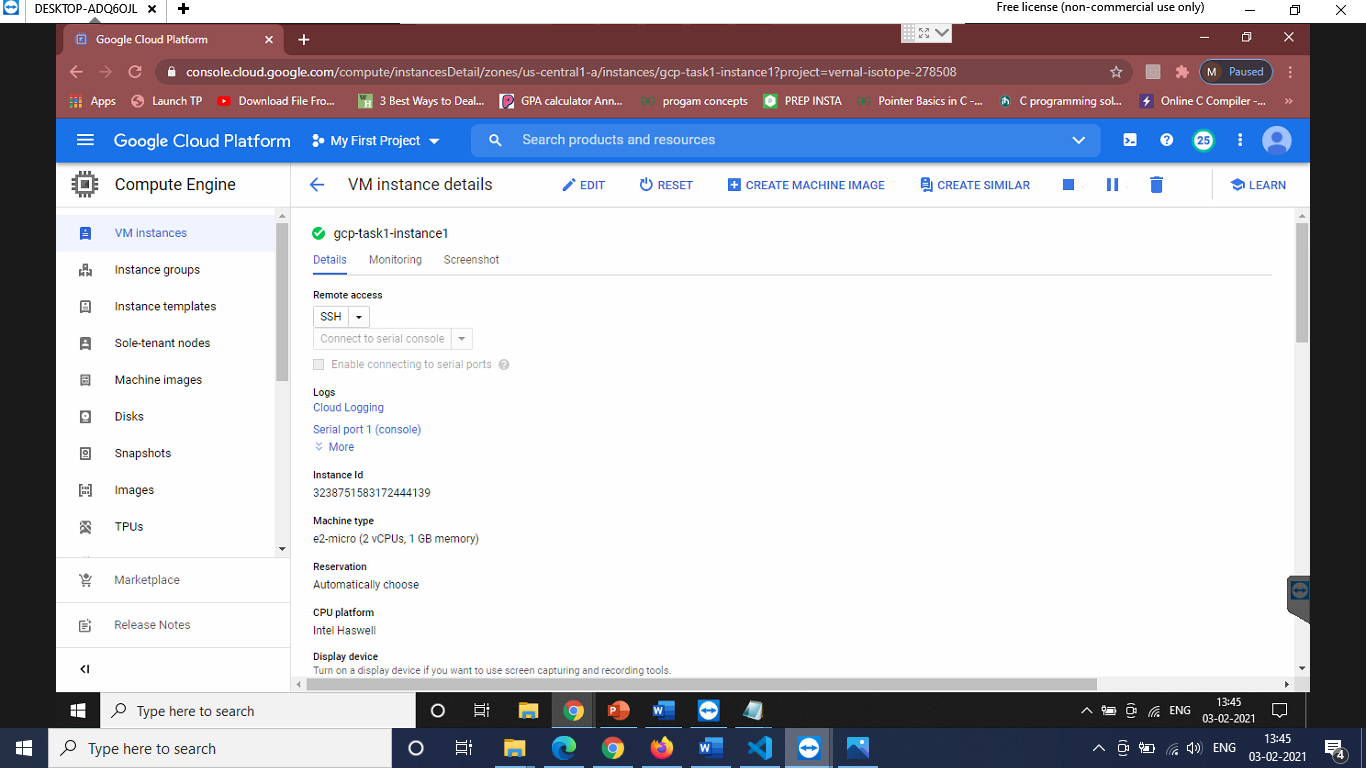
*service apache2 restart*

Be sure to supply that script as the value of the **Startup script** field. If you accidentally put it into another field, it won't be executed when the VM instance starts.

1. Leave the remaining settings as their defaults, and click **Create**.

Instance can take about two minutes to launch and be fully available for use.

1. On the **VM instances** page, copy the gcp-task1-instance1 VM instance's internal and external IP addresses to a text editor for use later.



**Task 3: Create a Cloud Storage bucket using the gsutil command line**

All Cloud Storage bucket names must be globally unique. To ensure that your bucket name is unique, these instructions will guide you to give your bucket the same name as your Cloud Platform project ID, which is also globally unique.

Cloud Storage buckets can be associated with either a region or a multi-region location: **US**, **EU**, or **ASIA**. In this activity, you associate your bucket with the multi-region closest to the region and zone that Qwiklabs or your instructor assigned you to.

1. On the **Google Cloud Platform** menu, click **Activate Cloud Shell** . If a dialog box appears, click **Start Cloud Shell**.
2. For convenience, enter your chosen location into an environment variable called LOCATION. Enter one of these commands:

export LOCATION=US

Or

export LOCATION=EU

Or

export LOCATION=ASIA

1. In Cloud Shell, the DEVSHELL\_PROJECT\_ID environment variable contains your project ID. Enter this command to make a bucket named after your project ID:

*gsutil mb -l $LOCATION gs://$DEVSHELL\_PROJECT\_ID*

1. Retrieve a banner image from a publicly accessible Cloud Storage location:

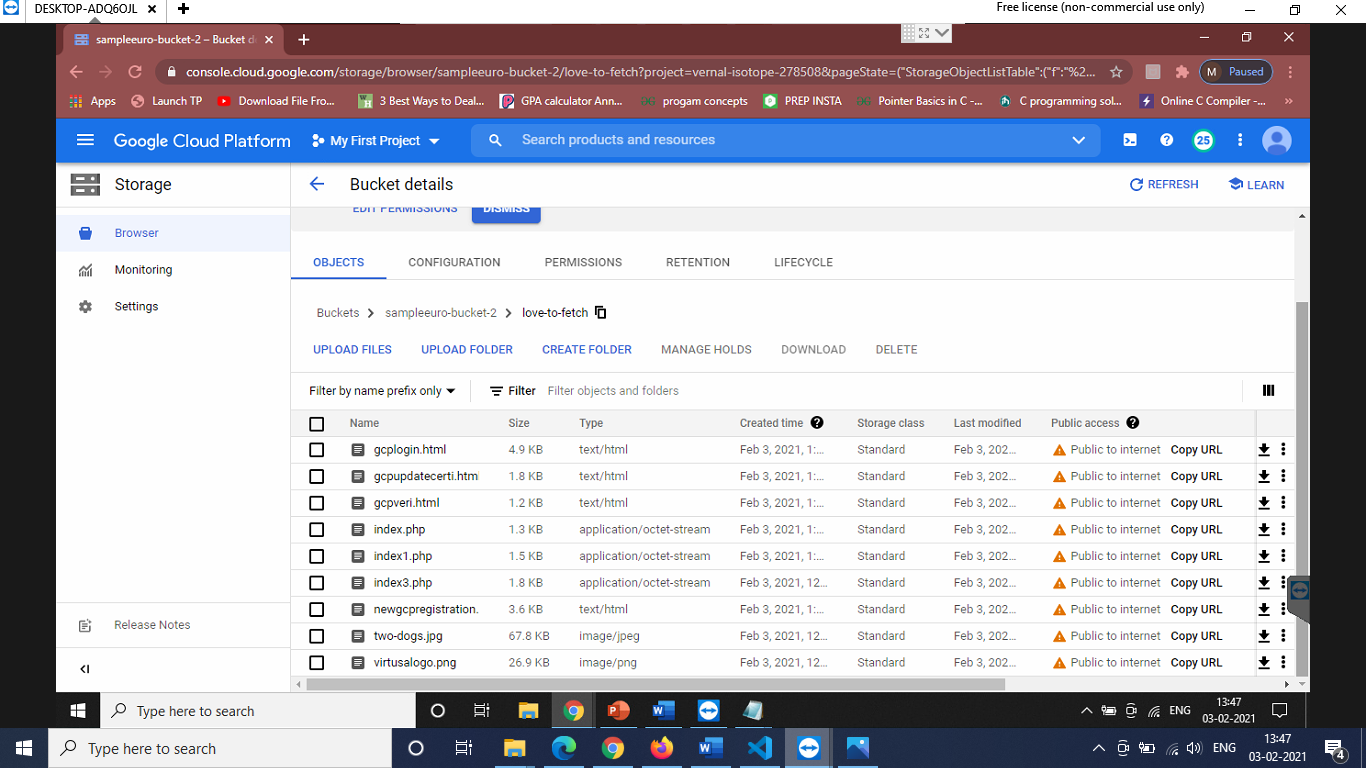
*gsutil cp gs://gcp-task1-bucket1/virtusalogo.png virtusalogo.png*

Copy the banner image to your newly created Cloud Storage bucket:

*gsutil cp virtusalogo.png gs://$DEVSHELL\_PROJECT\_ID/ virtusalogo.png*

Modify the Access Control List of the object you just created so that it is readable by everyone:

*gsutil acl ch -u allUsers:R gs://$DEVSHELL\_PROJECT\_ID/ virtusalogo.png*



**Task 4: Create the Cloud SQL instance**

1. In the GCP Console, on the **Navigation menu** (), click **Storage** > **SQL**.
2. Click **Create instance**.
3. For **Choose a database engine**, select **MySQL**.
4. For **Instance ID,** type **gcp-task1-sql1**, and for **Root password** type a password of your choice.

Choose a password that you remember. There's no need to obscure the password because you'll use mechanisms to connect that aren't open access to everyone.

1. Set the region and zone us-central1-a
2. This is the same region and zone into which you launched the gcp-task1-instance1 instance. The best performance is achieved by placing the client and the database close to each other.
3. Click **Create**.

Wait for the instance to finish deploying. It will take a few minutes.

1. Click on the name of the instance, **regitstrationdb**, to open its details page.
2. From the SQL instances details page, copy the **Public IP address** for your SQL instance to a text editor for use later in this lab.
3. Click the **Users** tab, and then click **Create user account**.
4. For **User name**, type root
5. For **Password**, type a password of your choice. Make a note of it.
6. Click **Create** to create the user account in the database.

Wait for the user to be created.

1. Click the **Connections** tab, and then click **Add network**.

If you are offered the choice between a **Private IP** connection and a **Public IP** connection, choose **Public IP** for purposes of this lab. The **Private IP** feature is in beta at the time this lab was written.

The **Add network** button may be grayed out if the user account creation is not yet complete.

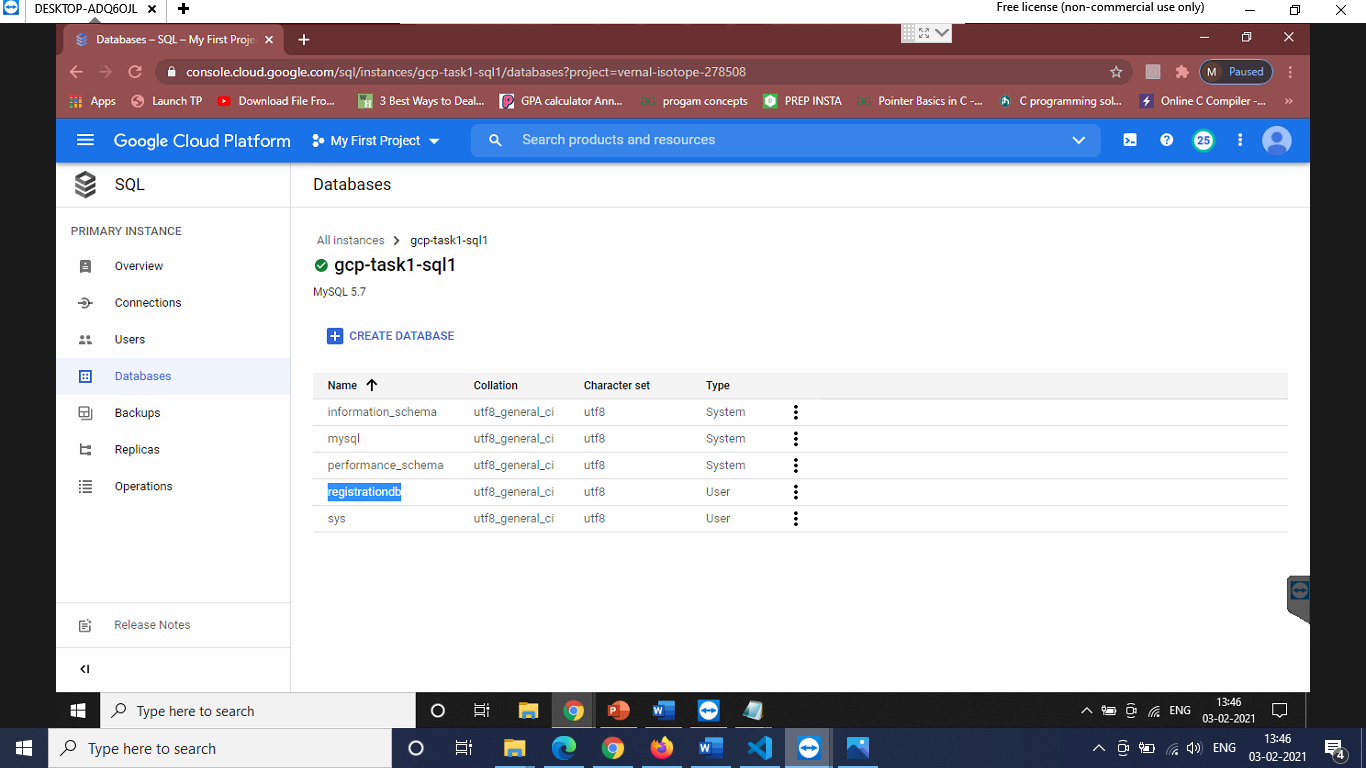
1. For **Name**, type **gcp-task1-network1**
2. For **Network**, type the external IP address of your gcp-task1-instance1**t** VM instance, followed by /32

The result will look like this:

34.70.129.72/32

Be sure to use the external IP address of your VM instance followed by /32. Do not use the VM instance's internal IP address. Do not use the sample IP address shown here.

1. Click **Done** to finish defining the authorized network.
2. Click **Save** to save the configuration change.



**Task 5: Configure an application in a Compute Engine instance to use Cloud SQL**

1. On the **Navigation menu** (), click **Compute Engine** > **VM instances**.
2. In the VM instances list, click **SSH** in the row for your VM instance **gcp-task1-instance1**
3. In your ssh session on gcp-task1-instance1, change your working directory to the document root of the web server:

cd /var/www/html

1. Use the **nano** text editor to edit a file called **gcplogin.html**:

sudo nano gcplogin.html

1. Paste the content below into the file:

|  |  |
| --- | --- |
|  | **<!DOCTYPE html>** |
|  | **<html>** |
|  | **<head>** |
|  | **<style>** |
|  | **.img-container1 {** |
|  | **text-align: center;** |
|  | **}** |
|  | **</style>** |
|  |  |
|  | **<script>** |
|  | **function validateForm()** |
|  | **{** |
|  | **var alphanumberic = /^[a-z0-9]+$/i;** |
|  | **var mailformat = /^[a-zA-Z0-9.!#$%&'\*+/=?^\_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)\*$/;** |
|  | **var passw = /^(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z]).{6,20}$/;** |
|  | **var x=document.forms["f1"]["t1"].value;** |
|  | **var y=document.forms["f1"]["t2"].value;** |
|  | **var z=document.forms["f1"]["t3"].value;** |
|  |  |
|  |  |
|  | **if(!x.match(alphanumberic))** |
|  | **{** |
|  | **alert("Please fill out the username in alphanumeric");** |
|  | **return false;** |
|  | **}** |
|  |  |
|  | **if(y.indexOf("@virtusa.com", y.length - "@virtusa.com".length) == -1)** |
|  | **{** |
|  | **alert("Please fill a valid email address");** |
|  | **return false;** |
|  | **}** |
|  | **if (!z.match(passw))** |
|  | **{** |
|  | **alert("Please enter the password in the given format");** |
|  | **return false;** |
|  | **}** |
|  | **Else** |
|  | **{** |
|  | **window.open("gcpupdatecerti.html");** |
|  | **return true;** |
|  | **}** |
|  |  |
|  |  |
|  |  |
|  |  |
|  | **var myInput = document.getElementById("psw");** |
|  | **var letter = document.getElementById("letter");** |
|  | **var capital = document.getElementById("capital");** |
|  | **var number = document.getElementById("number");** |
|  | **var length = document.getElementById("length");** |
|  |  |
|  | **// When the user clicks on the password field, show the message box** |
|  | **myInput.onfocus = function() {** |
|  | **document.getElementById("message").style.display = "block";** |
|  | **}** |
|  |  |
|  | **// When the user clicks outside of the password field, hide the message box** |
|  | **myInput.onblur = function() {** |
|  | **document.getElementById("message").style.display = "none";** |
|  | **}** |
|  |  |
|  | **// When the user starts to type something inside the password field** |
|  | **myInput.onkeyup = function() {** |
|  | **// Validate lowercase letters** |
|  | **var lowerCaseLetters = /[a-z]/g;** |
|  | **if(myInput.value.match(lowerCaseLetters)) {** |
|  | **letter.classList.remove("invalid");** |
|  | **letter.classList.add("valid");** |
|  | **} else {** |
|  | **letter.classList.remove("valid");** |
|  | **letter.classList.add("invalid");** |
|  | **}** |
|  |  |
|  | **// Validate capital letters** |
|  | **var upperCaseLetters = /[A-Z]/g;** |
|  | **if(myInput.value.match(upperCaseLetters)) {** |
|  | **capital.classList.remove("invalid");** |
|  | **capital.classList.add("valid");** |
|  | **} else {** |
|  | **capital.classList.remove("valid");** |
|  | **capital.classList.add("invalid");** |
|  | **}** |
|  |  |
|  | **// Validate numbers** |
|  | **var numbers = /[0-9]/g;** |
|  | **if(myInput.value.match(numbers)) {** |
|  | **number.classList.remove("invalid");** |
|  | **number.classList.add("valid");** |
|  | **} else {** |
|  | **number.classList.remove("valid");** |
|  | **number.classList.add("invalid");** |
|  | **}** |
|  |  |
|  | **// Validate length** |
|  | **if(myInput.value.length >= 8) {** |
|  | **length.classList.remove("invalid");** |
|  | **length.classList.add("valid");** |
|  | **} else {** |
|  | **length.classList.remove("valid");** |
|  | **length.classList.add("invalid");** |
|  | **}** |
|  | **}** |
|  |  |
|  |  |
|  | **}** |
|  | **</script>** |
|  |  |
|  | **<meta name="viewport" content="width=device-width, initial-scale=1">** |
|  | **<!-- Add icon library -->** |
|  | **<link rel="stylesheet" href="**[**https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css**](https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css)**">** |
|  | **<style>** |
|  | **body {font-family: Arial, Helvetica, sans-serif;}** |
|  | **\* {box-sizing: border-box;}** |
|  |  |
|  | **.input-container {** |
|  | **display: -ms-flexbox; /\* IE10 \*/** |
|  | **display: flex;** |
|  | **width: 100%;** |
|  | **margin-bottom: 15px;** |
|  | **}** |
|  |  |
|  | **.icon {** |
|  | **padding: 10px;** |
|  | **background: dodgerblue;** |
|  | **color: white;** |
|  | **min-width: 50px;** |
|  | **text-align: center;** |
|  | **}** |
|  |  |
|  | **.input-field {** |
|  | **width: 100%;** |
|  | **padding: 10px;** |
|  | **outline: none;** |
|  | **}** |
|  |  |
|  | **.input-field:focus {** |
|  | **border: 2px solid dodgerblue;** |
|  | **}** |
|  |  |
|  | **/\* Set a style for the submit button \*/** |
|  | **.btn {** |
|  | **background-color: dodgerblue;** |
|  | **color: white;** |
|  | **padding: 15px 20px;** |
|  | **border: none;** |
|  | **cursor: pointer;** |
|  | **width: 100%;** |
|  | **opacity: 0.9;** |
|  | **}** |
|  |  |
|  | **.btn:hover {** |
|  | **opacity: 1;** |
|  | **}** |
|  |  |
|  |  |
|  |  |
|  | **</style>** |
|  | **</head>** |
|  | **<body bgcolor="pink">** |
|  | **<div class="img-container1"> <!-- Block parent element -->** |
|  | **<img src="**[**https://storage.googleapis.com/gcp-task1-bucket1/virtusalogo.png**](https://storage.googleapis.com/gcp-task1-bucket1/virtusalogo.png)**" alt="virtusa logo" width="300" height="300">** |
|  | **</div>** |
|  |  |
|  |  |
|  | **<form name="f1" action="index.php" style="max-width:500px;margin:auto">** |
|  |  |
|  | **<marquee><font color="blue">Welcome to Virtusa</font></marquee>** |
|  | **<div class="input-container">** |
|  | **<i class="fa fa-user icon"></i>** |
|  | **<input class="input-field" type="text" placeholder="Username" name="t1">** |
|  | **</div>** |
|  |  |
|  | **<div class="input-container">** |
|  | **<i class="fa fa-envelope icon"></i>** |
|  | **<input class="input-field" type="text" placeholder="Email" name="t2" title="Must have virtusa domain" required>** |
|  | **</div>** |
|  |  |
|  | **<div class="input-container">** |
|  | **<i class="fa fa-key icon"></i>** |
|  | **<input class="input-field" type="password" placeholder="Password" name="t3" pattern="(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z]).{8,}" title="Must contain at least one number and one uppercase and lowercase letter, and at least 8 or more characters" required>** |
|  | **</div>** |
|  |  |
|  |  |
|  |  |
|  |  |
|  | **<button type="submit" class="btn" onclick="validateForm()">LOGIN</button>** |
|  | **</form>** |
|  |  |
|  |  |
|  |  |
|  | **</body>** |
|  | **</html>** |
|  |  |
|  |  |

1. Press **Ctrl+O**, and then press **Enter** to save your edited file.
2. Press **Ctrl+X** to exit the nano text editor.

*sudo nano index.php*

13.Paste the below php code

**<html>**

**<head><title>VIRTUSA</title></head>**

**<body bgcolor="pink">**

**<div class="img-container1"> <!-- Block parent element -->**

**<img src="https://storage.cloud.google.com/sampleeuro-bucket-2/love-to-fetch/virtusalogo.png" alt="virtusa logo" width="300">**

**</div>**

**<?php**

**$dbserver = "35.193.145.78";**

**$dbuser = "root";**

**$dbpassword = "classic123";**

**$database = "registrationdb";**

**// In a production blog, we would not store the MySQL**

**// password in the document root. Instead, we would store it in a**

**// configuration file elsewhere on the web server VM instance.**

**$conn = new mysqli($dbserver, $dbuser, $dbpassword, $database);**

**if (mysqli\_connect\_error()) {**

**echo ("Database connection failed: " . mysqli\_connect\_error());**

**} else {**

**echo ("Database connection succeeded.");**

**}**

**// Escape user inputs for security**

**$t1 = mysqli\_real\_escape\_string($conn, $\_REQUEST['t1']);**

**$t2 = mysqli\_real\_escape\_string($conn, $\_REQUEST['t2']);**

**$t3 = mysqli\_real\_escape\_string($conn, $\_REQUEST['t3']);**

**// Attempt insert query execution**

**$sql = "INSERT INTO logtable(Username,Email,Password) VALUES ('$t1', '$t2', '$t3')";**

**if(mysqli\_query($conn, $sql)){**

**echo "Records added successfully.";**

**} else{**

**echo "ERROR: Could not able to execute $sql. " . mysqli\_error($conn);**

**}**

1. In the **nano** text editor, replace with the Cloud SQL instance Public IP address that you noted above. Leave the quotation marks around the value in place.
2. In the **nano** text editor, replace DBPASSWORD with the Cloud SQL database password that you defined above. Leave the quotation marks around the value in place.
3. Press **Ctrl+O**, and then press **Enter** to save your edited file.
4. Press **Ctrl+X** to exit the nano text editor.
5. Restart the web server:

*sudo service apache2 restart*

**repeat the steps for gcpupdatecerti.html, index1.php, newgcpregistation.html, index3.php, gcpverifi.html**

*sudo nano* **gcpupdatecerti.html**

13.Paste the below code

|  |  |
| --- | --- |
|  |  |
|  | **<html>** |
|  | **<head>** |
|  | **<style>** |
|  | **.img-container {** |
|  | **text-align: center;** |
|  | **}** |
|  | **.btn {** |
|  | **background-color: dodgerblue;** |
|  | **color: white;** |
|  | **padding: 15px 20px;** |
|  | **border: none;** |
|  | **cursor: pointer;** |
|  | **width: 100%;** |
|  | **opacity: 0.9;** |
|  | **}** |
|  |  |
|  | **.btn:hover {** |
|  | **opacity: 1;** |
|  | **}** |
|  |  |
|  |  |
|  |  |
|  | **</style>** |
|  | **<script>** |
|  | **function update()** |
|  | **{** |
|  | **window.open("https://storage.cloud.google.com/sampleeuro-bucket-2/love-to-fetch/newgcpregistration.html");** |
|  | **}** |
|  | **function view()** |
|  | **{** |
|  | **window.open("https://storage.cloud.google.com/sampleeuro-bucket-2/love-to-fetch/index3.php");** |
|  | **}** |
|  | **function logout()** |
|  | **{** |
|  | **window.open("https://storage.cloud.google.com/sampleeuro-bucket-2/love-to-fetch/gcplogin.html");** |
|  | **}** |
|  |  |
|  | **</script>** |
|  | **</head>** |
|  | **<body bgcolor="pink">** |
|  | **<div class="img-container"> <!-- Block parent element -->** |
|  | **<img src="**[**https://storage.cloud.google.com/sampleeuro-bucket-2/love-to-fetch/virtusalogo.png**](https://storage.cloud.google.com/sampleeuro-bucket-2/love-to-fetch/virtusalogo.png)**" alt="John Doe" width="300" height="300">** |
|  | **</div>** |
|  | **<table class="center" width="300" border="0" align="center" cellspacing="2">** |
|  | **<tr > <td>** |
|  | **<table width="50%" border="0">** |
|  | **<tr>** |
|  | **<td>** |
|  | **<table width="50%" border="0">** |
|  |  |
|  |  |
|  | **<tr>** |
|  |  |
|  | **<td><input type="button" class="btn" name="t1" value="UPDATE CERTIFICATE" style="background-color:blue; color:white; font-size:24px;" onclick="update()"></td>** |
|  | **<td><input type="button" class="btn" name="t2" value="VIEW CERTIFICATE" style="background-color:blue; color:white; font-size:24px;" onclick="view()" ></td>** |
|  | **<td><input type="button" class="btn" name="t3" value="LOGOUT" style="background-color:blue; color:white; font-size:24px;" onclick="logout()" ></td>** |
|  | **</tr>** |
|  | **</table>** |
|  | **<font color="blue"><marquee behavior="alternate" ><h2>"SPARKING INNOVATION,ONE SPRINT AT A TIME"</h2></font></marquee>** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | **</body>** |
|  | **</html>** |
|  |  |

1. Press **Ctrl+O**, and then press **Enter** to save your edited file.
2. Press **Ctrl+X** to exit the nano text editor.

*sudo nano index1.php*

13.Paste the below php code

**<html>**

**<head><title>VIRTUSA</title></head>**

**<body bgcolor="pink">**

**<div class="img-container1"> <!-- Block parent element -->**

**<img src="https://storage.cloud.google.com/sampleeuro-bucket-2/love-to-fetch/virtusalogo.png" alt="virtusa logo" width="300" height="300">**

**</div>**

**<?php**

**$dbserver = "35.193.145.78";**

**$dbuser = "root";**

**$dbpassword = "classic123";**

**$database = "registrationdb";**

**// In a production blog, we would not store the MySQL**

**// password in the document root. Instead, we would store it in a**

**// configuration file elsewhere on the web server VM instance.**

**$conn = new mysqli($dbserver, $dbuser, $dbpassword,$database);**

**if (mysqli\_connect\_error()) {**

**echo ("Database connection failed: " . mysqli\_connect\_error());**

**} else {**

**echo ("Database connection succeeded.");**

**}**

**if(isset($\_POST['submit']))**

**{**

**$t1 = $\_POST['t1'];**

**$t2 = $\_POST['t2'];**

**$s1 = $\_POST['s1'];**

**$t3 = $\_POST['t3'];**

**$t4 = $\_POST['t4'];**

**$t5 = $\_POST['t5'];**

**$t7 = $\_POST['t7'];**

**$t8 = $\_POST['t8'];**

**$t6 = $\_POST['t6'];**

**$sql = "INSERT INTO regitable (Empname,**

**Empid ,**

**CSP ,**

**Certilevel ,**

**Certiname ,**

**Certiid ,**

**Date ,**

**Expiry ,**

**Validity)**

**VALUES ('$t1','$t2','$s1','$t3','$t4','$t5','$t7','$t8','$t6')";**

**if (mysqli\_query($conn, $sql)) {**

**echo "New record has been added successfully !";**

**} else {**

**echo "Error: " . $sql . ":-" . mysqli\_error($conn);**

**}**

**mysqli\_close($conn);**

**}**

**?>**

**</body></html>**

1. Press **Ctrl+O**, and then press **Enter** to save your edited file.
2. Press **Ctrl+X** to exit the nano text editor.

*sudo nano* **newgcpregistation.html**

13.Paste the below code

|  |  |
| --- | --- |
|  |  |
|  | **<html>** |
|  | **<head>** |
|  | **<style>** |
|  | **.img-container1 {** |
|  | **text-align: center;** |
|  | **}** |
|  | **</style>** |
|  |  |
|  | **<script type="text/javascript">** |
|  | **function validateForm() {** |
|  |  |
|  |  |
|  |  |
|  |  |
|  | **var letters = /^[a-zA-Z ]\*$/;** |
|  | **var numbers = /^[0-9]+$/;** |
|  | **var alphanumberic = /^[a-z0-9]+$/i** |
|  | **var x = document.forms["f1"]["t1"].value;** |
|  | **var y=document.forms["f1"]["t2"].value;** |
|  | **var z=document.forms["f1"]["t3"].value;** |
|  | **var a=document.forms["f1"]["t4"].value;** |
|  | **var b=document.forms["f1"]["t5"].value;** |
|  | **var c=document.forms["f1"]["t6"].value;** |
|  |  |
|  |  |
|  | **if(!x.match(letters))** |
|  | **{** |
|  | **alert("Please input alphabet characters only");** |
|  | **return false;** |
|  | **}** |
|  |  |
|  | **else if(!y.match(numbers))** |
|  | **{** |
|  | **alert("Emp Id must be filled out");** |
|  | **return false;** |
|  | **}** |
|  |  |
|  | **else if(!z.match(letters))** |
|  | **{** |
|  | **alert("Please fill out the certification level in words");** |
|  | **return false;** |
|  | **}** |
|  |  |
|  | **else if(!a.match(letters))** |
|  | **{** |
|  | **alert("Please fill out the certification name");** |
|  | **return false;** |
|  | **}** |
|  |  |
|  | **else if(!b.match(alphanumberic))** |
|  | **{** |
|  | **alert("Please fill out the certification id");** |
|  | **return false;** |
|  | **}** |
|  |  |
|  |  |
|  |  |
|  | **else if(!c.match(numbers))** |
|  | **{** |
|  | **alert("Please enter the validity of the certificate");** |
|  | **return false;** |
|  | **}** |
|  |  |
|  | **else** |
|  | **{** |
|  | **alert("Thank You, registration is completed:) and you will receive a mail shortely");** |
|  | **window.open("https://storage.cloud.google.com/sampleeuro-bucket-2/love-to-fetch/gcpveri.html");** |
|  | **return true;** |
|  | **}** |
|  |  |
|  | **}** |
|  |  |
|  | **</script></head>** |
|  | **<body bgcolor="pink">** |
|  | **<div class="img-container1"> <!-- Block parent element -->** |
|  | **<img src="**[**https://storage.cloud.google.com/sampleeuro-bucket-2/love-to-fetch/virtusalogo.png**](https://storage.cloud.google.com/sampleeuro-bucket-2/love-to-fetch/virtusalogo.png)**" alt="virtusa logo" width="125" height="125">** |
|  | **</div>** |
|  |  |
|  | **<table>** |
|  | **<form name="f1" action="https://storage.cloud.google.com/sampleeuro-bucket-2/love-to-fetch/index1.php" method="post" onsubmit="return validateForm(this)">** |
|  |  |
|  | **<tr>** |
|  | **<th align="center"><h2>Welcome </h2></th>** |
|  | **</tr>** |
|  | **<tr><td><h4> Employee Name<font color="red"><sup>\*</sup></font></h4></td><td><input type="text" name="t1" placeholder="Enter in words"></td></tr>** |
|  | **<tr>** |
|  | **<td><h4> Employee Id<font color="red"><sup>\*</sup></font></h4></td><td><input type="text" name="t2" placeholder="Enter the empid"></td></tr>** |
|  |  |
|  |  |
|  | **<tr>** |
|  | **<td><h4>Certification Service Provider<font color="red"><sup>\*</sup></font></h4></td>** |
|  | **<td>** |
|  | **<select name="s1">** |
|  | **<option value="1">GCP</option>** |
|  | **<option value="2">AWS</option>** |
|  | **<option value="3">AZURE</option>** |
|  | **</select>** |
|  | **</td>** |
|  | **<tr>** |
|  | **<td><h4>Certification Level<font color="red"><sup>\*</sup></font></h4></td><td><input type="text" name="t3" placeholder="Enter in words"></td>** |
|  | **</tr>** |
|  | **<tr>** |
|  | **<td><h4>Certification Name<font color="red"><sup>\*</sup></font></h4></td><td><input type="text" name="t4" placeholder="Enter in words"></td>** |
|  | **</tr>** |
|  | **<tr>** |
|  | **<td><h4>Certification Id<font color="red"><sup>\*</sup></font></h4></td><td><input type="text" name="t5" placeholder="Enter in alphanumeric"></td>** |
|  | **</tr>** |
|  | **<tr>** |
|  | **<td><h4>Date of certification<font color="red"><sup>\*</sup></font></h4></td><td><input type="date" id="start" name="t7"** |
|  | **value="2021-07-22"** |
|  | **min="2021-01-01" max="2030-12-31">** |
|  | **</td>** |
|  |  |
|  | **</tr>** |
|  | **<tr>** |
|  | **<td><h4>Expiry Date of Certification<font color="red"><sup>\*</sup></font></h4></td>** |
|  | **<td><input type="date" id="start" name="t8"** |
|  | **value="2021-07-22"** |
|  | **min="2021-01-01" max="2030-12-31">** |
|  | **</td>** |
|  | **</tr>** |
|  | **<tr>** |
|  | **<td><h4>Validity<font color="red"><sup>\*</sup></font></h4></td><td><input type="text" name="t6" placeholder="Enter in numeric"></td>** |
|  | **</tr>** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | **<tr>** |
|  | **<td><input type="SUBMIT" value="SUBMIT" name="submit" onclick="cal();" style="background-color:green; color:pink;"></td>** |
|  | **<td><input type="RESET" value="CANCEL" style="background-color:green; color:pink;"></td></tr>** |
|  | **</table>** |
|  | **</form></body>** |
|  | **</html>** |
|  |  |

1. Press **Ctrl+O**, and then press **Enter** to save your edited file.
2. Press **Ctrl+X** to exit the nano text editor.

*sudo nano index3.php*

13.Paste the below php code

**<html>**

**<head><title>VIRTUSA</title></head>**

**<body bgcolor="pink">**

**<div class="img-container1"> <!-- Block parent element -->**

**<img src="https://storage.cloud.google.com/sampleeuro-bucket-2/love-to-fetch/virtusalogo.png" alt="virtusa logo" width="300" height="300">**

**</div>**

**<form name="frmUser" method="post" action="" align="center">**

**<h3 align="center">Enter employee id </h3>**

**Employee ID:<br>**

**<input type="text" name="empid">**

**<br>**

**<input type="submit" name="submit" value="Submit">**

**</form>**

**<?php**

**$dbserver = "35.193.145.78";**

**$dbuser = "root";**

**$dbpassword = "classic123";**

**$database = "registrationdb";**

**// In a production blog, we would not store the MySQL**

**// password in the document root. Instead, we would store it in a**

**// configuration file elsewhere on the web server VM instance.**

**$conn = new mysqli($dbserver, $dbuser, $dbpassword,$database);**

**if (mysqli\_connect\_error()) {**

**echo ("Database connection failed: " . mysqli\_connect\_error());**

**} else {**

**echo ("Database connection succeeded.");**

**}**

**$result = mysqli\_query($conn,"SELECT \* FROM regitable WHERE Empid='" . $\_POST["empid"] . " ' ");**

**echo "<table border='1'>**

**<tr>**

**<th>Empname</th>**

**<th> Empid</th>**

**<th> CSP</th>**

**<th> Certilevel</th>**

**<th> Certiname</th>**

**<th> Certiid</th>**

**<th> Date</th>**

**<th> Expiry</th>**

**<th> Validity </th>**

**</tr>";**

**while($row = mysqli\_fetch\_array($result))**

**{**

**echo "<tr>";**

**echo "<td>" . $row['Empname'] . "</td>";**

**echo "<td>" . $row['Empid'] . "</td>";**

**echo "<td>" . $row['CSP'] . "</td>";**

**echo "<td>" . $row['Certilevel'] . "</td>";**

**echo "<td>" . $row['Certiname'] . "</td>";**

**echo "<td>" . $row['Certiid'] . "</td>";**

**echo "<td>" . $row['Date'] . "</td>";**

**echo "<td>" . $row['Expiry'] . "</td>";**

**echo "<td>" . $row['Validity'] . "</td>";**

**echo "</tr>";**

**}**

**echo "</table>";**

**mysqli\_close($conn);**

**?>**

1. Press **Ctrl+O**, and then press **Enter** to save your edited file.
2. Press **Ctrl+X** to exit the nano text editor.

*sudo nano* **gcpverifi.html**

13.Paste the below code

|  |  |
| --- | --- |
|  | **<html>** |
|  | **<head>** |
|  | **<style>** |
|  | **.img-container1 {** |
|  | **text-align: center;** |
|  | **}** |
|  | **</style>** |
|  |  |
|  | **<script>** |
|  | **function cal()** |
|  | **{** |
|  | **window.close("https://storage.cloud.google.com/sampleeuro-bucket-2/love-to-fetch/gcpveri.html");** |
|  | **}** |
|  | **function view()** |
|  | **{** |
|  | **window.open("https://storage.cloud.google.com/sampleeuro-bucket-2/love-to-fetch/index3.php");** |
|  | **}** |
|  | **</script>** |
|  | **</head>** |
|  | **<body bgcolor="pink">** |
|  | **<div class="img-container1"> <!-- Block parent element -->** |
|  | **<img src="**[**https://storage.cloud.google.com/sampleeuro-bucket-2/love-to-fetch/virtusalogo.png**](https://storage.cloud.google.com/sampleeuro-bucket-2/love-to-fetch/virtusalogo.png)**" alt="John Doe" width="300" height="300">** |
|  | **</div>** |
|  |  |
|  | **<form name="f1" action="index3.php">** |
|  | **<table>** |
|  | **<marquee><h1><b><font color="blue">Your Certification details and Credentials are registered sucessfully!!!</font></marquee></h1></b>** |
|  | **<h3><font color="blue"> A Mail is sent to your registered mail id.</font></h3>** |
|  | **<font color="white">For further queries contact helpdesk@virtusa.com</font>** |
|  | **<tr><td alin="center"><input type="button" value="EXIT" style="background-color:white; color:blue;"onclick="cal()"></td>** |
|  | **<td alin="center"><input type="button" value="VIEW MY DETAILS" style="background-color:white; color:blue;"onclick="view()"></td>** |
|  | **</tr>** |
|  | **</table>** |
|  | **</form>** |
|  | **</body>** |
|  | **</html>** |

1. In the **nano** text editor, replace with the Cloud SQL instance Public IP address that you noted above. Leave the quotation marks around the value in place.
2. In the **nano** text editor, replace DBPASSWORD with the Cloud SQL database password that you defined above. Leave the quotation marks around the value in place.
3. Press **Ctrl+O**, and then press **Enter** to save your edited file.
4. Press **Ctrl+X** to exit the nano text editor.
5. Restart the web server:

*sudo service apache2 restart*

**Task 6: Creation of logtable and regitable in cloud mysql database**

## Connect to your instance using the mysql client in Cloud Shell

1. In the [Google Cloud Console](https://console.cloud.google.com/), click the Cloud Shell icon (Cloud Shell icon) in the upper right corner.
2. At the Cloud Shell prompt, connect to your Cloud SQL instance:

gcloud sql connect gcp-task1-sql1 --user=root

1. Enter your root password.

The mysql prompt appears.

## Create a database and upload data:

1. Create a SQL database on your Cloud SQL instance:

CREATE DATABASE registrationdb;

1. Insert sample data into the guestbook database:

USE registrationdb;

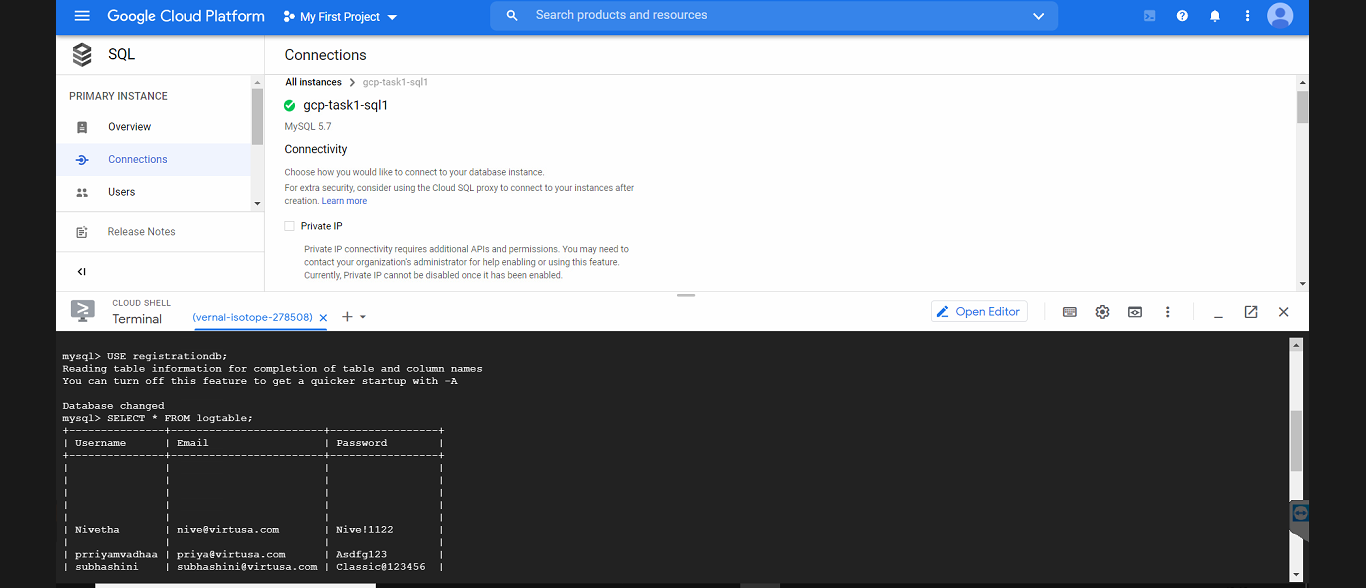
CREATE TABLE logtable(username VARCHAR(255), emailid VARCHAR(255), password VARCHAR(255));

CREATE TABLE regitable (Empname VARCHAR(255), Empid VARCHAR(255)PRIMARY KEY, CSP VARCHAR(255); Certilevel VARCHAR(255), Certiname VARCHAR(255), Certiid VARCHAR(255), Date VARCHAR(255), Expiry VARCHAR(255), Validity VARCHAR(255),

1. Retrieve the data:

SELECT \* FROM logtable;

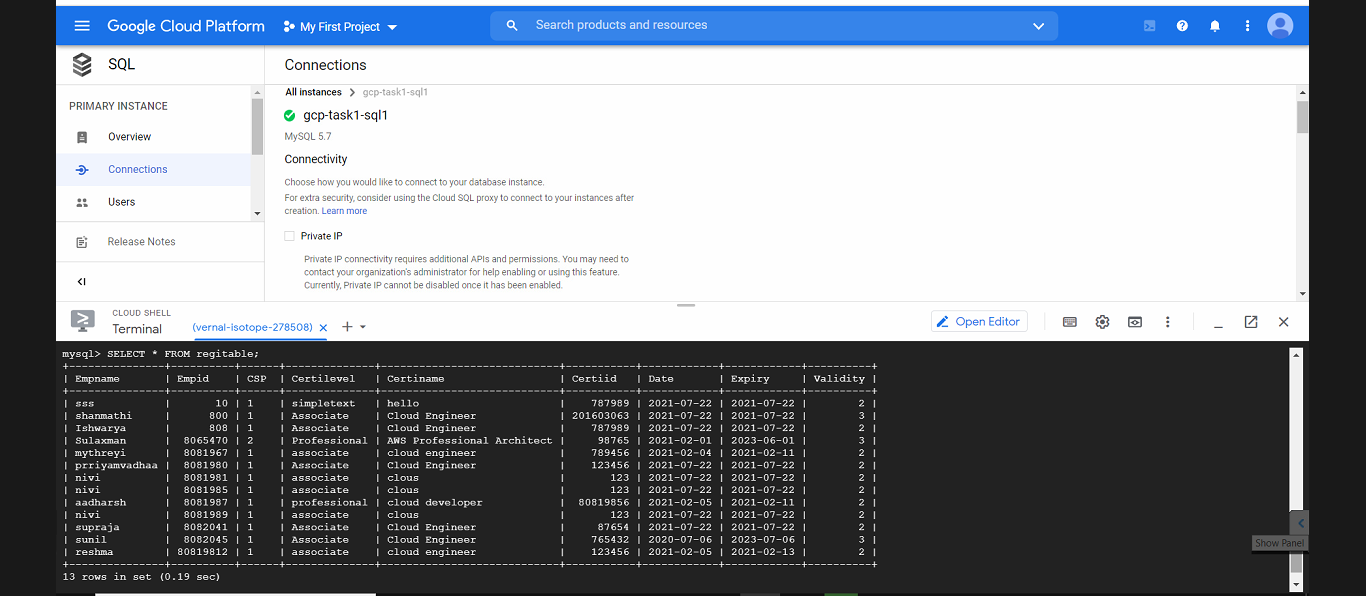
The result is:



1. Retrieve the data:

SELECT \* FROM regitable ;

The result is:

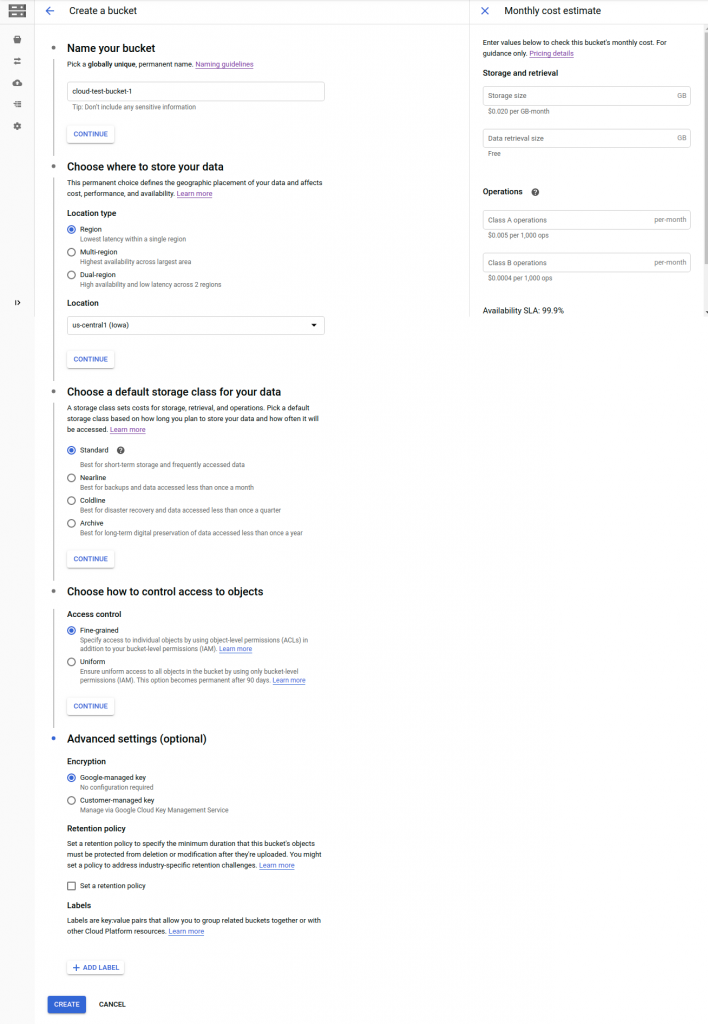


# **Task 7: Upload File To Google Cloud Storage and store the object URL in cloud sql database**

Step 1: Create Bucket  
Step 2: Make Bucket Public  
Step 3: Create Service Account & Download Key  
Step 4: Install Composer & Download Google Cloud Storage Library  
Step 5: Make Code Ready

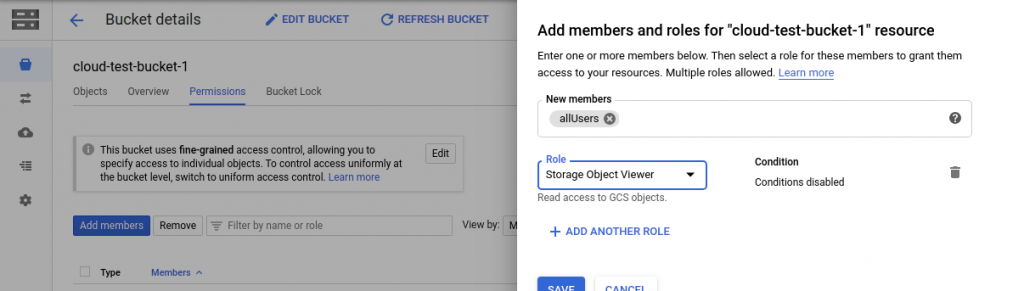
**Step 1: Create Bucket**

First of all we have to create a bucket in which we will upload/store our files. [Buckets](https://cloud.google.com/storage/docs/key-terms#buckets) are the basic containers that hold your data. Everything that you store in Cloud Storage must be contained in a bucket. To know more about bucket click buckets link in this paragraph. Below screenshot will help us to [create bucket](https://cloud.google.com/storage/docs/creating-buckets) from GCP [Console](https://cloud.google.com/cloud-console/) or we can execute “**gsutil mb gs://[BUCKET\_NAME]/”** command in [Cloud Shell](https://cloud.google.com/shell/).

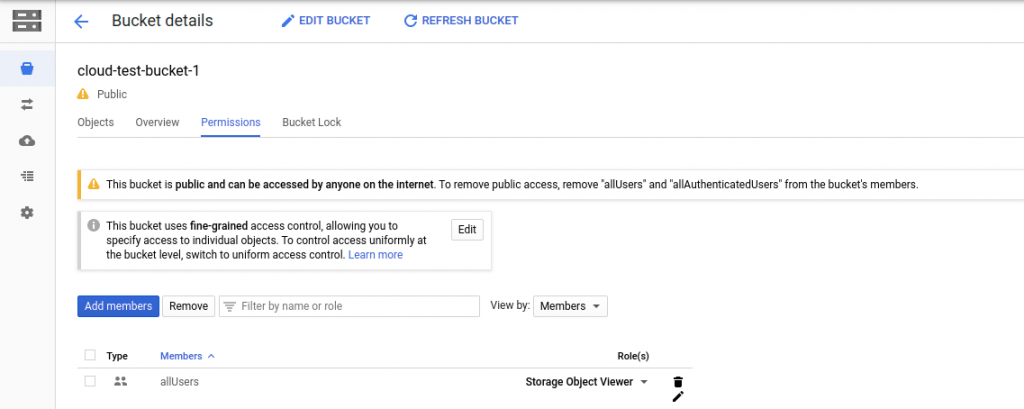


**Step 2: Make Bucket Public**

After creating bucket our next step is to making it public so that we can access it’s object/files at third party hosting or at localhost using uri like **“**[**https://storage.googleapis.com/[BUCKET\_NAME]/[OBJECT\_NAME]”**](https://storage.googleapis.com/%5bBUCKET_NAME%5d/%5bOBJECT_NAME%5d%E2%80%9D). Here also below screenshot will help us to make [bucket public](https://cloud.google.com/storage/docs/access-control/making-data-public) or we can make it public using **“gsutil acl ch -u AllUsers:R gs://[BUCKET\_NAME]/”** command also.



after saving and making it public



**Step 3: Create Service Account & Download Key**

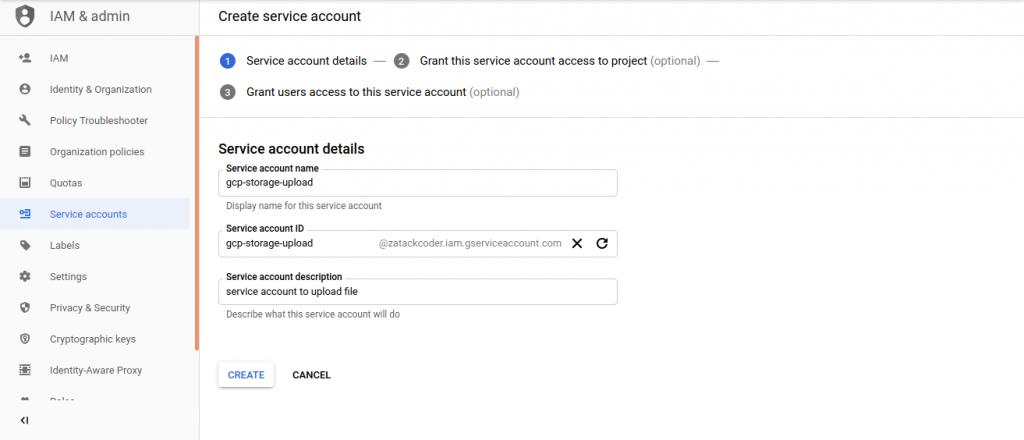
Our next step is to create [service account](https://cloud.google.com/iam/docs/service-accounts) and [downloading private key](https://cloud.google.com/iam/docs/creating-managing-service-account-keys) so that we can use that private key to  connect to Google Cloud Storage. Below screenshot or command will help us to create service account and download private key.

**step 1. creating service account**



|  |  |
| --- | --- |
| 1  2  3 | gcloud iam service-accounts create [SA-NAME] \  &nbsp; &nbsp; --description "[SA-DESCRIPTION]" \  &nbsp; &nbsp; --display-name "[SA-DISPLAY-NAME]" |

OR

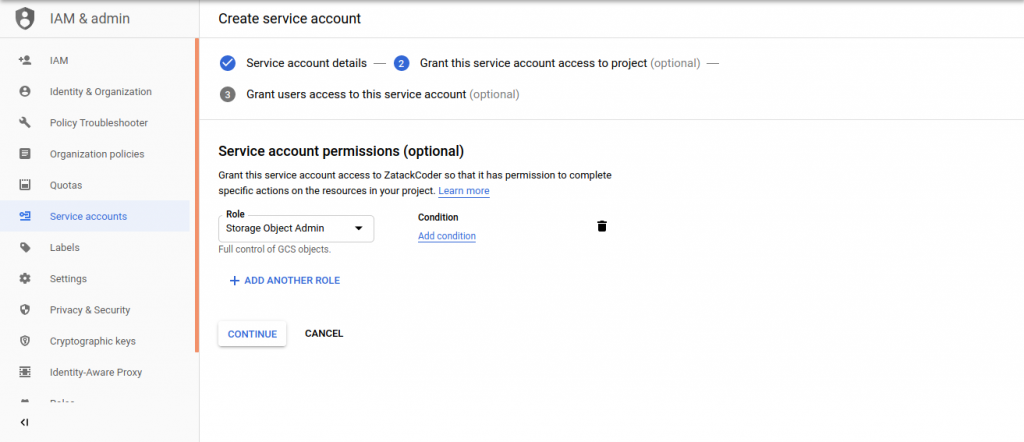


**step 2. granting roles to service account**



|  |  |
| --- | --- |
| 1  2  3 | gcloud projects add-iam-policy-binding my-project-123 \    --member serviceAccount:my-sa-123@my-project-[123.iam.gserviceaccount.com](http://123.iam.gserviceaccount.com/) \    --role roles/editor |

OR

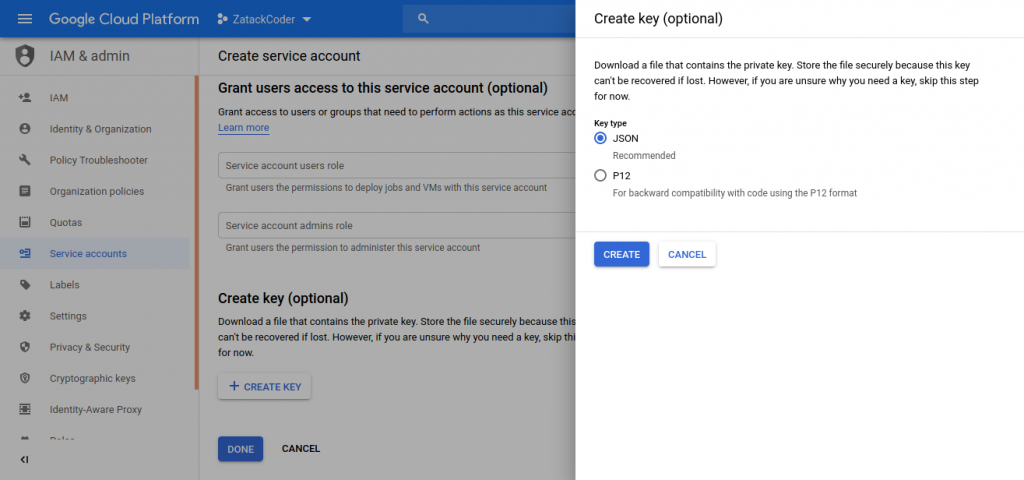


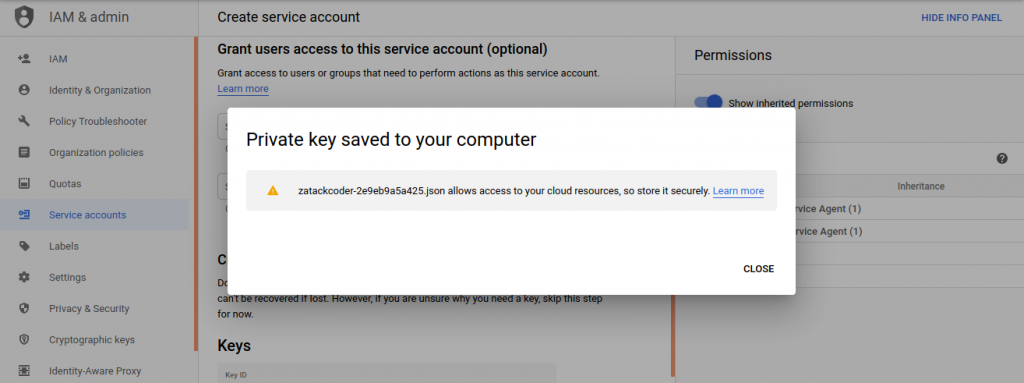
**step 3. creating key**



|  |  |
| --- | --- |
| 1  2 | gcloud iam service-accounts keys create ~/key.json \    --iam-account [SA-NAME]@[PROJECT-ID].iam.gserviceaccount.com |

OR





**Step 4: Install Composer & Download Google Cloud Storage Library**

Now in this step we’ll install [Composer (Dependency Manager for PHP)](https://getcomposer.org/). After installing composer download Google Cloud Storage Library which we are going to use in our code and to download it, we’ll use Terminal (Command Line). In command line we’ll create new directory go inside it or we can go to our project folder if already created then we’ll run below command.



|  |  |
| --- | --- |
| 1 | composer require google/cloud-storage |

Above command will download only [google/cloud-storage](https://googleapis.github.io/google-cloud-php/) library because we want to keep our code small as possible so that we can easily upload our code on third party hosting if needed. Composer will download latest stable version of Google Cloud Storage library after executing above command. After download finishes will have a folder called ‘vendor’ containing library files, we don’t need to change anything inside it.

**Step 5: Make Code Ready**

Now our next step is to code. We will create three php files here **config.php**, **requests.php**, **index.php** and will upload file through ajax. Here config.php will contain private key and necessary functions, requests.php will use to handle ajax request to upload file and index.php for sending ajax request and receiving response.

**Config.php**

|  |
| --- |
| <?php |
|  |  |
|  | // load GCS library |
|  | require\_once 'vendor/fupload/autoload.php'; |
|  |  |
|  | use Google\Cloud\Storage\StorageClient; |
|  |  |
|  | // Please use your own private key (JSON file content) which was downloaded in step 3 and copy it here |
|  | // your private key JSON structure should be similar like dummy value below. |
|  | // WARNING: this is only for QUICK TESTING to verify whether private key is valid (working) or not. |
|  | // NOTE: to create private key JSON file: https://console.cloud.google.com/apis/credentials |
|  | $privateKeyFileContent = '{ |
|  | "type": "service\_account", |
|  | "project\_id": "certificate-project-303705", |
|  | "private\_key\_id": "940375cd30224967d8fd6269ea1b5a30d3e0c124", |
|  | "private\_key": "-----BEGIN PRIVATE KEY-----\nMIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDkN5Tsv6JE5g5f\ngQ02pGZvEsChtl5eCkYu3GrHPPQNmHe2i6Ylksj8nqEuhDHe7fV5IQAl/VqbyYlt\nuec4ZIP5zG+1ltKbo7Phnw8hMWtRs/PAdn2VRWcJ4BemLArqlmP51eiu/A7Ziwo5\n19iM8I0paB1O7IlUYly5pAM3UBFXa1fIqjuq1w0iRpun/aCliGK+/xIipQwYIlDC\nWvb55wzRFvOmzZsGMZNk8tnrCax8o7nDYwImSn0eilrkyANd75hWtVH/kDwpFhMT\npeFal90feICSltOp4LrIkVQJkMyyNz5TzIqQdkdgVOQpDKWsllz/k519QlB5mzfI\nYYqURad5AgMBAAECggEAA1d3gihzsDHUrcwSVRWe7Nsv0CZfFx4sJSlD1H9FXRIZ\nlaYRlJ0DvaQ2fLuBje+6z49exM+jBSM0NJO7aaFaJ90ZGSX4V0BURNZbBrvGJCQA\nCpZJlDcYEUZTXlLohYMWe2zpC2/kKUYOjCaHlJe8n2m3fSV8RMysaij3RZbx7A9e\nnVRQTevxyyNr5tZwbOx/OtGhdvrDLYKqjLNIlBgV9nCPj7SigaOycKvwyQZ7ulwX\nIt2hAOIikhBnfTZ5yHqSa2P5USJXKQQJCf7nYUmcAI6BZfCP95NBCtn/+Nbe9sir\nKSywgB0ZZnQXNfXYb9D4otorPgq957F/4yuKJXpfcQKBgQD0iJqyv7FvsR5CYezc\n+LhIr3og4/eAJnCf45E+6S0Jn58GlMDax/E3FNpx3z3NX4LO31gssjVJIzzflToW\nYucDXJbrmWtdyozuMEjwPd48cER43EgknSrV5rHX97T0+2nwEkh/bZccQT61LBrd\n7K94pstfBriyBbklxBOL7RjhSQKBgQDu6x0BItXtLzU1sND4IdAHCR7kvm8Ujt0g\nDJ31/SRONzsrML4p6UwcUqlXWUFIfy6XNAC7PWyuz1NWIbZoxkBlxMJ5tgiDLAfP\n8lnOnUc0qiHvYkj5G8pv+aM90letQZZmI/G00ljYeZ/cNXOXfNVl2C8Xn/3HSTEc\n8UiKoubEsQKBgQDLt7pZTAXAbSguWXKBLNwqAbaXDBQ3OM7/BREN/ig9KjPLwcVg\n/s1f/Al4cGkGXZmWSs8kfVqTEb427hU0bTTTwiEhKfEedA4wqR06t/AbHdSNc8bO\nafkztjtXFtA0f/djv3eSYXRQX7KkMJg5ZmNQ+nPXOLxJ4ya55b37Bo2BsQKBgFST\nppDgZBqyu4NmNJOiZKIY+HbBc1EzwdO2o7SRCGkn5CpF9wufKvJb0Na7IgEoBLZC\nEoA2HmNDwZycEpbEl8du/+lWJ21ICPv0LxaVVr+t+pVjlbGZxPAez0rzS1ZqAXPn\nBYmdbRY0+AJcaa8W4fRLs2AJoy0JG/nC8IsjX84BAoGAYLsuve3mw84s+ybmV09c\nR9E6fSUoXbbCxn77zO3pB3foJffFBv278c+vvYzmJmpf0FxNVjs7/XFU6RjoxHQc\nZ5dEiOJtQ3Gv89bwV7g0i3FNF+tEJMGT7rP3HVMAoVNJ/bluCIGdg7UvLnQ9JiCi\nUOp9pYxWtnviOwrt5riC2h4=\n-----END PRIVATE KEY-----\n", |
|  | "client\_email": "gcp-storage-upload@certificate-project-303705.iam.gserviceaccount.com", |
|  | "client\_id": "113159614740399784358", |
|  | "auth\_uri": "https://accounts.google.com/o/oauth2/auth", |
|  | "token\_uri": "https://oauth2.googleapis.com/token", |
|  | "auth\_provider\_x509\_cert\_url": "https://www.googleapis.com/oauth2/v1/certs", |
|  | "client\_x509\_cert\_url": "https://www.googleapis.com/robot/v1/metadata/x509/gcp-storage-upload%40certificate-project-303705.iam.gserviceaccount.com" |
|  | }'; |
|  |  |
|  | /\* |
|  | \* NOTE: if the server is a shared hosting by third party company then private key should not be stored as a file, |
|  | \* may be better to encrypt the private key value then store the 'encrypted private key' value as string in database, |
|  | \* so every time before use the private key we can get a user-input (from UI) to get password to decrypt it. |
|  | \*/ |
|  |  |
|  | function uploadFile($bucketName, $fileContent, $cloudPath) { |
|  | $privateKeyFileContent = $GLOBALS['privateKeyFileContent']; |
|  | // connect to Google Cloud Storage using private key as authentication |
|  | try { |
|  | $storage = new StorageClient([ |
|  | 'keyFile' => json\_decode($privateKeyFileContent, true) |
|  | ]); |
|  | } catch (Exception $e) { |
|  | // maybe invalid private key ? |
|  | print $e; |
|  | return false; |
|  | } |
|  |  |
|  | // set which bucket to work in |
|  | $bucket = $storage->bucket('certdocs\_bucket'); |
|  | // upload/replace file |
|  | $storageObject = $bucket->upload( |
|  | $fileContent, |
|  | ['name' => $cloudPath] |
|  | // if $cloudPath is existed then will be overwrite without confirmation |
|  | // NOTE: |
|  | // a. do not put prefix '/', '/' is a separate folder name !! |
|  | // b. private key MUST have 'storage.objects.delete' permission if want to replace file ! |
|  | ); |
|  |  |
|  | // is it succeed ? |
|  | return $storageObject != null; |
|  | } |
|  |  |
|  | function getFileInfo($bucketName, $cloudPath) { |
|  | $privateKeyFileContent = $GLOBALS['privateKeyFileContent']; |
|  | // connect to Google Cloud Storage using private key as authentication |
|  | try { |
|  | $storage = new StorageClient([ |
|  | 'keyFile' => json\_decode($privateKeyFileContent, true) |
|  | ]); |
|  | } catch (Exception $e) { |
|  | // maybe invalid private key ? |
|  | print $e; |
|  | return false; |
|  | } |
|  |  |
|  | // set which bucket to work in |
|  | $bucket = $storage->bucket('certdocs\_bucket'); |
|  | $object = $bucket->object($fileContent); |
|  | return $object->info(); |
|  | } |
|  | //this (listFiles) method not used in this example but you may use according to your need |
|  | function listFiles($bucket, $directory = null) { |
|  |  |
|  | if ($directory == null) { |
|  | // list all files |
|  | $objects = $bucket->objects(); |
|  | } else { |
|  | // list all files within a directory (sub-directory) |
|  | $options = array('prefix' => 'uploads/'); |
|  | $objects = $bucket->objects(); |
|  | } |
|  |  |
|  | foreach ($objects as $object) { |
|  | print $object->name() . PHP\_EOL; |
|  | // NOTE: if $object->name() ends with '/' then it is a 'folder' |
|  | } |
|  | } |
|  | ?> |

### ****requests.php****

|  |
| --- |
| <?php |
|  | include\_once 'config.php'; |
|  |  |
|  | $action = filter\_var(trim($\_REQUEST['action']), FILTER\_SANITIZE\_STRING); |
|  | if ($action == 'upload') { |
|  | $response['code'] = "200"; |
|  | if ($\_FILES['file']['error'] != 4) { |
|  | //set which bucket to work in |
|  | $bucketName = "certdocs\_bucket"; |
|  | // get local file for upload testing |
|  | $fileContent = file\_get\_contents($\_FILES["file"]["tmp\_name"]); |
|  | // NOTE: if 'folder' or 'tree' is not exist then it will be automatically created ! |
|  | $cloudPath = 'uploads/' . $\_FILES["file"]["name"]; |
|  |  |
|  | $isSucceed = uploadFile($bucketName, $fileContent, $cloudPath); |
|  |  |
|  | if ($isSucceed == true) { |
|  | $response['msg'] = 'SUCCESS: to upload ' . $cloudPath . PHP\_EOL; |
|  | // TEST: get object detail (filesize, contentType, updated [date], etc.) |
|  | $response['data'] = getFileInfo($bucketName, $cloudPath); |
|  | } else { |
|  | $response['code'] = "201"; |
|  | $response['msg'] = 'FAILED: to upload ' . $cloudPath . PHP\_EOL; |
|  | } |
|  | }  $localhost = "104.155.58.209"; #localhost  $dbusername = "root"; #username of phpmyadmin  $dbpassword = "classic123"; #password of phpmyadmin  $dbname = "emaildb"; #database name    #connection string  $conn = mysqli\_connect($localhost,$dbusername,$dbpassword,$dbname);  if($conn)  {  echo "sucess";  }  else  {  echo "error";  }  $filename=$\_POST["t1"];  $link= 'https://storage.googleapis.com/certdocs\_bucket11/uploads/' . $\_FILES["file"]["name"];  #sql query to insert into database  $sql = "INSERT into gcsviewtb(name,link) VALUES('$filename','$link')";    if(mysqli\_query($conn,$sql)){    echo "File Sucessfully uploaded,Click to View";  }  else{  echo "Error";  } |
|  | header("Content-Type:application/json"); |
|  | echo json\_encode($response); |
|  | exit(); |
|  | } |

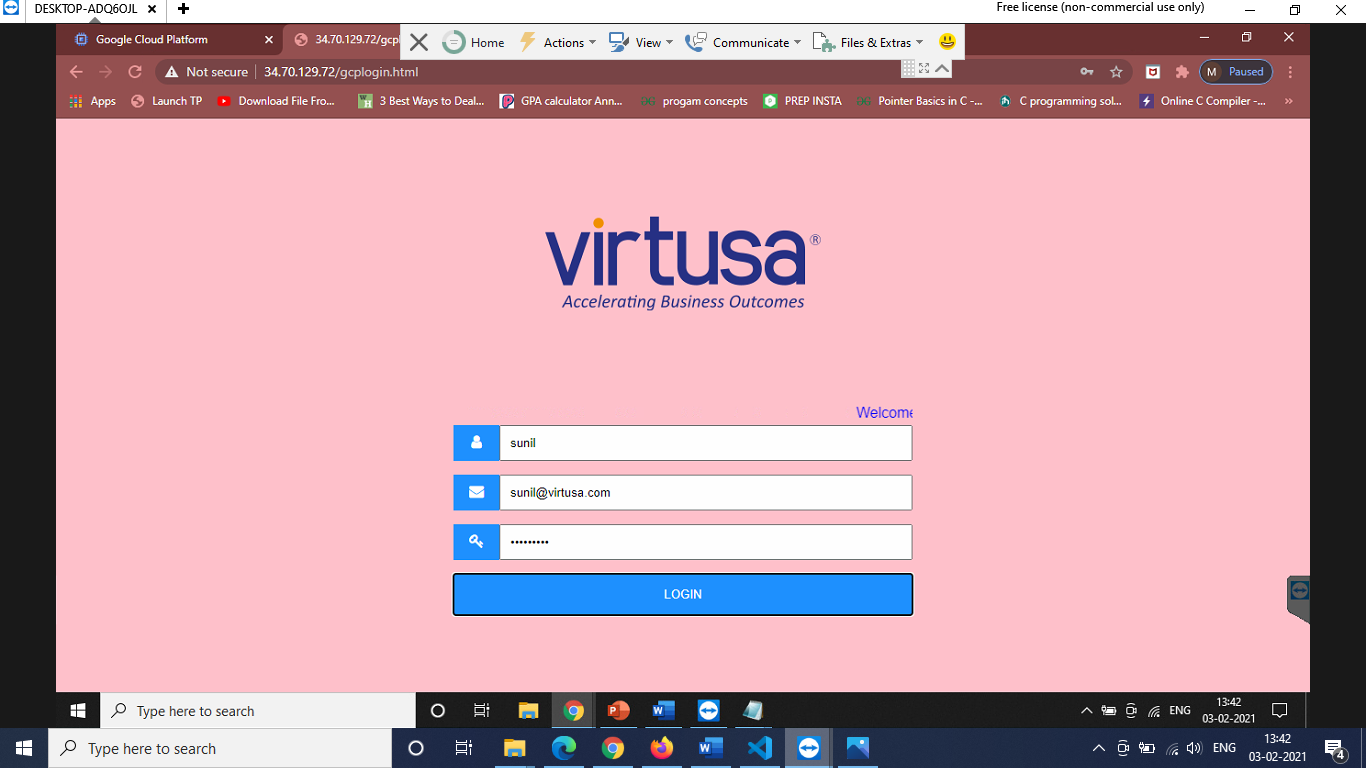
### ****index.php****

|  |
| --- |
| <html> |
|  | <head> |
|  | <meta charset="UTF-8"> |
|  | <title>GCP Storage File Upload using PHP</title> |
|  | </head> |
|  | <body> |
|  | <form id="fileUploadForm" method="post" enctype="multipart/form-data"> |
|  | <input type="file" name="file"/> |
|  | <input type="submit" name="upload" value="Upload"/> |
|  | <span id="uploadingmsg"></span> |
|  | <hr/> |
|  | <strong>Response (JSON)</strong> |
|  | <pre id="json">json response will be shown here</pre> |
|  |  |
|  | <hr/> |
|  | <strong>Public Link</strong> <span>(https://storage.googleapis.com/[BUCKET\_NAME]/[OBJECT\_NAME])</span><br/> |
|  | <b>Note:</b> we can use this link only if object or the whole bucket has made public, which in our case has already made bucket public<br/> |
|  | <div id="output"></div> |
|  | </form> |
|  | <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.4.1/jquery.min.js" integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo=" crossorigin="anonymous"></script> |
|  | <script> |
|  | $("#fileUploadForm").submit(function (e) { |
|  | e.preventDefault(); |
|  | var action = "requests.php?action=upload"; |
|  | $("#uploadingmsg").html("Uploading..."); |
|  | var data = new FormData(e.target); |
|  | $.ajax({ |
|  | type: 'POST', |
|  | url: action, |
|  | data: data, |
|  | /\*THIS MUST BE DONE FOR FILE UPLOADING\*/ |
|  | contentType: false, |
|  | processData: false, |
|  | }).done(function (response) { |
|  | $("#uploadingmsg").html(""); |
|  | $("#json").html(JSON.stringify(response, null, 4)); |
|  | //https://storage.googleapis.com/[BUCKET\_NAME]/[OBJECT\_NAME] |
|  | $("#output").html('<a href="https://storage.googleapis.com/' + response.data.bucket + '/' + response.data.name + '"><i>https://storage.googleapis.com/' + response.data.bucket + '/' + response.data.name + '</i></a>'); |
|  | if(response.data.contentType === 'image/jpeg' || response.data.contentType === 'image/jpg' || response.data.contentType === 'image/png') { |
|  | $("#output").append('<br/><img src="https://storage.googleapis.com/' + response.data.bucket + '/' + response.data.name + '"/>'); |
|  | } |
|  | }).fail(function (data) { |
|  | //any message |
|  | }); |
|  | }); |
|  | </script> |
|  | </body> |
|  | </html> |

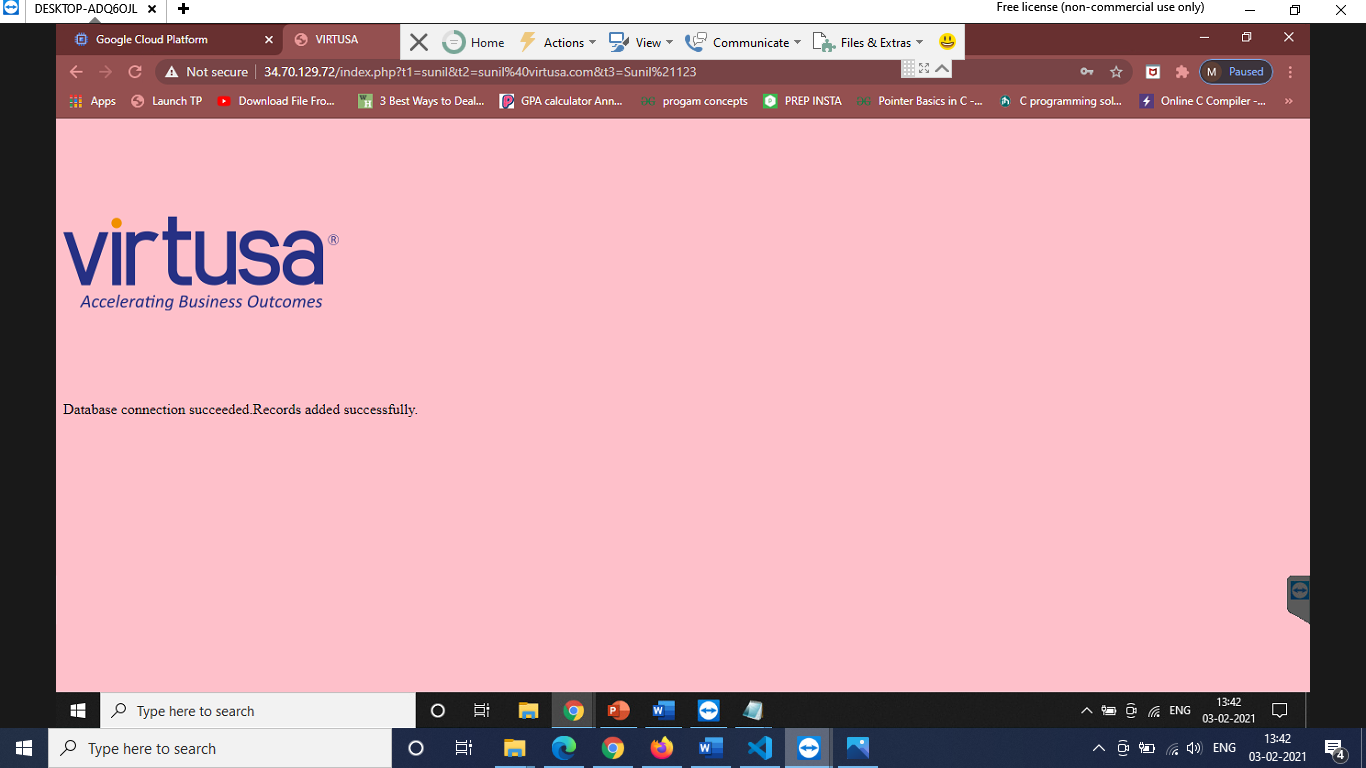
**Task 8: Check the web application**

Return to the web browser tab in which you opened your **gcp-task1-instance1** VM instance's external IP address/gcplogin.html.

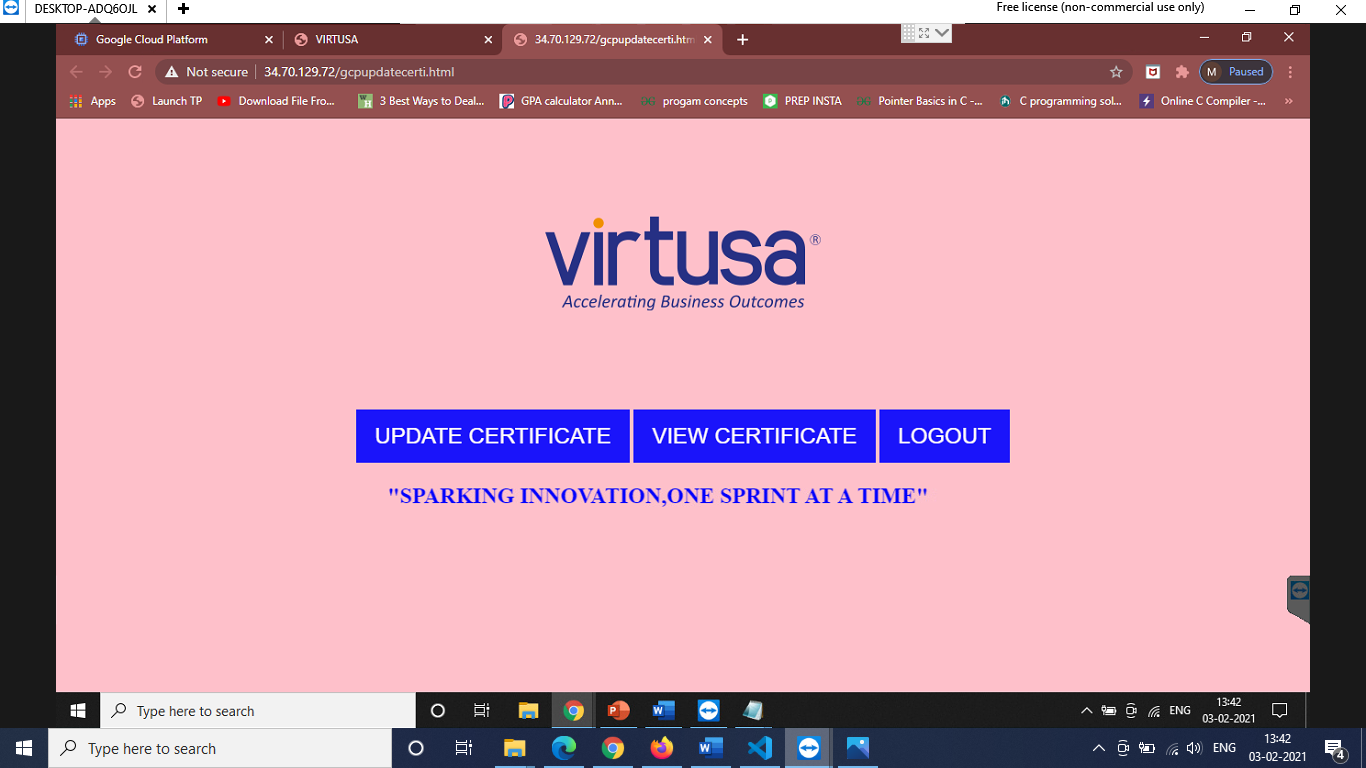
**Login Page**- User can login by specifying username ,email and password with the given constraints.



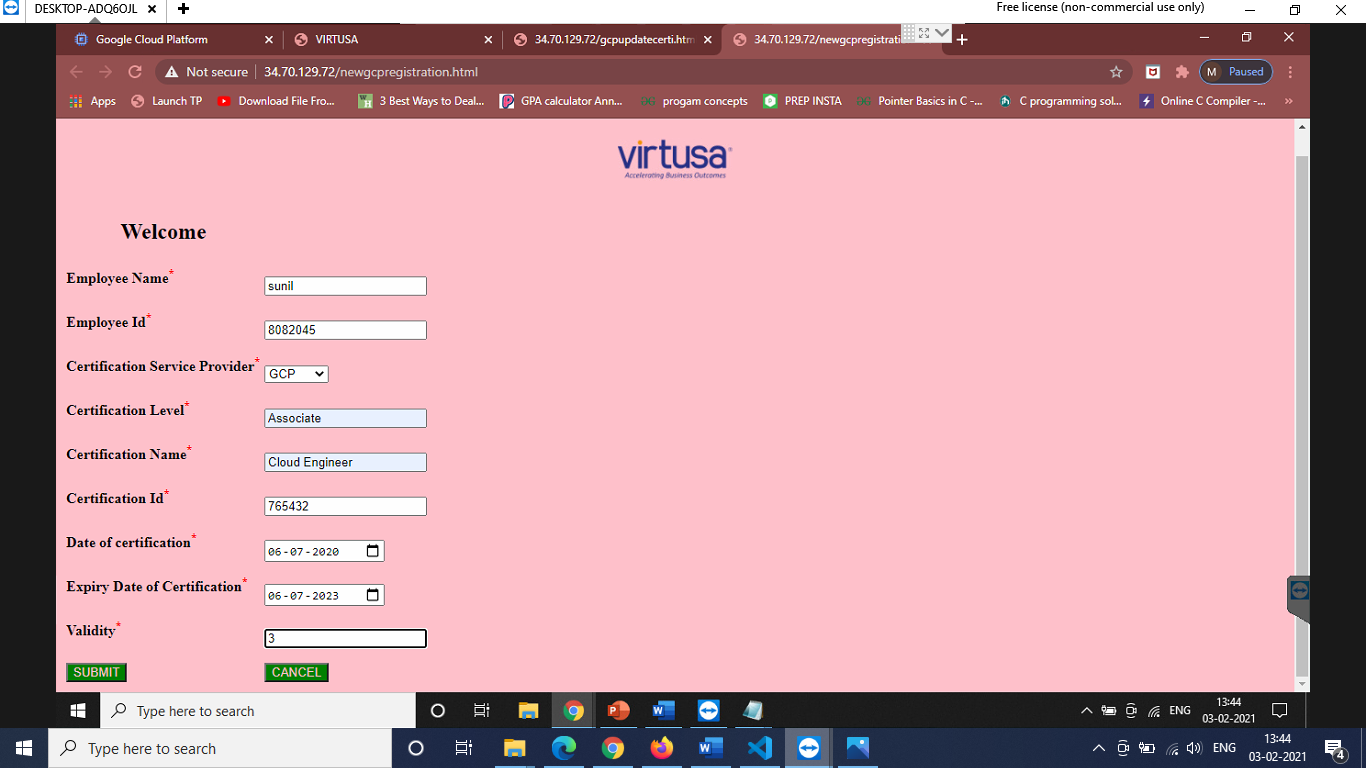
After successfully adding the user information to the database



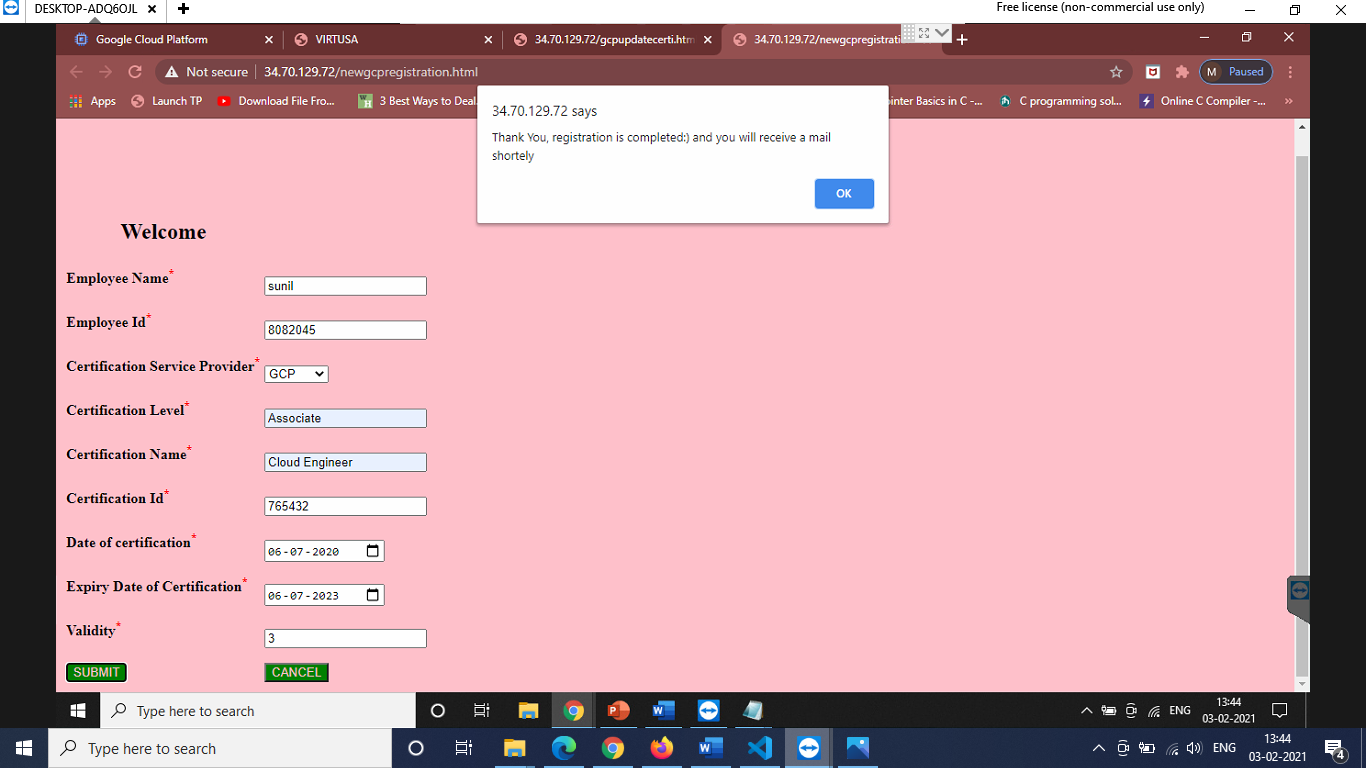
After successful login , user can choose to update or view their certification details.



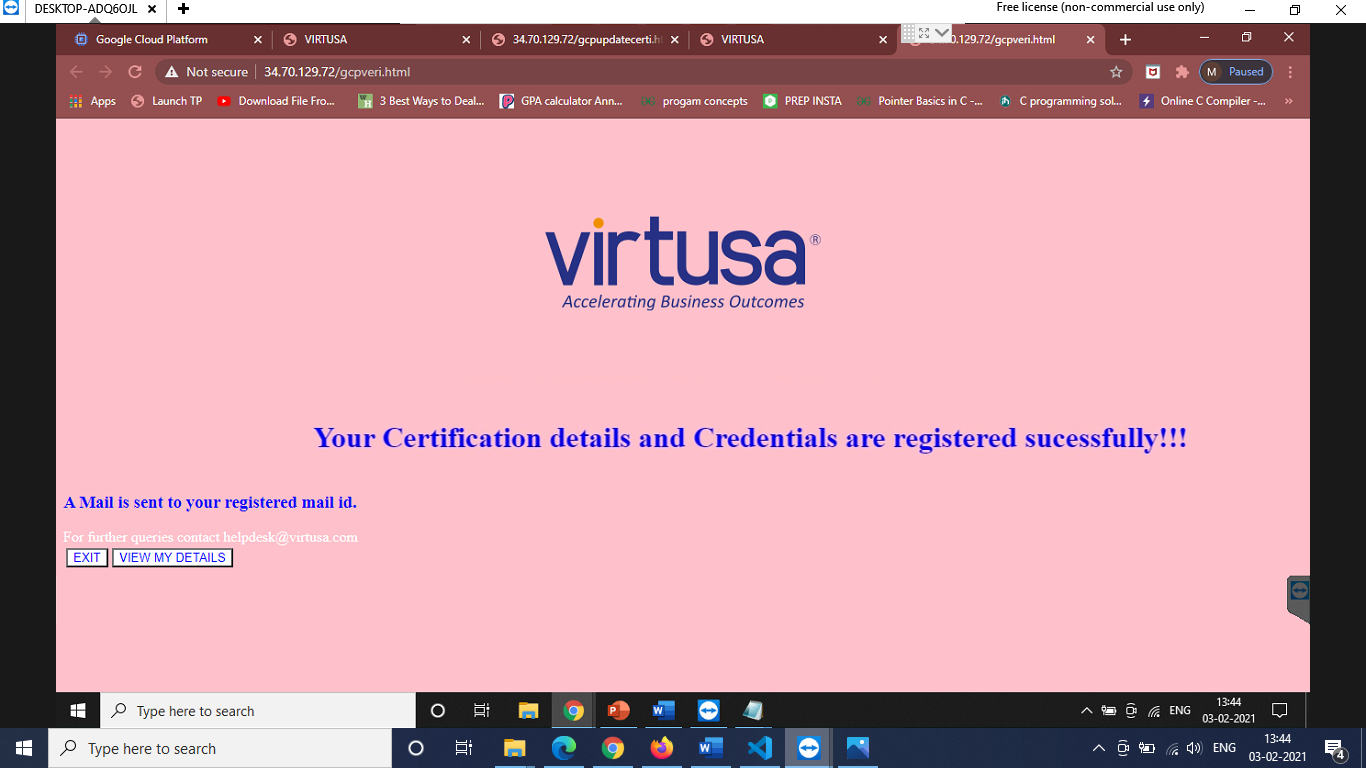
**Registration Page**-allows the user to fill in the details satisfying the functional requirements.



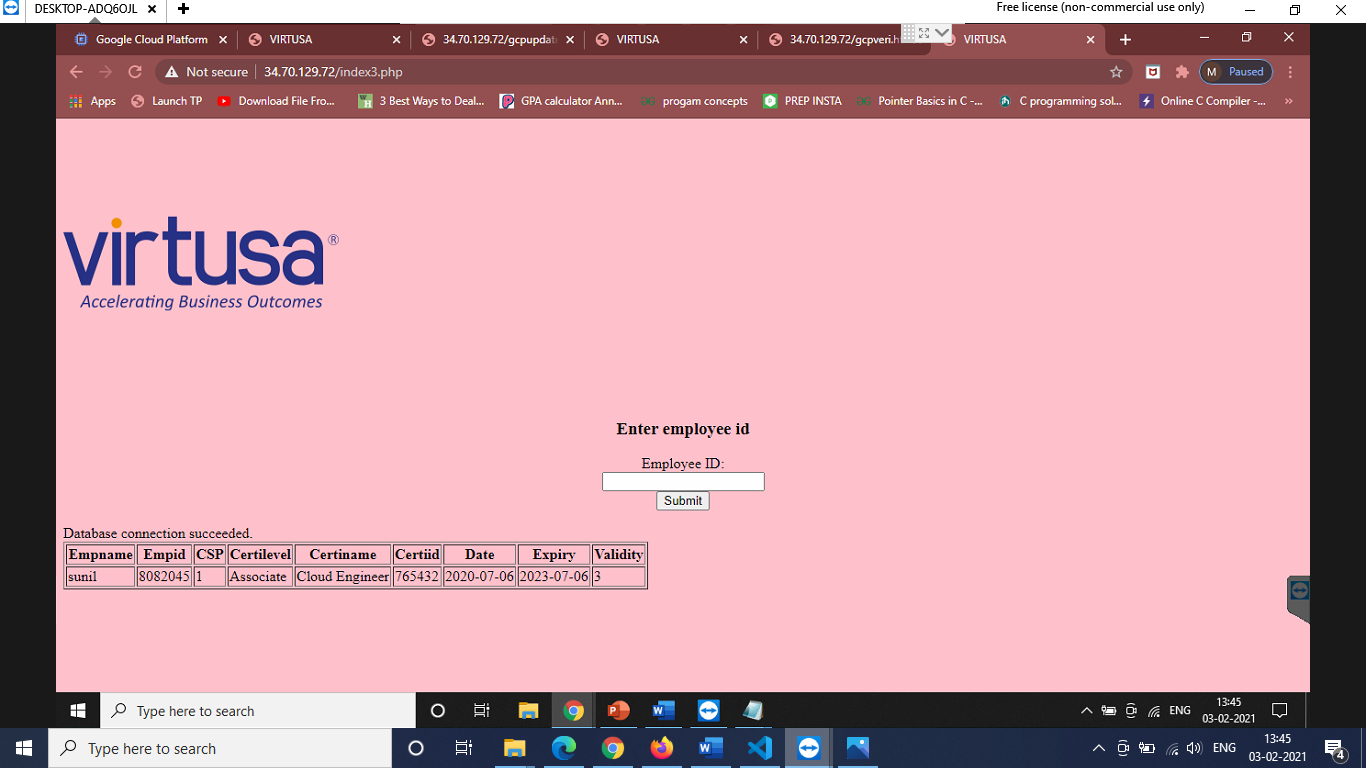
Providing confirmation message after successful registration.



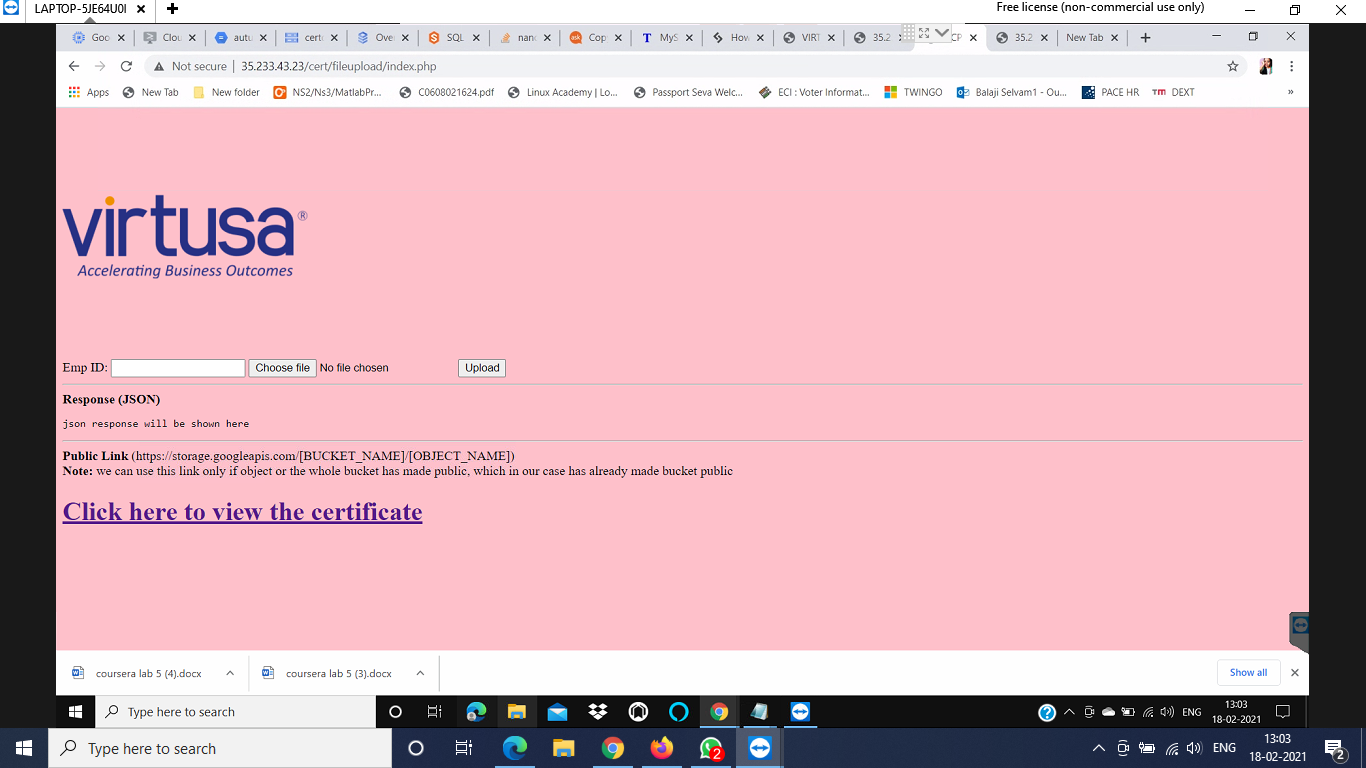
User can either exit or choose to view their details



**View Details Page**: allows the user to view their details by specifying the Employee id.

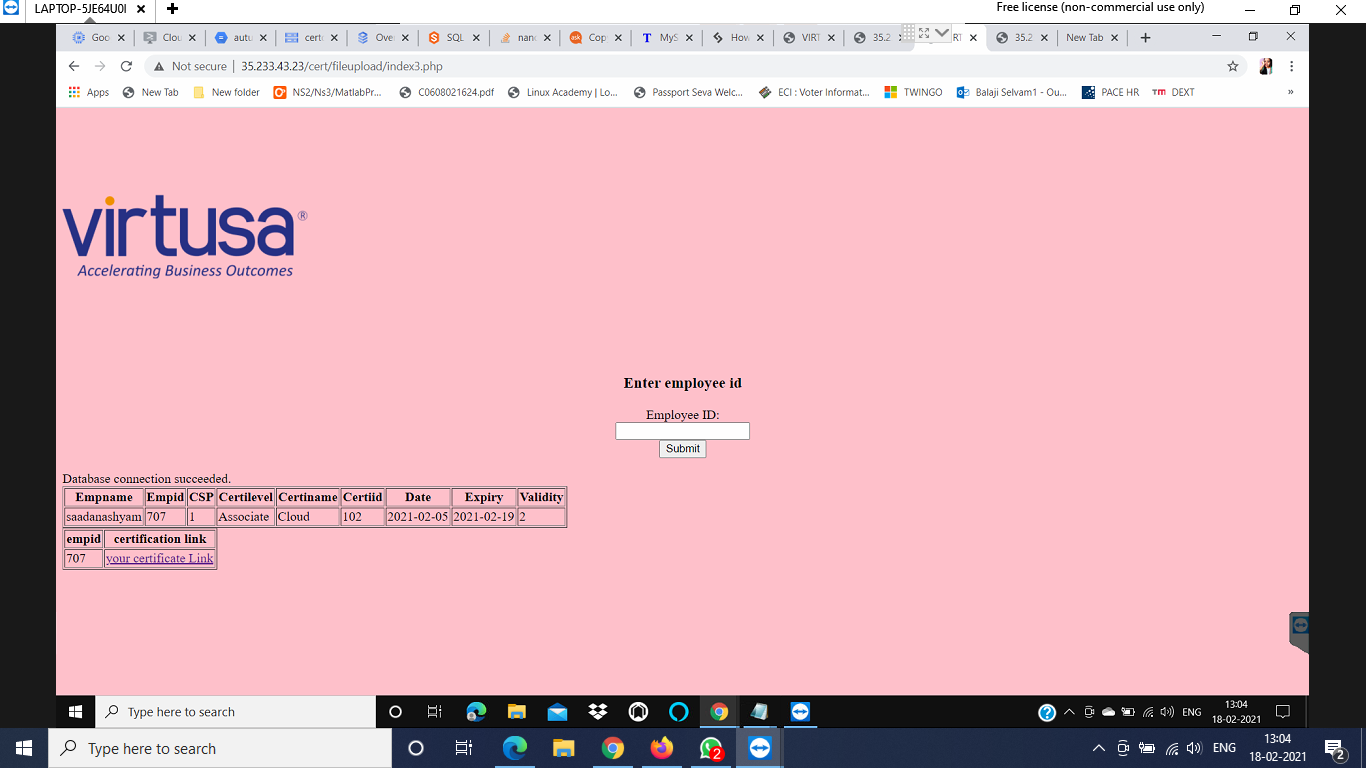


**Upload certificate page:** allows the user to upload their certificate by specifying the Employee id.





**View details page:** allows the user to view their certificate by specifying the Employee id.



.