SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 1

Q1. Working of Google Drive to make Spreadsheet and Notes (10 Marks)

Solution:

Google Drive is a cloud-based file storage and collaboration platform provided by Google. It allows users to **create, edit, and share** documents, spreadsheets, and notes online. Below are the steps to make a **Spreadsheet** and **Notes** using Google Drive:

Steps to create Spreadsheet:

1. Go to https://drive.google.com.
2. Click on **"+ New" → Google Sheets**.
3. A new spreadsheet opens; you can enter data in rows and columns.
4. Use formulas like =SUM(A1:A5) or formatting tools for better presentation.
5. Click **Share** → enter email IDs to share the sheet with others (view/edit access).
6. All changes are auto-saved in the cloud.

Steps to make Notes:

1. Click **"+ New" → Google Docs** (for textual notes).
2. Type your notes using headings, bullet points, and formatting tools.
3. Auto-saved in Drive and accessible from any device.
4. Can download in PDF, DOCX, or TXT format.

Advantages:

- Real-time collaboration
- Auto-saving and version control
- Accessible anywhere through the internet
- Secure and easy sharing option

Q2. Create and Host Static Web Page using AWS (20 Marks)

Answer:

We are using **AWS S3 (Simple Storage Service)** to host a static website.

1. **Create AWS Account:**
   Sign in to AWS Management Console → search for **S3** service.
2. **Create an S3 Bucket:**
   - Click **Create bucket**
   - Enter bucket name (e.g., mywebsite-bucket)
   - Choose region (e.g., Asia Pacific (Mumbai) region)
   - Uncheck “Block all public access” → Confirm and Create bucket.
3. **Upload Website Files:**
   - Prepare your HTML file (e.g., index.html) and CSS if any.
   - Upload files to the bucket using “Upload” button.
4. **Enable Static Website Hosting:**
   - Go to bucket → **Properties** tab → Enable **Static website hosting**.
   - Choose “Host a static website”.
   - Specify:
     - Index document: index.html
     - Error document: error.html (optional)
   - Click **Save changes**.
5. **Set Bucket Policy for Public Access:**
   - Go to **Permissions → Bucket policy**

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "PublicReadGetObject",
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::your-bucket-name/*"  }] }
```

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 2

Q1. Practical Implementation of Storage as a Service using OwnCloud

Theory:

Storage as a Service (STaaS) is a cloud service model in which a service provider gives access to data storage facilities over the internet.
Users can upload, manage, and access files remotely using the cloud.

Example Providers: AWS S3, Google Drive, Dropbox, OwnCloud, etc.

OwnCloud is an open-source platform that provides cloud-based storage similar to Google Drive.
It can be deployed on a server or cloud instance to allow users to upload, sync, and share files securely.

Steps to Implement Using AWS + OwnCloud

Step 1: Launch AWS EC2 Instance

1. Login to **AWS Management Console** → Go to **EC2 Service**.
2. Click **Launch Instance**.
3. Choose **Ubuntu Server 22.04 LTS (Free Tier Eligible)**.
4. Configure:
    - Instance Type: t2.micro
    - Key Pair: Create new or use existing one
    - Allow inbound HTTP (port 80) and HTTPS (port 443)
5. Launch the instance.

Step 2: Connect to EC2 Instance

Use terminal or command prompt:

ssh -i "keypair.pem" ubuntu@<Public-IP-of-EC2>

Step 3: Update System Packages

sudo apt update && sudo apt upgrade -y

Step 4: Install Apache, PHP, and MariaDB

sudo apt install apache2 mariadb-server libapache2-mod-php php php-mysql php-zip php-gd php-curl php-mbstring php-intl php-bcmath php-imagick php-xml php-ldap php-json -y

Start and enable services:

sudo systemctl enable apache2

sudo systemctl enable mariadb

Step 5: Configure Database for OwnCloud

sudo mysql -u root -p

Then run:

CREATE DATABASE owncloud;

CREATE USER 'owncloduser'@'localhost' IDENTIFIED BY 'password';

GRANT ALL PRIVILEGES ON owncloud.* TO 'owncloduser'@'localhost';

FLUSH PRIVILEGES;

EXIT;

Step 6: Download and Install OwnCloud

cd /var/www/

sudo wget https://download.owncloud.org/community/owncloud-complete-latest.tar.bz2

sudo tar -xjf owncloud-complete-latest.tar.bz2

sudo chown -R www-data:www-data /var/www/owncloud

sudo chmod -R 755 /var/www/owncloud

Step 7: Configure Apache for OwnCloud

sudo nano /etc/apache2/sites-available/owncloud.conf

Add:

<VirtualHost *:80>

DocumentRoot /var/www/owncloud

ServerName your-domain-or-public-ip

```
    <Directory /var/www/owncloud/>
        Options +FollowSymlinks
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

Then:

```
sudo a2ensite owncloud.conf
sudo a2enmod rewrite
sudo systemctl restart apache2
```

Step 8: Access OwnCloud via Browser

Open:
http://<EC2-public-IP>

- Set **Admin username** and **password**
- Database type: MySQL/MariaDB
- Database user: owncloduser
- Database name: owncloud
- Database password: password

Click **Finish Setup**

Step 9: Upload and Access Files

After login:

- Upload files (e.g., documents, images, PDFs).
- Create folders.
- Share files via links.
- View storage usage.

You've successfully implemented Storage as a Service using OwnCloud.

Q 2).Practical Implementation of Cloud Security

Steps to Implement Cloud Security using AWS

Step 1: Create an IAM User

1. Go to AWS Management Console → IAM (Identity & Access Management).
2. Click Users → Add User.
3. Enter username: clouduser.
4. Choose Access type:
    - Programmatic access
    - AWS Management Console access
5. Set custom password → Click Next.

Step 2: Assign Permissions

You can control what actions the user can perform.

- Choose Attach existing policies directly.
- Select AmazonS3ReadOnlyAccess policy.
  (This means the user can *view* but not *delete or modify* data in S3.)

This demonstrates Role-Based Access Control (RBAC).

Step 3: Create an S3 Bucket and Enable Security Features

1. Go to **S3 → Create bucket.**
    - Name: secure-data-bucket
    - Region: Asia Pacific (Mumbai)
    - **Enable Bucket Versioning** (to keep file history).
    - **Block Public Access:** Keep **ON** to restrict unauthorized access.
2. Upload a sample file: confidential.txt.

Step 4: Apply Bucket Policy for Controlled Access

3. Go to **Permissions → Bucket Policy**

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 3

Q1. Working of Google Drive to Create Exam Form and Generate Result (10 Marks)

Tools Used:

- **Google Forms** – to collect exam data.
- **Google Sheets** – to store responses and generate result.
- **Google Docs / Drive** – to store and share files.

Procedure:

Step 1: Create an Exam Form

1. Open https://forms.google.com.
2. Click **Blank Form → Title:** "M.Sc. Cloud Computing – Internal Exam Form".
3. Add the following fields:
    - Full Name (Short Answer)
    - PRN Number (Short Answer)
    - Subject Name (Multiple choice or dropdown)
    - Marks Obtained (Number input)
    - Email ID (Short Answer)
4. Go to **Settings → Responses → Collect email addresses (optional)**.
5. Click **Send → Get link** to share form.

Step 2: Store Responses in Google Sheets

1. After responses start coming, click **"Responses" → "Link to Sheets"**.
2. A new spreadsheet opens automatically storing all form responses.

Example:

Name	PRN	Subject	Marks	Email
Priya Patil	123	Cloud Computing	85	priya@gmail.com

Step 3: Generate Result Automatically

1. In Google Sheets, create a new column: **Result Status**.
2. Apply formula:
3. =IF(D2>=40, "Pass", "Fail")

(Assuming “Marks” are in column D)

4. Apply formula for Grade:
5. =IF(D2>=80,"A",IF(D2>=60,"B",IF(D2>=40,"C","F")))

Now the result is generated automatically based on marks.

Step 4: Share the Result

- Click **File → Share → View Link**
- Choose “View Only” access and share with students.

Q2. Practical Implementation of File Sharing and Storage as a Service (20 Marks)

Aim:

To demonstrate **File Sharing** and **Storage as a Service** using **AWS S3 (Simple Storage Service).**

Procedure:

Step 1: Create S3 Bucket

1. Login to AWS → open **S3 Service**.
2. Click **Create Bucket**.
3. Enter bucket name: student-file-storage.
4. Select Region: Asia Pacific (Mumbai).
5. Uncheck “Block all public access” (for demo purpose).
6. Click **Create Bucket.**

Step 2: Upload Files

1. Open the created bucket → Click **Upload**.
2. Choose sample files such as:
    - ExamResult.pdf
    - StudentRecord.csv
    - Notes.txt
3. Click **Upload.**

Step 3: Enable File Sharing

1. Select a file → **Actions → Share → Create a Pre-signed URL.**
2. Choose expiry time (e.g., 1 hour or 7 days).
3. Copy and share the URL with another user.
   That user can now access/download the file securely.

Step 4: Enable Versioning (Optional)

1. Go to **Properties → Bucket Versioning → Enable.**
2. Upload another file with same name → AWS keeps both versions safely.

Step 5: Apply Access Policy (Security)

Go to **Permissions → Bucket Policy**, add:

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::student-file-storage/*"
  }]
}
```

This allows read-only access to files.

Step 6: Test File Sharing

- Open the generated S3 file link in browser.
- Verify file access/download works.
- Try to delete file → not allowed (restricted permissions).

File sharing and cloud storage successfully implemented.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 4

Q1. Working and Implementation of Software as a Service (SaaS) using Google

[10 Marks]

Theory:

Software as a Service (SaaS) is a cloud service model that provides ready-to-use software over the internet.
Users do not need to install or maintain applications; they simply use them via a web browser.

Examples:

- Gmail (Email Service)
- Google Docs (Document Editing)
- Google Sheets (Spreadsheets)
- Google Drive (Cloud Storage)
- Google Meet (Video Conferencing)

Working:

1. **Access:** The software is accessed through a browser (no installation needed).
2. **Authentication:** User logs in with credentials (e.g., Gmail ID).
3. **Data Management:** All files are stored and synced in the cloud (Google Drive).
4. **Collaboration:** Multiple users can edit and comment in real-time.
5. **Maintenance:** Software updates and security patches are handled by Google automatically.

Implementation Example (Using Google Docs):

1. Open https://docs.google.com.
2. Click **Blank Document**.
3. Type content, format text, and insert images/tables.
4. Click **Share → Add Collaborator Email →** Choose permission (View/Edit).
5. Save automatically in Google Drive.

Demonstrates how SaaS applications are accessed and used over the cloud.

Advantages of SaaS:

- No installation required
- Cost-effective (subscription-based)
- Auto-updates & maintenance

- Accessible from any device
- Real-time collaboration

Q2. Create Google Form for Generating Certificates for Workshop

[20 Marks]

Aim:

To design a **Google Form** that collects participant details for a workshop and automatically generates participation certificates using **Google Forms + Google Sheets + Google Docs + Autocrat Add-on**.

Tools Used:

- **Google Forms** – For collecting participant data
- **Google Sheets** – To store responses
- **Google Docs** – For certificate template
- **Autocrat Add-on** – To auto-generate certificates

Procedure:

Step 1: Create the Google Form

1. Go to https://forms.google.com.
2. Click **Blank Form → Title:** "Workshop Registration & Certificate Form".
3. Add the following fields:
    - Full Name (Short answer)
    - Email ID
    - Workshop Topic (Short answer)
    - Date of Workshop (Date field)
4. Go to **Settings → Collect email addresses** (optional).
5. Click **Send → Copy the form link** to share.

Step 2: Link Responses to Google Sheet

1. After creating the form, click **Responses → Link to Sheets.**
2. All submissions will automatically appear in a spreadsheet.

Example:

Timestamp	Full Name	Email	Workshop Topic	Date
11/11/2025	Priya Patil	priya@gmail.com	Cloud Computing	10/11/2025

Step 3: Create Certificate Template in Google Docs

1. Open https://docs.google.com.

2. Create a new document titled **Workshop Certificate Template**.
3. Design certificate with text like:
4. Certificate of Participation
5. This is to certify that <<Full Name>>
6. has successfully attended the Workshop on <<Workshop Topic>>
7. held on <<Date>>.
8. Save and format it nicely with borders, logo, or signature.

Step 4: Use Autocrat Add-on to Generate Certificates

1. Open the linked Google Sheet (from Step 2).
2. Click **Extensions → Add-ons → Get add-ons → Search "Autocrat" → Install.**
3. After installation:
    - Go to **Extensions → Autocrat → Launch.**
    - Click **New Merge Job.**
    - Select the **Google Docs template** created in Step 3.
    - Map fields (e.g., <<Full Name>> → Form column "Full Name").
    - Choose output file type (PDF).
    - Choose to **Email each participant automatically** if needed.
4. Click **Run Job** → Certificates will be automatically generated and saved in Google Drive.

Each participant receives a personalized certificate generated automatically.

Step 5: Verify Output

- Open your Google Drive → A new folder "Autocrat Output" is created.
- Each participant's certificate (PDF) is saved with their name.

Example file name:
Certificate_Priya_Patil.pdf

Result:

Successfully created a **Google Form** that collects data and automatically **generates digital certificates** using **Google Workspace tools (SaaS model)**.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 5

Q1. Working and Implementation of Infrastructure as a Service (IaaS)

[10 Marks]

Theory:

Infrastructure as a Service (IaaS) provides **virtualized computing resources** (servers, storage, networking) over the internet.
It allows users to create and manage virtual machines (VMs) on demand.

Examples of IaaS Providers:

- Amazon Web Services (AWS) – EC2
- Microsoft Azure – Virtual Machines
- Google Cloud Platform (GCP) – Compute Engine
- IBM Cloud, Oracle Cloud, etc.

Working of IaaS:

1. **Provisioning:** User selects OS, CPU, RAM, and storage.
2. **Deployment:** A virtual server (instance) is created in the cloud.
3. **Configuration:** Software and runtime environments are installed.
4. **Usage:** User can connect, store files, and run programs.
5. **Billing:** Pay only for resources used (Pay-as-you-go).

Implementation using AWS EC2

Step 1: Login to AWS

- Go to https://aws.amazon.com → Sign in to **AWS Management Console**.
- Search for **EC2**.

Step 2: Launch a Virtual Machine

1. Click **Launch Instance**.
2. Enter name: MyIaaSInstance.
3. Choose AMI: **Ubuntu Server 22.04 LTS (Free Tier)**.
4. Select Instance type: **t2.micro**.
5. Create/Select key pair for SSH login.
6. Configure security group:
    - Allow **SSH (port 22)** and **HTTP (port 80)**.
7. Click **Launch Instance**.

Step 3: Connect to the Instance

Use terminal (Linux/macOS) or PuTTY (Windows):

ssh -i "keypair.pem" ubuntu@<public-ip-address>

Step 4: Verify System

Once logged in:

uname -a

Displays system information (verifies virtual machine is running).

Step 5: Install Software (Example: Apache Web Server)

sudo apt update

sudo apt install apache2 -y

Check service:

sudo systemctl status apache2

Open in browser:
http://<your-public-ip> → Default Apache page appears.

This proves successful IaaS setup (infrastructure running remotely).

Result:

Successfully implemented **Infrastructure as a Service (IaaS)** using **AWS EC2** by launching, connecting, and configuring a virtual server.

Q2. Developing Python Application (Addition of 10 Numbers) using Google App Engine

[20 Marks]

Theory:

Google App Engine (GAE) is a **Platform as a Service (PaaS)** under **Google Cloud Platform (GCP)**.
It allows developers to deploy and manage applications without handling infrastructure.

Procedure:

Step 1: Setup Google Cloud

1. Visit https://console.cloud.google.com.
2. Create a new project: **"PythonAdditionApp"**.

3. Enable **App Engine API**.
4. Install **Google Cloud SDK (gcloud)** on your system.

Step 2: Create Project Folder

On your local system:

mkdir addition_app

cd addition_app

Create a Python file:
main.py

```
from flask import Flask



app = Flask(__name__)



@app.route('/')

def add_numbers():

    numbers = [1,2,3,4,5,6,7,8,9,10]

    total = sum(numbers)

    return f"<h2>The addition of 10 numbers is: {total}</h2>"



if __name__ == "__main__":

    app.run()
```

This app adds 10 numbers and shows the total result.

Step 3: Create Requirements File

requirements.txt

```
Flask==2.0.3
```

Step 4: Create App Engine Configuration File

app.yaml

runtime: python39

entrypoint: gunicorn -b :$PORT main:app

Step 5: Initialize and Deploy Application

Run the following commands:

gcloud init

gcloud app create --region=asia-south1

gcloud app deploy

Step 6: Access Application

After successful deployment, note the URL (shown in terminal):
https://addition_app.appspot.com

Open in browser to view the output:
“The addition of 10 numbers is: 55”

The addition of 10 numbers is: 55

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 6

Q1. Write a Program for Web Feed (RSS Feed)

[10 Marks]

Theory:

A **Web Feed** (or **RSS Feed**) is a standardized XML file that provides frequently updated information — like blog posts, news headlines, or articles.
RSS stands for **Really Simple Syndication**.
Web feeds allow automatic updates without manually checking websites.

Example Use:
News websites and blogs provide RSS feeds for readers and applications to get updates automatically.

Algorithm:

1. Create XML structure with RSS version and channel information.
2. Add multiple <item> entries for articles/posts.
3. Save the file as feed.xml.
4. Open it in a browser to verify feed structure.

Program (Python):

```
from datetime import datetime



# Function to create RSS feed

def generate_rss():

    feed_content = f"""<?xml version="1.0" encoding="UTF-8" ?>

    <rss version="2.0">

    <channel>

        <title>My Web Feed</title>

        <link>https://www.mywebsite.com</link>
```

```
    <description>This is a sample RSS feed for latest updates.</description>
    <language>en-us</language>
    <lastBuildDate>{datetime.now()}</lastBuildDate>

    <item>
        <title>New Cloud Computing Course Launched</title>
        <link>https://www.mywebsite.com/cloud-course</link>
        <description>Learn AWS, Google Cloud, and Azure basics.</description>
        <pubDate>{datetime.now()}</pubDate>
    </item>

    <item>
        <title>AI Seminar Registration Open</title>
        <link>https://www.mywebsite.com/ai-seminar</link>
        <description>Participate in the upcoming AI & ML seminar.</description>
        <pubDate>{datetime.now()}</pubDate>
    </item>

</channel>
</rss>"""

# Save the RSS content to file
with open("feed.xml", "w", encoding="utf-8") as file:
    file.write(feed_content)
```

```
    print("RSS feed created successfully as feed.xml")


# Run the function

generate_rss()
```

Output (feed.xml file):

```
<rss version="2.0">
  <channel>
    <title>My Web Feed</title>
    <link>https://www.mywebsite.com</link>
    <description>This is a sample RSS feed for latest updates.</description>
    <item>
      <title>New Cloud Computing Course Launched</title>
      <link>https://www.mywebsite.com/cloud-course</link>
      <description>Learn AWS, Google Cloud, and Azure basics.</description>
    </item>
  </channel>
</rss>
```

Output file "feed.xml" can be opened in browser or RSS reader.

Result:

Successfully created a **Python program** that generates an **RSS Web Feed** (XML file) for publishing latest website updates.

Q2. Create Google Form for Generating Survey about Beauty Products

[20 Marks]

Aim:

To design a **Google Form survey** to collect opinions from users about various **Beauty Products** (e.g., cosmetics, skincare, haircare).

Tools Used:

- **Google Forms** – to create the survey
- **Google Sheets** – to collect and analyze responses

Procedure:

Step 1: Create a New Google Form

1. Go to https://forms.google.com.
2. Click **Blank Form → Title:** "Beauty Products Feedback Survey".
3. Add **Form Description:**
   "This survey aims to collect customer opinions about beauty and skincare products."

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 7

Q1. Working and Implementation of Identity Management (IAM)

[10 Marks]

Theory:

Identity Management (IAM) in cloud computing refers to the process of defining and managing the roles, privileges, and access rights of individual users to various cloud resources.

AWS IAM (Identity and Access Management) enables you to:

- Create and manage users and groups.
- Assign permissions using policies.
- Secure access using Multi-Factor Authentication (MFA).

Key Components of IAM:

1. **Users:** Individual identities (e.g., employees, students).
2. **Groups:** Collection of users with similar permissions.
3. **Policies:** JSON documents defining allowed or denied actions.
4. **Roles:** Temporary permissions for AWS services or users.
5. **MFA (Multi-Factor Authentication):** Adds extra security during login.

Steps to Implement Identity Management using AWS IAM

Step 1: Login to AWS Console

- Go to https://aws.amazon.com
- Open **IAM Service** from AWS Console.

Step 2: Create a New User

1. Click **Users → Add User.**
2. Enter username: student_user.
3. Select **Access Type:**
    - ✅ AWS Management Console Access.
    - ✅ Programmatic Access (for CLI/API use).

4. Set custom password (e.g., Student@123).

Step 3: Assign Permissions

- Choose **Attach Existing Policies Directly.**
- Select **AmazonS3ReadOnlyAccess** policy.
  → This allows the user to view files in S3 buckets but not delete or modify them.

Step 4: Create a Group (Optional)

- Go to **Groups → Create Group → Name:** Students.
- Attach same **ReadOnly policy** to the group.
- Add multiple users to the group.

Step 5: Apply MFA (Optional but recommended)

- Select user → **Security Credentials → Manage MFA.**
- Link virtual MFA app (e.g., Google Authenticator).
- Scan QR code → enter authentication code to verify.

Step 6: Test Access

- Log out and sign in using **new IAM user credentials**.
- Try accessing **S3 Service → Buckets.**
- You can **view files**, but delete/upload options are **restricted**.

This confirms IAM-based identity management and permission control.

Result:

Successfully implemented **Identity and Access Management (IAM)** in AWS to create and manage users with controlled permissions and security authentication.

Q2. Practical Implementation of File Sharing and Storage as a Service

[20 Marks]

Aim:

To demonstrate **File Sharing** and **Storage as a Service (STaaS)** using **AWS S3 (Simple Storage Service)**.

Theory:

Storage as a Service (STaaS) allows users to store, access, and manage data in the cloud.
File Sharing enables sharing of these files securely via links or permissions.

Example Providers: AWS S3, Google Drive, Dropbox, OneDrive.

Steps to Implement Using AWS S3

Step 1: Create an S3 Bucket

1. Go to **S3 Service → Create Bucket**.
2. Bucket name: student-file-storage.
3. Region: **Asia Pacific (Mumbai)**.
4. Uncheck **Block all public access** (for demo).
5. Click **Create Bucket.**

Step 2: Upload Files

- Select your new bucket.
- Click **Upload → Add Files → Upload sample documents** (e.g., notes.txt, exam_results.pdf).
- Click **Upload.**

Step 3: Enable File Sharing

1. Select uploaded file → Click **Share → Create Pre-Signed URL.**
2. Set expiration time (e.g., 1 hour).
3. Copy the generated URL and share it with others.
   ✅ This link allows temporary, secure access.

Step 4: Enable Versioning (Optional)

- Go to **Properties → Enable Bucket Versioning.**
- Upload new file with same name → AWS keeps both versions safely.

Step 5: Apply Access Policy (Security Control)

Go to **Permissions → Bucket Policy** and add:

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:GetObject",
```

```
    "Resource": "arn:aws:s3:::student-file-storage/*"
  }]
}
```

This allows **read-only access** to the public for testing.

Step 6: Test File Access

- Copy **Object URL** → open in browser.
- File is visible/downloadable based on permissions.
- Restricted user cannot delete or modify the file (tested with IAM user from Q1).

File sharing and storage implemented successfully.

Result:

Successfully implemented **File Sharing and Storage as a Service** using AWS S3 with secure access control and IAM-based identity management.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 8

Q1. Workings of Google Drive to Create Exam Form and Generate Result

[10 Marks]

Theory:

Google Drive provides multiple **cloud-based tools** (Google Forms, Sheets, Docs) to collect and process information in real time.
With **Google Forms**, we can collect exam responses; and using **Google Sheets**, results can be automatically calculated and shared.

Tools Used:

- **Google Forms** – For creating the online exam.
- **Google Sheets** – For storing responses and generating results.
- **Google Drive** – For file storage and sharing.

Steps to Implement:

Step 1: Create the Exam Form

1. Go to https://forms.google.com.
2. Click on **Blank Form** → Name it **"Online Exam Form."**
3. Add sections:
    - Student Name
    - Roll Number
    - Email ID
    - MCQ Questions (with correct answer options marked).
4. Click **Settings → Make Quiz → Assign Points** for each question.
5. Save and preview the form.

This form will automatically assign scores after submission.

Step 2: Collect Responses

- Open **Responses tab → Link to Google Sheets**.
- Each student's responses are automatically recorded.

Example Sheet:

Timestamp	Name	Roll No	Email	Q1	Q2	Q3	Score
11/11/2025	Riya Patil	12	riya@gmail.com	A	B	C	8

Step 3: Generate Result Sheet

1. Use formula in Google Sheets to compute total score:
2. =SUM(F2:H2)
3. To assign grades automatically:
4. =IF(I2>=8,"A",IF(I2>=5,"B","C"))
5. You can also use **Google App Script** to send results automatically to students via email.

Step 4: Share the Result

- Click **Share → Get Link → Anyone with the link (View only)**.
- Publish final result sheet or email results to students.

Demonstrates working of Google Drive for exam creation and result generation.

Result:

Successfully created and implemented **Exam Form** and **Automatic Result Generation** using **Google Drive, Google Forms, and Google Sheets**.

Q2. Developing Python Application (Factorial of a Given Number) Using Google App Engine

[20 Marks]

Aim:

To develop and deploy a simple **Python web application** on **Google App Engine (GAE)** that calculates the factorial of a given number.

Theory:

Google App Engine (GAE) is a **Platform as a Service (PaaS)** that allows developers to deploy web applications without managing servers.
It supports automatic scaling, version control, and integrates easily with Google Cloud.

Tools Required:

- Google Cloud SDK

- Python 3.x
- Flask framework
- Google App Engine environment

Steps to Implement:

Step 1: Setup Project

1. Create a project in https://console.cloud.google.com.
   Example Project ID: factorial-app-demo.
2. Enable **App Engine** for your project.
3. Install Google Cloud SDK on your system.

Step 2: Create Application Folder

Create a folder structure like:

```
factorial-app/
|
 ├── main.py
└── app.yaml
```

Step 3: Write Python Code (main.py)

```
from flask import Flask, request
app = Flask(__name__)
@app.route('/')
def home():
   return '''
   <h2>Factorial Calculator</h2>
   <form action="/result" method="get">
    Enter a number: <input type="number" name="num">
    <input type="submit" value="Calculate">
   </form>
   '''
@app.route('/result')
def result():
```

```
    try:
        n = int(request.args.get('num'))
        fact = 1
        for i in range(1, n + 1):
            fact *= i
        return f"<h3>Factorial of {n} is: {fact}</h3>"
    except:
        return "<h3>Invalid Input!</h3>"


if __name__ == "__main__":
    app.run()
```

Step 4: Create app.yaml (Configuration File)

```
runtime: python39
entrypoint: gunicorn -b :$PORT main:app
```

Step 5: Deploy on Google App Engine

1. Open terminal → Navigate to project folder.
2. Run commands:
3. gcloud init
4. gcloud app deploy
5. gcloud app browse
6. Your web app will open in the browser, e.g.:
7. https://factorial-app-demo.appspot.com

The application takes a number as input and displays its factorial.

Result:

Successfully developed and deployed a **Python-based factorial calculator** on **Google App Engine**, demonstrating **Platform as a Service (PaaS)** implementation.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 9

Q1. Create Virtual Machine using VirtualBox

[10 Marks]

Theory:

Virtualization is a core concept of cloud computing that allows multiple operating systems to run on a single physical machine.
VirtualBox is an open-source virtualization software developed by Oracle, used to create and manage virtual machines.

Key Terms:

- **Host Machine:** Physical computer where VirtualBox is installed.
- **Guest Machine (VM):** The virtual OS running inside VirtualBox.
- **ISO File:** Disk image used to install an operating system on the VM.

Steps to Create VM using VirtualBox

Step 1: Install VirtualBox

- Download and install **Oracle VM VirtualBox** from: https://www.virtualbox.org

Step 2: Create a New Virtual Machine

1. Open **VirtualBox** → Click **New**.
2. Enter name: Ubuntu_VM (or Windows10_VM).
3. Choose **Type:** Linux (or Windows).
4. Choose **Version:** Ubuntu (64-bit) or appropriate OS version.
5. Click **Next.**

Step 3: Allocate Memory (RAM)

- Assign at least **2048 MB (2 GB)** RAM for smooth performance.

- Click **Next.**

Step 4: Create Virtual Hard Disk

1. Choose **Create a virtual hard disk now → VDI (VirtualBox Disk Image)**.
2. Choose **Dynamically allocated.**
3. Set disk size: **20 GB**.
4. Click **Create.**

Step 5: Attach ISO File

- Select your VM → Click **Settings → Storage.**
- Under **Controller: IDE**, click **Empty → Choose a disk file.**
- Browse and select the **ISO file** (e.g., ubuntu-22.04.iso).

Step 6: Start and Install OS

1. Click **Start.**
2. The virtual machine boots from the ISO file.
3. Follow OS installation steps (language, region, username, password).
4. Once completed, the OS will boot into desktop mode.

Your Virtual Machine is now ready to use.

Step 7: Test Virtual Machine

- Open terminal or browser inside VM.
- Check internet connection and software installation.
- You can also take snapshots and clone VMs from VirtualBox.

Result:

Successfully created and configured a **Virtual Machine** using **Oracle VirtualBox** to install and run an operating system virtually.

Q2. Working of Google Drive to Make Spreadsheet (For 10 Students' Record)

[20 Marks]

Aim:

To create and maintain a **student record spreadsheet** for 10 students using **Google Drive (Google Sheets)**.

Theory:

Google Drive provides cloud storage and tools like **Google Sheets** for real-time data management.
Google Sheets can be used to maintain student details, marks, and generate total and average automatically using formulas.

Steps to Implement:

Step 1: Open Google Drive

- Go to https://drive.google.com.
- Click **New → Google Sheets.**
- Rename the file: Student_Record_Sem1.

Step 2: Create the Table

Enter the following columns:

Roll No Name Subject 1 Subject 2 Subject 3 Total Average Grade

Step 3: Enter Data for 10 Students

Example:

Roll No	Name	Sub1	Sub2	Sub3	Total	Average	Grade
1	Riya Patil	85	78	90			
2	Neha Shah	70	65	80			
3	Raj Mehta	88	92	85			
4	Aarav Joshi	75	80	70			
5	Priya Kale	95	94	90			
6	Omkar Jadhav	60	65	58			
7	Sneha More	72	78	80			
8	Tushar Deshmukh	88	85	80			
9	Kavita Shinde	92	90	96			
10	Aditya Pawar	65	70	68			

Step 4: Apply Formulas

- **Total Marks:**
- =SUM(C2:E2)
- **Average:**
- =AVERAGE(C2:E2)
- **Grade (Using IF Formula):**
- =IF(G2>=85,"A",IF(G2>=70,"B","C"))

Copy these formulas down for all 10 rows.

Step 5: Formatting & Sharing

- Apply **bold headers, borders, and colors** for clarity.
- Click **Share → Get Link → Anyone with link (View only)** to share with teacher.

The spreadsheet is now saved automatically in Google Drive.

Result:

Successfully created a **Google Spreadsheet** containing 10 students' records with automatic **Total, Average, and Grade** calculation using cloud-based tools.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 10

Q1. Workings of Google Drive to Create Exam Form and Generate Result

[10 Marks]

Theory:

Google Drive offers integrated tools like **Google Forms** and **Google Sheets** for data collection and processing in real time.
Using **Quiz Mode** in Google Forms, we can automatically evaluate answers and generate marks instantly.

Tools Used:

- **Google Forms** – For exam question creation.
- **Google Sheets** – For storing and calculating results.
- **Google Drive** – For saving and sharing data.

Steps to Implement:

Step 1: Create the Exam Form

1. Open https://forms.google.com.
2. Click **Blank Form → Title:** "Online Test – Cloud Computing."
3. Add the following fields:
    - Name
    - Roll Number
    - Email ID
4. Add MCQ Questions (choose "Multiple Choice" type).
5. Go to **Settings → Make this a quiz → Assign points** to each question.
6. Choose whether to show correct answers immediately or later.

Step 2: Collect Responses

- Share the form link with students.
- Under **Responses → Link to Sheets**, connect the form to a Google Sheet.
- Each response is automatically saved in the Sheet.

Example:

Timestamp	Name	Roll No	Q1	Q2	Q3	Score
11/11/2025	Riya Patil	12	A	C	B	8

Step 3: Generate Result

1. Use formula for **total score** (if not auto-calculated):
2. =SUM(D2:F2)
3. Assign grades using conditional formula:
4. =IF(G2>=8,"A",IF(G2>=5,"B","C"))
5. Format Sheet → Add colors or conditional formatting for better visibility.

Step 4: Share Results

- Go to **Share → Copy Link → Anyone with link (View only)**.
- Optionally, use Google Script to email scores automatically to each student.

This demonstrates working of Google Drive in creating, managing, and generating online exam results.

Result:

Successfully created and implemented an **Online Exam Form** and **Automated Result Sheet** using **Google Forms** and **Google Sheets** integrated with Google Drive.

Q2. Workings of Google Drive to Make Spreadsheet (Purchase Order for Grocery Items)

[20 Marks]

Aim:

To prepare a **Purchase Order Spreadsheet** for grocery items using **Google Sheets**, demonstrating cloud-based data entry, total calculation, and sharing.

Theory:

Google Sheets is a spreadsheet application that allows users to store, organize, and analyze data online.
It provides real-time collaboration and automatic saving in **Google Drive**, making it ideal for purchase orders, billing, and record maintenance.

Steps to Implement:

Step 1: Create Google Sheet

1. Open https://drive.google.com.
2. Click **New → Google Sheets.**
3. Rename the file: Grocery_Purchase_Order.

Step 2: Create the Table

Sr. No	Item Name	Quantity (kg/pcs)	Rate (₹)	Amount (₹)
1	Rice	5	60	
2	Wheat	10	45	
3	Sugar	3	50	
4	Oil	2	140	
5	Tea Powder	1	120	
6	Salt	2	20	
7	Soap	4	25	
8	Toothpaste	2	90	
9	Milk Packet	6	45	
10	Biscuits	5	30	

Step 3: Apply Formulas

- **Amount Calculation:**
- =C2*D2

(Then drag down to apply to all rows.)

- **Total Purchase Amount:**
- =SUM(E2:E11)

This automatically calculates total cost.

Step 4: Formatting

- Apply **borders, bold headers, and color formatting**.
- Use **Currency Format (₹)** for Rate and Amount columns.

- Add **company name or date** at the top (e.g., *"Smart Grocery – Purchase Order, Nov 2025"*).

Step 5: Share the Sheet

- Click **Share → Get Link → Anyone with link (View only)**.
- You can also allow collaborators (e.g., supplier or manager) to edit it.

This demonstrates real-time collaboration and cloud storage.

Result:

Successfully created a **Purchase Order Spreadsheet** using **Google Sheets** on **Google Drive** with automatic calculations and sharing functionality.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 11

Q1. Practical Implementation of Storage as a Service (Using Google Drive)

[10 Marks]

Theory:

Storage as a Service (STaaS) is a **cloud computing model** that allows users to store, manage, and share data on remote servers via the internet.
The service provider (e.g., Google) maintains the infrastructure, while users can access data anytime, anywhere.

Examples of STaaS Providers:

- Google Drive
- Dropbox
- OneDrive
- AWS S3

Features of Google Drive:

- Free storage up to 15 GB
- Real-time synchronization and sharing
- Integration with Google Docs, Sheets, Forms, etc.
- Version control and collaborative editing

Steps to Implement:

Step 1: Access Google Drive

- Go to https://drive.google.com.
- Sign in using your Gmail account.

Step 2: Upload Files

1. Click **New → File Upload**.
2. Choose files such as invoice.pdf, notes.docx, or photo.jpg.
3. Files are instantly uploaded to the cloud.

These files are now stored safely on Google's servers.

Step 3: Create Folders

1. Click **New → Folder → Name it "Cloud_Storage_Demo."**
2. Move uploaded files into the folder for better organization.

Step 4: Share Files or Folders

1. Right-click a file → Click **Share.**
2. Add email IDs or choose **"Anyone with the link"** → select **Viewer/Editor** permissions.
3. Copy the shareable link for access.

Demonstrates file sharing and permission control.

Step 5: Access Files Anywhere

- Log in from another device → same files and folders are available.
- You can also download, preview, or edit files directly online.

Advantages of Using Google Drive (STaaS):

- Data is stored safely in the cloud.
- Accessible from any device.
- Easy to share and collaborate.
- No need for external storage devices.

Result:

Successfully implemented **Storage as a Service** using **Google Drive** for uploading, organizing, and sharing files securely over the cloud.

Q2. Workings of Google Drive to Make Spreadsheet (Billing Invoice for Grocery Items)

[20 Marks]

Theory:

Google Sheets allows cloud-based data entry and automatic calculations using built-in formulas.
It is used to maintain billing records for businesses and calculate totals, GST, and final amounts easily.

Steps to Implement:

Step 1: Open Google Sheets

- Go to https://sheets.google.com.

- Click **Blank Sheet → Rename as "Grocery_Billing_Invoice."**

Step 2: Design the Billing Table

Sr. No	Item Name	Quantity	Rate (₹)	Amount (₹)
1	Rice	5 kg	60	
2	Sugar	2 kg	50	
3	Oil	1 L	140	
4	Tea Powder	1 pkt	120	
5	Salt	2 pkt	20	
6	Soap	4 pcs	25	
7	Biscuits	5 pkt	30	

Step 3: Apply Formulas

- **Amount Calculation (for each item):**
- =C2*D2
- **Total Amount:**
- =SUM(E2:E8)
- **GST Calculation (5%):**
- =E9*0.05
- **Final Bill:**
- =E9+E10

This automatically computes total, tax, and final bill amount.

Step 4: Add Headers and Formatting

- Add title: **"Smart Grocery Billing Invoice"**
- Merge top cells, center-align, and bold headers.
- Use ₹ currency format for price columns.
- Apply **borders and background colors** for neat presentation.

Step 5: Save and Share

- File is automatically saved to **Google Drive.**
- Click **Share → Get link → Anyone with the link (View only)** to share the invoice.

Demonstrates creation and sharing of a billing invoice using cloud storage.

Example Output:

Item	**Qty**	**Rate**	**Amount**
Rice	5	60	300
Sugar	2	50	100
Oil	1	140	140
Tea	1	120	120
Salt	2	20	40
Total			**700**
GST (5%)			**35**
Grand Total			**735**

Result:

Successfully created a **Billing Invoice Spreadsheet** using **Google Sheets**, demonstrating storage, automatic calculation, and cloud sharing via **Google Drive**.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 12

Q1. Demonstrate How to Manage Cloud Computing Resources

[10 Marks]

Theory:

Cloud Resource Management is the process of efficiently using and controlling computing resources (CPU, RAM, storage, and network) provided by a cloud service.
It ensures cost-efficiency, scalability, and proper allocation of resources according to demand.

Key Cloud Resources:

1. **Compute Resources** – Virtual Machines (e.g., AWS EC2, GCP VM Instances)
2. **Storage Resources** – Cloud storage buckets or drives
3. **Network Resources** – IPs, load balancers, firewalls
4. **User Management** – IAM (Identity and Access Management)

Steps to Demonstrate (Using Google Cloud Platform)

Step 1: Login to Google Cloud Console

- Visit https://console.cloud.google.com.
- Sign in using your Google account.

Step 2: Manage Compute Resources

1. Open **Compute Engine → VM Instances.**
2. Click **Create Instance.**
3. Configure:
    - Name: demo-vm
    - Region: asia-south1 (Mumbai)
    - Machine type: e2-micro
    - Boot disk: Ubuntu 22.04 LTS
4. Click **Create.**
   ✅ *VM created successfully.*

Step 3: Manage Storage Resources

1. Go to **Cloud Storage → Create Bucket.**
2. Set name: student-storage-bucket.
3. Choose default region and storage class (Standard).
4. Upload a sample file (e.g., notes.txt).
   ✅ *File is uploaded and stored securely.*

Step 4: Manage IAM (Access Control)

1. Go to **IAM & Admin → IAM.**
2. Click **Add Member → Enter Email → Assign Role:** Viewer / Editor.
   ✅ *This ensures resource-level permission control.*

Step 5: Monitor and Optimize Resources

- Open **Monitoring Dashboard** → view CPU usage and network load.
- Stop unused VMs to save billing cost.
  ✅ *Demonstrates efficient management of resources.*

Advantages of Cloud Resource Management:

- Cost optimization
- Resource scalability
- Centralized control
- High availability and performance monitoring

Result:

Successfully demonstrated how to **create, manage, and monitor cloud computing resources** using Google Cloud Platform, including virtual machines, storage, and IAM permissions.

Q2. Developing Python Application (Average of 5 Numbers) Using Google App Engine

[20 Marks]

Steps to Implement:

Step 1: Setup Environment

1. Create a Google Cloud project from https://console.cloud.google.com.
2. Enable **App Engine API.**
3. Install **Google Cloud SDK** on your computer.

Step 2: Create Project Folder

Folder structure:

```
average-app/
|
 ├── main.py
└── app.yaml
```

Step 3: Write Python Code (main.py)

```
from flask import Flask, request

app = Flask(__name__)
@app.route('/')
def home():
    return '''
    <h2>Average Calculator</h2>
    <form action="/result" method="get">
        Enter 5 numbers separated by commas:
        <input type="text" name="nums" placeholder="e.g. 10,20,30,40,50">
        <input type="submit" value="Calculate Average">
    </form>
    '''

@app.route('/result')
def result():
    try:
        numbers = [float(x) for x in request.args.get('nums').split(',')]
        if len(numbers) != 5:
            return "<h3>Please enter exactly 5 numbers!</h3>"
        avg = sum(numbers) / 5
        return f"<h3>Average of {numbers} is: {avg:.2f}</h3>"
```

```
    except:
        return "<h3>Invalid Input! Please enter numbers only.</h3>"


if __name__ == "__main__":
    app.run()
```

Step 4: Create Configuration File (app.yaml)

```
runtime: python39
entrypoint: gunicorn -b :$PORT main:app
```

Step 5: Deploy to Google App Engine

Run the following commands in terminal:

```
gcloud init
gcloud app deploy
gcloud app browse
```

Your app will be hosted at:

https://average-app.appspot.com

This web app calculates and displays the average of 5 user-input numbers.

Sample Output:

Input: 10, 20, 30, 40, 50
Output: Average of [10, 20, 30, 40, 50] is: 30.00

Result:

Successfully developed and deployed a **Python Average Calculator Application** on **Google App Engine**, demonstrating a **PaaS implementation**.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 13

Q1. Write a Program for Web Feed

[10 Marks]

Theory:

A Web Feed is a format (RSS or Atom) that allows websites to automatically distribute their latest updates to subscribers.

- RSS (Really Simple Syndication) provides XML-based updates.
- It is commonly used in blogs, podcasts, and news sites.

Users can subscribe to the feed using feed readers (e.g., Feedly, Inoreader).

Algorithm:

1. Import the required library (feedgen).
2. Create a feed object with metadata (title, author, description).
3. Add multiple entries (articles or posts).
4. Save the feed to an XML file (RSS format).

Program (Python)

```
from feedgen.feed import FeedGenerator


# Create Feed Object
fg = FeedGenerator()
fg.title('Cloud Computing Updates')
fg.link(href='https://mycloudupdates.com', rel='alternate')
fg.description('Latest updates and tutorials on Cloud Computing')
fg.language('en')


# Add Feed Entries
for i in range(1, 4):
    fe = fg.add_entry()
```

```
    fe.title(f'Cloud Topic {i}')
    fe.link(href=f'https://mycloudupdates.com/topic{i}')
    fe.description(f'Detailed article about Cloud Topic {i}')

# Generate and Save Feed
fg.rss_file('webfeed.xml')
print("Web Feed created successfully as webfeed.xml")
```

Output:

Web Feed created successfully as webfeed.xml

Generated File (webfeed.xml):

```
<rss version="2.0">
  <channel>
    <title>Cloud Computing Updates</title>
    <description>Latest updates and tutorials on Cloud Computing</description>
    <link>https://mycloudupdates.com</link>
    <item>
      <title>Cloud Topic 1</title>
      <link>https://mycloudupdates.com/topic1</link>
      <description>Detailed article about Cloud Topic 1</description>
    </item>
  </channel>
</rss>
```

Result:

Successfully created an RSS web feed in XML format that lists recent cloud computing topics.

Q2. Developing Python Application (Maximum of 10 Numbers) Using Google App Engine

[20 Marks]

Aim:

To develop and deploy a Python web application that accepts 10 numbers from the user and displays the maximum value using Google App Engine (GAE).

Theory:

Google App Engine (GAE) is a Platform as a Service (PaaS) that allows developers to run and scale applications easily.
It automatically handles server provisioning, scaling, and monitoring.

Tools Required:

- Google Cloud Account
- Google Cloud SDK
- Python 3.x
- Flask Framework

Steps to Implement:

Step 1: Setup Environment

1. Sign in to Google Cloud Console.
2. Create a new project: max-number-app.
3. Enable App Engine.
4. Install Google Cloud SDK and initialize using gcloud init.

Step 2: Create Project Files

Folder structure:

max-app/

|

├— main.py

└— app.yaml

Step 3: Write Python Code (main.py)

from flask import Flask, request

app = Flask(__name__)

```
@app.route('/')
def home():
    return '''
    <h2>Find Maximum of 10 Numbers</h2>
    <form action="/result" method="get">
        Enter 10 numbers separated by commas:<br>
        <input type="text" name="nums" placeholder="e.g. 5,12,8,20,3,7,9,15,6,10">
        <input type="submit" value="Find Maximum">
    </form>
    '''

@app.route('/result')
def result():
    try:
        numbers = [float(x) for x in request.args.get('nums').split(',')]
        if len(numbers) != 10:
            return "<h3>Please enter exactly 10 numbers!</h3>"
        max_num = max(numbers)
        return f"<h3>The maximum number is: {max_num}</h3>"
    except:
        return "<h3>Invalid Input! Please enter valid numbers only.</h3>"

if __name__ == "__main__":
    app.run()
```

Step 4: Create Configuration File (app.yaml)

```
runtime: python39
entrypoint: gunicorn -b :$PORT main:app
```

Step 5: Deploy on Google App Engine

Commands:

gcloud app deploy

gcloud app browse

App hosted successfully at
https://max-number-app.appspot.com

Sample Output:

Input:
5, 12, 8, 20, 3, 7, 9, 15, 6, 10

Output:
The maximum number is: 20

Result:

Successfully developed and deployed a Python application on Google App Engine that finds the maximum of 10 numbers.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 14

Q1. Practical Implementation of Storage as a Service (using Google Drive)

[10 Marks]

Theory:

Storage as a Service (STaaS) is a cloud computing model that allows users to store and manage data on the cloud instead of local storage.
Examples include Google Drive, Dropbox, AWS S3, and OneDrive.

Key Features of STaaS:

- Online storage & file synchronization
- File sharing & access control
- Data backup and versioning
- Accessibility from any device

Steps to Implement using Google Drive:

Step 1: Login

Open https://drive.google.com and sign in with your Google account.

Step 2: Create a Folder

- Click “+ New” → Folder
- Name it as “Cloud_Storage_Practical”

Step 3: Upload Files

- Click “+ New” → File Upload
- Select any files (e.g., report.docx, data.xlsx, image.png)

Step 4: Organize Files

- Create subfolders like Documents, Images, Reports
- Move files accordingly.

Step 5: Share File or Folder

- Right-click on the folder → Share
- Add email addresses or generate a shareable link

- Set permission:
    - Viewer
    - Commenter
    - Editor

Step 6: Access from Another Device

- Login to Google Drive on another device to verify cloud synchronization.

Result:

Successfully demonstrated Storage as a Service using Google Drive, including file upload, organization, sharing, and access control.

Q2. Write a Program for Web Feed

[20 Marks]

Aim:

To create a Python program that generates a web feed (RSS format) for automatically publishing website updates.

Theory:

A Web Feed allows automatic delivery of the latest updates (news, blog posts, or articles) from a website.
Two popular feed formats are:

- RSS (Really Simple Syndication)
- Atom

These feeds are written in XML format and can be read by feed readers.

Algorithm:

1. Import the feedgen library.
2. Create a feed object with metadata like title, link, and description.
3. Add multiple entries to the feed (like articles).
4. Save the output as an RSS file.

Program (Python)

```
from feedgen.feed import FeedGenerator
```

```
# Create Feed Object
fg = FeedGenerator()
fg.title('My Cloud Feed')
fg.link(href='https://cloudnewsportal.com', rel='alternate')
fg.description('Latest Cloud Technology Updates and Articles')
fg.language('en')

# Add Feed Entries
for i in range(1, 4):
    fe = fg.add_entry()
    fe.title(f'Cloud Computing Topic {i}')
    fe.link(href=f'https://cloudnewsportal.com/topic{i}')
    fe.description(f'Complete guide about Cloud Topic {i}')

# Generate and Save Feed
fg.rss_file('webfeed.xml')
print("RSS Web Feed generated successfully!")
```

Output:

RSS Web Feed generated successfully!

Generated XML File (webfeed.xml):

```
<rss version="2.0">
  <channel>
    <title>My Cloud Feed</title>
    <link>https://cloudnewsportal.com</link>
    <description>Latest Cloud Technology Updates and Articles</description>
    <language>en</language>
    <item>
      <title>Cloud Computing Topic 1</title>
```

```
    <link>https://cloudnewsportal.com/topic1</link>
    <description>Complete guide about Cloud Topic 1</description>
  </item>
 </channel>
</rss>
```

Result:

Successfully created a Web Feed (RSS file) that automatically lists multiple articles or updates in XML format.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 15

Q1. Working of Software as a Service (Amazon)

[10 Marks]

Theory:

Software as a Service (SaaS) is a cloud service model that delivers software applications over the internet.
Instead of installing and maintaining software locally, users access it via a web browser.

Examples of SaaS in Amazon:

1. Amazon WorkMail – Cloud-based business email and calendar service.
2. Amazon WorkDocs – Online document storage and collaboration platform.
3. Amazon Chime – Online meeting and communication tool.
4. Amazon Connect – Cloud contact center service.

Working of SaaS (Amazon Example):

1. User Access:
    - Users log in through a web browser (e.g., WorkDocs at https://workdocs.aws).
2. Service Hosting:
    - Software runs on AWS-managed infrastructure.
3. Subscription-Based Model:
    - Users pay for usage per month (per user).
4. Data Storage:
    - All user files, emails, and settings are stored in AWS Cloud.
5. Automatic Updates:
    - The software is updated automatically by Amazon, no manual installation required.
6. Access Anywhere:
    - Accessible from mobile, tablet, or desktop globally.

Advantages:

- No installation required
- Cost-effective and scalable
- Automatic updates and data backup
- Accessible from any device

Result:

Successfully studied and understood the working of Software as a Service (SaaS) using Amazon cloud platforms such as WorkMail, WorkDocs, and Chime.

Q2. Practical Implementation of File Sharing and Storage as a Service

[20 Marks]

Aim:

To practically implement File Sharing and Storage as a Service using a cloud storage provider such as Google Drive or AWS S3.

Theory:

Storage as a Service (STaaS) allows users to upload, manage, and share files online.
File Sharing provides secure access to files for multiple users.

Popular providers:

- Google Drive
- AWS S3 (Amazon Simple Storage Service)
- Dropbox

Implementation Steps (Using AWS S3):

Step 1: Login

- Sign in to AWS Management Console → https://aws.amazon.com

Step 2: Create a Storage Bucket

1. Go to Services → S3 → Create Bucket
2. Bucket name: file-sharing-demo
3. Region: Asia Pacific (Mumbai)
4. Keep “Block Public Access” ON initially (for safety).

5. Click Create Bucket

✅ *S3 bucket created successfully.*

Step 3: Upload Files

1. Open your bucket → Click Upload
2. Select files (e.g., report.pdf, project.docx, data.xlsx)
3. Click Upload

✅ *Files are now stored in AWS cloud.*

Step 4: Manage Access Permissions

1. Select a file → Permissions Tab
2. Choose “Grant public read access” (or share with specific users).
3. Copy the Object URL — users can access the file using this link.

✅ *File sharing implemented.*

Step 5: Organize Files

- Create folders for different types (e.g., “Reports”, “Images”).
- Move uploaded files accordingly.

Step 6: Verify Access

- Open the shared URL in another browser/device → file should open.
  ✅ *Demonstrates cloud-based file sharing successfully.*

Implementation Steps (Alternate using Google Drive):

1. Go to https://drive.google.com.
2. Upload files → Right-click → Share.
3. Generate a link and set permission (Viewer/Editor).
4. Open link from another account → Verify access.

Result:

Successfully implemented File Sharing and Storage as a Service using AWS S3 / Google Drive.
Users can store, manage, and share files securely over the internet.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 16

Q1. Show the Working and Implementation of Identity Management

[10 Marks]

Theory:

Identity Management (IdM) in cloud computing is the process of authenticating and authorizing users to access cloud resources securely.
It ensures that only authorized individuals or services can access specific data or functions.

AWS IAM (Identity and Access Management) helps manage:

- Users
- Groups
- Roles
- Permissions

Working of Identity Management:

1. Authentication – Verifies the user's identity (username & password).
2. Authorization – Grants permission to access resources.
3. User Roles and Policies – Defines what actions are allowed.
4. Access Keys and MFA – Adds secure login and access control.

Steps to Implement (Using AWS IAM):

Step 1: Login

- Go to AWS Management Console → https://aws.amazon.com
- Navigate to IAM (Identity and Access Management) service.

Step 2: Create a New IAM User

1. Click on Users → Add user.
2. Enter username: cloud_user.
3. Select Access type:
    - ✅ AWS Management Console Access
    - ✅ Programmatic Access

4. Set a password → Click Next.

Step 3: Assign Permissions

1. Choose Attach existing policies directly.
2. Select AmazonS3ReadOnlyAccess policy.
    - This allows the user to *view* but not *delete or modify* S3 objects.
      ✅ *Implements Role-Based Access Control (RBAC).*

Step 4: Create and Review User

- Click Create user.
- Note down login URL and user credentials.
  ✅ *User created successfully.*

Step 5: Verify Access

1. Login using new IAM credentials (cloud_user).
2. Try accessing AWS S3 → user can *view* but not *delete* files.

✅ *Identity and access control successfully demonstrated.*

Advantages of Identity Management:

- Secure access to cloud resources
- Prevents unauthorized activities
- Enables multi-user collaboration
- Supports auditing and compliance

Result:

Successfully implemented Identity Management using AWS IAM by creating users, assigning roles, and verifying access permissions.

Q2. Create and Host Static Web Page using Any Cloud Provider

Theory:

A static website consists of fixed content such as HTML, CSS, and images.
Cloud providers offer object storage services that can host static sites easily.

Examples:

- AWS S3 (Amazon Simple Storage Service)
- Google Cloud Storage

- Microsoft Azure Blob Storage

Steps to Implement (Using AWS S3):

Step 1: Create an HTML Page

Create a file named index.html with the following code:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Cloud Static Website</title>
</head>
<body style="background-color:#eaf3ff; text-align:center;">
  <h1>Welcome to My Static Web Page</h1>
  <h2>Hosted on AWS Cloud</h2>
  <p>This is a demo of cloud web hosting service.</p>
</body>
</html>
```

Step 2: Create an S3 Bucket

1. Go to AWS Console → S3 Service.
2. Click Create bucket.
    - Bucket name: my-static-web-demo
    - Region: Asia Pacific (Mumbai)
3. Uncheck “Block all public access.”
4. Click Create bucket.

Step 3: Upload the HTML File

1. Open your bucket → Click Upload.
2. Select the file index.html → Upload.

Step 4: Enable Static Website Hosting

1. Go to Properties → Static Website Hosting.
2. Choose “Enable” and set:
    - Index document: index.html

3. Save changes.

Step 5: Set Bucket Policy (to Make Public)

Go to Permissions → Bucket Policy and paste:

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "PublicReadGetObject",
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::my-static-web-demo/*"
  }]
}
```

Makes files publicly readable.

Step 6: Access Website

Copy the Bucket Website Endpoint URL, for example:
http://my-static-web-demo.s3-website-ap-south-1.amazonaws.com

Your static webpage is live and accessible globally.

Result:

Successfully created and hosted a static website using AWS S3 cloud storage.
The webpage displays a welcome message when accessed through the public URL.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 17

Q1. Practical Implementation of Storage as a Service (using OwnCloud)

[10 Marks]

Theory:

Storage as a Service (STaaS) is a cloud computing model that provides scalable storage to users over the internet.
Users can store, manage, and share data easily using web-based interfaces.

OwnCloud is an open-source cloud storage platform that allows users to:

- Store data on their private server
- Share and synchronize files
- Access files from anywhere securely

Requirements:

- Operating System: Ubuntu / Windows
- Software: **OwnCloud Server**, **XAMPP / Apache**, **Web Browser**

Steps to Implement:

Step 1: Install and Setup OwnCloud

1. Download OwnCloud from the official website: https://owncloud.org/download/
2. Install **XAMPP** or **Apache Server** on your system.
3. Extract the OwnCloud folder inside:
   C:\xampp\htdocs\owncloud (Windows)
   or /var/www/html/owncloud (Linux).
4. Start **Apache** and **MySQL** from XAMPP control panel.

Step 2: Create a Database

1. Open browser → visit: http://localhost/phpmyadmin
2. Click **New → Create Database** → name it owncloud_db

Step 3: Configure OwnCloud

1. Visit: http://localhost/owncloud
2. Enter details:
    - Username: admin
    - Password: 12345
    - Data folder: /owncloud/data
3. Under Database configuration:
    - Database user: root
    - Database name: owncloud_db
4. Click **Finish Setup**

OwnCloud dashboard appears.

Step 4: Upload and Manage Files

1. Click **+ → Upload file** → choose any document or image.
2. Create folders to organize files (e.g., *Projects*, *Reports*).
3. Click on any file → **Share** → generate a public or password-protected link.

Demonstrates file storage and sharing.

Step 5: Access from Another Device

- Open the shared link in another browser or mobile device.
  Files can be accessed remotely — STaaS successfully implemented.

Result:

Successfully implemented **Storage as a Service (STaaS)** using **OwnCloud**, demonstrating secure file storage, synchronization, and sharing.

Advantages:

- Secure private cloud storage
- Cross-platform access

- Data synchronization
- Access control and sharing

Q2. Show the Implementation of Cloud Single Sign-On (SSO)

[20 Marks]

Aim:

To demonstrate the **implementation of Single Sign-On (SSO)** in cloud computing for unified user authentication across multiple cloud applications.

Theory:

Single Sign-On (SSO) is an authentication process that allows a user to **log in once** and gain access to multiple applications without re-entering credentials.

Example:
When you sign in to your **Google Account**, you can access **Gmail, Drive, YouTube, and Calendar** without logging in again.

Working of SSO:

1. **User Authentication:**
   The user logs in once via a central Identity Provider (IdP).
2. **Token Generation:**
   IdP generates an authentication token.
3. **Service Access:**
   The token is shared securely across cloud applications.
4. **Session Management:**
   The user can switch between apps (e.g., Gmail, Drive) without repeated login.

Common Protocols Used in SSO:

- **SAML (Security Assertion Markup Language)**
- **OAuth 2.0**
- **OpenID Connect**

Implementation Example: Google Cloud Identity (SSO)

Step 1: Login to Google Admin Console

- Go to https://admin.google.com
- Sign in as administrator.

Step 2: Enable SSO

1. Navigate to **Security → Authentication → SSO Settings**.
2. Enable **Single Sign-On** with a third-party Identity Provider.
3. Provide required URLs:
    - SSO URL (IdP Login)
    - Entity ID
    - Certificate (for authentication)

Step 3: Integrate Cloud Apps

1. Add multiple Google Workspace apps (Gmail, Docs, Drive).
2. All apps will now use the **same login session**.

Step 4: Test SSO

1. Log in with your Google account once.
2. Access Gmail, Google Drive, YouTube, etc.
3. Notice — no need to log in again!

Single Sign-On successfully implemented.

Alternate Example: AWS SSO

1. Go to **AWS Console → IAM Identity Center (SSO)**.
2. Add users and connect multiple AWS accounts.
3. Users log in once to access all AWS services under one session.

Result:

Successfully demonstrated **Single Sign-On (SSO)** in cloud computing, showing how users can access multiple services with a single authentication process.

Advantages of SSO:

- Simplified user experience
- Improved security and centralized control
- Reduces password fatigue

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 18

Q1. Create Virtual Machine and Perform Basic Shell Commands

[10 Marks]

Aim:

To create a **Virtual Machine (VM)** using a cloud platform (AWS EC2 / VirtualBox) and execute basic **Linux shell commands** inside the VM.

Theory:

A **Virtual Machine (VM)** is a software-based emulation of a physical computer.
It allows users to run multiple operating systems on a single host machine using **virtualization technology**.

In cloud computing, VMs are used to create isolated computing environments for deploying applications securely.

Examples of VM providers:

- AWS EC2
- Google Compute Engine
- Microsoft Azure VM
- Oracle VirtualBox (for local setup)

Requirements:

- **Cloud Provider:** AWS / VirtualBox
- **OS:** Ubuntu / CentOS Linux
- **Tools:** SSH client or Terminal

Steps (Using AWS EC2):

Step 1: Login to AWS Management Console

1. Open https://aws.amazon.com
2. Navigate to **EC2 → Instances → Launch Instance**

Step 2: Configure VM

- Name: MyLinuxVM
- AMI: Ubuntu Server 22.04 LTS

- Instance Type: t2.micro (Free tier)
- Key Pair: Create or use existing key pair
- Network: Default VPC
- Storage: 8GB EBS volume

Click **Launch Instance** ✅

Step 3: Connect to the VM

1. In EC2 dashboard → Select instance → Click **Connect**
2. Copy SSH command (example):

ssh -i "mykey.pem" ubuntu@ec2-13-127-22-45.ap-south-1.compute.amazonaws.com

3. Press **Enter** — you are now inside the VM shell.

Step 4: Execute Basic Shell Commands

Command	**Description**
pwd	Print working directory
ls	List files and directories
mkdir testfolder	Create a directory
cd testfolder	Change directory
touch file1.txt	Create a new file
echo "Hello Cloud" > file1.txt	Write data to a file
cat file1.txt	Display file content
rm file1.txt	Delete a file
sudo apt update	Update packages
uname -a	Display system information

Output Example:

ubuntu@ip-172-31-23-11:~$ pwd

/home/ubuntu

ubuntu@ip-172-31-23-11:~$ mkdir testfolder

ubuntu@ip-172-31-23-11:~$ cd testfolder

ubuntu@ip-172-31-23-11:~/testfolder$ echo "Hello Cloud" > demo.txt

ubuntu@ip-172-31-23-11:~/testfolder$ cat demo.txt

Hello Cloud

VM created and shell commands executed successfully.

Result:

Successfully created a **Virtual Machine** and performed basic Linux shell operations like directory creation, file handling, and system info retrieval.

Q2. Practical Implementation of Cloud Security

[20 Marks]

Aim:

To implement and understand the key **cloud security mechanisms** such as **Identity & Access Management (IAM)**, **encryption**, and **secure data sharing**.

Theory:

Cloud Security ensures protection of data, applications, and services hosted in the cloud. It includes measures such as **authentication, encryption, firewall rules, and access control.**

Key Cloud Security Components:

1. **Authentication & Authorization** – Ensures only authorized users access resources.
2. **Encryption** – Protects data at rest and in transit.
3. **Network Security** – Controls inbound/outbound traffic using firewalls or security groups.
4. **Monitoring** – Tracks and logs activities for auditing.

Steps (AWS Example):

Step 1: Configure IAM (Identity & Access Management)

1. Go to AWS Console → **IAM → Users → Add User**
2. Set user name: test_user
3. Assign **programmatic access** and **console access**.
4. Attach policy: AmazonS3ReadOnlyAccess
   ✅ *Only allows reading S3 data.*

Step 2: Create S3 Bucket for Secure Storage

1. Go to **S3 → Create Bucket** (e.g., secure-bucket-demo).
2. Enable **Bucket Versioning** and **Default Encryption (AES-256)**.
3. Upload a test file (e.g., data.txt).

Step 3: Apply Access Control

- Navigate to **Permissions tab** → set access to **private**.
- Only the IAM user with AmazonS3ReadOnlyAccess can view the file.

Step 4: Enable Monitoring

- Go to **CloudTrail** → Enable for all AWS activities.
- Logs every user action for auditing.

Step 5: Test Security Setup

1. Log in as test_user.
2. Try to delete the file → Access Denied .
3. Try to read file → Access Granted.

Result:

Successfully implemented **cloud security** using IAM roles, encryption, and monitoring, ensuring data protection and access control.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 19

Q1. Practical Implementation of File Sharing and Storage as a Service (STaaS)

[10 Marks]

Theory:

Storage as a Service (STaaS) is a cloud computing model where users can **store and share data** online instead of local systems.
Cloud providers manage the storage infrastructure while users manage access and permissions.

Example Services:

- Google Drive
- Dropbox
- Amazon S3
- OwnCloud

Steps (Example: Using Google Drive)

Step 1 – Login to Google Drive

- Open **https://drive.google.com**
- Sign in with your Google account.

Step 2 – Create Folder for Storage

- Click **New → Folder**
- Name it **"CloudStorageDemo"**
- Open the folder.

Step 3 – Upload Files

- Click **New → File Upload**
- Upload some sample documents (e.g., Report.pdf, Image1.jpg, Notes.txt).

Step 4 – Share the Files

- Right-click on a file → **Share**.

- Choose access level:
    - *Viewer* (read-only)
    - *Editor* (full access)
    - *Commenter* (limited access).
- Copy the generated shareable link.

Step 5 – Access from Another Device

- Open the shareable link in another browser.

File opens successfully — shows **file sharing and storage** implemented.

Alternate (AWS S3 Example):

1. Login to AWS → S3 → *Create Bucket*.
2. Upload file → Enable *Public access*.
3. Generate *Object URL*.

Access file using the URL — secure storage and sharing verified.

Result:

Successfully implemented **File Sharing and Storage as a Service (STaaS)** using a cloud platform by uploading, accessing, and sharing files securely.

Q2. Create Virtual Machine using VirtualBox

[20 Marks]

Aim:

To create and configure a **Virtual Machine (VM)** using **Oracle VirtualBox**.

Theory:

A **Virtual Machine (VM)** is software that emulates a physical computer.
It allows running multiple operating systems on one physical host using **virtualization technology**.
VirtualBox is an open-source virtualization tool by Oracle.

Requirements:

- Oracle VirtualBox
- ISO Image (Ubuntu / Windows / Fedora)

Steps:

Step 1 – Install VirtualBox

1. Download from https://www.virtualbox.org
2. Install it on your system.

Step 2 – Create New Virtual Machine

1. Open VirtualBox → Click **New**.
2. Enter details:
    - Name: UbuntuVM
    - Type: *Linux*
    - Version: *Ubuntu (64-bit)*
3. Allocate **2 GB RAM** and **20 GB hard disk (VDI format)**.

Step 3 – Attach OS Image

- Under *Storage*, select **Empty CD icon → Choose a disk file → Browse Ubuntu ISO**.
- Click **OK**.

Step 4 – Start the VM

- Click **Start**.
- The Ubuntu installer will boot.
- Proceed with installation → Set username and password.

Step 5 – Access the VM

- Once installed, the VM starts normally.
- Try a few **basic commands** in terminal:

pwd

ls

mkdir demo

echo "Hello Cloud" > demo/test.txt

cat demo/test.txt

VM created and working successfully.

Result:

Successfully created a **Virtual Machine using VirtualBox** and performed basic operations inside the VM environment.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 20

Q1. Create Virtual Machine and Perform Basic Shell Commands

[10 Marks]

Theory:

A **Virtual Machine** is a software-based emulation of a computer system.
It runs an operating system and applications like a physical computer but inside a virtualized environment.

Common VM Platforms:

- Oracle VirtualBox
- VMware Workstation
- AWS EC2
- Microsoft Azure VM

Steps (Using Oracle VirtualBox):

Step 1 – Install VirtualBox

1. Download from https://www.virtualbox.org
2. Install it on your system.

Step 2 – Create New VM

1. Open VirtualBox → Click **New**.
2. Name: Ubuntu_VM
3. Type: **Linux**, Version: **Ubuntu (64-bit)**
4. Allocate:
    - Memory: 2 GB
    - Disk: 20 GB (VDI format)

Step 3 – Mount ISO

- Click **Settings → Storage → Empty Disk → Choose a disk file → Select Ubuntu ISO**.

Step 4 – Start and Install OS

- Click **Start** → Ubuntu setup begins.

- Follow installation steps and log in after setup completes.

Step 5 – Run Basic Shell Commands

Command	Description
pwd	Shows current directory
ls	Lists files/folders
mkdir CloudLab	Creates new directory
cd CloudLab	Moves to directory
echo "Hello Cloud" > demo.txt	Creates and writes text to file
cat demo.txt	Displays file contents
uname -a	Shows system details

Output Example:

student@ubuntu:~$ mkdir CloudLab

student@ubuntu:~$ cd CloudLab

student@ubuntu:~/CloudLab$ echo "Hello Cloud World" > demo.txt

student@ubuntu:~/CloudLab$ cat demo.txt

Hello Cloud World

Virtual Machine created successfully and basic shell commands executed.

Result:

Successfully created a **Virtual Machine** and performed basic Linux commands for directory, file, and system operations.

Q2. Create Google Form to Accept Student Details, Create Test Page, and Generate Result

[20 Marks]

Aim:

To design a **Google Form** that accepts student details, conducts a test, and automatically generates a result summary.

Theory:

Google Forms is a cloud-based tool under **Software as a Service (SaaS)** model that allows users to:

- Collect data via forms and quizzes,
- Store responses in Google Sheets,
- Automatically evaluate and generate results.

Steps:

Step 1 – Open Google Forms

- Visit https://forms.google.com
- Click **Blank Form**.

Step 2 – Create Student Details Section

Add following fields:

1. **Full Name** – Short answer
2. **Roll Number** – Short answer
3. **Class & Semester** – Dropdown (e.g., M.Sc. CS Sem-I)
4. **Email ID** – Short answer
5. **Contact Number** – Short answer

All fields set to "Required".

Step 3 – Create Test Questions

- Title: **Cloud Computing Test**
  Add 5 questions using **Multiple Choice / Checkbox** format. Example:

1. What is Cloud Computing?
2. Which is an example of IaaS?
3. Which company provides AWS service?
4. Expand SaaS.
5. Google Drive is an example of which service model?

Enable **Answer Key** for automatic grading.
Assign marks for each question (e.g., 2 marks each).

Step 4 – Convert Form into Quiz

1. Click ⚙️ (Settings icon).
2. Go to **Quizzes tab → Turn on "Make this a quiz"**.
3. Choose:
    - Release grade: *Immediately after submission*.
    - Show correct answers & points.

Step 5 – Collect Responses

- Share form link with students.
- Students fill details and complete test.
- Responses automatically saved in Google Sheets.

Step 6 – Generate Result

1. Click **Responses → View in Sheets**.
2. Sheet displays:
    - Student Name
    - Roll No
    - Marks
    - Score Percentage
3. Use formulas (e.g., =AVERAGE(B2:B10)) to calculate overall class performance.

Result:

Successfully created a **Google Form** that collects student data, conducts a quiz, and generates an automated results summary using Google Sheets.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 21

Q1. Workings of Google Drive to Create Exam Form and Generate Result

[10 Marks]

Theory:

Google Drive is a **cloud-based storage and collaboration platform** provided by Google. It allows users to:

- Create, store, and share documents and forms,
- Conduct online tests, and
- Generate and analyze results in real-time.

Google Forms (a part of Google Drive) helps in creating **quizzes and forms**, while **Google Sheets** stores responses and allows data analysis.

Steps:

Step 1 – Open Google Drive

- Visit https://drive.google.com
- Sign in with your Google account.

Step 2 – Create a New Google Form

- Click **New → Google Forms → Blank Form.**

Step 3 – Add Exam Details

Title: *Online Cloud Computing Exam*
Add fields:

1. **Student Name** (Short answer)
2. **Roll Number** (Short answer)
3. **Email ID** (Short answer)

✅ *Set all as required fields.*

Step 4 – Add Test Questions

Add 5 Multiple-Choice Questions:

1. What is Cloud Computing?

2. Which is an example of IaaS?
3. Expand SaaS.
4. AWS is developed by?
5. Which is a cloud storage service?

✅ Enable “Answer Key” for auto-grading (assign marks per question).

Step 5 – Turn Form into a Quiz

1. Click ⚙️ **Settings → Quizzes tab**
2. Turn ON **“Make this a quiz”**
3. Select “Release grade: Immediately after submission.”
   ✅ Students will see marks after submitting.

Step 6 – Collect Responses

- Share the form link via email or WhatsApp.
- Students fill and submit answers.

Step 7 – Generate Result

- Click **Responses → Link to Sheets → Create Spreadsheet**
- A new Google Sheet opens showing:
  - Student Name
  - Roll No
  - Email
  - Marks Scored
  - Total / Percentage

Use Sheet formulas like =AVERAGE() or =MAX() for analysis.

Output Example (Google Sheet):

Name	Roll No	Score	Percentage
Asha	01	8/10	80%
Rohan	02	9/10	90%
Priya	03	10/10	100%

✅ *Exam form created and result generated successfully.*

Result:

Successfully created an **exam form using Google Drive**, collected responses, and generated **automatic result sheet** using Google Sheets.

Q2. Show Practical Implementation of Cloud on Single Sign-On (SSO)

[20 Marks]

Theory:

Single Sign-On (SSO) is a **cloud authentication mechanism** that allows a user to log in once and access multiple related applications without re-entering credentials.

Example:
Logging into a Google account once gives access to Gmail, Drive, YouTube, and Calendar — all under the same session.

Working of SSO:

1. **User logs in** through a central **Identity Provider (IdP)** (like Google or AWS IAM).
2. The IdP generates an **authentication token**.
3. The token is passed to other apps that trust the IdP.
4. The user gains access to all apps without logging in again.

Common SSO Protocols:

- **SAML (Security Assertion Markup Language)**
- **OAuth 2.0**
- **OpenID Connect**

Implementation Example (Using Google Account):

Step 1 – Login to Google

- Go to https://accounts.google.com
- Sign in with one Google account.

Step 2 – Access Other Cloud Apps

Without logging in again, open:

- https://drive.google.com
- https://mail.google.com
- https://calendar.google.com

All services open without asking for login again — showing **SSO functionality**.

Alternate Example (AWS SSO):

1. Login to AWS → **IAM Identity Center (SSO)**.
2. Add user accounts and assign applications.
3. Once user logs in to SSO portal → gains access to all AWS apps (EC2, S3, Lambda) without multiple logins.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 22

Q1. Workings of Google Drive to Make Spreadsheet and Notes

[10 Marks]

Aim:

To create and maintain a **Spreadsheet** and **Notes document** using **Google Drive** as a cloud-based productivity tool.

Theory:

Google Drive is a **cloud-based file storage and synchronization service** that allows users to store, share, and collaborate on documents online.

It integrates tools such as:

- **Google Docs** – for creating text-based notes.
- **Google Sheets** – for spreadsheets and tabular data.
- **Google Slides** – for presentations.

These tools are part of **Software as a Service (SaaS)** under the cloud computing model.

Steps:

Step 1 – Open Google Drive

- Visit https://drive.google.com
- Sign in with your Google account.

Step 2 – Create a Spreadsheet

1. Click **New → Google Sheets → Blank Spreadsheet.**
2. Title: Student_Record_Sheet.
3. Enter sample data:

Roll No	Student Name	Marks	Grade
1	Asha	85	A
2	Rohan	78	B
3	Priya	90	A+
4	Kiran	60	C

4. Use simple formulas:
    - =AVERAGE(C2:C5) → To calculate average marks.
    - =MAX(C2:C5) → To find highest marks.

Shows basic spreadsheet operations.

Step 3 – Create Notes Document

1. In Google Drive → Click **New → Google Docs → Blank Document.**
2. Title: Cloud Computing Notes.
3. Write sample content:
4. Topic: Cloud Computing Overview
5. - Cloud computing delivers services over the internet.
6. - Models: IaaS, PaaS, SaaS.
7. - Providers: AWS, Azure, Google Cloud.
8. - Advantages: Scalability, Flexibility, Cost Efficiency.
9. Format text using headings, bullet points, and bold for keywords.

Step 4 – Organize Files

- Create a folder **"Cloud Practical Files"** in Google Drive.
- Move both **Spreadsheet** and **Notes** into it.

Step 5 – Share Files

- Right-click → **Share → Copy link → Set access as "Anyone with the link can view"**
  ✅ *Files shared via cloud successfully.*

Result:

Successfully created and shared **Google Sheet and Google Doc notes** using **Google Drive**, demonstrating SaaS cloud services.

Q2. Create and Host Static Web Page Using Any Cloud Provider (AWS Example)

[20 Marks]

Aim:

To **create and host a static web page** on a cloud platform (AWS S3).

Theory:

A **static website** consists of fixed content — HTML, CSS, and images.
Hosting a static website on the **cloud** provides scalability, reliability, and high availability.

Popular cloud providers:

- AWS S3
- Google Cloud Storage
- Microsoft Azure Static Web App

Here, AWS S3 (Simple Storage Service) is used to host a static website.

Steps (Using AWS S3):

Step 1 – Login to AWS Console

- Visit https://aws.amazon.com
- Open **S3** service.

Step 2 – Create S3 Bucket

1. Click **Create Bucket**.
2. Bucket name: my-static-website-demo.
3. Choose **Region:** Asia Pacific (Mumbai).
4. Uncheck "Block all public access."
   Allow public read access for hosting.

Step 3 – Upload Web Files

Create a simple HTML file named index.html:

<!DOCTYPE html>

<html>

<head>

```
  <title>My Cloud Web Page</title>
</head>
<body>
  <h1>Welcome to Cloud Computing Practical</h1>
  <p>This website is hosted on AWS S3.</p>
</body>
</html>
```

Upload index.html into the S3 bucket.

Step 4 – Enable Static Website Hosting

1. In S3 bucket → Go to **Properties** tab.
2. Scroll to **Static website hosting** → Enable it.
3. Set:
    - Index document: index.html
4. Click **Save changes.**

Step 5 – Get Website URL

- After enabling hosting, S3 provides a public **Website Endpoint URL**, e.g.:
- http://my-static-website-demo.s3-website.ap-south-1.amazonaws.com

Open this link — your web page is live!

Output Example:

Welcome to Cloud Computing Practical

This website is hosted on AWS S3.

Result:

Successfully created and hosted a **static website** using **AWS S3**, demonstrating deployment of web content on cloud storage.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ

(Cloud Computing)

Assignment 23

Q1. Practical Implementation of File Sharing and Storage as a Service (STaaS)

[10 Marks]

Theory:

Storage as a Service (STaaS) is a **cloud computing model** where users store, manage, and share data over the internet.
The provider maintains the infrastructure and ensures data availability, scalability, and security.

Examples:

- Google Drive
- Dropbox
- AWS S3
- OwnCloud

Steps (Using Google Drive as Example):

Step 1 – Open Google Drive

- Go to https://drive.google.com
- Sign in with your Google account.

Step 2 – Upload Files

- Click **New → File Upload**
- Select any file (e.g., Report.pdf, Notes.txt, or Presentation.pptx).
- The file is stored on the cloud.

Step 3 – Create Folders

- Click **New → Folder → Name it "Cloud_Storage_Demo"**
- Move uploaded files into this folder.

Step 4 – Share Files

- Right-click on a file → **Share**.
- Add user email or copy **Shareable link**.

- Set access permission: *Viewer*, *Editor*, or *Commenter.*

Step 5 – Access Shared File

- Open the shared link in another browser or device.
  File opens successfully — **file sharing and storage implemented**.

Output Example:

Google Drive Folder: Cloud_Storage_Demo

Files: Notes.txt, Report.pdf, Image1.png

Access: Shared with user@example.com (Viewer)

Result:

Successfully implemented **File Sharing and Storage as a Service (STaaS)** using a cloud platform for uploading, storing, and sharing data securely.

Q2. Workings of Storage as a Service using OwnCloud to Make Presentation on Types of Clouds (Minimum 5 Slides)

[20 Marks]

Theory:

OwnCloud is an open-source platform that provides private **cloud storage**.
It allows users to **upload, manage, and share** files just like Google Drive, but it runs on their own servers.

Service Model: Storage as a Service (STaaS)
Concept Covered: File management + Presentation sharing via OwnCloud

Steps:

Step 1 – Install and Setup OwnCloud

1. Download OwnCloud from https://owncloud.org/download/.
2. Install **XAMPP** or **Apache Server** on your system.
3. Copy OwnCloud folder to:
   - Windows: C:\xampp\htdocs\owncloud
   - Linux: /var/www/html/owncloud
4. Start **Apache** and **MySQL**.

Step 2 – Create Database

1. Visit http://localhost/phpmyadmin.
2. Create new database: owncloud_db.

Step 3 – Configure OwnCloud

1. Open browser → http://localhost/owncloud.
2. Enter:
    - Username: admin
    - Password: 12345
    - Database name: owncloud_db
3. Click **Finish Setup**.
   ✅ *OwnCloud dashboard opens.*

Step 4 – Create Presentation on "Types of Clouds"

Create a **PowerPoint (PPTX)** file titled *"Types of Cloud Computing"* with at least **5 slides**:

Slide 1: Introduction to Cloud Computing
Slide 2: Public Cloud (e.g., AWS, Google Cloud)
Slide 3: Private Cloud (e.g., VMware, OpenStack)
Slide 4: Hybrid Cloud (combination of public & private)
Slide 5: Community Cloud (for organizations with shared goals)
Slide 6 (Optional): Advantages of Cloud Deployment Models

Save file as Types_of_Clouds.pptx.

Step 5 – Upload File to OwnCloud

1. On dashboard → Click **+ → Upload File → Browse "Types_of_Clouds.pptx"**
2. The file is now stored in OwnCloud.

Uploaded presentation successfully.

Step 6 – Share the Presentation

- Click the **Share** icon next to the file.
- Generate a **Shareable Link** (with password if required).
- Open the link in another browser to verify access.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ
(Cloud Computing)
Assignment 24

Q1. Working and Implementation of Infrastructure as a Service (IaaS)

[10 Marks]

Theory:

Infrastructure as a Service (IaaS) is a **cloud computing service model** that provides virtualized computing resources over the internet.
It offers **virtual machines (VMs)**, **storage**, **networking**, and **operating systems** on-demand.

Examples:

- **AWS EC2** (Amazon Elastic Compute Cloud)
- **Google Compute Engine**
- **Microsoft Azure Virtual Machines**

Characteristics of IaaS:

- On-demand access to infrastructure
- Pay-as-you-go pricing
- Virtualization and scalability
- Resource pooling and automation

Steps (Using AWS EC2 Example):

Step 1 – Login to AWS Console

- Go to https://aws.amazon.com
- Sign in using your AWS account.

Step 2 – Launch an EC2 Instance

1. Navigate to **EC2 Dashboard → Launch Instance**.
2. Give name: My-IaaS-Demo.
3. Choose **Amazon Machine Image (AMI):** Ubuntu Server 22.04 LTS.
4. Choose instance type: t2.micro (Free Tier eligible).
5. Configure network and storage (default options).
6. Create a **key pair** for SSH access.

7. Click **Launch Instance**.

Your virtual machine (infrastructure) is now created on the cloud.

Step 3 – Connect to Instance

- Select your instance → Click **Connect**.
- Choose **EC2 Instance Connect → Open Terminal**
  or use SSH command:
- ssh -i mykey.pem ubuntu@ec2-13-233-XX-XX.ap-south-1.compute.amazonaws.com

Step 4 – Execute Basic Commands

sudo apt update

ls

mkdir test_folder

cd test_folder

echo "IaaS Practical" > demo.txt

cat demo.txt

✅ *Shows basic virtual machine operations.*

Step 5 – Stop and Terminate Instance

After practical completion:

- **Stop Instance → Terminate** to save cost.

Result:

Successfully launched and managed a **Virtual Machine (VM)** using **AWS EC2**, demonstrating the working of **Infrastructure as a Service (IaaS)**.

Q2. Workings of Google Drive to Make Presentation on Types of Cloud (Minimum 5 Slides)

[20 Marks]

Theory:

Google Slides is a cloud-based presentation tool under **Google Workspace**.
It enables users to create, edit, and share presentations online.
It demonstrates **Software as a Service (SaaS)** using cloud storage as backend infrastructure.

Step 1 – Open Google Drive

- Visit https://drive.google.com.
- Login using Google account.

Step 2 – Create New Presentation

- Click **New → Google Slides → Blank Presentation**.
- Rename file: Types_of_Clouds_Presentation.

Step 3 – Prepare Slides

Create minimum **5 slides** as follows:

Slide No.	Title	Content
1	Introduction to Cloud Computing	Definition and features of cloud computing
2	Public Cloud	Description + Examples (AWS, Google Cloud, Azure)
3	Private Cloud	Description + Example (VMware, OpenStack)
4	Hybrid Cloud	Combination of public and private
5	Community Cloud	Shared among organizations with common goals
6 (Optional)	Advantages	Scalability, Flexibility, Cost-efficiency

Add images, shapes, and icons for better visualization.

Step 4 – Save and Organize

- File automatically saves in Google Drive.
- Create a folder **"Cloud_Practicals"** and move the presentation into it.

Step 5 – Share Presentation

- Right-click → **Share → Copy link → Set access to "Anyone with link can view."**

Presentation stored and shared successfully via cloud.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ
(Cloud Computing)
Assignment 25

Q1. Create Virtual Machine and Perform Basic Shell Commands

[10 Marks]

Theory:

A **Virtual Machine (VM)** is a software-based environment that emulates a physical computer.
It allows users to install an operating system and run programs in an isolated environment.

In Cloud Computing, virtual machines are provided under **Infrastructure as a Service (IaaS)** model.

Steps (Using Oracle VirtualBox):

Step 1 – Install VirtualBox

- Download and install **Oracle VirtualBox** from https://www.virtualbox.org.

Step 2 – Create New Virtual Machine

1. Open VirtualBox → Click **New**.
2. Name: Ubuntu_VM.
3. Type: **Linux**, Version: **Ubuntu (64-bit)**.
4. Set:
    - **Memory Size:** 2048 MB
    - **Hard Disk:** Create a new virtual hard disk (VDI) → Size: 20 GB.
5. Click **Create**.

Step 3 – Install Operating System

1. Start VM → Browse ISO file (Ubuntu ISO).
2. Follow installation steps.
3. Once installed, login to Ubuntu.

Step 4 – Perform Basic Shell Commands

Open **Terminal** and execute the following commands:

Command	Description	Example Output
pwd	Shows current directory	/home/user
ls	Lists files and directories	Desktop Documents Downloads
mkdir demo	Creates a new directory	Folder “demo” created
cd demo	Changes directory	Navigated into demo
echo "Hello Cloud" > file.txt	Creates a text file	file.txt created
cat file.txt	Displays file content	Hello Cloud
rm file.txt	Deletes a file	File removed successfully

Basic shell commands executed successfully.

Result:

Successfully created a **Virtual Machine** and executed **basic shell commands** on the virtualized Linux OS environment.

Q2. Create Google Form for Generate Certificates for Workshop

[20 Marks]

Theory:

Google Forms is a **Software as a Service (SaaS)** tool used for creating surveys, registration forms, and feedback forms online.
It automatically stores data in **Google Sheets**, which can then be used to generate **certificates** using **Google Apps Script** or add-ons (like Autocrat).

Steps:

Step 1 – Open Google Forms

- Go to https://forms.google.com
- Click **Blank Form**.
- Title: *Workshop Registration and Certificate Form*

Step 2 – Add Form Fields

Add the following questions:

Field Type	Question	Response Type
Short answer	Full Name	Required
Short answer	Email ID	Required
Short answer	College Name	Optional
Multiple choice	Attended all sessions?	Yes / No
Paragraph	Feedback about workshop	Optional

Form is ready to collect details.

Step 3 – Configure Settings

- Click **Settings → Responses → Collect Email Addresses**.
- Enable **"Send Respondents a Copy"**.
- Optionally limit to one response per user.

Step 4 – Link with Google Sheets

- Click **Responses → Link to Sheets → Create New Spreadsheet**.
  Spreadsheet will store all participant data.

Step 5 – Generate Certificates (Optional)

Use **Autocrat Add-on**:

1. Open Google Sheets → Extensions → Add-ons → Get Add-ons.
2. Search for **Autocrat** → Install.
3. Create a **Google Docs certificate template** with placeholders like:
4. This is to certify that <<Full Name>>
5. has successfully participated in the Workshop on Cloud Computing.
6. In Autocrat → Select Template → Map form fields → Choose output folder → Click **Run Merge**.

Certificates generated automatically for all participants.

Step 6 – Share Form

- Click **Send → Link → Shorten URL.**
- Example: https://forms.gle/WorkshopCloudForm123

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ
(Cloud Computing)
Assignment 26

Q1. Working of Software as a Service (Amazon)

[10 Marks]

Theory:

Software as a Service (SaaS) is a **cloud computing service model** where software applications are hosted by a cloud provider and made available to users over the internet. Users can access these applications through a web browser without installation or maintenance.

Key SaaS Features:

- Subscription-based model
- Multi-user access
- Accessible from any device
- Automatic updates and backups

Amazon SaaS Examples:

1. **Amazon WorkDocs** – Online document storage, editing, and sharing.
2. **Amazon WorkMail** – Business email and calendar service.
3. **Amazon Chime** – Video conferencing and communication tool.
4. **Amazon Connect** – Cloud-based contact center software.

Steps (Using Amazon WorkDocs Example):

Step 1 – Login to AWS Console

- Go to https://aws.amazon.com
- Login or create an AWS account.

Step 2 – Open Amazon WorkDocs

- From **AWS Services**, search for **WorkDocs**.
- Click **Set up WorkDocs**.

- Choose region (e.g., Asia Pacific – Mumbai).
- Create a **WorkDocs site** with organization name: MyCloudDocs.

Step 3 – Access the WorkDocs Site

- After setup, open the provided URL (e.g., https://myclouddocs.awsapps.com/workdocs).
- Sign in with the admin credentials.

Step 4 – Create and Share Documents

1. Click **Upload → Files → Select Document (PDF or Word)**.
2. Add collaborators and share the document via email.
3. Collaborators can edit, comment, and download files in real time.

Step 5 – Manage Files and Permissions

- Create folders (e.g., "Project Reports").
- Set access permissions: *Read*, *Write*, or *Owner*.
  ✅ Demonstrates SaaS-based document management.

Result:

Successfully demonstrated **Software as a Service (SaaS)** using **Amazon WorkDocs**, showcasing document sharing and collaboration through a web interface.

Q2. Workings of Google Drive to Make Presentation on Types of Cloud (Minimum 5 Slides)

[20 Marks]

Aim:

To create a **Google Slides presentation** on **Types of Cloud Computing** using **Google Drive**, demonstrating **SaaS** in action.

Theory:

Google Drive provides multiple cloud-based tools such as **Docs**, **Sheets**, and **Slides** that represent **Software as a Service (SaaS)**.
These applications run entirely in the browser and save data automatically to the cloud.

Google Slides allows users to create, edit, and share presentations online.

Steps:

Step 1 – Open Google Drive

- Go to https://drive.google.com.

- Log in with your Google account.

Step 2 – Create a New Presentation

- Click **New → Google Slides → Blank Presentation.**
- Title the file: Types_of_Clouds_Presentation.

Step 3 – Create Slides (Minimum 5 Slides)

Slide No.	Title	Content
1	Introduction to Cloud Computing	What is cloud computing and its importance
2	Public Cloud	Examples: AWS, Azure, Google Cloud
3	Private Cloud	Used within organizations, e.g., OpenStack
4	Hybrid Cloud	Combines public and private cloud features
5	Community Cloud	Shared among organizations with common goals
6 (Optional)	Advantages of Cloud Models	Scalability, flexibility, pay-per-use

Add suitable **images, icons, and transitions** to enhance the presentation.

Step 4 – Format and Save

- Customize the theme using **Slide → Change Theme**.
- Google Slides auto-saves the presentation in Drive.

Step 5 – Share the Presentation

- Click **Share → Copy link → Set "Anyone with link can view."**
  Presentation is now stored and accessible online.

SUBJECT: CS-513-MJP: Lab Course on CS-512-MJ
(Cloud Computing)
Assignment 27

Q1. Practical Implementation of File Sharing and Storage as a Service (STaaS)

[10 Marks]

Theory:

Storage as a Service (STaaS) is a **cloud service model** that allows users to store, access, and share data over the internet.
The service provider manages storage infrastructure, ensuring **data availability, security, and scalability**.

Steps (Using Google Drive Example):

Step 1 – Login to Google Drive

- Visit https://drive.google.com.
- Login with your Google account.

Step 2 – Upload Files

- Click **New → File Upload**.
- Select files (e.g., Report.pdf, Notes.docx, Image.png).
- The files are uploaded to the cloud.

Step 3 – Create Folder

- Click **New → Folder → Name: Cloud_Storage_Demo**.
- Move uploaded files into this folder.

Step 4 – Share Files

- Right-click on the file → **Share**.
- Add user email or click **Copy Link**.
- Set access level: *Viewer*, *Editor*, or *Commenter*.

File successfully shared with another user.

Step 5 – Access from Another Device

- Open the link on another device or account.
- The file opens online — no download needed.

Result:

Successfully implemented **File Sharing and Storage as a Service (STaaS)** using Google Drive for uploading, storing, and sharing files online.

Q2. Show the Working and Implementation of Identity Management (IAM)

[20 Marks]

Theory:

Identity and Access Management (IAM) is a framework of **policies and technologies** that ensures the right individuals have the proper access to cloud resources.
It helps manage **user identities**, **authentication**, and **authorization** securely.

IAM Features:

- Centralized user management
- Role-based access control (RBAC)
- Policy-based permissions
- Secure access using credentials and MFA (Multi-Factor Authentication)

Steps (Using AWS IAM Example):

Step 1 – Login to AWS Console

- Visit https://aws.amazon.com.
- Sign in using your AWS account.

Step 2 – Open IAM Service

- In the AWS Console → Search **IAM**.
- Click to open the **Identity and Access Management Dashboard**.

Step 3 – Create a New User

1. Click **Users → Add User**.
2. Enter user name: StudentUser.
3. Select **Access type:** AWS Management Console access.
4. Set password: Cloud@123.

Step 4 – Assign Permissions

1. Choose **Attach existing policies directly**.
2. Select **AmazonS3ReadOnlyAccess** policy.

3. Click **Next → Create User**.

User created successfully with restricted S3 access.

Step 5 – Verify Access

- Sign out of admin account.
- Login as **StudentUser** using the credentials.
- Try accessing **S3 Service** → Only view permission is allowed.
- Try deleting or uploading files → Access denied.

Identity management and access control verified.

Step 6 – Optional: Enable Multi-Factor Authentication (MFA)

- Go to **Security Credentials → Assign MFA Device**.
- Scan QR code using Google Authenticator app.
- Add MFA code to complete setup.

Result:

Successfully implemented **Identity and Access Management (IAM)** using AWS by creating a user, assigning limited permissions, and verifying restricted access.