To deploy a CRUD (Create, Read, Update, Delete) operation Flask application on an Apache server and MySQL Server hosted on a Google Cloud Platform (GCP) Virtual Machines (VM) running Ubuntu, the following points outline the key steps:

1. **GCP VM’s Setup**

* **Create a Project**: Project named My Vcc Project on GCP.
* **Create a VM Instance**: Launch three new Ubuntu VM instances with configuration (Machine Type: e2-micro, OS: Ubuntu20.4 LTS) to handle your application's requirements.
* **Assign a Static IP Address**: Reserve a static IP address for the VM to ensure the application is accessible via the same IP.
* **Set up Firewall Rules**: Configure firewall rules to allow HTTP (port 80) and HTTPS (port 443) traffic.

1. **VM1 Instance Preparation**

* **SSH into VM Instance**: Use the SSH feature from the GCP console or a terminal to connect to the VM.
* **Update the System**: Access as a Super User using **sudo su**.

Run **apt update** and **apt upgrade** to ensure the system packages are up-to-date.



* **Install Python and Pip**: Install Python and pip (Python package installer) using **apt install python python3-pip**



1. **Flask Application Setup in VM1 Instance**

* **Create Flask App directory**: Use make directory to create app directory in apache location. **mkdir /var/www/myappenv**



* **Set Up Virtual Environment**: Create and activate a Python virtual environment in above directory using **python3 -m venv venv** and **source venv/bin/activate**.



* **Install Dependencies**: Install flask, pymysql using **pip install flask pymysql**



* **Add Student Model**: This model will add student details and view student details. The following attributes will be inserted for a student. ( **name, roll\_number, dob**).
* **Create the Student Table**: Add code to create the **students** table in the database **student** using **(/init\_db)**.

### ****Flask Routes for CRUD Operations****

* **Create Route (POST)**: Implement a Flask route that allows creating a new student entry in the database via a POST request (/add\_students).
* The route should accept JSON data with student details and insert them into the database (/add\_students\_json).
* **View Route (GET)**: Implement a Flask route that retrieves and displays student details from the database via a GET request (/view\_students).

1. **Apache Server Installation and Configuration**

* **Install Apache**: Install Apache2 on the VM using **apt install apache2**.
* **Install mod\_wsgi**: Install mod\_wsgi, which acts as a bridge between Apache and your Flask application, using **apt install libapache2-mod-wsgi-py3**.

1. **Apache Configuration for Flask Application**

* **Create Apache Configuration File**: Created a new Apache configuration file for Flask application, /etc/apache2/sites-available/myapp.conf.
  + Define ServerName, ServerAdmin, and other relevant directives.
  + Set up WSGIDaemonProcess, WSGIScriptAlias, and the Flask application directory.
  + Define static file serving if required.
* **Enable Your Flask App Site**: Enable your Flask application site using **a2ensite myappconf.**
* **Restart Apache**: Restart Apache with **systemctl restart apache2** to apply changes.



1. **VM2 Instance & Database Setup**

* **SSH into VM Instance**: Use the SSH feature from the GCP console or a terminal to connect to the VM.
* **Update the System**: Run **apt update** and **apt upgrade** to ensure the system packages are up-to-date.





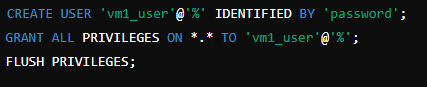
* **Install Database Software**: Install MySQL Database using the package manager commands (apt install mysql-server).



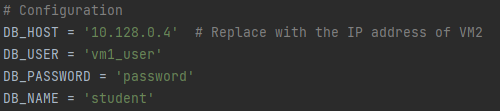
* **Configure Database**: Configure database for Remote Server, create the required User, database and configure user permissions.
* **Configure MySQL for Remote Access:** By default, MySQL only listens on localhost. To allow remote connections:
* Edit the MySQL configuration file: nano /etc/mysql/mysql.conf.d/mysqld.cnf



* **Find the line:** bind-address = 127.0.0.1
* **Change it to:** bind-address = 0.0.0.0
* **Create User:** created a MySQL user that VM1 can use to connect to the MySQL server



* **Update Flask Configurations**: Update your Flask application's configuration files with the database connection settings.



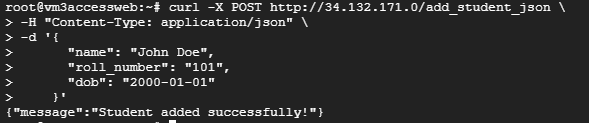
1. **VM3 Instance Setup:**

* **SSH into VM3 Instance**: Use the SSH feature from the GCP console or a terminal to connect to the VM.
* **Update the System**: Run **apt update** and **apt upgrade** to ensure the system packages are up-to-date.
* **Access the application:** Using curl to interact with the Flask application on the VM3 for creating a table, inserting student details, and viewing student details from the database.

**C:\Users\ShivaayPC\Pictures\curl.PNG**

### ****12. Testing CRUD Operations on the GCP VM3****

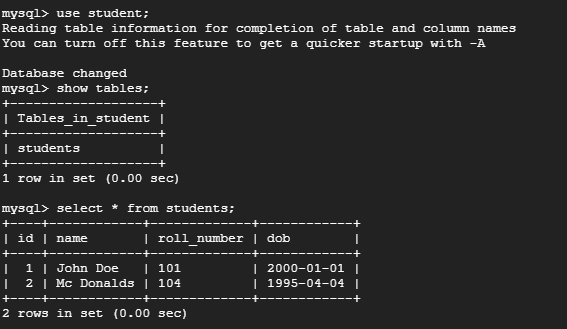
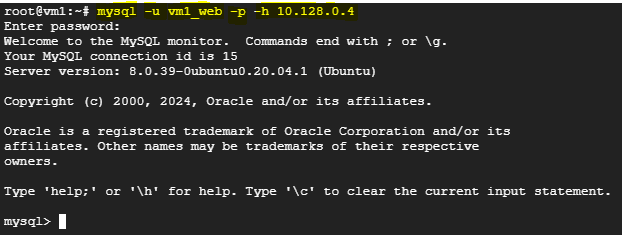
* **Deploy Flask App**: Ensure the Flask application with the CRUD routes is deployed and running on the VM.
* **Using curl : c**url commands to interact with the deployed Flask application:
  + **Create a Student**: curl -X POST <http://34.132.171.0/add_student_json> -H "Content-Type: application/json" -d '{"name": "Jane Doe", "age": 22, "grade": "B"}'



* + **View All Students**: curl <http://34.132.171.0/view_students>



* **Verify Database Entries**: Log in to the database (e.g., via MySQL CLI or PostgreSQL) to verify that student entries are correctly inserted and retrieved.



### ****14. Error Handling****

### **Error Handling**: Ensure that the Flask application returns appropriate error messages and status codes (e.g., 400 Bad Request for invalid data) for incorrect curl requests.

This report provides a comprehensive overview of the steps necessary to deploy a CRUD Flask application on an Apache server on a GCP VM running Ubuntu. Each step is crucial to ensure the successful deployment, security, and maintenance of the application.