**MADRAS INSTITUTE OF TECHNOLOGY**

**ANNA UNIVERSITY CHENNAI**

**CHENNAI - 600 044**

DEPARTMENT OF INFORMATION TECHNOLOGY

**5/8 B.TECH ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**NAAN MUDHALVAN PROJECT**

# TITLE: AWS-HOSTED VIRTUAL CLASSROOM AND LEARNING PLATFORM

Submitted by:

Thanya Gayathri N-2022510029

Sasikala S-2022510055

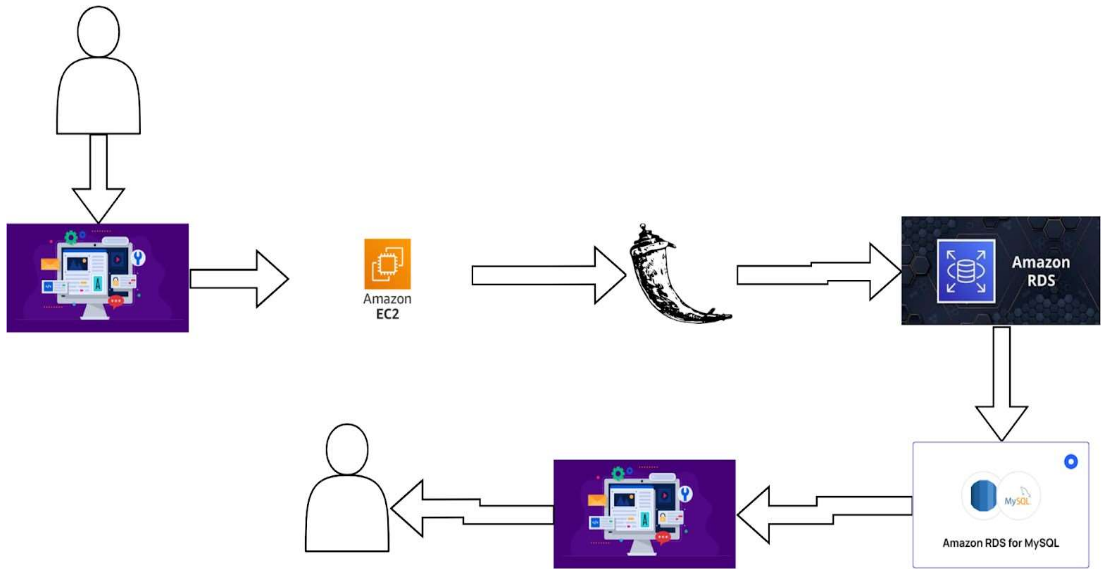
Janani S-2022510049

Lavanya J-2022510004

# VIRTUAL CLASS IN AWS

## ABSTRACT

The AWS-hosted virtual classroom and learning platform is a scalable and secure solution designed to facilitate online education. The platform offers interactive virtual classrooms, secure content hosting, real-time communication tools, and data analytics. Leveraging AWS services, the system ensures high availability, low latency, and robust security measures to meet the needs of educators and learners.



## Objectives:

1. Develop a user-friendly, secure, and scalable virtual classroom.
2. Integrate real-time communication tools for lectures and discussions.
3. Provide storage and retrieval systems for educational resources.
4. Enable data analytics to monitor student engagement and performance.
5. Ensure platform availability and fault tolerance.
6. Students or login details can be retrieved whenever and wherever required.
7. Used HTML for the backend so that it will be very user friendly.

## KEY FEATURES:

- **User Management**:

  - **Registration and Login**: Users can sign up and log in securely using an integrated authentication system. AWS Cognito or Flask-Login ensures data protection and seamless session management.
  - **Role-Based Access**: Separate dashboards and functionalities are tailored for students, teachers, and administrators.

- **Course Materials Management**:

  - **S3 Integration**: Course content, including lecture notes, videos, and other educational materials, are uploaded and stored in Amazon S3. S3 ensures high availability and secure access to these resources.
  - **Version Control**: Educators can update course materials, and students always have access to the latest versions.

- **Data Management**:

  - **Amazon RDS**: A relational database setup on RDS stores user information, course metadata, and other critical data. The database schema is designed for optimal performance and scalability.
  - **Secure Queries**: SQLAlchemy, the ORM used with Flask, ensures safe interactions with the database.

- **Deployment and Scalability**:

  - **AWS EC2**: The Flask application is deployed on an EC2 instance, providing the flexibility to scale resources based on user demand.
  - **Load Balancing**: Elastic Load Balancing (ELB) ensures even distribution of traffic across instances.

- **Real-Time Communication**:

  - Using AWS Chime SDK or WebRTC (optional for future development), the platform supports live lectures, discussions, and virtual office hours.

- **Analytics and Monitoring**:

  - **AWS CloudWatch**: Real-time monitoring tracks system health and performance.

- **Data Analytics**: AWS QuickSight generates insights on student progress, attendance, and course engagement metrics.

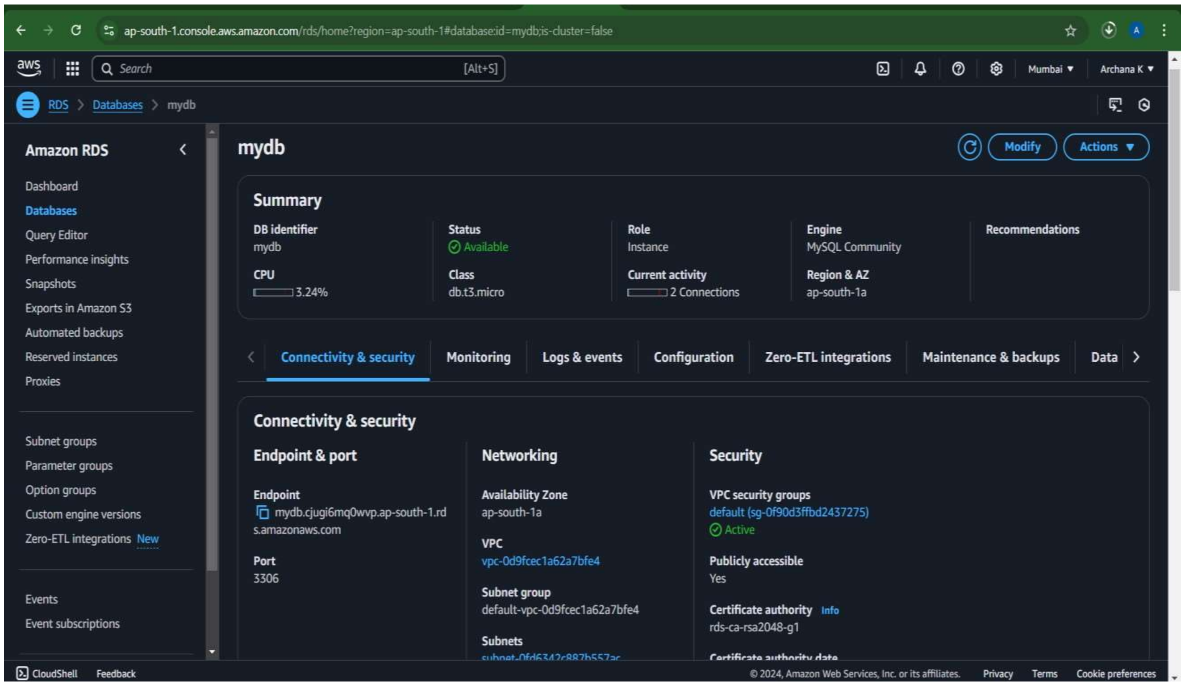
- **Security**:

- End-to-end encryption ensures secure transmission of sensitive data.
- Role-based IAM policies regulate access to AWS resources.

## Step by Step Process of Implementation :

**Step 1:** Log in to AWS Management Console

- Navigate to the AWS Management console.
- Sign in with your AWS credentials.
- In the AWS Management Console, search for RDS in the search bar.
- Click on RDS to open the service dashboard.
- On the RDS dashboard, click Create database.
- Standard create: Offers full control over configuration options.
- Easy create: Automates most configuration decisions for quicker setup.
- For full customization, select Standard create.
- Choose your preferred database engine, such as: MySQL
- Select the version of the database engine.
- Deployment Option: Choose between Production or Dev/Test.
- DB Instance Identifier: Provide a name for the database instance.
- Master Username: Set the admin username (e.g., admin).
- Master Password: Set a strong password for the admin user.
- DB Instance Class: Select the instance size based on performance needs (e.g., db.t3.micro for free tier-eligible setups).
- General Purpose SSD (gp2)
- Provisioned IOPS SSD (io1)
- Magnetic
- Set allocated storage size (e.g., 20 GB).
- Select a VPC (Virtual Private Cloud) or allow RDS to create one automatically.
- Configure Public Access:
- Yes if the database needs to be accessed over the internet.
- No for internal access only.
- Choose or create a Subnet Group for high availability.
- Set VPC Security Groups to control inbound and outbound traffic.


ap-south-1.console.aws.amazon.com/rds/home?region=ap-south-1#database:id=mydb;is-cluster=false

RDS > Databases > mydb

**mydb**

**Summary**

| DB identifier | Status | Role | Engine | Recommendations |
|---|---|---|---|---|
| mydb | Available | Instance | MySQL Community | |
| **CPU** | **Class** | **Current activity** | **Region & AZ** | |
| 3.24% | db.t3.micro | 2 Connections | ap-south-1a | |

**Connectivity & security**

**Endpoint & port**

Endpoint: mydb.cjugi6mq0wvp.ap-south-1.rds.amazonaws.com

Port: 3306

**Networking**

Availability Zone: ap-south-1a

VPC: vpc-0d9fcec1a62a7bfe4

Subnet group: default-vpc-0d9fcec1a62a7bfe4

Subnets

**Security**

VPC security groups: default (sg-0f90d3ffbd2437275) Active

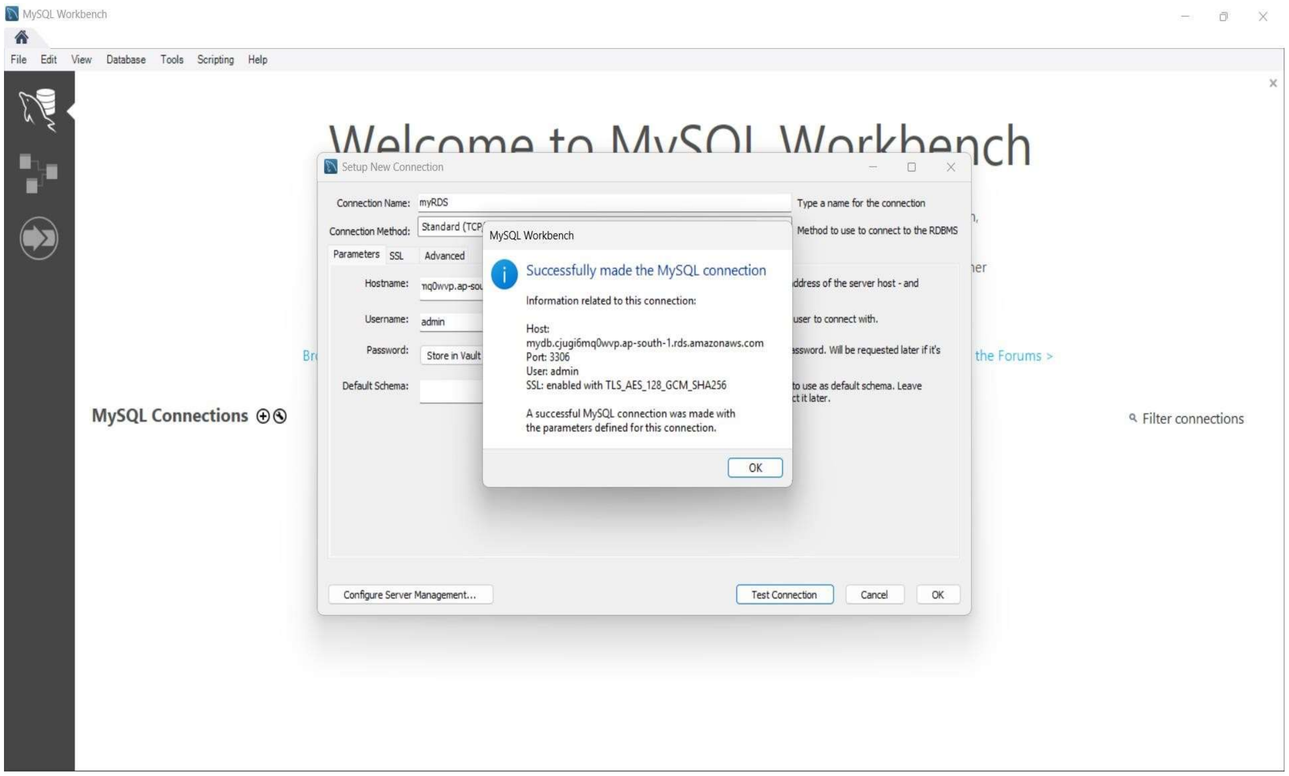
Publicly accessible: Yes

Certificate authority: rds-ca-rsa2048-g1

**Step 2** : Connect to MySQL Workbench.

- Download and install **MySQL Workbench** from the official MYSQL website.
- Launch MySQL Workbench after installation.
- Log in to the AWS Management Console.
- Navigate to the **RDS** dashboard.
- Select your MySQL RDS instance from the list.
- Locate the **Endpoint** and **Port** in the instance details.
  - Example: `mydb-instance.abc123xyz.us-east-1.rds.amazonaws.com`
  - Port: Default is `3306`.
- In the AWS Management Console, go to **EC2 > Security Groups**.
- Find the security group associated with your RDS instance.
- Add an inbound rule to allow your IP address to access the RDS instance:
  - **Type**: MySQL/Aurora
  - **Protocol**: TCP

  - **Port Range**: 3306
  - **Source**: Your IP (use **My IP** option for simplicity).
- Save the changes.
- Open **MySQL Workbench**.
- Click the + symbol next to "MySQL Connections" to create a new connection.
- **Connection Name**: Provide a name for the connection (e.g., `AWS RDS MySQL`).
- **Hostname**: Enter the **Endpoint** of your RDS instance (e.g., `mydb-instance.abc123xyz.us-east-1.rds.amazonaws.com`).
- **Port**: Enter the port number (default: `3306`).
- **Username**: Enter the master username you configured for your RDS instance.
- **Password**:
  - Click **Store in Vault** or **Store Password** and enter the master password.
- Click the **Test Connection** button.
- If prompted, select the appropriate authentication method (default is **Standard**).
- If the connection is successful, a confirmation dialog will appear.
- Click **OK** to save the connection.
- Double-click the new connection in MySQL Workbench to connect to your RDS instance.
- Once connected, you can:
  - View existing databases.
  - Run SQL queries.
  - Perform administrative tasks.



Setup New Connection

Connection Name: myRDS

Connection Method: Standard (TCP

Parameters | SSL | Advanced

Hostname: nq0wvp.ap-sou

Username: admin

Password: Store in Vault

Default Schema:

MySQL Workbench

**Successfully made the MySQL connection**

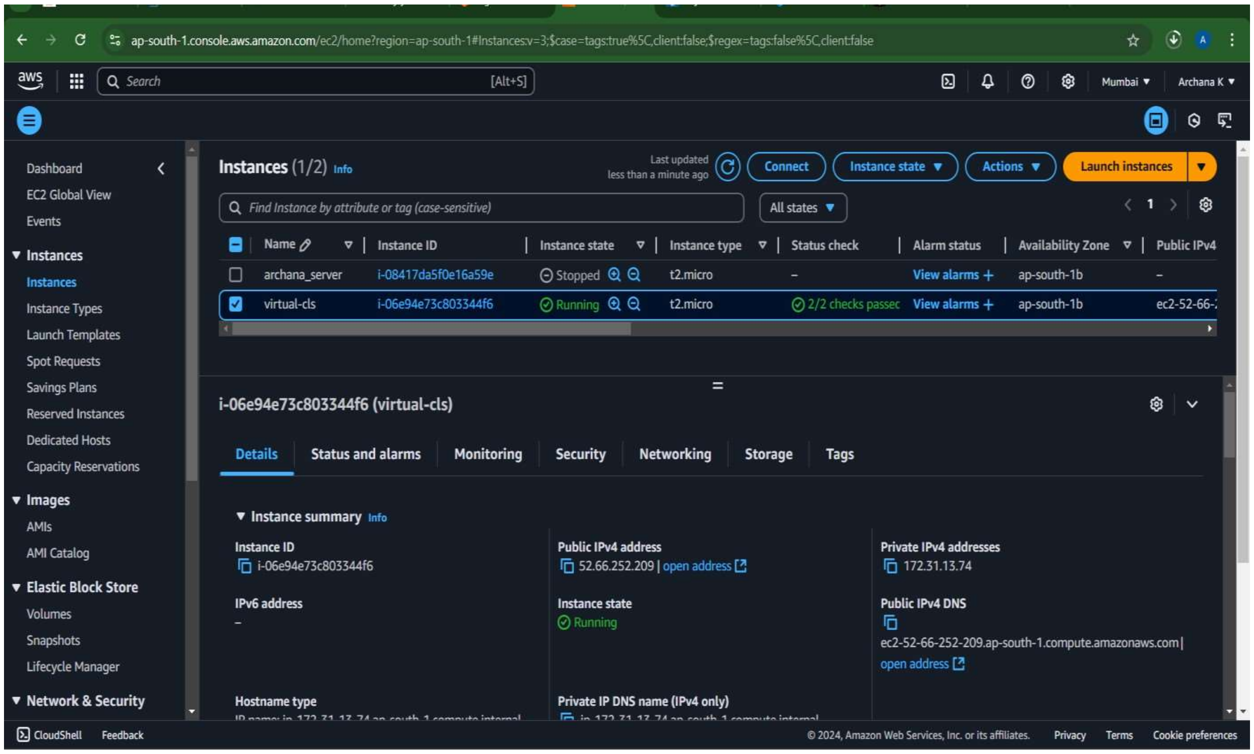
Information related to this connection:

```
Host:
mydb.cjugi6mq0wvp.ap-south-1.rds.amazonaws.com
Port: 3306
User: admin
SSL: enabled with TLS_AES_128_GCM_SHA256
```

A successful MySQL connection was made with the parameters defined for this connection.

MySQL Workbench
File Edit View Database Tools Scripting Help
Welcome to MySQL Workbench
MySQL Connections
Filter connections
OK
Configure Server Management...
Test Connection
Cancel
OK

## Step 3 : Create an EC2 Instance



ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#Instances:v=3;$case=tags:true%5C,client:false;$regex=tags:false%5C,client:false

Instances (1/2)

Last updated less than a minute ago

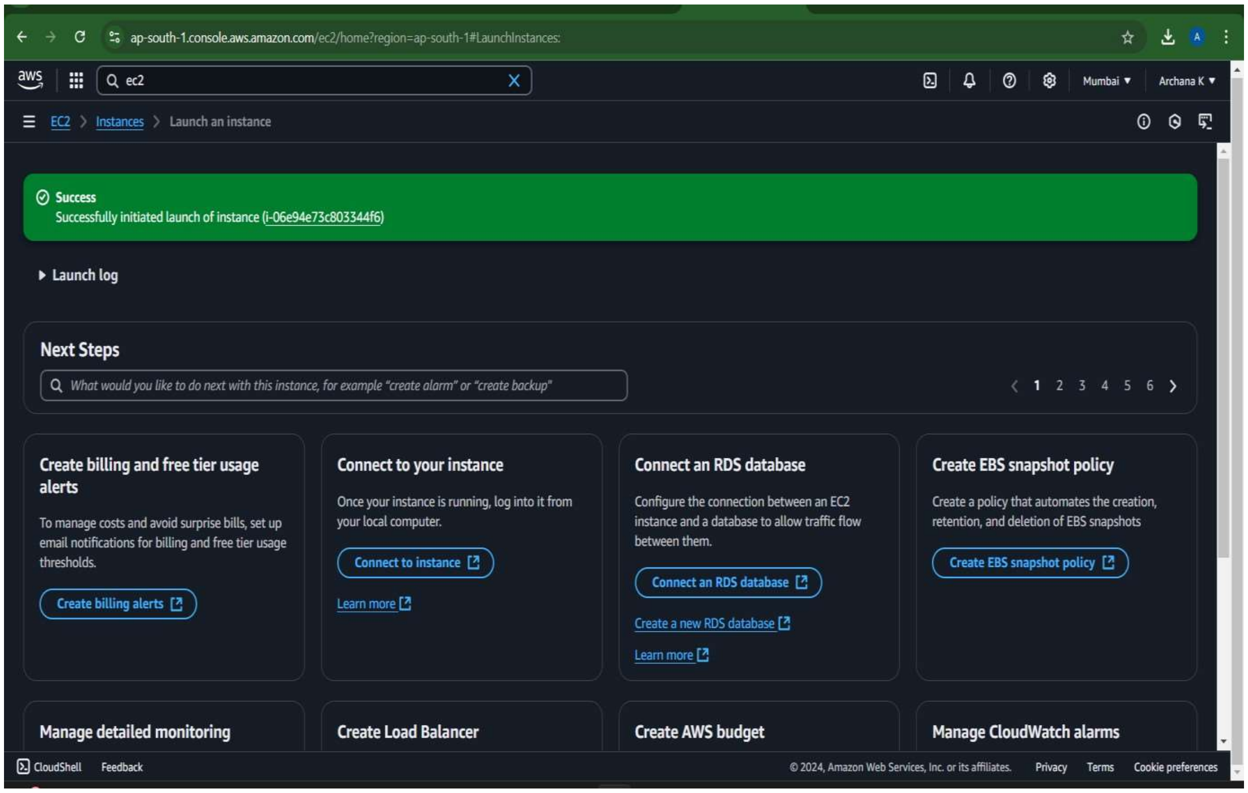
| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone | Public IPv4 |
|---|---|---|---|---|---|---|---|
| archana_server | i-08417da5f0e16a59e | Stopped | t2.micro | – | View alarms + | ap-south-1b | – |
| virtual-cls | i-06e94e73c803344f6 | Running | t2.micro | 2/2 checks passec | View alarms + | ap-south-1b | ec2-52-66-: |

i-06e94e73c803344f6 (virtual-cls)

Instance summary

| | | |
|---|---|---|
| Instance ID: i-06e94e73c803344f6 | Public IPv4 address: 52.66.252.209 \| open address | Private IPv4 addresses: 172.31.13.74 |
| IPv6 address: – | Instance state: Running | Public IPv4 DNS: ec2-52-66-252-209.ap-south-1.compute.amazonaws.com \| open address |
| Hostname type | Private IP DNS name (IPv4 only) | |

Step 4 : EC2 instance successfully created



ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LaunchInstances:

EC2 > Instances > Launch an instance

**Success**
Successfully initiated launch of instance (i-06e94e73c803344f6)

Launch log

**Next Steps**

What would you like to do next with this instance, for example "create alarm" or "create backup"

**Create billing and free tier usage alerts**
To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.
Create billing alerts

**Connect to your instance**
Once your instance is running, log into it from your local computer.
Connect to instance
Learn more

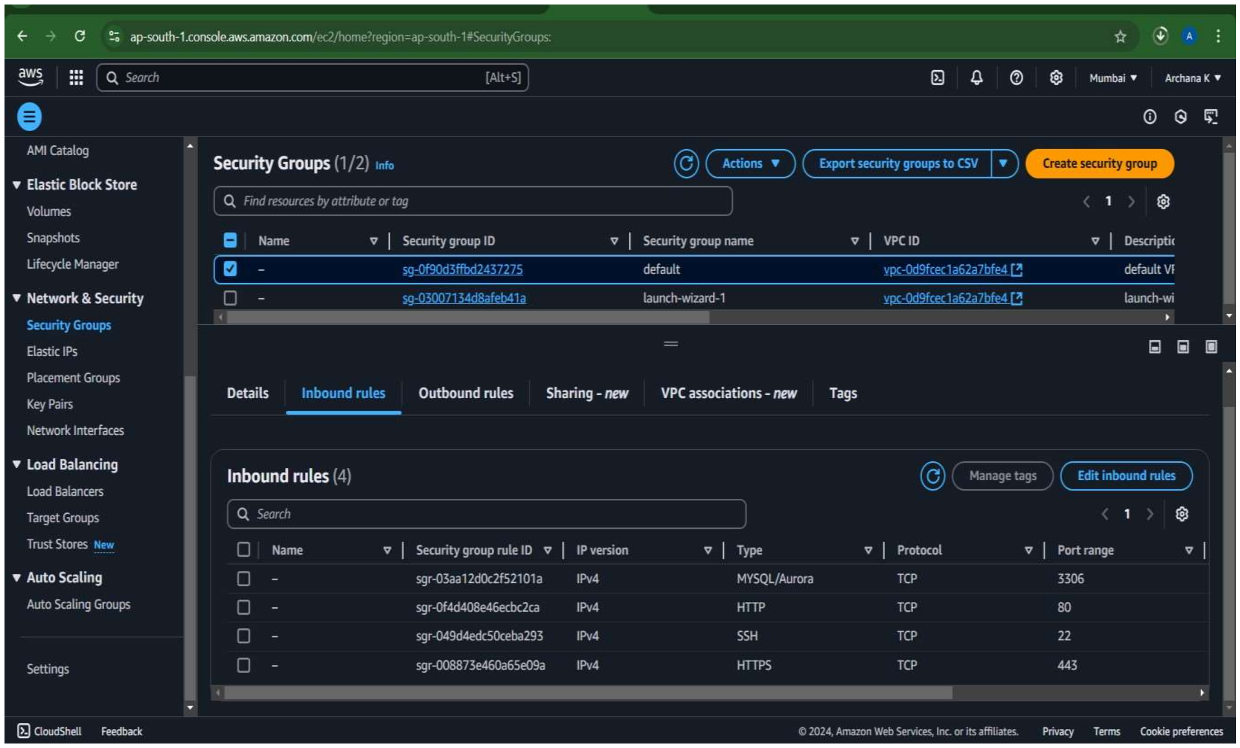
**Connect an RDS database**
Configure the connection between an EC2 instance and a database to allow traffic flow between them.
Connect an RDS database
Create a new RDS database
Learn more

**Create EBS snapshot policy**
Create a policy that automates the creation, retention, and deletion of EBS snapshots
Create EBS snapshot policy

**Manage detailed monitoring** | **Create Load Balancer** | **Create AWS budget** | **Manage CloudWatch alarms**

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 5 : Choose instance type, configure security groups, and key pair



ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#SecurityGroups:

**Security Groups** (1/2)

| Name | Security group ID | Security group name | VPC ID | Descriptic |
|---|---|---|---|---|
| – | sg-0f90d3ffbd2437275 | default | vpc-0d9fcec1a62a7bfe4 | default VF |
| – | sg-03007134d8afeb41a | launch-wizard-1 | vpc-0d9fcec1a62a7bfe4 | launch-wi |

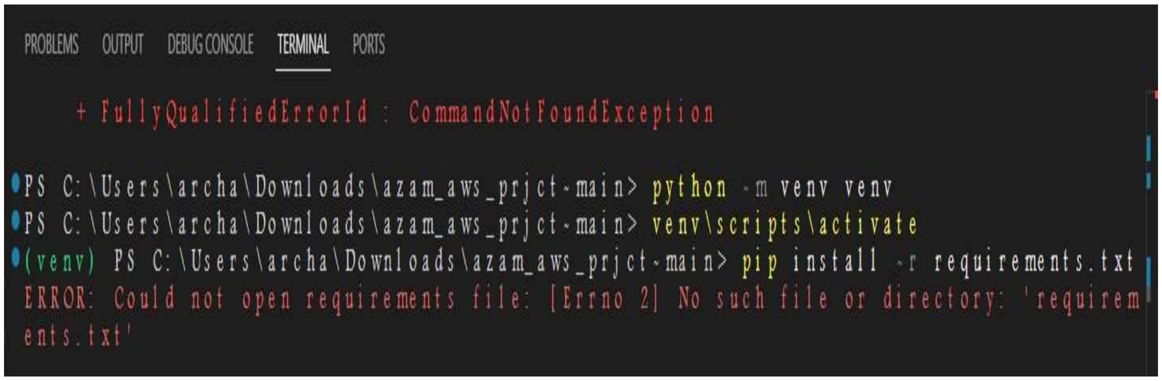
Details | Inbound rules | Outbound rules | Sharing - new | VPC associations - new | Tags

**Inbound rules** (4)

| Name | Security group rule ID | IP version | Type | Protocol | Port range |
|---|---|---|---|---|---|
| – | sgr-03aa12d0c2f52101a | IPv4 | MYSQL/Aurora | TCP | 3306 |
| – | sgr-0f4d408e46ecbc2ca | IPv4 | HTTP | TCP | 80 |
| – | sgr-049d4edc50ceba293 | IPv4 | SSH | TCP | 22 |
| – | sgr-008873e460a65e09a | IPv4 | HTTPS | TCP | 443 |

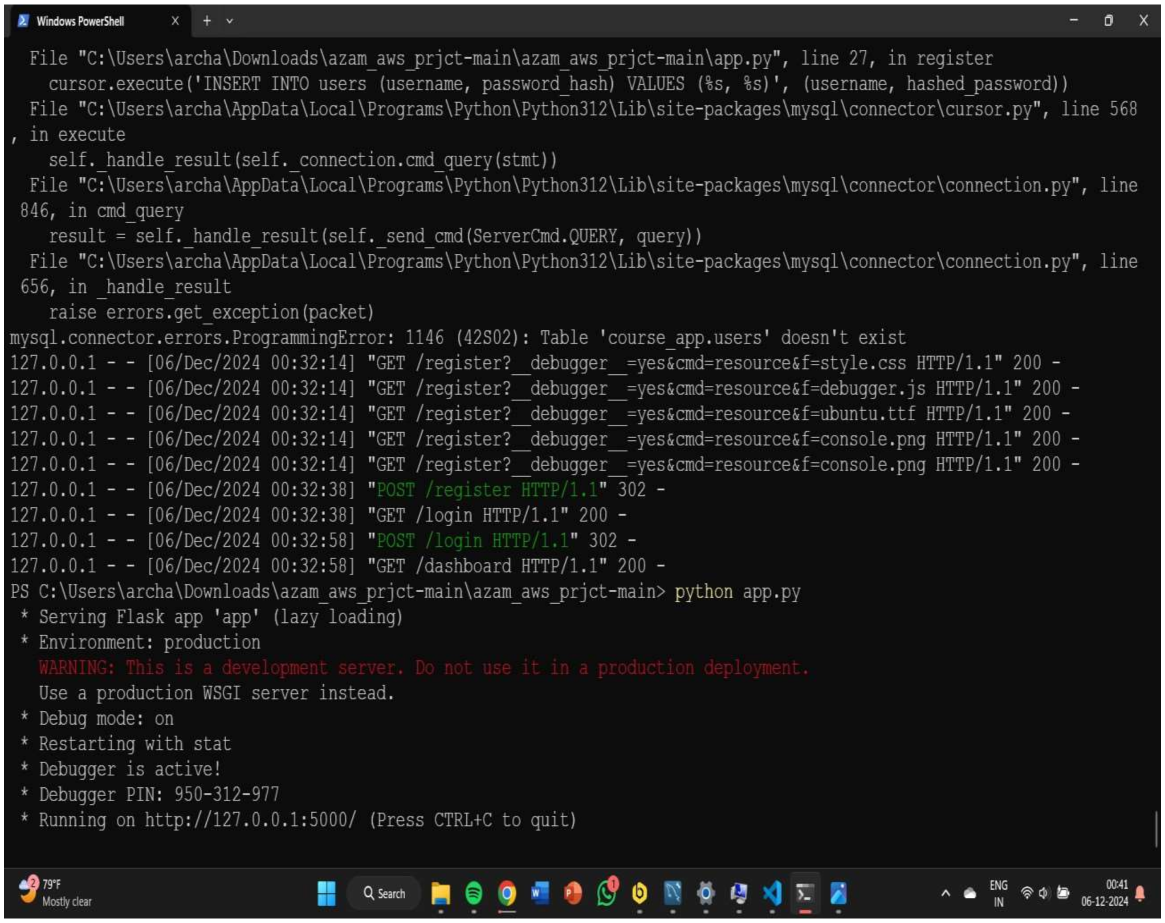
© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## Step 6 : Deploy Flask App on EC2



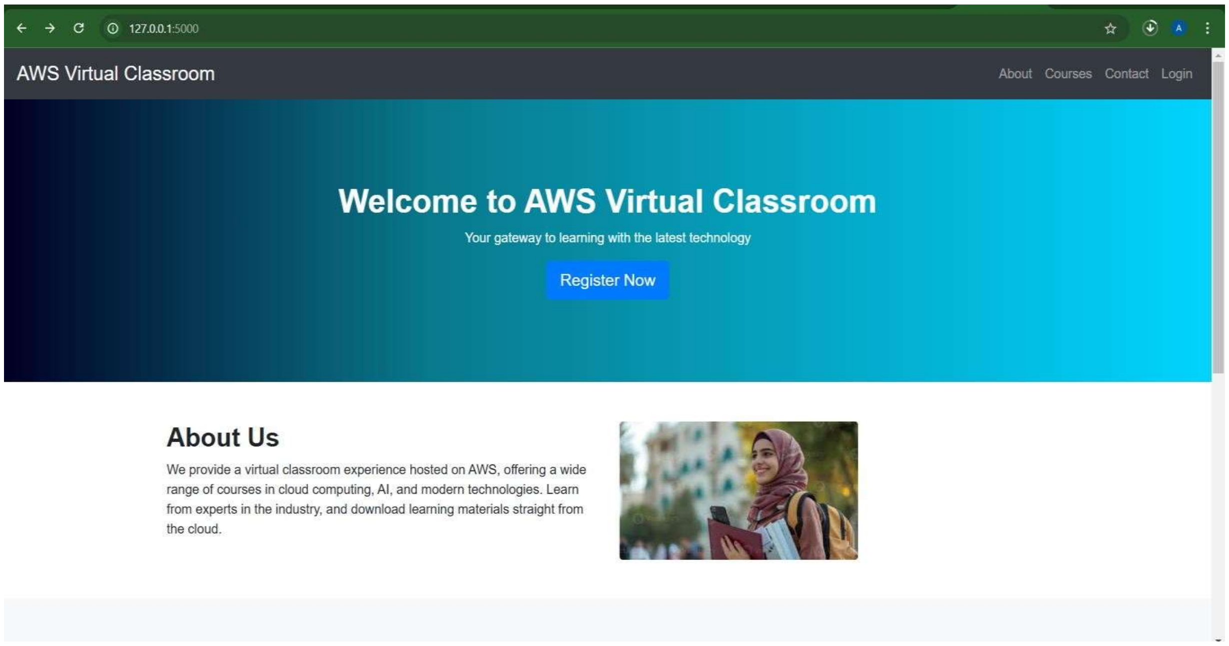
```
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\archa\Downloads\azam_aws_prjct-main> python -m venv venv
PS C:\Users\archa\Downloads\azam_aws_prjct-main> venv\scripts\activate
(venv) PS C:\Users\archa\Downloads\azam_aws_prjct-main> pip install -r requirements.txt
ERROR: Could not open requirements file: [Errno 2] No such file or directory: 'requirem
ents.txt'
```



```
  File "C:\Users\archa\Downloads\azam_aws_prjct-main\azam_aws_prjct-main\app.py", line 27, in register
    cursor.execute('INSERT INTO users (username, password_hash) VALUES (%s, %s)', (username, hashed_password))
  File "C:\Users\archa\AppData\Local\Programs\Python\Python312\Lib\site-packages\mysql\connector\cursor.py", line 568
, in execute
    self._handle_result(self._connection.cmd_query(stmt))
  File "C:\Users\archa\AppData\Local\Programs\Python\Python312\Lib\site-packages\mysql\connector\connection.py", line
 846, in cmd_query
    result = self._handle_result(self._send_cmd(ServerCmd.QUERY, query))
  File "C:\Users\archa\AppData\Local\Programs\Python\Python312\Lib\site-packages\mysql\connector\connection.py", line
 656, in _handle_result
    raise errors.get_exception(packet)
mysql.connector.errors.ProgrammingError: 1146 (42S02): Table 'course_app.users' doesn't exist
127.0.0.1 - - [06/Dec/2024 00:32:14] "GET /register?__debugger__=yes&cmd=resource&f=style.css HTTP/1.1" 200 -
127.0.0.1 - - [06/Dec/2024 00:32:14] "GET /register?__debugger__=yes&cmd=resource&f=debugger.js HTTP/1.1" 200 -
127.0.0.1 - - [06/Dec/2024 00:32:14] "GET /register?__debugger__=yes&cmd=resource&f=ubuntu.ttf HTTP/1.1" 200 -
127.0.0.1 - - [06/Dec/2024 00:32:14] "GET /register?__debugger__=yes&cmd=resource&f=console.png HTTP/1.1" 200 -
127.0.0.1 - - [06/Dec/2024 00:32:14] "GET /register?__debugger__=yes&cmd=resource&f=console.png HTTP/1.1" 200 -
127.0.0.1 - - [06/Dec/2024 00:32:38] "POST /register HTTP/1.1" 302 -
127.0.0.1 - - [06/Dec/2024 00:32:38] "GET /login HTTP/1.1" 200 -
127.0.0.1 - - [06/Dec/2024 00:32:58] "POST /login HTTP/1.1" 302 -
127.0.0.1 - - [06/Dec/2024 00:32:58] "GET /dashboard HTTP/1.1" 200 -
PS C:\Users\archa\Downloads\azam_aws_prjct-main\azam_aws_prjct-main> python app.py
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 950-312-977
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Step 7 : Display the home page of the website .


127.0.0.1:5000

AWS Virtual Classroom

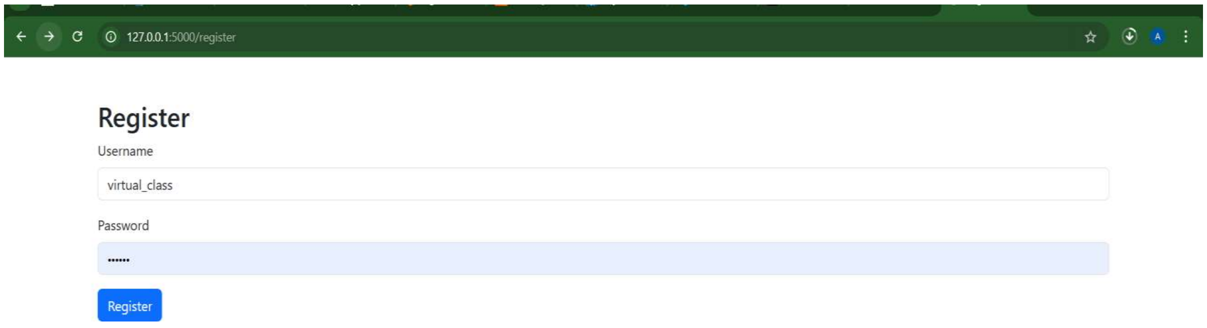
Welcome to AWS Virtual Classroom

Your gateway to learning with the latest technology

About Us

We provide a virtual classroom experience hosted on AWS, offering a wide range of courses in cloud computing, AI, and modern technologies. Learn from experts in the industry, and download learning materials straight from the cloud.

Step 8: Student Registration and Access

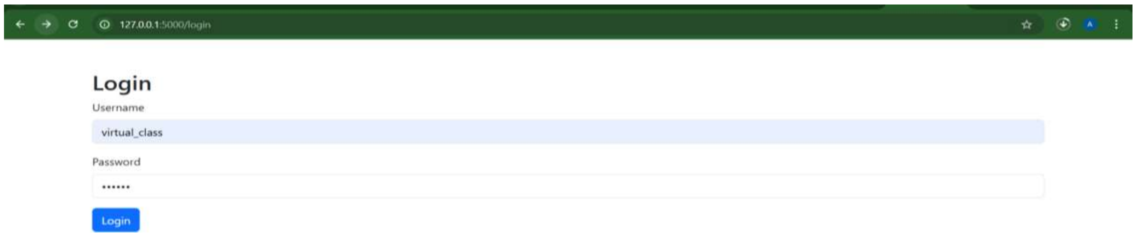

127.0.0.1:5000/register

Register

Username: virtual_class

Password: ••••••

Step 9 : User login

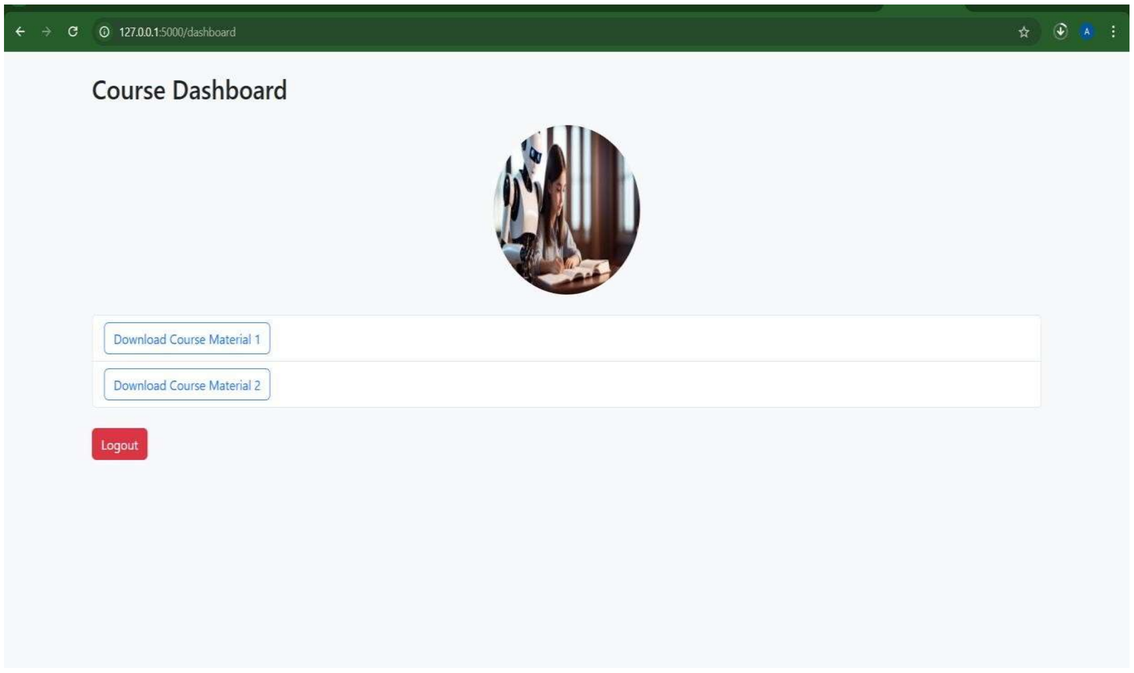

127.0.0.1:5000/login

Login

Username: virtual_class

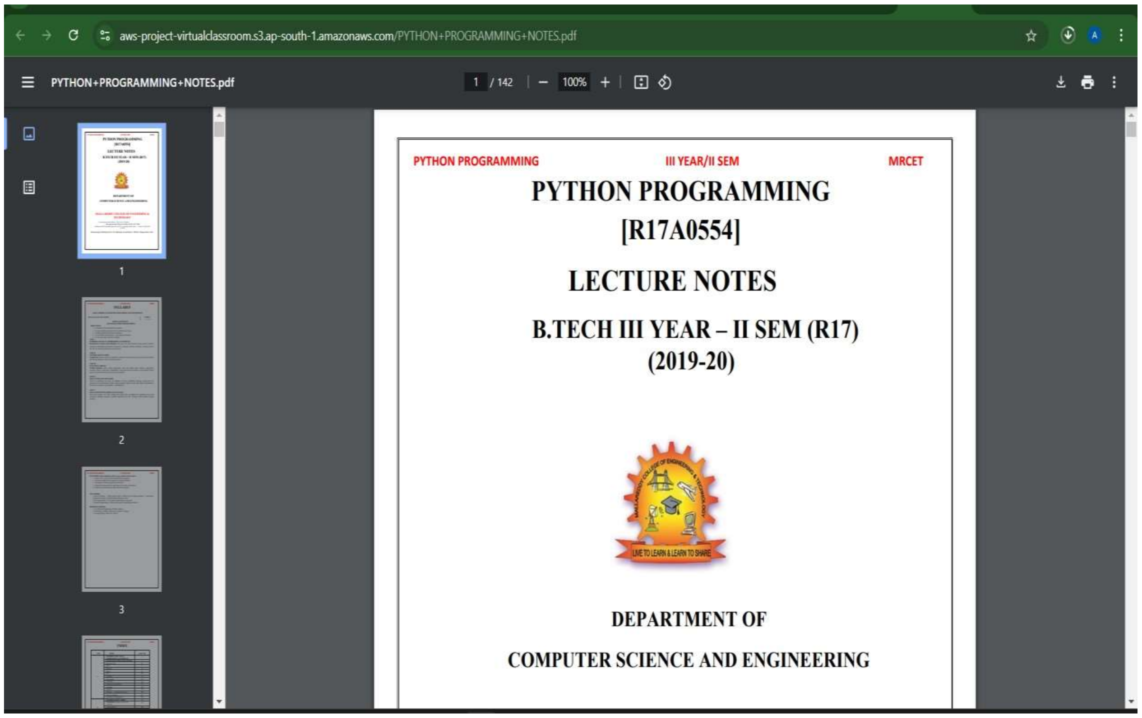
Password: ••••••

## Step 10 : Instructor Uploads Materials


127.0.0.1:5000/dashboard

Course Dashboard

## Step 11: Downloading Course Content


aws-project-virtualclassroom.s3.ap-south-1.amazonaws.com/PYTHON+PROGRAMMING+NOTES.pdf

PYTHON+PROGRAMMING+NOTES.pdf

PYTHON PROGRAMMING III YEAR/II SEM MRCET

PYTHON PROGRAMMING
[R17A0554]

LECTURE NOTES

B.TECH III YEAR – II SEM (R17)
(2019-20)

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

## SUMMARY:

- The project integrates Flask with AWS cloud services to build a scalable, secure, and user-friendly virtual classroom and learning platform. It leverages the flexibility of Flask for backend development and the robust capabilities of AWS for cloud hosting and data management. The platform allows users to register, log in, and access educational content hosted in Amazon S3, ensuring secure and high-availability storage.
- Data is managed in a relational database set up on Amazon RDS, which provides a scalable and reliable foundation for handling user data, course metadata, and other critical information. The application is deployed on Amazon EC2 instances, ensuring consistent performance and flexibility to scale resources based on user demand. To secure interactions, AWS Identity and Access Management (IAM) enforces role-based access controls, and VPC configurations ensure data protection.
- The solution incorporates key features such as real-time user authentication, course material management, and role-based dashboards tailored for students, educators, and administrators. The architecture includes load balancing and automated backup solutions, ensuring system reliability and data integrity. Additionally, AWS CloudWatch monitors system performance, and enhanced monitoring tools provide insights to improve operational efficiency.
- Designed to address the growing demand for remote and flexible education solutions, this platform combines AWS's powerful infrastructure with Flask's simplicity to deliver an intuitive and robust learning experience. Its modular design enables future enhancements, such as AI-driven learning recommendations, mobile applications, and advanced analytics. Overall, the platform offers a seamless and secure educational environment for modern e-learning needs.