



**PANIMALAR  
ENGINEERING COLLEGE**  
**CHENNAI CITY CAMPUS**  
Affiliated to Anna University, Chennai  
(JAISAKTHI EDUCATIONAL TRUST)



---

# **AWS-HOSTED VIRTUAL CLASSROOM AND LEARNING PLATFORM**

## **NAAN MUDHALVAN- AWS**

### **A PROJECT REPORT**

*Submitted by*  
**SHAMEETHA RAVIKUMAR**

*in partial fulfillment for the award of the degree  
of*

**BACHELOR OF TECHNOLOGY  
IN**

**ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**PANIMALAR ENGINEERING COLLEGE CHENNAI CITY CAMPUS  
ANNA UNIVERSITY : CHENNAI 600 025**

**November 2024**

SHAMEETHA RAVIKUMAR  
BTECH AIDS  
ROLL NO: 2022PCCADS133  
REG NO: 313522243094

## ABSTRACT

The AWS-hosted Virtual Classroom and Learning Platform project aims to harness the power of cloud technologies to provide a comprehensive, scalable, and secure virtual learning environment. In today's rapidly evolving digital education landscape, creating a reliable and accessible platform is essential to meet the growing demand for remote learning. This project is designed to address this need by building a cloud-native educational platform that integrates several core AWS services to deliver a seamless and interactive learning experience.

The platform's backend is developed using Flask, a lightweight and flexible web framework known for its simplicity and adaptability. Hosting this backend on an AWS EC2 instance ensures a robust, customizable, and scalable deployment that can accommodate varying workloads, allowing the platform to scale in response to user demand. Course materials, media files, and other educational content are stored in Amazon S3, leveraging its high availability and durability to provide secure and efficient access to resources.

For data management, Amazon RDS (MySQL) serves as the primary database, storing critical information such as user registration and login data, which are essential for user account management and platform functionality. The RDS database is configured to optimize performance and ensure data integrity, enhancing the reliability of the user experience.

Together, these components create a cohesive virtual learning environment that is both resilient and adaptable. By integrating Amazon's cloud infrastructure, the project not only facilitates the efficient management of resources but also ensures a high level of security, data accessibility, and ease of scalability. This virtual classroom setup exemplifies the potential of cloud-based solutions in supporting modern educational initiatives and represents a significant step forward in the delivery of online education.

# TABLE OF CONTENTS

<b>Chapter No.</b>	<b>Title</b>	<b>Page No.</b>
1.	<b>Abstract</b>	2
2.	<b>Introduction</b>	4
3.	<b>AWS Overview</b> <ul style="list-style-type: none"><li>• <b>Amazon Services Used</b><ul style="list-style-type: none"><li>- <b>Amzazon S3</b></li><li>- <b>Amazon EC2</b></li><li>- <b>Amazon RDS</b></li></ul></li></ul>	5
4.	<b>Steps Involved</b> <ul style="list-style-type: none"><li>• <b>Create AWS Account</b></li><li>• <b>AWS Resources</b><ul style="list-style-type: none"><li>- <b>Amzazon S3</b></li><li>- <b>Amazon EC2</b></li><li>- <b>Amazon RDS</b></li></ul></li><li>• <b>Environment Setup</b></li><li>• <b>Project Directory Structure</b></li><li>• <b>Explanation of Each Component</b></li><li>• <b>Cloning Repository</b></li><li>• <b>Environment and Dependencies</b><ul style="list-style-type: none"><li>- <b>Initial Commands for EC2 Setup</b></li><li>- <b>Required Packages</b></li></ul></li><li>• <b>Key File Conversion and Access Using PuTTY</b><ul style="list-style-type: none"><li>- <b>Convert PEM to PPK and Vice Versa with PuTTYgen</b></li><li>- <b>SSH Authentication with PuTTY</b></li><li>- <b>Using SCP for File Transfers</b></li></ul></li><li>• <b>Upload Main Files and Directories to EC2</b></li><li>• <b>Installing Dependencies</b></li><li>• <b>Running the Flask Application</b></li><li>• <b>Website Access</b></li></ul>	7
5.	<b>Conclusion</b>	22
6.	<b>References</b>	23

# INTRODUCTION

The rapid advancement of digital technology has revolutionized the way education is delivered, opening up new avenues for interactive, accessible, and scalable learning experiences. Cloud computing has emerged as a powerful enabler in this transformation, offering tools and infrastructure to develop flexible and robust educational platforms. In response to this need, the "AWS-Hosted Virtual Classroom and Learning Platform" project was developed, leveraging Amazon Web Services (AWS) to create a cloud-based, scalable, and secure educational environment.

This project focuses on building an integrated virtual classroom that uses AWS services such as Amazon S3, EC2, and RDS (MySQL) to facilitate seamless delivery and management of course content, user data, and other essential functionalities. The backend of this application is powered by Flask, a lightweight and versatile web framework, which allows for the creation of interactive features like user registration, login, and course material access.

Designed for students, educators, and administrators, the platform aims to enhance the learning experience by providing an easily accessible, centralized location for course resources, supporting asynchronous and self-paced learning. Through this project, users can manage learning content, perform registration, access materials, and benefit from a streamlined online learning system backed by AWS's reliability, security, and scalability.

# AWS OVERVIEW

Amazon Web Services (AWS) is a comprehensive cloud platform offering a vast range of services designed to help developers build scalable, secure, and cost-effective solutions. AWS provides infrastructure and tools that allow you to store data, host applications, and manage resources efficiently, making it ideal for creating applications like the Virtual Classroom platform.

## AWS Services Used

### 1. Amazon EC2 (Elastic Compute Cloud)

Amazon EC2 provides resizable virtual servers (instances) that host applications and services in the cloud. EC2 instances are flexible and can be configured with different amounts of memory, processing power, and storage, allowing you to run anything from simple applications to complex software solutions. In your project, EC2 is used to:

- Host the Flask application that serves as the main interface for users.
- Run Python scripts that manage backend processes, such as user authentication, registration, and course content display.
- Allow for real-time interaction, so users can log in, view, and access course materials.

**Integration:** You created an EC2 instance, configured it to run your Flask application, and exposed it to the internet using a public IP, making the Virtual Classroom accessible to users.

### 2. Amazon S3 (Simple Storage Service)

Amazon S3 is a storage service optimized for high durability, availability, and scalability. It is commonly used to store files such as images, documents, backups, and other static content. For the Virtual Classroom, S3 is utilized to:

- Store course materials, including documents, videos, and images.
- Provide a centralized location for user-uploaded content, which can be accessed by the Flask application.
- Manage large amounts of content efficiently without impacting EC2 storage.

**Integration:** You created an S3 bucket and configured your Flask application to retrieve course materials from this bucket. This setup allows for easy access to content and streamlines file management.

### 3. Amazon RDS (Relational Database Service)

Amazon RDS is a managed relational database service that supports popular databases like MySQL, PostgreSQL, and Oracle. RDS simplifies database administration by handling tasks such as backups, software patching, and scaling. In the Virtual Classroom project, RDS:

- Stores user information, such as usernames and hashed passwords, for authentication.

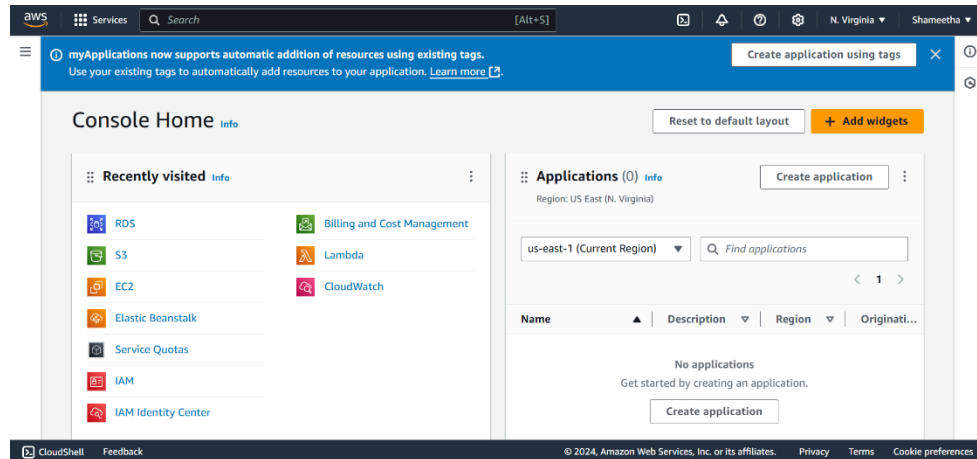
- Manages data related to courses, user interactions, and other app-specific information.
- Ensures secure and reliable data access with automatic backups and high availability.

**Integration:** You created an RDS MySQL instance, connected it to the Flask application on EC2, and used it to manage user data securely. With this setup, all database queries are routed through RDS, making the system more robust and maintainable.

# STEPS INVOLVED

## Create an AWS Account

- **Action:** Sign up for an AWS account if you don't already have one.

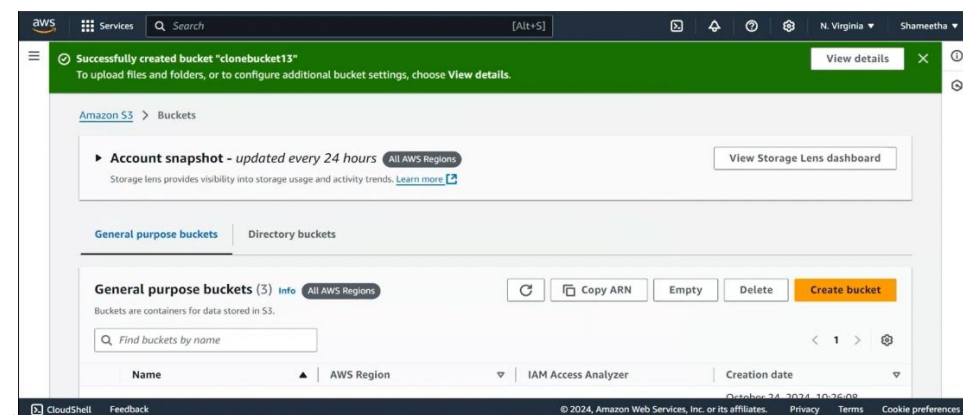


Create Account I

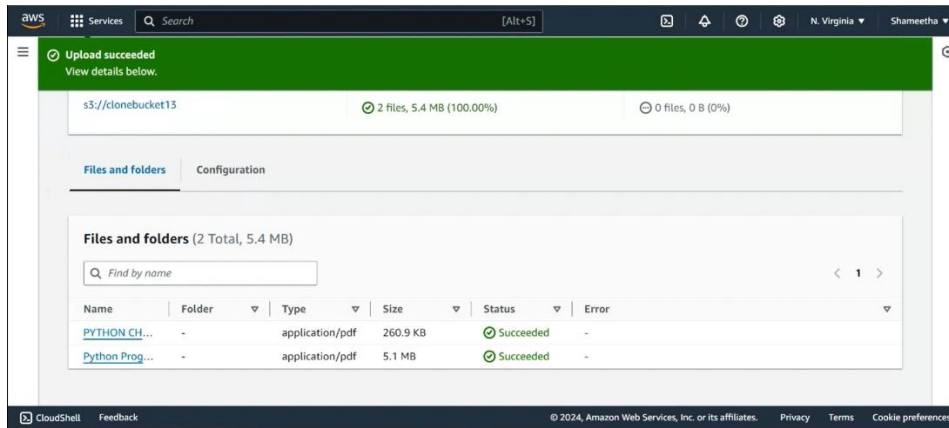
## AWS Resources Created

### 1. S3 Bucket

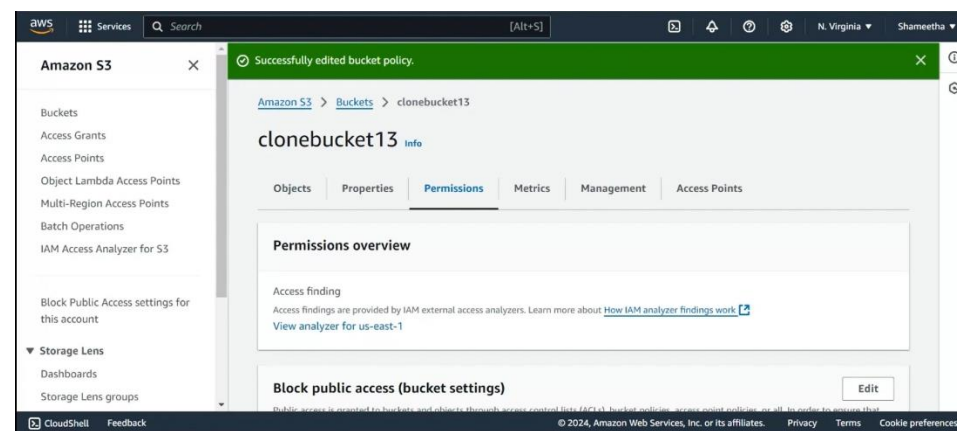
- **Name:** cloneBucket13
- **Purpose:** Store course materials and user-uploaded content for the platform



Create S3 Bucket 1



Upload Files in Bucket 1

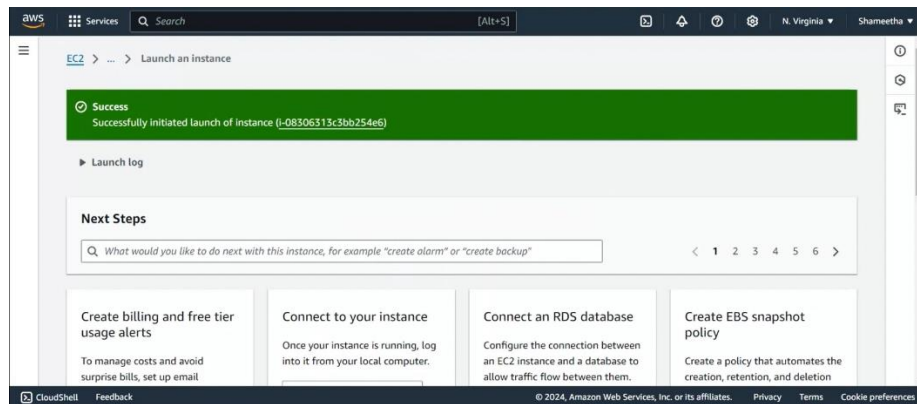


Update Bucket Policy 1

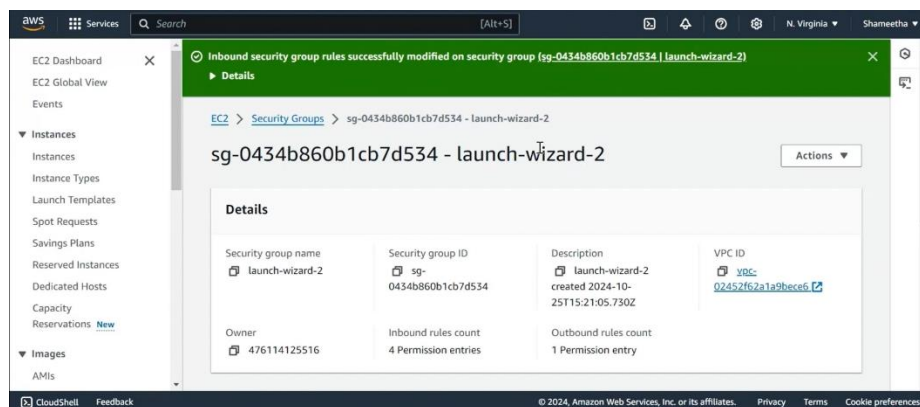
## 2. EC2 Instance

- **Name:** clone\_ec
- **Instance ID:** ec2-user@18.234.190.159
- **AMI:** Amazon Linux 2
- **Access Commands:**
  - `ssh -i "C:\Users\Lenovo\Downloads\clone.pem" ec2-user@18.234.190.159`
  - `ssh -i "Downloads/clone.pem" ec2-user@18.234.190.159`
- **Setup Commands:**
  - `source venv/bin/activate` – Activates the virtual environment.
  - `cd <git_repository_name>` – Navigates to the project repository.
  - `python3 app.py` – Runs the Flask application.
- **Purpose:** Host the Flask application and facilitate user interaction via web services.

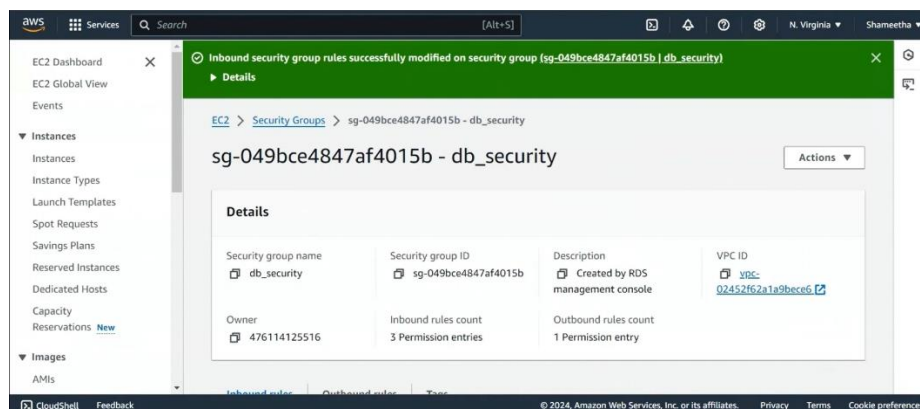




### Create EC2 Instance 1



### Modify Inbound Security Group Rules 1

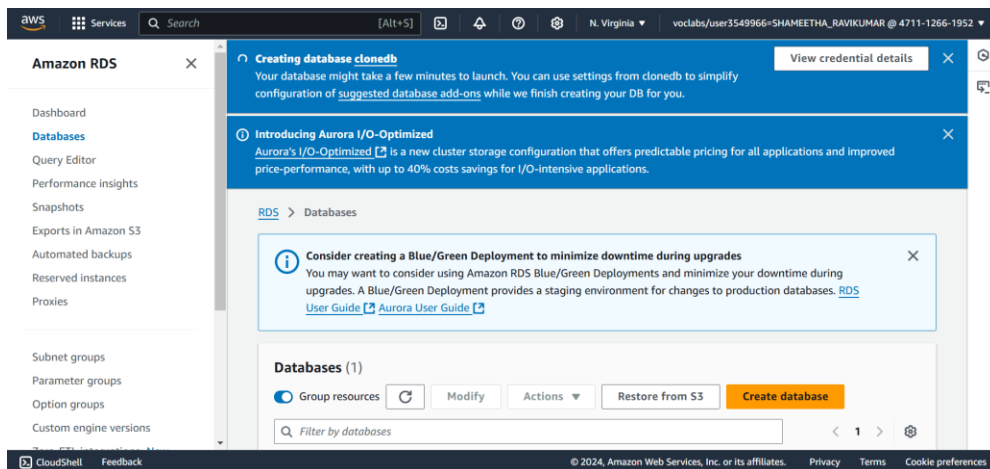


### Modify Inbound Security Group Rules 2

### 3. RDS MySQL Database

- **Name:** clonedb
- **Endpoint:** clonedb.c56g4kmq4b38.us-east-1.rds.amazonaws.com
- **Connection Command:**
  - `mysql -h clonedb.c56g4kmq4b38.us-east-1.rds.amazonaws.com -P 3306 -u admin1 -p`

**Purpose:** Store user-related data, such as registration and login information, managed using MySQL.



*Create RDS Database 1*

---

## MySQL Configuration and Database Setup

- **MySQL Connection Command:**

```
mysql -h clonedb.c56g4kmq4b38.us-east-1.rds.amazonaws.com -P 3306 -u admin1 -p
```

- **Database Creation:**

```
CREATE DATABASE clone_db;  
USE clone_db;
```

- **Table Creation:**

```
CREATE TABLE users (  
    username VARCHAR(100) PRIMARY KEY,  
    password_hash VARCHAR(255)  
);
```

- **Connect to MySQL Workbench:**
  1. Open MySQL Workbench and create a new connection.
  2. Enter the endpoint, port, username, and password of your RDS instance.
  3. Connect and manage the database.

```
C:\Windows\system32\cmd.exe - mysql -h clonedb.c56g4kmq4b38.us-east-1.rds.amazonaws.com -P 3306 -u admin1 -p  
Microsoft Windows [Version 10.0.19045.5011]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\Lenovo>mysql -h clonedb.c56g4kmq4b38.us-east-1.rds.amazonaws.com -P 3306 -u admin1 -p  
Enter password: *****  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 64  
Server version: 8.0.39 Source distribution  
  
Copyright (c) 2000, 2024, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> create database clone_db  
-> ;  
Query OK, 1 row affected (0.30 sec)  
  
mysql> use clone_db;  
Database changed  
mysql> create table users(username varchar(100) primary key,password_hash varchar (255));  
Query OK, 0 rows affected (0.29 sec)
```

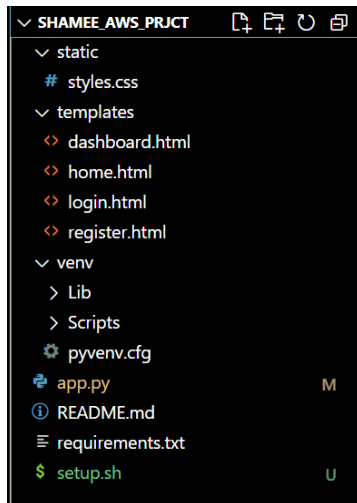
*Configure MySQL and Create Database 1*

---

## Environment Setup

- **EC2 Instance Configuration**
  - Launched an EC2 instance (clone\_ec) using Amazon Linux 2.
  - Created and activated a virtual environment to isolate dependencies.
- `ssh -i "path/to/your_key.pem" ec2-user@<EC2-Public-IP>`





Project Directory Structure 1

---

## Explanation of Each Component

- **shamee\_aws\_project/**: The root directory for your project.
  - **app.py**: The main application file where your Flask application is defined.
  - **requirements.txt**: A file listing all the dependencies required for your application, which can be installed using pip.
  - **static/**: This directory contains static files like CSS, JavaScript, and images. Static files are served directly to the client.
    - **styles.css**: The CSS file for styling your web application.
  - **templates/**: This directory holds all the HTML template files used in your Flask application, allowing for dynamic web page rendering.
    - **dashboard.html**: The HTML template for the dashboard page.
    - **home.html**: The HTML template for the home page.
    - **login.html**: The HTML template for the login page.
    - **register.html**: The HTML template for the registration page.
  - **venv/**: This directory contains the Python virtual environment for your project, where all the project-specific dependencies are installed.

---

## Cloning Repository

### 1. Navigate to Your Desired Directory

```
cd /path/to/your/directory
```

### 2. Clone the Repository

```
git clone https://github.com/Shameetha13/aws_vc.git
```

### 3. Change to the Cloned Directory

```
cd aws_vc
```

### 4. Verify the Remote Origin

```
git remote -v
```

### 5. Set the Main Origin (if necessary)

```
git remote set-url origin https://github.com/Shameetha13/aws_vc.git
```

### 6. Push Changes to the Origin

#### 1. Add Changes:

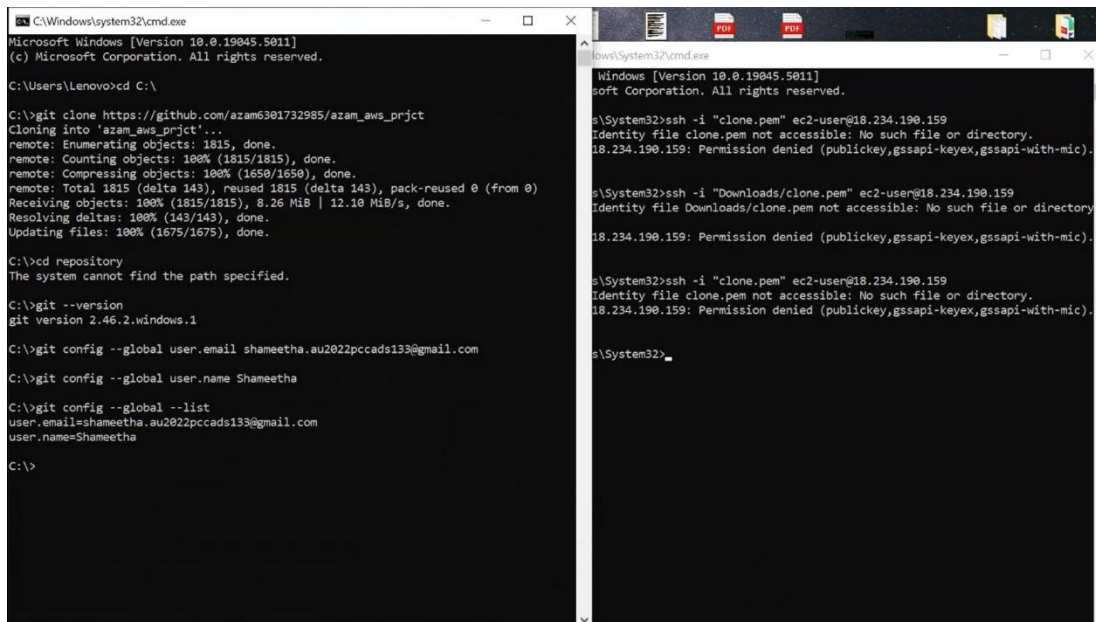
```
git add .
```

#### 2. Commit Changes:

```
git commit -m "Your commit message"
```

#### 3. Push Changes:

```
git push origin main
```



The image shows two terminal windows side-by-side. The left window is titled 'C:\Windows\system32\cmd.exe' and shows the following commands and output:

```
C:\Users\Lenovo>cd C:\
C:\>git clone https://github.com/azam6301732985/azam_aws_prjct
Cloning into 'azam_aws_prjct'...
remote: Enumerating objects: 1815, done.
remote: Counting objects: 100% (1815/1815), done.
remote: Compressing objects: 100% (1650/1650), done.
remote: Total 1815 (delta 143), reused 1815 (delta 143), pack-reused 0 (from 0)
Receiving objects: 100% (1815/1815), 8.26 MiB | 12.10 MiB/s, done.
Resolving deltas: 100% (143/143), done.
Updating files: 100% (1675/1675), done.

C:\>cd repository
The system cannot find the path specified.

C:\>git --version
git version 2.46.2.windows.1

C:\>git config --global user.email shameetha.au2022pccads133@gmail.com

C:\>git config --global user.name Shameetha

C:\>git config --global --list
user.email=shameetha.au2022pccads133@gmail.com
user.name=Shameetha

C:\>
```

The right window is titled 'aws\System32\cmd.exe' and shows the following commands and output:

```
aws\System32>ssh -i "clone.pem" ec2-user@18.234.190.159
Identity file clone.pem not accessible: No such file or directory.
18.234.190.159: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).

aws\System32>ssh -i "Downloads/clone.pem" ec2-user@18.234.190.159
Identity file Downloads/clone.pem not accessible: No such file or directory.
18.234.190.159: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).

aws\System32>ssh -i "clone.pem" ec2-user@18.234.190.159
Identity file clone.pem not accessible: No such file or directory.
18.234.190.159: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).

aws\System32>
```

*Clone Repository 1*

# Environment and Dependencies

## 1. Initial Commands for EC2 Setup

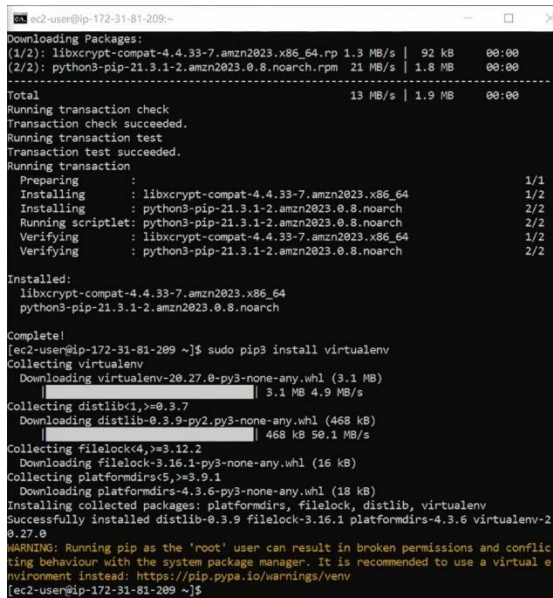
Run these commands only during the first-time setup:

```
sudo yum update -y
sudo yum install python3 -y
sudo yum install python3-pip -y
sudo pip3 install virtualenv
python3 -m venv venv
source venv/bin/activate
pip install flask
sudo yum install git -y
git clone https://github.com/Shameetha13/aws_vc
cd aws_vc
pip install mysql-connector-python
pip install -r requirements.txt
python app.py
```

## 2. Required Packages

Install dependencies with:

```
pip install mysql-connector-python
pip install -r requirements.txt
```



```
ec2-user@ip-172-31-81-209:~$ sudo yum update -y
Download Packages:
(1/2): libxcrypt-compat-4.4.33-7.amzn2023.x86_64.rpm 1.3 MB/s | 92 kB 00:00
(2/2): python3-pip-21.3.1-2.amzn2023.0.8.noarch.rpm 21 MB/s | 1.8 MB 00:00
-----
Total 13 MB/s | 1.9 MB 00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      : libxcrypt-compat-4.4.33-7.amzn2023.x86_64 1/1
  Installing     : python3-pip-21.3.1-2.amzn2023.0.8.noarch 1/2
  Installing     : python3-pip-21.3.1-2.amzn2023.0.8.noarch 2/2
  Running scriptlet: python3-pip-21.3.1-2.amzn2023.0.8.noarch 2/2
  Verifying      : libxcrypt-compat-4.4.33-7.amzn2023.x86_64 1/2
  Verifying      : python3-pip-21.3.1-2.amzn2023.0.8.noarch 2/2

Installed:
  libxcrypt-compat-4.4.33-7.amzn2023.x86_64
  python3-pip-21.3.1-2.amzn2023.0.8.noarch

Complete!
[ec2-user@ip-172-31-81-209 ~]$ sudo pip3 install virtualenv
Collecting virtualenv
  Downloading virtualenv-20.27.0-py3-none-any.whl (3.1 MB)
    | 3.1 MB 4.9 MB/s
Collecting distlib<1, >=0.3.7
  Downloading distlib-0.3.9-py2.py3-none-any.whl (468 kB)
    | 468 kB 50.1 MB/s
Collecting filelock<4, >=3.12.2
  Downloading filelock-3.16.1-py3-none-any.whl (16 kB)
Collecting platformdirs<5, >=3.9.1
  Downloading platformdirs-4.3.6-py3-none-any.whl (18 kB)
Installing collected packages: platformdirs, filelock, distlib, virtualenv
Successfully installed distlib-0.3.9 filelock-3.16.1 platformdirs-4.3.6 virtualenv-20.27.0
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
[ec2-user@ip-172-31-81-209 ~]$
```

EC2 Setup 1

```
C:\Windows\System32\cmd.exe
(6/8): perl-Git-2.40.1-1.amzn2023.0.3.noarch.rpm 768 kB/s | 42 kB 00:00
(7/8): perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64 1.6 MB/s | 36 kB 00:00
(8/8): perl-lib-0.65-477.amzn2023.0.6.x86_64.rpm 888 kB/s | 15 kB 00:00
-----Total
30 MB/s | 7.1 MB 00:00

Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      : git-core-2.40.1-1.amzn2023.0.3.x86_64 1/1
  Installing     : git-core-doc-2.40.1-1.amzn2023.0.3.noarch 1/8
  Installing     : perl-lib-0.65-477.amzn2023.0.6.x86_64 2/8
  Installing     : perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64 3/8
  Installing     : perl-File-Find-1.37-477.amzn2023.0.6.noarch 4/8
  Installing     : perl-Error-1:0.17029-5.amzn2023.0.2.noarch 5/8
  Installing     : perl-Git-2.40.1-1.amzn2023.0.3.noarch 6/8
  Installing     : git-2.40.1-1.amzn2023.0.3.x86_64 7/8
  Running scriptlet: git-2.40.1-1.amzn2023.0.3.x86_64 8/8
  Verifying      : git-2.40.1-1.amzn2023.0.3.x86_64 1/8
  Verifying      : git-core-2.40.1-1.amzn2023.0.3.x86_64 2/8
  Verifying      : git-core-doc-2.40.1-1.amzn2023.0.3.noarch 3/8
  Verifying      : perl-Error-1:0.17029-5.amzn2023.0.2.noarch 4/8
  Verifying      : perl-File-Find-1.37-477.amzn2023.0.6.noarch 5/8
  Verifying      : perl-Git-2.40.1-1.amzn2023.0.3.noarch 6/8
  Verifying      : perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64 7/8
  Verifying      : perl-lib-0.65-477.amzn2023.0.6.x86_64 8/8

Installed:
git-2.40.1-1.amzn2023.0.3.x86_64
git-core-2.40.1-1.amzn2023.0.3.x86_64
git-core-doc-2.40.1-1.amzn2023.0.3.noarch
perl-Error-1:0.17029-5.amzn2023.0.2.noarch
perl-File-Find-1.37-477.amzn2023.0.6.noarch
perl-Git-2.40.1-1.amzn2023.0.3.noarch
perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64
perl-lib-0.65-477.amzn2023.0.6.x86_64

Complete!
[ec2-user@ip-172-31-81-209 ~]$ client_loop: send disconnect: Connection reset
```

## EC2 Setup 2

```
ec2-user@ip-172-31-81-209:~/aws_vc
GNU nano 5.8 requirements.txt Modified
Flask==2.0.1
mysql-connector-python==8.0.26
werkzeug==2.0.1
Flask-Bootstrap==3.3.7.1
```

## EC2 Setup 3



```
ec2-user@ip-172-31-81-209:~/aws_vc
GNU nano 5.8 app.py Modified
conn = get_db_connection()
cursor = conn.cursor()
cursor.execute('SELECT password_hash FROM users WHERE username = %s', (username,))
result = cursor.fetchone()
cursor.close()
conn.close()

if result and check_password_hash(result[0], password):
    return redirect(url_for('dashboard'))
else:
    return 'Invalid credentials', 401
return render_template('login.html')

# Dashboard Route (after login)
@app.route('/dashboard')
def dashboard():
    course_urls = [
        'https://aws-project-virtualclassroom.s3.ap-south-1.amazonaws.com/python_code',
        'https://aws-project-virtualclassroom.s3.ap-south-1.amazonaws.com/PYTHON+PR'
    ]
    return render_template('dashboard.html', course_urls=course_urls)

# Home Route (Landing Page)
@app.route('/')
def home():
    return render_template('home.html')
```

EC2 Setup 4

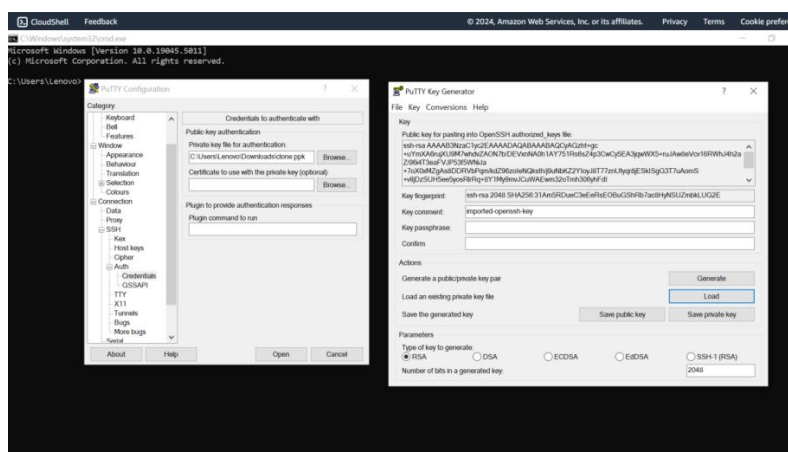
## Key File Conversion and Access Using PuTTY

### 1. Convert PEM to PPK and Vice Versa with PuTTYgen

- Open PuTTYgen, load the .pem file, then save the converted .ppk file.
- Repeat the process in reverse to convert .ppk back to .pem if necessary.

### 2. SSH Authentication with PuTTY

- Configure SSH by loading the .ppk key in PuTTY's SSH authentication settings.
- Browse for the .ppk file under Connection > SSH > Auth



PuTTY 1

### 3. Using SCP for File Transfers

Use scp to transfer files to the EC2 instance:

```
scp -i C:\Users\Lenovo\Downloads\clone.pem C:\shamee_aws_project\templates\home.html ec2-user@18.234.190.159:/home/ec2-user/aws_vc/templates/
```

Similarly, copy additional files (e.g., dashboard.html, login.html, register.html) as needed by adjusting the file path:

```
scp -i C:\Users\Lenovo\Downloads\clone.pem C:\shamee_aws_project\templates\dashboard.html ec2-user@18.234.190.159:/home/ec2-user/aws_vc/templates/
```

Upload Main Files and Directories to EC2:

- **Upload app.py**

```
scp -i C:\Users\Lenovo\Downloads\clone.pem C:\shamee_aws_project\app.py ec2-user@18.234.190.159:/home/ec2-user/aws_vc/
```

- **Upload requirements.txt**

```
scp -i C:\Users\Lenovo\Downloads\clone.pem C:\shamee_aws_project\requirements.txt ec2-user@18.234.190.159:/home/ec2-user/aws_vc/
```

- **Upload static files (e.g., styles.css)**

```
scp -i C:\Users\Lenovo\Downloads\clone.pem C:\shamee_aws_project\styles.css ec2-user@18.234.190.159:/home/ec2-user/aws_vc/static/
```

- **Upload templates (all HTML files)**

```
scp -i C:\Users\Lenovo\Downloads\clone.pem C:\shamee_aws_project\templates\*.html ec2-user@18.234.190.159:/home/ec2-user/aws_vc/templates/
```

---

## Installing Dependencies

Installed necessary Python packages using pip:

```
pip install Flask==2.0.1
mysql-connector-python==8.0.26
Flask-Bootstrap==3.3.7.1
werkzeug==2.0.1
```

```
ec2-user@ip-172-31-81-209:~/aws_vc
GNU nano 5.8 requirements.txt Modified
Flask==2.0.1
mysql-connector-python==8.0.26
werkzeug==2.0.1
Flask-Bootstrap==3.3.7.1
```

Installing Dependencies 1

## Running the Flask Application

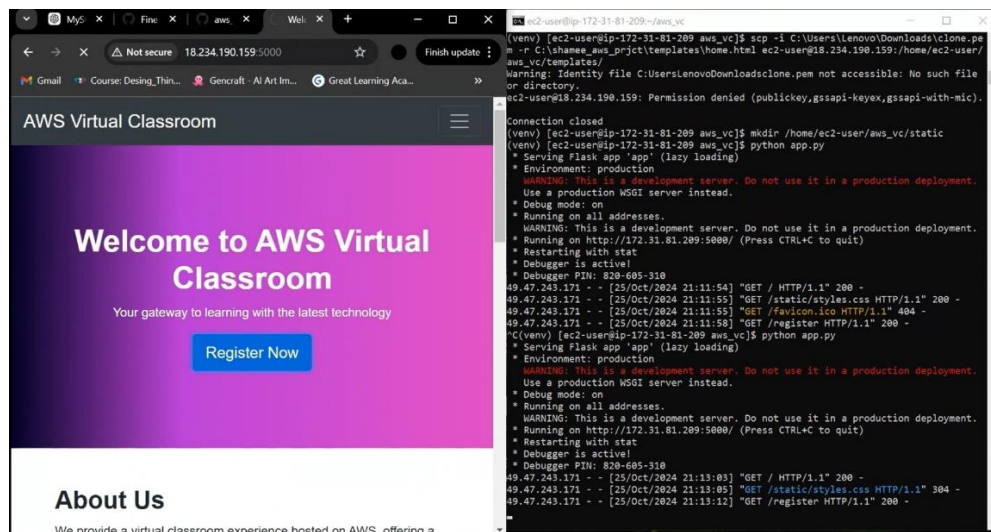
- Configured the Flask application to run on all addresses:

```
app.run(host='0.0.0.0', port=5000)
```

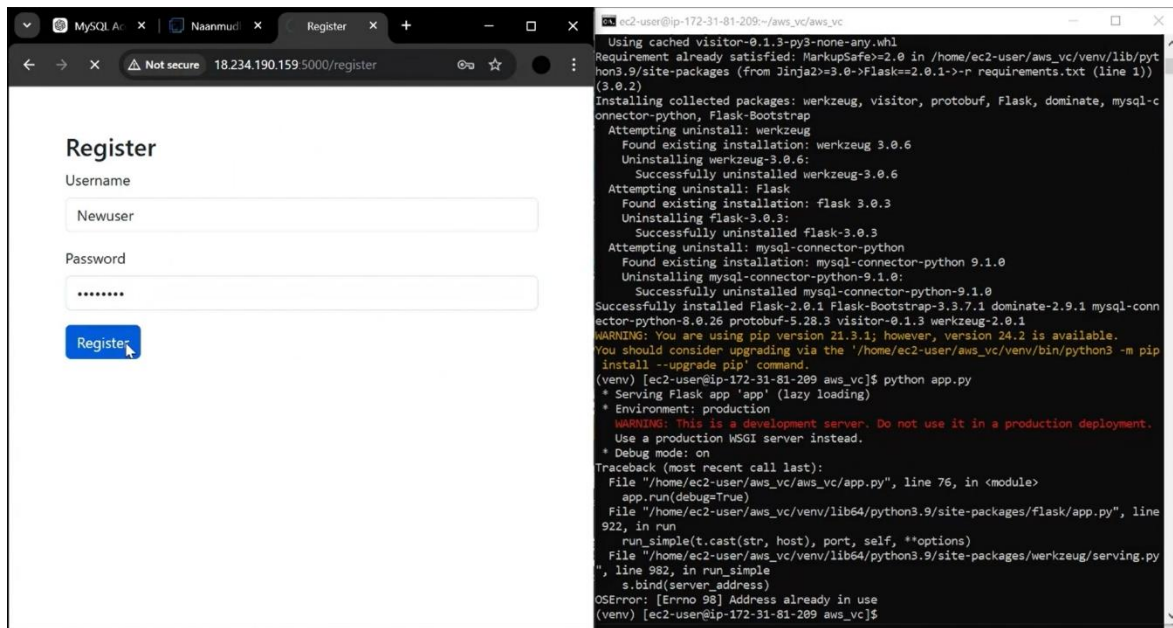
- Started the application using:

```
python app.py
```

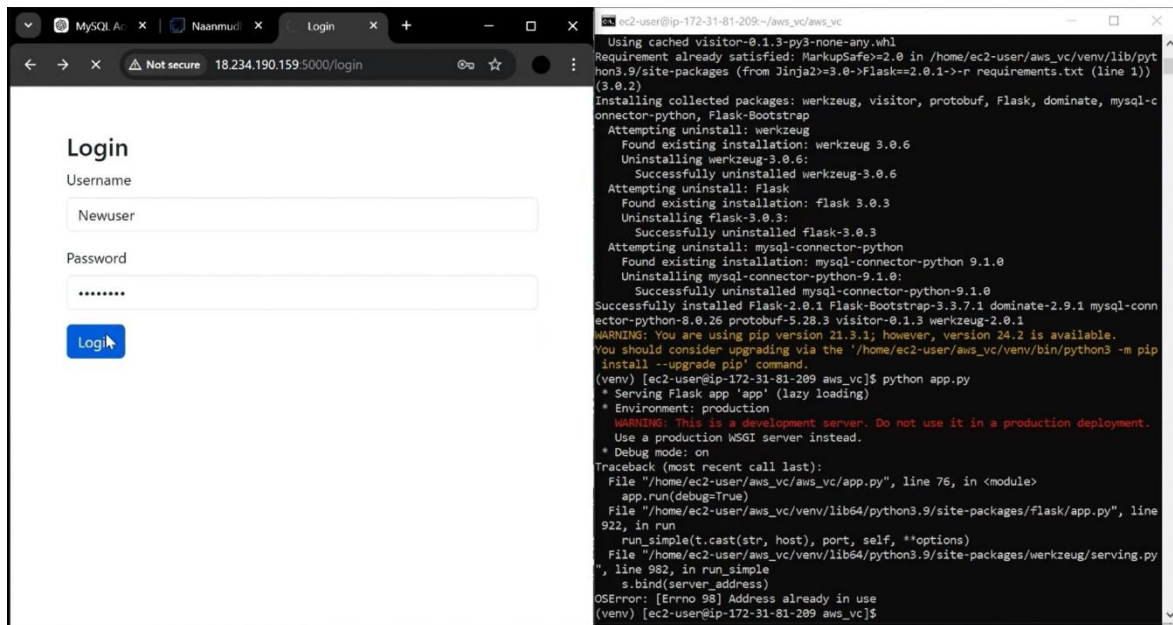
- Accessed the application at: <http://<EC2-Public-IP>:5000/>



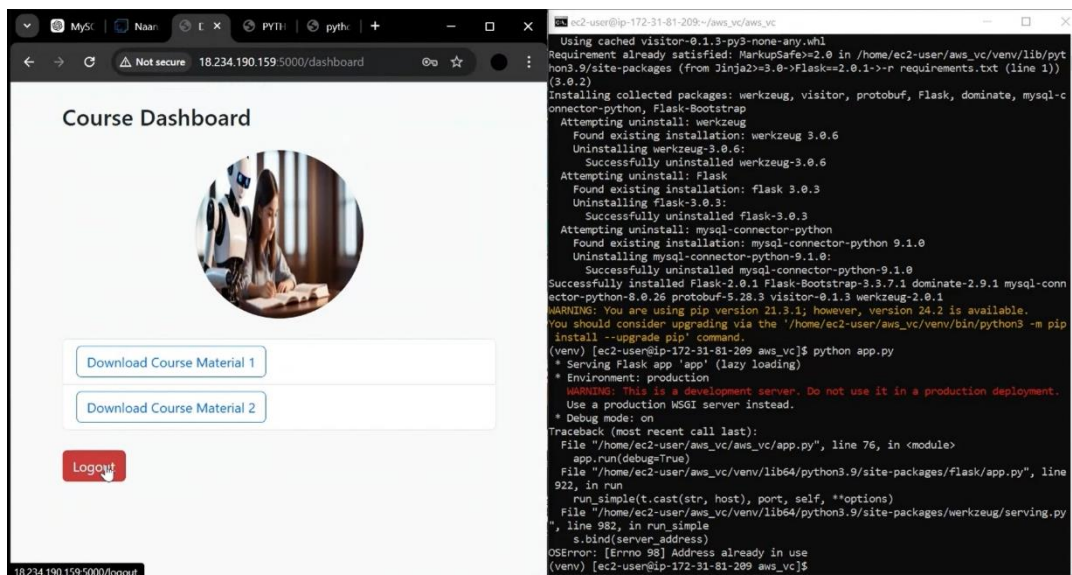
Access Application 1



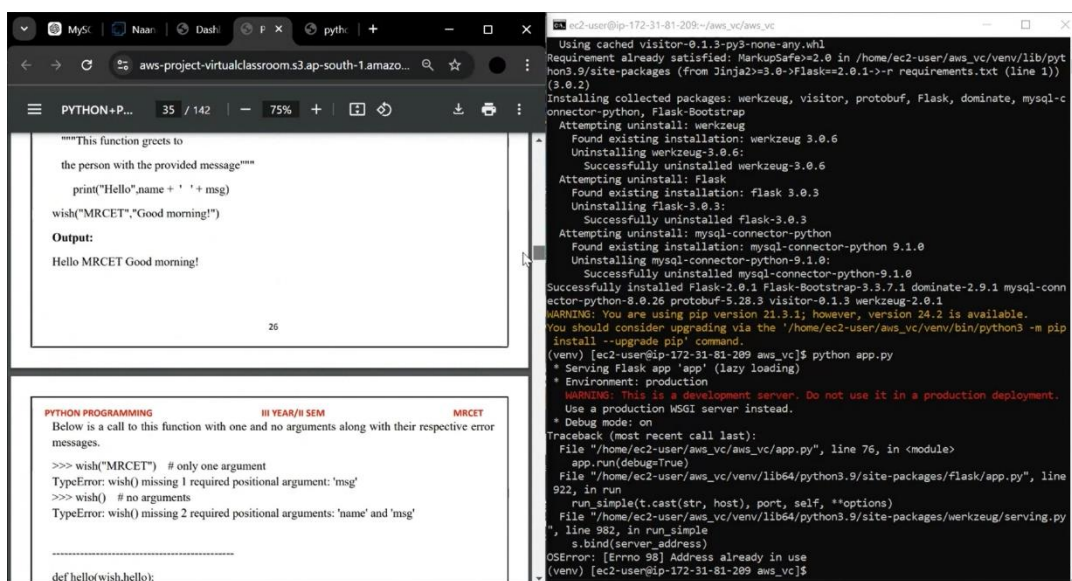
Registration In App 1



Logging In 1



Accessing Dashboard 1



Accessing the Files in Application 1

## Website Access

- URL: <http://18.234.190.159:5000>

## CONCLUSION

The AWS-hosted Virtual Classroom and Learning Platform project successfully demonstrates the integration of essential cloud services with Flask-based web application architecture to provide a scalable, accessible, and efficient learning platform. By utilizing Amazon EC2 for hosting, S3 for content storage, and RDS for managing user data, the application provides a robust backend infrastructure that ensures data security and application performance. Git and GitHub integration have streamlined version control and collaboration, while secure file transfer and key management practices, including SSH authentication and PuTTY configuration, have facilitated a secure deployment process.

Overall, this project highlights the capability of cloud-based solutions to support dynamic, real-time applications that can be expanded and adapted to meet the evolving needs of users, making it a valuable resource for virtual education.

## REFERENCES

- [https://explore.skillbuilder.aws/learn/public/learning\\_plan/view/82/cloud-foundations-learning-plan?la=cta&cta=topbanner](https://explore.skillbuilder.aws/learn/public/learning_plan/view/82/cloud-foundations-learning-plan?la=cta&cta=topbanner)
- <https://explore.skillbuilder.aws/learn/course/external/view/elearning/134/aws-cloud-practitioner-essentials>