

Deploying End-to-End Website on AWS

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of Bachelor of
Technology Degree in Information Technology

By

Shajith Ahamed
REG. NO. 39120106



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

APRIL 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI

CHENNAI- 600119

www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Shajith Ahamed (Reg. No: 39120106)** who carried out the project entitled “**Deploying End-to-End Website on AWS**” under my supervision from June 2021 to November 2021.

Internal Guide

Dr. P.Ajitha

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **Shajith Ahamed** hereby declare that the project report entitled **Deploying End-to-End Website on AWS** done by me under the guidance of **Dr.P.Ajitha** is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in Information Technology.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D, Dean, School of Computing, Dr. S. Vigneshwari, M.E., Ph.D. and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department of Information Technology and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.P.Ajitha** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology and Engineering** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE



ABSTRACT

Cyber security and secure software deployment are issues that cut across multiple sectors— aerospace, defense, energy, critical infrastructure, industrial automation, medical devices, telecommunications, and more. What these disparate

sectors have in common is that a malicious, network-borne intrusion can cause untold damage, whether financial or physical, and even threaten human safety. Fortunately, there are security-strengthening techniques and solutions for cyber security research that can apply in all of these areas. This paper explains why creating and testing security solutions is more effectively performed using virtual hardware and system simulation technology, rather than using the live systems that are being subjected to attacks.

TABLE OF CONTENTS		
CHAPTER	TITLE	PAGE NO
	ABSTRACT	
	LIST OF FIGURS	

1	INTRODUCTION	1
	1.1 CLOUD DEPLOYMENT MODEL	
	1.2 MANAGING THE RISKS OF PUBLIC CLOUDS	
	1.3 AMAZON WEB SERVICES	
2	SYSTEM SPECIFICATION	19
	2.1 HARDWARE SPECIFICATION	
	2.2 SOFTWARE SPECIFICATION	
3	SYSTEM IMPLEMENTATION	20
	3.1 SYSTEM ARCHITECTURE	
	3.2 MODULES	
	3.2.1 CREATING THE WEBSITE	
	3.2.2 SETTING UP THE AWS	
	3.2.3 CONNECTING EC2 TO OUR MACHINE AND RDS	
	3.2.4 CLONING THE WEBSITE CODE INTO EC2	
	3.2.5 DEPLOYING THE WEBSITE IN AWS CLOUD	
4	CONCLUSION	24
	REFERENCES	
	APPENDIX A: SNAP SHOTS	
	APPENDIX B: SOURCE CODE	

LIST OF FIGURES		
FIGURE NO	FIGURE NAME	PAGE NO
1.1.1	PUBLIC CLOUD	2

1.1.2	PRIVATE CLOUD	3
1.1.3	COMMUNITY CLOUD	4
1.1.4	HYBRID CLOUD	5
1.3.1	AMAZON RDS	
1.3.2	AMAZON S3	
1.3.3	AMAZON EC2	
3.1.1	ARCHITECTURE DIAGRAM	

CHAPTER 1

INTRODUCTION

1.1 CLOUD DEPLOYMENT MODEL

It works as your virtual computing environment with a choice of deployment model depending on how much data you want to store and who has access to the infrastructure.

Types of Cloud Deployment Model

Most cloud hubs have tens of thousands of servers and storage devices to enable fast loading. It is often possible to choose a geographic area to put the data “closer” to users. Thus, deployment models for cloud computing are categorized based on their location. To know which model would best fit the requirements of your organization.

Public Cloud

The name says it all. It is accessible by the public. Public deployment models in the cloud are perfect for organizations with growing and fluctuating demands. It also makes a great choice for companies with low-security concerns. Thus, you pay a cloud service provider for networking services, compute virtualization & storage available on the public internet. This is also a great delivery model for the teams with development and testing. Its configuration and deployment are quick and easy, making it an ideal choice for test environments.

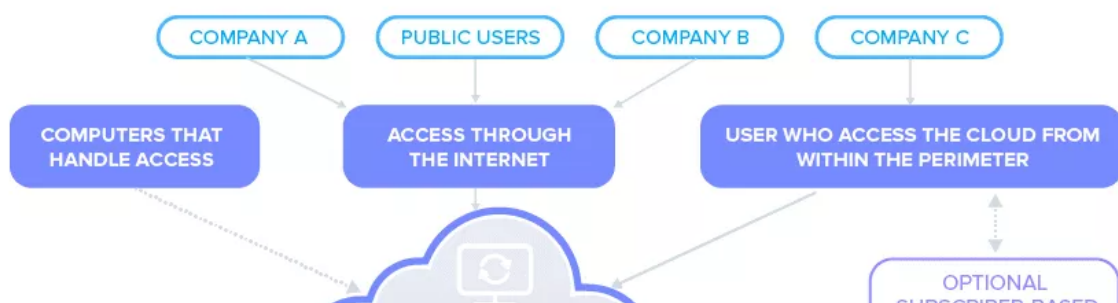
Benefits of Public Cloud

- Minimal Investment
- No Hardware Set-up
- No Infrastructure Management

Limitations of Public Cloud

- Data Security and Privacy
- Reliability Issues
- Service/License Limitation

Fig.1.1.1 Public Cloud



Private Cloud

Now that you understand what the public cloud could offer you, of course, you are keen to know what a private cloud can do. Companies that look for cost efficiency and greater control over data & resources will find the private cloud a more suitable choice. What it means is that it will be integrated with your data center and managed by your IT team. Alternatively, you can also choose to host it externally. When it comes to customization, the private cloud offers bigger opportunities that help meet specific organization's requirements. It's also a wise choice for mission-critical processes that may have frequently changing requirements.

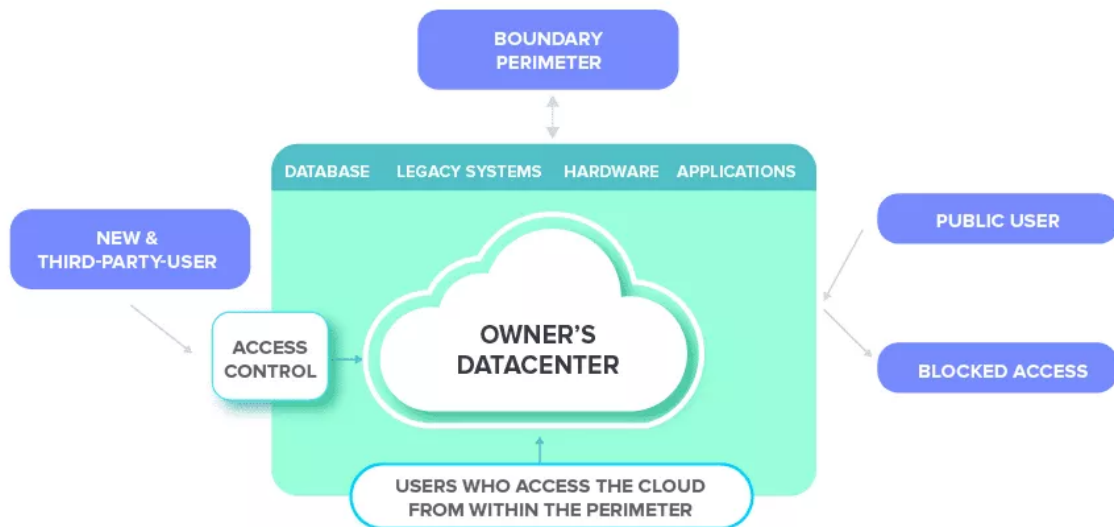


Fig.1.1.2 Private Cloud

Benefits of Private Cloud

- Data Privacy
- Security
- Supports Legacy Systems

Limitations of Private Cloud

- Higher Cost
- Fixed Scalability
- High Maintenance

Community Cloud

The community cloud operates in a way that is similar to the public cloud. There's just one difference – it allows access to only a specific set of users who share common objectives and use cases. This type of deployment model of cloud computing is managed and hosted internally or by a third-party vendor. However, you can also choose a combination of all three.

Benefits of Community Cloud

- Smaller Investment
- Setup Benefits

Limitations of Community Cloud

- Shared Resources
- Not as Popular

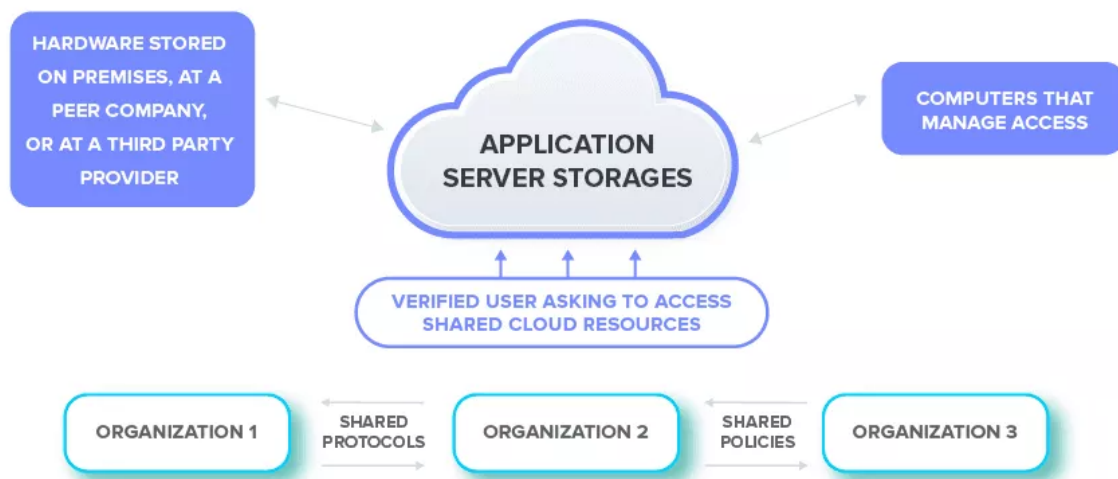


Fig.1.1.3 Community Cloud

Hybrid Cloud

As the name suggests, a hybrid cloud is a combination of two or more cloud architectures. While each model in the hybrid cloud functions differently, it is all part of the same architecture. Further, as part of this deployment of the cloud computing model, the internal, or external providers can offer resources. Let's understand the hybrid model better. A company that has critical data will prefer storing on a private cloud, while less sensitive data can be stored on a public cloud. The hybrid cloud is also frequently used for 'cloud bursting'. It means, suppose an organization runs an application on-premises, but due to heavy load, they can burst into the public cloud.

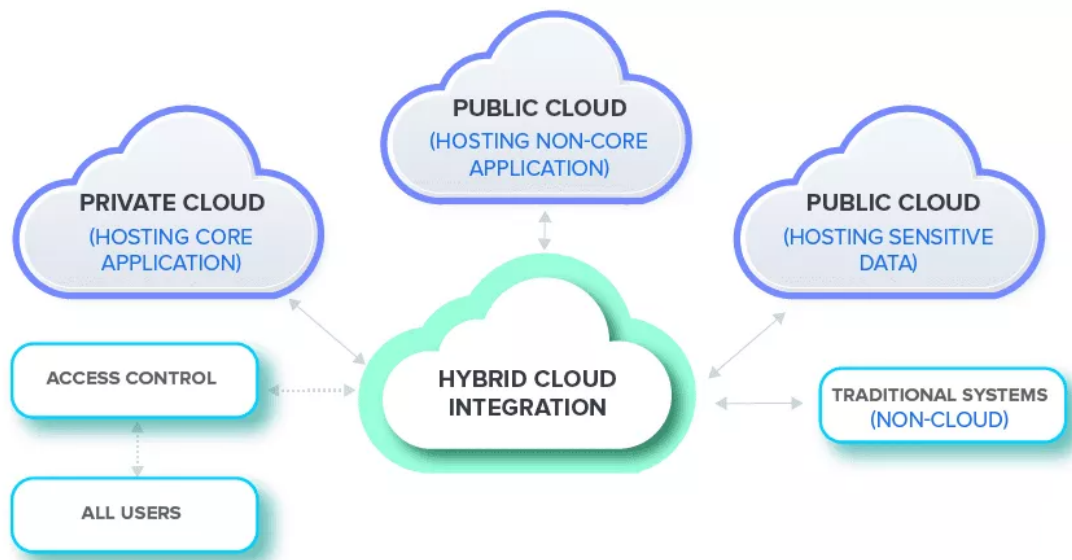


Fig.1.1.4 Hybrid Cloud

Benefits of Hybrid Cloud

- Cost-Effectiveness
- Security
- Flexibility

Limitations of Hybrid Cloud

- Complexity

- Specific Use Case

1.2 MANAGING THE RISKS OF PUBLIC CLOUDS

Although a public cloud deployment is suitable for most uses that are non sensitive, migrating sensitive, mission-critical, or proprietary data into any cloud environment that is not certified and designed for handling such data introduces high risk. A customer should first select a cloud deployment model and then make sure that sufficient security controls are in place.

These actions should be followed by a reasonable risk assessment:

- Data and encryption: If the data is stored unencrypted in the cloud, data privacy is at risk. There is the risk for unauthorized access either by a malicious employee on the cloud service provider side or an intruder gaining access to the infrastructure from the outside.
- Data retention: When the data is migrated or removed by the cloud provider or customer, there may be data residues that might expose sensitive data to unauthorized parties.
- Compliance requirements: Various countries have varying regulations for data privacy. Because some public cloud providers don't offer information about the location of the data, it is crucial to consider the legal and regulatory requirements about where data can be stored.
- Multi-tenancy risks: The shared nature of public cloud environments increases security risks, such as unauthorized viewing of data by other customers using the same hardware platform. A shared environment also presents resource competition problems whenever one of the customers uses most of the resources due either to need or to being exposed to targeted attacks, such as DDoS (distributed denial of service).
- Control and visibility: Customers have restricted control and visibility over the cloud resources because the cloud provider is responsible for administering the infrastructure. This introduces additional security concerns that originate

from the lack of transparency. Customers need to rethink the way they operate as they surrender the control of their IT infrastructure to an external party while utilizing public cloud services.

- Security responsibility: In a cloud the vendor and the user share responsibility for securing the environment. The amount of responsibility shouldered by each party can change depending on the cloud model adopted

1.3 AMAZON WEB SERVICES

Amazon Web Services (AWS) is the world's most comprehensive and broadly adopted cloud platform, offering over 200 fully featured services from data centers globally. Millions of customers- including the fastest-growing startups, largest enterprises, and leading government agencies- are using AWS to lower costs, become more agile, and innovate faster.

- Most Functionality
- Largest community of consumers and partners
- Most Secure
- Fastest pace of Innovation
- Most proven operational expertise

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

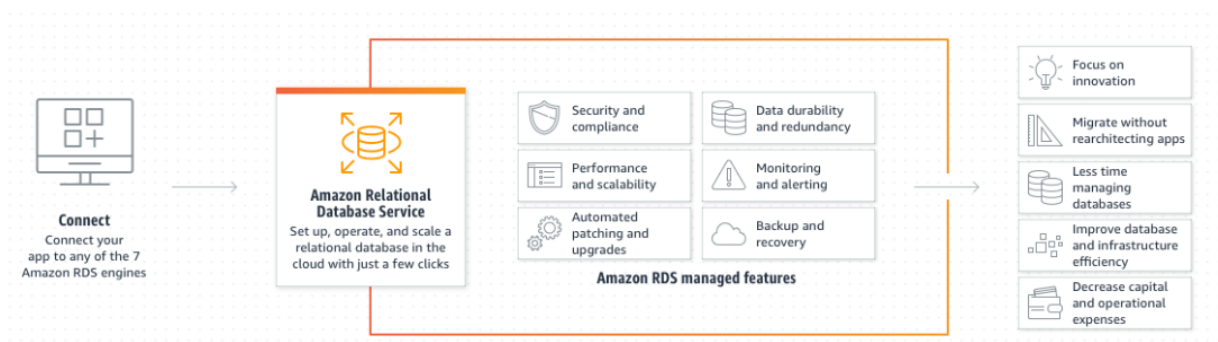


Fig.1.3.1 Amazon RDS

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can use Amazon S3 to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides management features so that you can optimize, organize, and configure access to your data to meet your specific business, organizational, and compliance requirements.

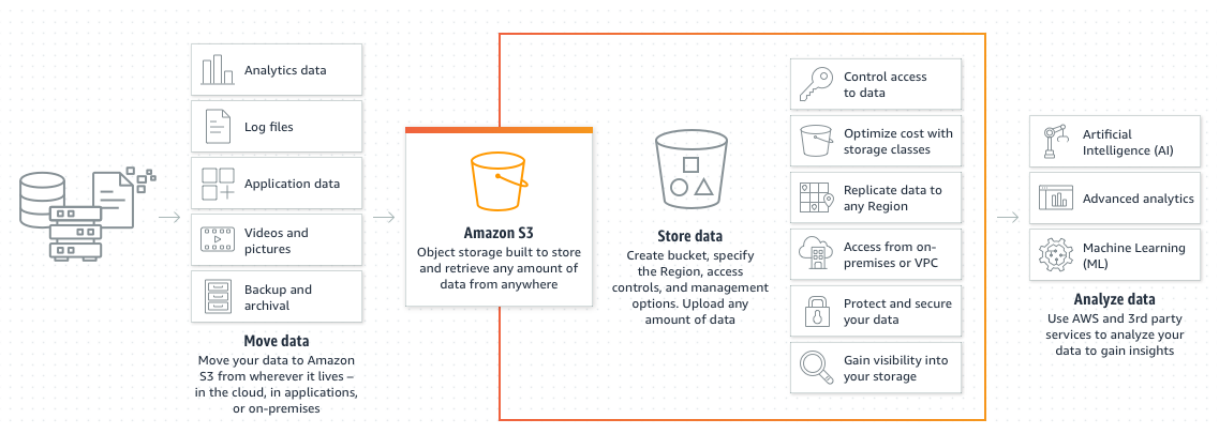


Fig.1.3.2 Amazon S3

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you

need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

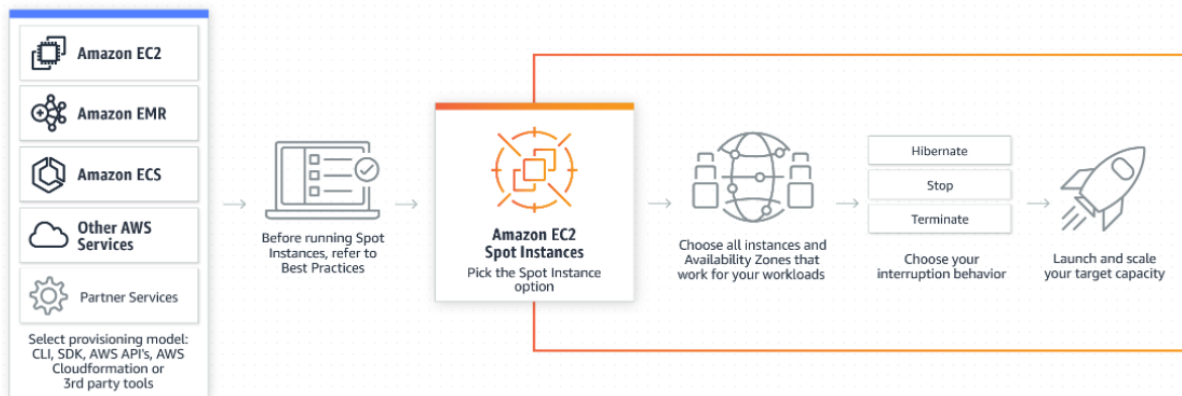


Fig.1.3.3 Amazon EC2

Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service. You can use Route 53 to perform three main functions in any combination: domain registration, DNS routing, and health checking.

CHAPTER 2

SYSTEM SPECIFICATION

2.1 HARDWARE SPECIFICATION

Processor	:	Apple M1 Chip or Intel Core i5
Internet Connection	:	Existing telephone lines, Data card, Fiber net
RAM	:	8 GB

2.2 SOFTWARE SPECIFICATION

Operating System	:	Mac, Windows, Linux
Language	:	HTML, CSS, Python
GUI	:	Ubuntu (from AWS)
CLI	:	Terminal, Command Prompt
IDE	:	Visual Code, Sublime Text

Whether you are moving to Amazon Web Services or are already running cloud-native, Ubuntu is the platform of choice for AWS. Canonical continuously tracks and delivers updates to Ubuntu images. Lean, stable, fast and powerful, Ubuntu

Server delivers services reliably, predictably and economically. It is the perfect base on which to build your instances. Ubuntu is very secure, built-in from the moment your machines and containers launch. Also it is free and will always be, and you have the option to get support and Landscape from Canonical.

Visual Studio Code is a lightweight but powerful code editor redefined and optimized for building and debugging modern web and cloud applications runs on your desktop and it is available for MacOS, Windows and Linux.

CHAPTER 3

SYSTEM IMPLEMENTATION

3.1 SYSTEM ARCHITECTURE

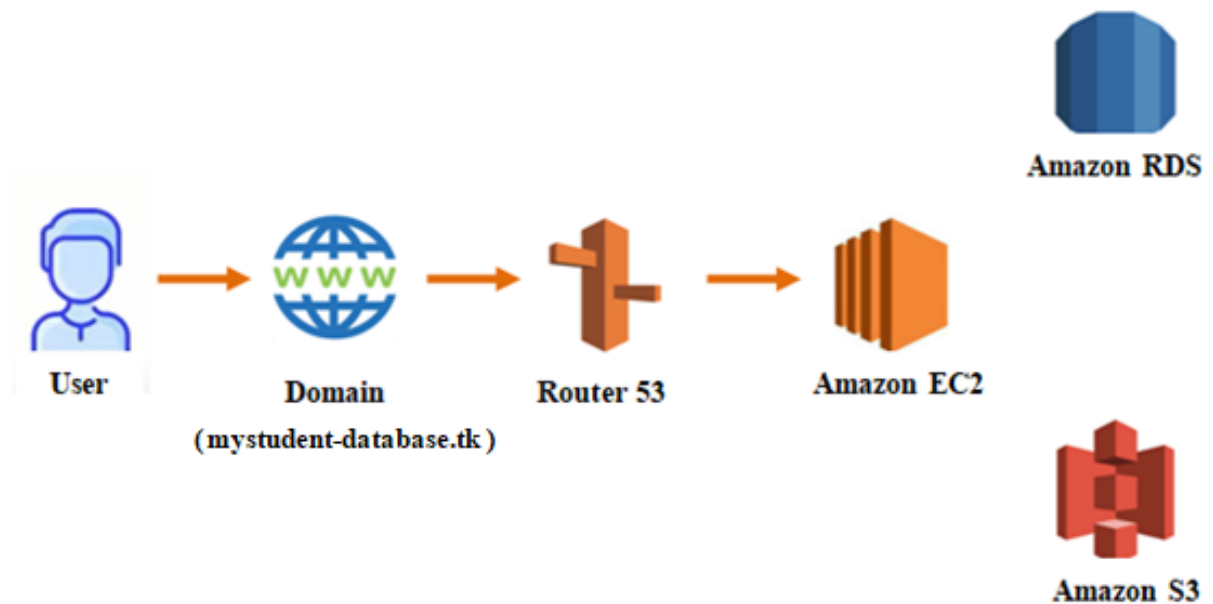


Fig.3.1.1 Architecture Diagram

The fig 3.1 shows the architectural diagram of this project defines the flow of data for deploying the website on AWS.

3.2 MODULES

There are five Modules

- Creating the Website
- Setting up the AWS
- Connecting EC2 to our Machine and RDS
- Cloning the Website code into EC2
- Deploying the Website in AWS Cloud

3.2.1 CREATING THE WEBSITE

Create a Student Database Website using HTML and CSS (frontend) and Python (backend) in Visual Studio.

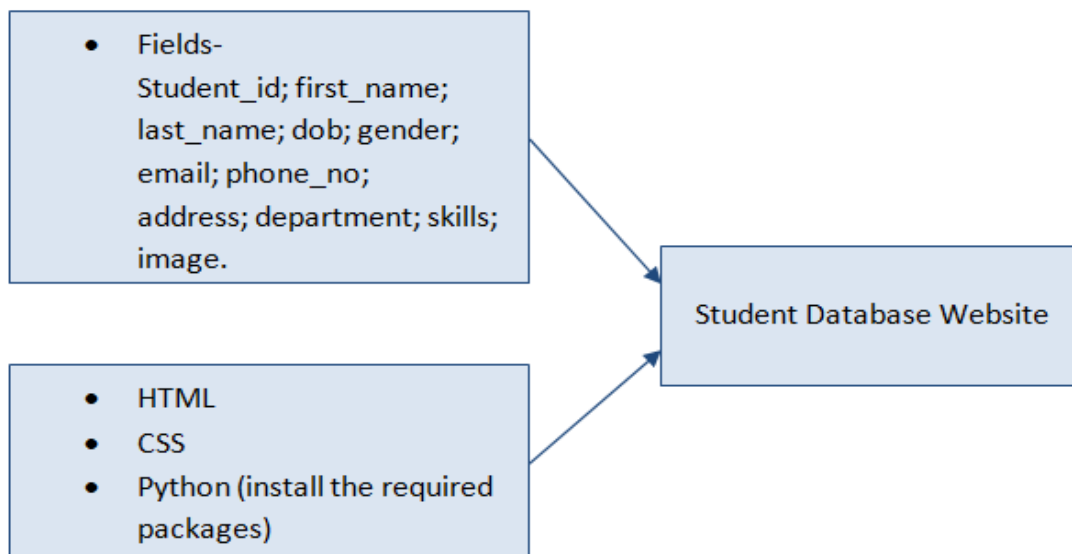


Fig.3.2.1.1

3.2.2 SETTING UP THE AWS

- Step-1: In Amazon Management Console, create a database '*studentinfo*' with engine option as MySQL in the RDS service.

- Step-2: Create a bucket '*addstudentinfo*' in the S3 service.
- Step-3: Create a instance i.e., server to deploy as '*mywebserver*' with AMI as Ubuntu server in the EC2 service.
- Step-4: Create a hosted zone '*mystudent-database.tk*' in the Route53 service.

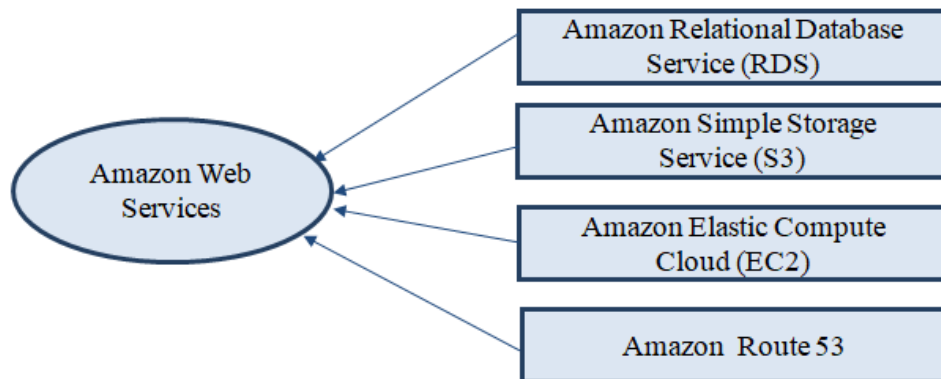


Fig.3.2.2.1

3.2.3 CONNECTING EC2 TO OUR MACHINE AND RDS

Connecting EC2 server to our Machine

- Step-1: In terminal, go to the place where you saved the Key-pair.
- Step-2: The command to connect is "ssh -i ./key-pair name ubuntu@ IP address of EC2".

Connecting EC2 server to RDS

- Step-1: In Ubuntu server, install mysql-client "sudo apt-get install mysql-client".
- Step-2: The command to connect is "mysql -h endpoint of database in RDS -username of the database -password of the database".

Now, we are inside MySQL Shell.

3.2.4 CLONING THE WEBSITE CODE INTO EC2

- Step-1: Go to EC2 server again.
- Step-2: Clone the code that was pushed into GitHub. The command is “git clone URL of your repository.git”.
- Step-3: Go into your repository inside EC2 server. The command is “cd name of your repository”.
- Step-4 : Install the required packages and deploy the website in EC2 server.

3.2.4 DEPLOYING THE WEBSITE IN AWS CLOUD

- Connect the registered domain name ‘mystudent-database.tk’ to Route53.

Connecting EC2 server to Route53

- Step-1: Create two record set in Route53 and add the EC2 IP address.
- Step-2: On the browser, type the domain name and deploy the website on AWS Cloud.

CHAPTER 4

CONCLUSION

Amazon Web Services provides number of tools to simplify and automate the provisioning of infrastructure and deployment of applications. Each deployment service has a unique approach for managing application deployments and offers various strategies for updating your application. For best results, focus on your workload and choose a deployment service that is tailored to your specific needs. As you plan, consider using hybrid deployment model approach, which uses a combination of deployment services for managing multiple applications throughout their lifecycle.

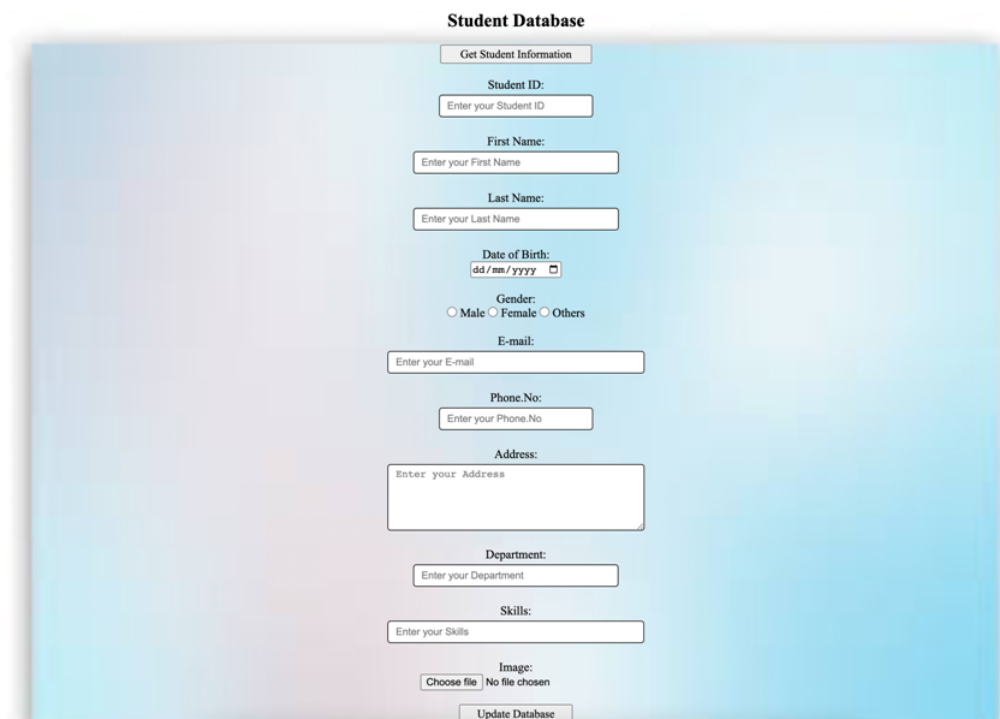
REFERENCES

1. Amazon Management Console. <http://aws.amazon.com/codedeploy/>.
2. Intellipaat. <https://intellipaat.com>.
3. Cloud Standards Customer Council (2017): Practical Guide to Cloud Computing, V3.0.
<https://www.omg.org/cloud/deliverables/practical-guide-to-cloud-computing>.
4. Bergmayr, A., J. Troya, P. Neubauer, M. Wimmer, and G. Kappel: UML-based Cloud Application Modeling with Libraries, Profiles, and Templates. Proceedings of the 2nd International Workshop on Model-Driven Engineering on and for the Cloud (CloudMDE 2014). <http://ceur-ws.org/> Vol. 242/paper7.pdf
5. Windriver.
<https://events.windriver.com/wrcd01/wrcm/2016/08/WP-cybersecurity-and-security-deployments.pdf>.

APPENDIX A

SNAP SHOTS

THE WEBSITE



Student Database

Get Student Information

Student ID:
Enter your Student ID

First Name:
Enter your First Name

Last Name:
Enter your Last Name

Date of Birth:
dd/mm/yyyy

Gender:
☐ Male ☐ Female ☐ Others

E-mail:
Enter your E-mail

Phone.No:
Enter your Phone.No

Address:
Enter your Address

Department:
Enter your Department

Skills:
Enter your Skills

Image:
Choose file | No file chosen

Update Database

CONNECTING EC2 TO OUR MAC

```
Downloads — ubuntu@ip-172-31-87-46: ~ — ssh -i mac-sam.pem ubuntu...
System load:  0.01          Processes:           101
Usage of /:   18.2% of 7.69GB Users logged in:      0
Memory usage: 21%          IPv4 address for eth0: 172.31.87.46
Swap usage:   0%

1 update can be applied immediately.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
```

Installing mysql-client

```
Downloads — ubuntu@ip-172-31-87-46: ~ — ssh -i mac-sam.pem ubuntu...
kB]
Get:33 http://security.ubuntu.com/ubuntu focal-security/main amd64 c-n-f Metadat
a [10.0 kB]
Get:34 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Package
s [859 kB]
Get:35 http://security.ubuntu.com/ubuntu focal-security/restricted Translation-e
n [122 kB]
Get:36 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 c-n-f M
etadadata [532 B]
Get:37 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages
[698 kB]
Get:38 http://security.ubuntu.com/ubuntu focal-security/universe Translation-en
[124 kB]
Get:39 http://security.ubuntu.com/ubuntu focal-security/universe amd64 c-n-f Met
adata [14.3 kB]
Get:40 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 Package
s [20.7 kB]
Get:41 http://security.ubuntu.com/ubuntu focal-security/multiverse Translation-e
n [5196 B]
Get:42 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 c-n-f M
etadadata [500 B]
Fetched 22.5 MB in 4s (5634 kB/s)
Reading package lists... Done
[ubuntu@ip-172-31-87-46:~$ sudo apt-get install mysql-client
```

CONNECTING EC2 TO RDS

```
Downloads — ubuntu@ip-172-31-87-46: ~ — ssh -i mac-sam.pem ubuntu@ec2-44-...
ubuntu@ip-172-31-87-46:~$ mysql -h studentinfo.chws2t1slh2.us-east-1.rds.amazonaws.com -u
sam -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 22
Server version: 8.0.28 Source distribution

Copyright (c) 2000, 2022, 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.
```

Creating table in MySQL Shell

```
Downloads — ubuntu@ip-172-31-87-46: ~ — ssh -i mac-sam.pem ubuntu@ec2-44-2...
[mysql> use studentinfo;
Database changed
[mysql> create table student(
  -> stuid varchar(20),
  -> fname varchar(20),
  -> lname varchar(20),
  -> dob date,
  -> gender set('male','female','others'),
  -> email varchar(40),
  -> phoneno varchar(10),
  -> address varchar(80),
  -> dept varchar(20),
  -> skill varchar(100));
Query OK, 0 rows affected (0.04 sec)

[mysql> show tables;
+-----+
| Tables_in_studentinfo |
+-----+
| student                |
+-----+
1 row in set (0.01 sec)

mysql>
```

CLONING THE WEBSITE CODE INTO EC2

```
Downloads — ubuntu@ip-172-31-87-46: ~/studentdatabase-aws — ssh -i mac-sam.p...
[ubuntu@ip-172-31-87-46:~]$ git clone https://github.com/Samritha03/studentdatabase-aws.git
Cloning into 'studentdatabase-aws'...
[Username for 'https://github.com': Samritha03
[Password for 'https://Samritha03@github.com':
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 13 (delta 3), reused 13 (delta 3), pack-reused 0
Unpacking objects: 100% (13/13), 5.35 KiB | 1.78 MiB/s, done.
[ubuntu@ip-172-31-87-46:~]$ ls
studentdatabase-aws
[ubuntu@ip-172-31-87-46:~]$ cd studentdatabase-aws/
ubuntu@ip-172-31-87-46:~/studentdatabase-aws$
```

DEPLOYING THE WEBSITE TO AWS CLOUD

Website deployed in EC2 and information in RDS

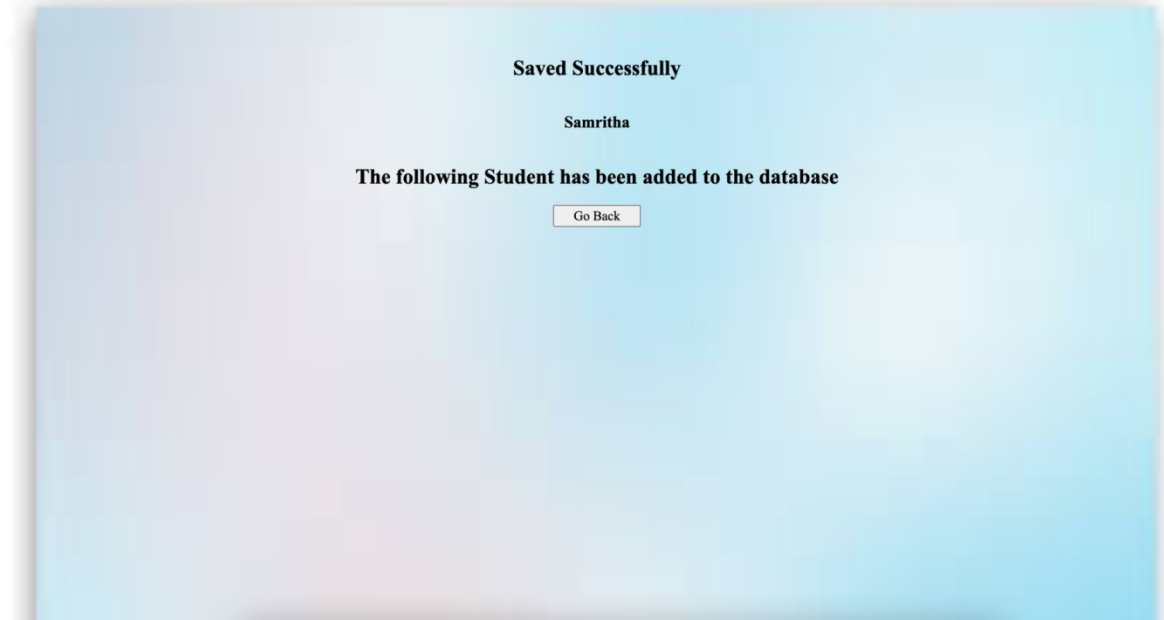
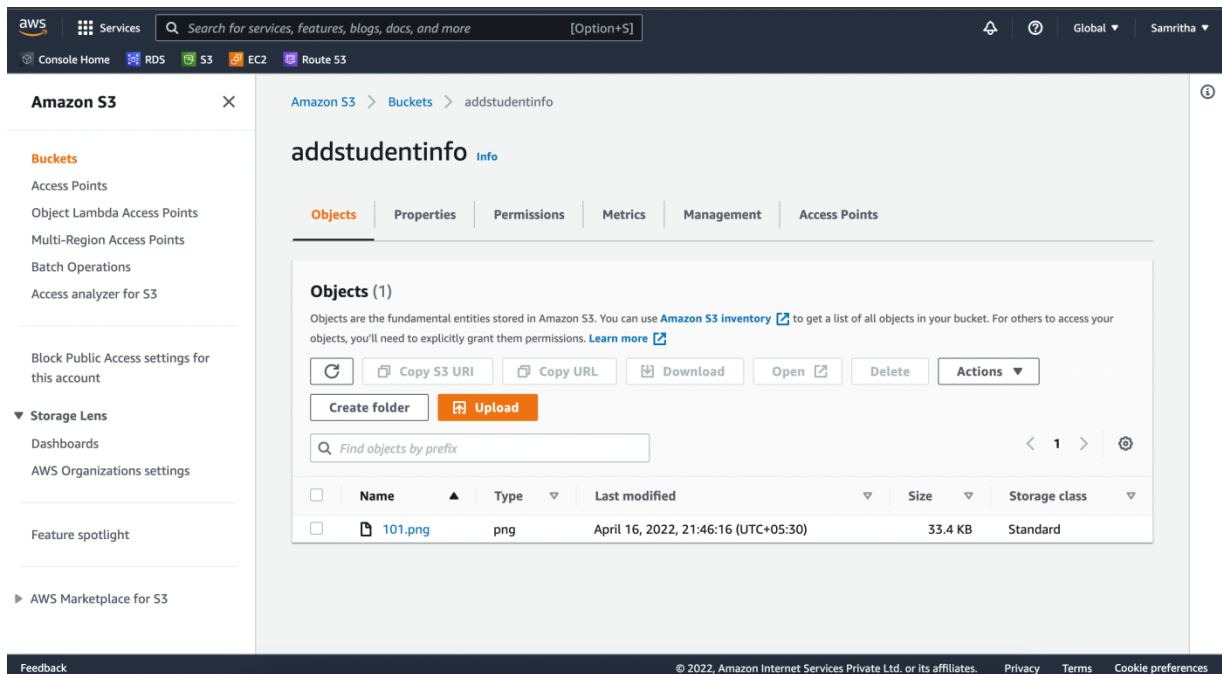


Image deployed in S3



APPENDIX B

SAMPLE CODING

Python Code for Website

```
from flask import Flask,render_template,request
from pymysql import connections
import os
import boto3
from config import *
```

```
app = Flask(__name__)
```

```
# DBHOST = os.environ.get("DBHOST")
# DBPORT = os.environ.get("DBPORT")
# DBPORT = int(DBPORT)
# DBUSER = os.environ.get("DBUSER")
# DBPWD = os.environ.get("DBPWD")
# DATABASE = os.environ.get("DATABASE")
```

```
bucket= custombucket
region= customregion
```

```
db_conn = connections.Connection(
    host= customhost,
    port=3306,
    user= customuser,
    password= custompass,
    db= customdb
)
```

```
output = {}
```

```
@app.route("/", methods=['GET', 'POST'])
```

```
def home():
```

```
    return render_template('AddStudent.html')
```

```
@app.route("/about", methods=['POST'])
```

```
def about():
```

```
    return render_template('www.intellipaat.com')
```

```
@app.route("/addstudent", methods=['POST'])
```

```
def AddStudent():
```

```
    student_id = request.form['student_id']
```

```
    first_name = request.form['first_name']
```

```
    last_name = request.form['last_name']
```

```
    dob = request.form['dob']
```

```
    gender = request.form['gender']
```

```
    email_id = request.form['email_id']
```

```
    phone_no = request.form['phone_no']
```

```
    address = request.form['address']
```

```
    department = request.form['department']
```

```
    skill = request.form['skill']
```

```
    student_image_file = request.files['student_image_file']
```

```
    insert_sql = "INSERT INTO %s VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s,  
%s)"
```

```
    cursor = db_conn.cursor()
```

```
    if student_image_file.filename == "":
```

```
        return "Please select a file"
```

```

try:
    cursor.execute(insert_sql,(student_id, first_name, last_name, dob, gender,
email_id, phone_no, address, department, skill))
    db_conn.commit()
    student_name = "" + first_name + " "+ last_name
    # Uplaod image file in S3 #
    student_image_file_name_in_s3 = "student-id-" + str(student_id) + "_image_file"
    s3 = boto3.resource('s3')

try:
    print("Data inserted in MySQL RDS... uploading image to S3...")
    s3.Bucket(custombucket).put_object(Key=student_image_file_name_in_s3,
Body=student_image_file)
    bucket_location =
boto3.client('s3').get_bucket_location(Bucket=custombucket)
    s3_location = (bucket_location['LocationConstraint'])

    if s3_location is None:
        s3_location = "
    else:
        s3_location = '-' + s3_location

    object_url = "https://s3{0}.amazonaws.com/{1}/{2}".format(
        s3_location,
        custombucket,
        student_image_file_name_in_s3)

    # Save image file metadata in DynamoDB #
    print("Uploading to S3 success... saving metadata in dynamodb...")

try:

```

```

dynamodb_client = boto3.client('dynamodb', region_name='us-east-1')
dynamodb_client.put_item(
    TableName='student_image_table',
    Item={
        'studentid': {
            'N': student_id
        },
        'image_url': {
            'S': object_url
        }
    }
)

except Exception as e:
    program_msg = "Flask could not update DynamoDB table with S3 object
URL"
    return str(e)

except Exception as e:
    return str(e)

finally:
    cursor.close()

print("all modification done...")
return render_template('AddStudentOutput.html', name=student_name)

@app.route("/getstudent", methods=['GET','POST'])
def GetEmp():
    return render_template('GetStudent.html')

```

```

@app.route("/fetchdata", methods=['GET','POST'])
def FetchData():
    student_id = request.form['student_id']

    output = {}
    select_sql = "SELECT student_id, first_name, last_name, dob, gender, email_id,
phone_no, address, department, skill from student2 where student_id=%s"
    cursor = db_conn.cursor()

    try:
        cursor.execute(select_sql,(student_id))
        result = cursor.fetchone()

        output["student_id"] = result[0]
        print('EVERYTHING IS FINE TILL HERE')
        output["first_name"] = result[1]
        output["last_name"] = result[2]
        output["dob"] = result[3]
        output["gender"] = result[4]
        output["email_id"] = result[5]
        output["phone_no"] = result[6]
        output["address"] = result[7]
        output["department"] = result[8]
        output["skill"] = result[9]
        print(output["student_id"])
        dynamodb_client = boto3.client('dynamodb', region_name=customregion)

    try:
        response = dynamodb_client.get_item(
            TableName='student_image_table',
            Key={

```

```

        'studentid': {
            'N': str(student_id)
        }
    }
)
image_url = response['Item']['image_url']['S']

except Exception as e:
    program_msg = "Flask could not update DynamoDB table with S3 object
URL"
    return render_template('AddStudentError.html', errmsg1=program_msg,
errmsg2=e)

except Exception as e:
    print(e)

finally:
    cursor.close()

return render_template("GetStudentOutput.html", id=output["emp_id"],
fname=output["first_name"], lname=output["last_name"], dob=output["dob"],
    gender=output["gender"], email=output["email_id"], phone=output["phone_no"],
address=output["address"], department=output["department"],
    skill=output["skill"], image_url=image_url)

if __name__ == '__main__':
    app.run(host='0.0.0.0',port=80,debug=True)

```