

USING CLOUD IMPLEMENTING WEB APPLICATION OF EMPLOYEE'S DATA



BY -

AKSHITH SIMHA KATRAGADA (yd8937)

GANGAREDDY NACHU (kt6301)

MANIKANTA KODALI (nv6200)

Table of contents

Chapter 1: Introduction	3
1.1 Problem summary	3
1.2 Solution	3
Chapter 2: Implementation Workflow	4
2.1 Architectural overview	5
2.1.1 Architecture of the system	5
2.1.2 Relation between services	5
Chapter 3: Development Process	7
3.1 Description of the development process	7
3.2 Challenges team faced during building this project	19
Chapter 4: Code & Technology	20
4.1 Programming language	20
4.2 Cloud services	20
Chapter 5: Git Access & Source Code	21
Chapter 6: Demo Screenshots from Website	25
Chapter 7: Outstanding Features of this Project	29
Chapter 8: Future Enhancements	29
Chapter 9: References	29

CHAPTER 1: INTRODUCTION

1.1 Problem Summary

An organization or company wants to store employee details and photos very efficiently and if any new data is added then they need to update in all their databases and map them accordingly. This company doesn't want to put upfront investment on resources like hardware, servers as they don't know about the company size in future.

The company does not want to rely on primitive technology such as manual storage or local server storage which creates many problems while the company grows and it's also very slow while retrieving the employee information from the database and they don't map according to all their databases. So, the company requirements are to provide a system where large data can be stored based on size and storing employee information and helping in retrieving the data with the help of the employee ID which we can use as the primary key where this overall process is done in a fast and efficient manner.

1.2 Solution

To solve the above mentioned problem, our proposed system will eliminate all the above issues and provide efficient results by using cloud computing technology, this system is implemented as a web application on AWS.

The web application we developed is deployed on AWS cloud platform where EC2 is used for launching instances S3 bucket is used for storing for example images and Relational Database Services for connecting to databases, the whole idea is to deploy web application on an EC2 instance which acts as Infrastructure as a Service (IaaS). In this web application we are providing search functionality to retrieve stored information about employees very quickly and efficiently.

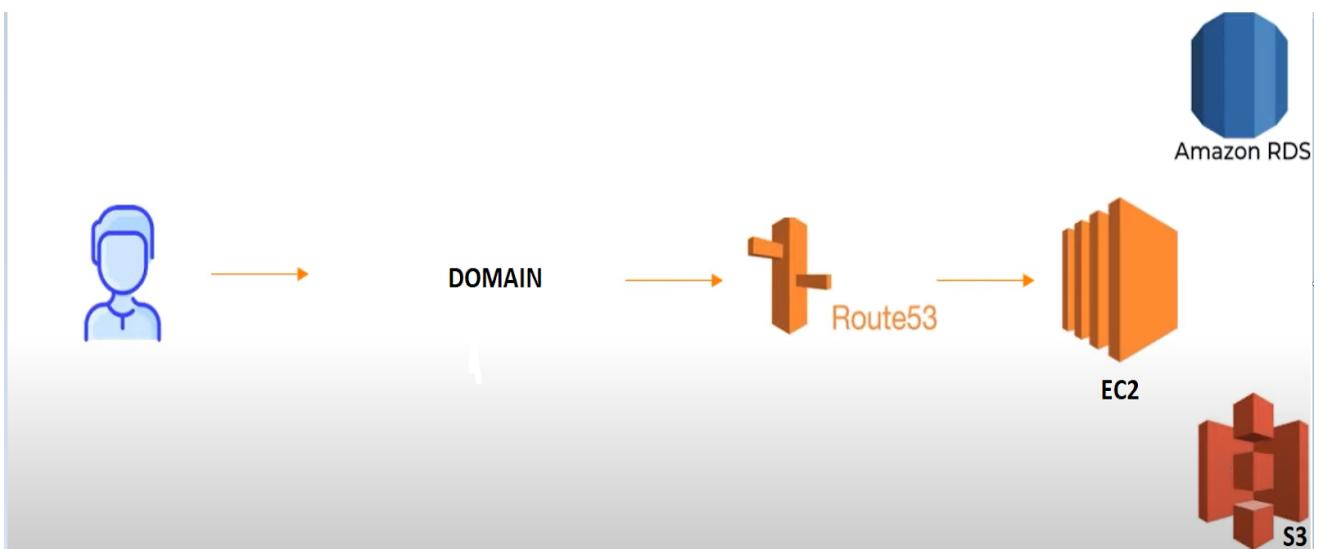
So, by this system the company or organization will be more benefited in numerous ways like companies don't need to put upfront investment on purchasing resources as scaling up and scaling down is done whenever it's needed automatically and pay as per usage of the resources only.

We are using many AWS services to implement this system like RDS, S3, EC2 and AWS boto and python for programming. The cloud platform offers more storage space, so large data can be stored, the computation speed is very fast so that administrators can retrieve or analyze employee data very efficiently.

CHAPTER 2: IMPLEMENTATION WORKFLOW



OVERVIEW DIAGRAM



WORKFLOW DIAGRAM

2.1 Architectural Overview

Here we are going to discuss the Architecture of the system and Relation between services.

2.1.1 Architecture of the system

The overall architecture of the system can be roughly divided into three components:

1. Selecting and configuring Amazon services
2. Building client-side application
3. Routing domain to EC2 server

In selecting and configuring Amazon services we are using three AWS namely -

1. RDS
2. S3
3. EC2

We need to select these services one by one From AWS dashboard and click appropriate options to set it up.

In building client-side applications, we are using Python and Html for the coding part.

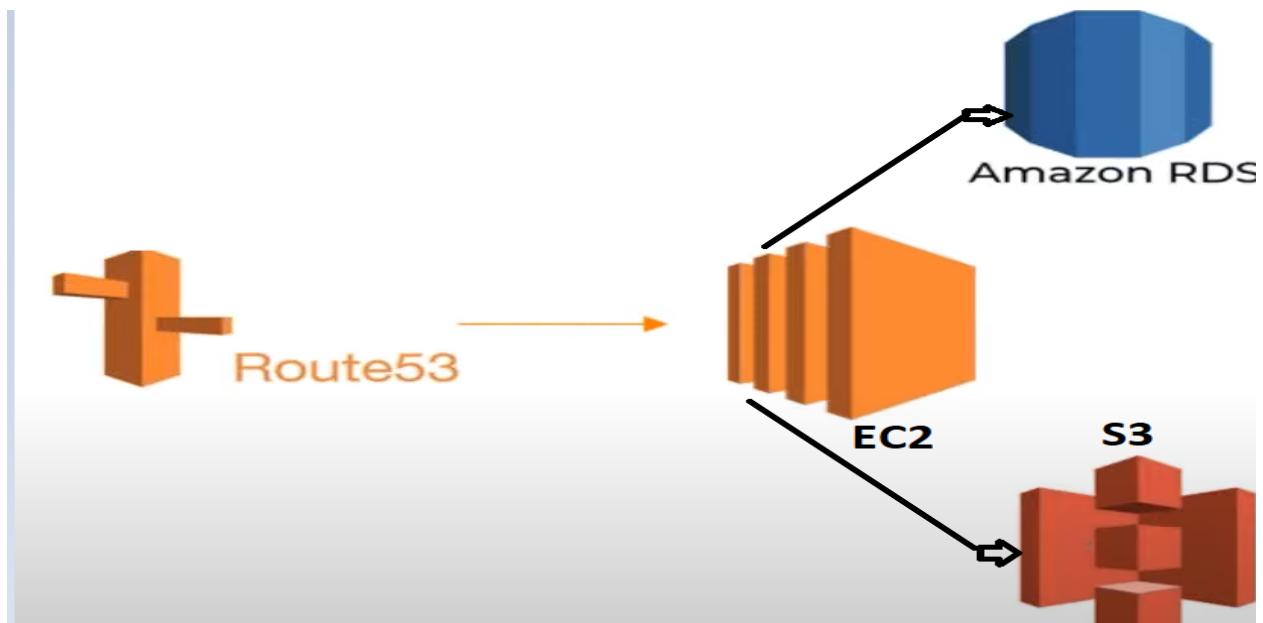
In routing domain to EC2 server we need to use another Amazon service called Route 53, which is used for routing purposes and next we should purchase domain names and then create hosted zones and configure them.

2.1.2 Relation between services

- We are using Amazon **RDS** service, which is used to operate and scale a relational database in the cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks. In this we are using MySQL database and this is used to store the entries. This RDS service will be connected by the EC2 server, this relation is used to access the database.
- We are using Amazon **S3** service, which is a Storage for the internet. We can use it to store and retrieve any amount of data at any time, from anywhere on the web. We should give EC2 instance access to S3, basically to upload the data. To make this relation we will give an IAM role to our instance.
- We are using Amazon **EC2** service, it is a web service for launching and managing Linux/UNIX and Windows Server instances in Amazon's data centers. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Here EC2

should be connected to RDS, S3 as described above, and the website is also deployed in EC2 server.

- We are using an **IAM** service, which is used when we connect with our RDS. We should give authentication to our S3 service, so we need to provide an IAM role for our instance. Then we are using **Route 53** service which is used to route domain to EC2 server.



RELATION DIAGRAM

CHAPTER 3: DEVELOPMENT PROCESS

Here we are going to discuss the Description of the development process and challenges the team faced during this project building.

3.1 Description of the development process

1) The first phase is to select and configure the Amazon services.

We are using three Amazon service initially -

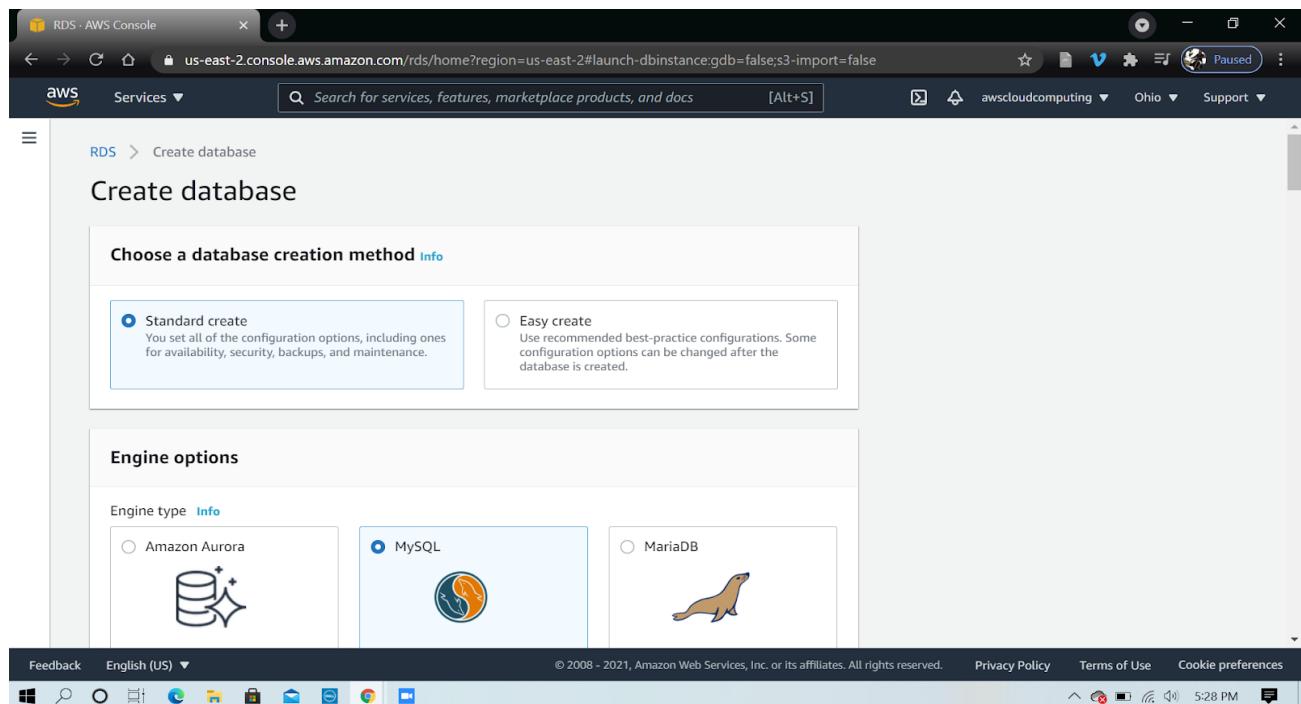
1. RDS
2. S3
3. EC2

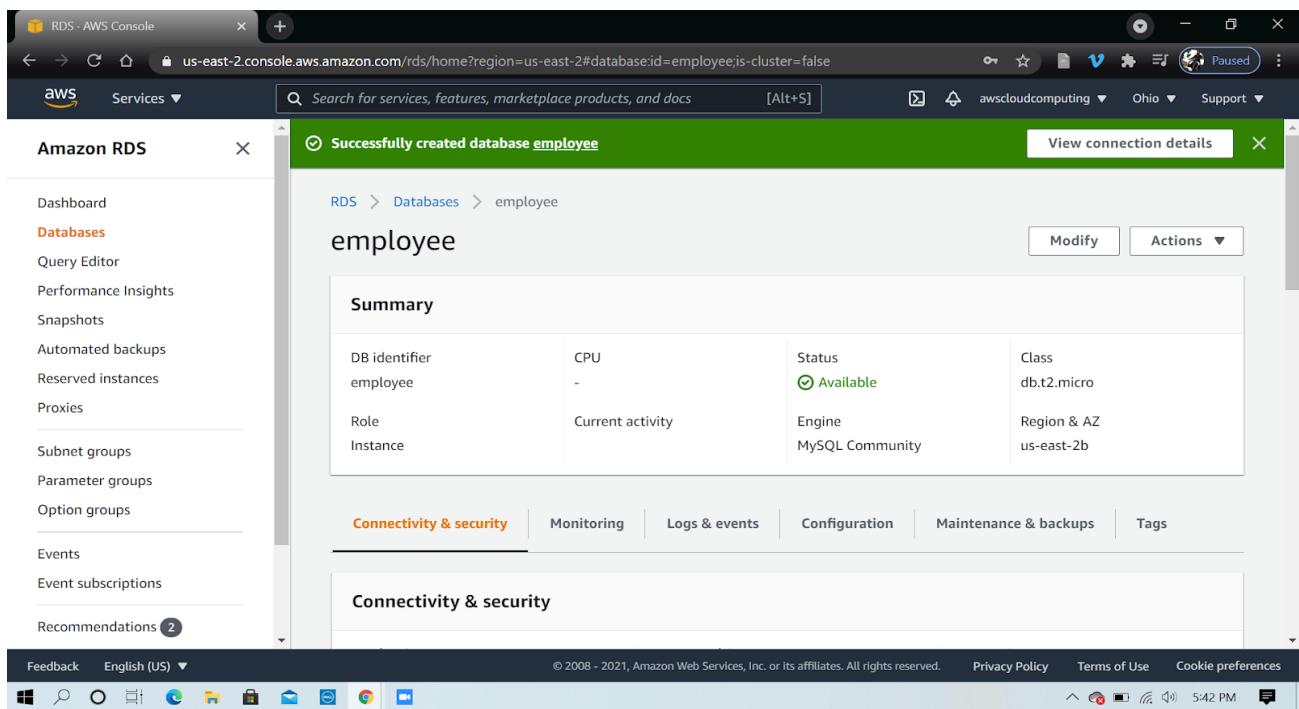
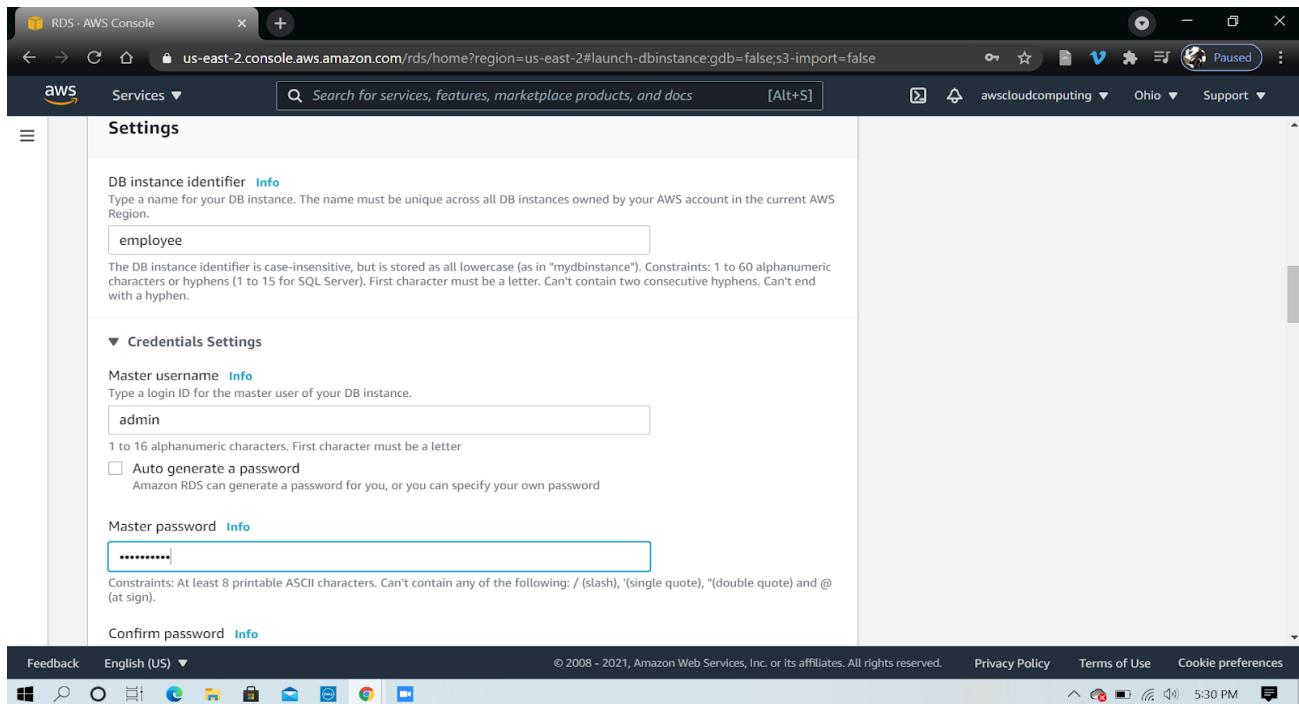
- Deploy MySQL database on AWS

It is done by using Amazon RDS, for this we should select RDS service from AWS management console dashboard and then select MySQL database engine. There are many engines for this project. We are using MySQL engine.

There are many advantages of using RDS, we don't need to launch a server, install MySQL on it and configure it manually instead of that we are just using AWS dashboard and select right options which will be managed automatically by AWS.

Some key advantages are easy to administer, highly scalable, available & durable, fast, secure and inexpensive.





• Create S3 bucket

It is done by using Amazon S3, for this we should select S3 service from AWS management console dashboard and create it. Check whether your bucket has been deployed in the correct region.

Some key advantages are Cost effective storage classes, easily managed data and access controls, query-in-place and process on-request, performance, scalability, availability and durability.

The screenshot shows the 'Create bucket' wizard in the AWS S3 Management Console. The 'General configuration' step is active. It includes fields for 'Bucket name' (set to 'addemployee08'), 'AWS Region' (set to 'US East (Ohio) us-east-2'), and a section for copying settings from an existing bucket. The browser interface at the top shows the URL 's3.console.aws.amazon.com/s3/bucket/create?region=us-east-2'.

The screenshot shows the 'Objects' tab for the 'addemployee08' bucket in the AWS S3 Management Console. The 'Objects (0)' section is shown, indicating no objects are currently stored. A toolbar at the top provides options for Copy URL, Open, Download, Delete, Actions, Create folder, and Upload. Below the toolbar is a search bar and a table header for object details (Name, Type, Last modified, Size, Storage class). The left sidebar shows navigation links for Buckets, Access Points, Object Lambda Access Points, Batch Operations, Access analyzer for S3, and Storage Lens. The browser interface at the top shows the URL 's3.console.aws.amazon.com/s3/buckets/addemployee08?region=us-east-2&tab=objects'.

- Deploy a EC2 server

It is done by using Amazon EC2 service, for this we should select EC2 service from AWS management console dashboard. Then launch an instance, where the ubuntu server is used. Select all the desirable options, configure security groups, create a new key pair or select existing one if you have one, and then review and launch. Some key advantages are reliable, scalable and infrastructure on demand.

The screenshot shows the AWS Launch Instance Wizard Step 1: Choose an Amazon Machine Image (AMI). It lists several AMI options:

- SUSE Linux**: ami-0b99ca359a84941ee (64-bit Arm) - Free tier eligible. Description: SUSE Linux Enterprise Server 15 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Amazon EC2 AMI Tools preinstalled; Apache 2.2, MySQL 5.5, PHP 5.3, and Ruby 1.8.7 available. Root device type: ebs, Virtualization type: hvm, ENA Enabled: Yes. Buttons: Select (disabled), Cancel and Exit.
- Ubuntu Server 20.04 LTS (HVM), SSD Volume Type**: ami-08962a4068733a2b6 (64-bit x86) / ami-064446ad1d755489e (64-bit Arm) - Free tier eligible. Description: Ubuntu Server 20.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>). Root device type: ebs, Virtualization type: hvm, ENA Enabled: Yes. Buttons: Select (disabled), Cancel and Exit.
- Microsoft Windows Server 2019 Base**: ami-0b697c4ae566cad55 - Microsoft Windows 2019 Datacenter edition, [English]. Root device type: ebs, Virtualization type: hvm, ENA Enabled: Yes. Buttons: Select (disabled), Cancel and Exit.
- Microsoft Windows Server 2019 Base with Containers**: ami-0235468fa1249482b - Microsoft Windows 2019 Datacenter edition, [English]. Root device type: ebs, Virtualization type: hvm, ENA Enabled: Yes. Buttons: Select (disabled), Cancel and Exit.

On the right, there is a sidebar with a radio button for "64-bit (x86)" which is selected.

The screenshot shows the AWS Launch Instance Wizard Step 6: Configure Security Group. It displays a table of existing security groups:

Security Group ID	Name	Description	Actions
sg-38960c4c	default	default VPC security group	Copy to new

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

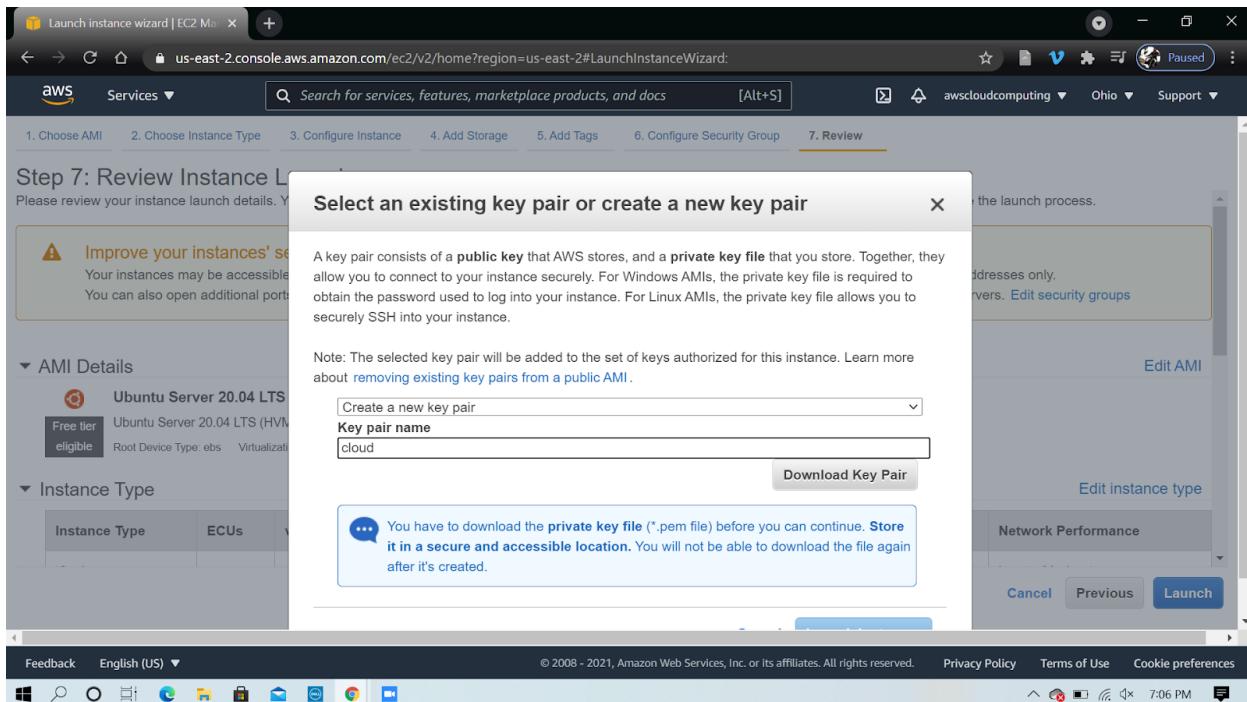
Assign a security group: Create a new security group Select an existing security group

The screenshot shows the Inbound rules for sg-38960c4c (Selected security groups: sg-38960c4c). The table has columns: Type, Protocol, Port Range, Source, and Description.

Type	Protocol	Port Range	Source	Description
All traffic	All	All	sg-38960c4c (default)	
SSH	TCP	22	0.0.0.0/0	

Buttons at the bottom: Cancel, Previous, Review and Launch.

The screenshot shows the Review and Launch step of the AWS Launch Instance Wizard. It includes a summary of the instance configuration and a "Review and Launch" button.



Instance ID	Public IPv4 address	Private IPv4 addresses
i-0cbfbbed141dedb4b3 (cloud-pro)	3.20.221.160 open address	172.31.44.222

Instance state	Public IPv4 DNS	Private IPv4 DNS
Running	ec2-3-20-221-160.us-east-2.compute.amazonaws.com open address	ip-172-31-44-222.us-east-2.compute.internal

Instance type	Elastic IP addresses	VPC ID
t2.micro	-	vpc-0847d263

AWS Compute Optimizer finding	IAM Role	Subnet ID
Opt-in to AWS Compute Optimizer for recommendations. Learn more	-	subnet-9d590bd1

Next connect the server to our machine.

To connect from windows to server one can use Putty or Git bash or Super Putty which is a premium tool.

To connect from a mac one can use the terminal itself, this will connect to an instance deployed in AWS.

Next connect to our RDS

First deploy MySQL client on the server which helps to connect to our RDS machine, for this install MySQL client.

```
ubuntu@ip-172-31-23-106: ~  
ubuntu@ip-172-31-23-106:~$ mysql -u admin -p -h employee.ccrxwdt9mefb.us-east-1.rds.amazonaws.com  
Enter password: |
```

```
ubuntu@ip-172-31-23-106: ~  
ubuntu@ip-172-31-23-106:~$ mysql -u admin -p -h employee.ccrxwdt9mefb.us-east-1.rds.amazonaws.com  
Enter password:  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 3036  
Server version: 8.0.20 Source distribution  
  
Copyright (c) 2000, 2021, 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> |
```

After installing MySQL client software now we connect to the RDS, so that now we will be in a MySQL shell.

Next deploy database on server

For this first create a database and then create a table in that database for the information to be stored.

// commands

After logging into databases:

Show databases; // gives list of Databases present

```
mysql> Show databases;
+-----+
| Database      |
+-----+
| employee      |
| information_schema |
| mysql          |
| performance_schema |
+-----+
4 rows in set (0.03 sec)

mysql> |
```

(need to create a database now)

create database <database-name>; // will create a database with the given name.

```
mysql> create database EMP;
Query OK, 1 row affected (0.05 sec)

mysql> |
```

(need to switch to created database)

Use <database name>;

```
mysql> use EMP;
Database changed
mysql> |
```

(now we can create tables in the created database)

Create table <table-name>(name varchar (20), age varchar (20)); // creates a table inside the database.

```
mysql> use EMP;
Database changed
mysql> create table EMP (name varchar (20), age varchar (20));
Query OK, 0 rows affected (0.09 sec)

mysql> |
```

Show tables;

```
mysql> show tables;
+-----+
| Tables_in_EMP |
+-----+
| EMP           |
+-----+
1 row in set (0.01 sec)

mysql> |
```

2) The second phase is to build the website, configure the website code and deploy the website on EC2 server.

- Build the website

Here to build the website we have used the Python Flask framework. Flask is known as a simple but extensible framework.

- Configure the website code

- 1) Provide Host name of RDS in host region of config code.
- 2) Provide username, password and database name in config code.
- 3) Provide bucket name and region in the configuration code.

- Deploy website on EC2 server

Make the website available on EC2 server.

After connecting to EC2 instance we need to clone a git repository:

```
ubuntu@ip-172-31-23-106: ~
ubuntu@ip-172-31-23-106:~$ git clone "https://github.com/mkodali37/cs_623_proj.git"
```

Then navigate to the project folder and install the required packages:

Execute all the below commands on the server to install them on server.

- **sudo apt-get install python3**

```
ubuntu@ip-172-31-23-106: ~/CS_623_proj
ubuntu@ip-172-31-23-106:~$ ls
cs_623_proj
ubuntu@ip-172-31-23-106:~$ cd cs_623_proj
ubuntu@ip-172-31-23-106:~/cs_623_proj$ sudo apt-get install python3 |
```

Python3 is the language that is coded for the website, we must install it on an existing system where we are running the website.

- **sudo apt-get install python3-flask**

```
ubuntu@ip-172-31-23-106: ~/CS_623_proj
ubuntu@ip-172-31-23-106:~$ ls
cs_623_proj
ubuntu@ip-172-31-23-106:~$ cd cs_623_proj
ubuntu@ip-172-31-23-106:~/cs_623_proj$ sudo apt-get install python3-flask |
```

Flask helps to go ahead and publish a website on to that server.

- **sudo apt-get install python3-pymysql**

```
ubuntu@ip-172-31-23-106: ~/CS_623_proj
ubuntu@ip-172-31-23-106:~$ ls
cs_623_proj
ubuntu@ip-172-31-23-106:~$ cd cs_623_proj
ubuntu@ip-172-31-23-106:~/cs_623_proj$ sudo apt-get install python3-pymysql |
```

It is a library which helps the website to connect to the MySQL server.

- **sudo apt-get install python3-boto**

```
ubuntu@ip-172-31-23-106: ~/CS_623_proj
ubuntu@ip-172-31-23-106:~$ ls
cs_623_proj
ubuntu@ip-172-31-23-106:~$ cd cs_623_proj
ubuntu@ip-172-31-23-106:~/cs_623_proj$ sudo apt-get install python3-boto|
```

It is an SDK which helps to connect to S3 service where we upload files.

Then give EC2 instance or server access to basically upload data to S3. We need to do this because on the server we are running some code and that code is trying to interact with the AWS S3 service, when we connect with our RDS we should give authentication to our S3 service, so for that we must give the IAM role to our instance.

To give IAM role to our instance, follow the below steps -

- Select IAM service from AWS management console dashboard.
- Then create an admin role for our AWS account.
- Then select roles for our EC2 instance.
- Then we should go to our EC2 service and click on the attach/replace IAM role and select the role which we created from it. Now EC2 is connected to RDS, EC2 is connected to S3, and website deployed in EC2 server.

3) The third phase is routing domain to EC2 server.

To route domain to EC2 server we need to use Route 53 service from AWS management console dashboard and then connect Route 53 service to EC2 server.

- Route 53 service

First, select Route 53 service from AWS management console dashboard.

Then, purchase a domain name for the website.

Then, create a hosted zone and mention the purchased domain name here.

The screenshot shows the Freenom Client Area interface. At the top, there's a navigation bar with links for 'Services', 'Partners', 'About Freenom', and 'Support'. Below the header, the main title 'My Domains' is displayed in blue, with a sub-instruction 'View & manage all the domains you have registered with us from here...'. A search bar labeled 'Enter Domain to Find' is followed by a blue 'Filter' button. A table lists the registered domain 'awse2ewebsitefree.tk' with details: Registration Date 2021-05-02, Expiry date 2022-05-02, Status ACTIVE, Type Free, and a 'Manage Domain' button. At the bottom, there are pagination controls for 'Results Per Page' (set to 10) and a note '1 Records Found, Page 1 of 1'.

Route 53 > Hosted zones > Create hosted zone

Create hosted zone [Info](#)

Hosted zone configuration

A hosted zone is a container that holds information about how you want to route traffic for a domain, such as example.com, and its subdomains.

Domain name [Info](#)
This is the name of the domain that you want to route traffic for.
awse2ewebsitefree.tk
Valid characters: a-z, 0-9, ! " # \$ % & ' () * + , - / : ; < = > ? @ [\] ^ _ { } . ~

Description - optional [Info](#)
This value lets you distinguish hosted zones that have the same name.
The hosted zone is used for...
The description can have up to 256 characters. 0/256

Type [Info](#)
The type indicates whether you want to route traffic on the internet or in an Amazon VPC.
 Public hosted zone
A public hosted zone determines how traffic is routed on the internet.
 Private hosted zone
A private hosted zone determines how traffic is routed within an Amazon VPC.

Tags [Info](#)
Apply tags to hosted zones to help organize and identify them.
No tags associated with the resource.
[Add tag](#)
You can add up to 50 more tags.

Route 53 Console Hosted Zones [X](#) [+](#)
 ← → C <https://console.aws.amazon.com/route53/v2/hostedzones#ListRecordSets/Z05411541XMCMLTAR8P70> Search for services, features, marketplace products, and docs [Alt+S] Incognito
 AWS Services ▾ awscloudcomputing Global Support
 Route 53 > Hosted zones > awse2ewebsitefree.tk
 awse2ewebsitefree.tk [Info](#) Delete zone Test record Configure query logging
 ▶ Hosted zone details [Edit hosted zone](#)
 Records (4) DNSSEC signing Hosted zone tags (0)
 Records (4) [Info](#) Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.
 Filter records by property or value Type Routing policy Alias

<input type="checkbox"/>	Record name	Type	Routin...	Differ...	Value/Route traffic to
<input type="checkbox"/>	awse2ewebsitefree.tk	A	Simple	-	54.160.76.95 ns-865.awsdns-44.net. ns-1049.awsdns-03.org. ns-422.awsdns-52.com. ns-1781.awsdns-30.co.uk.
<input type="checkbox"/>	awse2ewebsitefree.tk	NS	Simple	-	ns-865.awsdns-44.net. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400
<input type="checkbox"/>	awse2ewebsite...	A	Simple	-	54.160.76.95

Now, configure the Name Servers.

The screenshot shows the 'Nameservers' section of the Freenom DNS management interface. It includes a note about propagation times and two radio button options: 'Use default nameservers (Freenom Nameservers)' and 'Use custom nameservers (enter below)', with the latter being selected. Below these are five input fields for custom nameservers, each containing a yellow-highlighted IP address: NS-1049.AWSDNS-03.ORG, NS-1781.AWSDNS-30.CO.UK, NS-422.AWSDNS-52.COM, NS-865.AWSDNS-44.NET, and an empty field for Nameserver 5. A 'Change Nameservers' button is at the bottom left, and a 'Activate Win' link with a 'Go to Settings to' button is at the bottom right.

- Connect Route 53 service to EC2 server
- For this first create a record set and mention EC2 ip address in the value box.

The screenshot shows the 'Quick create record' interface in the AWS Route 53 console. It's creating a new record set for the domain 'awse2ewebsitefree.tk'. The 'Record name' is 'blog', 'Record type' is 'A - Routes traffic to an IPv4 address and so...', 'Value' is '54.160.76.95', and 'TTL (seconds)' is '300'. The 'Routing policy' is 'Simple routing'. At the bottom, there are buttons for 'Cancel' and 'Create records'.

Then, create another record set and mention the name as WWW and put the same ip address in the value box.

The screenshot shows the AWS Route 53 console interface for creating a new record set. The URL is `console.aws.amazon.com/route53/v2/hostedzones#CreateRecordSet/Z05411541XMCMLTAR8P7O`. The record set is being created for the domain `awse2ewebsitefree.tk`. The 'Quick create record' form is filled out as follows:

- Record name:** `www`
- Record type:** `A – Routes traffic to an IPv4 address and so...`
- Value:** `54.160.76.95`
- TTL (seconds):** `300`
- Routing policy:** `Simple routing`

At the bottom right, there are buttons for `Create records` and `Cancel`.

3.2 Challenges team faced during building this project

- Different Amazon services being used in this project and connection to RDS from Git bash/ Putty is one of the challenges we faced.
- All the commands to install flask, boto and pymysql in the server.
- In the second phase initially we didn't give EC2 instance access to upload data to S3, after that we made a research on IAM roles and used IAM service to upload pictures to S3 bucket.
- Usage of **Route 53** service to route domain to EC2 server as it is not available to all. We purchased a domain name for implementing Route 53 service.

CHAPTER 4: CODE & TECHNOLOGY

4.1 Programming language

1) Frontend

In frontend we used Python and Html languages.

2) Backend

In backend we used RDS (MySQL Database)

4.2 Cloud services

1) Amazon RDS:

We are using Amazon RDS service, which is used to operate and scale a relational database in the cloud. In this we are using MySQL database and this is used to store the entries. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

2) Amazon S3:

We are using Amazon S3 service, which is a Storage for the internet. We can use it to store and retrieve any amount of data at any time, from anywhere on the web.

3) Amazon EC2 :

We are using Amazon EC2 service, it is a web service for launching and managing Linux/UNIX and Windows Server instances in Amazon's data centers.

4) Amazon IAM

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. When we connect with our RDS we should give authentication to our S3 service, so that we must give the IAM role to our instance.

5) Amazon Route 53

AWS Route 53 is a service we are using to route domain to EC2 server.

CHAPTER 5: GIT ACCESS & SOURCE CODE

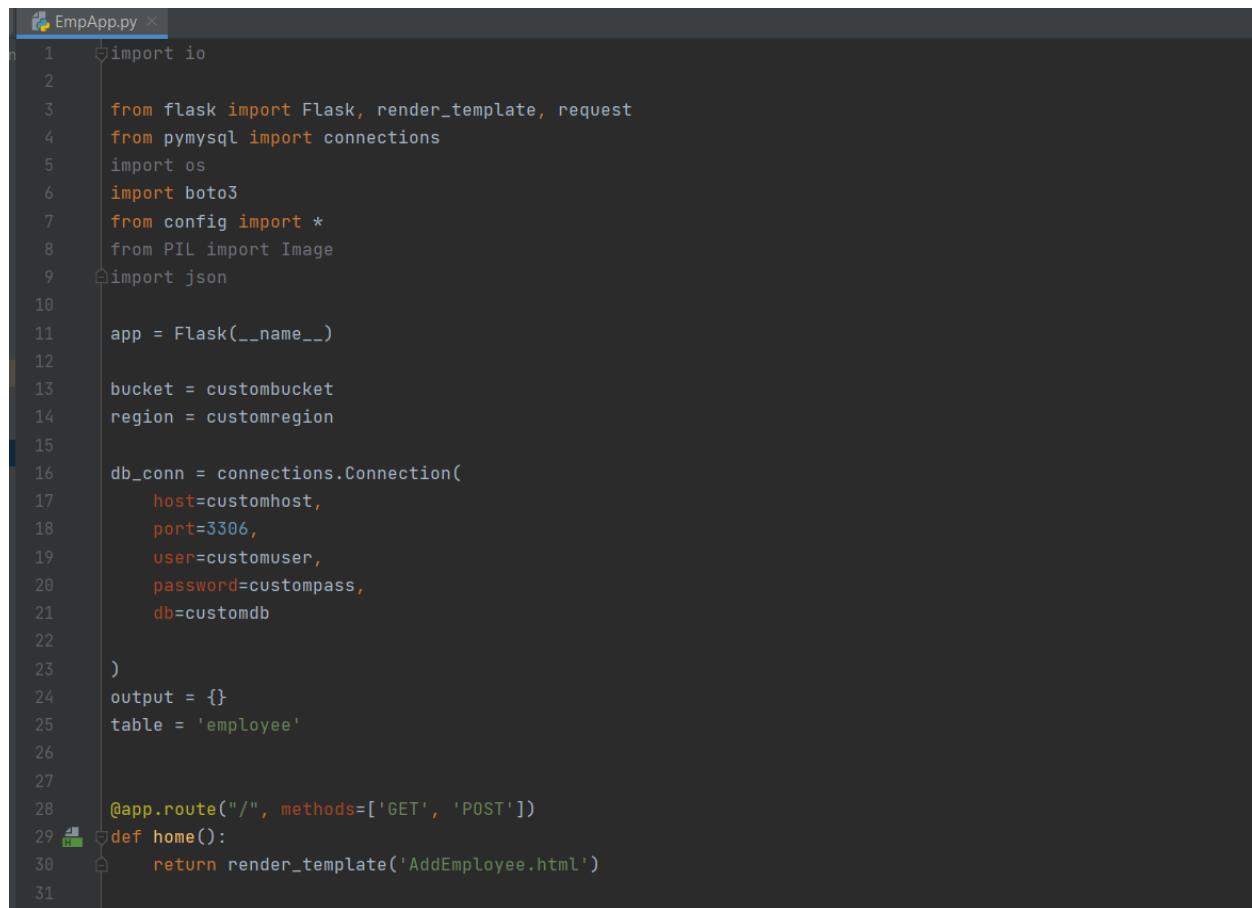
Below is the GitHub Link for our project:

https://github.com/gangareddynachu/AWS_E2E_Website

We have a component where employees can provide their information and when they click to update the database this code will run and establishes a connection to store data and to retrieve the data also. The python code for this is below -

The python code :

EmpApp.py is the entry point of our application:



A screenshot of a code editor showing the contents of the **EmpApp.py** file. The code is written in Python and uses the Flask framework. It imports various modules including io, flask, pymysql, os, boto3, config, PIL, and json. It defines a Flask application, sets up a connection to a MySQL database with custom host, port, user, password, and database names. It defines a route for the root URL ('/') that returns a template named 'AddEmployee.html'. The code is annotated with line numbers from 1 to 31.

```
 1  import io
 2
 3  from flask import Flask, render_template, request
 4  from pymysql import connections
 5  import os
 6  import boto3
 7  from config import *
 8  from PIL import Image
 9  import json
10
11 app = Flask(__name__)
12
13 bucket = custombucket
14 region = customregion
15
16 db_conn = connections.Connection(
17     host=customhost,
18     port=3306,
19     user=customuser,
20     password=custompass,
21     db=customdb
22 )
23
24 output = {}
25 table = 'employee'
26
27
28 @app.route("/", methods=['GET', 'POST'])
29 def home():
30     return render_template('AddEmployee.html')
```

```
EmpApp.py X
32
33     @app.route("/about", methods=['GET','POST'])
34     def about():
35         return render_template('about.html')
36
37
38     @app.route("/addemp", methods=['POST'])
39     def AddEmp():
40         emp_id = request.form['emp_id']
41         first_name = request.form['first_name']
42         last_name = request.form['last_name']
43         pri_skill = request.form['pri_skill']
44         location = request.form['location']
45         emp_image_file = request.files['emp_image_file']
46
47         insert_sql = "INSERT INTO employee VALUES (%s, %s, %s, %s, %s)"
48         cursor = db_conn.cursor()
49
50
51
52     try:
53
54         cursor.execute(insert_sql, (emp_id, first_name, last_name, pri_skill, location))
55         db_conn.commit()
56         emp_name = "" + first_name + " " + last_name
57         # Upload image file in S3 #
58         emp_image_file_name_in_s3 = "emp-id-" + str(emp_id) + "_image_file"
59         s3 = boto3.resource('s3')
60
61     try:
62         print("Data inserted in MySQL RDS... uploading image to S3...")
63         s3.Bucket(custombucket).put_object(Key=emp_image_file_name_in_s3, Body=emp_image_file)

```

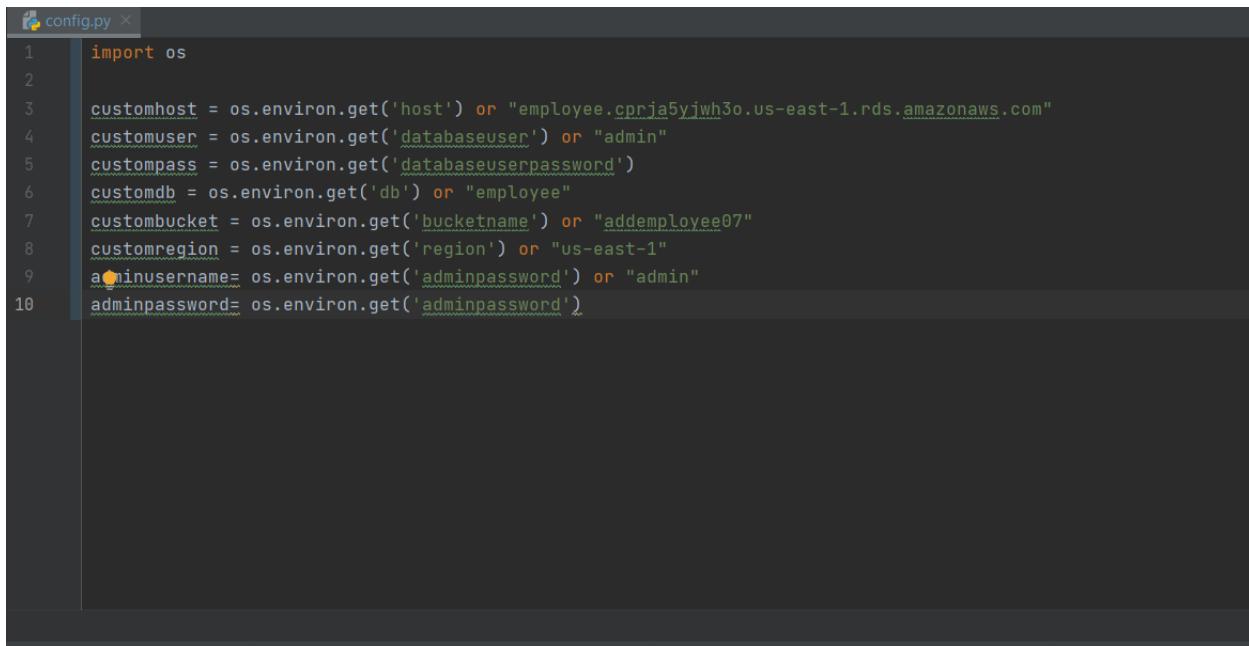
```
EmpApp.py X
59
60
61     s3 = boto3.resource('s3')
62
63     try:
64         print("Data inserted in MySQL RDS... uploading image to S3...")
65         s3.Bucket(custombucket).put_object(Key=emp_image_file_name_in_s3, Body=emp_image_file)
66         bucket_location = boto3.client('s3').get_bucket_location(Bucket=custombucket)
67         s3_location = (bucket_location['LocationConstraint'])
68
69         if s3_location is None:
70             s3_location = ''
71         else:
72             s3_location = '-' + s3_location
73
74         object_url = "https://s3{0}.amazonaws.com/{1}/{2}".format(
75             s3_location,
76             custombucket,
77             emp_image_file_name_in_s3)
78
79     except Exception as e:
80         return render_template('Error.html')
81
82     finally:
83         cursor.close()
84
85     print("all modification done...")
86     return render_template('EmployeeAdd_Success.html', name=emp_name)
87
88     @app.route('/admin', methods=['GET', 'POST'])
89     def admin():
90         return render_template("admin.html")
91
92     @app.route('/getemp', methods=['GET', 'POST'])
93
```

```

  EmpApp.py ×
90     @app.route("/getemp", methods=['GET', 'POST'])
91     def GetEmp():
92         return render_template("GetEmployeeInfo.html")
93
94     @app.route("/fetchdata", methods=['POST'])
95     def FetchEmp():
96         emp_id = request.form['emp_id']
97
98         output = {}
99         select_sql = "SELECT empid, fname, lname, pri_skill, location from employee where empid=%s"
100        cursor = db_conn.cursor()
101        emp_image_file_name_in_s3 = "emp-id-" + str(emp_id) + "_image_file"
102        s3 = boto3.resource('s3')
103
104        bucket_location = boto3.client('s3').get_bucket_location(Bucket=custombucket)
105        s3_location = (bucket_location['LocationConstraint'])
106
107        if s3_location is None:
108            s3_location = ''
109        else:
110            s3_location = '-' + s3_location
111
112        image_url = "https://s3{0}.amazonaws.com/{1}/{2}".format(
113            s3_location,
114            custombucket,
115            emp_image_file_name_in_s3)
116
117        try:
118            cursor.execute(select_sql, (emp_id))
119            result = cursor.fetchone()
120
  EmpApp.py ×  EmpImageFileInS3.py
116
117    try:
118        cursor.execute(select_sql, (emp_id))
119        result = cursor.fetchone()
120
121        output["emp_id"] = result[0]
122        print('EVERYTHING IS FINE TILL HERE')
123        output["first_name"] = result[1]
124        output["last_name"] = result[2]
125        output["primary_skills"] = result[3]
126        output["location"] = result[4]
127        print(output["emp_id"])
128
129
130
131
132
133        return render_template("EmployeeInfo_Output.html", id=output["emp_id"], fname=output["first_name"],
134                                lname=output["last_name"], interest=output["primary_skills"], location=output["location"])
135
136    except Exception as e:
137        print(e)
138        return render_template('Error.html')
139    if __name__ == '__main__':
140        app.run(host='0.0.0.0', port=80, debug=True)
141

```

There is a small python code below for the configuration - **Config.py**



```
config.py ×
1 import os
2
3 customhost = os.environ.get('host') or "employee.cprja5yjwh3o.us-east-1.rds.amazonaws.com"
4 customuser = os.environ.get('databaseuser') or "admin"
5 custompass = os.environ.get('databaseuserpassword')
6 customdb = os.environ.get('db') or "employee"
7 custombucket = os.environ.get('bucketname') or "addemployee07"
8 customregion = os.environ.get('region') or "us-east-1"
9 adminusername= os.environ.get('adminpassword') or "admin"
10 adminpassword= os.environ.get('adminpassword')
```

CHAPTER 6: DEMO SCREENSHOTS FROM WEBSITE

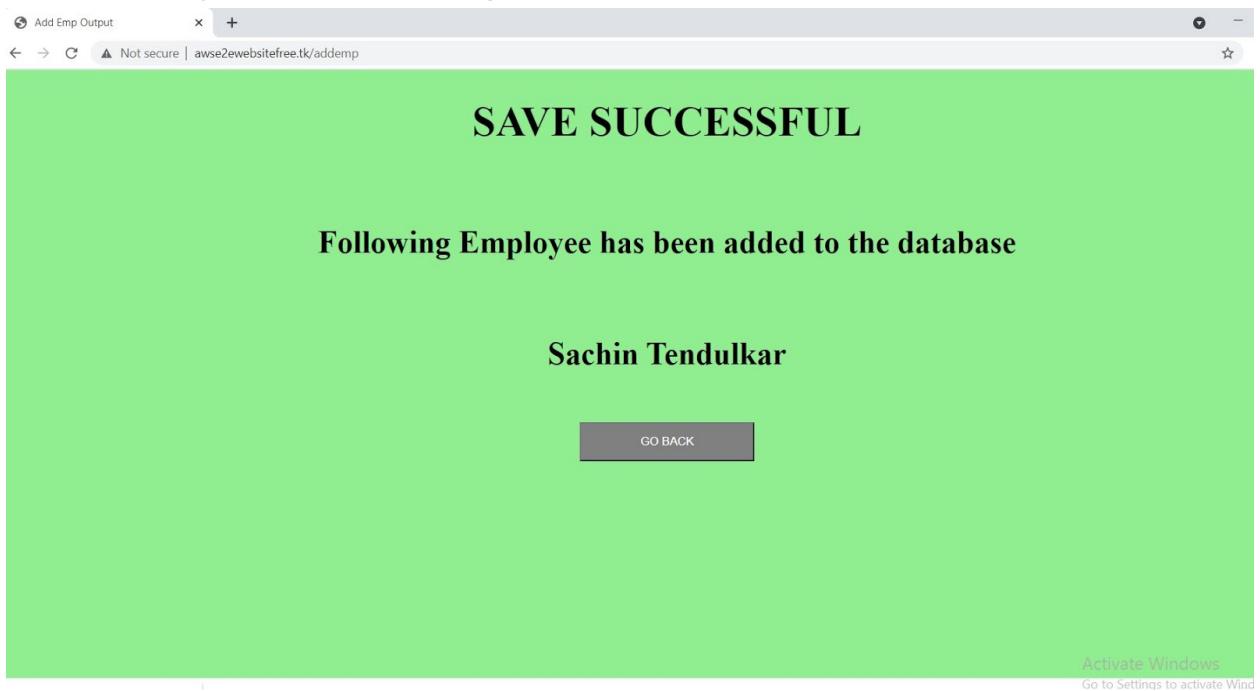
Home Page:

The screenshot shows a web browser window titled "Add Employee Information". The address bar indicates the site is "Not secure" and the URL is "awse2ewebsitefree.tk". The main content area is titled "Employee Database". It features a "GET EMPLOYEE INFORMATION" button at the top. Below it are five input fields: "Employee ID" (with a dropdown arrow), "First Name" (empty), "Last Name" (empty), "Primary Skills" (empty), and "Location" (empty). There is also a file input field labeled "Image:" with the placeholder "Choose File | No file chosen". At the bottom right are "UPDATE DATABASE" and "About Us" buttons.

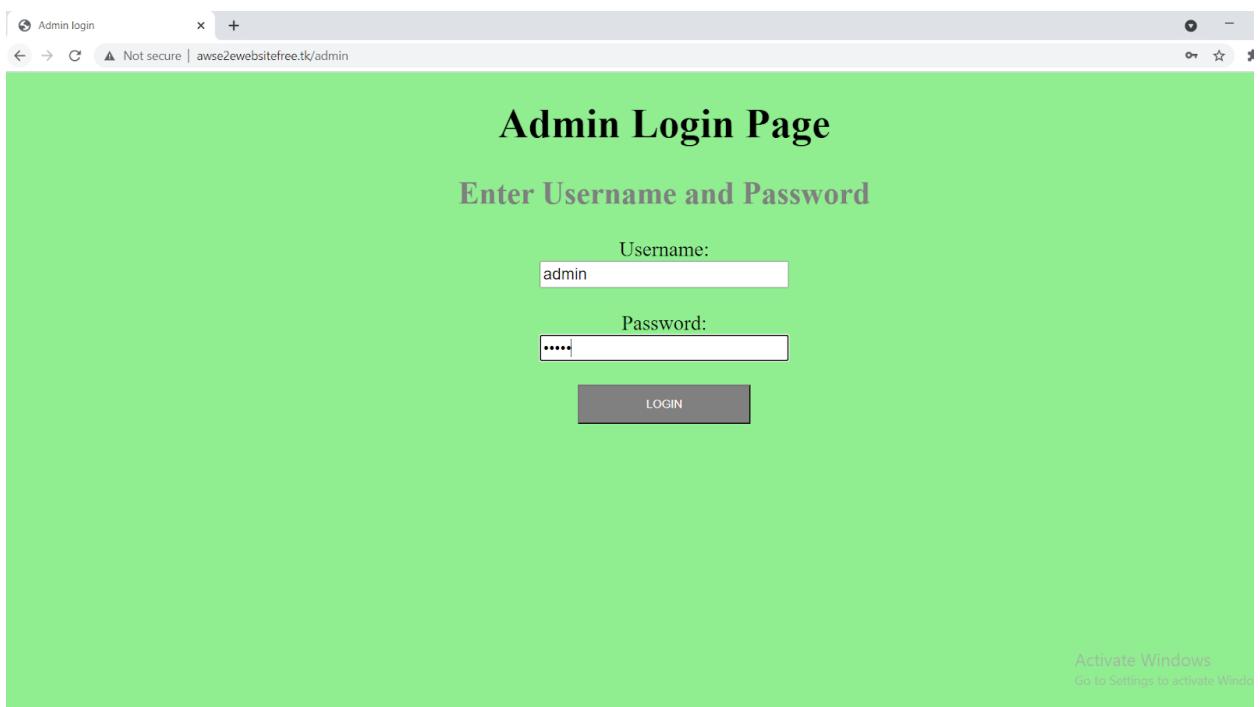
Add New Employee Page:

The screenshot shows the same "Add Employee Information" page as the previous screenshot, but with different data entered. The "Employee ID" field contains "100", the "First Name" field contains "Sachin", and the "Last Name" field contains "Tendulkar". The "Primary Skills" field contains "Angular JS", and the "Location" field contains "California". The "Image:" field shows "Choose File | sachin.jpg". The "UPDATE DATABASE" and "About Us" buttons are visible at the bottom.

Add Employee Success Page:



Logging as Administrator:



After Logging as Administrator, will redirect to Get employee page where we Admin needs to enter Employee ID to retrieve the employee details:

Get Employee Information +

Not secure | awse2ewebsitefree.tk/getemp

Employee Information

Enter Employee ID

Employee ID:

Activate Windows

Display Employee Info Page:

Employee Information +

Not secure | awse2ewebsitefree.tk/fetchdata

Employee Information

Employee ID: 100

First Name: Sachin

Last Name: Tendulkar

Primary Skill: Angular JS

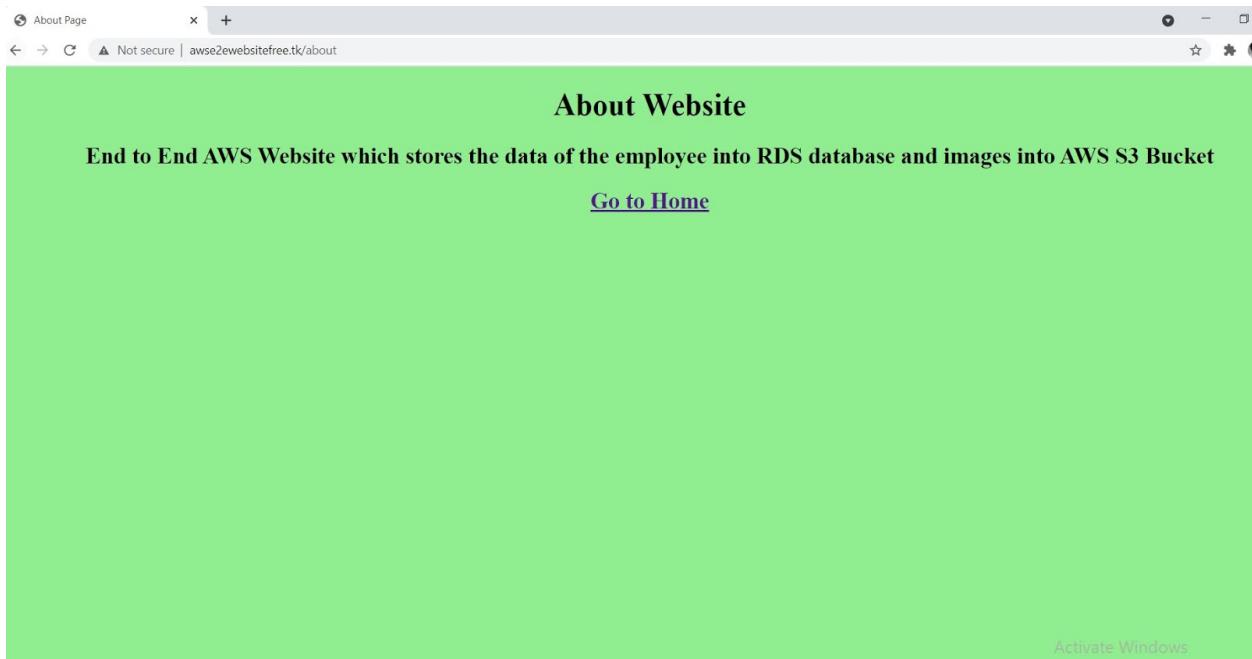
Location: California

Image



Activate Wind

About Page:



CHAPTER 7: OUTSTANDING FEATURES OF THIS PROJECT

1. Usage of different cloud services viz, EC2, S3, RDS, Route 53, IAM
2. Elimination of Infrastructure, and upfront investment.
3. Enhanced level of flexibility and scalability in comparison to traditional data centres.
4. Storing and retrieving of data in a fast and efficient manner.
5. Simple Project management.
6. Accessible from anywhere in the world and Increased productivity.

CHAPTER 8: FUTURE ENHANCEMENTS

1. Implementation and integration of AWS chatbot to our web application where employees can interact with it and have answers to their questions.
2. Implementation of AWS DynamoDB instead of MYSQL database, as AWS DynamoDB is a key-value and document database that delivers single-digit millisecond performance at any scale. DynamoDB can handle more than 10 trillion requests per day and can support peaks of more than 20 million requests per second. So, AWS DynamoDB can be one of our future enhancements.

CHAPTER 9: REFERENCES

1. <https://docs.aws.amazon.com/>
2. <https://docs.aws.amazon.com/s3/index.html>
3. <https://docs.aws.amazon.com/pythonsdk/>
4. <https://docs.aws.amazon.com/route53/>
5. [1] "Flask web development, one drop at a time",
<https://flask.palletsprojects.com/en/1.1.x/>
This is the official documentation of the Flask web application framework.
6. [2] "Introducing Python: modern computing in simple packages (Second edition.)", Bill Lubanovic, O'Reilly Media, November, 2019., Chapter 18.
This is a book that discusses the basic usage of Python programming language within various topics, including the web systems. Bill Lubanovic is a senior software engineer currently working for Internet Archive. O'Reilly is a popular publisher that has published various books related to software development.