Internship Portfolio

19011BC038
RADHA KRISHNA DESHPANDE
8 Sem B.Tech (DTDP)

About Me

Contact

9390936822

radhakrishna0801@gmail.com

NIZAMABAD

Education

JNAFAU
B.TECH DTDP
2019-2023

INTERMEDIATE
DELTA JUNIOR COLLAGE
2017-2019

SSC
KAKATIYA HIGH SCHOOL
2017

Skills

FRONTEND	82%
BACKEND	70%
SQL	50%
AWS	60%

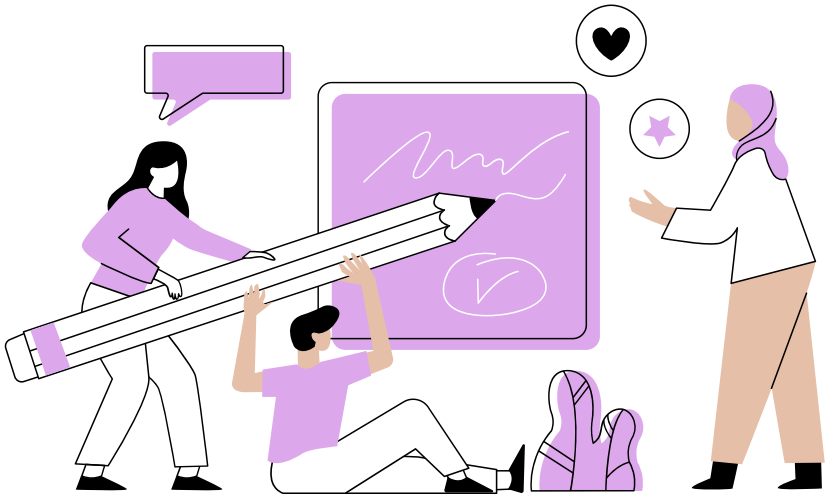
Carrer Objective

A highly motivated and experienced full stack developer with a strong background in machine learning. Seeking to utilize my skills and experience to contribute to the development and success of a dynamic and growing company by creating innovative and user-friendly web applications using machine learning techniques.

Social Media

https://www.linkedin.com/in/radha-krishna-deshpande-902281196/
@adhakrishnadeshpande663
@radha.krishna.deshpande
@radhakrishnadeshpande
https://resume-rkd.netlify.app/

CONTENTS



01 About the Company

Company name :Axiom Software Solutions Private Limited

At the start of the 21st century, two critical routes to securing talent and expertise were hiring permanent staff and engaging contingent workers. Fast forward to the present, and we have plenty of other opportunities. We now live in an open talent economy, bringing a more flexible and dynamic approach to talent acquisition. There are many more routes to source qualified candidates, and new ones continue to open as workforce dynamics and business needs are changing.The company offers various services like software solutions to different industries, corporate training, workshops, paper writing and internships.

Hyndava Techno Park, Plot No. 12, Survey No. 64, Madhapur Village, Serilingampally Mandal Sector 3, Phase 2, HUDA Techno Enclave, Opp. Rheja Mind space, Hitch City, Hyderabad, Telangana - 500081

Axiom Software Solutions Private Limited A-439, IT/ITES Building, Sector 132, Noida-201301, UP, India

+91 9581056666

Info@axiomsoftwaresolutions.com

www.axiomsoftwaresolutions.com

02 Nature of Work

I have worked as a PYTHON PROGRAM DEVELOPER and have done works in AWS CLOUD .For the back end portion, I have introduced to the Django framework, an open-source Python web framework. Django simplifies web development through its high-level abstractions, adhering to the "Don't Repeat Yourself" (DRY) principle. It offers features such as URL routing, ORM (Object-Relational Mapping) via Django ORM for database modeling, user authentication management, and a customizable admin interface for data administration. During your internship, i likely worked with a database, MySQL.
And also done work in the AWS CLOUD to deploy the websites in the server using lambda,EC2 and S3 .

03 Work Done in Django

```python
class Subject(models.Model):
    subject_title = models.CharField(max_length=250)
    subject_code = models.CharField(max_length=10)

    def __str__(self):
        return self.subject_title


class Student(models.Model):
    first_name = models.CharField(max_length=25)
    middle_name = models.CharField(max_length=25)
    surname = models.CharField(max_length=25)
    subject_taking = models.ForeignKey(Subject, on_delete=models.CASCADE)

    def __str__(self):
        return f"{self.surname}, {self.first_name} {self.middle_name}"
```

This code is a Django ORM for practising .

```
(k1) PS D:\Django_Projects\k1\myproject> python manage.py sqlmigrate ormapp 0001
BEGIN;
--
-- Create model Customer
--
CREATE TABLE `ormapp_customer` (`id` integer AUTO_INCREMENT NOT NULL PRIMARY KEY, `name` varchar(255) NOT NULL);
--
-- Create model Vehicle
--
CREATE TABLE `ormapp_vehicle` (`id` integer AUTO_INCREMENT NOT NULL PRIMARY KEY, `name` varchar(255) NOT NULL, `customer_id` integer NO
T NULL UNIQUE);
ALTER TABLE `ormapp_vehicle` ADD CONSTRAINT `ormapp_vehicle_customer_id_f3729894_fk_ormapp_customer_id` FOREIGN KEY (`customer_id`) REF
ERENCES `ormapp_customer` (`id`);
COMMIT;
(k1) PS D:\Django_Projects\k1\myproject>
```

```python
from django.db import models

#DataFlair #DjangoTutorials
# Create your models here.

class Customer(models.Model):
    name = models.CharField(max_length=255)

class Vehicle(models.Model):
    name = models.CharField(max_length=255)
    customer = models.ForeignKey(
        Customer,
        on_delete=models.CASCADE,
        related_name='Vehicle'
    )
```

03 Work Done in Django

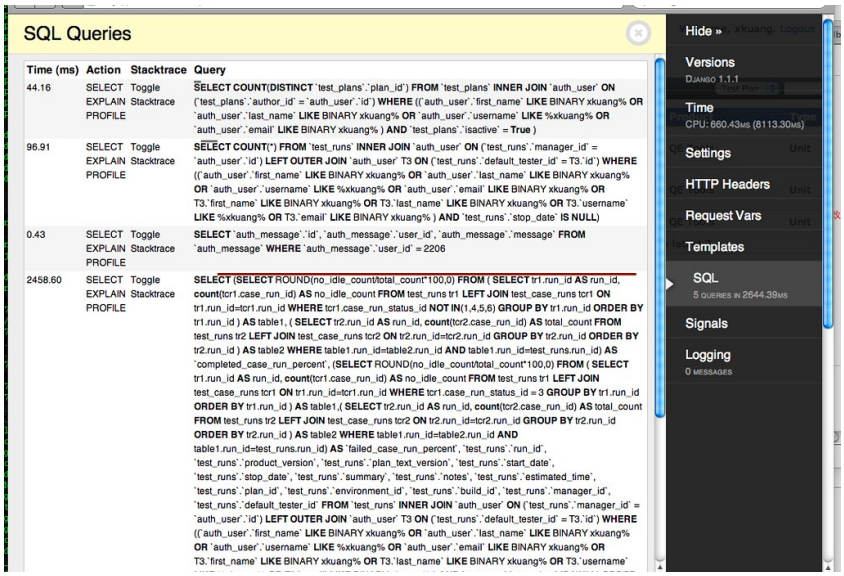
```python
from django.test import TestCase
from whatever.models import Whatever
from django.utils import timezone
from django.core.urlresolvers import reverse
from whatever.forms import WhateverForm

# models test
class WhateverTest(TestCase):

    def create_whatever(self, title="only a test", body="yes, this is only a test"):
        return Whatever.objects.create(title=title, body=body, created_at=timezone.now())

    def test_whatever_creation(self):
        w = self.create_whatever()
        self.assertTrue(isinstance(w, Whatever))
        self.assertEqual(w.__unicode__(), w.title)
```

Testing the code in the given company requirements.

```python
# views (uses selenium)

#pip install selenium==2.33.0

import unittest
from selenium import webdriver

class TestSignup(unittest.TestCase):

    def setUp(self):
        self.driver = webdriver.Firefox()

    def test_signup_fire(self):
        self.driver.get("http://localhost:8000/add/")
        self.driver.find_element_by_id('id_title').send_keys("test title")
        self.driver.find_element_by_id('id_body').send_keys("test body")
        self.driver.find_element_by_id('submit').click()
        self.assertIn("http://localhost:8000/", self.driver.current_url)

    def tearDown(self):
        self.driver.quit

if __name__ == '__main__':
    unittest.main()
```

Testing the code in the given company requirements.I have to not display the company database .

03 Work Done in Django



This is refered pic of django tool bar .

```python
if DEBUG:
    INSTALLED_APPS += ['debug_toolbar',]
    MIDDLEWARE += ['debug_toolbar.middleware.DebugToolbarMiddleware',]
    INTERNAL_IPS = [
        '127.0.0.1',
    ]

from django.conf import settings
if settings.DEBUG:
    import debug_toolbar
    urlpatterns += [
        path('__debug__/', include(debug_toolbar.urls)),
    ]

```

Tool Bars referenced code .

```python
from django.test import SimpleTestCase
from django.urls import reverse, resolve

from .views import HomePageView

class HomepageTests(SimpleTestCase):
    # def test_url_exists_at_correct_location(self):
    #     response = self.client.get("/")
    #     self.assertEqual(response.status_code, 200)
    # def test_homepage_url_name(self):
    #     response = self.client.get(reverse("home"))
    #     self.assertEqual(response.status_code, 200)
    # def test_homepage_template(self):
    #     response = self.client.get("/")
    #     self.assertTemplateUsed(response, "home.html")
    # def test_homepage_contains_correct_html(self): # new
    #     response = self.client.get("/")
    #     self.assertContains(response, "home page")
    # def test_homepage_does_not_contain_incorrect_html(self): # new
    #     response = self.client.get("/")
    #     self.assertNotContains(response, "Hi there! I should not be on the page.")
    # Change it to DRY code
    def setUp(self):
        url = reverse("home")
        self.response = self.client.get(url)
    def test_url_exists_at_correct_location(self):
        self.assertEqual(self.response.status_code, 200)
    def test_homepage_template(self):
        self.assertTemplateUsed(self.response, "home.html")
    def test_homepage_contains_correct_html(self):
        self.assertContains(self.response, "EPMS")
    def test_homepage_does_not_contain_incorrect_html(self):
        self.assertNotContains(self.response, "Hi there! I should not be on the page.")
    def test_homepage_url_resolves_homepageview(self):
        view = resolve("/")
        self.assertEqual(view.func.__name__, HomePageView.as_view().__name__)


```

03 Work Done in Django

```python
import os

ENVIRONMENT = os.getenv('ENVIRONMENT', 'development')

DEBUG = True
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
SECRET_KEY = '-05sgp9!deq=q1nltm@^^2cc+v29i(tyybv3v2t77qi66czazj'
ALLOWED_HOSTS = []

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.sites',
    'allauth',
    'allauth.account',
    'allauth.socialaccount',
    'allauth.socialaccount.providers.google',
    'crispy_forms',
    'django_countries',
    'core'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware'
]

ROOT_URLCONF = 'demo.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_L10N = True
USE_TZ = True

# static files (CSS, JS, Image)

STATIC_URL = '/static/'
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static_in_env')]
STATIC_ROOT = os.path.join(BASE_DIR, 'static_root')
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media_root')

DATABASES = {
```

```python

DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.sqlite3",
        "NAME": os.path.join(BASE_DIR, 'db.sqlite3')
    }
}

if ENVIRONMENT == 'production':
    DEBUG = True
    SECRET_KEY = os.getenv('SECRET_KEY')
    SESSION_COOKIE_SECURE = True
    SECURE_BROWSER_XSS_FILTER = True
    SECURE_CONTENT_TYPE_NOSNIFF = True
    SECURE_HSTS_INCLUDE_SUBDOMAINS = True
    SECURE_HSTS_SECONDS = 31536000
    SECURE_REDIRECT_EXEMPT = []
    SECURE_SSL_REDIRECT = True
    SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTO', 'https')

# Auth
AUTHENTICATION_BACKENDS = (
    'django.contrib.auth.backends.ModelBackend',
    'allauth.account.auth_backends.AuthenticationBackend'
)

SITE_ID = 1
LOGIN_REDIRECT_URL = '/'

# Provider specific settings
SOCIALACCOUNT_PROVIDERS = {
    'google': {
        # For each OAuth based provider, either add a ``SocialApp``
        # (``socialaccount`` app) containing the required client
        # credentials, or list them here:
        'APP': {
            'client_id': '123',
            'secret': '456',
            'key': '666'
        }
    }
}

# CRISPY FORM

CRISPY_TEMPLATE_PACK = 'bootstrap4'

STRIPE_PUBLIC_KEY = 'pk_test_lX3r6OMjOU2yzFsN5Hq6belT00EY82kZmH'
STRIPE_SECRET_KEY = 'sk_test_tn0CTDaIJHUJvAghsf39cfsC00LNjsqDnb'
```

This code is a Django importing the files from the requirement side and linking up with the requirements which are essential.

03 Work Done in Django

```python
# flake8: noqa
from .settings import *

DEBUG = True
ALLOWED_HOSTS += ['*']
WSGI_APPLICATION = 'market.wsgi.application'

AUTH_PASSWORD_VALIDATORS = [
    {'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator'},
    {'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator'},
    {'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator'},
    {'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator'},
]

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': os.getenv('APP_DB_NAME'),
        'USER': '{}@{}'.format(os.getenv('POSTGRES_ADMIN_USER'), os.getenv('POSTGRES_SERVER_NAME')),
        'PASSWORD': os.getenv('POSTGRES_ADMIN_PASSWORD'),
        'HOST': os.getenv('POSTGRES_HOST'),
        'PORT': '5432',
        'OPTIONS': {'sslmode': 'require'},
    }
}

STATICFILES_STORAGE = 'storages.backends.azure_storage.AzureStorage'
AZURE_ACCOUNT_NAME = os.getenv('AZ_STORAGE_ACCOUNT_NAME')
AZURE_CONTAINER = os.getenv('AZ_STORAGE_CONTAINER')
AZURE_ACCOUNT_KEY = os.getenv('AZ_STORAGE_KEY')
```

This code is a Django settings file that configures database, static file storage, and authentication settings for the project. Which I have developed

04 Work Done in Frontend & SQL

```javascript
// 1. Selecting an element and changing its text content
const element = document.getElementById('example');
element.textContent = 'Updated text';

// 2. Handling a button click event
const button = document.getElementById('btn');
button.addEventListener('click', () => {
  alert('Button clicked!');
});

// 3. Making an AJAX request using fetch API
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => {
    console.log(data);
  })
  .catch(error => {
    console.error('Error:', error);
  });

// 4. Dynamically creating an HTML element
const newElement = document.createElement('div');
newElement.textContent = 'New element';
document.body.appendChild(newElement);

// 5. Changing the CSS style of an element
const element = document.getElementById('example');
element.style.color = 'red';
element.style.fontSize = '20px';

// 6. Handling form submission
const form = document.getElementById('myForm');
form.addEventListener('submit', event => {
  event.preventDefault();
  const formData = new FormData(form);
  // Process form data or send it to the server
});

// 7. Adding a CSS class to an element
const element = document.getElementById('example');
element.classList.add('highlight');

// 8. Manipulating the browser's local storage
localStorage.setItem('key', 'value');
const storedValue = localStorage.getItem('key');
localStorage.removeItem('key');

// 9. Redirecting to a different page
window.location.href = 'https://example.com';
```

Front end code which is refference not the companies.

```sql
/-- 1. Creating a table
CREATE TABLE customers (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    email VARCHAR(100)
);

-- 2. Inserting data into a table
INSERT INTO customers (id, name, email)
VALUES (1, ' ', 'axiomsoftwaresolution@gmail.com');

-- 3. Querying data from a table
SELECT * FROM customers WHERE id = 1;

-- 4. Updating data in a table
UPDATE customers SET email = '' WHERE id = 1;

-- 5. Deleting data from a table
DELETE FROM customers WHERE id = 1;

-- 6. Creating an index on a column
CREATE INDEX idx_customers_email ON customers (email);

-- 7. Joining tables
SELECT c.name, o.order_number
FROM customers c
JOIN orders o ON c.id = o.customer_id;

-- 8. Aggregating data with GROUP BY
SELECT category, COUNT(*) AS count
FROM products
GROUP BY category;

-- 9. Using subqueries
SELECT name
FROM products
WHERE category IN (SELECT category FROM popular_categories);
```

SQL code

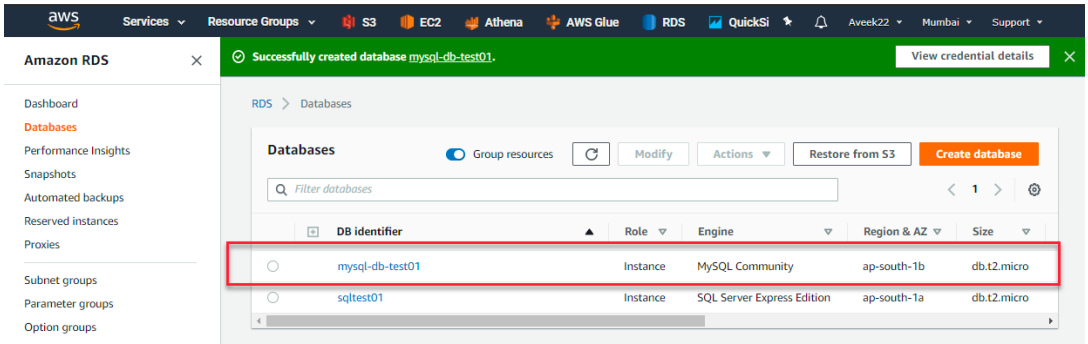
04 Work Done in Frontend & SQL

```javascript
// 1. Selecting elements and applying CSS styles
const elements = document.querySelectorAll('.item');
elements.forEach(element => {
  element.style.backgroundColor = 'red';
});

// 2. Handling a form input change event
const input = document.getElementById('myInput');
input.addEventListener('input', event => {
  const value = event.target.value;
  console.log('Input value:', value);
});

// 3. Creating and appending a new list item
const list = document.getElementById('myList');
const newItem = document.createElement('li');
newItem.textContent = 'New Item';
list.appendChild(newItem);

// 4. Fetching data from an API using async/await
async function fetchData() {
  try {
    const response = await fetch('https://api.example.com/data');
    const data = await response.json();
    console.log('Fetched data:', data);
  } catch (error) {
    console.error('Error:', error);
  }
}
fetchData();

// 5. Adding event listeners to multiple elements
const buttons = document.querySelectorAll('.btn');
buttons.forEach(button => {
  button.addEventListener('click', () => {
    console.log('Button clicked!');
  });
});

// 6. Toggling a CSS class on an element
const element = document.getElementById('myElement');
element.addEventListener('click', () => {
  element.classList.toggle('active');
});

// 7. Animating an element using CSS transitions
const box = document.getElementById('myBox');
box.style.transition = 'transform 0.3s';
box.addEventListener('click', () => {
  box.style.transform = 'translateX(100px)';
```

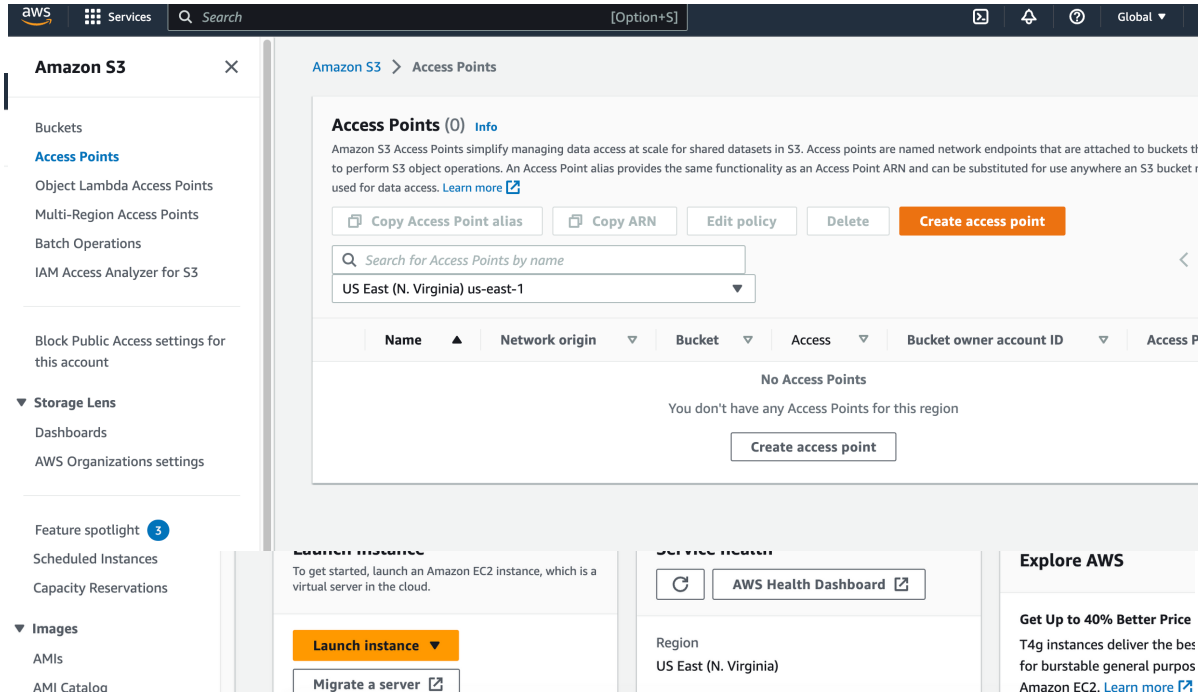
```sql
-- 1. Creating a table with foreign key constraint
CREATE TABLE orders (
  id INT PRIMARY KEY,
  customer_id INT,
  order_date DATE,
  FOREIGN KEY (customer_id) REFERENCES customers(id)
);

-- 2. Inserting multiple rows into a table
INSERT INTO orders (id, customer_id, order_date)
VALUES
  (1, 1, '2023-05-01'),
  (2, 2, '2023-05-02'),
  (3, 1, '2023-05-03');

-- 3. Querying data from multiple tables using JOIN
SELECT o.id, c.name, o.order_date
FROM orders o
JOIN customers c ON o.customer_id = c.id;

-- 4. Updating data in a table based on a condition
UPDATE products SET price = price * 1.1 WHERE category = 'Electronics';

-- 5. Deleting all records from a table
DELETE FROM customers;

-- 6. Using ORDER BY to sort query results
SELECT * FROM products ORDER BY price DESC;

-- 7. Using aggregate functions to calculate statistics
SELECT AVG(price) AS average_price, MAX(price) AS max_price FROM products;

-- 8. Limiting the number of results in a query
SELECT * FROM products LIMIT 10;

-- 9. Using subqueries to filter results
SELECT name, price
FROM products
WHERE price > (SELECT AVG(price) FROM products);
```

Referenced code pictures.

05 Work Done in AWS CLOUD

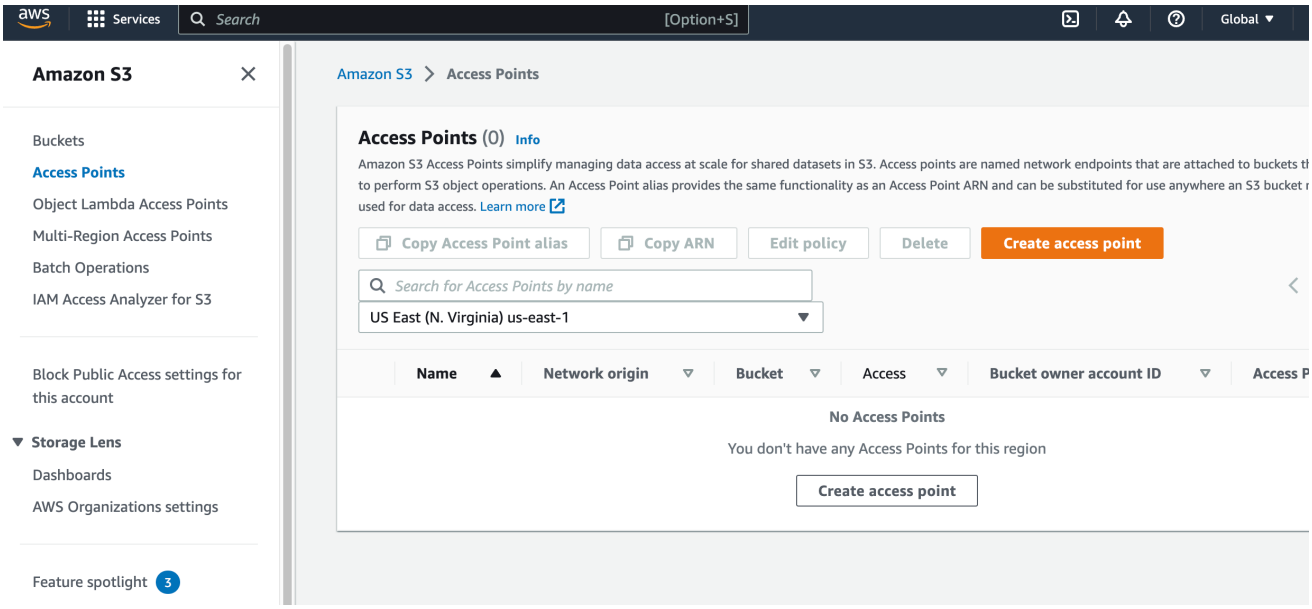


Practising the RDS(Relative DataBase System

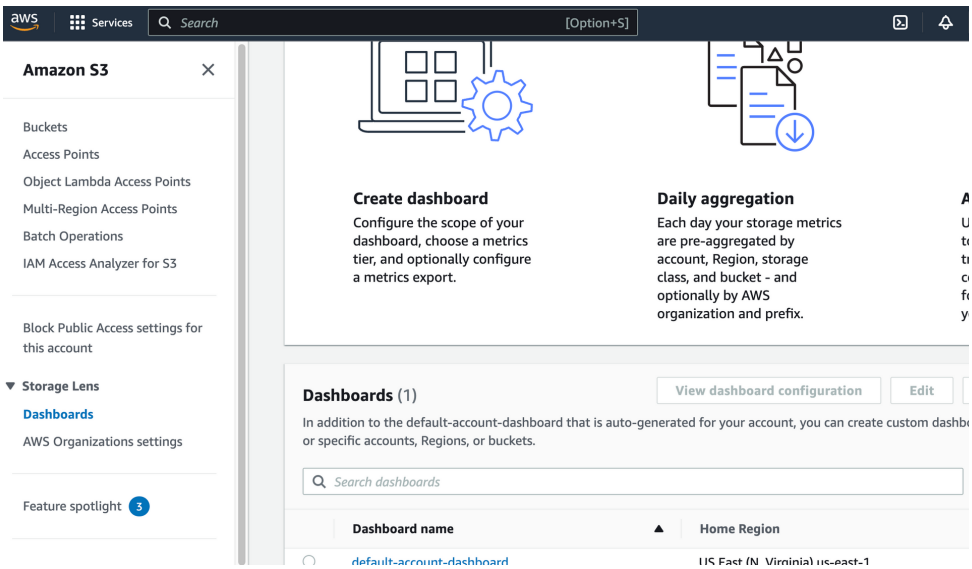


EC2 instace is best way of uploading the dyanmic website.It provides virtual servers (instances) that can be quickly provisioned and configured to meet various computing needs. EC2 instances offer flexible computing power, storage options, security features, and allow easy scalability to accommodate changing workload demands in the cloud.

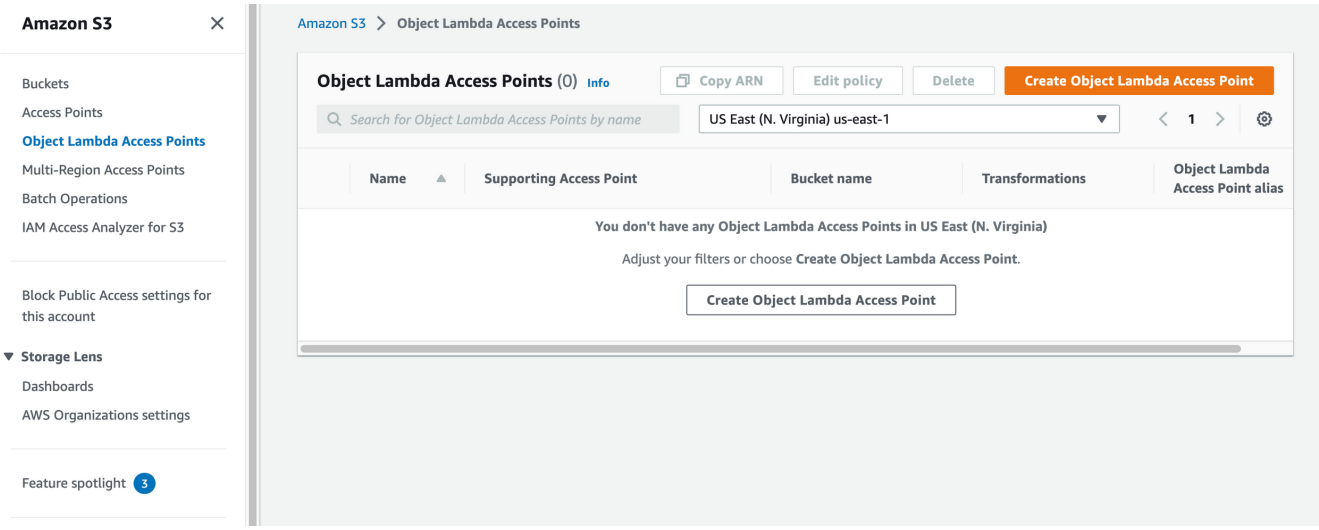
05 Work Done in AWS CLOUD



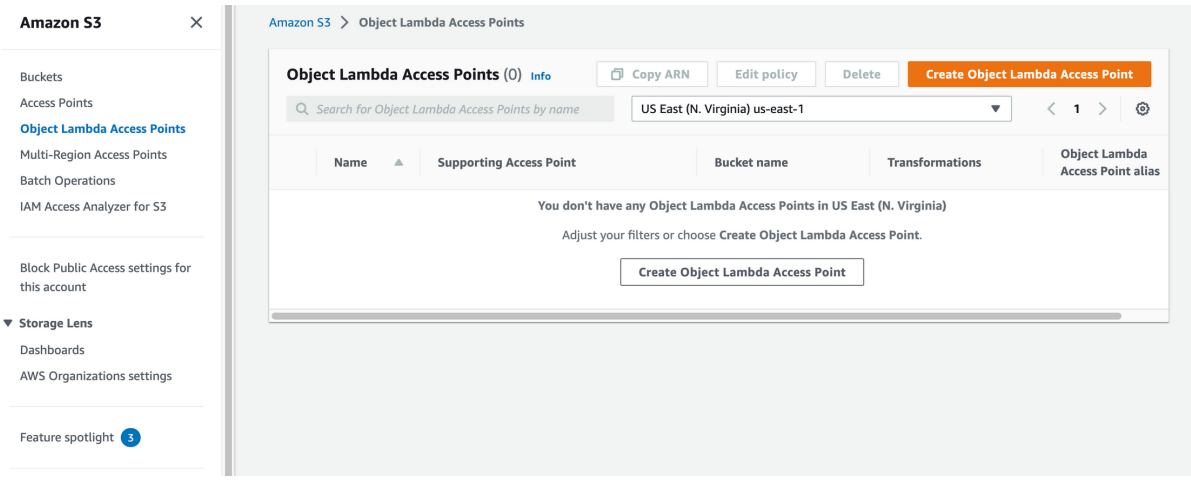
Amazon S3 (Simple Storage Service) is a highly scalable cloud storage service provided by AWS. It allows you to store and retrieve any amount of data from anywhere on the web. S3 provides a secure, durable, and highly available storage infrastructure, with data stored in multiple facilities and protected against hardware failures. It offers various features such as versioning, data encryption, lifecycle management, and access control policies. S3 is widely used for a range of applications, including data backup, content distribution, static website hosting, and big data analytics.



05 Work Done in AWS CLOUD

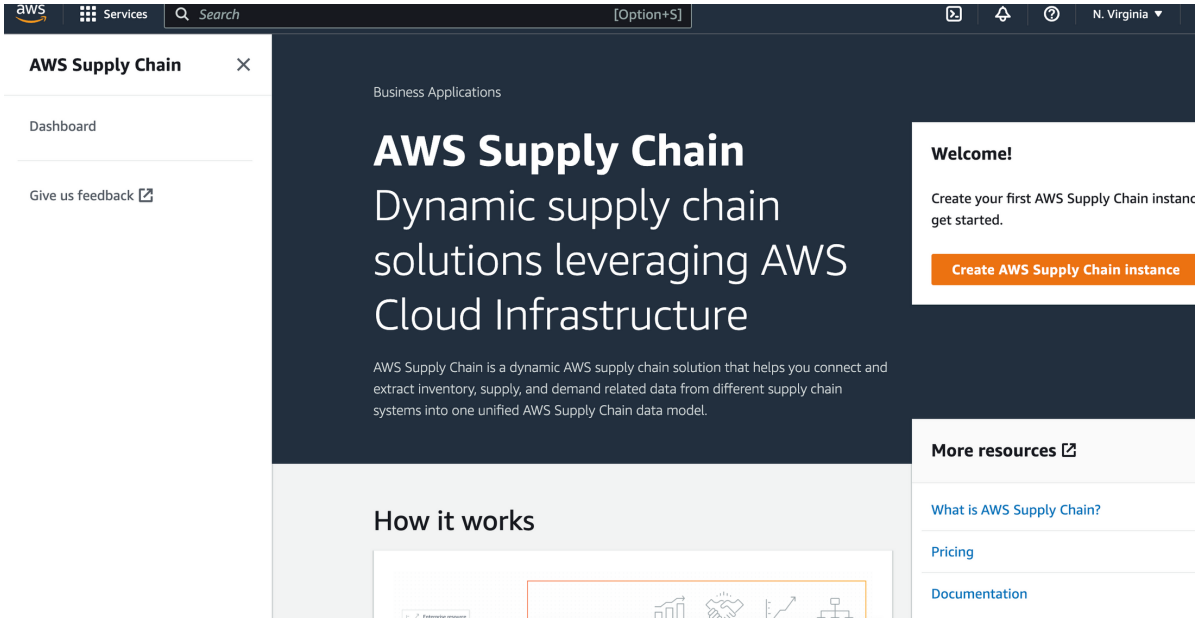


AWS Lambda enables serverless compute, running code without servers. Access is controlled via IAM policies, allowing secure execution and integration with other AWS services. CloudWatch monitors Lambda functions, providing logging and metrics for insights into performance.

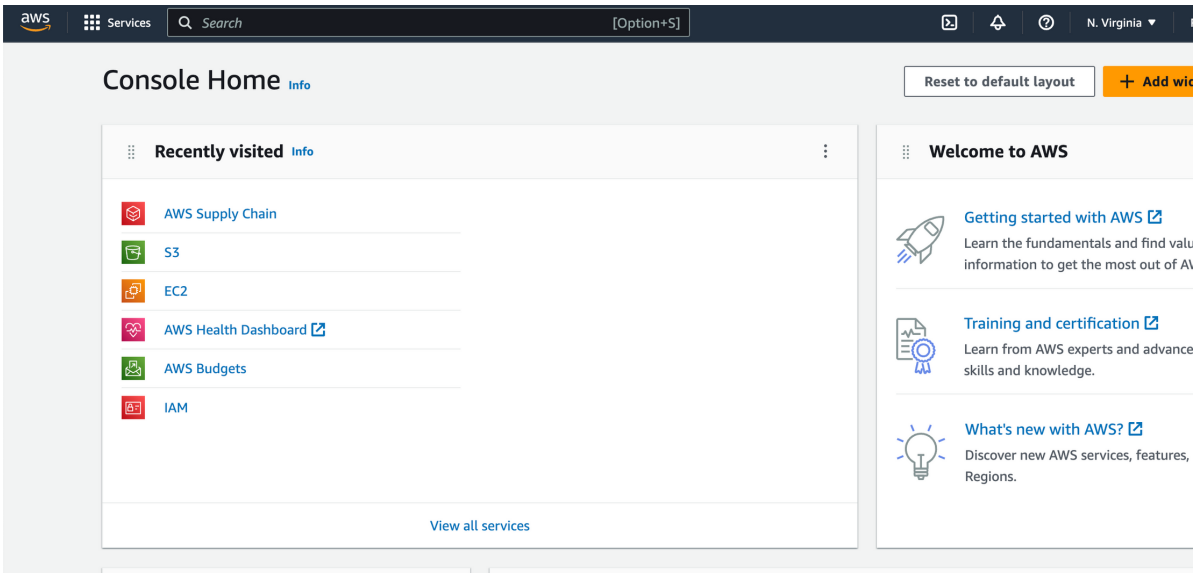


AWS Object Lambda Access Point is a feature in Amazon S3 that allows you to add custom code to process and modify data as it is retrieved, enabling dynamic transformations and custom responses.

05 Work Done in AWS CLOUD



AWS Supply Chain optimizes operations using AWS services like S3, Redshift, Lambda, Forecast, and Rekognition, enabling visibility, efficiency, and scalability. Integration with 3PLs enhances collaboration.



AWS Console Home is the web-based interface provided by Amazon Web Services (AWS) where users can access and manage their AWS services, configure settings, monitor resources, and access various tools and services for cloud computing.

06 Acquired Skills

My proficiency in HTML, CSS, and JavaScript empowers me to construct the foundation of web pages with semantic markup, style them precisely, and add interactive elements for enhanced user experiences. When it comes to Django, I can leverage its powerful framework to develop robust web applications, utilizing my skills in front-end frameworks like Bootstrap and responsive web design principles. Additionally, I have gained expertise in AWS services, allowing me to utilize cloud hosting, scalability, and storage solutions for efficient deployment. With MySQL, I can effectively design and manage databases, seamlessly integrating them with Django for seamless data storage and retrieval, enabling reliable and efficient web applications.

06 Acquired Skills

In addition to my Django expertise, I have also developed proficiency in utilizing AWS services for web application deployment, scalability, and storage. I am well-versed in leveraging AWS Console to manage resources, configure settings, and monitor application performance. Moreover, my experience with MySQL has allowed me to efficiently design and optimize databases for seamless data storage and retrieval. By combining my knowledge of Django, AWS, and MySQL, I am equipped to create powerful and reliable web applications with intuitive user interfaces, ensuring exceptional user experiences.

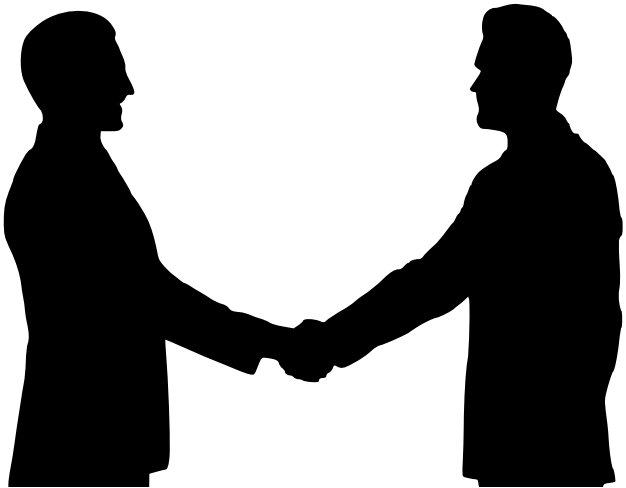
07 Final Conclusion

In conclusion, my web development internship has been a transformative journey that has significantly enhanced my skills in both front-end and back-end development. Through practical experience, I have gained a deep understanding of HTML, CSS, and JavaScript, allowing me to construct web pages with semantic markup, precise styling, and interactive elements.

Moreover, my internship introduced me to the powerful Django framework and SQL databases, which have been instrumental in my growth as a back-end developer. Working with Django, I have learned to develop robust and scalable web applications, utilizing its MVC architecture and extensive feature set.

Additionally, my experience with SQL databases has equipped me with the ability to efficiently manage and interact with data, ensuring data integrity and enabling sophisticated queries. Furthermore, I have gained valuable knowledge and skills in AWS services such as S3 and EC2. Leveraging S3 for storage and EC2 for hosting, I have learned to deploy and scale web applications in a flexible and reliable manner.

Overall, my internship has not only expanded my technical expertise but has also fostered important skills such as problem-solving, teamwork, and attention to detail. With my proficiency in HTML, CSS, JavaScript, Django, SQL, and knowledge of AWS services like S3 and EC2, I am confident in my ability to contribute effectively to future web development projects and deliver exceptional results.



SCAN FOR GETTING INTO THE WEBSITES OF MY PROFILE

19011BCO38
RADHA KRISHNA DESHPANDE

8SEM
B.TECH (DTDP)