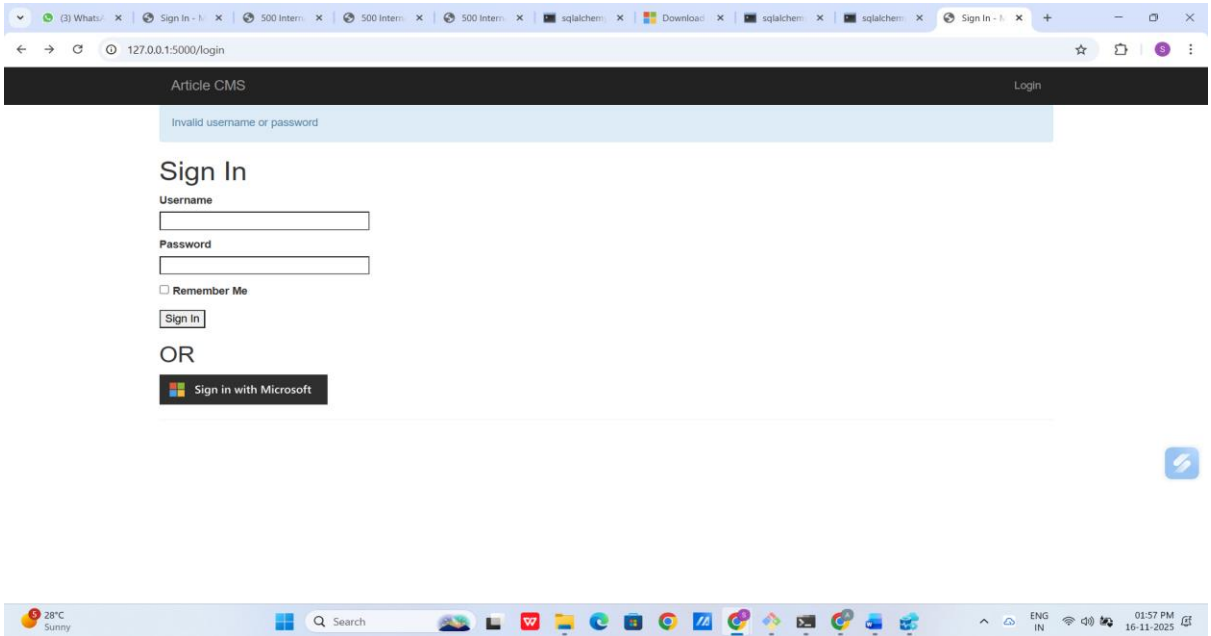
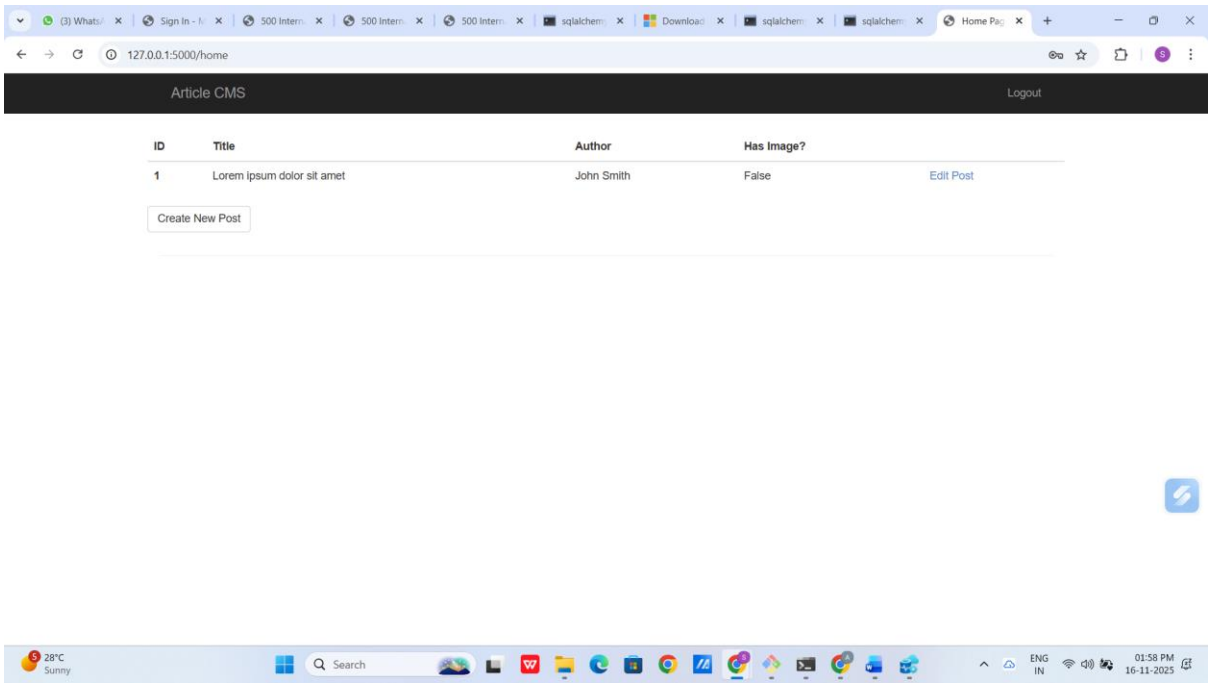
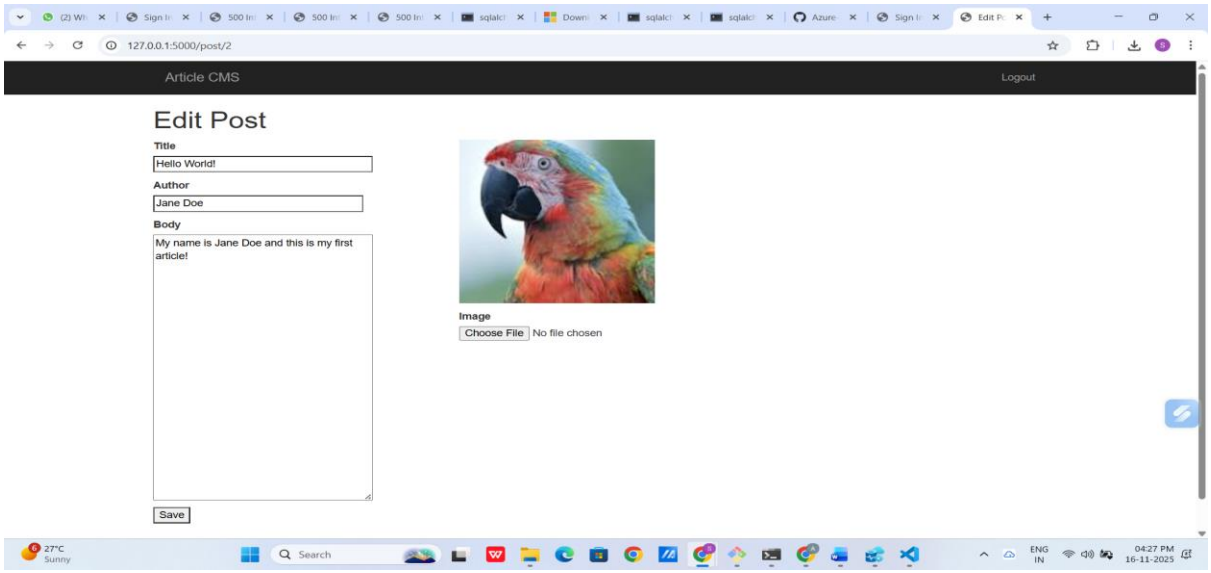
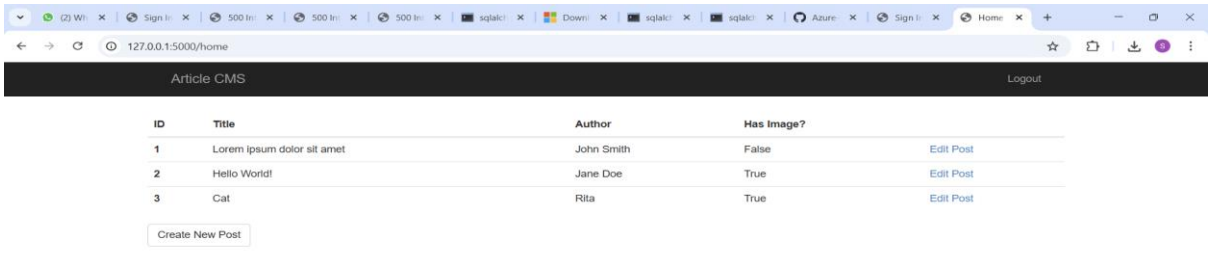
Project: Deploy An Article CMS to Azure

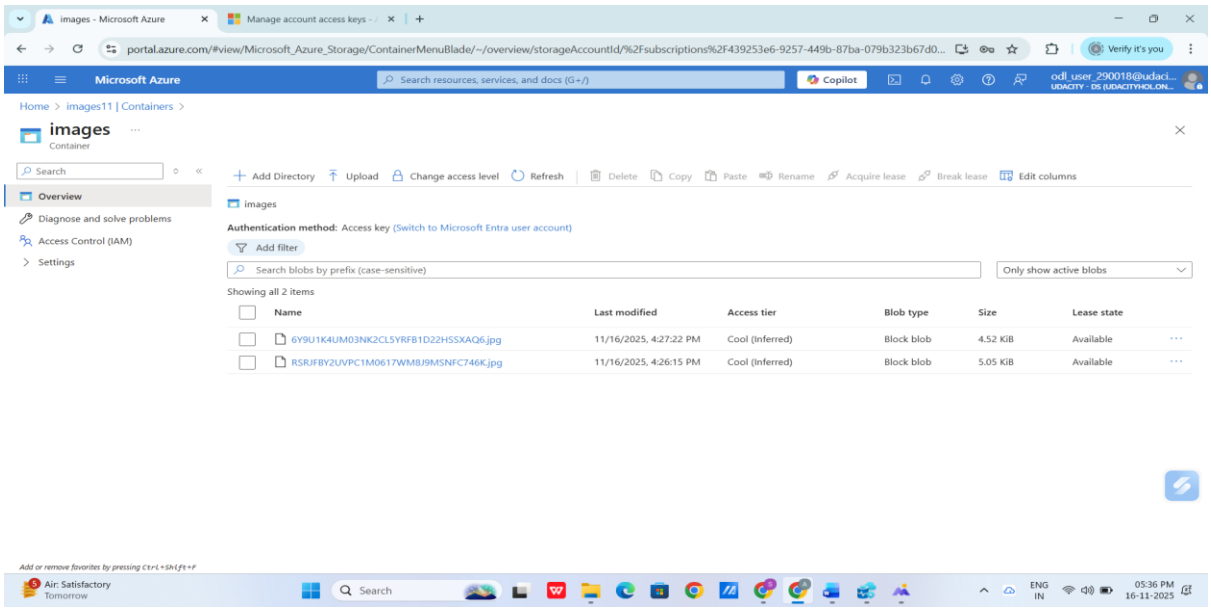
1. **A screenshot of an article created in the Article CMS on Azure. The screenshot must also include the URL. The article should have the following fields set:**
    - **Title: "Hello World!"**
    - **Author: "Jane Doe"**
    - **Body: "My name is Jane Doe and this is my first article!"**
    - **An image of your choice. It must be either a .png or .jpg.**



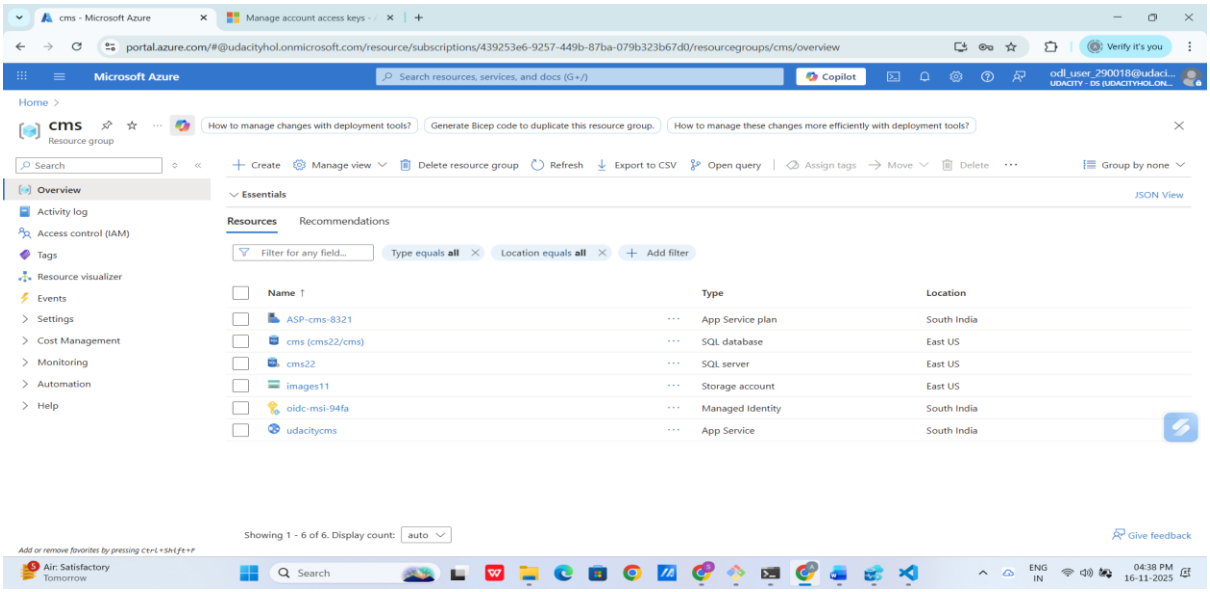



127.0.0.1:5000/post/2
Article CMS
Logout
Edit Post
Title
Hello World!
Author
Jane Doe
Body
My name is Jane Doe and this is my first article!
Image
Choose File
No file chosen
Save

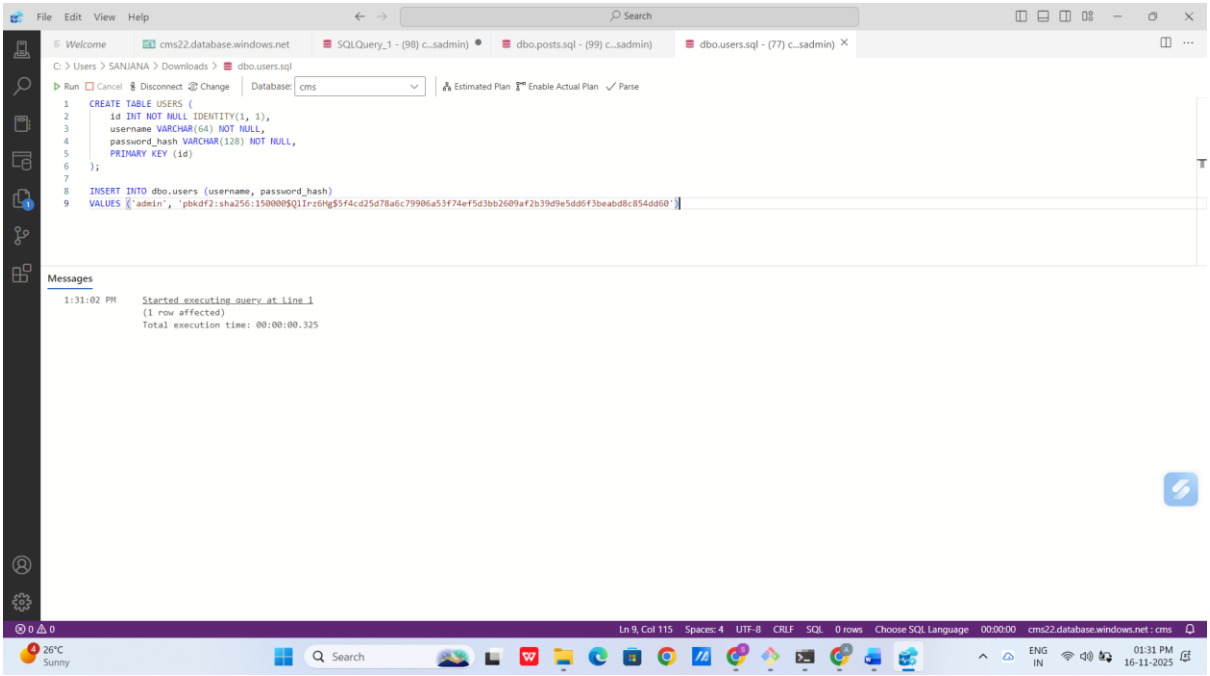

127.0.0.1:5000/home
Article CMS
Logout
ID
Title
Author
Has Image?
1
Lorem ipsum dolor sit amet
John Smith
False
Edit Post
2
Hello World!
Jane Doe
True
Edit Post
3
Cat
Rita
True
Edit Post
Create New Post

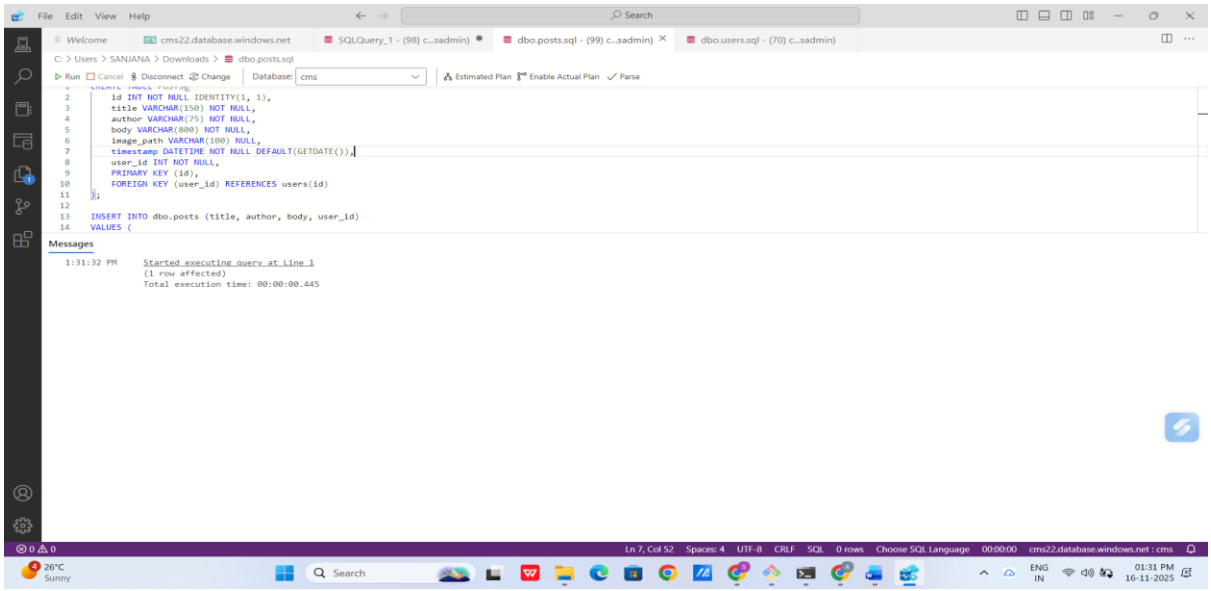

images - Microsoft Azure
Manage account access keys
portal.azure.com/#view/Microsoft_Azure_Storage/ContainerMenuBlade/~/overview/storageAccountId/%2Fsubscriptions%2F439253e6-9257-449b-87ba-079b323b67d0...
Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
Home > images11 | Containers >
images
Container
Overview
Diagnose and solve problems
Access Control (IAM)
Settings
Add Directory
Upload
Change access level
Refresh
Delete
Copy
Paste
Rename
Acquire lease
Break lease
Edit columns
Authentication method: Access key (Switch to Microsoft Entra user account)
Add filter
Search blobs by prefix (case-sensitive)
Only show active blobs
Showing all 2 items
Name
Last modified
Access tier
Blob type
Size
Lease state
6Y9U1K4UM03NK2CL5YRFB1D22HSSXAQ6.jpg
11/16/2025, 4:27:22 PM
Cool (Inferred)
Block blob
4.52 KiB
Available
RSRJFBY2UVPC1M0617WM8J9MSNFC746K.jpg
11/16/2025, 4:26:15 PM
Cool (Inferred)
Block blob
5.05 KiB
Available

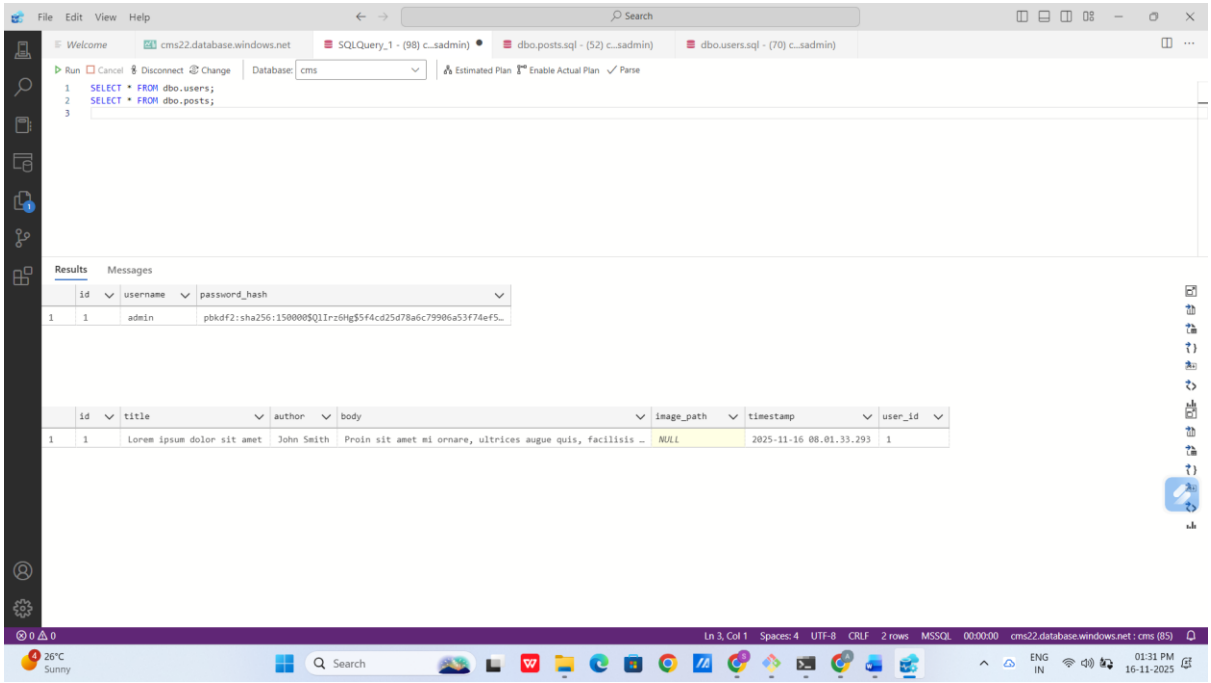
2. **A screenshot of the resource group from the Azure Portal including all of the resources that were created to complete this project. (see sample screenshot above).**

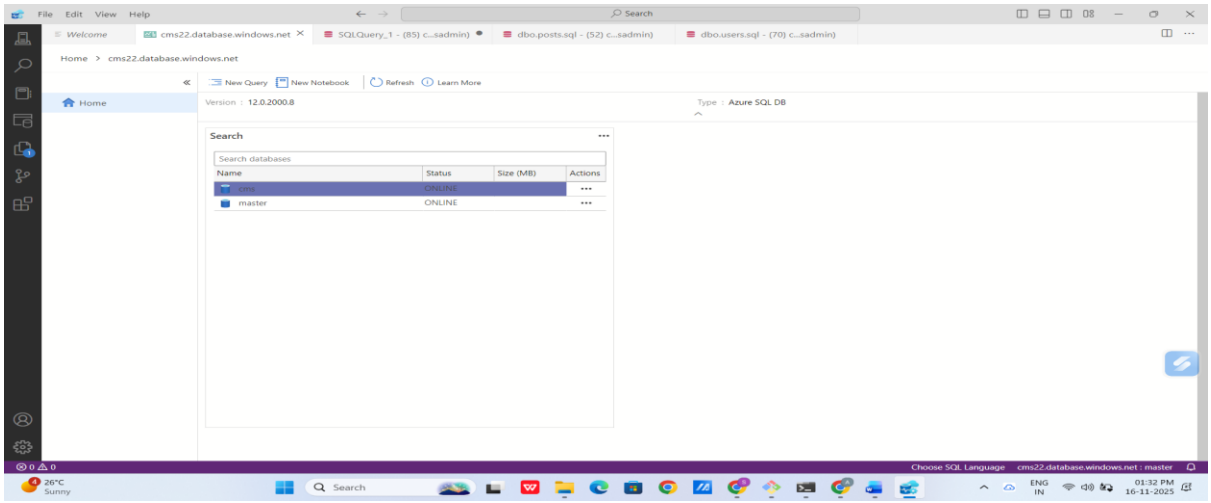


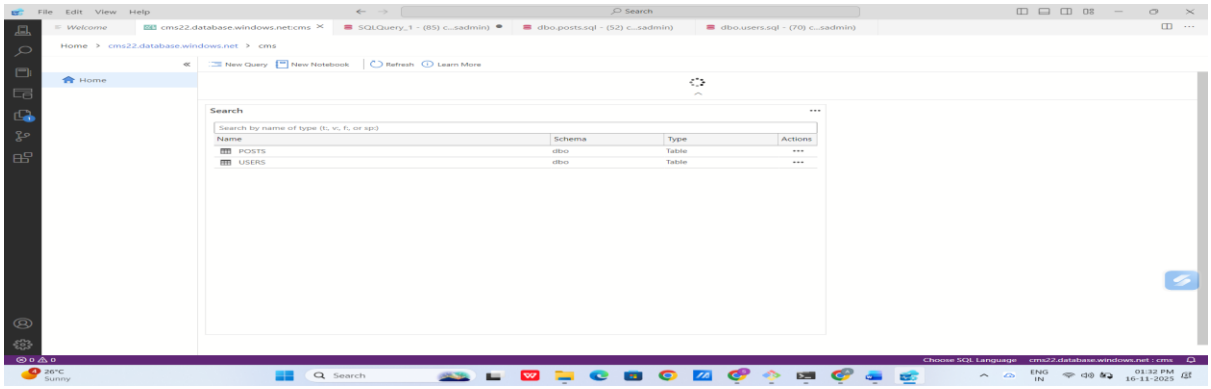
3. **A screenshot showing the created tables and one query of data from the initial scripts in the SQL database (see example in the project repository).**

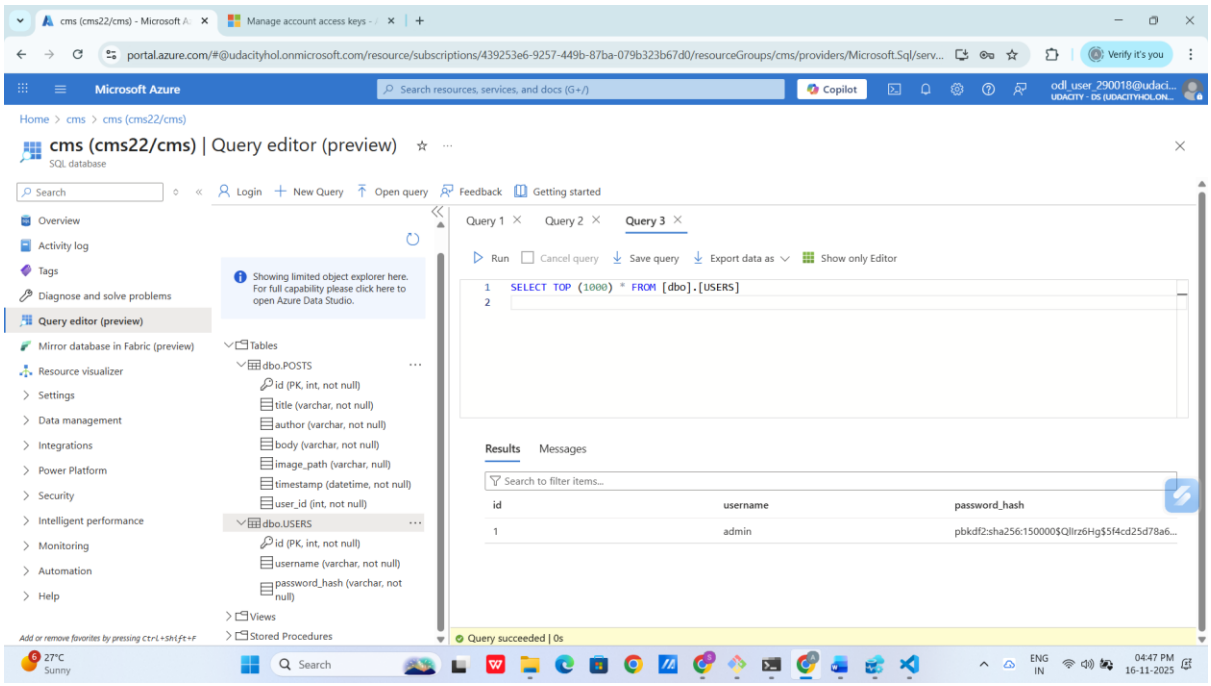


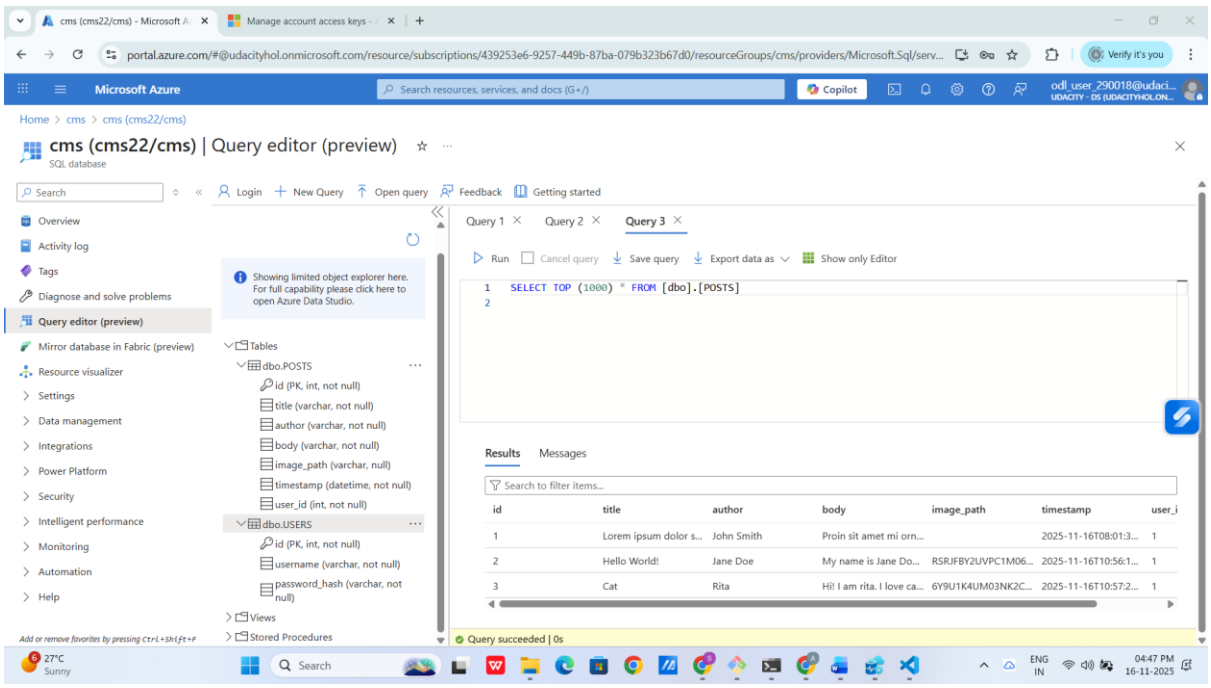

File Edit View Help
Search
Welcome
cms22.database.windows.net
SQLQuery_1 - (98) c...sadmin)
dbo.posts.sql - (99) c...sadmin)
dbo.users.sql - (70) c...sadmin)
Run Cancel Disconnect Change
Database: cms
Estimated Plan Enable Actual Plan Parse
id INT NOT NULL IDENTITY(1, 1),
title VARCHAR(150) NOT NULL,
author VARCHAR(75) NOT NULL,
body VARCHAR(800) NOT NULL,
image_path VARCHAR(100) NULL,
timestamp DATETIME NOT NULL DEFAULT(GETDATE()),
user_id INT NOT NULL,
PRIMARY KEY (id),
FOREIGN KEY (user_id) REFERENCES users(id)
);
INSERT INTO dbo.posts (title, author, body, user_id)
VALUES (
Messages
1:31:32 PM Started executing query at Line 1
(1 row affected)
Total execution time: 00:00:00.445
Ln 7, Col 52 Spaces: 4 UTF-8 CRLF SQL 0 rows Choose SQL Language 00:00:00 cms22.database.windows.net : cms
26°C Sunny
Search
ENG IN
01:31 PM 16-11-2025


File Edit View Help
Search
Welcome
cms22.database.windows.net
SQLQuery_1 - (98) c...sadmin)
dbo.posts.sql - (52) c...sadmin)
dbo.users.sql - (70) c...sadmin)
Run Cancel Disconnect Change
Database: cms
Estimated Plan Enable Actual Plan Parse
SELECT * FROM dbo.users;
SELECT * FROM dbo.posts;
Results
Messages
id username password_hash
1 1 admin pbkdf2:sha256:150000$Q1Irz6Hg$5f4cd25d78a6c79906a53f74ef5...
id title author body image_path timestamp user_id
1 1 Lorem ipsum dolor sit amet John Smith Proin sit amet mi ornare, ultrices augue quis, facilisis ... NULL 2025-11-16 08.01.33.293 1
Ln 3, Col 1 Spaces: 4 UTF-8 CRLF 2 rows MSSQL 00:00:00 cms22.database.windows.net : cms (85)
26°C Sunny
Search
ENG IN
01:31 PM 16-11-2025

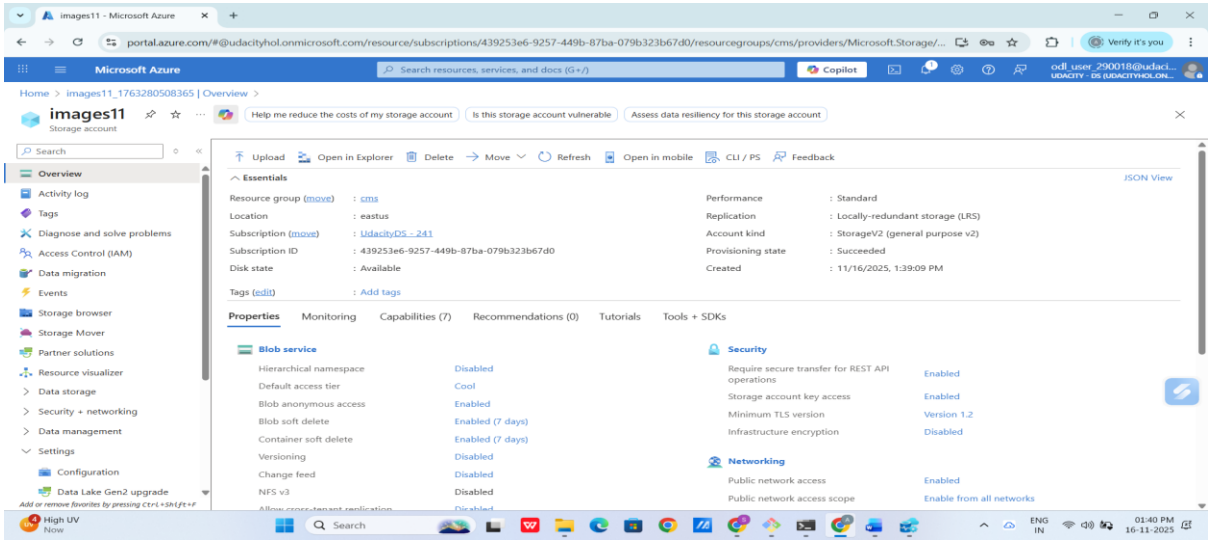

File Edit View Help
Search
Welcome
cms22.database.windows.net
SQLQuery_1 - (85) c...sadmin)
dbo.posts.sql - (52) c...sadmin)
dbo.users.sql - (70) c...sadmin)
Home > cms22.database.windows.net
New Query New Notebook Refresh Learn More
Home
Version : 12.0.2000.8
Type : Azure SQL DB
Search
Search databases
Name Status Size (MB) Actions
cms ONLINE
master ONLINE
Choose SQL Language cms22.database.windows.net : master
26°C Sunny
Search
ENG IN
01:32 PM 16-11-2025

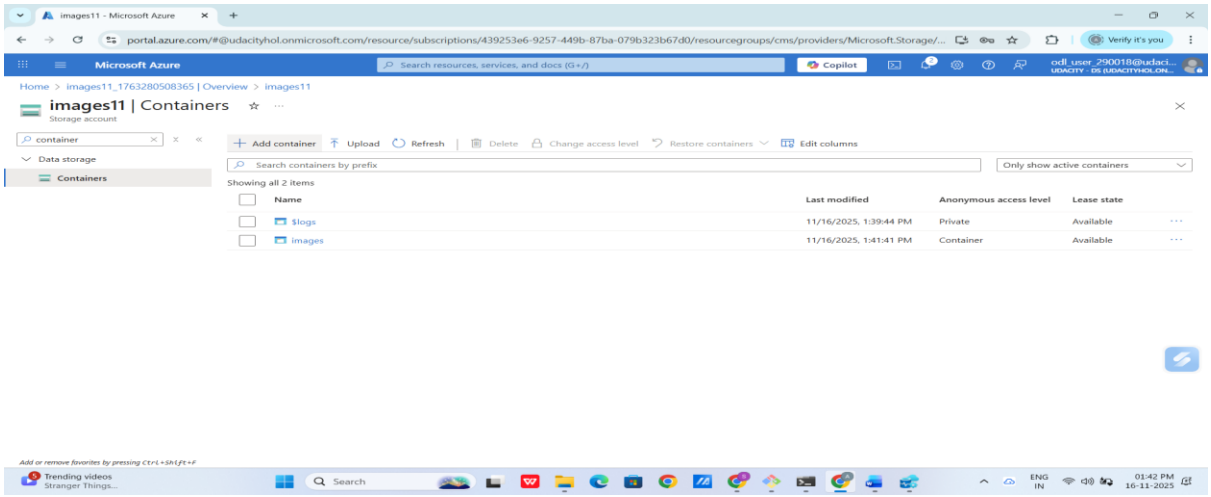


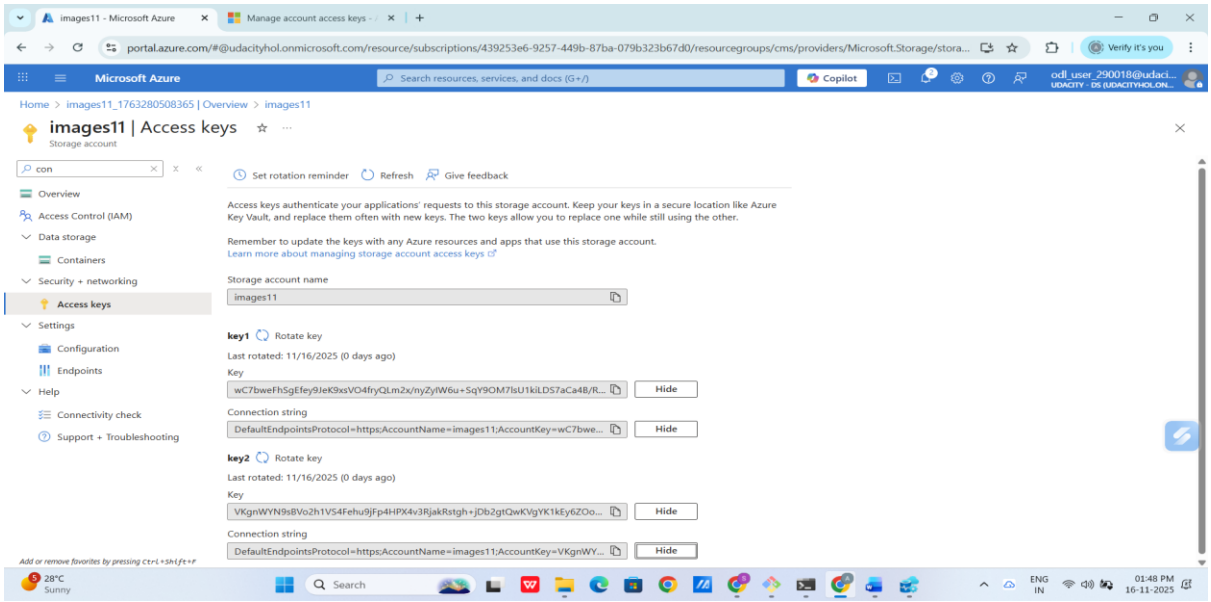

cms (cms22/cms) - Microsoft A
Manage account access keys -
portal.azure.com/#@udacityhol.onmicrosoft.com/resource/subscriptions/439253e6-9257-449b-87ba-079b323b67d0/resourceGroups/cms/providers/Microsoft.Sql/serv...
Verify it's you
Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
odl_user_290018@udaci...
UDACITY - DS (UDACITYHOLON...
Home > cms > cms (cms22/cms)
cms (cms22/cms) | Query editor (preview)
SQL database
Search
Login
New Query
Open query
Feedback
Getting started
Overview
Activity log
Tags
Diagnose and solve problems
Query editor (preview)
Mirror database in Fabric (preview)
Resource visualizer
Settings
Data management
Integrations
Power Platform
Security
Intelligent performance
Monitoring
Automation
Help
Add or remove favorites by pressing Ctrl+Shift+F
Showing limited object explorer here. For full capability please click here to open Azure Data Studio.
Tables
dbo.POSTS
id (PK, int, not null)
title (varchar, not null)
author (varchar, not null)
body (varchar, not null)
image_path (varchar, null)
timestamp (datetime, not null)
user_id (int, not null)
dbo.USERS
id (PK, int, not null)
username (varchar, not null)
password_hash (varchar, not null)
Views
Stored Procedures
Query 1
Query 2
Query 3
Run
Cancel query
Save query
Export data as
Show only Editor
1 SELECT TOP (1000) * FROM [dbo].[USERS]
2
Results
Messages
Search to filter items...
id
username
password_hash
1
admin
pbkdf2:sha256:150000$Qllrz6Hg$5f4cd25d78a6...
Query succeeded | 0s
27°C
Sunny
Search
ENG
IN
04:47 PM
16-11-2025

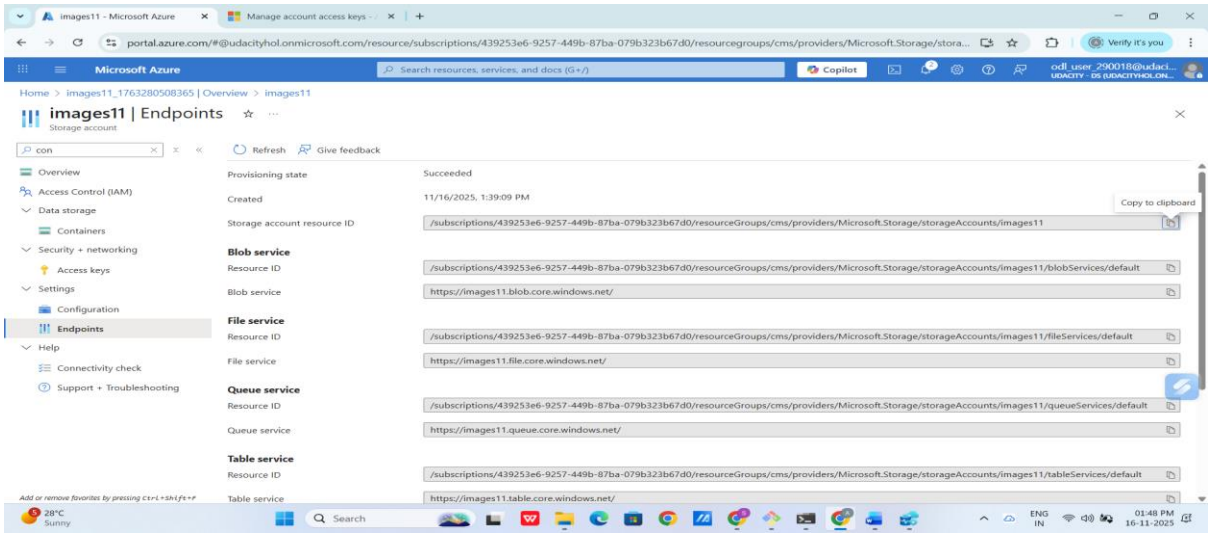

cms (cms22/cms) - Microsoft A
Manage account access keys -
portal.azure.com/#@udacityhol.onmicrosoft.com/resource/subscriptions/439253e6-9257-449b-87ba-079b323b67d0/resourceGroups/cms/providers/Microsoft.Sql/serv...
Verify it's you
Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
odl_user_290018@udaci...
UDACITY - DS (UDACITYHOLON...
Home > cms > cms (cms22/cms)
cms (cms22/cms) | Query editor (preview)
SQL database
Search
Login
New Query
Open query
Feedback
Getting started
Overview
Activity log
Tags
Diagnose and solve problems
Query editor (preview)
Mirror database in Fabric (preview)
Resource visualizer
Settings
Data management
Integrations
Power Platform
Security
Intelligent performance
Monitoring
Automation
Help
Add or remove favorites by pressing Ctrl+Shift+F
Showing limited object explorer here. For full capability please click here to open Azure Data Studio.
Tables
dbo.POSTS
id (PK, int, not null)
title (varchar, not null)
author (varchar, not null)
body (varchar, not null)
image_path (varchar, null)
timestamp (datetime, not null)
user_id (int, not null)
dbo.USERS
id (PK, int, not null)
username (varchar, not null)
password_hash (varchar, not null)
Views
Stored Procedures
Query 1
Query 2
Query 3
Run
Cancel query
Save query
Export data as
Show only Editor
1 SELECT TOP (1000) * FROM [dbo].[POSTS]
2
Results
Messages
Search to filter items...
id
title
author
body
image_path
timestamp
user_i
1
Lorem ipsum dolor s...
John Smith
Proin sit amet mi orn...
2025-11-16T08:01:3...
1
2
Hello World!
Jane Doe
My name is Jane Do...
RSRJFBY2UVPC1M06...
2025-11-16T10:56:1...
1
3
Cat
Rita
Hi! I am rita. I love ca...
6Y9U1K4UM03NK2C...
2025-11-16T10:57:2...
1
Query succeeded | 0s
27°C
Sunny
Search
ENG
IN
04:47 PM
16-11-2025

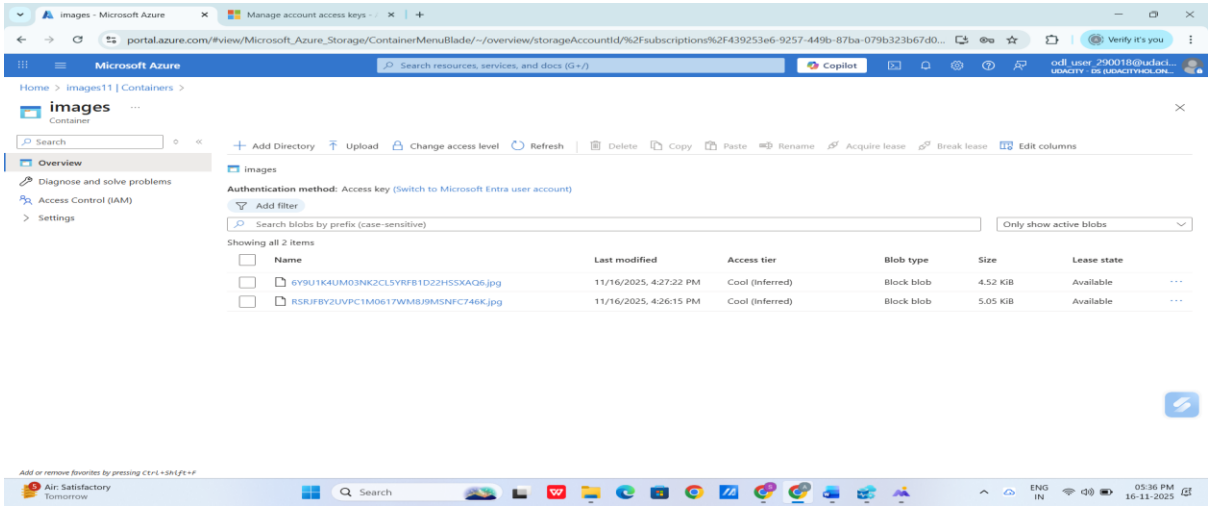
4. **A screenshot showing an example of blob endpoints where images are sent for storage (see example in the project repository).**



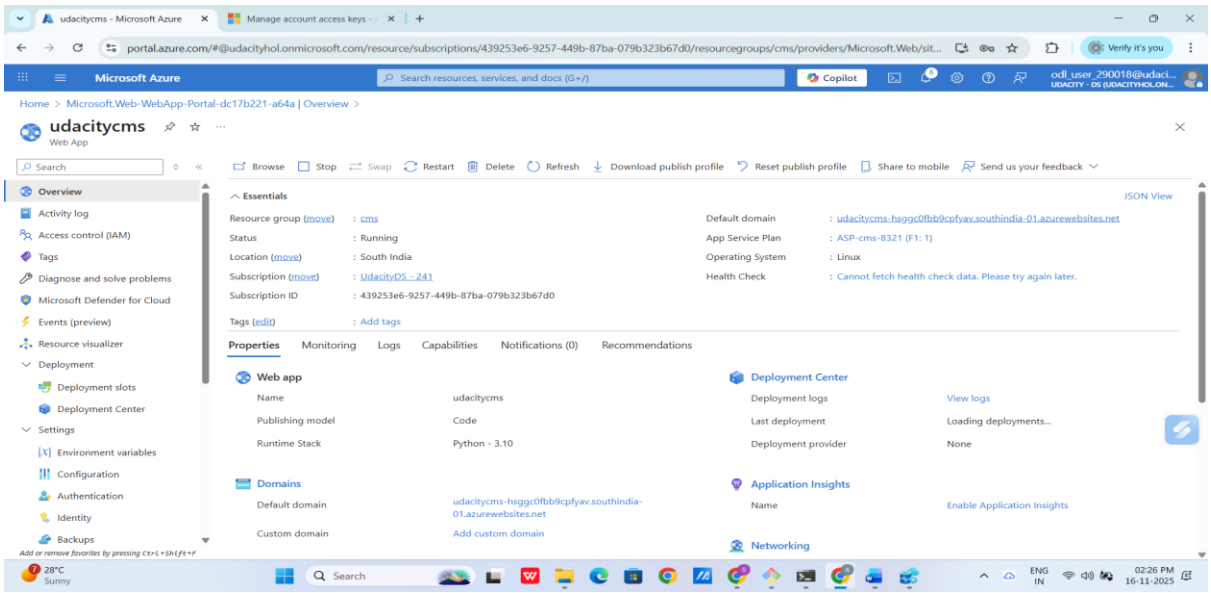


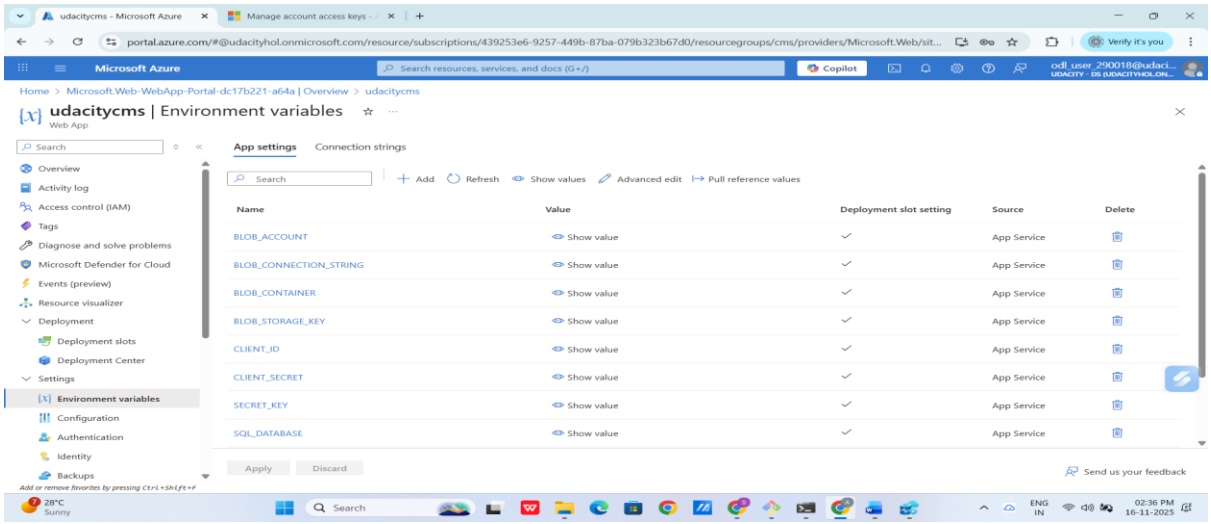


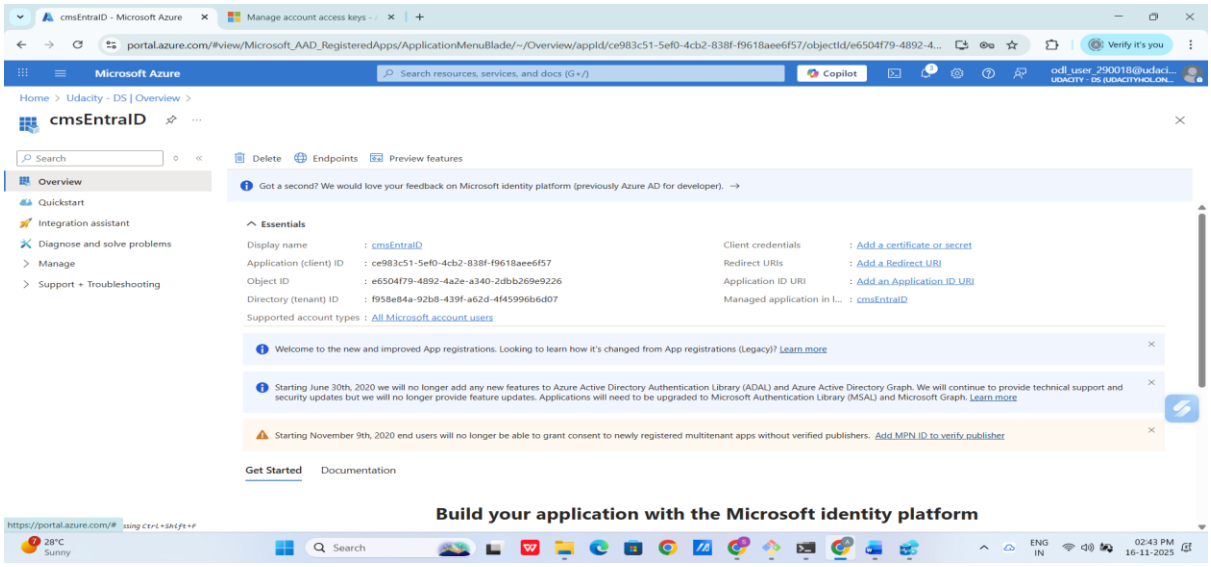


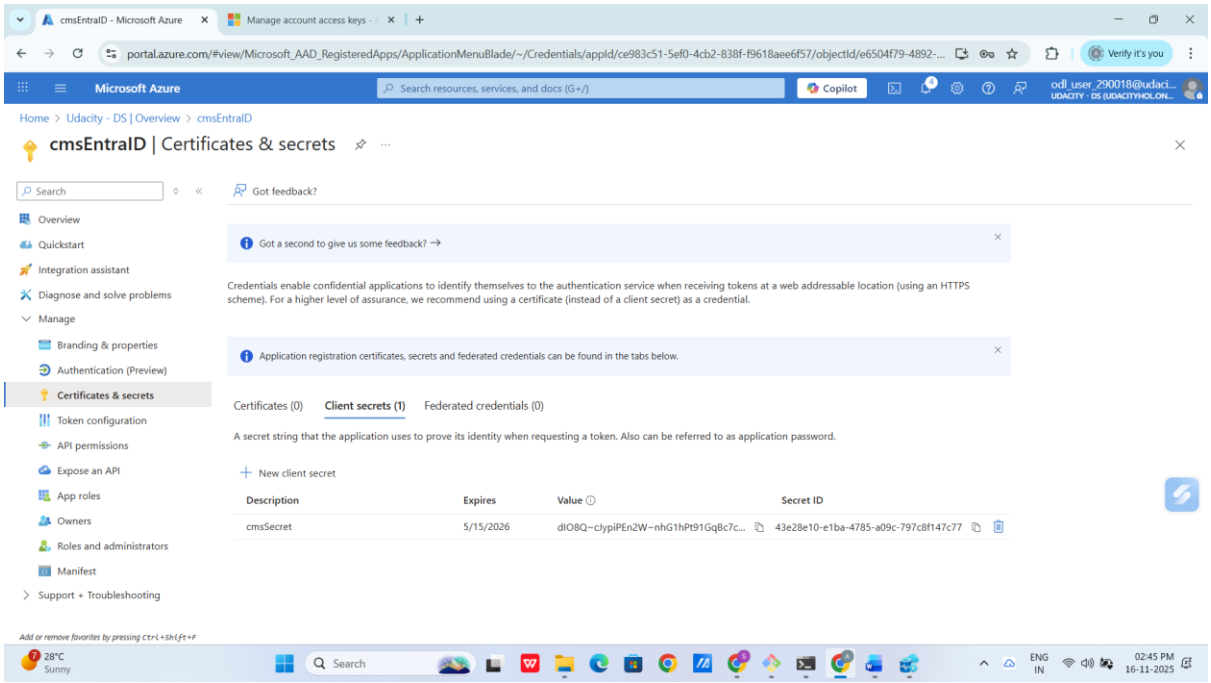


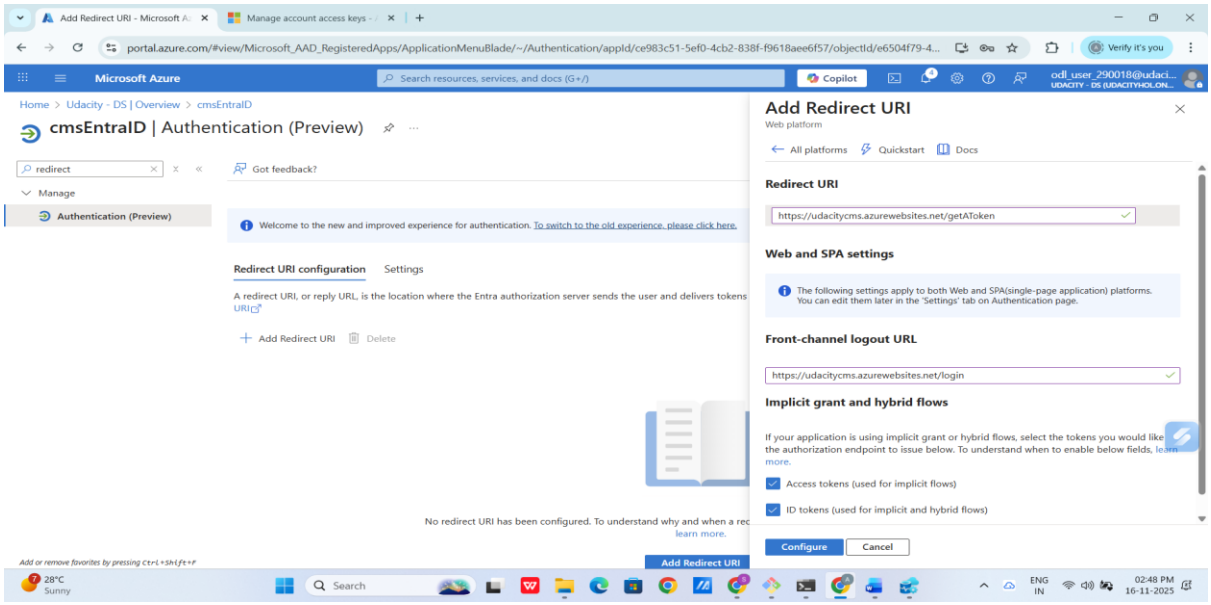
5. **A screenshot of the redirect URIs related to Microsoft authentication (see example in project repository).**

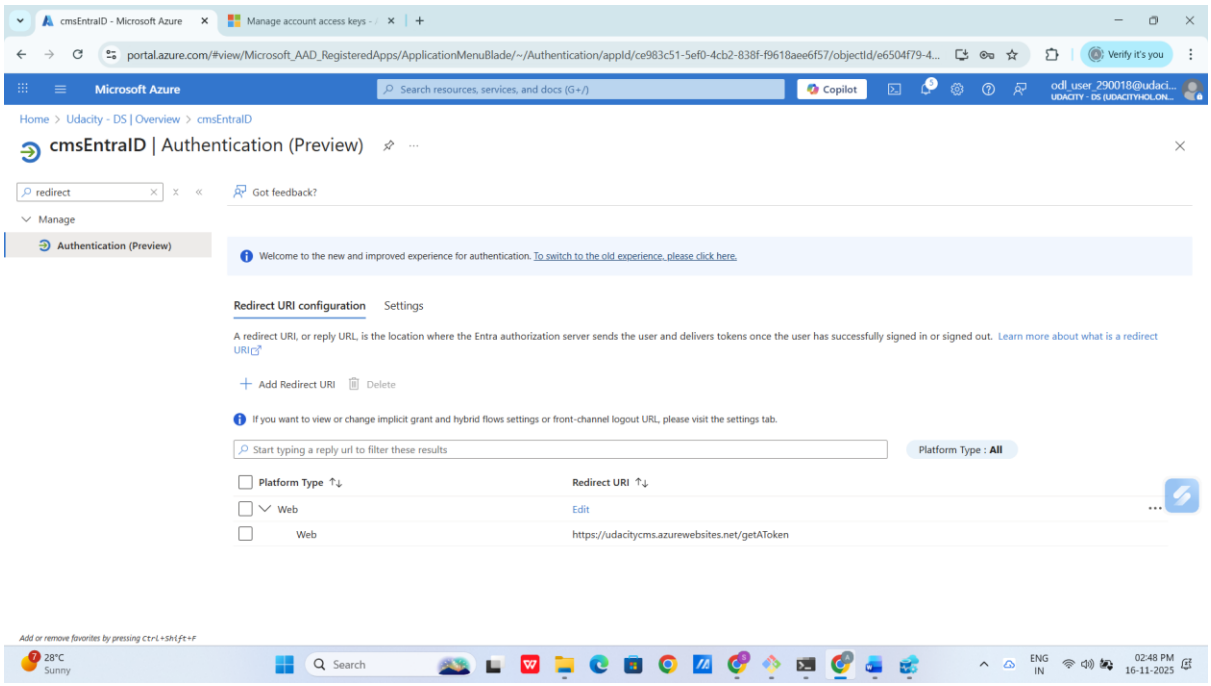



udacitycms - Microsoft Azure
Manage account access keys -
portal.azure.com/#@udacityhol.onmicrosoft.com/resource/subscriptions/439253e6-9257-449b-87ba-079b323b67d0/resourcegroups/cms/providers/Microsoft.Web/sit...
Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
Home > Microsoft.Web-WebApp-Portal-dc17b221-a64a | Overview > udacitycms
udacitycms | Environment variables
Web App
App settings
Connection strings
Add
Refresh
Show values
Advanced edit
Pull reference values
Name
Value
Deployment slot setting
Source
Delete
BLOB_ACCOUNT
BLOB_CONNECTION_STRING
BLOB_CONTAINER
BLOB_STORAGE_KEY
CLIENT_ID
CLIENT_SECRET
SECRET_KEY
SQL_DATABASE
Show value
App Service
Apply
Discard
Send us your feedback


cmsEntraID - Microsoft Azure
Manage account access keys -
portal.azure.com/#view/Microsoft_AAD_RegisteredApps/ApplicationMenuBlade/~/Overview/appId/ce983c51-5ef0-4cb2-838f-f9618aee6f57/objectId/e6504f79-4892-4...
Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
Home > Udacity - DS | Overview >
cmsEntraID
Delete
Endpoints
Preview features
Got a second? We would love your feedback on Microsoft identity platform (previously Azure AD for developer).
Essentials
Display name : cmsEntraID
Application (client) ID : ce983c51-5ef0-4cb2-838f-f9618aee6f57
Object ID : e6504f79-4892-4a2e-a340-2dbb269e9226
Directory (tenant) ID : f958e84a-92b8-439f-a62d-4f45996b6d07
Supported account types : All Microsoft account users
Client credentials : Add a certificate or secret
Redirect URIs : Add a Redirect URI
Application ID URI : Add an Application ID URI
Managed application in l... : cmsEntraID
Welcome to the new and improved App registrations. Looking to learn how it's changed from App registrations (Legacy)? Learn more
Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. Learn more
Starting November 9th, 2020 end users will no longer be able to grant consent to newly registered multitenant apps without verified publishers. Add MPN ID to verify publisher
Get Started
Documentation
Build your application with the Microsoft identity platform


cmsEntraID - Microsoft Azure
Manage account access keys -
portal.azure.com/#view/Microsoft_AAD_RegisteredApps/ApplicationMenuBlade/~/Credentials/appId/ce983c51-5ef0-4cb2-838f-f9618aee6f57/objectId/e6504f79-4892-...
Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
Home > Udacity - DS | Overview > cmsEntraID
cmsEntraID | Certificates & secrets
Got feedback?
Got a second to give us some feedback?
Credentials enable confidential applications to identify themselves to the authentication service when receiving tokens at a web addressable location (using an HTTPS scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a credential.
Application registration certificates, secrets and federated credentials can be found in the tabs below.
Certificates (0)
Client secrets (1)
Federated credentials (0)
A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.
New client secret
Description
Expires
Value
Secret ID
cmsSecret
5/15/2026
dIO8Q~clypiPEn2W~nhG1hPt91GqBc7c...
43e28e10-e1ba-4785-a09c-797c8f147c77





6. **A screenshot showing one potential form of logging with an "Invalid login attempt" and "admin logged in successfully", taken from the app's Log stream or other logs you create and store (see example in project repository). You can customize your log messages as you see fit for these situations.**

```
2020-06-04T20:01:39.236054360Z 172.16.0.1 - - [04/Jun/2020:20:01:39 +0000] "GET / HTTP/1.1" 302 237 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.61 Safari/537.36"
2020-06-04T20:01:39.517377405Z 172.16.0.1 - - [04/Jun/2020:20:01:39 +0000] "GET /login?next=%2F HTTP/1.1" 200 3015 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.61 Safari/537.36"

2020-06-04T20:01:50.539041390Z Invalid login attempt.
2020-06-04T20:01:50.603246546Z 172.16.0.1 - - [04/Jun/2020:20:01:50 +0000] "POST /login?next=%2F HTTP/1.1" 302 219 "https://cms-app-test.azurewebsites.net/login?next=%2F" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.61 Safari/537.36"
2020-06-04T20:01:50.900136504Z 172.16.0.1 - - [04/Jun/2020:20:01:50 +0000] "GET /login HTTP/1.1" 200 3009 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.61 Safari/537.36"

2020-06-04T20:02:07.294319770Z admin logged in successfully
```

7. **Your application code—most importantly __init__.py and views.py since they will contain your updates for Auth and Logging.**

7.1 __init__.py:

```
"""
The flask application package.
"""
import logging
from flask import Flask
from config import Config
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager
from flask_session import Session
from flask.logging import create_logger

app = Flask(__name__)
app.config.from_object(Config)
# TODO: Add any logging levels and handlers with app.logger
# DONE (Note: Instructor's original app.logger code not pylint compliant)
LOG = create_logger(app)
LOG.setLevel(logging.INFO)
streamHandler = logging.StreamHandler()
streamHandler.setLevel(logging.INFO)
LOG.addHandler(streamHandler)

Session(app)
db = SQLAlchemy(app)
login = LoginManager(app)
login.login_view = 'login'

import FlaskWebProject.views
```

7.2 views.py:

```
"""
Routes and views for the flask application.
"""

from datetime import datetime
from flask import render_template, flash, redirect, request, session, url_for
from werkzeug.urls import url_parse
from config import Config
from FlaskWebProject import app, db
from FlaskWebProject.forms import LoginForm, PostForm
from flask_login import current_user, login_user, logout_user, login_required
from FlaskWebProject.models import User, Post
import msal
import uuid
```

```
imageSourceUrl = 'https://'+ app.config['BLOB_ACCOUNT']  + '.blob.core.windows.net/' +
app.config['BLOB_CONTAINER']  + '/'

@app.route('/')
@app.route('/home')
@login_required
def home():
    user = User.query.filter_by(username=current_user.username).first_or_404()
    posts = Post.query.all()
    return render_template(
        'index.html',
        title='Home Page',
        posts=posts
    )

@app.route('/new_post', methods=['GET', 'POST'])
@login_required
def new_post():
    form = PostForm(request.form)
    if form.validate_on_submit():
        post = Post()
        post.save_changes(form, request.files['image_path'], current_user.id, new=True)
        return redirect(url_for('home'))
    return render_template(
        'post.html',
        title='Create Post',
        imageSource=imageSourceUrl,
        form=form
    )

@app.route('/post/<int:id>', methods=['GET', 'POST'])
@login_required
def post(id):
    post = Post.query.get(int(id))
    form = PostForm(formdata=request.form, obj=post)
    if form.validate_on_submit():
        post.save_changes(form, request.files['image_path'], current_user.id)
        return redirect(url_for('home'))
    return render_template(
        'post.html',
        title='Edit Post',
        imageSource=imageSourceUrl,
        form=form
    )

@app.route('/login', methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = LoginForm()
    if form.validate_on_submit():
```

```
        user = User.query.filter_by(username=form.username.data).first()
        if user is None or not user.check_password(form.password.data):
            flash('Invalid username or password')
            return redirect(url_for('login'))
        login_user(user, remember=form.remember_me.data)
        next_page = request.args.get('next')
        if not next_page or url_parse(next_page).netloc != '':
            next_page = url_for('home')
        return redirect(next_page)
    session["state"] = str(uuid.uuid4())
    auth_url = _build_auth_url(scopes=Config.SCOPE, state=session["state"])
    return render_template('login.html', title='Sign In', form=form, auth_url=auth_url)

@app.route('/getAToken')  # Its absolute URL must match your app's redirect_uri set in AAD
def authorized():
    if request.args.get('state') != session.get("state"):
        # State mismatch -> just go home
        return redirect(url_for("home"))

    if "error" in request.args:
        # Authentication/Authorization failure
        LOG.error('ERROR: Authentication/Authorization failure...')
        return render_template("auth_error.html", result=request.args)

    if request.args.get('code'):
        cache = _load_cache()
        # Acquire a token using the auth code from AAD
        result = _build_msal_app(cache=cache).acquire_token_by_authorization_code(
            request.args['code'],
            scopes=Config.SCOPE,
            # For local dev, you can use http. For Azure, you'll use https.
            redirect_uri=url_for('authorized', _external=True, _scheme='http')
        )
        if "error" in result:
            LOG.error('ERROR: Did not acquire a token for OAUTH...')
            return render_template("auth_error.html", result=result)

        # Save user claims from the id_token
        session["user"] = result.get("id_token_claims")

        # In this project, any MS-authenticated user is treated as "admin"
        user = User.query.filter_by(username="admin").first()
        login_user(user)
        _save_cache(cache)
        LOG.info('INFO: User Logged In via Microsoft...')

    return redirect(url_for('home'))

@app.route('/logout')
def logout():
    logout_user()
```

```
    if session.get("user"): # Used MS Login
        # Wipe out user and its token cache from session
        session.clear()
        # Also logout from your tenant's web session
        return redirect(
            Config.AUTHORITY + "/oauth2/v2.0/logout" +
            "?post_logout_redirect_uri=" + url_for("login", _external=True))

    return redirect(url_for('login'))

def _load_cache():
    # TODO: Load the cache from `msal`, if it exists
    cache = None
    return cache

def _save_cache(cache):
    # TODO: Save the cache, if it has changed
    pass

def _build_msal_app(cache=None, authority=None):
    # TODO: Return a ConfidentialClientApplication
    return None

def _build_auth_url(authority=None, scopes=None, state=None):
    # TODO: Return the full Auth Request URL with appropriate Redirect URI
    return None
```

8. **Your WRITEUP.md file analyzing and explaining your choice between a VM or an App Service.**

 # Write-up Template

Analyze, choose, and justify the appropriate resource option for deploying the app.

For **both a VM or App Service solution for the CMS app:**

- **Analyze costs, scalability, availability, and workflow**

- **Choose the appropriate solution (VM or App Service) for deploying the app**

- **Justify your choice**

a) Virtual Machine (VM)

Cost:

A VM is billed like a full server: you pay for the selected size (CPU, RAM, storage) as long as it's allocated, regardless of how busy the app is. For this CMS, a small VM like Standard B1ls is enough, but the cost is still higher than a free or low-tier App Service plan, especially if I forget to stop or delete the VM. I would also be responsible for managing OS patches and any extra software, which indirectly adds "operational cost" in terms of my time.

Scalability:

A single VM scales vertically: to handle more load, I'd have to resize it to a bigger SKU or manually add more VMs behind a load balancer. This is more complex to manage and usually requires planning and downtime. Auto-scaling is possible, but I have to configure it explicitly using scale sets or other Azure services.

Availability:

VM availability depends on how I configure it. A single VM has a single point of failure. To achieve higher availability, I'd need multiple VMs in an availability set or zone, plus a load balancer. I am responsible for OS updates, security patches, and restarts, which can cause downtime if not handled carefully.

Workflow:

Using a VM gives me full control over the OS and runtime stack. I can install any version of Python, configure Nginx/Gunicorn manually, and host other services on the same machine. However, deployment is more manual: SSH into the VM, clone the GitHub repo, create a virtual environment, install dependencies, configure Nginx, and manage services. This provides flexibility but also increases complexity and maintenance effort.

b) App Service (Web App)

Cost:

App Service offers a Free F1 tier, which is ideal for this project and development scenarios. I don't pay for idle VMs directly; instead, I pay per App Service plan, which can host multiple web apps. For small workloads like this article CMS, the free or basic tiers are cheaper than running a dedicated VM 24/7. Scaling up to higher tiers is still generally cost-effective compared to managing multiple VMs myself.

Scalability:

App Service has built-in horizontal scaling. I can increase the number of instances with a slider or configure auto-scale rules based on CPU, memory, or schedule. It abstracts away the individual servers, so I don't have to manage the underlying OS. This makes it easier to handle traffic spikes and future growth of the CMS without re-architecting the deployment.

Availability:

App Service is a managed platform (PaaS), so Microsoft handles OS patching, infrastructure redundancy, and many aspects of uptime. The service is integrated with Azure's availability features, and I can choose higher tiers with built-in SLA guarantees. I don't need to manually set up multiple machines or a load balancer just to get a highly available endpoint.

Workflow:

The developer workflow with App Service is very streamlined. I can connect the App Service to GitHub via Deployment Center so that pushes to a branch automatically deploy to Azure. Configuration like connection strings and secrets can be stored in Application Settings rather than hard-coded in the code. I don't manage the OS, web server, or runtime directly; I only focus on the app code and settings. This reduces operational overhead and speeds up development.

VM or App Service? (and why):

For this project, I chose to deploy the Flask CMS using Azure App Service (Web App).

The main reasons are:

Cost – The App Service free tier is enough for this small CMS, so I don't pay for a dedicated VM. It's more economical for a learning project and small workloads.

Simplicity & Workflow – App Service gives me an easy deployment pipeline via GitHub and a simple configuration experience through Application Settings. I don't have to manage the OS, patching, or web server directly, which lets me focus on the Flask app itself.

Built-in scalability & availability – If the CMS ever needs to handle more users or traffic, I can scale up or out using the App Service plan without redesigning the infrastructure or manually configuring load balancers and extra VMs.

Because of these factors, App Service is a better fit for this application than a VM for my current needs.

Assess app changes that would change your decision.

If the CMS application changed in ways that demanded more low-level infrastructure control, my choice might switch toward a VM (or even a container-based solution). For example, if I needed custom OS-level software, unusual networking requirements, or background services that App Service doesn't support easily, a VM would give me full control to install and configure everything exactly as needed.

Similarly, if the application expanded into a multi-component system (e.g., additional microservices, custom message queues, or specialized database engines) that needed tight control over the environment and network topology, managing everything on VMs (or using containers orchestrated by services like AKS) might become more suitable. In that case, I would accept the extra operational complexity of VMs in exchange for the flexibility they provide.

Right now, though, the CMS is a straightforward web app that primarily needs a Python runtime, connection to Azure SQL and Blob Storage, and an OAuth2 login flow. For that scenario, App Service remains the most practical and maintainable option.