

Meet Specifications

Site: <https://tannv-article-cms-project.azurewebsites.net>

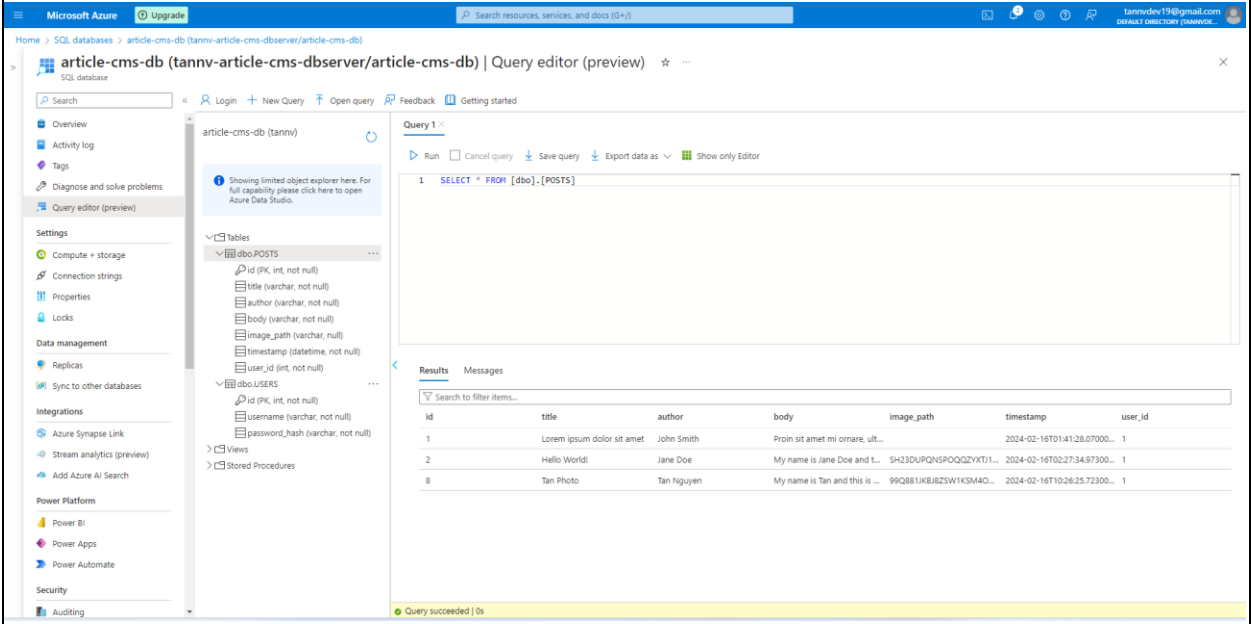
Resource Group

CRITERIA	MEETS SPECIFICATIONS
All relevant resources are contained in a single resource group.	The resource group must include a Storage Account, SQL Server, SQL Database, as well as any relevant services for deploying the web app. Provide a screenshot of the resource group in Azure, containing your running resources.

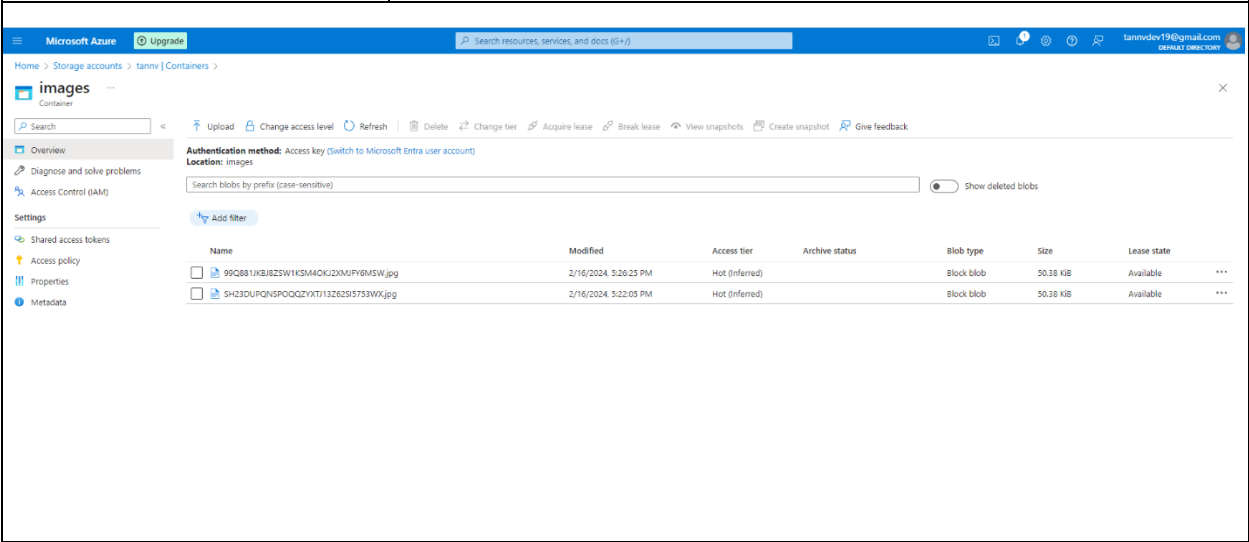
The screenshot shows the Azure portal interface for the resource group 'tannv-article-cms-resource'. The left sidebar contains navigation options like Overview, Activity log, Access control (IAM), Tags, Resource visualizer, Events, Settings, Deployments, Security, Deployment stacks, Policies, Properties, Locks, Cost Management, Monitoring, Alerts, Metrics, and Diagnostic settings. The main content area shows the 'Essentials' section with subscription details (Subscription ID: daef1c8f-6299-4b9d-ac44-cd38e65997e8) and deployment status (2 Deploying, 2 Failed, 12 Succeeded). Below this, the 'Resources' section lists 6 resources with columns for Name, Type, and Location. The resources are: article-cms-db (SQL database, East Asia), ASP-tannvarticlecmsresource-b5df (App Service plan, Southeast Asia), tannv (Storage account, Australia East), tannv-article-cms-id-91fd (Managed identity, Southeast Asia), tannv-article-cms-dbserver (SQL server, East Asia), and tannv-article-cms-project (App Service, Southeast Asia). The page also includes a search bar, filters, and a 'Give feedback' link.

Name	Type	Location
article-cms-db (tannv-article-cms-dbserver/article-cms-db)	SQL database	East Asia
ASP-tannvarticlecmsresource-b5df	App Service plan	Southeast Asia
tannv	Storage account	Australia East
tannv-article-cms-id-91fd	Managed identity	Southeast Asia
tannv-article-cms-dbserver	SQL server	East Asia
tannv-article-cms-project	App Service	Southeast Asia

Storage

CRITERIA	MEETS SPECIFICATIONS
Create and add article data to a SQL Server in Azure.	<p>A SQL Server is created in Azure and is capable of storing the necessary article data (title, author, body).</p> <p>Provide a screenshot from your SQL database within Azure, showing that both the POST and USER tables have been created. Alternatively, if the site is still live, provide the URL for the site.</p>
	

CRITERIA	MEETS SPECIFICATIONS
Create and upload images to a Storage Account.	<p>A Storage Account is created in Azure and is capable of storing the necessary image data for the article.</p> <p>Provide a screenshot from your Storage Account within Azure, with the blob storage endpoint URL visible (can be seen in "Settings"->"Properties"). Alternatively, if the site is still live, provide the URL for the CMS site to show images are able to be stored and viewed.</p>



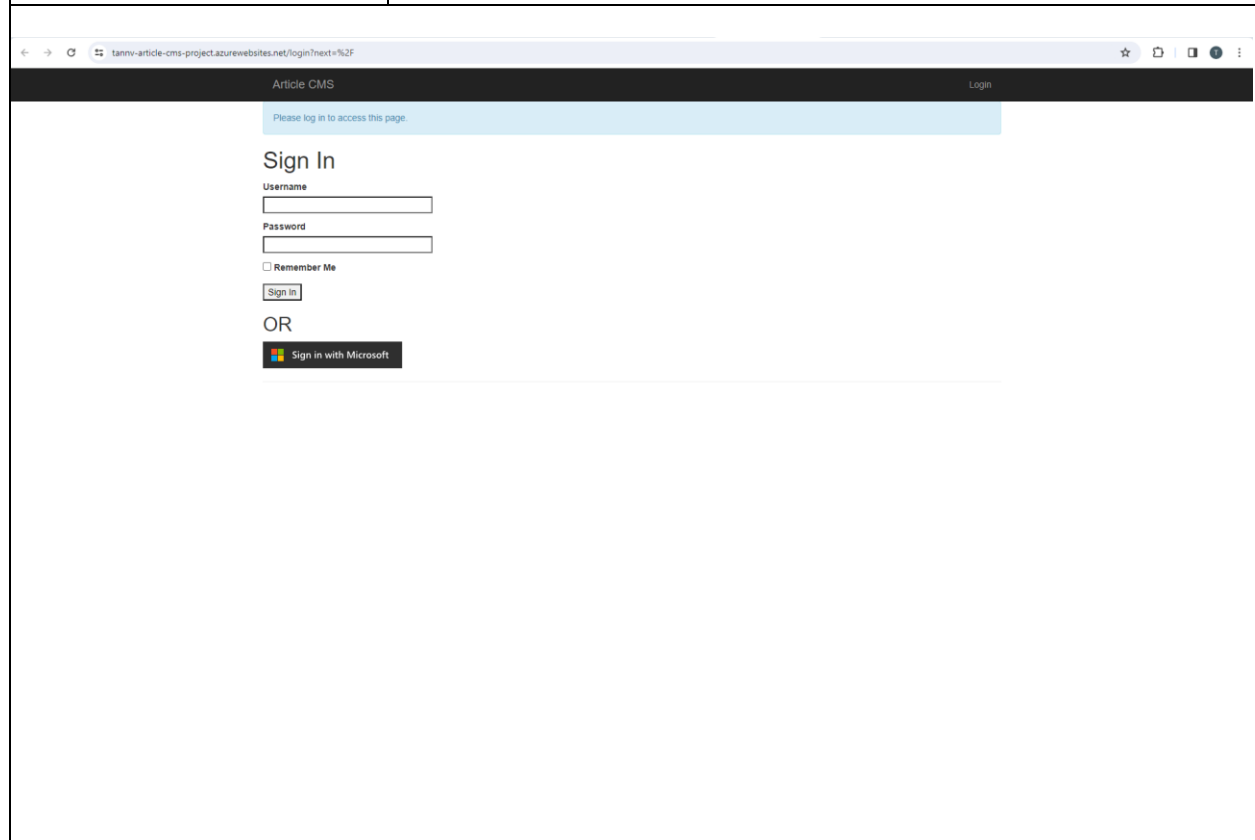
Resource Justification

CRITERIA	MEETS SPECIFICATIONS
Analyze, choose, and justify the appropriate resource option for deploying the app.	<p>In the provided writeup.md file, for both a VM or App Service solution for the CMS app:</p> <ul style="list-style-type: none">• Analyze costs, scalability, availability, and workflow• Choose the appropriate solution (VM or App Service) for deploying the app• Justify your choice <p>This does not need to be substantially long, but should include information on all four analysis points for each option, your choice, and at least 2-3 sentences on why you choose that option.</p>
<p>I choosing a web app service over a virtual machine. Cost-wise, the web app wins hands down with its pay-as-you-go model, saving cash. Plus, you ditch the VM maintenance headache, freeing up your time and energy. Deployment becomes a breeze thanks to the seamless GitHub integration, getting your app live in a flash. High availability is built-in, so your users enjoy consistent uptime. And when it comes to scaling, the web app adapts to your needs effortlessly, avoiding the manual VM configuration struggles. Furthermore, the web app service excels in simplicity and speed. The seamless integration with your GitHub repository allows for effortless deployment with just a few clicks. This streamlines your workflow significantly compared to the cumbersome setup and configuration of a VM. Based on the costs, scalability, availability, and workflow analysis I selected the web app service to deploy for my project article.</p>	

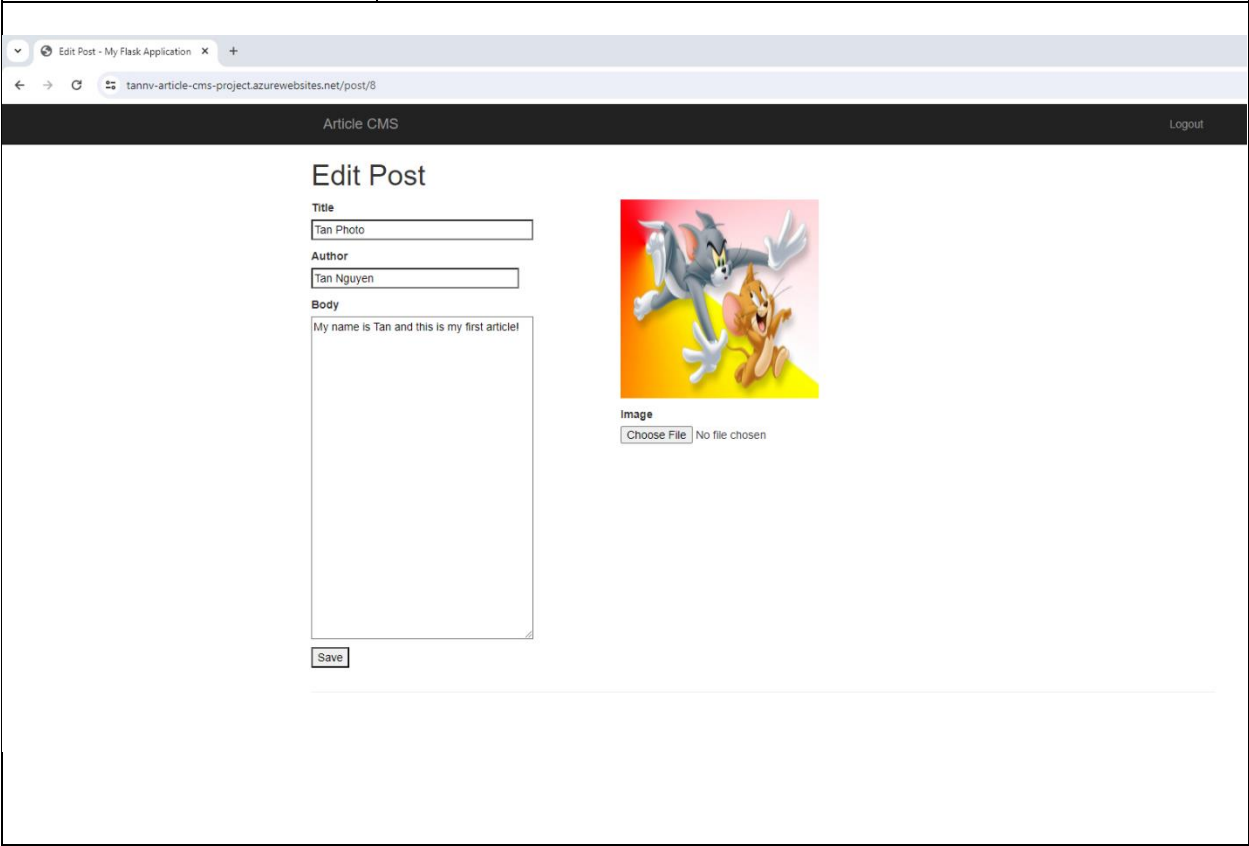
CRITERIA	MEETS SPECIFICATIONS
<p>Assess app changes that would change your decision.</p>	<p>In the provided writeup.md file, detail how the app and any other needs would have to change for you to change your decision in the last section.</p> <p>This should be at least 2-3 sentences, but feel free to add as much detail as you feel necessary.</p>
<p>I would choose virtual machines when security needs skyrocket and your application architecture explodes with microservices, diverse technologies and user growth, virtual machines (VMs) become attractive allies . Their inherent flexibility protects applications in isolated bubbles, enables granular resource allocation for security, and seamless integration of disparate technologies. Microservices thrive in individual VM houses, easily scaling to meet spikes in user demand. However, keep in mind the cost, management complexity when use virtual machine.</p>	

Deployment

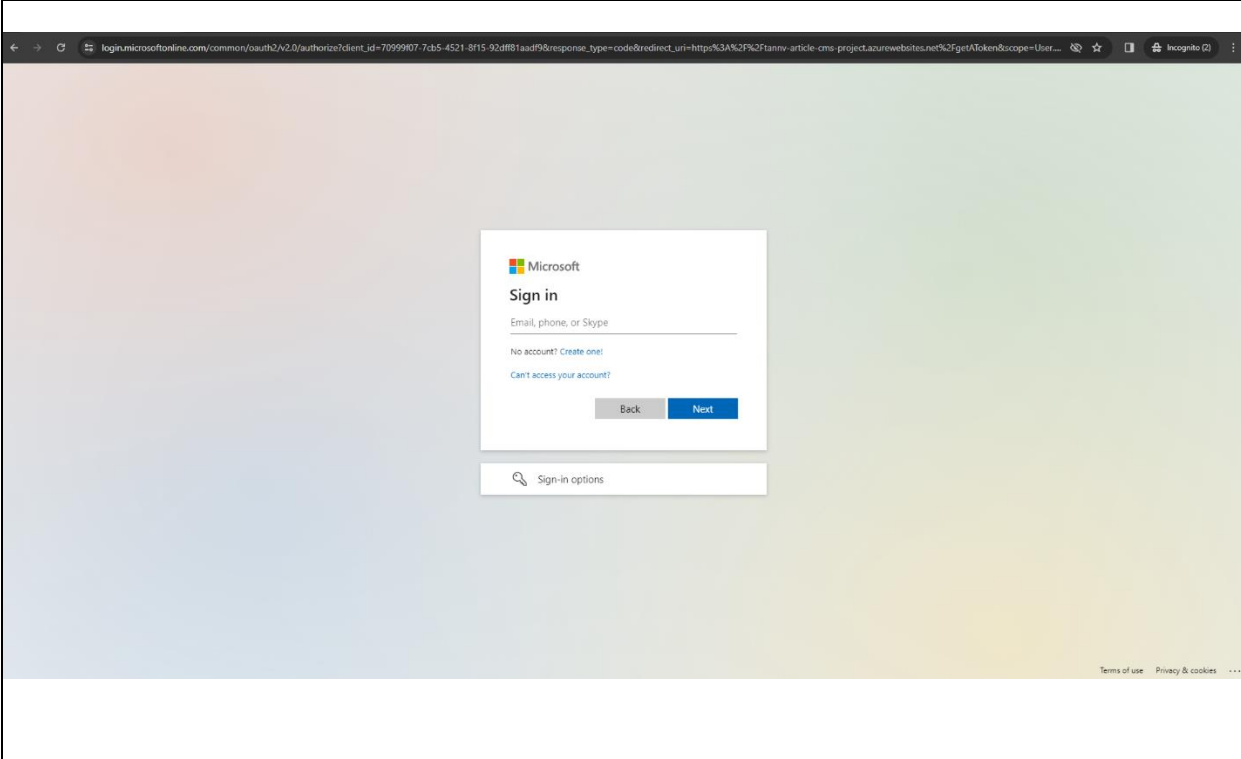
CRITERIA	MEETS SPECIFICATIONS
The Python web app is deployed to Azure.	<p>The Python web app has been deployed to Azure using the chosen resource in the previous section.</p> <p>As evidence, provide a screenshot of the Python application running from a browser (this can be part of the screenshot in the next section). The screenshot should include the URL and the black header that states “Article CMS”. Alternatively, you can provide a link to the deployed app, if it is still live.</p>

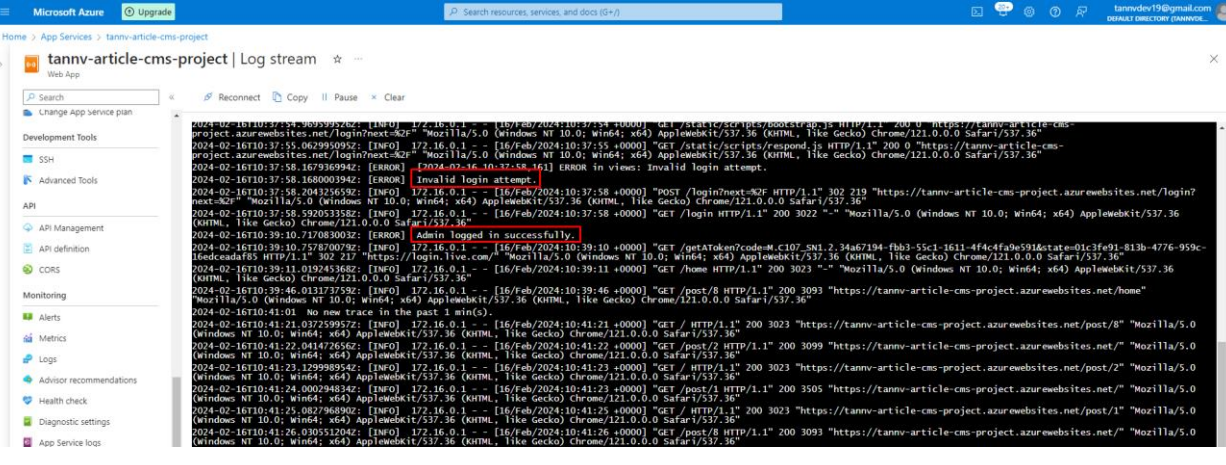


CRITERIA	MEETS SPECIFICATIONS
The Python web app is able to connect to storage.	<p>The Python web app is able to connect to the related storage solutions.</p> <p>As evidence, provide a screenshot of the Python application running from a browser. The screenshot should include the URL and at least one article containing title, author, body, and an image. Alternatively, you can provide a link to the deployed app, if it is still live.</p>



Security & Monitoring

CRITERIA	MEETS SPECIFICATIONS
Add a functioning “Sign in with Microsoft” option to the app.	<p>The Python web app has an additional, operational option to sign in with Microsoft.</p> <p>As evidence, provide a screenshot of the redirect URIs configured within the App Registration page in Azure. Alternatively, you can provide a link to the deployed app, if it is still live.</p> <p>Additionally, your code in <code>views.py</code> should appropriately implement the Microsoft sign-in button using the <code>msal</code> library.</p>
 A screenshot of a web browser showing the Microsoft Sign in page. The browser's address bar displays a long URL starting with 'login.microsoftonline.com'. The page has a light blue and white background. In the center, there is a white box with the Microsoft logo and the text 'Sign in'. Below this, there is a text input field labeled 'Email, phone, or Skype'. Underneath the input field, there are two links: 'No account? Create one!' and 'Can't access your account?'. At the bottom of the white box are two buttons: 'Back' and 'Next'. Below the white box, there is a link that says 'Sign-in options' with a magnifying glass icon. At the bottom right of the page, there are links for 'Terms of use', 'Privacy & cookies', and a three-dot menu icon.	

CRITERIA	MEETS SPECIFICATIONS
<p>Access attempts to the app are logged.</p>	<p>Both successful and unsuccessful attempts to access the web app are logged.</p> <p>As evidence, provide a screenshot or download the logs from Azure containing at least one successful and one unsuccessful access attempt, and include in your submission files. If otherwise submitting a URL, please include a link to screenshot/logs in the “Submission Details” box on the project submission page.</p>
	 <p>The screenshot shows the Azure portal interface for the 'tannv-article-cms-project' web app. The 'Log stream' tab is active, displaying a list of log entries. The logs include timestamps, IP addresses, user agents, and HTTP status codes. Key entries include:</p> <ul style="list-style-type: none">2024-02-16T10:37:55.062995995Z: [INFO] 172.16.0.1 - - [16/Feb/2024:10:37:55 +0000] "GET /static/scripts/respond.js HTTP/1.1" 200 0 "https://tannv-article-cms-project.azurewebsites.net/login?next=82f" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.362024-02-16T10:37:58.167936994Z: [ERROR] [2024-02-16 10:37:58.161] ERROR in views: Invalid login attempt.2024-02-16T10:37:58.168000394Z: [ERROR] Invalid login attempt.2024-02-16T10:37:58.592053358Z: [INFO] 172.16.0.1 - - [16/Feb/2024:10:37:58 +0000] "POST /login?next=82f HTTP/1.1" 302 219 "https://tannv-article-cms-project.azurewebsites.net/login?next=82f" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.362024-02-16T10:37:58.592053358Z: [INFO] 172.16.0.1 - - [16/Feb/2024:10:37:58 +0000] "GET /login HTTP/1.1" 200 3022 "-" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.362024-02-16T10:39:10.717083003Z: [ERROR] Admin logged in successfully.2024-02-16T10:39:10.757870079Z: [INFO] 172.16.0.1 - - [16/Feb/2024:10:39:10 +0000] "GET /getToken?code=M.C107.SN1.2.34a67194-fbb3-55c1-1611-4f4c4fa9e591&state=01c3fe91-813b-4776-959c-16edceadaf85 HTTP/1.1" 202 217 "https://login.live.com/" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.362024-02-16T10:39:11.019245368Z: [INFO] 172.16.0.1 - - [16/Feb/2024:10:39:11 +0000] "GET /home HTTP/1.1" 200 3023 "-" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.362024-02-16T10:39:46.013173759Z: [INFO] 172.16.0.1 - - [16/Feb/2024:10:39:46 +0000] "GET /post/8 HTTP/1.1" 200 3093 "https://tannv-article-cms-project.azurewebsites.net/home" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.362024-02-16T10:41:01.000000000Z: [INFO] No new trace in the past 1 min(s).2024-02-16T10:41:21.037259937Z: [INFO] 172.16.0.1 - - [16/Feb/2024:10:41:21 +0000] "GET / HTTP/1.1" 200 3023 "https://tannv-article-cms-project.azurewebsites.net/post/8" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.362024-02-16T10:41:22.041472656Z: [INFO] 172.16.0.1 - - [16/Feb/2024:10:41:22 +0000] "GET /post/2 HTTP/1.1" 200 3099 "https://tannv-article-cms-project.azurewebsites.net/" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.362024-02-16T10:41:23.041472656Z: [INFO] 172.16.0.1 - - [16/Feb/2024:10:41:23 +0000] "GET / HTTP/1.1" 200 3023 "https://tannv-article-cms-project.azurewebsites.net/post/2" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.362024-02-16T10:41:24.000294834Z: [INFO] 172.16.0.1 - - [16/Feb/2024:10:41:24 +0000] "GET /post/1 HTTP/1.1" 200 3505 "https://tannv-article-cms-project.azurewebsites.net/" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.362024-02-16T10:41:25.082796890Z: [INFO] 172.16.0.1 - - [16/Feb/2024:10:41:25 +0000] "GET / HTTP/1.1" 200 3023 "https://tannv-article-cms-project.azurewebsites.net/post/1" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.362024-02-16T10:41:26.030512042Z: [INFO] 172.16.0.1 - - [16/Feb/2024:10:41:26 +0000] "GET /post/6 HTTP/1.1" 200 3093 "https://tannv-article-cms-project.azurewebsites.net/" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36