ST10384670

Name: Zanenhlanhla. Tshenolo. Konjwayo

Module: CLOUD DEVELOPMENT A

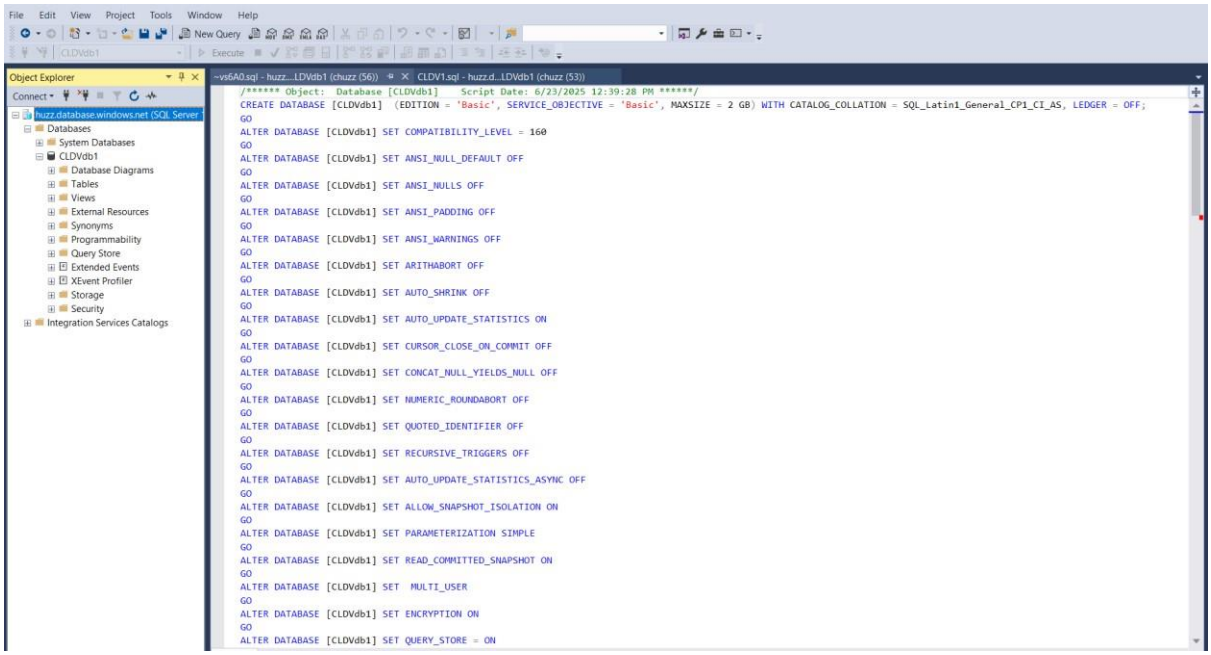
Component Discussion Technology Choices and Reasons

ASP.NET Core MVC

Because of its scalability, resilience, and close connection with Azure services, ASP.NET Core MVC was selected. It makes it possible for Models, Views, and Controllers to have their concerns clearly separated, which aids in logical system organization.

SQL Server

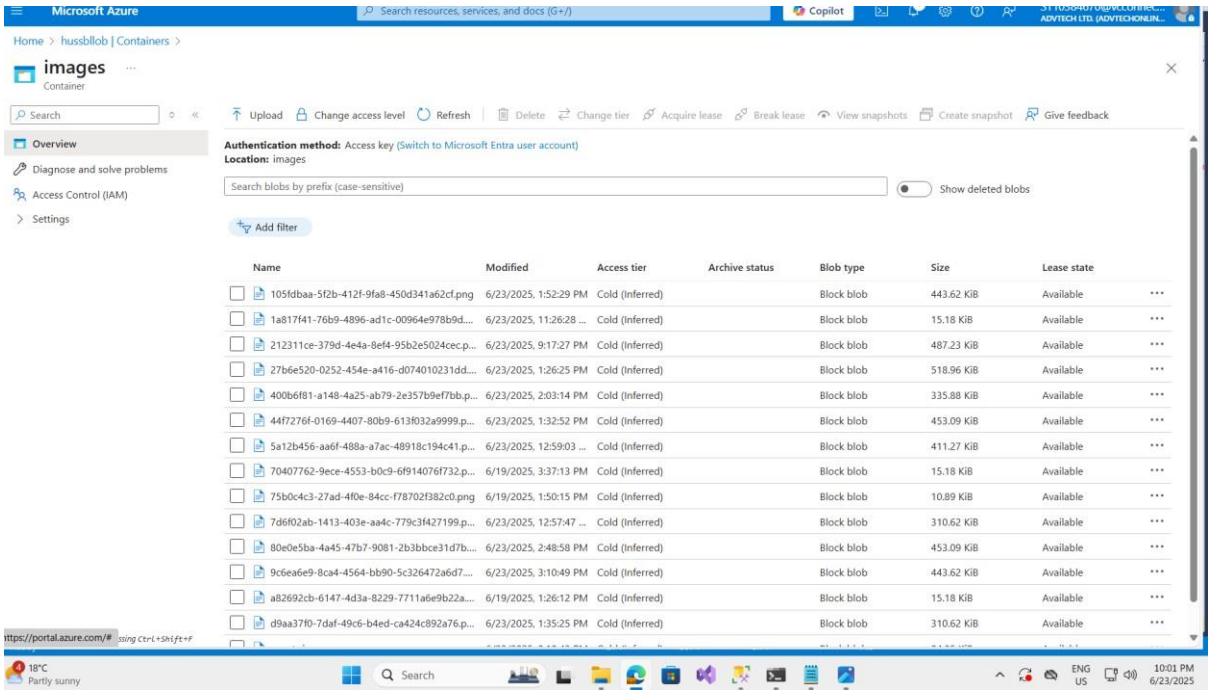
Because of its dependability, native cloud compatibility, and smooth Entity Framework integration, SQL Server was chosen. A scalable, cloud-hosted system that accommodates relational data and foreign key constraints was made possible by the use of Azure SQL Database.



Sql data base

Azure Blob Storage

Because Azure Blob Storage is very scalable and perfect for storing huge binary things like venue and event photographs, it was used to manage images. It also facilitates safe storage and speedy retrieval.



EF Core

ORM mapping was done with EF Core, which allowed for effective communication with the Azure SQL Database using LINQ queries and strongly typed C# classes. It made CRUD and database administration easier.

The Azure App Service

The web application was deployed in the cloud using the Azure App Service, which enables seamless scaling, continuous deployment, and direct GitHub connection for modifications in the future.

HTML and Bootstrap

A responsive and aesthetically pleasing front-end was developed using HTML and Bootstrap, while ASP.NET MVC's Razor Views enabled dynamic content rendering.

Alternative Elements (Taken into Account)

Database storage might have been done with Firebase or MongoDB Atlas, but unlike Azure SQL, they do not natively enforce relational fidelity.

Although Azure Blob Storage offered superior interoperability with the rest of the Azure ecosystem, Amazon S3 could have been utilized for picture storage.

Although they would have required more API levels and would have overcomplicated the project scope, Angular or React might have been utilized for the front end.

Reflection on the Project

Throughout my academic career, this cloud development project has been one of the most rewarding and challenging experiences I've had.
I lost the complete submission in Part 1 since I had trouble integrating GitHub and was unable to push my code in time. This was really discouraging, but I used it as a lesson rather than abandoning up.

Part 2 presented yet another set of challenges. Although I was able to get the system to function, my solution did not adhere to the rubric's requirements. Important aspects like adequate image storage and thorough validation were absent from my implementation. I came to the realization that in order to create a workable solution, I needed to focus more on the details and fully comprehend the rubric's requirements.

I made a self-promise that this would be my comeback when I got to Part 3. I pushed myself to create a completely working, cohesive solution by taking all of the criticism, errors, and frustrations from the preceding sections. I concentrated on:

.being aware of the connections between the models.
.enhancing my ability to handle errors.
.Understanding Azure deployment procedures to guarantee a functional system.

Along the process, I ran into a number of technical blunders, including SQL migration issues, mismatched model relationships, and foreign key concerns. However, I persevered this time. I investigated, debugged, and improved the application until it performed as I had anticipated.

Throughout this project, I encountered several **technical difficulties** that impacted my progress but also helped me grow significantly as a developer.

Deployment Issues:

One of the biggest problems I faced was **deploying the web app to Azure**. I struggled with setting up the correct Azure App Service configuration, and the web application did not work on the internet initially. It took time to understand how to troubleshoot deployment logs, adjust the publish profile, and link the correct database to the live system. Eventually, I was able to get the app running online, but this part taught me that deployment requires a different set of skills beyond just writing code.

Database/Table Challenges:

I also had issues with **creating and managing the SQL database tables**. At first, my table structure didn't support the filtering logic correctly — especially when it came to adding fields for venue availability and handling the date range. I had to go back and make several changes to the database schema, update my Entity Framework migrations, and ensure the data was correctly seeded or entered to test the features. It was a frustrating but important learning experience in designing database systems that match real user requirements.

Runtime Errors & Debugging:

Another major challenge was dealing with **runtime errors**. During development, my application crashed several times due to incorrect logic, null reference errors, or configuration mismatches. Debugging these errors, especially in a cloud environment, was difficult at first. However, I learned how to use Visual Studio's debugging tools more effectively and how to check error messages in Azure's logs. Each error helped me get better at solving problems quickly and understanding how different parts of the application interact.

By facing and overcoming these issues, I developed stronger **problem-solving, debugging, and cloud deployment skills**. I learned that creating a functional cloud-based application takes much more than just writing code—it also requires patience, testing, and the ability to adapt when things don't work as expected.

Knowledge Acquired

.Making a plan is important. I learned to always test my deployment processes before committing to them after missing out on Part 1's GitHub submission.

.Take note of the rubric. It's important to design the system they requested, not simply a system.

.Talent is surpassed by perseverance. Even though I made a lot of mistakes along the road, I managed to get everything together for Part 3 by persevering.

.Full-stack thinking is necessary while using cloud technologies. Getting the web application to function locally is insufficient; data integrity, scalability, and cloud deployment are equally crucial.

Present Knowledge about Cloud Applications

I now understand the design and development of cloud-based applications on a deeper level. Coding is only one aspect of cloud development; another is comprehension:

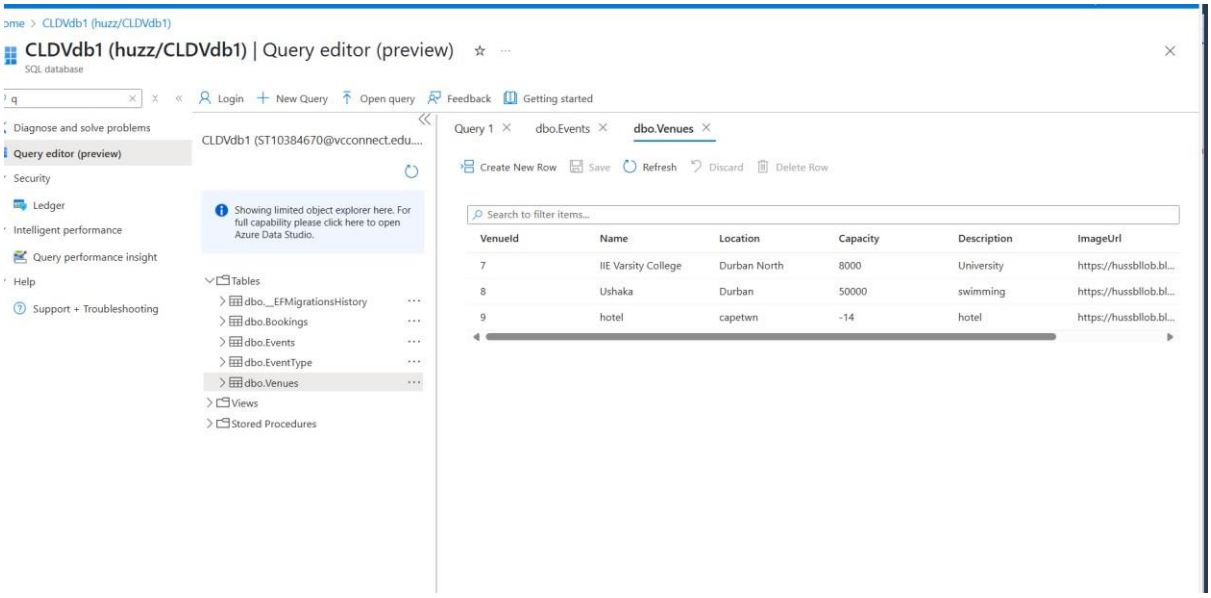
- Deployment pipelines
- Cloud storage solutions
- Data management at scale
- Error handling in a live environment
- Building for real users

Additionally, I now know how to manage cloud-hosted databases, use Azure services efficiently, create a system that is both practical and easy to use, and organize projects according to best practices.

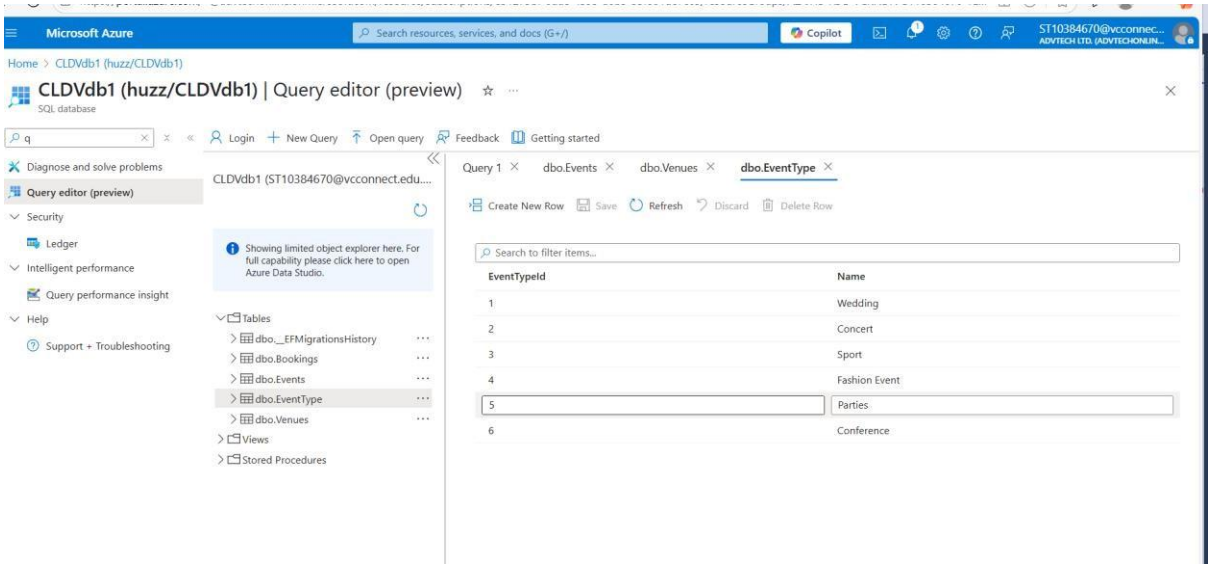
This project served as my atonement, and I am pleased with the results of this last section. Despite my past mistakes, I've learned a lot and can claim with confidence that I now know what it takes to create, implement, and maintain a cloud-based web service.

Screenshots of azure labs

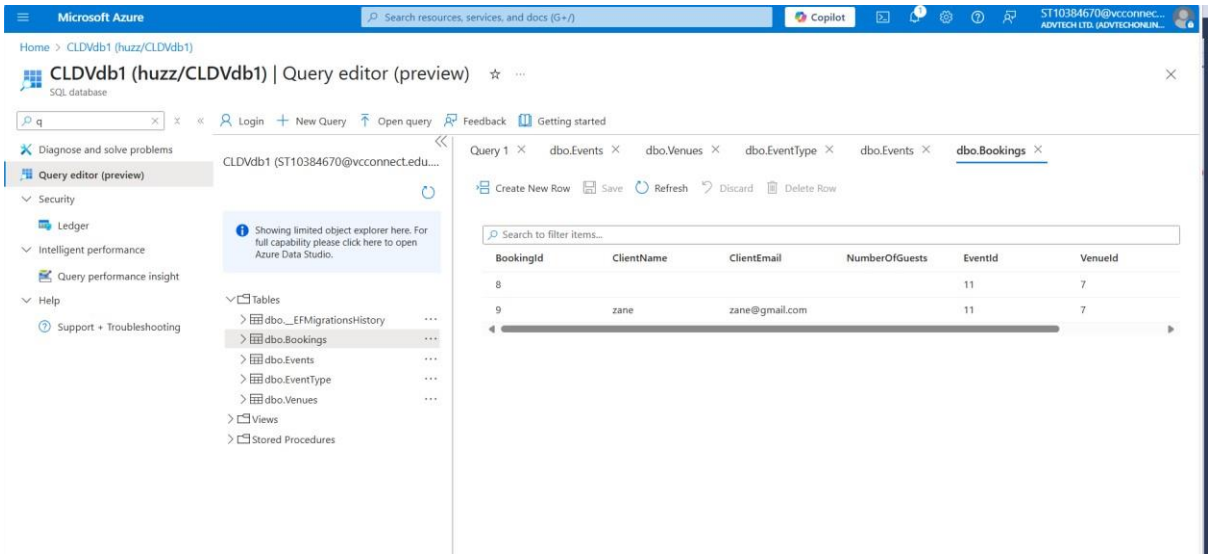
This is my new Venue table in the query editor



This is my my new Eventype table



This is my new Bookings table with filtering



Knowledge about Cloud Application Architecture
My comprehension of developing and implementing cloud applications with Microsoft Azure has greatly increased as a result of this project.

I now know how services like these are used to architect cloud-based systems:

The web application will be hosted via Azure App Service.

Azure SQL Database for safe data storage and querying

GitHub for integration with continuous deployment and code management.

I now know how to:

Use CI/CD pipelines to automatically deploy code.

Create and modify SQL tables to provide new functionality such as filtering,

Diagnose deployment faults and resolve problems in various environments.

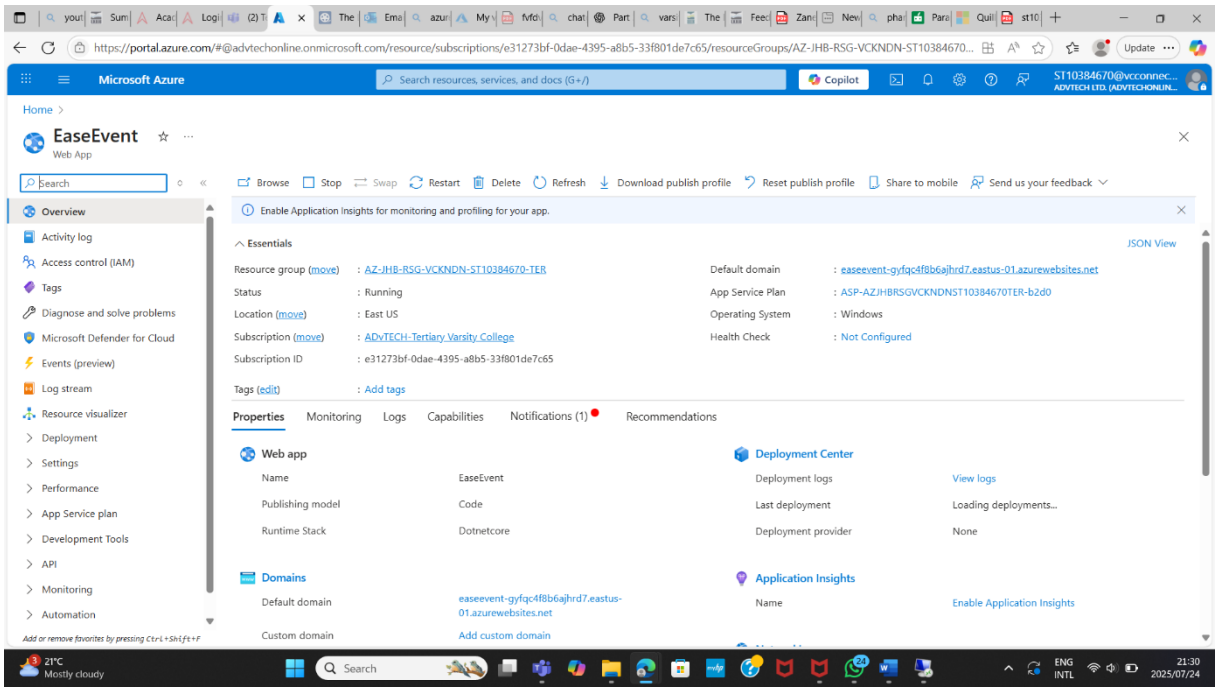
Create scalable solutions that keep front-end, back-end, and data storage issues apart.

This practical experience has equipped me to work well with others in cloud-based development environments and create future production-ready cloud applications.

🔗 Links

- **Live Website**: https://easeevent-gyfqc4f8b6ajhrd7.eastus-01.azurewebsites.net

- **GitHub Repository**: *(https://github.com/ST10384670/CLVD_EVENTEASE.git)*



Here is a screenshot of the webapp on azure labs

References

- Microsoft. (n.d.). *ASP.NET Core MVC overview*. Microsoft Learn. Retrieved June 23, 2025, from https://learn.microsoft.com/en-us/aspnet/core/mvc/overview
- Microsoft. (n.d.). *What is Azure SQL Database?*. Microsoft Learn. Retrieved June 23, 2025, from https://learn.microsoft.com/en-us/azure/azure-sql/database/sqldatabase-paas-overview
- Microsoft. (n.d.). *Introduction to Azure Blob Storage*. Microsoft Learn. Retrieved June 23, 2025, from https://learn.microsoft.com/enus/azure/storage/blobs/storage-blobs-introduction
- Microsoft. (n.d.). *Entity Framework Core Overview*. Microsoft Learn. Retrieved June 23, 2025, from https://learn.microsoft.com/en-us/ef/core/
- Microsoft. (n.d.). *What is Azure App Service?*. Microsoft Learn. Retrieved June 23, 2025, from https://learn.microsoft.com/en-us/azure/app-service/overview
- W3Schools. (n.d.). *Bootstrap 4 Tutorial*. Retrieved June 23, 2025, from https://www.w3schools.com/bootstrap4/
- Microsoft. (n.d.). *Razor syntax for ASP.NET Core*. Microsoft Learn. Retrieved June 23, 2025, from https://learn.microsoft.com/en-us/aspnet/core/mvc/views/razor