POE

Cloud Development

CLDV6211

ST10061509

Mohammed Moosa

Reflective Technical Report: EventEase Web Application

Introduction.

This report is the record of all the steps involved in the development process of the EventEase application, which is a web-based system for booking events that has been implemented in ASP.NET Core MVC and hosted in Microsoft Azure. In this report, the aim is to contemplate on the technical choices made during the development process and usage of Azure services in supporting cloud-native features. The discussion also contains an in-depth overview of the characteristics of the application, the technologies and services used, and what were some of the important lessons learnt through the project.

Application Features List.

EventEase system is a web-based application, which incorporates installation of cloud computing and possesses the following features:

Venue Management: Add, edit and remove venues. The images of the venues are uploaded and stored on the Azure Blob Storage.
Event Management: Add, view, update, and delete the events, and each of the events has an optional venue.
Booking System: The people will be able to book an event at the available venue. The system eliminates multi booking of the same date.
Booking and enhanced booking search: It is possible to search a booking through filtering by customer name, event name, venue name, venue availability, booking date parameter, and outlining the event.
Azure Deployment: All of the system is deployed and hosted on Azure Web App and has an Azure SQL Database as a backend and an Azure Blob Storage to store media files.

Component Discussion.

Azure Web App:
The EventEase application was hosted in Azure Web App. This one was chosen because it takes advantage of seamless compatibility with the ASP.NET Core, GitHub CI/CD pipelines, and embedded scaling and diagnostic features. This service saved the need of managing the underlying foundations, and version control and was quickly deployed.
Other Possible: Amazon elastic beans talk or Google app engine might have been the options, however the Azure Web App was the most direct and streamlined as to hosting ASP.NET core applications.

The Azure SQL database:

Any organized application information such as bookings, and events and location were stored on Azure SQL Database. It provided automated backups, maximum availability, and inbuilt support of the .NET applications. Azure SQL allowed using Entity Framework Core to map objects to the database and facilitated CRUD requests.
Alternate solution: Other options available like PostgreSQL or MySQL may have been applied in regards to the Azure Database services, yet the Azure SQL Database option was more appropriate, as it relates to the Visual Studio and the ef core framework.

The Azure Blob Storage:

Upload and storage of images of the venue was done using Azure Blob Storage. The pictures were not linked to the database directly, but they only saved the URL, whereas the binary data were stored to Azure Blob Storage. This structure also assisted in preventing unscrupulousness in the databases as well as offering a scalable management of media assets.
Alternative: Amazon S3 or Firebase Storage would have done the same. However, Azure Blob Storage became smoother to use along with Azure Web App and allowed a more convenient management of the SAS tokens.

Technologies applied and line of reasoning.

ASP.NET Core MVC (.NET 9): It will be utilized as the core framework because of its consistent model-view-controller framework structure, performance benefits, and compatibility with Azure services.

Entity Framework Core: Supplied the ORM capabilities to operate with the SQL database by using C# object and LINQ, so less sql code was required.

Azure Services: Blob Storage of media and Web App of deployment and SQL Database of relational data, can comprise a powerful cloud-native infrastructure.

Bootstrap: It has been used in responsive, front-end UI design and layout.

GitHub: Source control and azure pipelines integration of CI/CD.

Visual Studio: Offers complete resources of developing, debugging and publishing tool whether the ASP.NET specific.

Project Reflection 1 to 3.

Part 1: Initial Configuration and Database Connection.
The first stage of development took place and involved designing of the core models (Booking, Event, and Venue) and realization of the CRUD functionality. Database was linked on an instance of Azure SQL, and a simple management portal was created on each entity. It was a stage that set the platform of the app data layer and interface.
Key learning outcome: Acquiring knowledge on how to arrange an ASP.NET Core MVC project and to connect it with a SQL database that is hosted in the cloud.

Part 2: Upload of Image and Cloud Storage.
The phase entailed the incorporation of the Azure Blob Storage to upload images of the venue. This removed dependence of the file system on the web server as well as providing cloud-based scalable storage. Sharing of the media files was deemed secure through the shared access signature (SAS) URLs which allowed authorized users to access the media files.
Learning outcome key: I have gained experience in how to manage data storage on the binary and deploy safe access procedures with the help of cloud solutions.

Part 3 Advanced Search and Filtering.
During the last stage of location-based search, the feature that enabled filtering of bookings by type of events, period, and availability of a venue, was introduced. That necessitated more intensive utilization of LINQ, nullable types, and dynamic query construction. To organize the filtered data of booking, view models were presented and user data input was managed to prevent running time problems.
Important learning objective: Better knowledge concerning dynamic querying, model binding, user interface logic, and user experience enhancing custom filtering.

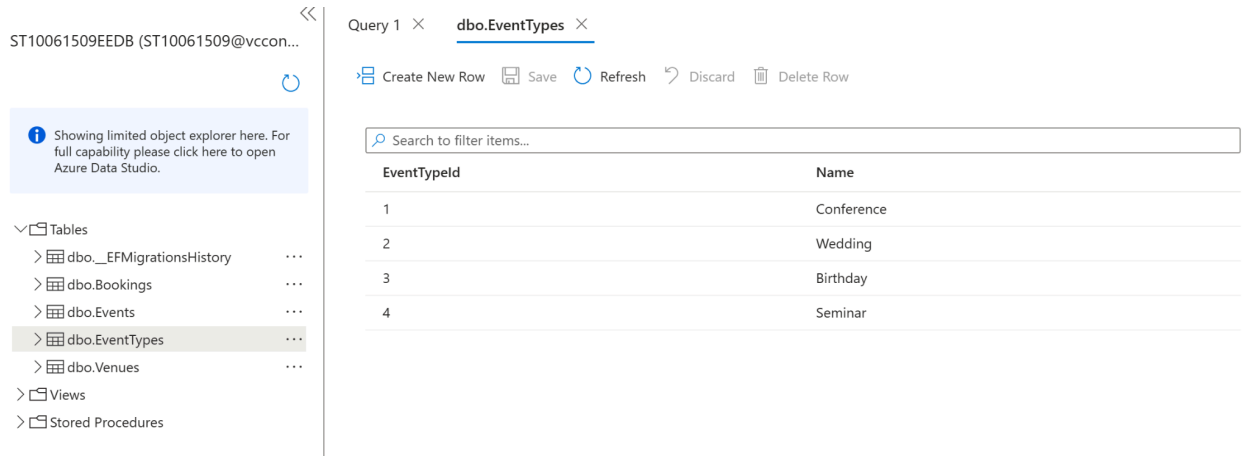
Lessons Learned and Reflection.

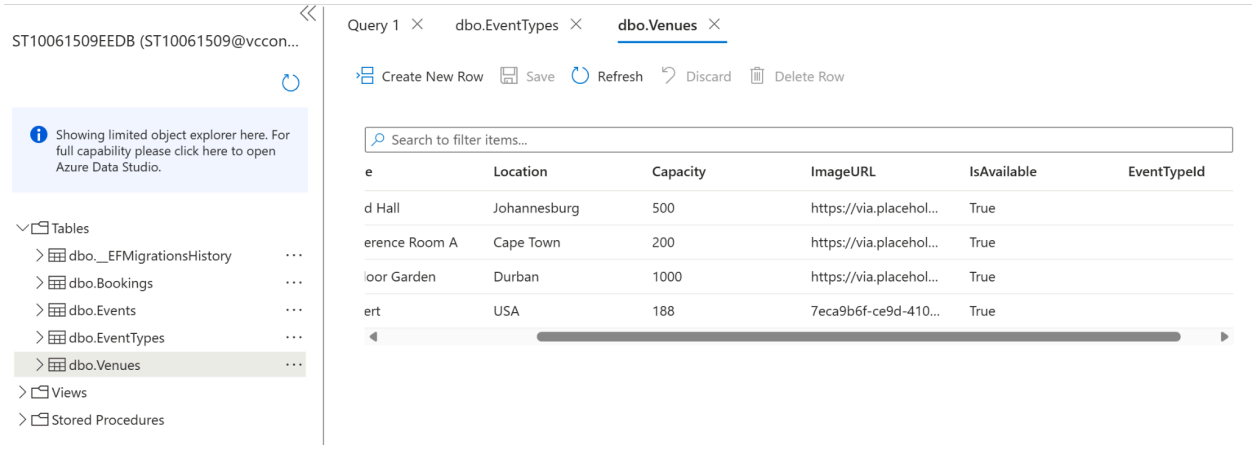
The project based on EventEase book taught practical skills on creating data-driven scalable Web applications using the Azure platform. It showed how cloud services may be used to outsource infrastructure in order to work on the application logic and the user experience.
Among several main findings, the following one can be taken into consideration:
The importance of isolating concerns: The application was scalable and maintainable by using Azure SQL, to work on structured data, and Azure Blob·Storage, to work on unstructured data.
The necessity of model validation: We wanted to prevent invalid and duplicate booking and this needed both server-side and client-side validation.
The significance of cloud integration: Azure services also maintained a huge advancement and implementation rate thus promoting the significance of cloud-first designs in currently utilized apps.
Good development practices: Source control system, structured models, reusable views and error treatment helped in contributing to maintainable and reliable code base.

Conclusion.

The creation of the EventEase app has been a fulfilling process in designing and implementation of contemporary cloud-based systems. By applying the techniques of Azure Web App, SQL Database, and Blob Storage, the application was integrated entirely with Microsoft cloud technologies. The project has enhanced technicalities and the architectural process of constructing real world web applications in an appropriate and scalable way.

Screenshots





Links.

https://st10061509eventease.azurewebsites.net/

https://github.com/IIEWFL/cldv6211-poe-ST10061509-Mohammed-Moosa

Reference List:

Microsoft. (2024). *What is Azure App Service?* [online] Microsoft Learn. Available at: https://learn.microsoft.com/en-us/azure/app-service/overview [Accessed 22 Jun. 2025].

Microsoft. (2024). *Overview of Azure SQL Database*. [online] Microsoft Learn. Available at: https://learn.microsoft.com/en-us/azure/azure-sql/database/sql-database-overview [Accessed 22 Jun. 2025].

Microsoft. (2024). *Introduction to Azure Blob Storage*. [online] Microsoft Learn. Available at: https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction [Accessed 22 Jun. 2025].

Microsoft. (2024). *Entity Framework Core overview*. [online] Microsoft Learn. Available at: https://learn.microsoft.com/en-us/ef/core/ [Accessed 22 Jun. 2025].

Microsoft. (2024). *ASP.NET Core MVC overview*. [online] Microsoft Learn. Available at: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-9.0 [Accessed 22 Jun. 2025].

GitHub. (2024). *Introduction to GitHub*. [online] GitHub Docs. Available at: https://docs.github.com/en/get-started/quickstart/hello-world [Accessed 22 Jun. 2025].

W3Schools. (2024). *Bootstrap 5 Tutorial*. [online] Available at: https://www.w3schools.com/bootstrap5/ [Accessed 22 Jun. 2025].

Amazon Web Services. (2024). *Elastic Beanstalk*. [online] Available at: https://aws.amazon.com/elasticbeanstalk/[Accessed 22 Jun. 2025].

Google Cloud. (2024). *Google App Engine documentation*. [online] Available at: https://cloud.google.com/appengine/docs [Accessed 22 Jun. 2025].

Amazon Web Services. (2024). *Amazon S3*. [online] Available at: https://aws.amazon.com/s3/ [Accessed 22 Jun. 2025].

Firebase. (2024). *Firebase Storage Overview*. [online] Available at: https://firebase.google.com/docs/storage [Accessed 22 Jun. 2025].