**EZEKIEL ISAAC NARAYANASAMI**

**ST10359128**

**CLDV6211**

**POE PART 3**

**REFLECTIVE TECHNICAL REPORT**

While doing my project for the EventEase booking system, my task at hand was to design, develop, and deploy a cloud-based venue booking system. This report with reflect on the development process all the way from part 1 to part 3, showing the key features that were implemented, the technology used, the tools used, how Azure cloud services were used, and the lessons and values I have learned while making this project.

The system has many features to meet the needs of an event management company. These were developed and structured around real-world requirements that were given in the business scenario. The core features include venue management, event management, booking functionality, validation, cloud integration, and searching capabilities.

The venue management part allows booking people to create, view, update, and delete venues and each venue has a description to show where it is, the capacity it can hold, and images. The images are added through urls but then was replaced by Azure Blob storage which is more secure and scalable. The event management part allows the staff to manage the events and link them to the venues. Each of them storing data such as name. start and end dates, and the type of event.

The bookings will be made by selecting an event and a venue of where it would be held. The system uses logic so that it will prevent double bookings, making sure that two events cannot overlap at the same venues on the same dates and times. Also, error handling and validation was implemented to ensure data integrity, such as not allowing deletion of the venues or events with existing bookings and displaying error messages when required fields are missing.

In part 3 of this poe we added in enhances search and filtering, which allows users to search for bookings by the booking ID or event name and filter by event type, data range, and venue availability. Booking display was created to keep all relevant booking data which improves the readability of the information and making it easier for the staff.

Because of its built-in dependency injection, support for the Model-View-Controller pattern, robust Razor view support, and ease of connection with Entity Framework Core and Azure services, ASP.NET Core MVC was used to build the system. To promote cloud-based storage with high availability and scalability, the back-end database was first planned and implemented using SQL Server and then later deployed as an Azure SQL Database.

The application first handled images using static urls, however this method was inflexible and unsecure. I added Azure Blob Storage to solve this, allowing users to upload and retrieve event and location photos from the cloud. This allowed safe access to media files, automatic backup, and scalable storage. I chose Azure App Service for the application's hosting, which made it possible to deploy it easily from Visual Studio. This solution was good for an expanding company like EventEase since it took away the work of managing infrastructure and guaranteed high availability. Azure SQL was more suitable given the type of relational, structured data. The MVC framework was utilized to efficiently render dynamic HTML information using Razor Pages.

My personal perspective on this project was that it proved to be a valuable learning experience in my journey of doing software development. In part one I was focused on building out the CRUD functionality and full understanding the basics of MVC. In part two of the project it introduced more advanced methods which I did struggle with but ended up getting a basic background to do more and further the project. This caused me to start thinking more about data integrity and user experience. One of the challenges I faced was Azure Blob storage because I could not get the hang of it but eventually I did, but, this taught me how to design user friendly systems that maintain data integrity and prevent more user errors. In part three I had to investigate the importance of designing databases that support flexible querying. I also learned how to create database views, which made the performance better and made it easier to display joined data.

Overall this project has proved to be a challenge for me but through careful work and gaining as much understanding on certain aspects and methods I was able to proceed to completing this poe. I have gained practical and mental knowledge by doing these tasks and somewhat confidence when it comes to doing another project like this. I will further my knowledge to make better projects and fix silly errors, making it so that I can apply this knowledge in real world situations.

**REFERENCES**

Microsoft, 2024. ASP.NET Core MVC overview.

<https://learn.microsoft.com/en-us/aspnet/core/mvc/overview>

Microsoft, 2024. Azure App Service documentation.

<https://learn.microsoft.com/en-us/azure/app-service/>

Microsoft, 2024. What is Azure SQL Database?

<https://learn.microsoft.com/en-us/azure/azure-sql/database/sql-database-overview>

Microsoft, 2024. Introduction to Azure Blob Storage.

<https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction>

Microsoft, 2024. Entity Framework Core Overview.

<https://learn.microsoft.com/en-us/ef/core/>

Microsoft, 2024. Azure Cognitive Search documentation

<https://learn.microsoft.com/en-us/azure/search/search-what-is-azure-search>

Microsoft, 2024. Azure SQL Database: Views.

<https://learn.microsoft.com/en-us/sql/relational-databases/views/views?view=sql-server-ver16>