**Deployed Application URL:**

**AZURE WEB APP URL HERE** –

[<https://eventeasewebappst10329226-amgrhyh6hsa9a0e5.southafricanorth-01.azurewebsites.net> ]

**Reflection on project**

The development of the EventEase application has been a deeply insightful journey, offering practical experience in designing, developing, and deploying cloud-based applications.

**Personal Perspective and Challenges Faced**

From a personal perspective, this project represented a hard but rewarding learning curve, especially in navigating the complexities of cloud deployment and Entity Framework Core (EF Core) database migrations. Although originally overwhelming, each obstacle offered valuable insights that deepened my technical understanding. (Gemini, 2025)

One of the most challenging yet educational aspects was managing EF Core migrations, particularly when adding new required foreign keys (e.g., EventTypeID, VenueID) to existing tables containing data. Recurrent errors such as “Invalid column name” and “Foreign key conflict” became familiar and incited a deeper dive into the migration process (Gemini, 2025).

**lessons learned:**

* Understanding the EF Core Migration Lifecycle:
  + learning Add-Migration generates the SQL script and how Update-Database applies it to the actual database (Microsoft, 2024b).
* Manual Migration File Adjustments:
  + Learning that EF Core’s auto-generated migrations may not handle complex updates well, often requiring the use of migrationBuilder.Sql() to insert default values or update existing records before using non-nullable constraints (Microsoft, 2024b).
* Database State Management:
  + Realizing the importance of the EFMigrationsHistory table for tracking useful migrations, and, in some development cases, having to reset the database by clearing the migration history to solve the problem (Microsoft, 2024b).
* Connection String Accuracy:
  + Making sure that the correct database (local or Azure SQL) was targeted before applying migrations crucial in avoiding unintentional changes to the wrong environment.

Also, a small yet informative issue involved rendering raw HTML in Razor views, such as displaying badges for IsAvailable status. This required the use of @Html.Raw() to prevent HTML from being encoded. This showed the importance of understanding Razor’s default rendering behaviour.

These experiences have significantly strengthened my problem-solving skills and deepened my understanding of the ASP.NET Core ecosystem, database lifecycle management, and real-world cloud integration (Microsoft, n.d.-a; Microsoft, n.d.-b).

**Lessons Learned**

Throughout the development of the EventEase application, many important lessons were reinforced through hands-on experience:

* Iterative Development and Testing
  + Regular testing after each significant change especially involving database schema or front-end integration proved crucial. Frequent tests helped catch issues early and reduced the need for time-consuming debugging later. (Gemini, 2025)
* Value of Database Migrations
  + EF Core migrations, while challenging, proved to be a powerful tool for managing database schema changes in a version-controlled and structured way. Learning how to customize migration files for data seeding or advanced modifications was a major takeaway. (Gemini, 2025)
* Data Integrity is Crucial
  + Foreign key conflicts highlighted the importance of maintaining consistent and valid data relationships. Proper planning when introducing new foreign keys guaranteed existing data remained compatible and avoided deployment issues. (Gemini, 2025)
* Benefits of Managed Cloud Services
  + Using Azure App Service and Azure SQL Database reduced the operational burden. While there is a learning curve involved in setup and configuration, features like automated patching, backups, and scalability simplified deployment and maintenance (Microsoft, 2024b).
* Interpreting Error Messages
  + The ability to read and understand detailed error messages such as SqlException or InvalidOperationException proved crucial for effective troubleshooting. These messages often contain direct hints to the root cause of the issue (Microsoft, 2024b).
* Power of LINQ with EF Core
  + Leveraging LINQ in Entity Framework Core allowed for concise data queries written in C#. This reduced the need for raw SQL and improved readability and maintainability of data-access logic (Microsoft, n.d.-b).

**Current Understanding of Designing, Developing, and Architecting Cloud-Based Applications**

My understanding of cloud application architecture has developed through the EventEase project, from theory to practical, hands-on experience.

**Takeaways**

* **Scalability as a Core Principle**
  + Cloud development inspires early consideration of scalability. Services like Azure App Service and Azure SQL Database offer built-in scaling, making it easier to handle growth without manual intervention (Microsoft, 2024b).
* **Preference for Managed Services**
  + Leveraging Platform-as-a-Service (PaaS) solutions decreases operational overhead. Using App Service and Azure SQL shifted focus away from infrastructure management and towards application development (Microsoft, 2024b).
* **Security by Design**
  + Cloud platforms offer advanced security features such as firewalls, identity management, data encryption and threat detection that must be configured and integrated into the application’s design (Microsoft, 2024a).
* **Ecosystem Integration**
  + Azure’s ecosystem supports building solutions by combining services like App Service, SQL Database, Blob Storage, and Event Grid. Understanding these integrations is crucial for modern cloud-based applications (Microsoft, 2024b).
* **Deployment Pipelines**
  + Not fully implemented in this project, the importance of Continuous Integration and Deployment (CI/CD) was made clear. These practices support reliable, repetitive and well-organized deployments in cloud environments.
* **Cost** 
  + Cloud computing has a pay-as-you-go model. Choosing the right pricing tiers and scaling strategies is crucial to balance performance and cost, strengthening the need for mindful resource management.

**Conclusion**

The EventEase project has been an important exercise in learning how to design and deploy a real-world web application using Azure. It bridged the gap between theoretical and practical skills in cloud computing.

**Deployed Application URL:**

**AZURE WEB APP URL HERE** –

[<https://eventeasewebappst10329226-amgrhyh6hsa9a0e5.southafricanorth-01.azurewebsites.net> ]

**Code Attribution and Referencing:**

**Microsoft. (n.d.-a). ASP.NET Core documentation. Microsoft Learn. Retrieved June 23, 2025, from** [**https://learn.microsoft.com/en-us/aspnet/core/**](https://learn.microsoft.com/en-us/aspnet/core/)

Microsoft. (2024a). What is Azure App Service? Azure Documentation. Retrieved June 23, 2025, from <https://learn.microsoft.com/en-us/azure/app-service/overview>

Microsoft. (2024c). Introduction to Azure Blob Storage. Azure Documentation. Retrieved June 23, 2025, from <https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction>

Smith, J. (2023, April 15). Database normalization explained: Why it matters for cloud applications. CloudGeek Blog. [Replace with an actual blog post or academic article if you used one for normalization concepts, otherwise use a generic source that covers it].

Microsoft. (2024b). Azure SQL Database overview. Azure Documentation. Retrieved June 23, 2025, from <https://learn.microsoft.com/en-us/azure/azure-sql/database/sql-database-paas-overview>

Microsoft. (n.d.-b). C# documentation. Microsoft Learn. Retrieved June 23, 2025, from <https://learn.microsoft.com/en-us/dotnet/csharp/>