**Reflective Technical report Component Discussion**

This section researches the technical choices made during the development of EventEase, covering the specific Azure services and core technologies used, along with a discussion of alternatives.

**Azure Services Used and Potential Alternatives**

**Azure App Service (Web App)**

**Description & Why It Was Used**

The EventEase application is hosted using **Azure App Service**, a **Platform as a Service (PaaS)** solution provided by Microsoft Azure. This service was selected for several strategic reasons: (Microsoft, 2024a)

* **Ease of Deployment**: Azure App Service supports continuous deployment directly from development tools such as **Visual Studio**, **GitHub**, or **Azure DevOps**, reducing the difficulty typically related with manual deployment processes.
* **Scalability**: The platform includes auto-scaling features, allowing the application to dynamically adjust resources based on user demand allowing for better performance during peak usage.
* **Managed Infrastructure**: Azure takes care of server maintenance, operating system patching and infrastructure updates. This allows a development team to concentrate on building applications and features instead of managing backend resources.
* **Cost-Effectiveness**: The pay-as-you-go pricing model offers flexibility for small to mid-sized projects, with numerous pricing tiers that can be adjusted based on the application’s growth and resource needs.

**Azure Virtual Machines (IaaS):**

**Potential Alternative & Why**

**Azure Virtual Machines (Infrastructure as a Service - IaaS)**

* Azure VMs provide full control over the operating systems (OS) and server infrastructure, making them ideal for custom configurations and specific deployment requirements. Though, they come with increased management overhead, needing manual setup and maintenance of tasks such as OS patching, security updates, and infrastructure scaling, which can slow down development and increase operational load. (Microsoft, 2024a)

**Azure SQL Database**

**Description & Why It Was Used**

Azure SQL Database was chosen as the main data storage solution because it is a fully managed interactive database service. It works well for structured data with clear relationships, such as venues, events, bookings, and event types. Being a managed service, it handles backups, updates, and maintenance automatically, reducing the need for manual management. It also offers strong security features like encryption and threat detection and integrates easily with ASP.NET Core and Entity Framework Core for efficient data management. (Microsoft, 2024a)

**Alternative Database Options**

**Potential Alternatives & Why**

**Azure Cosmos DB (NoSQL)**:

* Appropriate for use cases needing a flexible data structure, world-wide reach and low-latency access to non-relational data like user profiles or logs. however, using Cosmos DB would require a different data model approach.

**Azure Database for PostgreSQL/MySQL**:

* These are open-source relational database replacements, perfect if the development team prefers or has experience with these systems instead of SQL Server. (Microsoft, 2024b)

**Azure Table Storage (NoSQL Key-Value)**:

* reasonably priced option for storing large volumes of simple, non-relational data. Though, it's not great for handling complex queries or relationships between entities.

**Azure Storage (Blob Storage)**

**Description & Why It Was Used**

Azure Blob Storage was applied for storing venue images. It is a scalable and cost-effective solution for handling large amounts of unstructured data like images. It provides public URLs for easy access, inserting in web pages and automatic data replication. (Microsoft, 2024a)

**Alternative Storage Options**

**Potential Alternatives & Why**

* **Azure Files**: Designed for shared file access over the SMB (Server Message Block) protocol. It’s better suited for internal file sharing and is not great for public web content like images. (Microsoft, 2024a)
* **Azure Content Delivery Network (CDN)**: Often used alongside Blob Storage to cache and deliver content worldwide. It improves load times and performance for users in different parts of the world.

**Technology and Potential Alternatives**

**ASP.NET Core MVC (C#)**

**Description & Why It Was Used**

It was selected as the framework for developing the web application due to its scalability, making it suitable for enterprise-level projects. It is cross-platform, supporting deployment on Windows, Linux, and macOS. The framework benefits from a strong ecosystem. This includes powerful development tools like Visual Studio, extensive libraries, and active community support. Also, using C# provides advantages such as strong typing, object-oriented programming (OOPS) features and improved developer productivity. (Microsoft, n.d.-a)

**Alternative Frameworks**

**Potential Alternatives & Why**

* **Node.js (Express/NestJS)**: Usually used for building scalable APIs and microservices, mostly in full-stack JavaScript environments. It offers flexibility when using JavaScript or TypeScript across both front-end and back-end development.
* **Python (Django/Flask)**: Perfect for rapid development and data-driven applications. Also, for projects involving data analysis or machine learning.
* **Java (Spring Boot)**: A good, enterprise-grade framework known for consistency, scalability and performance. Widely implemented in large enterprise systems.

**Entity Framework Core (EF Core)**

**Description & Why It Was Used**

EF Core was used as the Object-Relational Mapper (ORM) to shorten database operations. It maps C# classes directly to database tables, reducing the need to write raw SQL. It supports querying through LINQ, making database access more instinctual for C# developers. EF Core also supports migrations, allowing database schema changes to be managed through code. It can be used with different database systems by switching providers. (Microsoft, n.d.-a; Microsoft, n.d.-b)

**Alternative Data Access Options**

Potential Alternatives & Why

* **Dapper**: A micro-ORM that provides better performance and more control over SQL queries. It's appropriate for performance applications but requires more manual SQL code. (Gemini, 2025).
* **ADO.NET**: Allows direct database interaction using raw SQL commands. While it offers maximum control, it’s harder to maintain and increases the risk of errors.

**HTML, CSS, Bootstrap, JavaScript**

Description & Why It Was Used

* **HTML5:** delivers the structure of web pages.
* **CSS3:** handles styling and visual presentation.
* **Bootstrap:** allows fast responsive UI development with pre-built components and grid system.
* **JavaScript:** improves client-side interactivity and dynamic content updates. (Gemini, 2025).

**Frontend Frameworks (React, Angular, Vue.js):**

Potential Alternatives & Why

* **React, Angular, Vue.js**: These frontend frameworks are great for building Single Page Applications (SPAs) with dynamic, component-based user interfaces. However, for this project, they were not used because of complexity and a steeper learning curve. (Gemini, 2025).
* **Tailwind CSS**: A CSS framework that offers better control over styling compared to Bootstrap. however, it requires more manual class composition, which can slow down development for smaller projects. (Gemini, 2025).

**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/>