Part 3 — Deployment and Finalisation

3.1 Deployment Approach

The Claim Management and Coordination System (CMCS) prototype was deployed to Azure App Service with an Azure SQL Database backend. This deployment model was chosen because Azure provides high availability, scalability, and integrated monitoring, which are crucial for enterprise-grade applications (Microsoft Docs, 2025a).

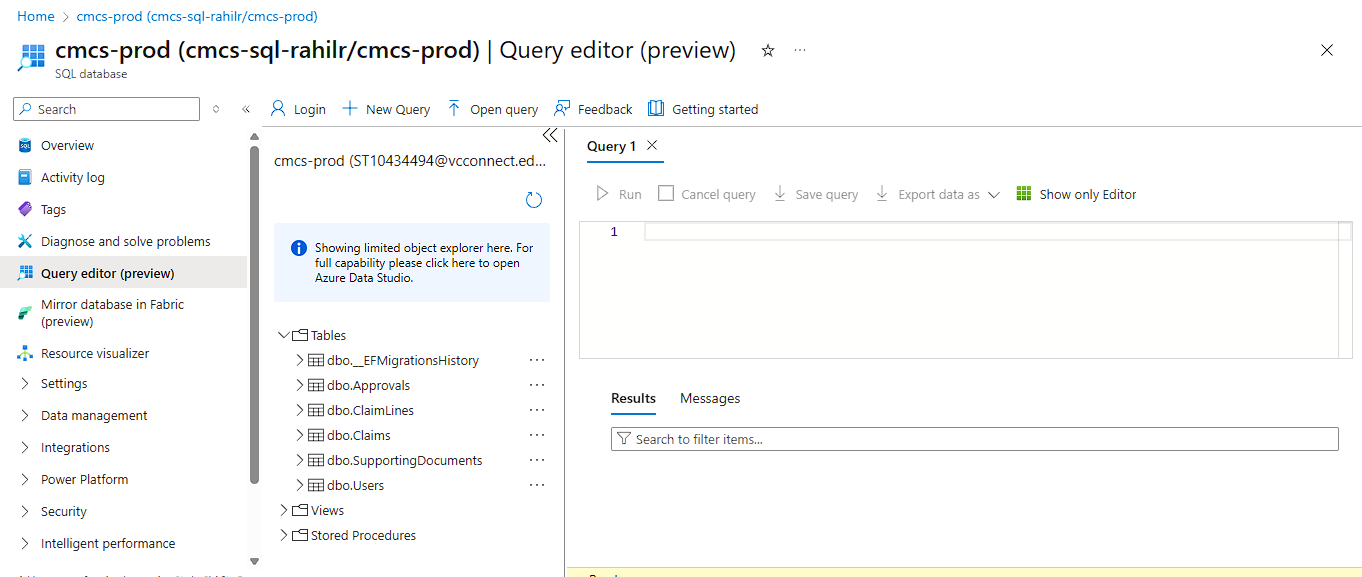
Compared to an on-premise deployment, Azure removes the need for physical infrastructure maintenance, offers automatic backups, and integrates easily with Visual Studio’s publishing tools. This simplified the process of publishing the ASP.NET Core MVC application directly from the development environment.

3.2 Database Migration

The Entity Framework Core migrations were executed against the Azure SQL instance using the following command:

dotnet ef database update --connection "Server=tcp:cmcs-sql-rahilr.database.windows.net,1433;Database=cmcs-prod;User ID=cmcsadmin;Password=\*\*\*\*\*\*;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;MultipleActiveResultSets=True"

This successfully created the necessary tables: Users, Claims, ClaimLines, SupportingDocuments, and Approvals. Seeded data (lecturers, coordinator, and manager) was inserted during the migration.



Screenshot: Azure Portal → SQL Database → Query Editor showing the tables list.

3.3 App Service Configuration

The application was configured in Azure App Service to use the DefaultConnection connection string. This was added under Configuration → Application settings as a secure environment variable of type SQLAzure.

A screenshot of a computer

AI-generated content may be incorrect.

Screenshot: Azure App Service → Configuration → Connection strings.

The Program.cs was adjusted to read the connection string dynamically from configuration:

builder.Services.AddDbContext<AppDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

This ensures that the deployed version uses Azure SQL rather than the local development database.

3.4 Application Publish

The application was published directly from Visual Studio using the Publish to Azure App Service wizard. Once deployed, the system became accessible via the public URL provided by Azure App Service.

A screenshot of a computer

AI-generated content may be incorrect.

Screenshot: Visual Studio publish success screen.

A screenshot of a computer

AI-generated content may be incorrect.  
Screenshot: Browser showing live application URL (e.g., Home page + Lecturer Claims page).

3.5 Testing on Cloud Environment

After deployment, functional testing confirmed that:

* Lecturers can submit new claims, with automatic total calculations.
* Coordinators can approve or reject claims.
* Managers can view claims grouped by status.
* Supporting document uploads work correctly within Azure.
* All seeded users (Lecturer, Coordinator, Manager) were accessible from the live site.

A screenshot of a computer

AI-generated content may be incorrect.

Screenshot: Example claim created and viewed on the live Azure-hosted app.

3.6 Reflection

Deploying to Azure demonstrates how cloud hosting simplifies application management. The environment removed dependency on local machines and provided scalability for future growth. The biggest challenge encountered was SQL server firewall restrictions, which initially blocked access. This was resolved by adding the client IP address and enabling “Allow Azure services” under networking.

This step highlights the importance of configuring secure yet flexible network access for cloud resources (Microsoft Docs, 2025b). Overall, the deployment process provided hands-on exposure to real-world hosting practices and strengthened understanding of cloud-based delivery.

Bibliography

* Microsoft Docs. (2025a). *ASP.NET Core app deployment to Azure*. Available at: https://learn.microsoft.com/azure/app-service/quickstart-dotnetcore (Accessed: 24 September 2025).
* Microsoft Docs. (2025b). *Azure SQL Database firewall rules*. Available at: https://learn.microsoft.com/azure/azure-sql/database/firewall-configure (Accessed: 24 September 2025).