**[Full Stack Development Demo Project](https://github.com/buddhika85/Full-Stack-Demo-App/tree/main)**

**July 2026 – Now**

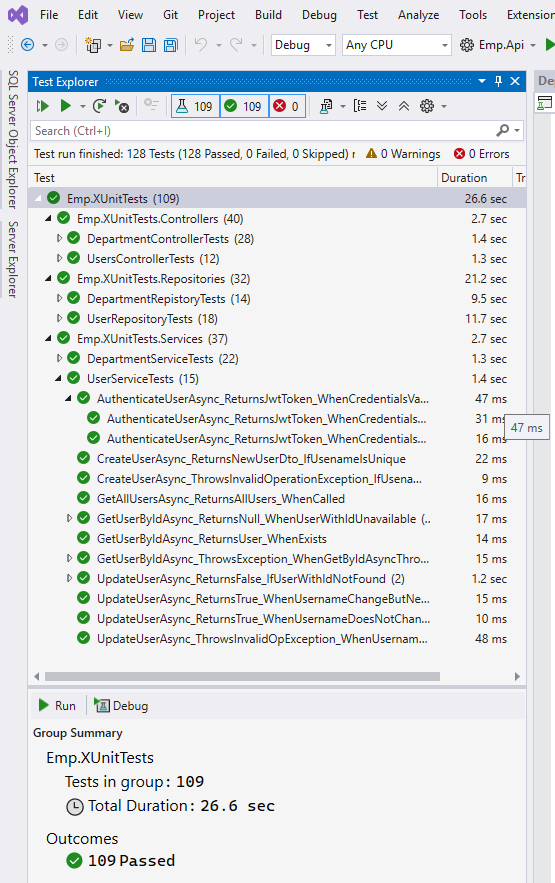
* Full Stack Web Development Project to showcase skills and experience I gained from previous industry projects.
* In summary it is an Azure Hosted Angular 20 Web Application which consumes an ASP.NET Core Web API written in .NET 9. This Web API encapsulates reusable business logic and data access to Azure Hosted MS SQL Server Database.
* Backend Project Architecture –

|  |  |
| --- | --- |
|  | **Emp.Api (ASP.NET Core Web API Project):**   * **Role:** The entry point for HTTP requests. Handles routing, authentication/authorization, and delegates business logic to the Application layer. * **Contents:** Controllers, Program.cs containing DI container service configurations, Middleware pipeline configuration, Compilation and Kestral Server Hosting, Custom Middleware, Filters, Seri Log Log Files and appsettings.json. * **References:** Emp.Application, Emp.Core, Emp.Infrastructure |
| **Emp.Application (Class Library):**   * **Role:** Contains the application's business logic and orchestrates operations. It uses DTOs for data exchange with the API layer. It consumes entities coming from Repositories of Infrastructure Layer. * **Contents:** Service Implementations * **References:** Emp.Core, Emp.Infrastructure |
| **Emp.Core (Class Library):**   * **Role:** The heart of the domain. Contains core entities, DTOs, Enums, Extensions, Custom Validation Attributes, shared interfaces, and common abstractions that are independent of any specific technology (like EF Core or ASP.NET). * **Contents:** Models (Entities: Employee, Department…etc), DTOs, Mapping Extenstions from Entities to DTOs, Repository Interfaces (IGenericRepository<T>, IEmployeeRepository, IDepartmentRepository), Unit of Work Interface (IUnitOfWork)…etc * **References:** None |
| **Emp.Infrastructure (Class Library):**   * **Role:** Handles data access, stores DB First migrations and external concerns. Implements the interfaces defined in Emp.Core. * **Contents:** ApplicationDbContext, Concrete Repository Implementations (GenericRepository<T>, EmployeeRepository, DepartmentRepository), Unit of Work Implementation (UnitOfWork), Migrations * **References:** Emp.Core, Microsoft.EntityFrameworkCore.\*. |
| **Emp.Tests (xUnit Test Project):**   * **Role:** Contains all unit tests written using XUnit and MOQ. * **Contents:** Test classes for Controllers, Services, Repositories. * **References:** Emp.Api, Emp.Application, Emp.Infrastructure (for testing concrete implementations), xunit, Moq, Microsoft.EntityFrameworkCore.InMemory. |

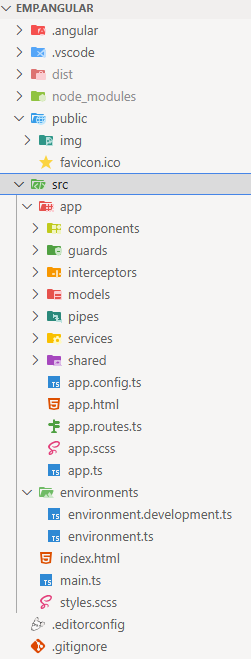
* **Backend Technologies utilized –**

|  |  |  |
| --- | --- | --- |
| Target Framework | .NET | net9.0 |
| Logging in Text Files | Serilog | 9.0.0 |
| API Testing – Web UI | Swashbuckle - Swagger | 9.0.1 |
| API Documentation and Testing postman collection | OpenApi | 9.0.7 |
| DB Communication | Microsoft.EntityFrameworkCore.SqlServer | 9.0.7 |
| DB Migrations | Microsoft.EntityFrameworkCore.Design | 9.0.7 |
| Unit Testing | xunit | 2.9.2 |
| Unit Testing – Mocking | moq | 4.2 |
| Unit Testing – Assertions | FluentAssertions | 8.5 |
| BCrypt.Net-Next | For Password hashing | 4.0.3 |

* **Backend Highlights** 
  + Modern architecture which uses
    - Controllers,
    - Services,
    - Unit of Work,
    - Generic and Specific Repositories encapsulate DB interactions.
  + JWT token generation, blacklisting, Authentication Authorisation
    - Program.cs is configured to Validate JWT Bearer tokens submitted via HTTP Request Authorisation Header
      * Token compromised or not
      * Issuer of token
      * Expiration
      * Blacklisted token or not
  + Caching
    - Output caching for some Action Method results and Cach Evicting
  + Seri Log Logging
  + Custom ASP.NET Middleware
    - CustomExceptionMiddleware - <https://github.com/buddhika85/Full-Stack-Demo-App/blob/main/Backend/Emp.Api/Middleware/CustomExceptionMiddleware.cs>
  + Custom ASP.NET Filter
    - ConsoleLoggerFilter - <https://github.com/buddhika85/Full-Stack-Demo-App/blob/main/Backend/Emp.Api/Filters/ConsoleLoggerFilter.cs>
  + Custom ASP.NET Validation Attributes
    - FirstLetterUpperCaseAttribute -<https://github.com/buddhika85/Full-Stack-Demo-App/blob/main/Backend/Emp.Core/ValidationAttributes/FirstLetterUpperCaseAttribute.cs>
* **Unit Testing** 
  + Wrote more than 100 unit tests showcasing ability to write unit tests for API controllers, service layer classes and repository class with the use of X-Unit, MOQ for mock dependency and Fluent Assertions for Assertions.

****

* + - Department Repository Unit Tests - <https://github.com/buddhika85/Full-Stack-Demo-App/blob/main/Backend/Emp.XUnitTests/Repositories/DepartmentRepistoryTests.cs>
    - Department Service Unit Tests - <https://github.com/buddhika85/Full-Stack-Demo-App/blob/main/Backend/Emp.XUnitTests/Services/DepartmentServiceTests.cs>
    - Department Controller Unit Tests - <https://github.com/buddhika85/Full-Stack-Demo-App/blob/main/Backend/Emp.XUnitTests/Controllers/DepartmentControllerTests.cs>
* **Front End**

****

* + Developed an Angular Application to showcase understanding of components, services, HTTP interceptor, Pipes, Decorators, Auth guards, Lazy loading of routes…etc
  + JWT Interceptor - <https://github.com/buddhika85/Full-Stack-Demo-App/blob/main/Emp.Angular/src/app/interceptors/jwt-interceptor.ts>
  + To User Role Pipe - <https://github.com/buddhika85/Full-Stack-Demo-App/blob/main/Emp.Angular/src/app/pipes/user-role-enum-to-user-role-pipe.ts>
  + Auth Guard – Check if authenticated user is an expected Role user - <https://github.com/buddhika85/Full-Stack-Demo-App/blob/main/Emp.Angular/src/app/guards/auth-guard.ts>
  + Service and Observables - <https://github.com/buddhika85/Full-Stack-Demo-App/blob/main/Emp.Angular/src/app/services/user.service.ts>
  + Component with Composite Subscriber Pattern to Unsubscribe from subscriptions –

<https://github.com/buddhika85/Full-Stack-Demo-App/blob/main/Emp.Angular/src/app/components/login/login.ts>

* + Reactive forms – same example as above / login form
  + Usage of Angular Different Life Cycle Hooks - <https://github.com/buddhika85/Full-Stack-Demo-App/blob/main/Emp.Angular/src/app/components/auth/admin/manage-app-users/users-form/users-form.ts>
  + Decoding JWT Tokens using JWT-DECODE package - <https://github.com/buddhika85/Full-Stack-Demo-App/blob/main/Emp.Angular/src/app/services/jwt.token.service.ts>
  + Lazy Loading of Routes - <https://github.com/buddhika85/Full-Stack-Demo-App/blob/main/Emp.Angular/src/app/app.routes.ts>
  + Snack bar service – To display snack bar error and success messages
  + Loading Intercept – To Display ‘Loading spinner’ when there is delay in API responses
  + …etc
* **Azure Free Tier Hosting**
  + MS SQL DB is hosted on Azure SQL Database
  + ASP.NET Core Web API is hosted on an Azure App Service
  + Angular Front End is hosted on Azure Static Web App.
  + Secrets were managed using Environment Variables.
* **Git** 
  + This git repository was used to store source code.
  + Branching and Tagging was used to manage good Dev Ops practised and important milestones/releases of the project
  + Pull requests and merging was used as they will be used in CI pipeline when it is developed.
* **CI / CD Pipeline**
  + Will be developed using Azure Dev Ops Free resources.
  + Currently waiting for Azure Dev Ops to Grant Free Parallelism request to develop CI pipeline.
* **Hosted URL** 
  + API - <https://api-interview-demo-gug3dwcbecg7emb6.australiaeast-01.azurewebsites.net/swagger/index.html>
  + Angular Front End - <https://delightful-desert-009c44900-preview.eastasia.2.azurestaticapps.net/>