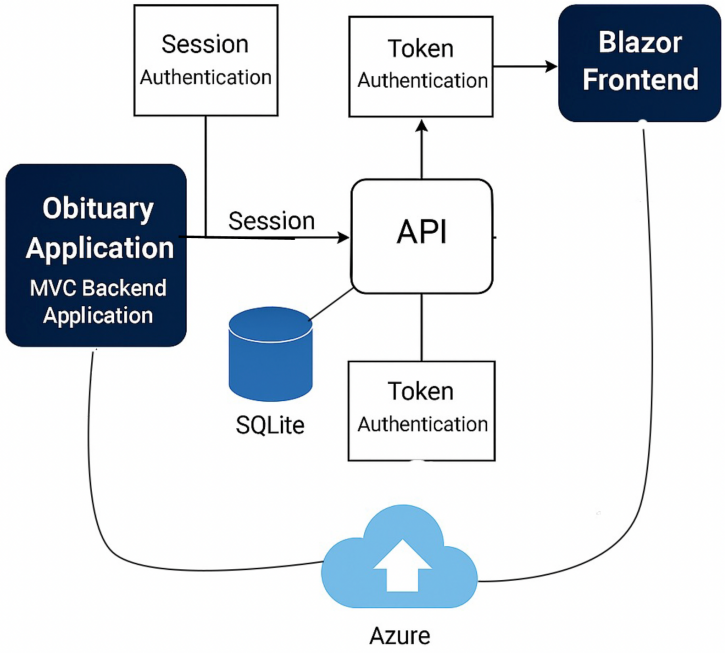
Obituary Application – Requirements Document

1. Project Overview

The goal of this project is to design and implement a web-based obituary management application using **ASP.NET MVC**. The application will allow authenticated users to create, view, edit, and delete obituary entries. The project will also expose obituary data through a *secure API* with **token-based authentication** and provide a **separate Blazor WebAssembly frontend** to consume and display this data. The final solution must be deployed to **Microsoft Azure** using DevOps pipelines triggered by **GitHub Actions**.



2. Team Structure

- Work in **groups of 2–3 students**.
- Each team member must contribute.
- Use **GitHub** for version control and collaboration.

3. Functional Requirements

3.1 User Authentication

- Implement **user registration** and **login** using ASP.NET Identity.
- Only authenticated users can create, edit, or delete obituaries.
- Public users can view obituaries without logging in.

3.2 Obituary Management

- **Create** obituary entries with:
  - Full name of the deceased

    - Date of birth
    - Date of death
    - Biography/tribute text
    - Photo upload
- **Edit** obituary entries (only by the creator or an admin).
- **Delete** obituary entries (only by the creator or an admin).
- **List** all obituaries with pagination.
- **Search** obituaries by name.

3.3 API Access

- Provide a RESTful API to:
    - Retrieve all obituaries
    - Retrieve a single obituary by ID
    - Create, update, and delete obituaries (only by the creator or an admin)
- Implement **token-based authentication** (e.g., JWT) for API endpoints.
- API must return JSON responses.

3.4 Blazor WebAssembly Frontend

Develop a **separate Blazor WebAssembly application** to consume the API.

- The Blazor app must:
    - Display a list of obituaries retrieved from the API.
    - Show obituary details on a dedicated page.
    - Allow authenticated users to create, edit, and delete obituaries via the API.
    - Implement token-based authentication for secure API calls.
    - Have a responsive and user-friendly UI.
- The Blazor app should be hosted alongside the main application in Azure or as a separate Azure App Service.

3.5 Both MVC and frontend apps must be professional looking.

4. Non-Functional Requirements

4.1 Technology Stack

Backend	ASP.NET MVC (.NET 9.0)
Database	SQLite
Authentication	ASP.NET Identity + JWT for API
Frontend	Blazor WebAssembly
Deployment	Azure App Service
CI/CD	GitHub Actions → Azure DevOps pipeline

4.2 Security

- Use common sense validation on all user inputs.
- Store sensitive data in Azure environment variables.

4.3 Performance

- API responses should be returned within **2 seconds** under normal load.

- Pagination must be implemented for large datasets.

5. Deployment Requirements

- Source code hosted in a GitHub repository.
- GitHub Actions must:
  - Build the project
  - Deploy to Azure App Service via Azure DevOps pipeline
- Deployment must be **automatic** on push to the main branch.
- Both the **ASP.NET MVC backend** and **Blazor WebAssembly frontend** must be deployed and accessible.

6. Deliverables

1. Source Code in GitHub repository.
2. API Documentation (Swagger/OpenAPI preferred).
3. Blazor WebAssembly Frontend consuming the API.

8. Additional Notes

- Use **Entity Framework Core** for database access with code first development.
- Use **Bootstrap** or similar for responsive UI in both MVC and Blazor apps.
- Commit changes frequently with meaningful commit messages.
- Each team member must be able to explain any part of the code.

Assignment 1: Backend Development – ASP.NET MVC & API

Objective

Build the core backend of the obituary management system using ASP.NET MVC and expose its functionality through a secure RESTful API.

Requirements

1. Authentication & Authorization

- Implement user registration and login using ASP.NET Identity.
- Restrict obituary creation/editing/deletion to authenticated users.
- Allow public viewing of obituaries.

2. Obituary CRUD Operations

- Create obituary entries with:
  - Full name
  - Date of birth & death
  - Biography/tribute
  - Optional photo upload
- Edit/delete entries (only by creator or admin).
- List obituaries with pagination.
- Search by name.

3. RESTful API

- Endpoints to:
  - Retrieve all obituaries
  - Retrieve obituary by ID
  - Create, update, delete obituaries (auth required)
- Secure API using JWT token-based authentication.
- Return JSON responses.

4. Database & ORM

- Use Entity Framework Core with SQLite.
- Code-first approach for model creation.

5. Deployment & CI/CD

- Host backend on Azure App Service.
- Set up GitHub Actions to trigger Azure DevOps pipeline on push to main branch.

6. Documentation

- Provide Swagger/OpenAPI documentation for the API.

7. Evaluation Criteria

Criteria	Weight
Functionality (meets requirements)	50%
Code Quality & Best Practices	15%
API Implementation & Security	15%
Deployment & CI/CD Setup	15%
Documentation	5%

Assignment 2: Frontend Development – Blazor WebAssembly

Objective

Create a responsive and user-friendly Blazor WebAssembly frontend that consumes the backend API.

Requirements

1. API Consumption

- Fetch and display obituary list from the API.
- Show detailed obituary view.
- Allow authenticated users to create, edit, and delete obituaries via API.

2. Authentication

- Implement token-based authentication for secure API calls.
- Store and manage JWT tokens securely.

3. UI/UX

- Use Bootstrap or similar for responsive design.
- Ensure intuitive navigation and clean layout.

4. Hosting & Deployment

- Host Blazor app on Azure (either alongside backend or as separate App Service).
- Integrate with CI/CD pipeline for automatic deployment.

5. Presentation

- Prepare a demo showcasing full functionality and user experience.

7. Evaluation Criteria

Criteria	Weight
Functionality (meets requirements)	50%
Code Quality & Best Practices	15%
Deployment & CI/CD Setup	15%
Documentation	5%
Presentation	15%