Rezilens MVP API & Policy Management Portal

This repository contains the Rezilens MVP API, built with .NET 8, Entity Framework Core, and SQL Server. The project is fully containerized using Docker for easy local development and includes automated migrations and seeding of initial users.

Table of Contents

Prerequisites

Before running this project, make sure you have:

- **Docker installed**
- **Docker Compose installed**
- **.NET 8 SDK (if running locally without Docker)**
- **Node.js and Angular CLI (for frontend integration)**

Setup and Running

Clone the repository

cd rezilens-mvp-api

Run with Docker Compose

The project includes a docker-compose.yml file that spins up:

- **SQL Server 2022 container (rezilens-sql)**
- **API container (rezilens-api)**

Run:

docker-compose up --build

This will:

- **Build the API container**
- **Start SQL Server**
- **Apply migrations automatically**
- **Seed initial users:**
  - **Admin: sam**
  - **User: nawal**
- **Make the API available at http://localhost:9090**

Docker Setup

Service	Container Port	Host Port
API	80	9090
SQL Server	1433	1433

Networks:
Both services are on a custom Docker network rezilens-network.

Volumes:
SQL Server data is persisted in Docker volume sql_data.

Environment Configuration

The API uses environment variables defined in docker-compose.yml:

ConnectionStrings__DefaultConnection="Server=sqlserver,1433;Database=rezilens_mvp_db;User Id=sa;Password=Str0ngP@ssw0rd!2025;TrustServerCertificate=True;"

ASPNETCORE_ENVIRONMENT=Development

JWT__Secret=" ByYM000OLlMQG6VVVp1OH7Xzyr7gHuw1qvUC5dcGt3SNM"

JWT__ValidIssuer="http://localhost:9090"

JWT__ValidAudience="http://localhost:9090"

Local Run Without Docker:
Copy these variables into appsettings.Development.json or set them in your local environment.

API Endpoints

The API endpoints are organized by feature. Key endpoints include:

Authentication

- **POST /api/Authentication/login**

Dashboard

- **GET /api/Dashboard/get-stats**

Health

- **GET /api/Health**

Policy

- **POST /api/Policy/create-policy**
- **GET /api/Policy/get-policies**
- **GET /api/Policy/get-policies-for-user**
- **GET /api/Policy/get-policy/{id}**
- **PUT /api/Policy/update-policy**
- **PUT /api/Policy/acknowledge-policy**
- **PUT /api/Policy/publish-policy**

Reports

- **GET /api/Reports/GetPoliciesWithExceptionsAndAcknowledgements**

RiskException

- **GET /api/RiskException/published-policies-for-exception**
- **POST /api/RiskException/submit-risk-exception**
- **GET /api/RiskException/all-risk-exceptions**
- **POST /api/RiskException/update-risk-exception-status**

Swagger

Once the API is running, Swagger UI is available at:
http://localhost:9090/swagger

This will show all available routes, request models, and responses.

Database

- **Database Name: rezilens_mvp_db**
- **SQL Server SA password: Str0ngP@ssw0rd!2025**
- **Migrations: Automatically applied on container start using Entity Framework Core.**
- **Initial Users Seeded:**
    - **Admin: sam**
    - **User: nawal**

Note: When containers are up just execute the rezilens_mvp_db_proc.sql file for adding all stored procedures in the database.

Frontend Integration

To connect an Angular frontend:

1. **Open your Angular project.**

2. **Edit src/environments/environment.ts (and environment.prod.ts) to point to the API:**

```
export const environment = {
  production: false,
  apiUrl: 'http://localhost:9090' // API base URL
};
```

3. **Install dependencies and serve the frontend:**

```
npm install
ng serve
```

Frontend will run on http://localhost:4200 and communicate with API on port 9090.

Sharing and Deployment

Locally:

1. **Share the repository with Dockerfile and docker-compose.yml.**
2. **Run:**

```
docker-compose up --build
```

- **API: http://localhost:9090**
- **SQL Server: localhost:1433**

For Public Hosting:

- **Deploy API to cloud (Azure App Service, AWS EC2, Docker hosting).**
- **Update environment variables for production.**
- **Share only API URL and authentication credentials.**

Notes

- **Data Protection:**
  Encryption keys are stored in /root/.aspnet/DataProtection-Keys. Configure persistent storage for production.

- **SQL Server Logs:**
  Informational messages from Service Broker / Mirroring do not require action.
- **Migrations & Seeding:**
  The API automatically creates tables and seeds initial users on first run.