# Technologies used

1. Dapper
2. Microsoft dotnet core 3.1.3
3. Microsoft.Extensions.DependencyInjection for dependency injection
4. MSTest for the test projects
5. Moq for mocking in tests
6. Microsoft.Extensions.Options for splitting the configuration in convenient settings classes. Works nicely with the DI
7. Microsoft.AspNetCore.Cryptography for handling password hashing
8. System.Security.Cryptography.Algorithms for password encryption
9. Microsoft.AspNetCore.Authentication.JwtBearer for handling tokens

# Design patterns used

1. Unit of work + Repository. The actual repository implementation is injected in the entry point, so the project is minimally dependent on it. There is not references to it from the Services project.
2. Service – stateless instance containing various operations
3. Mapper (data adapter) – converting data from one type into another
4. Observer – well … far from pure implementation but the way the DapperRepository’s commit and rollback are done is with callback … so in a sense it is kind of like a very specific observer (that registers for events in the constructor of the observable object)
5. Singleton – Techically we are not implementing it but setting it up. And it’s not a singleton for the whole application but for a http request. Due to the way asynchronous web requests are handled by the web server we are dependent on the DI provider to manage the life cycle of the SessionRepository, which we need to be alive for the whole request and the same instance to be injected in all constructors that need it (for the duration of the request).
6. Façade – the CryptoService class is like a façade to the Microsoft.AspNetCore.Cryptography and System.Security.Cryptography.Algorithms libraries.

# Sequence diagram

A picture containing outdoor, table, photo, screen

Description automatically generated

# Overview diagram

## Of projects

A close up of text on a white background

Description automatically generated

## A high level view

A screenshot of a cell phone

Description automatically generated