CS 319 - Object Oriented Software Engineering

Instructor: Eray Tüzün, TA: Muhammad Umair Ahmed

BilHealth

Design Report

Iteration 1

Mehmet Alper Çetin
21902324

Vedat Eren Arıcan
22002643

Uygar Onat Erol
21901908

Recep Uysal
21803637

Efe Erkan
21902248

April 6, 2022

Contents

1 Introduction

1.1 Purpose of the System

1.2 Design Goals

2 High-Level Software Architecture

2.1 Subsystem Decomposition

2.2 Hardware-Software Mapping

The project does not have any extraordinary hardware requirements. As it is a web application, it requires the users to have a device using which they can connect to the web server of the application over the internet. The main target devices for a supported user experience are desktop computers, laptops, and mobile phones.

In order to run the project the intended way, the host system's hardware must support stable containerization through Docker. It is possible to reconfigure certain entrypoints of the project to support bare-metal execution too, at the cost of system replicability and practicality. Similar to most other web services, the project is officially supported to run on a Linux environment, which may be of consideration when choosing hardware.

2.3 Persistent Data Management

The project makes use of PostgreSQL as its persistent data storage solution. The project does not have specific requirements that directly justify the use of PostgreSQL, however, this database system is well rounded enough to save the development process from any potential drawbacks. The `psql` command line tool is easy to use and supports our workflow in implementing database migrations. Also, the Docker images for the database are robust and simple, which has allowed the project to be supported by a containerized database.

On top of the actual database, the project uses EF Core as an object relational mapper (ORM), which is Microsoft's official library for data persistence. We use the recommended *code-first* approach of database management, which means our PostgreSQL tables are constructed by EF Core with respect to our entity definitions in C# code. The EF Core toolchain generates migrations, which it also converts to SQL scripts, using which a database can be configured to host the project in moments.

2.4 Access Control and Security

The project is secured by a typical authentication and authorization flow, supported by our identity provider of choice, ASP.NET Core Identity. Authentication is conducted over HTTPS with a user's username and password. The controller endpoints of the project are exposed to users within specific role types, which constitutes our authorization process. The authentication is persisted on the user's device in the form of browser cookies, enabling user session capability.

The project is mostly exclusive to authenticated users, meaning that there are not many substantial actions a guest user can take. Moreover, since users are divided across roughly

5 roles, most HTTP endpoints concerning management of the system are forbidden to access by non-staff users.

Lastly, user details are stored in a database with their passwords having been hashed by the *bcrypt* algorithm, which is among the top choices in secure password storage techniques at this time.

2.5 Boundary Conditions

3 Low-Level Design

3.1 Object Design Trade-Offs

3.2 Final Object Design

3.3 Packages

Below is a list of our namespaces with brief descriptions. Note that, to preserve the anonimity of the report, we've replaced the root namespace with a placeholder "`Project`".

- `Project.Utility`:
  This namespace contains the utility functions which were able to be decoupled from the project at large. In doing so, it can be used reliably across other namespaces.

- `Project.Utility.Enum`:
  This namespace contains the *enum* and *enum class* definitions that are used in many parts of the project.

- `Project.Services`:
  This namespace contains the main business logic interface through which boundary and entity systems are connected.

- `Project.Services.Users`:
  This namespace carries the same responsibility as its parent, but specifically for user-related entities.

- `Project.Model`:
  This namespace contains the entities that are persisted into a database. The main units of data are located here.

- `Project.Model.Identity`:
  This namespace contains entities that are specifically user information holding objects.

- `Project.Model.Dto`:
  This namespace contains *record* objects that are used to transfer data to and from the front-facing controllers. It is also used for some internal communication of unpersisted data. In other words, these are temporary representations of the actual data entities to be persisted.

- `Project.Data`:
  This namespace contains the database context built from EF Core's repository-like architecture.

- `Project.Controllers`:
  This namespace contains the front-facing API endpoints through which external communication with clients take place.

3.3.1 External Packages

- `Microsoft.AspNetCore`:
  This package is the framework supporting the entire web application.

- `Microsoft.EntityFrameworkCore.Design`:
  This package allows the EF Core migration tool to work on the project.

- `Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore`:
  This package allows debugging EF Core migrations.

- `npgsql.EntityFrameworkCore.PostgreSQL`:
  This package is the database driver providing EF Core its interoperation with PostgreSQL.

- `Microsoft.AspNetCore.Identity.EntityFrameworkCore`:
  This package provides authentication and authorization of users through integration with EF Core.

- `BCrypt.Net-Next`:
  This package provides the *bcrypt* algorithm which is currently among the best in securely hashing user passwords.

- `Microsoft.AspNetCore.Mvc.NewtonsoftJson`:
  This package is the most popular JSON library for .NET, used as the default solution in most cases.

- `Microsoft.AspNetCore.SpaProxy`:
  This package allows the .NET build process to launch and proxy with the client application.

3.4 Class Diagrams of Layers

3.4.1 Data Access Layer

3.4.2 Business Logic Layer

3.4.3 User Interface Layer

4 Glossary