Software Architecture Report

This document aims to describe to a potentially new member of the development team what the software project does at the time of redaction. It doesn't have to be a monolith document, the information can be stored in READ.MEs.

The intended audience for this document is a technical one. Different sections may target different stakeholders.

The document must be:

- Dated
- Versioned – in correlation with the project's versioning (you could use a commit ID as a form of versioning, if non other is implemented)

Contents:

1. What the purpose of the software project is
    a. This may be a summary based on the planning documentation.
    b. Fulfilled capabilities – what can the project do at this point, and what is to be done until project fulfillment.

- For the web application:
    - Everyone who has been to the gym so far has encountered some problems that can make many beginners feel overwhelmed: what exercises to do, how to perform them correctly, and in what order should you do them to optimize your workouts. Our application comes to the aid of all those who go to the gym and try to solve these problems.
    - Users can **post their workout programs** that they use, so that all other users have access to them and can try them out. The ranking system will ensure that the most appreciated programs by users, which have the highest ranking, will be displayed first, to help those looking for a good program to find it more easily. Additionally, **each program has its page** where **users can leave comments**. Also, in the application, **you can monitor your gym and weight progress**, having the option to **record each workout**: how many sets of each exercise you did, with what weights you worked, etc. An algorithm will analyze the data entered and notify you if you have surpassed your record in a certain exercise, taking into account the number of repetitions and the weight you worked with. A chart will also be generated showing your progress to see if you are starting to plateau and it's time to try something different or if you are on the right track.

- For the mobile application, we aimed to create an extension of the web one.
    - What practically happens in the application?
        - Upon installation, on the first opening of the app, the user is greeted with an Onboarding page, which serves to introduce the user to the app, like a virtual "Welcome". Then, the user encounters the Login page.

- Since the mobile application is an extension of the web one, we wanted it to be accessible only to users who already have an account (created through the web application), as the purpose of this app is simply for users to track their progress and workout history.
- After logging in, the user is directed to the page displaying all their workouts. If they tap on one, they are taken to the workout details and then to its history.
- What does the workout history entail?
  - It includes all repetitions, sets, everything the user has done since creating an account in the app until the present.
- Why did we choose to create this extension on mobile?
  - We wanted to offer users the convenience of viewing their progress from anywhere, including the gym, through a mobile app. In general, it's much more convenient, when we're not at home or don't have a laptop available, to have access to such mobile apps, where all the information we need to complete our activity is displayed.

2. Guides on how to

   In order to run the project locally you have to (FE + BE):

   - Clone to project locally
   - Go to the root directory in terminal
   - Run the following commands :
     - In order to start frontend:
     - npm install -> load node.js packages and create node_modules
     - npm run dev -> create a local socket for running your application on loopback
     - In order to start backend:
     - docker-compose up -d
     - docker exec -it <mssqlcontainerId> bash
     - /opt/mssql-tools/bin/sqlcmd -S localhost -U SA -P 'Follyestepisica@12' -Q 'CREATE DATABASE InginerieDB'
     - dotnet /opt/sqlpackage/sqlpackage.dll /tsn:localhost /tu:SA /tp:'Follyestepisica@12' /A:Import /tdn:InginerieDB /sf:/opt/downloads/df.bacpac
     - Now you got yourself a webapp with the backend on docker and frontend on local
   - To create an artifact of the project:
     - npm run build -> creates a bundle with our react project (js files compressed into a single chunk of data)
     - npm run start -> serve that bundle to our browser and starting the project on a PRODUCTION environment
   - To start tunneling session using ngrok for making mobile app communication with our backend(without docker environment):
     - ngrok http 80 -> creating a connection with Hypertext Transfer Protocol(http); in this way we can create as many connections with our web apps as we like (our REST api in this case)

    - ngrok http 7231 -> creating a connection with our .NET REST api (the resulting URI is served for our mobile app manually)
    - ngrok http 8082 -> creating a connection with our .NET REST api with docker container running on localhost, port 8082

- Contribution Guide(Design patterns):
    - Repository Pattern(backend): One of the most popular patterns used because it relies on SOLID principles, and if built correctly, it is both easy to use and reusable.
    - Provider Pattern(frontend): When it comes down to React, the provider pattern is implemented in the React context API. React is well known for supporting a unilateral downward data flow from a parent component to its children by default. (Chakra UI / Redux / React Router Dom 6)

- For the mobile application, things were much simpler. The project was run locally, straightforwardly, by pressing the "Command + B" key combination, or by clicking the Run button provided by Xcode. In Xcode, the option to build the application is the same as the option to run it.
    - Currently, the application isn't hosted somewhere, because there is only one place provided by Apple for such purposes, in the TestFlight application and we can't host it there, unless we want to add it to Apple Store for real. On TestFlight, developers can send invitations to other individuals, allowing them to download the applications hosted by the developer.

a. Run the project locally.
b. Build the project.
c. Deploy the project (either locally, or how is it hosted, where it is the case)
d. Contribution guide
    - ■ Patterns used in your application

3. Application entry points
    a. Data sources
    b. Data inputs
        - ■ Register form page
        - ■ Insert exercise
        - ■ Insert split
        - ■ Add progress
        - ■ Add weight
    c. Configuration files
        - ■ appconfig.json (WorkoutBuddy.Service\Backend.WebApp\appsettings.json)
        - ■ tsconfig.json (workout-buddy.client\tsconfig.json)

4. Deployment plan
    a. Where is the application deployed?
        - ■ Backend is deployed in docker containers using docker compose
    b. How the CI/CD pipeline works.

        - It is developed using actions from github, using 2 yaml workflows, 1 for checking the backend if it builds and also testing the frontend app if it builds and also running unit tests

5. Description of the QA process
    a. Test suites – what do they test.

- React Testing Library with Jest - Unit tests for making sure that pages remain consistent regarding initial data and snapshots
- For the mobile application, we used Xcode's unit tests to ensure that everything works correctly. Unit tests are essential for validating the correct behavior of the code and for detecting any errors or regressions during development.
    - The MVVM (Model-View-ViewModel) architecture is used to separate the logic components of the application and to improve the testability and maintainability of the code. In this architecture, ViewModels are responsible for managing data and the logic of displaying the user interface. Unit tests for ViewModels ensure that they work correctly in different scenarios and that the interaction with data and services is handled properly.
    - Since all the functionalities of the application are implemented in ViewModels according to the MVVM architecture, we created unit tests only for them. This way, we can verify the behavior and correctness of the ViewModels in isolation, without needing to interact with other components of the application.
    - It is important to have a comprehensive set of unit tests for ViewModels, covering different scenarios and functionalities, to ensure that our application is robust and stable during development and in production.

6. External dependencies included in the project
    a. APIs used - Our build-in REST API
    b. Libraries

- Frontend
    - Chakra UI
        - Chakra UI is a simple, modular and accessible component library that gives you the building blocks you need to build your React applications.
        - Used for enhancing our UI/UX with our personal custom theme and light/dark theming
    - Redux Toolkit
        - It helps you write applications that behave consistently, run in different environments (client, server, and native), and are easy to test.
        - Used as a global storage for our client application. Easy to store data used in multiple parts of the application, easy to debug and to use.
    - React Router 6
        - React Router enables "client side routing".

        - Client side routing allows your app to update the URL from a link click without making another request for another document from the server.
        - Used as a navigation tool for our application; Making it more responsive and interactive

- Backend
    - Newtonsoft.Json
        - Newtonsoft.Json is a popular third-party library for working with JSON (JavaScript Object Notation) data in .NET applications.
        - In the UserSplitController, we employed the Newtonsoft.Json library to seamlessly serialize and deserialize JSON data. Specifically, it was utilized for converting UserExerciseModel objects in the AddProgress action and handling JSON content in the GetDates action, enhancing data exchange capabilities in our .NET application
    - AutoMapper
        - AutoMapper is an open-source library for .NET that simplifies the process of mapping data between different types of objects.
        - In the UserSplitProfile class, we made use of AutoMapper to smooth out the process of mapping data between entities and models in the Backend.BusinessLogic.Implementation.UserSplitColection namespace
    - Fluent Validation
        - FluentValidation is an open-source .NET library that provides a fluent interface for defining and executing validation rules in a straightforward manner.
        - The AddProgressValidator class is using FluentValidation to define validation rules for the UserWorkoutModel class, which is typically used for adding progress in the context of user workouts
    - Entity Framework Core (EF Core)
        - Purpose: A lightweight, cross-platform ORM framework for .NET applications.
        - Key Features: Cross-platform support, Code-First and Database-First approaches, multiple database provider support, and designed for performance.
    - SQL Server
        - Purpose: SQL Server is a relational database management system (RDBMS) developed by Microsoft. It's used for storing and retrieving data in a structured format.
        - Key Features:

            - Support for SQL (Structured Query Language) for interacting with databases.
            - ACID (Atomicity, Consistency, Isolation, Durability) properties to ensure data integrity.
            - Scalability, security features, and support for stored procedures, triggers, and views.
    - Entity Framework Design
        - Purpose: Entity Framework Design is a part of Entity Framework Core, specifically designed for managing database migrations and scaffolding. It provides tools to create and apply database schema changes based on changes in your data model.
        - Key Features:
            - dotnet ef migrations: Commands for creating and applying database migrations.
            - dotnet ef database update: Applies pending migrations to the database.
            - Scaffolding tools to generate code and files based on an existing database.
    - Entity Framework Tools
        - Purpose: Tools and utilities for working with Entity Framework, including migrations, scaffolding, and design-time operations.
- If an item is already a project, reuse it, by using links. If an item is already present, but isn't up to date, update it and version it. It is not expected present in the p that you would fulfill everything that can be done for such a report. Do the most of what is possible, and acknowledge what can be done better.