

Documentation

# Project Overview

The Lightweight Gym is planning to create a web application that is a comprehensive fitness tool that offers a range of functionalities to help gym members and non-members achieve their fitness goals. The users can download the application for free and sign up, or sign in if they have registered, to get into the application and see the application's features and provided gym services.

# Design Description

The app has three functionalities in terms of user management, which are:

1. User registration to the web application
2. Log in to the web application
3. Password resetting/recovery

There are seven features that users can use. These are:

1. Viewing the list of activities and the schedule.
2. Viewing the diet plan
   1. Bulking
   2. Cutting
   3. Maintaining
3. Viewing workout plans
   1. Upper body
      1. Arms
      2. Backs
      3. Grip
   2. Lower body
      1. Calves
      2. Thighs
4. Viewing sauna schedule
5. Upgrade membership if your current one is either bronze or silver

**This feature only works for members.**

1. Downgrade membership if your current one is either silver or gold

**This feature only works for members**.

1. Unsubscribe from the gym
   1. Ask for feedbacks

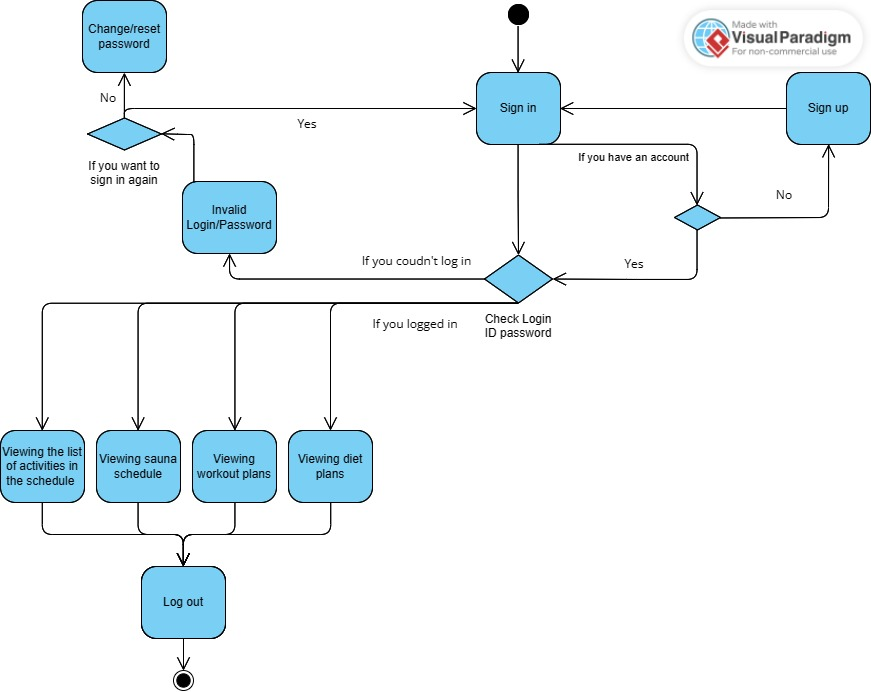
**This feature only works for members.**

# User Stories

1. As a new user, I would like to register for the Lightweight gym web application.
2. As a user, I would like to log in to my profile.
3. As a user, I would like to change my password or recover my password if I forget my password.
4. As a user, I would like to view the list of activities and the schedule of those activities.
5. As a user, I would like to view the diet plan for bulking, cutting, and maintaining my body.
6. As a user, I would like to view the workout plans for upper body and lower body.
7. As a user, I would like to view the sauna schedule.
8. As a user, I would like to upgrade membership if my current one is either bronze or silver.
9. As a user, I would like to downgrade membership if my current one is either silver or gold.
10. As a user, I would like to unsubscribe from the gym and share my feedback.

# Diagram

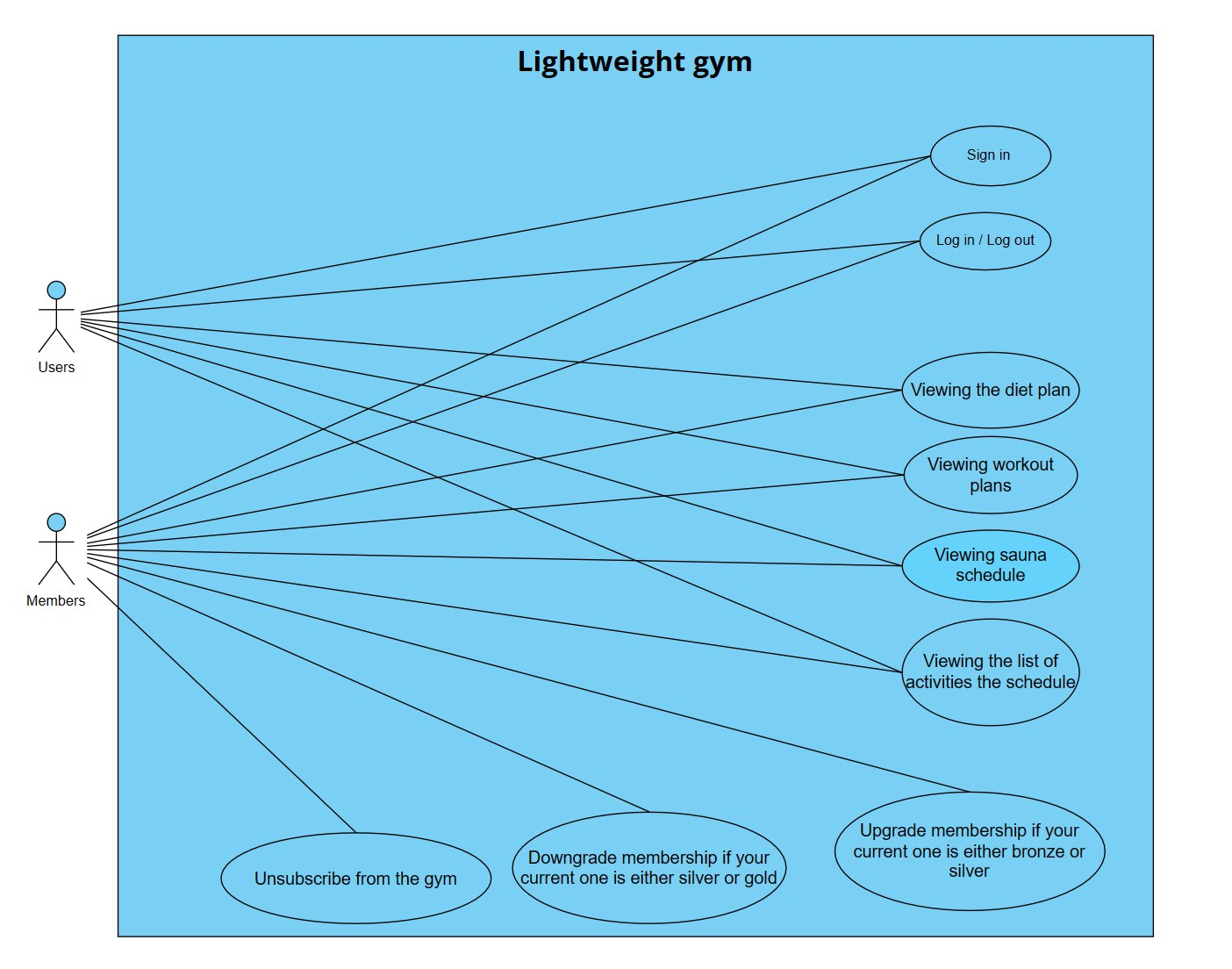
Activity diagram



This activity diagram illustrates the sequential flow of activities and processes within a system. The user initiates the process by accessing the "Sign In" page. In the case of new users, they have the option to first complete the "Sign Up" process before proceeding to the "Sign In" page. In the event that users encounter difficulties signing in, they can utilize the password recovery functionality to reset their password and attempt signing in once again.

Once successfully signed in, users gain access to various website features, including the activity schedule, sauna schedule, workout plans, and diet plans. Alternatively, users have the option to log out of the system, thereby concluding their session.

Use Case Diagram



We have distinguished our consumers into two distinct roles: Users and Members. Members refer to those consumers who have subscribed to Lightweight Gym, while Users encompass the general visitors of our website. At this initial stage of development, the primary distinction between these roles lies in the Members' ability to modify their subscription type and unsubscribe from the gym, subsequently becoming Users.

Our system comprises nine key features, six of which are accessible to both roles. These features include the options to sign up and sign in, view the diet plan, access workout plans, review the sauna schedule, and explore the activity schedule.

# Test Cases

* Use Case: Sign Up

Success Conditions:

* + User provides valid and unique registration details.
  + User's information is successfully stored in the database.

Failure Conditions:

* + User enters incomplete registration details.
  + User enters invalid password guidelines
  + User enters different password (password mismatch)
  + User attempts to sign up with an already existing email.
* Use Case: Sign In

Success Conditions:

* + User provides valid credentials (email and password).
  + User's credentials match an existing user in the database.
  + User is successfully authenticated and granted access.

Failure Conditions:

* + User enters incorrect or invalid credentials.
  + User's credentials do not match any existing user in the database.
* Use Case: See List of Activities

Success Conditions:

* + User is authenticated and authorized to access the list of activities.
  + List of activities is successfully fetched from the server.
  + Activities are displayed to the user.

Failure Conditions:

* + User is not authenticated and unauthorized to access the list of activities.
* Use Case: Upgrade Member

Success Conditions:

* + User is authenticated and authorized.
  + Member's member type is successfully upgraded.
  + Updated member information is stored in the database.

Failure Conditions:

* + User is not authenticated or authorized.
* Use Case: Downgrade Member Type

Success Conditions:

* + User is authenticated and authorized.
  + User selects a member to downgrade their member type.
  + Member's member type is successfully downgraded.
  + Updated member information is stored in the database.

Failure Conditions:

* + User is not authenticated or authorized.
  + Selected member does not has a membership.
* Use case:The diet plan

Success conditions:

* User is authenticated and authorized to access the list of activities.
* The diet plan is successfully fetched from the server.
* The diet plan is displayed to the user.

Failure conditions:

* User is not authenticated and unauthorized to access the diet plan.
* Use case:The workout plans

Success conditions:

* User is authenticated and authorized to access the workout plans.
* The diet plan is successfully fetched from the server.
* The diet plan is displayed to the user.

Failure conditions:

* User is not authenticated and unauthorized to access the workout plan.
* Use case:Password recovery
* Success conditions:
* User provides an existing email
* User has a mail for password recovery.

Failure conditions:

* User's credentials do not match any existing user in the database.
* Use case:Sauna schedule

Success conditions:

* User is authenticated and authorized to access the workout plans.
* The diet plan is successfully fetched from the server.
* The diet plan is displayed to the user.

Failure conditions:

* User is not authenticated and unauthorized to access the workout plan.
* Use case:Unsubscribe from the gym

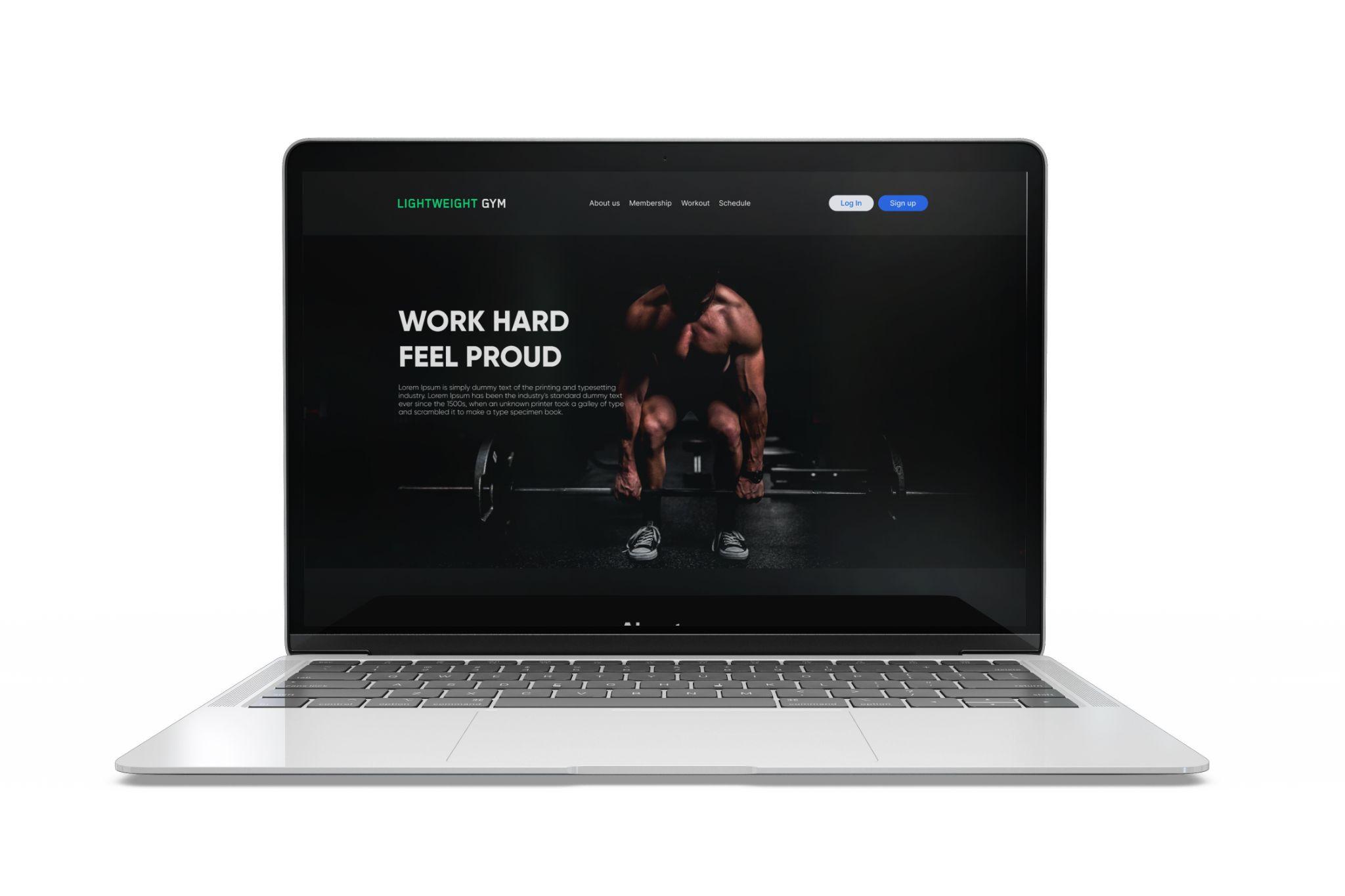
Success conditions:

* User is authenticated and authorized to unsubscribe from the gym.
* User unsubscribes from the gym.
* User is successfully unsubscribed from the gym
* Updated information is stored in the database

Failure conditions:

* User isn't authenticated and unauthorized to unsubscribe from the gym.
* User isn't subscribed to the gym

Web Design



# Frontend Technology

## HTML

**HTML (Hypertext Markup Language)** is the standard markup language for creating web pages and web applications. It provides the structure and defines the content of a web page. HTML consists of a series of elements, which are represented by tags, that define different parts of a webpage's structure and content.

CSS

**CSS (Cascading Style Sheets)** is a style sheet language used to describe the presentation and formatting of HTML (and XML) documents. It provides a way to control the appearance of web pages, including elements like layout, colors, fonts, and other visual aspects.

CSS works by applying styles to HTML elements using selectors. Selectors target specific elements or groups of elements on a webpage, and the associated styles define how those elements should be presented.

## JavaScript

**JavaScript (JS)** is a versatile scripting language used for adding interactivity and dynamic functionality to web pages. It allows developers to manipulate page content, respond to user actions, and interact with web APIs, making web pages more engaging and interactive.

# Backend Technology

## ASP .NET

**ASP.NET** is a web development framework developed by Microsoft. It is a part of the larger.NET framework and provides a programming model, tools, and libraries for building web applications and services. ASP.NET allows developers to create dynamic and interactive websites, web applications, and web services using various programming languages, such as C#, which will be used for developing the system.

ASP.NET offers different approaches for building web applications, and the Web API is one of them. The reason for choosing APIs is to promote a clear separation of concerns between the backend and the frontend. The backend can focus on providing data and functionality through the API endpoints, while the frontend can consume and present the data in the desired format for the user interface.

## REST API

For fetching data to the frontend, we are using the REST API architectural style for designing networked applications. REST APIs are widely used for building web services, enabling systems to interact and exchange data in a flexible and standardized manner.

By following the principles of REST, APIs can be designed to be stateless, scalable, and easily consumed by different clients, including web browsers, mobile applications, and other servers for scalability.

# Data Design

Data design provides efficient data exchange between clients and servers. They can be optimized for performance and scalability, allowing you to handle a large number of requests and efficiently manage resources.

This section outlines the design of the database management system (DBMS) files associated with the system. Changes to the logical data model may occur due to software requirements and data traffic.

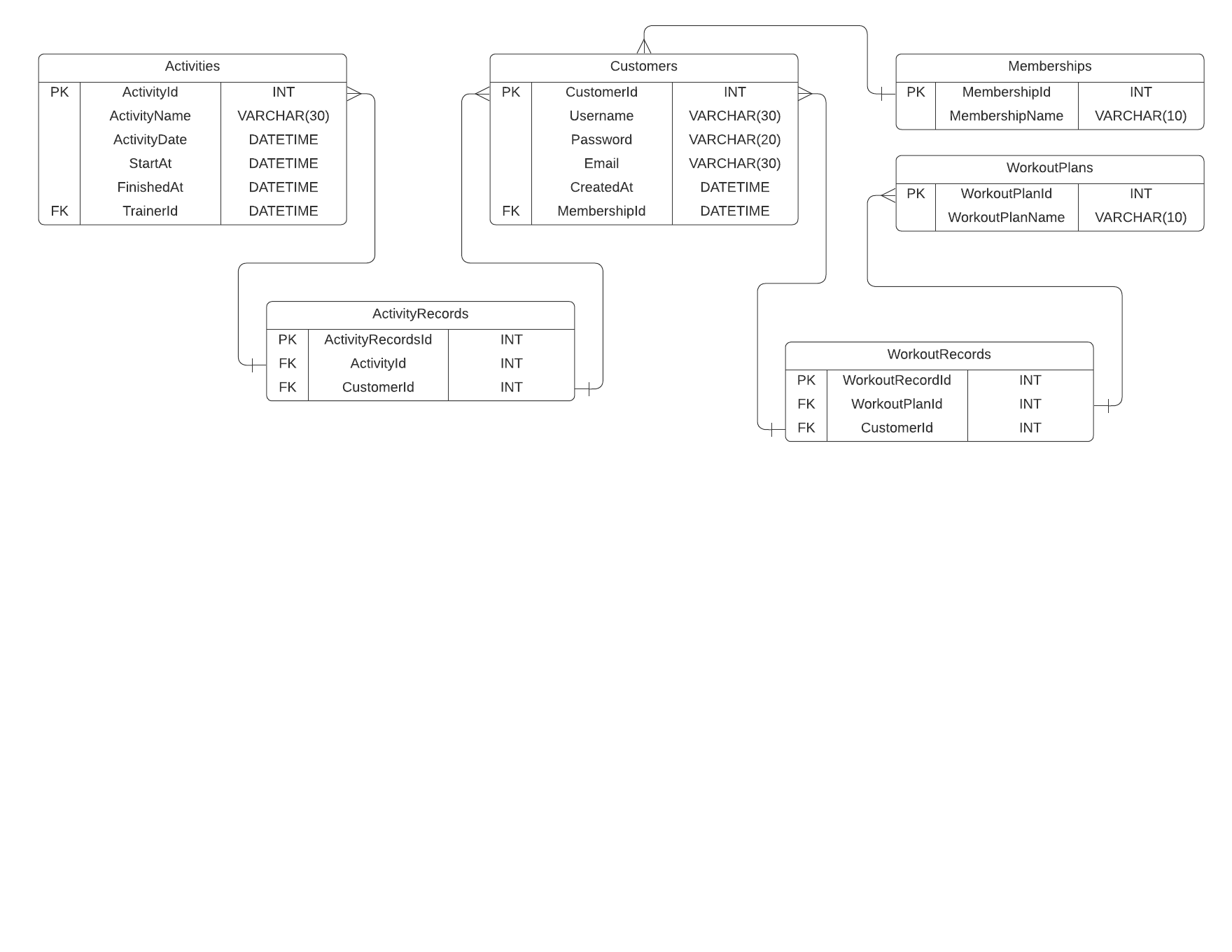
## SQLite

**SQLite** is a popular, lightweight, embedded relational database management system (RDBMS). It is a software library that provides a self-contained, serverless, zero-configuration, and transactional SQL database engine. Unlike traditional client-server database systems, SQLite is embedded directly into the application that uses it rather than running as a separate process.

SQLite has been chosen since ASP.NET includes built-in support for working with SQLite databases. The library can be downloaded from NuGet Packages under the name of Microsoft. EntityFrameworkCore.Sqlite and using the AddDbContext and UseSqlite methods.

SQLite is right for early development to create a database locally on the project to test every functionality that requires getting data from the database, such as user authentication and personal data.

## ERD



The ERD above describes relationships between entities. **Customers** entity has one foreign key, which is **Memberships** entity as the dictionary for membership types (bronze, silver, gold). There are two tables of features, those are **Activities** and **WorkoutPlans** tables. These tables have many-to-many relationships with the **Customers** table, and use records tables as mediators of both relationships with particular names.

# Prototype

## Database and API

<https://github.com/brianabeltimothy/LightweightGym>

## Frontend

<https://github.com/zhvnvrys/lightweight_gym_frontend/tree/main/Lightweight_Gym_Frontend>