System Design Document for TrainingTracker

1. Introduction

General info. What is this? What does it describe?

This document will provide documentation on the System Design for the application “TrainingTracker”, an application aimed at enhancing and simplifying a user’s experience at the gym.

2. System architecture

The most overall, top level description of the system. Which (how many) machines are involved? What are the system components, and what are they responsible for? Show the dependency between the different system components. If there are more computing entities (machines) involved: show dow they communicate. Describe the high level overall flow of some use stories. Describe how to start and stop the system. Any general principles in application? Flow, creations, . . .

2.1 Subsystem decomposition

Describe in this section each identified system component (that you have implemented).

2.2 ‘First Component’

What is this component responsible for and what does it do. Divide the component into subsystems (packages) and describe their responsibilities. Draw an UML package diagram for the top level. Describe the interface and dependencies between the packages. Try to identify abstraction layers. Think about concurrency issues.

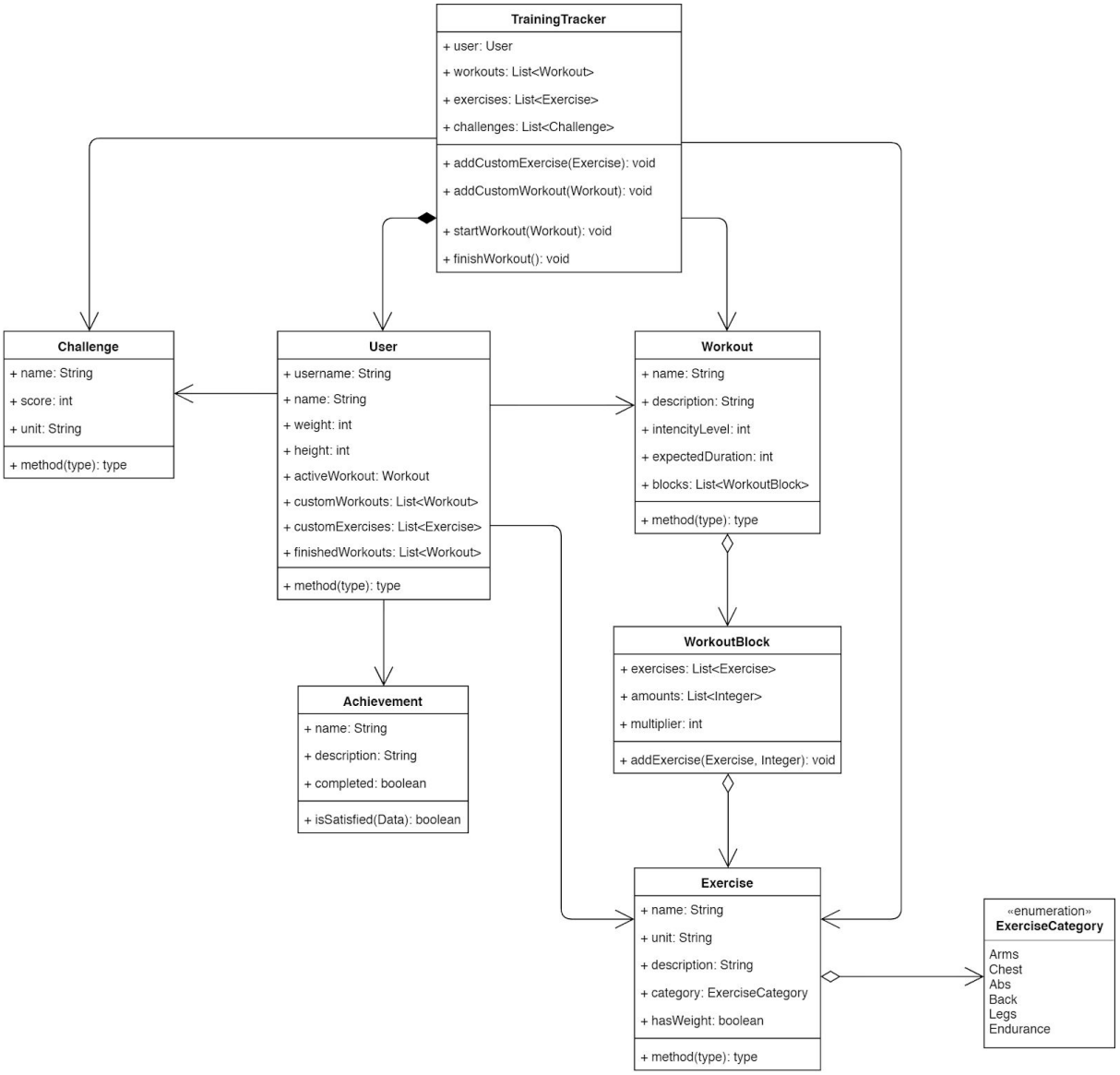
<u>If your application is a standalone then:</u>
• Describe how MVC is implemented
• Describe your design model (which should be in one package and build on the domain model)
• Give a class diagram for the design model.
<u>otherwise:</u>
• MVC and domain model described at System Architecture Diagrams

• Dependencies (STAN or similar)
• UML sequence diagrams for flow.
Quality
• List of tests (or description where to find the test)
• Quality tool reports, like PMD (known issues listed here) NOTE: Each Java, XML, etc. file should have a header comment: Author, responsibility, used by ..., uses ..., etc.

2.3 'Next component…'

As above, and continue for all components.

2.n 'Model package'



2.n ‘View package’

The “View package” contains the visual part of the application, the GUI. It is responsible for displaying the data in the model to the user, while also letting the user modify the data through interaction with the interface.

The main class of this package is the MainActivity, which is responsible for all communication with external packages. When another class in the package needs to access data from the model it does so via the MainActivity.

The view in this application is designed in such a way that each separate screen of the interface is represented by a Fragment. A Fragment can be displayed as a part of the screen or cover the entire screen. It holds the information of how it is to be displayed, and what components it is made of. Each fragment also has a corresponding .xml file which defines the visual elements of the fragment, these elements are then used and manipulated in the fragments class. Each fragment also has a reference to the MainActivity, which it can use to populate its components with the data from the model. When the app switches from one fragment to another it does so either through the bottom navigation bar (and its listener in the MainActivity class), or through the reference to the MainActivity in the current Fragment.

2.n ‘ViewModel package’

The “ViewModel Package” is responsible for handling the communication between the View and the Model. Whenever the MainActivity needs to access or modify data in the Model, it uses the viewModel object in the MainActivity class. The ViewModel, in turn, holds an instantiation of the TrainingTracker from the Model package in order to access data and logic in the Model.

2.n ‘Service package’

The “Service Package” contains various functionality for importing pre-created exercises and workouts, and functionality for serializing and de-serializing application data, in order to save information between app-sessions.

3. Persistent data management

How does the application store data (handle resources, icons, images, audio, ...). When? How? URLs, paths, ... data formats... naming..

The premade exercises and workouts are stored in xml format in the res\raw directory, and are loaded during startup by their respective service classes. The user specific data (user variables, custom exercises/workouts, challenges and achievements) are stored using Json,

utilizing Google’s implementation of Json, gson. Icons used in the application are stored in Vector format in the res\drawable directory, native to the android application.

4. Access control and security

Different roles using the application (admin, user, ...)? How is this handled?

The application currently does not have different roles for usage. Each user has the same options as every other user. When starting the application for the first time each user has the same application base, and through app usage their individual user data will differ, as more individual exercises and workouts are created .

5 References