WorkoutAnywhere Final Report

Version 1.0

May 12th, 2023

Alecsander Gonzalez

Liam Nelson

Table of contents

1. Introduction

1.1 Project Overview

WorkoutAnywhere is a smartphone application that was developed for the Android platform. The application takes in user information such as the user's height, weight, gender, and available equipment and generates a workout of their choice with their desired intensity. The application then walks the user through the workout, providing a rest timer and descriptions of the exercises upon request. Our intention with this application was to help lower the barrier of entry for new gym goers by providing a fast, effective way to generate an exercise routine and provide intermediate gym goers with a tool they can use when they are working out in a new gym.

2. Software Functions

2.1 Project Requirements

Below are the functional requirements of the system:

Req#	Requirement	Priority	Date Rvwd	Reviewed / Approved
WA_UI_01	The system shall collect basic user information (height, weight, gender) on its initial startup.	1	2/22/22	Alecsander Gonzalez, Liam Nelson
WA_UI_02	The system shall allow the user to choose the type and intensity of their workout.	1	2/22/22	Alecsander Gonzalez, Liam Nelson
WA_UI_03	The system shall allow the user to replace any workout from their plan.	2	2/22/22	Alecsander Gonzalez, Liam Nelson
WA_UI_04	The system shall allow the user to track their time between sets with a timer.	2	2/22/22	Alecsander Gonzalez, Liam Nelson
WA_UI_05	The system shall give the user more information about a workout on request.	2	2/22/22	Alecsander Gonzalez, Liam Nelson
WA_UI_06	The system shall accept a user's ultimate workout goal (lose fat/gain muscle) and will track their progress.	3	2/22/22	Alecsander Gonzalez, Liam Nelson
WA_BE_01	The system shall generate a workout plan based on the information from WA_UI_01 and WA_UI_02.	1	2/22/22	Alecsander Gonzalez, Liam Nelson
WA_BE_02	The system shall generate alternate exercises based on the user's choice from WA_UI_03.	2	2/22/22	Alecsander Gonzalez, Liam Nelson

In addition to those requirements, there were three design goals that we had:

3.3 Performance

The system shall provide a typical user an effective workout in less than 5 minutes.

3.4 Safety

The system shall recommend challenging workouts if the user desires, but will not recommend greater than 10% of the weight that the user has used in the previous week.

3.5 Reliability

The system shall include at least one paragraph in the "More Info?" section that informs the user how to perform the exercise.

3. Architecture

3.1 High-Level Architecture

We originally envisioned using the Model-View-Controller approach to build our system in our Architecture document. But as the coding phase went on, it became clear that our design was more appropriate for the Pipe-and-Filter method. As a result, based on this methodology, we have produced a new diagram that depicts our high-level architecture.

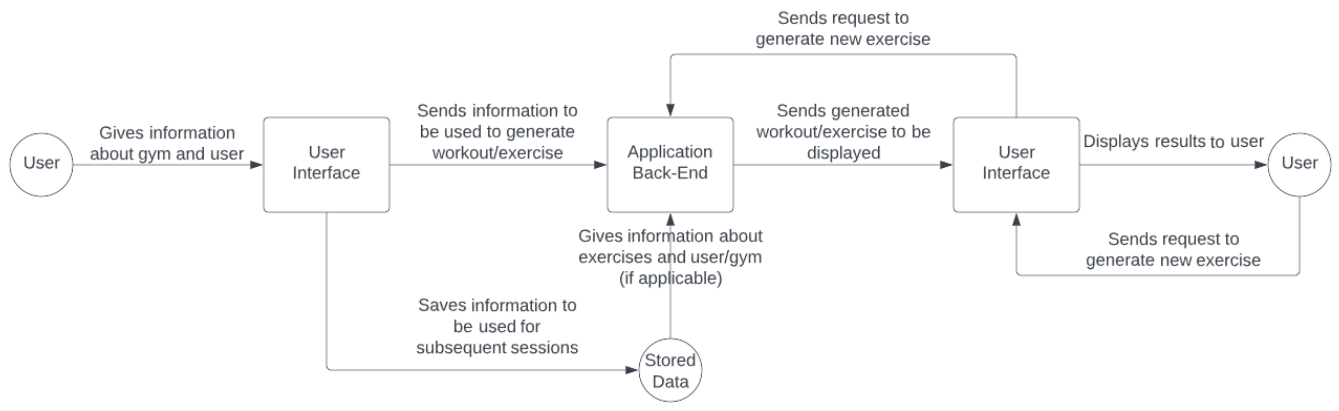
Sends request to generate new exercise
User
Gives information about gym and user
User Interface
Sends information to be used to generate workout/exercise
Application Back-End
Sends generated workout/exercise to be displayed
User Interface
Displays results to user
User
Sends request to generate new exercise
Gives information about exercises and user/gym (if applicable)
Saves information to be used for subsequent sessions
Stored Data

Fig 1. *High-Level Architecture of the System*

Our architecture consists of two pumps, namely the user and the stored data, and two filters, the user interface and the application back-end. The stored data is responsible for providing all the information about the exercises as well as any saved user data. On the other hand, the user provides the filter settings such as the type of workout they want to perform. This information is then passed through the user interface and into the application back-end where a workout is generated based on the provided parameters. The resulting workout is then sent back through the user interface to be displayed to the user. Additionally, the user can request for the same filters to be applied again to generate a new exercise.

3.2 User Interface

Purpose

The purpose of this module is to provide screens in which users can input data and receive information from the backend and stored data.

Rationale

This module is created to handle all requests related to the GUI. This includes displaying the appropriate screens, receiving user input to be added to stored data, and displaying input from both stored data and the application backend.

Required Interface

```
Void createUserProfile(float w, int feet, int inches,
boolean g, int lowerRange, int upperRange);
UserProfile getInstance();
public LinkedList generateWorkout(int intensity,
String type);
public String loadUserProfile(String fr1) throws
FileNotFoundException;
public String saveUserProfile(String fr1) throws
IOException;
public boolean isValidNumber(String input);
Excercise makeExcercise(String type);
String printExcercise();
String getDescription();
public static InstanceManager getInstance();
```

Provided Interface

This module does not provide an interface.

3.3 Stored Data

3.3.1 User Profile

Purpose

The purpose of this module is to save information about the user's physical profile (height, weight, gender) and their workout environment (what machines and weights are available) to be used in calculating weights, rep ranges, and available exercises for workouts.

Rationale

This module is created to collect information that can be stored in the user's profile, thus saving valuable time when it comes to inputting the necessary information to generate a workout.

Required Interface

```
public String loadUserProfile(String fr1) throws
FileNotFoundException;
public String saveUserProfile(String fr1) throws
IOException;
```

Provided Interface

```
Void updateGym(boolean kbAvailable, boolean
bbAvailable, boolean dbAvailable, int lowerRange, int
upperRange, boolean legExtensionMachine, boolean
latPulldownMachine, boolean smithMachine, boolean
legCurlMachine, boolean rowMachine, boolean
legPressMachine, boolean cableMachine, boolean
shoulderPressMachine, boolean lateralRaiseMachine,
boolean machineFly)
```

Description:
Updates the available equipment in the user's profile

Parameters:
kbAvailable: A boolean for the presence of kettlebells.
bbAvailable: A boolean for the presence of barbells.

dbAvailable: A boolean for the presence of dumbbells.
rowMachine: A boolean for the presence of a row machine.
legExtensionMachine: A boolean for the presence of a leg extension machine.
latPulldownMachine: A boolean for the presence of a lat pulldown machine.
smithMachine: A boolean for the presence of a smith machine.
legCurlMachine: A boolean for the presence of a leg curl machine.
legPressMachine: A boolean for the presence of a leg press machine.
cableMachine: A boolean for the presence of a cable machine.
shoulderPressMachine: A boolean for the presence of a shoulder press machine.
lateralRaiseMachine: A boolean for the presence of a lateral raise machine.
machineFly: A boolean for the presence of a chest flye machine.

```
Void createUserProfile(float w, int feet, int inches,
boolean g, int lowerRange, int upperRange)
```

Description:
Creates a user profile object.
Parameters:
w: The user's weight.
feet: The user's height in feet.
Inches: The user's height in inches.
G: The user's gender.
lowerRange: The lowest available weight in the gym
upperRange: The highest available weight in the gym

Exceptions:
If a user's profile has already been created, this function updates the existing profile.

```
UserProfile getInstance()
```

Description:
A singleton method is used to call the userprofile. This is needed in order to transfer information about the user into other classes.
Returns:
The current active user profile.

```
int getHeight()
```

Description:
Gets the height of the user.
Returns:
The height of the user in inches.

```
Float getWeight()
```

Description:
Gets the weight of the user.
Returns:
The weight of the user.

```
Boolean getGender()
```

Description:
Gets the gender of the user.

Returns:
The gender of the user.

```
Void updateWeight(float weight)
```

Description:
Updates the weight of the user.
Parameters:
weight: Weight to be updated.

```
Int calculateLoad(double groupFactor)
```

Description:
Calculates the load (weight) for the user to do during an exercise.
Parameters:
groupFactor: A variable that adjusts the load based on the muscle group being targeted.
Returns:
A load for the exercise.

Module ADT

```
UserProfile{
     private static UserProfile userProfile;
    //weight in pounds
    private float weight;
    //height in inches
    private int height;
    //true = man, false = woman
    private boolean gender;

    //lower and upper range of available weights
    public int lowerRange;
    public int upperRange;

    //all variables needed for gym equipment
    public boolean kbAvailable;
    public boolean bbAvailable;
    public boolean dbAvailable;
    public boolean rowMachine;
    public boolean legExtensionMachine;
    public boolean latPulldownMachine;
    public boolean smithMachine;
    public boolean legCurlMachine;
    public boolean legPressMachine;
    public boolean cableMachine;
    public boolean shoulderPressMachine;
    public boolean lateralRaiseMachine;
    public boolean machineFly;
}
```

3.3.2 Exercise Data

Purpose

The purpose of this module is to create a standardized representation of each workout that may become potentially generated for users.

Rationale

This module is created to make sure the program can search through and generate exercises according to a user's preferences.

Required Interface

```
public String[] requestFile(String fileName);
```

Provided Interface

This module does not have any provided interfaces.

3.4 Application Back-End

Purpose

The purpose of this module is to process the data from the Stored Data modules and to send processed data to the User Interface module.

Rationale

This module is created to perform the necessary calculations based on the user data and generate a workout plan. Since it needs all of the data from the previous sections, it makes sense to keep it separate rather than merge with one of them.

3.4.1 Exercise Generation

Purpose

The purpose of this module is to process the data from the Stored Data modules Into individual exercises.

Rationale

This module is created to construct exercise objects for each muscle group and pass them to the Workout Generation module.

Required Interface

```
public String[] requestFile(String fileName;
int calculateLoad(double groupFactor);
```

Provided Interface

```
ExcerciseGenerator()
```

Description:
A default constructor that allows the Workout Generation module to access the exercise factory.

```
Excercise makeExcercise(String type)
```

Description:
Generates exercises based on the requested type.
Parameters:
type: A reference to the muscle group of the exercise requested. (chest, bicep, etc.)

Returns:
An exercise object of the requested type.

```
String printExcercise()
```

Description:
Provides a string of the exercise object in a format that is readable.
Returns:
A string of the exercise object that includes the name, type, load, and set/rep range.

```
String getDescription()
```

Description:
Provides the description of the exercise.
Returns:
A string that contains information about how to perform the exercise.

Module ADT

```
public class Exercise {
    //name of exercise
    String name;
     //muscle group targeted (Chest), (Shoulder),
     etc...
    String descriptor;
    //description of exercise
    String description;
    String prerequisites;
     //stores the lower-bound of the rep range,
getting the upper bound by adding 2
    int repRange;
    int setRange;
    //weight to be lifted
    int load;
}
```

3.4.2 Workout Generation

Purpose

The purpose of this module is to simplify the process of generating multiple exercises. We can pass in a type of workout (push, pull, etc.) and generate a number of exercises that fit that workout plan.

Rationale

This module is created to construct a workout that is composed of five exercise objects.

Required Interface

```
Excercise makeExcercise(String type);
```

Provided Interface

```
public LinkedList generateWorkout(int intensity,
String type)
```

Description:
Provides a list of exercises that meet the user’s preferences and available equipment.
Parameters:
intensity: The intensity of the exercises. This is passed to the exercise constructor.
type: The type of workout the user would like to do.
Returns:
A linked list consisting of exercises that meet the user’s preferences and available equipment.

3.4.3 File Reader/Writer

Purpose

The purpose of this module is to allow the system to remember information between user sessions. It saves information about the user and information about available gym equipment.

Rationale

This module was created to improve the overall user experience and make workout generation faster. Without this module, the user will need to input all of their information everytime they want to use the application.

Required Interface

```
public static InstanceManager getInstance();
```

Provided Interface

```
public String saveUserProfile(String fr1) throws
IOException
```

Description:
Saves information about the session into the device's memory.
Parameters:
fr1: The file path constructed in local memory.
Returns:
A string that verifies the process was successful. Used for development purposes.

```
public String loadUserProfile(String fr1) throws
FileNotFoundException
```

Description:
Loads information about the previous session into the current instance.
Parameters:
fr1: The file path constructed in local memory.
Returns:
A string that verifies the process was successful. Used for development purposes.

```
public String[] requestFile(String fileName)
```

Description:
Loads a file that contains all of the exercises of a given type. Calls the labelMaker() function to populate the labels[] and descriptions[] with the names and descriptions of all the exercises.
Parameters:
fileName: The name of the file that has the exercises.
Returns:
An array that has the name of the exercise and the description of how to perform that exercise.

Module ADT

```
class FileRead extends Activity
{
    LinkedList<String> labels = new
    LinkedList<String>();
    LinkedList<String> descriptions = new
    LinkedList<String>();
    LinkedList<String> loggedFiles = new
    LinkedList<String>();
    String lastVisited;
    String toReturn = "n/a";
}
```

3.4.4 Instance Manager

Purpose

This module is created in order to facilitate the transfer of the instance-based information collected within the user interface module to the back-end and vice-versa.

Rationale

This module was required in order to make saving user information possible and makes the process of sending information between “screens” in the user interface simpler.

Required Interface

This module does not have a required interface.

Provided Interface

```
public static InstanceManager getInstance()
```

Description:
Provides a method to access the instance manager from anywhere.
Returns:
The current and only instance of the instance manager.

`public void changeContext(Context c)`

Description:
Changes the stored context variable. Note: Context is an object associated with an Android application. It stores information about the current active page.
Parameters:
c: The context of a current page

`public boolean isValidNumber(String input)`

Description:
Converts the input string into a float which is checked to see if it is in the valid range.
Parameters:
input: The float number to be checked, passed in as a string.
Returns:
A boolean value: true if the input is valid and false otherwise.

Module ADT

```
class InstanceManager
{
    private static InstanceManager instanceManager;
    public int difficulty;
    public String style;
    private Context c;
    public LinkedList<Excercise> workout;
    public Excercise e;
    public int example;
}
```

4. Implementation

In the upcoming section, we will discuss the technologies employed during the development of our system, how we have fulfilled the tasks outlined in our requirements, and the degree of consistency our final product has with our founding documents. Additionally, the final subsection of this section will include links to a video demo and our source code.

4.1 Technologies Used

We used Android Studio as our IDE and virtual testing environment. We chose this software because it has built-in tools to develop the user interface and it has the capability to create a virtual mobile device that we could use to test our code.

We used Brandcrowd.com to generate our app's logo.

4.2 Implementation Tasks

When the user opens the application, the Application Back-End module accesses the Stored Data module to pull information about the user. If this is a first time user, the User Interface module collects the necessary information about the user and their preferences for their workout. Once all of the information is collected, the User Interface gives a workout style to the Workout Generator, which calls the Exercise factory with the muscle group that's needed, which calls the File Reader/Writer to pull that exercise data from memory.

4.3 Consistency

Our project remained mostly consistent with our requirements document and has changed since our architecture document has been submitted. The changes from both documents are outlined in the subsections below:

4.3.1 Requirements

Our system slightly deviates from our original requirements document. We were successful in our implementation of all of the priority one and two members of our functional requirements but were unable to complete priority three requirements, the workout goal tracking. Our team pitched this feature to the project owner who agreed that it would be a good idea but was not necessary. Ultimately, we met all of the criteria that was listed in the project owner's proposal.

In addition to the functional requirements, we created three other requirements that we wanted to meet. These additional requirements were Performance, Safety, and Reliability (additional details in Section 2.1). We were able to deliver on our performance and reliability requirements but we did not meet the safety requirement that we had set out to achieve. The safety requirement was based upon the workout goal tracking feature which we did not have enough time to implement.

4.3.2 Architecture

Our system architecture has been greatly expanded upon since we worked on our initial documentation. We had an idea of how our system would be architectured in the beginning but, as we worked on it, more details became clear and we had to add more in order to make the system that we had wanted. This document includes the updated architecture and interfaces.

4.4 System Availability

Link to our video demo: https://www.youtube.com/watch?v=gVBiir4Dfeg
Link to our Github: https://github.com/LiamNelson9696/WorkoutAnywhere

5. Project Management

Below is a table that shows the work each team member did for our project.

Team Member	**Task**	**Notes**
Liam Nelson	Developed Stored Data Module	Collected all information about each exercise and created their names and descriptions.
Liam Nelson	Developed Back-End Application Module	Developed all tools and interfaces used by the back-end.
Alecsander Gonzalez	Developed User Interface Module	Developed all tools and interfaces used by the user

		interface.
Alecsander Gonzalez/Liam Nelson	Integrated the Modules	Implemented and created new interfaces to make the modules work together nicely.
Alecsander Gonzalez/Liam Nelson	Bug Fixes	
Alecsander Gonzalez/Liam Nelson	Created Documentation of the System and Requirements	All documents were worked on collaboratively.
Alecsander Gonzalez/Liam Nelson	Created Presentation and Presentation Materials	Alex provided all of the video demos. Liam provided the diagrams and slides.

We encountered some problems during the development of our system, mainly due to our unfamiliarity with Android Studio. We struggled to get information to be saved across screens to be used later on in the program. This was mainly due to our unfamiliarity with Android Studio and how it works compared to a regular computer program. This was a significant roadblock as our User Interface requires multiple screens to collect user information. Alex developed a technique in which we would use “bundles” to pipe the information to the next screen. Essentially every screen would collect information, pack it into the bundle, and transfer it to the next screen which would do the same thing. This was later improved by our Instance Manager which allowed any information to be passed between the User Interface and the Application Back-End.

Creating the File Reader/Writer component was also a struggle. Android Studio has a particular way of storing and accessing files which had initially made getting those files difficult. A context variable needed to be created for the active screen so that we could get to the Exercise Data component in the “raw” folder. However, since there are no active screens in the back-end code, it would be impossible to read the files without moving all of the File Reader/Writer code to the front end. However, by using the Instance Manager component, we were able to pass a context variable to the File Reader/Writer and resolve this issue.

6. Conclusion

WorkoutAnywhere meets all of the basic requirements set by the project owner, Nathan Raras. Our application is able to create workouts for a user based on their weight, workout intensity, and desired workout type. We would have liked to include the fitness goal tracking feature but difficulties during development led to this feature being scrapped.

Our team is satisfied with the final outcome of our application. Our team was of a reduced size when compared to the other groups but it had its advantages. We enjoyed the flexibility and spontaneity of meetings which allowed us to move on issues more quickly. There were some drawbacks, however, as the team size reduced the available man hours and made the larger segments of the system more time consuming to complete.

Despite that, our team enjoyed working on this project. We had no prior experience or knowledge of mobile app development and we've learned how to develop a mobile application over the course of the semester.

7. References

1. WorkoutAnywhere Requirements Specification Version 1.0
2. WorkoutAnywhere Architectural Design Version 1.0