|  |  |
| --- | --- |
| COMP6000  Workout Assistant with Generative AI Features | |
|  | |
|  | Team members: Nikola Kolev, Ethan Teather, Juan Salazar, Manuel Montero, and Misha Koroliovas |
| Supervised by: Sergey Ovchinnik |

# 

# Abstract

This project presents the development of a comprehensive fitness application designed to provide a personalised and intuitive workout experience. We drew inspiration from established fitness apps, such as Apple Health, Samsung Health, and Nike Training Club. We aimed to incorporate essential features, similar to theirs while introducing innovative elements. The novelty features, which are how we are differentiating from other fitness apps and are our selling points, are anonymous mode which does not require internet connection, AI Workout Generation and Enhance Workout with AI functionality. Other functionalities include manual workout creation, searching and filtering through already created workouts, step tracking, and detailed workout history.

Key to our development process was the integration of various APIs and technologies. We used Android Studio to code the app, incorporating Java and XML. TiDB Cloud served as our online database solution, while we used SQLite in the local database. The OpenAI ChatGPT API enabled AI-driven workout generation and enhancement, setting our app apart in the fitness app landscape.

Throughout development, a strong emphasis was placed on user experience, security, anonymity, and functionality. We implemented encryption methods to secure passwords, optimised navigation with fragments and activities, and integrated Google Fonts and Canva Color Wheel for a cohesive visual identity. The result is a user-friendly app that empowers individuals to tailor their fitness journey.

# Introduction

In the age of mobile technology, numerous workout apps exist on the market, catering to all categories of users, from new to experienced, from old to young. However, most of them fail to accommodate multiple categories of users. Some apps are extremely alien to those new to working out, demanding specific information that beginners may not comprehend. In other cases, the app can almost be too easy to understand, being great for new users, but failing to keep experienced people enticed afterwards, offering workouts that are now below their level.

In today’s rapidly advancing world, AI is quickly becoming one of the most powerful tools, and we believe it could be a major part of our project. It can also be the key to bridging the issue of captivating both new and experienced users.

However, the uprising of AI has caused numerous concerns about users' safety and identity online. As a result, people are fearful of the ethics of AI.

With that knowledge, we set out to create an app that could cater to both new and experienced users using the power of Artificial Intelligence, without losing out on essential features in other workout apps such as simply searching for workouts. The app will also prioritise user privacy, only using data the user provides and allowing for a full incognito mode. In incognito users are disconnected from the AI.

Our app will have 3 main pages: Home, Workouts and History:

* The home page is the landing page once a user logs in. It will show a greeting, along with several statistics related to their workout activity. This includes total completed workouts, favourite workout and total time spent in minutes. The home page also includes a step counter and some quick links to directly access popular workouts. Finally, we have a fully separate Home page dedicated to anonymous mode.
* In the Workout page users have 3 ways of receiving a workout. Firstly, the user can search for a workout from the database, filtering the results provided. Secondly, they can create their own workout, manually adding exercises from the database. Finally, AI can be utilised to generate a workout, by providing it with some data. Whilst each path is different, they all lead to the Workout Hub, where the user can either begin a workout or enhance it with AI, to modify the workout to their standards.
* Lastly, the History page is where users can see all workouts they have performed in chronological order. By default, only 4 workouts are shown to reduce the amount of data, but the user can click the “view all” button to see everything. The user can also choose to delete their history and erase their statistics.

# Background

A variety of established applications inspired our project, each of which contributed to some feature or design element that we decided to recreate. It also uses multiple APIs and refers to documentation, to implement certain functionalities.

From the beginning, Apple Health and Apple Fitness inspired the overall design aesthetic and functionality. Similarly, Samsung Health guided our approach to integrating a step counter feature, enhancing the app's usability for fitness tracking. These three are some of the most widely used workout apps globally, so we were sure that there was extensive research behind their UI/UX decisions. Following their steps in certain decisions was a safe choice we made, to ensure the app is user-friendly and intuitive.

The design of our History page owes its structure to the sleek layouts of Nike Run Club and Nike Training Club applications, offering users a visually appealing and organised view of their past workouts.

Regarding the development of functionalities, we read a lot of documentation and utilised APIs. Google Email (Gmail) Documentation, for instance, was used for implementing the "Forgot password" feature, ensuring users' access to secure verification codes via email. For data security, we turned to SHA256 Documentation, implementing encryption to safeguard sensitive user information within the app (passwords). Another documentation, that proved invaluable, was Android Studio API Documentation. It informed us about the project architecture, including the strategic use of Fragments, activities, and multithreading for optimal app performance. It was crucial for improving the app’s “smoothness”.

For the databases, we referred to TiDB and SQLite Documentation, since these were the online and local database solutions we decided to implement. Leveraging TiDB Documentation, we connected to the Online DB, enabling data retrieval from and storage on the cloud. Additionally, SQLite Documentation allowed us to create and manage a Local DB, facilitating offline functionality. That is used by default when one logs in anonymously.

The integration of OpenAI's ChatGPT API, next, allowed users to generate and enhance workouts using intelligent AI (the GPT 3.5-turbo model). These are cornerstone functionalities and are a selling point of our app. To connect to the ChatGPT API, we had to create a developer account in OpenAI and fund it.

Finally, practical tools such as Google Fonts, specifically the Manrope font, which became the app's global typography choice, helped us enhance readability and visual consistency. Canva’s colour wheel and Paletton.com helped us select vibrant and harmonious colours. For instance, Canva suggested the colour hex code for pastel green, red, and yellow that we used to showcase the difficulty of an exercise or a workout.

In essence, our app stands on the shoulders of these resources, making use of proven design elements and working with cutting-edge functionalities to create a user-friendly and AI-enhanced fitness companion.

# Aims

We aimed to create a Workout Assistant App with AI functionality and privacy in mind. It must also have the same basic functionality as anything else on the market.

The main selling point of the app is that it is heavily enhanced by AI, allowing for a more flexible approach to conventional health apps. One way is by generating dynamic workouts. These workouts would be constructed through carefully engineered prompts that would return the best results with the data provided by the user. The app also has the capability to enhance already existing workouts, allowing users to further adjust workouts to their needs.

Another key factor is data safety and anonymity within the app. There would be 2 modes available to the user: “Signed-in” and “Anonymous”. Signed-in would provide full access to the app, including all features, whilst using only the data the user gave. This data could be removed immediately upon request. Anonymous, instead, restricts the user to only a few features of the app, with no access to the AI feature. However, nothing is tracked within this mode, allowing the user to browse freely through all the available workouts in the local offline database.

# Workflow

Upon opening the app, users encounter a motivational quote that changes with each visit. They are presented with two buttons: "Login" and "Anonymous Browsing." There is also a clickable text below the login button that leads to the registration page.

Choosing "Anonymous Browsing" takes users to a separate Home page tailored for anonymous use. This mode provides access to core functionalities such as a "Search" button for local workouts and a "Logout" option returning users to the start page after confirmation. Previously, after clicking “Anonymous Browsing” we would lead people to the same pages that registered users could access and would restrict the buttons. However, too many of the buttons would need to be restricted in anonymous mode, so we developed two fully separate Home pages.

However, if a user is willing to exchange some privacy, they can log in regularly. To register, one must provide an email and a password. Additional details like name, date of birth, height, weight, and health conditions are optional. This data, used primarily for personalised AI-generated workouts, remains confidential. After registration, users proceed to the login page, where they enter their credentials.

We’ve also thought about the case where one has forgotten their password. In such a situation, they should click the “Forgot your password?” text, and then enter their email. After that, we send them a verification code, which they use to create a new password.

After logging in, users are led to the ActivityContainer, which holds the Home, Workout, and History fragments.

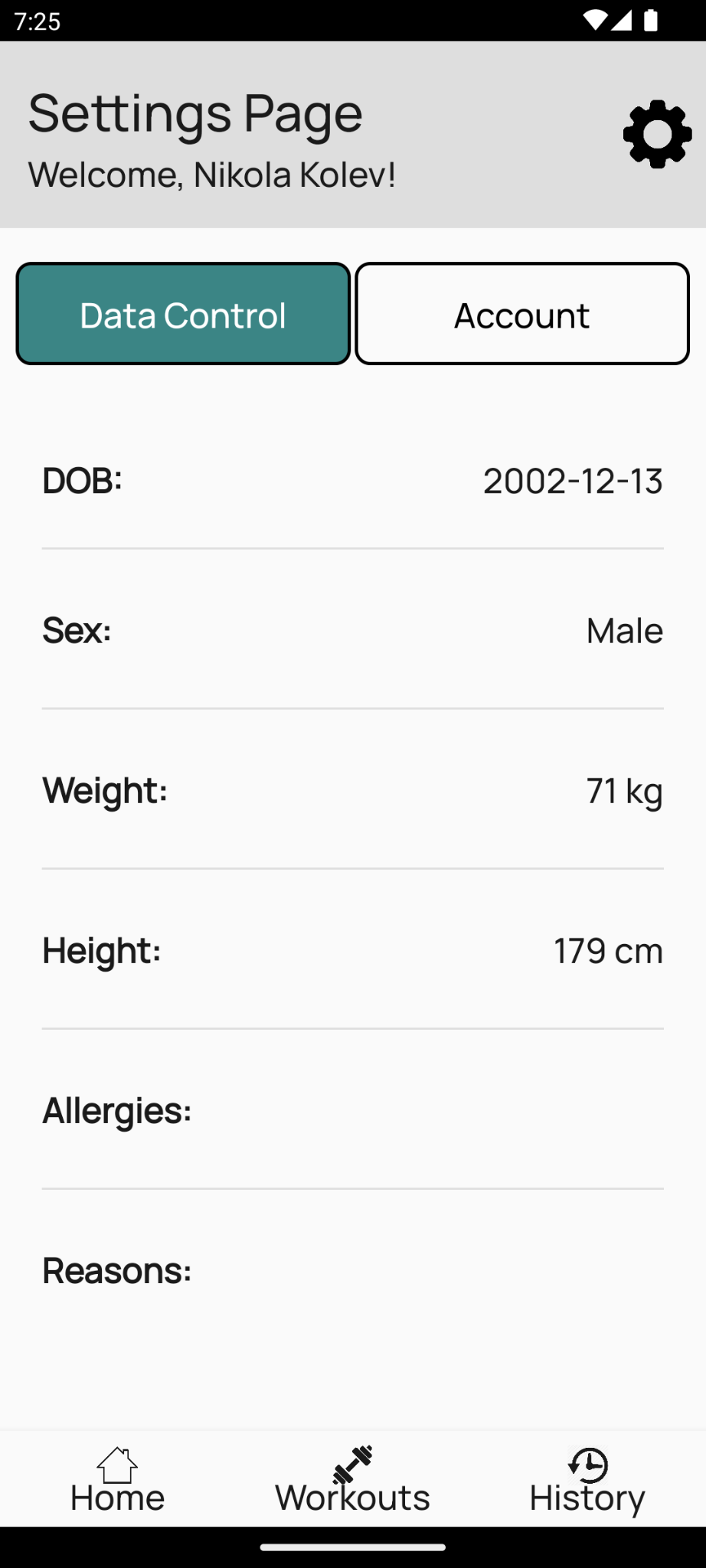
The Home fragment includes workout statistics, a step counter linked to the phone's pedometer sensor, and "Quick Selection" buttons for easy access to relevant workouts.

The Workout page contains two fragments: WorkoutManual and WorkoutAi2. WorkoutManual provides options to search for pre-created workouts or create custom ones using the online exercise table. WorkoutAi2 presents AI-generated personalised workout options.

The History page showcases the workouts a user has done in chronological order, starting with the most recent.

*See figures Wf.1, Wf.2 and Wf.3 in the appendix below for screenshots of the three pages as of the 30th of March 2024.*

The activity container also holds the Settings fragment, accessed via a top-right cog icon. The settings fragment holds two fragments itself. One is “Data Control” which allows users to adjust personal data like height. The other is “Account” which lets users change their passwords, log out, or delete their account.



Users typically navigate to the Workout page after logging in, to choose from three paths:

* Path A (Select): Visit "Search Workout" -> Filter (optional) -> Select workout.

*See figures Wf.4, Wf.5, Wf.6 and Wf.7.*

* Path B (Create): Enter details in "Create Workout" -> Choose exercises -> Confirm creation.

*See figures Wf.8, Wf.9, Wf.10 and Wf.11.*

* Path C (AI): Input preferences in AI Workout Gen -> Generate workout -> Select.

*See figures Wf.12, Wf.13 and Wf.14.*

All paths lead to the "Workout Hub" displaying workout information. In the “Workout Hub” users find "Enhance (AI)" to tailor workouts further and "Begin" to start their chosen workout.

During a workout, each exercise is displayed individually, detailing sets, reps, proper form, and sometimes images. Users navigate through exercises with "Next," and upon completion, select "Finish" to add the workout to their history.

All the above is showcased in the video that we submit along with the corpus of the report. You can also find a detailed diagram in the appendix, *figure Wf.15*.

# Tools

Throughout the development process, we relied on several tools. First, we used Android Studio as our IDE. It is a powerful software that allows us to build our app using Java and XML. All of us knew these languages before the project, which made collaboration smoother.

We also made use of TiDB Cloud for its valuable online SQL capabilities and amazing pricing plan. TiDB proved to be better than expected because it enabled us to run SQL queries from the web interface directly on the live database. This feature was incredibly important for our project, as it allowed us to test and refine our SQL queries before integrating them into the app's code. This meant we could see immediate results and ensure the app's functionality was on point. It probably has saved us hours in debugging.

Finally, we, of course, also incorporated the ChatGPT API into our app to connect with the AI GPT Model. This integration allowed us to use an AI without coding it from scratch ourselves. More information regarding our implementation of the ChatGPT API is available further in the report.

# Databases Overview

When creating the databases for storing information, we decided on having 2 databases that would operate differently. Both databases would have Workouts, Exercises and WorkoutExercisePairs. The online database would also have Users and History. The online database was the first to be created and was hosted by TiDB Cloud, due to its ease of use and great value. The offline database was created later inside the app itself since it would be operated locally within the app.

Exercise Tables hold the details of every exercise and have the following attributes: ExerciseID (primary key), ExerciseName, Description, Illustration, TargetMuscleGroup, Equipment, Difficulty, Sets, Reps and Time. It was decided to have a unique ID for the Exercise Table, given that none of the attributes are unique. Sets, Reps and Time are special since not all of them will have data at once, and depend on whether the exercise is time-based (where time will have data, but reps will not) or rep-based (the other way around).

Workouts are like the Exercise Table, in the attributes that it has, including WorkoutID (primary key), WorkoutName, Workout Duration, TargetMuscleGroup, Equipment and Difficulty. Once again, an ID was needed as a primary key.

ExerciseWorkoutPairs is used to create a relation between the Workouts and Exercises pairs. The relationship between Workouts and Exercises to ExerciseWorkoutPairs is one-to-many, as a workout can have many exercises, and an exercise can appear in many workouts. There are only 2 attributes for this table: ExerciseID and WorkoutID. The primary key is a composite key of both IDs, and both are foreign keys, relating to their respective tables.

“Users” stores details about all registered users. It contains Email, PreferredName, Password and other optional attributes that users provide upon registration.

# Workspace Directory Organisation

The workspace for our app is organised neatly within the "app\src\main" directory. Within the "Java\com\firstapp\group10app" path, we find the heart of our app – all the Java classes that it is built upon are there.

8.1. DB directory

In the "DB" directory, we handle everything related to databases. It has two sub-folders:

The "LocalDb" area focuses on operations with SQLite, our local database. It contains classes like LocalDbHelper and LocalDbConnection, which manage creating, upgrading, and querying the local database. These classes help us work smoothly with the three tables of the SQLite database: exercises, workouts, and pairs of exercises and workouts.

The "OnlineDb" section mirrors the functions of the "LocalDb" but with an online TiDB database connection. Here, we have OnlineDbConnection and OnlineDbHelper, along with DataChecker and DataFormatter classes. The first two connect to and query the database, while the latter ensures data is in the right format, preventing SQL errors.

Outside these two sub-directories, we have some common classes used by both databases. The key class is DatabaseManager, which acts as a central hub for database interactions. It is also a Java singleton class, ensuring there is access control to the local or online database connection. DatabaseManager is supposed to be the only class that accesses OnlineDbConnection and LocalDbConnection. It decides which database to query and holds instances of the connections, saving time by avoiding the creation of new connections each time. The optimization was quite substantial.

8.2. Other directory

Moving to the "Other" section, one finds utility classes with various functions used across the entire app. For example, ItemVisualiser helps display exercises and workout details, Encryption safeguards user passwords, and ExceptionHandler handles uncaught errors. Other important classes in the “Other” directory are Index, which holds commonly accessed variables, JsonToDb, a crucial class within our app, which we’ve designed to heavily rely on JSONs, and Session, which has private variables and getters for all important data of the current session. Within the latter one are the getSignedIn() and logout() methods, which are central to our app. Each of the files in the “Other” directory is accessed from different parts of the application, which is what groups them in the “Other” folder.

8.3. Pages directory

Another extremely important folder is the “Pages” directory, where all the code for the related activity layouts and fragment XMLs lay. These files handle button logic and page content. They also communicate with the files in the “ChatGPT”, “DB”, and “Other” directories to present relevant information to the users. The Java classes in the pages directory utilise asynchronous code execution whenever possible (especially when querying a database), to ensure the app is snappy.

8.4. res directory

Lastly, outside the "Java\com\firstapp\group10app" folder lies the "res" directory which holds all the resources our app uses. Animations for smooth transitions reside in the "anim" folder, while button shapes and backgrounds are in "drawable." Important XML files for layouts, navigation bars, and app values are neatly organised within "layout," "menu," and "values," respectively. This clear organisation ensures that each part of our app knows where to find what it needs to run effectively.

# Anonymous Mode

Anonymous mode allows users access to the basic functions of the app, without the need to input any data. It is also fully offline mode, which resulted in an offline Database being created and used during anonymous mode. Whilst anonymous, searching for workouts is the only option granted to the user, since everything else requires user data or internet. Settings are also disabled since everything within relates to the user. The Home page is also remodelled, informing the user they are in offline mode and showing them the limited options that they have.

# ChatGptClient class

The ChatGptClient class is a compact, 62-line file, and that’s with all the necessary imports, space for readability, and detailed documentation comments. The creator of this class is proud of how few lines he’s coded.

This class serves the purpose of communicating with the OpenAI GPT-3.5 API, allowing our app to send a chat message and retrieve the response.

Code-wise, it includes three essential static final variables:

* SECRET\_API\_KEY
* MODEL
* URL

It also has just one method named chatGPT. Here is a summary of how it works:

* The method takes the user's input, the chat message to send to the API, as a parameter.
* It then connects to the API using an HTTP POST request with the appropriate headers and authorization using the SECRET\_API\_KEY.
* The method constructs the request body, including the model to use (MODEL), the user's message, and the maximum number of tokens.
* Next, it sends the request to the API and retrieves the response.
* Finally, it extracts and returns the text part of the API's response, which we would then manipulate and present to the user neatly.

This method is crucial for our app’s functionality since two key features are built on top of it.

# Prompt Engineering

Prompt Engineering was a critical part of the project. The AI had to be requested to return results in the appropriate format, as the app uses JSON format.

These prompts are used twice within the app, first when generating a workout, and secondly when enhancing one. Although they have a similar structure, the actual query asked to ChatGPT differs between them, as explained below.

11.1. Generating a Workout

When generating a workout, the prompt starts by outlining how the result should be structured, and what each part of the JSON should include. During testing, certain parts of the JSON would be generated in incorrect formats. For example, set-based exercises would occasionally appear as time-based (AI gives time instead of reps). This was quickly resolved by adding metadata next to Sets, stating that it should only be a value if the exercise generated is set-based. The next part of the prompt gives some basic data about the user, such as their date of birth or their weight. These values are not necessary and depend on what the user has given the app. Additional data the user gives on the Workout Generation page is also used, such as the workout goal and equipment available to the user. The final line specifies some other information about the result, most notably that it needed to be in a single line.

11.2. Enhancing a Workout

The other prompt is used when enhancing a workout. This workout can either be searched for, manually created, or generated. Once enhanced, the “new” workout is added to the database, which means the prompt also needs to return the output in a specific format. The prompt itself is much simpler than the generate, due to a less variety of data required. The first prompt explains a workout would be given and the following condition needs to be applied. A final note added is the format, which is the same as the format when generating a workout - JSON.

# Design (UI)

We've dedicated a lot of time, maybe even more than we initially planned, to creating a beautiful design for our application. We went through multiple redesigns to reach the final version. Following the best design practices was key. We made sure buttons were large enough for easy tapping, maintained a coherent colour scheme with complementary and accent colours, kept pages clear with space between elements, added a dark mode option, and included subtle animations where they made sense. We also globalised our design within a few files to keep it as consistent throughout the app as possible.

First things first, choosing the right colours was crucial. We found popular gradients online and used tools like Canva's colour wheel and websites like paletton.com to suggest complementary colours. We also created a dark theme which opts for a dark grey tone. In the dark theme we flipped lack elements to white for contrast, and most colours, like our "tetradic1" complement, are lightened a bit for a balanced look.

Crafting our logo and selecting suitable icons and illustrations was also something we spent time on. These elements enhance the app's intuitiveness and overall aesthetic appeal, contributing to a more user-friendly experience.

Transitions within the app were kept minimal yet purposeful. For instance, between our main pages - Home, Workouts, and History - we implemented sliding animations. These pages are fragments that are meant to flow side by side, so we added left and right sliding animations. Smart logic determines the direction of the slide, enhancing user interaction.

Specific elements like buttons received a lot of attention to ensure they have a beautiful design. We settled on rounded corners, small text, and a subtle colour change when pressed. This combo proved effective after feedback from external testers, including Nikola’s girlfriend and Ethan’s friends from the swimming society. Similarly, toggle buttons now indicate the active selection with colour, adding vibrancy to our previously too monotone and black-and-white pages.

In the most recent stage of coding, dialogue boxes underwent a significant transformation, with added padding, rounded corners, and subtle animations. This change softened the app's overall feel, moving away from the boxy appearance prevalent in earlier versions.

To streamline our design across the app, we utilised styles.xml and themes.xml. In the former, we stored details about common elements such as buttons. That significantly reduced repetitive code and proved invaluable, because there are over 45 buttons in our app as of the 28th of March. Themes were split into light and dark mode files, aligning with Android's best practices. This setup allowed us to easily switch between and modify the two appearances.

Lastly, the dimes.xml file, added later in development, turned out to be incredibly useful for our coherent UI design. It houses global information like font sizes, button heights, corner radius, and more, referenced throughout our XML files. This centralised approach simplified tweaking design elements, making it easy to maintain our app's visual consistency.

# Code Optimizations

While working on the Workout App, we made sure to follow good coding practices like clear file and variable naming and basic Object-Oriented Programming (OOP) principles. All of that made it easier to identify potential areas for optimization and made it simpler to put them into action.

One minor optimization was using primitive types wherever possible. This streamlined our code, making it more efficient. It also made debugging easier, because Android Studio’s built-in debugger works better with primitives. However, that may very well be considered just a good coding practice.

However, transitioning to Fragments instead of creating new Activities for each page was a key optimization that we’ve implemented. Though we made this switch later in development, we prioritised converting essential pages like Home, Workout, and History, along with the Settings page and its tabs, to Fragments. This change improved the app's performance and navigation. It also made the transition between these specific pages feel smoother because we could implement transitions in a better way.

Also, recognizing the strain on the main thread, we optimised tasks by moving them to separate threads. Previously, we noticed skipped frames due to heavy querying of the online database. Sometimes more than 100 frames would be skipped. By executing these queries concurrently with view initialization, we reduced the strain on the main thread and improved overall responsiveness. Now the view would load in an instant and any important data that we need to get from the db would populate the view asynchronously a moment later.

Another optimization we focussed on was the efficient management of session data. Storing this information in a dedicated Session.java class allowed us to minimise calls to the online or local database. As of March 28th 2024, this class which has 58 calls from other parts of the app. That is 58 fewer database queries. Reducing the need for repeated db queries, but instead getting the data instantaneously from a saved variable in a Java class is a massive time improvement.

Finally, the creation of the "DatabaseManager" Java class as a singleton was a significant step. This class maintains an open connection to the Online or Local Database, eliminating the need to initialise a new connection every time we interact with the database. This optimization saved time and resources, ensuring smoother database operations.

# Testing

Testing the app was a continuous process throughout development. At the start of the project, most testing was to ensure that anything we developed simply worked, such as checking a button to get data from a database or a fragment correctly loaded. The majority of the structured and crucial testing occurred towards the end, once enhanced, anonymous mode, and AI systems were officially initialised. The tests would often run through specific scenarios of potential customers running the app, which would each use it differently.

Testing often would involve attempts to break the app, giving false data or trying to cause conflicts with the database.

Aside from each member of the project testing regularly, we also had external help from Asya and Ben Foster, each of whom gave valuable opinions after doing real-world testing.

Major issues that occurred are discussed in the next section.

# Issues and Resolutions

15.1. Signed-In Tests:

Issues related to the AI and how the data is handled arose. For example, the AI would often ignore some parts of the prompt, resulting in incorrect data being returned and the app crashing. The fix was to change the prompts to the AI to be more robust and make sure the result it returns is in the correct form of JSON.

Another major issue was the enhanced AI resulting in an error that no API key was issued. This was strange, since it runs through the same ChatGPT class file as the Generate Workout, which had no issues. This meant it could not be an issue with the key. After inspecting the prompt, it turned out ChatGPT doesn’t like line breaks when prompting it. We remove line breaks from our prompt and it has not broken since.

Next issue that we encountered was communication with the online database in the History page. We tried obtaining data rows using JSON\_ARRAYAGG, but the order in which the rows were returned would be random. We instead needed it to be chronological. This was fixed by using JSON\_ARRAY (instead of JSON\_ARRAYAGG) and handling the returned data manually in the Java file.

15.2. Anonymous:

After Anonymous was finished, we did a few test runs to check that it all worked. Oddly, when we tried to finish a workout, it would immediately crash. As it turned out, the logic behind the button was to immediately add the workout to the History table in the online database. The online database, however, is not active, because Anonymous uses the local database. We fixed that as well.

# Conclusions

15.1. How well did your final product work?

Our final product stands as a result of the efforts put into its development. Overall, users can expect smooth navigation and UX throughout the app while using the features we implemented. Feedback from potential users has been promising, with many expressing interest in using the app when available on Google Play Store. Highlights in the developing of the app include:

* The implementation of both online and offline databases was a significant milestone. This enabled users to utilise the application even without having an internet connection. Also provided greater anonymity for anonymous users.
* Another important aspect was our session management system. From the moment users log in until they log out, session management stores relevant info and eases the strain on the database.
* Additionally, the implementation of encryption mechanisms ensures the protection of user accounts, by employing SHA 256 encryption for users’s passwords.
* However, our most distinctive feature is the integration of AI in the app’s core. At this moment the use of OpenAI’s ChatGPT and their APIs has made the integration smooth, we have been able to successfully implement it in two distinct ways: generating workouts based on user’s preferences and enhance existing workouts either made by AI or the user themselves.

15.2. How does it compare with other, similar projects?

Comparing our app reveals significant strengths. In contrast to apps like Samsung Health and Nike Training, our app offers a distinct offline mode, a feature almost always overlooked in similar applications. Additionally, the integration of AI capabilities sets our app apart. Unlike most health apps, we have successfully implemented not one, but two AI functionalities: the ability to generate workouts based on user needs and the option to enhance existing workouts.

15.3. How novel are your ideas?

As stated above, almost no other fitness app has offline availability and AI features when online.

15.4. What guidance can you offer to others setting out with similar aims?

For those embarking on a similar journey, we offer valuable insights gained from our experience. First, a solid understanding of Android app architecture is crucial. Learning what Activities and Fragments are, along with the effective use of resource folders like res, save time and effort in development. Additionally, a focus on prompt engineering for GPT Models can yield more accurate and useful results.

15.5. What scope is there for further work on the topic?

Looking ahead, there are several exciting things for further development. Ideas include integrating workout reminders with personal calendars, implementing an AI-driven meal planning, and connecting the app with the GPS. The last one will be useful for running activities. Adding more workouts, incorporating short instructional videos/gifs, and enhancing user statistics are also cool things we can do. Finally, exploring iOS compatibility is also up for future consideration.

# Acknowledgements

We want to give special thanks to the numerous people who have helped in the success of this project. With special thanks to our supervisor Sergey Ovchinnik whom this project would not have been possible without. He has given us continuous support and guided us throughout. Asya Rizova, who has tested our app, suggested colour schemes as well as spent valuable time designing the final logo. More thorough testing was done by Ben Foster.

# Bibliography

Android Developers. (n.d.). *Android API Reference*. [online] Available at: <https://developer.android.com/reference>.

Android Developers. (n.d.). *SensorManager*. [online] Available at: <https://developer.android.com/reference/android/hardware/SensorManager>.

Android Studio Development Essentials. (2015). Available at:

<https://www.ebookfrenzy.com/pdf_previews/AndroidStudioEssentialsPreview.pdf>.

Apple (United Kingdom). (n.d.). *Apple Fitness+*. [online] Available at: <https://www.apple.com/uk/apple-fitness-plus/>.

Apple (United Kingdom). (n.d.). *iOS - Health*. [online] Available at: <https://www.apple.com/uk/ios/health/>.

Canva (2019). *Colour Wheel - Color Theory and Calculator | Canva Colors*. [online] Canva’s Design Wiki. Available at: <https://www.canva.com/colors/color-wheel/>.

Google Developers. (n.d.). *Gmail API Overview*. [online] Available at: <https://developers.google.com/gmail/api/guides>.

Google Fonts. (n.d.). *Manrope*. [online] Available at: <https://fonts.google.com/specimen/Manrope>.

Nike (2023a). *Nike Run Club App*. [online] Nike.com. Available at: <https://www.nike.com/gb/nrc-app>.

Nike (2023b). *Nike Training Club App. Home Workouts & More.* [online] Nike.com. Available at: <https://www.nike.com/gb/ntc-app>.

OpenAI. (n.d.). *OpenAI API*. [online] Available at: <https://platform.openai.com/docs/api-reference>.

OpenSSL Org. (n.d.). *SHA Documentation*. [online] Available at: <https://www.openssl.org/docs/man1.0.2/man3/sha.html> [Accessed 3 Apr. 2024].

Paletton. (2009). *Paletton - The Color Scheme Designer*. [online] Available at: <https://paletton.com/>.

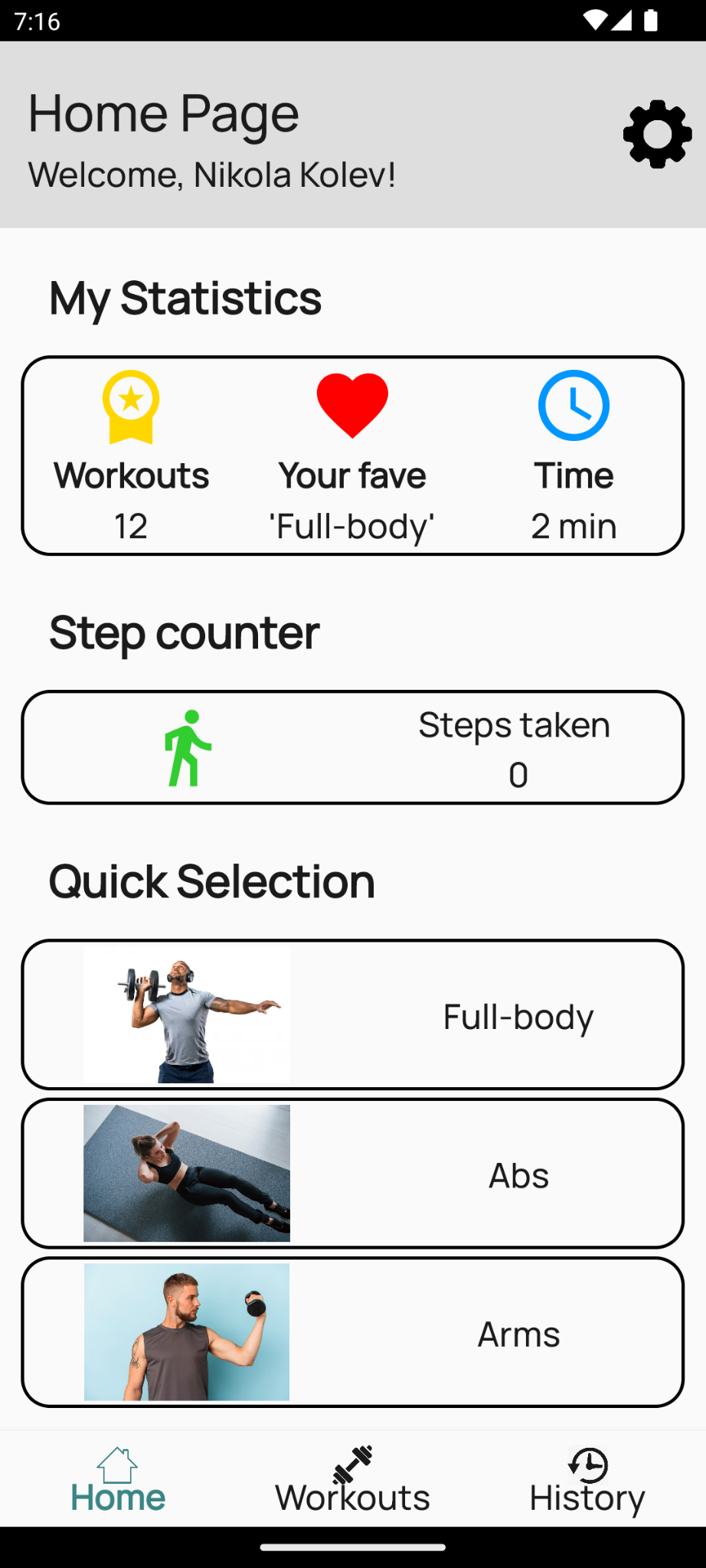
PingCap Org. (n.d.). *TiDB Cloud Documentation*. [online] Available at: <https://docs.pingcap.com/tidbcloud/> [Accessed 3 Apr. 2024].

Samsung UK. (n.d.). *Samsung Health | Apps & Services | Samsung*. [online] Available at: <https://www.samsung.com/uk/apps/samsung-health/>.

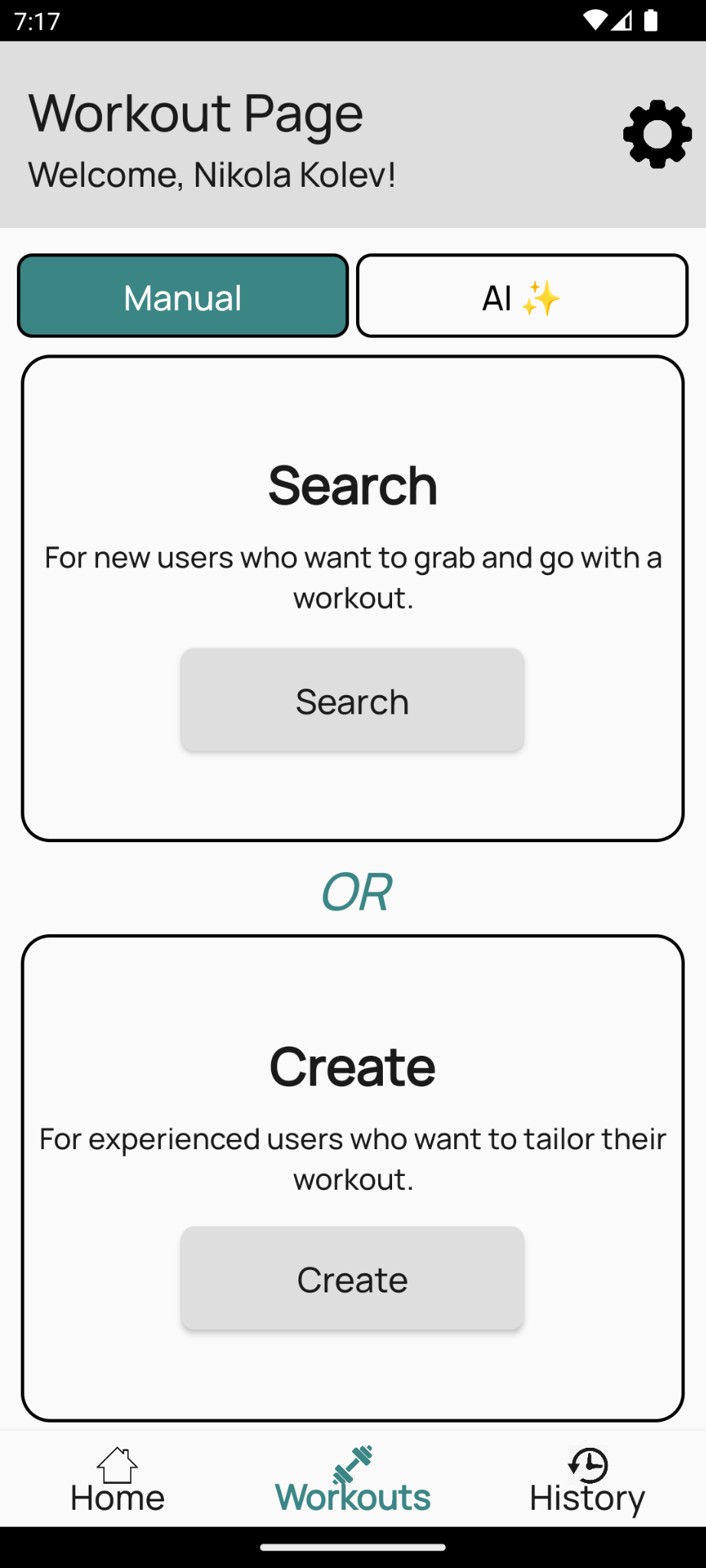
SQLite Organisation. (2019). *SQLite Documentation*. [online] Available at: <https://www.sqlite.org/docs.html>.

# Appendices

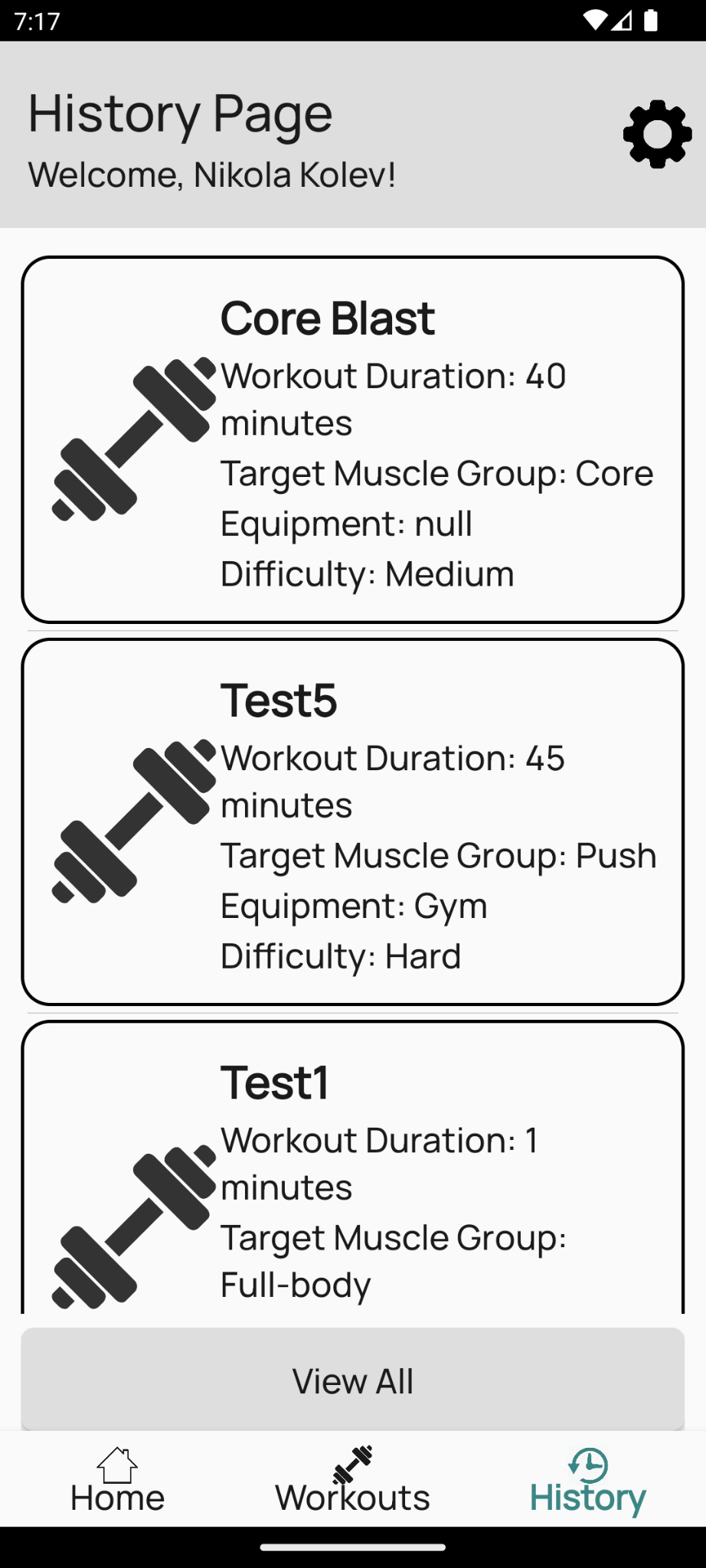
***Figure Wf.1***



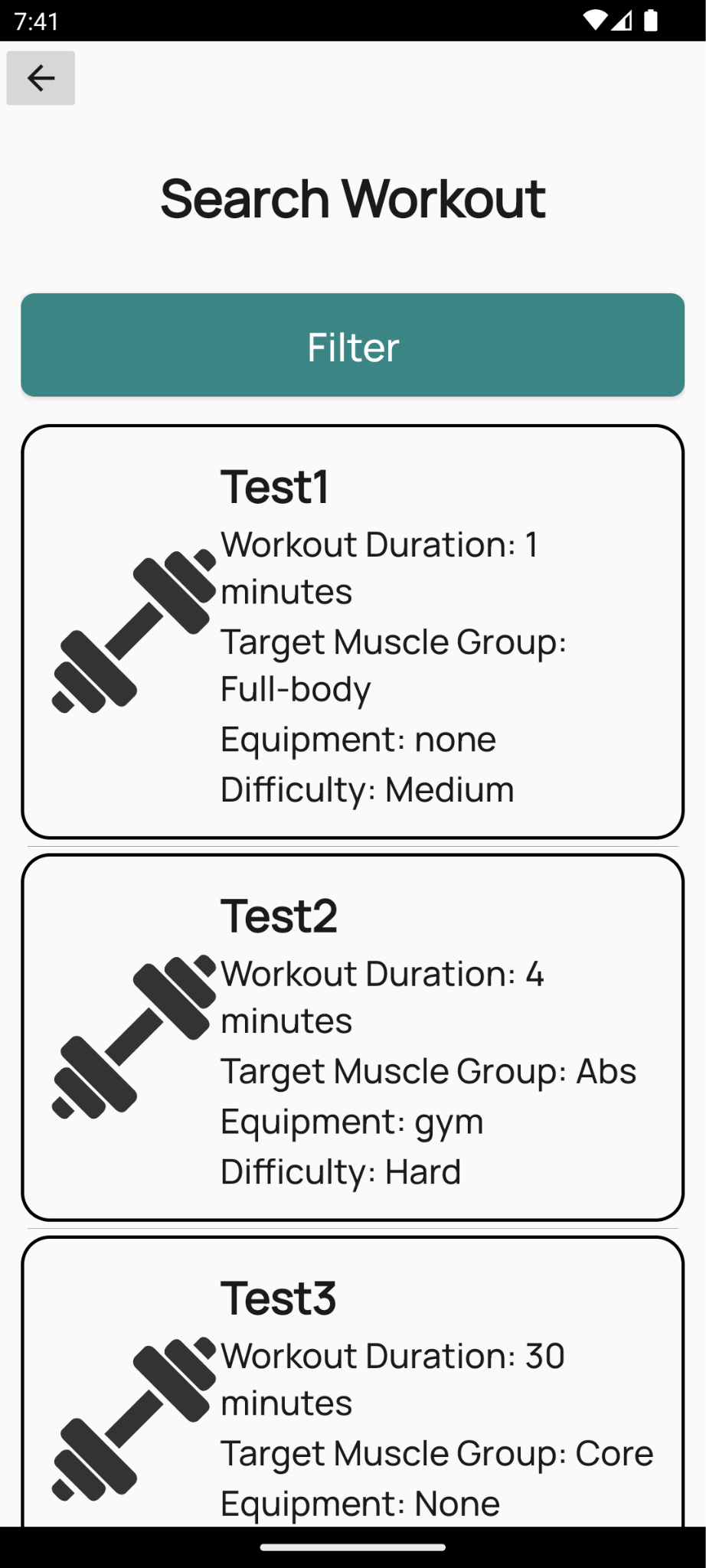
***Figure Wf.2***



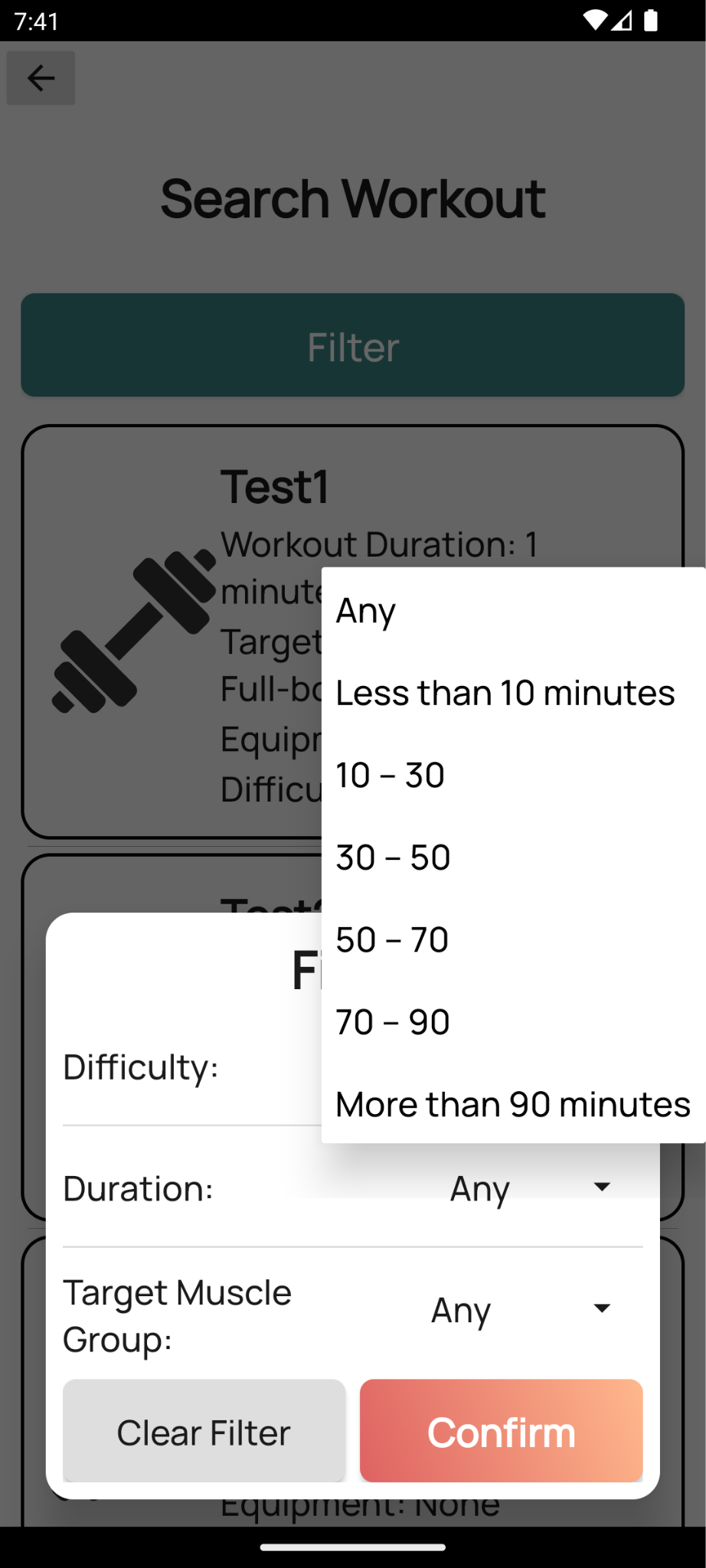
***Figure Wf.3***



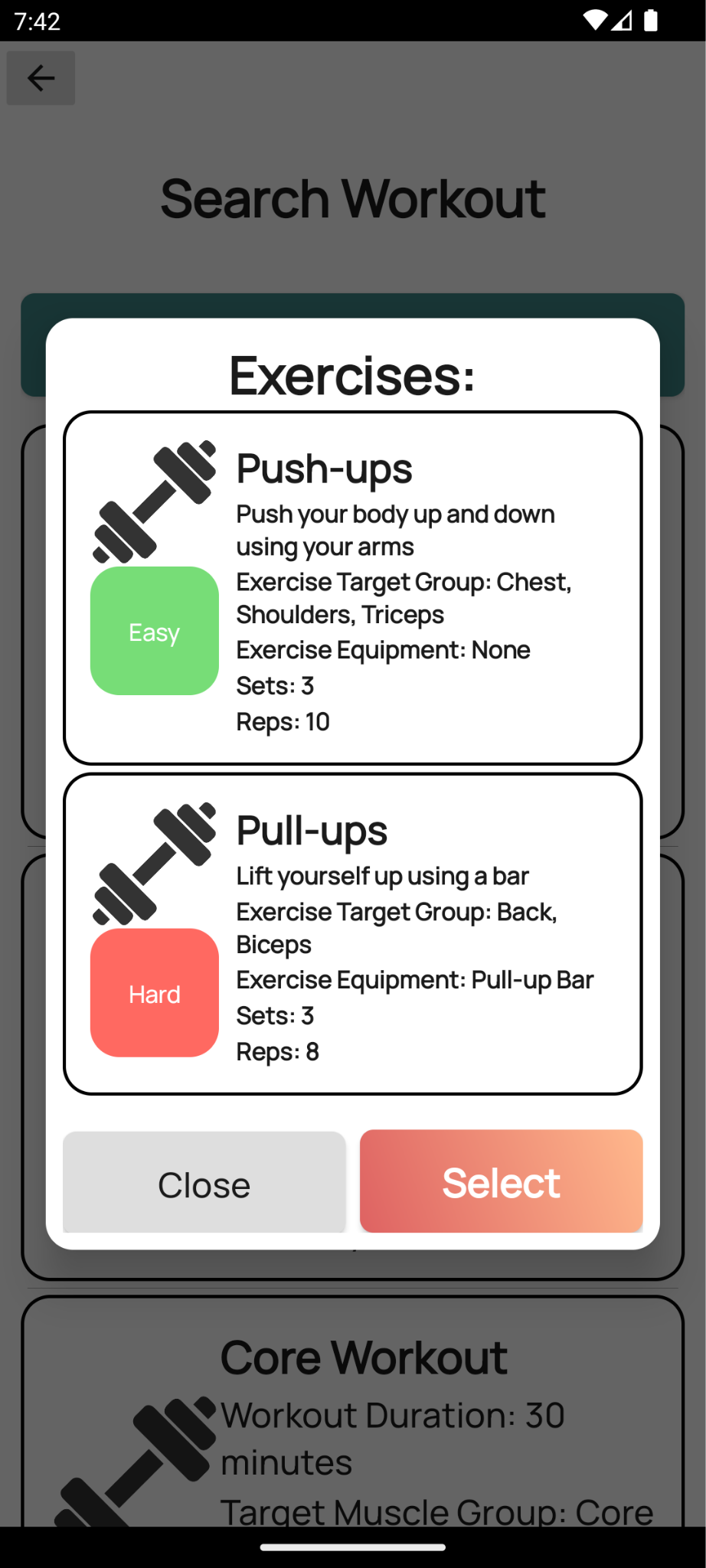
***Figure Wf.4***



***Figure Wf.5***



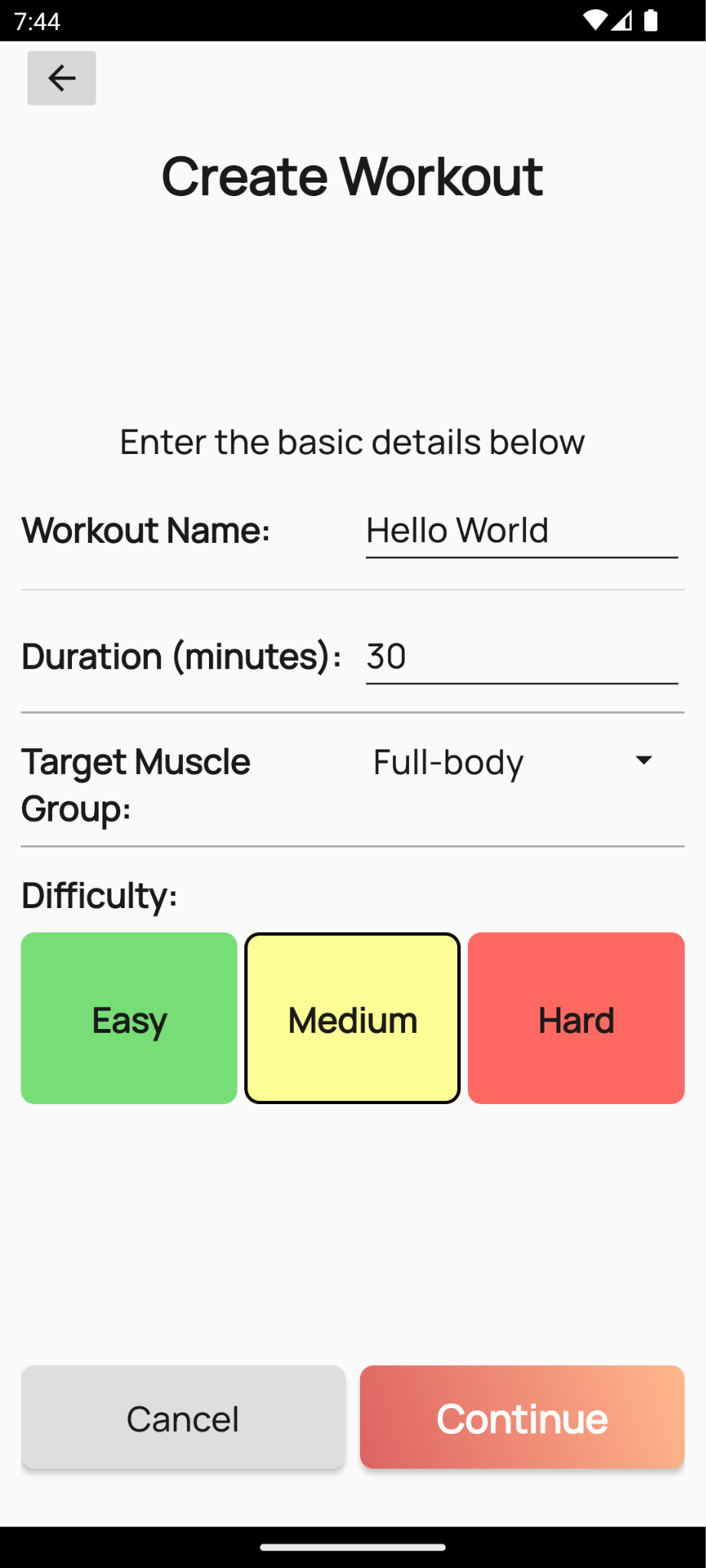
***Figure Wf.6***



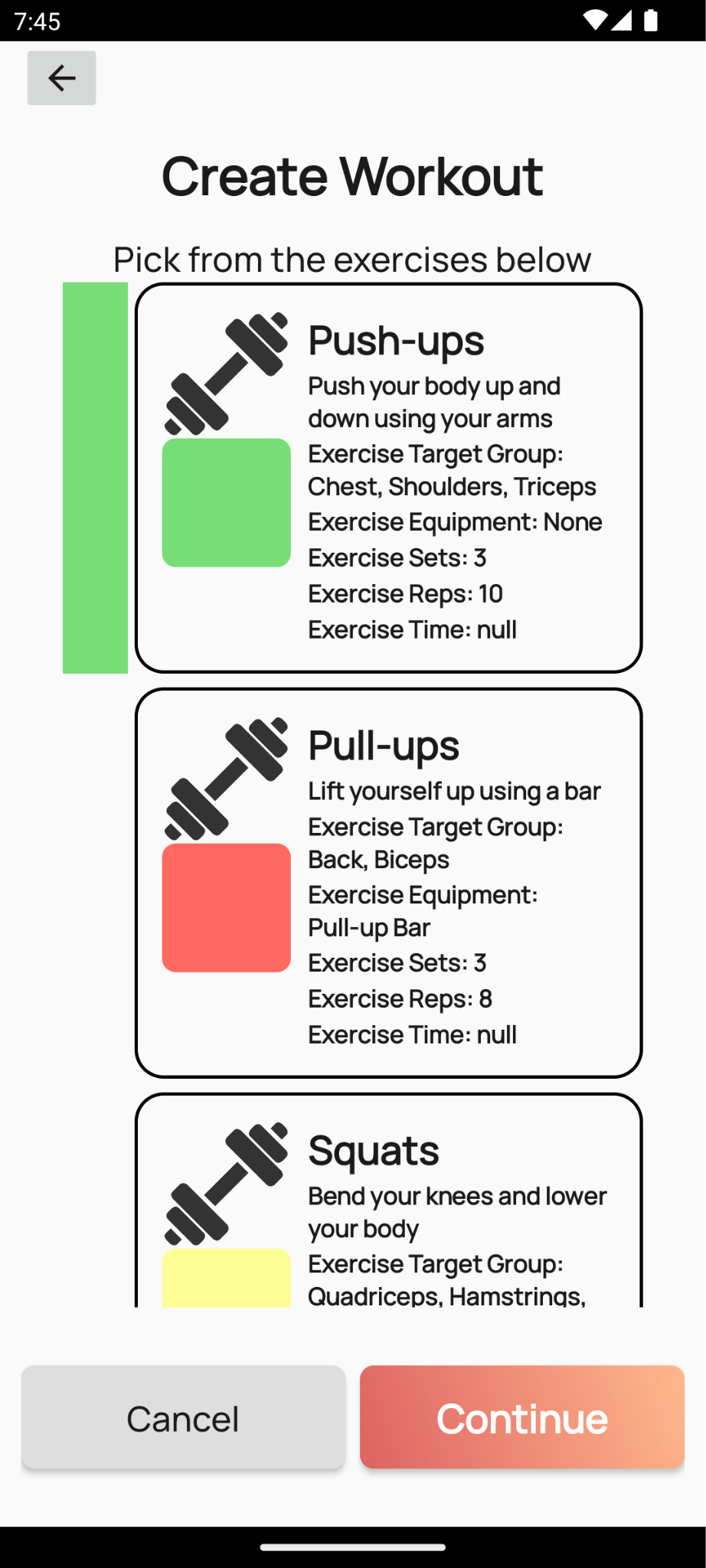
***Figure Wf.7***

******

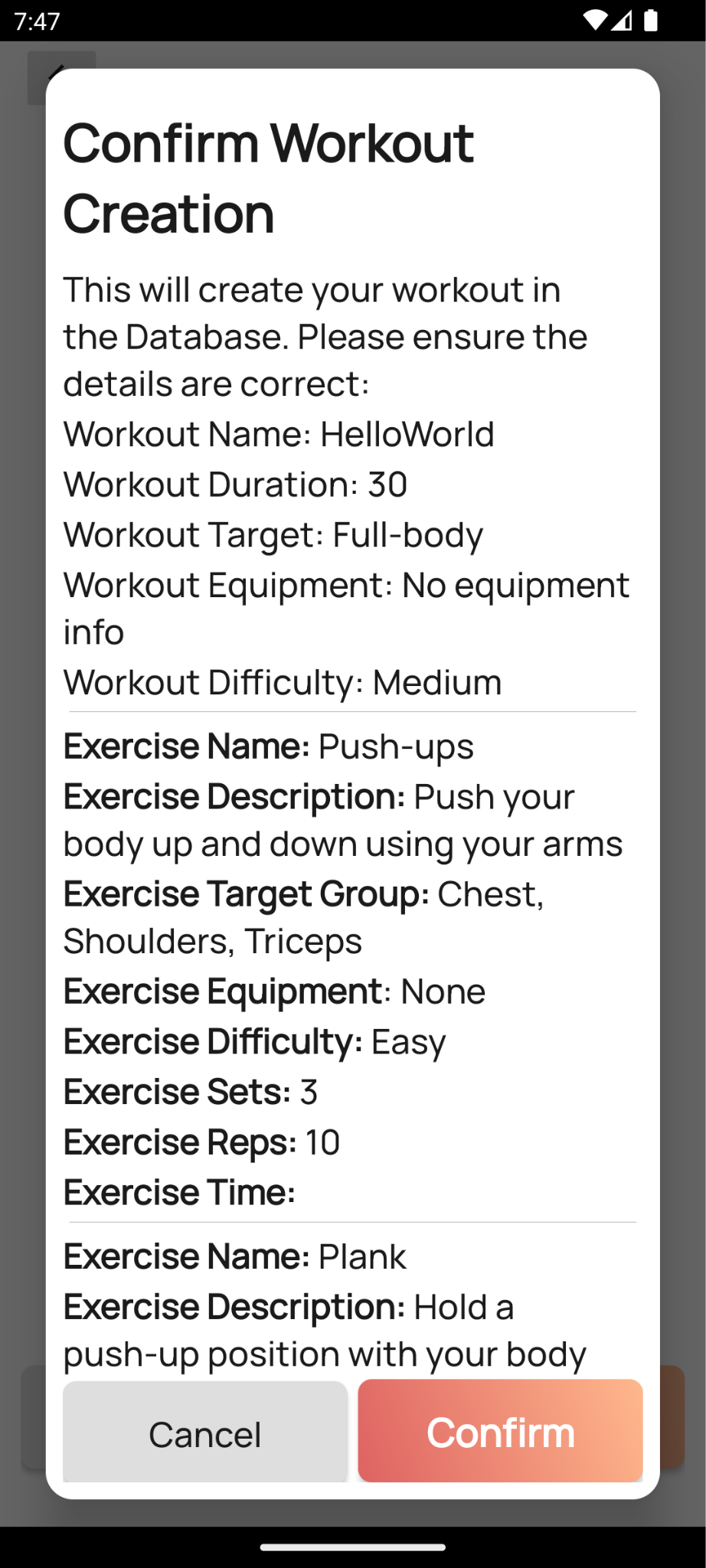
***Figure Wf.8***

******

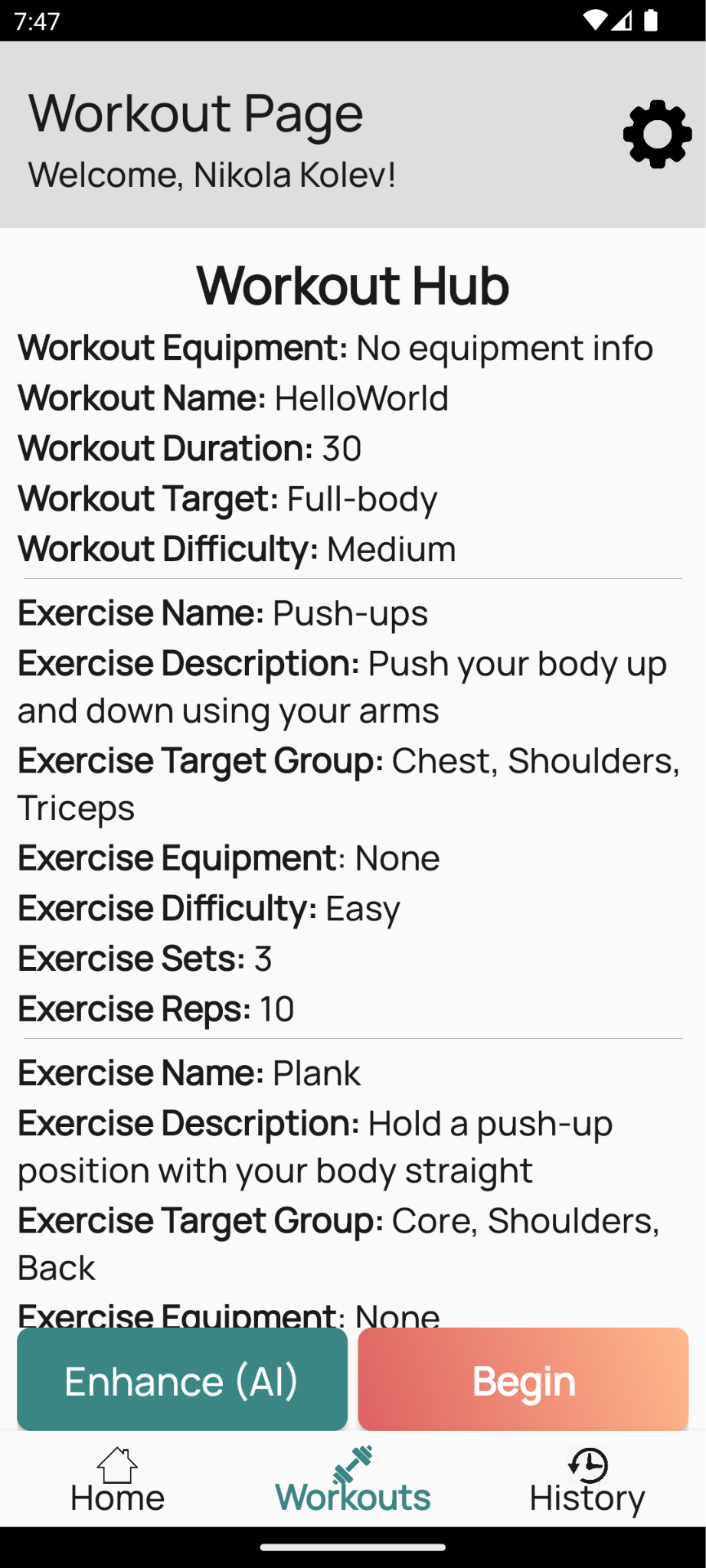
***Figure Wf.9***

******

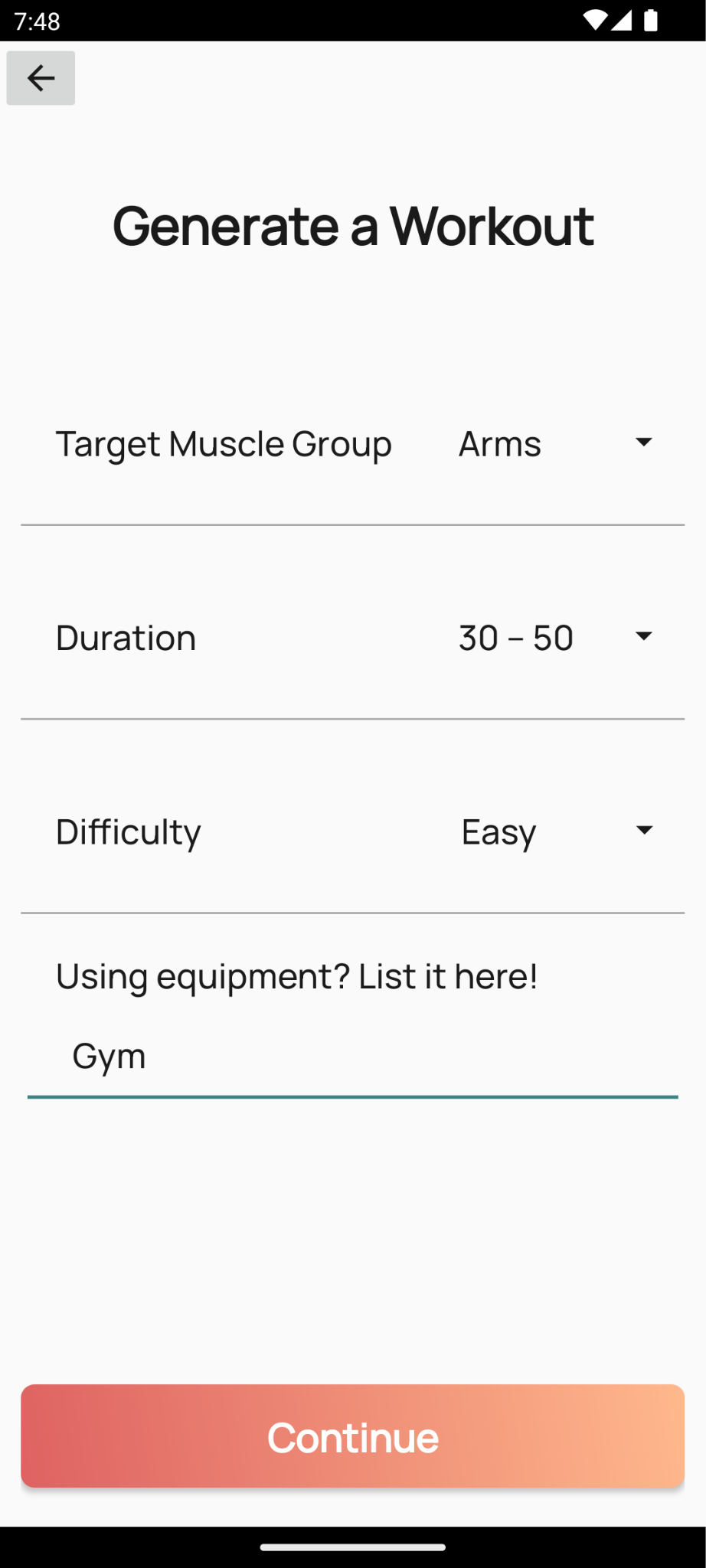
***Figure Wf.10***

******

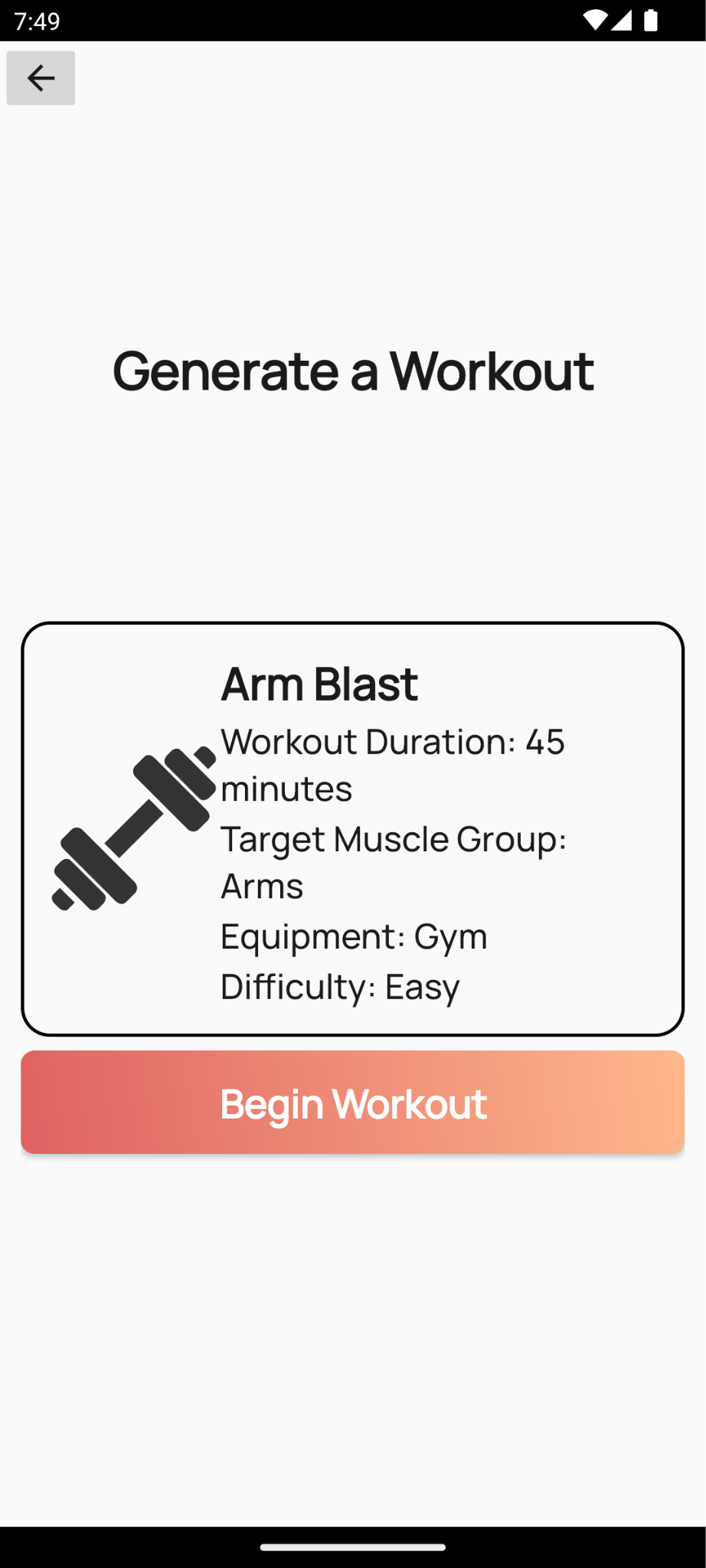
***Figure Wf.11***

******

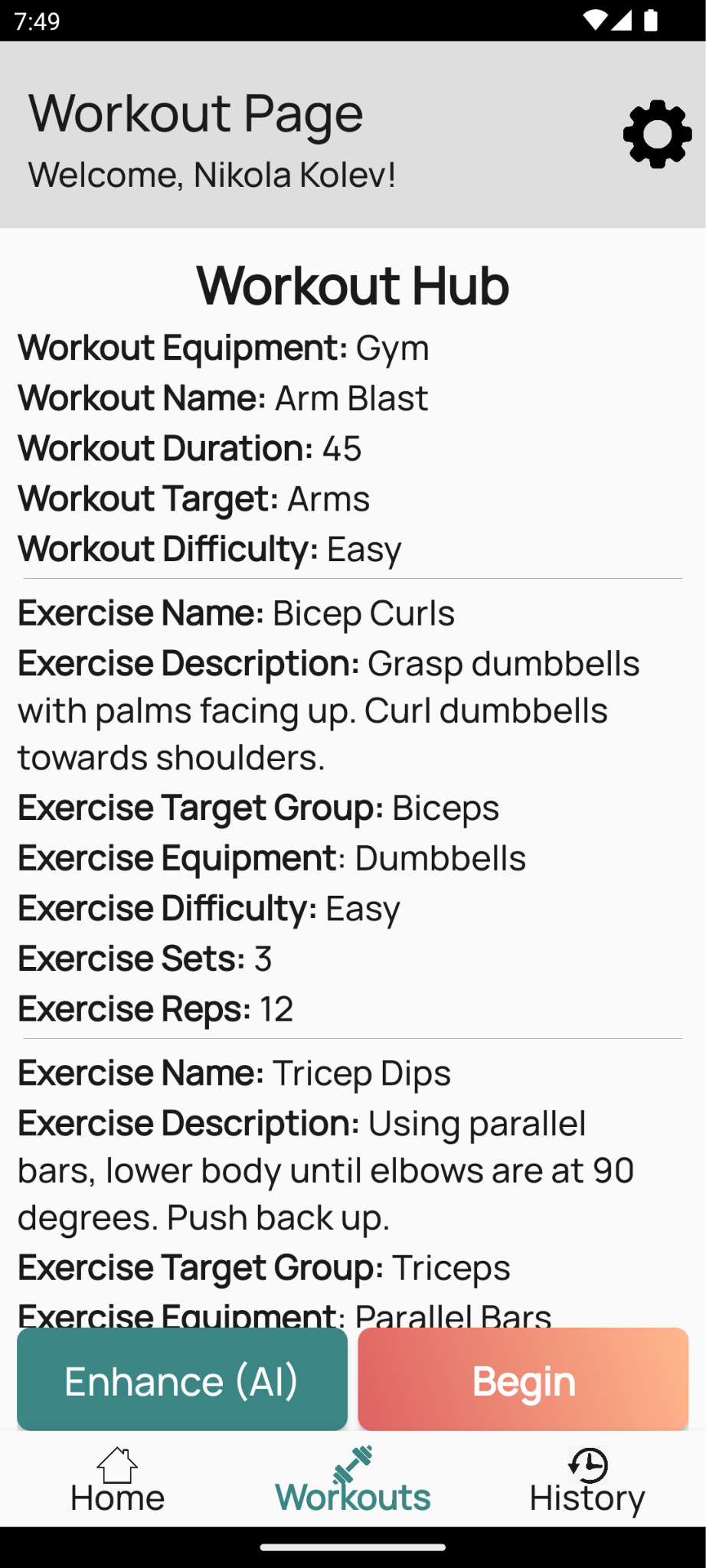
***Figure Wf.12***

******

***Figure Wf.13***

******

***Figure Wf.14***

******

***Figure Wf.15***

***A diagram of a program

Description automatically generated***