

Design and Implementation of Mobile Applications

A.A. 2022 – 2023

Design Document



FITUP

By

**Luca Cassenti
Nicolò Giannini**



POLITECNICO
MILANO 1863

Contents

Introduction	4
Purpose	4
Main Features	4
Definitions, Acronyms, Abbreviations	5
Revision History	7
Document Structure	7
Requirements	8
Implementation, Integration and Test Plan	10
Backend	10
Frontend	11
Interaction between WearOs and Mobile	11
Flow	12
Spotify API	15
Testing	15
Unit Testing	16
UI Testing	17
Architectural Design	19
Overview	19
Use Case View	21
Deployment View	23
Components View	24
Android Client	24
Firebase Server	27
WearOs Client	29
Sequence Views	30
Sequence Diagram - Login	30
Sequence Diagram - Edit Workout	31
Sequence Diagram - Play Workout	32
Future Server Replacement	33
Benefits of Server Replacement	33
Gamification	34
Introduction to Gamification	34
Gamification in Fitup	34
Advantages of Gamification in Fitness	34

Future Steps	35
Custom Layouts and Views	36
Custom Layouts	36
LineGraphLayout	36
WorkoutLayout	37
Custom Views	37
CircularCountdownView	37
LineGraphDataView	38
MinimizedExerciseView	38
ThreeDotsLoadingView	38
User Interface Design	39
Principles and Guidelines	39
Style and Visual Identity	40
Colour Schemes	41
Typography	42
Iconography	42
Mobile App UI	43
WearOs App UI	48
User Experience Diagrams	49
References	50
Table Of Figures	51

Introduction

Fitup is a versatile fitness application designed for both home and gym workouts. With customization at its core, the app empowers users to tailor their exercises based on personal preferences, adding exercises, with sets and repetitions to their workouts. Progress tracking is integrated into Fitup, enabling users to effortlessly monitor their achievements, including body weight, waist measurements, maximum repetitions, and calories burned.

Enhancing the experience further is Fitup's integration with WearOS. This feature enables real-time heart rate monitoring during exercises and grants users the convenience of managing their workout directly from their watch. Fitup takes motivation to the next level by dynamically pairing each workout with a Spotify playlist, ensuring rhythm matches the pace of exercises.

In a departure from traditional gym apps, Fitup transforms workouts into engaging games where users challenge virtual monsters through physical effort, making fitness both effective and enjoyable. Available in English and Italian, Fitup caters to a global audience, presenting a holistic and interactive approach to fitness.

Purpose

This is the Design Document of the Fitup mobile application. The purpose of the document is to provide a functional description of the main architectural components, showing their runtime behaviour and interactions and their interfaces. This document is mainly intended to be used by developers, testers and stakeholders.

Main Features

- **Workout Creation and Editing:** This app empowers users to create and customize their workout routines effortlessly. It offers a diverse range of presets and allows users to add custom exercises, tailoring their fitness regimen to meet their specific goals. Whether you're looking to build strength, improve endurance, or achieve a balanced workout, you can design a routine that suits your needs.
- **Workout Tracking:** Keeping track of your workouts is made easy with built-in timers and a gamified interface. This feature not only helps users stay organized but also adds an element of fun and motivation to their fitness journey. You can set goals, track progress, and celebrate your achievements as you work towards your fitness objectives.
- **Music Integration:** For an even more enjoyable workout experience, the app seamlessly integrates with Spotify. It provides curated playlists with music selected to match the tempo of your training. Exercising to the rhythm of the music can boost your motivation and performance, making your workouts more enjoyable.
- **Wearable Device Support (Wear OS):** If you own a Wear OS smartwatch, the app takes advantage of it to enhance your training experience. You can conveniently track your heart rate and bpm (beats per minute) in real-time, ensuring that you're exercising at the

right intensity level to meet your goals. This wearable integration adds both comfort and precision to your workouts.

- **Health and Exercise Data Monitoring:** In addition to tracking your workouts, the app lets you monitor essential health metrics. You can keep tabs on your weight, body fat percentage, and even specific fitness achievements like max reps in exercises such as bench presses. This comprehensive data allows you to gain insights into your overall health and fitness progress.
- **Custom Exercises:** The app goes the extra mile by allowing users to add their own unique exercises. This feature ensures that your workout plans are truly personalized, catering to your individual preferences and needs. Whether it's a specialized yoga pose or a unique strength exercise, you can include it in your routines.
- **Pre-set Workouts and Exercises:** For those looking to get started quickly or seek inspiration, the app offers a library of pre-set workouts and exercises. These pre-configured routines cover a wide range of fitness goals and provide an excellent starting point for users new to fitness or seeking variety in their workouts.
- **Graphs and Data Tracking:** Visualizing progress is crucial for motivation and goal tracking. The app provides easy-to-read graphs and detailed data tracking, allowing you to see how your efforts translate into results over time. This data-driven approach empowers users to make informed decisions about their fitness journey and make necessary adjustments for continuous improvement.

Definitions, Acronyms, Abbreviations

Android: Android is an open-source operating system based on Linux, developed by Google, designed for mobile devices. It offers a development environment rich in tools and libraries that empower developers to create innovative and interactive applications for smartphones, tablets, and wearable devices.

Kotlin: Kotlin is a modern open-source programming language developed by JetBrains. Kotlin has been officially embraced by Google as a supported programming language for Android application development. It provides concise, secure, and expressive syntax, along with an extensive array of features that simplify the creation of robust and performant applications.

Firebase: Firebase is a mobile and web app development platform offered by Google. It encompasses a set of tools and services that streamline the management of backend aspects of applications, including authentication, real-time databases, hosting, analytics, messaging, and more. Firebase accelerates the development process, enabling developers to focus on crafting innovative features.

XML (Extensible Markup Language): XML is a markup language that defines a set of rules for encoding documents in a format that is readable by both machines and humans. In the context of Android, XML is utilized to define the structure of the user interface, including layouts, graphical elements, text, and attributes. The declarative structure of XML separates interface definition from programming logic.

Fragment: a fragment is a fundamental concept within Android app development. Represents a reusable user interface component and corresponds to a portion of interface and functionality

within an activity. An Android application can have one or more activities, and each activity can contain one or more fragments.

API (Application Programming Interface): A set of definitions and protocols that allow different software components to communicate with each other. APIs facilitate interaction between applications and services.

UI (User Interface): The visual interface through which users interact with an application. It includes elements such as buttons, text fields, images, and layouts.

UX (User Experience): The overall experience of a user while interacting with an application. It encompasses aspects like ease of use, intuitive navigation, and user satisfaction.

MVVM (Model-View-ViewModel): Architectural pattern that separates the application into three components: the Model represents data, the View represents the user interface, and the ViewModel manages presentation logic and interface states.

AsyncTask: An Android class that simplifies the execution of asynchronous operations to prevent blocking the main UI thread.

Intent: An object used for communication between different components of an application or between different Android applications. Intents are often used to start new activities or services.

ViewModel: An architectural component that stores and manages data needed for the user interface. It's part of the MVVM architecture.

Revision History

Date	Version	Comments
August 2023	1.0	First release

Document Structure

The document is structured according to the following structure:

- **Introduction:** Containing a general overview of the document.
- **Requirements:** Showing the user and system requirements that the app should satisfy.
- **Implementation Integration and Test Plan:** Detailing the implementation, integration of subcomponents occurred and testing of the system.
- **Architectural Design:** Showing the main architectural components of the system and their relationships, underlining the architectural choices in the design process.
- **Gamification:** Detailing the system's use of gamification and future plans, detailing the importance of this feature for an engaging experience.
- **Custom Views and Layouts:** Reviewing the custom layouts and views coded for the application and the importance of custom software in mobile applications.
- **User Interface Design:** Showing the user interface and the design choices behind the development.

Requirements

R.1	The User must be able to sign up or to log in if he is already registered
R.2	The System must check if the User credentials are valid
R.3	The System must allow to log in using personal credentials
R.4	The System must allow changing password if it has been forgotten, through the personal email
R.5	The System must allow the User to insert personal information at the first access
R.6	The User must be able to insert personal information at the first access
R.7	The User must be able to see the available Workout
R.8	The User must be able to edit a Workout
R.9	The User must be able to create a new Workout
R.10	The User must be able to create a new Exercise
R.11	The User must be able to delete a Workout/Exercise only if he is the creator of that Workout/Exercise
R.12	The User must be able to start and play a Workout
R.13	The User must be able to play the Spotify playlist associated to the current Workout
R.14	The User must be able to control the Workout flow either from the Mobile or from the WearOs
R.15	The User must be able to see the bpm from WearOs and from Mobile when using the companion app
R.16	The User must be able to see the correct info about the current Workout either from Mobile or from WearOs when using the companion app
R.17	The User must be able to finish the Workout
R.18	The User must be able to see the info about Workout
R.19	The User must be able to see their analytics
R.20	The User must be able to add analytics data about their body composition
R.21	The User must be able to add analytics data about their personal records

R.22	The User must be able to categorize Workouts by type
R.23	The User must be able to search for Workouts
R.24	The User must be able to search for Exercises
R.25	The User must be able to categorize Exercises by type
R.26	The System must show Workouts categorized by the User's last use
R.27	The User must be able to change their username
R.28	The User must be able to change the System's language
R.29	The User must be able to log out

Implementation, Integration and Test Plan

In this chapter, we will delve into the technical aspects of the Fitup system and how it is structured to provide a seamless fitness experience. The Fitup system can be categorized into the following components:

- **Backend:** This section is responsible for data management, authentication, security, and integration with external APIs, including Facebook and Google. It forms the foundation of Fitup's functionality, ensuring secure and efficient data management.
- **Frontend:** Fitup offers both a mobile application and a dedicated WearOS application, designed to provide user-friendly and engaging fitness experiences. These interfaces serve as the primary touchpoints for users to interact with the Fitup ecosystem.
- **API for Exchanging Messages:** The Fitup system utilizes an API for communication between the mobile and WearOS applications. This messaging API facilitates real-time data exchange, enabling users to monitor and manage their workouts seamlessly.
- **Spotify API:** Fitup seamlessly integrates with the Spotify API to curate dynamic playlists that match the pace of users' workouts. This integration enhances motivation and enjoyment by combining fitness with music.

In the upcoming sections, the implementation details of each of these components will be explored, along with insights into the testing strategies employed to ensure the reliability, performance, and security of the Fitup system.

Backend

The application's backend infrastructure plays a pivotal role in ensuring a seamless and secure user experience. Leveraging Firebase for data management and authentication, coupled with the Kotlin programming language, has facilitated the creation of a robust and dependable backend system.

The integration of Firebase into the application offers several notable advantages:

1. **Authentication:** Firebase Authentication is employed to manage secure access and user authentication. It offers versatile login options, including email and password access, as well as access via social providers such as Google and Facebook.
2. **Realtime Database:** Firebase Realtime Database is utilized to store and synchronize user activity data. This feature provides a real-time cloud database that facilitates the seamless synchronization of data across users' devices.
3. **Simple APIs:** Firebase provides user-friendly APIs that streamline the development of authentication and data management features, significantly reducing overall development time.
4. **Security:** Firebase takes charge of infrastructure security, encompassing authentication and data authorization, thus mitigating concerns related to vulnerabilities and ensuring robust security measures are in place.

Frontend

The application's frontend was crafted using Kotlin and XML, focusing on creating a user-friendly interface. Kotlin was chosen for its modern features, making interface development, event handling, and view updates straightforward and maintainable. XML was employed to define the interface's structure, maintaining a clear separation between design and logic.

Views and user events were efficiently managed using Android's layout managers and Kotlin. Data binding played a crucial role, allowing to directly bind data from your ViewModel to the user interface (UI) of the Android app. This makes it easier and more efficient to update the UI based on changes in the underlying data and vice versa. With data binding, you can directly bind data model variables to the UI XML. This means you no longer need to search for UI elements using `findViewById` and manually update them whenever the data changes.

Since data binding is able to directly bind data to the UI, it can be more efficient than manual and `findViewById`-based approaches.

Styling and customization were simplified through XML resources, ensuring a consistent look and feel.

The frontend follows the Model-View-ViewModel (MVVM) design pattern, comprising the Model, View, and ViewModel. Kotlin Entities defined the data structures and business rules in the Model. XML layouts defined the visual elements in the View, enhancing collaboration between designers and developers. The ViewModel acted as a bridge between the Model and View, managing presentation logic and data synchronization.

XML layouts and fragments were used to create modular and adaptable UI components, contributing to a more organized and flexible frontend. The frontend interacted seamlessly with the backend through dedicated APIs and services, providing a responsive and intuitive user experience.

Interaction between WearOs and Mobile

This section examines the implementation of a message exchange mechanism between mobile devices and Wear OS devices using Google Play Services APIs. The process involves sending and receiving messages between Wear OS wearable devices and Android mobile devices, enabling synchronized bidirectional communication.

Message Sending: The sending of messages from mobile devices to Wear OS devices is managed through a class called `SendThread`. This class extends the `Thread` class and represents an asynchronous action to send messages to Wear OS devices.

1. **Data Preparation:** The `SendThread` object takes two parameters: `path`, representing the message path, and `message`, containing the content of the message to be sent.
2. **Getting Available Nodes:** The list of available nodes (Wear OS devices) is obtained by calling `wearableList`.

3. **Iterating Over Nodes:** For each node in the list, the code uses `Wearable.getMessageClient()` to send the message. The node's identifier (`node.id`) and the path (`path`) are used as parameters along with the message converted to a byte array.
4. **Result and Log:** The number of nodes to which the message has been sent is logged using a log statement.

Message Receiving: Receiving messages from Wear OS devices is handled by overriding the `onMessageReceived(messageEvent)` method of the `WearableListenerService` class.

1. **Interpreting the Path:** Based on the received message's path (`path`), the service decides which action to take. For example, for paths like `/bpm`, `/start`, `/requestExercise`, etc., the service creates specific intents with relevant actions and data.
2. **Local Broadcast:** Using `LocalBroadcastManager`, the created intent is sent as a local broadcast to notify the mobile application of the received event.
3. **Handling Non-Matching Messages:** If the message path doesn't match any defined options, the service handles the message in a default way.

Advantages:

- **Bidirectional Communication:** The message exchange mechanism allows bidirectional communication between Wear OS and mobile devices, enabling them to interact in real time.
- **Coordination:** Synchronized sending and receiving of messages enable effective coordination between devices, facilitating synchronized actions and timely notifications.

Flow

In this section we analyze the exchange of messages between the two devices. The flow of is as follows:

1. Upon opening the `WorkoutPlayActivity` the mobile sends a message with path `"/workout"` to the associated wearOs with an attached message containing the name of the currently running workout.
2. Once the wearOs receives the message from mobile via the `MessageService` listener class, the watch starts the `playWorkout` activity, starts the 5-second countdown and then the chronometer, and sends a message to the mobile to request the name of the current exercise.
3. The mobile receives the request from wearOs and sends the name of the exercise which once received from the watch will be displayed on the device
4. When the exercise is changed from mobile a message of `"/next"` will be sent to wearOs with the name of the next exercise, and when the next exercise is started a message of `"/start"` will be sent.
5. When the wearOs instead requests a `"/next"`, it will be sent to the mobile which will simulate the click of the next Button and behave as in step 4. same thing if the wearOs requests the `"/play"` of the exercise
6. When the exercise is finished, the mobile sends a `"/stop"` message and `"finish"` will be displayed on the wearOs

7. Clicking on "finish workout" on mobile will send a "/finish" message which will be handled by wearOs to return to its MainActivity waiting for a new "/workout" message.
- Bpms are handled in the same way, with the difference that whenever wearOs notices a change in bpm it sends these to the mobile.

A few sequence diagrams illustrating the interactions follow:

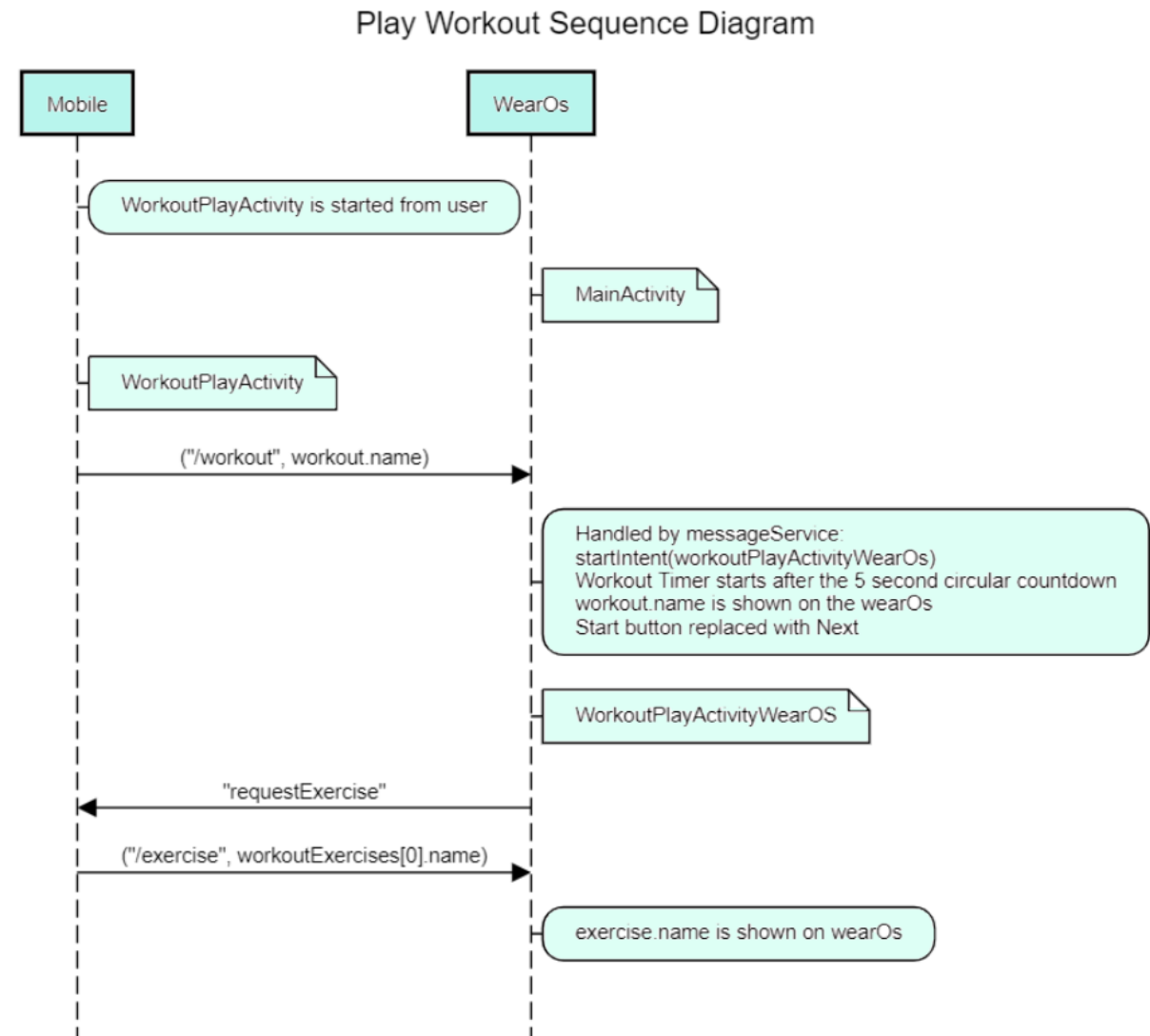
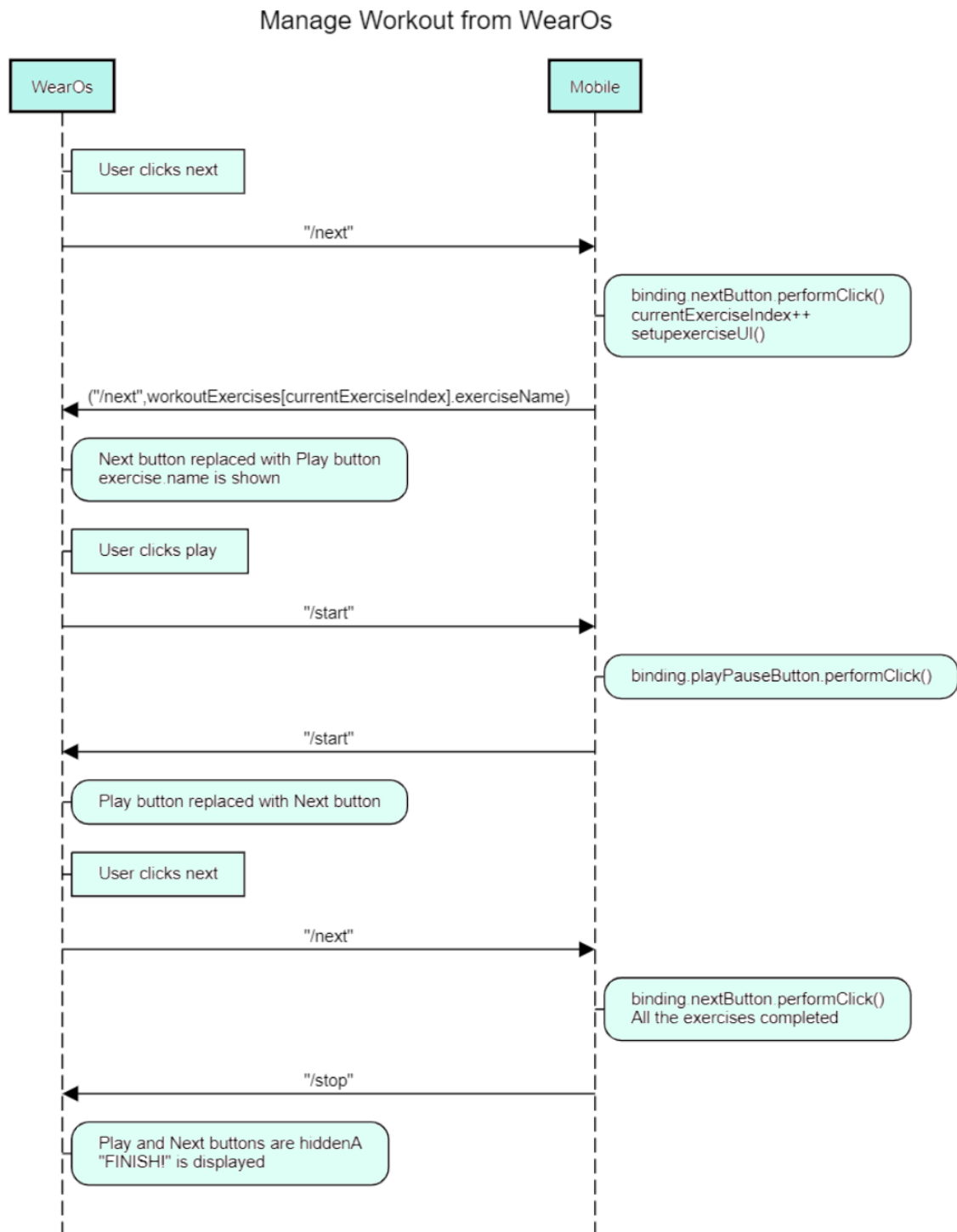


Figure 1: Sequence Diagram of Play Workout

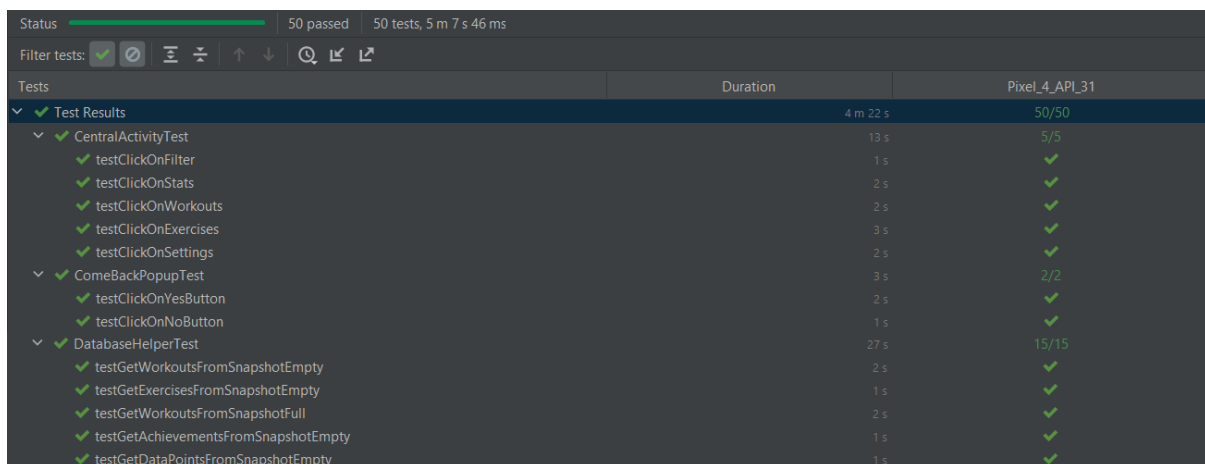
*Figure 2: Sequence Diagram Messages*

Spotify API

To initiate a Spotify playlist from an Android app written in Kotlin, a series of steps must be followed. Firstly, it involves registering the app on the Spotify Developer Dashboard. Following this, a crucial aspect is handling the authentication process, which is pivotal for securing access.

Once the access token is secured through authentication, it can then be utilized to make an API call to Spotify. This API call is instrumental in commencing the specific playlist that the user aims to play within the app. This procedure necessitates specifying preferences to the API, which will then orchestrate the playlist accordingly. In essence, it offers the user the ability to customize their music-workout experience seamlessly.

Testing



Status	50 passed	50 tests, 5 m 7 s 46 ms
Filter tests:		
Tests	Duration	Pixel_4_API_31
✓ Test Results	4 m 22 s	50/50
✓ CentralActivityTest	13 s	5/5
✓ testClickOnFilter	1 s	✓
✓ testClickOnStats	2 s	✓
✓ testClickOnWorkouts	2 s	✓
✓ testClickOnExercises	3 s	✓
✓ testClickOnSettings	2 s	✓
✓ ComeBackPopupTest	3 s	2/2
✓ testClickOnYesButton	2 s	✓
✓ testClickOnNoButton	1 s	✓
✓ DatabaseHelperTest	27 s	15/15
✓ testGetWorkoutsFromSnapshotEmpty	2 s	✓
✓ testGetExercisesFromSnapshotEmpty	1 s	✓
✓ testGetWorkoutsFromSnapshotFull	2 s	✓
✓ testGetAchievementsFromSnapshotEmpty	1 s	✓
✓ testGetDataPointsFromSnapshotEmpty	1 s	✓

Figure 3: Run Of UI Tests

Android Application Testing is a fundamental process to ensure the quality and reliability of applications designed for the Android platform. In this report, we will examine the key phases of testing:

- **Unit Testing:** This involves testing individual units of code, such as methods and functions, to ensure that they produce the expected results. These tests are usually automated and help to identify errors at a very granular level.
- **User Interface (UI) Testing:** It focuses on the user experience, ensuring that the user interface is intuitive and responsive across different screen.

Unit Testing

Unit tests were focused primarily on two classes, namely, HelperFunctions and DatabaseHelper. These classes were chosen due to their substantial logic and complexity within the application.

The DatabaseHelper class serves as an intermediary with the Firebase database, facilitating data retrieval for the application. It manages interactions with Firebase, encompassing data retrieval, transformation, and error handling. The goal of testing the methods within this class was to ensure the validity of data obtained from Firebase. These tests also verified accurate data processing and robust handling across various scenarios.

Specifically, successful data retrieval from Firebase was simulated, covering aspects like workouts, exercises, and user data. Mockito played a pivotal role in creating mock Firebase objects, including DataSnapshots. This approach allowed for the replication of Firebase database behavior during testing, granting control over data and the ability to test scenarios without involving a live database. It also enabled the examination of functions and methods interacting with the Firebase database without the necessity of reading or writing actual data during testing.

The HelperFunctions class encapsulates an array of utility functions aimed at supporting operations like data transformation, input parsing, and resource management. These functions are integral to the seamless operation of other components within the application. To ascertain their accuracy and reliability, unit tests were carried out.

Here's some examples among the most important Unit Tests:

- **testParseIntInput:** This test checks whether the parseIntInput method correctly converts a string into an integer. Three scenarios are tested: when the string represents a valid integer, when the string is empty, and when the string does not represent a valid integer.
- **testParseFloatInput:** This test checks whether the parseFloatInput method correctly converts a string into a floating-point number (float). Three scenarios similar to the previous test are tested.
- **testGetWorkoutType:** This test verifies the behavior of the getWorkoutType method, which returns a workout type based on a list of exercise types. Three assertions are performed with different combinations of exercise types to verify if the method returns the expected values.
- **testGetSerializableExtra:** This test checks whether the getSerializableExtra method correctly returns a serializable extra from an Intent object. An Intent with an extra value is created, and it is verified whether the method returns the correct value.
- **testSecondsToFormatString:** This test checks whether the secondsToFormatString method correctly converts seconds into a valid time format. Three assertions are performed to verify that the input values are converted correctly.

UI Testing

To evaluate the fundamental functionalities of the application, the Espresso framework was employed for the execution of User Interface (UI) tests. Espresso was chosen as the preferred framework for UI testing, given its status as one of the most popular and potent frameworks for Android app testing. Notably, Espresso provides a robust and intuitive API for simulating user interactions and validating app behavior.

Furthermore the use of Intents was fundamental to simulate the various app's activities in all their components, whereas Mockito was used to replicate the behaviour of some objects needed for throughout testing such as Firebase to simulate authentication in the SignIn Activity testing.

Interaction with the views within the app primarily relied on the usage of Espresso's `onView()` function. This function facilitates view selection based on attributes such as ID, text, or other customized attributes. Accurate view selection is vital for precisely identifying views and executing actions on them during testing.

Once a view was selected using `onView()`, Espresso enabled actions to be performed on it via `ViewActions`. These actions encompass a range of activities, including button clicks, text input, scrolling, and more. This versatility allowed the simulation of user interactions within the application.

`ViewMatchers` played a pivotal role in assessing the state of views and ensuring that the app adhered to expected behavior. These tools enabled the comparison of view properties with predetermined values. This verification phase was critical in confirming the correctness of the app's responses to user actions. For example checks on the visibility of menus were executed to test the correct change in status following interaction to open or close them, using `ViewMatcher.VISIBLE` when the menu had to be visible or `ViewMatcher.GONE` when the interaction closed the menu.

Other important function used in UI testing were the `.perform()` with `typeText()` functions, to simulate users' interactions, such as writing text when creating a new workout or logging in, or the use of `match()` to perform checks on specific lines of text or values, such as for testing the creation of a new workout.

UI tests were designed and executed to evaluate various aspects of the application's functionality, the most important ones being the following:

- To validate the accurate creation of custom workouts. This involved simulating interactions with the workout creation interface, including inputting workout details such as name, type, and associated music. The subsequent step involved verifying the correct display of the created workout within the workout list.
- To emulate workout tracking. These tests encompassed the initiation of a workout, display of the timer, and recording of workout data.
- To simulate the UX flow between various screens, making sure that each and every link to a new activity correctly changed the view.
- To verify the creation and use of an account by a user, whether in the sign up stage or in the login stage.

In essence, UI tests executed with Espresso, Intents and Mockito, with custom test code played a vital role in confirming the correct implementation of the app's core functionalities and the expected functionalities of the user interface to user interactions. These tests held significance in ensuring that the app delivered a reliable and error-free user experience.

Architectural Design

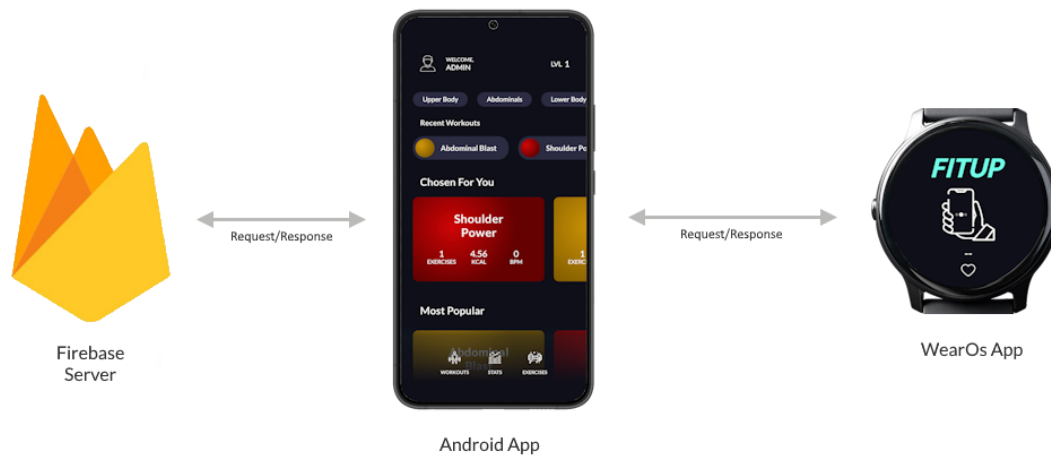


Figure 4: Overview of Fitup's Architecture

In this section, we provide an in-depth analysis of the Fitup fitness application's architectural structure. Our focus is to provide a comprehensive overview of the architectural framework that underpins the operational dynamics of Fitup. By delving into the core design elements, component interactions, and data flows, we aim to elucidate the methodical approach employed in establishing a robust and effective fitness tracking solution.

Overview

Fitup is designed to provide users with an intuitive app for tracking and managing their workout routines. To do this we employed a two tier architecture between our application and a Firebase server with an MVVM approach to enhance the features of our application with the inclusion of a WearOs companion app.

In particular the architecture involves seamless communication with Firebase for data storage and user authentication, along with the smartwarch app to extend the experience to wearable devices.

The Android application employs the MVVM (Model-View-ViewModel) architecture to ensure a clear separation of concerns, improved code organization, and enhanced testability. In this design pattern, the Model represents the data and business logic, retrieved from Firebase. The View is responsible for rendering the user interface elements, including workout data and session tracking. The ViewModel acts as an intermediary, facilitating communication between the Model and the View while also containing presentation logic. This architecture empowers developers to manage UI-related logic without compromising the scalability and maintainability of the codebase, promoting a streamlined development process.

This integration of the MVVM architecture enhances the overall quality of the application, making it more modular, maintainable, and adaptable to potential future changes and improvements.

MVVM offers several advantages for software development:

- **Separation of Concerns:** MVVM enforces a clear separation between the user interface and the application logic, leading to better code organization and maintainability.
- **Testability:** The ViewModel can be tested independently of the View, as it contains the application logic separate from the UI.
- **Reusability:** Models and ViewModels can often be reused across different UIs, improving code reusability.
- **Collaborative Development:** MVVM's clear separation of components allows different teams to work simultaneously on various parts of the application.

The Android app serves as the primary user interface for interacting with the workout features. It manages user inputs, displays workout data, and facilitates communication between the user and the Firebase server.

Firebase offers a cloud-based solution for storing workout-and-fitness-related data. It employs a NoSQL database structure to store user-specific information, including workout routines, progress, and settings.

In particular, the app communicates with Firebase's Realtime Database to retrieve and synchronize workout data. This encompasses the fetching of workout routines, tracking progress, and updating any relevant user-related information.

The Wear OS companion app complements the Android app by providing users with the ability to track their workouts directly from their wearable devices. This enables the users to use our app during a workout without the need to physically use their phone, which could be problematic in an environment such as a gym, while maintaining the most valuable information at their fingertips.

The companion app communicates with the Android app to fetch ongoing workout sessions, updating real-time progress.

Use Case View

In Figure 5, you can observe the use case view of the Fitup app. The Fitup application is a versatile solution tailored to meet a range of fitness tracking needs. From custom workout creation to personalized exercise management, the app empowers users to engage with their fitness routines comprehensively. Administrators enjoy direct access to crafting public content, ensuring a diverse array of fitness options. By seamlessly integrating user and administrator actions, along with real-time interaction through both Android and smartwatch apps, Fitup offers a holistic fitness experience.

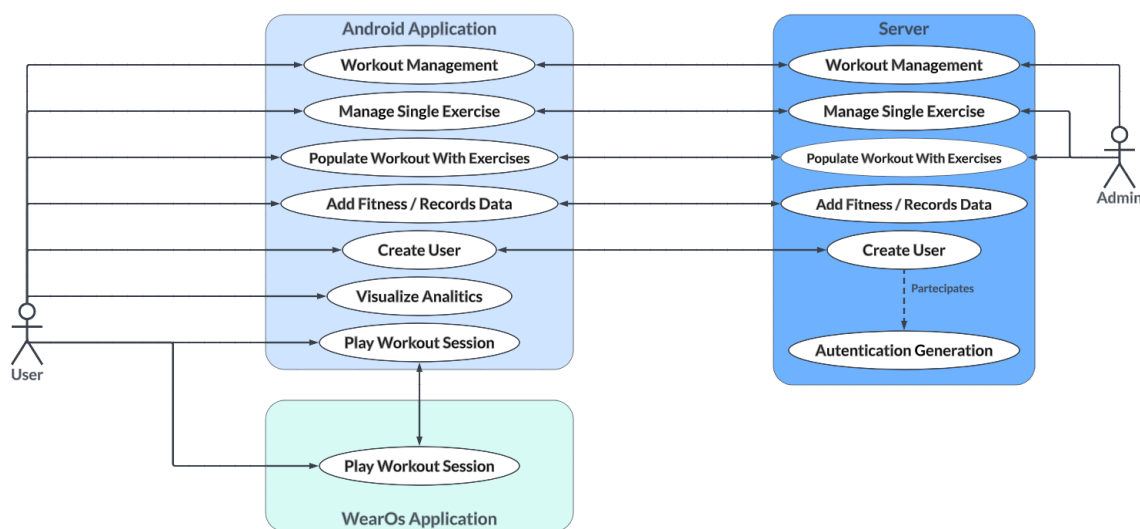


Figure 5: Use Case Diagram for Fitup

Here's a list of the use cases more in detail:

- **Effortless Account Creation:** Users can effortlessly establish accounts through conventional email/password credentials or choose streamlined access via Facebook or Google accounts. The Firebase server creates unique authentication credential along with the user profile
- **Custom Workout Management:** Users have the ability to create, edit, and delete custom workout routines according to their fitness preferences and objectives. Their workout are private and securely stored in the database.
- **Exercise Customization:** Users can create and remove individual exercises while assigning customized calorie values, enhancing accurate fitness progress monitoring. They can furthermore categorize the exercise by their type, whether it's an upper body

exercise or a yoga pose, to further enhance customization and the creation of workouts tailored to a user's goals.

- **Admin-Generated Public Content:** Administrators hold the privilege of generating public workouts and exercises, broadening the available fitness options for all users. Any user can access these workout or use the exercises in their own workout, making it possible to jump into a workout without any hassle, or creating custom workouts with few click. Users can't delete or modify any admin created data, ensuring the correct functionality of the app for everyone despite possible malicious actors.
- **Comprehensive Workout Creation:** Users can populate workouts with personalized or admin-provided exercises, specifying factors such as reps, sets, weights, reps buffer, and rest time. The app automatically updates these workouts with precise calorie expenditure data. Everything is automated to ensure ease of use and avoid unnecessary time waste for the user.
- **Fitness Data Tracking:** Users can log crucial fitness information, encompassing metrics like body weight, fat percentage, and personal records, facilitating meticulous self-monitoring and goal evaluation. They can also insert historical information on their data in order to not lose anything they already collected.
- **Data Analytics and Trends:** The app furnishes analytical insights into data trends over time, allowing users to make informed adjustments to their fitness routines based on precise progress assessments.
- **Dual-Interface Compatibility:** Fitup seamlessly interfaces with Android devices and smartwatches, enabling users to interact with their exercise routines through both the Android and smartwatch apps. This allows users to use our app during a workout in whatever way they prefer, whether through the more complete and engaging main android application, or through the hassle-free and compact smartwatch app on their wrist.
- **Real-Time Interaction:** During workout sessions, users can interact with their exercise routines using both the Android app and the smartwatch app, facilitating convenient real-time progress tracking and exercise visualization.

By intertwining these use cases, the Fitup fitness app, driven by the Android app, Wear OS app, and Firebase server, presents a comprehensive and adaptable solution that caters to the diverse fitness needs of users and administrators alike.

Deployment View

In Figure 6, you can observe the deployment of the entire Fitup system. The Fitup app operates on Android devices, while the backend is currently powered by Firebase, offering cloud-based services for data storage and user authentication. In the future, this should transition from Firebase to a custom backend server solution. This custom server will be hosted on dedicated infrastructure, ensuring greater control over data management and user interactions. For security and data privacy, communication between the client app and the server will be established using HTTPS, guaranteeing the integrity and confidentiality of information exchanged.

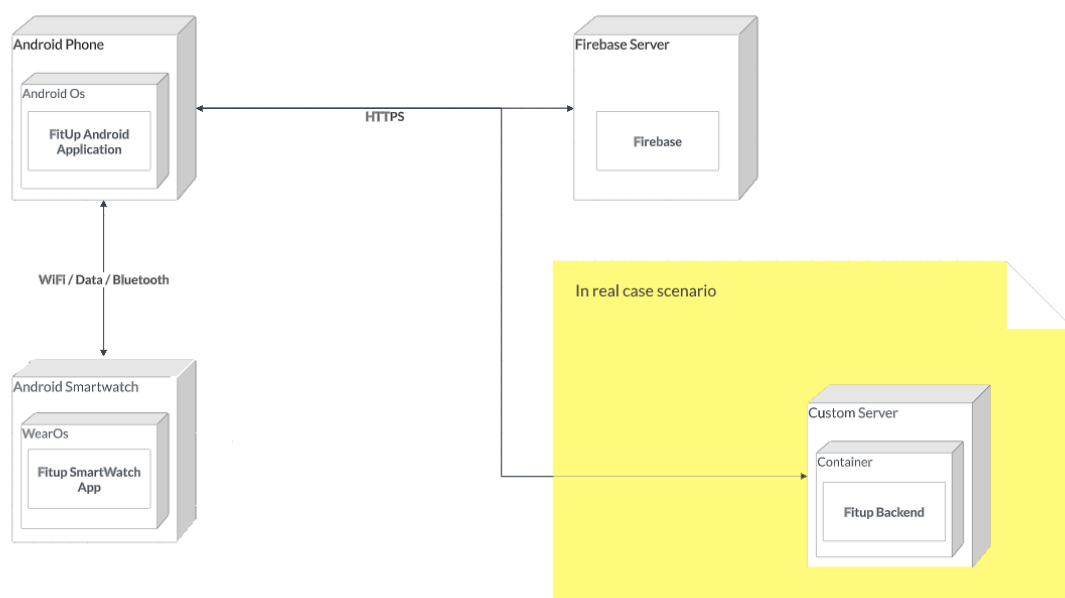


Figure 6: Deployment Diagram of Fitup

Components View

To explain how the components are organized internally, we present their connections in the UML component diagram shown in Figure 7.

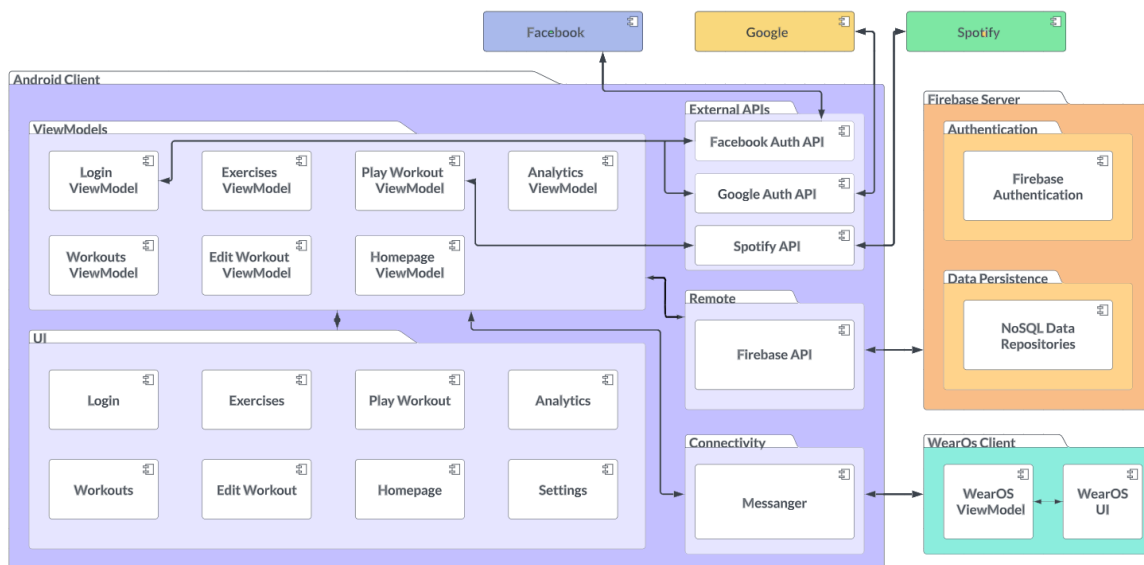


Figure 7: Component Diagram for Fitup

Android Client

- Login ViewModel:** The Login ViewModel manages user authentication during the login process. It exposes methods to handle user actions such as inputting credentials and initiating the login procedure. This ViewModel ensures secure login attempts, communicates with the authentication service, and validates user data. It collaborates with the Users Repository to authenticate users and maintain a seamless login experience.
- Exercises ViewModel:** The Exercises ViewModel facilitates the retrieval and management of exercise-related data. It provides methods to fetch exercises from the Exercise Repository and present them to the user interface. This ViewModel oversees the creation and deletion of exercises.
- Play Workout ViewModel:** The Play Workout ViewModel orchestrates the process of playing a selected workout. It interacts with the WorkoutExercises Repository to retrieve workout details, exercise sequences, and user preferences on exercises such as number of sets or rest time. This ViewModel manages the progression through exercises, records user input, and calculates workout progress via interaction with the WorkoutUserData Repository. It ensures a smooth and interactive workout experience for users.

- **Analytics ViewModel:** The Analytics ViewModel handles the generation and presentation of fitness-related data analytics. It accesses user data from the Graphs and Points Repository and computes trends and insights. This ViewModel prepares data for visualization, allowing users to monitor their progress and make informed decisions to optimize their fitness routines.
- **Workouts ViewModel:** The Workouts ViewModel oversees the management of workout information. It communicates with the Workout Repository to fetch, create, and destroy workout data. This ViewModel supports interactions such as viewing available workouts, selecting workouts for play or customization, and presenting workout details to users.
- **Edit Workout ViewModel:** The Edit Workout ViewModel enables users to modify existing workouts. It interacts with the WorkoutExercises Repository to retrieve workout details and associated exercises. This ViewModel facilitates the addition, removal, and customization of exercises within a workout, ensuring personalized and adaptable fitness routines.
- **Homepage ViewModel:** The Homepage ViewModel serves as the central control for the application's homepage or main screen. It coordinates the presentation of relevant content, including featured workouts, user recommendations, and navigation options. This ViewModel connects with various data sources, such as the Workout Repository and WorkoutUserData Repository, to curate a dynamic and engaging user experience and to tailor it to the user based on factors such as recent access and popularity.
- **Login:** The Login UI component provides users with an interface to initiate the authentication process. Users can input their credentials, such as email and password, to gain access to the app's features or use the convenient Facebook and Google login buttons. This component validates user input and communicates with the authentication service to verify the provided information. Successful authentication grants users access to their personalized fitness data and functionality.
- **Exercises:** The Exercises UI component offers users a platform to explore and manage exercise-related content. Users can browse through a variety of exercises, create new ones, and view detailed information about each exercise, such as its type.
- **Play Workout:** The Play Workout UI component delivers an immersive, gamified workout experience to users. It presents workout details, exercise sequences, and real-time progress indicators. Users can interact with this component to initiate, pause, and complete workout sessions. The Play Workout component offers an engaging interface that guides users through their fitness routines.
- **Analytics:** The Analytics UI component transforms fitness data into visual insights. It presents users with graphs, charts, and trends based on their workout history and performance metrics. This component empowers users to monitor their progress, identify trends, and make informed decisions to optimize their fitness strategies.

- **Workouts:** The Workouts UI component enables users to explore and manage available workout options. It showcases a list of workouts, provides essential details about each workout, and allows users to take actions such as selecting workouts for customization or play. The Workouts component facilitates the discovery of diverse fitness routines.
- **Edit Workout:** The Edit Workout UI component empowers users to modify existing workouts according to their preferences. It offers an intuitive interface for adding, removing, and customizing exercises within a workout. Users can fine-tune workouts to align with their evolving fitness goals and preferences.
- **Homepage:** The Homepage UI component serves as the app's main landing page. It presents users with featured content, personalized recommendations, and navigation options. This component provides users with a starting point to explore the app's offerings.
- **Settings:** The Settings UI component offers users a dedicated space to personalize application settings. It enables users to customize preferences, including language preference. The Settings component enhances user satisfaction by allowing individuals to tailor the app's behavior to suit their preferences and needs.
- **FireBase API:** The Firebase API serves as the communication bridge between the fitness app and the Firebase backend. It facilitates data exchange, authentication, and storage operations, ensuring seamless interaction between the app's frontend and the backend infrastructure.
- **Messenger:** The Messenger component enables communication and notifications with the WearOs companion. It manages the delivery of messages and updates to and from the smartwatch, enhancing user engagement and keeping them informed about their workout no matter the device they are using.

External APIs

- **Facebook Auth API:** The Facebook Auth API offers a seamless way for users to authenticate within the fitness app using their Facebook credentials. By leveraging the widely recognized authentication mechanism of Facebook, users can sign up and log in without the need to create new usernames and passwords.
- **Google Auth API:** The Google Auth API enhances user authentication within the fitness app by allowing users to log in using their Google accounts. This API streamlines the user experience by providing a familiar and secure authentication process through Google's authentication services.
- **Spotify API:** The integration of the Spotify API elevates the fitness app's user experience by seamlessly integrating new music into the workout environment. Users can connect their Spotify accounts to the app, enabling them to play playlist that adapt to their chosen workout.

Firestore Server

Firestore Authentication: The Firestore Authentication component within the Firestore Server serves as a robust and secure solution for user identity and access management. It handles user registration, login, and authentication processes, ensuring that only authorized users can access the fitness app and personal data. This component employs encryption and industry-standard security practices to safeguard user credentials and sensitive data during transmission. By integrating Firestore Authentication, the fitness app benefits from a reliable and scalable authentication mechanism, enhancing user data protection and overall application security.

NoSQL and Data Models

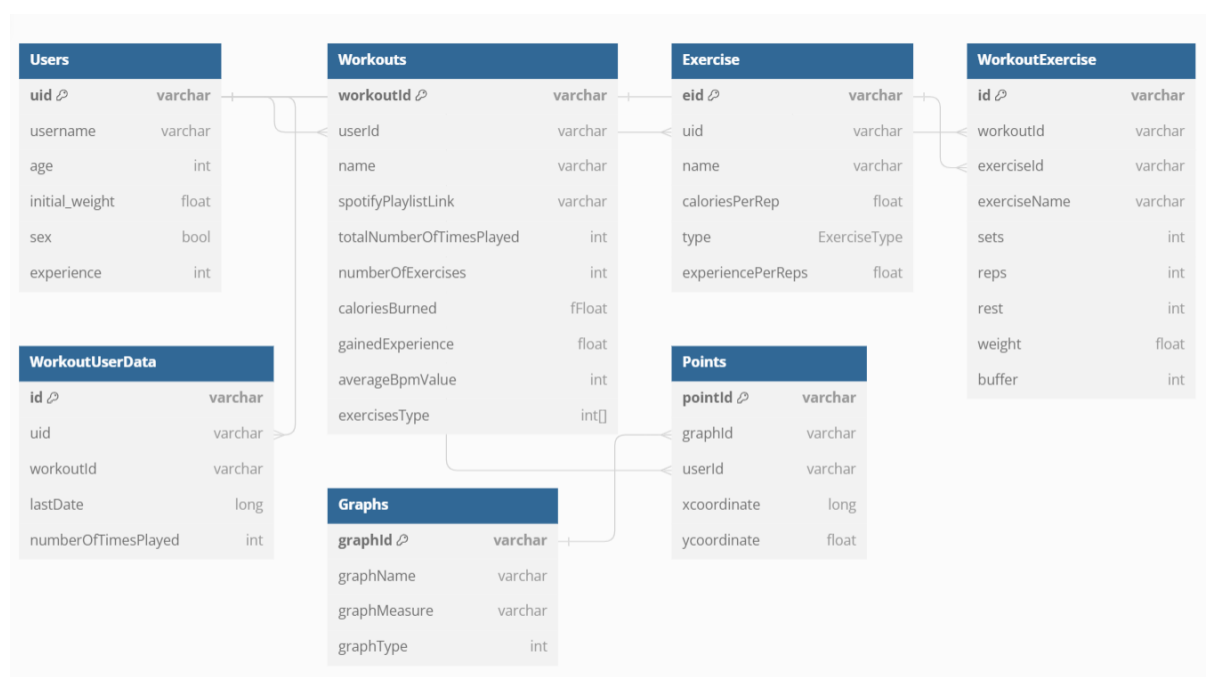


Figure 8: Data Schemas inside Fitup

The provided NoSQL database schema is intricately tailored to efficiently manage data within a fitness tracking application. This schema embraces the versatility of NoSQL databases while accommodating fitness-specific attributes and relationships. Core tables include "Users," capturing essential user details, "Workouts," storing workout metadata, and "Exercises," recording exercise specifics such as calories burned. The "WorkoutExercise" table bridges workouts and exercises, enabling tailored routines, while "WorkoutUserData" maintains user-specific workout histories.

Moreover, the schema accommodates data visualization with "Graphs" for varied graph types, and "Points" for data points, fostering a holistic fitness perspective. NoSQL's flexibility aligns with the dynamic fitness landscape, supporting semi-structured data and scalability. Furthermore, this schema's attributes and relationships could be translated to a relational SQL

database if required. However, NoSQL's straightforward querying and adaptability suit the fitness app's complexities, ensuring efficient data access and enhancing user fitness experiences.

WearOs Client

- **WearOs ViewModel:** The WearOs ViewModel plays a pivotal role in facilitating communication and data flow between the Wear OS app's user interface and the main application. It handles interactions specific to the wearable device, such as tracking workout chronometers, displaying current exercise details, and managing user inputs to navigate a workout. This ViewModel ensures seamless synchronization between the Wear OS app and the primary Android app, providing users with a consistent and efficient fitness experience across devices.
- **WearOs UI:** The WearOs UI component is tailored to the unique form factor of wearable devices like smartwatches. It offers users a compact and glanceable interface to interact with their workouts directly from their wrist. This component presents essential workout details, exercise progress, and interactive controls optimized for the wearable environment. The WearOs UI enhances user convenience by minimizing the need to frequently access the smartphone app, promoting a more seamless and comfortable fitness tracking experience.

Sequence Views

The sequence diagrams presented in this section encapsulate the most important interactions of users within the Fitup fitness application. These diagrams illustrate the step-by-step interactions that unfold as users interact with the app's core features. For instance, the sequence diagram detailing the process of a user creating a new workout provides an insightful journey through the user's actions, the app's responses, and the corresponding communication with the server.

Sequence Diagram - Login

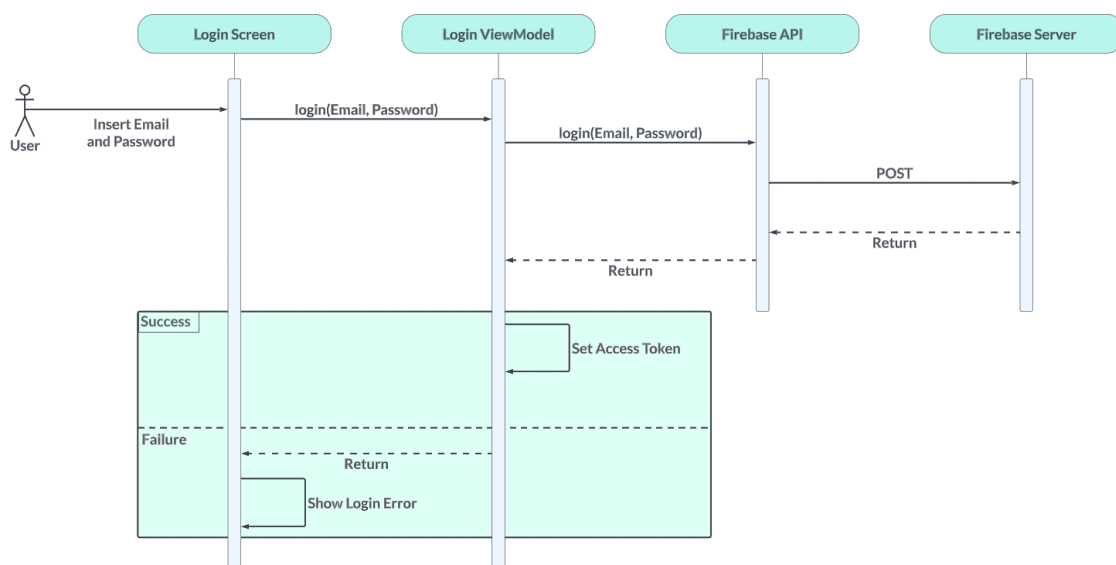


Figure 9: Login Sequence Diagram

This sequence diagram provides a detailed view of the login process within the Fitup app. It illustrates how user authentication is managed seamlessly. When a user inputs their email and password, the app initiates the login procedure. Subsequently, the login data is transmitted to Firebase Authentication using the Firebase API. The Firebase Authentication service processes the login request and verifies the user's credentials against the stored user data. If the authentication is successful, the system generates an authentication token, signifying the user's valid access. This token is then securely managed by the app, allowing the user to interact with authenticated features and data. Conversely, if the authentication process encounters an error, an appropriate error message is generated and sent back to the app, ensuring the user is promptly informed about the issue.

Sequence Diagram - Edit Workout

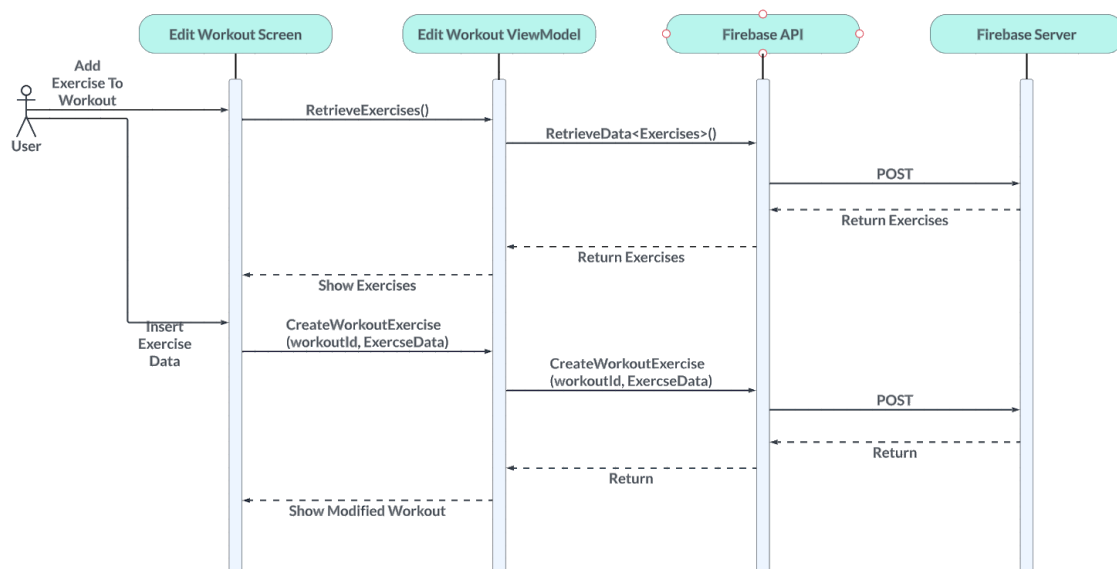


Figure 10: Edit Workout Sequence Diagram

This sequence diagram delves into the intricacies of editing a workout within the Fitup app. When a user opts to modify a workout, the app navigates to the corresponding workout screen, displaying the existing exercises associated with that workout. To facilitate exercise selection and customization, the app leverages the Firebase API to fetch up-to-date data about available exercises. The user interacts with the interface to select desired exercises and provides comprehensive details, including sets, reps, rest, weight, and buffer for each exercise. Upon the user's confirmation, the app marshals this new data and sends it back to Firebase. The Firebase server processes the incoming data, updating the workout structure with the user's modifications. Upon successful completion of this process, the app promptly refreshes the workout view to reflect the changes, including the newly added exercise and its corresponding details.

Sequence Diagram - Play Workout

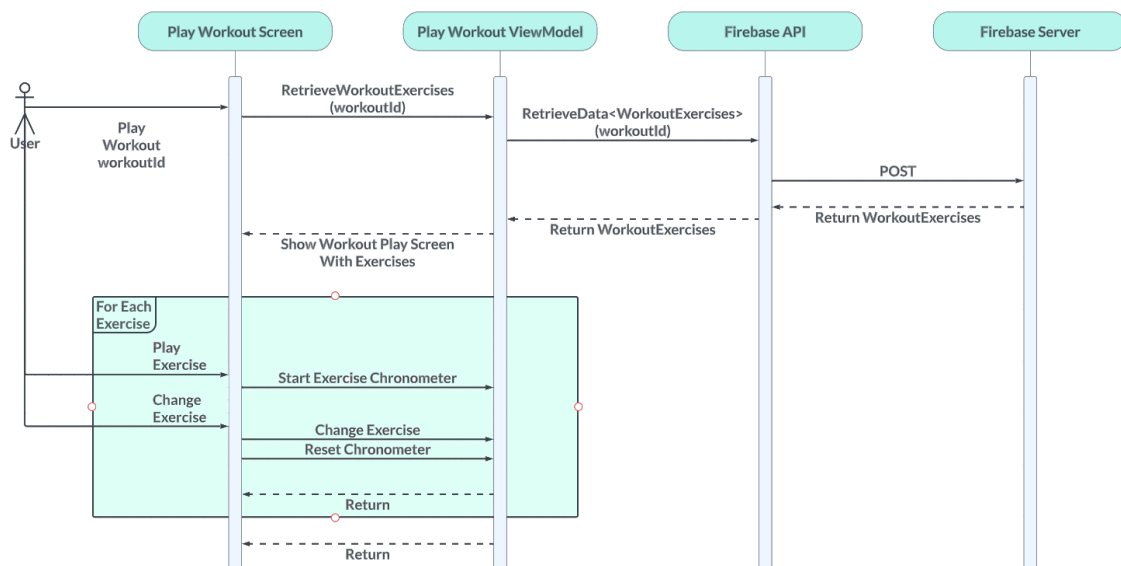


Figure 11: Play Workout Sequence Diagram

This sequence diagram provides an in-depth look into the journey of playing a workout within the Fitup app. Initiated by the user's desire to begin a workout session, the app's "Play Workout" activity comes into play. The activity communicates with the Firebase server through the Firebase API to obtain the comprehensive workout data, which includes exercise information, such as sets and other relevant details. With the arrival of this data, the app transitions to the workout play interface, presenting the user with a structured environment to navigate through their exercises. As the user progresses through the exercises, they can initiate each exercise, proceed to the next exercise at their discretion, and keep track of their overall workout time. Upon the completion of the final exercise, the app redirects the user to a workout recap, summarizing their session's achievements and encouraging a sense of accomplishment.

Future Server Replacement

For enhanced customization, the current Firebase server, utilized for demonstration, should be substituted with a dedicated backend server in a real-world application scenario.

This backend server will expose RESTs APIs catering to user authentication and workout data management. Both the Android app and Wear OS app will interact with these APIs, like their interaction with Firebase but with enhanced functionalities. The server infrastructure will encompass robust authentication mechanisms, adaptable database systems, and stringent security measures to guarantee data integrity and user privacy.

Benefits of Server Replacement

- **Advanced Security:** Custom server setups allow for tighter security configurations and better safeguarding of user data, when it comes to input sanitization.
- **Customization:** A dedicated server permits the incorporation of enhanced features tailored to the application's unique requirements.

Gamification

Introduction to Gamification

Gamification, a concept that fuses the allure of games with everyday activities, has swept through industries, redefining engagement and motivation. At its core, gamification infuses elements like challenges, rewards, and competition into non-game scenarios to trigger enthusiasm and commitment. In recent years, this innovative approach has permeated various sectors, showing particular promise in promoting healthier lifestyles. A notable example is Nintendo's "Ring Fit Adventure," a video game that ingeniously merges physical workouts with a captivating gaming narrative and a fun, fitness-focused rpg-like gameplay. This blend exemplifies how gamification can revolutionize fitness, transforming exercise from a chore into a thrilling adventure. As we embark on our fitness app's gamified journey, let's delve into the profound impact this approach can have on your pursuit of well-being.

Gamification in Fitup

Fitup tackles gamification in fitness in two main ways. Firstly the app rewards users with experience points each time they complete a workout and this translates into the user gaining levels and thus increased satisfaction. Secondly, and most importantly, the workout itself is presented to the user as an interactive story in which the user acts as the hero that needs to defeat the monster threatening to destroy the kingdom: exercises are steppingstones to reach the monster, and framed as the necessary training the hero must complete to be able to defeat the monster. This creates an engaging experience that stimulates the user to complete each exercise, not only for the delayed gratification fitness is known to give, but also for the instant gratification of the game, that makes the gamified approach to fitness so strong.

Furthermore our approach to gamified fitness doesn't rely on punishing the player, for example by having streaks the player can lose if they don't workout: this was done considering how fitness can be a challenging journey to take and using punishing mechanics, although common and appreciated in other gamified experience, would not make sense in this context.

Advantages of Gamification in Fitness

Incorporating gamification principles into fitness regimens brings forth a range of compelling advantages that elevate the entire experience:

- **Enhanced Motivation:** By intertwining rewards, achievements, and challenges into workouts, we cultivate a dynamic atmosphere that propels motivation. As you conquer each session, the promise of tangible rewards and personal triumphs fuels the drive for consistent progress.

- **Increased Engagement:** Interactivity lies at the heart of gamification. Through our gamified approach, workouts transcend mere routines. Interactive elements seamlessly weave into exercises, adding an enjoyable layer of engagement that keeps the user looking forward to each session.
- **Goal Achievement:** Setting and achieving goals is a cornerstone of any fitness journey. Our gamification strategy amplifies this process, offering milestones to conquer. Each accomplishment fuels a sense of achievement that further encourages dedication.
- **Long-Term Adherence:** The journey towards health and fitness is a marathon, not a sprint. Our gamified workouts are designed to instill a sense of fun and fulfillment in the user's routine. This, in turn, nurtures a habit-forming experience, ensuring commitment to well-being over the long term.

Future Steps

In the future the gamified approach in Fitup could benefit from new additions, the most important ones are listed here:

- **Social Interactions:** Incorporating social-network-like features to engage the users would be one of the most important steps in supporting the gamified aspects in Fitup: having users rival their friends and social circles in fitness challenges, using leaderboards and even just being public about one's fitness journey is proven to be an effective method when the objective is to maximise the user's engagement. That being said, there needs to be a careful calibration to avoid transforming this feature into something that users see as negative and that brings bad emotions.
- **Achievements:** The introduction of an achievement system adds a layer of personal accomplishment to a user's fitness journey. Unlocking achievements provides a tangible representation of progress and dedication. As the user's conquer new feats and surpass personal bests, the sense of pride and satisfaction boosts motivation, turning every workout into a interactive step toward their goals.
- **Daily/Weekly/Monthly Challenges:** The integration of diverse challenges at different intervals adds an exciting dimension to users' routines. Daily, weekly, and monthly challenges introduce variety and anticipation to workouts, ensuring that monotony is nowhere in sight. These challenges encourage users to explore new exercises, push their limits, and continuously engage with the app, creating a dynamic experience that transcends the traditional fitness routine.
- **Rewards:** The inclusion of rewards serves as a powerful incentive, uniting users' effort with meaningful outcomes. By earning rewards through their engagement and achievements, users are tangibly acknowledged for their dedication. These rewards, whether they're virtual badges, discounts, or other incentives, not only reinforce users' commitment but also hold the potential to pave the way for future enhancements. Furthermore, rewards like discounts or even the possibility of earning free fitness accessories could be the foundation of a potential in-brand store that offers these items, further expanding the Fitup business. This imaginative prospect adds an extra layer of excitement to users' fitness journey, while also opening the door for the app's evolution. Each reward becomes a marker of their progress, celebrating their journey towards better health and well-being.

Custom Layouts and Views

User interfaces are the frontline of interaction between users and applications. While standard UI components provide a baseline, the diverse requirements of our fitness app demanded a more tailored approach. The utilization of custom layouts and views emerged as a response to these demands, enabling us to sculpt an interface optimized for both functionality and user experience.

The necessity for custom layouts and views arises from the need to harmonize complex features and present data in novel ways. This chapter delves into the description of these components, outlining their role within the app's architecture. By not relying exclusively on off-the-shelf solutions, we have engineered an interface that reflects the app's character and enhances its usability.

Each of these elements fulfils a specific purpose, ranging from visualizing data to accommodating complex interactions. Through the exploration of each custom entity, we will discuss the underlying design decisions and implementation intricacies.

Custom Layouts

LineGraphLayout

LineGraphLayout is a custom layout used to generate beautiful, rich, and engaging line graphs for our application. User can register their data on body measurements and personal fitness records through the app's UI. This data is then aggregated and provided to the custom layout to be visualized.

This is done by using various draw functions. The most important one is used to visualize the trend in the graph data by connecting the various data points using straight lines. The beauty of the visualized data is then further accentuated by two distinct elements: the first one is the semi-transparent area underneath the graph, that helps the screen feel more full and provides an instant feedback to the user; the second are the data points themselves, that emerge from the background thanks to their border. To complete the ensemble, the dotted line augments the three-dimensional feeling of the layout and the measures on the right side provide immediate information to the user.

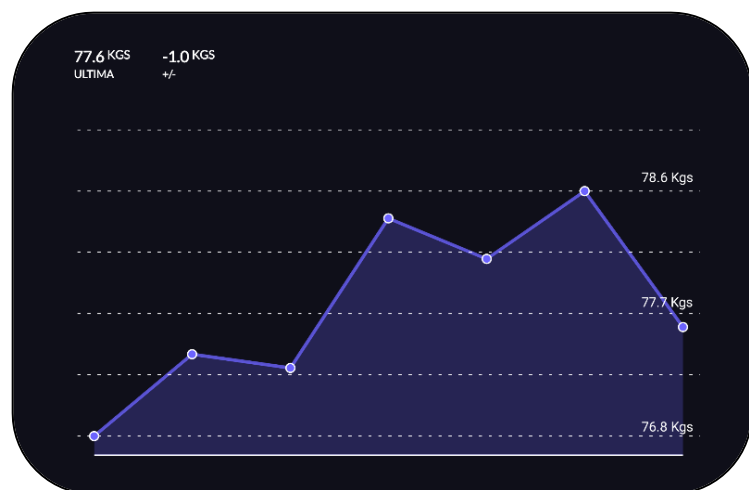


Figure 12: Analytics Custom Layout

Lastly, each data point in the graph is clickable and interacting with it reveals a custom view, the `MinimizedExerciseView` discussed later.

WorkoutLayout

`WorkoutLayout` is a custom layout that provides a gamified experience of their workouts to the users. This is done by reimagining the workout as a quest the user must undertake to defeat the villain (a monster in our case) to save the realm: each exercise is thus a steppingstone in the path and completing it brings the user one step closer to the objective.

The custom layout thus was needed to visualize the path in a way that was intuitive and pleasing to the user, while at the same time being informative enough with icons and colours to deliver the type of the exercise to the user.

The layout is a collection of `MinimizedExerciseView` views, discussed later, and a series of dotted line connecting them.

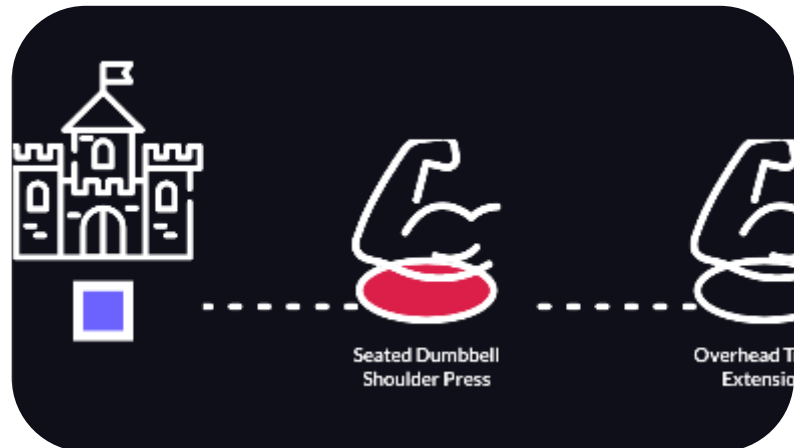


Figure 13: Workout Adventure Custom Layout

Custom Views



Figure 14: Countdown Custom View

CircularCountdownView

`CircularCountdownView` is a simple view to visualize a circular countdown. The necessary calculations for the countdown are done elsewhere and the view is responsible only of updating the annulus with the data received from the activity,

In particular the annulus is obtained by drawing two circles: the first one, is the internal one, that is never updated and is used to mask anything below it, thus giving the feeling the middle is empty; the second one is instead the one responsible for the actual visualization

of the countdown, and it is simply achieved by continuously redrawing the external circle with different sweep angles by invalidating it at a constant rate.

LineGraphDataView

LineGraphDataView is a simple data visualization view used in the LineGraphLayout layout. Its main purpose is that of visualizing the corresponding data value when a user taps one of the data points. This view consists simply of a circle and a text visualizing the data. After a few seconds from when the point gets clicked and this view visualized, the view automatically disappears.

MinimizedExerciseView

The MinimizedExerciseView view corresponds to one exercise in the WorkoutLayout layout and is mostly used visualize a step in the path the user must take in our gamified version of a workout.

ThreeDotsLoadingView

The ThreeDotsLoadingView is a WearOs-only custom view that simulates a three-dots-waiting animation. It was used to signal the user the need to use the smartphone application to start a workout without the need of long TextViews but only relying on visual cues, thus enhancing the immediacy of our application, especially considering the difficulties encountered when visualizing text on the smaller screens of smartwatches.

User Interface Design

User interface design stands as a bridge between technology and human interaction. Our fitness app's strength lies not only on its underlying functionality but also on its ability to engage and guide users toward their fitness goals. The User Interface (UI) design, therefore, emerges as a critical component, shaping how users perceive, interact with, and derive value from the application.

Our approach focuses on the union of user-centred design methodologies and innovative visual aesthetics. By integrating these components, we have crafted an interface that is not only visually compelling but also intuitively navigable, fostering a sense of empowerment and accomplishment for the user.

From colour schemes that evoke emotion to typography choices that enhance readability, every element has been meticulously curated to create a cohesive and captivating user experience.

Principles and Guidelines

At the core of our fitness app's design philosophy, a set of principles and guidelines build an immersive and intuitive user experience. Primarily, we recognize that images possess the power to convey information swiftly and effectively, surpassing the limitations of text. As such, our design choices prioritize imagery, maximizing the impact of every interaction.

We focused on clarity and immediacy: each screen is organized to ensure that essential information is readily accessible, enabling users to navigate the app effortlessly and engage with their fitness goals directly. This design aligns with the pursuit of cleanliness and a modern, minimalistic, aesthetic, cultivating an interface that feels sleek, contemporary, and aligned with current trends. By stripping away the unnecessary, we created an interface that is not only visually appealing but also efficient in its functionality.

Furthermore, we harnessed the psychology of colour to establish a mental connection between exercises and their associated hue, aiding in exercise recall and creating a visual pattern that helps the user when using our application.

Big, short bold text is another important piece of our design. By emphasizing key information, we ensure that users quickly grasp key details, eliminating confusion. Coupled with our minimalistic approach, this bold text acts as a navigational guide, directing users' attention to principal elements.

Lastly, our user interface makes heavy use of icons. These recognizable symbols facilitate navigation and interaction. The placement of icons alongside textual elements optimizes both aesthetics and functionality, offering users a familiar and cohesive experience.

Style and Visual Identity

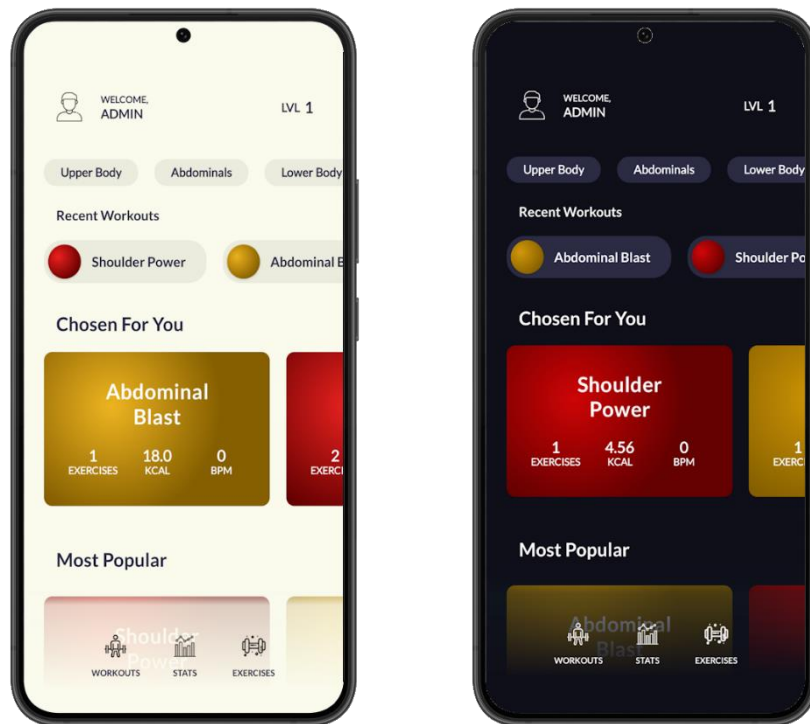


Figure 15: Light and Dark Theme for Fitup

Central to the app's identity are the dynamic day and night themes that adapt seamlessly to users' surroundings and preferences, enhancing usability and comfort. These themes not only ensure optimal visibility but also underscore our commitment to tailoring the user experience.

Beyond themes, our visual identity bursts forth with vibrant colours that energize the app's landscape. These hues infuse vitality into every interaction, creating a connection between users and their fitness pursuits.

Colour Schemes



Figure 16: Fitup Color Schemes

Colour is a powerful tool that transcends mere aesthetics. Our fitness app's colour scheme is more than a visual embellishment; it is a strategic component aimed at elevating user experience and interaction. Using both day and night themes, we have designed an assorted colour palette that embody contrast and vibrancy, bringing both practicality and emotion to our application.

Creating a contrast-rich environment within our app has been a guiding principle. The day theme is characterized by calm hues, enabling users to seamlessly navigate and engage with the interface. The night theme adopts subdued and darker tones that promote relaxation and ease of use during low-light conditions.

The integration of bold accent colours acts as a beacon, directing users' attention to critical elements and actions. These accents inject vitality into the app, enabling users to use interactive components better and navigate effortlessly through various sections.

Furthermore, a universal colour intertwines both versatility and harmony. While the palette encompasses an extensive range of hues, each colour is tempered to maintain a minimalistic touch, often akin to pastel shades. This decision augments the visual appeal of the app.

Despite the profusion of colours, they blend seamlessly, giving rise to a harmonious interface. Each shade possesses cohesiveness. This ensures that even with a wide array of visual elements, the overall experience remains unified and soothing to the user's eye.

Our colour scheme serves as a guiding light, ensuring that our fitness app caters to both form and function. The strategic use of contrasts, the vitality of accent colours, and the delicate balance of the universal palette culminate in an interface that speaks to users on both practical and emotional levels.

Typography

Our fitness app embraces the Lato font for clear, legible, and harmonious communication. Lato's clean lines and versatile weights empower us to present information with finesse, ensuring that users experience both the content's meaning and the aesthetic appeal.

The Lato font's balance of elegance and functionality represents our commitment to user-centric design. Its carefully chosen weights and styles enhance readability, making it ideal for information that is both inviting and intelligible.



Figure 17: Fitup Main Font

Iconography

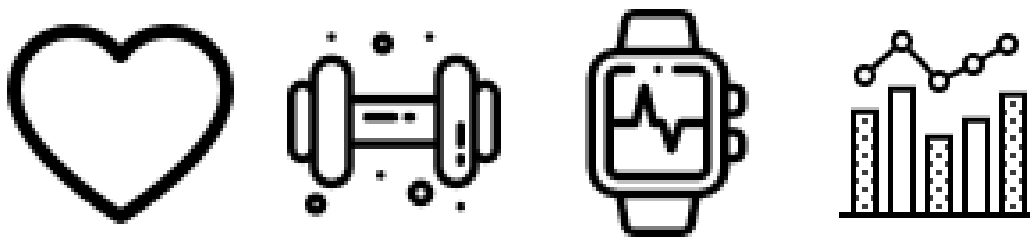


Figure 18: Some of Fitup's Icons

Icons transcend language barriers. Our fitness app's iconography guides users through the app's interface with clarity and efficiency. Each icon encapsulates actions, categories, and concepts, creating a bridge between user intent and app response.

Our iconography design aligns with the broader UI strategy, embracing simplicity and visual coherence. Each icon was designed to be recognizable and comprehensible.

Mobile App UI

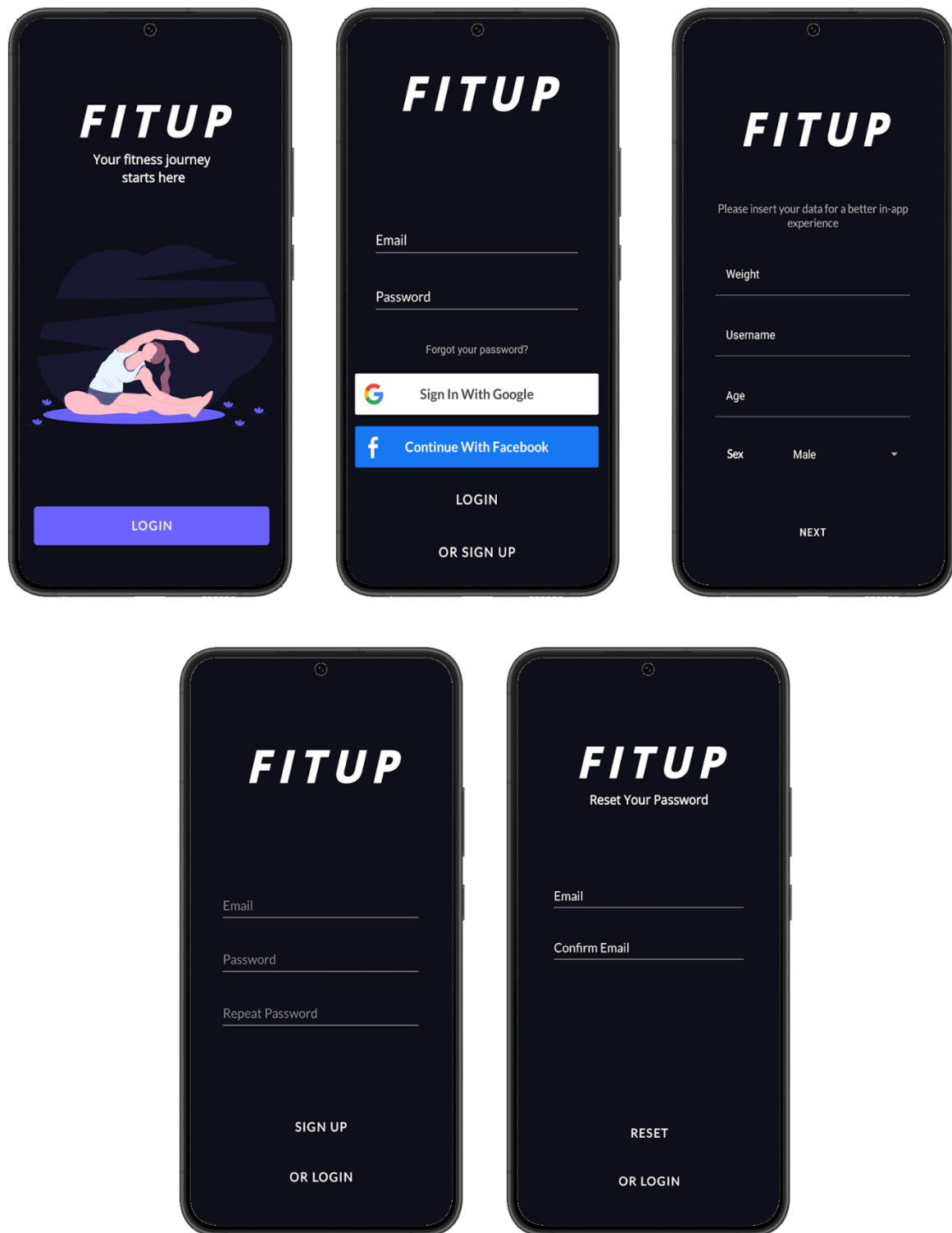


Figure 19: Enter, Login, Signup and Password Reset Screens

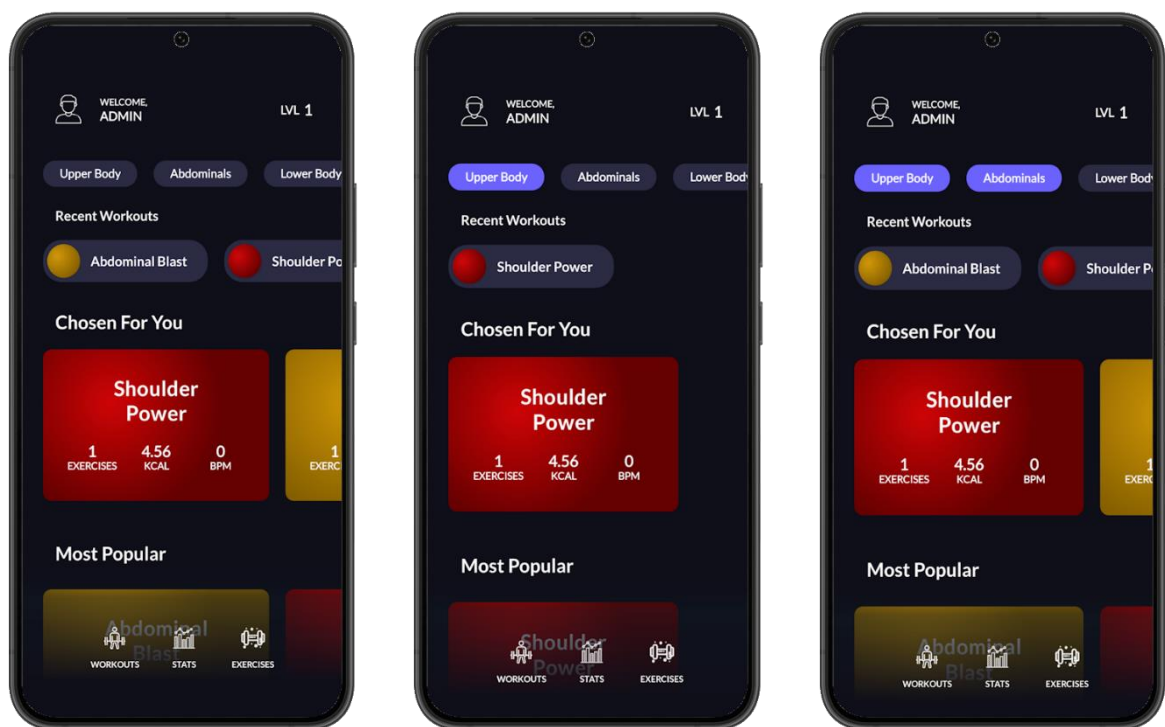


Figure 20: Home Page with and without Filters Active

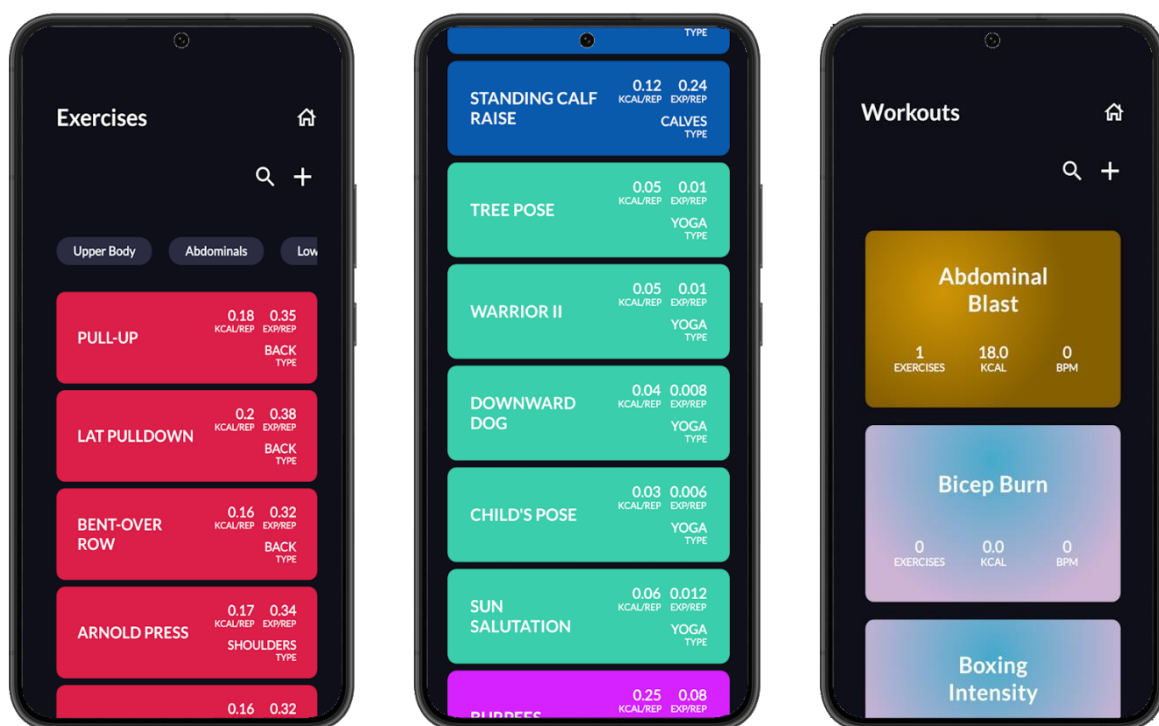


Figure 21: Exercises and Workout Lists



Figure 22: Analytics Screens

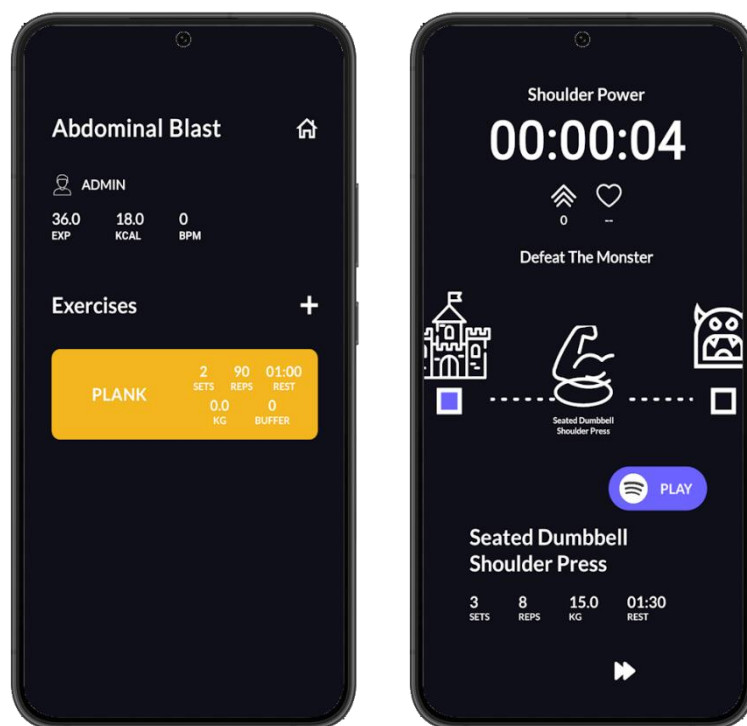


Figure 23: Screens of Editable Workout and Play Workout

WearOs App UI



Figure 24: Mockups of the WearOs App

User Experience Diagrams

Here it is possible to see the UX diagram of the key features of the Android app.

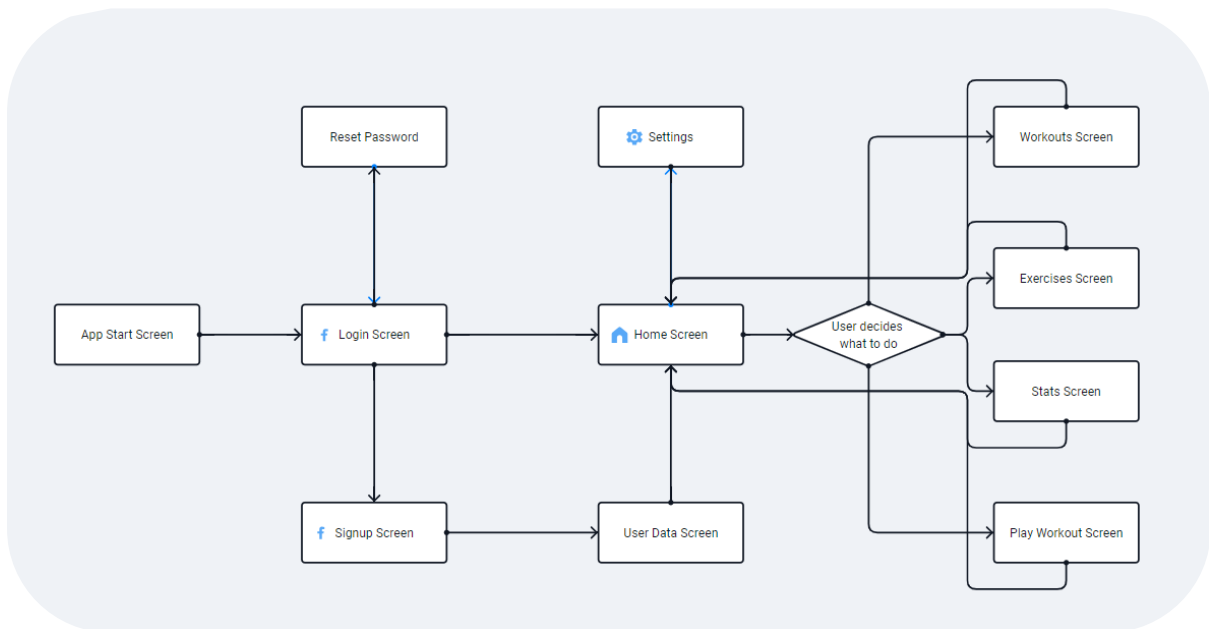


Figure 25: Diagram of the App's UX Main Flow

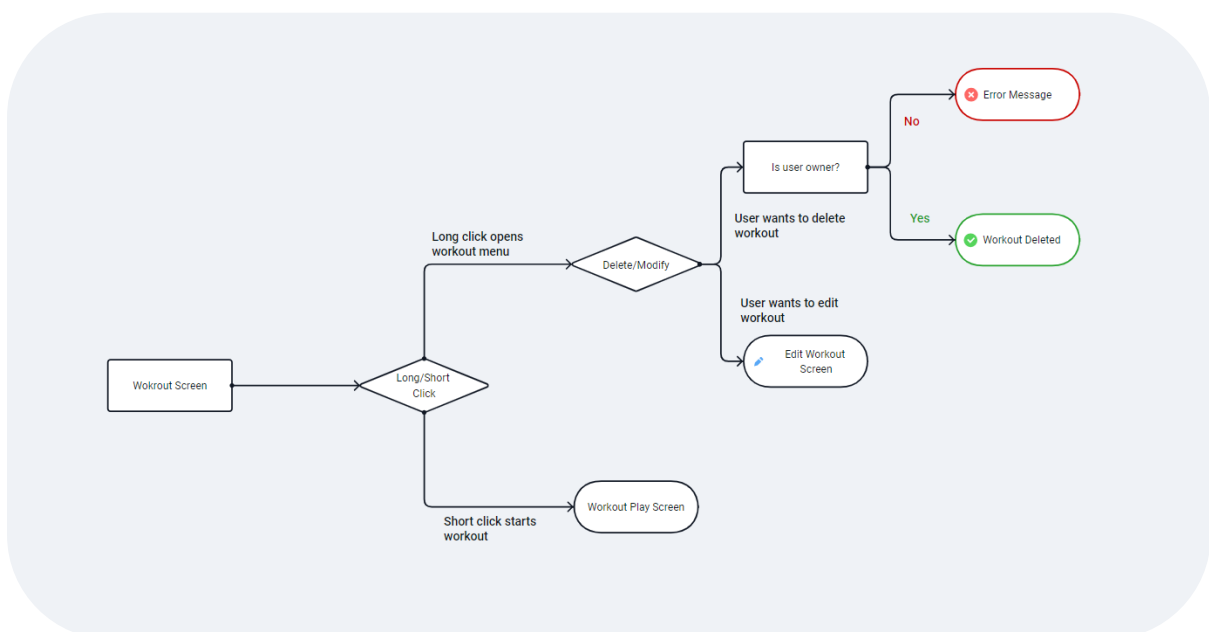


Figure 26: Diagram of Workout List UX with Possible Actions

References

- [Introduction to Android](#)
- [Glossary](#)
- [User Interface](#)
- [Firebase Authentication](#)
- [Firebase for Android](#)
- [Firebase Realtime Database](#)
- [App Navigation](#)
- [Spotify Web API](#)
- [App Architecture](#)
- [Android Components](#)
- [Building a Dynamic UI with Fragments](#)
- [Custom Views](#)
- [Design Guidelines](#)
- [Designing for Multiple Screens](#)
- [User Interface Design](#)

Table Of Figures

Figure 1: Sequence Diagram of Play Workout	13
Figure 2: Sequence Diagram Messages	14
Figure 3: Run Of UI Tests	15
Figure 4: Overview of Fitup's Architecture.....	19
Figure 5: Use Case Diagram for Fitup	21
Figure 6: Deployment Diagram of Fitup.....	23
Figure 7: Component Diagram for Fitup	24
Figure 8: Data Schemas inside Fitup.....	27
Figure 9: Login Sequence Diagram.....	30
Figure 10: Edit Workout Sequence Diagram	31
Figure 11: Play Workout Sequence Diagram.....	32
Figure 12: Analytics Custom Layout.....	36
Figure 13: Workout Adventure Custom Layout.....	37
Figure 14: Countdown Custom View	37
Figure 15: Light and Dark Theme for Fitup	40
Figure 16: Fitup Color Schemes	41
Figure 17: Fitup Main Font.....	42
Figure 18: Some of Fitup's Icons	42
Figure 19: Enter, Login, Signup and Password Reset Screens.....	43
Figure 20: Home Page with and without Filters Active	44
Figure 21: Exercises and Workout Lists	45
Figure 22: Analytics Screens	46
Figure 23: Screens of Editable Workout and Play Workout.....	47
Figure 24: Mockups of the WearOs App.....	48
Figure 25: Diagram of the App's UX Main Flow	49
Figure 26: Diagram of Workout List UX with Possible Actions.....	49