Flutter Gym App Documentation WIP

Cole Westerveld

Table Of Contents

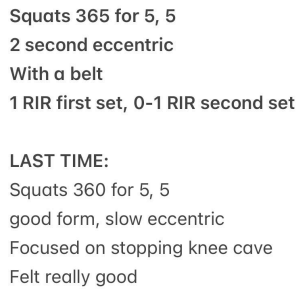
Project Overview

Motivation (Backstory of Project Idea)

When it comes to working out, visual changes come slowly, and not really noticeable workout-to-workout (despite them being noticeable long term). But at the same time, without some way to measure changes, it's hard to know if what you're doing is working without having to wait 5 years to realise you look the same and are just as strong. That's why I think tracking is one of the most important parts of reaching a fitness goal (or any goal really).

It's motivating and re-assuring to be able to clearly see that the weight you could barely lift for 6 reps last time - you just did for 7, or even just 6 and a half. These changes are small, and hard to track without some way to write them down.

For most of my three years (as of writing this) of lifting, I tracked my exercises in a long note on my phone - my sets, reps, machine settings, how I was feeling that day, and more.



example snippet of a log from my leg day

The problem was, this note became very long, it became impractical to save data past 2 workouts ago, and sometimes hard to find information. I also had a separate spreadsheet to track progress, which I had to manually populate from this note, which was exhausting and as a student I don't really have time to do that.

I found some apps which did some of this for me, and work fairly similarly to how my app works. With all of them, though, there was still something to be desired. Notifications right before my workout is scheduled so that I don't forget to bring my belt when I have squats planned, a good way to schedule, easy program editing, good and customizable analytics, or a simple and easy to use interface. Plus, some of them were paid for many of the features that they did include. I'm always looking for an excuse to build a new project anyways, and so I decided to build this app and get some experience with mobile dev. I plan to put this on the app store and play store for mobile when it's ready (it feels like it's never going to be ready), and keep the code free and open source on my GitHub.

Techstack

I came across **Flutter** when looking for how an app might be built, and I found the code structure easy to pick up, and the cross-platform functionality is really nice. Overall, I am glad I picked Flutter over other frameworks like React Native - the tooling is unmatched, Dart is a really nice language (>> JS), and the widget library is vast.

With Flutter I am using the **Provider** library for state management, **SQFLite** for the database, and various other libraries for UI components such as fl_charts for the graphs and table_calendar for the schedule calendar.

Features

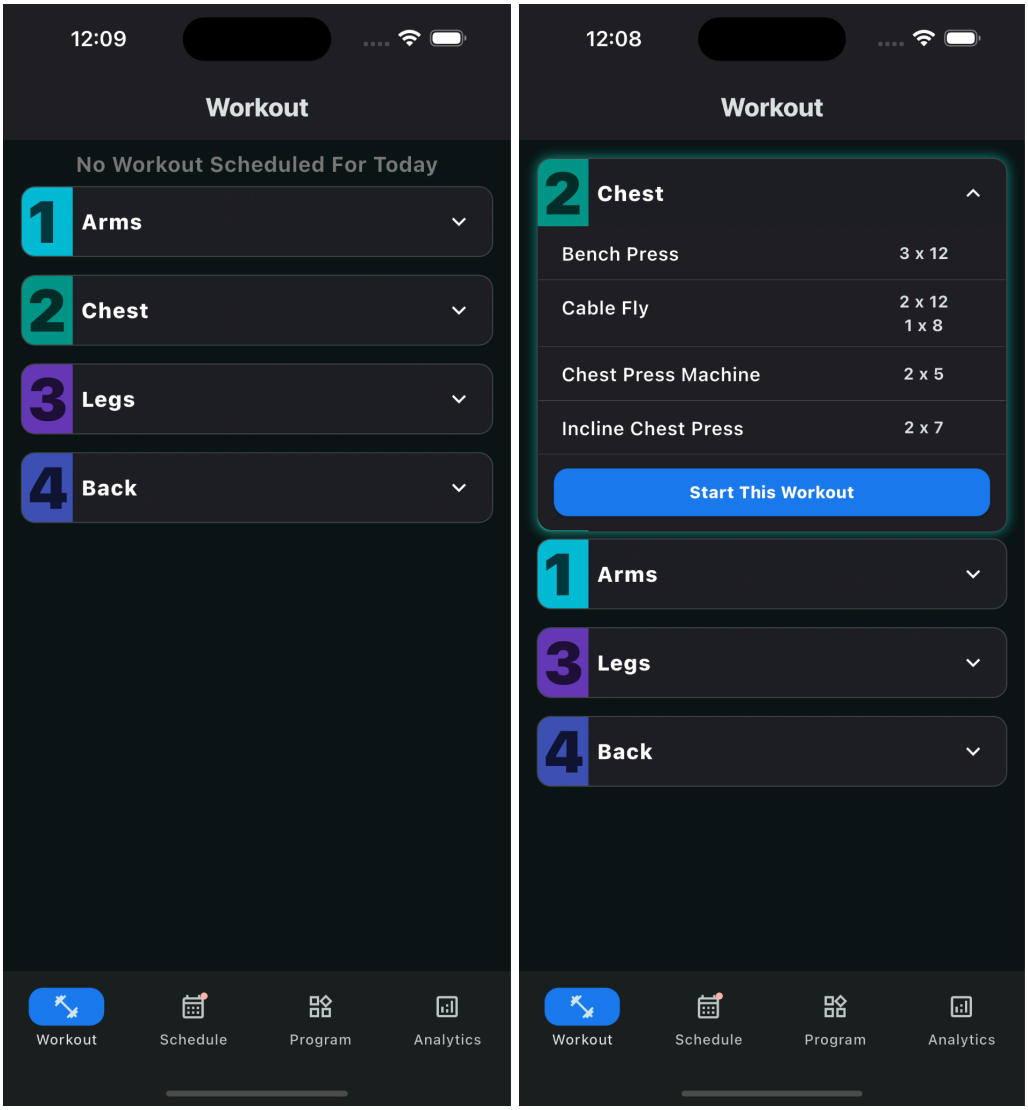
See screenshots folder in the root directory for an idea of how some of the features look. NOTE: earlier screenshots have outdated UIs, from when I was working out how to design a decent UI and was becoming comfortable with Flutter. See more recent ones for more up-to-date info.)

Currently, there are 4 simple tabs in the bottom navigation bar of the app: Workout, Schedule, Program, and Analytics.

Workout Page

“Workout” is where the user starts on opening the app. Here, they can view all the workouts in the current split, and start whichever one they like. The idea is that they have this open while they are actively at the gym - they start a workout once they get to the gym, and they can view all of the exercises and sets they have planned for the workout. Here, they can log data between sets while they rest.

This page will notify the user if there is no workout schedule for today, or if there is one, it will display it and highlight it at the top. Whether or not a day is planned for today is set by the user when they create the program - it fills in by default, and can be changed in the “Schedule” Page.



Two possible options, depending on if there's a workout scheduled for today.

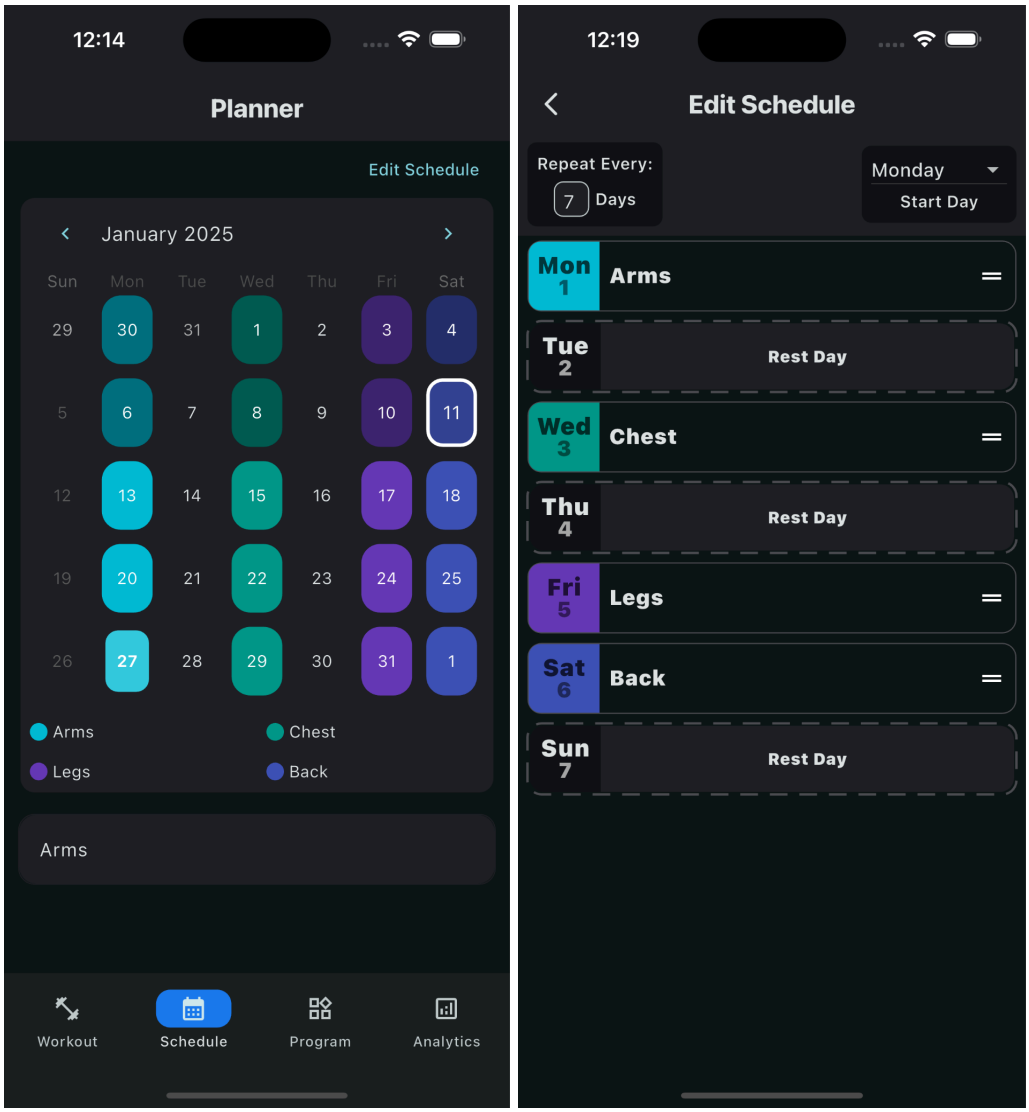
The first screen is on a sunday, when no workout happens to be scheduled. The second screen is on a wednesday, when "Chest" is scheduled. Notice how even though "Chest" was the second day in the split, it gets brought to the top and expanded. The user may click on the others to expand them instead, and do that workout if they wish to go off-program.

Schedule Page

The schedule page allows users to see a calendar of the workouts they have done, and the workouts they have planned. By pressing on a day, the day title comes up at the bottom for that day. They can also see this as each day is colour coded, and there is a legend.

The outlined day is today, and you can see the 27th is smaller and lighter, it is selected, and it happens to be when “Arms” is scheduled. The user may scroll through the months to view history/future workouts, if they wish to reschedule a workout for a one-off time because they have another event in their life.

The edit schedule button takes the user to a page where they can customize the period At which the program loops back, and the day of the week to start. It uses a drag and drop to add rest days or move days around.



Program Page

Analytics Page

Setup

File Structure

The file is mostly straightforward, and I tried to modularize things as much as I reasonably could. Under **lib/** directory, for each of the 4 main pages there is a separate file containing the Dart to run the logic and UI for that page. There is a main file which is the entrypoint of the project, and displays the scaffold with the navigation bar which directs to the different pages.

The **user.dart** file stores the multiprovider class, which is used for state management. Here, variables which need to be accessed across different pages are stored, and there are corresponding functions to change or perform an action on them. These also call the database functions **asynchronously**, so that the database gets updated to reflect what the user sees, but the rendering of the UI is not slowed down by the database functions. That being said, the database functions still execute very quickly, regularly <0.25s just from me eyeballing it. It's all local, so it's fast, I just didn't want UI frame stutters.

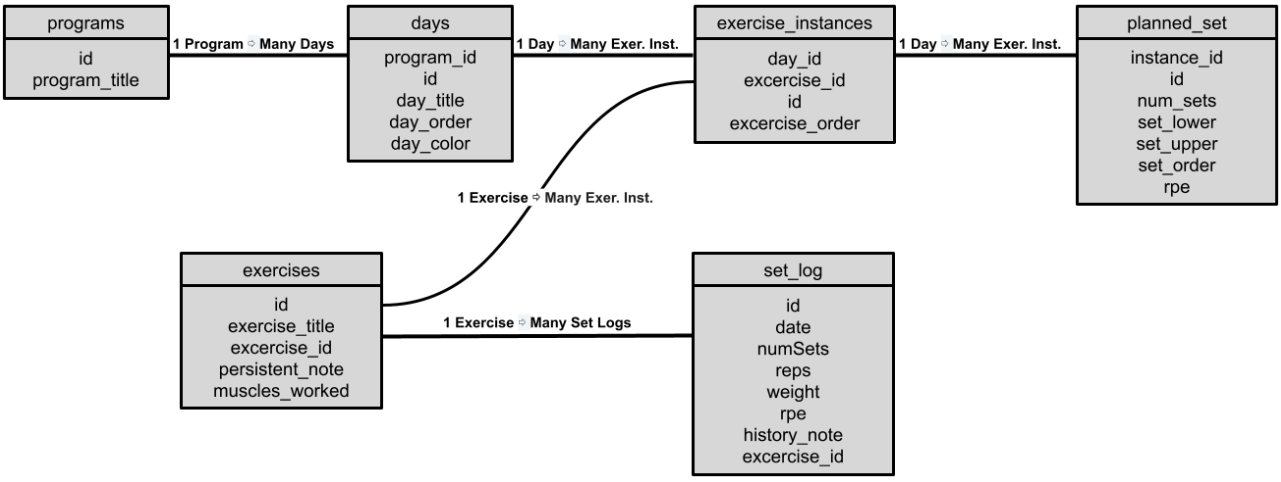
Under the **database/** directory, there is the **database_helper.dart** file which contains the database helper class. This class is what runs all of the SQL, and makes interactions with the database easier by having async methods which can be called to update the database. The **profile.dart** file contains all of the classes for the tables, so that data can be easily moved to and from disk to memory. They have toMap and fromMap methods to retrieve/provide information to the database. These classes are what are used everywhere in the project to store data for their corresponding structure (eg. exercises stores excercise_title, muscles_worked, etc., database ID, etc.)

Under **assets/** directory, there is the **exercises.txt** file which contains all the exercises initially in the library of exercises. This file is parsed when the user first downloads and opens the app, and the data is added to the database.

There are also other assets in this folder (and kinda floating around - it needs a bit of cleanup lol) of designs I was thinking of using for some background somewhere, such as **background.png**. You can find **screenshots** of the UI throughout development in the **screenshots/** directory.

Database

The database is done using the SQFLite library, and so the database follows a RDBMS structure and uses SQL for creating tables, queries, CRUD, etc. The database follows the following schema:



The user can have multiple distinct programs, each with their own days that make them up. eg. a student could have a program for exam season, when they have less time to workout, so they do full body 3x/week, and an upper/lower split 4x/week otherwise. The exercises table will store a large dataset of 899 distinct exercises which I found from here. The user can also add custom exercises if their exercise isn't on the list. From there, the user takes an exercise and can add an instance of it to their day. Then, the sets planned for that exercise for that day are specified. There could be multiple instances of the same exercise, but the instances themselves are kept separate. The set_log table will store data logged for an exercise during workouts, and is used to show analytics.

Dates are stored as strings, using the **ISO 8601 Standard** date format:

yyyy-MM-ddTHH:mm:ss