CS31620

Assignment – Part B

Student ID: 200054045

Contents

Software Design

My application relies on effective storage of data in Data Store Preferences and an SQLite database, as well as its robust and object-oriented nature. As a result, the app is virtually bug-free and functions as intended, providing a seamless workout experience for the user.

The colours used in this app are hand-generated based on the Figma template developed for part A, and relies on a blue-white colour scheme to provide a friendly and welcoming user interface.

Data Persistence Storage

In order to store data persistently across instances of the app, I chose to use two different methods for two different purposes.

1. SQLite Database

To store the data required for the workouts and exercises, I opted to use an SQLite database. As the app currently only requires data to be stored locally, there was no need to store data on a cloud service. As there needs to be easy access to creating, editing and deleting exercises, I deemed the table layout of the SQLite database to be the most suitable.

The database contains 3 tables:

- A table to store the workout data, containing the workout day and the custom name set by the user.
- A table to store the exercise data, containing the exercises and their details: a unique ID, a name, the sets and reps, the weight of the exercise, the custom icon int reference, and the tint of the icon as an int.
- A table that creates links between the workouts and exercises. This table uses the day of the table, the exercise ID, and the position in the workout routine to create a unique resource. It also has a unique ID column, which in hindsight is redundant as the 3 other columns create a unique compound key.

To effectively use the databases, I relied on a tutorial found on GeeksForGeeks that teaches the reader how to read and write data to an SQLite database in Jetpack Compose.

https://www.geeksforgeeks.org/how-to-read-data-from-sqlite-database-in-android-using-jetpack-compose/

2. Data Store Preferences

For data that only needed one value to be stored, unlike in a table-based database, I deemed the data store preferences solution to be the most effective, as it works as a library to store key-value pairs. This database stores things such as the user settings, as well as the status of the workouts such as whether or not the user has already completed the workout of the day, and the last day the workout was completed. This is mainly used in the running of the workout itself so the user can be informed they have already completed the workout.

App Navigation / Workout Implementation

Navigation

The user has 4 different navigable pages accessible from the bottom navigation bar featured on the app. These are:

- The home page, where all the workouts and their contents are displayed
- The exercises page, where the user can access, edit and remove existing exercises, or create new ones, independent of workouts
- The settings page, where various options are present
- The run workout page, where the workout of the day is run for the user to complete.

The current page the user is on is displayed in the top bar, with the exception of the run workout page as its designed to discourage you from exiting the workout early.

Home page

At the time of writing, there are 7 workouts available to the user shown on the home page, each representing a day of the week. When the user first opens the app, these workouts have no custom names and no exercises set. The user has the ability to set a custom name for the workouts as well as add, remove and reposition exercises in the workout to their desires. There is also a clear button that removes all exercises from the workout in one go, if the user wanted to start from scratch for that particular day. The user also has access to create a new exercise from this screen when trying to add an exercise to the workout. The user can add as many or as few exercises to their workout as they like, and can add multiple of the same exercise if they wish, as well as use the same exercise on multiple workouts.

Exercises page

This page allows the user to view, edit and delete any existing exercises, as well as create a new exercise if they wish. The exercises are displayed with all their information present, including the icon, sets, reps and weights, shown in a scrollable column.

Settings page

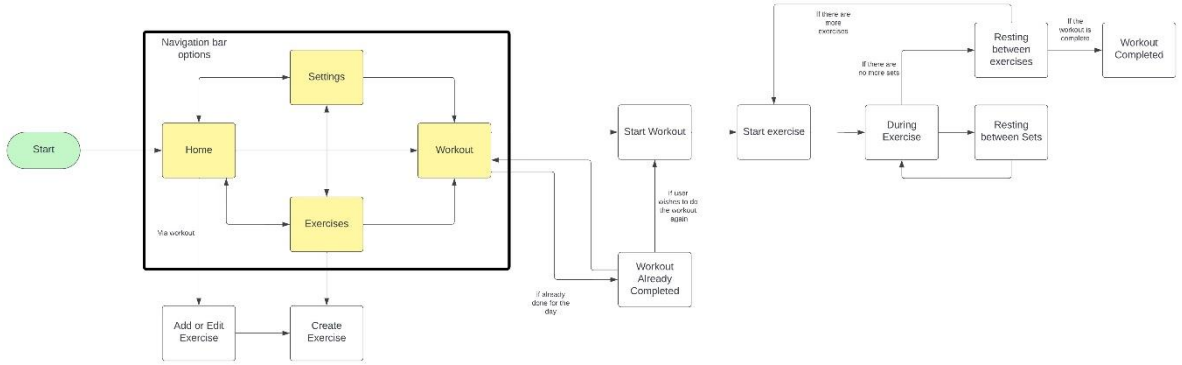
Contained on this page are a few options, some active and some inactive. There are buttons to clear various parts of data, which function, and some toggles and buttons that do not have an effect on the app, acting instead as placeholders for possible future implementations.

Run Workout page

This page is where the user runs the workout. The program goes through each exercise, switching between resting and exercising screens. Each step of the process has a different coloured background to make it more distinct as the content of each page is quite similar.

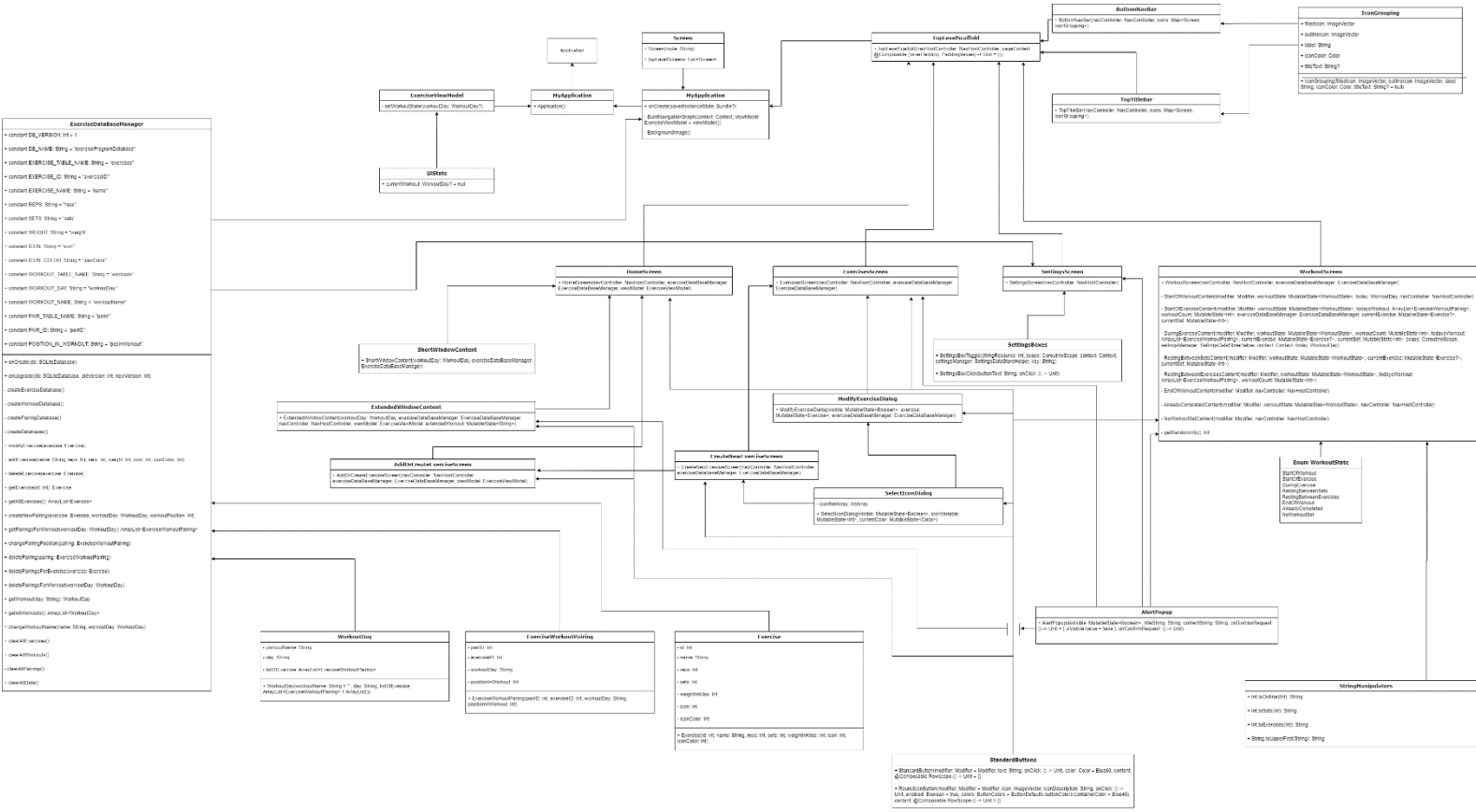
UML Diagrams

Navigation Diagram



Database Model

UML Diagram



provided in ZIP file as SVG

Testing

To test my application, I originally decided to create some unit tests to test the SQLite database. However, setting the tests up proved to be far more tedious than I had imagined, and I unfortunately had to instead create a manual testing table.

Test Table

Test Case	Description	Input	Expected Result	Result	Pass/Fail
1	Test naming the workouts	Expand a workout on the home page, set a name in the textbox	Name saves to workout and persists across app instances	Name saves to workout and persists across app instances	pass
2	Test creating an exercise with expected inputs	Navigate to exercises page, press “Create New”, fill in the textboxes, click “Save and Exit”	Exercise is created and saved in database	Exercise is created and saved in database	pass
3	Test creating an exercise with invalid inputs	Navigate to exercises page, press “Create New”, fill in the textboxes with values exceeding limits or with invalid types, click “Save and Exit”	Invalid inputs are not taken and character limits are not exceeded	Invalid inputs are not taken and character limits are not exceeded	pass
4	Text cancelling the creation of an exercise doesn’t still create one	Navigate to exercises page, press “Create New”, fill in the textboxes with values exceeding limits or with invalid types, click “Cancel”	Exercise not created	Exercise not created	pass
5	Test creating an exercise without changing default values	Navigate to exercises page, press “Create New”, click “Save and Exit” without changing inputs	Default exercise created with “Unnamed” name	Default exercise created with “Unnamed” name	pass

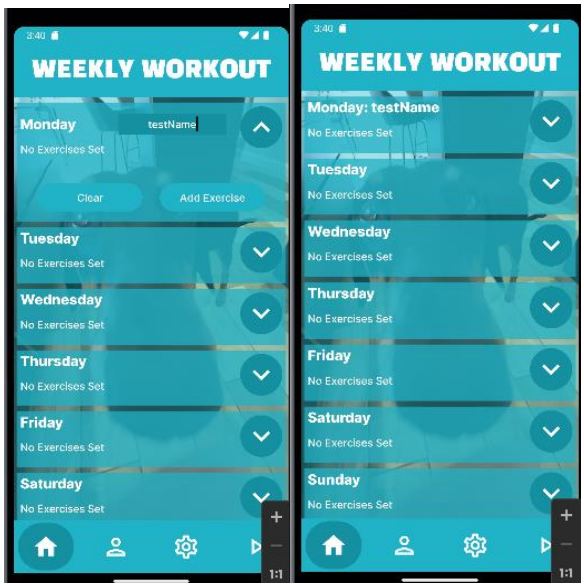
6	Test editing an exercise	Navigate to exercises page, press cog next to an exercise, make changes to each input	Exercise is modified and displays the new data	Exercise is modified and displays the new data	pass
7	Test deleting an exercise	Navigate to exercises page, press delete button on an exercise, press confirm on dialog	Exercise is removed from the screen and the database	Exercise is removed from the screen and the database	pass
8	Test adding an exercise to a workout	Open dropdown for a workout on homepage, press “add exercise”, add an exercise	Exercise is added to workout	Exercise is added to workout	pass
9	Test moving exercises in workout	Open dropdown for a workout containing 2 or more exercises, move exercise up or down the order	Exercises are successfully swapped	Exercises are successfully swapped	pass
10	Test removing exercise from workout	Open dropdown for a workout containing an exercise, press delete button	Exercise is removed from workout but persists as an exercise	Exercise is removed from workout but persists as an exercise	pass
11	Test clearing all exercises from a workout	Open dropdown for a workout containing multiple exercises, press clear button and confirm	All exercises are removed from workout	All exercises are removed from workout	pass
12	Test resetting all workouts	Navigate to settings, press “Reset all Workouts” and confirm	All workouts are reset to default with no exercises or name, exercises are not deleted	All workouts are reset to default with no exercises or name, exercises are not deleted	pass

13	Test deleting all exercises	Navigate to settings, press “Delete all exercises” and confirm	All exercises are deleted, including being removed from workouts	All exercises are deleted, including being removed from workouts	pass
14	Test deleting all data	Navigate to settings, press “Delete all data” and confirm	App is reset to defaults, workouts are unnamed and there are no exercises	App is reset to defaults, workouts are unnamed and there are no exercises	pass
15	Test starting a workout with no set exercises	Navigate to run workout page without any exercises set for the day	Page is displayed stating there are no workouts and presenting a button for the user to return to the page they left, which works	Page is displayed stating there are no workouts and presenting a button for the user to return to the page they left, which works	pass
16	Test starting a workout with 1 exercise with 1 set	Navigate to run workout, start workout, start exercise, end set	Exercise should say 1st and final, the name of the exercise, the reps sets and weight. Starting exercise should say set 1 of 1. Ending set should show screen saying exercise completed and workout done	Exercise should say 1st and final, the name of the exercise, the reps sets and weight. Starting exercise should say set 1 of 1. Ending set should show screen saying exercise completed and workout done	

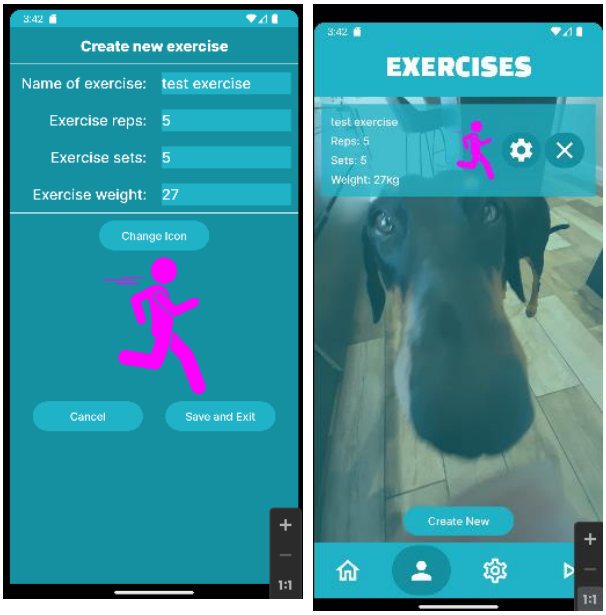
17	Test starting a workout with 2 exercises with 2 sets each	Navigate to run workout, go through steps	Exercise should say 1st and final, the name of the exercise, the reps sets and weight. Starting exercise should say set 1 of 2. Ending the exercise should lead to the 2nd one which follows the same process. Ending the 2nd exercise leads to screen saying workout completed.	Exercise should say 1st and final, the name of the exercise, the reps sets and weight. Starting exercise should say set 1 of 2. Ending the exercise should lead to the 2nd one which follows the same process. Ending the 2nd exercise leads to screen saying workout completed.	pass
18	Test starting a workout when workout has already been completed on the day	Navigate to run workout	Screen shows workout already completed, gives option to redo	Screen shows workout already completed, gives option to redo	pass
19	Test follows 18 – attempt to redo workout	Click redo workout	Screen returns to start workout page	Screen returns to start workout page	pass

Evidence

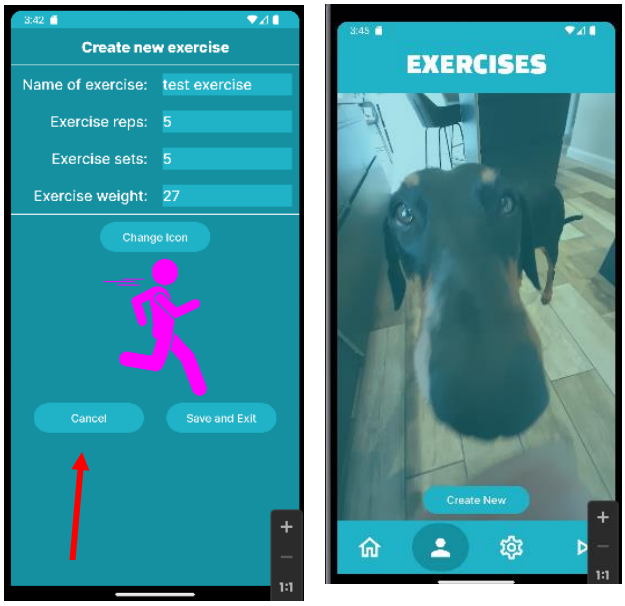
1:



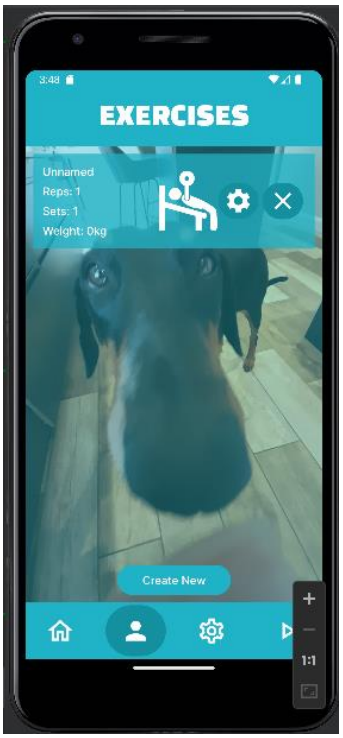
2, 3:



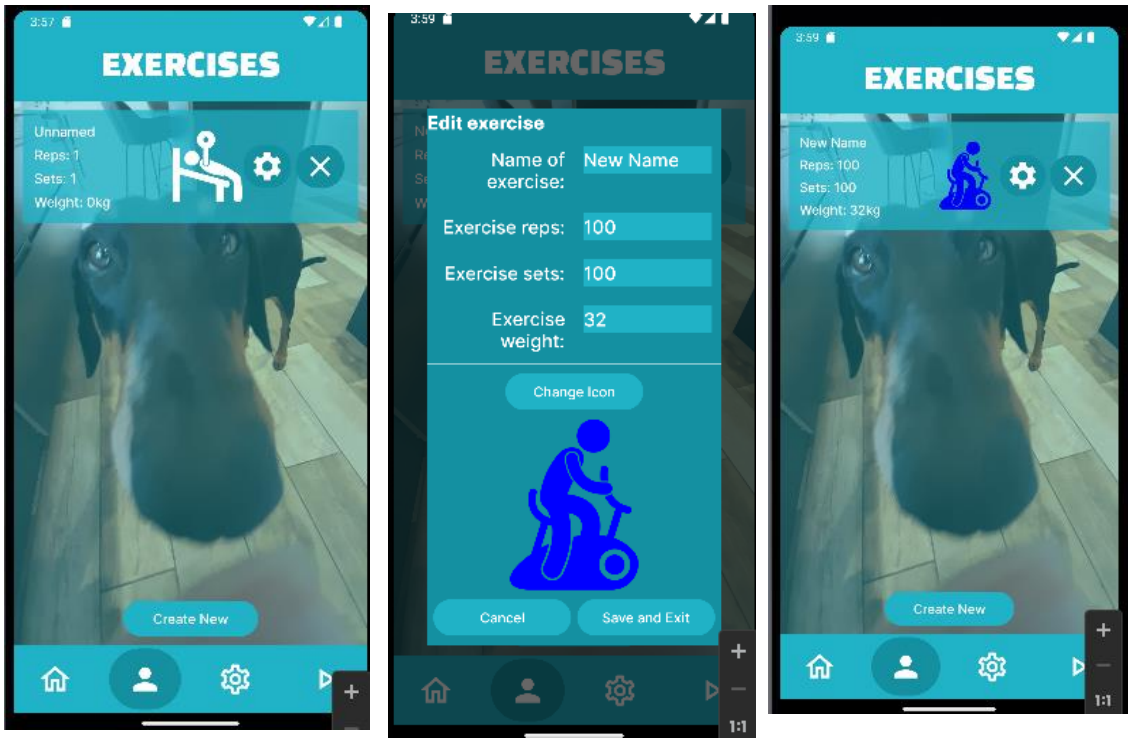
4:



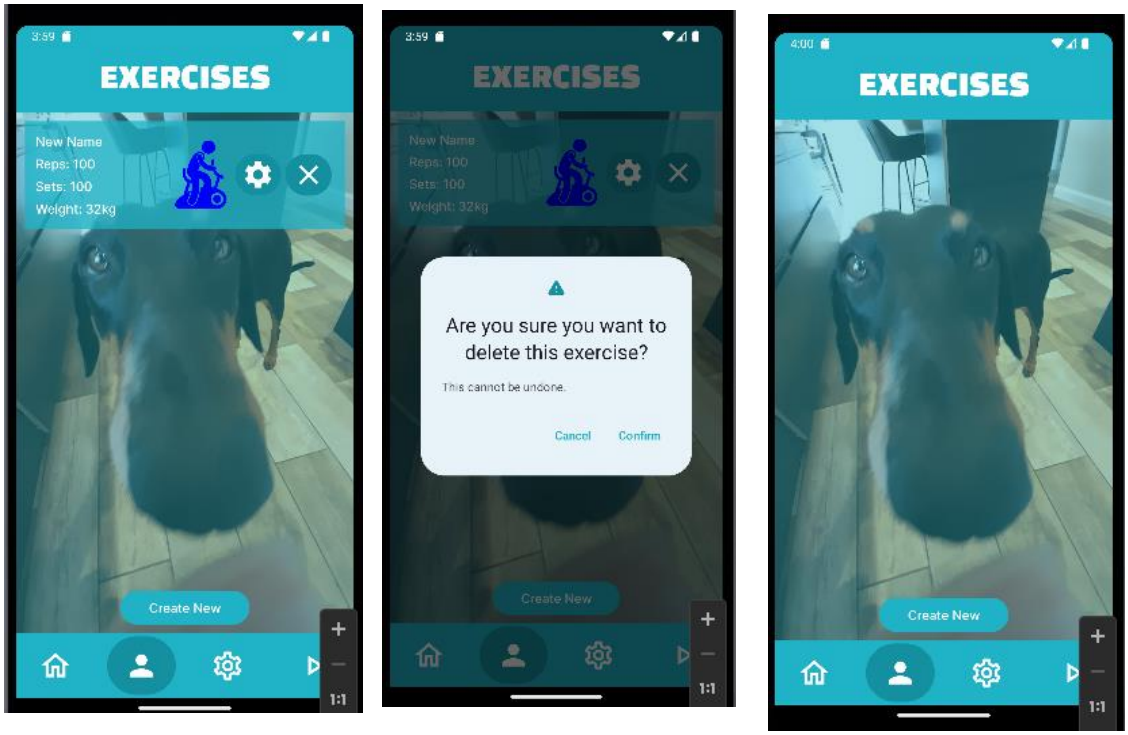
5:



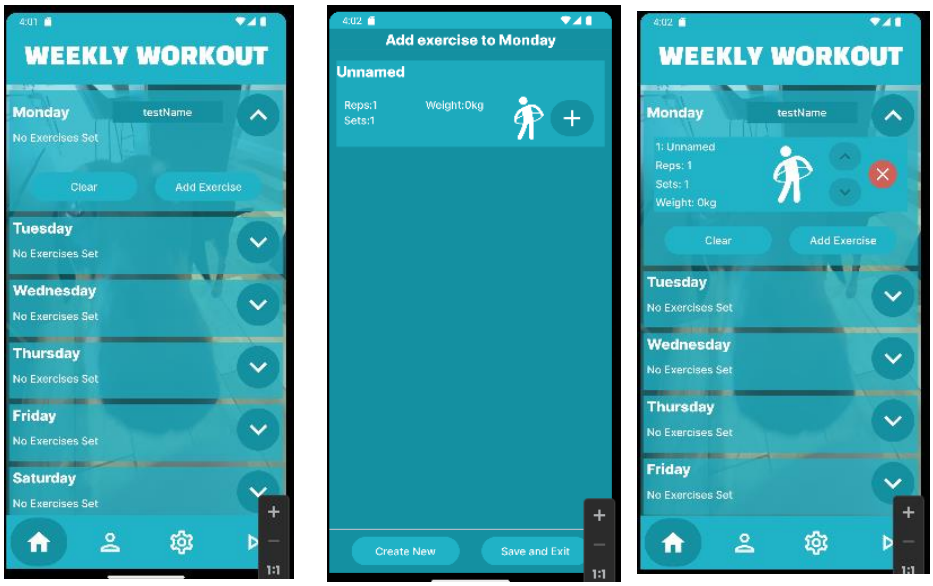
6:


EXERCISES
Unnamed
Reps: 1
Sets: 1
Weight: 0kg
Create New
EXERCISES
Edit exercise
Name of exercise:
New Name
Exercise reps:
100
Exercise sets:
100
Exercise weight:
32
Change Icon
Cancel
Save and Exit
EXERCISES
New Name
Reps: 100
Sets: 100
Weight: 32kg
Create New

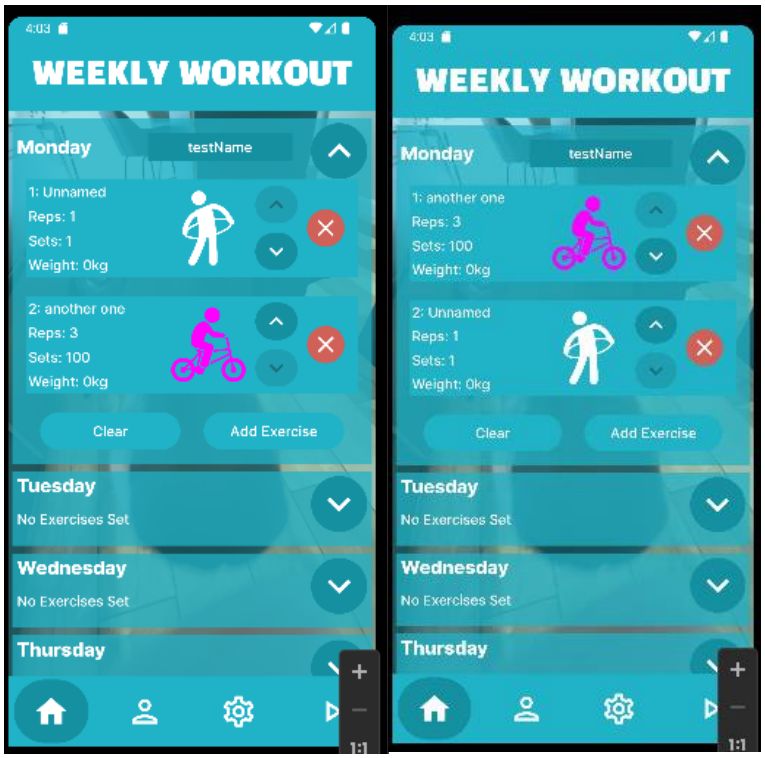
7:


EXERCISES
New Name
Reps: 100
Sets: 100
Weight: 32kg
Create New
EXERCISES
Are you sure you want to delete this exercise?
This cannot be undone.
Cancel
Confirm
EXERCISES
Create New

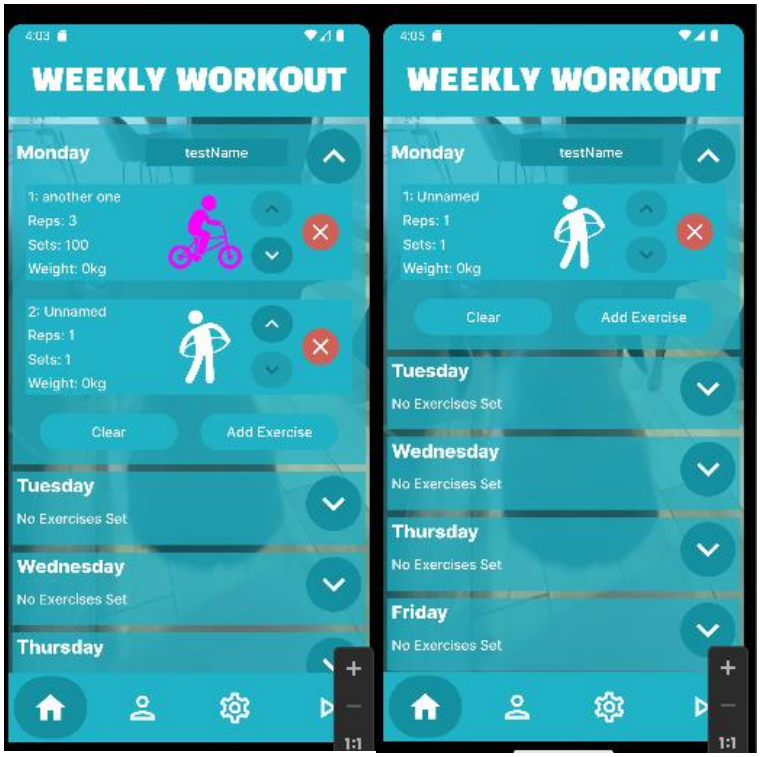
8:


WEEKLY WORKOUT
Monday
testName
No Exercises Set
Clear
Add Exercise
Tuesday
No Exercises Set
Wednesday
No Exercises Set
Thursday
No Exercises Set
Friday
No Exercises Set
Saturday
No Exercises Set
Add exercise to Monday
Unnamed
Reps:1
Sets:1
Weight:0kg
Create New
Save and Exit
WEEKLY WORKOUT
Monday
testName
1: Unnamed
Reps: 1
Sets: 1
Weight: 0kg
Clear
Add Exercise
Tuesday
No Exercises Set
Wednesday
No Exercises Set
Thursday
No Exercises Set
Friday
No Exercises Set

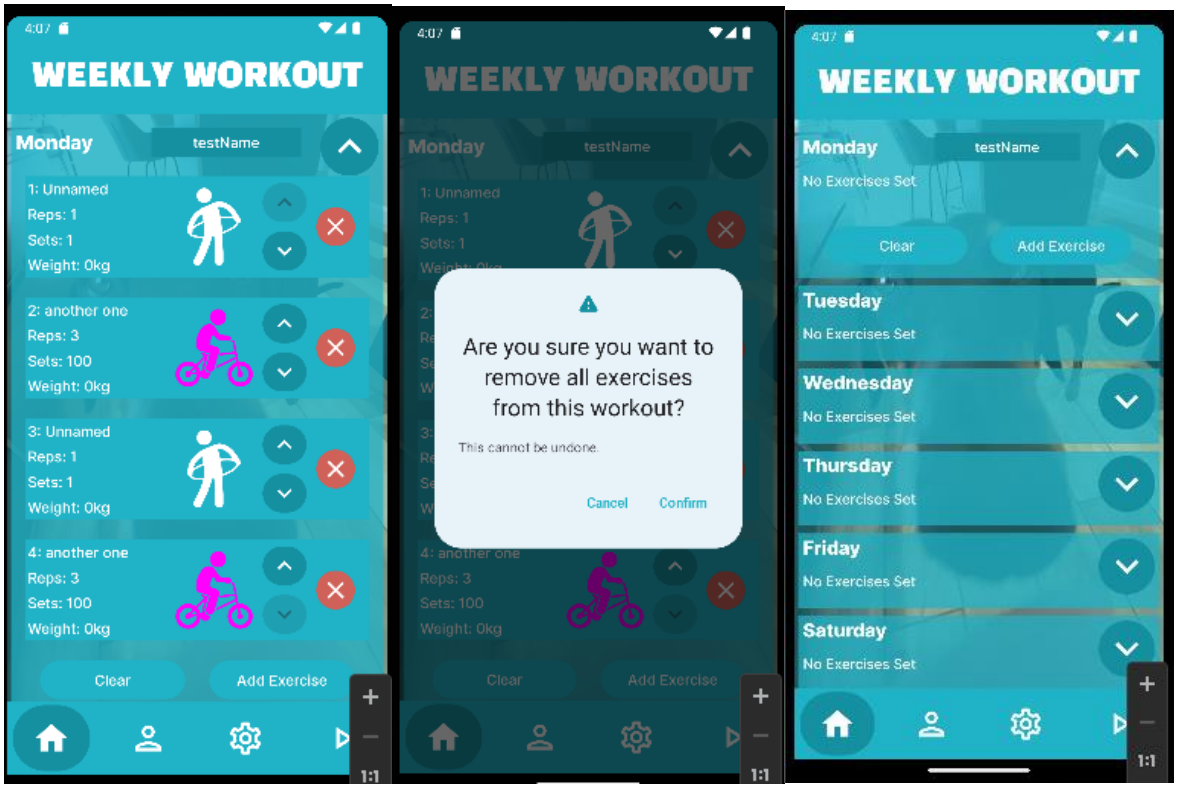
9:


WEEKLY WORKOUT
Monday
testName
1: Unnamed
Reps: 1
Sets: 1
Weight: 0kg
2: another one
Reps: 3
Sets: 100
Weight: 0kg
Clear
Add Exercise
Tuesday
No Exercises Set
Wednesday
No Exercises Set
Thursday
WEEKLY WORKOUT
Monday
testName
1: another one
Reps: 3
Sets: 100
Weight: 0kg
2: Unnamed
Reps: 1
Sets: 1
Weight: 0kg
Clear
Add Exercise
Tuesday
No Exercises Set
Wednesday
No Exercises Set
Thursday

10:


WEEKLY WORKOUT
Monday
testName
1: another one
Reps: 3
Sets: 100
Weight: 0kg
2: Unnamed
Reps: 1
Sets: 1
Weight: 0kg
Clear
Add Exercise
Tuesday
No Exercises Set
Wednesday
No Exercises Set
Thursday
WEEKLY WORKOUT
Monday
testName
1: Unnamed
Reps: 1
Sets: 1
Weight: 0kg
Clear
Add Exercise
Tuesday
No Exercises Set
Wednesday
No Exercises Set
Thursday
No Exercises Set
Friday
No Exercises Set

11:


WEEKLY WORKOUT
Monday
testName
1: Unnamed
Reps: 1
Sets: 1
Weight: 0kg
2: another one
Reps: 3
Sets: 100
Weight: 0kg
3: Unnamed
Reps: 1
Sets: 1
Weight: 0kg
4: another one
Reps: 3
Sets: 100
Weight: 0kg
Clear
Add Exercise
WEEKLY WORKOUT
Are you sure you want to remove all exercises from this workout?
This cannot be undone.
Cancel
Confirm
WEEKLY WORKOUT
Monday
testName
No Exercises Set
Clear
Add Exercise
Tuesday
No Exercises Set
Wednesday
No Exercises Set
Thursday
No Exercises Set
Friday
No Exercises Set
Saturday
No Exercises Set

Possible REST API for remote persistence

Using a REST API for external data storage would provide a far greater range of opportunities for the user. Not only would the data persist even if the user was to uninstall and reinstall the app, but it would also be protected from any possible corruptions, and it would allow the user to use the data on multiple devices. For a workout app, this is particularly useful, as many users may prefer to use the app on a smartwatch, whilst setting the workouts may be more suited to a larger-screened mobile device. The REST API would have to be able to support the requesting, adding, removing and editing of data for both workouts and exercises, as well as the bonds between them.

A suitable RESTful API would allow the user indirect access to GET, POST, PUT, DELETE and PATCH methods, so they can manipulate their exercise and workout data freely via the app. The API should also have security methods to prevent unauthorised users from accessing someone else's data; as a result, a username and password system would need to be implemented. Unwanted users should be turned away with a 401 or 403 error depending on if the user is signed in or not.

The structure of the API would rely on the user's username, followed by two branches: workout and exercise. When calling the workout, an additional value would be used to determine which day the user is attempting to access, modify or delete from. An example URL would be workoutplanner.com/user/workout?day=Sunday or workoutplanner.com/user/exercise.

Suitable API methods would look as follows:

GET /workoutplanner.com/user/workout?day=Monday
Sending headers: Accept: applications/json; q=1.0, application/xml; q=0.8, text/*; q=0.1
Received headers: Content-Type: application/json
Content received: JSON resource representation
Status codes: 200 OK, 401 Unauthorized, 404 not found

POST /workoutplanner.com/user/exercises
Sending headers: Accept: applications/json; q=1.0, application/xml; q=0.8, text/*; q=0.1
Received headers: Content-Type: application/json
Content received: Text response confirming addition of new exercise
Status codes: 201 Created, 401 Unauthorized

PUT /workoutplanner.com/user/workout?day=Tuesday
Sending headers: Accept: applications/json; q=1.0, application/xml; q=0.8, text/*; q=0.1
Received headers: Content-Type: application/json
Content received: Text response confirming modification of workout
Status codes: 200 OK, 201 Created, 204 No Content, 401 Unauthorized

DELETE /workoutplanner.com/user/exercises?id=12
Sending headers: Accept: applications/json; q=1.0, application/xml; q=0.8, text/*; q=0.1
Received headers: Content-Type: application/json
Content received: Text response confirming deletion of workout
Status codes: 200 OK, 204 No Content, 401 Unauthorized

Reflections

What went well:

I am very happy with the design of my app, as well as the functionality that it provides. It provides a seamless and clean experience for the user with very few bugs, if any. It achieves most functional requirements that were set as part of the assignment. I am especially proud of the way my app interacts with the database, efficiently giving and taking data, as well as the way the database effectively uses a 3rd database to form the well-needed connections between the exercises and the workouts.

I am also satisfied with the abstract nature of the code; many elements have been repurposed as reusable chunks of code to clean up the code, including multi-use dialogs and buttons. It clearly demonstrates my understanding of mutable states and UI development, as well as lambda functions.

I am also happy with my changes compared to the original Figma-based UI design. I have used a lot less green in the app, instead focusing on using blues and white, with red for the more important buttons.

What could have gone better:

My biggest grief with the assignment program is that it does not implement dropsets in any way. This is mainly due to my lack of understanding of the concept, and led me to decide I would be able to create a better program without attempting to add functionality I didn't fully know.

I am also disappointed that, whilst the app notifications setting was purely for demonstration of functionality, the change background setting was initially intended to be a fully developed feature. However, I simply did not have the time to implement it. Fortunately, as that feature was intended to be flair anyway, it does not take away from the app's overall functionality.

What I have learnt:

As a result of this assignment, I have learnt a great deal about working with jetpack compose as well as Kotlin. It has helped me fully understand the processes of how an app is developed professionally. It has also reinvigorated my knowledge of using SQL databases, something I had not done for a long time.

UI Changes

The UI has remained largely the same when compared to the original prototypes in part A of the assignment. The biggest change is the colour scheme. Originally, the prototype relied on blues and greens, whereas the app uses different shades of blue. The background image has also changed. The final difference is the running of the workout, where different stages are now represented by different background colours as there was not enough differentiation between the stages in the prototype.

Self Grading

I would personally grade myself a 2/1. Whilst the program contains most of the functionality, it is missing a few key points. Similarly, the REST API section of this report is not as good as it could have been.

References used in the code

[1] GeeksForGeeks (2022) “How to Read Data From SQLite Database in Android using Jetpack Compose?” (Online) https://www.geeksforgeeks.org/how-to-read-data-from-sqlite-database-in-android-using-jetpack-compose/

[2] Leremy (Various dates) Free Icons (Online) https://www.flaticon.com/authors/leremy