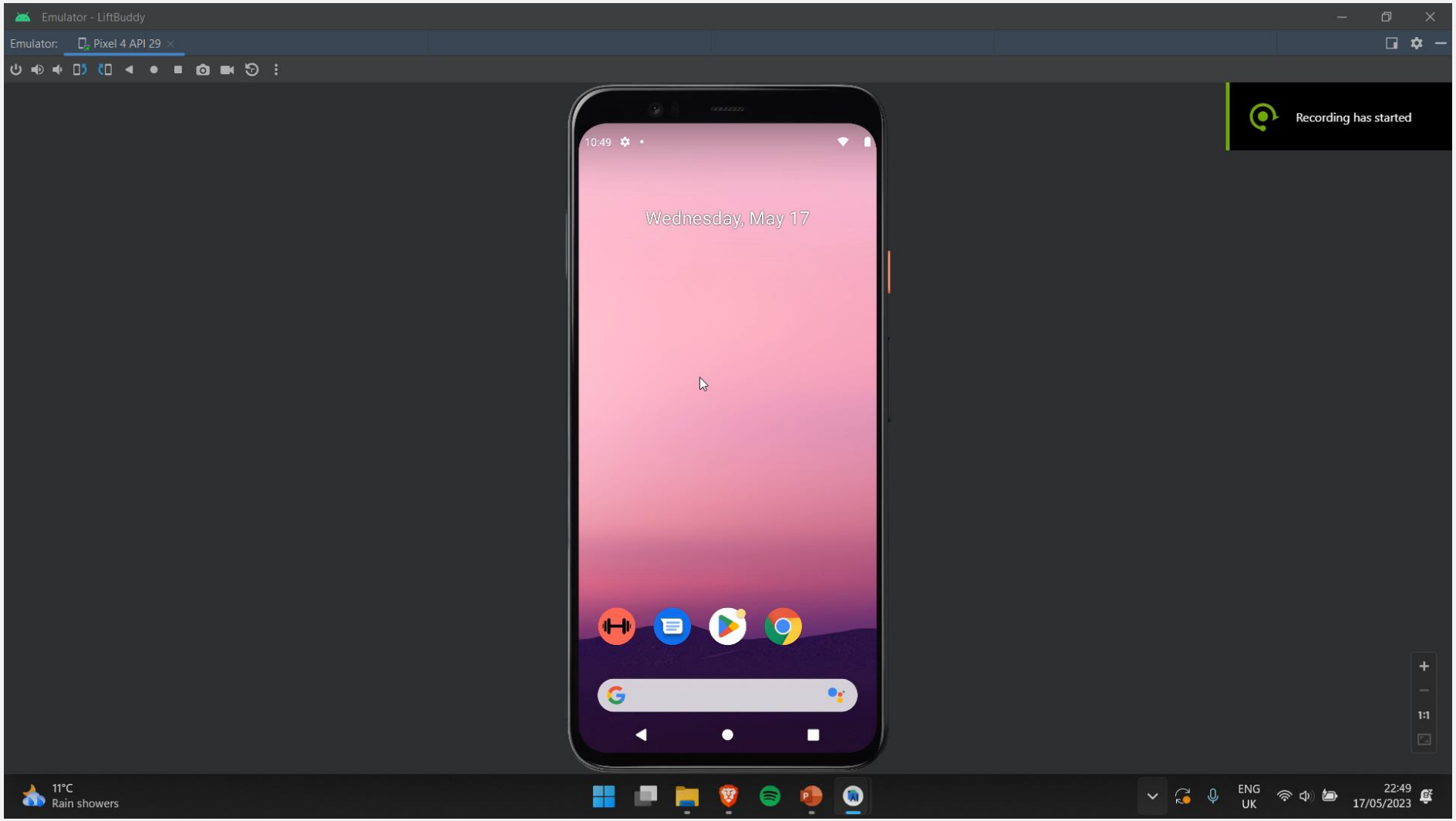
LIFTBUDDY

Martin Zhelev 2002985

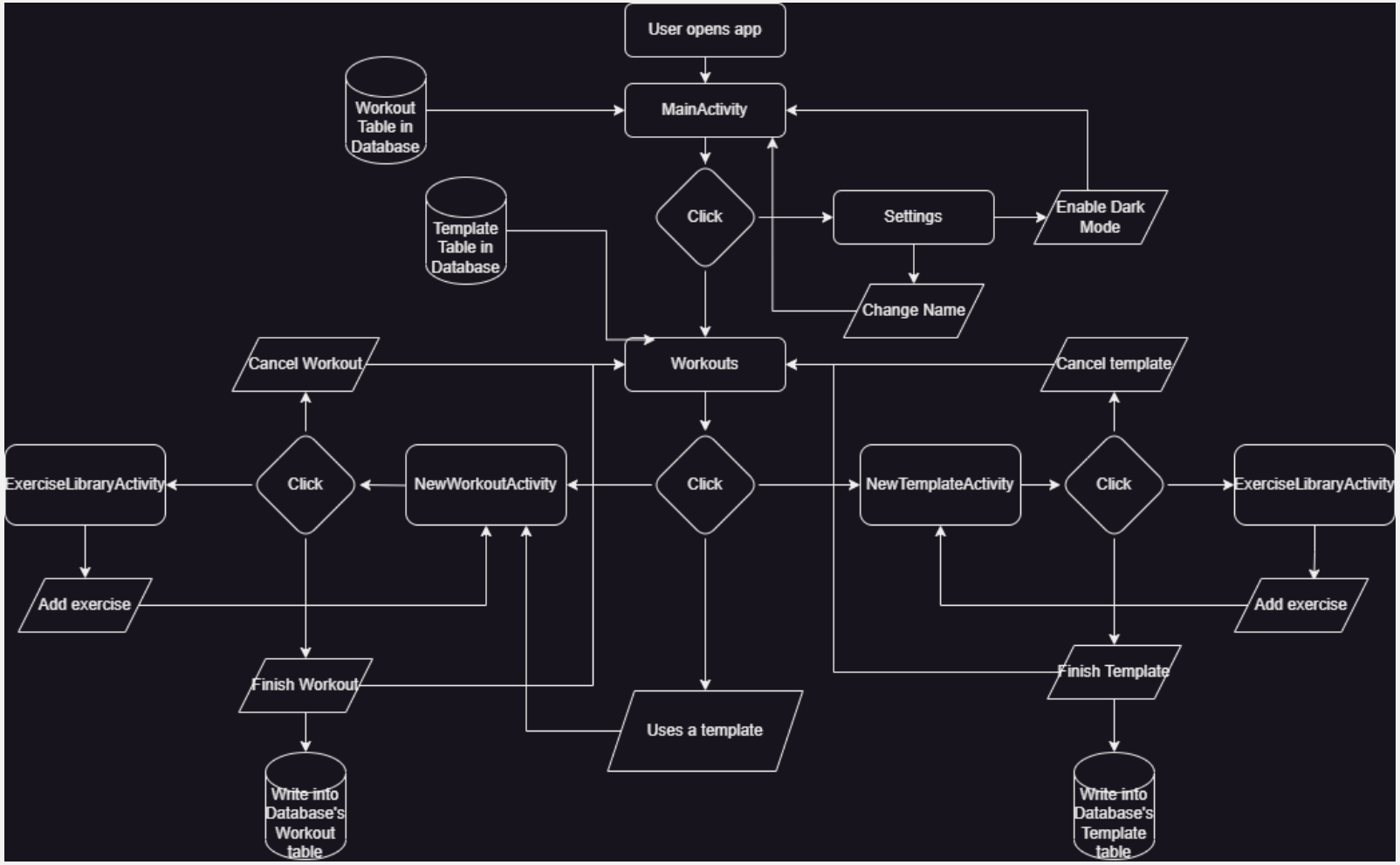
APP OVERVIEW

- Lift Buddy is a workout tracking app build using Kotlin with the following menus:
  - Profile Menu
  - Settings Menu
  - Workouts Menu
  - New Workout Menu
  - New Template Menu

DEMO


Emulator - LiftBuddy
Emulator:
Pixel 4 API 29
Recording has started
10:49
Wednesday, May 17
11°C
Rain showers
ENG
UK
22:49
17/05/2023

FLOW CHART


User opens app
Workout Table in Database
MainActivity
Template Table in Database
Click
Settings
Enable Dark Mode
Change Name
Cancel Workout
Workouts
Cancel template
ExerciseLibraryActivity
Click
NewWorkoutActivity
Click
NewTemplateActivity
Click
ExerciseLibraryActivity
Add exercise
Add exercise
Finish Workout
Finish Template
Uses a template
Write into Database's Workout table
Write into Database's Template table

STRUCTURE


Classes
convertUtilities
ExerciseClass
WorkoutClass
dynamicLayout
dynamicLayoutClass
dynamicNewTemplateClass
dynamicNewWorkoutClass
dynamicProfileClass
dynamicWorkoutsClass
Database
DatabaseHelper
Profile
MainActivity
Settings
SettingsActivity
Workouts
WorkoutsActivity
New Template
New Workout
New Template Activity
NewWorkoutActivity
Rename Dialog
renameDialogFragment

KEY FEATURES

KEY FEATURES

- Possibility to set a custom profile name (Shared Preference)
- Dark theme switching which is consistent across the app (Shared Preference)
- Ability to start a new workout and create a workout template
- Saving of workout and templates in database
- SQLite Database containing a table for templates and completed workouts
- Rename Dialog using fragment
- History displayed in profile
- Workout templates displayed in workout menu

```kotlin
private fun settings() {
    val prefs = PreferenceManager.getDefaultSharedPreferences( context: this
    val name = prefs.getString("name", "Profile")

    val tv_profile_name = findViewById<TextView>(R.id.tv_profile_name)

    binding.apply { this: ActivityMainBinding
        tv_profile_name.text = name
    }

}
```

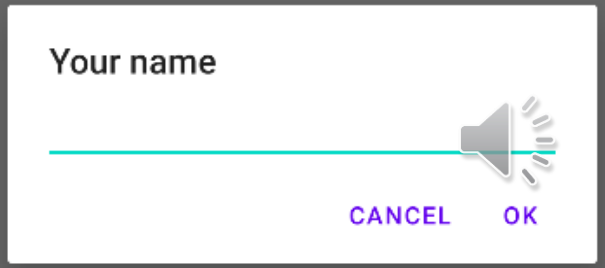
```xml
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/r
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <PreferenceCategory app:title="Basic settings">

        <EditTextPreference
            app:key="name"
            app:title="Your name"
            app:useSimpleSummaryProvider="true" />
```

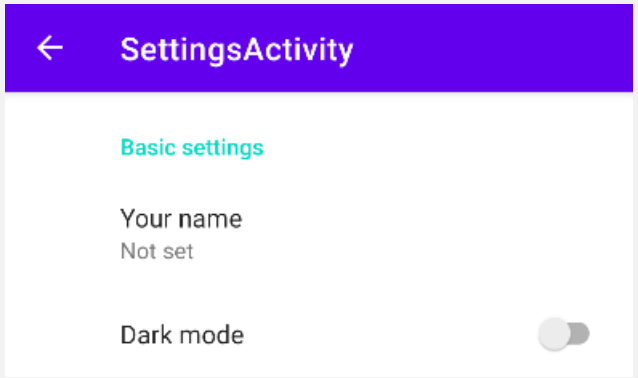
CUSTOM PROFILE NAME

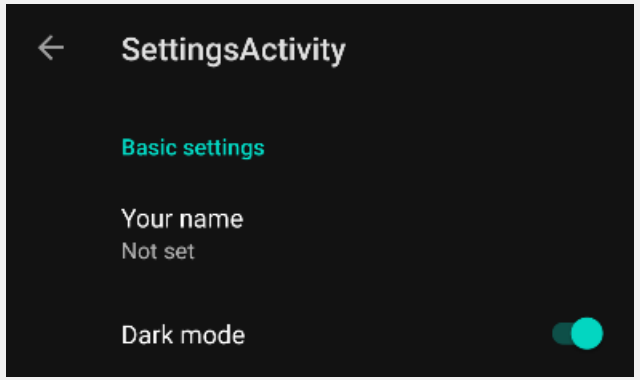
- Uses a Settings Activity
- Changed name is added as a shared preference
- Uses view binding to update name



DARK THEME

- Is a shared preference
- Achieved using AppCompatDelegate






```
    <SwitchPreferenceCompat
        app:key="dark_mode"
        android:defaultValue="false"
        app:title="Dark mode" />

  </PreferenceCategory>

PreferenceScreen>
```

```
override fun onSharedPreferenceChanged(sharedPreferences: SharedPreferences?, key: String?) {
    if (key == "dark_mode") {
        val preference = sharedPreferences?.getBoolean(key, false)
        if (preference == true) {
            AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES)
        } else {
            AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO)
        }
    }
}
```

```kotlin
fun clickHandler(view: View) {
    when (view.id) {
        R.id.btn_workout_start -> {
            val intent = Intent( packageContext: this, NewWorkoutActivity::class.jav
            getContent.launch(intent)
            overridePendingTransition( enterAnim: 0, exitAnim: 0)
        }
        R.id.btn_template_add -> {
            val intent = Intent( packageContext: this, NewTemplateActivity::class.ja
            getContent.launch(intent)
            overridePendingTransition( enterAnim: 0, exitAnim: 0)
        }
        else -> {
        }
    }
}
```

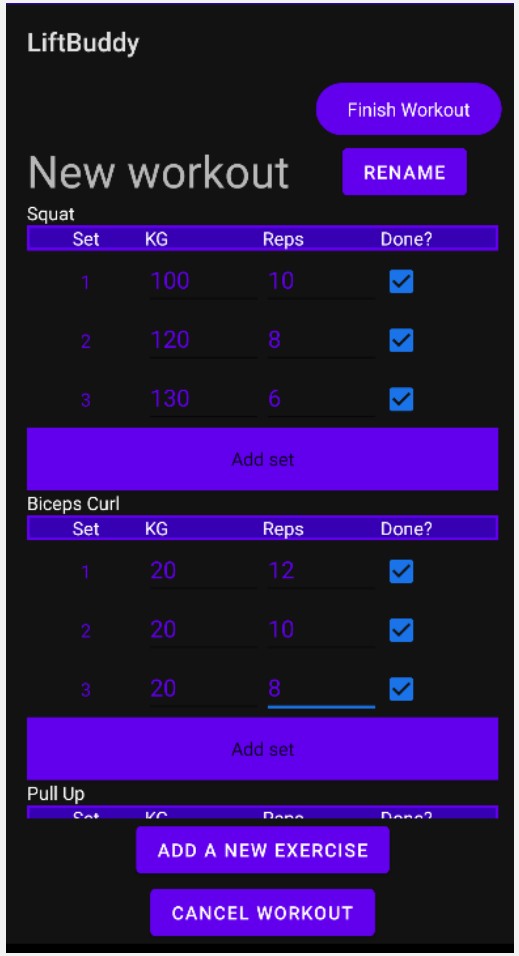
```kotlin
private val getContent = registerForActivityResult(ActivityResultContracts.StartActivityForResult()) {
    // Handle the returned result
    if (result.resultCode == Activity.RESULT_OK) {
        val data: Intent? = result.data
        // Handle the result here
        if (data != null) {
            // Check if the data received is from the new templates or new workout acitivty
            if (!data.getBooleanExtra( name: "template", defaultValue: false)) {
                //Saves workout result to database
                doneWorkoutObject = getSerializable(data, name: "workoutObject", WorkoutClass::class.ja
            } else {
                //Saves template result to databsae
                doneTemplateObject = getSerializable(data, name: "doneTemplateObject", WorkoutClass::cla
                dynamicWorkouts.addNewExerciseTemplate(doneTemplateObject)
                templatesArray.add(doneTemplateObject)
            }
        }
    } else {
        Toast.makeText( context: this, text: "Cancelled!", Toast.LENGTH_SHORT).show()
    }
}
```

LAUNCHING OF NEW WORKOUT AND NEW TEMPLATE

- Achieved usina a clickHandler that checks which button is pressed
- Activity is started for a result using the Activity Result API's

ABILITY TO START A NEW WORKOUT AND SAVE IT

- Able to choose exercises to add
- Can write down weight and reps that were achieved.
- Saved to database
- Send as intent to Workouts Activity




```
        } else {
            removeUnfinished()
            val name = workoutObject.name
            val date = workoutObject.date
            workoutObject.exercises = dynamicNewWorkout.exerciseObjects
            Toast.makeText( context: this,  text: "Workout Finished + $name
                .show()
            returnIntent.putExtra( name: "workoutObject", workoutObject)
            setResult(Activity.RESULT_OK, returnIntent)
            // Saves workout in database
            val thread: Thread
            saveWorkout(workoutObject)
            this.finish()
        }
    }
```

```
private fun saveWorkout(workoutObject: WorkoutClass) {
    val saveWorkoutThread = Thread {
        db = DatabaseHelper( context: this,  factory: null)
        /// creating variables for values
        val workoutName = workoutObject.name
        val date = workoutObject.date
        //Converts workoutObject of type WorkoutClass to ByteArray
        val objectBlob = convertUtilities.makebyte(workoutObject)
        db.addWorkout(date, workoutName, objectBlob)
    }
    startThread(saveWorkoutThread)
}
```

EXERCISE LIBRARY

- Exercises are added to the workout or template using the ExerciseLibraryActivity
- Dynamically changing text based on intent extra
- Exercises are send to the activity that launched the exercise library using intents

LiftBuddy

```
//Change displayed text based on type of activity. Adding to workout or tem
val type = this.intent.getStringExtra( name: "typeAdd")
val text = findViewById<TextView>(R.id.text_info)
if (type == "workout") {
    text.text = "Add to workout"

} else if (type == "template"){
    text.text = "Add to template"
} else {
    text.text = "Error"
    Toast.makeText( context: this,  text: "Error", Toast.LENGTH_SHORT).show()
}
```

Squat Add

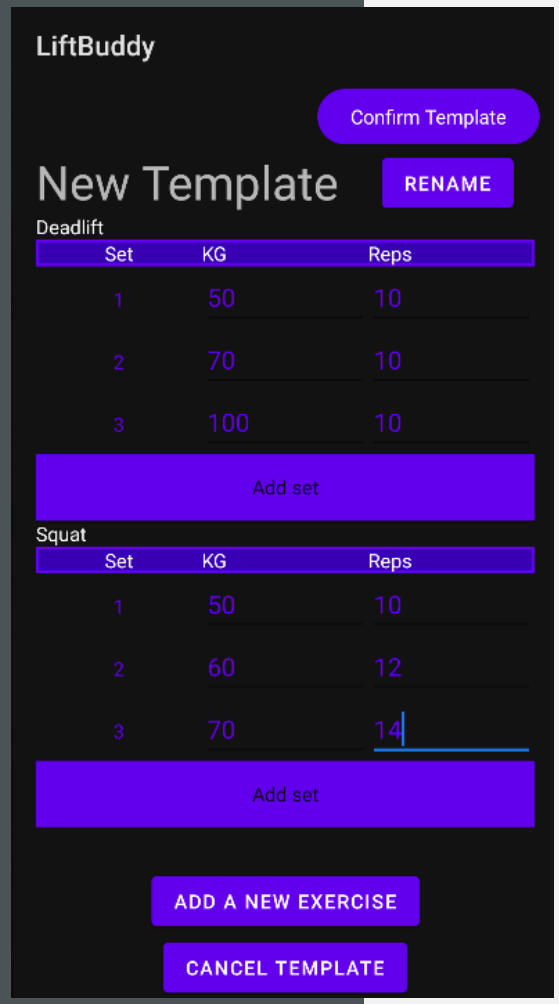
```
R.id.btn_exercise_add -> {
    val intent = Intent( packageContext: this, ExerciseLibraryActivity::class.java)
    val type = "workout"
    intent.putExtra( name: "typeAdd", type)
    getContent.launch(intent)
}
```

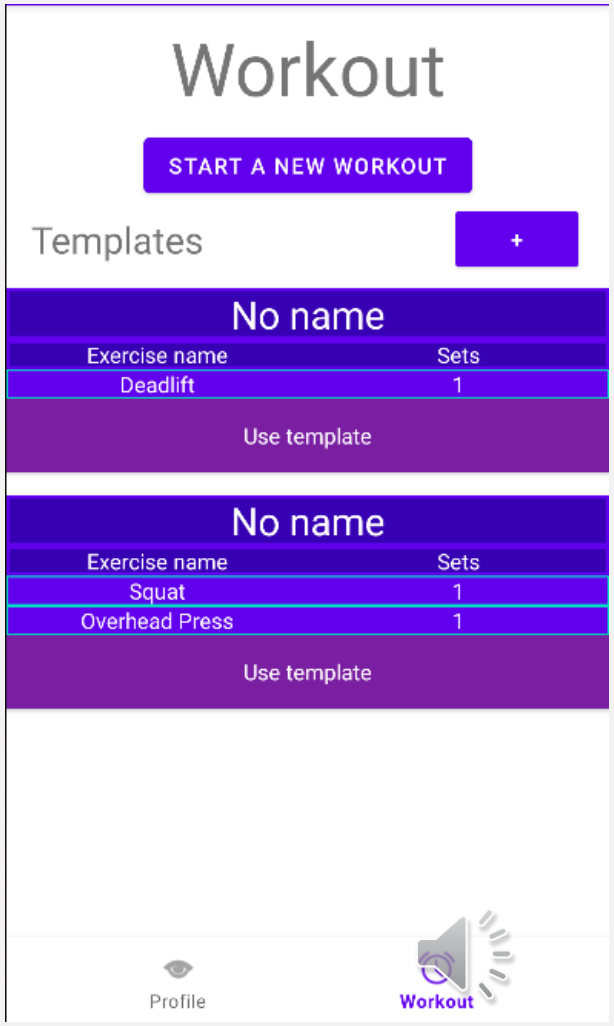
Chest Dip Add

```
} else {
    //Return array of exercise
    val returnIntent = Intent()
    returnIntent.putExtra( name: "selectedExercisesNames", selectedExercisesNames)
    setResult(Activity.RESULT_OK, returnIntent)
    this.finish()
}
```

ABILITY TO CREATE A NEW TEMPLATE AND SAVE IT

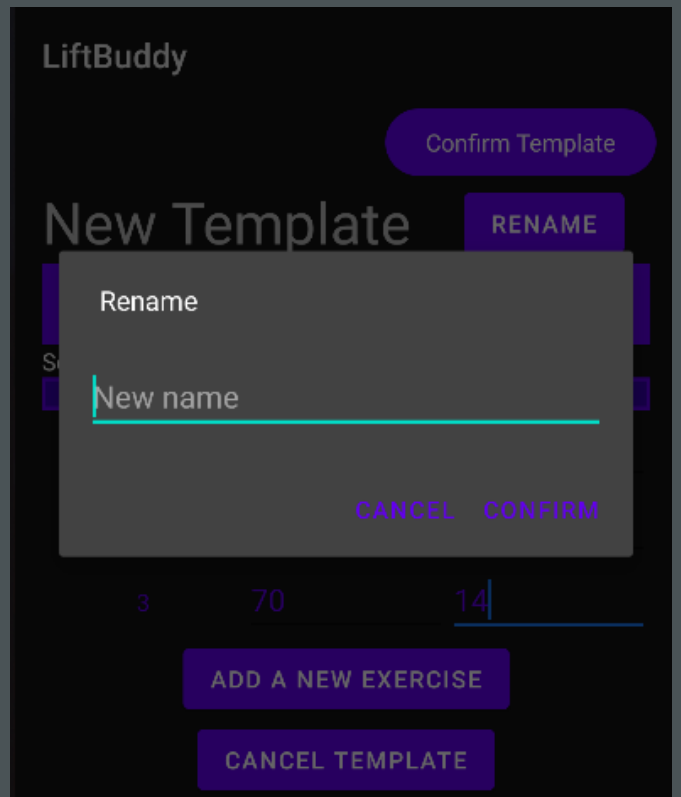
- Works similarly to creation of new workouts
- Gets saved into Database
- Templates get displayed in workout menu





RENAME DIALOG

- Used to give the user the option of renaming their workout or template
- Achived using Fragments




```
class RenameDialogFragment : DialogFragment() {
    private lateinit var et_rename_field: EditText
    private lateinit var listener: DialogListener
    override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {
        return activity?.let { it: FragmentActivity
            val builder = AlertDialog.Builder(it)
            val inflater = it.layoutInflater
            val view = inflater.inflate(R.layout.layout_dialog,  root: null

            builder.setView(view)
                .setMessage("Rename")
                .setPositiveButton( text: "Confirm",
                    DialogInterface.OnClickListener { dialog, id ->
                        val renamed: String = et_rename_field.text.toStri
                        listener.applyRename(renamed)
                    })
                .setNegativeButton( text: "Cancel",
                    DialogInterface.OnClickListener { dialog, id ->

                    })
            et_rename_field = view.findViewById(R.id.et_rename_field)
            return builder.create()
        } ?: throw IllegalStateException("Activity cannot be null")
    }

    interface DialogListener {
        fun applyRename(name: String)
    }
```

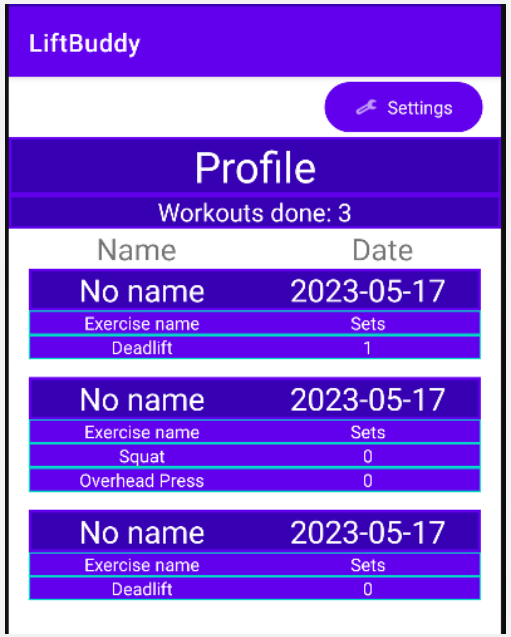
SQLITE DATABASE

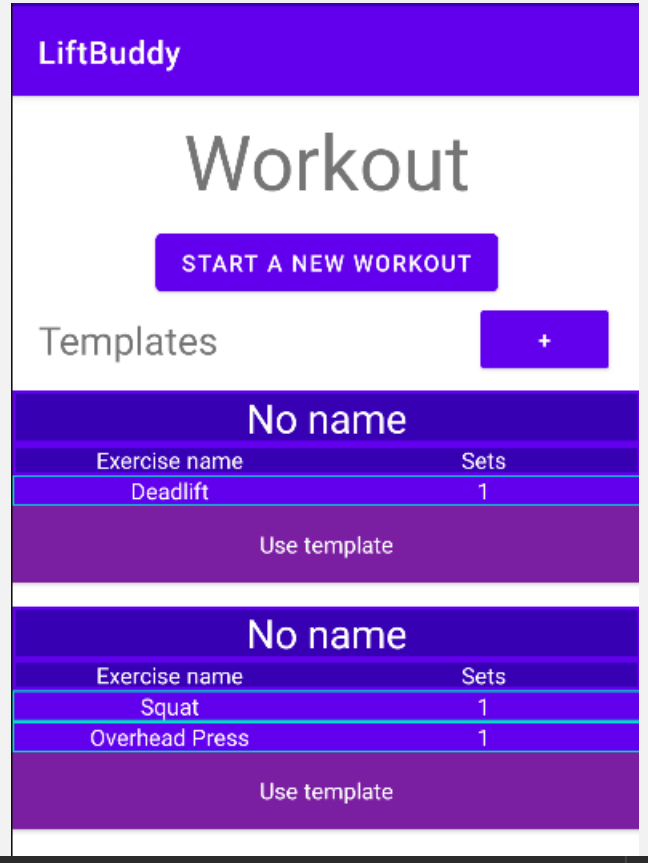
- One database containing 2 tables – Workouts and Templates
- Uses threads to save and load from database to prevent app hanging

```
//Saves templates into templates table in database
private fun saveTemplate(doneWorkoutObject: WorkoutClass) {
    val saveTemplateThread = Thread {
        db = DatabaseHelper( context: this,  factory: null)
        /// creating variables for values
        val workoutName = doneWorkoutObject.name
        val date = doneWorkoutObject.date
        val objectBlob = makebyte(doneWorkoutObject)
        db.addTemplate(date, workoutName, objectBlob)
    }
    startThread(saveTemplateThread)
}
```

HISTORY AND WORKOUT TEMPLATES

- Get dynamically updated when new workout or template is completed
- Loaded from database or when an intent is received






```
if (result.resultCode == Activity.RESULT_OK) {
    val data: Intent? = result.data
    if (data != null) {
        // Converts extra back into object
        doneWorkoutObject = getSerializable(data, name: "doneWorkoutObject", WorkoutClass::class.
        // Programmatically adds history
        dynamicProfileLayout.addNewHistory(doneWorkoutObject)
        workoutsNumber++
        updateCounter()
    }
} else {
```

OVERALL

WHAT DID THE APP ACHIEVE

It works as a good workout tracking app

The UI is easy to understand

Works fast and does not hang

App is compatitble with all devices down to SDK 29

Support light and dark mode

Input validation in place to ensure crashes can't be caused by user input

WHAT COULD BE IMPROVED

- Ability to see weight and reps performed
- Ability to add new exercises to the exercise library
- More attractive UI
- Removal of all Edit Text field listeners when app is paused to improve performance
- Ability to send a template to a friend as an SMS, which they can then use to open the template on their app
- Test app with lower API

SECURITY

WHAT WAS ACHIEVED

- Used up to date coding practices
- There is no deprecated code
- No personal data is saved
- All user data is stored locally using an SQL database

WHAT COULD BE IMPROVED

- Vulnerability scanner can be ran against the app to see if there is any potential vulnerabilities

PERFORMANCE

WHAT WAS ACHIEVED

- The app uses multithreading to save and load into the Database
- The listeners for certain buttons in the UI are released are restored based on app lifecycle.

WHAT COULD BE IMPROVED

- Removal of all listeners based on app lifecycle
- Checks to prevent database from being loaded when not needed

THANKS FOR LISTENNING

REFERENCES

- annianni (2022) *How to create a new fragment in Android Studio?*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/how-to-create-a-new-fragment-in-android-studio/ (Accessed: 10 May 2023).
- ayushpandey3july (2022) *Bottom Navigation Bar in Android using Kotlin*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/bottom-navigation-bar-in-android-using-kotlin/ (Accessed: 10 May 2023).
- baeldung (2023) *Get the current date/time in Kotlin*, *Baeldung on Kotlin*. Available at: https://www.baeldung.com/kotlin/current-date-time (Accessed: 17 May 2023).
- Coding In Flow (2017) *Custom dialog + sending information to Activity - Android studio tutorial*, *YouTube*. Available at: https://www.youtube.com/watch?v=ARezg1D9Zd0 (Accessed: 12 May 2023).
- Daily Coding (2021) *Android activityresultlauncher | howto start activity for result | startactivityforresult deprecated*, *YouTube*. Available at: https://www.youtube.com/watch?v=DfDj9EadOLk (Accessed: 01 May 2023).

REFERENCES

- MaterialDesign (2022) *Material design*. Available at: https://m2.material.io/components/bottom-navigation/android#using-bottom-navigation (Accessed: 17 May 2023).
- *Remove elements from a list while iterating in Kotlin* (2021) *Techie Delight*. Available at: https://www.techiedelight.com/remove-elements-from-list-while-iterating-kotlin/ (Accessed: 13 May 2023).
- scoder13 (2021) *Android sqlite database in Kotlin*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/android-sqlite-database-in-kotlin/ (Accessed: 15 May 2023).
- Seyedi, M. (2022) *GetSerializableExtra and getparcelableextra deprecated, what is the alternative?*, *Stack Overflow*. Available at: https://stackoverflow.com/questions/72571804/getserializableextra-and-getparcelableextra-deprecated-what-is-the-alternative (Accessed: 11 May 2023).
- Smith, S. (2022) *Moving from android startactivityforresult to registerforactivityresult*, *Medium*. Available at: https://medium.com/@steves2001/moving-from-android-startactivityforresult-to-registerforactivityresult-76ca04044ff1 (Accessed: 10 May 2023).