Deliverable 4

Quantum Quants

Housefy

Team 04

Team members

Member 1: Wenyuan Yu, Student ID: n01403697

Member 2: Kyrylo Lvov, Student ID: n01414058

Member 3: Artem Tsurkan, Student ID: n01414146

Table of Contents

1. Brief Description of the Project

Housefy project consists of two parts - a mobile application and an IoT (hardware) device. Both are designed to work together and provide a Smart Home solution.

2. Member's Info and Participation Table

Name	ID	Signature	Effort
Kyrylo Lvov	n01414058	Kyrylo Lvov	100%
Wenyuan Yu	n01403697	Wenyuan Yu	100%
Artem Tsurkan	n01414146	Artem Tsurkan	100%

3. GitHub Repo Link

Android app: https://github.com/WenyuanYu3697/housefy
Backend: https://github.com/WenyuanYu3697/housefy-asp.net-core-web-app

4. Sprint Goals

Our goal for this sprint is to reach a state of project completion by resolving all remaining tasks, addressing our technical debt, and implementing necessary refactoring and polishing, uploading the app to the Google Play and formally closing the project as much as it can be done in the current conditions of our Agile development process simulated for learning purposes.

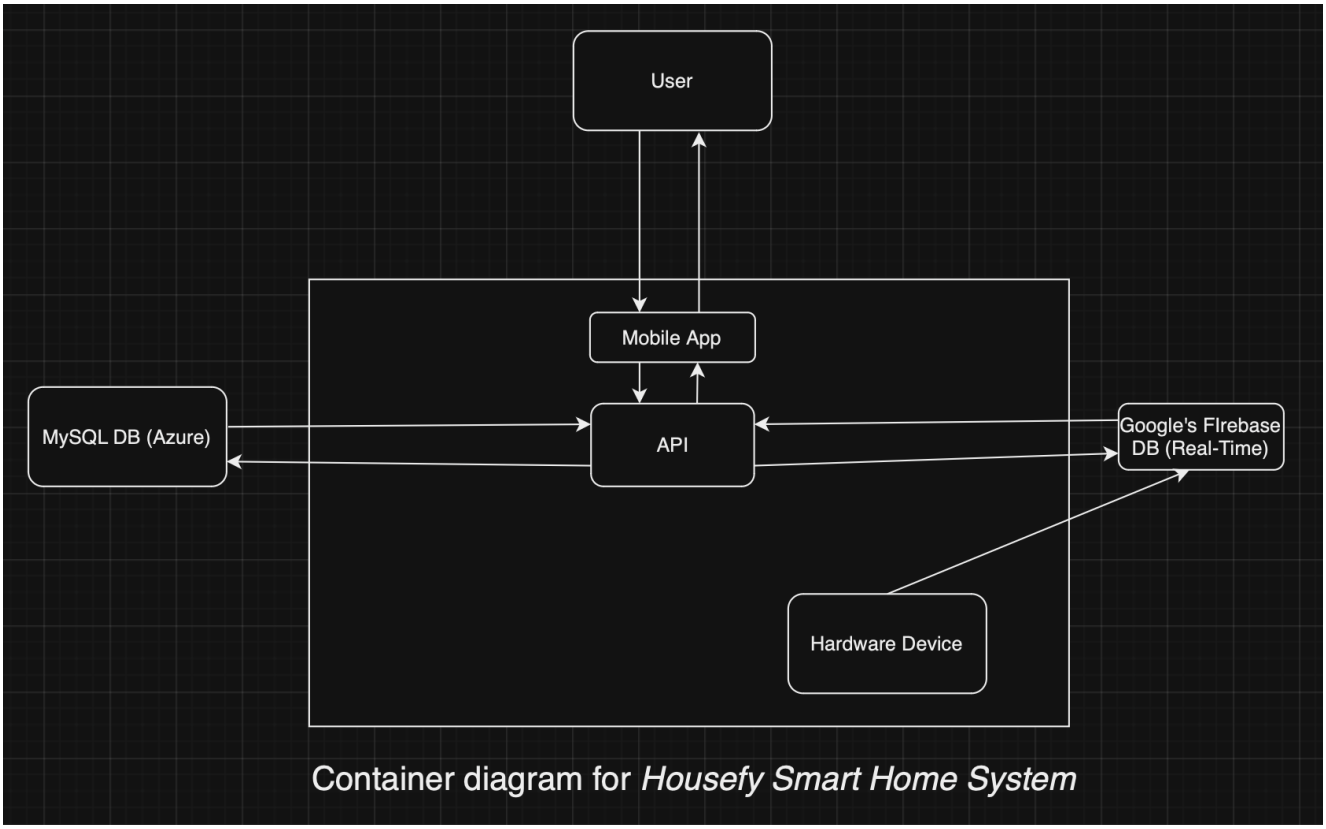
To achieve this goal, the team will need to prioritize the remaining tasks and technical debt issues. Technical debt, often incurred during the rush of early development, includes sub-optimal code or rushed design decisions that need to be revisited. Addressing this debt will improve the code's maintainability and reduce the likelihood of future issues.

The refactoring component of the goal refers to altering the code structure without changing its external behavior, thereby improving its internal quality. The team will need to identify areas where the code can be made more efficient, cleaner, or easier to understand. This will make the code more manageable and less prone to bugs, enhancing its overall stability.

Polishing refers to the process of refining and finalizing the product. This could involve fine-tuning the user interface, fixing minor bugs etc. The aim here is to make the product the best as it can be before delivery.

5. Container Diagram

Link: https://bit.ly/housefy-container-diagram



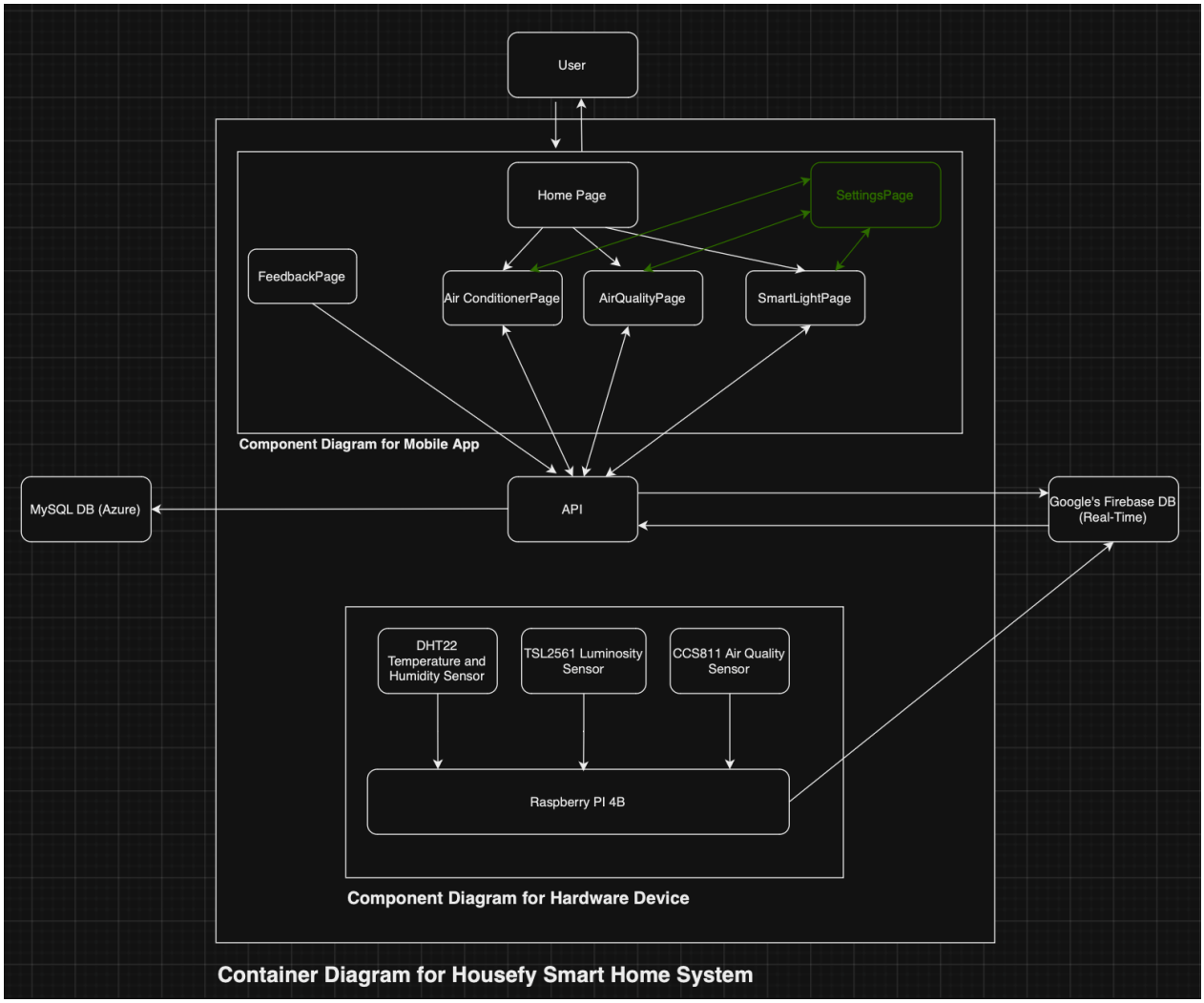
<u>Justification (TBU):</u>

https://www.youtube.com/watch?v=x2-rSnhpw0g

The diagram was built according to the guidelines provided in the the video from the link above shown in class.

6. Component Diagrams

Link: https://tinyurl.com/componentdiagrams

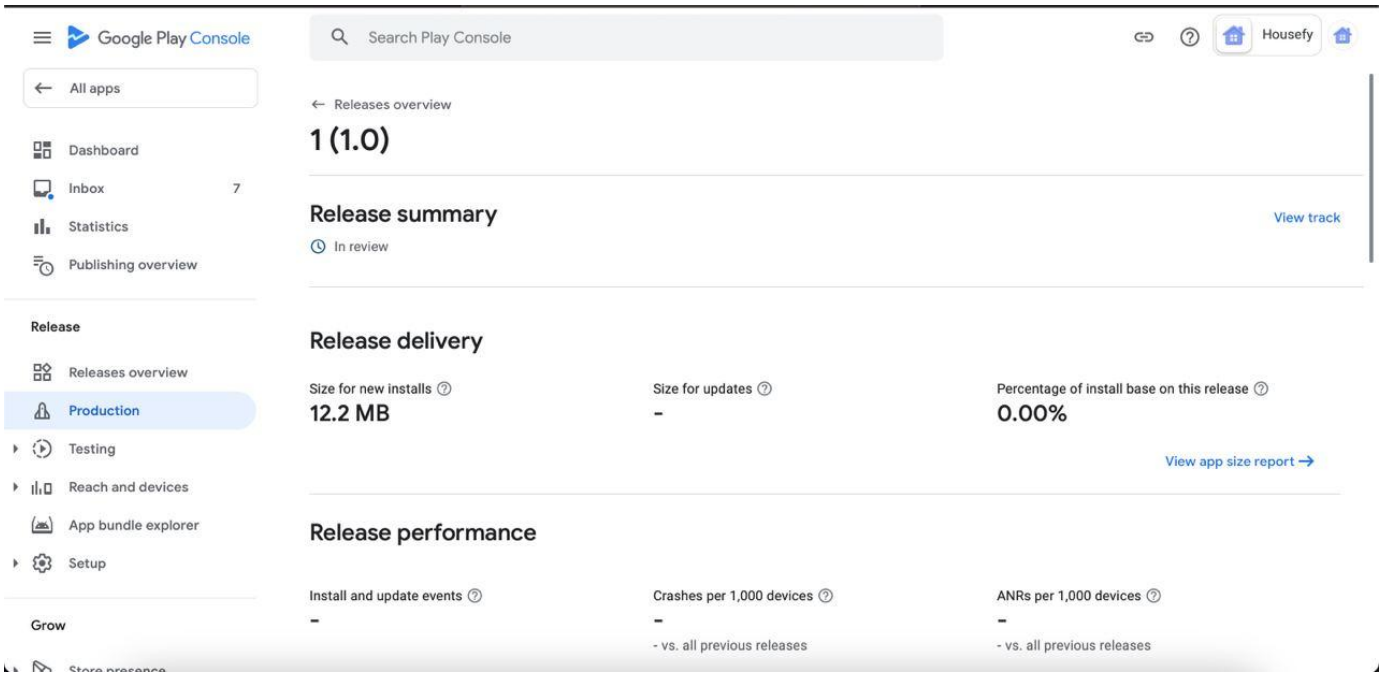


Justification (TBU):

https://www.youtube.com/watch?v=x2-rSnhpw0g

The diagram was built according to the guidelines provided in the the video from the link above shown in class.

7. Screenshot of app submission to Google Play



8. Offline Mode Functionality

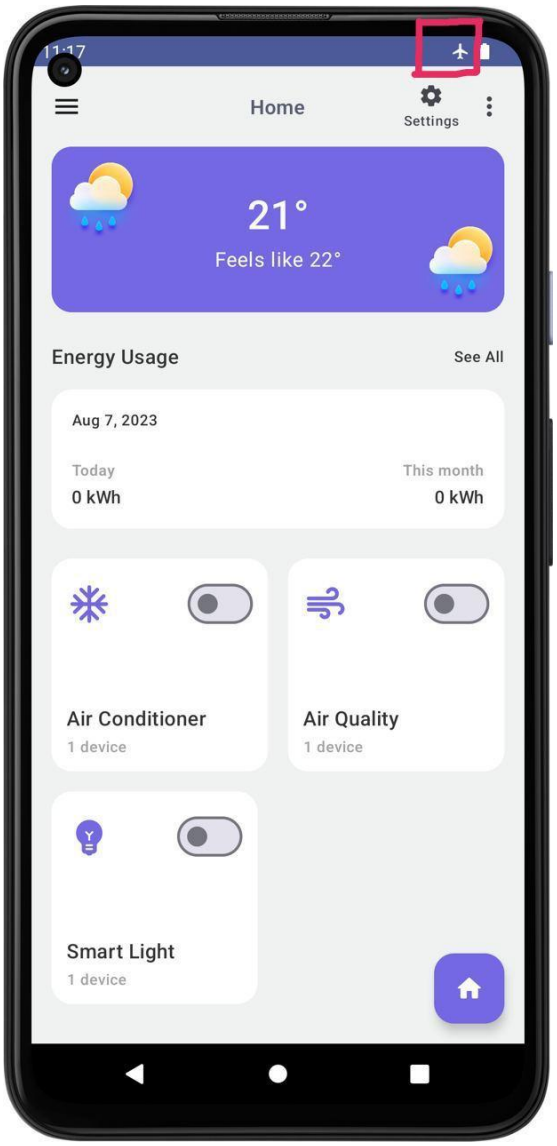
The weather is saved into Shared Preference. If you run the app in the airplane mode, the last saved weather will be shown before the internet connection was lost.

Code snippet:

```
fun saveWeatherData(context: Context, temp: String, feelsLike: String, icon: String) {
    val sharedPreferences = context.getSharedPreferences("weather_data", Context.MODE_PRIVATE)
    with(sharedPreferences.edit()) {
        putString("temp", temp)
        putString("feelsLike", feelsLike)
        putString("icon", icon)
        apply()
    }
}
fun getSavedWeatherData(context: Context): Map<String, String?> {
    val sharedPreferences = context.getSharedPreferences("weather_data", Context.MODE_PRIVATE)
    return mapOf(
        "temp" to sharedPreferences.getString("temp", "N/A"),
        "feelsLike" to sharedPreferences.getString("feelsLike", "N/A"),
        "icon" to sharedPreferences.getString("icon", "N/A")
    )
}
```

Path to file:

housefy/src/main/java/ca/quantum/quants/it/housefy/asynctasks/FetchWeatherTask.kt

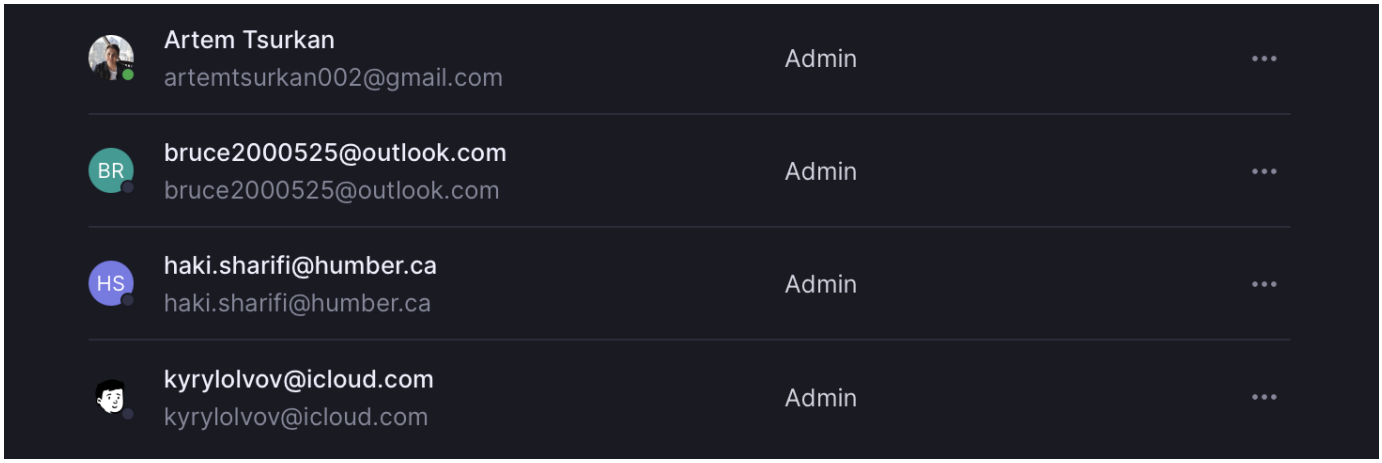


The weather is also the runtime permission that will be asked for during the first launch.

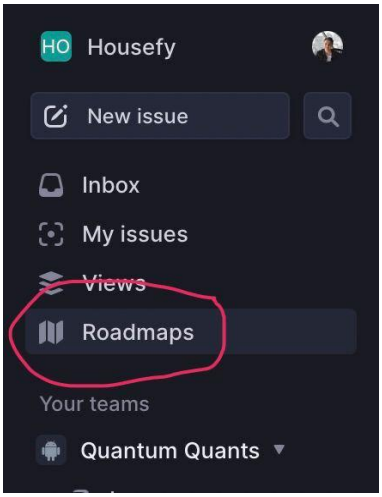
9. Scrum Dashboard w./ stories, tasks, owner, status, start/end date, size

https://linear.app/housefy/roadmap/all

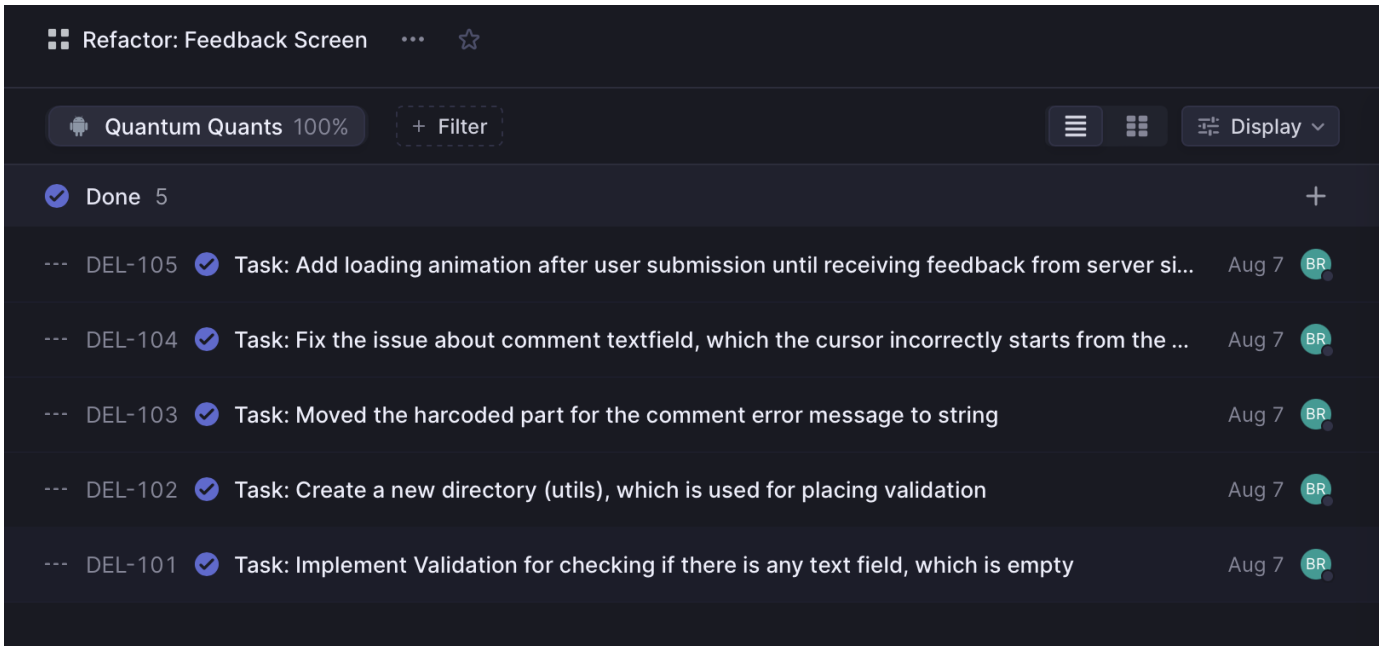
Screenshot of invitation:

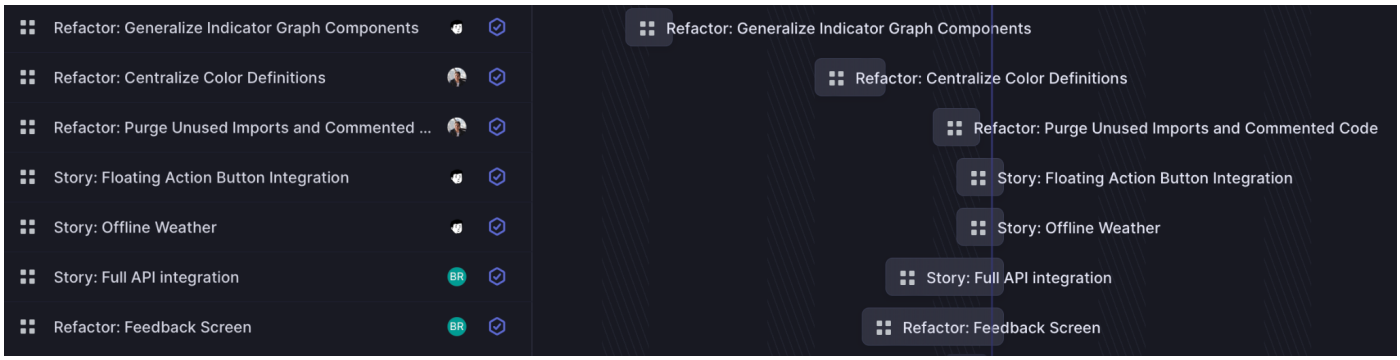


The stories are located in the “Roadmaps” section of the side menu on the left:



You can click on each story to view the tasks and details. For example:





10. Post Mortem / Project Review meeting

Link:

https://full-morning-7fc.notion.site/Post-mortem-meeting-181cda5698b34663b656690be6f87773?pvs=4

Post-mortem meeting

💡 **Notion Tip:** Use this template to analyze a project's success (or lack thereof) and create action items that will help prevent repeat mistakes. You can use this ✏️ Meeting Notes template to capture all key highlights from your post-mortem meeting.

Meeting details

Project:
Housefy

Team Members:
Kyrylo Lvov, Artem Tsurkan, Wenyuan Yu

Note taker:
Artem Tsurkan

Project Performance

Cost — Initial planned budget was not exceeded

Schedule — Things went well as per the agreed timeline

Quality — Overall, the quality can be assessed as good.

Time Management

Major points in the timeline of the project, achievements and obstacles experienced.

👇Click `Add a new milestone` to create a new milestone

Add a new milestone

11. Addressing technical debt

Technical debt was addressed by prioritizing refactoring of the code into a cleaner code, and eliminating the code that follows bad programming practices. Removing commented out code, unused import statements and clearing out the warnings raised by Android Studioю. In addition, a considerable effort was spent on UI enhancements.

12. Two areas of refactoring

Example 1 - Moving the functionality to set the threshold to the settings page and changing the method of setting the threshold from text field to slider.

Green - Added

Red - Removed

Git provides a great way to illustrate the changes. Two commits in which this refactoring was done are:

1. REFACTORED & CHANGED by Artem Tsurkan on August 3
2. ADDED by Artem Tsurkan on August 4

Main changes are also explained below.

Changes and Removals in the Energy Consumption Screen:

```
@Composable

fun EnergyConsumptionPage() {

    val context = LocalContext.current

    val preferences = context.getSharedPreferences("housefy_preferences",
Context.MODE_PRIVATE)

    val threshold = remember {

        mutableStateOf(preferences.getFloat("energyConsumptionThreshold", 0.7f))

    }


    Box(

        modifier = Modifier
```

```
            .fillMaxSize()
            .background(color = BackgroundGrey),
        contentAlignment = Alignment.Center
    ) {
        Chart(
            data = listOf(
                Pair(0.9f, 1),
                Pair(0.8f, 5),
                Pair(0.7f, 6),
                Pair(0.7f, 7),
            ), max_value = 50, threshold = threshold.value
            ), max_value = 50, threshold = threshold
        )
        ThresholdSettings(threshold) { newThreshold ->
            threshold = newThreshold
        }

    }
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun ThresholdSettings(threshold: Float, onThresholdChange: (Float) -> Unit) {
```

```
    Box(
        modifier = Modifier
            .padding(30.dp, 70.dp, 30.dp, 500.dp)
            .absoluteOffset(0.dp, 460.dp)
            .fillMaxWidth()
            .fillMaxHeight()
            .clip(RoundedCornerShape(5))
            .background(color = Color.White),
        contentAlignment = Alignment.Center
    ) {
        Column() {
            Text(
                text = stringResource(R.string.set_threshold),
                color = TextBlack,
                textAlign = TextAlign.Start,
            )
        }

        Column() {
            TextField(
                value = threshold.toString(),
                onValueChange = { onThresholdChange(it.toFloatOrNull() ?: threshold)
},
```

```
                label = { Text(text = stringResource(R.string.set_threshold)) },
                modifier = Modifier
                    .background(color = Color.White),
            )
        }
    }
}
```

Changes and Removals in the Settings Screen:

```
@SuppressLint("MissingPermission")
 @Composable
fun SettingsPage() {
…
var energyConsumptionThreshold by remember {
mutableStateOf(preferences.getFloat("energyConsumptionThreshold", 0f)) }
…
SettingsRow(
                title = "Energy Consumption Threshold",
                control = {
                    Slider(
                        value = energyConsumptionThreshold,
                        onValueChange = { newValue ->
                            energyConsumptionThreshold = newValue
```

```
                    preferences.edit {
                        putFloat("energyConsumptionThreshold", newValue)
                    }
                },
                steps = 5,
                valueRange = 0f..1000f, // Adjust this range based on your needs
                modifier = Modifier.width(150.dp),
                colors = SliderDefaults.colors(
                    thumbColor = Purple,
                )
            )
        }
    )
    }
  }
}
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun ThresholdSettings(threshold: Float, onThresholdChange: (Float) -> Unit) {
    Box(
        modifier = Modifier
            .padding(30.dp, 70.dp, 30.dp, 500.dp)
```

```
            .absoluteOffset(0.dp, 460.dp)
            .fillMaxWidth()
            .fillMaxHeight()
            .clip(RoundedCornerShape(5))
            .background(color = Color.White),
        contentAlignment = Alignment.Center
    ) {
        Column() {
            Text(
                text = stringResource(R.string.set_threshold),
                color = TextBlack,
                textAlign = TextAlign.Start,
            )
        }
        Column() {
            TextField(
                value = threshold.toString(),
                onValueChange = { onThresholdChange(it.toFloatOrNull() ?: threshold)
},
                label = { Text(text = stringResource(R.string.set_threshold)) },
                modifier = Modifier
                    .background(color = Color.White),
            )
```

```
        }
    }
}
```

Example 2 - Extracting the common functionality between the AirConditionerGraph and AirQualityGraph components into a more general, reusable component called IndicatorGraph. This was done to enhance modularity, reduce redundancy and facilitate code maintainability.

The commit in which this refactoring was done is:

> REFACTORED by Kyrylo Lvov on July 23
>
> Link: https://bit.ly/kyrylo_refactoring_commit

housefy/src/main/java/ca/quantum/quants/it/housefy/components/air_quality/AirQualityGraph.kt - deleted

```diff
@Composable
- fun AirConditionerGraph(
    indicatorValue: Int,
-   canvasSize: Dp = 300.dp,
    maxIndicatorValue: Int = 40,
    backgroundIndicatorColor: Color = MaterialTheme.colorScheme.onSurface.copy(alpha = 0.1f),
) {
    var allowedIndicatorValue by remember {
        mutableStateOf(maxIndicatorValue)
```

```diff
@Composable
+ fun IndicatorGraph(
+     foregroundIndicatorColor: Color = Color(0xFF7468E4),
    indicatorValue: Int,
    maxIndicatorValue: Int = 40,
    backgroundIndicatorColor: Color = MaterialTheme.colorScheme.onSurface.copy(alpha = 0.1f),
+     canvasSize: Dp = 300.dp,
+     indicatorText: @Composable (Int) -> Unit
) {
    var allowedIndicatorValue by remember {
        mutableStateOf(maxIndicatorValue)
```

```diff
@@ -83,14 +85,13 @@ fun AirConditionerGraph(
                foregroundIndicator(
                    sweepAngle = sweepAngle,
                    componentSize = componentSize,
+                   indicatorColor = foregroundIndicatorColor,
                )
            },
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
-       EmbeddedElements(
-           aqiValue = receivedValue,
-       )
+       indicatorText(receivedValue)
    }
}
```

```diff
    drawArc(
        size = componentSize,
-       color = Color(0xFF7468E4),
+       color = indicatorColor,
        startAngle = 150f,
        sweepAngle = sweepAngle,
        useCenter = false,
@@ -134,18 +136,4 @@ fun DrawScope.foregroundIndicator(
            y = (size.height - componentSize.height) / 2f
        )
    )
-}
-
-@Composable
-fun EmbeddedElements(
-    aqiValue: Int,
-) {
-    Text(
-        text = "$aqiValue°C",
-        color = Color(0xFF353336),
-        fontSize = 64.sp,
-        textAlign = TextAlign.Center,
-        fontWeight = FontWeight.Bold,
-        modifier = Modifier.offset(y = (-8).dp),
-    )
}
```

housefy/src/main/java/ca/quantum/quants/it/housefy/pages/AirConditionerPage.kt

```
@Composable
@@ -47,8 +52,18 @@ fun AirConditionerPage() {
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
-       AirConditionerGraph(
            indicatorValue = 25,
        )

        Row(
```

```
@Composable
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
+       IndicatorGraph(
            indicatorValue = 25,
+           indicatorText = {
+               Text(
+                   text = "25°C",
+                   color = Color(0xFF353336),
+                   fontSize = 64.sp,
+                   textAlign = TextAlign.Center,
+                   fontWeight = FontWeight.Bold,
+                   modifier = Modifier.offset(y = (-8).dp),
+               )
+           }
        )

        Row(
```

housefy/src/main/java/ca/quantum/quants/it/housefy/pages/AirQualityPage.kt:

```
@Composable
@@ -101,9 +102,19 @@ fun AirQualityPage() {
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        item {
-           AirQualityGraph(
                indicatorValue = value,
-               foregroundIndicatorColor = getAQIColor(value)
```

```
@Composable
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        item {
+           IndicatorGraph(
                indicatorValue = value,
+               foregroundIndicatorColor = getAQIColor(value),
+               indicatorText = {
+                   Text(
+                       text = "$value",
+                       color = getAQIColor(value),
+                       fontSize = 84.sp,
+                       textAlign = TextAlign.Center,
+                       fontWeight = FontWeight.Bold,
+                       modifier = Modifier.offset(y = (-8).dp),
+                   )
+               }
```

```
150     fun calculateAQI(co2: Float): Int {
151         return (co2 / 10).roundToInt()
152     }
```

```
161     fun calculateAQI(co2: Float): Int {
162         return (co2 / 10).roundToInt()
163 + }
164 +
165 + fun getAQIColor(aqi: Int): Color {
166 +     return when {
167 +         aqi in 0..25 -> Color(0xFF8CD456)
168 +         aqi in 26..50 -> Color(0xFFFFE24C)
169 +         aqi in 51..75 -> Color(0xFFFFA500)
170 +         else -> Color(0xFFFF0000)
171 +     }
172 + }
173 +
174 + @Composable
175 + fun getAQIDescription(aqi: Int): String {
176 +     return when {
177 +         aqi in 0..25 -> stringResource(R.string.excellent)
178 +         aqi in 26..50 -> stringResource(R.string.good)
179 +         aqi in 51..75 -> stringResource(R.string.moderate)
180 +         else -> stringResource(R.string.poor)
181 +     }
182     }
```

13. Suggestions for the instructor

Dear Professor Haki,

As a team, we appreciate the modern and real-life-oriented approach this course is being delivered in, particularly:

- Real-world context
- Hands-on experience
- Scrum Ceremonies Simulations (daily stand-ups, retrospectives etc.)
- Modern Tools and especially the freedom of choice between them (Trello, Miro, Asana, Linear, Notion etc.)
- Role Responsibilities simulation (Scrum Master, Technical Team, Product Owner etc.)

However, with all due respect, we have a humble suggestion to put forward. It pertains to the apparent clash between the course's expectation for us to multitask and Agile's preference against it. The challenge is that many of us are currently enrolled in seven technical courses this semester, which demand considerable time and focus.

We completely understand if the nature of academic course schedules might not lend itself to easy changes in this aspect. Nonetheless, we felt it would be helpful

to bring this to your attention, especially since your openness to feedback has always been commendable. Our suggestion is made with the intention of further enriching the learning experience, and we are grateful for your understanding and receptivity.

We appreciate your continuous support and your passion for teaching, which clearly shines through in every class.

Best Regards,

Quantum Quants (Team 4)

14. Additional Notes (Important)

- **Regarding pictures (6 pictures are required):**
  We are using dynamic pictures in our application. The weather component on Home fragment contains 4 pictures which switch depending on OpenWeatherApi response. The other 2 pictures are on Smart Light fragment, when you toggle the switch, image changes.

- **Regarding menu:**
  As of now, Jetpack Compose (the new way to style Android applications which are build using Kotlin) doesn't come with always, ifroom, never menu items, that's why we needed to create them ourselves, the functionality is the same, you can find the code in /components/navigation/Menu.kt

- **Regarding landscape mode:**
  In the new paradigm (Jetpack Compose), rather than having separate layout files for each orientation, Jetbrains suggest having one, which adapts to any landscape. We still implemented explicit change of layout to landscape mode using if - else statement on Splash and Home screens

```
fun HomePage(navController: NavHostController, snackbarHostState: SnackbarHostState) {
    val configuration = LocalConfiguration.current

    if (configuration.orientation == Configuration.ORIENTATION_LANDSCAPE) {
        HomePageLayout(navController, 16.dp, snackbarHostState)
    } else {
        HomePageLayout(navController, 24.dp, snackbarHostState)
    }
}
```

- **Regarding energy consumption screen:**

  Our Team member on the hardware course is not taking this course with us. His sensor is not ready, and most likely he won't be able to do the capstone project with us. Due to this reason, we will most likely opt-out of using the energy consumption screen in future. For now, we just leave it as an experimental / prototype feature.