Software Project

Team Name: TeamBot
Project Name: Cave Examination Bot
Group Number: 9

Student Names and IDs:
Adrian Portal Calcines - n01489363
Alfred Dowuona - N01490404
Ali Mohebi - N01477361
Hassan Noorani - N01485518

Project summary:

The project is an Android studio application that can display sensor readings gathered from a robot stored in a database. It has various screens such as main dashboard, configuration screen, data readings, robot controlling, notifications screen, location map and other minor functionality screens like login, registration and feedback. The app is as user-friendly as possible so the user can interact with the data and manage the robot easily. The app integrates and modifies a database, displays and manipulates data gathered from the robot within the app.

Participation and signatures:

Name	Student Id	Github Id	Signature	Effort
Adrian Portal C	n01489363	AdrianPortal9363	A.P.	100%
Hassan Noorani	N01485518	HassanNoorani5518/HaBNo11(gmail sign-in from android studio)	H.N	100%
Alfred Dowuona	N01490404	AlfredDowuona0404	A.D.	100%
Ali Mohebi	N01477361	AliMohebi7361	A.M.	100%

Github repo link:

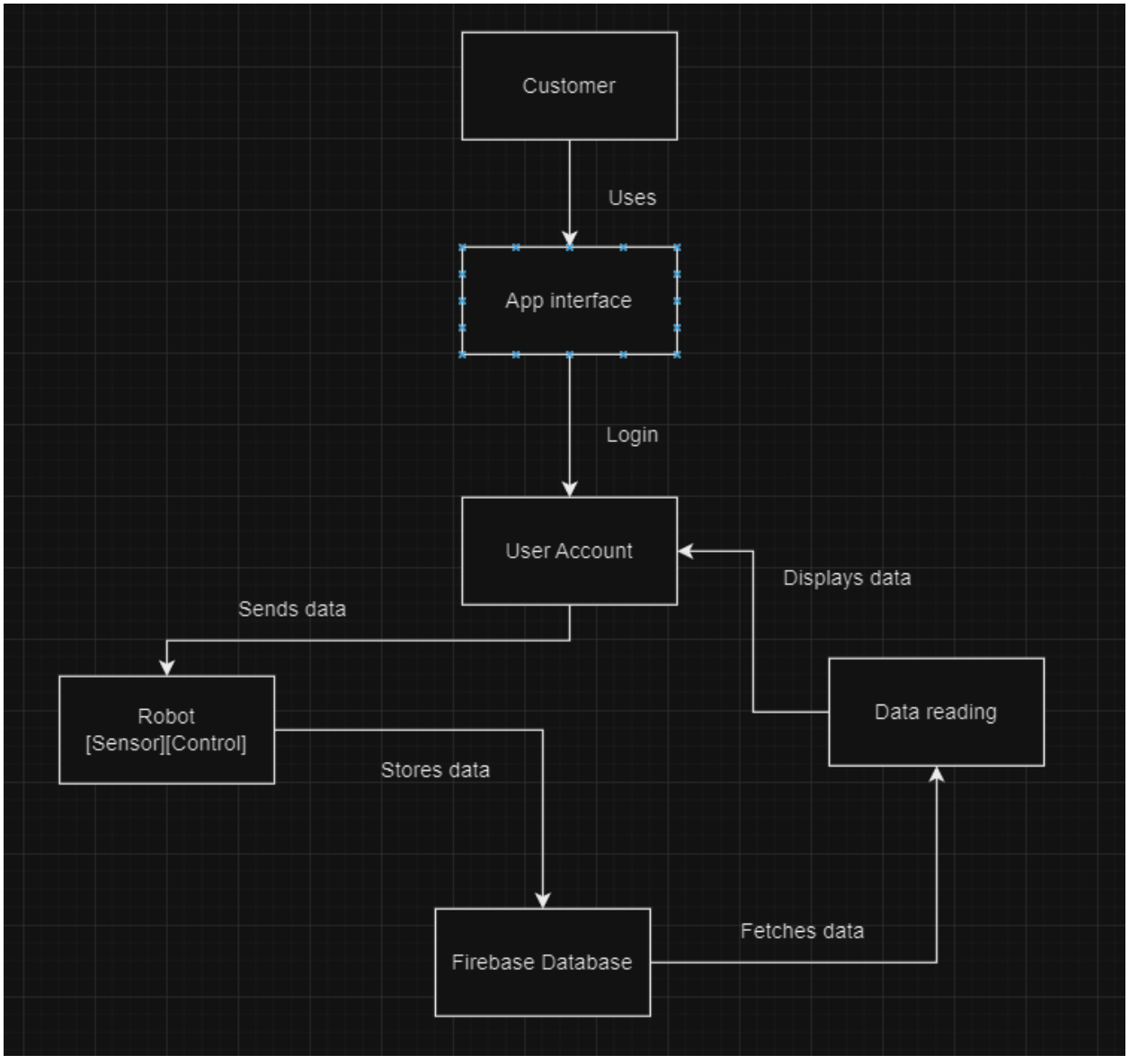
https://github.com/HassanNoorani5518/CaveExaminationBot

Sprint goals:

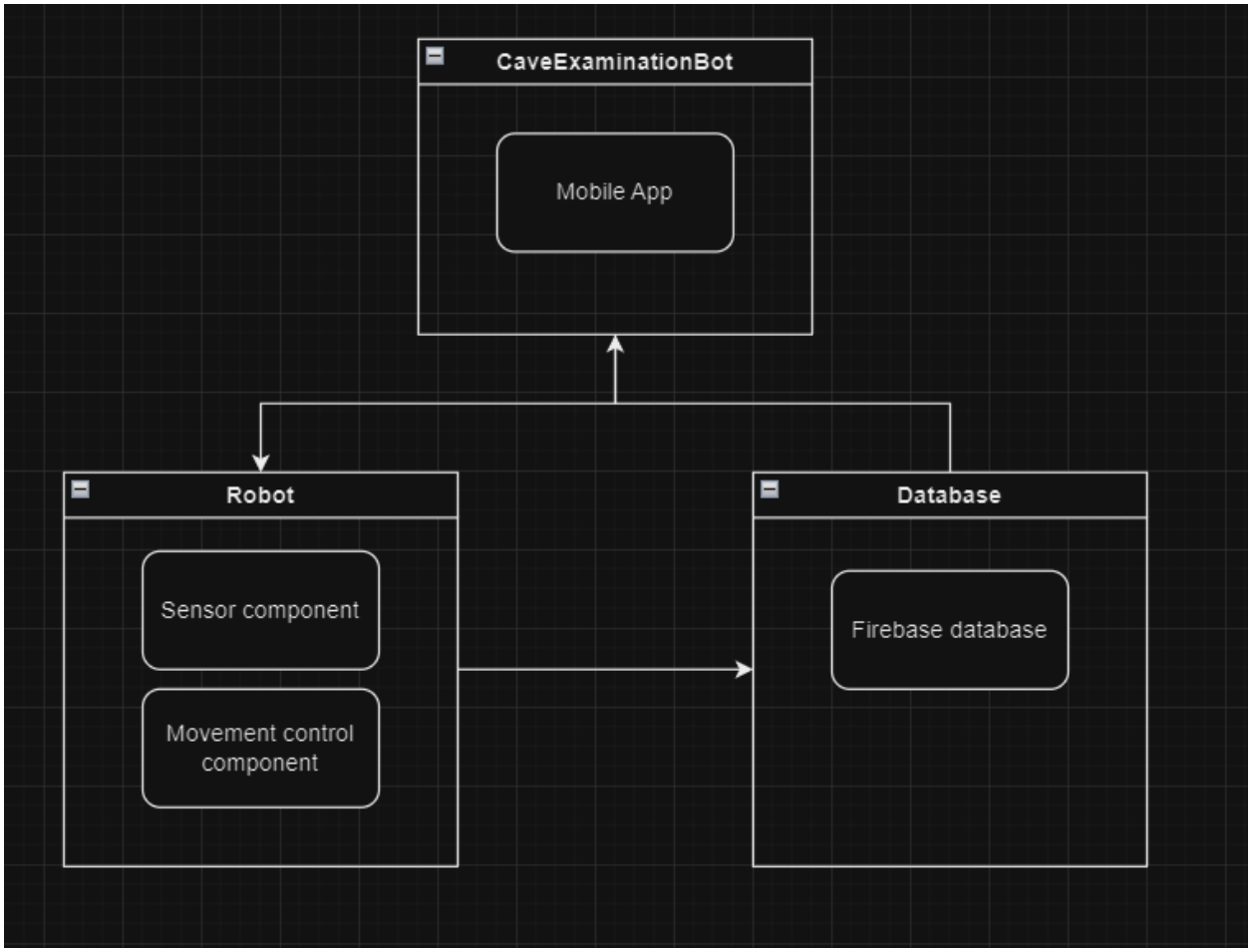
Our spring goal was to have a functioning app by the end of the deadline. More specifically we wanted a fully functional login/registration screen with proper validation checks. Our goal was to implement shared preference to every aspect of the app so it saves unique data to each unique user. We needed to make a robot controlling screens with functionalities as well as data reading and notifications screens. Further improve the map function and work on the feedback screen and adding a timer restriction to its submission. Another goal was to implement an offline feature that will work when the app is in airplane mode. Lastly, our expectations were to surround our code with test cases to validate the data shown to the user.

C4 Model:

Container Diagram:



Component Diagram:



Use of Alerts:

AlertDialog:

We used AlertDialog in the feedback screen when a user submits the forum.

Snackbar:

We used the snackbar to warn the user when pressing the refresh button before refreshing notifications.

Toast:

We use the toast feature when the user logs in and error handling for the user.

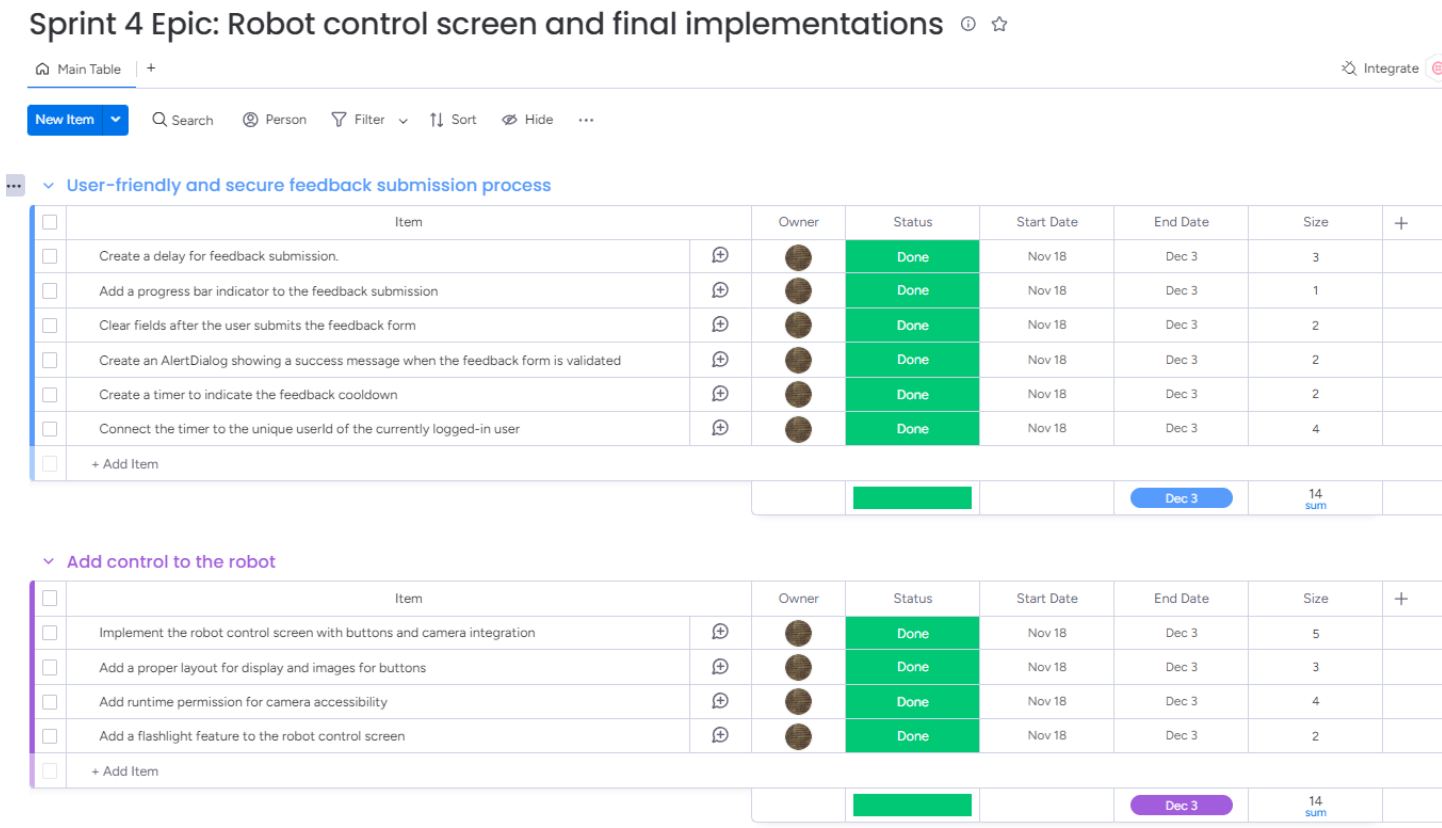
Notification:

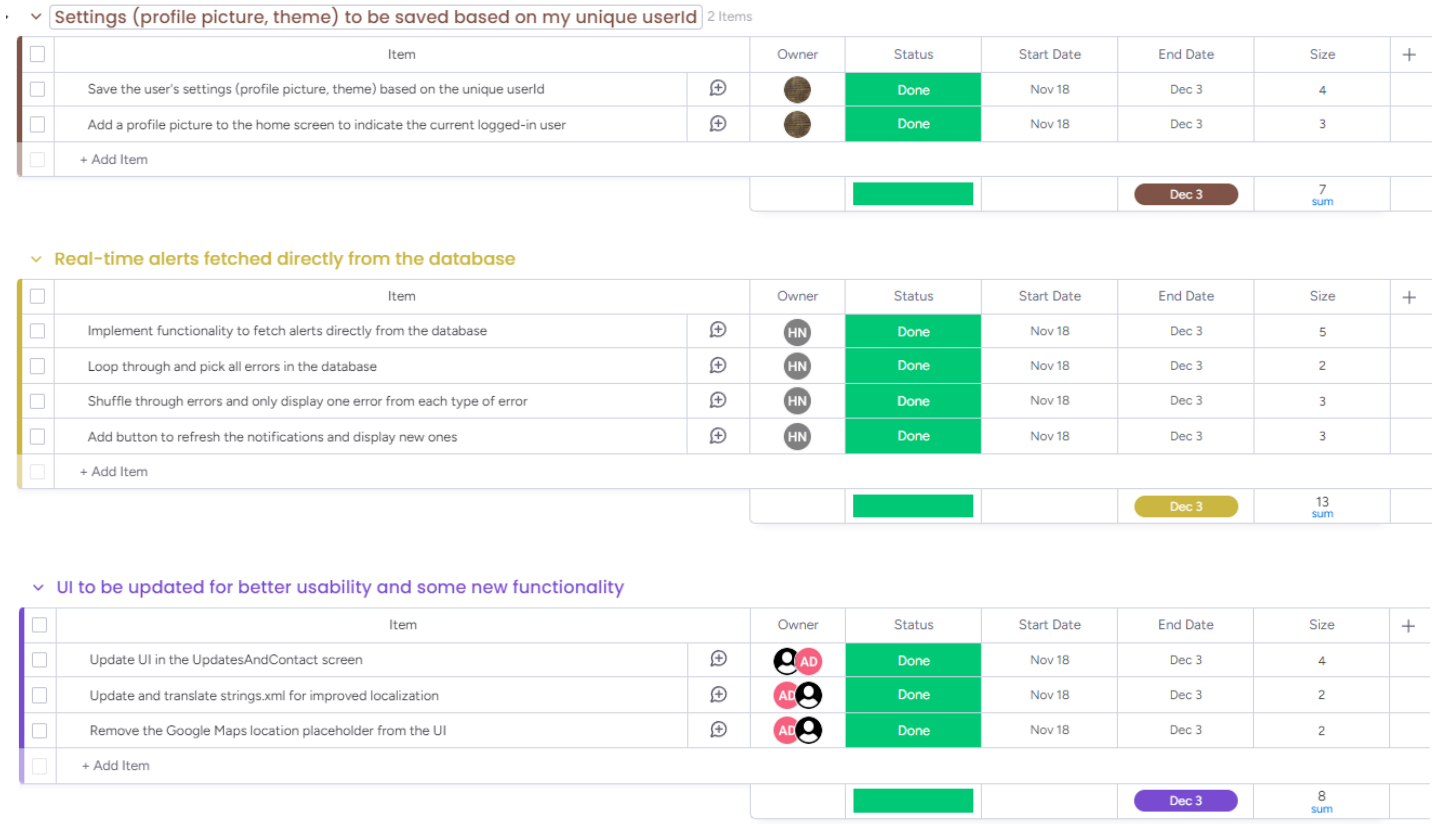
We have a notification screen showing different notifications to the user based on the robot's input and data on the database.

Offline feature:

We will have a section where the user can save the sensor data gathered by the robot and later review it in one of the screens, for analysing and reading. This will be using shared preference and the unique user can access his/her data.

Scrum dashboard:





Settings (profile picture, theme) to be saved based on my unique userId 2 Items

Item	Owner	Status	Start Date	End Date	Size
Save the user's settings (profile picture, theme) based on the unique userId		Done	Nov 18	Dec 3	4
Add a profile picture to the home screen to indicate the current logged-in user		Done	Nov 18	Dec 3	3
+ Add Item					
				Dec 3	7 sum

Real-time alerts fetched directly from the database

Item	Owner	Status	Start Date	End Date	Size
Implement functionality to fetch alerts directly from the database	HN	Done	Nov 18	Dec 3	5
Loop through and pick all errors in the database	HN	Done	Nov 18	Dec 3	2
Shuffle through errors and only display one error from each type of error	HN	Done	Nov 18	Dec 3	3
Add button to refresh the notifications and display new ones	HN	Done	Nov 18	Dec 3	3
+ Add Item					
				Dec 3	13 sum

UI to be updated for better usability and some new functionality

Item	Owner	Status	Start Date	End Date	Size
Update UI in the UpdatesAndContact screen	AD	Done	Nov 18	Dec 3	4
Update and translate strings.xml for improved localization	AD	Done	Nov 18	Dec 3	2
Remove the Google Maps location placeholder from the UI	AD	Done	Nov 18	Dec 3	2
+ Add Item					
				Dec 3	8 sum

Post-mortem:

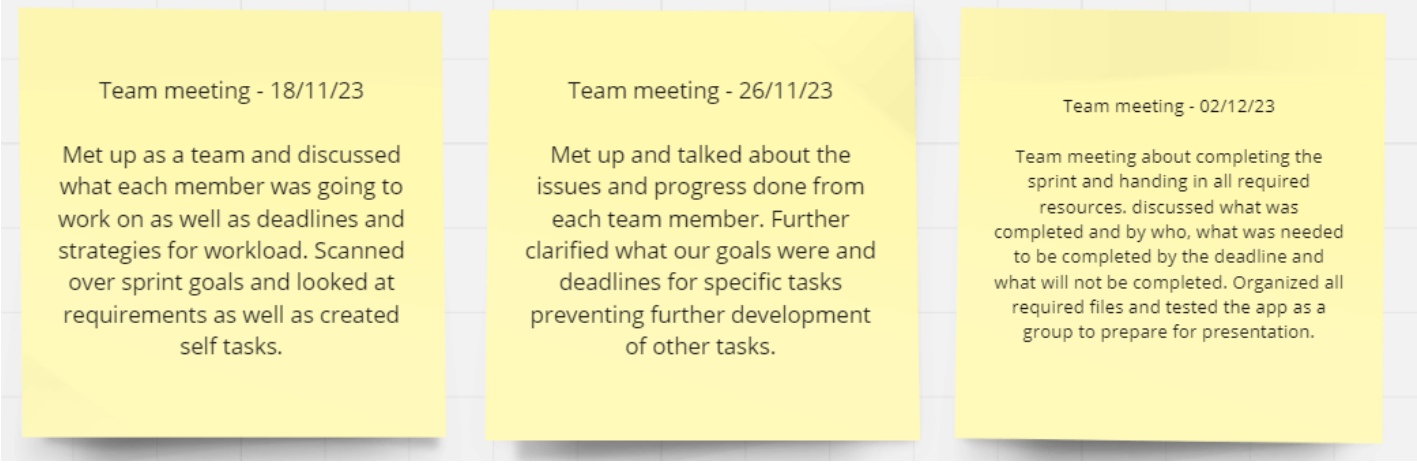
The CaveExaminationBot project was overall a good performance that needed a bit more work. In terms of cost the project was free, it has good performance for being free, multiple features that can be used and enough scope that more can be added on. Schedule wise the project was built over a 3 month period, for first time developers of a project of this scale we believe we developed a good performing application. In terms of quality the app could use more work, a lot of features could be added and others further improved. All the app functionalities work and can be further expanded on, the UI and design of the app is good for the most part.

In terms of time management we all could have done better, most of the workload was left for the last day or 2. The team did not communicate strategies, workload and task assigning, or struggles/problems. Even through this we managed to execute a functional app with good performance.

There were no issues with quality due to mostly compromises. If there was a task that was too high in quality or took too much time we compromised and scaled it down to fit the time schedule. We adapted and prioritized performance/functionality instead of quality due to the team situation.

We learned overall that projects of this scale take time, patience and a lot of effort to complete. We should have managed our time better and got together more times as a team to discuss strategies. We left most issues and work to the last minute and rushed a lot of functionalities because of this. We made a lot of mistakes such as pushing code that was broken, merging branches that contradicted code and not following the design principles that make it easy for us to code. There are always improvements we can work on such as managing time better, working together and communicating, and assigning a team leader who can manage the project, implement and research design principles.

Project review meetings:



Technical debt:

We had technical debt from rushing sprint goals to match the deadline, things such as copy and pasting code instead of using design principles and implementing a lot of placeholders to stall having to implement the functionality. The way we tackled this issue is we handled it and implemented/fix the code in the following sprint. Another way we solved it was by compromising and lowering the quality of the code and functionality to a lesser (but working) version.

Area of refactoring:

```
public void populateData()
{
   DataGeneration dataGeneration = new DataGeneration();
   String time = generateUniqueKey("test");

   addItem("SensorData", time, "Time", dataGeneration.getCurrentTime());
   addItem("SensorData", time, "Temperature",
DataGeneration.generateTemperature());
```

```
    addItem("SensorData", time, "Gas Level",
DataGeneration.generateGasConcentration());
    addItem("SensorData", time, "Humidity", DataGeneration.generateHumidity());
    addItem("SensorData", time, "Air Pressure",
DataGeneration.generateAirPressure());
    addItem("SensorData", time, "Magnetic Field",
DataGeneration.generateMagneticField());
    addItem("SensorData", time, "Cave Integrity",
DataGeneration.generateCaveIntegrity());
}
```

```
public class DataGeneration
{
    private static final Random random = new Random();

    public static String generateTemperature() {
        // Range: 10 to 30 °C
        double i = 10 + random.nextDouble() * 20;
        return String.valueOf(i);
    }

    public static String generateGasConcentration() {
        // Assume normal air with a variation
        // Oxygen: 20.5 - 21.5%
        // Nitrogen: 77.5 - 78.5%
        // Argon: 0.8 - 1.2%
        double i = 20.5 + random.nextDouble() * 1;
        return String.valueOf(i);
    }
```

We created a separate class with the specific purpose of generating a number within a range for simulating various sensor data. This is done so the populateData function is more readable and any changes in the future to the ranges of numbers will not impact the original function.

```
private void toggleFlashlight() {
    try {
        if (isFlashAvailable()) {
            if (isFlashlightOn) {
                cameraManager.setTorchMode(cameraId, false);
                isFlashlightOn = false;
            } else {
                cameraManager.setTorchMode(cameraId, true);
                isFlashlightOn = true;
            }
        }
    } catch (CameraAccessException e) {
        e.printStackTrace();
    }
}
```

```
private boolean isFlashAvailable() {
    try {
        // Get the characteristics of the camera with the given ID
        CameraCharacteristics characteristics =
cameraManager.getCameraCharacteristics(cameraId);

        // Check if the flash unit is available
        Boolean flashAvailable =
characteristics.get(CameraCharacteristics.FLASH_INFO_AVAILABLE);

        return flashAvailable != null && flashAvailable;
    } catch (CameraAccessException e) {
        e.printStackTrace();
        return false;
    }
}
```

In this toggleFlashlight function we refactored the code that checks if the device supports flash and made it into its own function. The isFlashAvailable function will never change and it's good practice to isolate the function so it doesn't get altered in the future when called again.

Suggestions for instructor:

The ideas and execution we liked about the project was that it was project based, adding on to it every week and setting goals from the very beginning to challenge ourselves and have accurate expectations of what we will learn and be able to do by the end of the course. The content of the deliverables themselves were both challenging and made real-world implementations we see everyday in apps. The fact that all coding issues were left to us to solve and implement is good and helped us learn the material through debugging and working together as a team.

The things that should be done differently is how the project is organized as a team. The project is done in 3 months which means you need team members (at least 3 total), so if 2 or more team members don't decide to help the rest are doomed to fail or sacrifice other course's grades in order to succeed. It would be better if the professor was the project manager and as a group of 4 we present what each will work on based on the sprint goals, that way the professor can tell who worked on which tasks. The professor currently sees the commits, but behind the scenes they have no idea who works on the docs, presentations and who fixes the copy/pasted broken code other team members push.