

**Software Project Deliverable 4**  
Buckle Up Inc.  
Smart Aqua



**Team Number**

Group 6

**Student's names and ID's**

Alvaro Rodrigo Chavez Moya	N01455107
Denis Shwaloff	N01422583
Nicholas Dibiase	N01367109
Paolo Adrian Quezon	N01424883

## Table of Contents

Table of Contents.....	2
Project Description.....	3
Team Signature List.....	3
Members Info and Participation.....	3
GitHub Repository Link.....	4
Sprint Goals.....	4
C4 Model.....	5
Container Diagram.....	5
Component Diagram.....	5
Project submitted into Google Play.....	6
Offline Functionality.....	6
Runtime Permission Feature.....	6
Accessing location runtime permission.....	6
Validating location runtime permission.....	7
Breakdown of location runtime logic.....	7
Scrum Dashboard.....	8
Post-Mortem - Project Review Meeting.....	11
How did you address technical debt in your project.....	11
Refactoring.....	12
First Case:.....	12
Second Case:.....	14
Suggestions for future projects.....	17


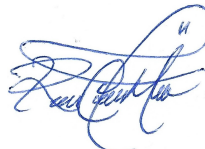


## Project Description

Smart Aqua is an intuitive Android application designed to provide real-time water quality monitoring along with proactive notifications that facilitate the upkeep of water treatment systems. The project takes an integrated approach, incorporating requirement gathering, user-centric user interface design, effective database architecture, and seamless sensor integration. We place a high priority on ongoing improvements and extensive testing, which are driven by valuable user feedback and ensure the exact and complete fulfillment of all stated goals and criteria.

## Team Signature List

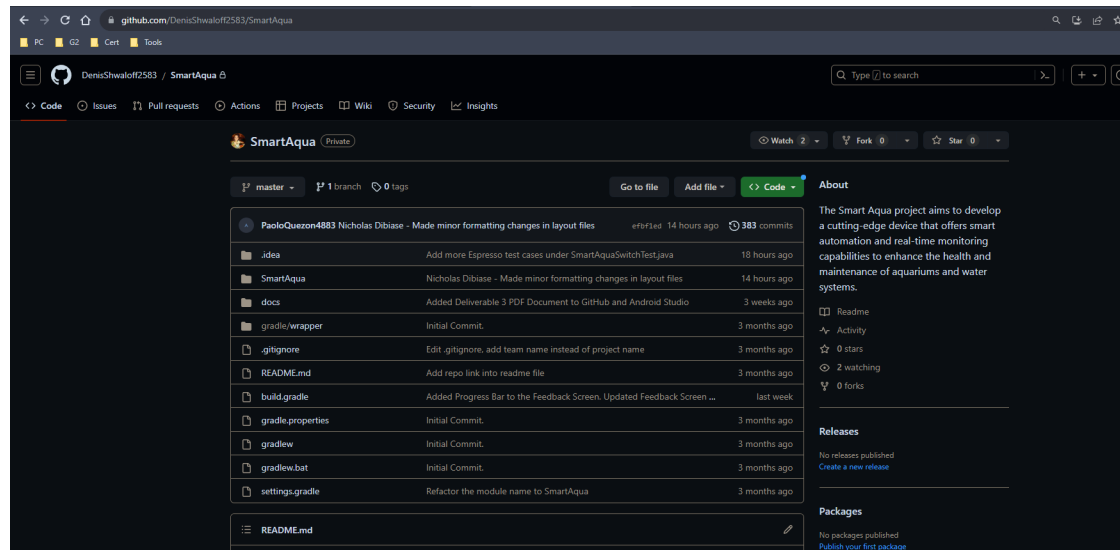
Alvaro Rodrigo Chavez Moya	Denis Shwaloff	Nicholas Dibiase	Paolo Adrian Quezon
			

## Members Info and Participation

Name	ID	Signature	Effort
Denis Shwaloff	N01422583		100
Alvaro Rodrigo Chavez Moya	N01455107		100
Nicholas Dibiase	N01367109		100
Paolo Adrian Quezon	N01424883		100

## GitHub Repository Link

<https://github.com/DenisShwaloff2583/SmartAqua.git>



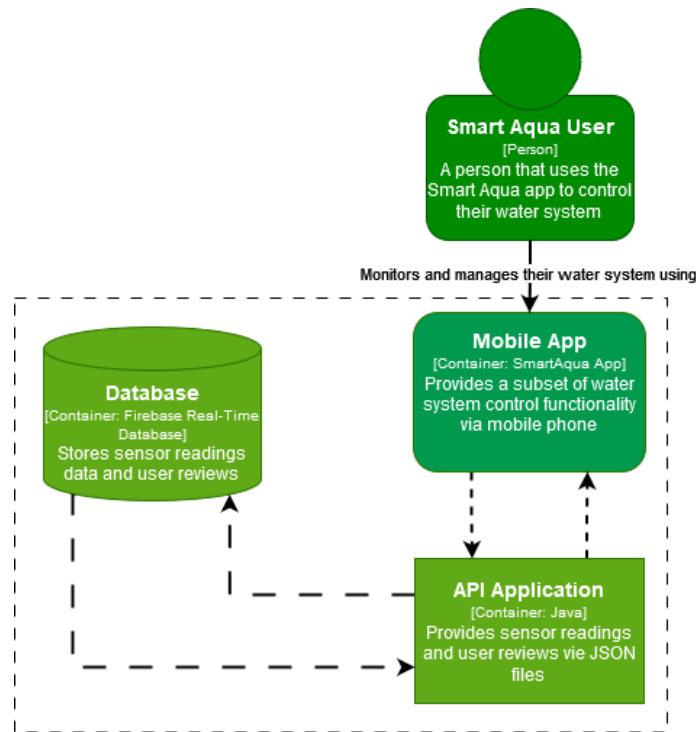
## Sprint Goals

As our team transitions from deliverable 3 to 4 our sprint goals for the current sprint are as follows:

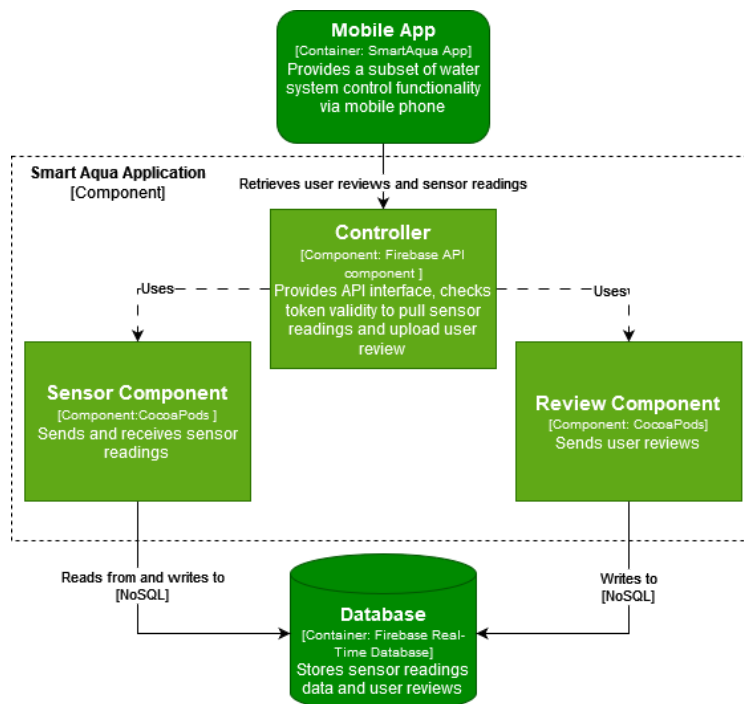
- 1. Enhance User Interaction for Offline and Online Modes:**
  - Explore techniques to improve user engagement and experience in both offline and online modes of the Smart Aqua Project.
- 2. Implement Data Download Feature:**
  - Develop a user-friendly data download functionality that allows users to retrieve relevant information to their phones as needed.
- 3. Study Different App Testing Methodologies:**
  - Research and understand different methodologies used for testing mobile applications to ensure the reliability of our app.
- 4. Refactor Code for Efficiency, Skill Improvement, and Future Maintenance:**
  - Explore code refactoring techniques to optimize the app's performance, enhance coding skills, and facilitate easier maintenance in the future.
- 5. Explore the Implementation of the C4 Model in Software Development:**
  - Gain insights into the principles and practical implementation of the C4 Model in the context of software development projects.
- 6. Collaborate as a Team to Publish the App on the Play Store:**
  - Work collaboratively as a team to successfully submit the Smart Aqua Project app to the Google Play Store.

## C4 Model

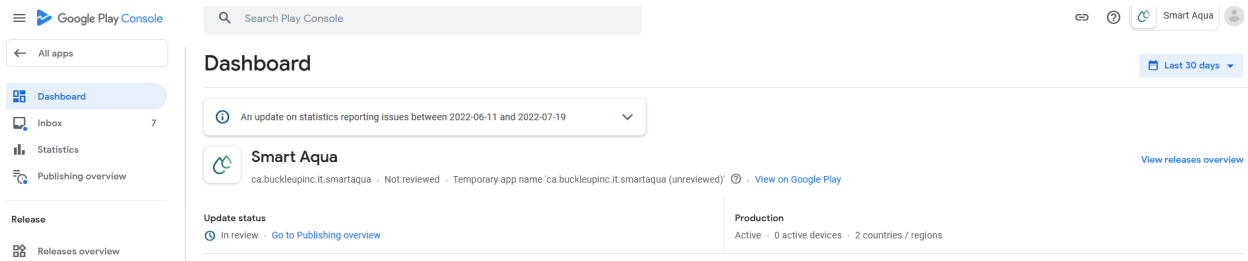
### Container Diagram



### Component Diagram



## Project submitted into Google Play



## Offline Functionality

The functionality that offers an offline mode is the reading feature in SmartAquaQuality. The behavior of the class involves retrieving simulated data from our database, which is then displayed to indicate one of two statuses (Good or Danger). Prior to this deliverable, I had noticed that the readings weren't displaying when the phone lost Wi-Fi connection. As a result, this implementation is a commendable solution to that issue.

When the phone is online, the app actively and randomly selects one reading to display. However, when the app goes offline or enters airplane mode, it stores the readings from the database in shared preferences and presents them to the user. Moreover, this functionality persists even if the app is closed and subsequently reopened in offline mode. This seamless transition between online and offline modes adds significant value to the user experience and demonstrates the sophistication of the Smart Aqua application.

## Runtime Permission Feature

### Accessing location runtime permission

The app's manifest file must have the necessary permission constants, such as 'Manifest.permission.ACCESS\_COARSE\_LOCATION,' in order to make use of the location runtime permission functionality in Android Studio. The basis for permission checks is set up by these constants. For the Smart Aqua application to manage location services and updates properly, initializing both the 'LocationManager' and 'LocationListener' aspects becomes essential.

The 'ActivityCompat.checkSelfPermission()' method is used to check the status of the current authorization. When the permission has already been granted, the application may freely reveal the current location by calling 'LocationManager.requestLocationUpdates()'. If the permission has not yet been granted, however, the approach switches to using the 'ActivityCompat.requestPermissions()' method to request the permission. Additionally,

the 'onRequestPermissionsResult()' function is where the results of these permission requests are managed. It is feasible to create an instance of a permission-handling class like "SmartAquaLocation" and use its "onRequestPermissionsResult()" function within the setting of this method with the appropriate arguments.

## **Validating location runtime permission**

The SmartAquaLocation class is in charge of managing the runtime permission validation for location access on the device. By carefully verifying for permissions and location services, this class controls the process. It makes use of the LocationPermissionTask AsyncTask subclass to asynchronously verify whether the app has the required permission (PERMISSION\_ACCESS\_COARSE\_LOCATION). Whether or not permission is given depends on the outcome of this test. The changed onRequestPermissionsResult() function reads the result based on grantResults once the user responds to the permission request, streamlining further actions. Depending on permission and location service status, the displayCurrentLocation() function takes care of displaying the current location.

The class also uses the isLocationEnabled() function to confirm the availability of network-based or GPS location providers, which is necessary for precise location information. The class uses the locationManager.requestLocationUpdates() method to ask for location updates when permission is given and location services are enabled. The user is requested to enable location services through settings if they are currently disabled. The application supports Android's runtime permission structure by using this planned approach, improving user privacy and security and providing seamless location-based functionality.

## **Breakdown of location runtime logic**

The LocationPermissionTask class extends AsyncTask and serves the purpose of verifying whether the app holds location permission. If the permission is granted, the onPostExecute method gets triggered, leading to the display of the current location. In case the permission is not granted, the app resorts to requesting it via ActivityCompat.requestPermissions().

In onPostExecute(), when permission is confirmed, the method displayCurrentLocation() is invoked. This function establishes a listener for location updates, allowing retrieval of the current location, which is then displayed via a Toast notification.

The displayCurrentLocation() function is responsible for inspecting the availability of location services. If these services are accessible and permission is granted, the app initiates location updates using locationManager.requestLocationUpdates(). If, however,

location services are deactivated, the app prompts the user to enable them by opening the device settings.

The `isLocationEnabled()` method performs a check to ascertain whether either GPS or network location providers are active on the device.

The `requestLocationPermission()` method acts as a starting point for the permission request process within the application.

`OnRequestPermissionsResult()` gets invoked upon the user's response to the permission request. The method verifies if permission was granted and, if affirmative, proceeds with displaying the current location.

## Scrum Dashboard

Task name	Assignee	Due date	Priority
▼ Implement test cases for the Smart Aqua App			
✓ Write JUnit 4 test cases to validate the behaviour and correctness of these components in isolation.	Nicholas Dib...	Yesterday	
✓ Write Robolectric test cases to verify the behavior of your app under different Android platform configurations.	Paolo Quezon	Yesterday	
✓ Write Espresso test cases to simulate user interactions, such as button clicks	DS Denis Shwaloff	Yesterday	
✓ Identify key user interface elements and interactions in your app that need testing.	Nicholas Dib...	Yesterday	
✓ Write valid and invalid test cases to meet requirements given by Deliverable guideline	DS Denis Shwaloff	Yesterday	
Add task...			
▼ Create a floating action button for user interaction			
✓ Create a floating action button in the SmartAquaQuality class	RC Rodrigo Cha...	Aug 4	Medium
✓ Design with proper icon and colour the floating action button	RC Rodrigo Cha...	Aug 4	Medium
✓ Implement functionality to download the readings into the phone storage	RC Rodrigo Cha...	Aug 4	Medium
✓ Download two different files for the functionality (json and txt)	RC Rodrigo Cha...	Aug 4	Medium
✓ Accommodate the floating action button in the fragment	RC Rodrigo Cha...	Aug 4	Medium
✓ Ensure the floating action button downloads both files to phone storage	RC Rodrigo Cha...	Aug 4	Medium
Add task...			

Task name	Assignee	Due date	Priority
▼ Add Offline Feature			
✓ Implement functionality to perceive if the user is online or offline	RC Rodrigo Cha...	Aug 3	Medium
✓ Store the readings coming from the database to Shared Preferences	RC Rodrigo Cha...	Aug 3	Medium
✓ Verify that the data is displayed when switching from offline to online	RC Rodrigo Cha...	Aug 3	Medium
✓ Adapt existing method to retrieve readings from database	RC Rodrigo Cha...	Aug 3	Medium
✓ Implement functionality to display readings from shared preferences	RC Rodrigo Cha...	Aug 3	Medium
Add task...			
▼ Store the prefer settings from the user			
✓ Add the Shared Preferences code for the dark mode feature in the settings screen	RC Rodrigo Cha...	Jul 13	Medium
✓ Add the Shared Preferences code in the main activity for the dark mode feature	RC Rodrigo Cha...	Jul 13	Medium
✓ Verify that shared preferences work well when user exits and opens the app again	RC Rodrigo Cha...	Jul 13	High
✓ Debug to find any issues in case of app crashing (Optional but important)	RC Rodrigo Cha...	Jul 13	High
✓ Add the Shared Preferences code for the mute app feature in the settings screen	RC Rodrigo Cha...	Jul 13	Medium
✓ Add the Shared Preferences code for the lock landscape mode feature in the settings screen	DS Denis Shwaloff	Jul 13	Medium
Add task...			



Task name	Assignee	Due date	Priority
▼ Complete the functionality for the individual settings for Smart Aqua App users			
✔ Implement and enable the dark mode feature to work with the toggle button	RC Rodrigo Cha...	Jul 13	Medium
✔ Implement and enable the mute app feature to work with the toggle button	DS Denis Shwaloff	Jul 13	Medium
✔ Implement Run-time permission for getting approximate location	PA Paolo Quezon	Jul 13	Medium
✔ Display toast when the settings are reset to default	RC Rodrigo Cha...	Jul 13	Low
✔ Implement functionality of reset settings button	RC Rodrigo Cha...	Jul 13	Medium
✔ Display toast when dark mode is on and when the mute app is enabled and disabled	RC Rodrigo Cha...	Jul 13	Low
✔ Implement the use of AsyncTask with location runtime permission	PA Paolo Quezon	Jul 13	Medium
Add task...			
▼ Retrieve reviews from customers and store them in a real-time database			
✔ Update the feedback screen design with the new requirements	RC Rodrigo Cha...	Jul 11	Medium
✔ Implement the real-time database for users to send reviews	DS Denis Shwaloff	Jul 11	High
✔ Create Firebase Database for our customers reviews	DS Denis Shwaloff	Jul 11	High
✔ Add Alert Dialog when the customer wants to send a review	DS Denis Shwaloff	Jul 11	Medium
✔ Display toast when review has been sent	DS Denis Shwaloff	Jul 11	Medium
✔ Display toast if not all the fields in the review are filled	DS Denis Shwaloff	Jul 11	Medium
✔ Verify that the database successfully connects to our app	DS Denis Shwaloff	Jul 11	High
Add task...			

Task name	Assignee	Due date	Priority
▼ Complete functionality for all sensor with simulated data and actions			
✔ Send notification when toggle button in light screen is enabled and disabled	N Nicholas Dib...	Jul 14	Medium
✔ Send notification when toggle button in temperature screen is enabled and disabled	PA Paolo Quezon	Jul 14	Medium
✔ Change image view when toggle button in light screen is enabled and disabled	PA Paolo Quezon	Jul 14	Medium
✔ Display status and snack bars when toggle buttons are enabled and disabled in the switch screen	DS Denis Shwaloff	Jul 14	Medium
✔ Save simulated data from TDS readings into our real-time database	RC Rodrigo Cha...	Jul 14	Medium
✔ Read data from database for TDS simulated readings in the water quality screen	RC Rodrigo Cha...	Jul 14	Medium
✔ Add Shared Preference for the switch status on Switch screen	DS Denis Shwaloff	Jul 14	Medium
✔ Add Shared Preference for the seekbar and toggle button on Temp screen	PA Paolo Quezon	Jul 14	Medium
Add task...			
▼ Update the Layout design for our Smart Aqua App			
✔ Update the Home Screen Frame Layout to Constraint Layout with ScrollView	RC Rodrigo Cha...	Jul 15	Low
✔ Update the Settings Screen Frame Layout to Constraint Layout with ScrollView	RC Rodrigo Cha...	Jul 15	Low
✔ Update the Water Quality Screen Frame Layout to Constraint Layout with ScrollView	RC Rodrigo Cha...	Jul 15	Low
✔ Update the Water Temperature Screen Frame Layout to Constraint Layout with ScrollView	PA Paolo Quezon	Jul 15	Low
✔ Update the Light Control Screen Frame Layout to Constraint Layout with ScrollView	N Nicholas Dib...	Jul 15	Low
✔ Update the Switch Screen Frame Layout to Constraint Layout with ScrollView	DS Denis Shwaloff	Jul 15	Low
✔ Update the Review Screen Frame Layout to Constraint Layout with ScrollView	RC Rodrigo Cha...	Jul 15	Low
✔ Update the Splash Screen Frame Layout to Constraint Layout with ScrollView	RC Rodrigo Cha...	Jul 15	Low

Task name	Assignee	Due date	Priority
▼ Create general QOL features			
✔ Implement OnBackButton User Alert	DS Denis Shwaloff	Jun 9	Medium
✔ Create the SplashScreen on startup	PA Paolo Quezon	Jun 12	Medium
✔ Create location feature with Run-Time user permissions to access user's device general locality	DS Denis Shwaloff	Jun 10	Medium
✔ Implement Contact US feature in the Options Menu for user feedback	DS Denis Shwaloff	Jun 11	Medium
✔ Implement visual design elements in the Options Menu like icons and optional button visibility	RC Rodrigo Cha...	Jun 11	Low
Add task...			
▼ Design pages for each hardware piece			
✔ Design the Temperature Control screen	PA Paolo Quezon	Jun 10	Medium
✔ Design the Light Control screen	N Nicholas Dib...	Jun 10	Medium
✔ Design the Switch Control Screen	DS Denis Shwaloff	Jun 10	Medium
✔ Design the Water Quality Screen	RC Rodrigo Cha...	Jun 10	Medium
✔ Verify that both landscape and portrait layouts are supported	N Nicholas Dib...	Jun 10	High
Add task...			

Task name	Assignee	Due date	Priority
▼ Create Cloud Database			
✔ Add Alert Dialog to the Submit Review button to confirm that user wants to send the review	DS Denis Shwaloff	Jun 11	Low
✔ Create Server to host Cloud Database on Microsoft Azure	DS Denis Shwaloff	Jun 11	Low
✔ Setup MySQL Workbench client to work with the Cloud DB Server	DS Denis Shwaloff	Jun 11	Low
✔ Create new database through MySQL Workbench for later implementation	DS Denis Shwaloff	Jun 11	Low
✔ Add a screenshot of the Database to the project document	DS Denis Shwaloff	Jun 11	Low
Add task...			
▼ Implement functionality for movement between fragments			
✔ Create an Implemented Menu	RC Rodrigo Cha...	Jun 10	Medium
✔ Implement Navigation Drawer to navigate fragments	RC Rodrigo Cha...	Jun 9	High
✔ Add links to Settings, Feedback, and Home fragments in the Options Menu for quick user access	RC Rodrigo Cha...	Jun 10	Medium
✔ Design an Implemented Menu based on requirements	RC Rodrigo Cha...	Jun 10	Medium
✔ Code the functionality for the menu	RC Rodrigo Cha...	Jun 10	High
Add task...			

Task name	Assignee	Due date	Priority
▼ Adapt application for Landscape device layouts			
✔ Design landscape layout for Splash Screen	PA Paolo Quezon	Jun 10	Medium
✔ Design landscape layout for Temperature Fragment	PA Paolo Quezon	Jun 10	Medium
✔ Design landscape layout for Light Fragment	N Nicholas Dib...	Jun 10	Medium
✔ Design landscape layout for Switch Fragment	DS Denis Shwaloff	Jun 10	Medium
✔ Design landscape layout for Home Fragment	RC Rodrigo Cha...	Jun 10	Low
✔ Design landscape layout for Quality Fragment	RC Rodrigo Cha...	Jun 10	Medium
✔ Design landscape layout for Settings Fragment	DS Denis Shwaloff	Jun 10	Medium
Add task...			
▼ Feedback and help for user issues			
✔ Create Feedback fragment	RC Rodrigo Cha...	Jun 10	Low
✔ Design the feedback fragment in portrait and landscape	RC Rodrigo Cha...	Jun 10	Low
✔ Create a menu option to contact a team member	DS Denis Shwaloff	Jun 10	Low
✔ Add the feedback option on both the navigation drawer and implemented menu	RC Rodrigo Cha...	Jun 10	Low
✔ Functionality for the feedback transfer from both menus	RC Rodrigo Cha...	Jun 10	Low
✔ Functionality for the contact us option from the menu	DS Denis Shwaloff	Jun 10	Low
Add task...			

## Post-Mortem - Project Review Meeting

### POST-MORTEM

Project Review Meeting

**1. Performance review of the project. Performance in terms of cost, schedule, and quality.**

Cost of the project was minimal in monetary sense, our team avoided using paid services or any other possible expenses. On the other hand, this project has taken up a significant chunk of every team member's not only allocated but free time as well. Scheduling, was good and there was not a significant lack of organization however, we would often stay after hours, or do work outside of our pre-determined schedule, due to the amount of objectives to be completed in time.

**2. Did the team members involved manage their time wisely? Or everything was done last minute.**

The team has always tried to stay on track and start work on each deliverable as soon as possible, all things considered. At times, with our busy schedules, it was hard to stay on track, however, through hard work and appropriate allocation of team members to their respective tasks, all of the requirements were completed on time.

**3. Were there issues with the quality or compromises along the way?**

There were a few instances, where we would be out of time to completely refine our documentation, although it only happened in extreme cases.

**4. Lessons learned, mistakes, and area of improvements.**

With some the software app requirements, we were met with great challenges that would take a considerable amount of time and effort to complete. Only after the completion, sometimes, we would find the proper guides or information that would have greatly helped in the development process.

Another aspect of it was that in some instances the tasks were not assigned equally (capability wise) and we would run into problems with time constraints and re-writes to account for that. Also, on top of that, at times, we would face similar situations with the documentation work.

**5. Who attended and who missed the meeting**

Every team member has attended the meeting.

## How did you address technical debt in your project

By deliberately identifying important application-related areas that needed attention, our team addressed technical debt in our project. The possible effects on the functionality, user experience, and development efficiency of the app were taken into consideration as we prioritized these areas. Setting priorities for high-impact issues before moving on to lower-priority ones was the primary aspect of our strategy. During our daily standup meetings, when we carefully laid out plans to deal with each item of technical debt, we often reviewed and improved this process. We distributed tasks, deadlines, and responsibilities to each team member by dividing larger work into smaller, more manageable chunks. In order to show this process, we used our Scrum dashboard.

We started reworking our application's code to improve its general quality, readability, and maintainability after assigning ourselves tasks. Our SmartAquaSettings class changed as an example of this effort. We identified the technical debt in its poor code structure, which was formerly filled with several methods. To combat this, we added several classes, each of which is responsible for containing a certain method or functionality of our settings screen. The readability of the codebase was significantly improved through this

method of organization. Our Java classes underwent similar reformatting, with closely removed unnecessary and redundant code segments.

To avoid disagreements and keep the team working in sync, communication was essential. We created a routine of notifying each other before making changes to the application, particularly if several team members were working on related classes. Our code was kept updated and well-organized by routinely downloading updates. We carefully checked to make sure that all dependencies and libraries were continuously updated in order to reduce any vulnerabilities and compatibility problems.

During our daily standups, we implemented collaborative techniques like pair programming and code reviews to improve the standard of our codebase. These processes not only made it easier to spot errors but also made sure that the changes made to reduce technical debt were smoothly incorporated into the codebase overall.

## Refactoring

Upon analyzing our code in the last deliverable, we noticed that the settings fragment was handling multiple tasks simultaneously. Therefore, we made the decision to separate the functionality of each setting into distinct classes. This will help improve the organization and maintainability of our codebase.

### First Case:

#### Dark Mode Section: Before refactoring

##### SmartAquaSettings > OnCreateView

```
//=====DARK MODE=====
ToggleButton darkTB = view.findViewById(R.id.SmartAquaDarkModeToggleBtn);
ToggleButton darkTB = view.findViewById(R.id.SmartAquaDarkModeToggleBtn);
darkTB.setOnCheckedChangeListener(null); // Remove previous listener temporarily
SmartAquaDarkMode darkModeHandler = new SmartAquaDarkMode(darkTB, settingsPreferences);

darkTB.setOnCheckedChangeListener(darkModeHandler);
// Get the initial state from SharedPreferences
final boolean[] darkModeCheckState = {settingsPreferences.getBoolean("DarkModeToggleState", false)};
darkTB.setChecked(darkModeCheckState[0]);

darkTB.setOnCheckedChangeListener((buttonView, isChecked) -> {
    // Check if the state has actually changed
    if (isChecked != darkModeCheckState[0]) {
        if (isChecked) {
            AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES);
            Toast.makeText(getActivity(), R.string.darkModeON, Toast.LENGTH_SHORT).show();
        } else {
            AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
        }
    }
}
```

```

        settingsPreferences.edit().putBoolean("DarkModeToggleState", isChecked).apply();
        darkModeCheckState[0] = isChecked; // Update the initial state
    }
});

```

The snippet of code above is in charge of the dark mode feature. It was in the `onCreateView` function, along with the rest of the code. This made our code not efficient enough to make specific changes if we wanted to implement them in different cases. Also the amount of tasks were too condensed in just one function. We decided to create a separate class

## Dark Mode Section: After refactoring

### **SmartAquaSettings > onCreateView**

```

//=====DARK MODE=====
ToggleButton darkTB = view.findViewById(R.id.SmartAquaDarkModeToggleBtn);
SmartAquaDarkMode darkModeHandler = new SmartAquaDarkMode(darkTB, settingsPreferences);
darkTB.setOnCheckedChangeListener(darkModeHandler);

```

### **SmartAquaDarkMode**

```

/* CENG-322-0NA: Group 6
   Denis Shwaloff - N01422583
   Alvaro Rodrigo Chavez Moya - N01455107
   Paolo Adrian Quezon - N01424883
   Nicholas Dibiase - N01367109 */

package ca.buckleupinc.it.smartaqua;

import android.content.SharedPreferences;
import android.widget.CompoundButton;
import android.widget.Toast;
import android.widget.ToggleButton;

import androidx.appcompat.app.AppCompatActivity;

public class SmartAquaDarkMode implements CompoundButton.OnCheckedChangeListener {

    private final ToggleButton darkTB;
    private final SharedPreferences settingsPreferences;
    private static final String DARK_MODE_TOGGLE_STATE_KEY = "DarkModeToggleState";

    public SmartAquaDarkMode(ToggleButton darkTB, SharedPreferences settingsPreferences) {
        this.darkTB = darkTB;
        this.settingsPreferences = settingsPreferences;
    }

    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        // Check if the state has actually changed
        if (isChecked != settingsPreferences.getBoolean(DARK_MODE_TOGGLE_STATE_KEY, false)) {
            if (isChecked) {

```

```

        AppCompatActivity.setDefaultNightMode(AppCompatActivity.MODE_NIGHT_YES);
        Toast.makeText(buttonView.getContext(), R.string.darkModeON, Toast.LENGTH_SHORT).show();
    } else {
        AppCompatActivity.setDefaultNightMode(AppCompatActivity.MODE_NIGHT_NO);
    }
    settingsPreferences.edit().putBoolean(DARK_MODE_TOGGLE_STATE_KEY, isChecked).apply();
}
}
}
}

```

As we created the new SmartAquaDarkMode class, this made simpler to call the method in the SmartAquaSetting, but not only for simplicity but also to give maintenance in case of improvement in the future.

## Second Case:

### DownloadManagerSection: Before refactoring

#### SmartAquaQuality > OnCreateView

```

FloatingActionButton quality_fab = view.findViewById(R.id.SmartAquaFAB);
quality_fab.setOnClickListener(viewFAB -> {
    String reading_TDS_str = readings_TDS.getText().toString();
    String status_TDS_str = status_TDS.getText().toString();

    // Save the data to a text file
    saveDataToFile(reading_TDS_str, status_TDS_str);

    // Show a toast message to indicate the download
    Toast.makeText(getActivity(), R.string.wq_download_message, Toast.LENGTH_SHORT).show();
});

```

#### SmartAquaQuality - Methods:

```

private void saveDataToFile(String reading, String status) {
    String folderName = "SmartAquaQualityReadings";
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HH:mm:ss", Locale.getDefault()).format(new Date());

    // Save JSON data
    String jsonFileName = "SmartAquaReading_" + timeStamp + ".json";
    File folder = new File(getActivity().getExternalFilesDir(null), folderName);
    if (!folder.exists()) {
        folder.mkdirs();
    }

    File jsonFile = new File(folder, jsonFileName);

    try {
        FileOutputStream jsonFileOutputStream = new FileOutputStream(jsonFile);
        OutputStreamWriter jsonOutputStreamWriter = new OutputStreamWriter(jsonFileOutputStream);

        JSONObject dataObject = new JSONObject();
        dataObject.put("Reading", reading);
        dataObject.put("Status", status);
    }
}

```

```

        dataObject.put("DateTime", timeStamp); // Include date and time in JSON

        jsonOutputStreamWriter.write(dataObject.toString());

        jsonOutputStreamWriter.close();
        jsonFileOutputStream.close();
    } catch (IOException | JSONException e) {
        e.printStackTrace();
        Toast.makeText(getActivity(), R.string.wq_json_error, Toast.LENGTH_SHORT).show();
    }

    // Save text data
    String textFileName = "SmartAquaReading_" + timeStamp + ".txt";
    File textFile = new File(folder, textFileName);

    try {
        FileOutputStream textFileOutputStream = new FileOutputStream(textFile);
        OutputStreamWriter textOutputStreamWriter = new OutputStreamWriter(textFileOutputStream);

        textOutputStreamWriter.write("Reading: " + reading + "\n");
        textOutputStreamWriter.write("Status: " + status + "\n");
        textOutputStreamWriter.write("DateTime: " + timeStamp + "\n"); // Include date and time in text

        textOutputStreamWriter.close();
        textFileOutputStream.close();
    } catch (IOException e) {
        e.printStackTrace();
        Toast.makeText(getActivity(), R.string.wq_txt_error, Toast.LENGTH_SHORT).show();
    }
}

```

This is the previous code before separating the download functionality to its own class. We decided to do this to adapt download features in the future for our different screens such as light, temperature and switch.

## DownloadManagerSection: **After refactoring**

### SmartAquaQuality > OnCreateView

```

FloatingActionButton quality_fab = view.findViewById(R.id.SmartAquaFAB);
quality_fab.setOnClickListener(viewFAB -> {
    SmartAquaDownloaderManager fileDownloader = new
    SmartAquaDownloaderManager(requireContext());
    fileDownloader.saveWQDataToFile(readings_TDS.getText().toString(),
    status_TDS.getText().toString());
    Toast.makeText(getActivity(), R.string.wq_download_message, Toast.LENGTH_SHORT).show();
});

```

### SmartAquaDownloadManager

```

/* CENG-322-0NA: Group 6
   Denis Shwaloff - N01422583
   Alvaro Rodrigo Chavez Moya - N01455107
   Paolo Adrian Quezon - N01424883
   Nicholas Dibiase - N01367109 */

```

```

package ca.buckleupinc.it.smartaqua;

import android.content.Context;
import android.widget.Toast;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

public class SmartAquaDownloaderManager {
    private final Context context;
    private static final String FOLDER_NAME = "SmartAquaQualityReadings";
    private static final String DATE_FORMAT = "yyyyMMdd_HH:mm:ss";
    private static final String FILE_NAME = "SmartAquaReading_";
    private static final String JSON_EXT = ".json";
    private static final String TXT_EXT = ".txt";
    private static final String READING = "Reading";
    private static final String STATUS = "Status";
    private static final String DATE_TIME = "Date_Time";
    private static final String SPACE = "\n";
    private static final String COLON = ":";

    public SmartAquaDownloaderManager(Context context) {
        this.context = context;
    }

    public void saveWQDataToFile(String reading, String status) {
        String folderName = FOLDER_NAME;
        String timeStamp = new SimpleDateFormat(DATE_FORMAT,
        Locale.getDefault()).format(new Date());

        // Save JSON data
        String jsonFileName = FILE_NAME + timeStamp + JSON_EXT;
        File folder = new File(context.getExternalFilesDir(null), folderName);
        if (!folder.exists()) {
            folder.mkdirs();
        }

        File jsonFile = new File(folder, jsonFileName);

        try {
            FileOutputStream jsonFileOutputStream = new FileOutputStream(jsonFile);
            OutputStreamWriter jsonOutputStreamWriter = new
        OutputStreamWriter(jsonFileOutputStream);

            JSONObject dataObject = new JSONObject();
            dataObject.put(READING, reading);
            dataObject.put(STATUS, status);
            dataObject.put(DATE_TIME, timeStamp);

```



```

        jsonOutputStreamWriter.write(dataObject.toString());

        jsonOutputStreamWriter.close();
        jsonFileOutputStream.close();
    } catch (IOException | JSONException e) {
        e.printStackTrace();
        Toast.makeText(context, R.string.wq_json_error, Toast.LENGTH_SHORT).show();
    }

    // Save text data
    String textFileName = FILE_NAME + timeStamp + TXT_EXT;
    File textFile = new File(folder, textFileName);

    try {
        FileOutputStream textFileOutputStream = new FileOutputStream(textFile);
        OutputStreamWriter textOutputStreamWriter = new
OutputStreamWriter(textFileOutputStream);

        textOutputStreamWriter.write(READING + COLON + reading + SPACE);
        textOutputStreamWriter.write(STATUS + COLON + status + SPACE);
        textOutputStreamWriter.write(DATE_TIME + COLON + timeStamp + SPACE);

        textOutputStreamWriter.close();
        textFileOutputStream.close();
    } catch (IOException e) {
        e.printStackTrace();
        Toast.makeText(context, R.string.wq_txt_error, Toast.LENGTH_SHORT).show();
    }
}
}

```

If we want to add more download methods for different screens, they can be just added here since each screen has different parameters to download data.

## Suggestions for future projects

- The entire team found it to be a very exciting process to build an app from the bottom up and gradually add new features over the course of the project.
- Being a part of a group significantly enhanced each group member's ability to form a team while encouraging a stronger understanding of teamwork and the importance of working together.
- It could have been a good idea to plan a weekly time where other teams could rate our app for each deliverable we completed on a scale of 1 to 5. Through feedback from our classmates and instructors, this could allow us to identify our strengths and potential areas for improvement to check if all the required features and functionality were met. The adjustments we've made since the last deliverable might be highlighted in a quick presentation that lasts 5 to 10 minutes, which can potentially be weekly/bi-weekly depending on the availability of each team.