CENG-322 Deliverable 4

Team #1 - **Algaerithms Inc.**

Project name - Phytoplankton-based air purifiers

Name	Student ID	Github ID	Signature	Effort
Julian Imperial	N01638310	JulianImperial8310	J.I	100%
Dhairya Pal	N01576099	DhairyaPal6099	D.P	100%
Sanskriti Mansotra	N01523183	SanskritiMansotra3183	S.M	100%
Dharmik Shah	N01581796	DharmikShah1796	D.S	100%

Brief description of the project - A smart and eco-friendly air purification system that uses phytoplankton to naturally convert CO_2 into oxygen. The system connects with an Android app for real-time monitoring, automated alerts, and sustainability-focused feedback.

What issue your product will solve -
Our system addresses poor indoor air quality caused by high CO_2 levels and low oxygen circulation. Unlike passive air purifiers, it uses live algae to naturally photosynthesize and purify the air while providing a clean energy cycle and an interactive dashboard for education and engagement.

Compare your application with at least two apps in the market. Provide links and description of the two apps you selected.

1. LIQUID3

LIQUID3 is a large outdoor urban solution that combines an algae tank with street furniture. It serves as an air purifier, public seating, USB charger, night light, and traffic barrier. Designed primarily for city spaces, it brings greenery into urban environments and is backed by UNDP Serbia and other government agencies.

2. Carbelim

An indoor modular “living wall” using photobioreactor panels; captures CO_2 (up to 330 kg/year) and emits oxygen (~1.5 million L/year).Removes >90% of $PM_{2.5}$, PM_{10}, VOCs, NO_x, microbes, and neutralizes pathogens with integrated UV-C sterilization. IoT-enabled real-time monitoring dashboard; scalable indoor installation from compact T-1 units to large panels

Highlight the differences between your app and these two Why you believe your app is better than these two apps. Create a table pros and cons of the three different apps.

Feature	Phytoplankton Air System	LIQUID3	Carbelim AirForest™
Deployment	Indoor (home, small office)	Outdoor urban spaces	Indoor walls, lobbies, clinics
Portability	Compact, modular	Large, fixed bench-style	Modular living-wall panels
Sensors & Feedback	CO_2, light, humidity, algae health + app alerts	Manual maintenance, no sensors	IoT dashboard (CO_2, AQI, PM, etc.)
Pollutants Targeted	$CO_2 \rightarrow O_2$ photosynthesis	CO_2, CO, PM, heavy metals	CO_2, PM, VOCs, microbes, NO_x + UV-C sterilization
User Interaction	Active via Android app, real-time controls	Passive, public usage	Passive display; facility-managed
Maintenance	Regular algae care, refill cycle	Monthly biomass removal, refill	Low-maintenance, UV-C, IoT diagnostics
Scale	Single-unit indoor	Fixed public installation	Scalable from small to large indoor environments
Primary Purpose	Personal air quality, education	Public greenery, multi-utility bench	Commercial air purification, ESG compliance

13. Why you believe your app is better than these two apps?

1. **Designed for individuals**—compact, affordable, and smart-enabled for home and office use.

2. **Interactive control**—users receive real-time feedback and can adjust settings via an Android interface.

3. **Optimal indoor performance**—unlike LIQUID3's outdoor focus or AirForest's installation requirements.

4. **Educational appeal**—provides insight into air quality, algae health, and eco-behavior—a gamified experience missing in large-scale systems.

5. **Cost-effective**—lower installation and maintenance costs compared to infrastructure-heavy alternatives.

14. Create a table pros and cons of the three different apps.

Feature	**Phytoplankton Air system**	**LIQUID3**	**Carbelim AirForest™**
Pros	• Smart sensor suite• Real-time app interaction• Compact and modular• Educational and engaging	• Urban-friendly design• Processes pollutants & heavy metals• Multi-functional (bench, charger, light)	• Broad pollutant removal (CO_2, PM, VOCs, microbes)• Modular scaling• IoT monitoring + UV-C sterilization
Cons	• Requires maintenance and algae updates• Needs app setup	• Outdoor only• Requires city permits and power• Large footprint	• High cost and infrastructure needs• Geared for commercial, not domestic
Best For	Individual users, small workplaces	City planners, public urban spots	Corporations, hospitals, public buildings

GitHub Repo link - https://github.com/Algaerithms-Inc/PhytoplanktonAirSystems

Login functionality, I will use the following credentials to test your app:

Email: aaa@bbb.com Password: Admin101!

Describe in detail, the work that has been completed by each team member in this sprint only.

1) Dhairya -
   1.1) Contact Support - Improved the UI of the contact support screen in the support action bar in accordance with the rest of the app. Made both the contact buttons (phone number dialing, and emailing functional.
   1.2) Lifetime CO2 logic - All the managers (achievements, notifications, leaderboard, etc.) would either get their carbon dioxide data from the realtime database or wouldn't be implemented at all yet. Now the logic to update 'Lifetime carbon dioxide' from the realtime database is in place and functional so all the handlers working with lifetime carbon dioxide can start fetching that data from the firestore directly.
   1.3) Achievements - Made it so it fetches the lifetime carbon dioxide data from the firestore instead of the real time database.
   1.4) Share Dashboard - A custom view of the dashboard is shared when clicked on the Share button in the support action bar. This view contains the app watermark, the highest achievement the currently logged in user has received, and the lifetime amount of carbon dioxide converted.
   1.5) Leaderboard - Made this screen completely functional.
   1.6) Feedback screen - Once the user submits the form successfully, gray out the submit button, and display a timer showing how many hours and minutes remaining when the user can submit another feedback.
   1.7) Authentication class - Moved all the authentication logic from LoginActivity to Authentication class.

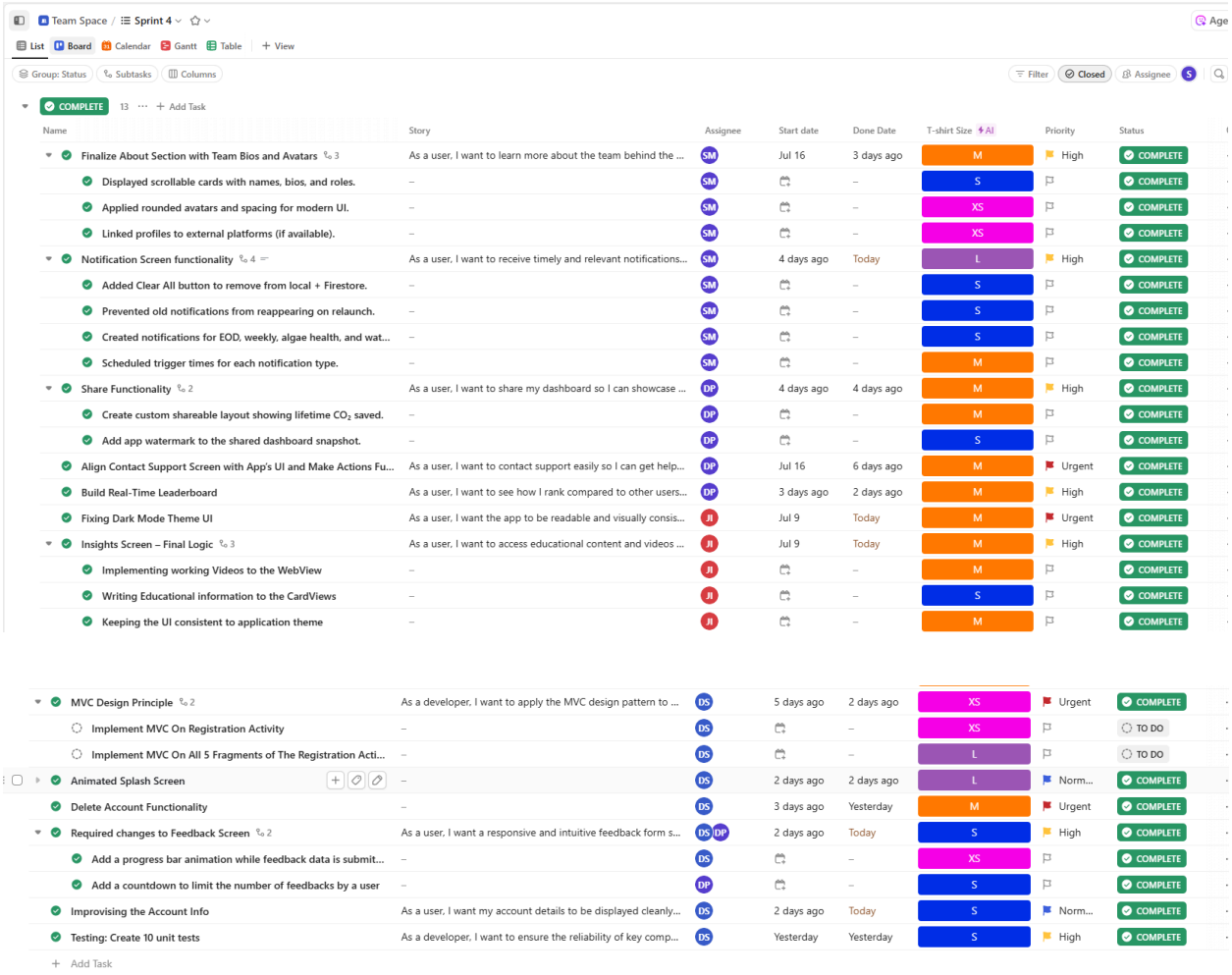
2) Dharmik -
   2.1) Password - Enforced the requirement of the password being minimum 6 characters.
   2.2) Splash Screen - Created an animation instead of a still image as the splash screen.
   2.3) Test cases - Wrote ten test cases for AccountInfo class.
   2.4) Settings - Delete Account implemented. Name, email should be fetched from the database of the current user. Also in that cardview, remove the angle bracket on the right so people don't think it's clickable.
   2.5) Deliverable - Took screenshots showing the team discussion, topics discussed, team members. Provide screenshots for three different dates, i.e. WhatsApp, Discord, …etc.

2.6) Feedback screen - Progress bar is shown while submitting the feedback form. Implemented a delay for 5 seconds. Once received a confirmation from the DB, display an AlertDialog with OK confirming the form has been submitted.
2.7) MVC - Implemented MVC for RegistrationActivity and registration fragments. Also sorted all the files in the project in different folders based on this design pattern.

3) Julian -
   3.1) C4 diagrams - Created both level diagrams as per the requirement.
   3.2) Dashboard - Removed the GraphView option as it is not aligned with the rest of our app.
   3.3) Dark theme switch - Changed the colors in themes so all the labels and texts are readable in both light and dark mode.
   3.4) Gantt Chart - Updated.
   3.5) Insights screen - Finished implementation of this screen.
   3.6) Deliverable - Described the main functionality added in the deliverable.
   3.7) Strings hardcoding review - Reviewed and moved any hardcoded strings to strings.xml

4) Sanskriti -
   4.1) Sprint tasks and goals - Created and organized Sprint 4 tasks and user stories in ClickUp, ensuring the team had clear goals, priorities, and ownership.
   4.2) Sprint retrospective - Created entirely after taking notes during the retrospective meeting.Built using LucidChart.
   4.3) About Screen - Finished UI and implementation of this screen.Sanskriti implemented the About screen layout with scrollable cards for each team member.
   Each card displays the member's avatar, name, role, and short bio.
   The layout was styled to be simple, clean, and consistent with the rest of the app.
   4.4) Notifications Screen - Sanskriti implemented logic on the Notification screen to show one-time notifications on app launch and scheduled notifications (e.g., Weekly Stats) using WorkManager.She added a "Clear All" button that removes notifications from both the UI and Firebase. Once cleared, notifications do not repopulate on relaunch, ensuring only future scheduled alerts appear.This ensures a clean, non-repetitive notification flow, combining real-time data with controlled scheduling.
   4.5)Built custom icon you see in the splash screen animation.
   4.6) Deliverable - Explained what issue our product will solve. Highlighted the differences between our app and two other apps and why we believe our app is better than these two apps. Created a table pros and cons of the three apps.

Scrum Dashboard:



Team Space / Sprint 4

List | Board | Calendar | Gantt | Table | + View

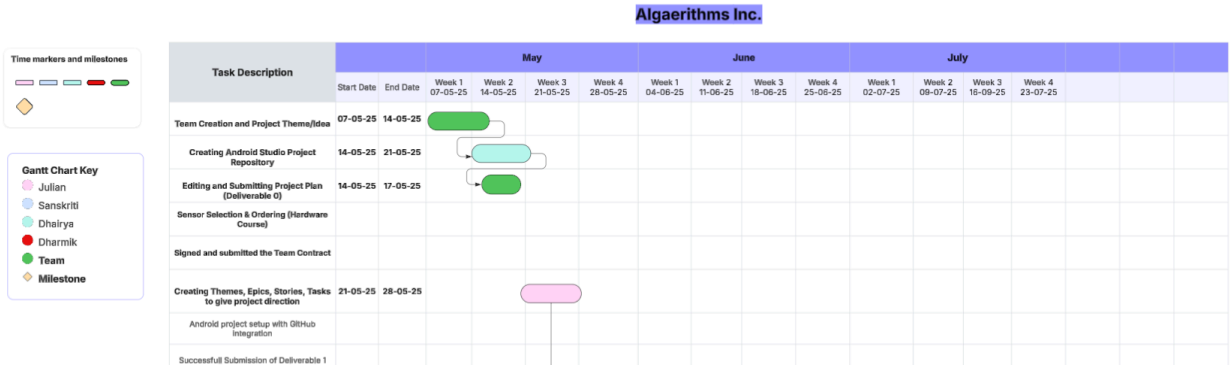
Group: Status | Subtasks | Columns | Filter | Closed | Assignee

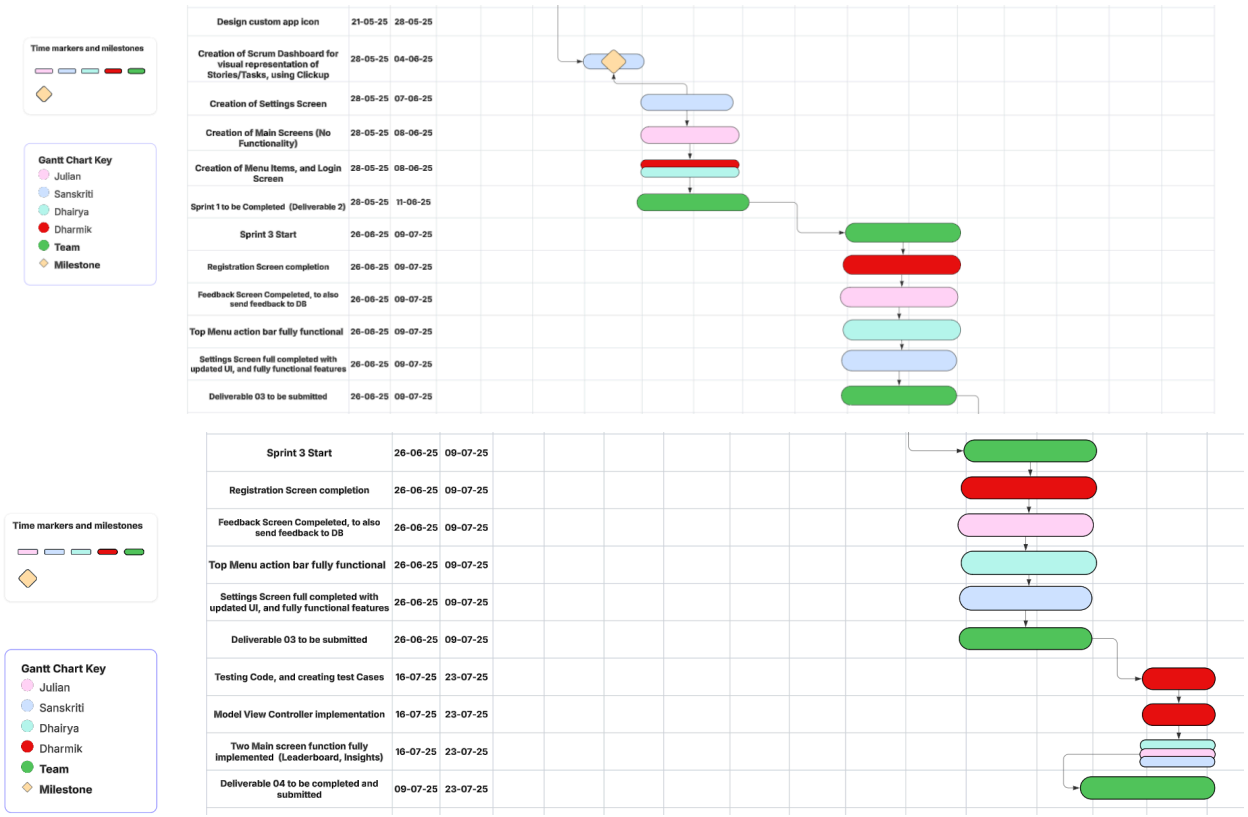
COMPLETE 13 + Add Task

Name	Story	Assignee	Start date	Done Date	T-shirt Size	Priority	Status
Finalize About Section with Team Bios and Avatars 3	As a user, I want to learn more about the team behind the ...	SM	Jul 16	3 days ago	M	High	COMPLETE
Displayed scrollable cards with names, bios, and roles.	–	SM	–	–	S		COMPLETE
Applied rounded avatars and spacing for modern UI.	–	SM	–	–	XS		COMPLETE
Linked profiles to external platforms (if available).	–	SM	–	–	XS		COMPLETE
Notification Screen functionality 4	As a user, I want to receive timely and relevant notifications...	SM	4 days ago	Today	L	High	COMPLETE
Added Clear All button to remove from local + Firestore.	–	SM	–	–	S		COMPLETE
Prevented old notifications from reappearing on relaunch.	–	SM	–	–	S		COMPLETE
Created notifications for EOD, weekly, algae health, and wat...	–	SM	–	–	S		COMPLETE
Scheduled trigger times for each notification type.	–	SM	–	–	M		COMPLETE
Share Functionality 2	As a user, I want to share my dashboard so I can showcase ...	DP	4 days ago	4 days ago	M	High	COMPLETE
Create custom shareable layout showing lifetime CO_2 saved.	–	DP	–	–	M		COMPLETE
Add app watermark to the shared dashboard snapshot.	–	DP	–	–	S		COMPLETE
Align Contact Support Screen with App's UI and Make Actions Fu...	As a user, I want to contact support easily so I can get help...	DP	Jul 16	6 days ago	M	Urgent	COMPLETE
Build Real-Time Leaderboard	As a user, I want to see how I rank compared to other users...	DP	3 days ago	2 days ago	M	High	COMPLETE
Fixing Dark Mode Theme UI	As a user, I want the app to be readable and visually consis...	JI	Jul 9	Today	M	Urgent	COMPLETE
Insights Screen – Final Logic 3	As a user, I want to access educational content and videos ...	JI	Jul 9	Today	M	High	COMPLETE
Implementing working Videos to the WebView	–	JI	–	–	M		COMPLETE
Writing Educational information to the CardViews	–	JI	–	–	S		COMPLETE
Keeping the UI consistent to application theme	–	JI	–	–	M		COMPLETE
MVC Design Principle 2	As a developer, I want to apply the MVC design pattern to ...	DS	5 days ago	2 days ago	XS	Urgent	COMPLETE
Implement MVC On Registration Activity	–	DS	–	–	XS		TO DO
Implement MVC On All 5 Fragments of The Registration Acti...	–	DS	–	–	L		TO DO
Animated Splash Screen	–	DS	2 days ago	2 days ago	L	Norm...	COMPLETE
Delete Account Functionality	–	DS	3 days ago	Yesterday	M	Urgent	COMPLETE
Required changes to Feedback Screen 2	As a user, I want a responsive and intuitive feedback form s...	DS DP	2 days ago	Today	S	High	COMPLETE
Add a progress bar animation while feedback data is submit...	–	DS	–	–	XS		COMPLETE
Add a countdown to limit the number of feedbacks by a user	–	DP	–	–	S		COMPLETE
Improvising the Account Info	As a user, I want my account details to be displayed cleanly...	DS	2 days ago	Today	S	Norm...	COMPLETE
Testing: Create 10 unit tests	As a developer, I want to ensure the reliability of key comp...	DS	Yesterday	Yesterday	S	High	COMPLETE

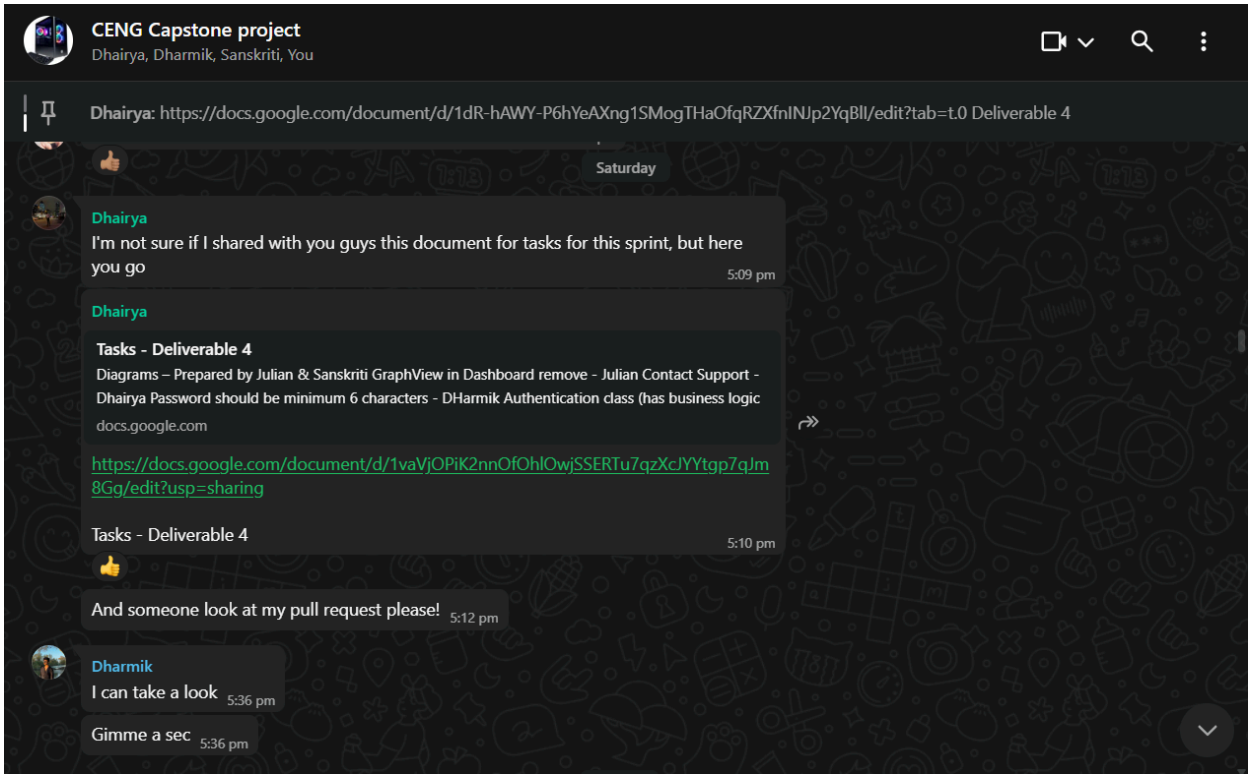
+ Add Task

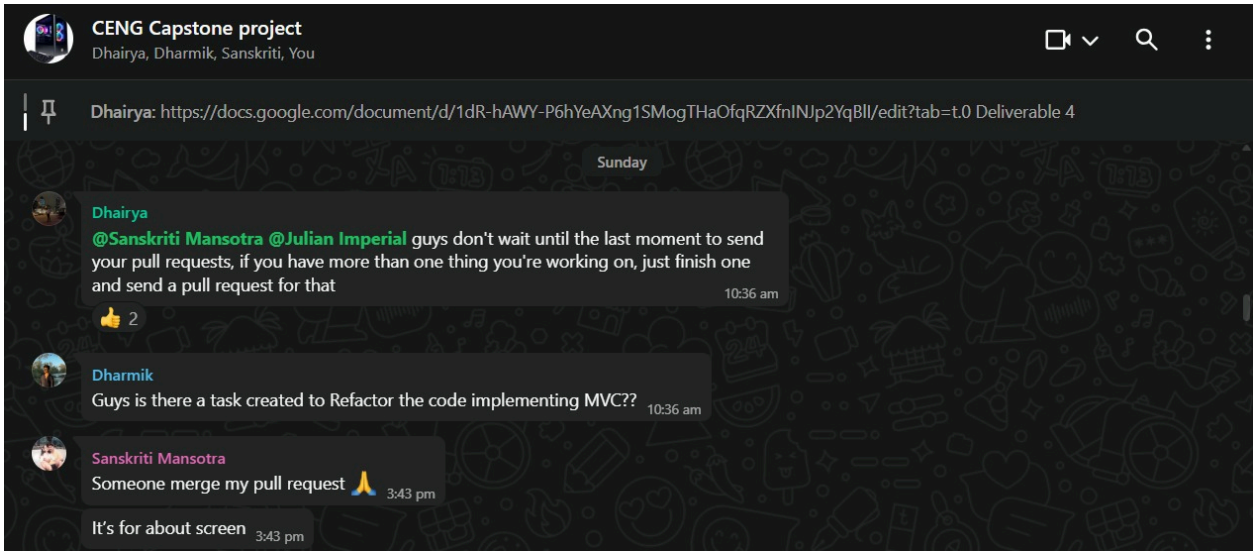
Gantt chart:

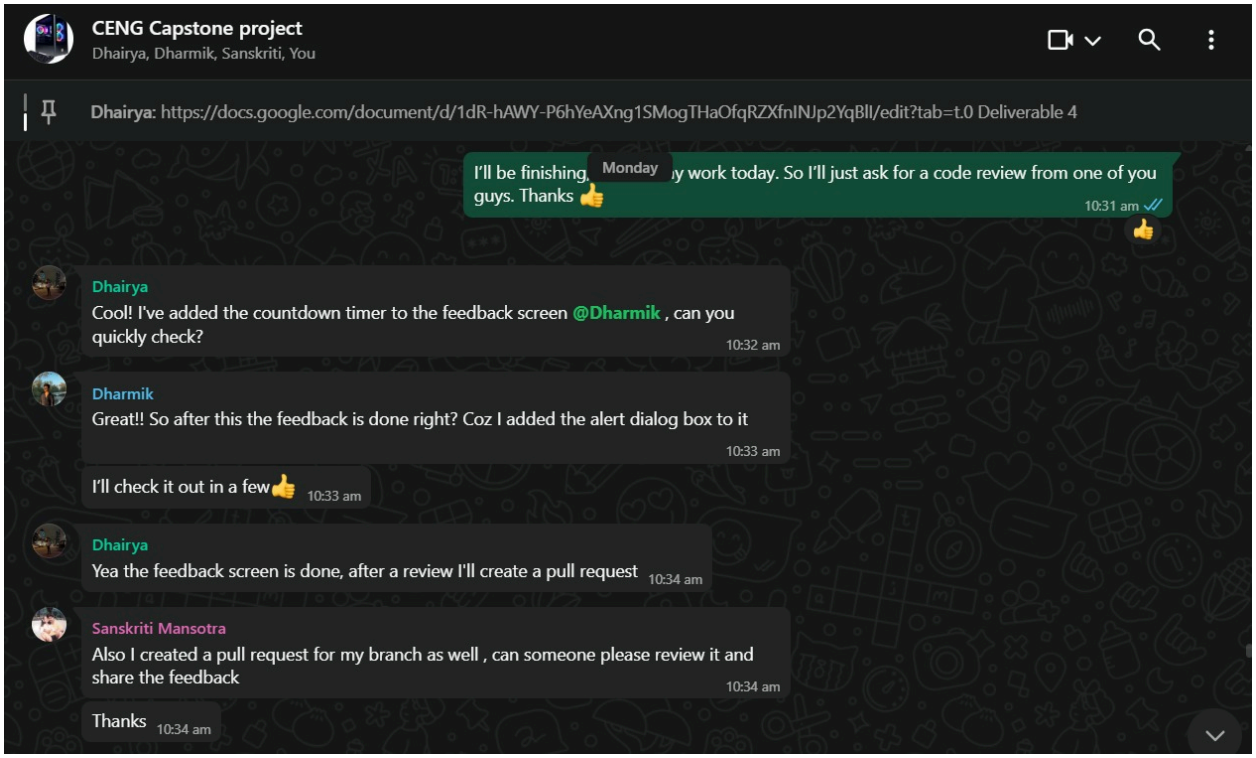




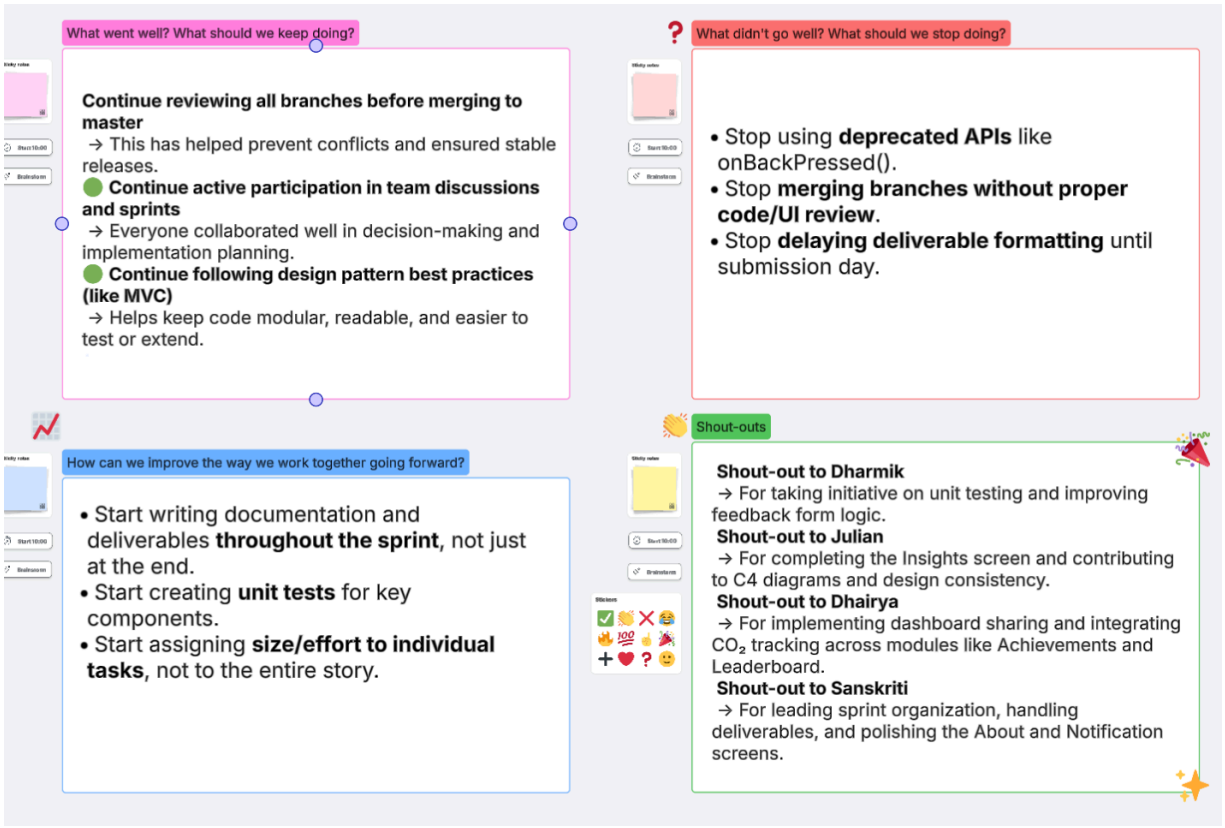
Take screenshots showing the team discussion, topics discussed, team members. Provide screenshots for three different dates, i.e. WhatsApp, Discord, …etc.


CENG Capstone project
Dhairya, Dharmik, Sanskriti, You
Dhairya: https://docs.google.com/document/d/1dR-hAWY-P6hYeAXng1SMogTHaOfqRZXfnINJp2YqBll/edit?tab=t.0 Deliverable 4
Saturday
Dhairya
I'm not sure if I shared with you guys this document for tasks for this sprint, but here you go
5:09 pm
Dhairya
Tasks - Deliverable 4
Diagrams – Prepared by Julian & Sanskriti GraphView in Dashboard remove - Julian Contact Support - Dhairya Password should be minimum 6 characters - DHarmik Authentication class (has business logic
docs.google.com
https://docs.google.com/document/d/1vaVjOPiK2nnOfOhlOwjSSERTu7qzXcJYYtgp7qJm8Gg/edit?usp=sharing
Tasks - Deliverable 4
5:10 pm
And someone look at my pull request please! 5:12 pm
Dharmik
I can take a look 5:36 pm
Gimme a sec 5:36 pm


CENG Capstone project
Dhairya, Dharmik, Sanskriti, You
Dhairya: https://docs.google.com/document/d/1dR-hAWY-P6hYeAXng1SMogTHaOfqRZXfnINJp2YqBll/edit?tab=t.0 Deliverable 4
Sunday
Dhairya
@Sanskriti Mansotra @Julian Imperial guys don't wait until the last moment to send your pull requests, if you have more than one thing you're working on, just finish one and send a pull request for that
10:36 am
2
Dharmik
Guys is there a task created to Refactor the code implementing MVC??
10:36 am
Sanskriti Mansotra
Someone merge my pull request
3:43 pm
It's for about screen 3:43 pm



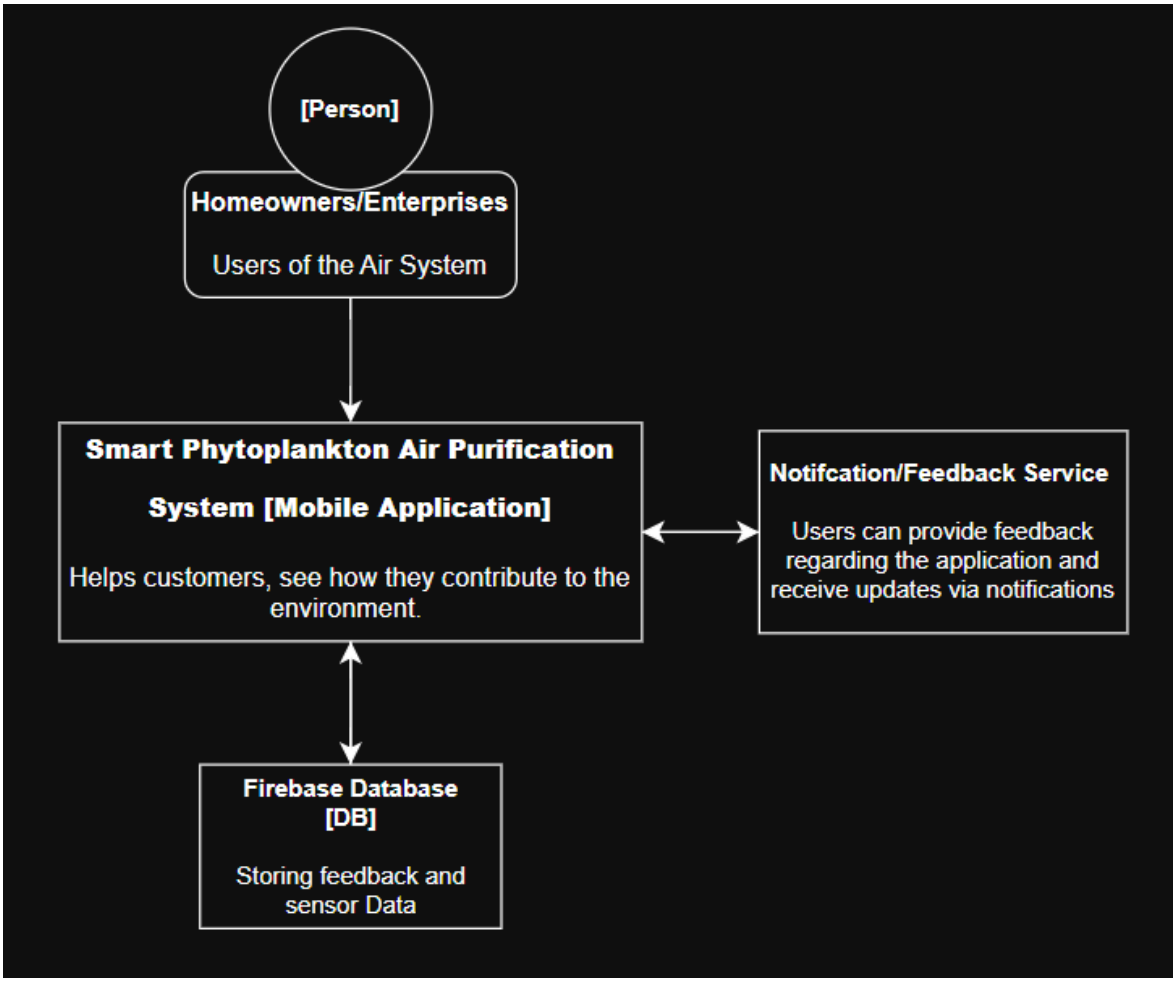
31. Use a tool to record your Sprint Retrospective (i.e. https://lucidspark.com/landing/create/online-sticky-notes https://miro.com/ 1. Who missed the meeting, marks will be deducted for missing the retro. 1. Answer the questions below, a minimum of 3 for each of the following: 1. Start doing. 2. Stop doing. 3. Continue doing. 2. Take a screenshot. 3. Must use online tool for the Retrospective.



We used Lucidspark for Sprint retrospective

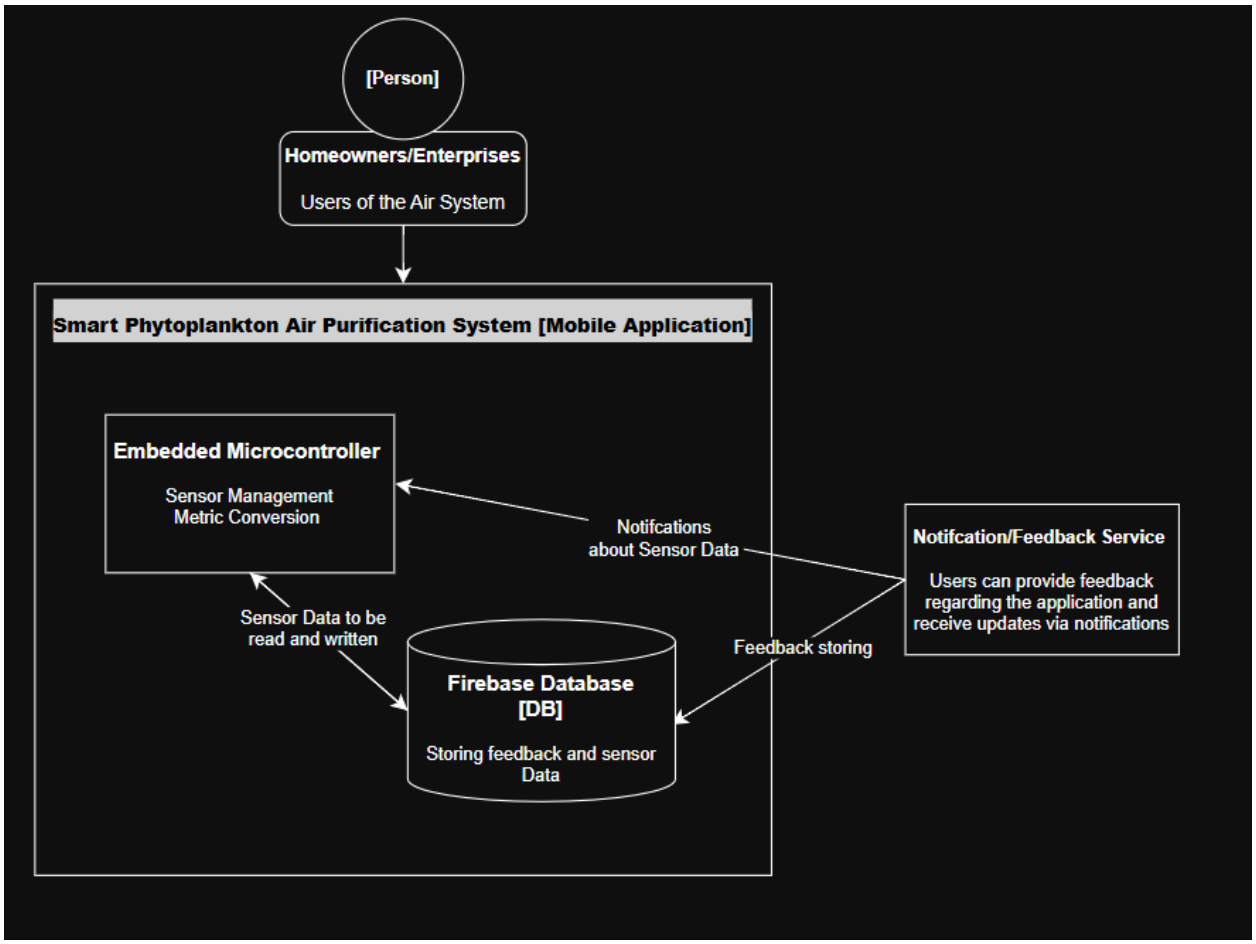
Using C4 Model, show “System Context Diagram”. Use a tool to draw your diagram, hand drawing will not be accepted, i.e. you can use https://app.diagrams.net/

System Context Diagram



Using C4 Model, show “Container Diagram”. Use a tool to draw your diagram, hand drawing will not be accepted, i.e. you can use https://app.diagrams.net/

Container Diagram



Two different design patterns used in our code:

1) **Creational pattern - Builder**

```
//Co2 sync from RTDB to Firestore periodic work request
PeriodicWorkRequest co2SyncRequest = new
PeriodicWorkRequest.Builder(CO2Updater.class, 1, TimeUnit.HOURS).build();
WorkManager.getInstance(this).enqueueUniquePeriodicWork("CO2SyncWork",
ExistingPeriodicWorkPolicy.KEEP, co2SyncRequest);
```

We're adding our carbon dioxide data from the realtime database to a field in the firestore every hour. This field in the firestore contains the user's lifetime carbon dioxide amount converted. To achieve this, we use the PeriodicWorkRequest class and use its Builder method. This way we only have to specify the values that we want to set for the PeriodicWorkRequest instance created and the rest are set by default.

2) **Singleton**

```
public class SensorDataManager {
   private final DatabaseReference deviceRef;
   private final MutableLiveData<SensorData> sensorLiveData = new
MutableLiveData<>();
   private static SensorDataManager instance;
```

```
    private SensorDataManager() {
        deviceRef = FirebaseDatabase.getInstance().getReference("device_001");
        deviceRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                SensorData data = snapshot.getValue(SensorData.class);
                sensorLiveData.setValue(data);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                //Failed to fetch sensor data
                //If this happens often generate a notification
            }
        });
    }

    public static SensorDataManager getInstance() {
        if (instance == null) {
            instance = new SensorDataManager();
        }
        return instance;
    }

    public LiveData<SensorData> getSensorLiveData() {
        return sensorLiveData;
    }

    public void getSensorLatestData(SensorDataCallback callback) {
        deviceRef.get().addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                DataSnapshot snapshot = task.getResult();
                SensorData data = snapshot.getValue(SensorData.class);
                callback.onDataFetched(data);
            } else {

callback.onError(DatabaseError.fromException(task.getException()));
            }
        });
    }

    public interface SensorDataCallback {
        void onDataFetched(SensorData data);
        void onError(DatabaseError error);
    }
}
```

The SensorDataManger class in our app interacts with the realtime database and updates fields of another class called SensorData. SensorData is basically a model class that is supposed to contain all the real time values of the sensors fetched from the realtime database. SensorDataManager makes this value fetching happen and since there should only be the need of one manager class fetching the realtime data from the realtime database to set values in a class, we have made SensorDataManager singleton. If there were more instances allowed then the values in SensorData would start changing unpredictably.

What additional features/functionality added since deliverable 3.

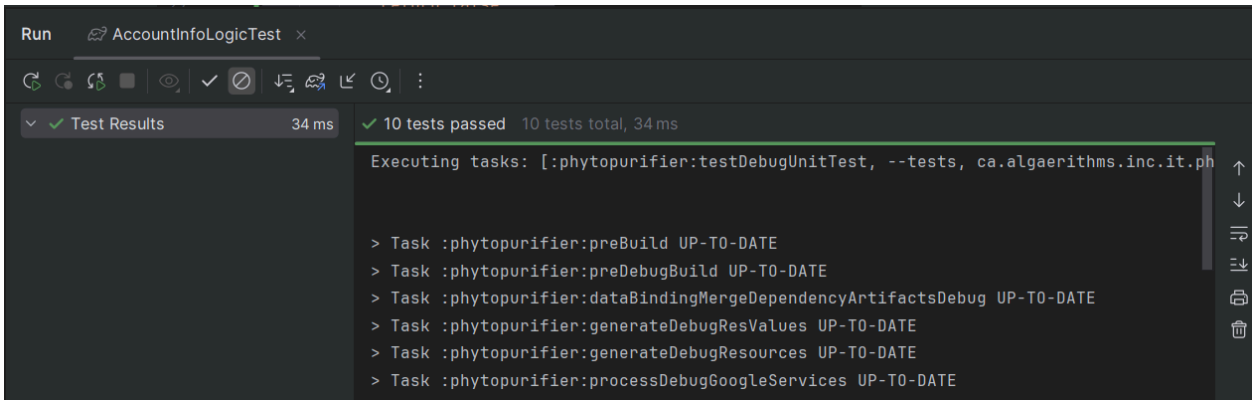
1) Sharing dashboard - The share feature in the support action bar used to take a screenshot of the app screen at the moment it was tapped and it would prepare it as an image to share. Now it creates a custom view to share which includes the amount of carbon dioxide converted by the product of the user signed in, and the title of their highest achievement, along with the watermark of our app. And this custom view is opened in the Android share tray.
2) Splash Screen - Updated to show an animation instead of a still image.
3) Leaderboard - This screen is now completely functional. It retrieves CO2 converted data from the firestore of all the users and shows the top ten users with the most amount of CO2 converted lifetime.
4) Insights - This screen is now completely functional as well. It includes information about phytoplanktons and how they play an important part in our lives. This screen is meant for environment and sustainability enthusiasts to gain more information about the core ingredient of our product - phytoplanktons.
5) Dark mode visibility of text - We added the option of dark mode in the last sprint but this time we made sure that all the text and labels change colors in both light and dark mode so everything is readable.

Coding work progress since the last deliverable -

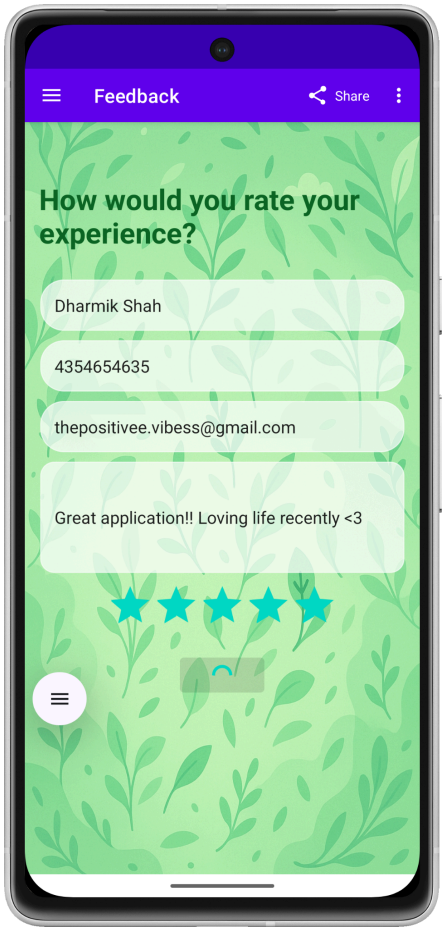
In addition to the coding required for adding/updating the above tasks, we also worked on the following things:

1) Achievements - They were earlier being fetched from the realtime database. Now they are being fetched from firestore.
2) RTDB to Firestore data transfer - Every hour, the carbon dioxide value from the realtime database is now being added to the “Lifetime CO2 converted” field in Firestore.
3) Contact Support screen - This screen now is UI-wise and implementation-wise in accordance with the rest of our app. There is also a GIF added on this screen.
4) Notifications - More kinds of notifications have been created and they are being sent at different points of time, for example - a critical alert if algae health drops significantly is sent immediately, while weekly/daily stats of carbon dioxide converted to oxygen is being sent weekly/daily respectively.

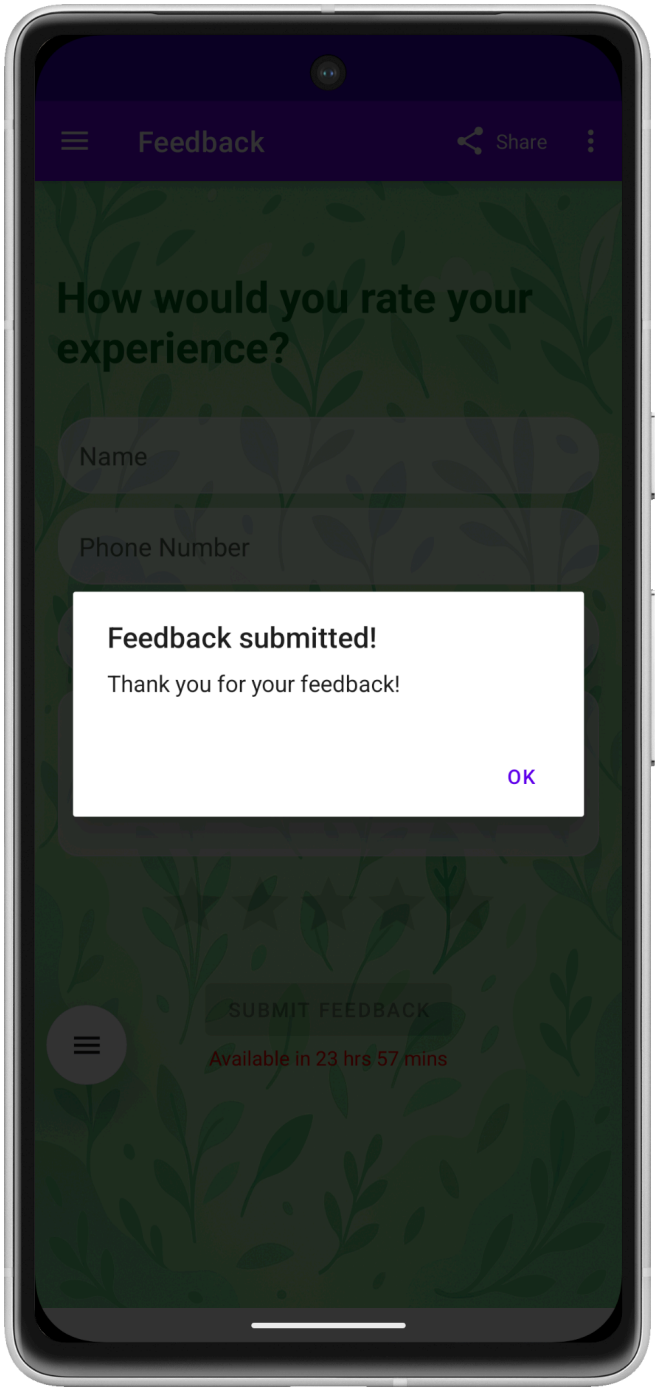
Write unit test cases. Select one of your Java classes and write a minimum of 10 test cases. 38. Demonstrate the use of assertEqual, assertTrue, assertFalse, and asssertNotNull in your testing. 39 All test cases must pass. Take screenshot showing all you test cases are passing.



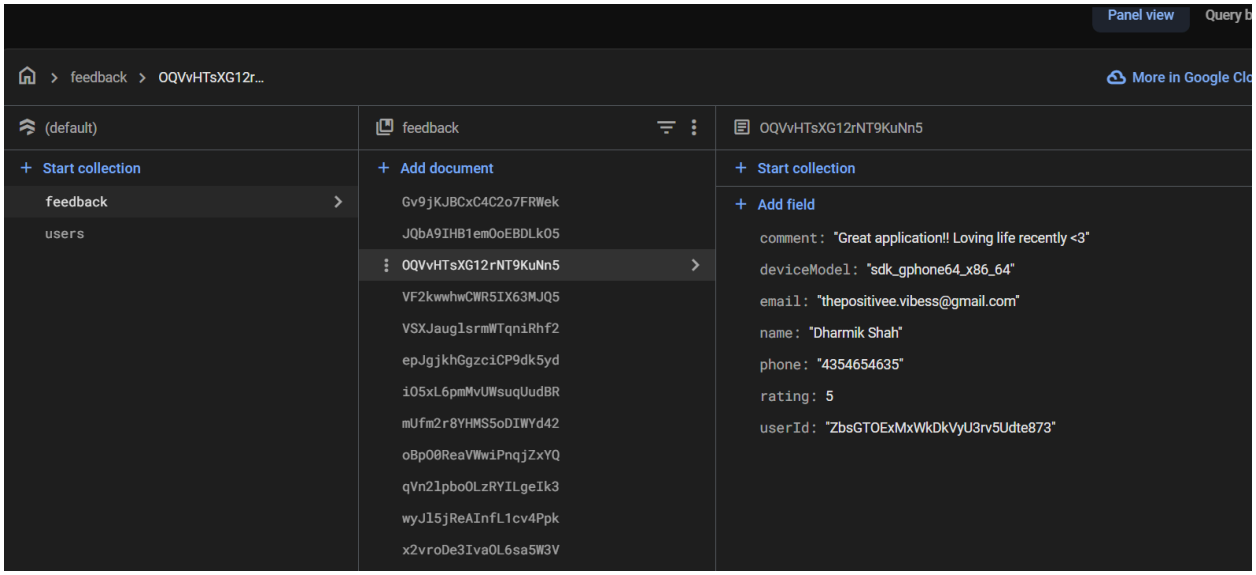
Display progress bar while the info is getting submitted, implement some delay for 5 seconds. 44. Once you receive a confirmation from the DB, display an AlertDialog with OK confirming the form has been submitted. Add into pdf file screenshot showing the progress bar while the form is getting submitted.



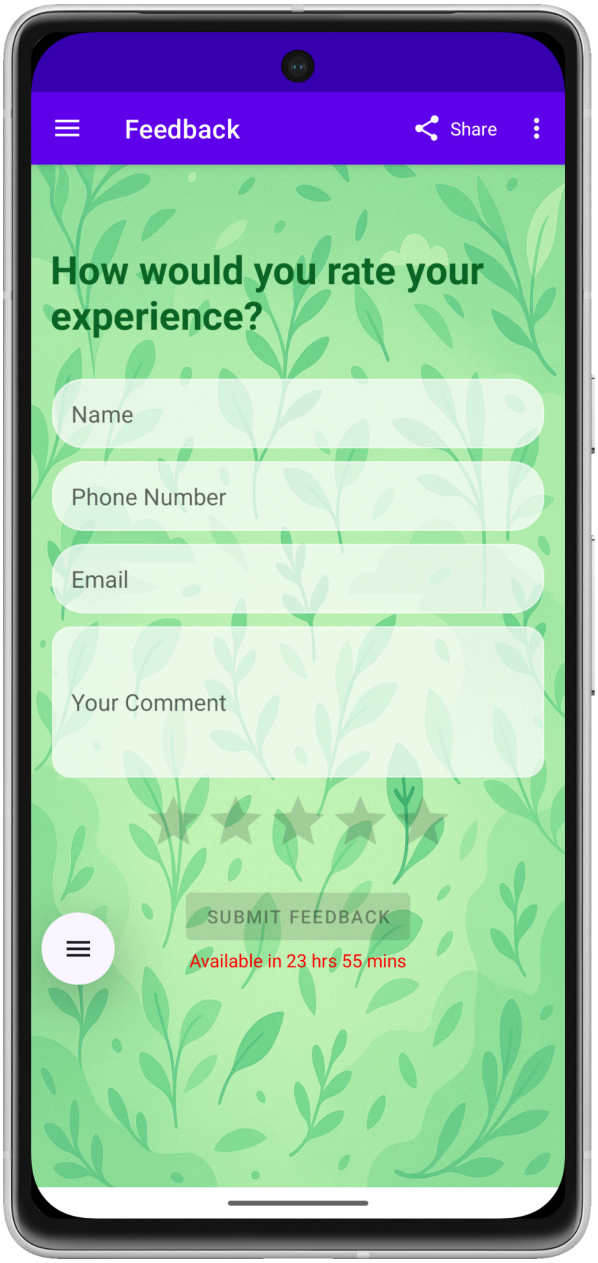
47. Add into pdf file screenshot showing the AlertDialog once the form is submitted successfully.



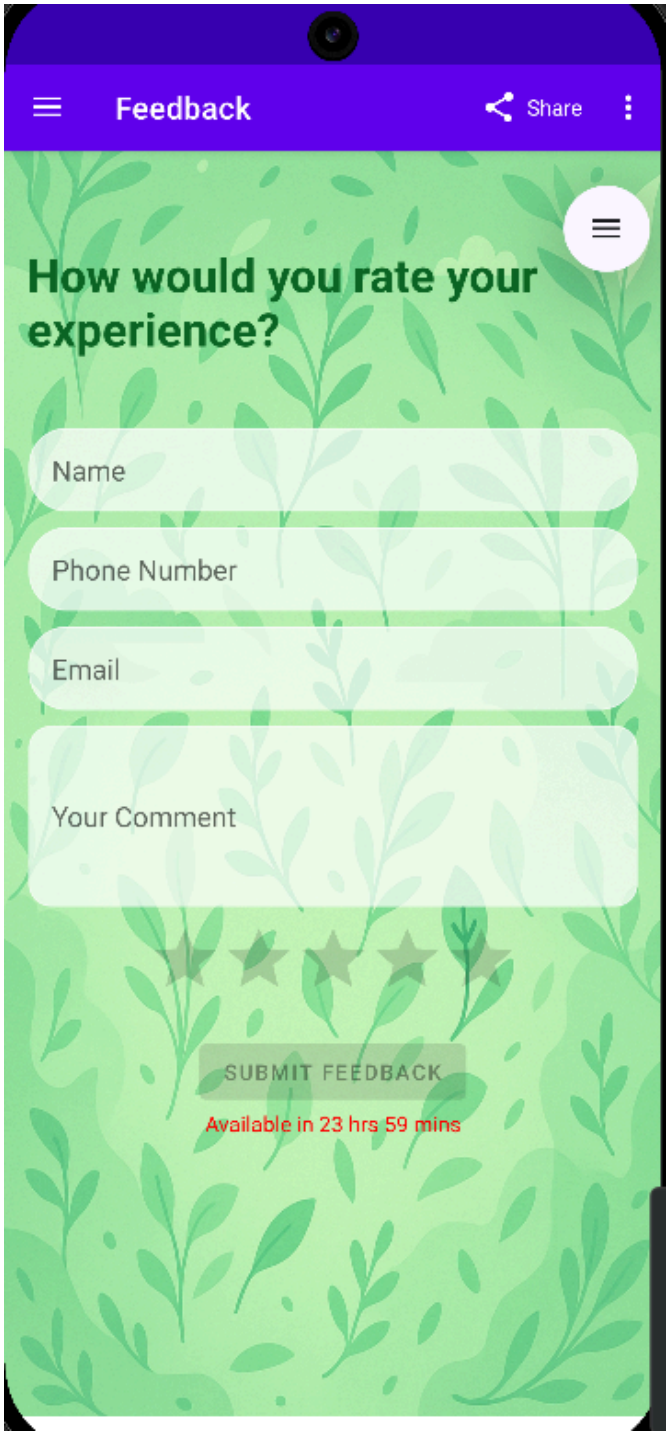
48. Add into pdf file screenshot showing the data stored into the DB. Must have at least 3 different entries.



49. Clear the user input once the form is submitted. Restrict user submission to once per 24 hrs.



50. Once the user submits the form successfully, gray out the submit button, and display a timer showing how many hours and minutes remaining when the user can submit another feedback. Describe in the pdf file on how you satisfied this requirement.



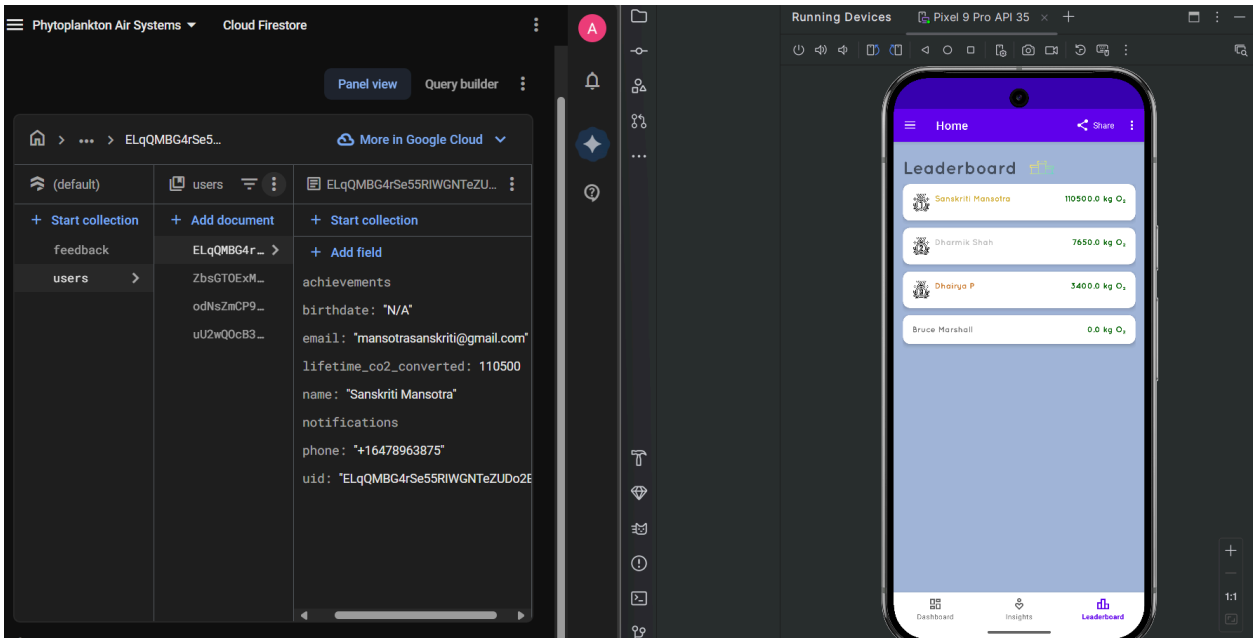
When the user submits the form and the alert dialog box is closed by the user, we set the submit button to grayed out using its property '.setEnabled(false)'. When this happens we store the current system time in a variable and create an if condition to check if the current system time - the saved time = 0. If it is greater than zero, we create an instance of the abstract class

‘CountDownTimer’ with the countdown interval time 1 minute (that means the method onTick will run every minute). Within the onTick method, we get the hours and the minutes left until 24hours are over and show them on a TextView right beneath the button in red color. This textview updates every minute (since the onTick method runs every minute). After the timer runs out, the onFinish method runs within it. We re-enable the submit button and clear the countdown text. This value is stored in SharedPreferences so it makes sure that the time is also counted when the app is not actively running.

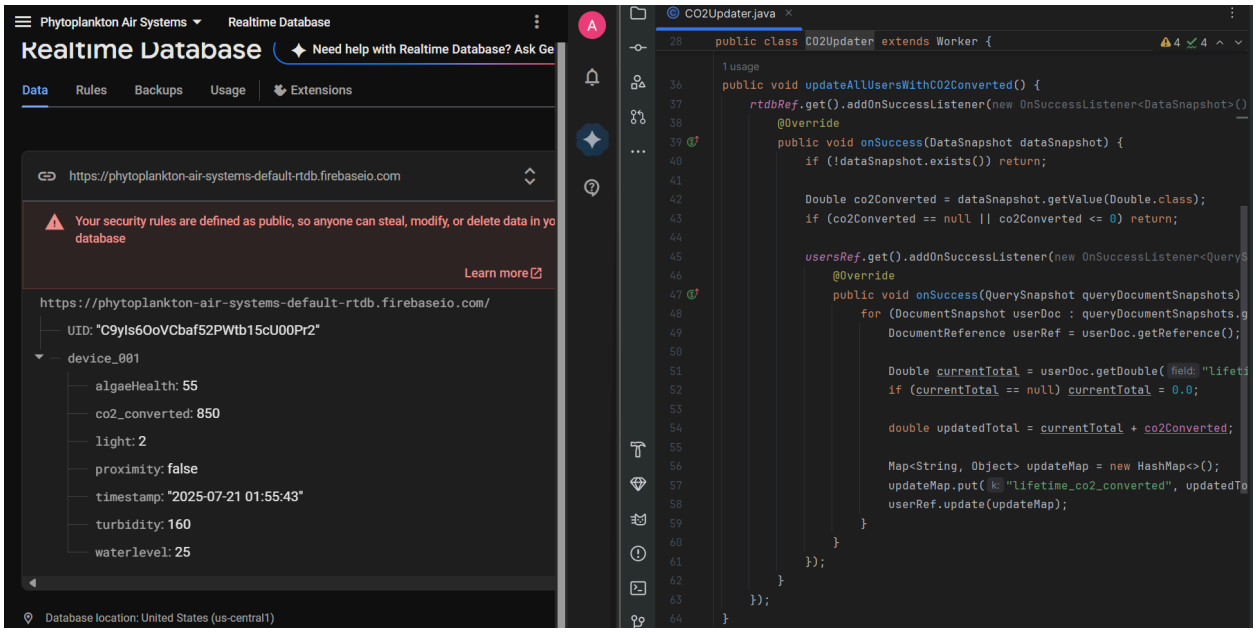
Describe what main functionality added.

So, some of the main functionalities added to our Application is that, the Leaderboard and Insights screen is to be fully functional, by that I mean the Leaderboard now fetches sample sensor data from the database and shows that. The Insights screen has a fully functional webview, and educational cardviews for users that want to learn more about our system and its certain aspects. On the side, we just improved the quality of life to our application alongside other necessary requirements.

Take a screenshot showing all sensor data fetched and updated from the DB.



In this screenshot, we can see a variable that holds the cumulative of Sanskriti’s CO2 converted amount, and that is fetched from the database onto the app as shown in the leaderboard screen - Sanskriti is at the top having converted 11050.0 kg of CO2 to O2. And since the conversion ratio of 1 molecule of CO2 to 1 molecule of O2 is 1:1, that’s why it could be rewritten as 11050 kg of CO2 converted or 11050 kg of O2 produced.



This class ‘CO2Updater’ has a function that runs every hour and adds the value of co2_converted from the real time database to the lifetime_co2_converted variable in firestore. The field ‘co2_converted’ in the real time database is supposed to be updated by Raspberry Pi using the sensor data, and this function ‘updateAllUsersWithCO2Converted’ in ‘CO2Updater’ class updates firestore from real time database every hour.

Registration Screen Explained:

We are using **5 fragments** within the Registration Activity, each handling a different step of the registration process:

1. **RegistrationEmailFragment**
    - This is the first step where users enter their email address.
    - A **verification email** is sent via Firebase Authentication.
    - Only verified emails are allowed to proceed — **no dummy or unverified emails** can log in or continue.
2. **RegistrationNameFragment**
    - Users are asked to enter their **first and last name**.
    - The input is validated to ensure proper name formatting.
3. **RegistrationPhoneFragment**
    - Users select their **country** (e.g., Canada, USA), and the phone number is validated based on that country's code (e.g., 437 for Canada).
    - This ensures the phone number format is appropriate and real.
4. **RegistrationBirthdateFragment**
    - Users choose their **birthdate** using a date picker.
    - This step is **optional** and can be skipped.
    - If provided, the birthdate is used to send **personalized notifications**, offers, or **birthday discounts**.
5. **RegistrationPasswordFragment**
    - Users set their password, which must be at least **6 characters**, and include **an uppercase letter, a lowercase letter, and a number**.
    - This fragment also **collects all user inputs** from the previous fragments (email, name, phone, birthdate) and:

- **Creates the user in Firebase Authentication**,
- **Saves the complete user profile** to Firestore or your database **in one final step**.