CENG-322 Deliverable 1

Team #1 - **Algaerithms Inc.**

Project name - Phytoplankton-based air purifiers

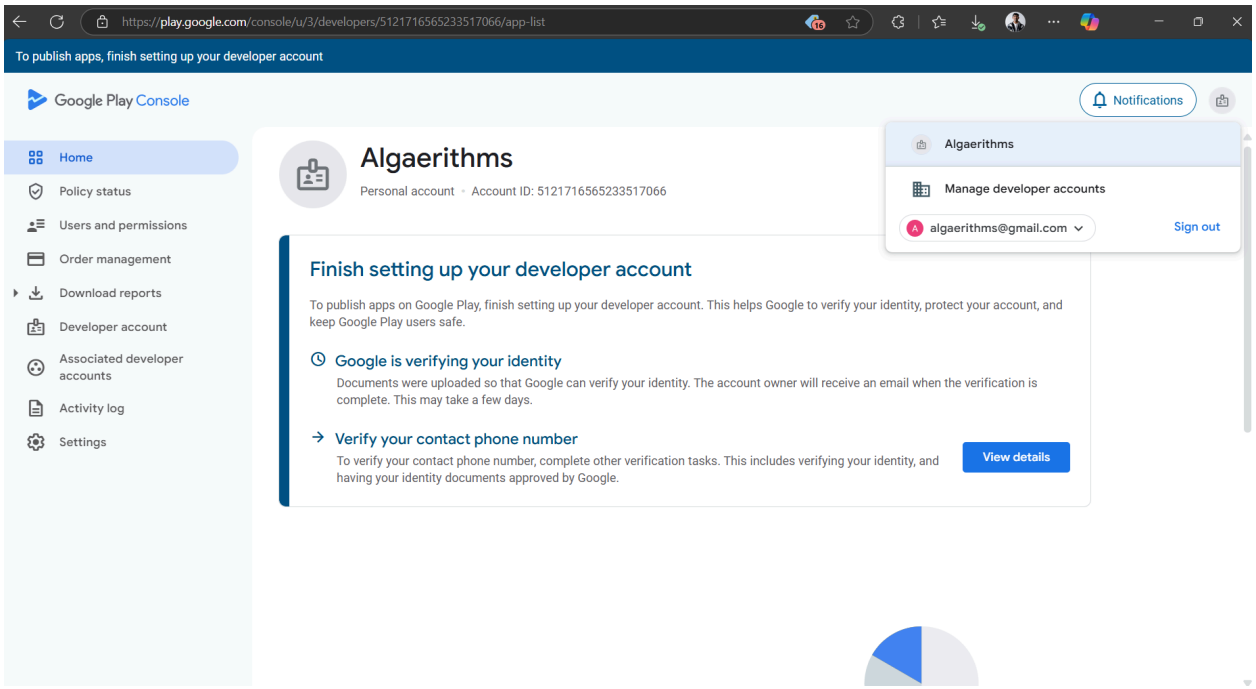
Student names and IDs -

- Dhairya Pal N01576099
- Dharmik Shah N01581796
- Julian Aldrich Imperial N01638310
- Sanskriti Mansotra N01523183

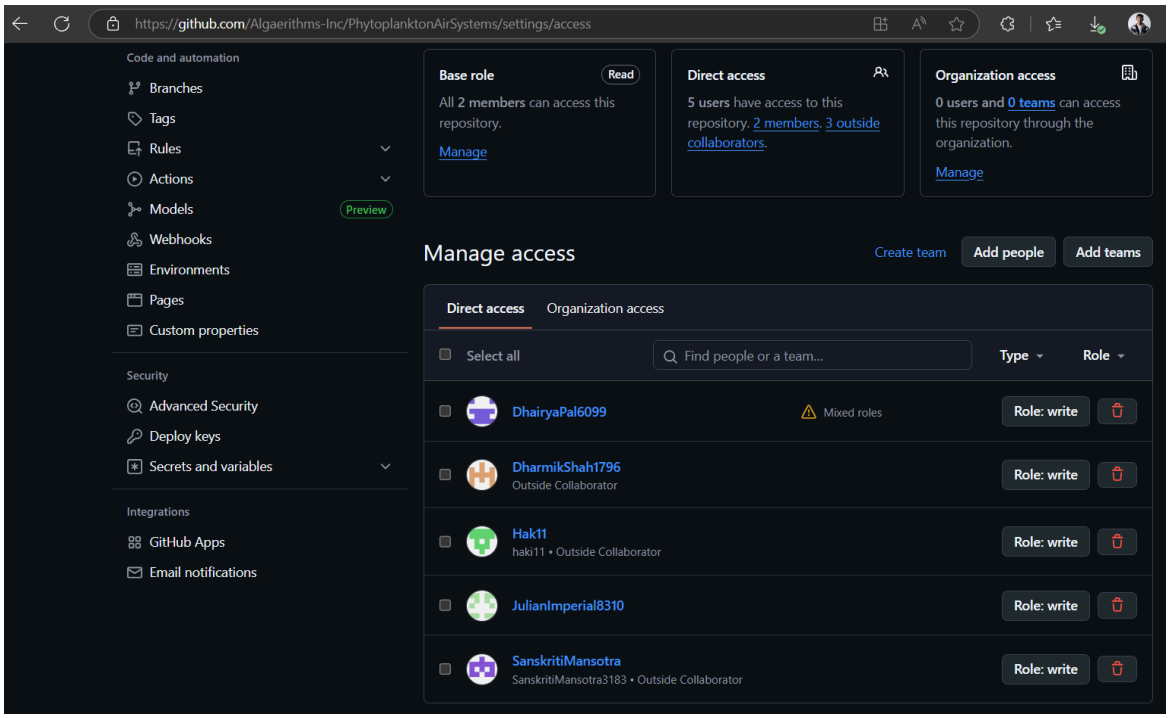
Table of contents -

Item	Page number
Team Contract	Attached separately with this document
Google Play Developer Account screenshot	Page 2
Prof invitation screenshot	Page 2
Github Repository Link	Page 3
Project Background and Description	Page 3 - 4
Project Scope	Page 4 - 5
Integration of software with hardware	Page 5 - 6
Project Layout	Page 6 - 7
Theme	Page 7 - 8

Google Play Developer Account Screenshot



Github prof invitation screenshot



Github Repository Link

[Algaerithms Repository]

Project Background and Description

<1. Describe the project goals and final vision.

The goal of our project is to create a smart, algae-based air purification system that helps improve indoor air quality in a sustainable and self-sufficient way. We are combining phytoplankton, which naturally convert CO_2 into oxygen, with real-time monitoring through an Android app. The final vision is to give users a visually engaging and interactive way to monitor air quality, while also promoting eco-conscious behavior. The system not only improves indoor environments by reducing pollution and increasing oxygen levels, but also raises awareness about climate-friendly technologies. Through features like gamified achievements, alerts, and live data tracking, we aim to make clean air more accessible, measurable, and meaningful for everyday users.

2. Describe the software aspect and hardware.

Software:

- Developed in Android Studio using Java
- Uses Firebase Realtime Database for cloud-based data storage
- The app displays real-time air quality data and includes features like:
    - Live sensor data (CO_2, humidity, temperature)
    - Gamified badges and achievements
    - Alerts when CO_2 levels are high
    - Clean UI for easy interaction

Hardware:

- Raspberry Pi (Main microcontroller)
- Connected sensors:
    - CO_2 sensor
    - Temperature & Humidity sensor
    - Proximity sensor (used to trigger app alerts or visual changes when someone is near the setup)
    - Light sensor (tracks sunlight, which is essential for phytoplankton photosynthesis)
    - Water level sensor (External – will be added in the final hardware phase to measure the water level)

- Raspberry Pi reads sensor values and sends them to Firebase over Wi-Fi

Software Components:

Android app: Developed in Java using Android Studio, it displays real-time air quality data, proximity sensor status, and system alerts. It also includes gamified features to engage users.

Firebase: Cloud-hosted real-time database used for data synchronization between hardware and the app.

3. Describe the screen flows.
The app flows in a simple, user-friendly manner:

- **Home Screen:**
  - Shows live readings: CO_2, temperature, humidity, light levels
  - Color-coded status for air quality
  - Proximity-based visual feedback if a user is near

- **Achievements Page:**
  - Rewards for keeping air clean or regular usage

- **Alerts / Notifications:**
  - Warnings of CO_2 levels exceeding a threshold

- **Info Screen:**
  - Educational content on phytoplankton and sustainability

4. How you Incorporated the feedback provided through the interview. N/A

5. Demonstrate how you are planning to satisfy to read / write from the DB which is hosted on the cloud. >

- The Firebase Realtime Database is used to store all sensor data.

- The Raspberry Pi collects sensor values and uploads them to Firebase

- The Android app reads this data live from Firebase to:
  - Display real-time readings on the home screen
  - Trigger visual changes, alerts, or achievements

- Write operations (e.g., resetting achievements, updating logs) can also be handled from the app if needed.

Project Scope

<Describe the technical scope of the project by talking about the project plan, and how you will know when the project is complete.>

The technical scope of this project involves designing and developing a smart, algae-based air purification system that integrates hardware sensor data with a real-time monitoring Android application. The software will read live air quality metrics, display them on a dashboard, and send user notifications based on predefined thresholds.

Key components of the project plan:

- Developing an Android app with a Navigation Drawer layout to navigate between Dashboard, Settings, and Summary pages.
- Integrating Firebase for user authentication, database storage, and cloud messaging for notifications.
- Implementing data visualization features such as graphs and air quality trend indicators.
- Interfacing the app with sensor data through mock inputs (initially) and later through Firebase/HTTP from the IoT module.
- Setting up logic to track and notify users about air quality status and system performance.
- Allowing users to customize notification preferences and view a daily summary.

The project will be considered complete when:

- Real-time air quality data is displayed in the app using simulated or live data.
- Users can view trends, receive notifications, and customize preferences.
- The app can successfully communicate with a cloud-hosted database (Firebase).
- The design is responsive and tested on Android devices.
- A final demonstration proves all main features function as intended.

Integration of Software with Hardware components

The integration between software and hardware in our system is designed using Firebase as the communication bridge. The Raspberry Pi collects real-time data from a variety of sensors and pushes that data to the Firebase Realtime Database. The Android app then pulls that data live and reacts accordingly, both functionally and visually.

Here's how each sensor plays a role in our system:

- Light Sensor: This checks whether the algae (phytoplankton) is getting enough sunlight for photosynthesis. If the light intensity drops below a threshold, the app triggers an alert and the system can activate an LED light source as a backup to maintain

photosynthesis. The app visually notifies the user when artificial light has been turned on.

- Turbidity Sensor: This sensor detects water clarity. If the turbidity increases (indicating dead algae or buildup on the container walls), the app will alert the user to perform maintenance or replace water. This ensures the algae remains effective at purifying air.

- Proximity Sensor: When someone approaches the system, the proximity sensor detects it and sends data to Firebase. The Android app uses this event to trigger visual animations (like simulating water waves or glowing light) to make the system look aesthetically pleasing and interactive.

- CO_2 Sensor: This is the core sensor for measuring the effectiveness of our purification system. It tracks the ambient CO_2 level and sends that data to Firebase. The app then uses this to display live air quality stats, issue alerts when levels are high, and contribute to the gamified achievement system for maintaining clean air.

Project Layout

We plan to implement the Navigation Drawer in our Android app as the main method for switching between different sections of the application. This layout provides a smooth, intuitive user experience by enabling users to easily navigate between screens without cluttering the UI.

- **Implementation Approach.**

We'll use Android Studio with Java.

The Navigation Drawer will be implemented using the DrawerLayout and NavigationView components.

It will be integrated with a Fragment-based architecture, where each menu item loads a different fragment (like Dashboard, Achievements, Info Page, etc.).

For combining layouts, we will use a tabs layout to move between the numerical stats of our product to the graph stats of our product + showing the citywide leaderboard and the countrywide leaderboard.

- **Why are we using a Navigation Drawer?**

It keeps the UI clean and minimal by hiding the menu until the user chooses to expand it.

Allows for scalable navigation across multiple app sections without overwhelming the user.

Follows Material Design Guidelines and is widely recognized by users, enhancing usability.

Use Cases in Existing Apps:
Gmail uses a Navigation Drawer to let users switch between inboxes, labels, and settings. And it also uses a bottom navigation bar to let users move between emails and Google Meet.

YouTube Music uses a Navigation Drawer when you click on your profile initial to let the users access user based settings that are categorized in the drawer. It also uses a bottom navigation bar to let users access different music related pages such as Search, Library, Explore, etc.

When to Use Navigation Drawer vs. Alternatives:
Use Navigation Drawer when you have more than 5 major sections or when the app's features are secondary in priority (e.g., settings, history, about).

Use Bottom Navigation when your app has 3–5 top-level sections that are frequently accessed.

In our app, since the features include multiple views (Live Dashboard, Achievements, Info, Settings, possibly Logs), the Navigation Drawer is more appropriate for organizing the app without cluttering the screen. With it, we can implement a tab layout to let the user switch between looking at a graph view of the stats or the numerical view of the stats. It can also be used to show the leaderboard countrywide, and the leaderboard citywide, etc.

Theme

Theme **Enhance the air quality and monitor the health of our sustainable air system.**	
Epic 1 **Implement Real-time air quality monitoring/dashboard**	**Epic 2** **Building smart notifications to inform users of their product's health**

Story 1.1 As a user I want to see the air quality on my floor in real time.			Story 1.2 As a user I want to see the patterns and trends over time, of the improved air quality.			Story 1.3 As a user I want to know when there may be a problem with the efficiency of my product.			Story 2.1 As a user I want to receive alerts when the air quality is poor so I can take action.			Story 2.2 As a user I want to be able to choose the preferences and frequency of notifications .			Story 2.3 As a user, I want a daily summary of air quality metrics based on my preferences		
T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T

	Task 1	Task 2	Task 3
Story 1.1	- Designing UI layout/dashboard for metrics	- Connecting mock data from sensors, which data will update from	- Color coded, air quality thresholds (green - good, red - poor, etc.)
Story 1.2	- Choosing a chart library to work with, what format	- Toggle buttons to change the metrics to be seen by a user	- Coding for data to be fetched from database (tbd)
Story 1.3	- Logic to check for sensor activity	- Snackbar/Warning to inform user, about sensor status	- UI when sensor, is inactive and not providing data (test-case)
Story 2.1	- Define air quality threshold ranges/values.	- Implementing Firebase messaging in relation to threshold values	- Notification logic (silent notification for off-hours)
Story 2.2	- Creating a settings fragment for user preferences	- Storing these preferences in the SharedPreferences or Firebase DB	- Updating the notification logic, with these user preferences if any
Story 2.3	- Scheduling a daily summary notification	- Summary screen to be opened when notification is pressed	- Collect data in a 24-hour cycles, to check metrics