Document Title: Deliverable 3

Team Name: Smart Sprout

Project Name: Smart Home-Garden System

Group Number: Group 9

Team Member & Student ID:

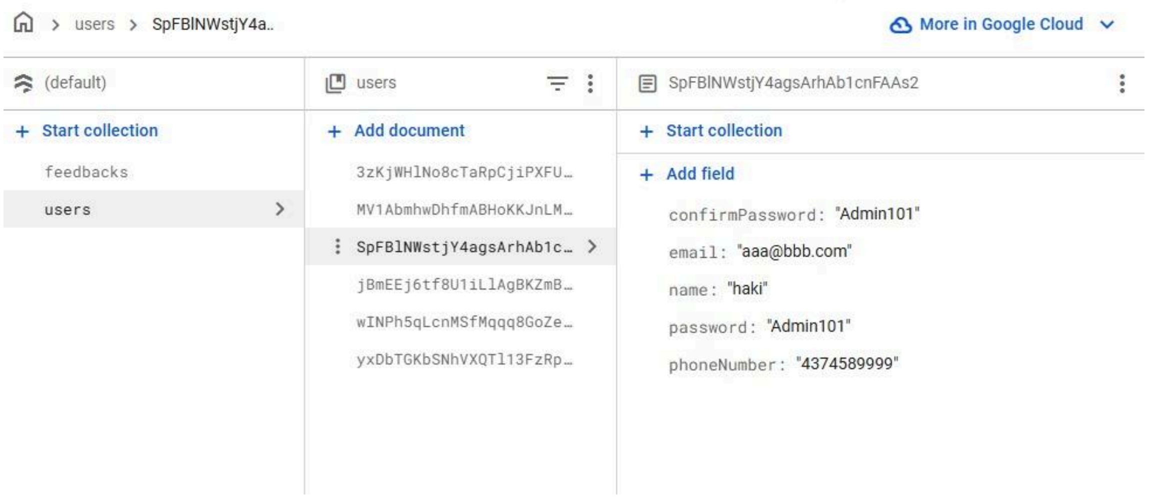
Aditi Patel	n01525570
Birava Prajapati	n01579924
Darshankumar Prajapati	n01574247
Zeel Patel	n01526282

Table Of Content

Name	Student Id	Github Id	Signature	Efforts
Aditi Patel	n01525570	AditiPatel5570	Apatel	100%
Birava Prajapati	n01579924	BiravaPrajapati9924	Biravap	100%
Darshankumar Prajapati	n01574247	DarshanPrajapati4247	Darshan	100%
Zeel Patel	n01526282	ZeelPatel6282	Zeel	100%

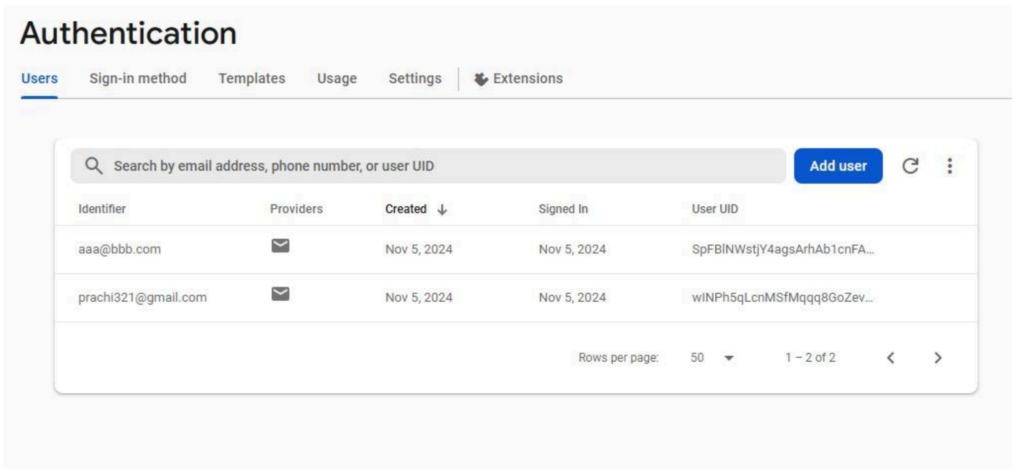
- **GitHub Repo Link**: https://github.com/DarshanPrajapati4247/SmartHomeGardenSys

Database for login


users > SpFBlNWstjY4a..

More in Google Cloud

(default)	users	SpFBlNWstjY4agsArhAb1cnFAAs2
+ Start collection	+ Add document	+ Start collection
feedbacks	3zKjWHlNo8cTaRpCjiPXFU…	+ Add field
users	MV1AbmhwDhfmABHoKKJnLM…	confirmPassword: "Admin101"
	SpFBlNWstjY4agsArhAb1c…	email: "aaa@bbb.com"
	jBmEEj6tf8U1iLlAgBKZmB…	name: "haki"
	wINPh5qLcnMSfMqqq8GoZe…	password: "Admin101"
	yxDbTGKbSNhVXQTl13FzRp…	phoneNumber: "4374589999"


Authentication

Users | Sign-in method | Templates | Usage | Settings | Extensions

Search by email address, phone number, or user UID — Add user

Identifier	Providers	Created	Signed In	User UID
aaa@bbb.com	✉	Nov 5, 2024	Nov 5, 2024	SpFBlNWstjY4agsArhAb1cnFA…
prachi321@gmail.com	✉	Nov 5, 2024	Nov 5, 2024	wINPh5qLcnMSfMqqq8GoZev…

Rows per page: 50 1 – 2 of 2

<u>Sprint 3: Work done by all the members.</u>

Zeel Patel: I have successfully completed several tasks in my project. First, I integrated Firestore to store user details such as name, email, phone, and password. I also developed the feedback functionality, which displays user information and saves feedback (including description, rating, and device model) to Firestore. Additionally, I began the integration of Google Sign-In to streamline user authentication. On the login screen, I added the "Remember Me" feature, validated the email format, and set constraints for the password length. Throughout these tasks, I conducted thorough testing to ensure smooth user experience and proper data handling.

Darshan Prajapati: I have mainly worked on the search and diagnosis screens, which I designed. For now, functionality has not been added. Also, I worked on the feedback screen and its functionality with Zeel.

Birava Prajapati: I have worked on the profile screen UI and settings screen functionalities. I added features to the app's settings screen. There's now an option to enable or disable notifications, which saves the preference even if the app is closed or restarted. I also added a "Log Out" button that clears session data and sends the user back to the login screen. A "Feedback" option is included in the UI. I added a footer that shows the app's version number. The settings screen now includes options for Dark Mode and notification preferences, which are saved using SharedPreferences so they stay the same after the app restarts or the device reboots.

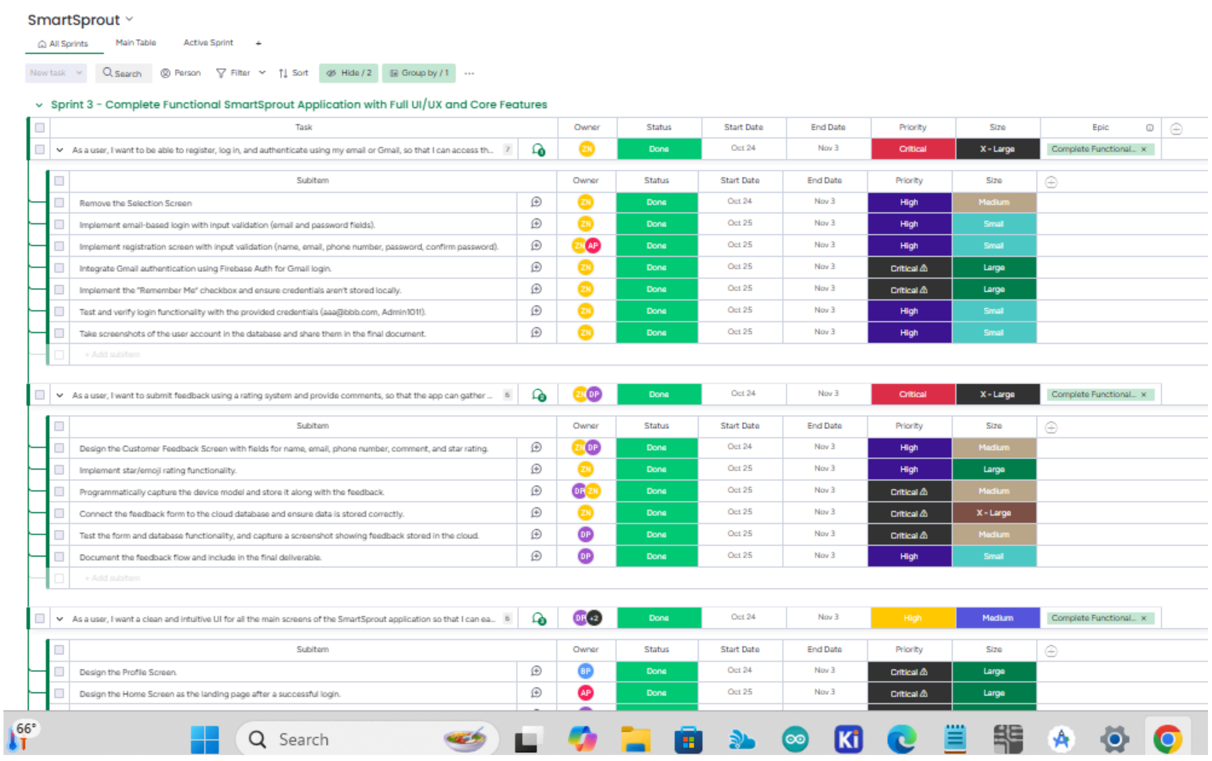
Aditi Patel: I have implemented the UI and functionality for both the Home and Sensor Data screens. The Home screen features a button that takes users to the search screen, where they can add a plant. In the future, this button's functionality will evolve to allow users to add tasks. For the Sensor Data screen, I've added hard-coded values in the fragment, which are sent to the real-time database and stored. These values are then fetched and displayed, updating the progress bar and TextView. Eventually, this will be upgraded to pull live data from actual sensors. I've integrated the real-time database for this feature.

<u>Sprint Goals</u>

The goal of this sprint is to complete core functionalities, database integrations, UI designs, and settings required in Deliverable 3, ensuring full compliance with project standards. This includes:

1. **UI Design Completion**:
    - Finalize the UI design for **all screens** (Home, Add Plant, Login, Registration, Feedback, Settings, Sensor Data).
    - Ensure each screen follows a consistent design language and aligns with UX best practices for intuitive navigation and usability.
2. **User Authentication and Navigation**:
    - Implement **Google Login and Sign Up** functionality, allowing users to register or log in using their Gmail credentials, with secure data handling.
    - Create the **“Add Plant” button** on the Home screen, with navigation to the Search screen for plant selection. Validate button responsiveness and navigation as per user flow requirements.
3. **Settings Screen Functionality**:
    - Develop and complete the **Settings screen** with essential options including:
        - **Theme selection**: Allow users to switch between Dark, Light, and System themes, with preferences stored in SharedPreferences and retained across app restarts.
        - **Notification settings**: Enable toggle to turn notifications on or off.
        - **Logout functionality**: Provide a clear logout option for users.
        - **Feedback option**: Ensure space and UI elements are in place for future functionality.
4. **Sensor Data Transmission**:
    - Integrate Firebase real-time database for **sensor data management**, including moisture, sunlight, and temperature readings. Transmit hardcoded values for now to demonstrate the data pipeline.
    - Verify and document data transfer from the app to Firebase and back, preparing for dynamic sensor data in future sprints.
5. **Runtime Permissions for Notifications**:
    - Implement Android 13 **runtime permissions for notifications**. Provide users with a notification prompt, directing them to settings if permissions are denied.
    - Include utility methods to manage permission requests, ensuring app compliance with Android standards.

<u>Tasks and Stories:</u>



Sprint 3 - Complete Functional SmartSprout Application with Full UI/UX and Core Features

Task		Owner	Status	Start Date	End Date	Priority	Size	Epic
As a user, I want to be able to register, log in, and authenticate using my email or Gmail, so that I can access th...	7	ZN	Done	Oct 24	Nov 3	Critical	X - Large	Complete Functional...

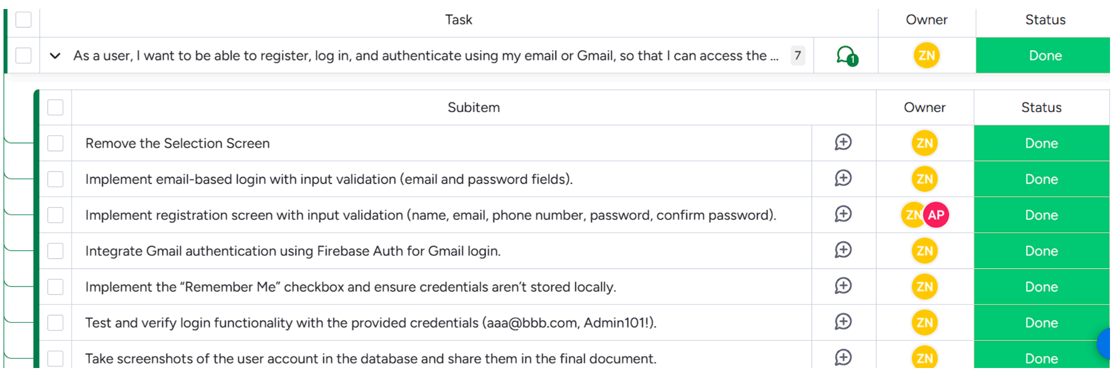
Subitem	Owner	Status	Start Date	End Date	Priority	Size
Remove the Selection Screen	ZN	Done	Oct 24	Nov 3	High	Medium
Implement email-based login with input validation (email and password fields).	ZN	Done	Oct 25	Nov 3	High	Small
Implement registration screen with input validation (name, email, phone number, password, confirm password).	ZN AP	Done	Oct 25	Nov 3	High	Small
Integrate Gmail authentication using Firebase Auth for Gmail login.	ZN	Done	Oct 25	Nov 3	Critical ⚠	Large
Implement the "Remember Me" checkbox and ensure credentials aren't stored locally.	ZN	Done	Oct 25	Nov 3	Critical ⚠	Large
Test and verify login functionality with the provided credentials (aaa@bbb.com, Admin101!).	ZN	Done	Oct 25	Nov 3	High	Small
Take screenshots of the user account in the database and share them in the final document.	ZN	Done	Oct 25	Nov 3	High	Small
+ Add subitem						

Task		Owner	Status	Start Date	End Date	Priority	Size	Epic
As a user, I want to submit feedback using a rating system and provide comments, so that the app can gather ...	6	ZN DP	Done	Oct 24	Nov 3	Critical	X - Large	Complete Functional...

Subitem	Owner	Status	Start Date	End Date	Priority	Size
Design the Customer Feedback Screen with fields for name, email, phone number, comment, and star rating.	ZN DP	Done	Oct 24	Nov 3	High	Medium
Implement star/emoji rating functionality.	ZN	Done	Oct 25	Nov 3	High	Large
Programmatically capture the device model and store it along with the feedback.	DP ZN	Done	Oct 25	Nov 3	Critical ⚠	Medium
Connect the feedback form to the cloud database and ensure data is stored correctly.	ZN	Done	Oct 25	Nov 3	Critical ⚠	X - Large
Test the form and database functionality, and capture a screenshot showing feedback stored in the cloud.	DP	Done	Oct 25	Nov 3	Critical ⚠	Medium
Document the feedback flow and include in the final deliverable.	DP	Done	Oct 25	Nov 3	High	Small
+ Add subitem						

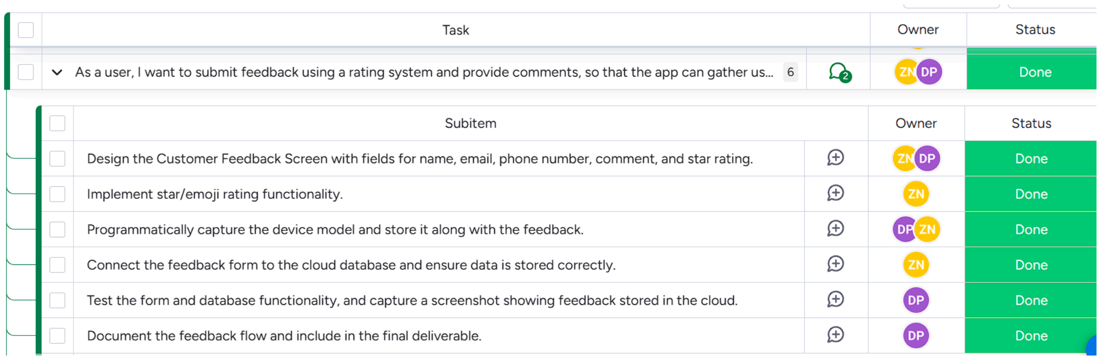
Task		Owner	Status	Start Date	End Date	Priority	Size	Epic
As a user, I want a clean and intuitive UI for all the main screens of the SmartSprout application so that I can ea...	6	DP +2	Done	Oct 24	Nov 3	High	Medium	Complete Functional...

Subitem	Owner	Status	Start Date	End Date	Priority	Size
Design the Profile Screen.	BP	Done	Oct 24	Nov 3	Critical ⚠	Large
Design the Home Screen as the landing page after a successful login.	AP	Done	Oct 25	Nov 3	Critical ⚠	Large



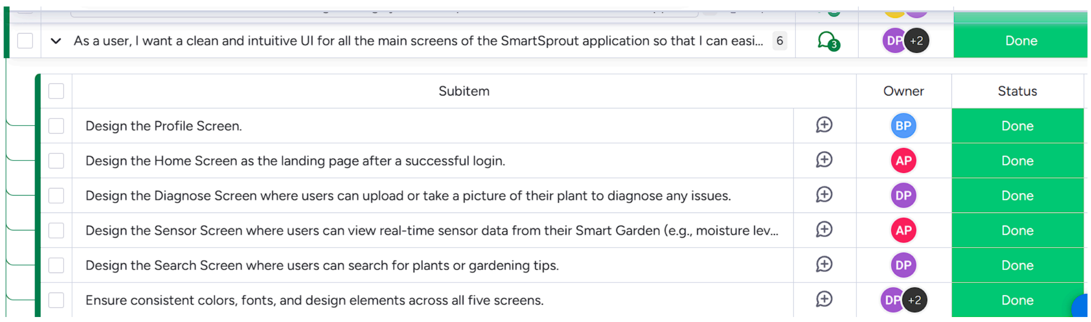
Task		Owner	Status
As a user, I want to be able to register, log in, and authenticate using my email or Gmail, so that I can access the ...	7	ZN	Done

Subitem	Owner	Status
Remove the Selection Screen	ZN	Done
Implement email-based login with input validation (email and password fields).	ZN	Done
Implement registration screen with input validation (name, email, phone number, password, confirm password).	ZN AP	Done
Integrate Gmail authentication using Firebase Auth for Gmail login.	ZN	Done
Implement the "Remember Me" checkbox and ensure credentials aren't stored locally.	ZN	Done
Test and verify login functionality with the provided credentials (aaa@bbb.com, Admin101!).	ZN	Done
Take screenshots of the user account in the database and share them in the final document.	ZN	Done



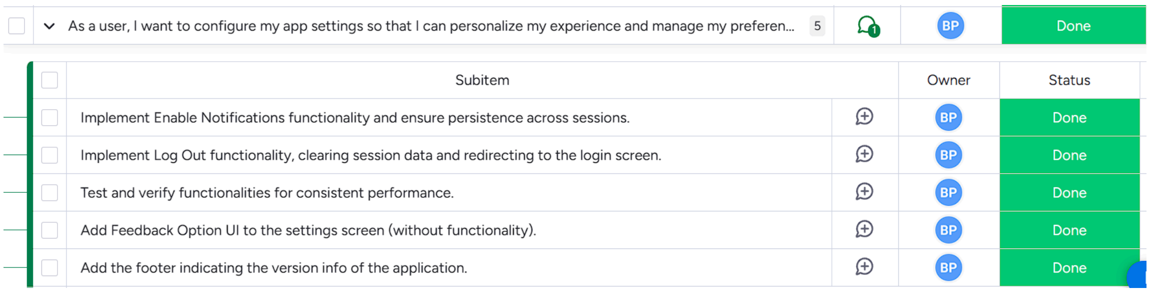
Task		Owner	Status
As a user, I want to submit feedback using a rating system and provide comments, so that the app can gather us...	6	ZN DP	Done

Subitem	Owner	Status
Design the Customer Feedback Screen with fields for name, email, phone number, comment, and star rating.	ZN DP	Done
Implement star/emoji rating functionality.	ZN	Done
Programmatically capture the device model and store it along with the feedback.	DP ZN	Done
Connect the feedback form to the cloud database and ensure data is stored correctly.	ZN	Done
Test the form and database functionality, and capture a screenshot showing feedback stored in the cloud.	DP	Done
Document the feedback flow and include in the final deliverable.	DP	Done



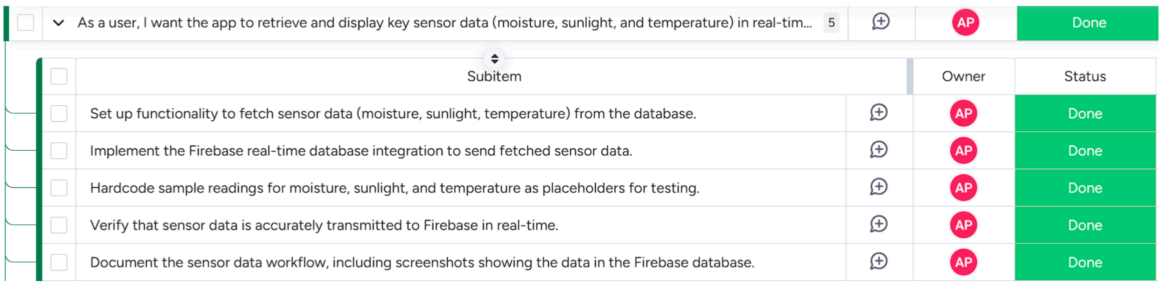
As a user, I want a clean and intuitive UI for all the main screens of the SmartSprout application so that I can easi...	6		DP +2	Done

Subitem		Owner	Status
Design the Profile Screen.		BP	Done
Design the Home Screen as the landing page after a successful login.		AP	Done
Design the Diagnose Screen where users can upload or take a picture of their plant to diagnose any issues.		DP	Done
Design the Sensor Screen where users can view real-time sensor data from their Smart Garden (e.g., moisture lev...		AP	Done
Design the Search Screen where users can search for plants or gardening tips.		DP	Done
Ensure consistent colors, fonts, and design elements across all five screens.		DP +2	Done



As a user, I want to configure my app settings so that I can personalize my experience and manage my preferen...	5		BP	Done

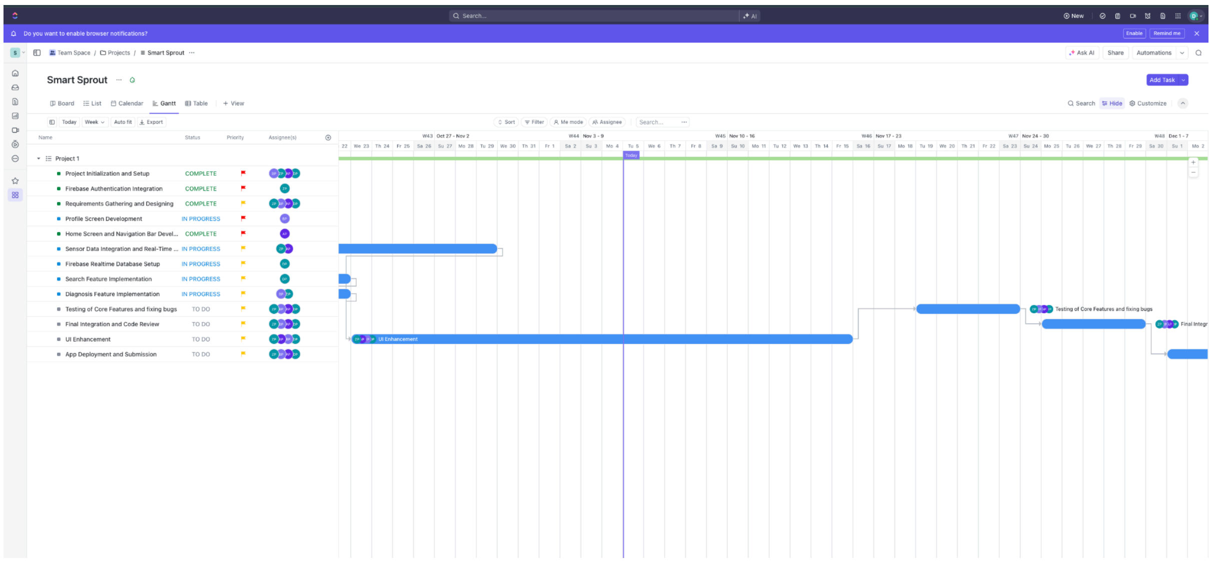
Subitem		Owner	Status
Implement Enable Notifications functionality and ensure persistence across sessions.		BP	Done
Implement Log Out functionality, clearing session data and redirecting to the login screen.		BP	Done
Test and verify functionalities for consistent performance.		BP	Done
Add Feedback Option UI to the settings screen (without functionality).		BP	Done
Add the footer indicating the version info of the application.		BP	Done



As a user, I want the app to retrieve and display key sensor data (moisture, sunlight, and temperature) in real-tim...	5		AP	Done

Subitem		Owner	Status
Set up functionality to fetch sensor data (moisture, sunlight, temperature) from the database.		AP	Done
Implement the Firebase real-time database integration to send fetched sensor data.		AP	Done
Hardcode sample readings for moisture, sunlight, and temperature as placeholders for testing.		AP	Done
Verify that sensor data is accurately transmitted to Firebase in real-time.		AP	Done
Document the sensor data workflow, including screenshots showing the data in the Firebase database.		AP	Done

Gantt Chart



Team Space / Projects / Smart Sprout

Smart Sprout

Name	Status	Assignee(s)
Project 1		
Project Initialization and Setup	COMPLETE	BP ZP AP DP
Firebase Authentication Integration	COMPLETE	ZP
Requirements Gathering and Designing	COMPLETE	ZP BP AP DP
Profile Screen Development	IN PROGRESS	BP
Home Screen and Navigation Bar Devel...	COMPLETE	AP
Sensor Data Integration and Real-Time ...	IN PROGRESS	ZP AP
Firebase Realtime Database Setup	IN PROGRESS	DP
Search Feature Implementation	IN PROGRESS	DP
Diagnosis Feature Implementation	IN PROGRESS	BP DP
Testing of Core Features and fixing bugs	TO DO	ZP BP AP DP
Final Integration and Code Review	TO DO	ZP BP AP DP
UI Enhancement	TO DO	ZP AP BP DP
App Deployment and Submission	TO DO	ZP BP AP DP

Daily Standup

Oct 24, 2024

	What did I do yesterday?	What will I do today?	Any Blockers?
Darshan Prajapti	Create the user stories.	Build the sprint dashboard and assinging tasks.	None
Birava Prajapati	Implemented the back pressed funtionality in settings screen.	Will implemente the functionalities in settings screen.	None
Aditi Patel	Was trying to open Android studio	Will add runtime implementation according to MVVM model	Was having error for opening android studio
Zeel Patel	Research on working with firestore database	Create firestore database	None

Oct 30, 2024

	What did I do yesterday?	What will I do today?	Any Blockers?
Darshan Prajapti	Research on google sign in implementation.	Design the search screen fragment.	None.
Birava Prajapati	Implemented the some functionality to settings screen.	Design the ui of profile screen.	None
Aditi Patel	Added Sensor screen UI and hardcoded values	Will add a button that navigates to the search screen	no blockers yet
Zeel Patel	Add sign in gmail provider in firebase authentication	Reemove selection screen and addedGmail functionality	no blockers yet

Nov 4, 2024

	What did I do yesterday?	What will I do today?	Any Blockers?
Darshan Prajapti	Helping team members for resolving errors.	Test and review the code for merging in the main branch.	None.
Birava Prajapati	Implemented the notification and theme functionality.	Will try to enhance the ui of profile screen.	None.
Aditi Patel	Added the button theme	I will add realtime database to fetch data on the senor screen.	Pull issue.
Zeel Patel	Design the layout of feedback screen in figma	Test the screens of login and registration	Google declared some method depricated which I used for sign in with gmail

SMART SPROUT

Design Principles

MVVM Architecture Explanation

- **Model**: The `Model` layer is responsible for handling the data logic of the application. It includes data classes and repositories, which fetch data from the Firebase real-time database or other data sources. This layer is represented by the `data` folder, where the `model` and `repository` subfolders reside.
    - **Model Classes**: `Feedback`, `Plant`, `SensorData`, `User`, and `WeatherResponse` define the data structure for different entities.
    - **Repository Classes**: `FirebaseRepository`, `ProfileRepository`, and `WeatherRepository` manage data operations, providing a single source of truth and ensuring data consistency across the application.
- **ViewModel**: The `ViewModel` layer acts as a bridge between the Model and the View. It exposes data from the Model to the View in a lifecycle-aware manner, preventing memory leaks and ensuring data persists during configuration changes. Each feature has a dedicated ViewModel class under the `viewmodels` folder.
    - **Example ViewModels**: `AuthViewModel`, `FeedbackViewModel`, `NavigationViewModel`, `NotificationViewModel`, `PlantViewModel`, `SensorViewModel`, `SessionViewModel`, `ThemeViewModel`, and `WeatherViewModel` handle data for their respective views and provide necessary logic for user interactions.
- **View (UI)**: The `View` layer is responsible for displaying the data to the user and capturing user interactions. In Android, it consists of `Activities` and `Fragments` that display UI elements.
    - **Activities and Fragments**: The `HomeFragment`, `ProfileFragment`, `SearchFragment`, and others represent different screens of the application. Activities such as `LoginActivity` and `SettingsActivity` handle user login and settings options, while fragments provide a modular UI structure.

SmartSprout Project Directory Structure

Below is the directory structure of the SmartSprout project, highlighting the organization of the MVVM components

```
SmartHomeGarden
|
├── manifests
|
├── java
|   └── ca.smartsprout.it.smart.smarthomegarden
|       ├── data
|       |   ├── model
|       |   |   ├── Feedback
|       |   |   ├── Plant
|       |   |   ├── SensorData
|       |   |   ├── User
|       |   |   └── WeatherResponse
|       |   └── repository
|       |       ├── FirebaseRepository
|       |       ├── ProfileRepository
|       |       ├── WeatherRepository
|       |       └── WeatherService
|       ├── ui
|       |   ├── adapter
|       |   |   └── PlantAdapter
|       |   └── fragments
|       |       ├── DiagnoseFragment
|       |       ├── HomeFragment
|       |       ├── ProfileFragment
|       |       ├── SearchFragment
|       |       ├── SensorFragment
|       |       ├── BaseActivity
|       |       ├── FeedbackActivity
|       |       ├── LoginActivity
|       |       ├── RegistrationActivity
|       |       ├── SettingsActivity
|       |       └── SplashActivity
|       └── utils
|           ├── Constants
|           └── Util
└── viewmodels
    ├── AuthViewModel
    ├── FeedbackViewModel
    ├── NavigationViewModel
    ├── NotificationViewModel
    ├── PlantViewModel
    ├── SensorViewModel
    ├── SessionViewModel
    ├── ThemeViewModel
    └── WeatherViewModel
```

Advantages of MVVM in SmartSprout

1. **Separation of Concerns**: Each component in the MVVM architecture is responsible for a specific role, making the codebase easier to maintain and extend.
2. **Testability**: ViewModels can be tested independently from the Views, allowing for effective unit testing.
3. **Lifecycle Awareness**: By leveraging ViewModel and LiveData (if used), data survives configuration changes (e.g., screen rotations), improving user experience and app stability.
4. **Scalability**: MVVM architecture provides a scalable structure for adding new features, as each ViewModel, Model, and View can be developed independently.

This directory structure and architecture ensure a clean and modular design, making SmartSprout robust and ready for future enhancements.

Work Progress

Since Deliverable 2, significant advancements have been made in the Smart Sprout project. All screen UIs have been developed, providing a seamless user experience across the app. The functionality within the Home fragment has been enhanced to improve navigation and user interaction. Notifications have been integrated, ensuring users are promptly alerted to relevant updates. Sensor data is now being successfully stored in the database, laying the foundation for future real-time updates from actual sensors. Additionally, user registration functionality has been added to the database, allowing for user-specific data storage. The settings screen has been completed, giving users control over app preferences and a dark theme has been implemented for improved accessibility and aesthetics. These improvements align with the objectives of Deliverable 3 and contribute to the overall functionality and user experience of the Smart Sprout app.

Runtime permission implemented.

– We have implemented location runtime permission and notification runtime permission.

- **Location runtime permission:** // This will take the user permission for accessing the location as we have integrated weather api.

```
private final ActivityResultLauncher<String[]> requestPermissionLauncher =
        registerForActivityResult(new ActivityResultContracts.RequestMultiplePermissions(),
result -> {
            Boolean fineLocationGranted =
result.getOrDefault(Manifest.permission.ACCESS_FINE_LOCATION, false);
            Boolean coarseLocationGranted =
result.getOrDefault(Manifest.permission.ACCESS_COARSE_LOCATION, false);

            if (fineLocationGranted != null && fineLocationGranted) {
                fetchWeatherData();
                Snackbar.make(requireView(), R.string.location_permission_granted,
Snackbar.LENGTH_SHORT).show();
            } else if (coarseLocationGranted != null && coarseLocationGranted) {
                fetchWeatherData();
                Snackbar.make(requireView(), R.string.location_permission_granted,
Snackbar.LENGTH_SHORT).show();
            } else {
                handlePermissionDenied();
            }
        });
```

- **Notification runtime permission: //** This runtime permission setup ensures that the app requests notification permissions from the user when needed, particularly for Android 13 (API level TIRAMISU) and above, as required for managing notifications.

```
private ActivityResultLauncher<String> requestPermissionLauncher =
registerForActivityResult(
    new ActivityResultContracts.RequestPermission(),
    isGranted -> {
        if (isGranted) {
            notificationViewModel.updateNotificationPermission(true);
        } else {
            notificationViewModel.updateNotificationPermission(false);
            Util.showSnackbar(requireView(),
getString(R.string.notification_permission_snackbar), true);
        }
    }
);

private void checkNotificationPermission() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
        if (ContextCompat.checkSelfPermission(requireContext(),
Manifest.permission.POST_NOTIFICATIONS)
                != PackageManager.PERMISSION_GRANTED) {
            requestPermissionLauncher.launch(Manifest.permission.POST_NOTIFICATIONS);
        } else {
            notificationViewModel.updateNotificationPermission(true); }}}
```

Functionalities

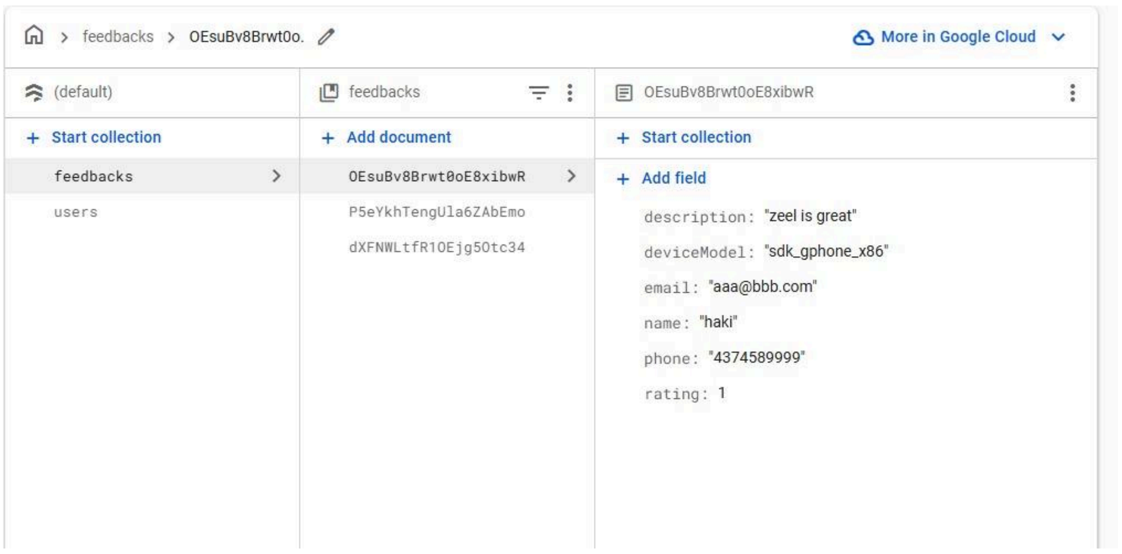
Add Button on Home Screen:

- **Description**: A button was added to the Home screen to allow users to navigate to the search screen. This button is designed to enable users to add a plant to their smart home garden. The feature is a part of the ongoing development of the app's task management system, where users will eventually be able to add specific tasks related to their plants.
- **Purpose**: To streamline plant management by providing an easy way for users to add new plants to their gardens.

Sensor Data Screen:

- **Description**: The Sensor Data screen was implemented with hard-coded values for sensors. These values are displayed on the screen and stored in the Firestore database for future use. The feature will eventually be upgraded to pull live data from real sensors.
- **Purpose**: To allow users to monitor and store sensor data, enabling the app to manage environmental conditions in real-time, which will enhance plant care and maintenance features.

Feedback Database:



```
description: "zeel is great"
deviceModel: "sdk_gphone_x86"
email: "aaa@bbb.com"
name: "haki"
phone: "4374589999"
rating: 1
```