Document Title: Deliverable 5

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

Brief description of the project

Our project, **Smart Sprout**, is a smart home gardening app designed to simplify plant care by integrating sensor data. It helps users monitor their plants' needs, such as moisture and temperature. The app automates plant care by offering notifications, weather updates, and a user-friendly interface for managing and tracking plant health, making it easy for users to care for their plants effortlessly.

Name	Student Id	Github Id	Signature	Efforts
Aditi Patel	n01525570	AditiPatel5570		100%
Birava Prajapati	n01579924	BiravaPrajapati9924		100%
Darshankumar Prajapati	n01574247	DarshanPrajapati4247		90%
Zeel Patel	n01526282	ZeelPatel6282		80%

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

Sprint 5 goals

1. **Complete Application Functionality**:
    - Finalize and implement all remaining features to ensure the SmartSprout app is fully functional and user-ready.

2. **Seamless User Experience**:
    - Enhance navigation flows (e.g., back navigation, clear notifications, feedback submission) and improve UI/UX consistency across all screens.

3. **Robust Data Management**:
    - Ensure all data is dynamically retrieved and stored using Firebase Authentication, Firestore, and Firebase Realtime Database, eliminating hardcoded data.

4. **Offline Mode Implementation**:
    - To ensure the app remains usable without internet connectivity, add offline functionality for specific features (e.g., viewing stored plant data).

5. **Technical Debt Resolution**:
    - Refactor code for maintainability, improve input validation and centralize database interactions to address technical debt from previous sprints.

6. **Testing and Stability**:
    - Implement comprehensive unit tests and integration tests for critical components to ensure functionality, stability, and performance.

7. **Menu and Notification Refinement**:
    - Redesign the menu structure, relocate misplaced features, and implement notification runtime permissions for API 33 and higher.

8. **Plant API Integration**:
    - Replace the existing plant API with a new one, updating the search functionality and ensuring smooth data handling.

9. **Feedback Integration and Validation**:
    - Complete the feedback screen with form validation, progress indicators, and confirmation dialogs.

10. **Deliverable Completion**:
    - Prepare and finalize all necessary deliverable documents, including updated diagrams, test reports, and a fully functional app for submission.

3 tasks from the Scrum dashboard that are related to addressing technical debt

- **Moving Unit Test Cases into the Correct Package**
    1. Relocating unit test cases from the Android test package to the test package.
    2. Enhances code organization by separating unit tests from integration and UI tests.
    3. Ensures faster execution of unit tests and simplifies debugging.
- **Refactoring Large Classes into Smaller Ones**
    1. Splitting big classes into smaller, more focused classes.
    2. Improves code readability and maintainability.
    3. Reduces complexity and promotes adherence to the Single Responsibility Principle.
- **Encrypting Passwords in the Database**
    1. Implementing encryption for passwords stored in the database.
    2. Protects sensitive user information and mitigates risks of data breaches.
    3. Ensures compliance with modern security standards and best practices.

The work that has been completed by each team member in this sprint only.

Aditi: In this sprint, I introduced a task custom bottom sheet feature to the Smart Sprout app, providing users with an intuitive way to create and manage tasks seamlessly. This functionality ensures that tasks are displayed as timely reminders at the user-specified time and includes an additional notification 30 minutes before the set time, enhancing preparedness and organization. Moreover, I completed the notification screen, which offers a dedicated interface for users to view and manage their notifications effortlessly, promoting better engagement with app alerts . Finally, I finalized the sensor data functionality, ensuring the integration of real-time or stored sensor data within the app. This feature enables users to monitor essential metrics effectively, fostering informed decision-making and an enhanced user experience. I also changed the password encryption in the Database and made espresso test cases.

Birava: In this sprint, I worked on enhancing the Smart Sprout app's functionality and user experience. A key improvement was implementing a Pinterest-style photo grid in the ProfileFragment, allowing users to view their photos in a visually appealing layout. This was supported by adding database and UI groundwork for seamless integration and introducing spacing between grid items in the RecyclerView for a polished presentation. To enhance navigation, I implemented tab-based RecyclerView toggling in the ProfileFragment, making it easier for users to switch between views.

For image management, I centralized logic into the ImagePickerHandler, added immediate permission handling for smoother interactions, and introduced an ImagePickerDialog for the "Add Plant Picture" FAB button. Additionally, I enhanced app usability by refactoring unit conversion logic and adding a toggle in the settings screen to switch between units. I also aligned password validation with the MVVM architecture to ensure code maintainability.

To address connectivity issues, I added a method to inflate a "No Internet Connection" dialog in NetworkUtils and designed the corresponding `dialog_no_internet.xml` layout, improving the app’s resilience and user communication during offline scenarios. These updates collectively enhance the app's usability, reliability, and maintainability.

Darshan: In this sprint, I focused on implementing key features and resolving critical issues to enhance the app's functionality and reliability. One of my main contributions was developing the offline activity, which allows users to access essential features even without an internet connection. I also integrated live network checking functionality, ensuring that the app can seamlessly switch between offline and online modes while providing real-time feedback to users about their connectivity status.

Additionally, I addressed and resolved several major errors related to exception handling, improving the app's stability and user experience. To further enhance the app's capabilities, I implemented the functionality to add plants by searching for their names. This feature leverages the Plant API, allowing users to quickly and accurately find and add plants to their profiles. These updates reflect my focus on creating a more robust, user-friendly, and efficient application.

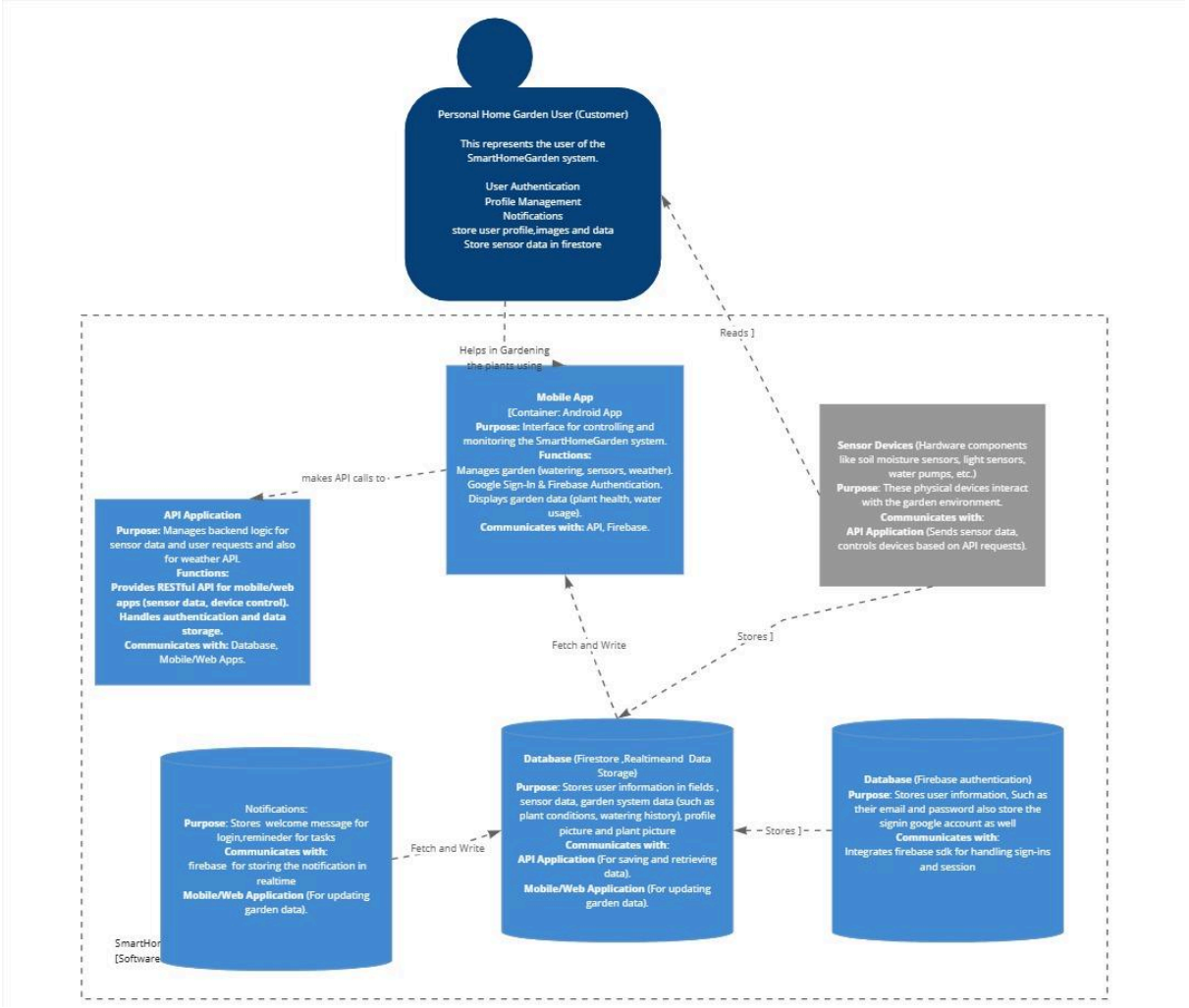
Zeel:

Throughout the sprint, I focused on implementing significant updates and new features to enhance the app's functionality and user experience. One of my primary contributions was developing the FAQ functionality, which included creating a new FAQActivity, updating the FAQ toolbar, and integrating back-pressed functionality for seamless navigation. I also replaced the "Help" menu with an FAQ option and removed hardcoded strings across the FAQ and other activities to improve localization and maintainability.

I worked on refining the user interface across various screens, including updating the registration and forgot password screens, removing unnecessary toolbars from the Login and Registration screens, and improving the layout of the RegistrationActivity. To simplify navigation, I replaced the Google Sign-In feature with a "Back to Login" option. Additionally, I implemented a "Remember Me" feature to provide persistent login functionality, making the app more convenient for users.

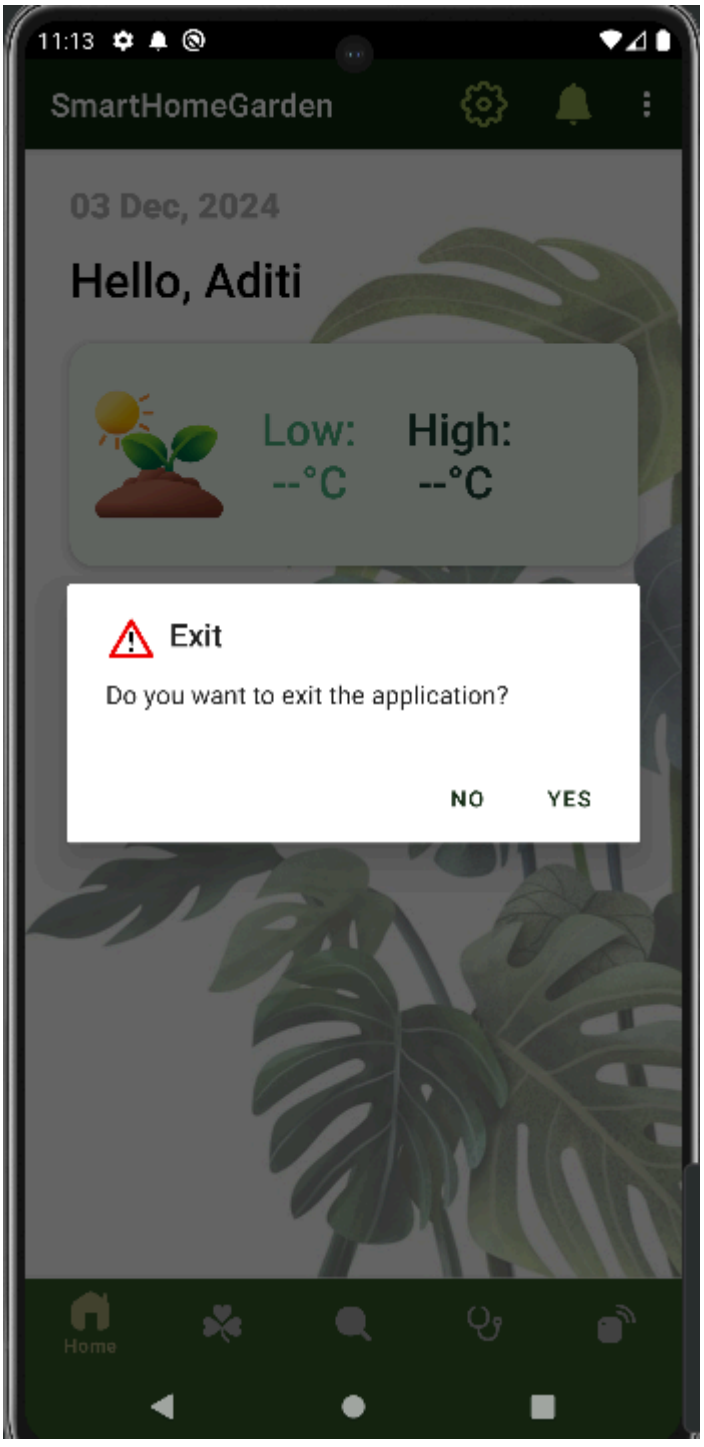
To support these improvements, I updated the MenuHandler class to incorporate the new FAQ functionality. I also created JUnit test cases for critical features like the FAQ screen and RecyclerView setup to ensure reliability. These updates reflect my commitment to improving usability, maintaining clean and scalable code, and adhering to best practices.

Using the C4 Model



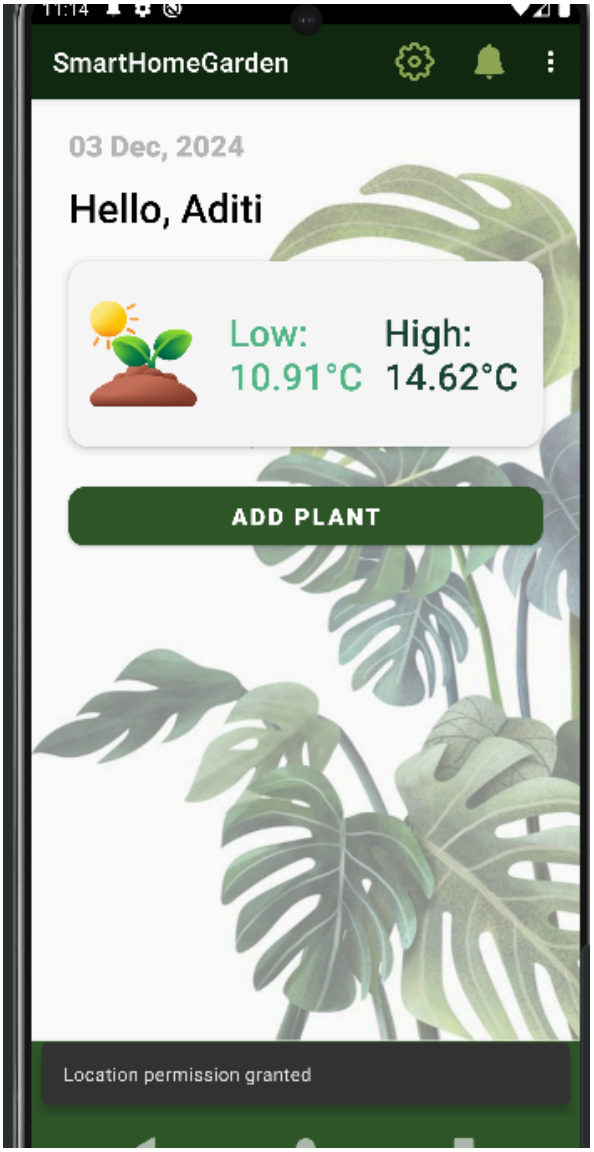
Demonstrating usage

A. AlertDialog



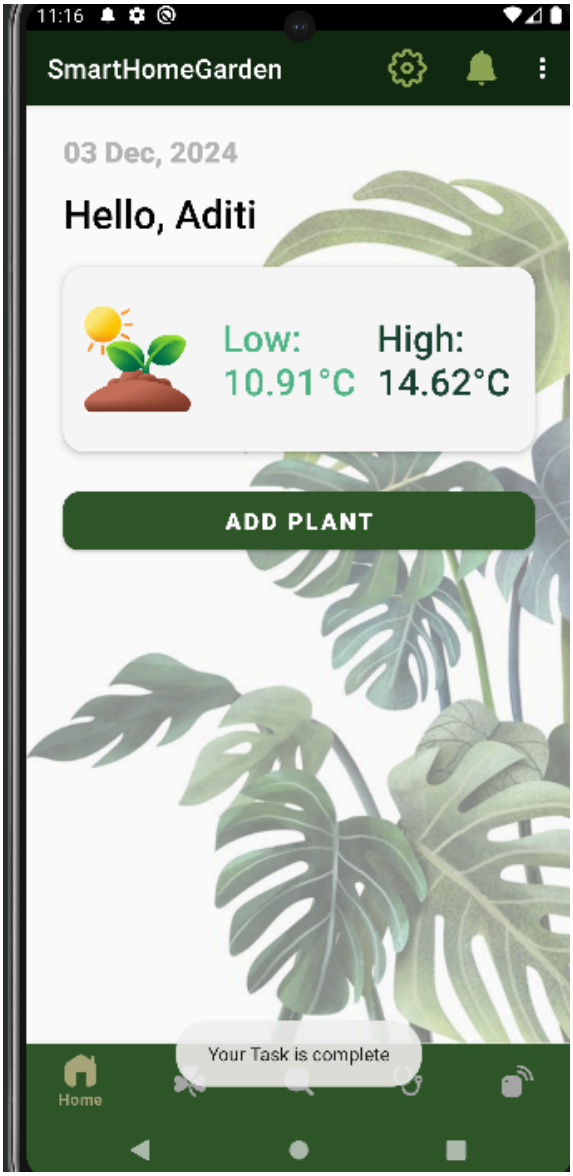
The Alert dialog box we are using appears when the user clicks on the back button on the app and tries to exit the application. It gives the user two option if he wants to exit or stays in the application.

B. Snackbar



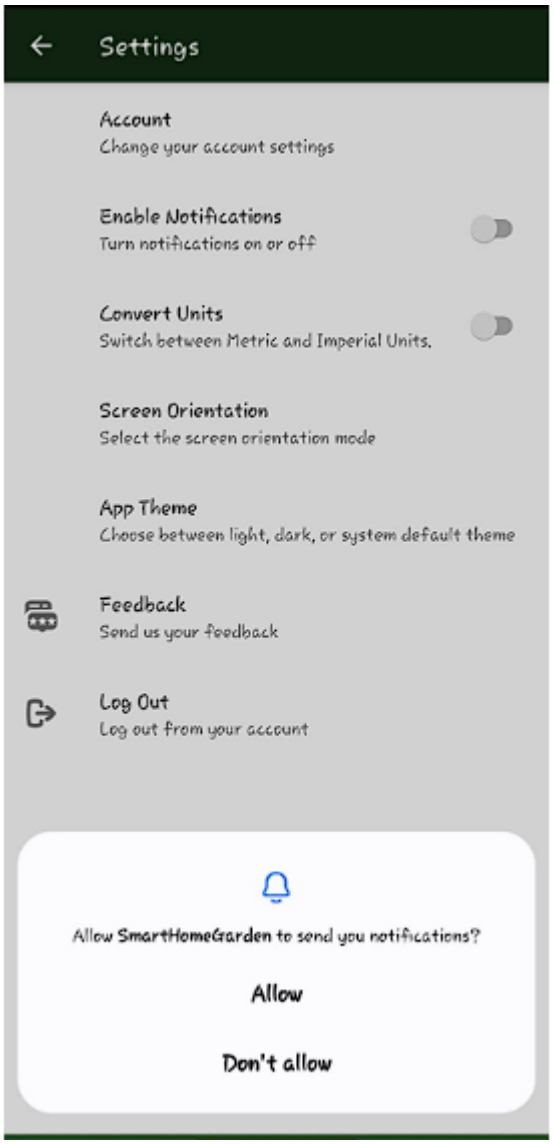
This snackbar appears when the user grants or denies location permission.

C. Toast



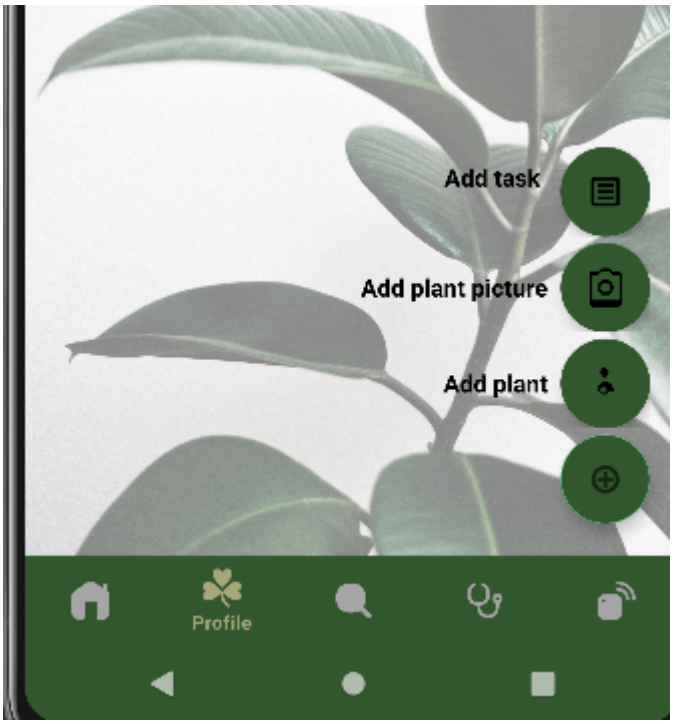
The toast is appearing when the user clicks on the checkbox on the task cardview.

D. Notification, handle runtime for API 33 and higher



When enabling notification the runtime permission is being asked.

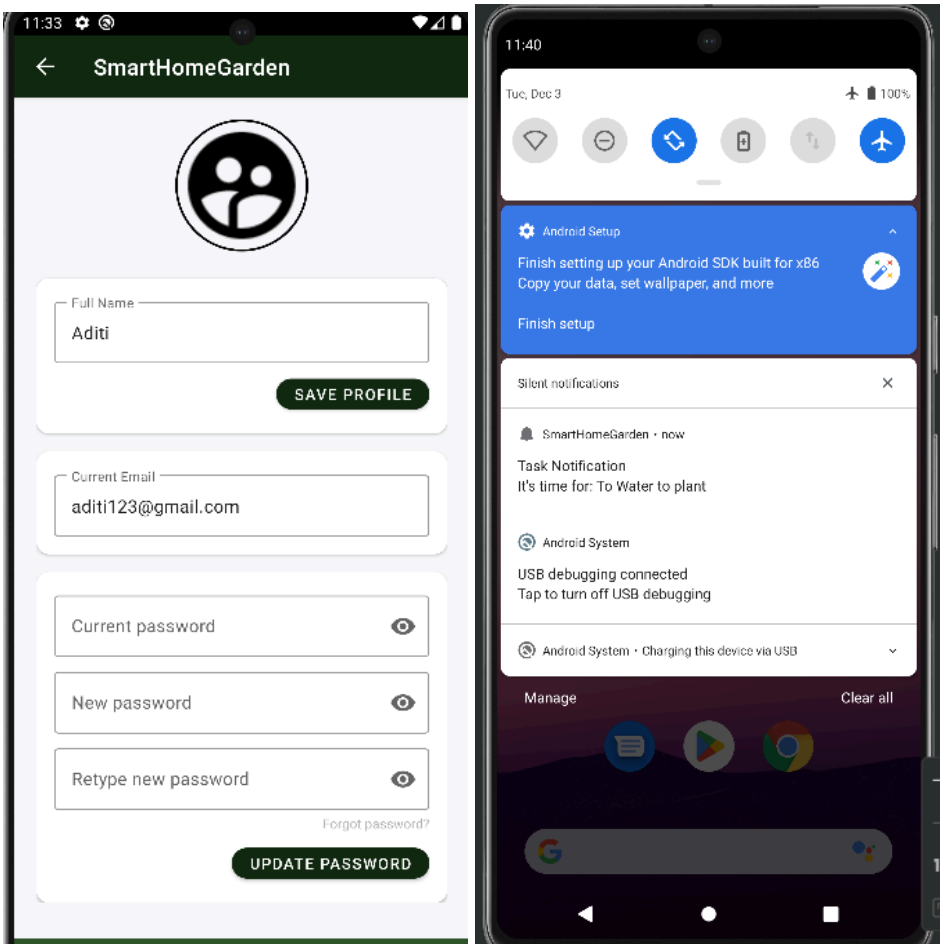
fab button and functionality



Here is the fab button we are using it has three functionality

1. Add Task - the user can add tasks for the plant-related work and it works as a reminder.
2. Add Plant Picture—This option allows the user to take pictures of the plant, which can then be used to diagnose it.
3. Add Plant - It allows the user to add whichever plant users want.

Offline Features



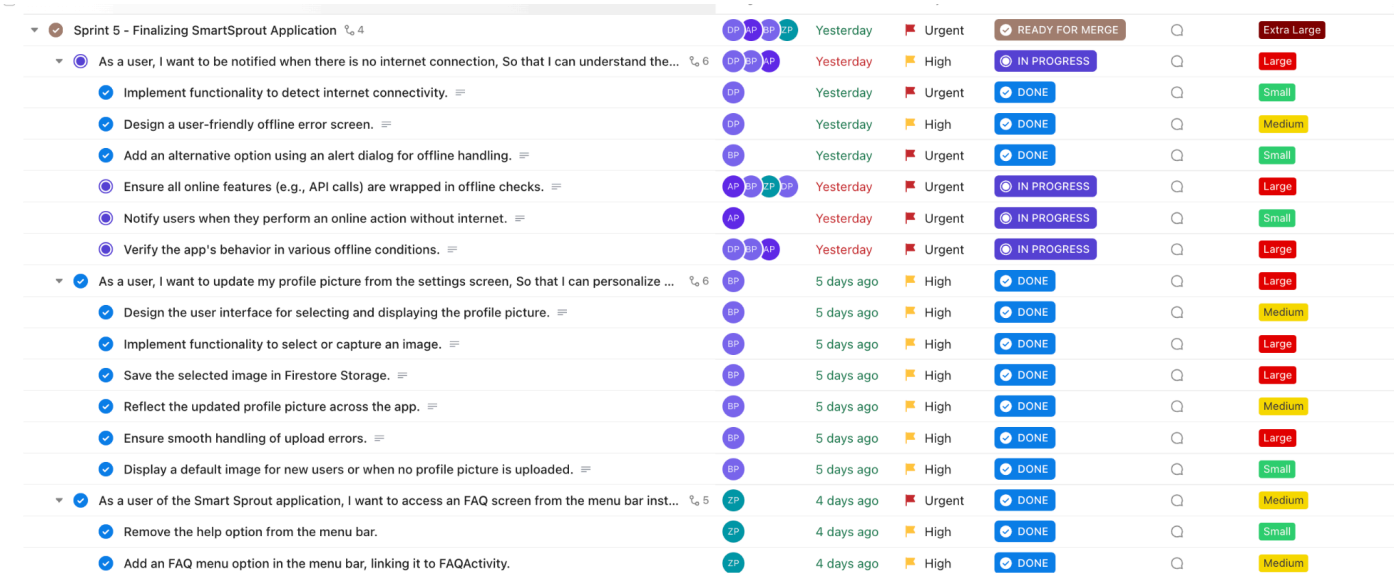
Users can change the user name of the user in airplane mode and it will be modified in the database when online again.
Also, the task reminder still works in the online mode. It will still notify users at the set time.

Exceptions Handling

The exception handling in code ensures the app remains robust and user-friendly during network operations. In `searchAndFetchPlantDetail`, the `try-catch` block captures potential `IOException` errors from network calls, preventing crashes during connectivity issues. Validation checks confirm the response's success, ensure the body is not null, and verify that search results contain valid data. If any errors occur, detailed logs provide insights into the failure, and the `plantDetail` LiveData is updated with `null` to gracefully inform the UI of issues.

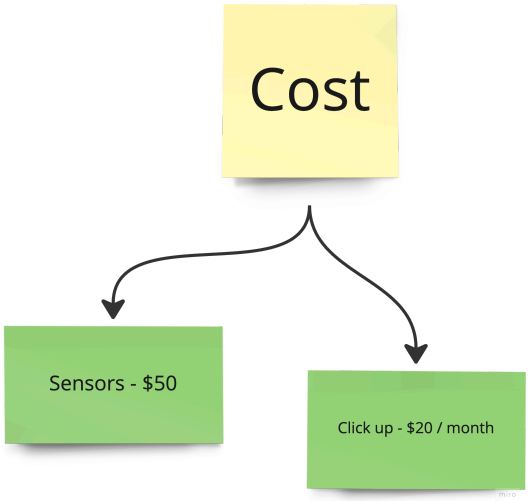
Similarly, in `fetchPlantDetail`, network calls are wrapped in `try-catch` blocks to handle `IOException`. Response validation ensures the API returns valid data before processing. Errors like invalid access tokens or failed API requests are logged, helping developers identify issues. Additionally, `plantDetail` is updated with `null` on failure, allowing the app to degrade gracefully while maintaining a smooth user experience. This approach ensures reliability, scalability, and ease of debugging.

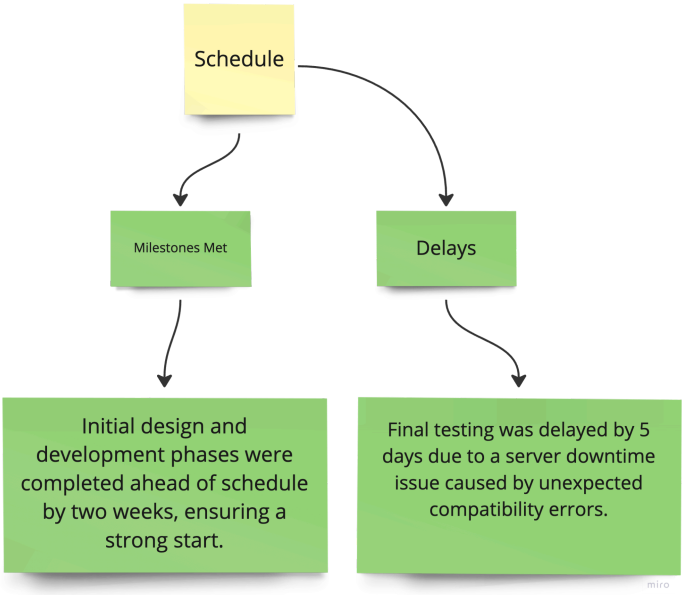
Scrum dashboard

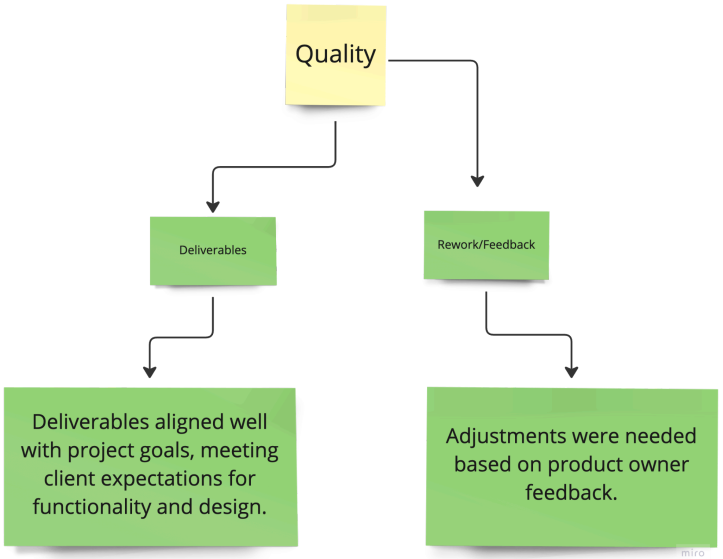


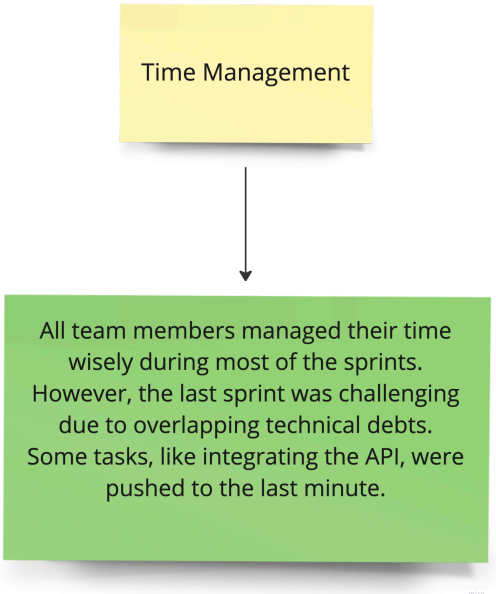
Sprint 5 - Finalizing SmartSprout Application 4	DP AP BP ZP	Yesterday	Urgent	READY FOR MERGE		Extra Large
As a user, I want to be notified when there is no internet connection, So that I can understand the... 6	DP BP AP	Yesterday	High	IN PROGRESS		Large
Implement functionality to detect internet connectivity.	DP	Yesterday	Urgent	DONE		Small
Design a user-friendly offline error screen.	DP	Yesterday	High	DONE		Medium
Add an alternative option using an alert dialog for offline handling.	BP	Yesterday	Urgent	DONE		Small
Ensure all online features (e.g., API calls) are wrapped in offline checks.	AP BP ZP DP	Yesterday	Urgent	IN PROGRESS		Large
Notify users when they perform an online action without internet.	AP	Yesterday	Urgent	IN PROGRESS		Small
Verify the app's behavior in various offline conditions.	DP BP AP	Yesterday	Urgent	IN PROGRESS		Large
As a user, I want to update my profile picture from the settings screen, So that I can personalize ... 6	BP	5 days ago	High	DONE		Large
Design the user interface for selecting and displaying the profile picture.	BP	5 days ago	High	DONE		Medium
Implement functionality to select or capture an image.	BP	5 days ago	High	DONE		Large
Save the selected image in Firestore Storage.	BP	5 days ago	High	DONE		Large
Reflect the updated profile picture across the app.	BP	5 days ago	High	DONE		Medium
Ensure smooth handling of upload errors.	BP	5 days ago	High	DONE		Large
Display a default image for new users or when no profile picture is uploaded.	BP	5 days ago	High	DONE		Small
As a user of the Smart Sprout application, I want to access an FAQ screen from the menu bar inst... 5	ZP	4 days ago	Urgent	DONE		Medium
Remove the help option from the menu bar.	ZP	4 days ago	High	DONE		Small
Add an FAQ menu option in the menu bar, linking it to FAQActivity.	ZP	4 days ago	High	DONE		Medium

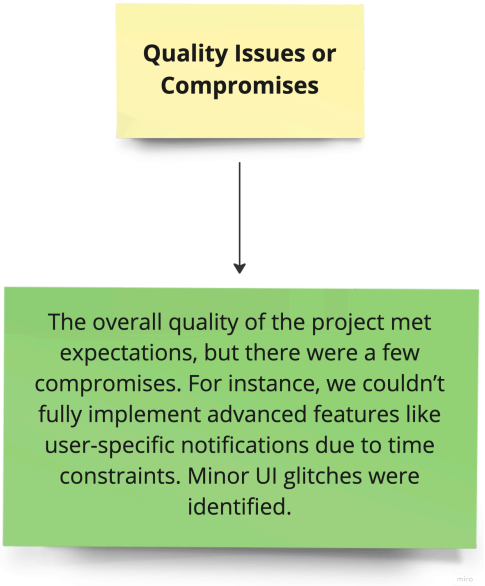
Post-Mortem

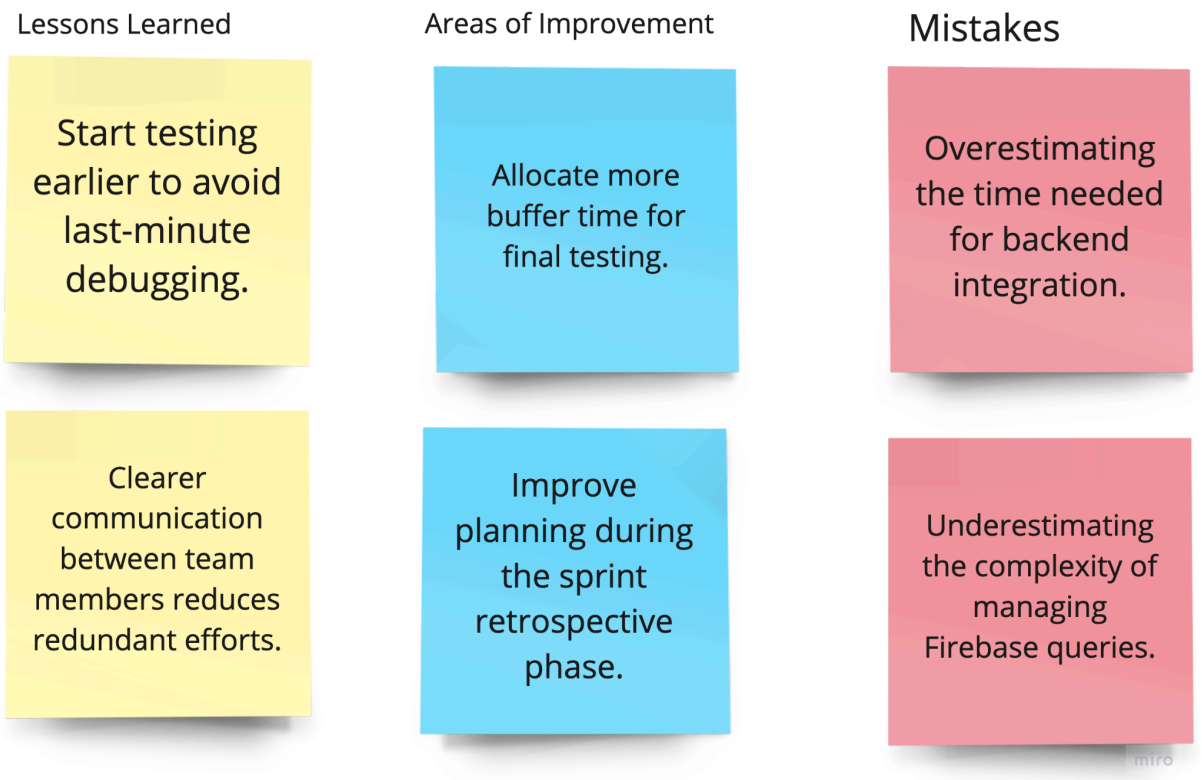

Cost
Sensors - $50
Click up - $20 / month


Schedule
Milestones Met
Delays
Initial design and development phases were completed ahead of schedule by two weeks, ensuring a strong start.
Final testing was delayed by 5 days due to a server downtime issue caused by unexpected compatibility errors.


Quality
Deliverables
Rework/Feedback
Deliverables aligned well with project goals, meeting client expectations for functionality and design.
Adjustments were needed based on product owner feedback.


Time Management
All team members managed their time wisely during most of the sprints. However, the last sprint was challenging due to overlapping technical debts. Some tasks, like integrating the API, were pushed to the last minute.


Quality Issues or Compromises
The overall quality of the project met expectations, but there were a few compromises. For instance, we couldn't fully implement advanced features like user-specific notifications due to time constraints. Minor UI glitches were identified.


Lessons Learned
Start testing earlier to avoid last-minute debugging.
Clearer communication between team members reduces redundant efforts.
Areas of Improvement
Allocate more buffer time for final testing.
Improve planning during the sprint retrospective phase.
Mistakes
Overestimating the time needed for backend integration.
Underestimating the complexity of managing Firebase queries.

How did you address technical debt in your project

Improving Test Organization:

- **Task**: Moving all test cases from the `AndroidTest` package to the `test` package to streamline unit testing.
- **Action Taken**: Restructured the testing framework to separate unit tests from integration tests, improving test execution speed, and aligning with standard development practices.

Error Handling:

- **Task**: Ensuring exceptions are properly handled to prevent application crashes.
- **Action Taken**: Audited the codebase to add robust exception handling mechanisms, ensuring the application gracefully recovers from errors.

Document two areas of refactoring and why you did it!. Real examples from your code and not just statements. Copy the code from your project and comment on it.

1. Code for database:

```
// Register the user

   authViewModel.registerUser(email, password).observe(this,
authResult -> {

       if (authResult != null && authResult.getUser() != null) {

           String uid = authResult.getUser().getUid();

           User user = new User(name, phone, email, password,
confirmPassword);

           authViewModel.saveUserDataToFirestore(uid,
user).observe(this, success -> {

               if (success) {

                   Toast.makeText(this,
getString(R.string.registration), Toast.LENGTH_SHORT).show();

                   startActivity(new Intent(RegistrationActivity.this,
MainActivity.class));

                   finish();

               } else {

                   Toast.makeText(this,
getString(R.string.faileddata), Toast.LENGTH_SHORT).show();

               }

           });

       } else {

           Toast.makeText(this,
getString(R.string.registration_failed), Toast.LENGTH_SHORT).show();

       }

   });

}
```

Updated:

```
authViewModel.registerUser(email, password).observe(this, authResult
-> {

       if (authResult != null && authResult.getUser() != null) {

           String uid = authResult.getUser().getUid();

           try {

               String encryptedPassword =
EncryptionUtils.encrypt(password);

               String encryptedConfirmPassword =
EncryptionUtils.encrypt(confirmPassword);

               User user = new User(name, phone, email,
encryptedPassword, encryptedConfirmPassword);

               authViewModel.saveUserDataToFirestore(uid,
user).observe(this, success -> {

                   if (success) {

                       Toast.makeText(this,
getString(R.string.registration), Toast.LENGTH_SHORT).show();

                       startActivity(new
Intent(RegistrationActivity.this, MainActivity.class));

                       finish();

                   } else {

                       Toast.makeText(this,
getString(R.string.faileddata), Toast.LENGTH_SHORT).show();

                   }

               });

           } catch (Exception e) {

               Toast.makeText(this,
getString(R.string.encryption_failed), Toast.LENGTH_SHORT).show();

               Log.e("RegistrationActivity", "Encryption failed", e);

           }

       } else {

           Toast.makeText(this,
getString(R.string.registration_failed), Toast.LENGTH_SHORT).show();

       }
```

```
    });

}
```

Here I modified the code because the password was not encrypted.

2. For database:

```
if (currentUser != null) {

    String userId = currentUser.getUid();

    databaseReference =
FirebaseDatabase.getInstance().getReference("Users").child(userId).chi
ld("notifications");

    loadNotifications();

}
```

Here I changed to User -> notifications first it was just notifications.

Describe how DevOps would have helped your project

What coding standards did you use, and how did you use them

List at least one security vulnerability in your code, copy the code and comment on it. How to address in the future.

DevOps Benefits:

- Automates testing, building, and deployment (CI/CD).
- Improves collaboration and monitoring for faster delivery and better quality.
  Example: Using GitHub Actions to run automated unit tests for your app when code is pushed.

Coding Standards:

- Follow best practices like meaningful variable names, consistent formatting, and error handling.
  Example:

```
private void loginUser() {  1 usage
    String email = emailInput.getText().toString().trim();
    String password = passwordInput.getText().toString().trim();
    if (email.isEmpty() || password.isEmpty()) {
        Toast.makeText( context: this, getString(R.string.email_password), Toast.LENGTH_SHORT).show();
    } else if (password.length() > 10) {
        passwordInput.setError(getString(R.string.exceed));
    } else if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
        emailInput.setError(getString(R.string.invalidemail));
    } else {

        authViewModel.loginUser(email, password).observe( owner: this, this::handleLoginResult);
    }
}
```

Security Vulnerability:

- **Issue**: Hardcoding sensitive data like API keys in code.
  Example:

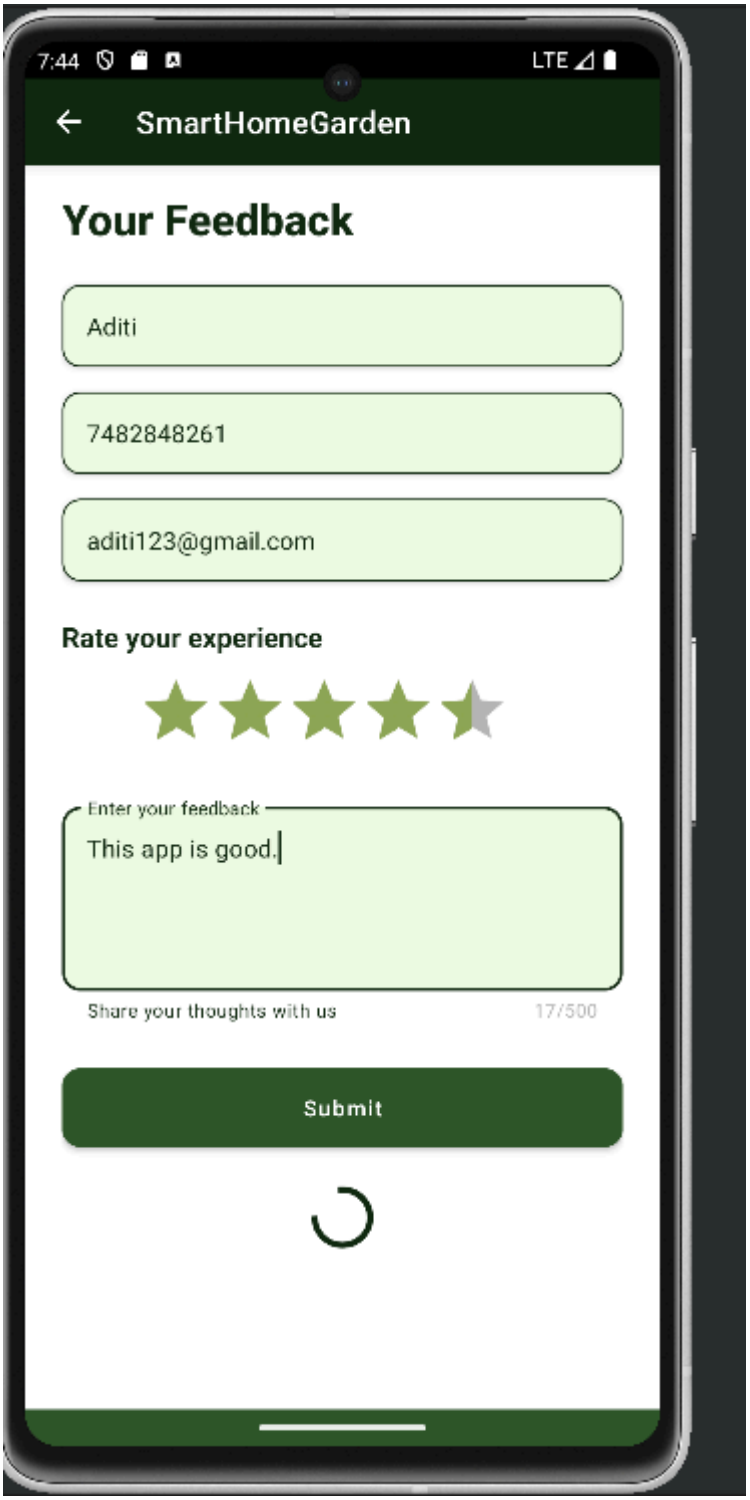
```
<string name="webclient">204065205787-1g7a97om3768e9lp66bopohr43mia4pr.apps.googleusercontent.com</string>
```

- **Fix**: Use secure storage or server-side authentication.
  Example:

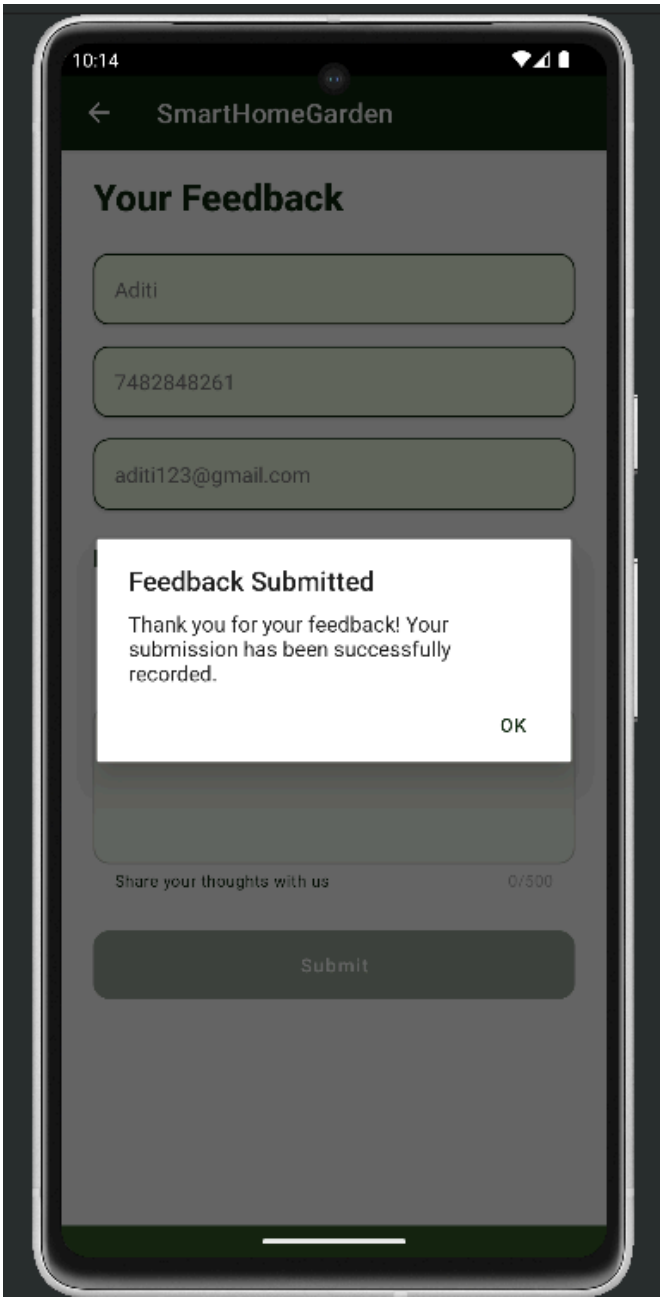
```
public void searchAndFetchPlantDetail(String query) {  1 usage
    executorService.execute(() -> {
        try {
            String searchUrl = SEARCH_URL + "?q=" + query + "&limit=1";
            Request searchRequest = new Request.Builder()
                    .url(searchUrl)
                    .addHeader( name: "Content-Type",  value: "application/json")
                    .addHeader( name: "Api-Key", API_KEY)
                    .build();
```

Avoid exposing sensitive data in your APK by using **Proguard** for obfuscation.

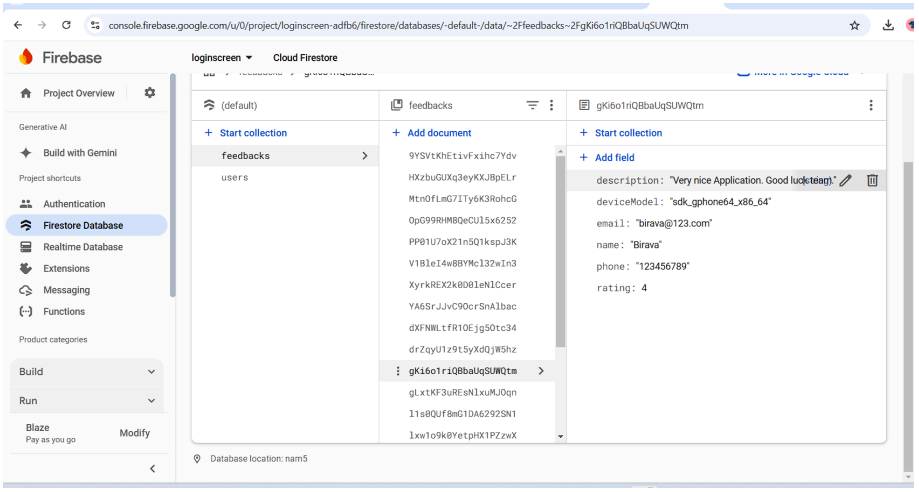
Feedback

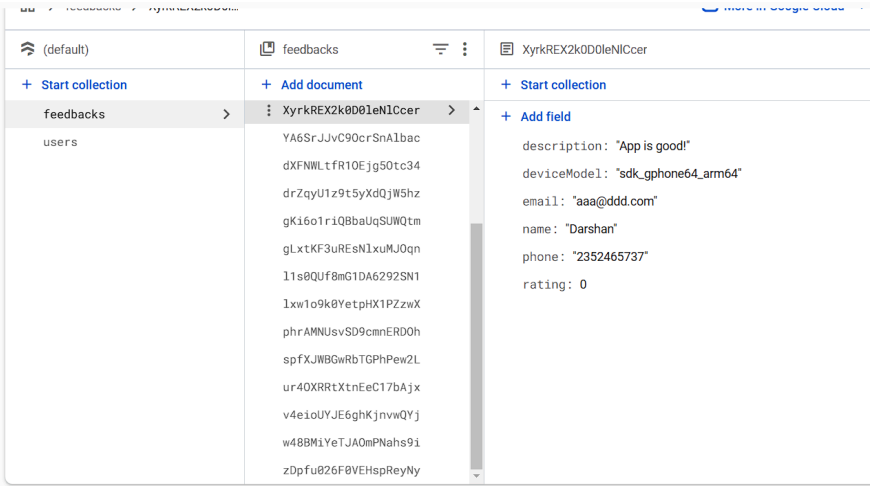


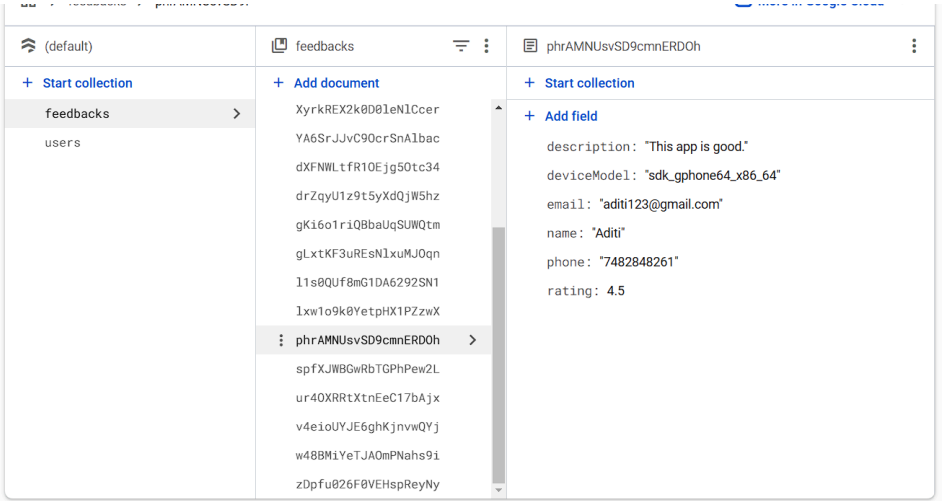
Add into pdf file screenshot showing the AlertDialog once the form is submitted successfully on the feedback screen.


10:14
SmartHomeGarden
Your Feedback
Aditi
7482848261
aditi123@gmail.com
Feedback Submitted
Thank you for your feedback! Your submission has been successfully recorded.
OK
Share your thoughts with us
0/500
Submit

Database:





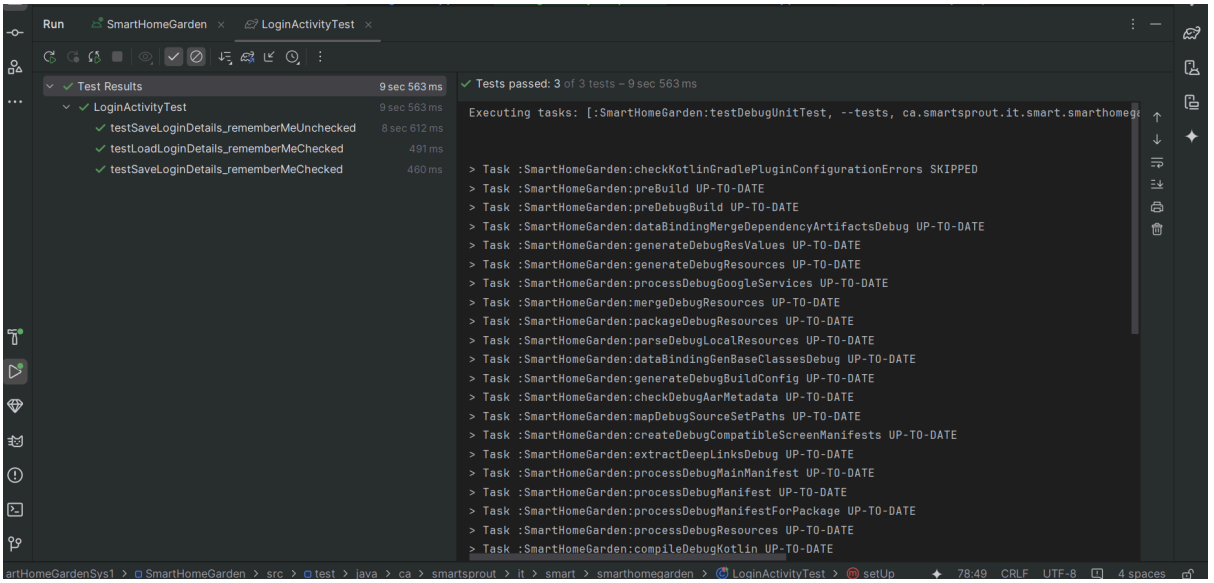


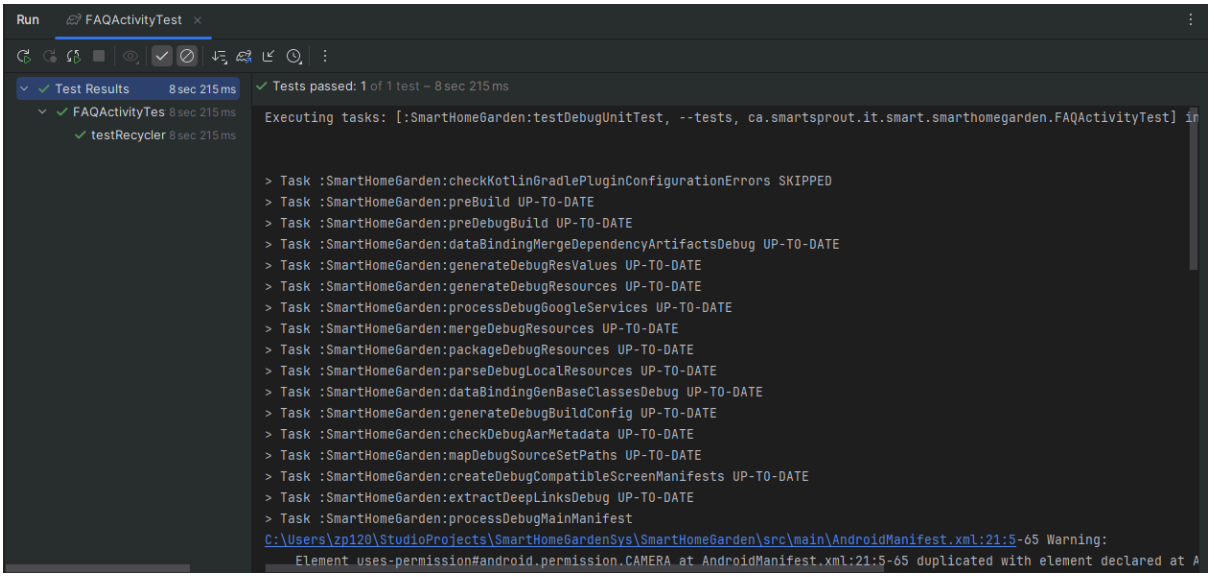
SMART SPROUT

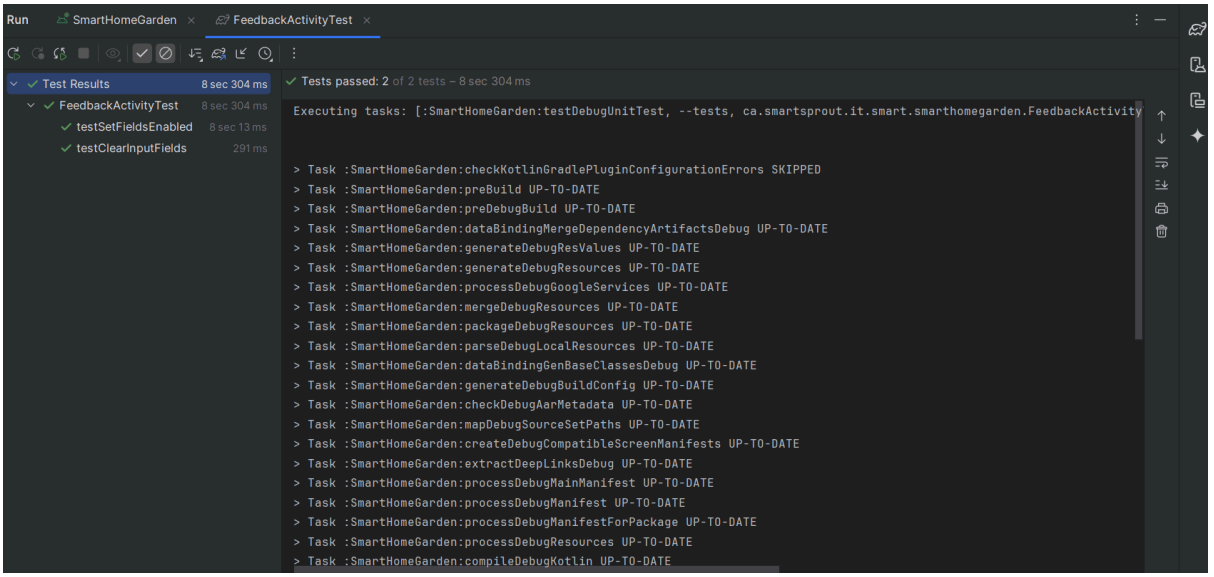
Test Cases

Strategy:

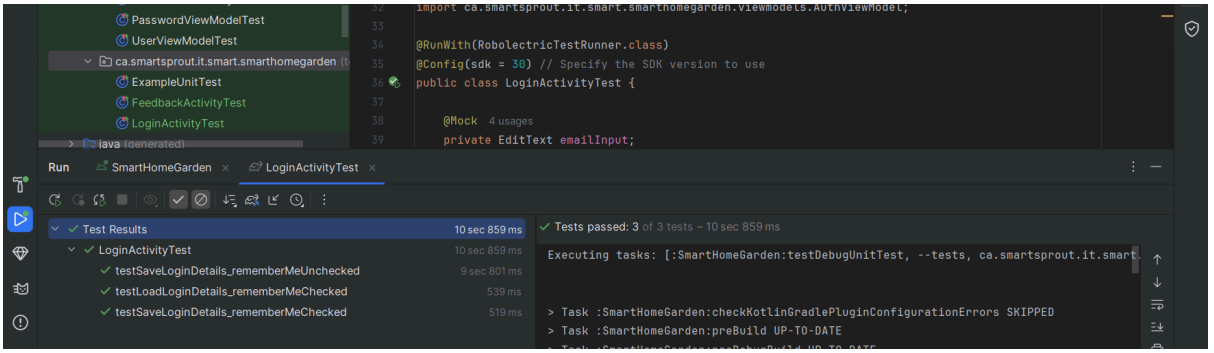
Screenshots showing 3 classes junit test cases are passing.



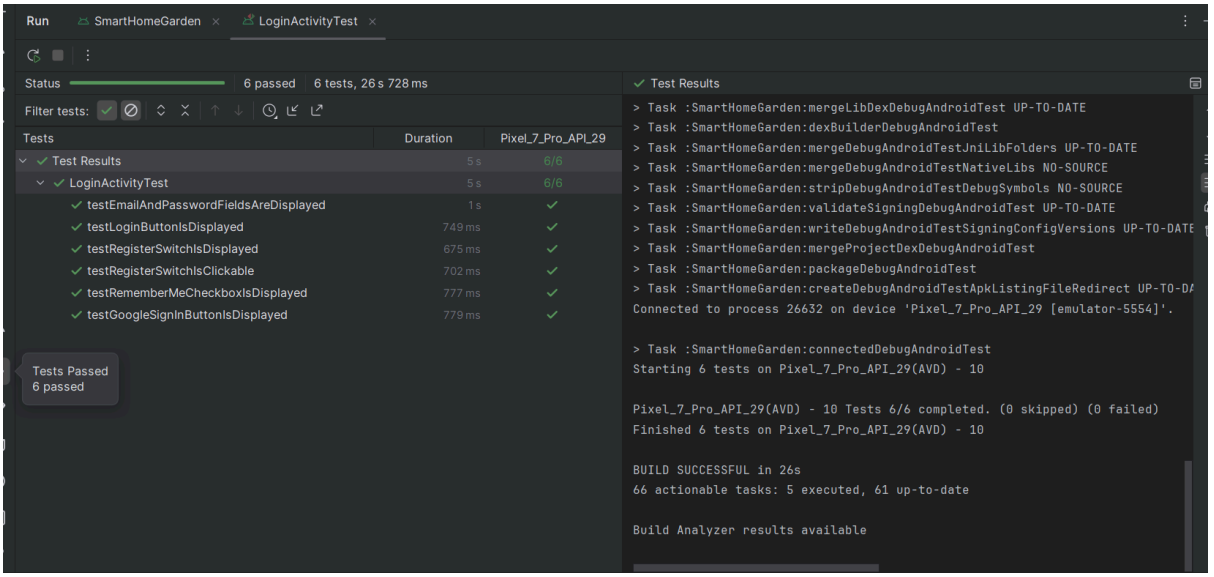




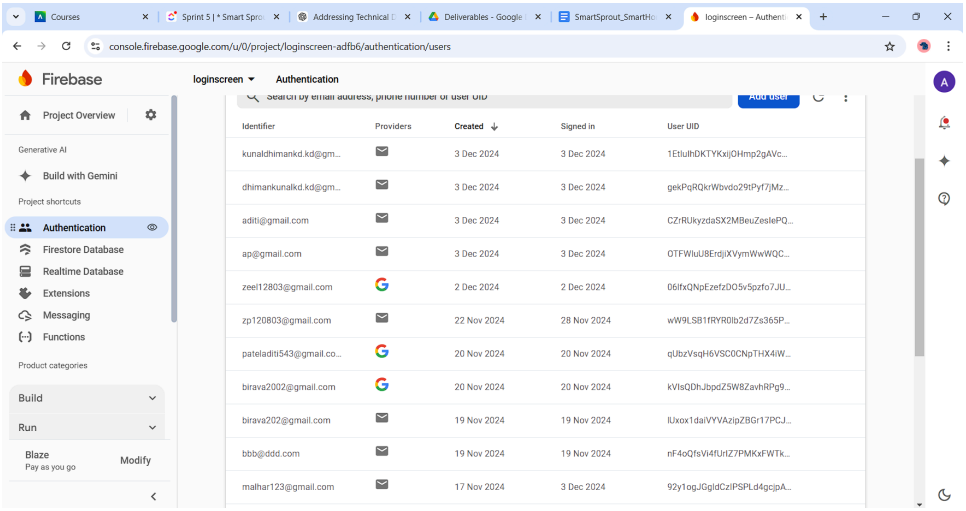
Screenshot showing Robleteric test cases are passing

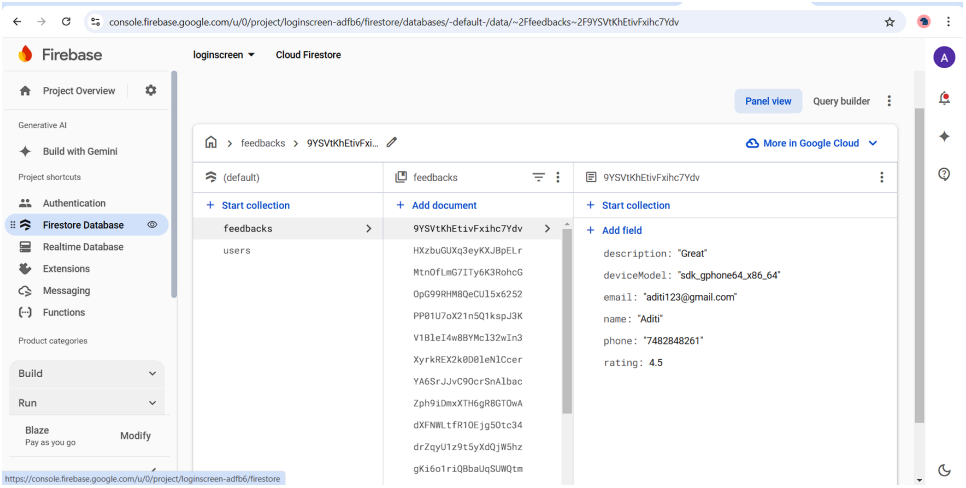


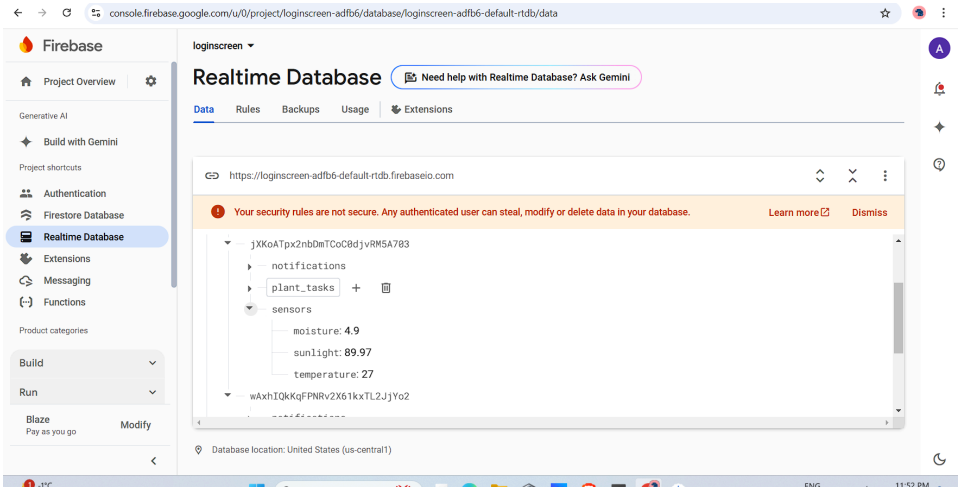
Screenshot showing Espresso test cases are passing:



Screenshots showing the data in your DB, sensors data, login data (including Google login), and Feedback Screen data







SMART SPROUT