

CENG 322

Deliverable 3





Team Name: Sleep Deprived
Project Name: Eevee the Pet Companion
Team Number: Group 5

Chloe Chai - N01447321
Ubay Abdulaziz - N01437353
John Aquino - N01303112
Jennifer Nguyen - N01435464

Table of Contents

Signatures of Participants	3
Project Summary	3
GitHub Repo Link	4
Account Creation in Database	4
Sprint Goals	4
Sprint Retrospective	5
Sprint Dashboard	5
Gantt Chart	6
System Context Diagram	7
Progress	8
Customer Reviews in Database	9
Runtime Permission	9
Design Patterns and Examples	9
Design Principles and Examples	11
Daily Stand-ups	15

Signatures of Participants

Name	ID	Signature	Effort
Wei Wen Chai	N01447321		100%
Jennifer Nguyen	N01435464		100%
John Aquino	N01303112		100%
Ubay Abdulaziz	N01437353		100%

Project Summary

Evee the Pet Companion is a pet monitoring device that allows users to remotely interact with their pets. The project plan includes designing the Android mobile app, integrating it with the physical device, and conducting thorough testing. The device will feature a high-quality camera for live streaming, obstacle avoidance, a treat dispenser, and autonomous navigation. The aim is to enhance convenience, safety, and the bond between pets and their owners by providing easy monitoring and control options.

GitHub Repo Link

<https://github.com/WeiWenChai7321/EveeThePetCompanion3>

WeiWenChai7321 / EveeThePetCompanion3

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

General

Access

Collaborators

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Codespaces

Pages

Security

Code security and analysis

Deploy keys

Secrets and variables

Integrations

GitHub Apps

Email notifications

Who has access

PRIVATE REPOSITORY

Only those with access to this repository can view it.

Manage

DIRECT ACCESS

4 have access to this repository. 4 collaborators.

Manage access

Add people

Select all

Type

Find a collaborator...

Collaborator	Type	Remove
Hak11 haki11 • Collaborator		Remove
JenniferNguyen5464 Collaborator		Remove
JohnAquino3112 Collaborator		Remove
UbayAbdulaziz7353 Collaborator		Remove

Account Creation in Database

Firebase

Project Overview

Project shortcuts

Authentication

Firestore Database

Product categories

Build

Release and monitor

Analytics

Engage

All products

Evee The Pet Companion

Cloud Firestore

Data Rules Indexes Usage Extensions

Panel view Query builder

More in Google Cloud

Collection	Document ID	Fields
users	imz1JR4rZwcvfK8H4d7QrE1pz911	email: 'aaa@bbb.com', password: 'Admin101!'
users	p2sJcKcUjCXre260GtCm	

Sprint Goals

Sprint Retrospective

Continue ➡
What helped us move forward?

- Constantly pushing updated to code to Github
- Also continue pulling code
- Helping each other when we're stuck on our tasks
- Having meetings to see where we are in the project

Stop ⛔
What held us back?

- Stop removing code from another team member before checking with them
- Last minute attempts in finishing our tasks

Who missed meetings??
Who disappeared?

Happy to say that everyone was present for all meetings

Start 🏁
What should we do next?

- Focusing on the Remember Me function
- Making some simulated data for some of the functionality

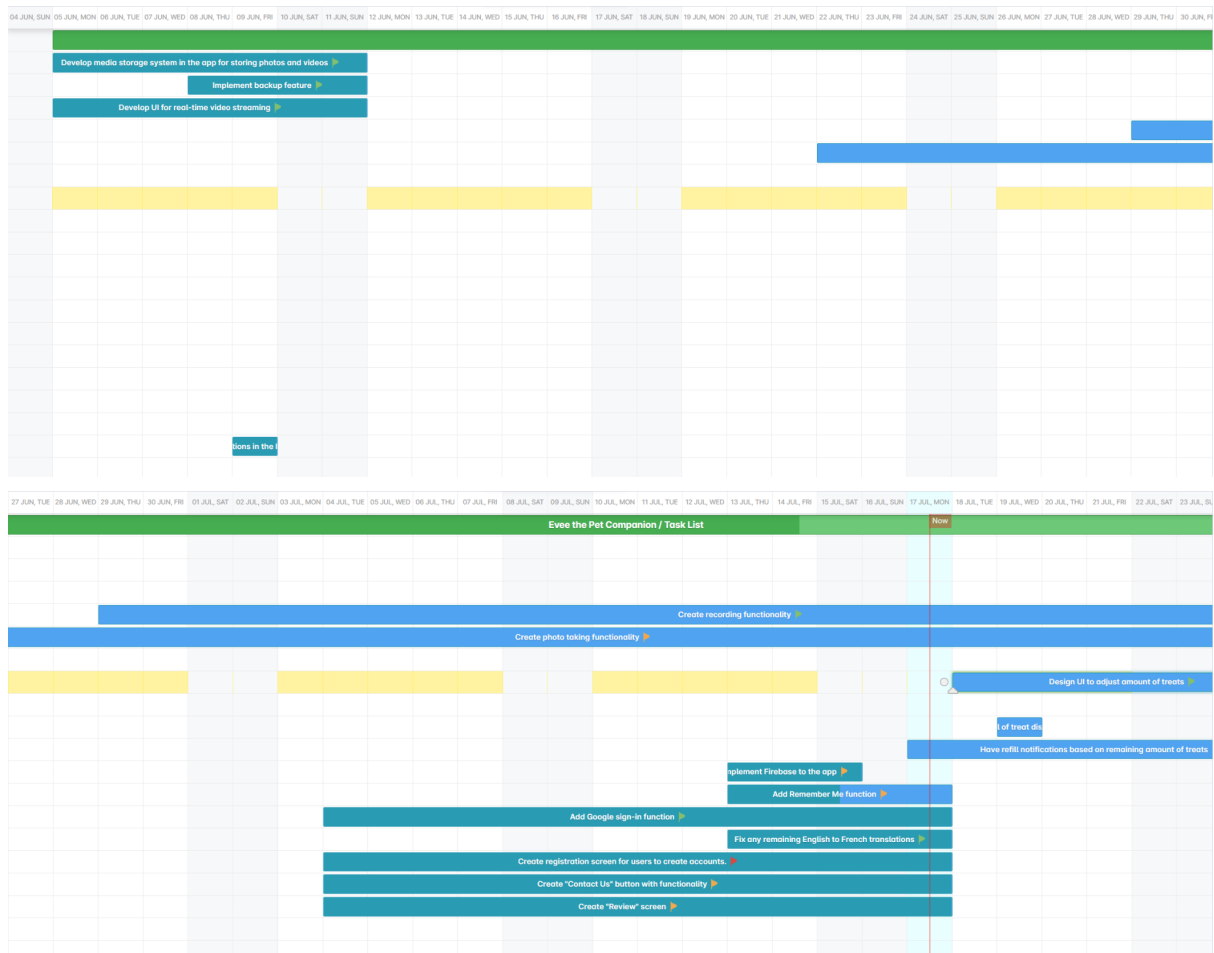
Sprint Dashboard

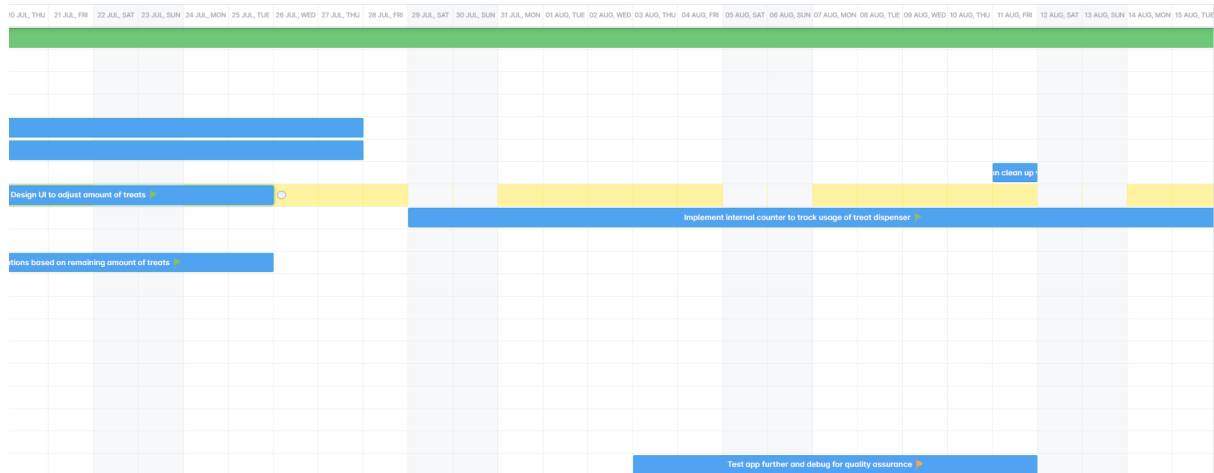
▼ Sprint 3

✓ Enhance the Stream Fragment UI to include an interactive treat button.	Treat Dispensing	Jun 22	Jun 22	S	High	Chloe
✓ Implement the logic to trigger treat dispensing actions.	Treat Dispensing	Jun 18	Jun 24	M	Medium	Chloe
✓ Display a toast message to provide visual feedback on treat dispensing.	Treat Dispensing	Jun 23	Jun 25	M	Low	Chloe
✓ Test and verify the treat dispensing functionality.	Treat Dispensing	Jun 23	Jun 25	M	Low	Chloe
✓ Handle error cases and provide appropriate error messages.	Treat Dispensing	Jun 26	Jun 26	S	Low	Chloe
✓ Implement the logic to add new reminders and display them in the UI.	Reminder Control	Jun 30	Jun 30	S	Low	John
✓ Add UI elements to the Dashboard Fragment for adding and deleting reminders.	Reminder Control	Jun 27	Today	S	Low	John
✓ Allow users to delete unwanted reminders from the list.	Reminder Control	Jul 1	Jul 1	S	Low	John
✓ Update the database with the changes in the reminders list.	Reminder Control	Jul 1	Jul 14	M	Medium	John
✓ Test and verify the reminder control functionality.	Reminder Control	Aug 10	Aug 10	S	Low	John
✓ Create a "Contact Us" button in the Help Fragment UI.	Contact Support	Jun 23	Jun 30	M	Low	Jennifer
✓ Implement the logic to open the default email client with a pre-filled template.	Contact Support	Jun 25	Jun 25	S	Low	Jennifer
✓ Pre-fill the email template with appropriate recipient and subject.	Contact Support	Jul 12	Aug 1	L	High	Jennifer
✓ Test and verify the "Contact Us" functionality.	Contact Support	Aug 11	Aug 11	S	Low	Jennifer
✓ Handle error cases and provide appropriate error messages.	Contact Support	Aug 1	Aug 1	S	Low	Jennifer
✓ Add an editable email field to the Settings Fragment UI.	Update Email Address	Aug 15	Jul 29	M	Low	Jennifer
✓ Implement the logic to update the email address in the database.	Update Email Address	Wednesday	Wednesday	S	Low	Jennifer
✓ Handle case where user logged in with Google	Update Email Address	Jul 6	Jul 6	S	Low	Ubay
✓ Test and verify the email address update functionality.	Update Email Address	Jun 16	Jun 23	M	High	Chloe
✓ Provide visual feedback to the user on successful email address update.	Update Email Address	Jun 18	Jun 23	L	Low	Chloe

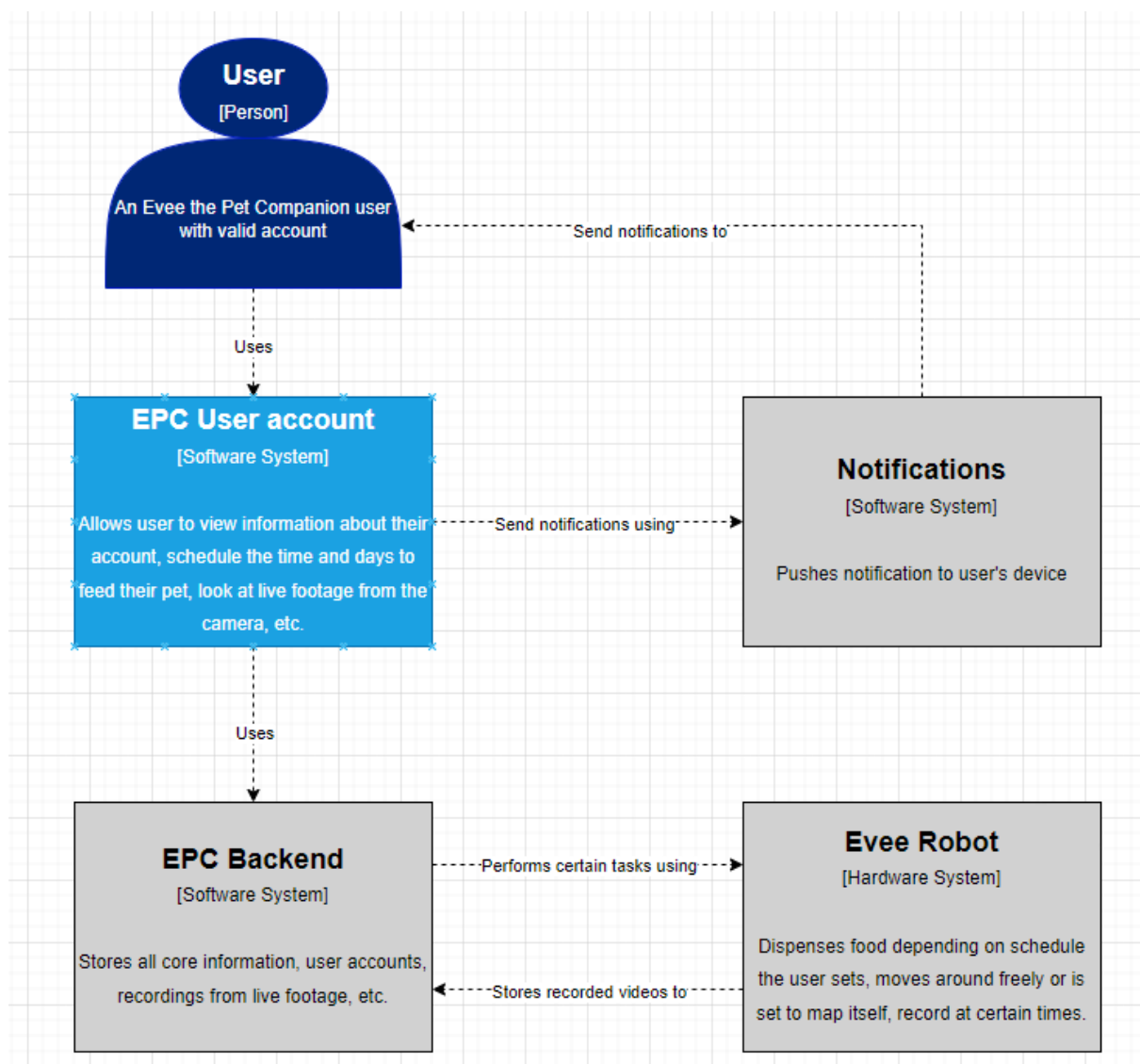
▼ Sprint 4							
Update the Stream Fragment to handle storing pictures and videos in the databas	Storing Pictures and Videos	Wednesday	Aug 11	S	High	Chloe	
Implement the logic to save captured pictures and videos in the appropriate datal	Storing Pictures and Videos	Wednesday	Aug 11	M	Medium	Chloe	
Store metadata such as timestamps and relevant information along with the medi	Storing Pictures and Videos	Wednesday	Aug 11	M	Low	Chloe	
Test and verify the storage functionality for pictures and videos.	Storing Pictures and Videos	Wednesday	Aug 11	M	Low	Chloe	
Handle error cases and provide appropriate error messages.	Storing Pictures and Videos	Wednesday	Aug 11	S	Low	Chloe	
Enhance the Pet Profile Fragment UI to allow users to add entries to the calendar.	Calendar Feature	Wednesday	Aug 11	S	Low	John	
Implement the logic to save the entered calendar entries in the database.	Calendar Feature	Wednesday	Aug 11	S	Low	John	
Validate the entered entries and handle error cases.	Calendar Feature	Wednesday	Aug 11	S	Low	John	
Update the database with the changes in the reminders list.	Calendar Feature	Wednesday	Aug 11	M	Medium	John	
Test and verify the functionality of saving entries in the calendar.	Calendar Feature	Wednesday	Aug 11	S	Low	John	
Update the push notification switch in the Settings Fragment UI.	Enabling Push Notifications	Wednesday	Aug 11	M	Low	Jennifer	
Implement the logic to enable push notifications in the app when the switch is tur	Enabling Push Notifications	Wednesday	Aug 11	S	Low	Jennifer	
Integrate with a push notification service provider (e.g., Firebase Cloud Messaging	Enabling Push Notifications	Wednesday	Aug 11	L	High	Jennifer	
Test and verify the functionality of push notifications in the app.	Enabling Push Notifications	Wednesday	Aug 11	S	Low	Jennifer	
Handle error cases and provide appropriate error messages.	Enabling Push Notifications	Wednesday	Aug 11	S	Low	Jennifer	
Add a "Delete Account" button to the Settings Fragment UI.	Account Deletion	Wednesday	Aug 11	M	Low	Jennifer	
Implement the logic to handle account deletion when the button is clicked.	Account Deletion	Wednesday	Aug 11	S	Low	Jennifer	
Prompt the user for confirmation before proceeding with the account deletion.	Account Deletion	Wednesday	Aug 11	S	Low	Ubay	
Remove the user's data from the database upon successful account deletion.	Account Deletion	Wednesday	Aug 11	M	High	Chloe	
Provide visual feedback to the user on successful account deletion.	Account Deletion	Wednesday	Aug 11	L	Low	Chloe	

Gantt Chart





System Context Diagram



Progress

Firebase Database was linked to the login and registration screens, allowing for the integration of user data storage and secure authentication.

CRUD operations were enabled on the database, enabling the app to create, read, update, and delete records as needed.

The LoginActivity was enhanced to support Google sign-in, allowing users to authenticate using their Google credentials. Additionally, a "Remember Me" feature was added to enable automatic login on subsequent app launches.

The PetProfileFragment was extended to allow users to edit their pet's information, such as name, breed, and age. Users could also add entries to the calendar for important pet-related events and appointments.

In the StreamFragment, the treat button became interactive, triggering actions to dispense treats to the pet. A toast message was displayed to provide visual feedback to the user.

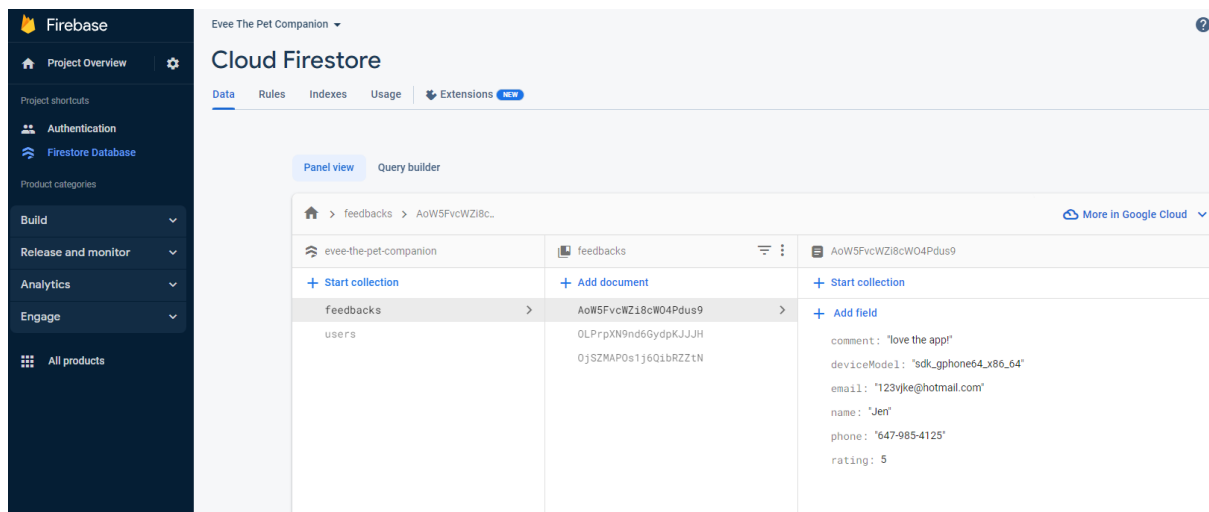
The DashboardFragment was enhanced to enable users to add and delete reminders, with the changes being saved in the database. This provided users with better control over their reminders.

The HelpFragment integrated a "Contact Us" feature, allowing users to tap a button and open an email client with a pre-filled template addressed to the support team. This streamlined communication between users and the app's support team.

In the SettingsFragment, users could update their email addresses, with the changes being reflected in the database. This functionality was available for users who did not sign in with their Google accounts.

The FeedbackFragment was expanded to include a feedback submission feature, allowing users to provide ratings, enter their name, phone number, email address, and leave comments. The submitted feedback was stored in the database for review and analysis by the development team.

Customer Reviews in Database



Runtime Permission

The HighlightsFragment includes a "Download All" button, which prompts the user for runtime permission to write to the SD card and allows them to save all the videos and photos from the page to their SD card.

Design Patterns and Examples

To demonstrate the use of Observer and Builder patterns, we will be looking at code snippets from the SettingsFragment and MainActivity.

Observer Pattern

The Observer pattern is used to listen for changes in the user's email address stored in Firestore. The onStart method sets up a Firestore snapshot listener using the addSnapshotListener method. Whenever the email address document is modified in Firestore, the onEvent callback is triggered, and the email address is updated in the UI.

This pattern allows the SettingsFragment to observe changes in the user's email address and automatically update the UI without requiring manual intervention. The fragment acts as an observer, and the Firestore snapshot listener acts as the subject. When a change occurs, the observer (fragment) is notified and can react accordingly.

By using the Observer pattern, the SettingsFragment remains decoupled from the specific implementation details of Firestore and can easily handle updates to the email address without tightly coupling with the Firestore API.

```
public class SettingsFragment extends Fragment {  
    // ...  
}
```

```

@Override
public void onStart() {
    super.onStart();

    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();

    if (user != null) {
        emailListener = firestore.collection("users")
            .document(user.getId())
            .addSnapshotListener(new EventListener<DocumentSnapshot>() {
                @Override
                public void onEvent(@Nullable DocumentSnapshot
documentSnapshot, @Nullable FirebaseFirestoreException e) {
                    if (e != null) {
                        // Handle error
                        return;
                    }

                    if (documentSnapshot != null && documentSnapshot.exists()) {
                        String email = documentSnapshot.getString("email");
                        if (email != null) {
                            emailEditText.setText(email);
                        }

                        // Disable editing
                        emailEditText.setEnabled(false);
                    }
                }
            });
    }
}

@Override
public void onStop() {
    super.onStop();

    // Unsubscribe from Firestore listener
    if (emailListener != null) {
        emailListener.remove();
    }
}

// ...
}

```

Builder Pattern

The Builder design pattern is used here to simplify the construction of a complex object, which is the notification. It allows you to set different properties of the notification step by step without exposing the underlying implementation details.

By using the builder pattern, the code becomes more readable and maintainable, and it provides a clear and intuitive way to create the desired notification object with all the necessary configurations.

```
public void showReviewNotification() {
    // ...

    // Build the notification
    NotificationCompat.Builder notificationBuilder = new
    NotificationCompat.Builder(this, CHANNEL_ID)
        .setSmallIcon(R.drawable.notification_icon)
        .setContentTitle("Enjoying our app?")
        .setContentText("Leave us a review!")
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)
        .setContentIntent(okayPendingIntent)
        .addAction(0, "No thanks", noThanksPendingIntent)
        .addAction(0, "Okay", okayPendingIntent)
        .setAutoCancel(true);

    // ...
}
```

Design Principles and Examples

To demonstrate the use of Single Responsibility and Open-Closed principles, we will be looking at code snippets from the DashboardFragment and SettingsFragment.

Single Responsibility Principle (SRP)

Each method in the DashboardFragment class has a specific responsibility and performs a single task. For example, the addReminder method is responsible for adding a reminder, the removeReminderDelayed method handles the delayed removal of a reminder, and the saveReminders method is responsible for saving the reminders to SharedPreferences. This adherence to the SRP ensures that each method has a clear purpose and makes the code easier to understand, test, and maintain.

```
private void addReminder(String reminderText) {
    if (remindersLayout == null) {
        return;
    }

    CheckBox reminderCheckBox = new CheckBox(requireContext());
    reminderCheckBox.setText(reminderText);
    reminderCheckBox.setChecked(false);
}
```

```

        reminderCheckBox.setOnCheckedChangeListener((buttonView, isChecked) -> {
            if (isChecked) {
                removeReminderDelayed((CheckBox) buttonView);
            }
        });
        remindersLayout.addView(reminderCheckBox);
        reminderCheckboxes.add(reminderCheckBox);
        savedReminders.add(reminderText);
    }

    private void removeReminderDelayed(CheckBox checkBox) {
        new Handler().postDelayed(() -> {
            remindersLayout.removeView(checkBox);
            reminderCheckboxes.remove(checkBox);
            savedReminders.remove(checkBox.getText().toString());
        }, 5000);
    }

    private void addSavedReminders() {
        for (String reminder : savedReminders) {
            addReminder(reminder);
        }
    }

    private void saveReminders() {
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putStringSet("reminders", savedReminders);
        editor.apply();
    }

```

Open-Closed Principle (OCP)

The code snippet demonstrates the Open-Closed Principle by implementing the `onResume()` method. Here, the `SettingsFragment` class is open for extension but closed for modification.

The `onResume()` method is responsible for updating the UI elements based on stored preferences. It reads the stored preferences for orientation locking and push notifications, updates the switches accordingly, and attaches listeners to handle changes.

If new preferences need to be added in the future, the `onResume()` method can be extended by adding the necessary logic for handling those preferences, without modifying the existing code. This allows the class to be easily extended without directly modifying its implementation.

By adhering to the Open-Closed Principle, the code promotes maintainability and extensibility. It allows for the addition of new preferences or modifications to the existing ones without altering the core behavior of the `SettingsFragment` class. This leads to more flexible and reusable code.

```

public class SettingsFragment extends Fragment {
    // ...

    @Override
    public void onResume() {
        super.onResume();
        String storedEmail =
sharedPreferences.getString(getString(R.string.saved_email), "");

        // Read the stored preferences and update the switches accordingly
        boolean isOrientationLocked =
sharedPreferences.getBoolean(getString(R.string.lock_orientation), false);
        boolean isPushNotificationsEnabled =
sharedPreferences.getBoolean(getString(R.string.push_notifications), false);

        lockOrientationSwitch.setChecked(isOrientationLocked);
        pushNotificationSwitch.setChecked(isPushNotificationsEnabled);

        lockOrientationSwitch.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                // Save the orientation preference in SharedPreferences
                sharedPreferences.edit().putBoolean(getString(R.string.lock_orientation),
isChecked).apply();

                if (isChecked) {
                    // Lock screen orientation to portrait

getActivity().setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTR
AIT);
                    Toast.makeText(getActivity(),
R.string.screen_orientation_locked_to_portrait, Toast.LENGTH_SHORT).show();
                } else {
                    // Reset screen orientation to sensor

getActivity().setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_SENSEO
R);
                    Toast.makeText(getActivity(), R.string.screen_orientation_unlocked,
Toast.LENGTH_SHORT).show();
                }
            }
        });

        pushNotificationSwitch.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {

```

```

        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
            // Save the push notifications preference in SharedPreferences

sharedPreferences.edit().putBoolean(getString(R.string.push_notifications),
isChecked).apply();

            if (isChecked) {
                Toast.makeText(getActivity(), R.string.push_notifications_enabled,
Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(getActivity(), R.string.push_notifications_disabled,
Toast.LENGTH_SHORT).show();
            }
        }
    });
}

// ...
}

```

Daily Stand-ups

📅 Sunday, July 9th @ 10:00 AM

	Name	Priorities 🧐	Progress 😊	Problems 😞
1	Wei Wen (Chloe) Chai	Delegate tasks for the next deliverable	Distributed tasks to work on before next meeting	None
2	John Aquino	Work on "Remember Me" function from the login page	Have created a <u>savedInstance</u> method for the function but anything further than that there is no progress.	The account will still stay logged in even after disabling Remember Me on the login page. Keeping it checked by default for now until a fix is found.
3	Jennifer Nguyen	Make Registration Screen for user to create account	Created the registration screen by adding a function on the Register button in the Settings screen to direct it to the Register Screen	None
4	Ubay <u>Abdulaziz</u>	Make <u>snackbars</u> and other notifications appear on the app	Created the ability to make notifications and <u>snackbar</u> appear on the app	Needed to find a use case for the notifications and <u>snackbar</u> to appear as that is the only way for them to appear in a meaningful way.

📅 Sunday, July 12th @ 10:00 AM

	Name	Priorities 🤔	Progress 😊	Problems 😞
1	Wei Wen (Chloe) Chai	Implement Firebase to the app	Firebase has now been implemented to the app	None
2	John Aquino	Keeping an eye on the English to French translations	Found some hardcoded text as placeholders for the calendars and some buttons, but they have now been fixed	None
3	Jennifer Nguyen	Allow users to edit entries in calendar	Added clickable cells in calendar schedule that can let users type what they want in that cell and it will appear	Trouble saving it even after exiting the app
4	Ubay <u>Abdulaziz</u>	Work on the Contact Us functionality on the button	Managed to create the functionality for directing to the Contact Us on the button	None



Jul 15, 2023 @ 9:00 AM

	Name	Priorities 🤔	Progress 😊	Problems 😞
1	Wei Wen (Chloe) Chai	Fix the auto-login after using Google sign-in	Finished the task	None
2	John Aquino	<ul style="list-style-type: none">-Update Sprint dashboard and Gantt charts-Check app for corrections-Update Standup notes for report-Create System Context Diagram	<ul style="list-style-type: none">-Finishing up last standup note-Went through app one final time to see if the app is still working with no crashes or errors-Sprint dashboard and Gantt chart have been updated to be more focused on software-System context diagram has been created and completed	Checking through the app, the Remember Me functionality is still giving us trouble, but the app can still run with most functionalities in place.
3	Jennifer Nguyen	Create <u>snackbar</u> in feedback button and login screen	Updated the <u>snackbar</u> for the login screen	None
4	Ubay <u>Abdulaziz</u>	Make a Review screen that will appear when tapping on a button called "Feedback"	Updated and added some fields for the Review screen. Task was finished much earlier than we thought.	None