CENG 322

Deliverable 4

Team Name: Sleep Deprived
Project Name: Evee 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

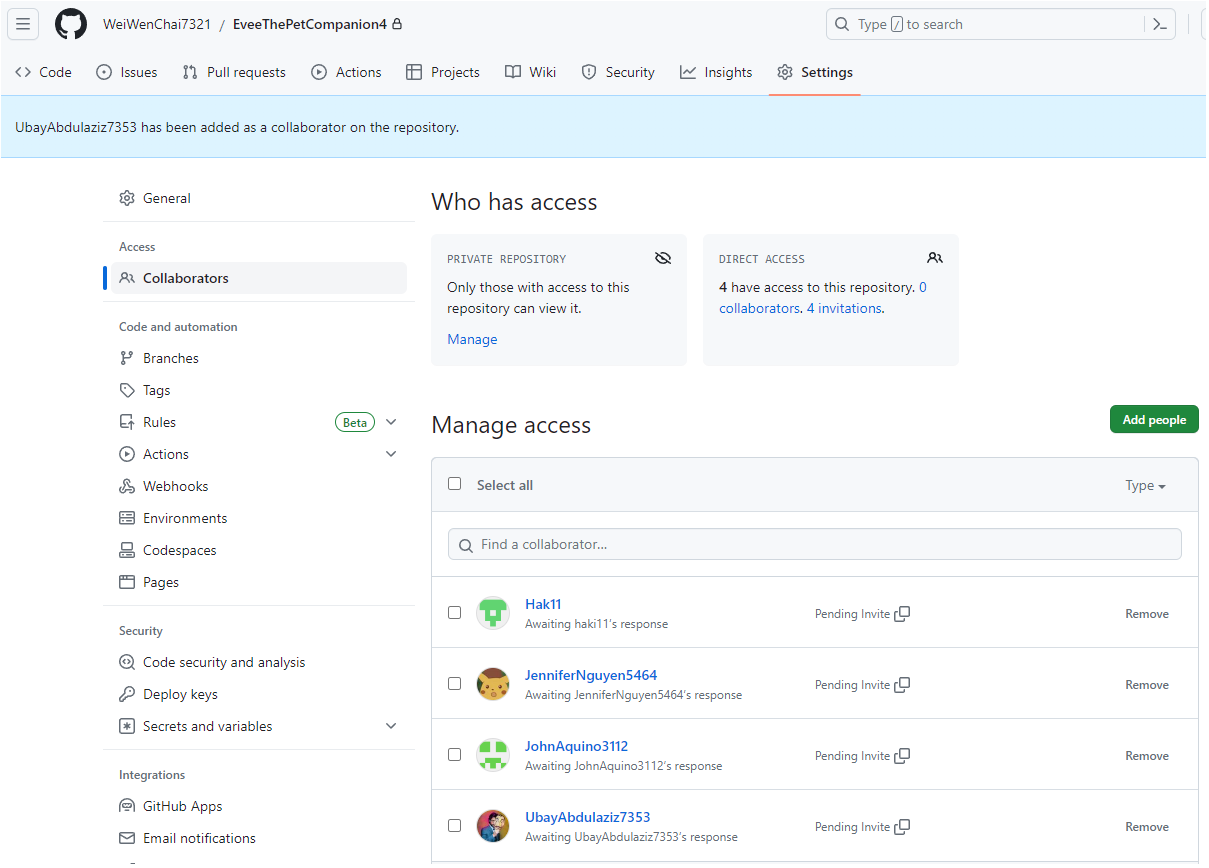
Name	ID	Signature	Effort
Wei Wen Chai	N01447321		100%
Jennifer Nguyen	N01435464		100%
John Aquino	N01303112		100%
Ubay Abdulaziz	N01437353		100%

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.

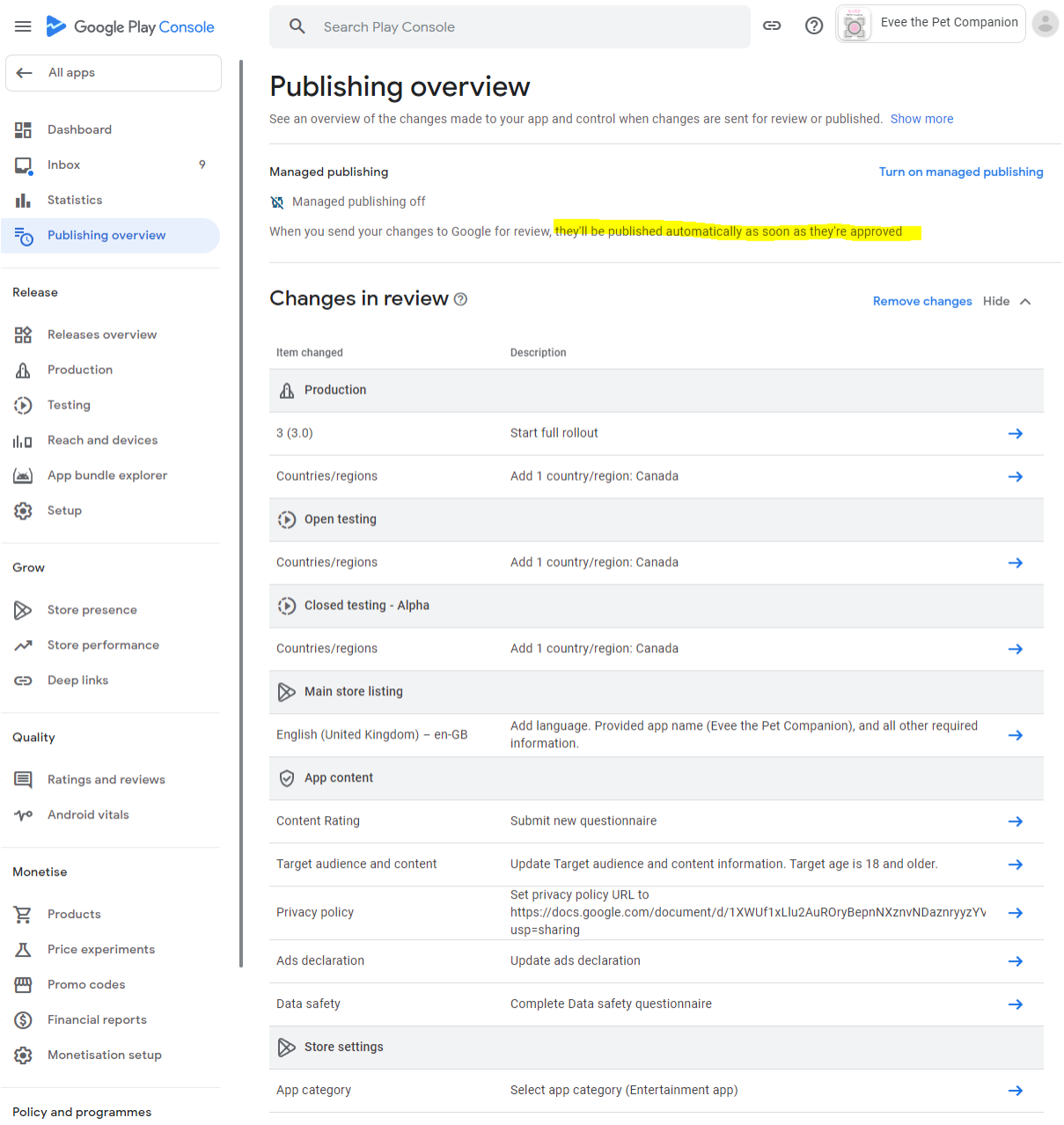
GitHub Repo Link

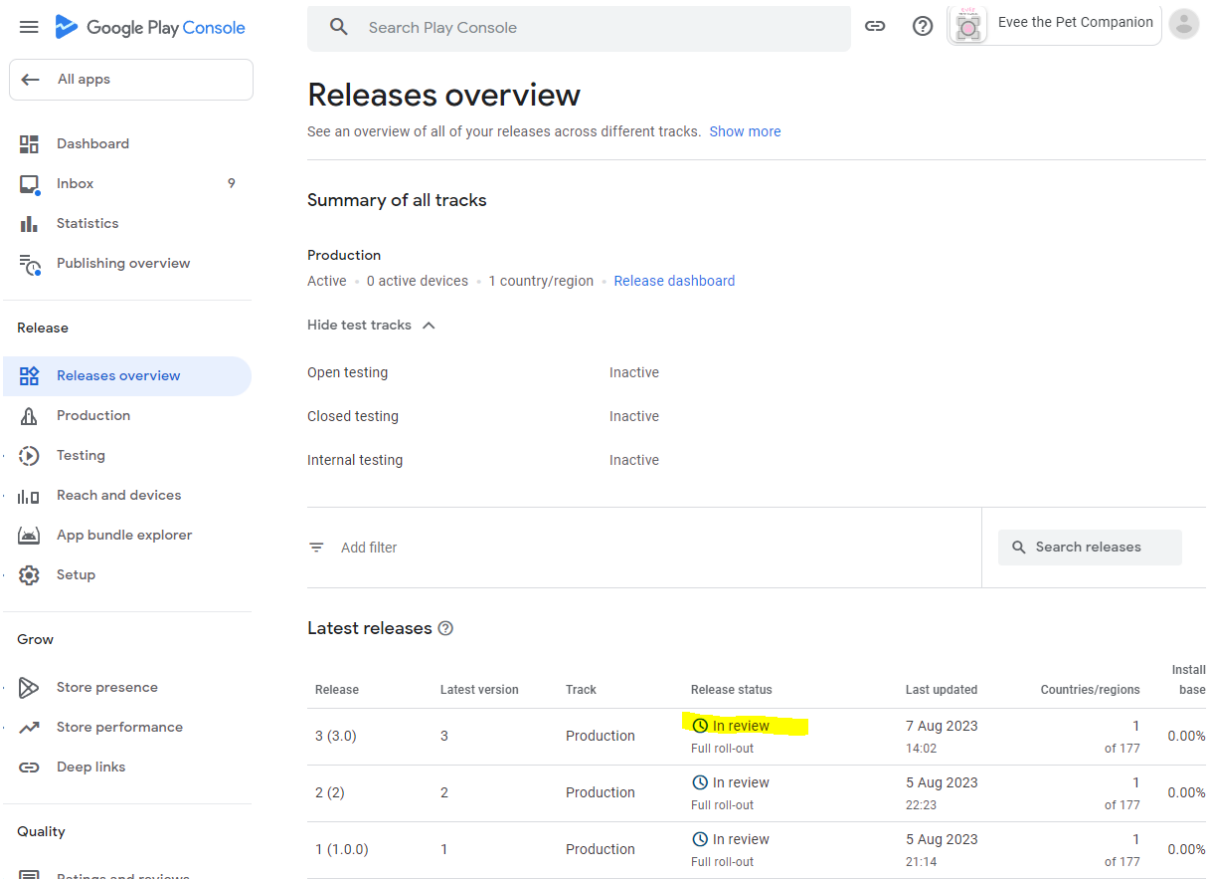
https://github.com/WeiWenChai7321/EveeThePetCompanion4/



Google Play Console

The app has been submitted to the Play Store and is under review as of August 5th, 2023.

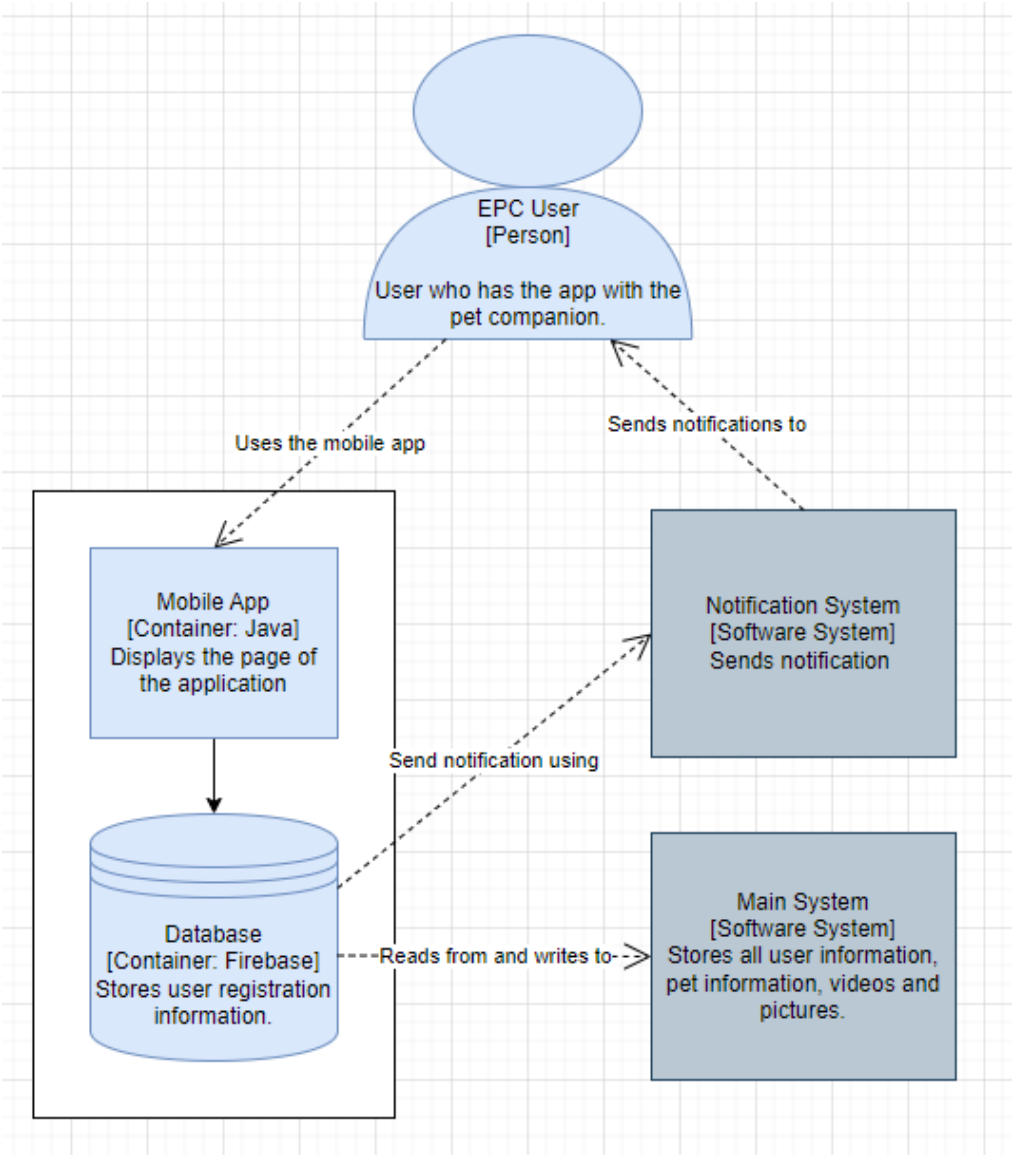


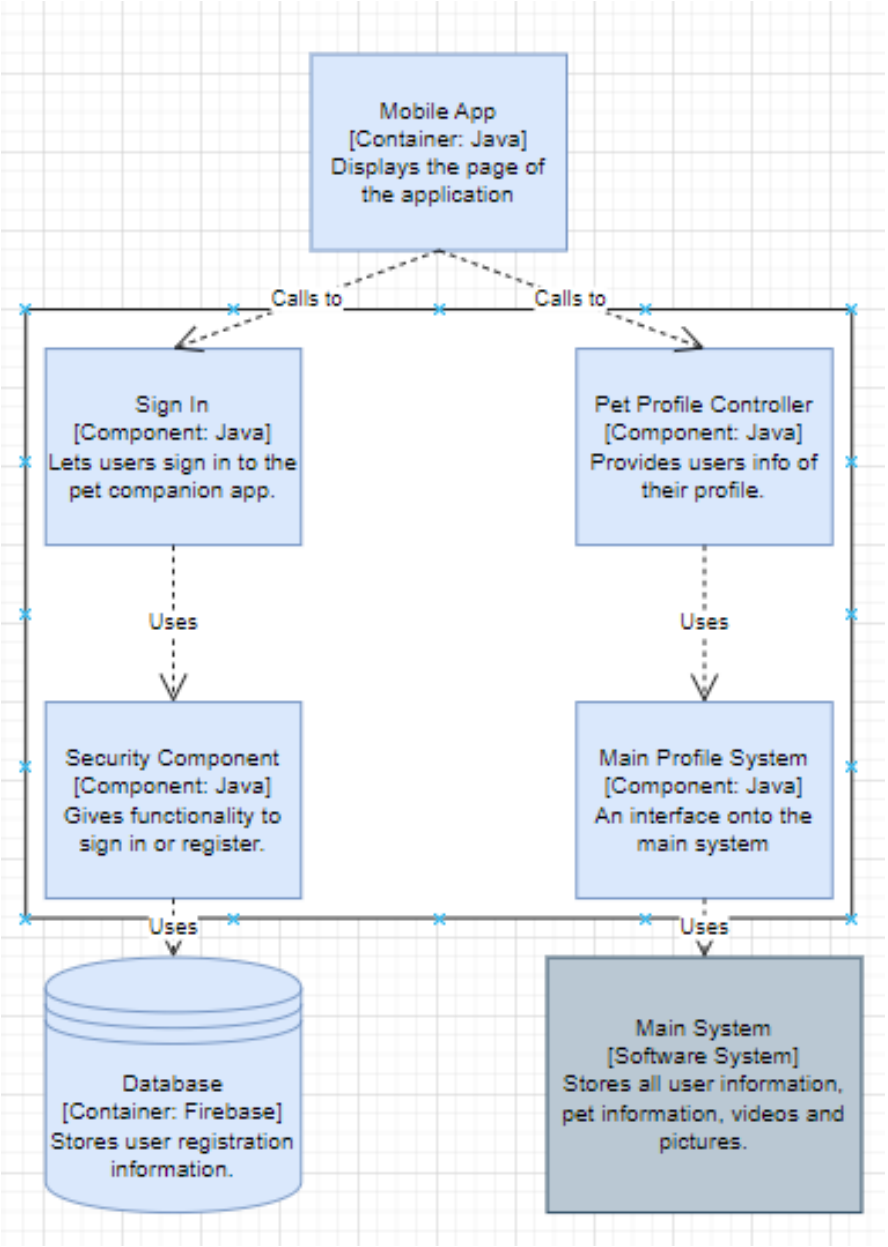


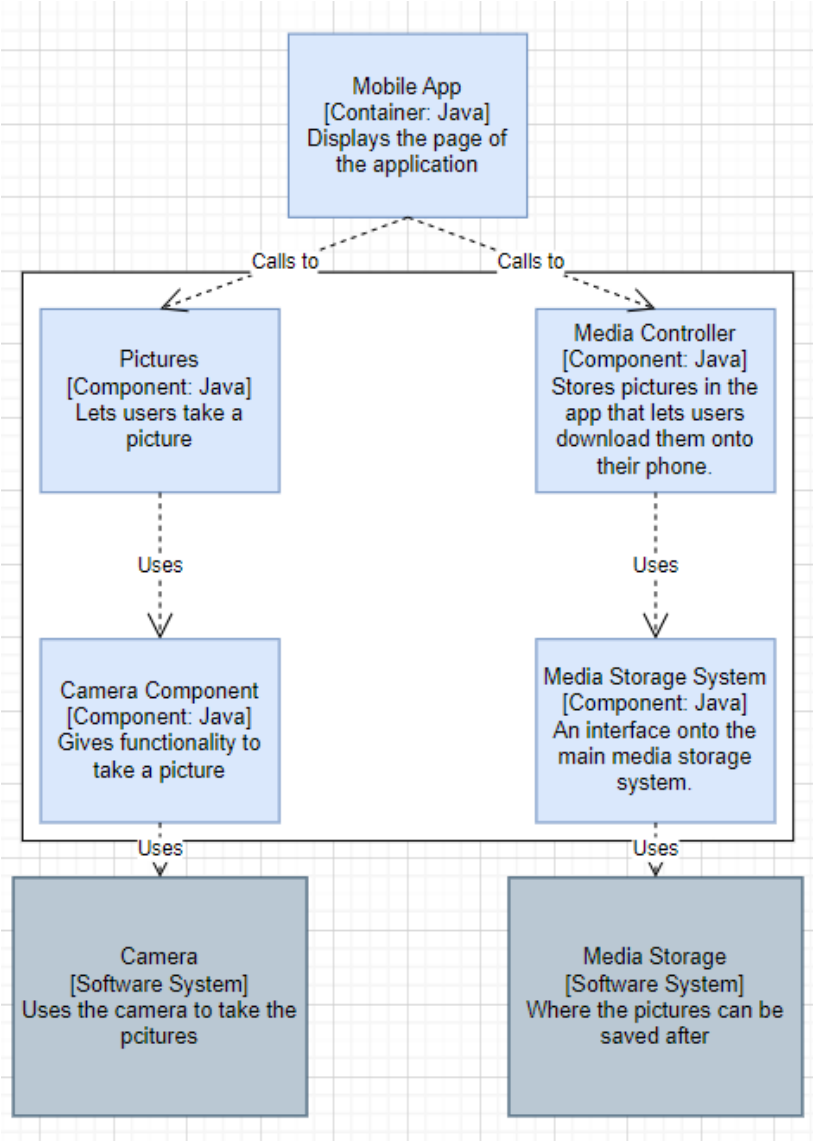
Sprint Goals

1. Icon Update: Change icons for the bottom navigation bar to better fit the purpose of each menu option
2. Implement some functionality for off-line mode.
3. Complete all functionality and connectivity to both Firebase Firestore (text) and Storage (images) databases - no hardcoded data
4. Create floating Fab button with appropriate functionality
5. Address previously given feedback: e.g. phone number field not accepting numbers without hyphens, path for downloading images is too long to be useful for users
6. Complete Functionality and UI on All Screens: Finalise all functionality and user interface elements across all screens and ensure image assets have different resolutions for various devices.
7. Test Cases: Write three types of test cases using JUnit 4, Robolectric, and Espresso, each with a minimum of five test cases for different classes and scenarios.
8. Delay and Progress Bar in Review Screen: Introduce a delay in submitting the form on the Review Screen and display a progress bar for a few seconds during the submission process.
9. App Refactoring and Code Comments: Refactor the entire app to improve code quality and maintainability, adding comments where code is refactored.
10. Upload the final app to the Google Play store with full functionality.

C4 Model: Container and Component Diagrams



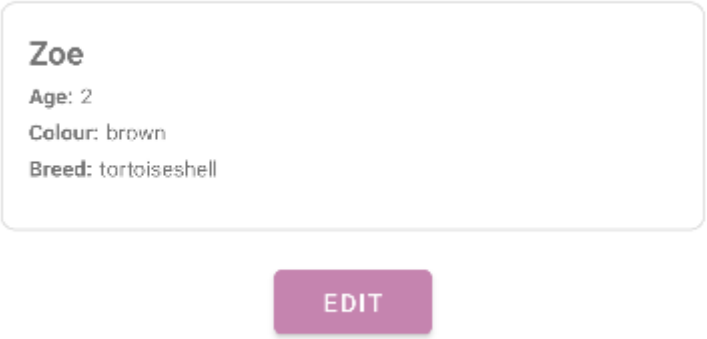

Mobile App
[Container: Java]
Displays the page of the application
Calls to
Calls to
Sign In
[Component: Java]
Lets users sign in to the pet companion app.
Pet Profile Controller
[Component: Java]
Provides users info of their profile.
Uses
Uses
Security Component
[Component: Java]
Gives functionality to sign in or register.
Main Profile System
[Component: Java]
An interface onto the main system
Uses
Uses
Database
[Container: Firebase]
Stores user registration information.
Main System
[Software System]
Stores all user information, pet information, videos and pictures.


Mobile App
[Container: Java]
Displays the page of the application
Calls to
Calls to
Pictures
[Component: Java]
Lets users take a picture
Media Controller
[Component: Java]
Stores pictures in the app that lets users download them onto their phone.
Uses
Uses
Camera Component
[Component: Java]
Gives functionality to take a picture
Media Storage System
[Component: Java]
An interface onto the main media storage system.
Uses
Uses
Camera
[Software System]
Uses the camera to take the pcitures
Media Storage
[Software System]
Where the pictures can be saved after

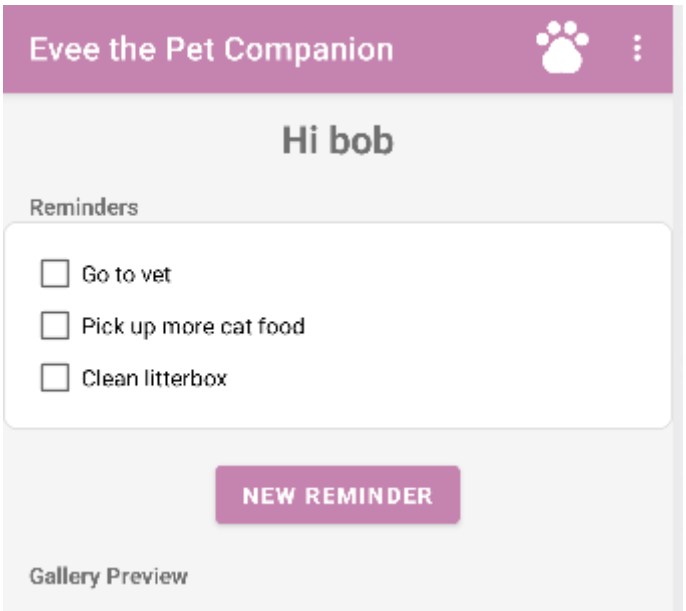
Offline Functionality

We have two offline features that allow users to still utilise parts of the app even when they are not connected to the Internet.

The "PetProfileFragment" retrieves and displays pet information (name, age, color, breed) from SharedPreferences, where it stores the latest data fetched from the Firebase Firestore database when the device was online. It also supports editing while offline, to be updated in the database when the Internet connection is restored.



Similarly in the “DashboardFragment”, the reminders list is also retrieved and displayed from SharedPreferences where the latest data was fetched the last time the user was online. The user can also create/delete reminders offline, and have their database updated when they access the fragment with an Internet connection. Additionally, in the same fragment the user will also still see the greeting with their first name even if they are offline.



Runtime Permission Features

The HighlightsFragment includes a "Download All" button, which prompts the user for runtime permission to write to external storage and allows them to save all their photos they have taken on the app to their Gallery.

The StreamFragment uses a runtime permission feature for accessing the camera. The app checks if the CAMERA permission is granted before starting the camera preview. If the permission is not granted, it requests the user for the permission. This camera preview is used as the background for this fragment and will allow the user to see their pet and capture photos.

Sprint Dashboard

Evee the Pet Companion — On track

Overview | List | Board | Timeline | Calendar | Workflow | Dashboard | Messages | Files

Add task | Filter | Sort: 1 | Hide

Task name	Story	Start Date	Due date	Done Date	T-shirt sizing	Priority	Designee
Update the Stream Fragment to upload pictures in Firebase Storage upon clicking button	Image Capture and Upload	Aug 3	Today	Aug 3	S	High	Chloe
Update Firebase Storage permissions to allow read/write for photos bucket	Image Capture and Upload	Aug 3	Today	Aug 3	M	Medium	Chloe
Read and display pictures from Storage database in Dashboard fragment	Image Capture and Upload	Aug 3	Today	Aug 3	M	Low	Chloe
Read and display pictures from Storage database in Gallery fragment	Image Capture and Upload	Aug 3	Today	Aug 3	M	Low	Chloe
Test and verify the storage functionality for pictures.	Image Capture and Upload	Aug 3	Today	Aug 3	M	Low	Chloe
Handle error cases and provide appropriate error messages.	Image Capture and Upload	Aug 3	Today	Aug 3	S	Low	Chloe
Save pet profile display picture in Firebase Storage	Image Capture and Upload	Aug 2	Today	Aug 2	S	Low	Chloe
Make pet profile display picture clickable	Image Capture and Upload	Aug 2	Today	Aug 2	S	Low	Chloe
Allow user to update pet profile picture in Firebase Storage	Image Capture and Upload	Aug 2	Today	Aug 2	S	Low	Chloe
Display toast once pet profile picture is successfully updated in database	Image Capture and Upload	Aug 2	Today	Aug 2	S	Low	Chloe
Test class using Junit 4 and minimum of 5 test cases.	Unit Testing	Jul 18	Today	Jul 22	S	High	Ubay
Test class using Roboletric and minimum of 5 test cases.	Unit Testing	Jul 20	Today	Jul 21	M	High	Ubay
Test class using Espresso and minimum of 5 test cases.	Unit Testing	Jul 19	Today	Jul 20	M	High	Ubay
Write Junit test cases to validate Login credentials. Minimum of 7 test cases.	Unit Testing	Jul 21	Today	Jul 21	M	High	Ubay
Update build.gradle to use Junit, Roboletric, Espresso	Unit Testing	Aug 5	Today	Today	M	High	Ubay
Handle error cases where Google sign in process is interrupted	Updating Account Information	Jul 27	Today	Aug 27	S	Medium	John
Fix toast that says you cannot edit a google email (always appears)	Updating Account Information	Jul 20	Today	Jul 20	S	High	John
Save pet information to shared preferences to display in offline mode	Updating Account Information	Aug 5	Today	Aug 5	M	Medium	John
Save pet information to database	Updating Account Information	Aug 2	Today	Today	L	Medium	John
Update pet information in shared preferences with last pulled info from database	Updating Account Information	Aug 4	Today	Aug 4	L	Medium	John
Address previous feedback: allow phone number to be entered without dashes	Login and Authentication	Jul 19	Today	Jul 19	M	Low	Jennifer
Implement the logic to handle account deletion when button is clicked.	Login and Authentication	Jul 19	Today	Jul 19	S	Low	Jennifer
Prompt the user for confirmation before proceeding with the account deletion.	Login and Authentication	Jul 19	Today	Jul 19	S	Low	Ubay
Save firstName and lastName in Firestore Firebase	Login and Authentication	Aug 3	Today	Aug 3	M	High	Chloe
Handle error cases for Google-sign in	Login and Authentication	Jul 19	Today	Aug 19	L	Low	Chloe
Pull first name from database to greet user upon logging in	Login and Authentication	Aug 4	Today	Aug 4	S	Low	Ubay
Create floating Fab button to allow user to scroll to top of page	Troubleshooting & Feedback	Jul 21	Today	Jul 21	S	Low	John
Make counter to track total number of treats available	Pet Interaction	Aug 3	Today	Aug 3	S	Low	Jennifer
Display toast every time treat button is pressed, showing number of treats remaining	Pet Interaction	Aug 4	Today	Aug 4	S	Low	Jennifer
Save reminders in database	Reminders	Jul 26	Today	Aug 26	M	Medium	Jennifer
Change button to edit reminders to a toggle logic (update, save)	Reminders	Jul 27	Today	Jul 27	S	Medium	Jennifer
Change database rules to add write and read access for reminders collection	Reminders	Jul 27	Today	Jul 27	M	High	Jennifer
Sprint 3							
Add an editable email field to SettingsFragment UI	Updating Account Information	Jul 4	Jul 17	Jul 4	S	Medium	Chloe
Implement logic to update the email address in the database	Updating Account Information	Jul 4	Jul 17	Jul 4	M	Medium	Chloe
Handle cases where user logged in with UI	Updating Account Information	Jul 4	Jul 17	Jul 4	S	Low	John
Test and verify email address update functionality	Updating Account Information	Jul 4	Jul 17	Jul 4	S	Medium	John
Provide visual feedback to user on successful email address update	Updating Account Information	Jul 4	Jul 17	Jul 4	S	Low	Jennifer
Make logout screen appear if user presses on logout button	Login and Authentication	Jul 6	Jul 17	Jul 6	S	High	Jennifer
Move logout button to settings page	Login and Authentication	Jul 6	Jul 17	Jul 6	S	Low	Jennifer
Create snackbar to show that user is being logged in	Login and Authentication	Jul 6	Jul 17	Jul 6	S	Low	Jennifer
Create a "Contact Us" button in HelpFragment UI	Troubleshooting & Feedback	Jul 14	Jul 17	Jul 14	S	Low	Ubay
Implement the logic to open the default email client with a pre-filled template	Troubleshooting & Feedback	Jul 14	Jul 17	Jul 14	S	Low	Ubay
Pre-fill the email template with appropriate recipient and subject	Troubleshooting & Feedback	Jul 14	Jul 17	Jul 14	S	Low	Ubay
Test and verify the "Contact Us" functionality	Troubleshooting & Feedback	Jul 14	Jul 17	Jul 14	S	Low	Jennifer
Handle error cases and provide appropriate error messages	Troubleshooting & Feedback	Jul 14	Jul 17	Jul 14	S	Low	Ubay
Make clickable button for treat dispensing	Pet Interaction	Jul 12	Jul 17	Jul 12	S	Medium	John
Make clickable button for obstacle avoidance	Pet Interaction	Jul 12	Jul 17	Jul 12	S	Medium	John
Make clickable button for line following	Pet Interaction	Jul 12	Jul 17	Jul 12	S	Medium	John
Make clickable arrow buttons for motion	Pet Interaction	Jul 12	Jul 17	Jul 12	S	Medium	John
Make clickable arrow buttons for turning left and right	Pet Interaction	Jul 12	Jul 17	Jul 12	S	Medium	John
Implement logic to add new reminders and display them in UI	Reminders	Jul 9	Jul 17	Jul 9	M	Medium	Chloe
Add UI elements to Dashboard Fragment for adding and deleting reminders	Reminders	Jul 9	Jul 17	Jul 9	S	Medium	Chloe
Allow users to delete unwanted reminders from the list	Reminders	Jul 9	Jul 17	Jul 9	S	Medium	Chloe
Test and verify reminder control functionality	Reminders	Jul 9	Jul 17	Jul 9	S	Medium	Chloe
Save reminders in sharedPreferences	Reminders	Jul 9	Jul 17	Jul 9	S	Low	Chloe

Add task

Sprint 2

Create download all button in gallery fragment	Image Capture and Upload	Jun 8	Jun 11	Jun 8	S	Low	John
Create logic to download all images on page once button is pressed	Image Capture and Upload	Jun 8	Jun 11	Jun 8	S	Medium	John
Display toast with location of downloaded images upon successful action	Image Capture and Upload	Jun 8	Jun 11	Jun 8	S	Low	John
Create settings option in overflow menu	Updating Account Information	Jun 9	Jun 11	Jun 9	S	Low	Ubay
Create Settings page UI	Updating Account Information	Jun 9	Jun 11	Jun 9	S	Medium	Jennifer
Create pet profile page UI	Updating Account Information	Jun 9	Jun 11	Jun 9	S	Medium	Chloe
Create toggle switch in Settings page for locking screen orientation	Updating Account Information	Jun 9	Jun 11	Jun 9	S	Medium	Chloe
Create field to edit email	Updating Account Information	Jun 9	Jun 11	Jun 9	S	Medium	Jennifer
Implement logic for locking screen orientation	Updating Account Information	Jun 9	Jun 11	Jun 9	M	Medium	Chloe
Create login page UI with necessary fields	Login and Authentication	Jun 11	Jun 11	Jun 11	S	Medium	Jennifer
Make login page appear before MainActivity	Login and Authentication	Jun 11	Jun 11	Jun 11	M	Low	Jennifer
Create logout page UI	Login and Authentication	Jun 11	Jun 11	Jun 11	S	Low	Jennifer
Create database for account management	Login and Authentication	Jun 11	Jun 11	Jun 11	M	High	Jennifer
Create introduction for help page to describe how to use it	Troubleshooting & Feedback	Jun 6	Jun 11	Jun 6	S	Low	Ubay
Create commonly asked questions section	Troubleshooting & Feedback	Jun 6	Jun 11	Jun 6	S	Low	Ubay
Create help option in overflow menu	Troubleshooting & Feedback	Jun 6	Jun 11	Jun 6	S	Low	Ubay
Create multiple resolutions for each image	Media Playback	Jun 10	Jun 11	Jun 10	M	Medium	John
Create landscape views for all pages	Media Playback	Jun 10	Jun 11	Jun 10	S	Low	John
Create layout for stream fragment layout	Pet Interaction	Jun 9	Jun 11	Jun 9	M	Medium	Chloe
Create layout for dashboard fragment	Reminders	Jun 8	Jun 11	Jun 9	S	Medium	Chloe
Create reminders UI in dashboard fragment	Reminders	Jun 8	Jun 11	Jun 9	S	Medium	Chloe

Add task...

Sprint 1 (Project Planning)

Project Review Meeting

Initial Expectations

- Complete 4 deliverables, 2-3 weeks apart across 3 month span
- Build a fully functioning Android app

Project Recap

- Achieved primary goals, successfully implementing core features and functionalities.
- Delivered project within set timeline
- The app exhibits excellent performance (proper error handling, all features work as intended), offering a seamless user experience.

Unexpected Roadblocks

- Learning curve in integrating Firebase Firestore with the app
- Setting up camera orientation and taking photo feature (StreamFragment) was more complex than we expected, took a long time to troubleshoot
- Displaying images from Firebase Storage database in proper grid format and sizes was very challenging
- Google sign-in and creating account features did not initially include Authentication. This took us a long time to figure out why we could not sign in after creating accounts

Learnings

- Integrating Firebase databases taught us the importance of thorough documentation review and dedicating time to understand new technologies fully.
- Dealing with camera orientation and photo feature complexities emphasized the need for meticulous testing and anticipating potential issues early on.
- The challenges in displaying images highlighted the significance of optimizing database queries and handling image sizes for better app performance.
- The oversight in account creation underscored the necessity of implementing comprehensive authentication mechanisms from the start, ensuring a seamless user experience.

Stakeholder Feedback

- Stakeholders acknowledged the seamless integration of Firebase functionalities, especially the real-time updates in the StreamFragment, which enhanced the app's responsiveness.
- Team members provided positive feedback on the effective collaboration and communication within the team, resulting in a well-coordinated development process.
- Some stakeholders suggested adding more pet-related features, such as a pet health tracker or pet training tips, to further engage users and offer a comprehensive pet companion experience.

Next Steps

- Based on the feedback and learnings, the team has planned future feature enhancements to further elevate the app's capabilities.
- Continuous improvement will be emphasized to maintain the app's performance and keep it up-to-date with evolving user needs.
- Comprehensive documentation in the form of the report will be finalized.
- PowerPoint presentation will be created

Post-Mortem

Project Performance Review:

Cost: The project was completed within the allocated budget, and expenses were managed effectively throughout the development process.
Schedule: The project was delivered on time, meeting the planned milestones and deadlines.
Quality: The overall quality of the app was satisfactory, with most functionalities and features working as intended.

Time Management:
The team members demonstrated good time management skills, adhering to project timelines and delivering their assigned tasks promptly.
There were minimal instances of last-minute rushes, and most tasks were well-planned and executed on time.

Quality and Compromises:
There were some minor issues with the quality of certain features, particularly related to association of reminders, photos, and pet information with accounts. Due to time constraints, this nice-to-have feature was not implemented but it was not listed as a project requirement. However, no significant compromises were made on core functionalities or user experience.

Lessons Learned and Areas of Improvement:
Integrating Firebase Firestore proved to be a valuable learning experience, highlighting the importance of understanding the technology thoroughly before implementation.
The team learned the significance of thorough testing, especially when dealing with features involving camera and image handling.
Continuous code reviews and collaboration could be improved to catch potential issues early in the development cycle. As previously mentioned in the “quality and compromises” section, we could improve the database in terms of associating certain data with the account that added it (this would allow the user to delete all associated data with their account).

Attendees and Absences:
The project review meeting was attended by all team members involved in the app development process.

In conclusion, the project performed well in terms of cost, schedule, and quality. The team members efficiently managed their time and successfully delivered the project on time. While there were minor quality issues, the overall app functionality met expectations. The project provided valuable learning experiences, and the team identified areas for improvement in future projects. The project review meeting was well-attended, fostering collaboration and effective communication among all stakeholders.

Technical Debt

In our project, we took a proactive approach to tackle technical debt and maintain a robust codebase. This was our strategy:

Prioritization: We assessed each identified problem and ranked them based on their potential impact on the app in meetings. We considered factors like how it could affect the app's stability, performance, and long-term maintenance. By prioritizing the most critical issues, we could focus our efforts where they mattered the most.

Involvement of the Team: We held regular meetings to share our findings and insights about large bug fixes and features to implement. Everyone had a chance to contribute and pitch ideas on how to address it effectively. This collaborative approach allowed us to make sure that we were all on the same page and offer solutions that others may not have thought of.

Frequent Testing: We were serious about not introducing new problems while trying to fix the existing ones. To ensure that our efforts to implement new features didn't lead to unexpected regressions, we adopted a comprehensive testing strategy. We conducted frequent testing throughout the development process which helped us catch potential issues early, giving us the confidence to refactor and improve the code without fear of breaking anything unintentionally.

By following these strategies, we managed to tackle technical debt effectively in our project while meeting important deadlines. It resulted in a codebase that was easier to maintain and extend.

Refactoring

RegisterActivity

The refactoring separates validation logic from the main registerUser() method into a new isValidInput() method, improving code modularity. Additionally, a helper method showToast() is introduced to handle Toast messages, reducing code duplication. The Firebase authentication checks are now handled in a separate method called checkEmailAndRegister(), making the code more organized and easier to understand.

(Before)

```
private void registerUser() {
    final String fullName = fullNameEditText.getText().toString().trim();
    final String email = emailEditText.getText().toString().trim();
    String password = passwordEditText.getText().toString().trim();
    String confirmPassword = confirmPasswordEditText.getText().toString().trim();
    String phoneNumber = phoneEditText.getText().toString().trim();

    if (email.isEmpty() || password.isEmpty() || confirmPassword.isEmpty() || fullName.isEmpty() || phoneNumber.isEmpty()) {
        Toast.makeText(RegisterActivity.this, R.string.please_fill_fields, Toast.LENGTH_SHORT).show();
    } else if (!password.equals(confirmPassword)) {
        Toast.makeText(RegisterActivity.this, R.string.passwds_not_match, Toast.LENGTH_SHORT).show();
    } else if (!isValidPassword(password)) {
        Toast.makeText(RegisterActivity.this, R.string.invalid_password, Toast.LENGTH_SHORT).show();
    } else if (!android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
        Toast.makeText(RegisterActivity.this, R.string.invalid_email, Toast.LENGTH_SHORT).show();
    } else if (phoneNumber.length() != 10 || !Patterns.PHONE.matcher(phoneNumber).matches()) {
        Toast.makeText(RegisterActivity.this, R.string.invalid_phone_number, Toast.LENGTH_SHORT).show();
    } else {
        // Check if the email already exists in Firebase Authentication
        firebaseAuth.fetchSignInMethodsForEmail(email).addOnCompleteListener(new
OnCompleteListener<SignInMethodQueryResult>() {
            @Override
            public void onComplete(@NonNull Task<SignInMethodQueryResult> task) {
                if (task.isSuccessful()) {
                    SignInMethodQueryResult result = task.getResult();
                    if (result != null && result.getSignInMethods() != null && result.getSignInMethods().size() > 0) {
                        // Email already exists in Firebase Authentication, show error message
                        Toast.makeText(RegisterActivity.this, R.string.account_already_exists, Toast.LENGTH_SHORT).show();
                    } else {
                        // Email doesn't exist in Firebase Authentication, proceed with account creation
                        createAccount(email, password);
                    }
                } else {
                    // Error occurred while accessing Firebase Authentication, show error message
                    Toast.makeText(RegisterActivity.this, R.string.failed_access_firebase, Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

(After)

```
private void registerUser() {
    final String fullName = fullNameEditText.getText().toString().trim();
    final String email = emailEditText.getText().toString().trim();
    String password = passwordEditText.getText().toString().trim();
    String confirmPassword = confirmPasswordEditText.getText().toString().trim();
    String phoneNumber = phoneEditText.getText().toString().trim();

    if (isValidInput(fullName, email, password, confirmPassword, phoneNumber)) {
```

```
        checkEmailInFirebase(email, password);
    }
}

private boolean isValidInput(String fullName, String email, String password, String confirmPassword, String phoneNumber) {
    if (email.isEmpty() || password.isEmpty() || confirmPassword.isEmpty() || fullName.isEmpty() || phoneNumber.isEmpty()) {
        showToast(R.string.please_fill_fields);
        return false;
    } else if (!password.equals(confirmPassword)) {
        showToast(R.string.passwds_not_match);
        return false;
    } else if (!isValidPassword(password)) {
        showToast(R.string.invalid_password);
        return false;
    } else if (!android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
        showToast(R.string.invalid_email);
        return false;
    } else if (phoneNumber.length() != 10 || !Patterns.PHONE.matcher(phoneNumber).matches()) {
        showToast(R.string.invalid_phone_number);
        return false;
    }
    return true;
}

private void showToast(int stringId) {
    Toast.makeText(RegisterActivity.this, stringId, Toast.LENGTH_SHORT).show();
}

private void checkEmailAndRegister() {
        final String email = emailEditText.getText().toString().trim();
        final String password = passwordEditText.getText().toString().trim();

        firebaseAuth.fetchSignInMethodsForEmail(email).addOnCompleteListener(new
OnCompleteListener<SignInMethodQueryResult>() {
            @Override
            public void onComplete(@NonNull Task<SignInMethodQueryResult> task) {
                if (task.isSuccessful()) {
                    SignInMethodQueryResult result = task.getResult();
                    if (result != null && result.getSignInMethods() != null && result.getSignInMethods().size() > 0) {
                        Toast.makeText(RegisterActivity.this, R.string.account_already_exists, Toast.LENGTH_SHORT).show();
                    } else {
                        createAccount(email, password);
                    }
                } else {
                    Toast.makeText(RegisterActivity.this, R.string.failed_access_firebase, Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
```

DashboardFragment

The refactoring involves extracting the toggle visibility and button text update logic into a separate method named toggleReminderEditText(), improving code readability and eliminating duplication.

(Before)

```
@Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_dashboard, container, false);
        remindersLayout = view.findViewById(R.id.reminders_card);
        editReminderEditText = view.findViewById(R.id.edit_text_reminder);
```

```
    editReminderEditText.setVisibility(View.GONE); // Initially hide the EditText
    Button newReminderButton = view.findViewById(R.id.button_new_reminder);
    updateNoRemindersVisibility();
    newReminderButton.setOnClickListener(v -> {
        String reminderText = editReminderEditText.getText().toString().trim();
        if (isEditTextVisible) {
            // If the EditText is visible, hide it and change the button text to "New Reminder"
            editReminderEditText.setVisibility(View.GONE);
            newReminderButton.setText(R.string.new_reminder_button_text);
            hideKeyboard(); // Hide the keyboard when canceling
        } else {
            // If the EditText is not visible, show it and change the button text to "Cancel"
            editReminderEditText.setVisibility(View.VISIBLE);
            editReminderEditText.requestFocus();
            newReminderButton.setText(R.string.cancel_button_text);
            showKeyboard(); // Show the keyboard when creating a new reminder
        }
        isEditTextVisible = !isEditTextVisible;
        updateNoRemindersVisibility();
    });
```

(After)

```
@Override
  public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle
savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_dashboard, container, false);
    remindersLayout = view.findViewById(R.id.reminders_card);
    editReminderEditText = view.findViewById(R.id.edit_text_reminder);
    editReminderEditText.setVisibility(View.GONE);
    Button newReminderButton = view.findViewById(R.id.button_new_reminder);
    updateNoRemindersVisibility();
    newReminderButton.setOnClickListener(v -> {
        String reminderText = editReminderEditText.getText().toString().trim();
        if (isEditTextVisible) {
            toggleReminderEditText(false);
            hideKeyboard();
        } else {
            toggleReminderEditText(true);
            showKeyboard();
        }
    });

private void toggleReminderEditText(boolean isVisible) {
    Button newReminderButton = view.findViewById(R.id.button_new_reminder);
    if (isVisible) {
        editReminderEditText.setVisibility(View.VISIBLE);
        editReminderEditText.requestFocus();
        newReminderButton.setText(R.string.cancel_button_text);
    } else {
        editReminderEditText.setVisibility(View.GONE);
        newReminderButton.setText(R.string.new_reminder_button_text);
    }
    isEditTextVisible = isVisible;
    updateNoRemindersVisibility();
  }
```

Suggestions

We recommend presenting the assignment instructions in a more user-friendly format for students. For example, in the Deliverable 4 PDF, there are three different numbered lists, which always prompts us to make our own list in a different format. Combining all the tasks into one list would make it easier for students to understand what is expected of them.

We also suggest updating the assignment PDF instructions. Some deadlines mentioned are outdated, leading to confusion. Additionally, in one instance, the instructions mentioned adding a professor as a collaborator who isn't teaching this semester.

Regarding app design feedback, we valued the input received. However, we hope that students won't have marks deducted for subjective choices, such as adding features not explicitly requested, as long as they meet all the assignment requirements. We agreed with all your suggestions, such as focusing on a single pet in the pet profile fragment instead of supporting multiple pets.

Despite these points, we enjoyed the app development process and the opportunity to learn how to integrate it with a database. We appreciate the clarity provided in each deliverable's requirements.