D424 –Software Engineering

Task 3

Capstone Proposal Project Name: Pet Health Companion Mobile Application

Student Name: Andrea Hayes

Table of Contents

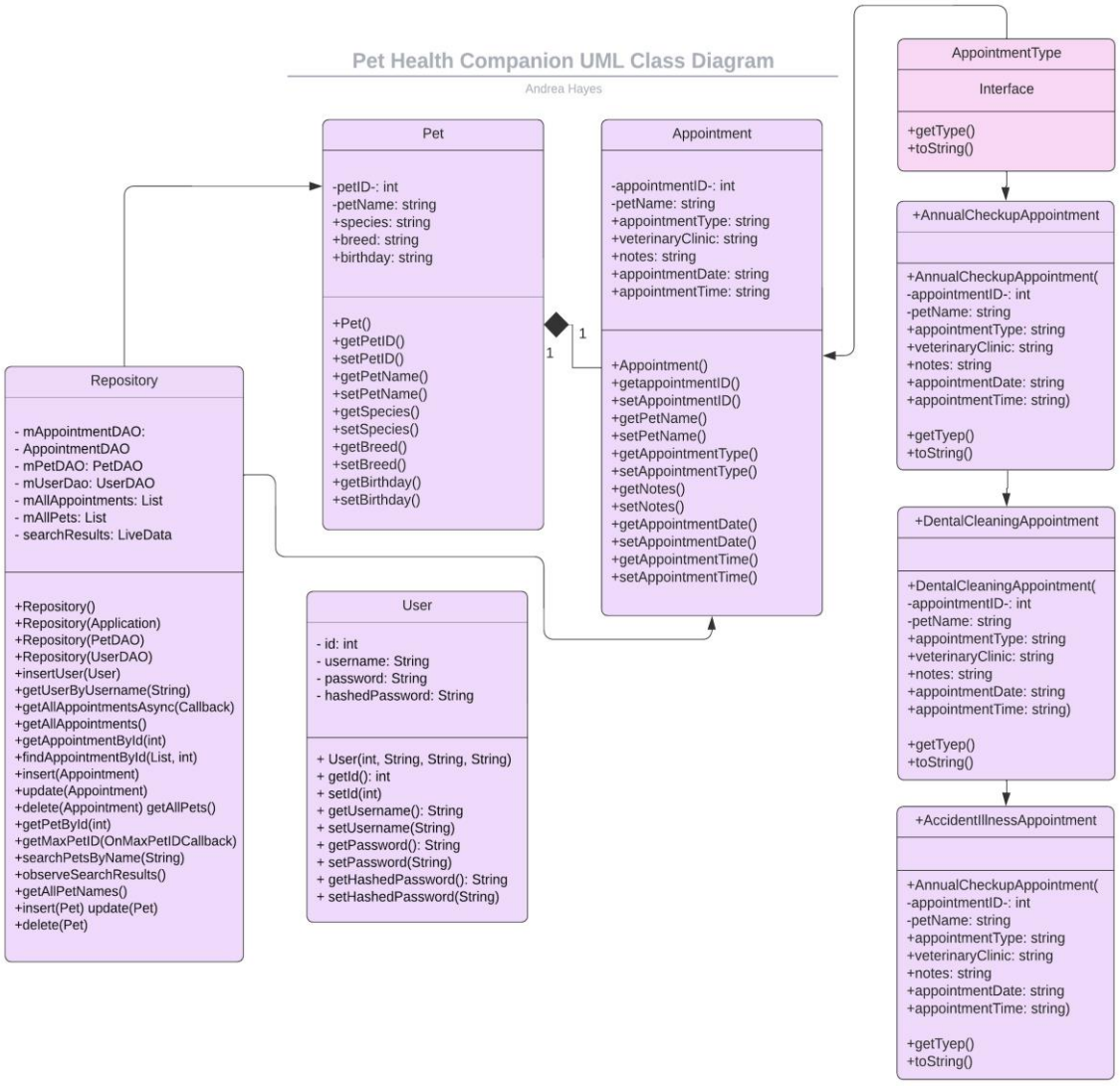
Application Design and Testing

Class Design

The Pet Health Companion is a minimum viable product (MVP) and was created in Android Studio, an Android app develop environment. To represent the class diagram for the Pet Health Companion, a UML diagram has been included below, depicting the main class components and their interrelationships. The class structure is designed to address pet owners' requirements for managing their pet health, primary in the area of scheduling pet health care appointments. The mobile application's primary classes - Pet and Appointment - handle crucial data and facilitate key interactions within the application.

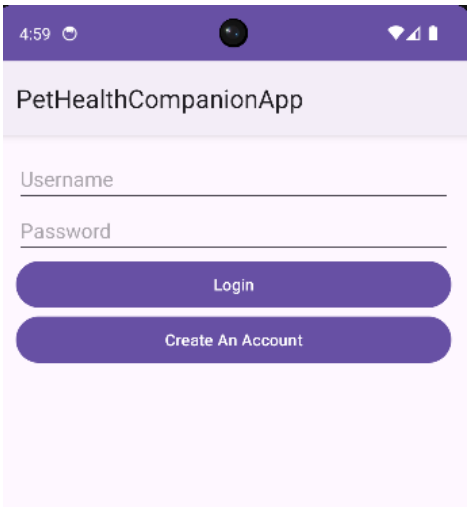
The class diagram visually represents the attributes, methods, and relationships among these components, providing a comprehensive illustration of the structure and functionality offered within the application. Android Studio's intuitive interface and powerful development tools make it possible for developers to implement multiple features. This includes appointment tracking, a wpet ' w z ' w feature, and customized reminders for appointments, ensuring that pets receive timely care when they need it. The Pet Health Companion embraces Android Studio's flexible and scalable set of development tools to steamline the pet care process, strengthening the bond between pet owners and their furry friends.

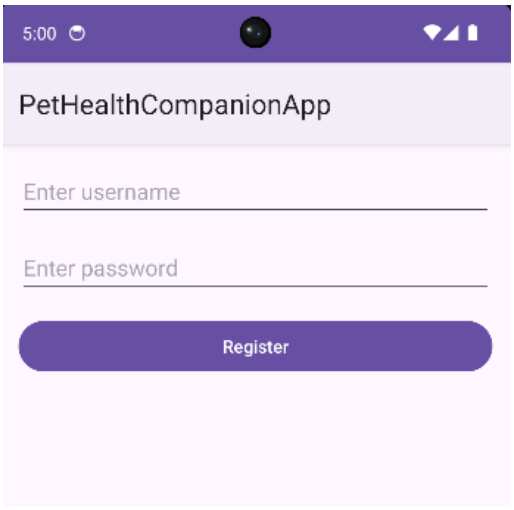
The Pet Health Companion app enables pet owners to easily incorporate pet care into their everyday routine, using the app as a consistent aid in overseeing their pets' health. With features designed to prioritize preventative care and improve communication with veterinarians, the Pet Health Companion enables pet owners to focus less on administrative tasks and more on the enjoyable activities of being a pet parent. By leveraging the capabilities of Android Studio, the app allows pet owners to maximize the joy of pet ownership while minimizing the stress.


Pet Health Companion UML Class Diagram
Andrea Hayes
Pet
-petID-: int
-petName: string
+species: string
+breed: string
+birthday: string
+Pet()
+getPetID()
+setPetID()
+getPetName()
+setPetName()
+getSpecies()
+setSpecies()
+getBreed()
+setBreed()
+getBirthday()
+setBirthday()
Appointment
-appointmentID-: int
-petName: string
+appointmentType: string
+veterinaryClinic: string
+notes: string
+appointmentDate: string
+appointmentTime: string
+Appointment()
+getappointmentID()
+setAppointmentID()
+getPetName()
+setPetName()
+getAppointmentType()
+setAppointmentType()
+getNotes()
+setNotes()
+getAppointmentDate()
+setAppointmentDate()
+getAppointmentTime()
+setAppointmentTime()
1
1
AppointmentType
Interface
+getType()
+toString()
+AnnualCheckupAppointment
+AnnualCheckupAppointment(
-appointmentID-: int
-petName: string
+appointmentType: string
+veterinaryClinic: string
+notes: string
+appointmentDate: string
+appointmentTime: string)
+getTyep()
+toString()
+DentalCleaningAppointment
+DentalCleaningAppointment(
-appointmentID-: int
-petName: string
+appointmentType: string
+veterinaryClinic: string
+notes: string
+appointmentDate: string
+appointmentTime: string)
+getTyep()
+toString()
+AccidentIllnessAppointment
+AnnualCheckupAppointment(
-appointmentID-: int
-petName: string
+appointmentType: string
+veterinaryClinic: string
+notes: string
+appointmentDate: string
+appointmentTime: string)
+getTyep()
+toString()
Repository
- mAppointmentDAO:
- AppointmentDAO
- mPetDAO: PetDAO
- mUserDao: UserDAO
- mAllAppointments: List
- mAllPets: List
- searchResults: LiveData
+Repository()
+Repository(Application)
+Repository(PetDAO)
+Repository(UserDAO)
+insertUser(User)
+getUserByUsername(String)
+getAllAppointmentsAsync(Callback)
+getAllAppointments()
+getAppointmentById(int)
+findAppointmentById(List, int)
+insert(Appointment)
+update(Appointment)
+delete(Appointment) getAllPets()
+getPetById(int)
+getMaxPetID(OnMaxPetIDCallback)
+searchPetsByName(String)
+observeSearchResults()
+getAllPetNames()
+insert(Pet) update(Pet)
+delete(Pet)
User
- id: int
- username: String
- password: String
- hashedPassword: String
+ User(int, String, String, String)
+ getId(): int
+ setId(int)
+ getUsername(): String
+ setUsername(String)
+ getPassword(): String
+ setPassword(String)
+ getHashedPassword(): String
+ setHashedPassword(String)

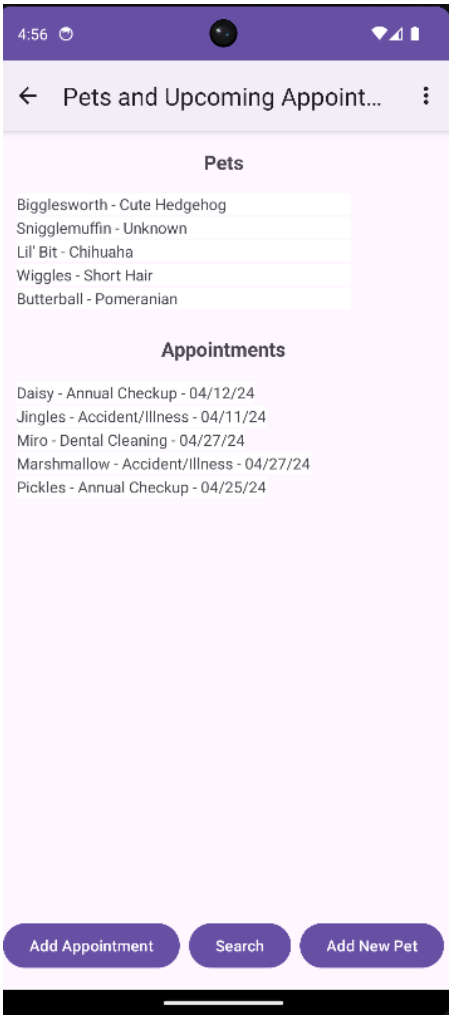
UI Design

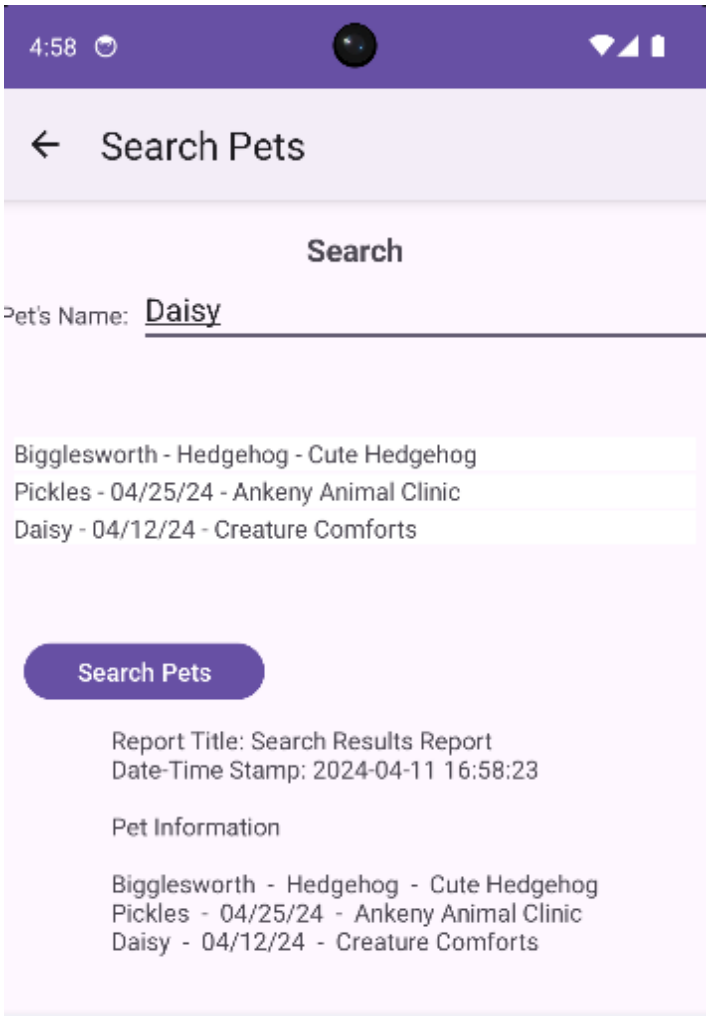
The Pet Health Companion application posseses a user interface (UI) that is easy to use, ensuring a reliable, consistent experience for pet owners. Navigation through the app starts with a login page where users can either log in or, ' ' w 'yet, can go to a registration page where they register with a new username and password. The login functionality guarantees that all pet data will be secure. If the user logs in successfully, they will be able to view data, as well as the date, time, and type of any upcoming appointments.



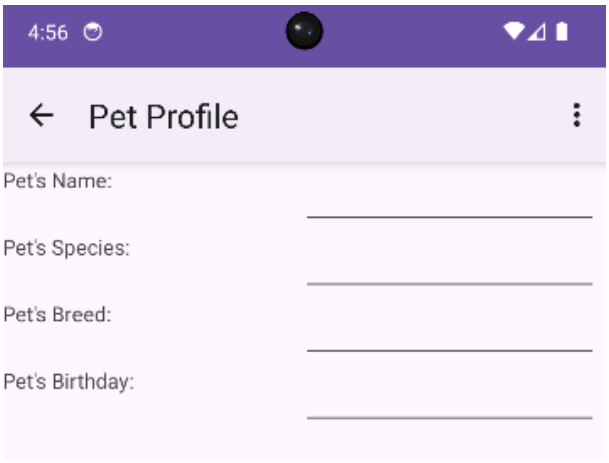


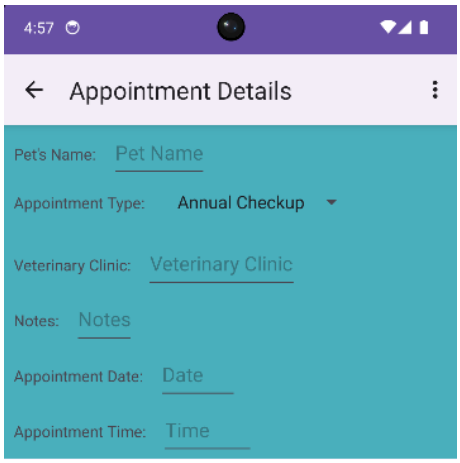
Upon entrance to the app, users are greeted with a list of all their pets and upcoming appointments. There is also a report section that allows users to search pets by pet name and to create reports that list any upcoming appointments. Used properly, the application allows the user to be proactive about managing ' ' y w timely checkups or dental care visits are missed. ' w '





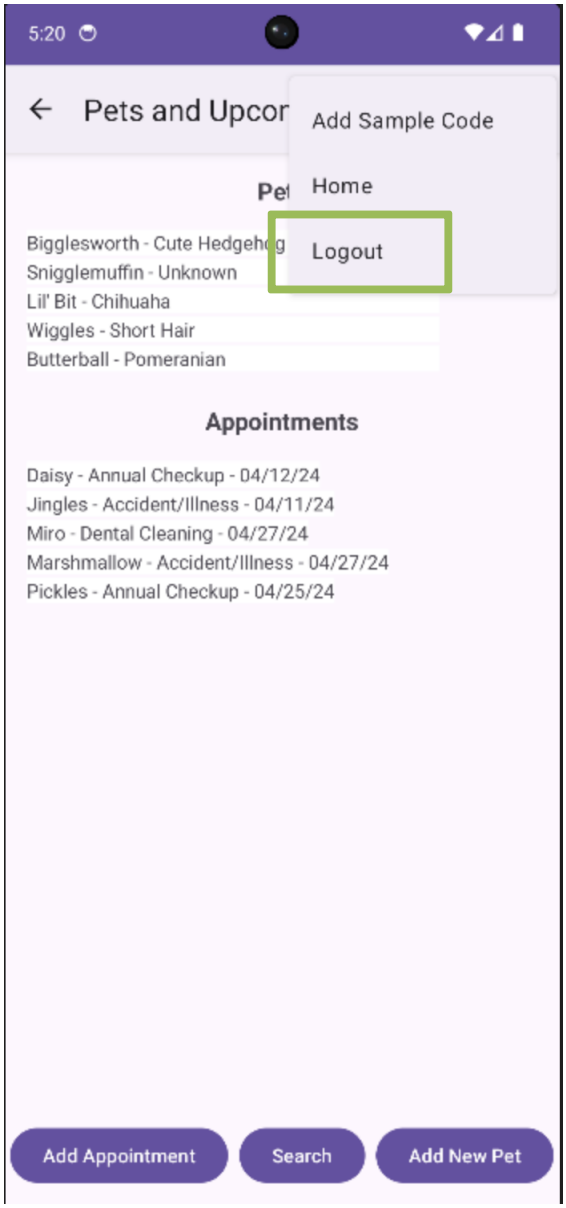
The key feature of the Pet Health Companion is managing and tracking pet healthcare appointments. The intuitive interface offers pet owners the ability to add, edit, and delete upcoming w 5 ' w ' ' w z z 5 ' z 5 ' ' z -legged ' w friends change in size. With the option of setting customized reminders, the application makes it very easy to check and monitor ' y ' w 7 '

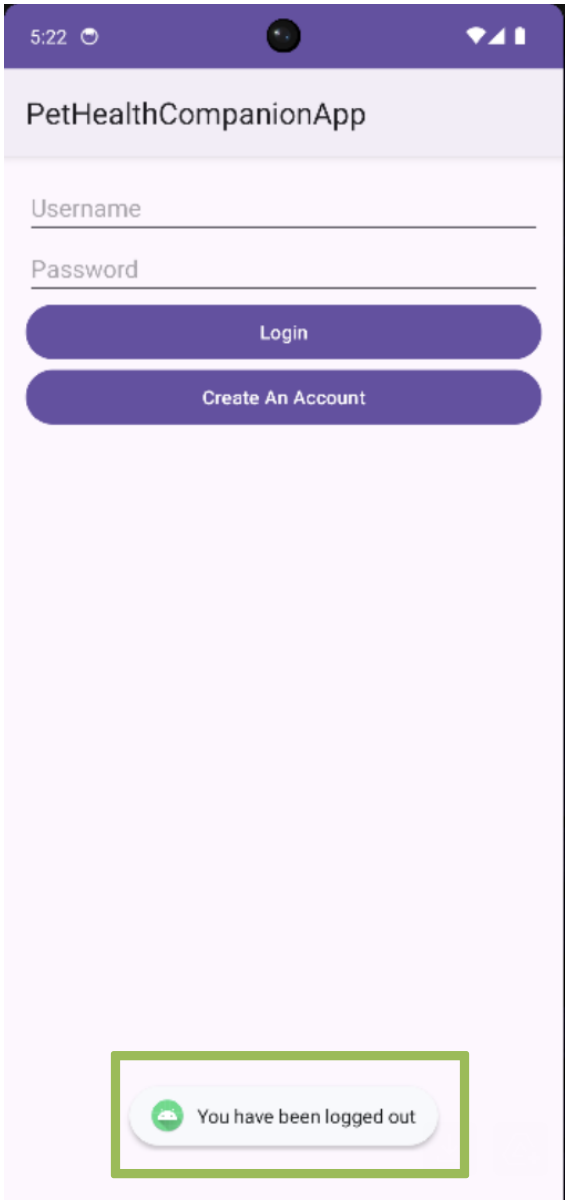




When users have finished monitoring their pet health care activities within the application, they are able to easily log out, and are provided with a message that verifies they have successfully logged out.

j ' ' w ' ata always 'remains secure. z ' ' z





Unit Test Plan

Introduction

Purpose

The Pet Health Companion's App unit tests are performed to ensure that the application is reliable and that it performs as intended. The tests are conducted to ensure that every part of the application has been tested systematically, that its functionality has been verified, and that it works together as an integrated, seamless whole.

Overview

The unit test process for the Pet Health Companion ensures that the application's codebase is reliant and robust. The plan uses a custom methodology to test specified parts of the application. Core functions such as adding and deleting pets and appointments are assessed to ensure their reliability and accuracy. These tests are valuable for checking the functionality ' ' w core features both in isolation and when integrated as a whole.

Addition and Deletion of Pet: The unit tests created for the Pet Health Companion mobile app provide a thorough assessment of the functionality for adding and deleting pets. These tests involve the submission of simulated input data - or a mock pet- to thoroughly assess and validate the application's mechanism for properly storing and removing data.

Addition and Deletion of Appointment: In regards to the addition and deletion of appointments, the unit tests utilize a mock appointment to test how the data process behaves, discerning and evaluating the precision of data persistence and elimination from the database. The implication of these tests results are ultimately used to provide pet holders with software that is trustworthy and reliable for w w ' ' ' w 7 '

Items

Development Environment: A development environment with Android Studio, the official Integrated Development Environment (IDE) for Android app development.

Database: The Room database, which is a part of the Android Jetpack library used for local data storage in Android applications. The Repository class is called by the unit testing scripts to serve as a buffer between the code in the program and the actual database. Mockito is used to create mock implementations of the DAO interfaces in the program. These mocks simulate the behavior of the actual DAOs without having to access the actual database. By doing this, the tests focus solely on verifying the functionality of the repository methods without having to execute actual database interactions, which may otherwise have an unintended impact on the program.

Test Framework: JUnit was used, which is a popular programming tool for writing and running unit tests in Java. Mockito was also used, which is a mocking framework used with JUnit to create mock objects for testing.

Application Source Code: To run the tests, clone the source code from the Git repository. As the Pet Health Companion is a MVP, the test scripts are already integrated into the codebase.

Features

Appointment Management: Test the addition and deletion of appointments from the database using a mock appointment.

Pet Management: Test the addition and deletion of pets from the database using a mock pet.

Deliverables

Test Scripts: The source code includes a set of test scripts, which are written using the JUnit testing framework, JUnit assertion library, and Mockito for integration testing.

Test Results: A report provides an overview of the test outcomes, encompassing successful tests and if present - any failures, accompanied by error messages.

Tasks

Prepare the development environment with Android Studio and configure the Room Database.

Ensure that test scripts for each outlined feature in the test plan, such as the Pet Records Manager Test script and the Appointment Records Manager test script, are available. As this is an MVP, the test scripts are not extensive and are integrated into the codebase.

Execute the test scripts using the Android Studio testing framework by selecting either the Pet Records Manager Test script or the Appointment Records Manager test script and running it. The results of the test will be displayed in the console for you to monitor and assess. Alternatively, you can read test results in an HTML

' w ' ' y w z ' ' ' y '

PetHealthCcompanionapp > app build > reports> tests > testDebugUnitTest folder.

Analyze the test results and identify any failed tests or unexpected behavior. The completed tests results would appear as below:

Package com.example.pethealthcompanionapp

all > com.example.pethealthcompanionapp

2	0	0	1.406s
tests	failures	ignored	duration

100% successful

Classes

Class	Tests	Failures	Ignored	Duration	Success rate
PetRecordsManagerTest	2	0	0	1.406s	100%

Generated by Gradle 8.2 at Apr 16, 2024, 12:55:39 PM

Class com.example.pethealthcompanionapp.PetRecordsManagerTest

all > com.example.pethealthcompanionapp > PetRecordsManagerTest

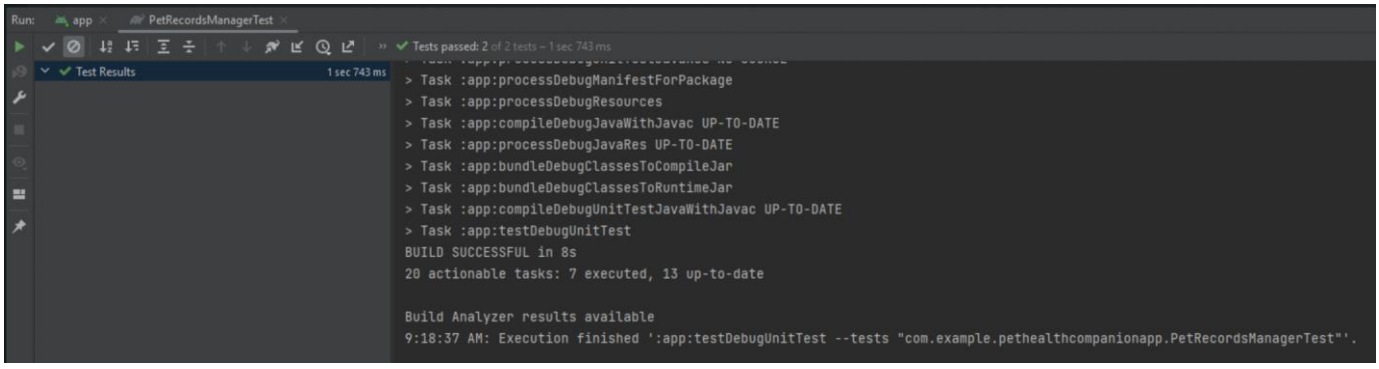
2	0	0	1.406s
tests	failures	ignored	duration

100% successful

Tests

Test	Duration	Result
testAddNewPetRecord	0.001s	passed
testDeletePetRecord	1.405s	passed

Generated by Gradle 8.2 at Apr 16, 2024, 12:55:39 PM



Package com.example.pethealthcompanionapp

all > com.example.pethealthcompanionapp

2	0	0	0.392s
tests	failures	ignored	duration

100% successful

Classes

Class	Tests	Failures	Ignored	Duration	Success rate
AppointmentRecordsManagerTest	2	0	0	0.392s	100%

Generated by Gradle 8.2 at Apr 16, 2024, 1:13:14 PM

Class com.example.pethealthcompanionapp.AppointmentRecordsManagerTest

all > com.example.pethealthcompanionapp > AppointmentRecordsManagerTest

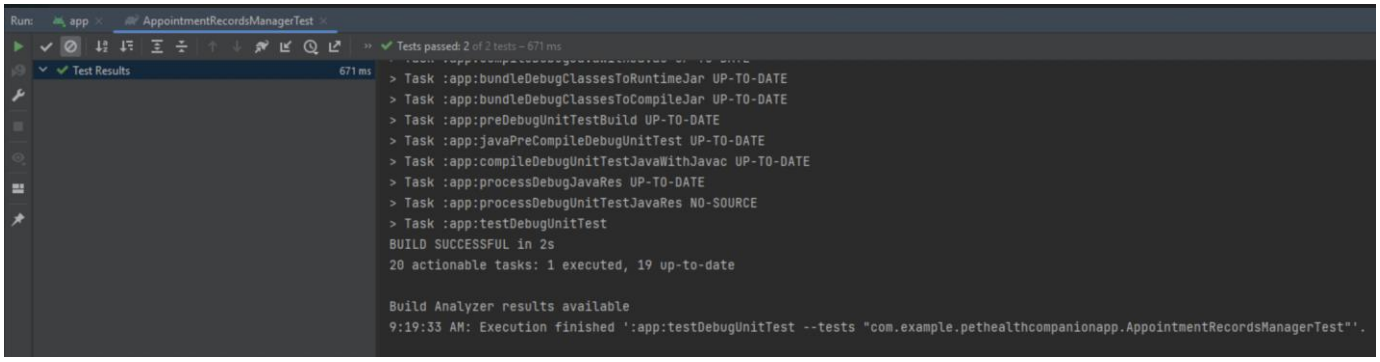
2	0	0	0.392s
tests	failures	ignored	duration

100% successful

Tests

Test	Duration	Result
testAddNewAppointmentRecord	0s	passed
testDeleteAppointmentRecord	0.392s	passed

Generated by Gradle 8.2 at Apr 16, 2024, 1:13:14 PM



Needs

Software Requirements: Within the build.gradle script, configure the required dependencies to conduct the unit testing, including Room, JUnit, and Mockito. These dependencies are essential for the unit tests to function properly.

Testing Tools and Libraries: JUnit and Mockito are used as the testing framework, along with the JUnit library for assertions. Ensure that the proper version of these frameworks are installed so to maintain compatability and stability within the Android developer environment.

Test Scripts: Ensure that the tests scripts within the original codebase are available and accessible so to execute the tests within the Android Studio developer environment. It may be necessary to employ a version control system such as Git or Github to handle code alterations and to monitor any adjustments made to the code as a result of the test results.

Pass/Fail Criteria

Appointment Addition:

- Pass: The appointment is successfully added to the repository and the method returns true.
- Fail: The appointment is not added to the repository or if the method returns false.

Appointment Deletion:

- Pass: The appointment is successfully deleted from the repository and the method returns true.
- Fail: the appointment is not deleted from the repository or if the method returns false.

Pet Addition:

- Pass: The pet is successfully added to the repository and the method returns true.
- Fail: The pet is not added to the repository or if the method returns false.

Pet Deletion:

- Pass: The pet is successfully added to the repository and the method returns true.
- Fail: The pet is not added to the repository or if the method returns false.

If a test fails during the testing process, the following measures are recommended to remediate the issue:

Identify the root cause: Investigate the failed test to find out the underlying cause of the problem. Analyze bug reports and error messages, logs, and other reports to discern the source of the issue.

Document the issue: Create a bug report that describes the problem thoroughly, including details such as the error messages, which tests failed, and any insights into the resolution of the problem.

Specifications

Displayed below are screenshots from the Pet Health Companion source codebase:

APPOINTMENT RECORDS MANAGER TEST:

```
package com.example.pethealthcompanionapp;

import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;
import static org.mockito.Mockito.mock;

import com.example.pethealthcompanionapp.dao.AppointmentDAO;
import com.example.pethealthcompanionapp.database.Repository;
import com.example.pethealthcompanionapp.entities.Appointment;

new *
public class AppointmentRecordsManagerTest {

    4 usages
    private Repository repository;
    2 usages
    private AppointmentDAO appointmentDAO;

    new *
    @Before
    public void setUp() {
        appointmentDAO = mock(AppointmentDAO.class);
        repository = new Repository(appointmentDAO);
    }
```

```
new *
@Test
public void testAddNewAppointmentRecord() {
    Appointment newAppointment = new Appointment( appointmentID: 0, petName: "Garfield", appointmentType: "Dental Cleaning", veterinaryClinic: "Animal Heart Hospital",
            notes: "Garfield will stay all day", appointmentDate: "04/20/24", appointmentTime: "2:00pm");
    boolean isAdded = repository.addNewAppointmentRecord(newAppointment);
    assertTrue(isAdded);
}

new *
@Test
public void testDeleteAppointmentRecord() {
    Appointment appointmentToDelete = new Appointment( appointmentID: 0, petName: "Buffy", appointmentType: "Annual Bloodwork", veterinaryClinic: "Creature Comforts",
            notes: "Fast 12 hours before appointment", appointmentDate: "04/19/24", appointmentTime: "1:00pm");
    repository.insert(appointmentToDelete);
    boolean isDeleted = repository.deleteAppointmentRecord(appointmentToDelete);
    assertTrue(isDeleted);
}
```

REPOSITORY:

```
1 usage  new *
public boolean addNewAppointmentRecord(Appointment appointment) {
    try {
        mAppointmentDAO.insert(appointment);
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}

1 usage  new *
public boolean deleteAppointmentRecord(Appointment appointment) {
    try {
        mAppointmentDAO.delete(appointment);
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
```

PET RECORDS MANAGER TEST:

```java
import org.junit.Test;
import static org.junit.Assert.*;
import static org.mockito.Mockito.mock;
import com.example.pethealthcompanionapp.dao.PetDAO;
import com.example.pethealthcompanionapp.database.Repository;
import com.example.pethealthcompanionapp.entities.Pet;

new *
public class PetRecordsManagerTest {

    4 usages
    private Repository repository;
    2 usages
    private PetDAO petDAO;

    new *
    @Before
    public void setUp() {
        petDAO = mock(PetDAO.class);
        repository = new Repository(petDAO);
    }

    new *
    @Test
    public void testAddNewPetRecord() {
        Pet newPet = new Pet( petID: 0, petName: "Buttons", species: "Dog", breed: "German Shepard", birthday: "01/02/2024");
        boolean isAdded = repository.addNewPetRecord(newPet);
        assertTrue(isAdded);
    }

    new *
    @Test
    public void testDeletePetRecord() {
        repository.insert(new Pet( petID: 0, petName: "Muffin", species: "Cat", breed: "Siamese", birthday: "02/16/24"));
        boolean isDeleted = repository.deletePetRecordByName( petName: "Muffin");
        assertTrue(isDeleted);
    }
}
```

REPOSITORY:

```java
1 usage  new *
public boolean addNewPetRecord(Pet pet) {
    try {
        mPetDAO.insert(pet);
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}

1 usage  new *
public boolean deletePetRecordByName(String petName) {
    try {
        mPetDAO.delete(petName);
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
```

Procedures

1. Test Preparation: Identified the key functionalities of the Pet Health Care Companion, wrote test cases to cover various scenarios, and defined the pass/fail conditions involved in each case.
2. Testing Environment Setup: Configured the test environment by adding JUnit and Mockito dependencies in Gradle. Ensured proper installation and configuration of the Android Studio development environment and Room.
3. Test Execution: Ran the tests within Android Studio and reviewed the test results. Analyzed any failed tests. When tests failed, the test cases and code for the application were reviewed and revised.
4. R y w F ' d z ' ' ' y 5 ' w ' code. In other cases, tracked the failures and recorded the problems identified during testing.

Results

After running the test scripts with the JUnit testing framework, the following test results for the Pet Health Companion were observed:

Test	What is Expected	What Happened	Pass?
User adds a new appointment	Appointment will be saved to the database	Mock appointment was saved to the database	Yes
User deletes an appointment	Appointment will be deleted from the database	Mock appointment was deleted from the database	Yes
User adds a new pet	Pet will be saved to the database	Mock pet was saved to the database	Yes
User deletes a pet	Pet will be deleted from the database	Mock pet was deleted from the database	Yes

Results Screenshot - all tests passed.

Package com.example.pethealthcompanionapp

all > com.example.pethealthcompanionapp

2	0	0	1.406s
tests	failures	ignored	duration

100% successful

Classes

Class	Tests	Failures	Ignored	Duration	Success rate
PetRecordsManagerTest	2	0	0	1.406s	100%

Generated by Gradle 8.2 at Apr 16, 2024, 12:55:39 PM

Class com.example.pethealthcompanionapp.PetRecordsManagerTest

all > com.example.pethealthcompanionapp > PetRecordsManagerTest

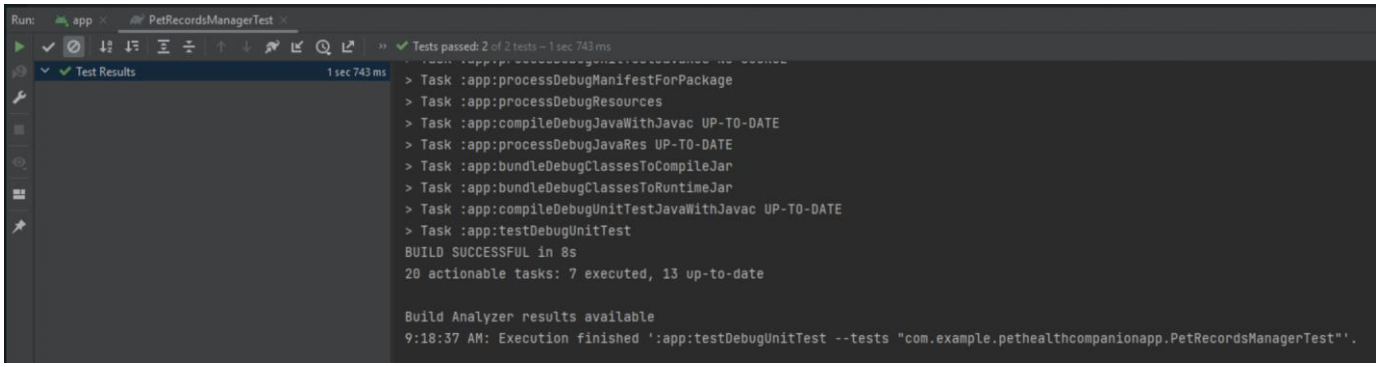
2	0	0	1.406s
tests	failures	ignored	duration

100% successful

Tests

Test	Duration	Result
testAddNewPetRecord	0.001s	passed
testDeletePetRecord	1.405s	passed

Generated by Gradle 8.2 at Apr 16, 2024, 12:55:39 PM



Package com.example.pethealthcompanionapp

all > com.example.pethealthcompanionapp

2	0	0	0.392s
tests	failures	ignored	duration

100% successful

Classes

Class	Tests	Failures	Ignored	Duration	Success rate
AppointmentRecordsManagerTest	2	0	0	0.392s	100%

Generated by Gradle 8.2 at Apr 16, 2024, 1:13:14 PM

Class com.example.pethealthcompanionapp.AppointmentRecordsManagerTest

all > com.example.pethealthcompanionapp > AppointmentRecordsManagerTest

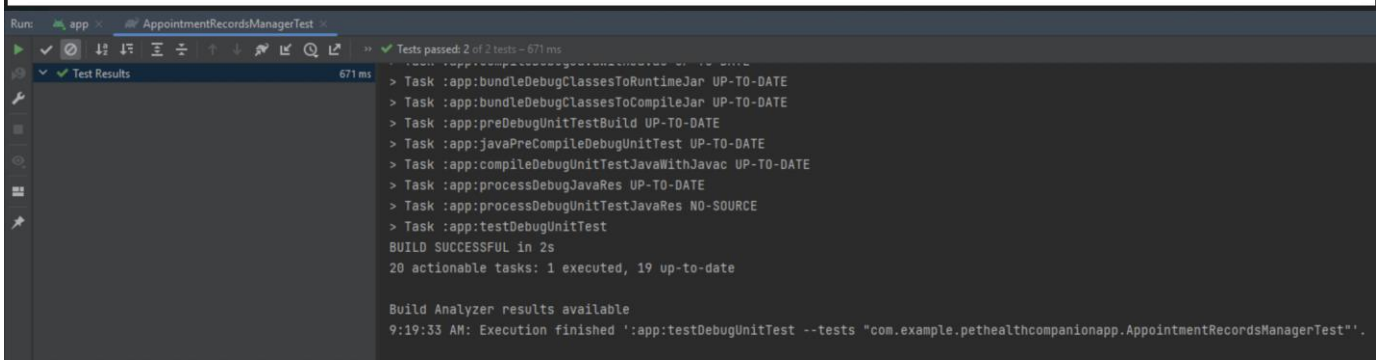
2	0	0	0.392s
tests	failures	ignored	duration

100% successful

Tests

Test	Duration	Result
testAddNewAppointmentRecord	0s	passed
testDeleteAppointmentRecord	0.392s	passed

Generated by Gradle 8.2 at Apr 16, 2024, 1:13:14 PM



As a result of the current successful execution of the unit tests, no modifications were made to the Pet Health Care Companion.

Hosted Web Application

Hosted Web Application Link:

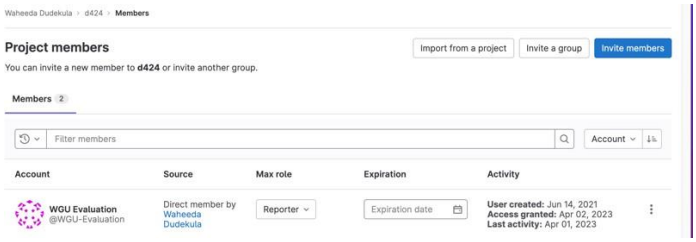
The Pet Health Care Companion is deployed on the Google Play Console and does not have a Hosted Web Application Link.

GitLab Repository & Branch History

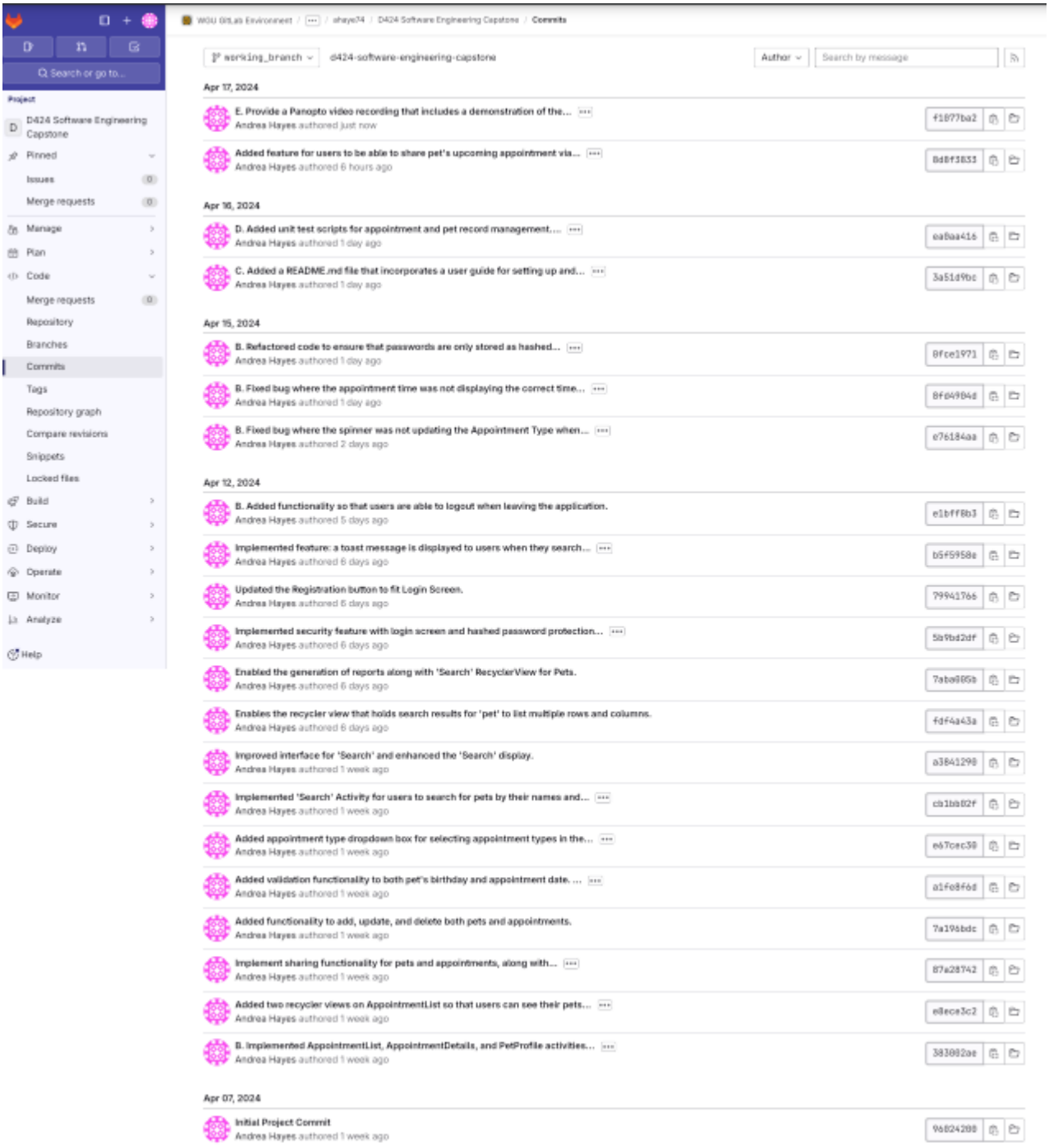
GitLab Repository Link:

https://gitlab.com/wgu-gitlab-environment/student-repos/ahaye74/d424-software-engineering-capstone.git

Added WGU-Evaluation as a member



GitLab Branch history:



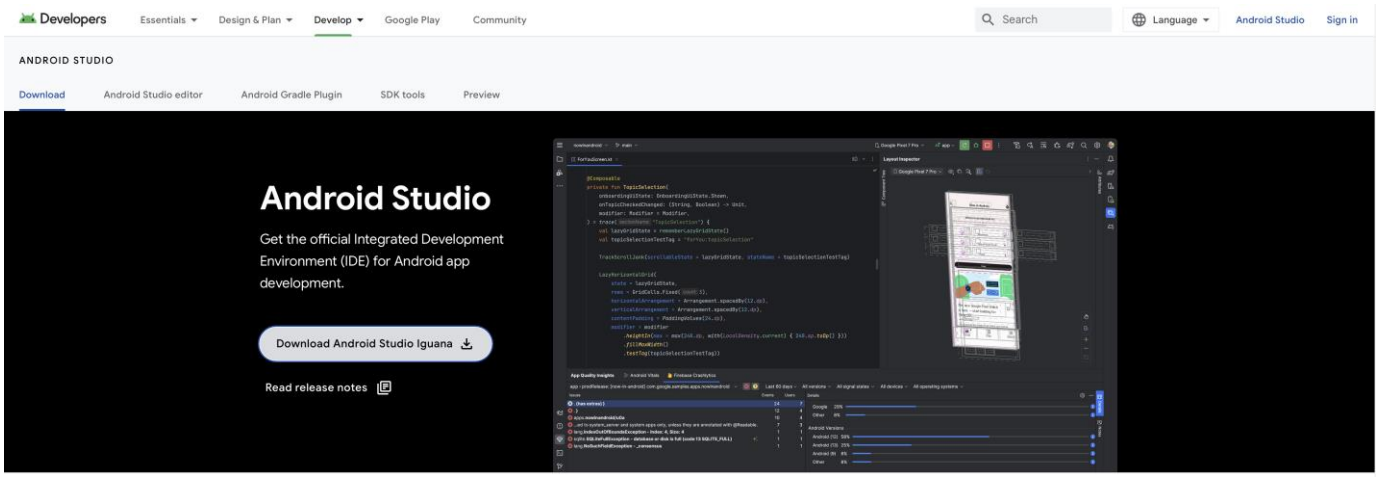
User Guide for Initial Setup & Running the Application

Introduction

This User Guide provides a step-by-step walkthrough for setting up the Pet Health Companion application. By following these instructions, users can configure their developer environment with ease and begin making any desired modifications within the application.

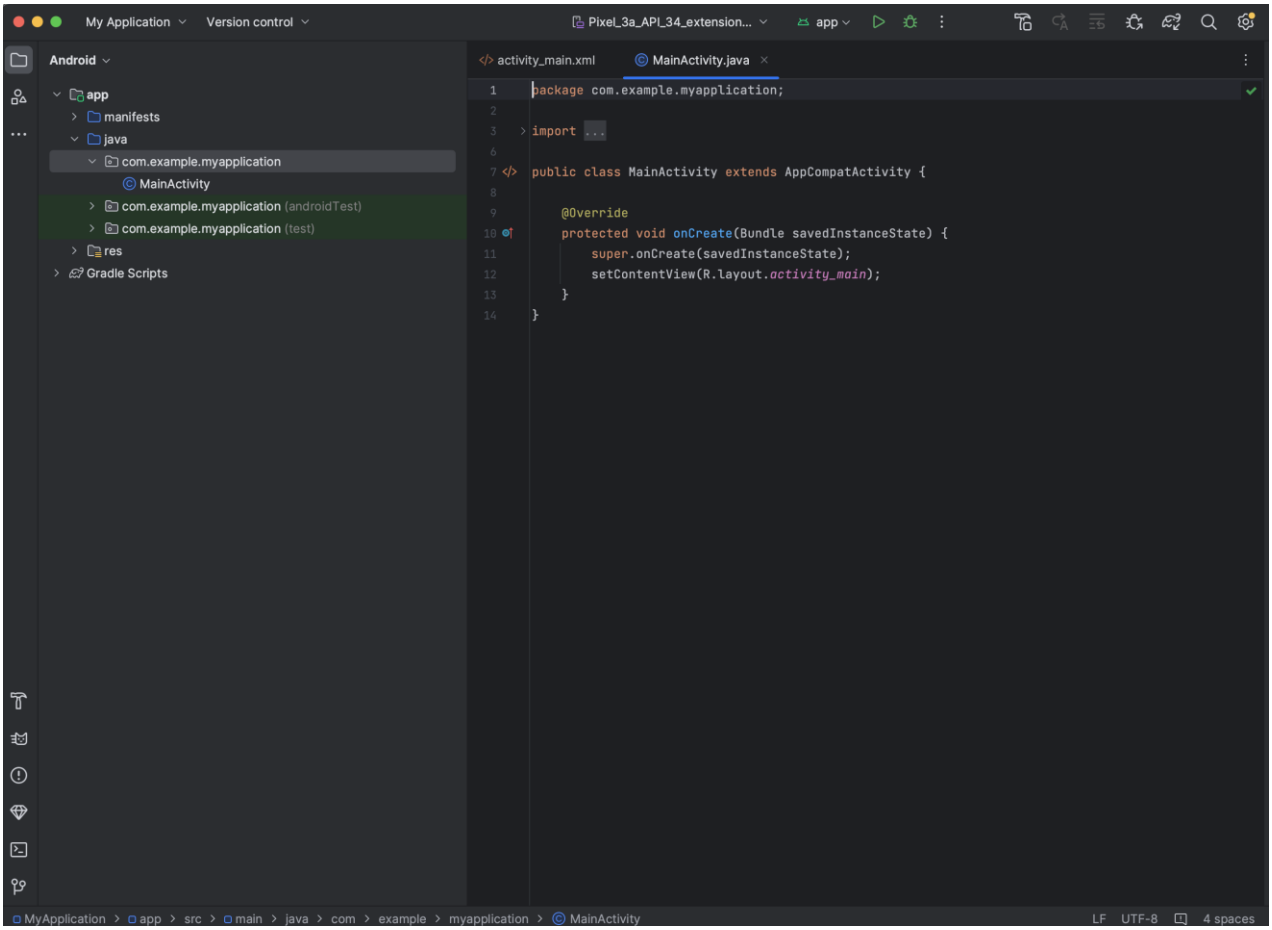
Installation of Android Studio:

1. Download Android Studio: Visit the official Android Studio website (https://developer.android.com/studio) and download the latest version of Android Studio that is congruent with your operating system (Windows, macOS, or Linux).



2. Install Android Studio: Once the download is complete, launch the installer and follow the on-screen instructions to install Android Studio onto your computer. Ensure that your computer possesses the system requirements that are specified by the installer.

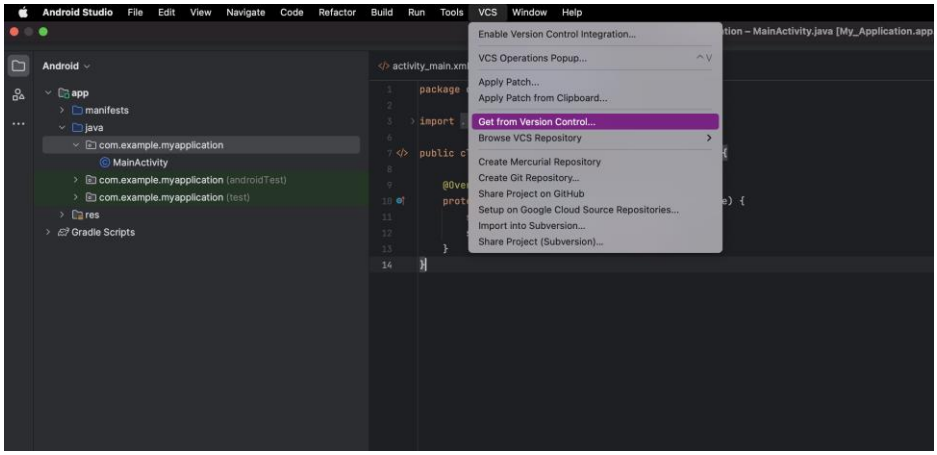
3. Configure Android Studio: Upon successful installation, open Android Studio. You may be prompted to import settings from a previous installation or customize your development environment. Follow the prompts to configure Android Studio according to your preferences.



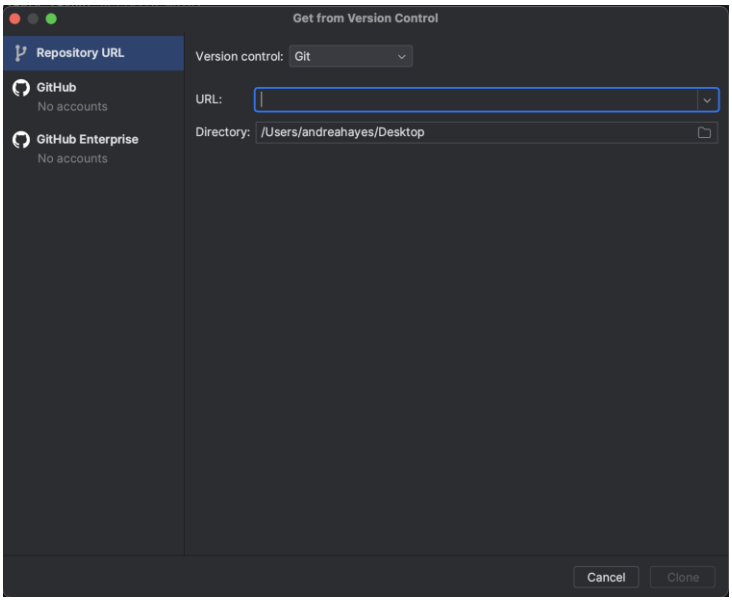
Cloning the Project from GitLab:

1. Access the GitLab Repository: Navigate to the GitLab repository hosting the Pet Health Companion project at https://gitlab.com/wgu-gitlab-environment/student-repos/ahaye74/d424-software-engineering-capstone.git.

2. Open Android Studio: Launch Android Studio and ensure that it is fully loaded before starting to make any modifications.
3. Select "Check out project from Version Control": In the Android Studio welcome screen, choose "Check out project from Version Control" and select "Git" from the dropdown menu.



4. Enter GitLab Repository URL: In the "Clone Repository" dialog, paste the URL of the GitLab repository hosting the Pet Health Companion project.
5. Choose Project Directory: Enter the directory where you want to clone the project on your local machine.



6. GitLab Authentication: If required, provide your GitLab credentials (username and password) to authenticate and access the repository.
7. Clone the Repository: To start the cloning process, click on the "Clone" button. Android Studio will download the project files from the GitLab repository and initiate the project structure.
8. Open the Project: Once the cloning process is complete, Android Studio will automatically open the project. Users can now examine project files, modify them, and run the application using the built-in emulator or a physical Android device.

Conclusion:

This User Guide has provided clear instructions for installing Android Studio and cloning the Pet Health Companion project from its GitLab repository. By following these instructions, users can easily set up their development environment up and start making modifications to the application.

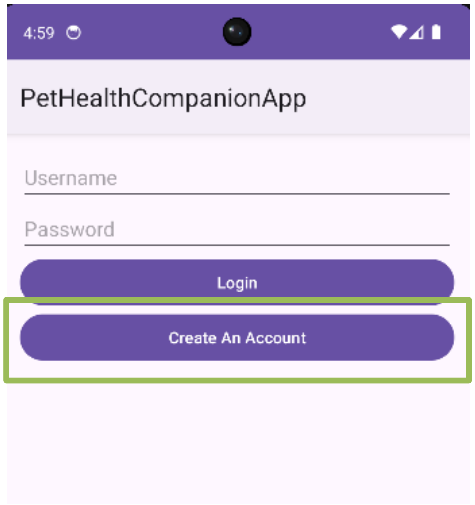
User Guide for Running the Application from User Perspective

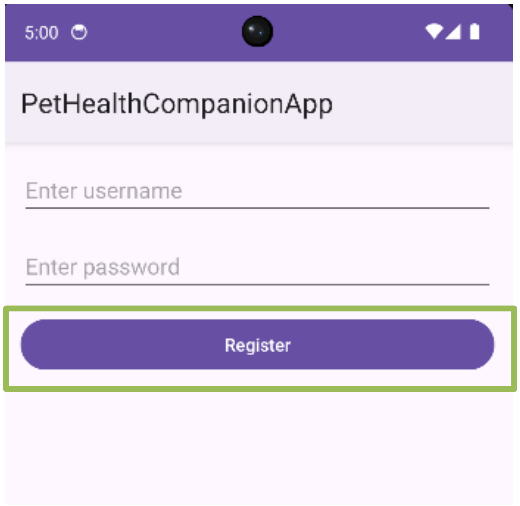
Introduction

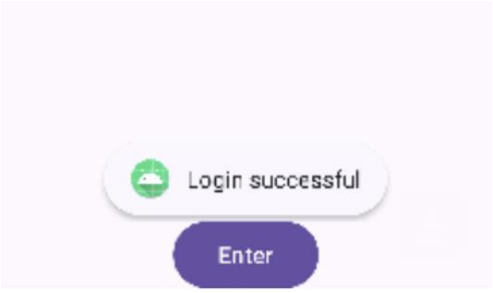
This user guide is designed to offer a comprehensive overview of all available feature within the Pet Health Companion application and to help users navigate through them.

Registration & Login

Before users are able to access the Pet Health Companion app, they must first register an w y y 7 ' ' j ' ' w ' w y y 5 ' y y ' ' h w ' w 7 ' ' S ' w ' w 5be re- w directed back to the Login page, where you can enter your username and password to enter the app. k ' y y ' ' 5 ' ' ' x '







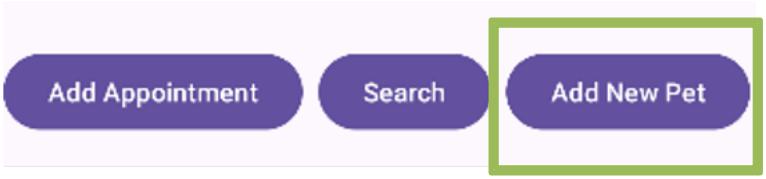
Dashboard page

e y ' z ' 5 ' ' w y w ' w ' '

this page, you can perform searches, add, edit and update appointments, and add, edit, and update pets.

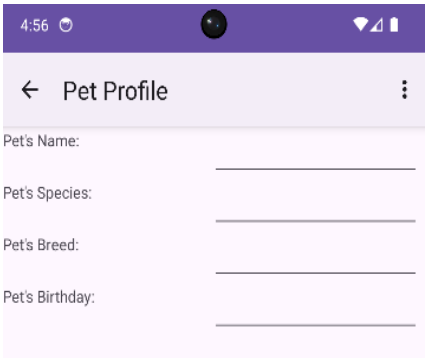
Add New Pet

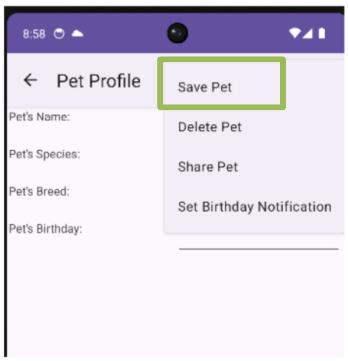
j ' w z z ' w ' ' 5 ' y ' ') N z z ' d ' f)
N ' w 7 '



P ' w ' z ' ' ' w ' w z ' '
' ' ' 5 ' ' w y w ' ' z

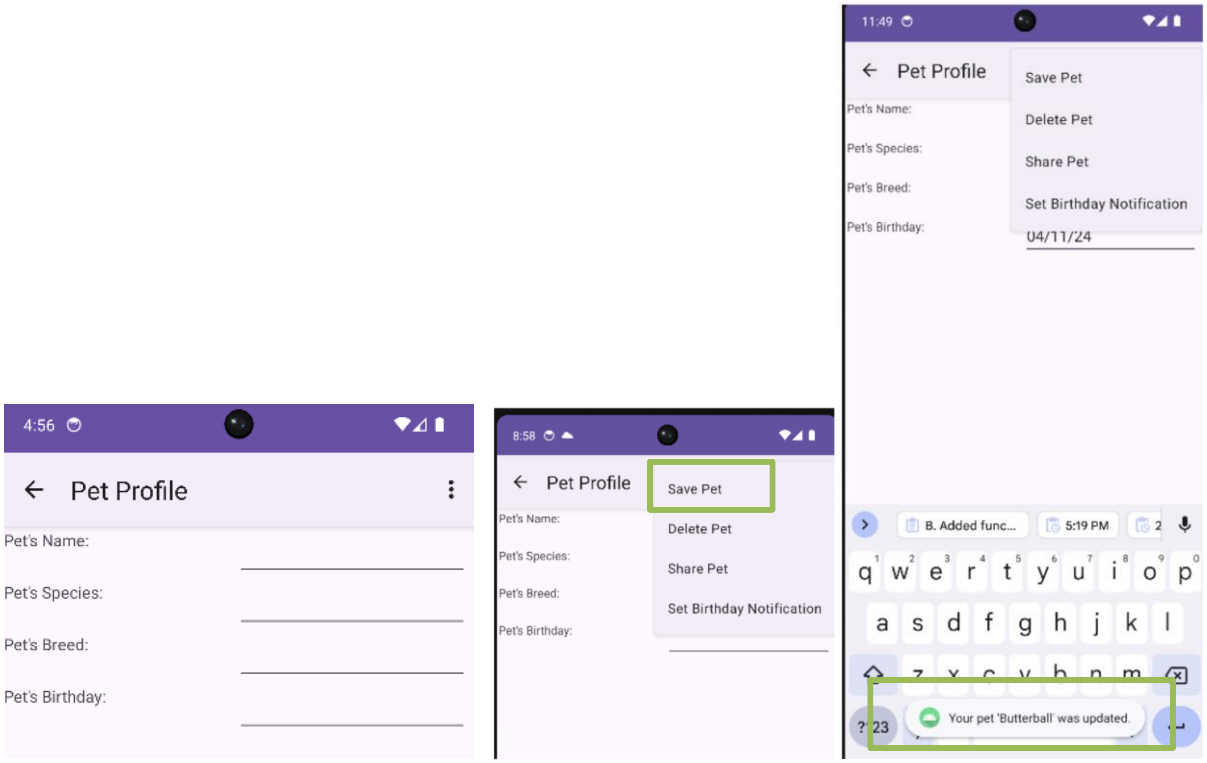
page, where you should now see the newly added pet in the list. In addition, you should see a message confirming that your pet has successfully been saved.





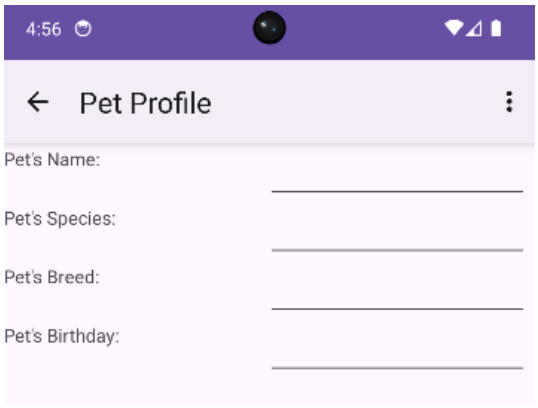
Edit Pet

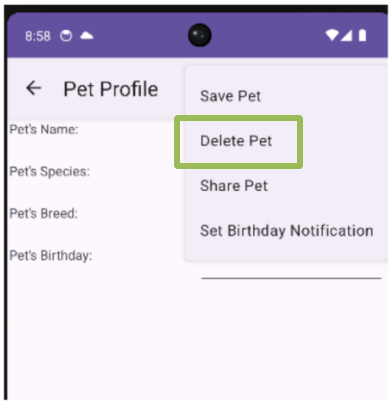
j ' z ' w ' ' 5 ' w w ' ' ' f ' ' ' ' z ' ' ' you will be redirected to the ' ' f ' f ' w 7 ' ' c w ' ' y w ' ' successful, you will see a message confirming that your pet has been updated. When you click the i w ' f ' x 5 ' '' w ' yf w ' w' z ' k 'y z where you will now see the updated information for the selected pet.

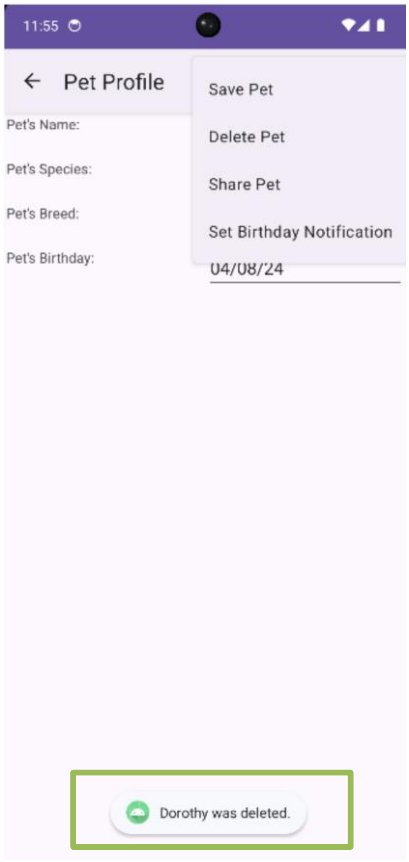


Delete Pet

j ' z ' w ' 5 ' ' ' ' f ' w z ' k y w ' ' 7 ' j ' ' w ' ' ' ' f R ' f ' 7 ' m ' ill show a confirmation message that your pet has been deleted, will remove the pet from the list, and will return you to the z w z ' f ' w z ' k y ' N ' w





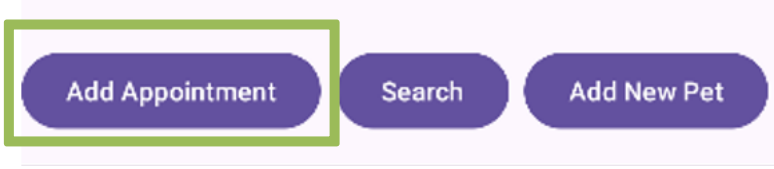


Appointments page

Add Appointment

j ' w z z ' w ' ' w 5 ' y ' ' N z z '

f ' w z ' k y ' N ' w 7 '

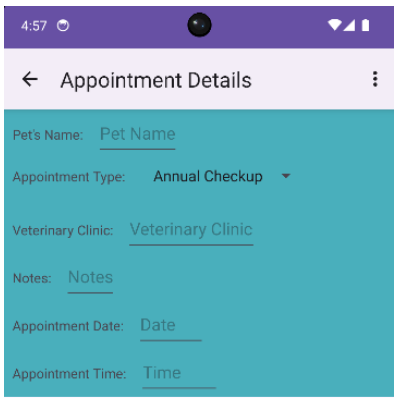


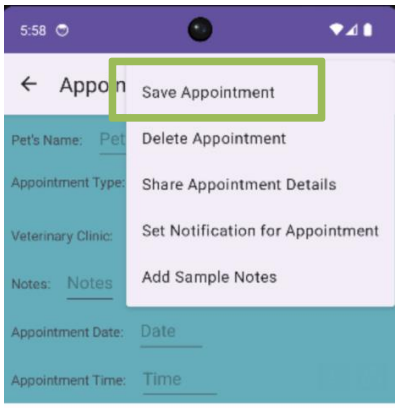
Fill out the requested information and ensure that all fields are completed on this page.

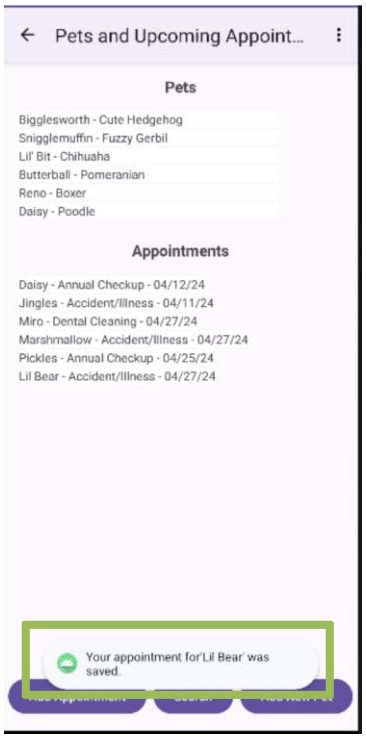
k ' y y ' ' i w ' N ' '

f ' w z ' k y ' N newly added appointment '

in the list along with a message confirming that your appointment has been saved.

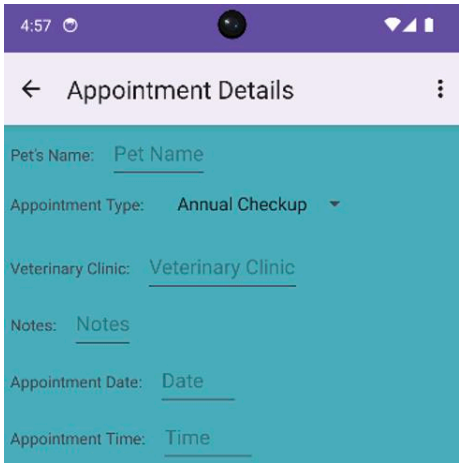


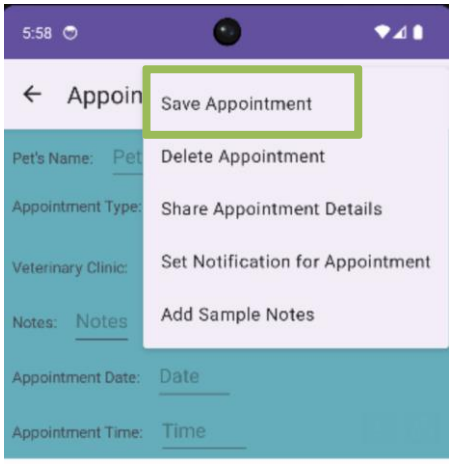


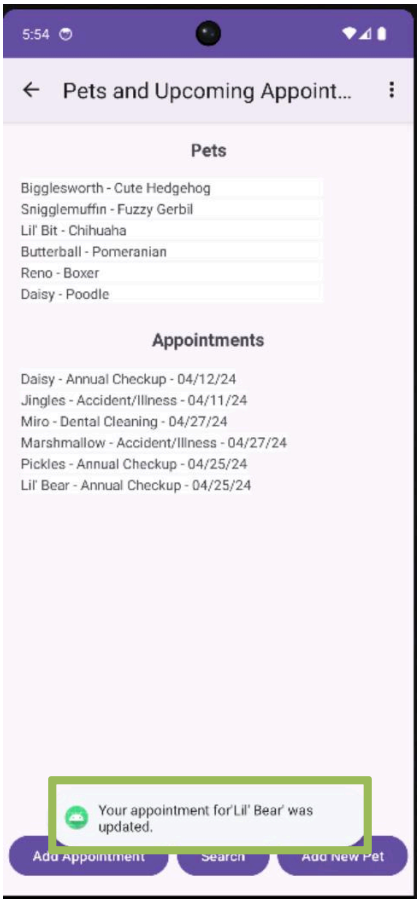


Edit Appointment

j ' z ' w ' ' w 5 ' w w ' and locate the appointment you would like to update. Click on the appointment in the Appointment ' ' w z ' ' ' x ' z y z ' ' ' z ' w z ' ' option. If successful, you will see a message y ' w ' ' w ' x ' z x 5 ' ' w y w ' ' z y ' ' ' will now see the updated information for the selected appointment.



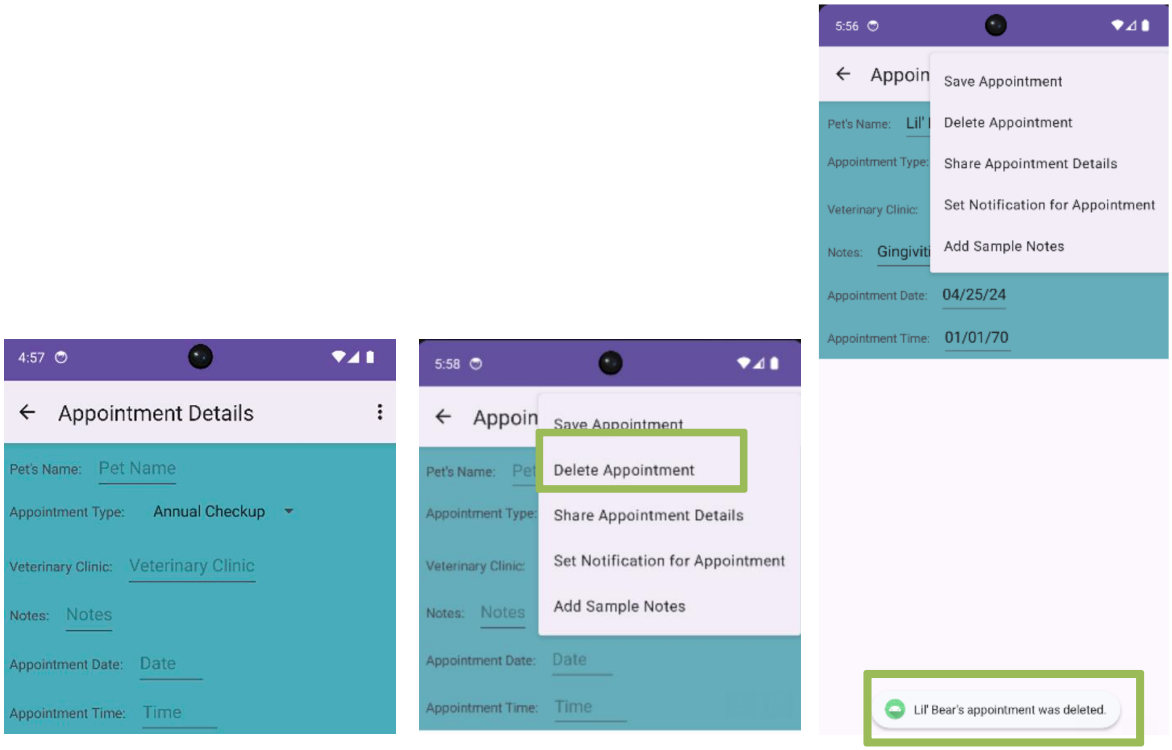




Delete Appointment

j ' z ' w ' w 5 ' ' ' ' f '
w ' ' w ' ' 7 ' j ' ' w
' 5 ' y ' ' R ' N ' N '

the application will show a confirmation message that your appointment has been deleted and will remove the appointment from the list. You will then be re-directed to the ' ' ' f k y ' N , where the deleted appointment should no longer be visible.



Search & Reports page

e ' ' i w y ' w 5 ' ' y w ' w ' w

' ' w z ' ' y ' w 7 ' O

z y z ' ' ' i w y ' w 5 ' ' 'w 7 " y w ' '

' w ' ' ' ' ' z ' ' ' w y 5

w z ' w ' z w ' w ' w x '

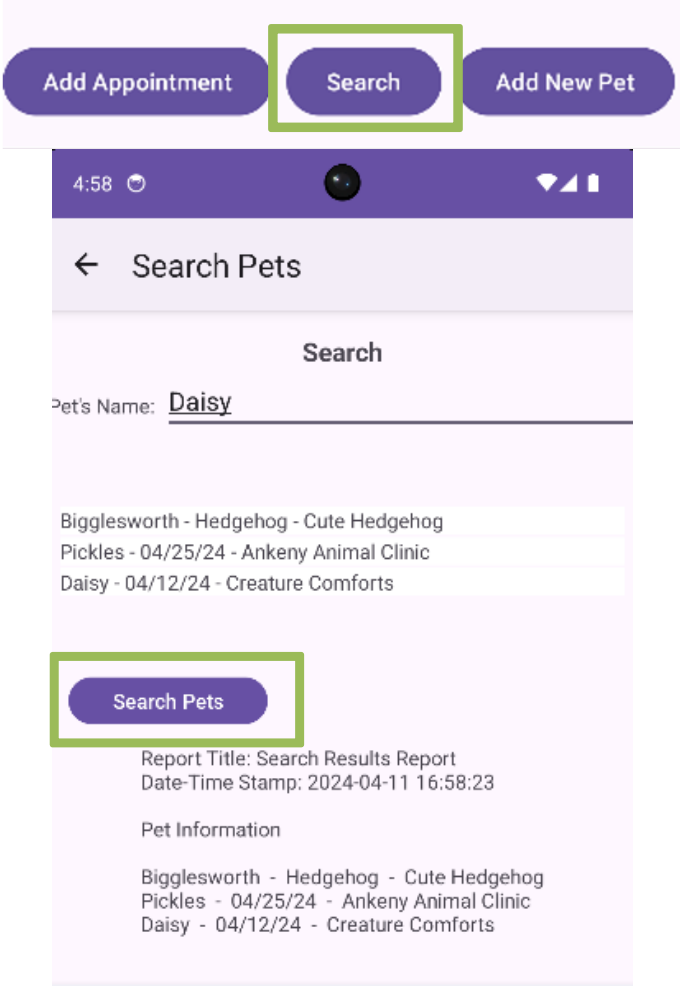
the application will display ' ' w 5 ' x z 5 ' w z ' y

w ' ' ' 5 ' ' ' ' z

appointment, and the veterinary clinic where your appointment is scheduled. If you have added all

' ' w ' ' ' w 5 ' ' w y w

upcoming appointment information in the generated report.



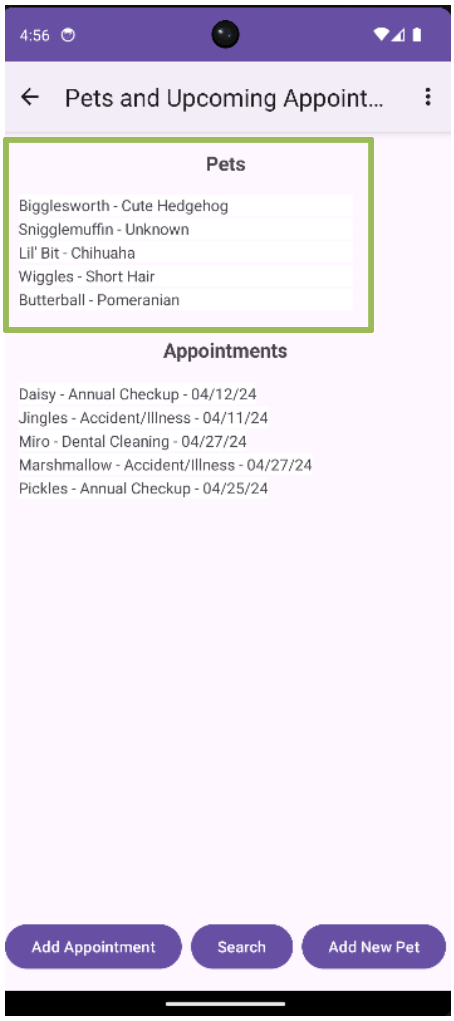
Share Pet Information

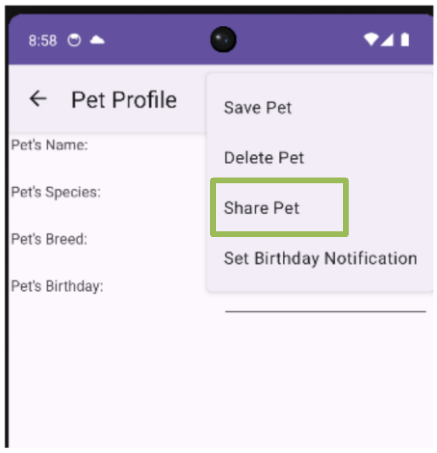
j ' w ' w ' w x ' ' 5 ' y

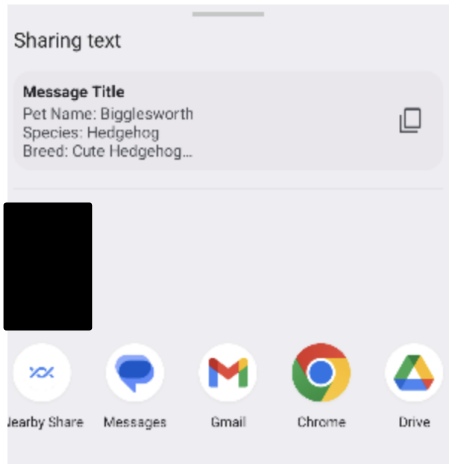
k y ' N ' f w 7 ' ' j ' ' z y

w ' w x ' ' ' 7w y' W ' ' ' ' x w5 ' 5y' w y

would like to share the information (either through notepad, email, or text).







Share Appointment Information

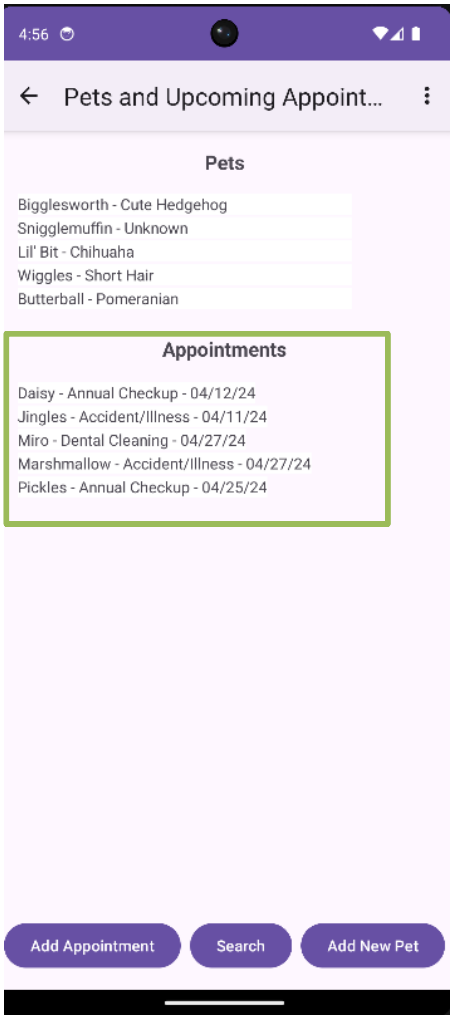
j ' w ' w ' w x ' ' ' y

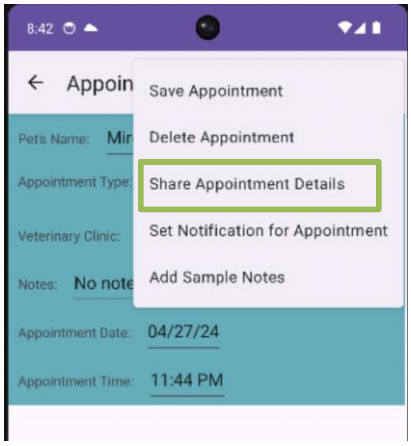
' ' w ' ' ' f ' w z ' k y ' N

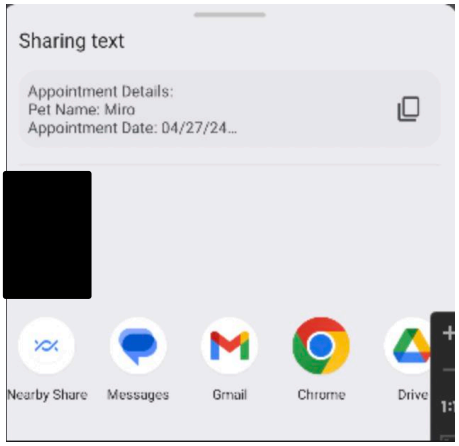
R w ' w 5 ' ' ' ' ' w z ' 7ẇ ' W

i w ' N ' R w ' 5 ' w z ' '

through notepad, email, or text).

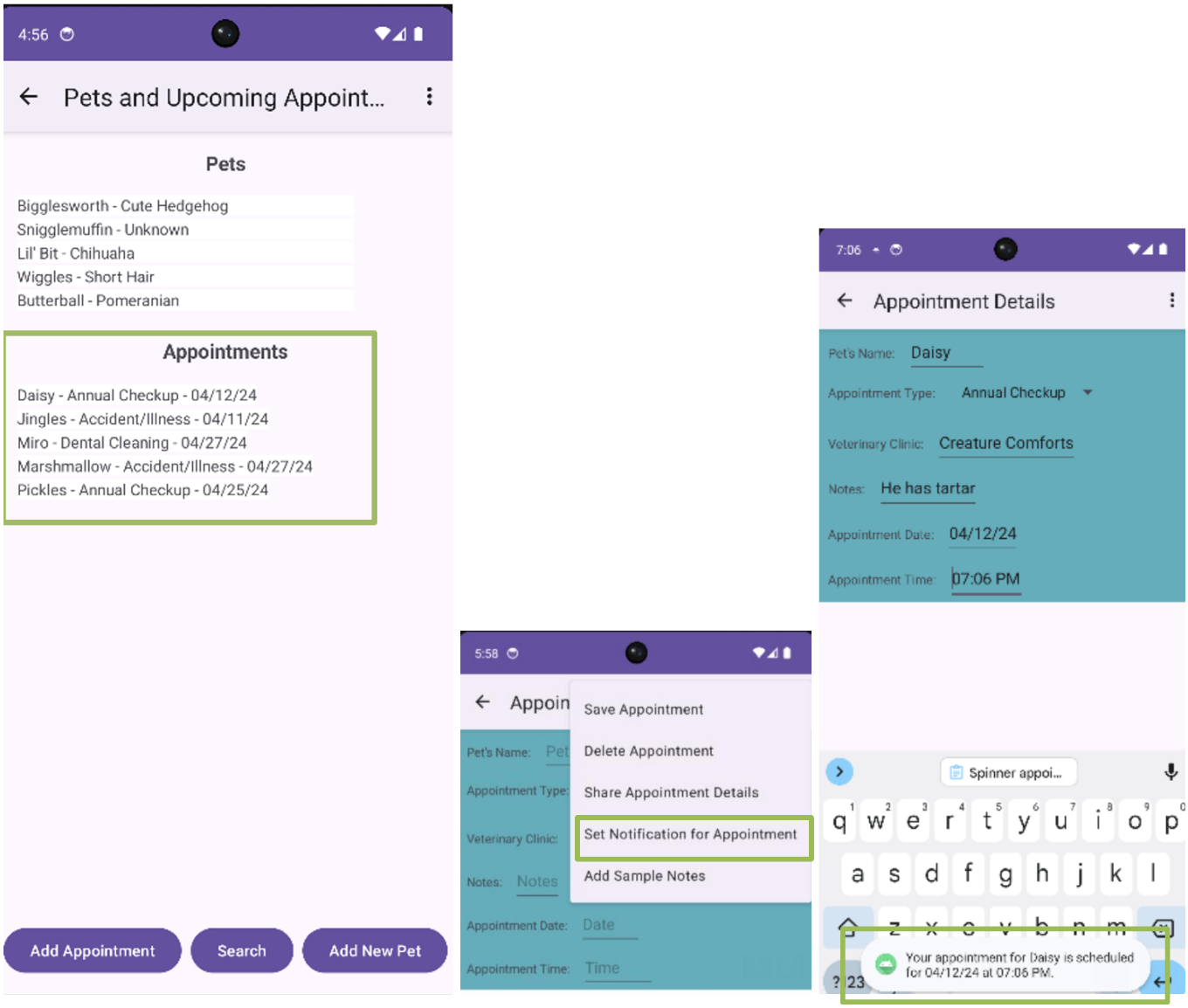






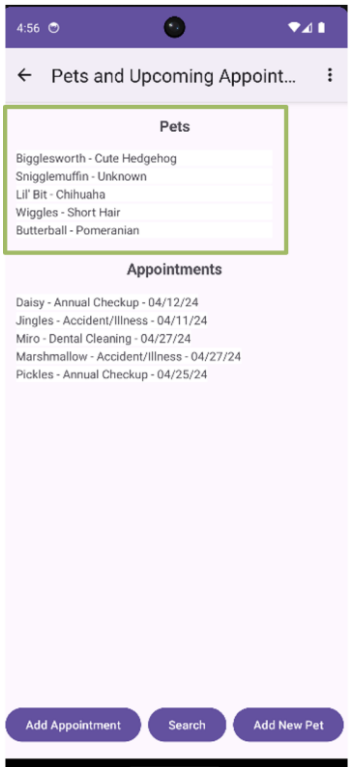
Set Appointment Notifications

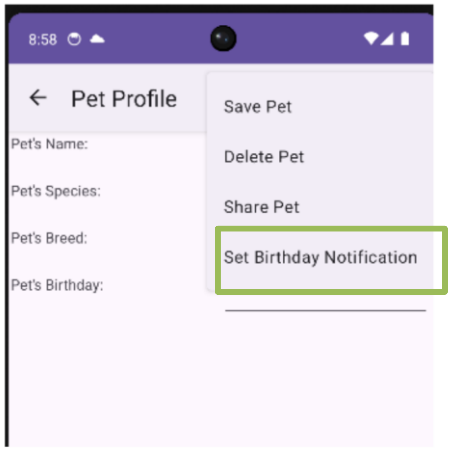
With the Pet Health Companion App, you can set y w ' ' ' y ' ' w ' ' ' N ' y ' ' z z ' ' ' i ' d appointment, you will receive a notification reminding you.



Set Birthday Notifications

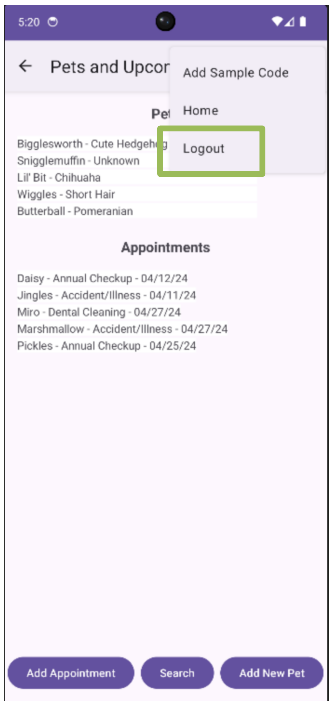
m ' ' f ' V w ' P w ' N 5 ' ' y w '
' ' ' ' f ' b ' ' ' f ' w z
' ' i ' O z w ' d ' x y w z w 5 '7 ' ' m' '
' ' ' o ' d ' x ' z w ' ' ' w '

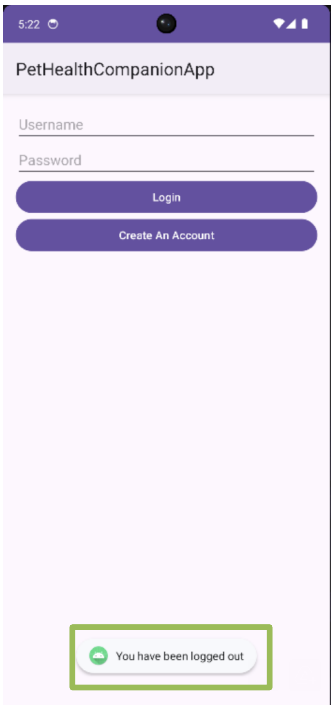




Log Out

When you are all finished using the Pet Health Companion App, you can easily log out so to secure ' ' z w w 7 ' ' W ' ' f ' w z ' k y ' N z z ' 5 ' y ' directed to the Login page and will receive a ' ' o confirmation message that you have been logged out.





Panopto Video Link

Name:

D424CapstonePetHealthCompanionTask3

Link:

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=e3a63597-dcec-4d6c-a6fe-b1550143665e