

Deliverable 5

Team Name: FurGuardian

Project Name: Pet Wellness App

Team Group: 8

Student Names and ID's: Justin Chipman – N01598472

Imran Zafurallah - N01585098

Zane Aransevia - N01351168

Tevadi Brookes - N01582563

Name	Student Id	Github Id	Signature	Effort
Justin Chipman	N01598472	Chipman8472	Justin C	100%
Imran Zafurallah	N01585098	ImranZafurallah5098	Imran Z	100%
Zane Aransevia	N01351168	Zayne917	Zane A	100%
Tevadi Brookes	N01582563	TevadiBrookes2563	Tevadi B	100%

GitHub Link: <https://github.com/Chipman8472/FurGuardian.git>

Sprint 5 Goals

Tasks from Scrum Dashboard

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

Heart rate processing - update libraries and framework to use stable versions to address know vulnerabilities

Optimized UI Design - create and enhance automated testing for existing functionality to ensure stability.

History Tracking - Update and clean up existing code to improve readability, maintainability, and scalability without changing its external behavior.

The work completed by each member

Justin: I worked on fixing our google sign in as it seems the logic was broken at some point. I updated the database logic by using FirebaseAuth which allowed better storage of our users and increased our security. I implemented our 20 junit tests using both standard methods and roboelectrics. I implemented an eye toggle button for the password on our login page to toggle between plain text and "***". As well I just overall worked on fixing imperfections across.

Tevadi: This sprint I added different functionalities that were not there before. I added a snackbar to the reminders fragment, as well as assisted Imran with the clean up of the code. My main focus this sprint

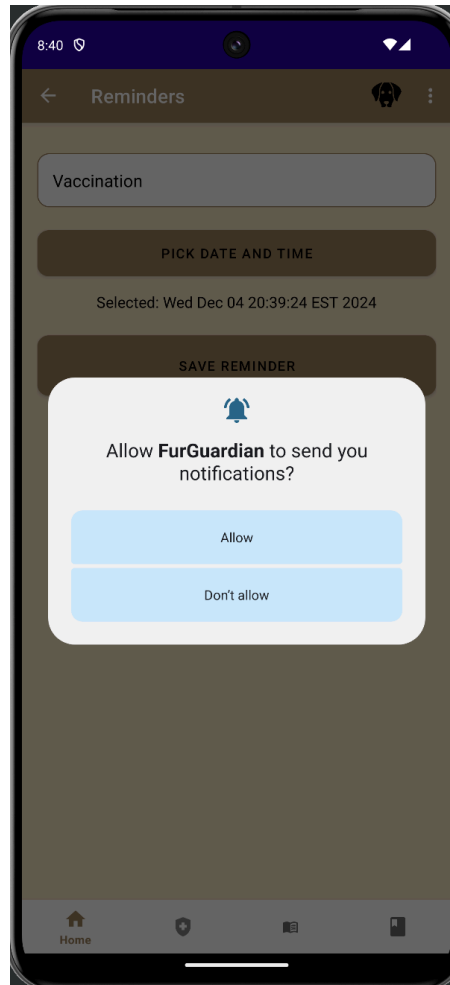
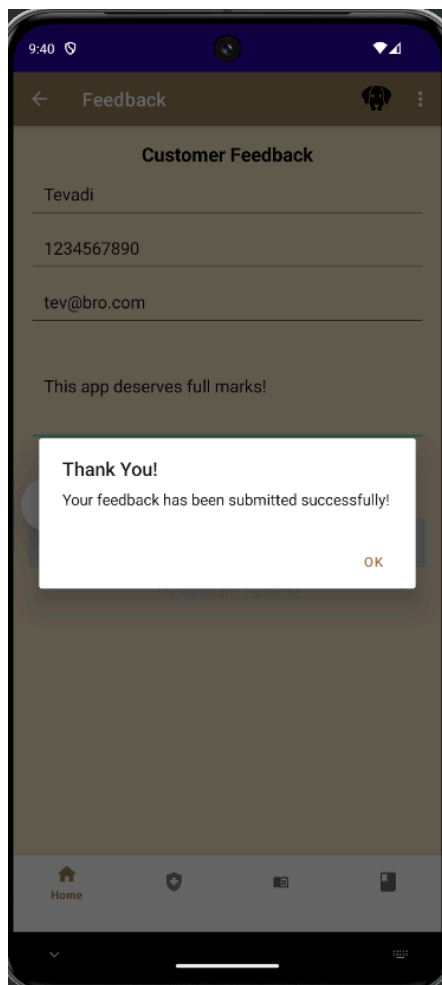
Imran: Worked on cleaning up the java class codes, taking out and adjusting deprecated codes and unused imports for all classes and making sure the app runs with the adjusted codes

Zayne:

C4 Model Component Diagram



Screenshot showing AlertDialog, Snackbar, Toast & Notification



8:35

FurGuardian

Username


tev@bro.com

Password


☐ Remember me?

LOGIN

OR

 Sign in

REGISTER

 The supplied auth credential is incorrect, malformed or has expired.

FurGuardian

Reminder • now

Check-up at Thu Dec 05 20:50:09 EST 2024

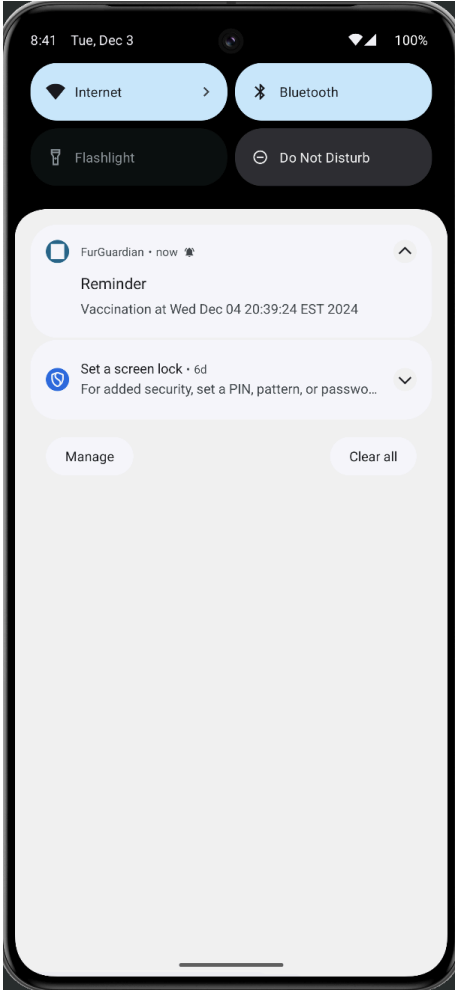
Check-up

PICK DATE AND TIME

Selected: Thu Dec 05 20:50:09 EST 2024

SAVE REMINDER

Thank you! Your reminder has been set!



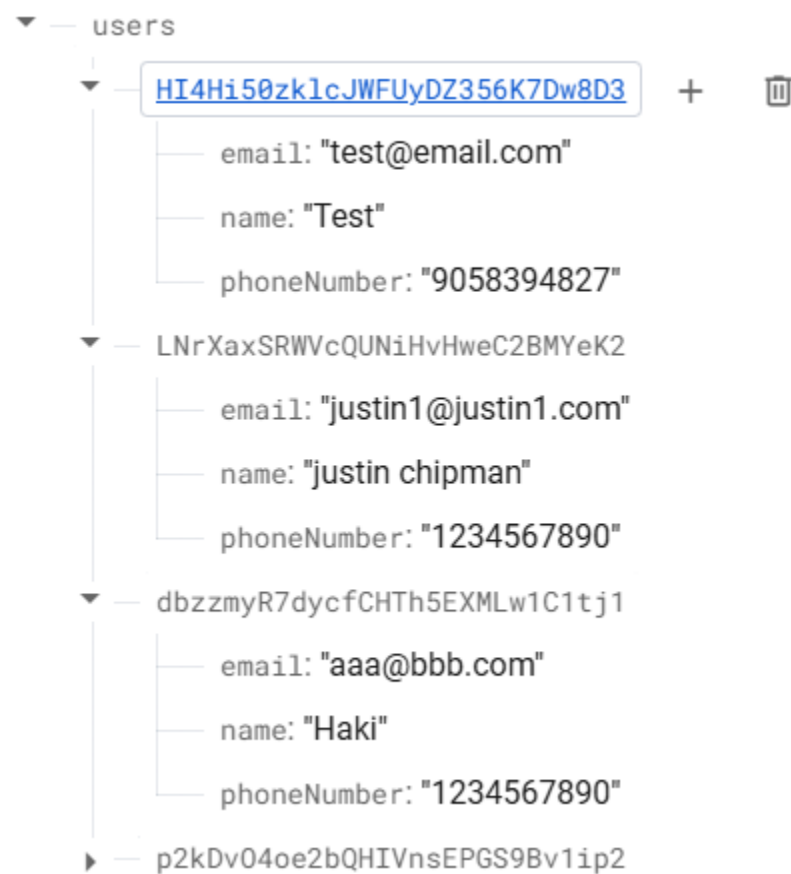
Screenshot of feedback in DB

- ▼ — -OC6W_C0RJ5PycXZKXQ6
 - comment: "nice app"
 - deviceModel: "sdk_gphone64_x86_64"
 - email: "justin@justin.com"
 - name: "Tev"
 - phone: "4166602648"
 - rating: 5
- ▼ — -OCfwhpcXgPLJ0P70krB
 - comment: "sdfsd"
 - deviceModel: "sdk_gphone64_arm64"
 - email: "afa@abc.com"
 - name: "asdfas"
 - phone: "1234567890"
 - rating: 3
- ▶ — -OCkWw77Es0ecAaYWrv8
- ▼ — -0DEhIuCQpMA7oWKJoe2
 - comment: "very good app"
 - deviceModel: "sdk_gphone64_x86_64"
 - email: "justin@justin.com"
 - name: "justin"
 - phone: "5678900981"
 - rating: 4.5

Screenshots of Data being stored in DB



Screenshot of user data in DB



Screenshot of users

Search by email address, phone number, or user UID					Add user	↺	⋮
Identifier	Providers	Created ↓	Signed In	User UID			
tev@bro.com	✉	Dec 3, 2024	Dec 3, 2024	p2kDvO4oe2bQHIVnsEPGS9B...			
test@email.com	✉	Dec 3, 2024	Dec 3, 2024	HI4Hi50zklcJWFUyDZ356K7D...			
aaa@bbb.com	✉	Dec 1, 2024	Dec 3, 2024	dbzzmyR7dyfcCHTh5EXMLw1...			
justin1@justin1.com	✉	Dec 1, 2024	Dec 2, 2024	LNrXaxSRWVcQUNiHvHweC2...	📄	⋮	
chipman32@yahoo.ca	🌐 ✉	Dec 1, 2024	Dec 3, 2024	uCuxLDUxyFVQ7FYUO10bb9A...			
Rows per page: 50					1 – 5 of 5	⏪	⏩

Screenshots of Tests

✓ InputValidatorTest	28 ms
✓ testValidPassword	20 ms
✓ testInvalidPassword_NoUppercaseLetter	1 ms
✓ testInvalidPhoneNumber_TooShort	1 ms
✓ testInvalidPassword_NoSpecialCharacter	0 ms
✓ testInvalidPhoneNumber_ContainsLetters	1 ms
✓ testValidEmail	1 ms
✓ testValidPhoneNumber	1 ms
✓ testInvalidPassword_LengthOutOfBounds	1 ms
✓ testInvalidEmail_NoDomain	1 ms
✓ testInvalidEmail_NoAtSymbol	1 ms

✓ LoginActivityTest	8 sec 749 ms
✓ testLoginButtonClickDoesNotCrash	6 sec 957 ms
✓ testRegisterButtonNavigatesToRegistrationActivity	274 ms
✓ testActivityInitialization	243 ms
✓ testUIElementsAreInitialized	196 ms
✓ testLoginNavigatesToMainActivity	208 ms
✓ testGoogleSignInButtonExists	197 ms
✓ testEmailValidationLogic	181 ms
✓ testRememberMeCheckboxDefaultState	170 ms
✓ testPasswordFieldIsPasswordType	184 ms
✓ testEmptyEmailAndPasswordShowsToast	139 ms

How Did we address technical debt

One instance of technical debt we faced was when we initially created our database we did not store our database logic correctly in a separate package and class we put it inside of our UI elements directly. On top of that we did not use Authentication right away we simply just stored and retrieved data in plain text.

How we tackled this is by refactoring our code to have different database models such as UserModel in which that class did all the interacting with the database user table. We did that same thing for our data, pet, and feedback table in the database.

Then we upgraded our code to use FirebaseAuth so that we can store users correctly without using plain text and also incorporate other sign in methods such as Google Sign in.

Refactored Code

Before refactoring our health data simulation code the health markers were completely random. After refactoring we decided to make the values represent more realistic values. So we refactored the code to mimic more real life scenarios. The code has been commented to show how it represents real life examples

After Refactor

```
// Method to simulate more realistic health data
public Map<String, Object> simulateData() {
    // Heart rate (bpm): Normal resting heart rate is between 60 and 100
    int heartRate = generateHeartRate();

    // Respiratory rate (breaths per minute): Normal is between 12 and 20
    int respiratoryRate = generateRespiratoryRate();

    // Steps: Average steps per day can range between 4000 and 12000, depending
    on activity
    int steps = generateStepCount();

    // Distance (km): Daily distance walked based on steps, average stride
    length is 0.762 meters
    double distance = steps * 0.000762; // Convert steps to km (0.762m per
    step)

    // Sleep hours: 7-9 hours, but it can vary slightly for shorter or longer
    sleepers
    double sleepHours = generateSleepHours();

    // Weight (kg): Slight fluctuations based on meals, hydration, and activity
    double weight = generateWeight();
}
```

```

    // Prepare the data map to store in Firebase
    Map<String, Object> data = new HashMap<>();
    data.put("heartRate", heartRate);
    data.put("respiratoryRate", respiratoryRate);
    data.put("steps", steps);
    data.put("distance", distance);
    data.put("sleepHours", sleepHours);
    data.put("weight", weight);

    return data;
}

// Simulates realistic heart rate
private int generateHeartRate() {
    double activityFactor = Math.random();
    if (activityFactor < 0.2) { // Resting
        return 60 + random.nextInt(10); // 60-69 bpm
    } else if (activityFactor < 0.7) { // Light activity
        return 70 + random.nextInt(20); // 70-89 bpm
    } else { // High activity
        return 90 + random.nextInt(20); // 90-109 bpm
    }
}

// Simulates realistic respiratory rate
private int generateRespiratoryRate() {
    double activityFactor = Math.random();
    if (activityFactor < 0.3) { // Resting
        return 12 + random.nextInt(3); // 12-14 breaths/min
    } else if (activityFactor < 0.8) { // Light activity
        return 15 + random.nextInt(4); // 15-18 breaths/min
    } else { // High activity
        return 19 + random.nextInt(2); // 19-20 breaths/min
    }
}

// Simulates realistic step count
private int generateStepCount() {
    double activityLevel = Math.random();
    if (activityLevel < 0.3) { // Sedentary day
        return 2000 + random.nextInt(2000); // 2000-3999 steps
    } else if (activityLevel < 0.7) { // Moderately active day
        return 4000 + random.nextInt(4000); // 4000-7999 steps
    } else { // Very active day
        return 8000 + random.nextInt(5000); // 8000-12999 steps
    }
}

```

```
// Simulates realistic sleep hours
private double generateSleepHours() {
    double sleepQuality = Math.random();
    if (sleepQuality < 0.3) { // Poor sleep
        return 5 + random.nextDouble(); // 5-6 hours
    } else if (sleepQuality < 0.7) { // Average sleep
        return 7 + random.nextDouble() * 2; // 7-9 hours
    } else { // Good sleep
        return 9 + random.nextDouble(); // 9-10 hours
    }
}

// Simulates realistic weight fluctuations
private double generateWeight() {
    double baselineWeight = 7.0; // Assuming 7.0kg as average for the subject
    return baselineWeight + (random.nextDouble() * 0.3 - 0.15); // +/- 0.15kg
    fluctuation
}
```

Another example of how we refactored our code is with the Registration activity. Before we refactored it we had all the input validation inside the UI class and we also had all the database logic in there as well. After refactoring it the code became a simple if-else statement with all the rest of the logic being handled by the InputValidator class and the UserModel class.

After refactor

// InputValidator now handles all the logic for input validation making the code much cleaner.

```
if (!InputValidator.isValidEmail(email)) {
    Toast.makeText(this, getString(R.string.invalid_email_format),
        Toast.LENGTH_SHORT).show();
} else if (!InputValidator.isValidPhoneNumber(phone)) {
    Toast.makeText(this, getString(R.string.phone_number_must_be_10_digits),
        Toast.LENGTH_SHORT).show();
} else if (!InputValidator.isValidPassword(password)) {
    Toast.makeText(this,
        getString(R.string.password_must_be_8_16_characters_contain_at_least_1_uppercase_letter_1_number_and_1_symbol), Toast.LENGTH_LONG).show();
} else if (!password.equals(confirmPassword)) {
    Toast.makeText(this, getString(R.string.passwords_do_not_match1),
        Toast.LENGTH_SHORT).show();
} else {
    SharedPreferences sharedPreferences = getSharedPreferences("userPrefs",
        Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putString(getString(R.string.email), email);
    editor.putString(getString(R.string.password1), password);
    editor.apply();
}
```

```

    // User Model now handles the Database logic we simply just pass the
    information gathered by the UI elements.
    userModel.registerUser(email, password, name, phone, this, new
    UserModel.RegistrationCallback() {
        @Override
        public void onRegistrationSuccess() {
            Toast.makeText(RegistrationActivity.this,
            getString(R.string.registration_successful), Toast.LENGTH_SHORT).show();
            startActivity(new Intent(RegistrationActivity.this,
            MainActivity.class));
            finish();
        }

        @Override
        public void onRegistrationFailed(String errorMessage) {
            Toast.makeText(RegistrationActivity.this, errorMessage,
            Toast.LENGTH_LONG).show();
        }
    });
}

```

below is a look at our registerUser method in UserModel

```

// Register a new user
public void registerUser(String email, String password, String name, String
phoneNumber, Context context, RegistrationCallback callback) {
    auth.createUserWithEmailAndPassword(email,
    password).addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            FirebaseUser firebaseUser = auth.getCurrentUser();
            if (firebaseUser != null) {
                String userId = firebaseUser.getId();

                // Store user details in Realtime Database
                User newUser = new User(email, name, phoneNumber);

                usersRef.child(userId).setValue(newUser).addOnCompleteListener(dbTask -> {
                    if (dbTask.isSuccessful()) {
                        callback.onRegistrationSuccess();
                    } else {
                        callback.onRegistrationFailed(context.getString(R.string.failed_to_save_user_data));
                    }
                });
            }
        } else {

```

```

        callback.onRegistrationFailed(task.getException().getMessage());
    }
});
}

```

Below is a look at our InputValidator class

```

// Validates if the email is in a valid format
public static boolean isValidEmail(String email) {
    if (email == null) return false;
    String emailRegex = "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,6}$";
    return email.matches(emailRegex);
}

// Validates if the phone number is exactly 10 digits (for North American
numbers)
public static boolean isValidPhoneNumber(String phoneNumber) {
    return phoneNumber != null && phoneNumber.matches("\\d{10}");
}

// Validates if the password meets the required criteria
public static boolean isValidPassword(String password) {
    if (password == null || password.length() < 8 || password.length() > 16) {
        return false;
    }

    Pattern pattern =
Pattern.compile("(?=.*[A-Z]) (?=.*[0-9]) (?!.*[!@#$%^&*+=]).{8,16}$");
    Matcher matcher = pattern.matcher(password);

    return matcher.matches();
}

```

Describe how DevOps would have helped your project

Streamlined Development and Deployment- by Continuous Integration/Continuous Deployment (CI/CD) pipelines automating building, testing, and deploying code, ensuring rapid delivery of new features and updates.

Reduced Technical Debt-having regular refactoring and maintenance as part of automated pipelines and collaborative workflows will reduce the technical debt significantly.

Risk Mitigation- Incremental changes, automated testing, and rollback mechanisms reduce the risk associated with deploying updates.

Security Vulnerability

The security vulnerability here is that we are storing the users email and phone number in plain text. This is an issue because if there is a data breach this information will be leaked and the users could receive an increase in spam or robo calls. Potentially make them more susceptible to phishing attempts.

How we would change this in the future is by using encryption to store this data so that even if there is a breach the data will be unreadable.

Security threat Code

```
public void submitFeedback(String name, String phone, String email, String
comment, float rating, Context context, FeedbackCallback callback) {
    // Prepare feedback data
    String deviceModel = android.os.Build.MODEL;
    Map<String, Object> feedback = new HashMap<>();
    feedback.put("name", name);
    feedback.put("phone", phone);
    feedback.put("email", email);
    feedback.put("comment", comment);
    feedback.put("rating", rating);
    feedback.put("deviceModel", deviceModel);

    // Add feedback data to Firebase
    feedbackRef.push().setValue(feedback)
        .addOnSuccessListener(aVoid -> {
            // Notify success through callback
            if (callback != null) {
                callback.onSuccess();
            }
        })
        .addOnFailureListener(e -> {
            // Notify failure through callback
            if (callback != null) {
                callback.onFailure(e.getMessage());
            }
        });
}
```


Project Review Meeting (names of who attended written on the sticky notes)

Project Review Meeting

Begin your post-mortem, conduct a performance review of the project. In other words, calculate the project's performance in terms of cost, schedule, and quality.

Cost - In terms of cost I believe the projects hardware is capable of being implemented with a reasonable cost.

Justin Chipman

This project would be cost effective. All hardware should not be too expensive.

Tevadi Brookes

The cost for most of the project would go mainly into R&D and would be cost effective

Imran Zafurallah

Quality - There are many aspects of the code I would like to improve on in the future had I had the time. So I believe due to this the quality is lower than what we are capable of.

Justin Chipman

Schedule - I believe in terms of schedule the project suffered a bit because of the other 6 courses there wasn't enough time to dedicate as I would have liked too.

Justin Chipman

The quality again could be much better if more time could've been allocated to the task at hand.

Tevadi Brookes

For this project to be a big success, ample time needed to be allocated .

Tevadi Brookes

Were there issues with the quality or compromises along the way?

Yes we had issues implementing certain test cases that we had to just abandon for different types.

Justin Chipman

There was a difference of opinion on the records screen design but Zane and I were able to find common ground.

Tevadi Brookes

We also did not get to fix some of the security issues such as storing email/phone number in plain text in the DB

Justin Chipman

Finding non deprecated methods to use was hard

Imran Zafurallah



Did the team members involved manage their time wisely? Or everything was done last minute.

I believe time definitely could have been reallocated to working on the project more than it was. However this semester it proved difficult to manage the 7 courses time wise. Things were for sure done last minute in a few cases.

Justin Chipman

We could've managed our time better, but with all other assignments and life we did the best we could.

Tevadi Brookes

The time allotted to the last sprint was too short. We had too much due at once and it was overwhelming.

Imran Zafurallah



Lessons learned, mistakes, and areas of improvement.

Time management.

Tevadi Brookes

A lesson I learned is to go out of my way to chip a little away at a project over time even if it is a small chunk.

Justin Chipman

Start the project early and do little by little

Imran Zafurallah

Learned how to effectively work together in the master with group without affecting code.

Tevadi Brookes

I ran into a lot of configuration issues along the course of this project and it cost me a lot of spent time.

Justin Chipman

I learned how to write code using better design principles in order to create more readable and abstracted code.

Justin Chipman

