Project 5: Wasteagram

Introduction

Demonstrate proficiency of prior material, and challenge yourself to demonstrate how to apply new concepts to meet functional requirements. Build an application for recording food waste. Practice applying the concepts of location services, camera / image picker, permissions, forms, navigation, lists, asynchronous programming, streams, and Firebase backend services. Enhance your application with analytics, crash reporting, accessibility, internationalization, debugging and automated testing.

Scenario

Your client, Matthew Peter, is the owner of TwentySix Cafe, a Portland coffee shop.

"Man, I am so tired of these wasted bagels and pastries we have at the end of every day!" he says. "I'm losing money, and it's so wasteful... I feel like there's an episode of Portlandia about this. I mean, why waste a donut? A *donut*!"

Mr. Peter wants his employees to run an application that, "is like Instagram, but for food waste," he says. Every night the person closing the shop can gather up the leftover baked goods, take out their phone, start the app, and create a post consisting of a photo of the wasted food and the number of leftover items.

"If only I could see a list of these posts over time, then at least I'd know how much money I'm losing, and I could make adjustments to my pastry orders," he says, dreamily. "No more forsaken donuts!"

You have engaged Matthew Peter in a paid contract to develop a functioning version of the application that he and his employees can try out at the coffee shop. "Hey, I know," he says, "Let's call it *Wasteagram*."

What to Do

Implement *Wasteagram*, a mobile app that enables coffee shop employees to document daily food waste in the form of "posts" consisting of a photo, number of leftover items, the current date, and the location of the device when the post is created. The application should also display a list of all previous posts. After discussing the requirements with the client and sketching out the UX flow.

The **functional requirements** are:

1. Display a circular progress indicator when there are no previous posts to display in the List Screen.
2. The List Screen should display a list of all previous posts, with the most recent at the top of the list.

3. Each post in the List Screen should be displayed as a date, representing the date the post was created, and a number, representing the total number of wasted items recorded in the post.
4. Tapping on a post in the List Screen should cause a Detail Screen to appear. The Detail Screen's *back* button should cause the List Screen to appear.
5. The Detail Screen should display the post's date, photo, number of wasted items, and the latitude and longitude that was recorded as part of the post.
6. The List Screen should display a large button at the center bottom area of the screen.
7. Tapping on the large button enables an employee to capture a photo, or select a photo from the device's photo gallery.
    1. Note: you need to support either capturing a photo or selecting a photo from the device's photo gallery, but not both.
8. After taking a new photo or selecting a photo from the gallery, the New Post screen appears.
9. The New Post screen displays the photo of wasted food, a *Number of Items* text input field for entering the number of wasted items, and a large *upload* button for saving the post.
10. Tapping on the *Number of Items* text input field should cause the device to display its numeric keypad.
11. In the New Post screen, tapping the *back* button in the app bar should cause the List Screen to appear.
12. In the New Post screen, tapping the large *upload* button should cause the List Screen to appear, with the latest post now appearing at the top of the list.
13. In the New Post screen, if the *Number of Items* field is empty, tapping the *upload* button should cause a sensible error message to appear.

In addition to the functional requirements above, your application should meet the following **technical requirements**:

1. Use the *location*, *image_picker*, *cloud_firestore*, *and firebase_storage* packages to meet the functional and technical requirements.
2. Incorporate the use of Firebase Cloud Storage and Firebase Cloud Firestore for storing images and post data.
3. Data should not be stored locally on the device.
4. On the List Screen, the application should display the posts stored in the Firestore database.
5. On the Detail Screen, the application should display the image stored in the Cloud Storage bucket.
6. On the New Post screen, tapping the large *upload* button should store a new post in the Firestore database.
7. Each "post" in Firestore should have the following attributes: *date*, *imageURL*, *quantity*, *latitude* and *longitude*.
8. The application should incorporate the Semantics widget in multiple places, such as interactive widgets like buttons, to aid accessibility.

9. The codebase should incorporate a model class.
10. The codebase should incorporate a few (two or three) simple unit tests that test the model class.

The functional and technical requirements specifically exercise the Exploration content and our module learning outcomes. In addition, here are **some optional extra credit requirements.** You can implement up to 2 of these requirements to get extra credit. Each of these 2 optional requirements that you implement will add an extra 1% to the grade. So in total you can add an extra 2% to your class grade.

1. The app bar of the List Screen should display the total sum of the number of wasted items in all posts.
2. Add integration tests that verify any one particular part of the UX flow.
3. Integrate the use of in-app analytics (Analytics) to monitor application usage.
4. Integrate the use of crash reporting (Sentry or Crashlytics) to record application crashes.