Project 2A - DreamCatcher List-Detail

39/40 Points

3/3/2022

Attempt 1

REVIEW FEEDBACK
3/2/2022

Attempt 1 Score: **39/40**

View Feedback

Unlimited Attempts Allowed

Details

Due: Thursday, March 3
Points: 50 points
Deliverables: 2 screenshots and a ZIP file
Grading: 40 for tests (20 points for 10 base tests + 20 points for 20 instructor tests) + 10 points for 2 screenshots
Resources:

- build.gradle (https://drive.google.com/file/d/1wzmUQfzkC7hsCW39z0ubvdZGVjIg_U7x/view?usp=sharing) (subject to minor tweaks)
- DreamRepository (https://drive.google.com/file/d/19UesXyKR57bf9h3A7a9hOUccTH8CAtx-/view?usp=sharing)
- BaseDreamCatcherListDetailTest (https://drive.google.com/file/d/1JadeB-0NwFjOrVLPwu5tG38yWF2jLOxW/view?usp=sharing)

Additional Code

In addition to the files provided above, please copy-paste the code below into appropriate Kotlin files. These are model classes that you will use in your project.

Dream.kt

```
data class Dream(
    val id: UUID = UUID.randomUUID(),
    var title: String = "",
    var date: Date = Date(),
    var isFulfilled: Boolean = false,
    var isDeferred: Boolean = false
)
```

DreamEntry.kt

```
data class DreamEntry(
    val id: UUID = UUID.randomUUID(),
    val date: Date = Date(),
    val text: String = "",
    val kind: DreamEntryKind = DreamEntryKind.REFLECTION,
    val dreamId: UUID
)
```

DreamEntryKind.kt

```
enum class DreamEntryKind {
    CONCEIVED, DEFERRED, FULFILLED, REFLECTION
}
```

DreamWithEntries.kt

```
data class DreamWithEntries(
    var dream: Dream,
    var dreamEntries: List<DreamEntry>
)
```

Finally, in addition to the above model classes, you will need the following application class to initialize your repository (see BNR Chapter 11, section "Accessing the Database using the Repository Pattern").

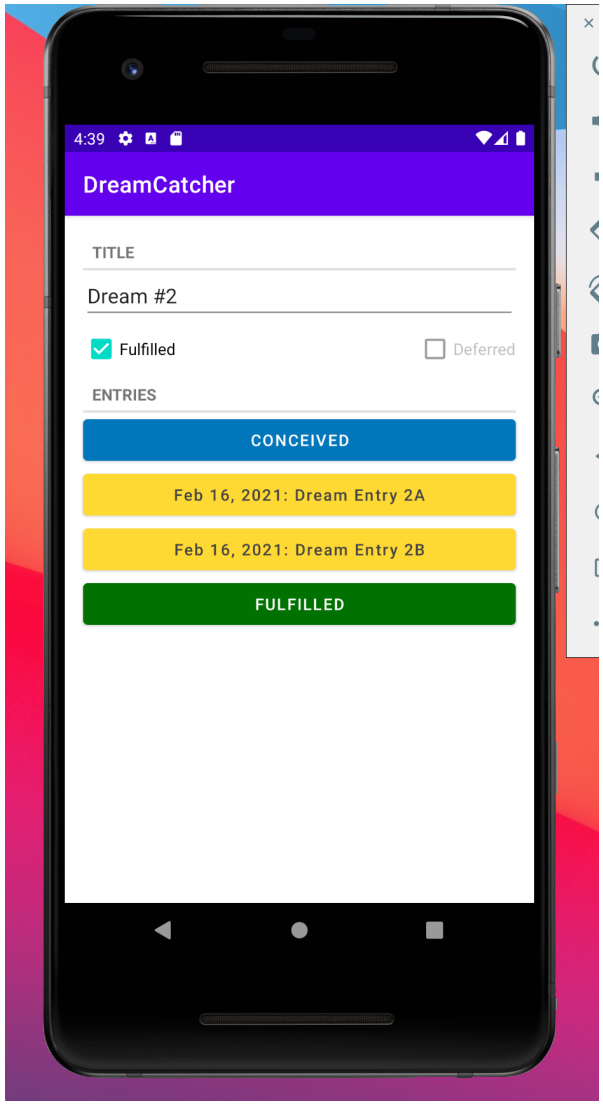
```
class DreamCatcherApplication : Application() {

    override fun onCreate() {
        super.onCreate()
        DreamRepository.initialize(this)
    }
}
```

We are not going to use a database in this first DreamCatcher project, but we will use a repository, which is given to you above. **Please do not modify any code we give to you unless we explicitly tell you that you can.**

Overview

The DreamCatcher application will span multiple projects. The application is similar in many respects to the CriminalIntent application in BNR. It is a list-detail application in that it contains both a list view and a detail view, and you will implement both of these views as fragments rather than activities. When the application opens, the list view appears. The list of dreams will be stored as a list of Dream objects and you will display the list with RecyclerView.

Instead of documenting workplace crime, as in the CriminalIntent application, you will document your dreams - the big things that you would like to do in your life (kind of like a "bucket list"). Each dream will have a description, a date that the dream was conceived, a list of updates (each updates includes a date and a short text comment), a checkbox that indicates whether the dream was fulfilled, and a checkbox that indicates whether the dream is deferred. The detail-view screen should look something like this.

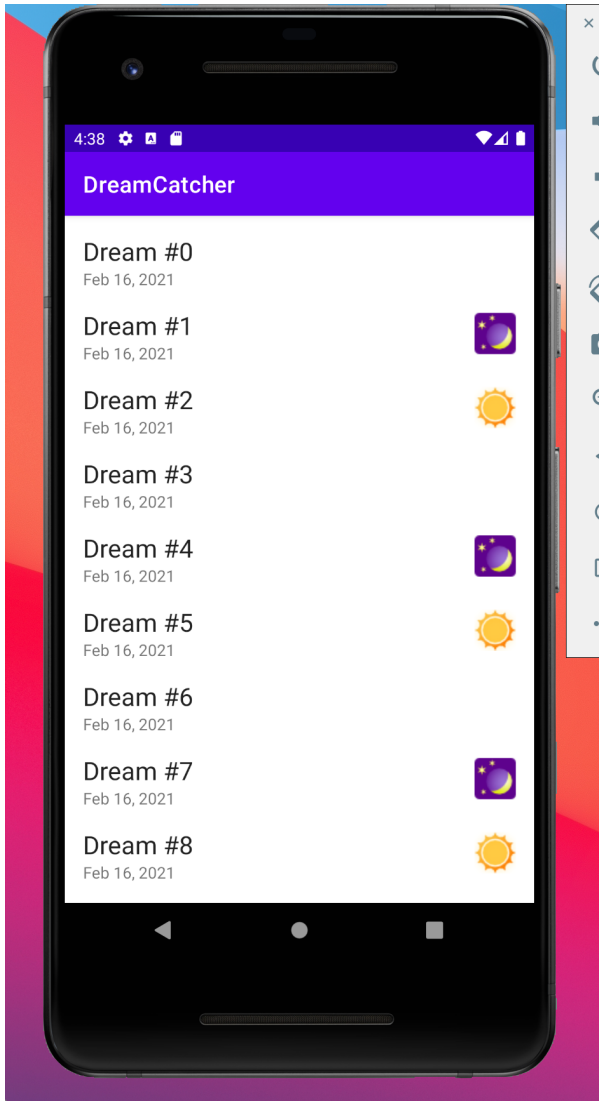


Some detail view requirements:

- Both labels (title and entries) should have the same styling
- If labels are underlined, the line should span the width
- No elements are right up against the edge of the screen (16dp margin recommended)
- Fulfilled checkbox and Deferred checkbox are on the same horizontal
- Fulfilled checkbox is flush left and Deferred checkbox is flush right
- Buttons should have a small amount of space between them

- The colors of the different buttons should all be different
- Do not use the same color buttons that are used in the image
- Make sure you text is easy to read (for example, white text on the yellow buttons above would **not** be readable)
- The date is included with the text on certain button (of type "reflection")
- The date does not include the time
- If you have any questions about the requirements, ask on Piazza

The list-view screen displays a list of all the dreams. It will display the short text description of the dream with the date below it. To the right of the text and the date, an image will be present if the dream has been fulfilled. The list-view screen will look something like this.



Some list view reqruirements:

- Dream titles should be styled more prominently than dream dates
- No text or icons should ever be right up against the edge of the screen or another element
- Put space between list items. It should be apparent that there is more space between list items than there is between the title and date in the same list item
- The spacing of list items should always be consistent regardless of whether the items are displaying an image or not
- Find images online (or create your own) for dream fulfilled and dream deferred

- Images should be in color. No black and white icons please!
- Be creative with your images. Do not use stars, hearts, checks, or X's
- The padding on the left should roughly equal the padding on the right

Layout Names

For the tests to work correctly, you must use exact names in the layouts. You will have 4 layouts in the application, and the views inside them should have the following IDs:

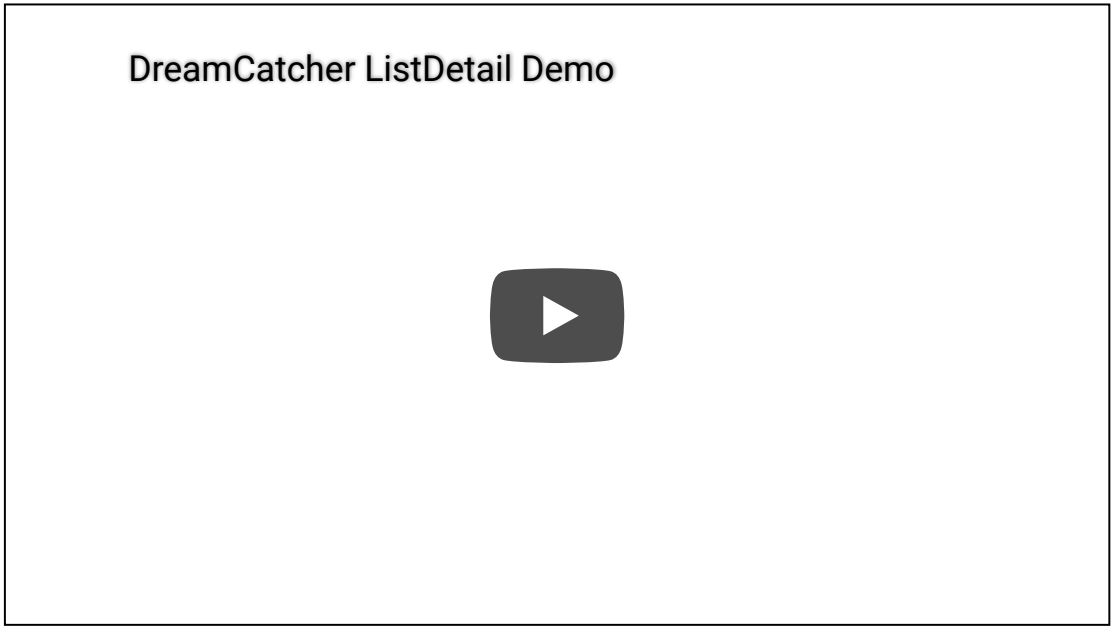
- activity_main.xml (fragment_container)
- fragment_dream_detail.xml (dream_detail_layout)
  - dream_title_label
  - dream_title_text
  - dream_fulfilled_checkbox
  - dream_deferred_checkbox
  - dream_entries_label
  - dream_entry_0_button
  - dream_entry_1_button
  - dream_entry_2_button
  - dream_entry_3_button
  - dream_entry_4_button
- fragment_dream_list.xml (dream_recycler_view)
- list_item_dream.xml (dream_item_layout)
  - dream_item_title
  - dream_item_date
  - dream_item_image

If there is any confusion about which ID goes with which view, please ask on Piazza.

Functionality

When a dream from the list is selected, the detail view for that dream is displayed. When the device's back button is pressed, it will take you back to the list view. When the user checks the box in detail view indicating that a dream has been fulfilled, the image should be

present for that dream in list view.



Getting Started

To get started with DreamCatcher, we suggest first creating the detail-view and list-view fragments as described in chapters 8 and 9 of BNR (you will not need the repository for this). The DreamCatcher detail view is more complicated than the detail view of CriminalIntent. Here are a few important requirements.

- The detail view has two checkboxes: fulfilled and deferred. When the fulfilled checkbox is check, the deferred checkbox should be disabled, and vice-versa. Therefore, it will be impossible to have both boxes checked at the same time.
- The current detail view layout has exactly 5 buttons to display a list of dream-entries. The repository has set up the model so that you will never need more than 5 buttons to display the entries. In future projects we will allow an unlimited number of entries to be displayed.
- There are four different kinds of dream entries: conceived, deferred, fulfilled, and reflection.
- Every dream that is created must have a "conceived" dream entry as the first entry in the list.
- If the "deferred" checkbox is checked, the last dream entry in the list will be a "deferred" dream entry
- If the "fulfilled" checkbox is checked, the last dream entry in the list will be a "fulfilled" dream entry
- Unchecking either checkbox gets rid of its corresponding dream entry.

Testing

In order for the test cases to work, you must name your layout elements as shown above.

Furthermore, your images should be named dream_fulfilled_icon and dream_deferred_icon. One relatively easy way to upload and use PNG images in your code is to do the following.

- Find or create two PNG images for dream fulfilled and dream deferred. Make the images relevant and non-trivial (don't use the black-and-white vector images from Android Studio)
- The image should not be too large (they will fit in about a 40 dp square area)
- Ensure that their names are dream_fulfilled_icon.png and dream_deferred_icon.png
- In the file system (with Android Studio closed), drag and drop the images to the drawables-v24 folder of your project
- Open Android Studio again. From your list_item_dream.xml layout, load a default image into your imageView with the following.

```
        app:srcCompat="@drawable/dream_fulfilled_icon"
```

Use the bind function in DreamHolder to swap out images. Something like the following code should work.

```
when {
    dream.isDeferred -> {
        itemBinding.dreamItemImage.setImageResource(R.drawable.dream_deferred_icon)
        itemBinding.dreamItemImage.tag = R.drawable.dream_deferred_icon
    }
    dream.isFulfilled -> {
        itemBinding.dreamItemImage.setImageResource(R.drawable.dream_fulfilled_icon)
        itemBinding.dreamItemImage.tag = R.drawable.dream_fulfilled_icon
    }
    else -> {
        itemBinding.dreamItemImage.setImageResource(0)
        itemBinding.dreamItemImage.tag = 0
    }
}
```

Note that updating the image-view tags is **mandatory**. It's how we test that you have the correct image in your recycler view item.

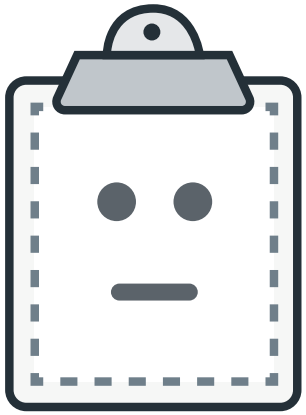
Grading: Test Cases & Screenshots

10 base test cases are provided. Please ensure that your code passes all of these tests before you submit.

The testing portion of your grade will be worth 40 points. 2 points for each of the 10 base test case you pass + 1 point for each of the 20 instructor tests you pass.

You should submit 2 screenshots to a separate assignment. The screenshots do *not* need to contain AndroidStudio running in the background.

- Screenshot #1: Emulator with list view starting at dream #5. The screenshot should be similar to the one above except that dream #5 should be the first dream. Also, make sure that your icons are different from those above and they are consistent with the requirements
- Screenshot #2: Emulator with detail view containing dream #2. As in the screenshot above, **the fulfilled checkbox should be selected**. Your button colors should not be the same as in the screenshot above.



Preview Unavailable

DreamCatcher.zip

Download

(https://canvas.vt.edu/files/22281440/download?download_frd=1&verifier=T3VEEtKFHNNefR5kgYt4ewxmZLdE55Qxk65XjZid)

You are unable to submit to this assignment as your enrollment in this course has been concluded.