DreamCatcher App (Part 1 of 3)

In this first of three parts, we will begin building an extensive app very similar to the CriminalIntent app covered by BNRG Chapters 9 through 19. Each part will iteratively build upon the earlier parts, introducing new features and making a few changes along the way. This part closely corresponds to Chapters 9 through 11 in developing the two major fragments.

Instead of documenting workplace crimes as in the BNRG example, the DreamCatcher app will allow users to document their dreams. Dreams here can be any aspirations, achievements, or other goals the user might have. It can be as lofty as lifelong ambitions, or as mundane as a shopping list. It's all up to the user.

Each dream is tracked along with sequence of intermediate reflections, which can be used to record short notes or to indicate partial progress toward the goal. A dream may be marked as deferred, meaning that it's on hold and not being actively pursued, or a dream may be marked as fulfilled, meaning that the goal has been reached.

Overview

The DreamCatcher app will consist of a single activity with two fragments to form a list-detail interface. The DreamListFragment is a screen with a scrollable list containing a brief summary of every dream. The DreamDetailFragment is a screen where the user can view and update the details of a single dream.

In this first part of the project, the main objective is to create these two main screens as fragments, hosted by a single simple activity. The activity will host one full-screen fragment at a time, and at the end of this part there will be no mechanism yet to navigate from either fragment to the other. Thus switching between fragments will require manually updating the activity's layout file and relaunching the app.

Getting Started

A project template with starter code is once again provided via Git. Please start Project 2A by selecting "Get from VCS" from the Android Studio welcome screen, or use File | New | Project from Version Control from the menu of any existing project. In either case, specify the following remote URL, and specify any convenient directory on your development system:

* https://www.prof-oliva.com/cs5254/2024.Spring/git/P2-DreamCatcher.git

This will create a new project by cloning a local copy of the remote repository. The project will contain the all starter code needed for this assignment in a single assignment branch.

Beyond the basic files normally created in a new Android Studio project, the repository includes the following changes and additions specific to this project:

* **build.gradle.kts** (Module :app)
  + The configuration file has been updated according to BNRG Chapters 9 and 10 (with current versions) plus additional options and dependencies required to support instrumented testing
* **Dream.kt**
  + Dream is the central data class for this project. Each Dream object contains a list of at least one DreamEntry object, along with computed properties based on which kinds of DreamEntry objects are present/absent. The DreamEntry data class and the DreamEntryKind enum class are also defined within this file.
* **DreamEntryButtonUtil.kt**
  + Provides a setBackgroundWithContrastingText() extension function to the Button class. As suggested by its rather long name, this function changes the background color of a Button, and ensures the text of the button is displayed in a contrasting color. The descriptive name is meant to clearly distinguish it from similar built-in functions of the Button class.
* **DreamDetailViewModel.kt**
  + This initial view model should be associated with the DreamDetailFragment. It constructs a single Dream instance (containing several entries) as a placeholder for initial development and testing purposes.
* **DreamListViewModel.kt**
  + This initial view model should be associated with the DreamListFragment. It constructs a list of 100 Dream instances (with varying combinations of entries) as a placeholder for initial development and testing purposes.
* **DreamDetailFragmentTest.kt**
  + Provides initial instrumented tests for the DreamDetailFragment.
* **DreamListFragmentTest.kt**
  + Provides initial instrumented tests for the DreamListFragment.

You must integrate these files, as provided, into your solution.

Phase 1: Dream Detail

Detail Screen Layout

The detail screen must display a single Dream with an editable title, the last updated date and time, two checkboxes, and a list of up to five entries displayed as disabled buttons. The layout should be as follows:

Portrait orientation (default):

A screenshot of a computer screen

Description automatically generated

Landscape orientation:

A screenshot of a computer

Description automatically generated

Detail Screen Requirements

* Ensure both section labels (Dream and Entries) have the same style
* Include a 16dp margin surrounding the entire interface (apply this to the root LinearLayout)
* Align the checkboxes on the same horizontal line
  + Spread the checkboxes all the way flush to the left/right of the section
* Allow sufficient room for all five buttons to be displayed in each orientation
* Set all buttons to be disabled in the layout
* Use the following view component ID values:
  + title\_text
  + last\_updated\_text
  + fulfilled\_checkbox
  + deferred\_checkbox
  + entry\_0\_button
  + entry\_1\_button
  + entry\_2\_button
  + entry\_3\_button
  + entry\_4\_button
* Specific to the landscape orientation only:
  + The Dream section and Entries section must have exactly equal widths
  + Include a 16dp width space between the two sections

Detail Screen Appearance

* Ensure that all 5 dream entry buttons can fit entirely on the screen – in both portrait and landscape modes – on a **Pixel 4** device
* Select a unique button color for each entry kind
  + Don't use the same color combinations as shown above
* Ensure the reflection text is displayed in mixed case (not all caps)
  + Display all other entry kinds as the string representation of the kind
* Populate the buttons starting from entry\_0\_button at the top and work down
  + Set the visibility of any buttons without associated entries to View.GONE
* For the last updated date, use the format function of android.text.format.DateFormat
  + Double-check the imports, as there are several other DateFormat classes available in other packages
  + Use the following format string:
    - 'Last updated' yyyy-MM-dd 'at' hh:mm:ss A
  + Note that the above format string can just be hard-coded, rather than stored as a string resource, since we haven't quite yet covered how to manage general purpose string resources

Detail Screen Behavior

* Checkbox behavior (see Piazza for a brief video)
  + Checking either checkbox must:
    - Disable the other checkbox, so at most only one may be checked
    - Append a new entry of the proper kind (Deferred or Fulfilled) to the end of the entry list
  + Unchecking either checkbox must:
    - Enable the other checkbox, so both checkboxes are enabled only when neither is checked
    - Remove the entry of the proper kind (Deferred or Fulfilled) from the list
* Any user changes (dream title and/or checkbox state) must update the Dream object in the view model
* The entire system state must persist across a screen rotation
* The last updated timestamp shouldn't be changed by the code in this project
  + This value gets set once, when the DreamDetailViewModel instantiates the Dream object, and remains the same until the app is closed

Phase 2: Dream List

List Screen Layout

The list screen must scroll using a recycler view component showing each dream's title and number of reflections, along with an icon indicator if the dream is deferred or fulfilled. Only one layout is required for both portrait and landscape:

A screenshot of a cell phone

Description automatically generated

List Screen Requirements

* You may use either ConstraintLayout (as in BNRG) or LinearLayout for the root of the list\_item\_dream layout
* Style the dream titles to be much more prominent than the reflection counts
* Provide margins sufficient to allow some space between each list item
  + Allow more space between list items than between the title and the reflection count
  + Ensure spacing of list items is consistent regardless of whether an image is present
  + Use approximately the same amount of padding on the left as on the right
* Set the ImageView size to approximately 40dp x 40dp
  + Image drawable names must be:
    - ic\_dream\_deferred
    - ic\_dream\_fulfilled
* Use the following view component ID values:
  + In the fragment\_dream\_list.xml layout:
    - dream\_recycler\_view
  + In the list\_item\_dream.xml layout:
    - list\_item\_title
    - list\_item\_reflection\_count
    - list\_item\_image

List Screen Appearance

* Ensure that at least 10 dreams are always fully visible on the screen – in portrait mode only – on a **Pixel 4** device
  + Dream #0 through Dream #9 inclusive must be visible initially at the top
  + Dream #90 through Dream #99 inclusive must be visible after scrolling to the bottom
* Find images online (or create your own) to act as fulfilled and deferred indicators
  + Look for very small (256 x 256) source images for best results
    - You may edit larger images down to this size if you prefer
    - Image files must be no larger than 100kB each
  + Use color or grayscale images, rather than monochrome (black and white)
  + Be creative with the images; avoid emojis or simple shapes such as stars, hearts, checks, or similar
* Adjust the visibility property of the list\_item\_image as appropriate:
  + Use View.GONE if the dream is neither deferred nor fulfilled
  + Use View.VISIBLE otherwise, and set the image resource to the ID of the drawable

List Screen Behavior

* No user interaction is required, beyond scrolling through the list
  + Scrolling should be smooth and responsive