GitHub Username: **angela-aciobanitei**

Pin It

Description

Organization is a key skill in both work and life! PinIt is a personal Pinterest-style app that allows users to record photos or videos and store them in collections. The collections will organize related photos and videos and include cover photos, titles, tags, and more.

Intended User

This app is intended to be used by anyone who wants to organize their photos and videos in the familiar Pinterest app style, using boards and pins, without publicly revealing the content of their collections.

Features

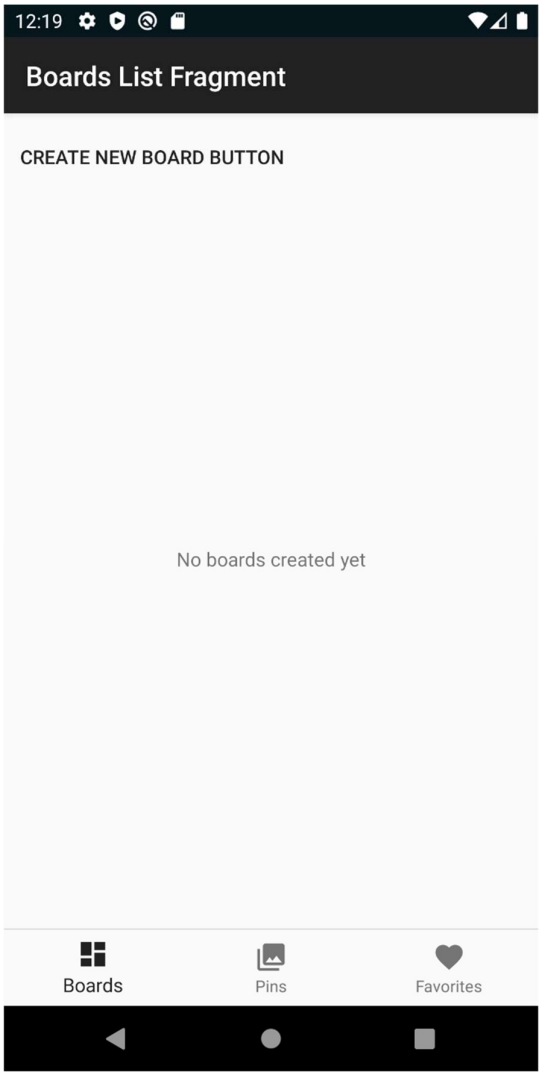
- The app allows users to capture or choose pictures or videos.
- Individual pins (photos or videos) include a title, comments, and tags.
- The app allows users to create sets of pictures and/or videos called Boards.
- Each board has a title and a cover image selected from the collection.
- Tapping on a specific board will display the containing pins in a grid.
- Tapping on a video will display it directly in this app (instead of playing it via an intent).

User Interface Mocks

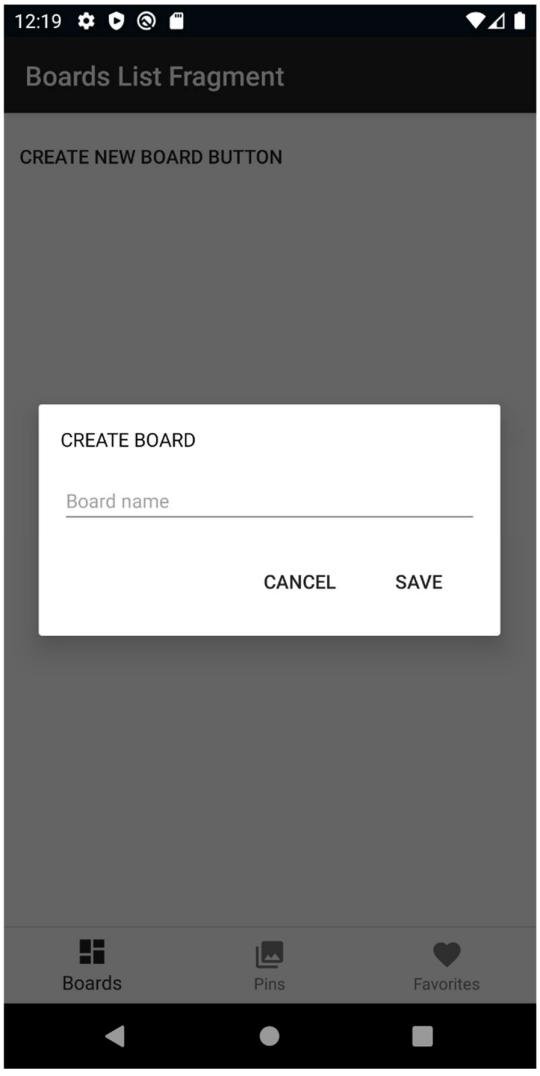
The app has 3 main screens: boards, pins and favourites. Navigation to each of these screens is done using a bottom navigation view as shown in the pictures below.

- The Home screen is the Boards Fragment, which displays a list of boards.
- Initially, this fragment displays a “no data yet” message to the user.
- User can create a new board by clicking on a button, which will open up a dialog for this purpose.
- Once the user has created a bunch of boards, he/she can go ahead and populate these boards with pins of their choice.

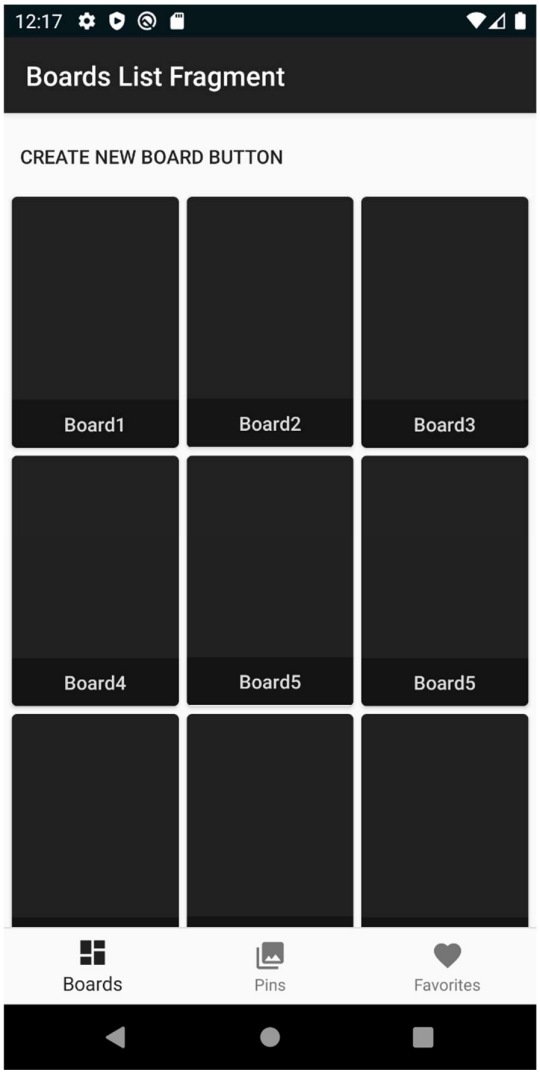
Boards Fragment, empty state



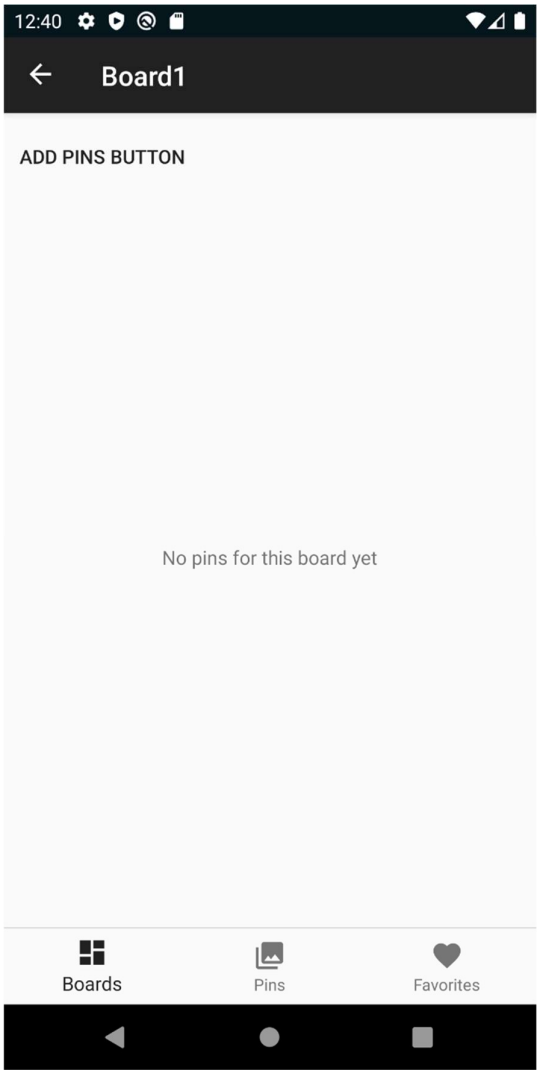
Creating a new board



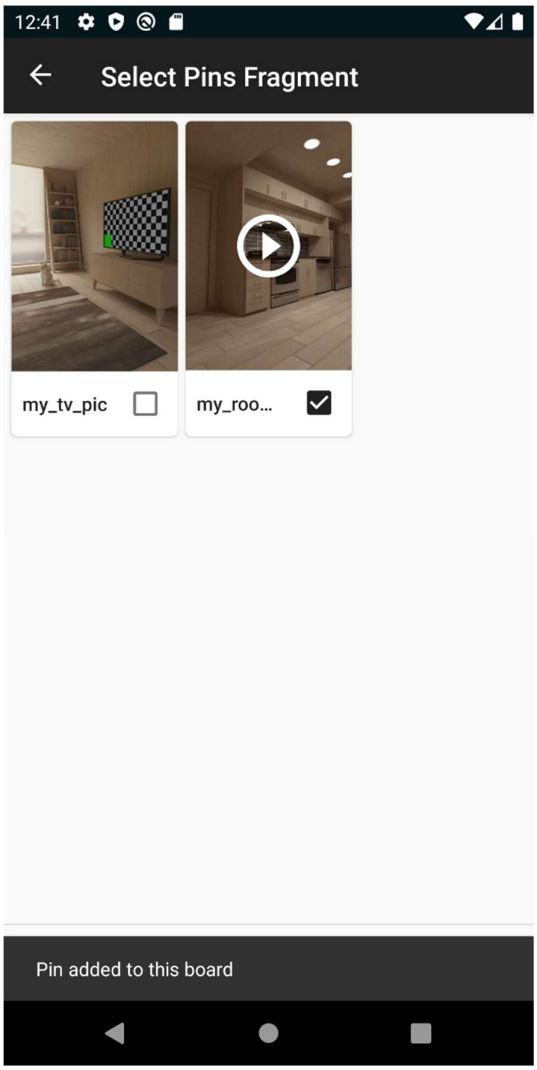
Boards Fragment, populated



Clicking on a board navigates user to Board Details fragment



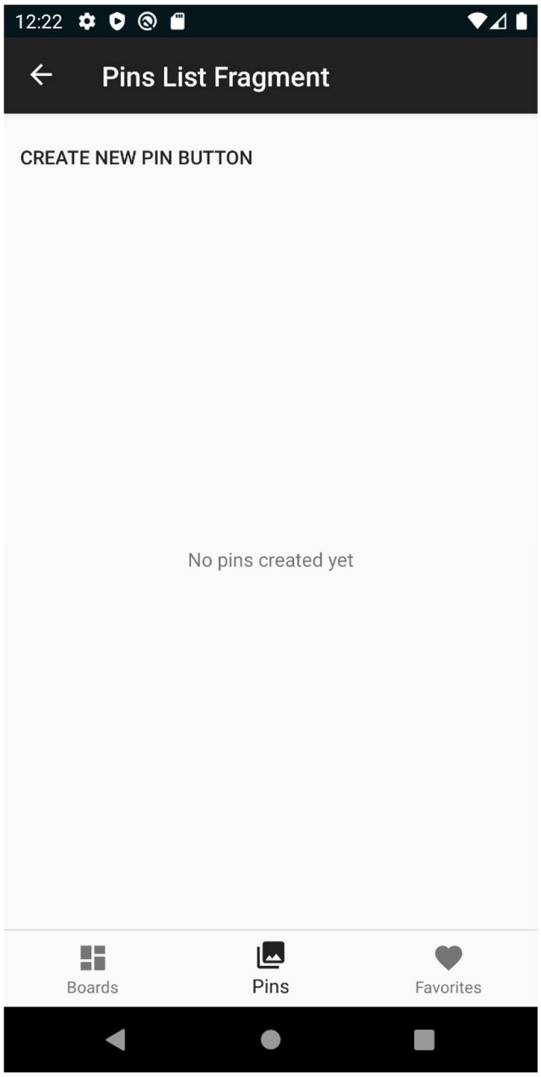
Clicking on ADD PINS button navigates user to Select Pins fragment
Note: user is notified every time a pin is added to the current board via a Snackbar message.



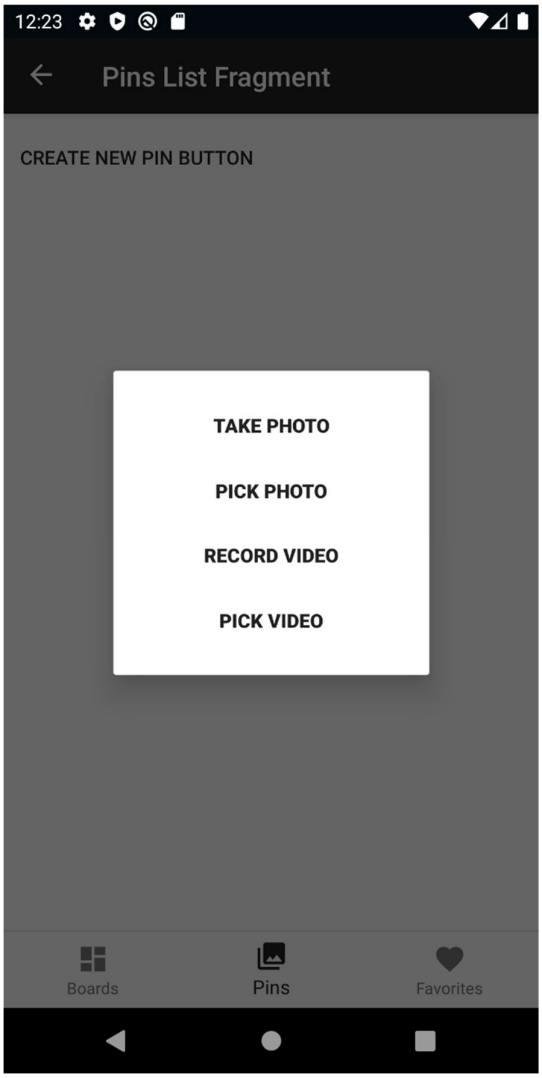
The next main screen is Pins Fragment, which displays a list of pins.

- Initially, this fragment displays a "no data yet" message to the user.
- User can create a new pin by clicking on a button, which will open up a dialog for this purpose.
- The "New Pin" dialog offers a list of choices to the user on how to create a pin: take a photo, pick an already existing photo, record a video, pick a video.

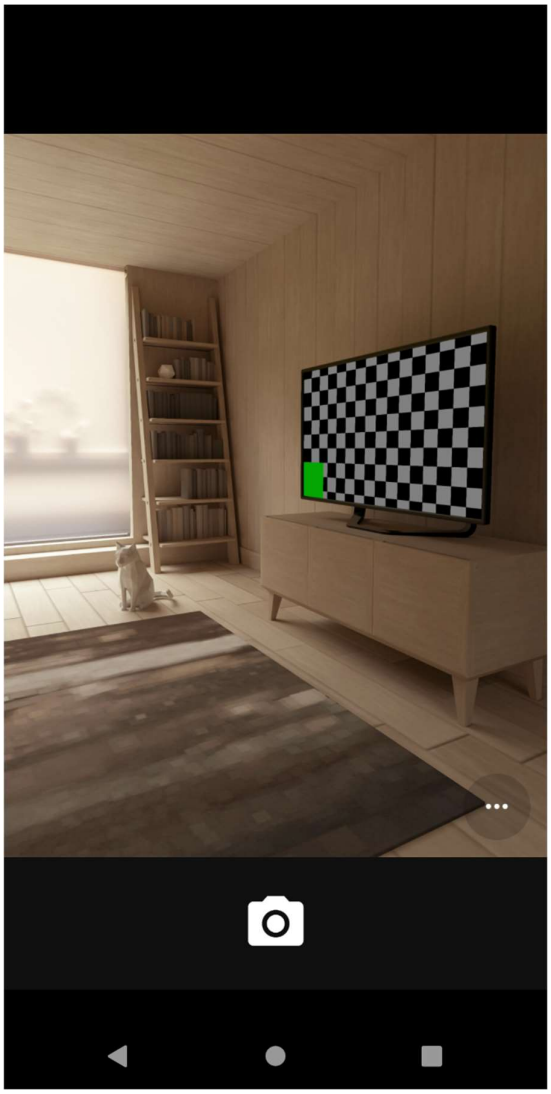
Pins Fragment, empty state



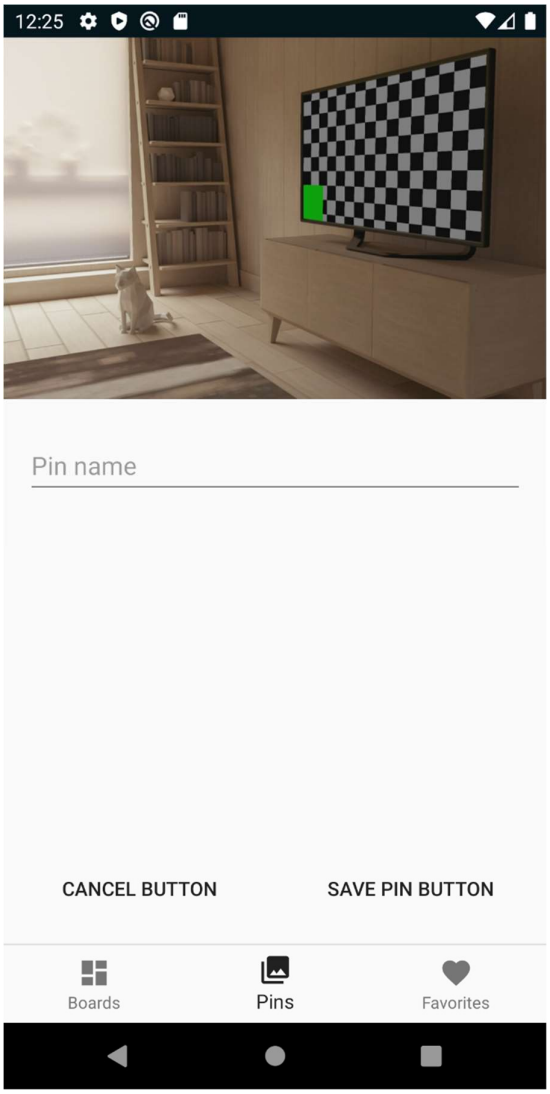
Create new pin dialog



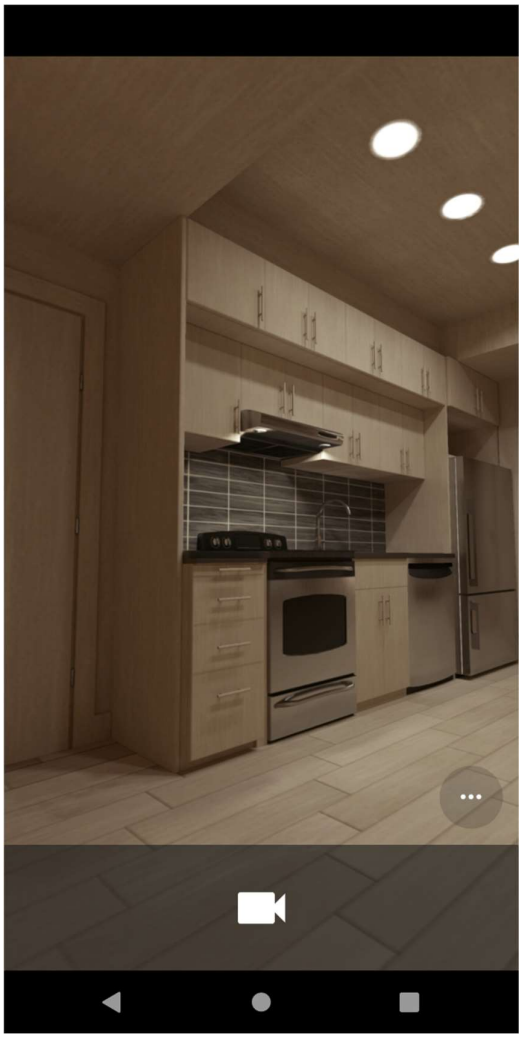
Creating new pin: take photo



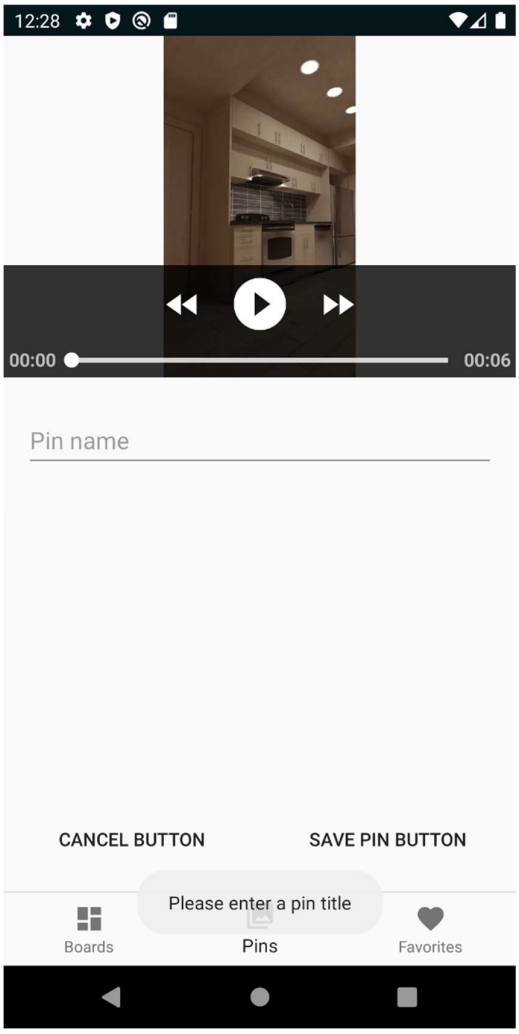
Creating new pin: save pin



Create pin: record video



Create pin: save video

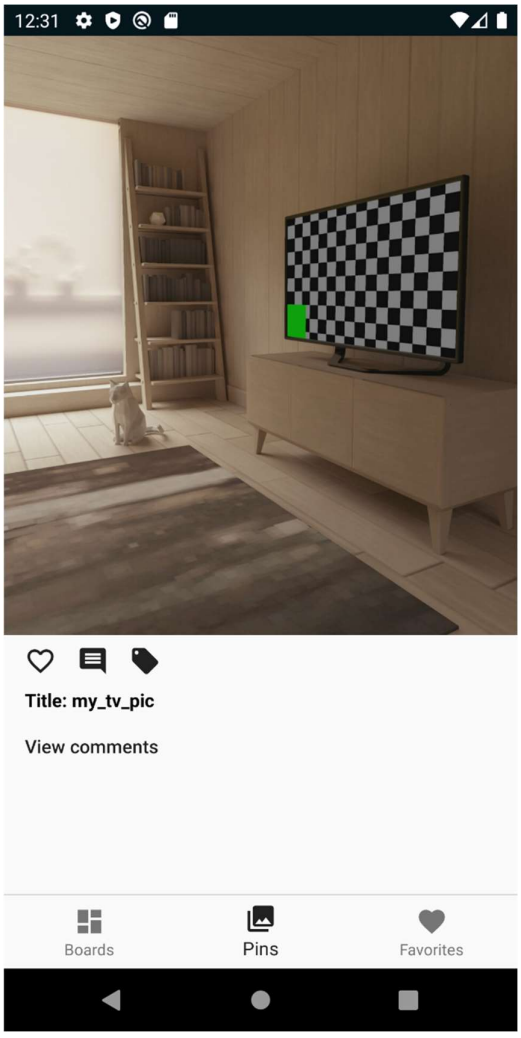


Note: user can't save a pin with an empty title. A Toast is shown if the user hits the Save button without providing a Pin title.

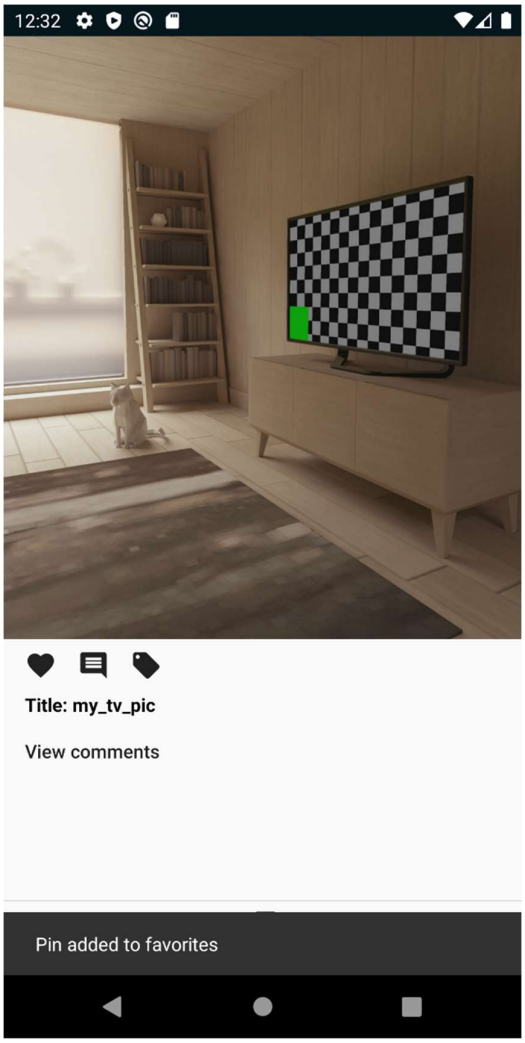
Clicking on a pin, navigates the user to Pin Details fragment. Here, the user can:

- add or remove the pin to “Favourites”
- create pin tags
- add pin comments

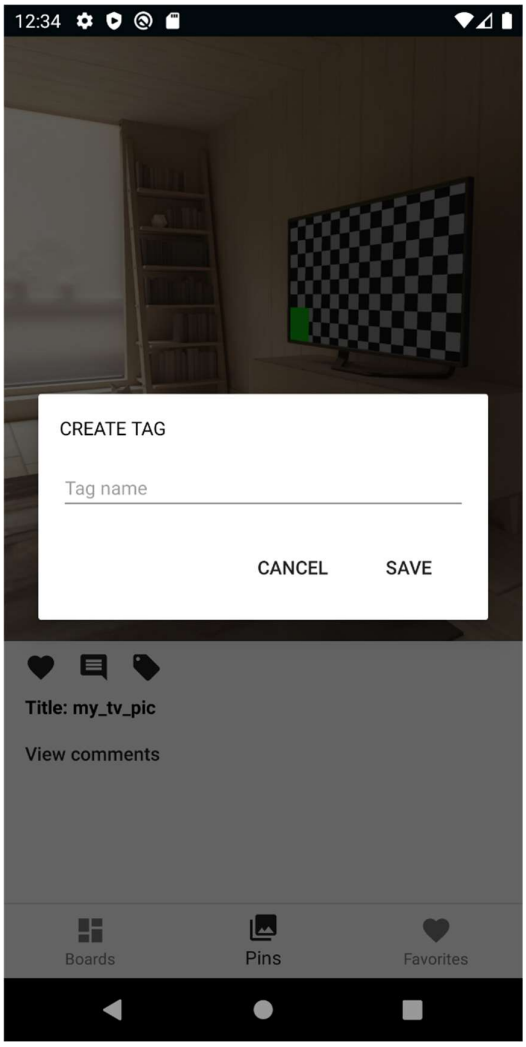
Edit pin



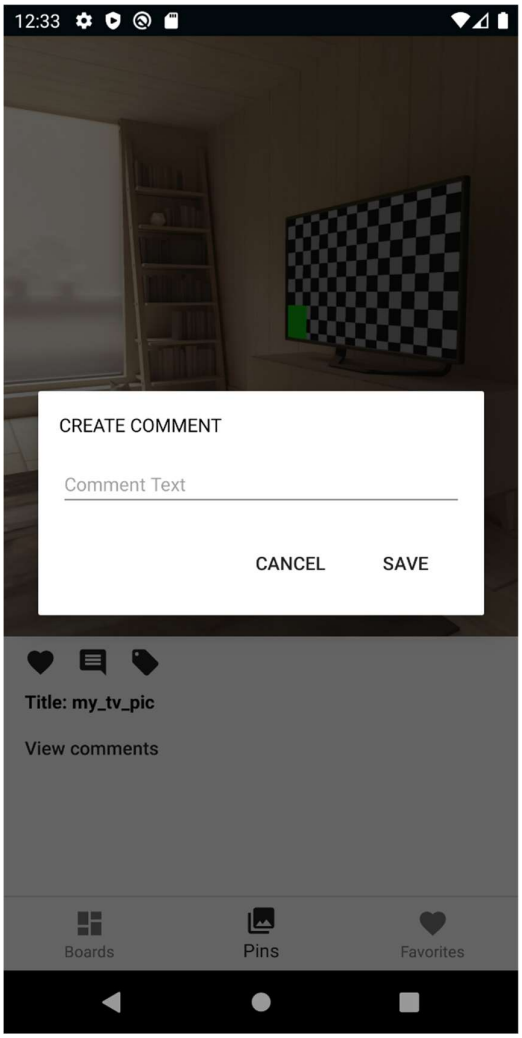
Edit pin: favourite or unfavourite pin



Create new tag dialog



Create new comment dialog



Key Considerations

How will your app handle data persistence?

Data persistence will be handled using Room. I will use the following entities:

- a Board, which has a title and a photo cover URL.
- a Pin, which can be a photo item or a video item.
- a BoardPin - since a board can have any number of pins, and a pin can be included in any number of boards, we need to define a many-to-many relationship between them. (https://developer.android.com/training/data-storage/room/relationships#many-to-many).
- a Tag – since a Pin can have any number of Tags, we need to define a one-to-many-relationship for this entity
- a Comment – this also has a one-to-many-relationship with a Pin.

Describe any edge or corner cases in the UX.

- In case of an empty list content, the user is informed with an appropriate status message ("No boards created yet" or "No pin created yet").
- If the user favourites or unfavourites a pin, the app will display a Snackbar message ("Pin added to favourites" or "Pin removed from favourites")
- Marking a pin as favourite or not updates the favourites list accordingly.

Describe any libraries you'll be using and why.

- Room for data persistence.
- Other Jetpack components: Navigation, Data Binding.
- Glide for image loading.
- ExoPlayer for streaming videos.
- Timber for logging.
- Dagger 2 for dependency injection.

Describe how you will implement Google Play Services or other external services.

- AdMob (https://developers.google.com/admob/android/quick-start)
  - Import the Mobile Ads SDK.
  - Add the AdMob App ID to the Manifest.
  - Initialize Mobile Ads SDK.
  - Enable test ads.
- Firebase Analytics (https://firebase.google.com/docs/analytics/get-started)
  - Enable Google Analytics in my project .
  - Add the Analytics SDK to my app.
  - Start logging events.

Required Tasks

Task 1: Project Setup

- Create the project and the git repository
- Add the necessary libraries dependencies to build.gradle
- Structure the project packages: data, ui, utils etc.
- Add necessary permissions to the Manifest:
    - WRITE_EXTERNAL_STORAGE
    - CAMERA
- Specify in the Manifest that this app uses the Camera (<uses-feature android:name="android.hardware.camera2" />)

Task 2: Sketch the UI to create the Nav Graph

- The app will have a single host activity with multiple fragments.
- The navigation will be handled via the Jetpack Navigation Component.
- In order to create the graph, create new empty fragments that will represents the points in the app to which the user can navigate: boards list, boards details, pin list, pin details, pin edit, select pins and favourite pins. We'll populate these fragments later on.
- Create the nav graph and use it in the main activity layout.
-

Task 3: Create the Database

- Create the Room database (must be a Singleton).
- Create the Room entities (Board, Pin, Comment, Tag etc) and define the necessary
  relationships between them (one-to-many or many-to-many).
- Create the DAOs for each entity.

Task 4: Add DI with Dagger

- Add necessary dependencies to build.gradle
- Centralize all singleton creation with Dagger: the app database, the DAOs etc.
- Create the necessary Dagger modules (see: https://github.com/codepath/android_guides/wiki/Dependency-Injection-with-Dagger-2)

Task 5: Create the Repository modules

- There are 2 main entities used throughout this app: a Board and a Pin, so we need 2 repositories: a Boards Repository and a Pins Repository.
- Each Repository handles creating a new item, inserting an item in the database, updating an item, and deleting an item (the 4 CRUD operations).
- Each repository needs access to the database in order to perform these operations.
- Careful: database operations can't happen on the main thread; we'll use app executors (a global executor pool for the whole application, see: https://github.com/android/architecture-components-samples/blob/master/PersistenceMigrationsSample/app/src/main/java/com/example/android/persistence/migrations/AppExecutors.java)
- To sum it up, each repository will need access to the app database and the app executors, which will be provided thanks to Dagger.
- Also, we need to ensure that each repo is a Singleton (see: https://en.wikipedia.org/wiki/Singleton_pattern). Dagger will help us with that too.

Task 6: Create the Boards View Model and the associated Fragment

- The Boards View Model will use the Boards Repository to access the database in order to save, update or delete the list of boards.
- Finish layout for Boards Fragment.
- Initially the fragment displays a message to the user "No boards yet".
- The user can create a new Board via the "CREATE BOARD" button which pops up a dialog for this purpose.
- Boards are displayed in a RecyclerView as either a list or grid, so we need to create a Boards Adapter.
- Handle click events: tapping on a specific board should display a secondary RecyclerView that displays the board's pins in a list or grid.

Task 7: Create the Pins View Model and the associated Fragment

- The Pins Repository handles creating a new pin, inserting a pin in the database, updating a pin, and deleting a pin.
- The Pins View Model will use the Pins Repository to access the database in order to save, update or delete the list of pins.
- Finish the layout for Pins Fragment.
- Pins are displayed in a RecyclerView as either a list or grid.
- Create Pins Adapter.
- Tapping on a specific pin should display the pin details.

Task 8: Creating and saving a new pin

- User can create a new pin with the following scenarios:
    - Taking a new photo
    - Picking an already existing photo
    - Recording a new video
    - Selecting an existing video
- For each of these cases, save the appropriate pin details in the database.

Task 9: Pin Details

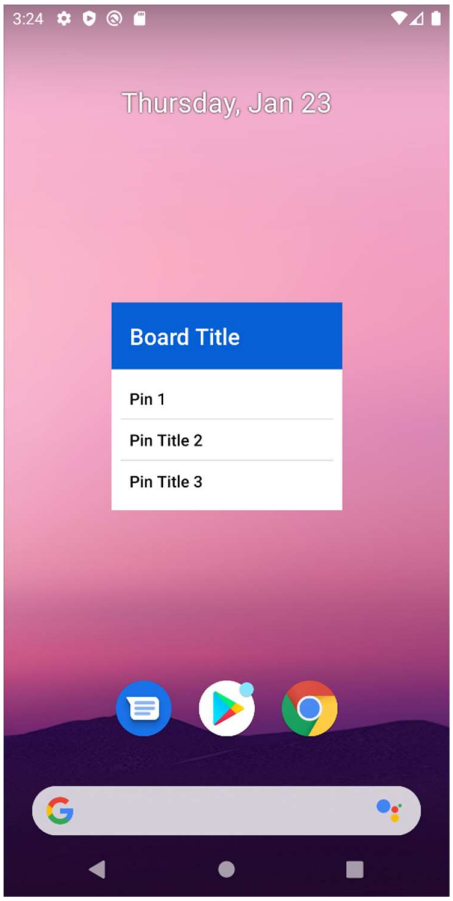
- Create the Pin Details View Model and the associated Fragment.
- User can favourite or unfavourite a pin.
- Handle creating new pin tags.
- Allow user to add pin comments.
- Users should be able to tap on a video to play it.
- Use ExoPlayer to display the video directly in the app (instead of playing it via an intent).

Task 10: Adding pins to boards

- After creating a new board, the user can select which pins to add to it.
- Create the Select Pins View Model and the associated Fragment.
- Careful when retrieving the pins that already belong to this board (the many-to-many Room relationship) when displaying the list of available pins.

Task 11: Create App Widget

- Create widget layout.
- Add the AppWidgetProviderInfo in XML and declare it in the Manifest.
- Create the RemoteViewsService and declare it in the Manifest.
- Create the RemoteViewsService.RemoteViewsFactory which will access the Database DAOs via Dagger injection.
- Use the RemoteViewsService to fetch the pins for a specific board from the database.
- Handle widget click events.



Task 12: Optional Features

- Implement a way for users to share pins using Android's Share intent.
- When selecting tags, create an easy way for users to select tags from existing tags they've already used.
- Allow users to sort and display pins by specific tags.