GitHub Username: CEThompson

Time Lapse Gallery

Description

A gallery application that stores photo sequences in order to track changes in a subject over time. This application will be written solely in the Java Programming Language.

Intended User

The intended user of the application wishes to track changes in a subject over time. A time-separated series of images could be used to track the height of a child, the progression of a fitness goal, or simply to watch plants grow.

Features

The main features of the application will be:

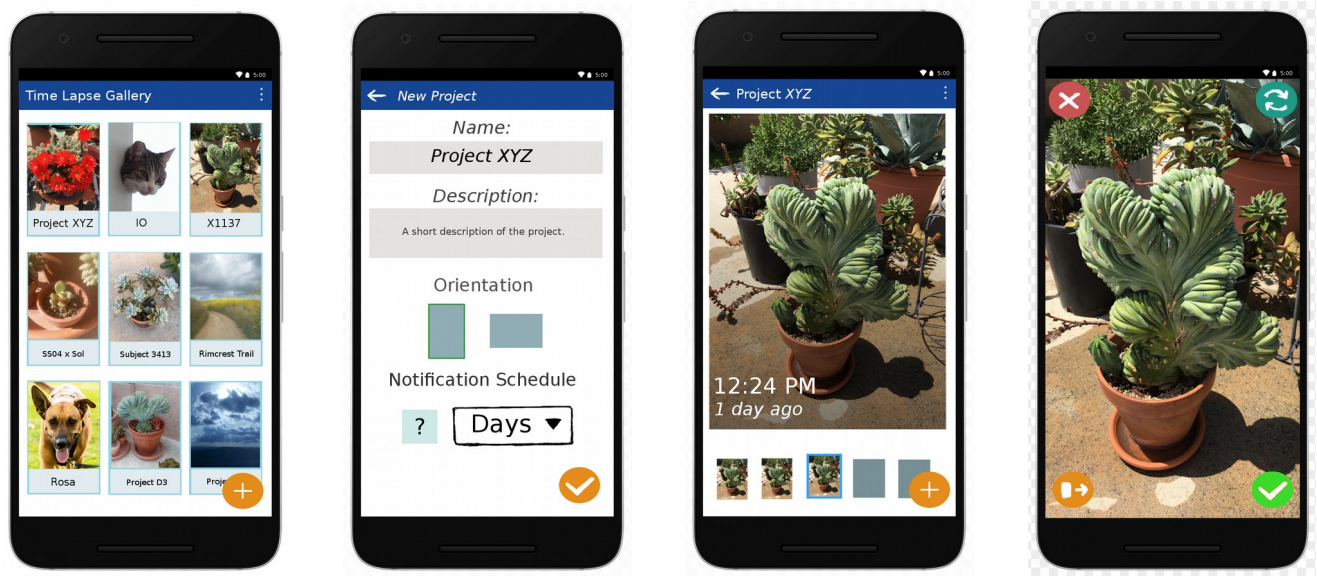
- To create and manage time-lapse projects
- To add sequential photos to projects

Additional features are possible for future development:

- Backing up photo sequences to the cloud
- Exporting photo sequences and data
- Importing photo sequences
- Sharing photo sequences
- Creating video / animations from the sequence of images

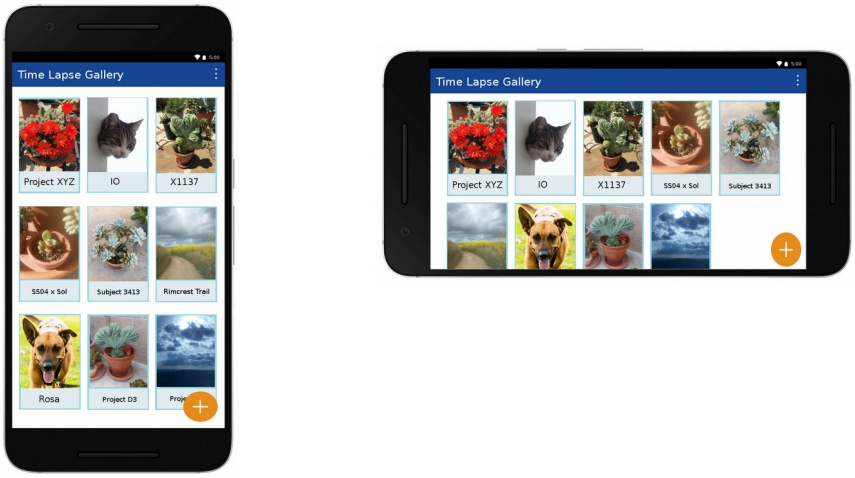
User Interface Mocks

Overview



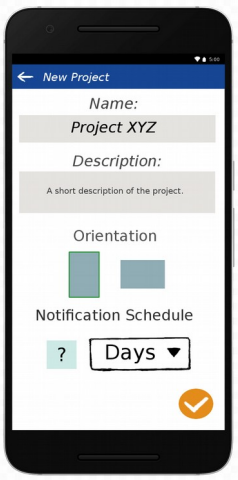
The user interface will consist of four screens and a call to a camera application. (1) The Main Activity will consist of a gallery of user created projects. (2) The New Project Activity will manage the creation of new projects. (3) The Details Activity will show the latest taken photo and the set of photos belonging to a project. (4) The Add Photo Activity will manage the process of adding a photo to the set.

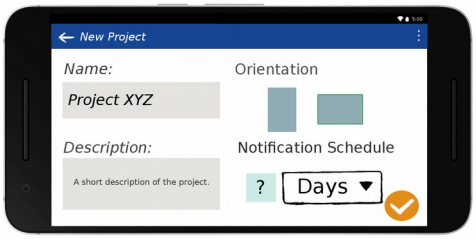
Screen 1 – Main Activity



The Main Activity will consist of a gallery of projects. A settings option will be provided to enable or disable notification for the entire application. A Floating Action Button will launch the New Project Activity. Clicking on an element in the gallery will launch the Details Activity for the selected project.

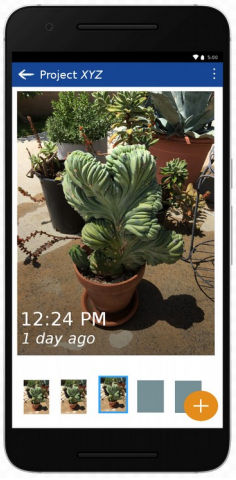
Screen 2 – New Project Activity

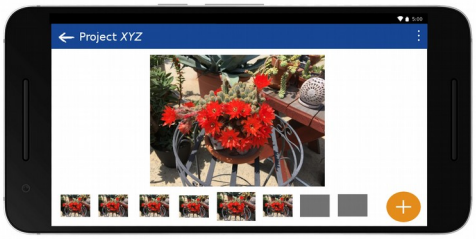




The New Project Activity creates a new project. Projects will require a unique name identifier. The description may be blank. Landscape versus portrait orientation should be determined in this stage for the project. The notification schedule shall default to NEVER, however the user shall be able to customize a notification schedule.

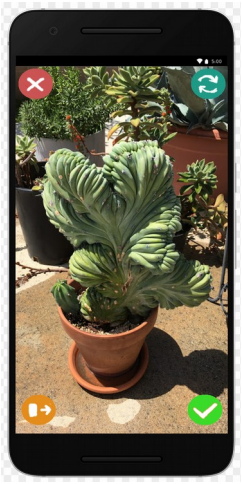
Screen 3 – Details Activity





The Details Activity will show the collection of photos for the project. A recycler view will be implemented to display thumbnails for the set of photos. A main view will display the last taken photo or a selection from the set of photos. A floating action button will launch the Add Photo Activity. Settings options will be provided to delete the current photo, edit the project, or delete the project. Editing the project will launch a modified version of the New Project Activity.

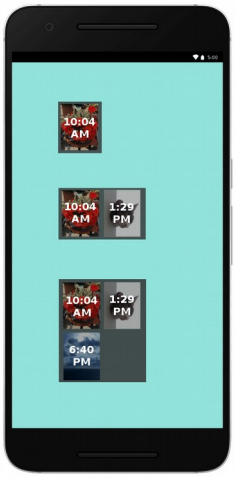
Screen 4 – Add Photo Activity



The Add Photo activity will launch an implicit intent and show the result to the user. The user will be presented with the result photo. A FAB in the lower left hand corner will be provided for quick comparison to the previous photo. A FAB will be provided in the upper left corner to cancel the activity (navigate up). Another FAB in the top right corner will be provided to retake the photo. Lastly, a checkmark FAB in the bottom right corner will be provided to accept the result. This activity shall be fullscreen and no toolbar will be displayed.

Widget

The widget shall display the next scheduled project and the time. Expanding the widget shall display additional projects for the day dependent upon available space. If there are no projects scheduled for the day a text view shall display “NO PROJECTS SCHEDULED FOR TODAY”.



Key Considerations

How will your app handle data persistence?

Room will be used to store local references to projects and photos. Settings options will be persisted as shared preferences. Photos will be saved in an appropriate file structure such as TimeLapseGallery/{$project_name}/{$image_number}.jpeg

A concrete example: TimeLapseGallery/Amy/00000001.jpeg

It will be perhaps preferable to use a timestamp as the filename rather than an ordered number sequence.

Describe any edge or corner cases in the UX.

Handling landscape vs portrait orientation for photos seems to be the primary edge case. For example: a user begins adding landscape photos to a portrait project. The simplest solution seems to be design the UI in such a way that it guides the user into the appropriate orientation. Therefore, if a project is a landscape project the Add Photo Activity launched from it will default to a landscape orientation.

This application is intended primarily for mobile usage. Tablet layouts will not be optimized for, but designs will be intended for general translation. Specifically no master/detail flow will be implemented.

Describe any libraries you'll be using and share your reasoning for including them.

- Butterknife will be used for view binding.
- Analytics will be used to monitor the application.
- Room will be used to persist references to the projects photo resources.
- Admob will be utilized to serve banner ads to the free version of the application.

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

- Analytics:
    - The service will need to be registered in the manifest, project-level build.gradle, application build.gradle, and an xml resource tracker will need to be implemented in order to monitor specific activities
- Admob:

- This service will require importing the Mobile Ads SDK by handling both project level build.gradle and application build.gradle
- Admob App ID will need to be added to the manifest

Next Steps: Required Tasks

Task 1: Project Setup

This will be a single module project. In order to set up the project the following dependencies will be utilized in Android Studio 3.2:

- Configure dependencies
    - Butterknife 10.1.0
    - Analytics
        - Project level build.gradle: Google Services 3.0.0
        - App level build.gradle: Play Services Analytics 10.2.4
    - Room 2.1.0
    - Admob
        - Play Services Ads 18.1.1
- Handle free and paid, as well as debug and release variants
- Declare relevant permissions, activities, and services in the manifest

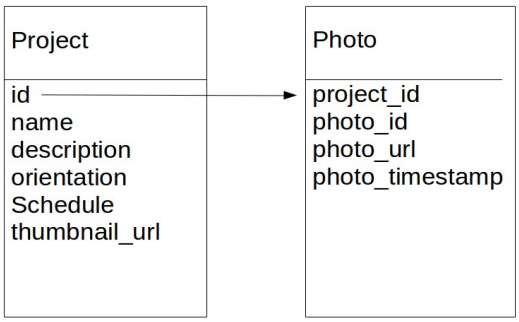
Task 2: Build the UI

1. Build the MainActivity
    - Bind a recyclerview to projects
    - Set up a Toolbar displaying the application name
    - Create a setting to disable / enable notifications
    - Set up FAB to launch the New Project Activity
2. Build the DetailsActivity
    - Get selected project information from main activity
    - Get references to the set of photos
    - Bind photo thumbnails to a recyclerview for a preview
    - Display latest taken photo in main photo view or selected photo
    - Display appropriate data for selected photo
    - Set up toolbar displaying project name
    - Set up all options

    - Build FAB that launches the Add Photo Activity
3. Build the AddPhotoActivity
    - Launch implicit camera intent for result
    - Display returned photo
    - Create cancel FAB
    - Create retake picture FAB
    - Create quick compare FAB
    - Create submit FAB
    - On submissions save photo to file structure, add references to database, and return to project details
4. Build the NewProjectActivity
    - Make edit text for the project name
    - Make edit text for the project details
    - Create selector for portrait vs landscape orientations
    - Create a spinner and a number edit text for the notification schedule
    - Create FAB that adds the project to the database

Task 3: Set up the database

This will require implementing a room database singleton, entry definitions for the database objects, a database access object, app executors to handle asynchronous access, and appropriate converters for any dates or timestamps. Additionally, view models, their factories, and live data will need to be implemented for any activities representing the database. This implementation will necessarily follow the pattern outlined in the popular movies application and during the class. The database shall implement the following schema.



Task 4: Build the widget

The widget shall display the notification schedule for projects. This will require implementing a widget update service, implementing a provider for the widgets, and implementing any remote views factories

and services in order to create the views for a list of projects and their schedules.

Task 5: Build the notification service

A notification service will be created that sends out notifications based upon the project schedules. The notification service shall be implemented using Job Dispatcher to schedule and execute notifications. When clicked the notification shall take the user to the appropriate project. The service shall be suspended if the main activity setting is disabled.

Task 6: Implement shared element transitions

A shared element transition shall be set up from the selected photo element in the Main Activity gallery to its corresponding view in the Details Activity.

Task 7: Handle Accessibility

In order to consider accessibility all relevant UI elements shall be appropriately labeled for use with TalkBack.

- ImageViews shall be labeled with the android:contentDescription XML attribute
    - Content descriptions for irrelevant graphical elements attribute android:contentDescription shall be set to “@null”.
    - For API level 16 or higher the android:importantForAccessibility shall be set to “no”.
- Editable elements shall be labeled using the android:hint XML attribute
    - For dynamic elements the setHint() method shall be used
- For API level 17 or higher the android:labelFor attribute should be used for labeling View objects that serve as content labels for other View objects

Task 6: Handle Project Resources

All colors shall be handled in a colors.xml resource file. Dimensions shall be handled in the appropriate dimens.xml resource file. All strings shall be handled in strings.xml. Project themes shall reference resources appropriately.
