
Description

Intended User

Features

User Interface Mocks

- MapsActivity (main screen)
- Photos Viewer
- Favorites Viewer
- Widget

Key Considerations

- How will your app handle data persistence?
- Describe any corner cases in the UX.
- Describe any libraries you'll be using and share your reasoning for including them.
- Describe how you will implement Google Play Services.

Next Steps: Required Tasks

- Task 1: Project Setup
- Task 2: Implement UI for Each Activity and Fragment
- Task 3: Implement MapsFragment
- Task 3: Create ContentProvider for saving locations and implement related methods in UI
- Task 4: Create AsyncTask for retrieving pictures
- Task 5: Implement delegate methods for PhotosView RecyclerView
- Task 6: Implement ContentProvider for photos and implement related methods in UI
- Task 7: Implement methods for favoriting photos
- Task 8: Implement methods for sharing photos
- Task 9: Create Widget UI
- Task 10: Implement Widget UI
- Task 11: Prepare for Release and Final Cleanup

GitHub Username: jwright798

Virtual Traveler

Description

Be the ultimate couch explorer. With the Virtual Traveler app, you can navigate the globe and view fantastic photos from Flickr users all over the world. Search for pictures from your current location, or search for that perfect destination. You can save your favorite pictures and view them from your home screen with the Virtual Traveler widget. Perfect for planning your photography tour, or getting the lay of the land before your next vacation.

Go explore today!

Intended User

Intended for world travelers and couch explorers alike.

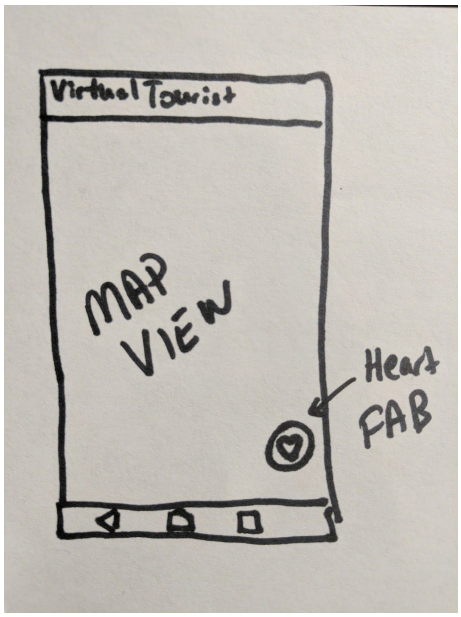
Features

The Virtual Traveler app will allow you to:

- Search the globe for a specific location or center on the user's current location via Google Maps and the Location API
    - These locations are saved so users always have reference to them.
- Search for pictures using the Flickr API
- Save favorite pictures for display on the widget (and in app via FAB)
- Share pictures to Facebook or Twitter or in a MMS message

User Interface Mocks

MapsActivity (main screen)



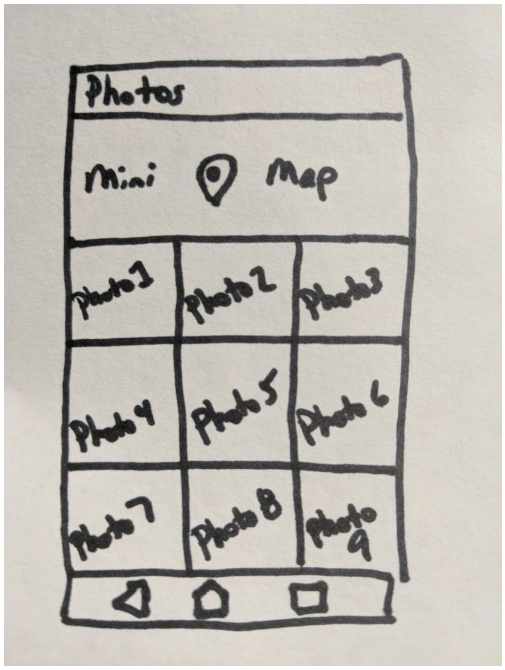
This first page is the main screen users will see when they launch the app. It will be a full screen map view, and users can tap on a specific point on the map to drop a pin for that location.

To remove the pin, they can tap and hold to remove the pin (after confirming on an alert)

To view photos related to that pin, just tap on the pin

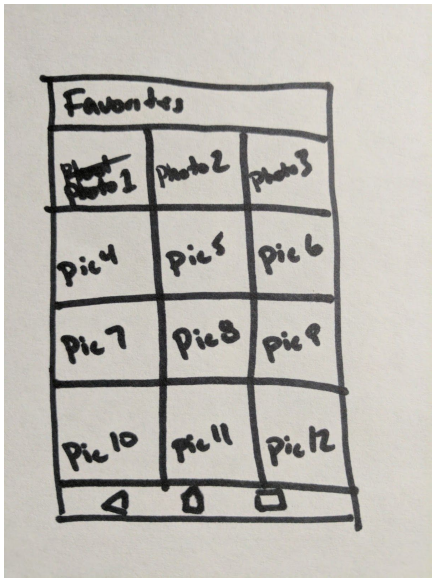
To view your favorite photo tap the heart FAB

Photos Viewer



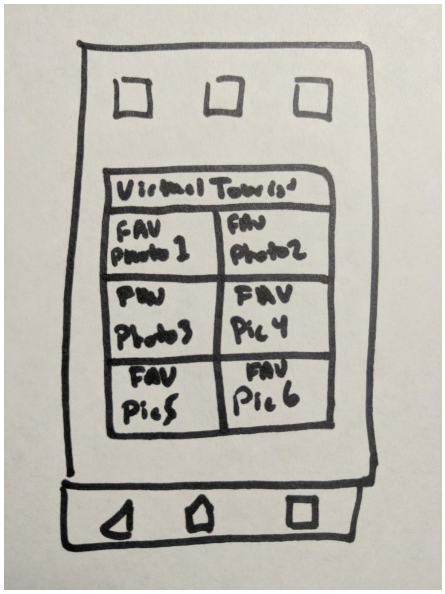
In this view you can see photos for that area (from Flickr). Double tap on a photo to save it to your favorites, and long press on a photo to bring up the sharing screen

Favorites Viewer



In this view you can see all of your favorite photos that you've saved across all of your locations.

Widget



This widget lets you view your favorited photos. Empty message appears if there are no favorited photos

Key Considerations

How will your app handle data persistence?

To handle storing the user's favorite photos (just urls) and saved locations, I will be creating 2 Content Providers, one for photos, one for locations

Describe any corner cases in the UX.

Users will tap and hold on a map location to set pins (Instructions will be in a dialog that will pop on first launch).

Tap the pin to be transitioned to the Photos view

Tap and hold on an existing pin to delete it. (Confirmation dialog will appear)
Tap and hold on a photo to launch the sharing screen

Double tap on a photo to save it to the favorites list.

Back button will take you from the photos view to the map screen

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

Images will be retrieved via the Picasso library

Describe how you will implement Google Play Services.

1. Location - This will be used to retrieve the currrent location in LatLong of the user. This will also be used for geocoding and reverse geocoding locations.
2. Google Maps - This is the key service since this will be used on the main screen to display a map users can navigate around (using the standard map view)

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

Task 1: Project Setup

- Starting with the Maps sample app from Android Studio
- Create Flickr account and get an API key
- Add Picasso library to gradle
- Set up permissions in AndroidManifest

Task 2: Implement UI for Each Activity and Fragment

First task is to create the base files(with layout files):

- Build file and base layout for MapsActivity
    - Reference to Fragment, FAB and toolbar
- Build file and base layout for MapsFragment
    - Maps layout
- Build file and base layout for PhotosActivity
    - Reference to PhotosFragment
- Build file and base layout for PhotosFragment
    - Contains RecyclerView

Task 3: Implement MapsFragment

For this task, I will be implementing the necessary delegate methods and creating the UI pieces for the MapFragment

- Implement Google Maps delegate methods (add maps to build.gradle)
    - Methods for adding pins, removing pins
    - Add dialog for removing pins
- Implement Location API delegate methods (add Location to build.gradle)
- Add FAB for accessing favorite pictures
- Add theme (extends AppCompat)
- Add toolbar (uses theme)
- Test for accessibility and make appropriate changes

Task 3: Create ContentProvider for saving locations and implement related methods in UI

Creating the locations ContentProvider

- Create LocationDO
- Create LocationsContentProvider class
- Implement ContentProvider methods for standard CRUD operations
- Add methods to the MapActivity so I can call ContentProvider
- Add unit tests

Task 4: Create AsyncTask for retrieving pictures

Creating the AsyncTask for retrieving images

- Create class
- Create PhotoDO
- Set up request with input validations and logging
- Create method for parsing JSON response
- Add validations for response and logging
- Create callbacks
- Add unit tests for JSON parsing

Task 5: Implement delegate methods for PhotosView RecyclerView

Setting up the RecyclerView and layout for the PhotosFragment

- Create layout
    - Add mini map layout (only show if non-favorite mode)
    - Add RecyclerView
- Add recycler view delegate methods
- Add Loader to load pictures into the views
- Set up recycler view for grid layout
- Add mini map methods for zoomed in view in selected location
- Disable user interaction for the map
- Add call to AsyncTask if it is not the favorites view
- Ensure accessibility standards are met and make any needed changes

Task 6: Implement ContentProvider for photos and implement related methods in UI

Creating the favorite photos ContentProvider

- Create PhotosContentProvider class

- Implement ContentProvider methods for standard CRUD operations
- Add check if photo is already favorited to avoid duplication
- Add methods to the PhotosActivity so I can call ContentProvider
- Add unit tests

Task 7: Implement methods for favoriting photos

Need to create ability to favorite photos

- Add gesture recognizer to photo item
- Add dialog for removing favorite
- Add unit tests

Task 8: Implement methods for sharing photos

- Add sharing intent

Task 9: Create Widget UI

Need to setup the initial layouts and get the widget started for viewing favorited photos

- Create Layouts
- Add Widget to Manifest
- Create file packages

Task 10: Implement Widget UI

More implementation tasks

- Add Remote views code
- Implement Picasso loading in Remote views

Task 11: Prepare for Release and Final Cleanup

Final steps and tasks for submission

- Set up certs and profiles for the installRelease Gradle task
- Ensure keystore is up to date and functional
- Add signing configuration
- Final dev testing

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"