Team member: Jonathan Espino

EID: jie296

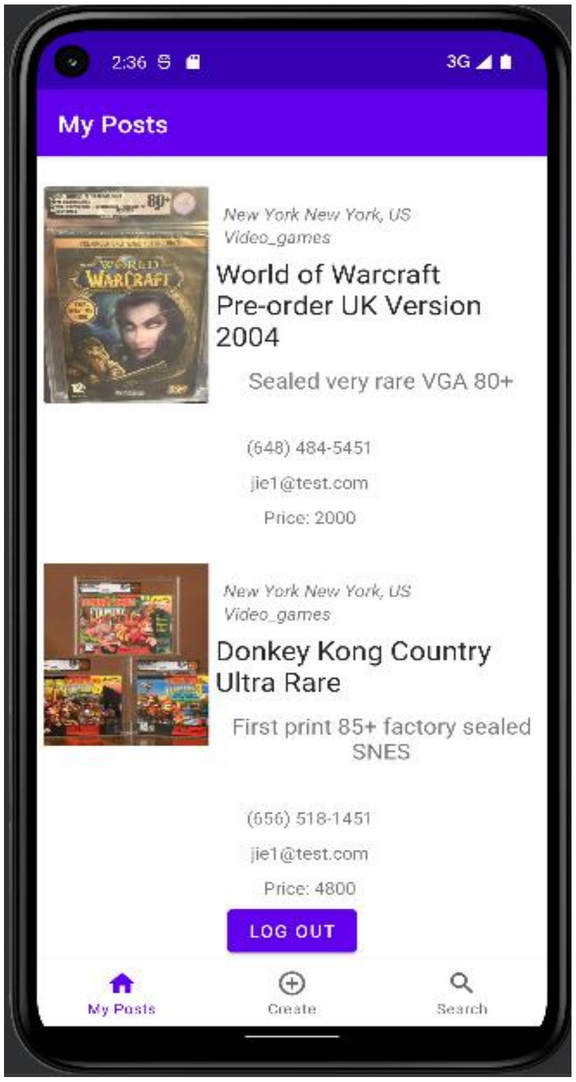
Email: Jonathan.i.espino@outlook.com

Application name: Bazaar

Description:

The application attempts to mimic the functionality of Craigslist. Users can create posts advertising products and list them and users can search for products listed by other users.

Viewing your posts:



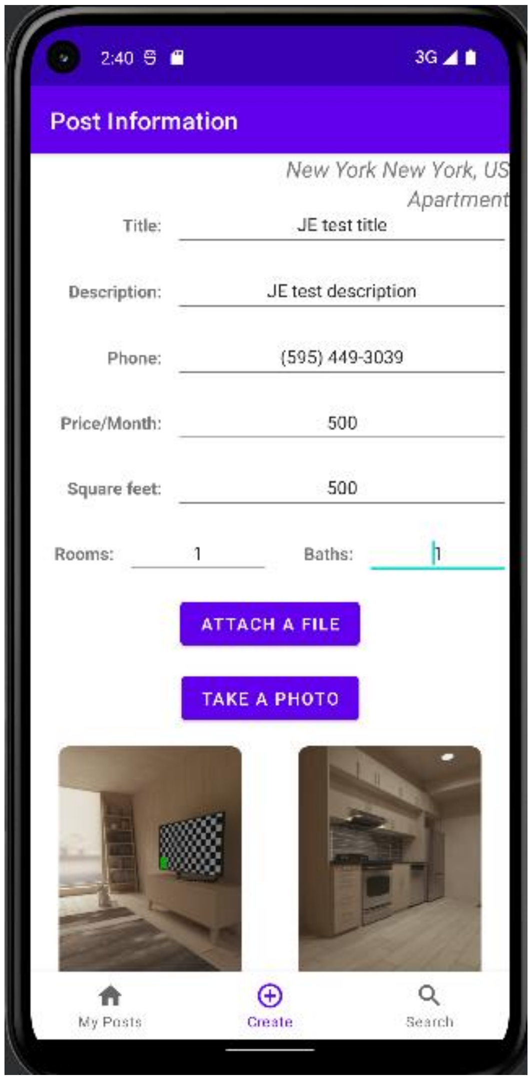
2:36 3G

My Posts

New York New York, US
Video_games
World of Warcraft Pre-order UK Version 2004
Sealed very rare VGA 80+
(648) 484-5451
jie1@test.com
Price: 2000

New York New York, US
Video_games
Donkey Kong Country Ultra Rare
First print 85+ factory sealed SNES
(656) 518-1451
jie1@test.com
Price: 4800

Creating a post:



2:40 3G

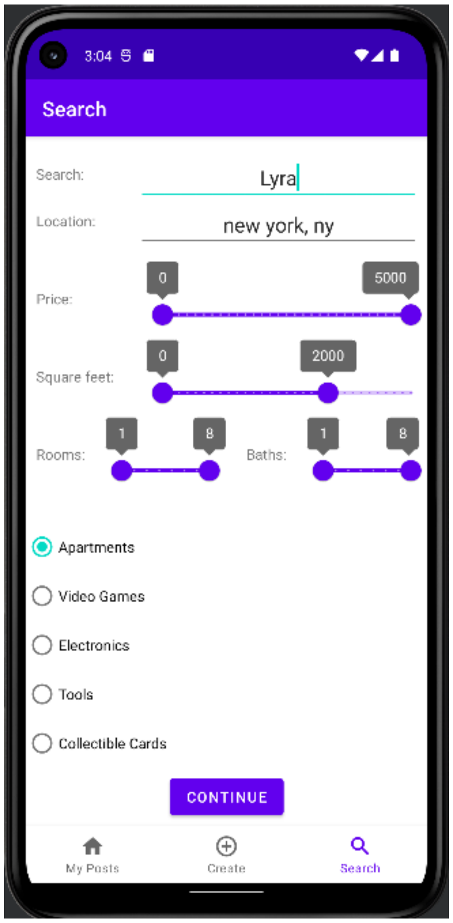
Post Information

New York New York, US
Apartment

Title:	JE test title
Description:	JE test description
Phone:	(595) 449-3039
Price/Month:	500
Square feet:	500

Rooms: 1 Baths: 1

Searching for posts:



Search: Lyra

Location: new york, ny

Price: 0 – 5000

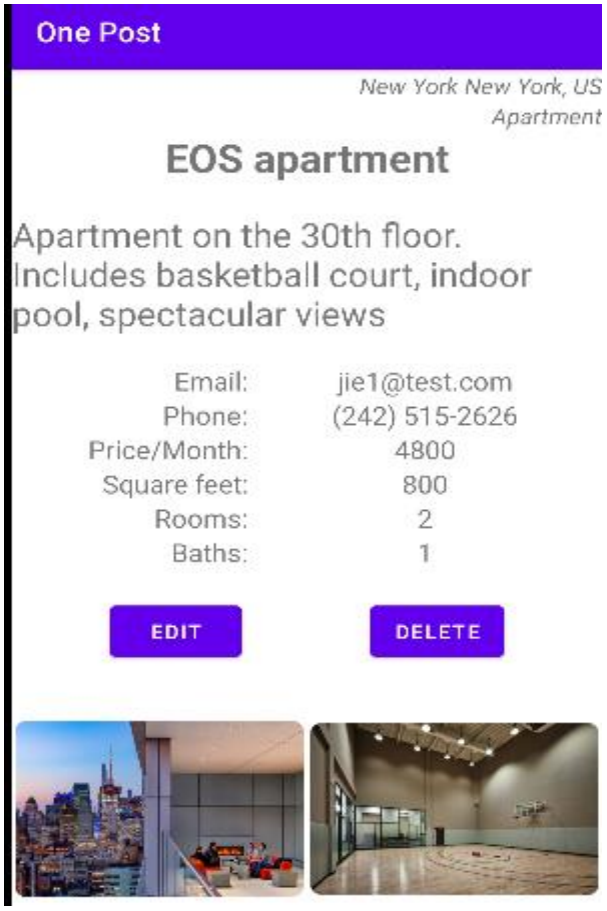
Square feet: 0 – 2000

Rooms: 1 – 8

Baths: 1 – 8

[✓ Apartments] Video Games Electronics Tools Collectible Cards

Single post detail view:



New York New York, US
Apartment

EOS apartment

Apartment on the 30th floor. Includes basketball court, indoor pool, spectacular views

Email:	jie1@test.com
Phone:	(242) 515-2626
Price/Month:	4800
Square feet:	800
Rooms:	2
Baths:	1

APIs:

Geocoder: used for getting a location from input user text.

Identity Toolkit, Token Service: Used for authentication.

Android features:

A navigation menu was used and a Navigation Graph was used to control the fragment flow. Most if not all the material we covered in class was used for the project (Intents, Fragments, LiveData, Firebase, etc.).

Third-Party Libraries:

Glide was used to display user images.

Third-Party Services:

Firebase's Storage, Authentication, Firestore Database.

Storage was used to store all the image data for created posts.

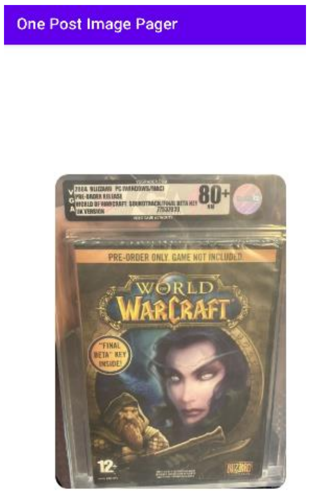
Firebase Authentication was used for logging in. Users must be logged in to use the application.

The Firestore Database was used to store user post data. Unfortunately, this Firebase feature is not super flexible. In the app, users can search for posts by search text, category, price, etc. The Firestore doesn’t have a good way of querying for matching substrings and the best workaround only allows for matching the starting characters of a word (e.g., in the phrase “Born in the USA”, “bor” would be a match but “orn” would not). Since this a bad solution I decided to retrieve results from the database and filter the results using Kotlin.

There were also issues when storing data. In the application users can view their own posts or search for posts using criteria. I originally wanted to store two database collections, one for posts by user and the other for posts split by category. This would’ve made searching faster and simpler, but doing this results in duplicate data and makes updating data more complicated because a post can be in multiple documents but only gets assigned one firestore ID. I didn’t find a good way to use this method correctly, so I decided to have one collection for all user posts. This is the simplest structure possible, allows for easily creating/updating posts, has the least amount of data, but isn’t the most efficient solution for retrieving user posts.

UI/UX:

The UI/UX for the project is straightforward and probably not worth going into detail. Constraint layouts were used for nearly everything. To display photos in user posts, I used similar logic to the FireNote demo. The only noteworthy thing would be the PagerAdapter that is used when a user clicks on a photo for a post. After a photo is clicked, the user can swipe to the right to see each individual photo for a post.



Back end:

The only thing out of ordinary here is not using filtering in the Firestore database query itself but instead doing this in Kotlin.

Interesting thing:

Probably the navigation functionality. There was a demo in class that used it, but I didn't realize how useful it was. There are eight fragments in this activity but with navigation they all interact with each other in a straightforward and simple way.

Most difficult challenge:

Unfortunately, the most difficult thing didn't make it inside the application. Allowing users to capture video and upload videos and allowing the user to play the videos while viewing a post. I was able to take video and store the data but couldn't get video playing to work inside the application. I did see some information online where others got video playing to work using Firebase, but this would've taken too much time for me to finish before the project deadline.

Besides this, there were some minor annoyances with uploading and taking photos in quick succession that caused errors (fixed by displaying a toast message in this scenario), handling database data correctly (mentioned in the Third-Party Services section) and showing the correct display with constraint layouts when display elements are shown or hidden in certain scenarios. However, there were no major issues that took several days to resolve.

Instructions:

Caution: the first time I launch the app sometimes I get a "grpc failed" error for Geocoder. I've never hit the error twice so relaunching the app should fix it.

On launching the app, you need to login. You can either create a new email address or use jie1@test.com, password 123456. This account has existing posts all associated with New York, New York, US.

There are three different tabs in the app, My Posts, Create, and Search.

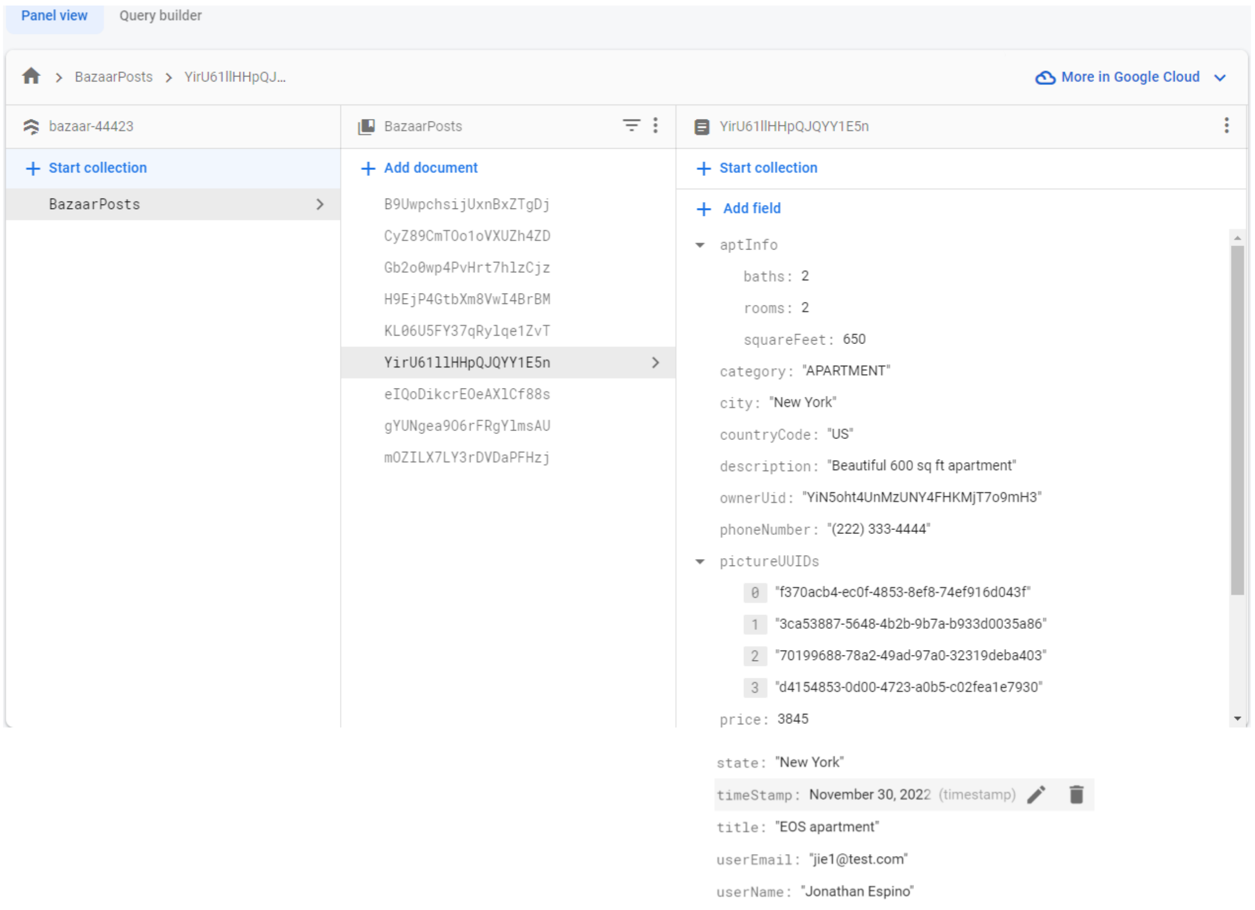
The My Posts tab will display all posts associated with your user; you can swipe up to refresh the list. You can select any post here to view it and see all the post's photos. After selecting it you can edit/ or delete the post. Editing allows you change all post information, long clicking on a photo will remove it from the post, and clicking the save button will update the database. Clicking a photo on the single post screen will display a blown-up version of the image and allow right-swiping through the photos.

The Create tab allows you to make a new post. After selecting the location and category, you can input the post information. Here you can take a new image or load one from your phone. Long clicking a photo will delete it. Selecting the save button will save the post in the database.

The Search tab allows searching for posts. The search text will look for the text in either the post title or post description, the input location will need to match the post's location (geocoder finds the location from the input text), and for the other criteria (price, square feet, etc.), the post will need a value within the selected range. After getting search results, clicking a post will show the post details and show all photos associated with it. Clicking the phone number here will launch a text message.

Note: there are five post categories and only "Apartments" has unique attributes (square feet, rooms, baths). The other four categories do not have these attributes.

Database Schema:



BazaarPosts:
B9UwpchsijUxnBxZTgDj
CyZ89CmTOo1oVXUZh4ZD
Gb2o0wp4PvHrt7hlzCjz
H9EjP4GtbXm8VwI4BrBM
KL06U5FY37qRylqe1ZvT
YirU611lHHpQJQYY1E5n
eIQoDikcrE0eAX1Cf88s
gYUNgea906rFRgYlmsAU
mOZILX7LY3rDVDaPFHzj

YirU61lIHHpQJQYY1E5n:

```
aptInfo
    baths: 2
    rooms: 2
    squareFeet: 650
category: "APARTMENT"
city: "New York"
countryCode: "US"
description: "Beautiful 600 sq ft apartment"
ownerUid: "YiN5oht4UnMzUNY4FHKMjT7o9mH3"
phoneNumber: "(222) 333-4444"
pictureUUIDs
    0 "f370acb4-ec0f-4853-8ef8-74ef916d043f"
    1 "3ca53887-5648-4b2b-9b7a-b933d0035a86"
    2 "70199688-78a2-49ad-97a0-32319deba403"
    3 "d4154853-0d00-4723-a0b5-c02fea1e7930"
price: 3845
state: "New York"
timeStamp: November 30, 2022 (timestamp)
title: "EOS apartment"
userEmail: "jie1@test.com"
userName: "Jonathan Espino"
```

Cloc output:

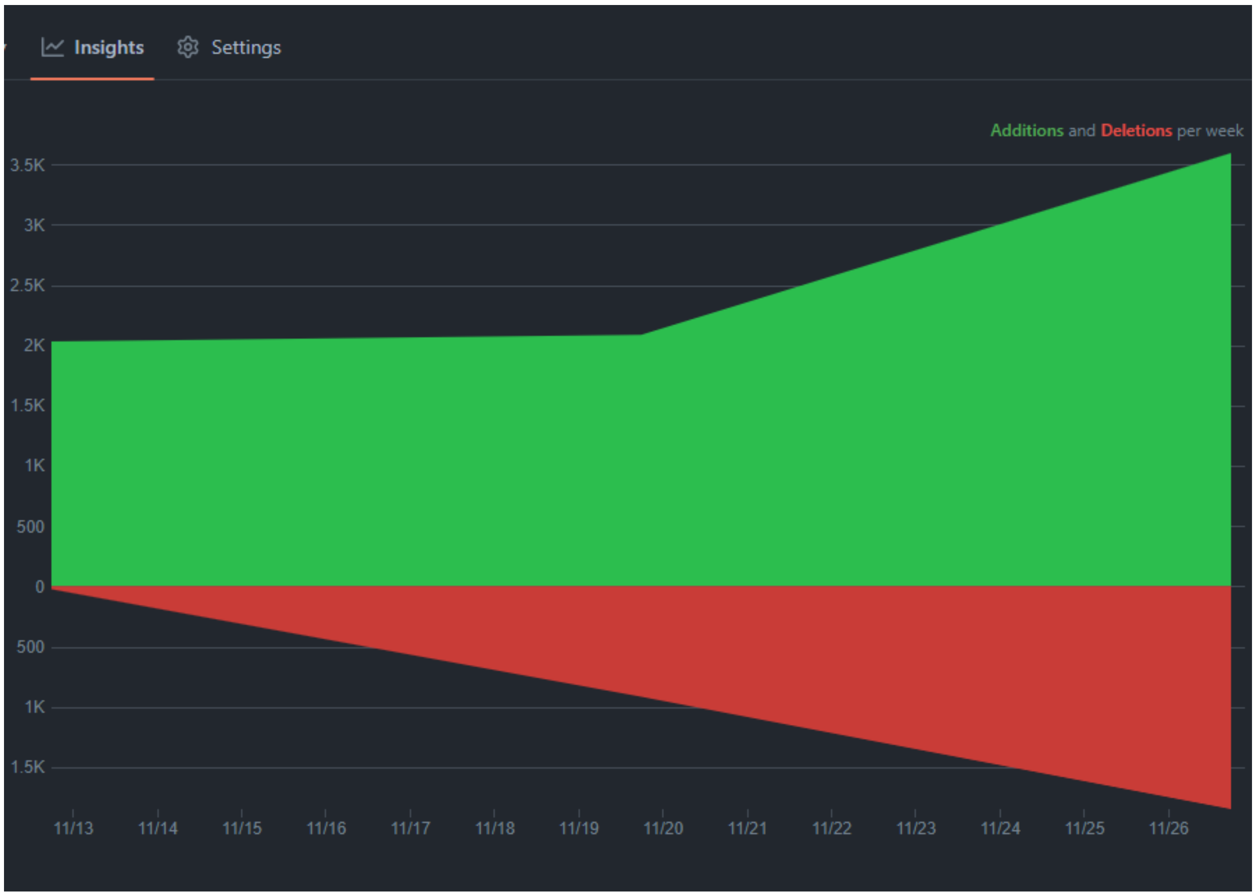
XML lines: 2126

Kotlin lines: 1490

Total lines: 3616

A lot of XML and Kotlin code was re-used in this project as many fragments shared the same code but had slight twists in their implementation. For the unique Kotlin code lines, I'd say you could reduce the total amount by 40%.

Code frequency:



Raw Cloc output:

```
-------------------------------------------------------------------------------------------------------------------
File                                                              blank    comment      code
-------------------------------------------------------------------------------------------------------------------
main/res/layout/fragment_create_post.xml                             13          0       279
main/java/com/example/bazaar/ui/myPosts/EditPost/EditPostFragment.kt 64          7       274
main/java/com/example/bazaar/ui/createPost/CreatePostFragment.kt     62          7       266
```

main/res/layout/fragment_one_post.xml	21	0	228
main/res/layout/fragment_search_criteria.xml	11	0	198
main/res/drawable/ic_launcher_background.xml	0	0	170
main/java/com/example/bazaar/Storage/DBHelper.kt	32	16	169
main/res/layout/search_result_row.xml	13	0	137
main/java/com/example/bazaar/ui/createPost/CreatePostViewModel.kt	32	27	121
main/java/com/example/bazaar/ui/myPosts/MyPostsViewModel.kt	28	27	117
main/java/com/example/bazaar/ui/search/SearchCriteriaFragment.kt	40	2	116
main/java/com/example/bazaar/ui/myPosts/OnePost/OnePostFragment.kt	24	2	96
main/java/com/example/bazaar/ui/search/SearchResultsAdapter.kt	14	1	91
main/java/com/example/bazaar/ui/myPosts/MyPostsAdapter.kt	16	1	88
main/java/com/example/bazaar/ui/search/OnePost/OnePostFragment.kt	20	2	87
main/res/navigation/mobile_navigation.xml	7	0	84
main/java/com/example/bazaar/Storage/Storage.kt	7	13	74
main/res/layout/fragment_criteria_create_post.xml	3	0	73
main/java/com/example/bazaar/ui/myPosts/MyPostsFragment.kt	14	6	71
main/java/com/example/bazaar/ui/createPost/PostCriteriaFragment.kt	21	3	70
main/java/com/example/bazaar/ui/search/SearchResultsViewModel.kt	14	2	66
main/java/com/example/bazaar/ui/search/SearchResultsFragment.kt	14	2	55
main/AndroidManifest.xml	3	0	51
main/java/com/example/bazaar/FireBaseAuth/AuthInit.kt	5	9	46
main/java/com/example/bazaar/ui/createPost/MediaAdapter.kt	5	1	44
main/java/com/example/bazaar/MainActivity.kt	13	7	42
main/java/com/example/bazaar/ui/myPosts/OnePost/OnePostMediaAdapter.kt	5	1	40
main/java/com/example/bazaar/ui/search/OnePost/OnePostMediaAdapter.kt	5	1	39
main/java/com/example/bazaar/glide/AppGlideModule.kt	3	5	38
main/res/layout/fragment_search_results.xml	5	0	36
main/res/values/strings.xml	12	0	35
main/java/com/example/bazaar/ui/search/OnePost/OnePostImagePagerAdapter.kt	11	20	31

```
main/res/drawable-v24/ic_launcher_foreground.xml                         0      0     30
main/res/layout/activity_main.xml                                        5      0     29
main/res/layout/photo_list_row.xml                                       4      0     28
main/java/com/example/bazaar/Model/UserPost.kt                           3      3     27
main/java/com/example/bazaar/ui/search/OnePost/OnePostImagePager.kt     11      1     25
main/java/com/example/bazaar/FireBaseAuth/FirestoreAuthLiveData.kt       4      0     24
main/res/layout/one_post_image_pager.xml                                 2      1     16
main/res/menu/bottom_nav_menu.xml                                        6      0     15
main/res/values/colors.xml                                               1      0     11
main/res/values-night/themes.xml                                         0      5     11
main/res/values/themes.xml                                               0      5     11
main/java/com/example/bazaar/ui/createPost/Category.kt                   1      0      9
main/res/drawable/ic_home_black_24dp.xml                                 0      0      9
main/res/drawable/ic_baseline_logout_24.xml                              0      0      5
main/res/drawable/ic_baseline_add_circle_outline_24.xml                  0      0      5
main/res/mipmap-anydpi-v26/ic_launcher_round.xml                         0      0      5
main/res/drawable/ic_baseline_arrow_forward_24.xml                       0      0      5
main/res/drawable/ic_baseline_search_24.xml                              0      0      5
main/res/values/dimens.xml                                               0      1      4
main/res/xml/data_extraction_rules.xml                                   0     15      4
main/res/xml/file_paths.xml                                              0      0      4
main/res/xml/backup_rules.xml                                            0     11      2
-----------------------------------------------------------------------------------------
SUM:                                                                   574    204   3616
-----------------------------------------------------------------------------------------

-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Kotlin                          26            468            166           2126
XML                             28            106             38           1490
```

```
-------------------------------------------------------------------------------
SUM:                     54          574          204         3616
-------------------------------------------------------------------------------
```