
GitHub Username: GregIzzo

Tag Sale Now

Description

'Tag Sale Now' helps you track tag sales in your area, list them by distance, tell friends when you've arrived at a sale, and let you rate the tag sale. When you arrive or rate a tag sale, your friends will be instantly notified. Planning to attend a tag sale this weekend? Check the 'attending' box, and your friends will be notified. Get notified when a new tag sale is added in your area.

Intended User

People who go to Tag Sales with friends.

Features

- Track Tag Sale dates/times
- Invite friends to attend a Tag Sale
- Receive notifications when your friends select ‘attending’ for an upcoming tag sale
- Share photos from Tag Sales
- Announce your own Tag Sale
- Rate Tag sales (up to 5 stars)
- Weather forecast for Tag Sale days

User Interface Mocks

Screen : Main Activity Screen

- Forecast fragment to show current weather conditions
- Tag Sale List - scrolling list of tag sales. Sort list by “Number of Friends Attending”, “Date”, “Distance”, or “Attending”.
    - Tap the “+” button to add a new Tag Sale (See “Add a Tag Sale” screen)
    - Tap on a tag sale entry line to see the detail screen (See “Tag Sale Detail” screen)
    - Tap on the ‘Im HERE’ heart to tell your friends that you’ve arrived at the tag sale.

- Friend List - scrolling list of users who have accepted a friend request.
    - Tap the “+” button to send a new friend request (See “Add a Friend” screen)
- “Find Nearby” - tap this button to view the Tag Sale List (See “Tag Sale List” screen), where the user can sort the tag sales based on distance
- Advertisements will appear along the bottom

Screen : Add A Tag Sale

- Enter details for a new tag sale - location, date, time, description. Check boxes to further describe the tag sale.
- Tap tool bar back arrow to return to Main Activity Screen

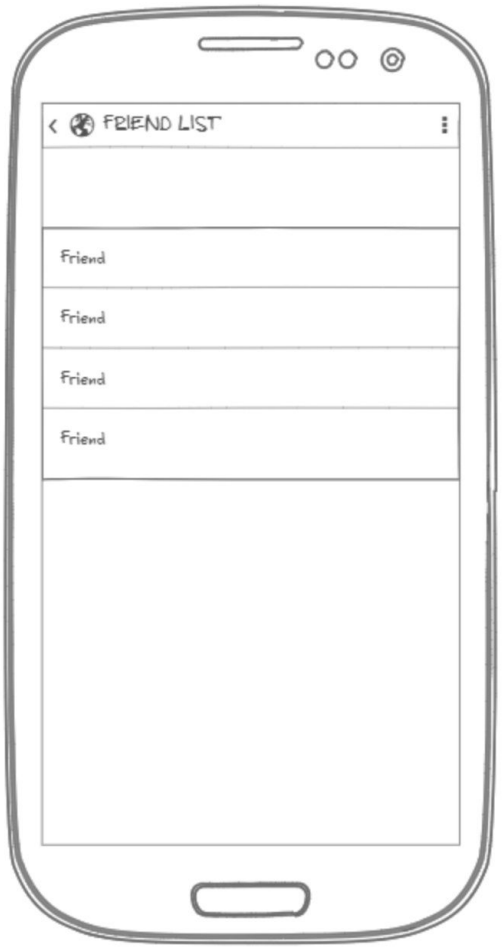
Screen : Tag Sale Detail

- Details for a tag sale are shown - location, date, time, description. Check boxes to further describe the tag sale.
- Tag Sale reviews appear in a scrolling list
- Tap the '+' button to add a new Review (See 'Tag Sale Review Entry' screen)
- A list of friends appear showing those that have marked the 'Attending' checkbox for this tag sale (i.e. they were planning to go), and those that are currently attending the tag sale (shown by a Red heart.)
- Tap tool bar back arrow to return to Main Activity Screen

Screen : Tag Sale Review Entry

- User can write a review of the tag sale
- User can rate the tag sale by selecting from 1 to 5 stars
- Tap the Toolbar back arrow to return to the Tag Sale Detail screen.

Screen : Friend List

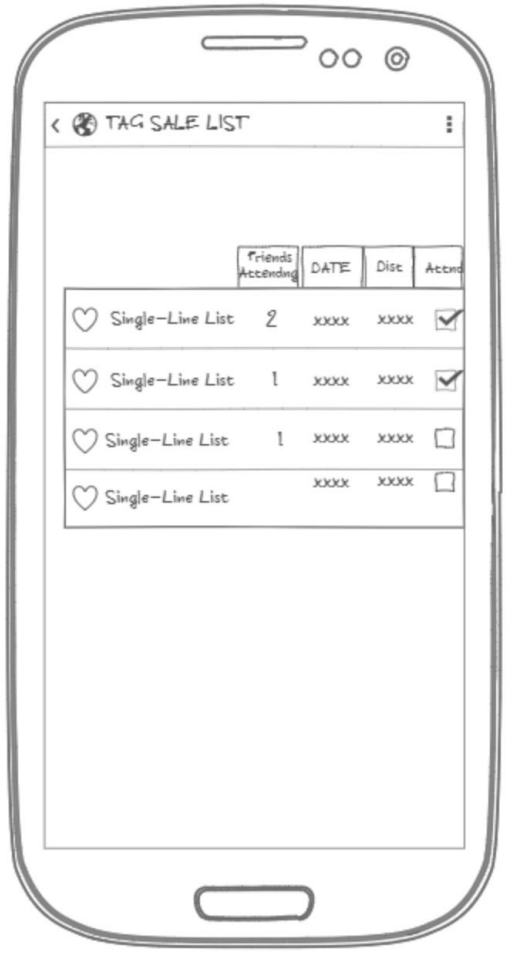


- A list of users who have accepted a friend request
- Tap the Toolbar back arrow to return to the Tag Sale Detail screen.

Screen : Send Friend Request

- Fill out your friend's email address, and (possibly) include a message which will appear in the email.
- (possibly) The code will check to see if this user is already in the user database but not yet a friend, it will send them a friend request push notification with an 'accept' button. If they tap on it a new friend record is created
- Tap the Toolbar back arrow to return to the Tag Sale Detail screen.

Screen : Tag Sale List



- List of all tag sales
- Column heading buttons allow user to sort by “Friends Attending”, “Date”, “Distance”, “Attending”
- The “Distance” sorting will use the GPS/Maps api to determine the distance from the user
- Tap the Toolbar back arrow to return to the Tag Sale Detail screen.

Screen : Home Screen Widget

- Homescreen Widget will display the 3 most recent tag sales
- Tapping on the widget will open the app to the Main Activity Screen

Key Considerations

- This app will be written solely in the Java Programming Language
- The app utilizes stable release versions of all libraries, Gradle, and Android Studio.
- The app will keep all strings in a ‘strings.xml’ file, to simplify string changes and to simplify localization
- To conform to accessibility standards, the app will include content descriptions, and navigation using a D-pad
- The app will update data in its cache at regular intervals using SyncAdapter
- The app will use IntentService to pull data from web services and API’s
- The app will use AsyncTask for tasks which should not use the main thread.

How will your app handle data persistence?

Describe how your app with handle data. (For example, will you build a Content Provider or use Firebase Realtime Database?)

- Firebase Realtime Database. It will need to grow dynamically as the number of people, tag sale events, rankings increase.
- Also, I'll use Room Library to maintain a local DB

Describe any edge or corner cases in the UX.

The screen layout and navigation is simple in this app, and I have not been able to find any edge/corner cases where navigation might be ambiguous.

Lack of Internet Connection

- If the internet connection is missing, then message will appear at the very top of the screen saying, "Internet is not available. Some functions may not work"
- In this situation, the app will rely on it's local DB for information.
- If the 'find nearby' functionality fails, the app will display a Toast message saying the internet is unavailable
- If the "GPS" functionality is unavailable, then a Toast message will be displayed saying "GPS is unavailable"

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

For example, Picasso or Glide to handle the loading and caching of images.

- Picasso - image display/handling. It's easy and free to use, and is used by many people
- Room - to maintain a local DB so the app will work when network isn't available
- Firebase Messaging - to notify a user when a friend joins, chooses to 'attend' a tag sale, or reviews a tag sale
- Firebase DB - to keep data for the app in the cloud. One of the main benefits for the app is to share information about tag sales.
- Firebase Ads - to show ads at the bottom of the main activity
- Google Play Service - Maps - To allow the app to show tag sales near to the user's GPS position.

Library/Software Versions (using stable release versions):

- Picasso : implementation 'com.squareup.picasso:picasso:2.71828'
- Room:
    - implementation 'android.arch.persistence.room:runtime:1.1.1'

    - annotationProcessor ‘android.arch.persistence.room:compiler:1.1.1’
- Firebase Messaging : implementation 'com.google.firebase:firebase-messaging:17.3.2'
- Firebase DB : implementation 'com.google.firebase:firebase-database:16.0.1'
- Firebase Ads : implementation 'com.google.firebase:firebase-ads:15.0.1'
- Google Maps: com.google.android.gms:play-services-maps:15.0.1
- Android Studio 3.2
- Android Gradle Plugin 3.2.0
- Gradle 4.6

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

Describe which Services you will use and how.

- Google Maps - to provide a map, or ‘nearby’ functionality for tag sales.
- Cloud Messaging for notifying users when their friends select ‘attending’ for a future tag sale, or they ‘check in’ to a tag sale that’s underway
- Firebase DB - to maintain a cloud DB of tag sale data. This will make the data available to all app users.
- Firebase Ads - show Admob ads at bottom of main activity
- OpenWeatherMap.org - Use api to retrieve weather info for given location

Next Steps: Required Tasks

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

1. Create class for retrieve/handle ‘Tag Sale Event records’
2. Create class for retrieve/handle ‘user records’
3. Create class to retrieve/handle ‘Friend records’
4. Create class to retrieve/handle ‘Attending records’
5. Create class to retrieve/handle ‘Tag Sale Rating records’
6. Create class to handle GPS location services (capture phone location, ‘Find nearby’,etc)
7. Create class to retrieve Forecast for a given Zip code
    a. Use api from OpenWeathermap.org
8. Create fragment for ‘Tag Sale List’ to display list of ‘Tag Sale Event Records’.
    a. Select a Tag Sale to show details (location, time, etc)
    b. ‘add new record’ functionality, which displays the detail screen and allows entry
        i. Entry fields for details (address, time from-to, etc)
        ii. List of checkboxes or keywords for user to select, to describe the tag sale (furniture, clothing, multifamily, etc)
    c. Show icon for ‘attending’. Allow user to mark records as ‘attending’ or not.

    d. ‘Find Nearby’ function- Use phone GPS location to sort Tag Sale Event records by distance
9. Create fragment to create Tag Sale Review
    a. Text entry
    b. Rating interface (1-5 stars are selectable)
10. Create fragment to display ‘Friends list’,
    a. Button to ‘send friend request’. Show detail screen to enter friend email, then launch email intent.
11. Create ‘Forecast’ fragment
    a. Pull current weather data given a location (zip code or GPS) and display info.
12. Create Main Activity
    a. Integrate Forecast fragment
    b. Integrate Tag Sale List fragment
    c. Integrate Friend List fragment
13. Connect ‘Firebase Realtime Database for Android’
14. Integrate ‘Firebase Cloud Messaging’ for notifications when friends select ‘attending’ for a given tag sale.

Task 1: Project Setup

- Setup a new Project in Firebase
- Setup Firebase DB from web console
- Setup Firebase Messaging on web console
- Use these guides to setup basic files/config for Firebase DB and cloud messaging:
    - https://firebase.google.com/docs/cloud-messaging/android/client
    - https://firebase.google.com/docs/database/android/start/
- Setup account at openWeatherMap.org
- Obtain Google Maps API key
- Sign Up for Admob account

Task 2: Implement UI for Each Activity and Fragment

- Build UI for each Fragment:
    - Tag Sale List - display many tag sale records
    - Tag Sale Entry - create a tag sale record
    - Tag Sale Detail - show details about a single tag sale, enter/change rating, add review
    - Friends List - display many friend records
    - Friend request - enter information to bundle with an external email intent to allow a friend to get the app

    - Weather Forecast fragment - display current weather forecast
- Build UI home screen widget
- Build Main Activity UI - displays ‘Weather Forecast’ fragment, ‘Tag Sale List’ fragment, and Ad Mob ad.

Task 3: Database Tables

- Create tables:
    - Tag Sale Event
    - User
    - Friend
    - Attending
    - Tag Sale Rating
- Create classes to interact with each table

Task 4: GPS

Create GPS util class to

- translate zip code to GPS, and GPS to zip code
- Compute distance from a given location to each tag sale record. This will be used to find ‘nearby tag sales’

Task 5: Weather Forecast Utility

Create Weather Forecast class to:

- Retrieve current forecast for a given place (GPS coords, zip, etc) and date.

Task 6: Main Activity - connect to database

- Combine fragments to create main activity
- Retrieve data and populate fragments
- Connect ‘Tag Sale List’ fragment, and build functionality.
- Connect ‘Tag Sale Entry’ fragment and build functionality.
- Connect ‘Tag Sale Detail’ fragment and build functionality

Task 7: Add-Friend Functionality

- Link Friend-Request fragment
- Code and Test the request functionality based on implicit email intent.
- Connect 'Friend List' fragment and build functionality

Task 8: Push Notification

Create app code and server setup to send notifications:

- When user clicks 'attending' button for a given tag sale, send notifications to all friends
- When user enters a rating for a given tag sale, send notifications to all friends.

Task 9: Add 'Ad Mob' advertisements

- Connect the app to Ad Mob
- Add code to display add at bottom of main activity.

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
    - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"