CS 501 Mobile Application Development

Spring 2023

Assignment 4 – Fragments, RecyclerView, and Beginning Semester Project

**Due:** Night of lecture 5, before class.

Do not email homework, please submit on Blackboard. You should also take screenshots or recording of your assignment and submit. Post the code on GitHub and submit the link and the videos in a zip file. Other than the reading, this is a group assignment. When working as a group, do not divide and conquer, do not assign one problem to one or two people. Everyone must work on each problem. Please be sure to include your team’s name and members for the group submission. Only one group submission per team please.

Note the programming assignments will require you to do some research by your team. And, as before, unit and instrument tests (at least 2 each) must be included with your apps.

**Assignments:**

1. **(Individually)** Read all of chapters 11, 12, and 13 before the next class.
2. **(Team)** Implement the challenge in chapter 10 of the book. That is:

* Using the CriminalIntent example reviewed in class, create two types of rows in your RecyclerView: a normal row and a row for more serious crimes. To implement this, you will need to work with the view type feature available in RecyclerView.Adapter.
* Add a new property, requiresPolice, to the Crime object and use it to determine which view to load on the CrimeListAdapter by implementing the getItemViewType(Int) function (developer.android.com/​reference/​androidx/​recyclerview/​widget/​RecyclerView.Adapter#getItemViewType(int)).
* In the onCreateViewHolder(ViewGroup, Int) function, you will also need to add logic that returns a different ViewHolder based on the new viewType value returned by getItemViewType(Int). Use the original layout for crimes that do not require police intervention and a new layout with a streamlined interface containing a button that says “contact police” for crimes that do.

1. **(TEAM)** Create a Hangman Game. Target both a Portrait and Landscape views as shown below. This is a design challenge as well, so please make design choices on your own, (eg.,- how to render the hangman stick figure, how to tell the user they won or lost, etc.) Your game should be able to run at least one complete game and handle both winning and losing, no scoring is necessary. Include a "New Game" Button which will reset the current game and start a new and different game when clicked.

Your game must remember state when rotating!

Hard code the words for the game, but randomly present different words when run. Ensure the maximum length of any word is 6 characters. This will simplify rendering the word on the screen.

Panel 1: Contains a "Choose the letter" set of buttons. Provide a menu of letters, just use buttons. Be sure to disable the button after it's been selected.

Panel 2: Contains a Hint Button. Clicking the Hint Button can do several things:

* The first time it is clicked it displays a hint message.
* The second time it is clicked it disables half of the remaining letters (that are not part of the word) BUT it costs the user a turn.
* The third time it is clicked, it shows all the vowels, BUT it costs the user a turn. Be sure to disable all the vowel buttons so they user doesn't click them again.
* If clicking the hint button would cause the user to be completely hung, ie, lose the game, then show a toast, "Hint not available".

Panel 3: Contains the Main Game Play Screen.

General Game Play:

After choosing a letter, the user will be notified whether the letter was correct or not. If correct the letter will be displayed in the proper area, if incorrect, another body part will hang. The game ends when the user successfully identifies the word or is completely hung. IMPLEMENT ONLY THE REQUIREMENTS ABOVE.

REMEMBER: you must pass requests between fragments through the parent activity, using an interface. This ensures “loose coupling”. Fragments should not talk to each other directly.

A picture containing graphical user interface

Description automatically generated A screenshot of a computer

Description automatically generated with low confidence

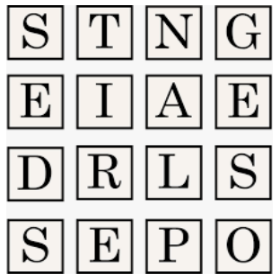
1. (Team) Project proposal. Submit a proposal for what your semester project app will do.

* List the features and capabilities is the form of:
  + Scenarios (the plays that users will do with your app)
  + Personas (the actors in the play)
  + Stories (the specific features that when combined, create a scenario)
    - Note that stories may be made up of multiple tasks.
    - Think Agile – work in sprints.
* Identify at least 2 meaningful APIs that you will use in your app. Describe how they will be used to support the functionality described in the scenarios.
  + Examples of APIs are
    - Augmented Reality
    - Drawing a path on a map to track a user's movement (eg, while jogging).
    - Pull JSON data from Amazon or eBay or Walmart
    - Interact meaningfully with Google Earth's API
    - Interact with Fitbit or other fitness tracker AP
    - Interact with Spotify's API to play music
    - Interact with payment services like PayU
    - IBM Watson's SDK
    - TicketMaster/Eventbrite
    - Image Recognition
* Write up your proposal as a presentation which each team will give in the next class.

1. (Team) For fun! **Simple Boggle**

Boggle is a simple word game that involves forming as many words as you can from a grid of letters. Sort of like a word search.

For example, given the following 4 x4 block of random letters, you can form the word “POSE”, by clicking on the letters, ‘P’, ‘O’, ‘S’, and ‘E’ in order.



**A few other words in the example above: “TIRE”, “REPO”, “TIED”.**

For this task you will implement a modified Boggle game.

**Read these Rules Carefully:**

* Your first letter can start anywhere.
* Subsequent letter connections must “touch” the previous letter. This can occur in any direction, including diagonally.
* You may only use a letter in the current word once (no doubling back or reconnecting).
* All words must utilize a minimum of two vowels.
* You cannot generate the same word more than once, even if it’s from different letters.
* Words must be at least 4 chars long.
  + For example, the word “SLOPE” above would be valid. Another valid word above would be “PRIDE” (*See if you can find it above.*)

**For clarity some things that do NOT qualify as words, based on the rules are: “PLANT” “STING”, AIR, ….**

Scoring:

* Every consonant in a word is worth one point.
* Every vowel is worth 5 points.
* If you use one or more of the consonants ‘S’, ‘Z’, ‘P’, ‘X’ or ‘Q’ the value of the word is doubled.
* Incorrect responses reduce the score by 10 points, regardless of word length.
* There is no time limit.

Word List:

* Download a word dictionary to check answers.
* You can use any list of dictionary of words you like, many abound on the web, eg, <https://raw.githubusercontent.com/dwyl/english-words/master/words.txt> . No need to use a Database, just check if the users answer is in the dictionary.

GAME PLAY:

* User selects a word in the grid.
* User hits Clear or Submit.
* If the user hits Clear, the word entry so far is removed with no penalty.
* If the user hits Submit, the entry is checked against the dictionary.
* If it is incorrect, 10 points are deducted from the score.
* If it is correct, the proper number of points are added to the score, according to the rules above.

OTHER:

* Keep things simple, use toasts to communicate with the user, eg, “That’s correct, +4”, “That’s incorrect, -10”, “You may only select connected letters”, etc.
* Again, Users should only be allowed to click on contiguous letters.
* You can hardcode the layout or generate it dynamically.
* The letters must be different each time.

IMPLEMENTATION REQUIREMENTS:

* Design the game using the template suggested.
* You must use fragments, these are identified with red boxes around the template.
* As discussed in class, fragments may not communicate directly.
* The two fragments must pass messages through an interface to the Main Activity.
* Do not create separate layouts for phones and tablets. **Just one layout in Portrait Mode.**
* Be considerate to the user. Apply simple “UI Polish” to the game, eg, after a letter is selected, color the button differently and disable it, so the user won’t accidentally select it again. You must think about other opportunities for making the UI user friendly yourself.

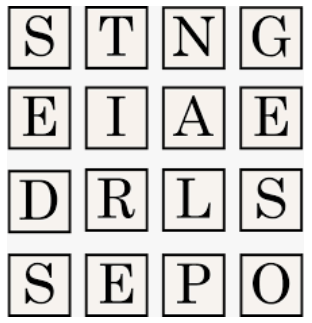
SCORE: 6

NEW GAME

Enter a word then hit submit.

P O S E

CLEAR SUBMIT



GAME TEMPLATE, PORTRAIT MODE ONLY

**Note: To save time, lock the orientation in portrait mode. Don’t worry about saving state when the App is destroyed.**