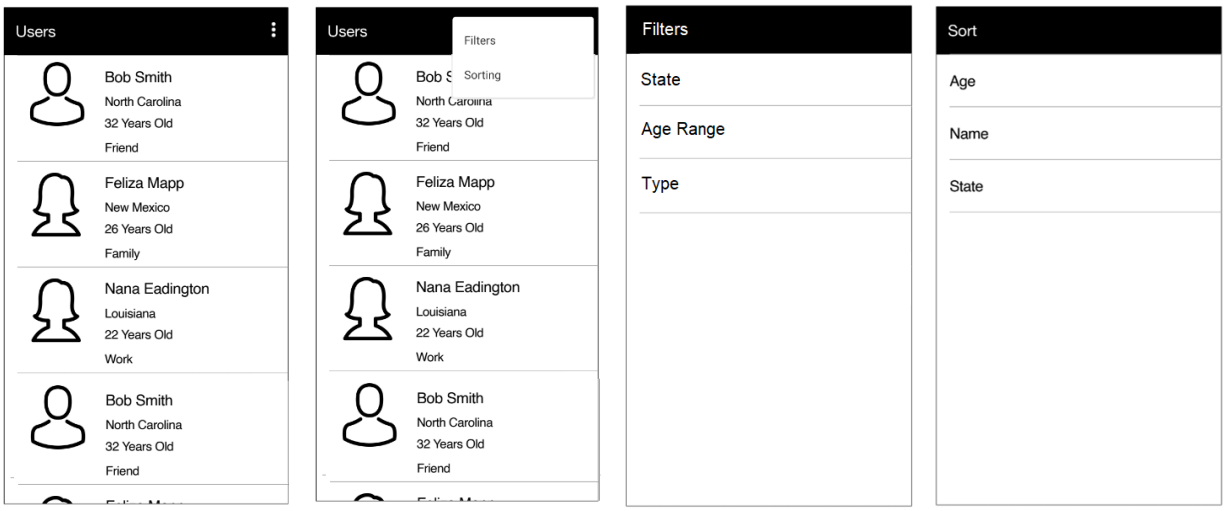
Mobile App Development
Midterm Exam

Basic Instructions:

1. This is the Midterm Exam, which will count for 20% of the total course grade.
2. This Midterm is an individual effort. Each student is responsible for her/his own Midterm and its submission.
3. Once you have picked up the exam, you may not discuss it in any way with anyone until the exam period is over.
4. During the exam, you are allowed to use the course videos, slides, and your code from previous home works and in class assignments. You can use the internet to search for answers. You are NOT allowed to use code provided by other students or solicit help from other online persons.
5. Answer all the exam parts, all the parts are required.
6. Please download the support files provided with the Midterm and use them when implementing your project.
7. Your assignment will be graded for functional requirements and efficiency of your submitted solution. You will loose points if your code is not efficient, does unnecessary processing or blocks the UI thread.
8. Create a zip file which includes all the project folder, any required libraries, and your presentation material. Submit the exported file using the provided canvas submission link.
9. **Do not try to use any Social Messenger apps, Emails, Or Cloud File Storage services in this exam.**
10. **Failure to follow the above instructions will result in point deductions.**
11. **Any violation of the rules regarding consultation with others will not be tolerated and will result disciplinary action and failing the course.**

Midterm Exam (100 Points)

The app provides features to list users, filter and sort. Figure 1 shows the main screens.



(a) Users Fragment (b) User Fragment options (c) Filters Fragment (d) Sorting Fragment

Figure 1. Application Wireframes

Main Requirements and Data:

1. This app should contain only **ONE** activity and all the screens should be implemented using fragments.
2. You are provided with DataServices class that includes a method to retrieve a list of User objects. Each user has a name, age, gender, group and state. You should include the DataServices.java class in your project and not modify it.

Part 1: Users Fragment (40 Points)

The interface should be created to match the UI presented in Figure 1(a). The requirements are as follows:

1. This fragment should display the list of users as shown in Figure 1(a). This fragment is the main fragment that should be displayed when the app starts.
2. You have to retrieve the users by calling the method getUsers() from the DataServices class, which emulates a network connection so **it should be called in a worker thread**. You are free to use Thread Pool or AsyncTask.
3. You are free to use ListView or RecyclerView to implement the users list. Note that for each row item you should include the user name, state, age and group. In addition the user icon should be selected based on the user's gender as shown in Figure 1(a).
4. An options menu should be implemented at the top bar, which will contain the "Sorting" and "Filters" options, as shown in Figure 1(b).
   a. For more information about menus, please refer to https://developer.android.com/guide/topics/ui/menus or any other documentation or site that you find appropriate for this matter.

   b. Keep in mind that adding options to the android bar is a very common practice in Android.
   c. The options should be visible only when in the Users fragment.
5. Clicking the “Filters” option should display “Filters” fragment and place the current fragment on the back stack.
   a. Upon selecting a filter, the selected filter criteria should be returned to the Users fragment and the list should be filtered based on the selected criteria.
   b. Filtering should make sure to maintain the preselected sorting order of the users.
6. Clicking the “Sort” option should display Sort fragment and place the current fragment on the back stack.
   a. Upon returning from the “Sort” fragment, the selected sort criteria should be returned to the Users fragment and the list should be sorted based on the selected criteria.
   b. Note that the “Sort” fragment does not affect the filter criteria selected if any.

Part 2: Filters Fragments (40 Points)

The interface should be created to match the UI presented in Figure 1(c), and each option will lead to a fragment for that specific filter, as shown in Figure 2. The requirements are as follows:

1. Upon selecting a filter value, for example a state or an age range, the Users fragment has to be presented again to the user, and the respective filter should be applied.
2. For example, showing only the Users that match the selected state, or that meet any provided criteria.
3. Note that for going back to the Users fragment, one “popFromBackStack()” might not be enough.

4. Filter by state:

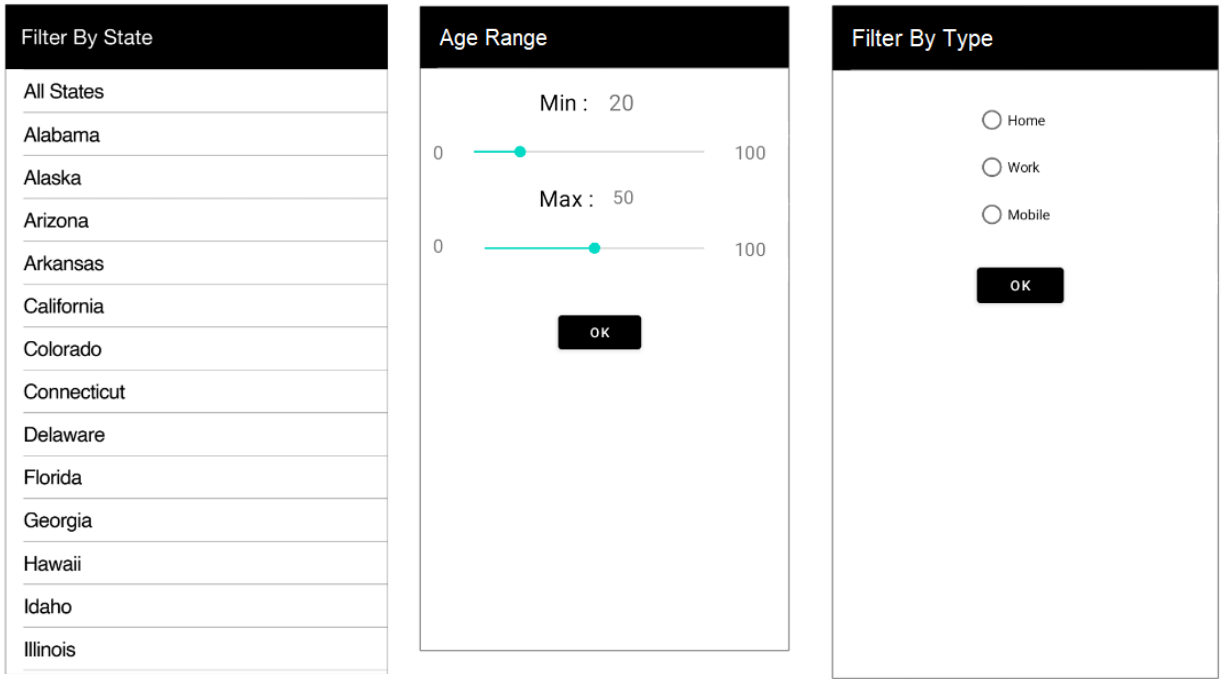
   a. The list of states that will be used for populating this fragment has to be calculated from the distinct states available across all the users.
   b. Manually creating and typing a list of states is not allowed.
   c. The list of unique states should be sorted in ascending order and the top row should include “All States” as shown in Figure 2(a).

5. Filter by age range:

   a. There will be two seek bars, one for the minimum age to filter with, and the other one for the maximum one, as shown in Figure 2(b).
   b. Both have to be configured to accept values from 0 to 100, inclusive.
   c. When submitting, it must validate that the minimum age is less than the maximum age, and show toast messages if there are any validation issues.

6. Filter by type:

   a. The type options (mobile, home, work) will be presented with radio buttons, as shown in Figure 2(c).
   b. The radio buttons should belong to the same group so that only one of them can be selected at the same time.
   c. When submitting the selection, validations have to be performed, in order to make sure that there is an item selected.



(a) Filter by state (b) Filter by age range (c) Filter by type

Figure 2, Filter Wireframes

Part 3: Sorting Fragment (20 Points)

The interface should be created to match the UI presented in Figure 1(c). The requirements are as follows:

1. This fragment allows the user to select the sorting criteria to be used to sort the list presented in the Users Fragment.
2. This fragment should use a ListView or RecyclerView to present the three rows shown in Fig 1(c). Each row should display the name of the sort attribute.
3. Clicking on the attribute should perform the following:
   a. The selected sort attribute be sent back to the Users Fragment, which should sort the users based on the selected sort attribute. Then refresh the list to display the sorted list of users in ascending order.
   b. The Sorting Fragment should be popped from the back stack and the Users Fragment should be displayed to reflect the selected sort criteria.