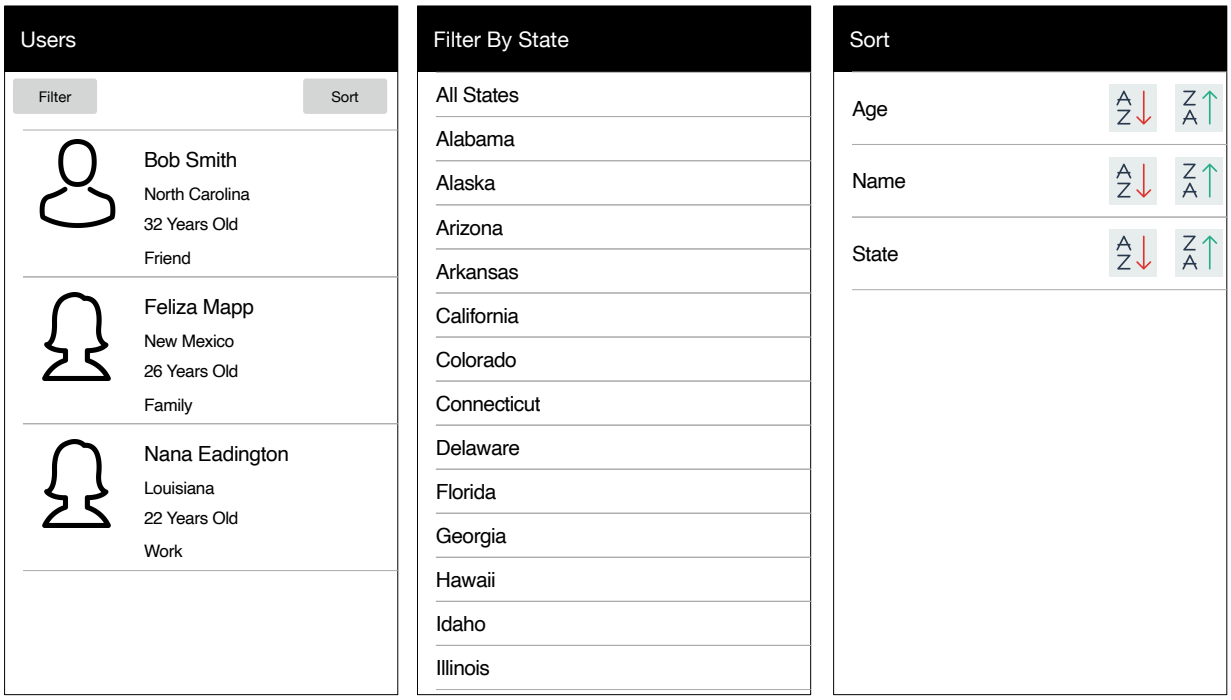
Mobile App Development
Midterm Exam

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 app screens.



(a) Users Fragment (b) Filter By State Fragment (c) Sort Fragment

Figure 1, Application Wireframe

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 Data class that includes a list of User objects. Each user has a name, age, gender, group and state. You should include both Data.java and User.java classes in your project.

Part 1: Users Fragment (35 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 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).
3. Clicking the "Filter" button should display "Filter By State" fragment and place the current fragment on the back stack.
   a. Upon returning from the "Filter By State" fragment, 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.
4. Clicking the “Sort” button 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: Filter By State Fragment (35 Points)

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

1. The list of unique states should be retrieved from the users list in the Data class provided.
2. The list of unique states should be sorted in ascending order and the top row should include “All States” as shown in Figure 1(b).
3. Clicking on a state row item:
   a. The selected state should be sent back to the Users Fragment, which should filter the users to display only the users at the selected state. Then refresh the list to display the filtered list of users. Selecting “All States” row should signal the Users Fragment to display all the users from all states.
   b. The Users Fragment should be popped from the back stack and should be displayed to reflect the selected filter criteria.

Part 3: Sort Fragment (40 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. The sort criteria include sort ascending/descending based on age or name or state.
2. This fragment should use **RecyclerView** to present the three rows shown in Fig 1(c). Each row should display the name of the sort attribute, and two buttons descending and ascending as shown in Fig 1(c).
3. Clicking on the Descending or Ascending buttons on a given row:
   a. The selected sort attribute and criteria should be sent back to the Users Fragment, which should sort the users based on the selected sort attribute and criteria. Then refresh the list to display the sorted list of users.
   b. The Users Fragment should be popped from the back stack and should be displayed to reflect the selected sort criteria.

Item	Grade
Part 1: The list of users is presented as shown in Figure 1(a).	25
Part 1: Clicking the Filter button shows the Filter by State Fragment and puts the Users fragment on the back stack.	5
Part 1 : Clicking the Sort button shows the Sort Fragment and puts the Users fragment on the back stack.	5
Part 2: The list unique states is retrieved from the Data class, and is presented in ascending order, and includes the “All States” at the top, as shown in Figure 1(b).	15
Part 2: Selecting a filter state is communicated with the Users fragment and the user list is updated to reflect the selected filter criteria. The Users fragment is popped from the back stack and shows the updated list.	10
Part 3: The RecyclerView is implemented to show the list of sort criteria as shown in Figure 1(c).	20
Part 3: Selecting a filter state is communicated with the Users fragment and the user list is updated to reflect the selected filter criteria. The Users fragment is popped from the back stack and shows the updated list.	20
Total	**100**