Mobile App Development

In-Class Assessment 2

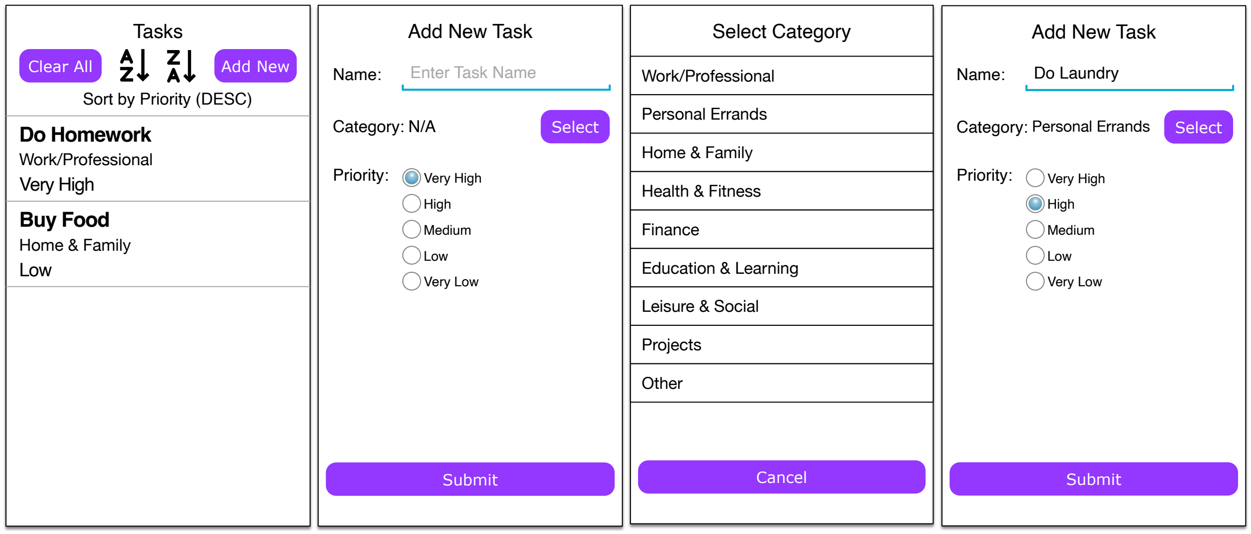
Basic Instructions:

1. This is and In Class Assessment, which will count for 8% of the total course grade.
2. This assessment is an individual effort. Each student is responsible for her/his own assessment and its submission.
3. Once you have picked up the assessment, you may not discuss it in any way with anyone until the assessment period is over.
4. During the assessment, 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 assessment parts, all the parts are required.
6. During the assessment the teaching assistants and Instructors will pass by each student and ask them to demonstrate their application. Your interaction with the teaching assistants and instructors will be taken into consideration when grading your assessment submission.
7. Please download the support files provided with the assessment and use them when implementing your project.
8. 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.
9. 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.
10. **Do not try to use any Social Messenger apps, Emails, Or Cloud File Storage services in this exam.**
11. **Failure to follow the above instructions will result in point deductions.**
12. **Any violation of the rules regarding consultation with others will not be tolerated and will result disciplinary action and failing the course.**

In-Class Assessment 2 (100 Points)

In this assignment you will be building an application that uses ListViews to manage a tasks list. This app is composed of one activity (Main Activity) and multiple fragments. The app requirements are as follows:

1. Please use the provided skeleton app.
2. The ArrayList of Task objects should be stored in the Main Activity to maintain the list of tasks added.
3. All communication between fragments should be performed through the Main Activity.
4. Main Activity should load the “Tasks” Fragment as the initial fragment.



(a) Tasks Screen **(b) Add New Task** **(c) Select Category** **(d)Add New Task**

Figure 1, Application User Interface

Part 1, Tasks Fragment (45 Points):

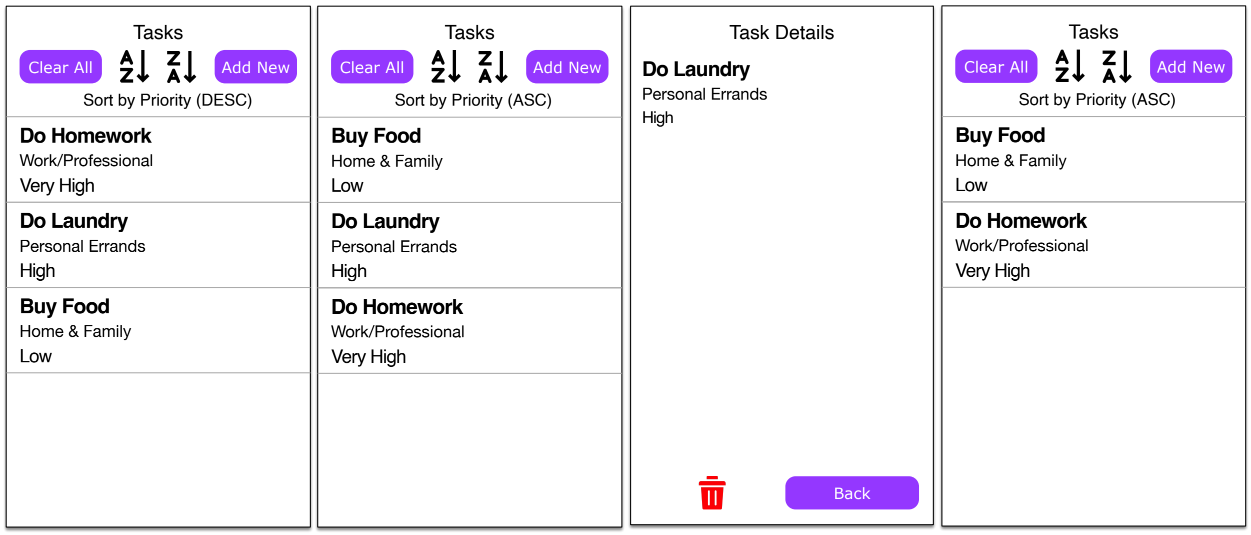
This fragment displays the ListView of tasks as shown in Figure 1(a). Requirements are:

1. This fragment should receive the ArrayList of Task objects from the Main Activity, and should display the list of tasks as shown in Figure 1(a).
   a. You should create and use an interface to request the ArrayList of Task objects from the Main Activity.
   b. Use a ListView, extend the ArrayAdapter which should be implemented inside the this fragment.
   c. Each row item should display the name, category and priority.
2. Clicking the “Clear All” button should communicate with the Main Activity to:
   a. Remove all the tasks from the ArrayList of Task objects.
   b. Reload the ListView to display no tasks as the tasks ArrayList is empty.
3. Clicking the “Add New” button should communicate with the Main Activity to:
   a. Replace the current fragment with the Add Task fragment.
   b. Push the current fragment on the back stack.
   c. Upon returning from the Add Task fragment this fragment should request the updated ArrayList of tasks and refresh the ListView to show the list of tasks which contains the newly added task as shown in Figure 2(a).

4. Clicking the “Z→A” button should:
   a. Sort the list of Tasks in descending order by task priority, where Very High is the largest value and Very Low is the lowest value.
   b. Refresh the ListView to display the sorted array of Tasks.
   c. Update the TextView under the buttons to indicate that “Sort by Priority (DESC)” as shown in Figure 2(a).
5. Clicking the “A→Z” button should:
   a. Sort the list of Tasks in ascending order by task priority, where Very High is the largest value and Very Low is the lowest value.
   b. Refresh the ListView to display the sorted array of Tasks.
   c. Update the TextView under the buttons to indicate that “Sort by Priority (ASC)” as shown in Figure 2(b).
6. Clicking on the row item should communicate with the Main Activity to:
   a. Replace the current fragment with the Task Detail fragment, send the selected Task object to the Task Detail fragment.
   b. Push the current fragment on the back stack.

Part 2, Add New Task Fragment (20 Points):
This fragment is used to add a new task. The requirements are listed below:
1. Clicking on the “Select” should communicate with the Main Activity to:
   a. Replace the current fragment with the Select Category fragment.
   b. Push the current fragment on the back stack.
   c. Upon returning from the Select Category fragment the selected category value should be received through the Main Activity, sent to the Add Task fragment and should be displayed see Fig 1(d).
2. Clicking on the Submit button should:
   a. If any of the entries is not entered or selected then show a toast message indicating the missing input.
   b. If all the required data is entered then create a Task object which contains all the entered and selected values. Send the created Task object to the Main Activity which should:
      i. Add the new Task object to the Tasks ArrayList hosted in the Main Activity.
      ii. Pop the back stack which should go back to the Tasks fragment.

Part 3, Select Category Fragment (20 Points):
This fragment allows the user to select a category value as shown in Fig 1(c). The requirements are listed below:
1. List the category selections in a ListView as shown in Figure 1(c). The category selections can be retrieved using the provide Data class.
2. You should use the simple ArrayAdapter to display the category string values.
3. Clicking a list item should communicate with the Main Activity to:
   a. Find the “Add Task” fragment by tag and send it the selected category value.
   b. Pop the back stack which should display the Add Task fragment and display the selected category value as shown in Figure 1(d).
4. Clicking “Cancel” should simply communicate with the Main Activity to:
   a. Pop the back stack which should go back to the Add Task fragment.



Sort by Priority (DESC)

Do Homework
Work/Professional
Very High

Do Laundry
Personal Errands
High

Buy Food
Home & Family
Low

Sort by Priority (ASC)

Buy Food
Home & Family
Low

Do Laundry
Personal Errands
High

Do Homework
Work/Professional
Very High

Do Laundry
Personal Errands
High

Sort by Priority (ASC)

Buy Food
Home & Family
Low

Do Homework
Work/Professional
Very High

(a) Tasks Screen **(b) Tasks Screen (ASC)** **(c) Task Details** **(d) Task Deleted**

Figure 2, Application User Interface

Part 4, Task Details Fragment (15 Points):

This fragment displays the selected task details as shown in Figure 2(c). The requirements are listed below:

1. This fragment should receive a Task object from the Tasks fragment through the Main Activity.
2. Display the name, category and priority as shown in Figure 2(c).
3. Clicking “Delete” should simply communicate with the Main Activity to:
   a. Send the current Task object to the Main Activity, and delete that Task object from tasks ArrayList that is stored in the Main Activity.
   b. Pop the back stack which should go back to the Tasks fragment which should show the new list of tasks not containing the deleted task, see Figure 2(d).
4. Clicking “Back” should simply communicate with the Main Activity to:
   a. Pop the back stack which should go back to the Tasks fragment.

Section:	
Student Name:	
Student ID:	

Part #	Features	Total	Grade
Part 3	Select Category: Displays the list of categories. Sends the selected category value through an interface to the Main Activity. The selected value is sent to the Add Task fragment and is displayed in the Add Task fragment.	20	
Part 2	Add Task: Validation and sends the new Task object through an interface to the Main Activity. The new task is added to the ArrayList of Tasks in the Main Activity.	20	
Part 1	Tasks: Retrieves the ArrayList of Tasks from the Main Activity though an interface. Displays the list of tasks (Extended adapter).	15	
Part 1	Tasks: Clear all - Through the adapter, deletes all the Tasks in the ArrayList of Tasks and notifies the adapter to refresh the list.	10	
Part 1	Tasks: Sorts ASC, sorts the ArrayList of Tasks in ascending order by priority and notifies the adapter to refresh the list.	10	
Part 1	Tasks: Sorts DESC, sorts the ArrayList of Tasks in descending order by priority and notifies the adapter to refresh the list.	10	
Part 4	Task Details: Receives and displays the Task object information. The delete feature, deletes the Task object from the ArrayList of Tasks through the interface, and the Task Screen reloads to show the updated list.	15	
	Total	**100**	

Table 1: Grading Key