logit development log

## 27 dec 2022

### implementing nav drawer

#### step 1: convert activity main log to a fragment

activityMainLog has to be converted to fragment for nav drawer to work.

A parent activity “ActivityParent” will be created. This parent activity will instantiate the current ViewModel, too. This way, saved assignment data will only have to be pulled once and shared to both FragmentLog and FragmentCalendar.

|  |  |
| --- | --- |
| **Issues** | **Solutions** |
|  |  |

#### step 2: implement nav drawer

I’ll follow the steps I’ve outlined in my Notes document and take reference from the Marcellina Pizzas app.

|  |  |
| --- | --- |
| **Issues** | **Solutions** |
|  |  |

## 30 dec 2022

### issue: calling parent fragment on a nav host fragment returns null

**Description:** I wanted to call ParentFragment to call the methods showFab and hideFab when user scrolls, but calling ParentFragment returns a navHostFragment, and calling parentFragment of that returns null.

**Solution:** Instead of calling parentFragment, I used setFragmentResultListener and setFragmentResult instead as follows:

**FragmentTodo.kt**

if (dy > 0) { // scrolling down and fab is shown  
 *setFragmentResult*("hideFab", *bundleOf*())  
} else { // scrolling up and fab is not shown  
 *setFragmentResult*("showFab", *bundleOf*())  
}

**FragmentLog.kt** (FragmentTodo’s parent fragment)

// show/hide fab on user scroll  
*setFragmentResultListener*("showFab") **{** \_, \_ **->** showFabAddTask()  
**}***setFragmentResultListener*("hideFab") **{** \_, \_ **->** hideFabAddTask()  
**}**

## 31 dec 2022

### moving settings from activity to fragment

#### issue: context as settingsfragment – cast never succeeds

|  |  |
| --- | --- |
| **Issue description** | I want to call a method *updateSettings* in SettingsFragment within the RecyclerView Adapter class, so I tried doing *(context as SettingsFragment).updateSettings* but this throws an error: “cast never succeeds”. |
| **Solution** | Initially, I thought I could implement an interface, but then I realized it would be easier if I just moved *updateSettings* to ParentActivity and did *(context as ParentActivity).updateSettings* in the Adapter. This worked. |

## 1 jan 2023

### implementing search function in log

|  |  |  |
| --- | --- | --- |
| **The idea** | In FragmentLog, users can search by a) Subject, or b) Task. User’s input (*p0*) will be looked up against each Task item’s Subject and Task. Task items which have valid Subjects and/or Tasks will be shown in the RecyclerView. | |
| **Implementation** | Refer to Notes document on implementing filtering / search functions. | |
| **Issues** | | **Solutions** |
| Some back-end *ArrayLists* get modified when user searches and doesn’t go back to the original ones when user exits search. If user swipes after exiting search, there will be a duplicate of another taskItem being shown in the RecyclerView.  If I swipe to Done and back to Todo though, the RecyclerView fixes itself and shows the correct items – possibly to do with onResume? It resets back-end lists there? | | I could call *getLists* upon user swipe and that fixes the duplicate item issue but that would cause the RecyclerView animations to be jerky since *getLists* calls *createRV*. I think this solves the duplicate issue because it recreates the RV, not because it updates the lists – I tried updating lists but it still had duplicate items. Not sure how to fix the duplicate items without recreating the RV. |
| FragmentLog’s todoList and doneList are not updated when user swipes after search. | | Called *fragmentResultListener* before calling *filter* to update lists in FragmentLog. |
| If user swipes while still in search (ie the RecyclerView is still showing the modified lists), mapOfIndex in FragmentTodo is not updated to reflect the new position-index pairs. | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |

## 2 jan 2023

### creating calendar fragment

|  |  |
| --- | --- |
| **The idea** | Fragment with a CalendarView showing all the user’s tasks, similar to the Calendar app with all your upcoming events. |
| **Implementation** |  |

## 12 jan 2023

### calendar – mark tasks as done

**Plan:** Create an interface in RecyclerViewAdapter which is called by PagerAdapter, which also has an interface which is in turn called by CalendarFragment, which uses a ‘for’ loop to find the task with the corresponding ID as the completedTask (‘for’ loop is needed here since RecyclerViewAdapter doesn’t have the full todoList). CalendarFragment then calls *markAsDone* in ViewModel, just like what would be done in TodoFragment (in the Log part of the app).

**Implementation:** Created interface in RecyclerViewAdapter which is called by PagerAdapter. PagerAdapter finds the actualPosition of the completedTask (its index in todoList) and adds a (Task, Int) object to a list ‘completedTaskIndexes’. When user unchecks the task (ie mark as undone), its corresponding TaskIndex is removed from this list. When the ViewPager is closed (AlertDialog dismissed), PagerAdapter marks all the tasks in completedTaskIndexes as done by calling *(context as ParentActivity).viewModel.markAsDone()*.

## 17 jan 2023

### calendar: “checked” mark as done gets reset when user swipes through pages

**Issue:** When user clicks the Check icon to mark a Task as done, then swipes to other pages in the ViewPager, the Check icon gets marked as undone (on the front end) when user swipes back. However, it is still marked as done on the back end. This disparity causes crashes.

**Solution:** Initially, the code to check for whether Checked or Unchecked was placed in the ViewHolder class, but this always caused the mapChecked[*adapterPosition*] to return null, possibly since mapChecked is not yet initialized when the ViewHolder class is called. To fix this, the code was moved to the **onBindViewHolder** method.

// set check color  
checked = if (mapChecked[holder.*adapterPosition*] != null) {  
 mapChecked[holder.*adapterPosition*]  
} else {  
 false  
}  
if (checked == true) {  
 holder.checkIcon.*imageTintList* = ColorStateList.valueOf(getColor(context, androidx.appcompat.R.attr.*colorAccent*))  
} else {  
 holder.checkIcon.*imageTintList* = ColorStateList.valueOf(getColor(context, R.attr.*calendarDialogCheckColor*))  
}

### revamped settings page

|  |  |  |  |
| --- | --- | --- | --- |
| **Before**  Settings fragment had a RecyclerView populated with SettingsItems and a horizontal line between each item. This looked very unprofessional. | **After**   |  |  | | --- | --- | | I replaced the dynamic RecyclerView with a static xml layout – since the number of SettingsItems will always be fixed (unless I decide to change it in development), a static layout is suitable here. |  | |