Demo One

Cal-Task

Brad Stiff

Table Of Contents

Scope

Project Description

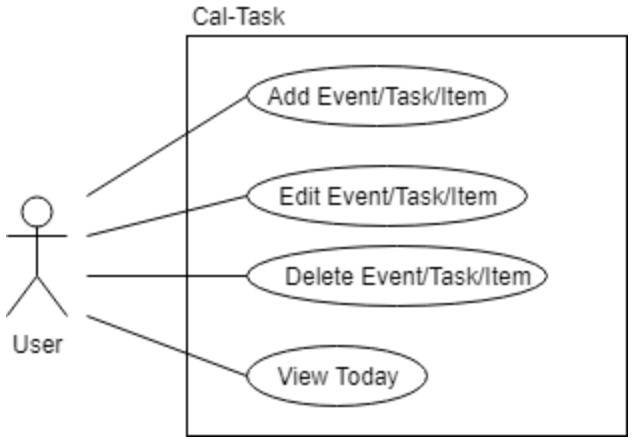
Goal

This project will consist of creating an android application that allows linked users to share tasks, events, and grocery lists. The user will be able to list these objects as private or public among the linked users. A wall screen, essentially a wall calendar, will act as a home entry point for the system, or display the events/lists of the household.

Deliverables

- Android Application
- Monthly Demo Reports
- Wall Screen Running Android

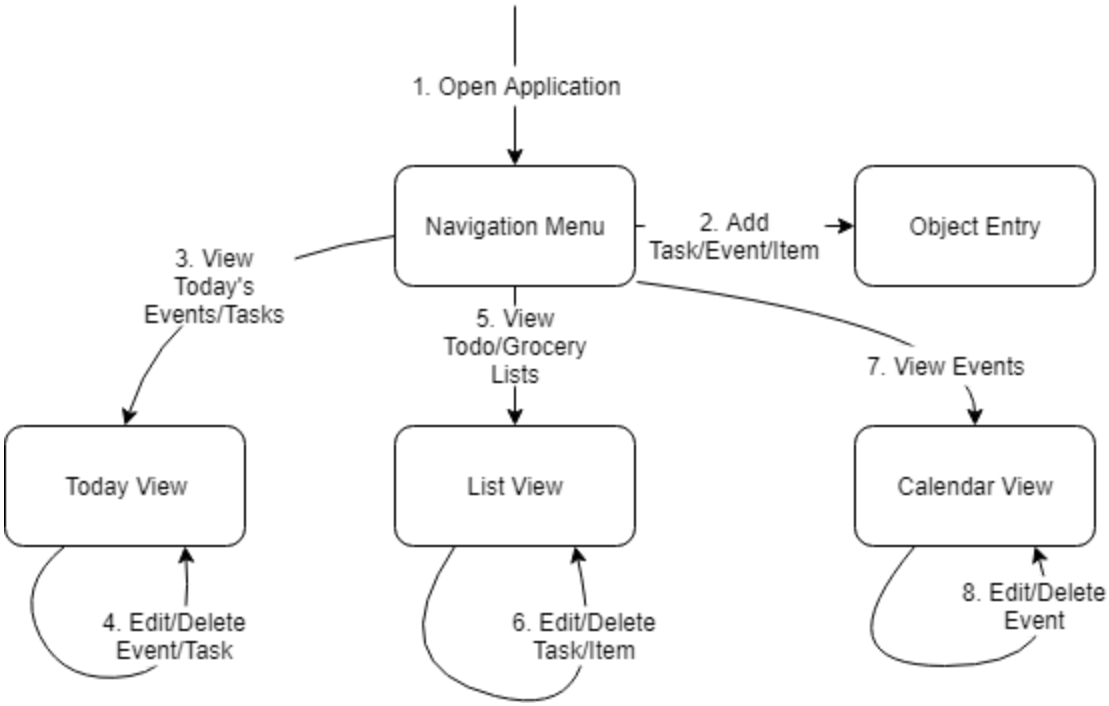
Use Case Diagram



Use Case	Description
Add Task/Event/Item	**Task:** Adds task to database. The user has the option of giving their task entry a due date. If the task has an associated date, it will also appear on the calendar. **Event:** Adds event to the database. The user can add a date, time, and location to the event.

	Item: Adds an item to the database. The item will appear on the Lists tab in the Grocery List Section. **All:** Objects that are listed as public will show up for all linked users. Public objects can be edited by linked users, but can not be changed to private by a linked user.
Edit Task/Event/Item	**Task:** Allows the user to edit the task entry or date. Tasks can be entered without a due date, if a date is added, it will also appear on the calendar. If a date is removed, it will also be removed from the calendar, but remain on the Todo list. **Event:** Allows the user to edit an event, date, time, or location. **Item:** Allows the user to edit a Grocery List item. **All:** If an object is changed from public to private, all linked users outside of the object creator will lose access to said object. Only the object creator can change object back to private.
Delete Task/Event/Item	**Task:** Removes the task from the database. Completing the task essentially does the same action as deleting. **Event:** Removes the event from the database. **Item:** Removes the item from the database. Items that have been picked up essentially does the same action as deleting the item. **All:** Public objects that are deleted are also removed from linked users. Both users can delete objects regardless of object owner.
View Today	Polls the events and tasks with the current date.

Communication Diagram



Use Case	**Communication Path/Description**
View Today	1. Open Application 3. View Today's Events/Tasks **Description:** Upon opening the application, the Today tab/view is already selected to show today's events and tasks.
Add Task/Event/Item	1. Open Application 2. Add Task/Event/Item **Description:** On the toolbar above the tabs, there will be a button that displays a pop-up window will allow the user to enter any of the objects into the database. Since it is placed here, all views will have access to creating new objects.
Edit Task/Event/Item	**Task:** 1. Open Application 3. View Today's Events/Tasks -or- 5. View Todo/Grocery Lists 4. Edit/Delete Event/Task -or- 6. Edit/Delete Task/Item

	Description: When the user is viewing either the Today View or the Lists View, they can click on the task item (row) and choose to edit or delete the object. **Event:** 1. Open Application 3. View Today's Events/Tasks -or- 7. View Events 4. Edit/Delete Event/Task -or- 8. Edit/Delete Event **Description:** When the user is viewing either the Today View or the Calendar View, they can click on the event item (row), or date on the CalendarView object to edit or delete the event. **Item:** 1. Open Application 5. View Todo/Grocery Lists 6. Edit/Delete Task/Item **Description:** When the user is viewing the Lists View, they can click on the grocery item (row) to edit or delete the item.
Delete Task/Event/Item	**All:** Same paths as edit

Implementation

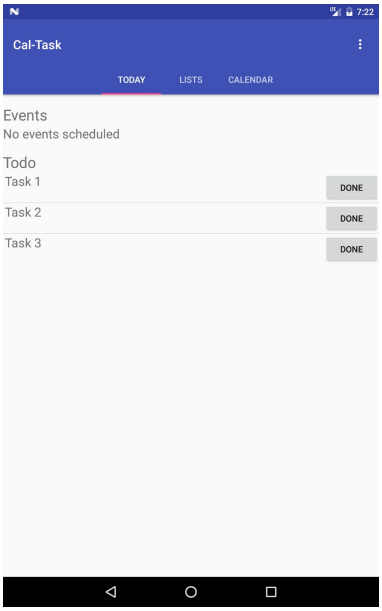
Activity Navigation

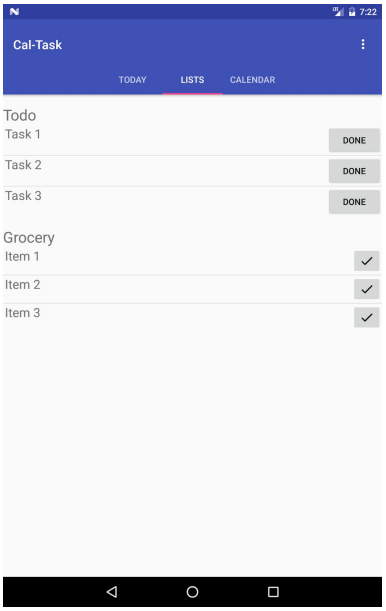
Problem

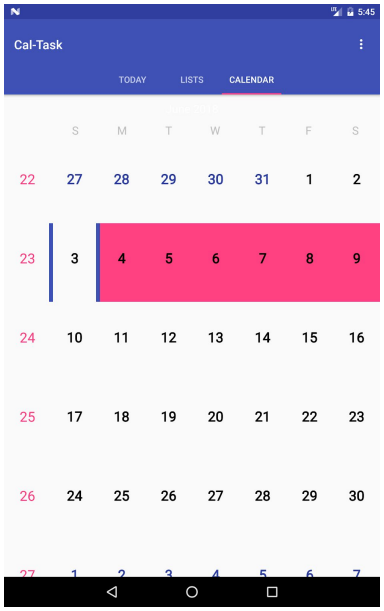
The application being developed has many features that need to be displayed on the screen of the device. The activities need to be presented in a user friendly way, and the navigation between these activities needs to be fluid and operate quickly. Activities of similar relevance should be contained inside the same view.

Views	Use Case
Today	• View Today

	• Edit Task/Event • Delete Task/Event
Lists	• Edit Task/Item • Delete Task/Item
Calendar	• Add Event • Edit Event • Delete Event
Main (Toolbar)	• Add Task/Event/Item





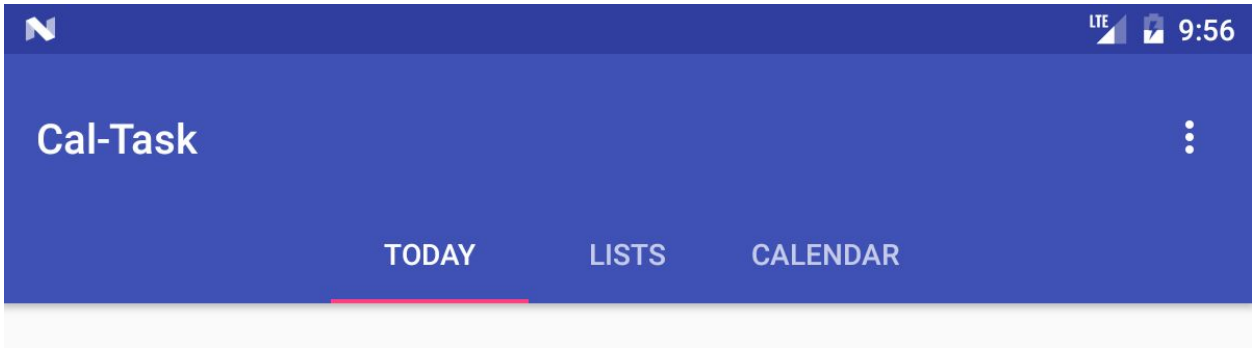


Solutions

Tab Navigation

Description

Navigation between the activities will be handled with buttons held inside a toolbar at the top of the screen. Swiping left or right will also change the displayed layout fragment and activity.



Pros

Since the navigation is held inside a toolbar, the user can easily travel between the different activities views. Each class is created as a fragment, allowing quick navigation between the different activities. The user can also swipe left or right to change between the different activities as well. This method also allows for room to grow, and additional activities to be added in the future.

- Modular design
- Easy implementation
- Expandable

Cons

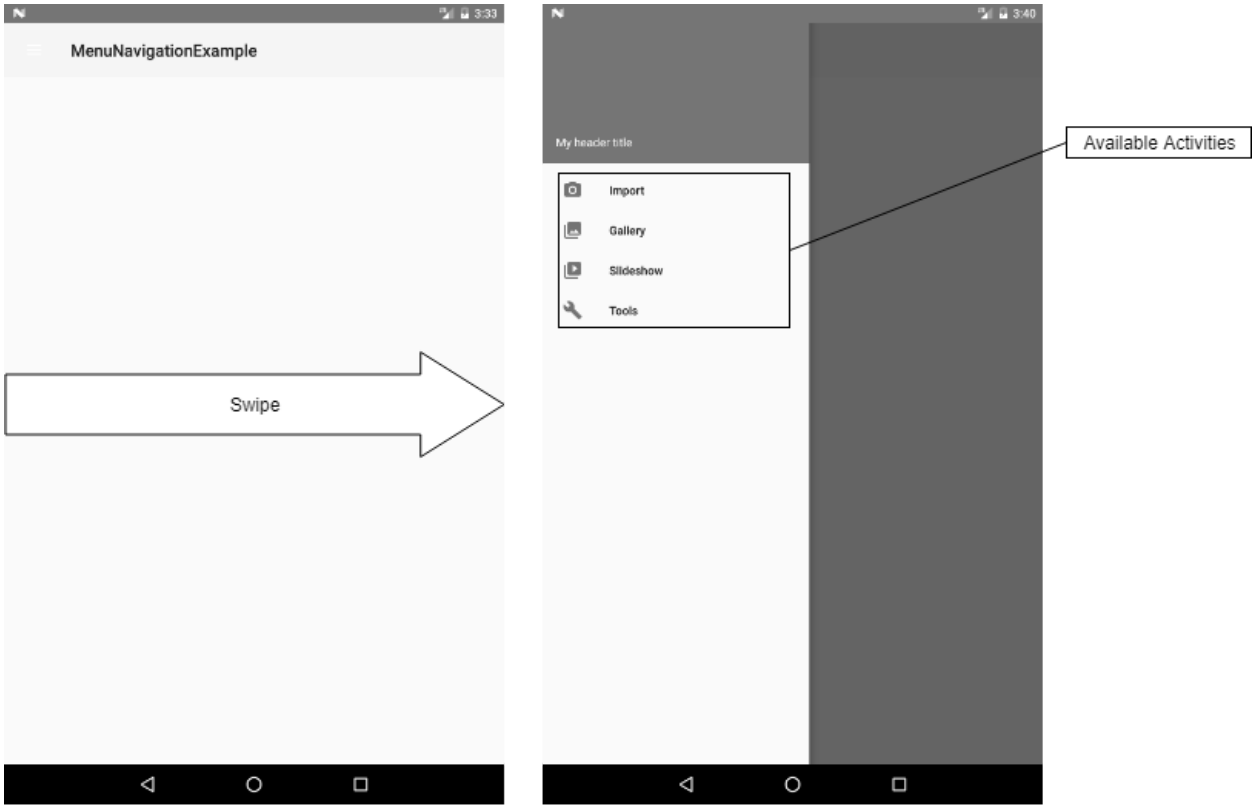
The top navigation takes up the upper section of the screen from all layouts, allowing for less space to be utilized by the activities.

- Space used

Navigation Drawer

Description

This solution creates a menu that slides out from the side of the screen. The navigation is therefore hidden off of the screen, allowing for the whole screen to be used for the activity. Selecting an item in the drawer menu, changes to the desired activity. The drawer menu will remain across activities, similar to the tabbed navigation. The navigation back to the main activity is the same as the button navigation solution.



Pros

Since the drawer is hidden off of the screen, the entire screen below the application title can be used for the activity. The code can implemented in a modular fashion with relative ease.

- Hides off screen
- Modular implementation

Cons

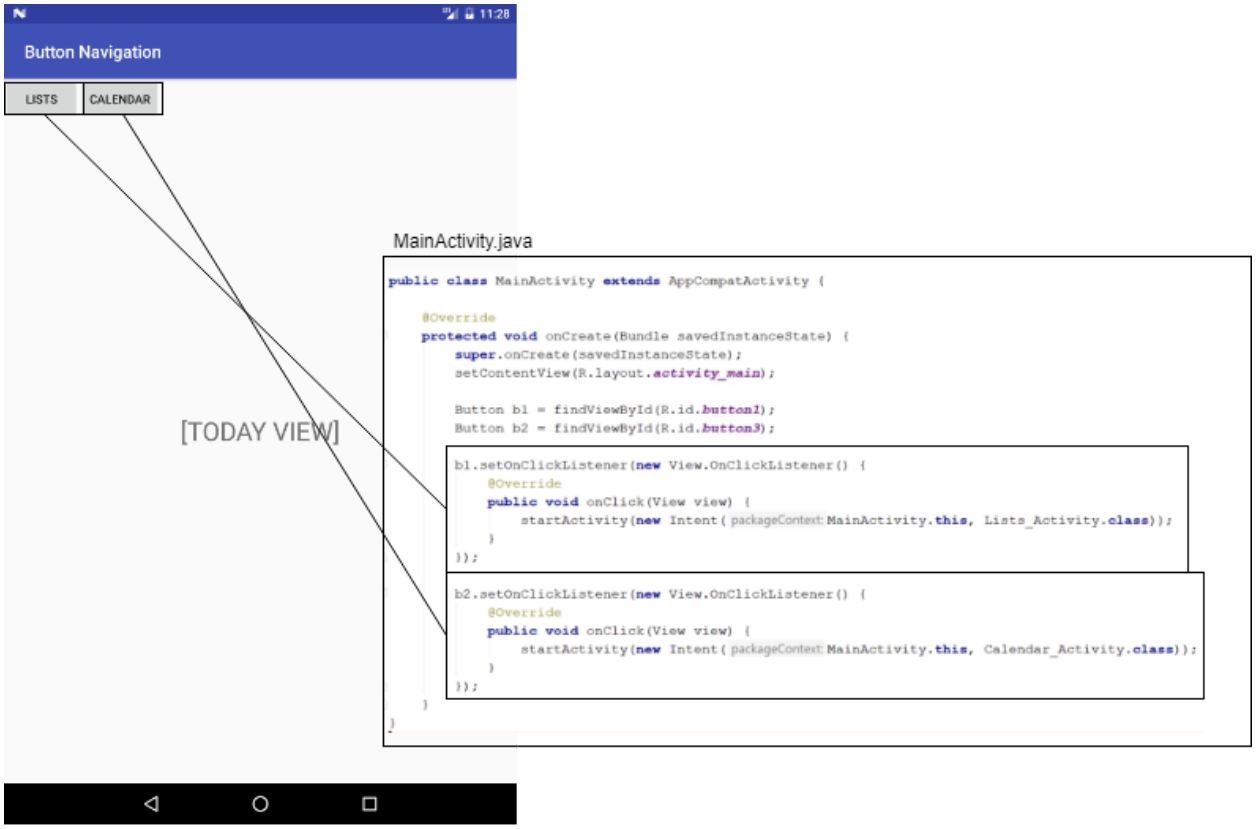
This navigation option seems bulky for its design with the desired final product. The user would have to pull out the navigation drawer and then select an option before navigating to the new activity. Once there, the navigation back to the main activity is basically the same as with button navigation. Together, these two things slow down the navigation between activities.

- Bulky (More steps required in navigation)
- Slow

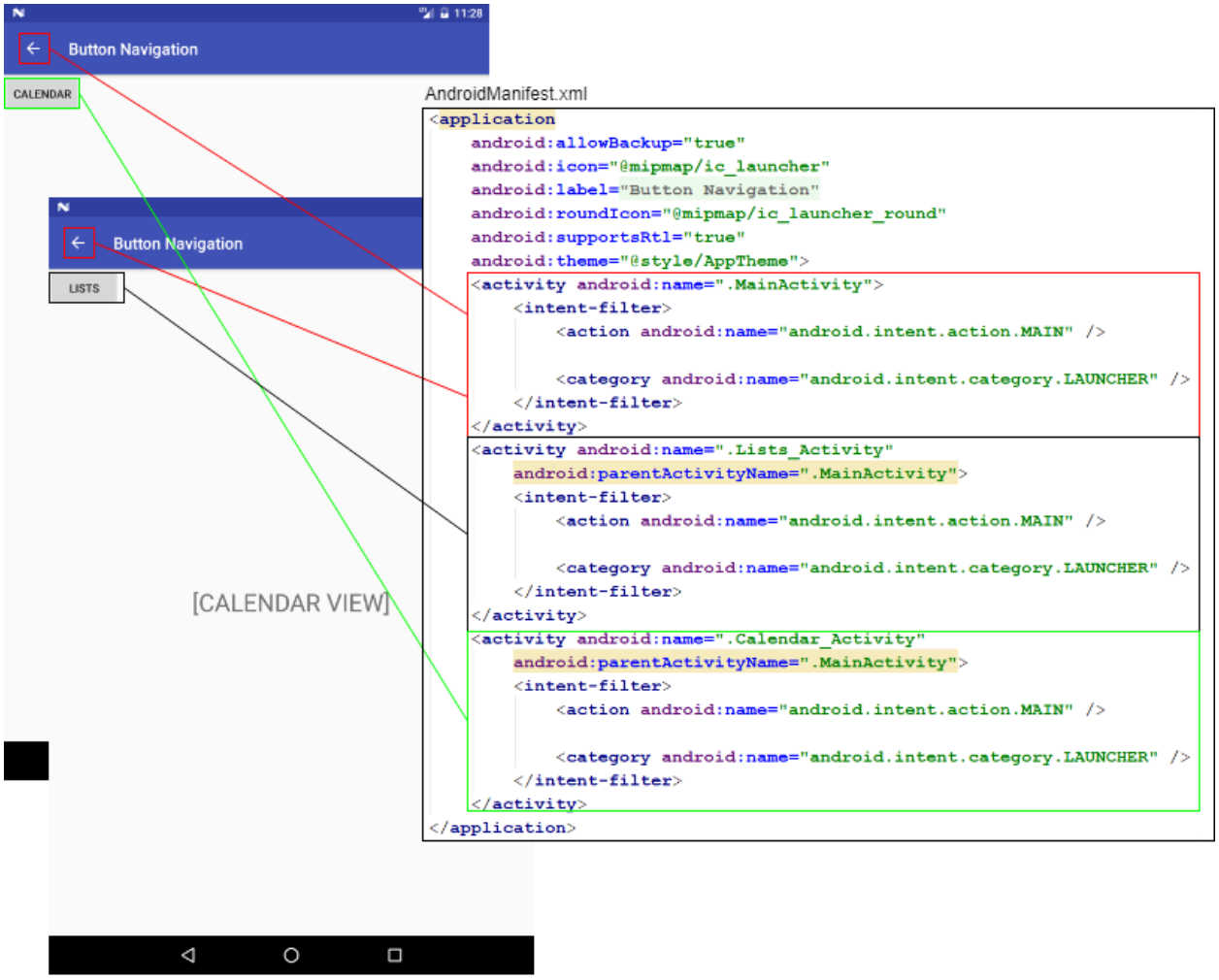
Button Navigation

Description

The navigation between activities will be done by buttons placed inside the view of the activity. Each view will need its own set of buttons, the button IDs can be the same across views. To navigate back to the main activity, the back arrow in the toolbar will return the user.



The above implementation shows how the activities are changed upon button press. When the button is clicked, it starts a new activity with a new view. The new class being passed, with created its new view with its own onCreate method.



Each java class associated with an activity has to be listed inside the manifest file for the application. By listing the MainActivity as the parent activity, whenever the other activities are running, a back arrow appears in the application title bar to allow the user to navigate back to the main activity.

Pros

This solution offers a way to separate the different activities pretty easily. This solution also allows for the navigation to be placed anywhere since it is held solely to a button object.

- Semi-modular
- Easy customization

Cons

In order to add a new java class, you have to create a layout, a java class, custom button navigation, and then add the class to the manifest file. In tabbed navigation, the new tabs just require the java class, layout, and added to the tab list. The navigation between the layouts is kinda slow and bulky due to the fact that each activity starts and pauses each other upon activation.

- Semi-modular

- Need to list classes in multiple files
- Slow

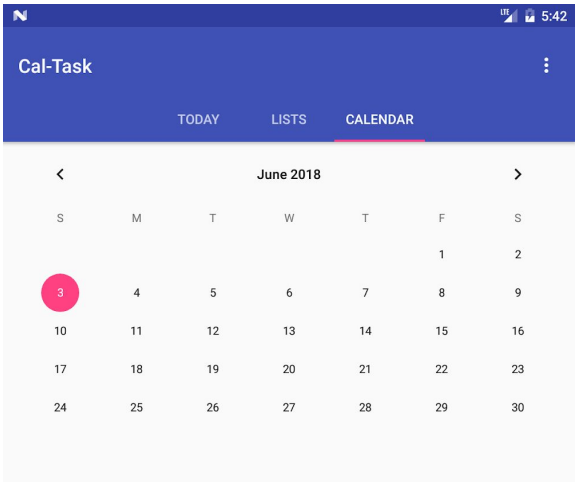
Design Decision

The tabbed navigation option was chosen due to its perks aligning with the desired final product. Tabbed navigation provides quick navigation between activities by either a swipe or clicking on the desired activity. The fragment implementation keeps all the activities running almost simultaneously, allowing for the navigation to feel smooth and fast. The ease to add new features was also an added bonus if the project required further additions.

CalendarView Depreciation

Problem

The built-in CalendarView object is a necessary asset, but it needs to be customized away from the default layout. As it stands, once the user gets on the Calendar tab, swiping left or right only changes the months. The ability to swipe between tabs could remain if the functionality was changed to a vertical implementation.

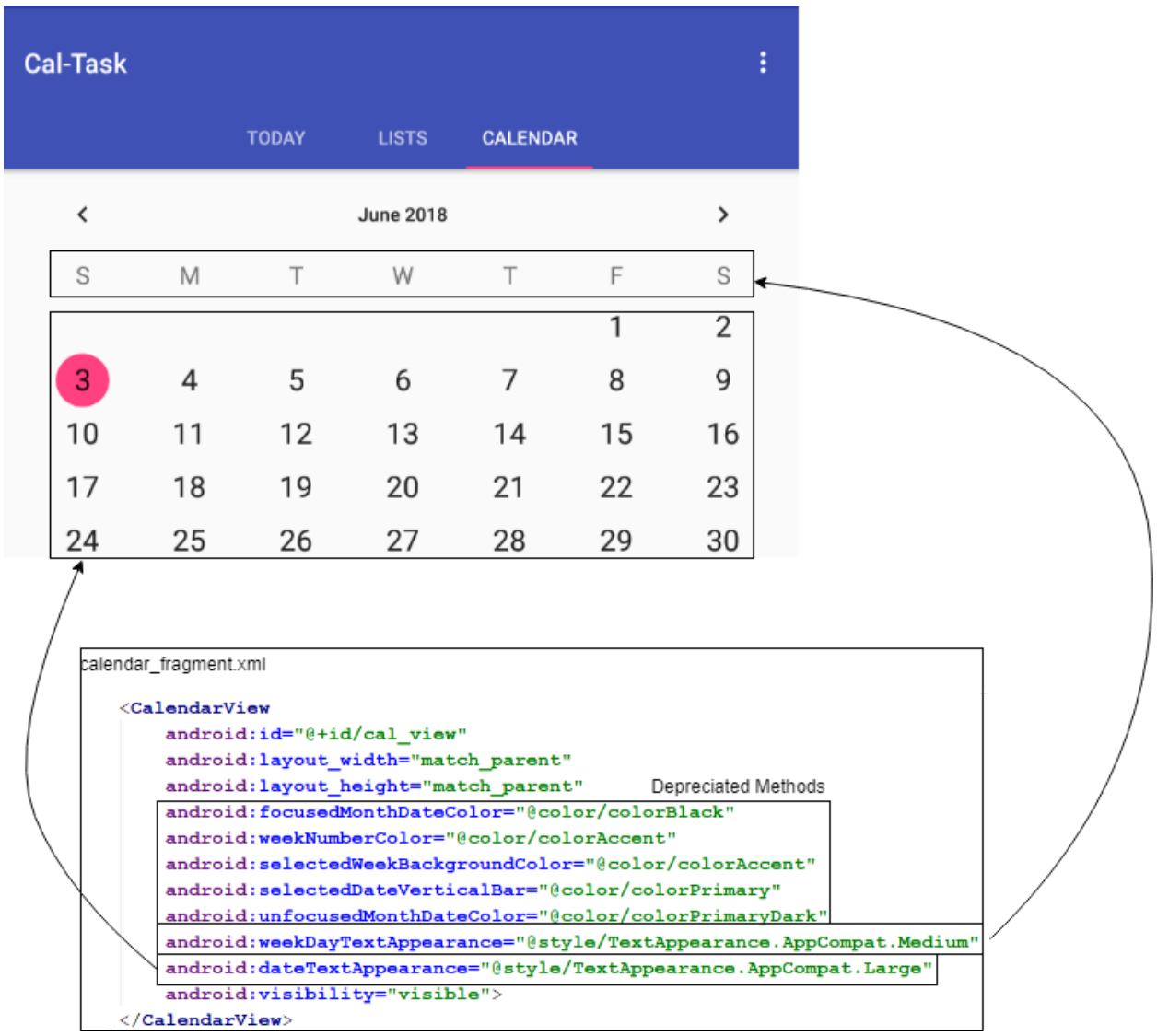


Many of the methods associated with the CalendarView object have depreciated throughout the many Android API Levels. The most current API level is 27. This creates a problem with using the built-in CalendarView object.

Method	Depreciated	Description
getFocusedMonthDateColor()	API level 23	Returns the color for the dates in the focused month.
setFocusedMonthDateColor(int i)	API level 23	Sets the color of the dates in the focused month. *int: The desired color*

getSelectedDateVerticalBar()	API level 23	Returns the drawable object used for the vertical bars of the selected date.
setSelectedDateVerticalBar(int i)	API level 23	Sets the drawable object to be used for the selected date. The object appears on either side of selected date. *int: The desired drawable ID*
setSelectedDateVerticalBar(drawable d)	API level 23	Sets the drawable object to be used for the selected date. The object appears on either side of selected date. *drawable: The desired drawable*
getSelectedWeekBackgroundColor()	API level 23	Returns the background color of the selected week.
setSelectedWeekBackgroundColor(int i)	API level 23	Sets the background color of the selected week to the desired color. *int: The desired color*
getShowWeekNumber()	API level 24	Returns a boolean of whether or not to show the week numbers.
setShowWeekNumber(boolean b)	API level 24	Sets whether the week numbers show or not in the CalendarView object. *boolean: true or false*
getShownWeekCount()	API level 23	Returns the number of weeks that are to be shown.
setShownWeekCount(int i)	API level 23	Sets the number of weeks to be shown in the CalendarView object. *Int: The desired number of weeks*
getUnfocusedMonthDateColor()	API level 23	Returns the color of the dates of the unfocused month.
setUnfocusedMonthDateColor(int i)	API level 23	Sets the color for the dates in the unfocused month. *int: The desired color*
getWeekNumberColor()	API level 23	Returns the color used for the week numbers.
setWeekNumberColor(int i)	API level 23	Sets the color for the week number. *int: The desired color*

getWeekSeparatorLineColor()	API level 23	Returns the color used for the line separator between weeks.
setWeekSeparatorLineColor(int i)	API level 23	Sets the color for the line separator between weeks in the CalendarView object. *int: The desired color*

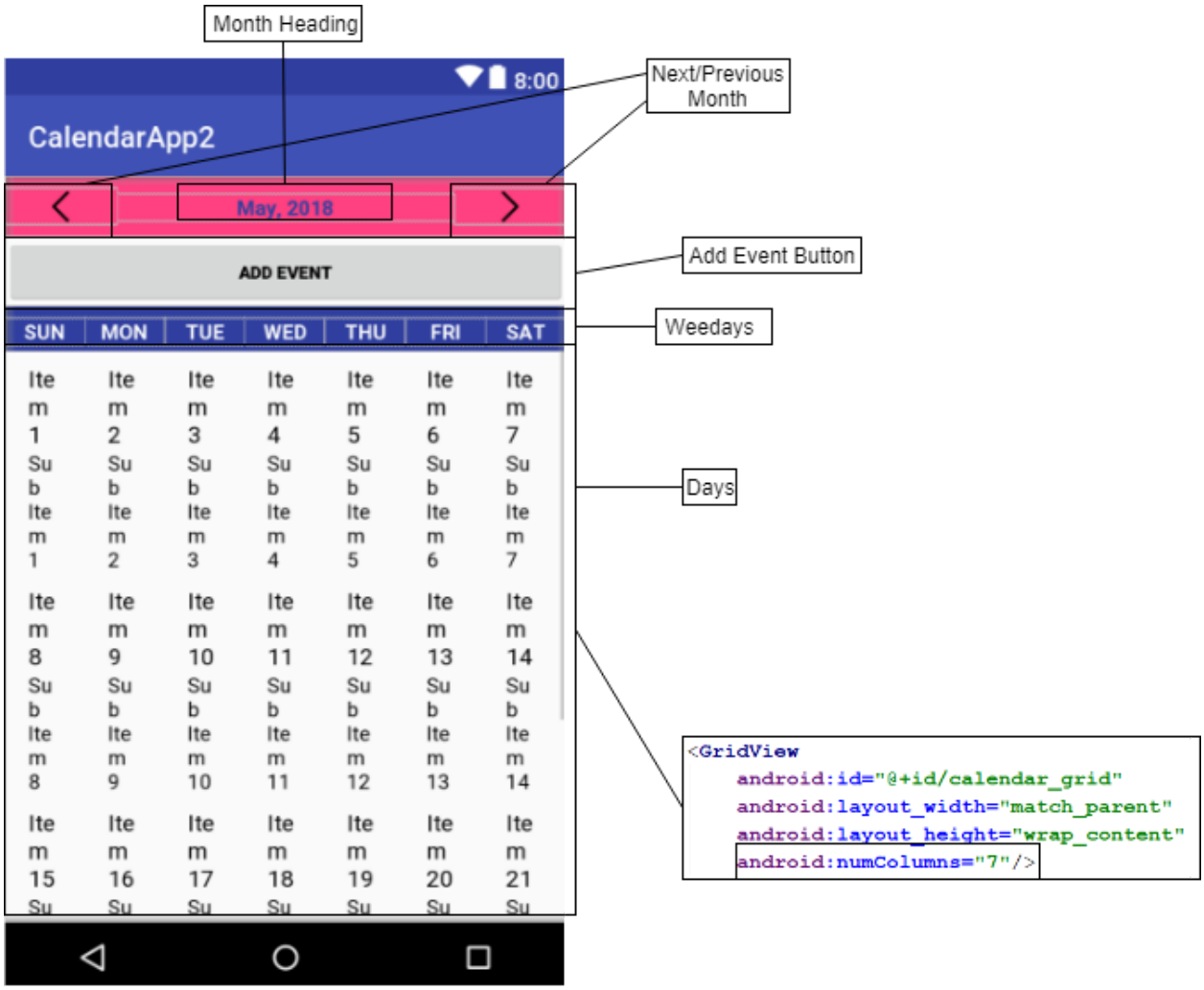


Solutions

Build From Scratch

Description

Various objects could be used to create a calendar layout. This option would give complete customization. Using a GridView object and limiting the number of columns, this object could be used as way to display the days of the month. Creating a Calendar object in java, and updating the view using java would make this a viable option.



Pros

This option would allow for complete customization in design. The GridView object will scroll vertically and ActionListerners could be used for handling the changing of months.

- Complete Customization
- Vertical Scroll

Cons

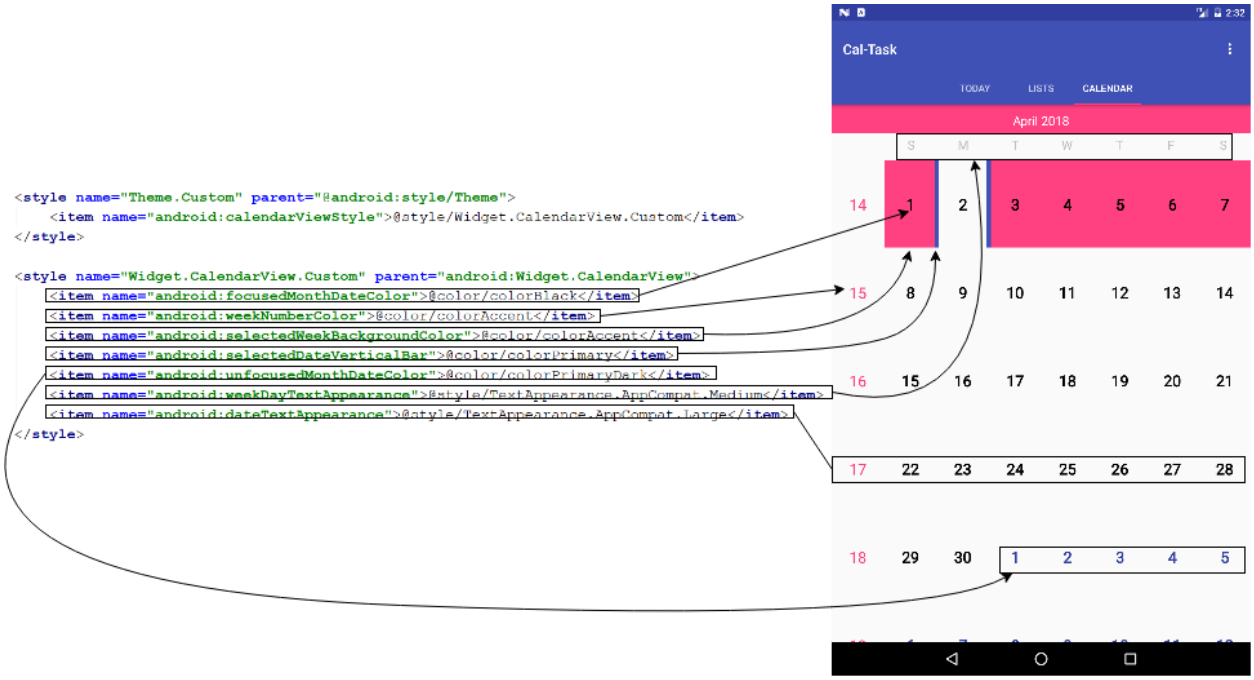
Since this option is build from scratch, every built-in method that the CalendarView has has to be re-implemented as well.

- Reinventing the wheel

CalendarView Custom Theme

Description

A custom theme



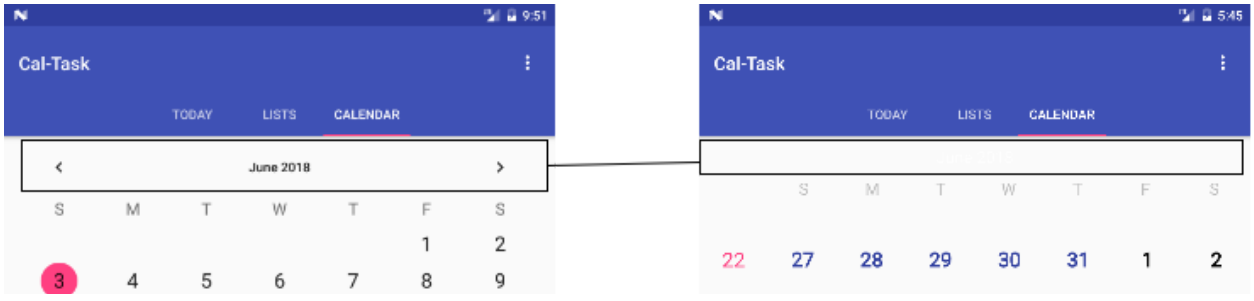
Pros

This method allows me to get around some of the depreciated methods.

- Use of built-in methods
- Vertical Scroll

Cons

This option does allow me to somewhat use most of the depreciated methods, but for some reason, I lose the month heading when creating a custom theme for the CalendarView.



- Lose month heading

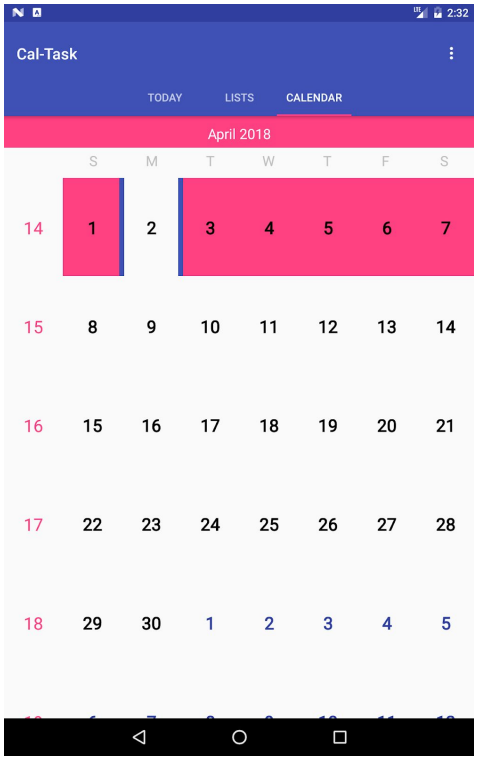
Google Calendar API

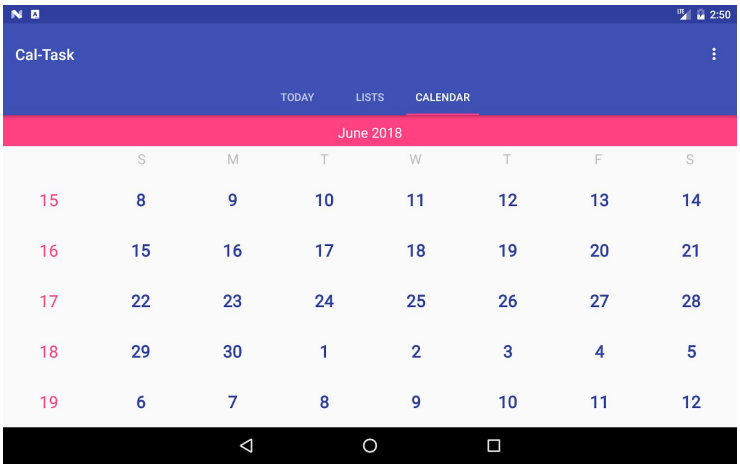
This API was considered, but ultimately ruled out. The choice was made to avoid this because in order to use the calendar aspect at all, the user would be required to have internet

connection. Internet usage is an aspect wanted for the project, but not a requirement. Events created would have to be done twice, once for the API, and once for the system database.

Design Decision

The custom theme option provided almost everything desired, except for the losing the month header. After trying to implement another with a TextView, the text seemed to be white, appearing invisible over the white app background. So instead of using text, the background of the TextView was just colored to the app accent color. Since the text size doesn't change when the device is rotated, making the TextSize attribute of the TextView equal 30sp creates the desired effect.





Tutorial

Overview

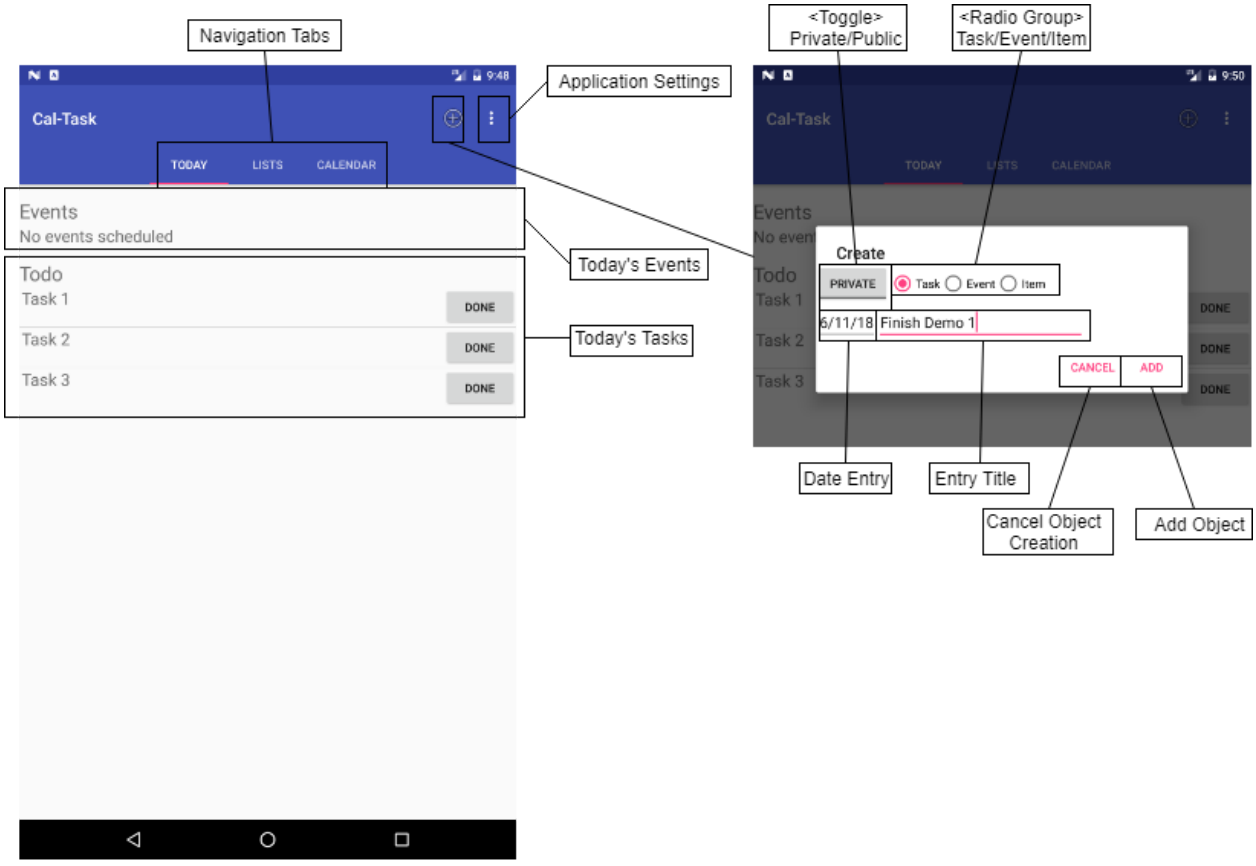
For demo one, the main focus was to get a basic understanding of android development and to create the layout for the application. The application currently allows navigation between the views, and presents an object creation form upon button click.

Goals For Next Demo

For demo two, the main focus will be developing the back-end of the application. This includes:

- Database creation/update/deletion
- Database Hosting
- Action Listeners for database manipulation objects
- User Accounts
- Public and private object management

Main and Today View



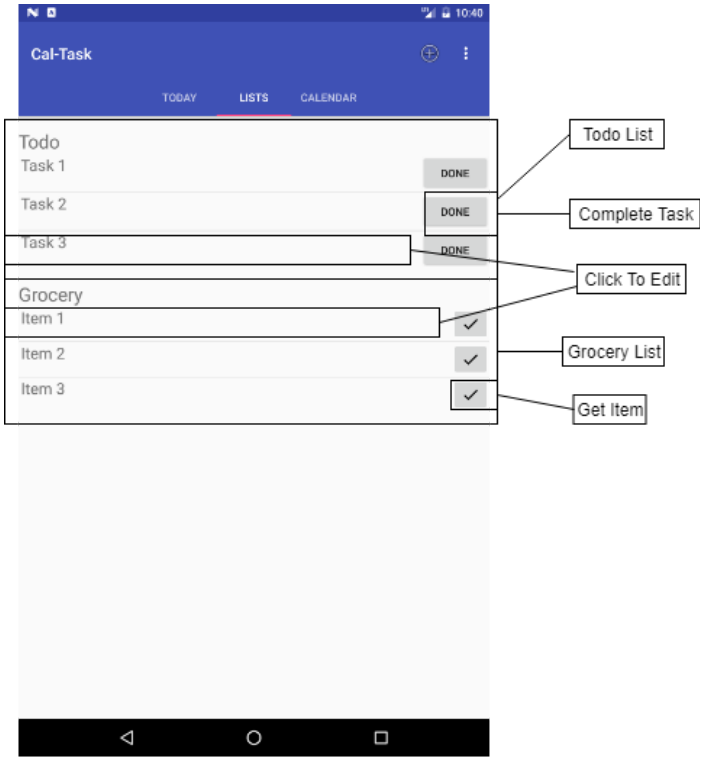
Walkthrough

The main activity sits in the background of the app, providing the Create button, Setting dropdown, and the tabbed navigation. Everything below the main toolbar is part of the currently selected fragment layout. (Today, Lists, Calendar)

Object	**Description**

Navigation Tabs	These tabs contain the titles of the various views in the application. These views can be accessed by either clicking the title of the desired view or swiping left or right.
Application Settings	This dropdown menu will allow me to add settings to the application. The activity will be shown similar to that of the button layout, with the tabbed fragments being listed as the main activity, and the settings page receiving the back arrow to exit the Settings view. **[Settings view to come in future demo]**
Create Button	The "⊕" button in the toolbar triggers an alert window containing an object creation form. This form contains: ● <Toggle> Private/Public ● <Radio Group> Task/Event/Item ● Date Entry ● Entry Title ● Cancel Object Creation ● Add Object
<Toggle> Private/Public	This toggle button determines if the object created will be shared among linked users. If the object is listed as private, only the user that created the object can see it.
<Radio Group> Task/Event/Item	This radio group lets the user select one of the radio buttons contained inside. This way an object cant get double listed by mistake.
Date Entry	This text object allows the user to enter in a date for the task/event. The input type attribute to this textarea object is set to "date," which only allows numbers and dashes to be entered into the area.
Entry Title	This text object allows the user to enter in a title for their task/event/item.
Cancel Object Creation	Cancels the current object being created.
Add Object	Adds the object to the database with the filled in attributes. **[To be implemented]**
Today's Events	ListView object that contains the events with today's date.
Today's Tasks	ListView object that contains the tasks with today's date.

Lists View



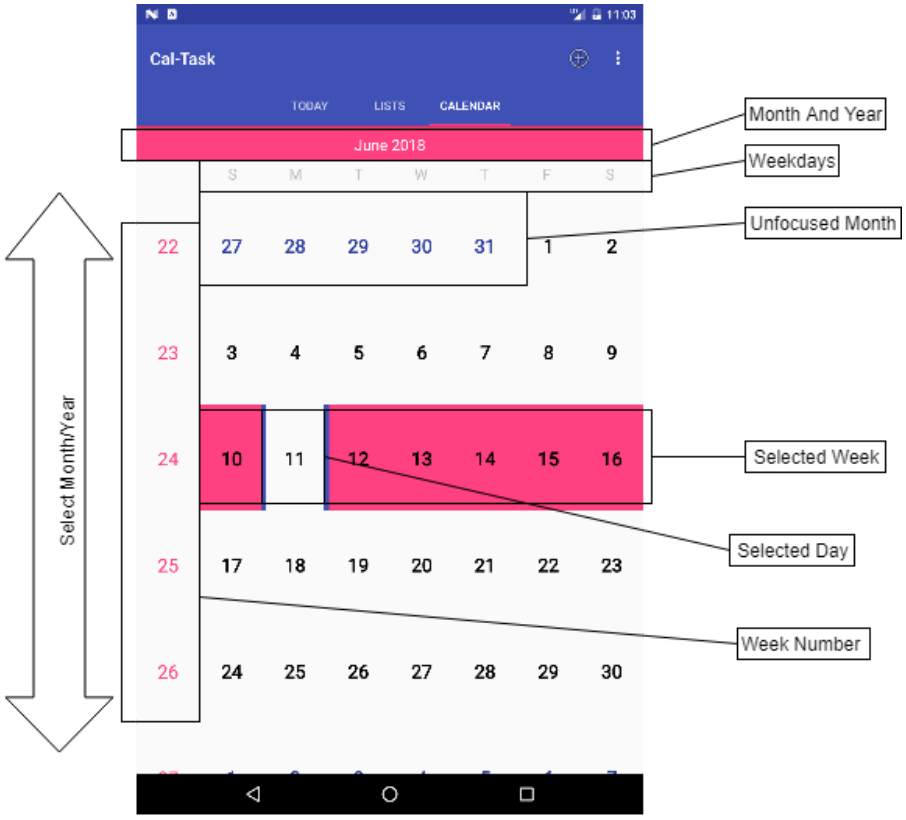
Walkthrough

The Lists view contains a Todo list to store tasks, and a Grocery list to hold items. This view will grow or shrink based on the number tasks/items that appear in the respected lists.

Object	**Description**
Todo List	ListView object that contains the tasks to be completed.
Grocery List	ListView object that contains the requested items.
Click To Edit	When the textarea is clicked on an object inside the ListViews, it will allow the user to change the attributes of the object. It does this by loading an altered version of the creation form. **[To be implemented]**
Complete Task	When the user clicks this button the task will be removed from this ListView object. If the task is public, from the online database as well. **[To be implemented]**
Get Item	When the user clicks this button the item will be removed from this ListView object. If the item is public, from the online

	database as well. **[To be implemented]**

Calendar View



Walkthrough

The Calendar view solely contains a custom CalendarView. To change to a different month or year, scroll up or down. Scrolling up will go back in time and down will advance the user forward in time.

Object	**Description**
Month And Year	Shows the month and year of the month currently in focus.
Weekdays	Shows the days of the week.
Unfocused Month	These days are from either month surrounding the currently selected month.
Selected Week	The currently selected week.
Selected Day	The currently selected day.

Week Number	Lists the week numbers of the weeks currently in view. Weeks are numbered from one to fifty-two and start over upon the new year.

References

1. Tabbed Navigation
   https://www.youtube.com/watch?v=00LLd7qr9sA
2. Navigation Drawer
   https://developer.android.com/training/implementing-navigation/nav-drawer
3. Button Navigation
   https://www.youtube.com/watch?v=n21mXO1ASJM
4. CalendarView Depreciation
   https://developer.android.com/reference/android/widget/CalendarView
5. Calendar From Scratch
   https://inducesmile.com/android/how-to-create-android-custom-calendar-view-with-events/
6. Custom Themes
   https://stackoverflow.com/questions/9412402/change-calendarview-style?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa