CIS4350_Lab06_Task List App

Due Date
Friday, Dec 15th — 03:30 pm.

Points
100 points.

Background

This project is designed to give you some hands on experience with…

- Using additional iOS widgets — e.g. `UIPickerView`, `UIDatePicker`, `UITextField`, `UISegmentedControl`, etc.
- Use of the Navigation-based app template
- Use of various delegates — e.g. `UITableViewDelegate`, `UITextFieldDelegate`, `UIPickerViewDelegate`
- Use of various delegates and data sources — e.g. `UITableViewDataSource`, `UIPickerViewDataSource`
- Persistence of custom data types

Task

For this lab you will be implementing a navigation-based app that allows a user to manage a list of tasks.

You are welcome to reuse your `Task` and `TaskList` classes from Lab 04. You may also change the implementation of these classes as you see fit for this lab. There is no requirement that you preserve the same API that was detailed for the first lab.

Behavior

When your app initially loads it should look similar to the image shown below.

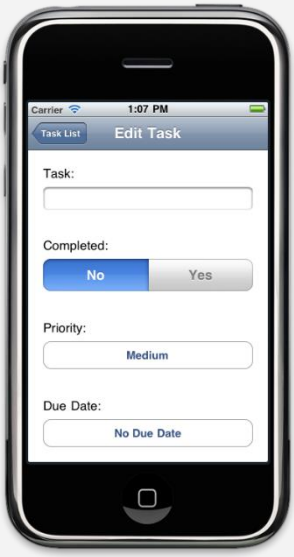
The user should be presented with a large table view with a task bar at the top. The task bar should contain 3 elements:

- A text title
- An add button
- An edit button

Task List View — Initial View

If the user presses the add button, the next view should slide in (right to left) that allows the user to edit the 4 properties of a task. There should be several UI elements on this page:

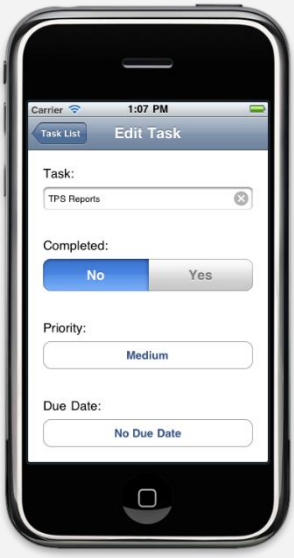
- The title bar (with a back navigation button and title)
- A text field for entering the task name
- A widget to select between completed and not completed. The actual widget you use it up to you (a `UISwitch` probably isn't appropriate since the labels of the switch are "ON" and "OFF" — here I've used a segmented control)
- A widget to display the priority of the task and bring up a view to specify the priority (widget selection is up to you — here I've used a button)
- A widget to display the due date of the task and bring up a view to specify the due date (again, widget selection is up to you — here I've used another button)



Initial Edit Task Screen

In the screenshot below, the user has specified some text as the task text and has dismissed the keyboard (you'll need to implement a `UITextField` delegate to dismiss the keyboard).

The task text field should display the clear widget icon (the "X" at the right) once text has been typed into it.



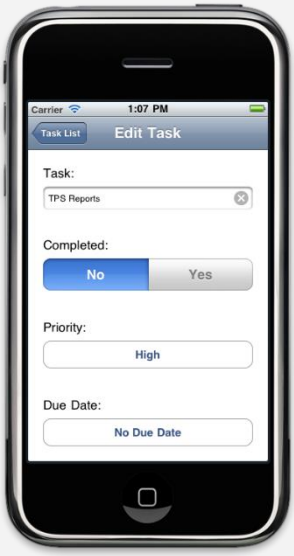
Edit Task Screen — with Text

If the user selects the priority widget, another view should be brought in (again right to left) that allows the user to change the priority. This priority screen must contain the following widgets:

- A toolbar with an appropriate title
- A custom picker that allows the user to select a priority of Low, Medium or High
- A button that allows the user to complete the selection (in the implementation shown below, it's the "Done" button in the toolbar above the picker

Priority Screen

If the user changes the priority, the widget representing the priority must be updated to reflect the current priority as shown below.



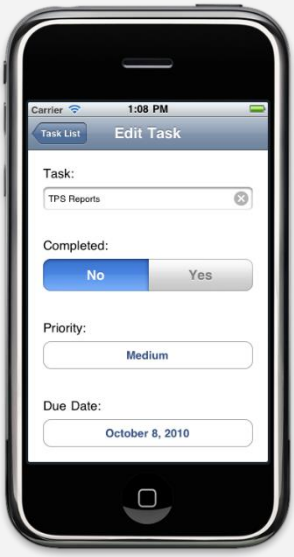
Edit Task Screen — Changed Priority

If the user selects the due date widget, another view should be brought in (again right to left) that allows the user to change the due date. This priority screen must contain the following widgets:

- A toolbar with an appropriate title
- A date picker (just date, no time) that allows the user to select a due date
- A button that allows the user to complete the selection (in the implementation shown below, it's the "Done" button in the toolbar above the picker
- A button that allows the user to specify that there's no due date associated with the given task (in the implementation shown below, this is the "No Due Date" button in the toolbar).

Due Date Screen

If the user specifies a due date, the widget representing the due date must be updated to reflect the current due date. This date should be formatted as shown below.



Edit Task Screen — Changed Due Date

Once the user enters his/her data and taps the arrow to go back to the main screen the item they edited should be added to the list. Take note that the table cell displays 2 lines of information:

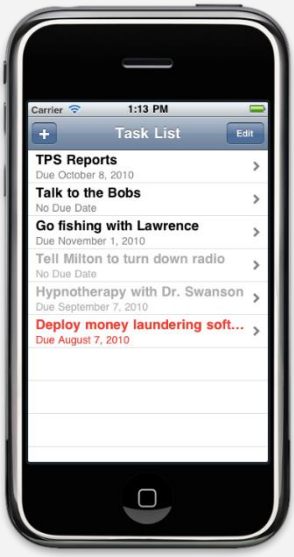
- The task text
- The due date in the same format as the task editing view (or a reasonable message if no date is specified)

Task List View — with Added Task

Below we see the main screen after several tasks have been added to the list.

Tasks should be color-coded based on the following rules:

- If the task is complete (regardless of due date) it should be colored gray
- If the task is past due (and not complete) it should be colored red
- If the task is not past due and not complete it should be colored normally



Task List View — with Multiple Tasks

Editing and Deleting Tasks

Above I've shown screenshots for creating a new task. However, in addition to listing the tasks on the main screen and adding tasks your app must also support editing and deleting tasks.

The editing flow should be pretty much the same as what's shown above. When a task is selected, it should be shown in the editing view with all of its current values populated in the UI — meaning that the text should be in the text field, the widget represented completeness should be selected to the correct value, the priority should be the task's specified priority and the date should reflect the current value for the task's due date (or a message if not specified). Likewise when the priority and due date screens are shown for an existing task, they should be pre-set to that task's values.

The app should also support the standard deletion mode as shown below. When the user presses Edit, this mode should be enabled.

Delete Mode

Persistence

Your application must persist the task list between runs of your application. There are many ways to do this, however the underlying implementation of how to do this is up to you.

Additional Ideas

The following items are provided as thoughts as to how you might extend or improve the project beyond the basic requirements. You are not required to implement any of these.

- Your task list from the first lab should have many capabilities not leveraged in this sample implementation. You could choose to leverage them here to provide additional functionality (e.g. provide another toolbar/button to remove completed tasks)
- You could add sorting options (e.g. sort by priority, due date, etc.)
- Author a more complex table cell view that displayed additional info (e.g. render a check mark in the cell if the task is complete)

Getting Started

Open Xcode, then create a new project by selecting File → New Project… then under *iOS* select *Application* and select *Single View Application*. Type name the project "**`FirstName_LastName_Lab06`**", Choose Organization name to be your first name followed by your last name (e.g., Ola Ajaj) and choose Swift for language, and finally choose *Universal* for Devices.

`** Comments **` Add a comments section on top of your **`main`** file to add your name, TUID, class, Project number, and purpose of this project.

Testing

No sample test harnesses will be provided, though you are strongly encouraged to develop your own testing scenarios.

Grading

Your projects will be graded on the following criteria:

- Correctness of application
- Adherence to Objective-C and iPhone coding conventions
- Neatly formatted and indented code
- Well documented header files

Submission

To prep your project for submission, you need to zip up your entire project directory. To do this right click (or control click) on the project folder **`FirstName_LastName_Lab06`** and select "Compress Lab06".

This will create a **`FirstName_LastName_Lab06.zip`** file that contains your project.

Simply upload **`FirstName_LastName_Lab06.zip`** through Lab06 link on Blackboard.