Calendar Event and To-Do List Manager: A Mobile Application

Prepared by Team Thundercats

Shubhangi Rakhonde
David Schechter
Zayd Hammoudeh

Date of Submission: April 30, 2015

Table of Contents

1. Summary

On April 23, 2015, Team Thundercats presented the prototype of our calendar appointment and to-do list manager mobile application. While we had previously done a calendar and to-do list application for our web application, we felt very valuable experience would be gained as we worked through the process of translating a web application to a mobile device while still ensuring a good user experience.

This document provides a brief description of our application including its key features, a source code overview, our presentation methodology as well as a discussion and comparison of five design patterns that were shared between our mobile and web applications.

2. Application Overview[1]

It is common for a person to have multiple daily calendars stored on different, disconnected platforms. Managing and visualizing these disparate calendars can be cumbersome and difficult. Our application simplifies this otherwise burdensome task by integrating all of a user's different calendars into a unified platform where a user can visualize and modify all of his/her calendars through a single, cohesive interface.

In addition to scheduled meetings and appointments, an individual usually must also complete a set of tasks, chores, errands, etc. Our application also integrates the ability to create and manage a user's tasks in the form of an advanced "to-do list".

By incorporating into a single platform an individual's calendar with the tasks s/he must perform, a user is able to easily visualize and prioritize all of his/her daily activities. Therefore, our applications' integrated approach helps prevent the inefficiencies and issues (e.g. belated completion of tasks) associated with what for most is an unstructured system to daily activity management.

3. Application Files and Source Code

Our tool was written in Objective-C and incorporated multiple open source utilities including:

- **JTCalendar**
  - **Description:** Skeleton calendar application
  - **Location:** https://github.com/jonathantribouharet/JTCalendar

- **UIView+Toast**
  - **Description:** Tool for providing simple, semi-modal messages
  - **Location:** https://github.com/taruntyagi697/RateView

- **UITextField+Shake**
  - **Description:** Tool that can shake a text field (either horizontally or vertically) on demand.
  - **Location:** https://github.com/andreamazz/UITextField-Shake

- **RateView**
  - **Description:** Slider style interface for selecting a star rating based on a scale of 0 to 5.
  - **Location:** https://github.com/taruntyagi697/RateView

[1] The "Application Overview" section is very similar to the text in our web application final report. Since the two applications are intended to have been developed theoretically by the same company targeting a single market and user base, we did not see that it made logical sense to have meaningfully different overviews.

When running this application, we used the following configuration:

- **Operating System:** Apple OS X Yosemite ver. 10.10.1
- **Runtime Environment:** Apple Xcode ver. 6.3

To run the application, first unzip the source code in the file "Assignment #4 – iOS Prototype.zip". Next, open Xcode, and import the project by clicking on the "Open another project…" button in the bottom right corner of the Xcode window as shown in figure 1.

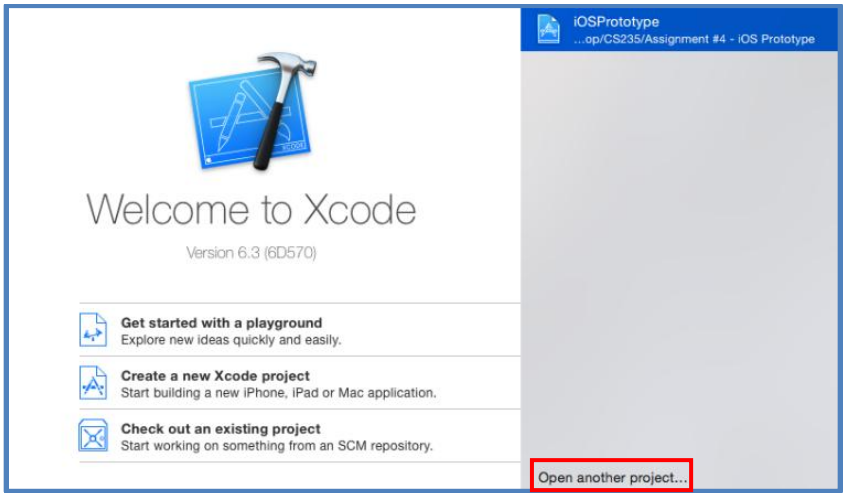

Figure 1 – Importing the Project into Xcode

A file browser will then open that allows you to browse to our Xcode project. First, navigate to the folder containing the source code and then select the subfolder named "iOSPrototype." In there, you will find an Xcode project file named "iOSPrototype.xcodeproj." Select the project file and click "Open" as shown in figure 2.

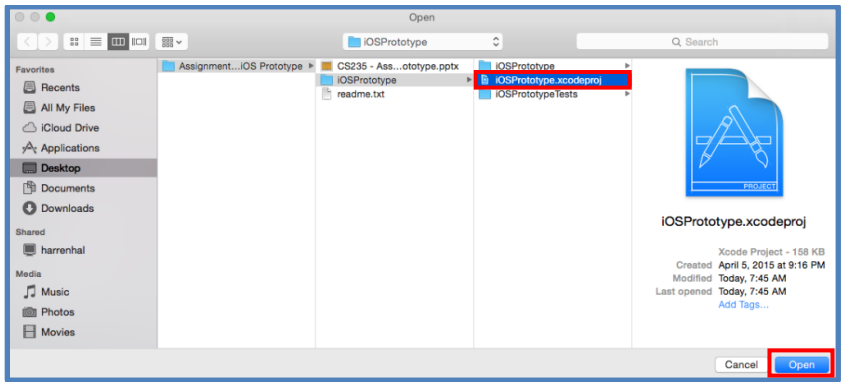

Figure 2 – Selecting the Project in the Xcode File Browser

After clicking the "Open" button, the project will automatically be displayed in Xcode. Verify that the "iPhone 6" runtime environment is selected, and click the run button as shown in figure 3. A dialog box stating "Build Successful" should appear at which point the iOS Simulator will open running our iOS application.

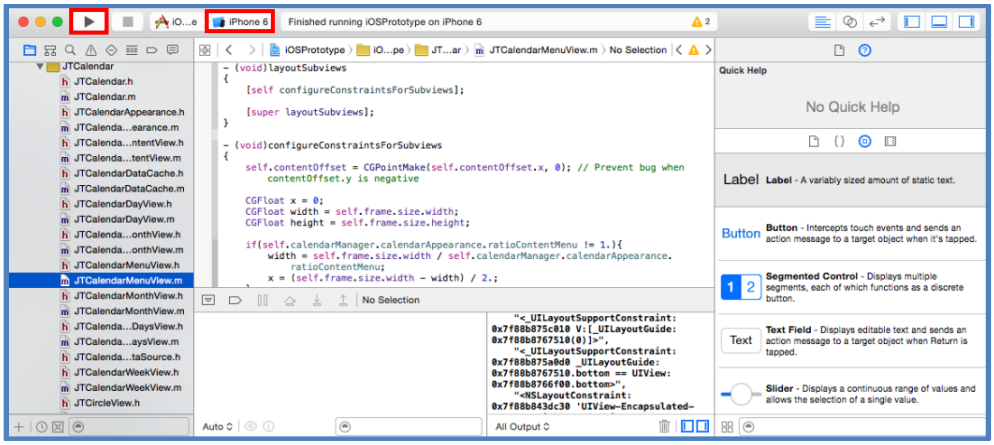

Figure 3 – Running the Project in Xcode

4. Prototype Presentation

Similar to the approach we used with our web application, our mobile application presentation only had a very small number of slides. A slide-focused presentation can quickly cause the audience to lose interest and/or become distracted. It was our position that a very short presentation followed by a longer demonstration would be more engaging and informative to the audience. Given the very positive feedback we received from both Professor Mak and the class after our web application demo, we saw no need to change our approach for this presentation. Overall, we felt the mobile application presentation was successful and engaged the audience. Our PowerPoint presentation, named "CS235 - Assignment #4 - Mobile Prototype.pptx", is included with this submission.

We received three specific points of feedback during our presentation. They are summarized below with our comments:

- **Displaying the Current Year:** In our main calendar view, we do not display the year. In one of our earlier versions, we had been displaying the year, but it made the calendar appear too cluttered so we removed it. We understand the user's desire to see the year, but in the absence of more data from usability testing, we still feel that our decision to leave it off is the right one. However, we agree that the year number should be displayed when the calendar is displaying a month not from the current year. While not implemented in our prototype, we described in our presentation the methodology we would have used which would be to shorten the month's name to an abbreviation and then display the year with it (e.g. "January 2016" would be shortened to "Jan 2016").

- **Pressing and Holding on a Date Bringing Up the Appointment Editor:** One user mentioned that if he were to double click or "press and hold" on a date that our application should bring up the appointment creation panel. Our web application demo had this feature, and it was an oversight on our part not including a similar one in this design. While we have a feature that can do a similar thing by clicking on the date then clicking on the appointment creation button, we should have simplified the process to improve the user overall experience.

- **Text Size:** Professor Mak mentioned that when displayed on the projector, the text size appeared small. David explained that this is more an artifact of a deliberately reduced resolution to make the entire simulator window fit on the display. Since none of us have paid the Apple annual subscription fee for developer mode, we are not able to load it on to a phone to verify or otherwise disprove the feedback. Hence, while we acknowledge Professor Mak's critical viewpoint, we are not able to speak more conclusively on it at this time.

5. Design Patterns

Design patterns provide solutions to often encountered software challenges; they serve as best practices that have been refined through proven design experience. In this section, we describe five of the design patterns that were incorporated into our mobile application. Note that this list is not intended to be exhaustive; rather, we selected the same five design patterns from the web application's report. While not specifically requested in the report requirements, we felt it made for a more insightful and interesting report if we compared and contrasted how we implemented the patterns in the web space versus the mobile space.

5.1. Inlay List Design Pattern

An inlay list displays a list of items (usually text based) as a single column; when a user selects/clicks on an item in the list, that item's details are displayed below it. This pattern allows for a significant amount of content to be displayed in a relatively compact space.

In our original web application, we used the Inlay List Pattern to display the user's to do list. We had decided on this methodology because a user could have dozens of items or more in their to-do list. What is more, each to-do list item will have several pieces of information associated with it including: a title, description, priority, completion due date, etc. To display all of this information at once for each to-do list item would be a large visual cognitive load and would greatly reduce the scanability of the list. In contrast, the inlay list allowed us to display the minimum amount of information possible while at the same time giving the user the flexibility to display additional details as required by that user's specific goals.

When transitioning to a web application, all of the previously enumerated benefits of inlay lists applied; what is more, the smaller screen size of a smartphone made it even more necessary to be efficient when displaying the items in the to-do list.

Figure 4 shows our mobile application's to-do inlay list in its unexpanded form. Note that for the four items in the list, only the title for each task is displayed. When the user clicks on an item in the list, the item expands to display all of its associated information. Figure 5 shows the expanded information associated with the "Mow the Lawn" to-do list item. Note that the description, due date, and priority are now displayed below the item's title.

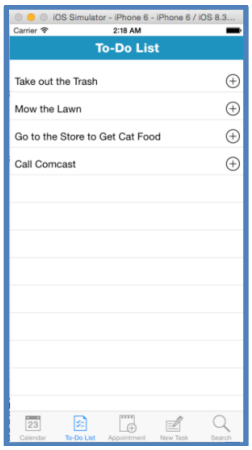

Figure 4 – Mobile Application's Unexpanded Inlay To-Do List

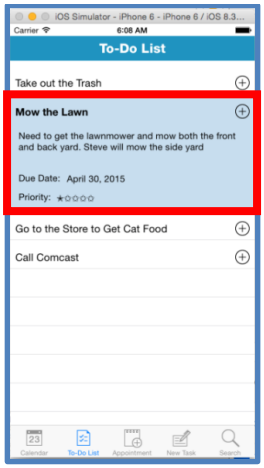

Figure 5 – Mobile Application's Expanded Inlay To-Do List

During our in class presentation, we discussed the importance of brand cohesion between the mobile and web applications. When developing the mobile application, we strived to keep the interface similar to the web application when possible. Figures 6 and 7 show the unexpanded and expanded inlay lists respectively from our web application. Similarity between the two applications allows users to build a single base of practice frequency. The following are examples of deliberate similarity between the two to-do list implementations:

1. "To-Do List" title bar text.

2. Color scheme for the title bar and text. In addition, the background colors of expanded (light blue) and unexpanded (white) to-do list tasks were preserved.

3. Organization of the text in the expanded to-do list description.

4. The plus ("+") sign to indicate that a to-do list can be expanded.

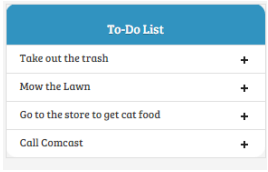

Figure 6 – Web Application's Unexpanded Inlay To-Do List

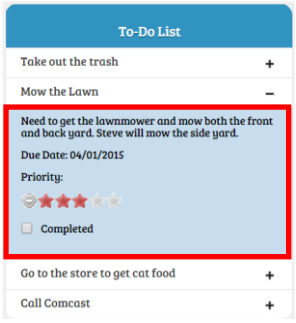

Figure 7 – Web Application's Expanded Inlay To-Do List

5.2. Alternate Views Design Pattern

When using a software program or tool, often a single, "one-size-fits-all" view is insufficient to allow a user to extract all of the requisite information from a set of data. For example, a user's calendar may be particularly full on a given day leading the user to want to view as many of that day's scheduled events as possible. In contrast, if the same user wants to know what days in the next month s/he can schedule an all day trip, s/he may want to view the whole month's calendar at once. We used the Alternate Views Pattern in both our mobile and web applications to addresses these types of varying user needs by allowing a user to select the calendar view that best suits his/her current goals. In our web application, a user can select between three primary calendar views; they are:

- Day View – View an hour by hour breakdown of a user's appointments for a specific day.
- Week View – View an hour by hour breakdown of a user's appointments for a specific week.
- Month View – A higher level view of a user's appointments for an entire month. It can also display up to three events scheduled per day.

The web application's three alternate views are shown in figures 8, 9, and 10 respectively.

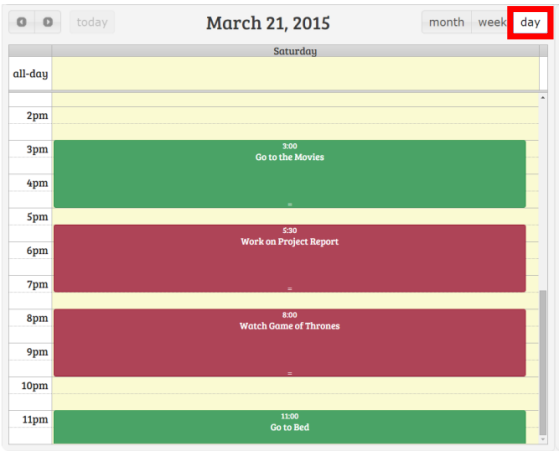

Figure 8 – Web Application's Calendar Appointments Day View

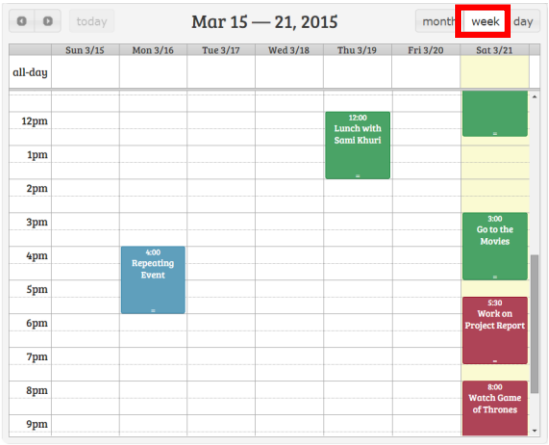

Figure 9 – Web Application's Calendar Appointments Week View

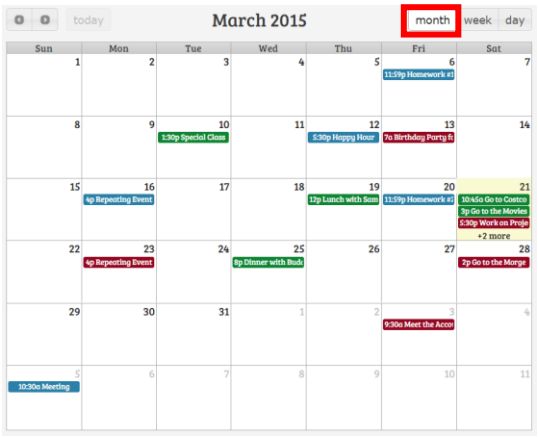

Figure 10 – Web Application's Calendar Appointments Month View

For the mobile application, we knew the Alternate Views Design Pattern would add tremendous value. For example, given the smaller screen of a mobile device, the Alternate Views Design Pattern allows us to be even more judicious in what we can display to the user. However, we had to rethink how we implemented the pattern to make it more mobile friendly. The changes we made migrating from a web interface to a mobile one are described below.

Change #1: The calendar month view changed because on a smaller screen it was not possible to show the entire month *and* some events on each day. In our mobile application, the user can select a single day (shown as a red circle), and only that selected day's tasks are displayed below the calendar month view as shown in figure 11. Note that in this view a maximum of three appointments can be displayed at once. To view other appointments, the user scrolls through the list via a "drag" gesture.

Another subtle feature we added in this revised view is a dot ("●") below those days in the calendar where the user has at least one appointment scheduled. This allows the user to not rely on recall to determine which days in the calendar are fully open.

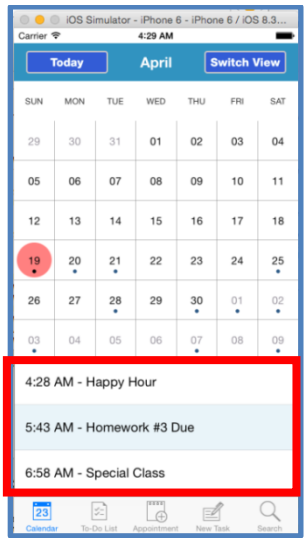

Figure 11 – Mobile Application's Calendar Appointments Month View

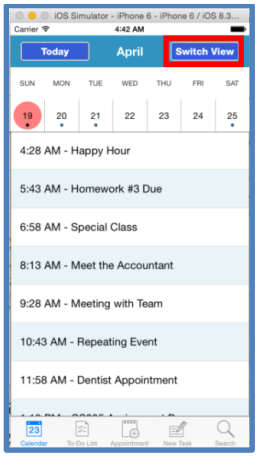

Figure 12 – Mobile Application's Calendar Appointments Day View

Change #2: For the user to get an expanded view of the tasks on a given day, the user selects that specific day in the calendar and then clicks the "Switch View" button. This will collapse the calendar to only show a week view as well as up to seven calendar appointments at once. Note that unlike the web application which provided an hour by hour breakdown of the calendar (see figure 8), we decided to simply show the tasks in chronological order since that would reduce scrolling for the user on the smaller screen. To make this view more readable, we also took advantage of the Striped Rows Design Pattern as shown in figure 12.

5.3. Prompting Text Field Design Pattern

An application is more likely to make traction with users if it is intuitive and easy to use. One of the ways to increase the intuitiveness of an application is to reduce the amount users must think when using it. The Prompting Text Field Pattern includes prefilled information inside a text field to inform the user regarding the nature of the input.

The Prompting Text Field design pattern reduces the amount a user must think while also reducing the likelihood of user error. While our original web application did use some prompting text fields, it failed to take full advantage of the pattern. Figures 13 and 14 show our web application's modal dialogs that are used to create new calendar appointments and to-do list tasks respectively. Note that the text boxes for "Name", "Task Name", "Description", "Add Invitees," and "Due Date" are empty by default with no guidance to the user. In contrast, figures 15 and 16 show how these fields are all filled with prompting text in our mobile application. This revised approach allows us to subtly and unobtrusively remind users which fields are optional and which are required.

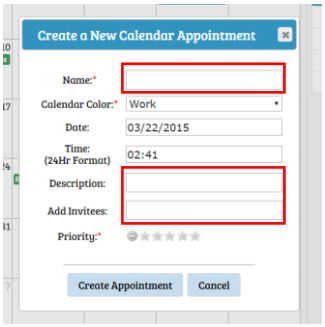

Figure 13 – Web Application's Create Calendar Appointment Modal Dialog without Prompting Text

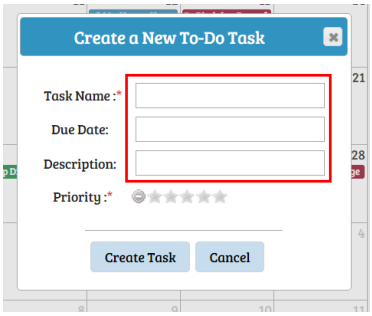

Figure 14 – Web Application's Create To-Do Task Modal Dialog without Prompting Text

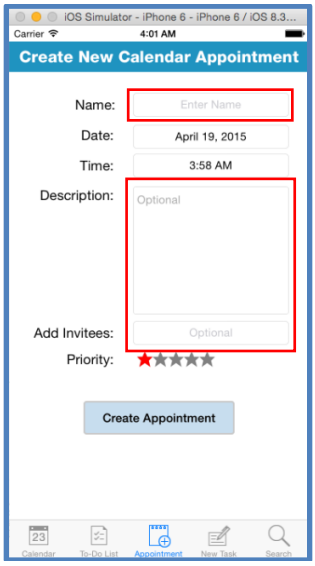

Figure 15 – Mobile Application's Create Calendar Appointment Form with Prompting Text

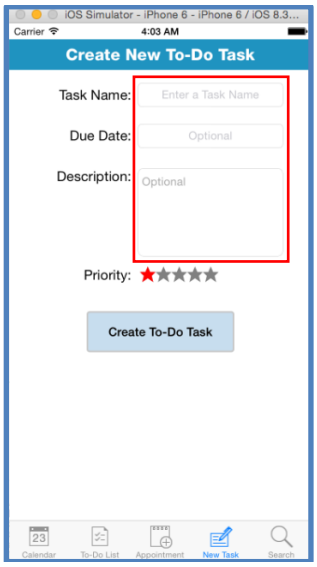

Figure 16 – Mobile Application's Create To-Do Task Form with Prompting Text

5.4. "Toast" Semi-Modal Dialog Messages

For our web application, we relied heavily on modal dialog boxes to streamline navigation and create a "hub and spoke" style navigation model. In contrast, modal dialogs generally do not translate well to a smartphone because of the smaller screen and less precise control for a finger versus a mouse. Despite these limitations, modal dialogs can still have a place in a mobile environment when used strategically.

Starting with Jelly Bean (version 4.1), Android provided an API for creating simple "Toast" semi-modal dialogs messages[2]; these prompts are intended as "simple feedback about an operation in a small pop-up". The two locations we used semi-modal "toast" messages were after calendar appointment and to-do task creation (shown in figures 17 and 18 respectively). Some may consider it wrong to use an Android feature in an iOS application. However, we felt that since this scheme is used in other applications (e.g. the now defunct Windows messenger was the originator of the idea), it was not a case of forcing something where it did not belong.

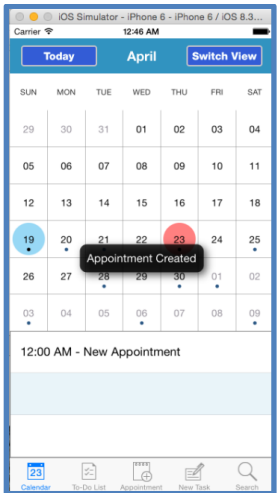

Figure 17 – "Toast" Semi-Modal Dialog after Calendar Appointment Creation

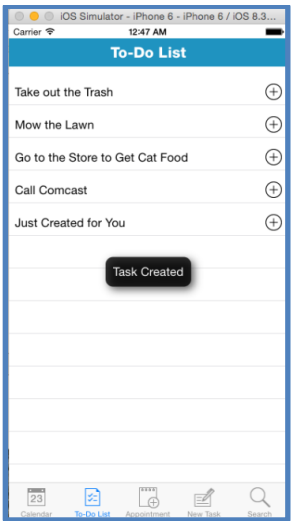

Figure 18 – "Toast" Semi-Modal Dialog after To-Do Task Creation

[2] For more information on Android's toast, please see here: http://developer.android.com/guide/topics/ui/notifiers/toasts.html.

The advantages of the "Toast" messaging scheme are:

1. **Size** – On a smartphone screen, size is often paramount leading to everything being miniaturized. This model is specifically designed for short (generally 2-3 words) messages.

2. **Instant Feedback** – For users to feel comfortable with an application, they should receive instant, obvious feedback that a specific control's action has taken place. This paradigm provides that feedback.

3. **Semi-modality** – The "Toast" message appears in front of all text so the user will not be able to accidentally hide it behind other menus. At the same time though, it is non-blocking so users can continue to perform other actions as needed.

4. **Ephemerality** – Modal dialogs, in a particular on a mobile device, can become very annoying if dismissing them requires a user action. Toast dialogs are minimally intrusive as they are only displayed for a preprogrammed length of time (ideally the display time would be determined based off feedback in a usability test). Note for this prototype, we made the toast dialogs longer than they normally would be to make it more obvious in the demo.

5.5. Dropdown Chooser Design Pattern

When a web page or mobile application is not well-designed, users often become confused when trying to enter information into a form as they may be unsure of how to correctly format the input as well as the set of possible values the field accepts. In addition, when users are forced to manually type such information, they are more likely to make a mistake. To address these types of issues with user input, we used multiple dropdown choosers in both our web and mobile applications.

Our web application used two types of dropdown choosers. The first was to select a specific date from a calendar (shown in figure 19), and the second was a time chooser (shown in figure 20).

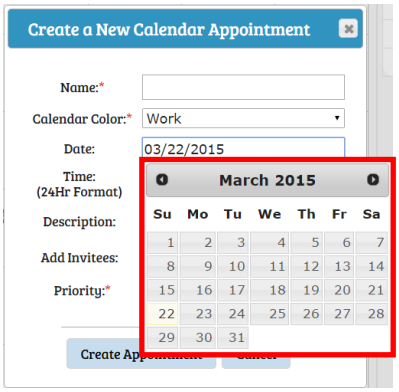

Figure 19 – Web Application's "Point-and-Click" Style Date Dropdown Chooser

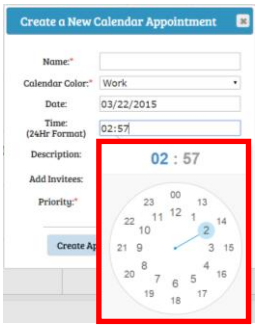

Figure 20 – Web Application's "Point-and-Click" Style Time Dropdown Chooser

While these dropdown choosers work well for a web application, they are fundamentally flawed for a mobile application as both require high precision when selecting the date/time. Figures 21 and 22 show the dropdown choosers we used in our mobile application; the date chooser was used in both the "Create New Calendar Appointment" and "Create New To-Do Task" forms while the time chooser was only used in the "Create New Calendar Appointment" form. It is important to note that the choosers were changed to a "rolodex" style in the mobile application as to the opposed to a "point and click" style used in the web application. The benefits of the rolodex style for mobile applications include:

1. **Reduction in the Requisite Precision** – Precisely controlled selections on a mobile device are very difficult or even impossible, especially for specific categories of users (e.g. the elderly). In contrast, the rolodex only requires users to move their finger up and down to move through the list of choices, which most users will find significantly more manageable.

2. **Error Forgiveness** – In a rolodex dropdown chooser, the user is able to perform fine scrolling back and forth to hone in on their target date or time. In contrast, a web dropdown chooser usually closes as soon as the user selects an item, even if the selection was not what the user intended.

While the change in dropdown chooser represents a difference between our two applications, we felt the previously enumerated benefits greatly outweighed the associated costs (e.g. requiring users to learn a data entry model).

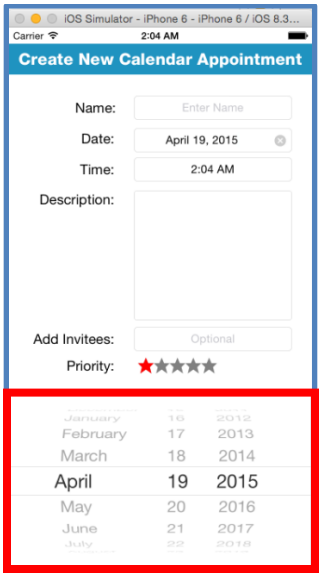

Figure 21 – Mobile Application's "Rolodex" Style Date Dropdown Chooser

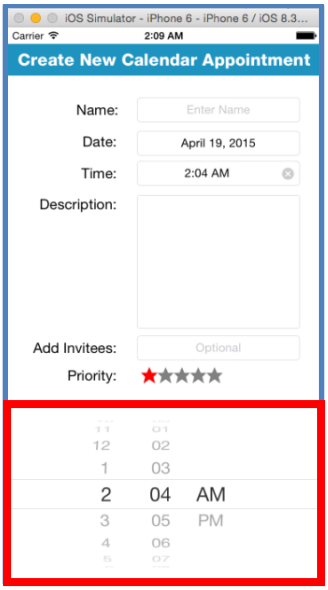

Figure 22 – Mobile Application's "Rolodex" Style Time Dropdown Chooser