**Calendar Event and   
Task List Manager:**

**Final Report**



**Prepared by Team Thundercats**

Shubhangi Rakhonde

David Schechter

Zayd Hammoudeh

**Date of Submission: April 3, 2015**

**Table of Contents**

[1. Summary 2](#_Toc414756670)

[2. Application Overview 2](#_Toc414756671)

[3. Application Files and Source Code 2](#_Toc414756672)

[4. Prototype Presentation 2](#_Toc414756673)

[5. Design Patterns 3](#_Toc414756674)

[5.1. Inlay List Design Pattern 3](#_Toc414756675)

[5.2. Alternate Views Pattern 4](#_Toc414756676)

[5.3. Prompting Text Field Pattern 4](#_Toc414756677)

[5.4. Modal Dialog Pattern 6](#_Toc414756678)

# Summary

On March 12, 2015, Team Thundercats presented the prototype for our calendar appointment and to-do list manager application. We received positive feedback from both the class as well as Professor Mak. The design met all of our key design objectives and was a success according to both our personal standards as well as in comparison to our peers.

This document provides a brief description of our application including its key features, a source code overview, our presentation methodology, and examples of five design patterns we used.

# Application Overview

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 calendar where the user can visualize and modify all of his/her calendars using 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 the user’s tasks in the form of an advanced “to-do list”.

By juxtaposing in a single interface 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, this application’s 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.

# Application Files and Source Code

Our tool used a combination of jQuery and JavaScript, including many open source utilities (e.g. jQuery FullCalendar[[1]](#footnote-1)). As with many jQuery-based websites, the application will not render correctly unless it is run inside a web server. To solve this, we used the built in web server capabilities provided by GitHub. Below is a link to our two main web pages served by GitHub.

* **Login Page:** [**http://rawgit.com/ZaydH/CS235/master/Assignment\_2\_ToDo\_Web\_Application/login.html**](http://rawgit.com/ZaydH/CS235/master/Assignment_2_ToDo_Web_Application/login.html)
* **Main Page:** [**http://rawgit.com/ZaydH/CS235/master/Assignment\_2\_ToDo\_Web\_Application/mainpage.html**](http://rawgit.com/ZaydH/CS235/master/Assignment_2_ToDo_Web_Application/mainpage.html)

In our submission, we also included our source code and libraries in a zip file named: “Assignment\_2\_ToDo\_Web\_Application.zip”. This contains all of the source code; we used Aptana Studio[[2]](#footnote-2) to locally serve the page during debug. However, for evaluation purposes, the GitHub server should be sufficient.

# Prototype Presentation

In contrast to the other teams, our prototype presentation utilized only a very small number of slides. A slide-focused presentation can quickly result in the audience losing interest or becoming 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 feedback we received from both Professor Mak and our peers in the class, we felt this strategy was very successful. Our PowerPoint presentation, named “CS235 - Assignment #3 - Application Prototype.pptx”, is included in this submission.

# Design Patterns

Design patterns provide solutions to regularly encountered programming challenges; they serve as best practices that have been refined through proven design experience. In this section, we describe five of the design patterns we included in our design. Note that this list is not intended to be exhaustive; rather, when selecting which design patterns to describe in the subsequent sections, we strove to balance the criticality of the pattern to our overall design with how well we thought the implementation could be described in this format.

## 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, the details of that item are displayed below it. This pattern allows for a significant amount of content to be displayed in a relatively compact space.

In our application, we used the inlay list to display the user’s to do list. We 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 may have several pieces of information associated with it including: a name, description, priority, a 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.

Figure 1 shows our application’s to-do inlay list in its unexpended form. Note that for the four items in the list, only the task title is display. When the user clicks on an item in the list, the item expands to display all of its associated information. Figure 2 shows the expanded information associated with the “Mow the Lawn” to-do list item. Note that the description, due date, priority, and completion check box below that item are displayed.

We included two additional features in our to-do inlay list to improve the overall implementation of this pattern. First, when a to-do list item is unexpanded, it has a “+” next to its name as indication to the user that this item can be expanded. Similarly, when a to-do list item is expanded, the “+” changes to a “-“ to indicate to the user that this item can be collapsed. While this may appear subtle to some users, for others, it is a clear affordance to indicate that the structure is an inlay list.

The second additional feature we added was to use color to distinguish between a to-do list item’s title (which always has a white background) and its description field (which always has a light blue background). This will allow a user to quickly and visually recognize the nature of a particular section of displayed text without relying recall.



Figure – Unexpanded Inlay To-Do List



Figure – Expanded Inlay To-Do List

## 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 in any calendar application, a user may want to see all of the meetings s/he has on one specific day; if his/her calendar is particularly full on that day or if s/he has little interest regarding the other days in his/her calendar, s/he may just want to see exclusively that day’s scheduled meetings. 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. The Alternate Views Pattern addresses these types of varying user goals by allowing a user to pick the view that best suits their goals. In our application, a user can select between three primary calendar views:

* 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.

These three alternate views are shown in figures 3, 4, and 5 respectively. Note that the view is selectable by clicking on the name of the view (e.g. “day”, “week”, “month”) in the upper right corner of the calendar as shown in each of these figures.

## Prompting Text Field Design Pattern

The more intuitive and obvious a web page is, the more likely it is that it will make traction with users. One of the ways to increase the intuitiveness of a page is to reduce the amount users must think. The Prompting Text Field Pattern includes prefilled information inside a text field to inform the user regarding the nature of the expected input.

We used this pattern in two different places in our application. The first is on the login page as shown in figure 6; note that in this form, the username field is prepopulated with the text “Username” while the password field has the characters masked to prompt the user this is the password field. While the password prompting is somewhat more subtle than the model used for the “Username” field, we felt this was sufficient for the general user.

The second place we used the Prompting Text Field Pattern was in our page’s search box as shown in figure 7. Since the search field was placed above the to-do list, we had concerns that some users may that that the search feature applied only to the to-do list and not the whole tool. As such, we placed inside the search field the prompt “Search Calendar & To-Do List” to inform the user of the tool’s full functionality.



Figure – Calendar Appointments Day View



Figure – Calendar Appointments Week View



Figure – Calendar Appointments Month View

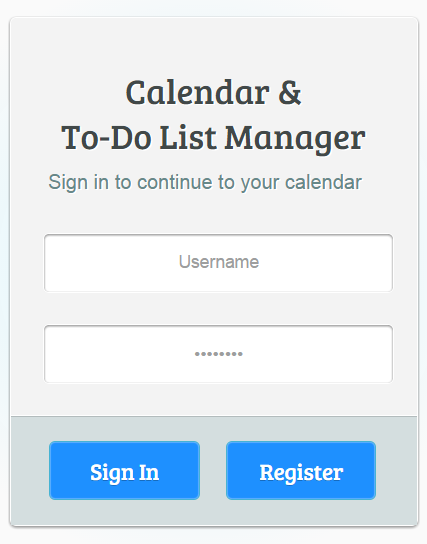


Figure – Login Page with Prompting Text Fields for Username and Password

Search Prompting Text Field.png

Figure – Search Box as a Prompting Text Field

## Modal Dialog Design Pattern

If a site requires significant amounts of navigation to access different features, then users may find the page difficult to use and/or confusing. One way to minimize the amount of page navigation is through modal dialog, which open on top of the existing. Through modal dialogs, a page can create a hub and spoke like affect where the user never actually leaves the main page (i.e. hub) but accesses features through numerous modal dialogs (i.e. spokes).

Our design relies heavily on modal dialog boxes to handle almost all types of user interaction including: creating a calendar appointment, creating a to-do task, and marking a to-do task as completed; these modal dialog boxes are shown in figures 8, 9, and 10 respectively.

## Drop Down Chooser Design Pattern

If a web page is not designed well, users can become confused when trying to enter information into a form as they may be unsure of how to properly format the information as well as the set of possible values the field accepts. One of the solutions to this problem is to use a drop down chooser.

In our applications, we recognized two specific types of dropdown choosers that would improve the user experience. The first type is a date or calendar style drop down chooser where a user can select a specific day

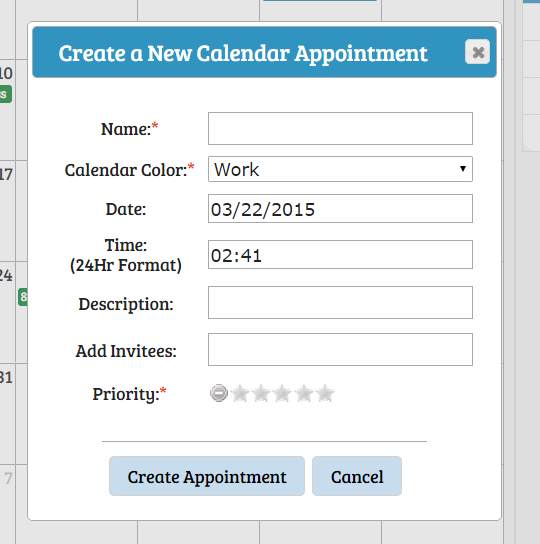


Figure – Modal Dialog to Create a Calendar Appointment

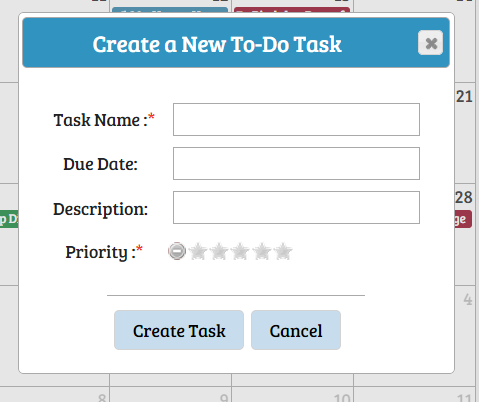


Figure – Modal Dialog to Create a To-Do Task

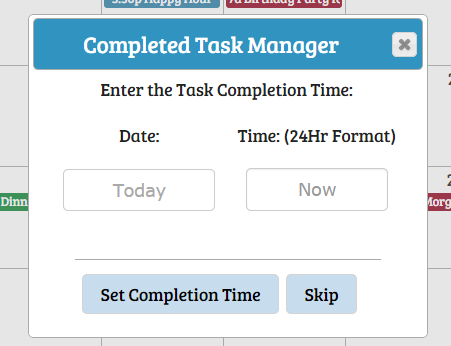


Figure – Modal Dialog to Mark a To-Do Task Completed

from a monthly view. This type of drop down chooser is shown in figure 10 and was used in all three of our modal dialogs shown in figures 8, 9, and 10.

The second type of drop down chooser we utilized was a time chooser where the user could select time specific time that would be displayed in 24 hour formatting. The drop down chooser is shown in figure 10 as was used in the calendar appointment creation and to-do task completion modal dialogs shown figures 8 and 10.



Figure – Calendar Date Selector Drop Down Chooser



Figure – Time Selector Drop Down Chooser

1. JQuery Full Calendar is available at: <http://fullcalendar.io/>. [↑](#footnote-ref-1)
2. Aptana Studio 3 is available at: <http://www.aptana.com/>. It is based off of Eclipse and includes a built in web server in its debug mode. [↑](#footnote-ref-2)