**Event and Task List Manager**



**Prepared by Team Thundercats**

David Smith

Shubhangi Rakhonde

David Schechter

Zayd Hammoudeh

# Table of Contents

[Table of Contents ii](#_Toc410499921)

[1. Application Name 1](#_Toc410499922)

[2. Application Description 1](#_Toc410499923)

[3. Functional Requirements 1](#_Toc410499924)

[3.1. Functional Requirements Related to Tasks and Events 1](#_Toc410499925)

[3.2. Functional Requirements Related to Event and Task Priority 1](#_Toc410499926)

[3.3. Functional Requirements Related to Synchronous Calendar Events 2](#_Toc410499927)

[3.4. Functional Requirements Related to Asynchronous Task Completion 2](#_Toc410499928)

# Application Name

Event and Task List Manager

# Application Description

This section provides a summary of the application intended purpose as well as the

This application simplifies the management of a user’s multiple daily calendards/planners as well as the user’s to-do/task list by integrating them into a single, cohesive interface.

This application retrieves a user’s appointment ecalendar from across multiple competing platforms (e.g. Google calendar, Facebook, etc.) and displays them

# Functional Requirements

This section reviews the different functional features that the event and task manager and scheduler must include. They have been divided into different categories based on their logical role in the application.

## Functional Requirements Related to Events and Tasks

1. The application needs to support two distinct types of events and tasks. The application shall track these differing types separately since there is limited overlap in their nature.
   1. The first type of event is “synchronous” in nature in that they are associated to a specific time and date on the calendar. Examples of synchronous events would be a doctor’s appointment, business meeting, and a birthday party since they all occur at specific times.
   2. The second type of task is “asynchronous” in that they still need to be done, but the user has significant flexibility regarding the time they can be done. For instance, a possible asynchronous task could be to cut the grass or to go shopping since the user can do them. In contrast, if the user only had access to the lawnmower from 2-3pm on Saturday, mowing the grass would turn from being asynchronous to synchronous.
2. The application shall display synchronous events and asynchronous events in a side-by side two panel view as shown in Figure 1.

## Functional Requirements Related to Event and Task Priority

1. The user shall be able to assign each synchronous calendar event as well as all asynchronous tasks a priority.
2. The calendar events and asynchronous task priorities shall be ranked on a scale of zero to five stars, with zero stars being the lowest priority and five stars being the highest priority.
3. Tasks and events with different priorities shall be color coded in the system. The color scheme used must make higher priority events and tasks appear more prominently than their lower priority counterparts.



Figure – Basic Overview of Interface Structure

## Functional Requirements Related to Synchronous Calendar Events

1. The application shall allow the user the user to integrate their synchronous calendar information from other applications/services, including but not necessarily limited to Google calendar, Facebook, and Apple’s calendar application.
2. The application must support the ability to synchronous calendar events that were not imported from another application (e.g. Facebook). The application must allow the user to specify the following information as part of these user created events:
   1. Event name
   2. Event time and date
   3. Event description (if any)
   4. Invitee List (if any)
   5. Event recurrence (e.g. once a week, every Tuesday/Thursday - if any)
3. The application must support the ability to create and appropriately display multiple synchronous calendar events that overlap in time (e.g. two separate meetings scheduled at exactly the same time).

## Functional Requirements Related to Asynchronous Task Completion

1. The application must support the ability to categorize asynchronous tasks as either uncompleted or completed.
2. Once an asynchronous task has been completed, the user shall be able to mark it as “Completed” by clicking a check box next to the task.
3. Once a task has been marked as completed, the application must allow the user to specify a task completion time. This completion time can be either the current time or a time specified by the user.
4. Upon an asynchronous task’s completion, the application must automatically remove the task from the set of uncompleted tasks and instead include it in the set of completed tasks.

Functional