Gabriel Martinez-Amezaga, Kaz Tam, Gabriella Best, Andisha Safdariyan, Danish Safdariyan

WIP Report

CS 3110: Data Structures and Functional Programming

March 28, 2024

**WIP Report**

Vision

To reiterate what we reported in MS1, we seek to create a niche calendar program that has various functions and a consistent, aesthetically pleasing, graphical user interface. A specific goal we have in mind is to create three main uses: a regular calendar, a schedule manager, and a task manager. The regular calendar will include normal functionality such as creating repeating events or notating important dates. The schedule manager will be integrated with Cornell scheduler (as in, importing a file exported from scheduler), and be very similar to the implementation of scheduler. The task manager will be a normal task manager that keeps track of impending tasks. The purpose of this program is to provide a time management method that is easy to use and convenient as well.

Some of us within the group are currently frustrated with the different calendar and task management options available to us. More often than not, a calendar app does not encompass what we want to use out of it (ex: different views that include different events/schedules). Additionally, we want to combine this calendar functionality with an integrated task manager window in the program as well. The goal is to create a computer program for a task management system and organizational system that encompasses different uses and is simple to use.

We decided to name the program: *Trek*.

**WIP Report Continued**

Summary of Progress

During MS1, we created a vision for what we want our program to do. For MS2, we wanted to do two essential things that would set the foundation for the rest of the project. First, we wanted to figure out a data structure to represent calendar data. For that we decided on using a simple dictionary/map that currently has the structure of [(k , v)] where the keys are strings that represent a day, and the values are strings representing events such as a birthday. In other terms, the structure is [( day , event )]. This is fine for our current implementation, but we don’t foresee using this exact implementation for the rest of the project. Second, we wanted to find a GUI library that we would work with for the project. After experimenting with the OCaml graphics library and deciding it would not work long term, we settled on using Bogue. The link to Bogue is below: <https://github.com/sanette/bogue/blob/master/README.md>.

For our WIP, we implemented a basic GUI using Bogue and represented the map/dictionary on that GUI. This is just a basic GUI and representation that will definitely not stay the same long term, but we wanted to establish a basic foundation to work off of, which we did. Although these foundations (the current implementation of our data structure and GUI) will probably not work in the long run, the goal of establishing a foundation has been achieved and we are more comfortable with moving forward.

The current implementation of the program takes simple user input. It takes an event that you would like to put on a “calendar” (a 35-day grid) and a day (ranging from Day 01 to Day 35) to put that event on. Once these are inputted, the GUI is shown with inputted event on the inputted date. This is all that we have implemented so far, just to start a foundation.

**WIP Report Continued**

Activity Breakdown

*Danish:*

* **Responsibilities:** Back-end functionality (data structure). Developing dictionary data structure. Collaborated primarily with Gabriella and Andisha on back-end.
* **Activities:** Group meetings.
* **Features:** [CalDict] compilation unit polishing. Implemented part of [main.ml] that involved integrating dictionary use with GUI.
* **Number of hours:** 4.5 hours.

*Andisha:*

* **Responsibilities:** Back-end functionality (data structure). Developing dictionary data structure. Collaborated primarily with Gabriella and Danish on back-end.
* **Activities:** Group meetings.
* **Features:** [CalDict] compilation unit functionality and implementation.
* **Number of hours:** 4.0 hours.

*Gabriella:*

* **Responsibilities:** Back-end functionality (data structure). Developing dictionary data structure. Collaborated primarily with Danish and Andisha on back-end.
* **Activities:** Group meetings.
* **Features:** [CalDict] compilation unit functionality and implementation.
* **Number of hours:** 4.0 hours.

**WIP Report Continued**

Activity Breakdown Continued

*Kaz:*

* **Responsibilities:** Front-end (GUI). Implementing GUI functionality. Collaborated with Gabriel on front-end.
* **Activities:** Group meetings.
* **Features:** [Display] compilation unit functionality and implementation. Partially implemented integration of the GUI and dictionary in [main.ml].
* **Number of hours:** 5.5 hours.

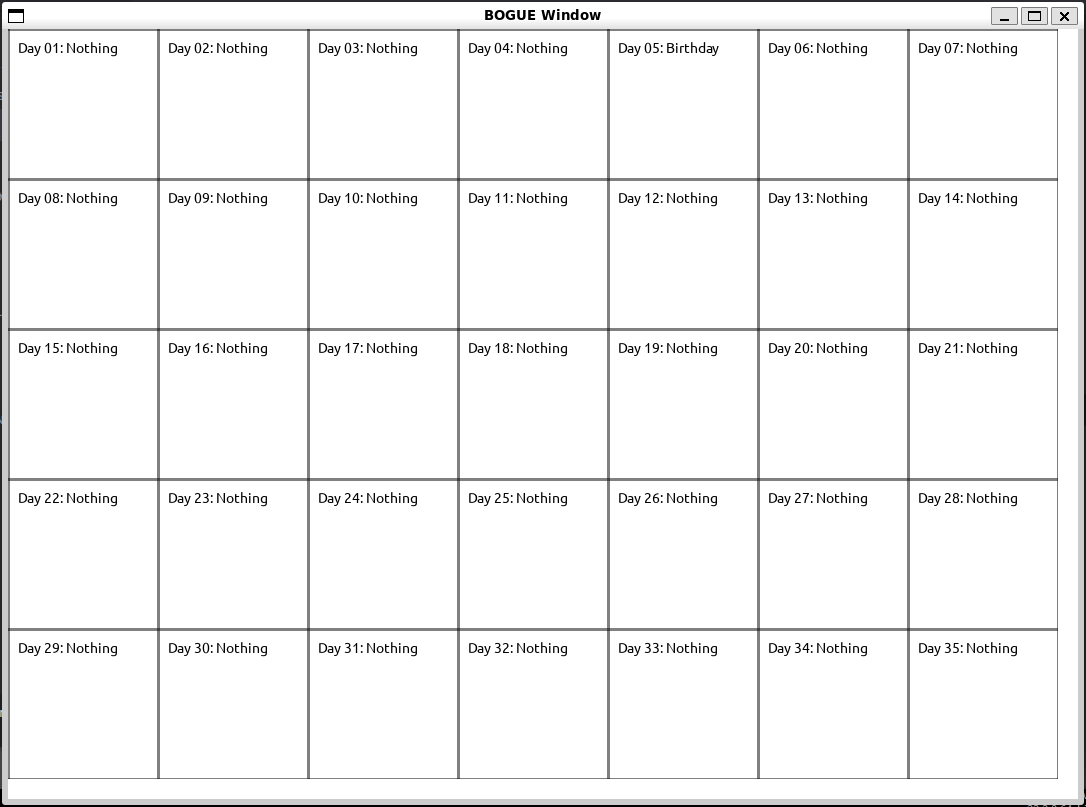
*Gabriel:*

* **Responsibilities:** Front-end (GUI). Choosing GUI library (bogue). Polishing GUI implementation. Writing MS2 report. Collaborated with Kaz on front-end.
* **Activities:** Group meetings.
* **Features:** [Display] compilation unit polishing. Finished and polished the integration of the GUI and dictionary in [main.ml]. Tested both GUI and [main.ml].
* **Number of hours:** 5.0 hours.

**WIP Report Continued**

Productivity Analysis

Our team accomplished what we wanted to during the sprint. It was a journey that had uncertainty to it, but we pulled through in the end. Unfortunately, the majority of our team had a very, very busy schedule during the week of the sprint, so communication was lacking/limited at times. At times, the team was not as productive as it could be and the aforementioned uncertainty during the sprint was caused by this, but after communicating and resolving toward the end of the sprint, the sprint was completed to satisfaction. Our goal was to print a month of days onto some kind of graphical user interface using a data structure. We accomplished this to satisfaction and even took in some user input that could change one of the data cells. We are satisfied with this foundation and proof of concept for this sprint.

*GUI Example:*