Home Base
for UTM CSCI 352 Fall 2018

Adam Chisolm
Lukas Saul

Abstract

Our intention with this project is to create a simple-to-use calendar/reminder program to run on a desktop/laptop computer. It will keep track of events, reminders, and the weather while also being more customizable than most similar programs.

1. Introduction

Our goal for this program is to create a calendar system that will integrate locally stored events and reminders with Google Calendar, through the provided Google Calendar API. We expect people who struggle keeping track of their busy schedules to run the program on startup so that they can be reminded of their schedule, weather forecasts, reminders, etc. so they may be kept on track and prepare for the day. We hope that people will find that their lives are made a little easier having this program reminding them of what they need to do.

The ultimate goals for this semester regarding this program is to create an independent reminder and event system that will help someone keep track of their busy lives. We also want to integrate a weather forecast so that one may be better prepared once they leave the house.

1.1. Background

This program will be fairly similar to Google Calendar or Window's calendar even but much more customizable in that you will be able to change the background and layout to a couple different presets. We have witnessed that a great many users have trouble staying on track with their busy schedules, which is why we'll be creating this app to help people stay organized.

1.2. Challenges

We believe that the hardest part of this project for now will be creating a reliable system for reminders and events. This might require making the progam runnable in the background and adding Windows reminders for the user in the fashion of the notifications that you receive in the bottom right in Windows 10.

2. Scope

This project will be finished once the user can store an event/reminder in the calendar and be successfully reminded of the event. The reminders should synchronize with the user's Google calendar. We also want to have a number of customization options available to the user to make the program look how they want it to look. This would include customizable background colors/pictures and layouts.

2.1. Requirements

We developed these requirements from what we have seen as good ideas in calendar programs and what we feel like most users would like to see in a calendar program.

2.1.1. Functional.

- User needs to have a Google account. This will allow them to syncronize their local calendar with their Google Calendar.
- The program will need to store events locally and keep track of them.
- The user should be able to alter the aesthetic of the program via background pictures, colors, sizes, layout.
- The user should be able to add a "reminder" or short term event, and be notified of its happening on the desktop.

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Add task to calendar	User	Med	1
2	Complete task in calendar	User	Med	1
3	Delete task in calendar	User	Med	1

TABLE 1. SAMPLE USE CASE TABLE

2.1.2. Non-Functional.

- Reliability – user created events must be easily accessed again and again without the loss of an event.
- Backup – user created events must be able to be stored as a whole to local memory in case of failure.

2.2. Use Cases

The User should be able to add a task/reminder to their calander. They should then be able to ”check off” the task if they completed it, or delete the task if they are no longer planning on completing it. See Table 1.

Use Case Number: 1
Use Case Name: Add item to cart
Description: A shopper on our site has identified an item they wish to buy. They will click on a “Add to Cart” button. This will kick off a process to add one instance of the item to their cart.

You will then go on to (minimally) discuss a basic flow for the process:

1) User navigates to page listing desired item
2) User left-clicks on “Add to Cart” button.
3) User cart is updated to reflect the new item, this also updates the current total.

Termination Outcome: The user now has a single instance of the item in their cart.

You may need to also add in any alternative flows:
Alternative: Item already exists in the cart

1) User navigates to page listing desired item
2) User left-clicks on “Add to Cart” button.
3) User cart is updated to reflect the new item, showing that one more instance of the existing item has been added. This also updates the current total.

Termination Outcome: The user now has multiple instances of the item in their cart.

You will often also need to include pictures or diagrams. It is quite common to see use-case diagrams in such write-ups. To properly reference an image, you will need to use the `figure` environment and will need to reference it in your text (via the `ref` command) (see Figure 1). NOTE: this is not a use case diagram, but a kitten.

After fully describing a use case, it is time to move on to the next use case:

Use Case Number: 2
Use Case Name: Checkout
Description: A shopper on our site has finished shopping. They will click on a “Checkout” button. This will kick off a process to calculate cart total, any taxes, shipping rates, and collect payment from the shopper.

You will then need to continue to flesh out all use cases you have identified for your project.

2.3. Interface Mockups

At first, this will largely be completely made up, as you get further along in your project, and closer to a final product, this will typically become simple screenshots of your running application.

In this subsection, you will be showing what the screen should look like as the user moves through various use cases (make sure to tie the interface mockups back to the specific use cases they illustrate).

3. Project Timeline

Go back to your notes and look up a typical project development life cycle for the Waterfall approach. How will you follow this life cycle over the remainder of this semester? This will usually involve a chart showing your proposed timeline, with specific milestones plotted out. Make sure you have deliverable dates from the course schedule listed, with a plan to meet them (NOTE: these are generally optimistic deadlines).

Figure 1. First picture, this is a kitten, not a use case diagram

4. Project Structure

At first, this will be a little empty (it will need to be filled in by the time you turn in your final report). This is your chance to discuss all of your design decisions (consider this the README's big brother).

4.1. UML Outline

Show the full structure of your program. Make sure to keep on updating this section as your project evolves (you often start out with one plan, but end up modifying things as you move along). As a note, while Dia fails miserably at generating pdfs (probably my fault), I have had much success with png files. Make sure to wrap your images in a `figure` environment, and to reference with the `ref` command. For example, see Figure 2.

4.2. Design Patterns Used

Make sure to actually use at least 2 design patterns from this class. This is not normally part of such documentation, but largely just specific to this class – I want to see you use the patterns!

5. Results

This section will start out a little vague, but it should grow as your project evolves. With each deliverable you hand in, give me a final summary of where your project stands. By the end, this should be a reflective section discussing how many of your original goals you managed to attain/how many desired use cases you implemented/how many extra features you added.

5.1. Future Work

Where are you going next with your project? For early deliverables, what are your next steps? (HINT: you will typically want to look back at your timeline and evaluate: did you meet your expected goals? Are you ahead of schedule? Did you decide to shift gears and implement a new feature?) By the end, what do you plan on doing with this project? Will you try to sell it? Set it on fire? Link to it on your resume and forget it exists?

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

Figure 2. Your figures should be in the *figure* environment, and have captions. Should also be of diagrams pertaining to your project, not random internet kittens