Productive Project Manager

Aaron Alden and Spencer Karpati

Abstract

For our semester project, we decided to implement our own version of a project manager. Our target audience is any group of people that are working on either small or large-scale projects that are more complex in scope and need sufficient time management. Features will include a calendar, member-to-member communication, deadline reminders, etc. So far, we have determined what we want to do for our project and will now begin to figure out design patterns and high-level implementation.

1. Introduction

Have you ever found yourself absolutely swamped with an overbearing workflow due to a high workload, lack of organization, and poor time management? Are you frustrated that big names in project management software require a monthly subscription making them not as accessible? Do you wish there was a straightforward program that can do the most necessary tasks of any type of project management software but also does not require a monthly subscription? Fret not dear user, as we have a solution to your issue.

Productive is a project management software that plans to provide the most needed and required features without bloating itself with unnecessary additions like AI or webinars like other project management software making the user experience as straightforward and hassle-free as possible. With this scope of the project in mind, Productive can be accessible to all as it doesn't need a monetization scheme to cover operating costs as the tools we will use to create the program are free and accessible to us. The combination of an easy-to-use interface as well as the overall accessibility of Productive is what makes it a powerful tool for any user.

1.1. Background

Initially, we found it challenging to decide what we wanted to do for our project. We realized that being that this is our first time doing a project of this scope, we recognized that planning out the implementation of a large project could prove difficult if not properly planned/tracked well. This led us to want to build a project revolving around project planning and management. Other project management applications, such as Hive, Nifty, and Microsoft Teams are excellent, but usually, come with a monthly subscription and are usually bloated with many different sets of features. With Productive, we hope to "trim the fat" by implementing only the most necessary features and being free to all. With this, we hope that it will be able to improve overall efficiency and productivity between individuals and the group as a whole as well as to be completely accessible to everyone.

1.2. Impacts

With this application, our goal is to ensure that teams use their time and resources efficiently and that productivity is consistent with the overall expectations and timelines of the project as well as wanting the user experience to be straightforward and easy to use without unnecessary complications or distractions. We predict that by starting early, we can make design decisions that aren't bloated but streamline the usage of the application. With these design concepts in mind, we want our application to overall ease the workflow and stress of our users when dealing with larger-scale projects that require more sufficient time management, organization, and overall efficient workflow by allowing the user access to a variety of project management tools.

1.3. Challenges

The biggest challenge of the project would be the networking aspect as we want to implement a chat room for team members to speak with each other as well as make sure the user UI is simple and user-friendly to make the application more accessible. We will look into this further as we get closer to implementation. Another challenge will be making sure that the application will continuously update information continuously to ensure proper communication between all users in a group whenever a new data object is added as well as making sure that the application's database is also properly updated.

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Add event to calendar	User	Med	1
2	Quick-note feature	User	Low	3
3	Notifications for events	User	Med	2

TABLE 1. USE CASE TABLE

2. Scope

We believe that the application should have a complete suite of tools for making sure that project reminders/deadlines are sufficiently communicated to the user and that the user or team of users can segment their workflow across periods of the project's lifespan to consider the project to be finished. This will include a calendar with planning tools (adding/deleting events at a certain date, forwarding events to other users, changing event details, assigning events to other users in a group, etc) notifications (usually showing up from the user's taskbar) for communicating upcoming deadlines and messages received, and user-to-user text and voice communication.
Our three initial stretch goals are:

1) A voice-chat feature for quicker and more efficient communication between two users.
2) Group voice chat between a multitude of users for group meetings or discussions.
3) Ability for users to import or export events from other calendar applications such as Google Calendar, Apple Calendar, etc.

2.1. Requirements

When looking at other project management software like Teams, we wanted to look at the most basic and necessary features when deciding what tools Productive should have. Once we decided what tools we wanted for Productive, we then brainstormed what was absolutely needed to ensure that Productive met all of its requirements. Then, we decided on some non-functional requirements that we believe a project manager should have.

2.1.1. Functional.

- The user needs to have access to a database. By having a database, the user's relevant information and calendar data can be stored. For example, importing or exporting a calendar, storing inputted events and tasks, etc.
- The user needs an in-software notebook. This will allow the user to write important notes within the project interface. This can also be exported as a PDF to be shared offline.
- As a key part of a project manager, the user needs to have a reliable and easy-to-use planning and task-scheduling tool. This includes features such as a shareable team calendar, the ability to prioritize tasks, etc.
- If we do manage to reach the goal of implementing network features into our program, we need to implement basic networking features such as a P2P structure so that users can act as both a client and server to receive and send messages.
- The user needs to have a "home screen" which in this case would be the project dashboard. It should allow the user to visualize the progress of each project task, as well as shortcuts to other key features of the program.

2.1.2. Non-Functional.

- Usability - Ergonomic and practical performance whilst traversing the project dashboard/the application itself.
- Accessibility - This program will be open-sourced and free, available to anyone who could take advantage of its features.
- Reliability - During the implementation of the program, user changes will constantly be updated, and saved upcoming events are properly brought to the user's attention.

2.2. Use Cases

Our table for the use cases can be found here 1

Use Case Number: 1 (see Figure 1)
Use Case Name: Adding an event to a calendar.

Description: A user has an event they need to record in the calendar. They click the “Add Event” button. The user will then be prompted to enter the name, date and time, and description of the event.
Here is the flow of that process:

1) User navigates to the calendar tab at the top of the program.
2) User left-clicks on the calendar tab to open up the drop-down menu.
3) User left-clicks on the “Add Event” button.
4) Once the user inputs the event name, date and time, and description, the new event will be added to the corresponding date on the calendar.

Termination Outcome: The user now has an event added to the calendar and will be notified of its presence when getting closer to the event deadline.

Alternative: User modifies existing event.

1) User navigates to the ”Calendar” tab and then clicks on “Events”.
2) User right-clicks on the event from the list of events and clicks “Details”.
3) User events are updated with the newly updated event name, date and time, and description given by user input.
4) Once the user inputs the event name, date and time, and description, the new event will be added to the corresponding date on the calendar.

Termination Outcome: The user now has an updated event on their calendar that reflects the user modification in the “Details” sections.

Alternative: User deletes existing event.

1) User navigates to the ”Calendar” tab and then clicks on “Events”.
2) User right-clicks on the event and clicks “Remove”.
3) User is given a prompt asking if the user really wants to remove this event.
4) If the user clicks “Yes”, the event is removed from the calendar. If the user clicks “No”, the event is not removed.

Termination Outcome: The user now has an updated calendar that reflects the user’s removal of the event at the specified date and time.

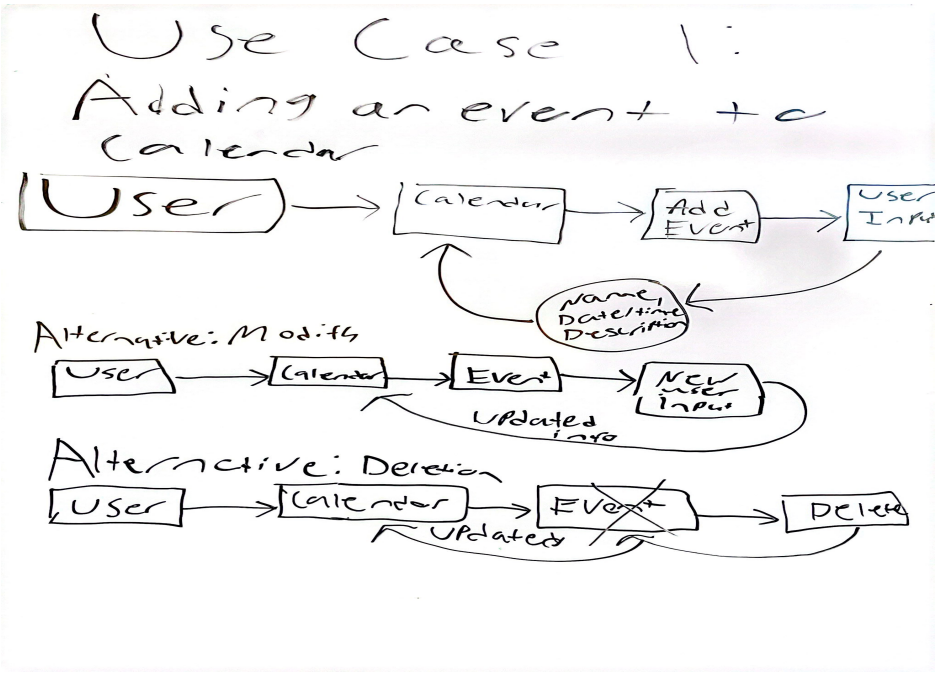

Figure 1. Use case 1

Use Case Number: 2 (see Figure 2)
Use Case Name: Accessing the note-book part of the application

Description: A user needs to record the notes of a team meeting that occurred earlier in the day. They click on the “Create New Quick-Note” button. The user will then be presented with a new blank note that they can write in, save, export, etc. Here is the flow of that process:

1) User navigates to the “Notes” tab at the top of the program.
2) User left-clicks on the “Notes” tab to open up the drop-down menu.
3) User left-clicks on the “Create New Quick-Note” button and begins using the feature.
4) Once finished, the user clicks on “Save” from the options at the top of the quick-note section.

Termination Outcome: The user has saved the quick note. It will now be sorted by the last date modified.

Alternative: The user wishes to export the quick note as a PDF after saving.

1) User navigates to the top of the quick-note screen.
2) Once finished, the user clicks on “Save” from the options at the top of the quick-note section.
3) User then left-clicks on the “Export as PDF” button at the top of the quick-note section.

Termination Outcome: The user now has a saved quick-note version in the application that is sorted by the last date modified. As well as a downloaded PDF version that can be shared offline with others if desired.

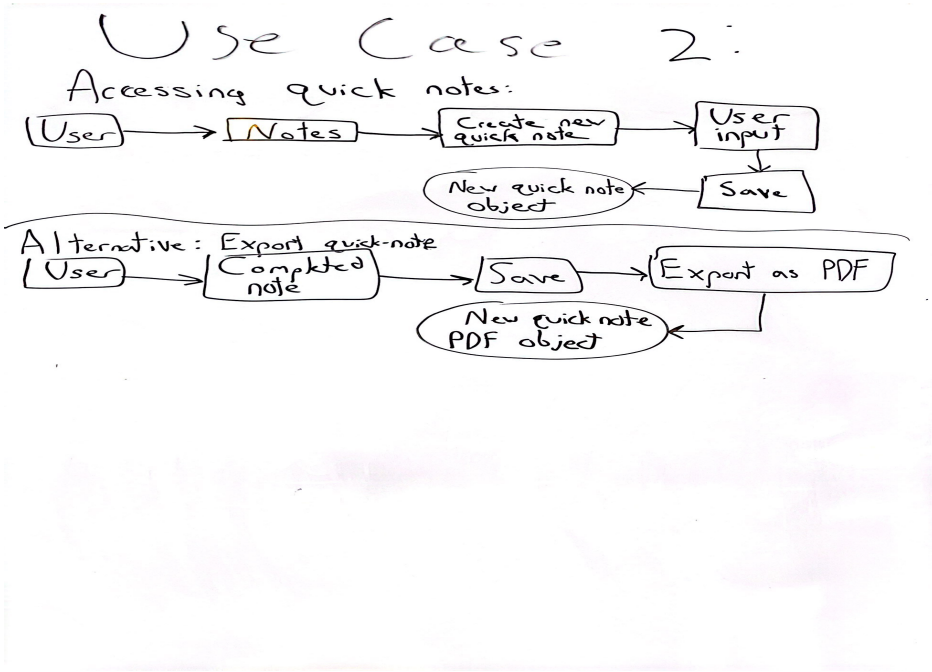

Figure 2. Use case 2

Use Case Number: 3 (see Figure 3)
Use Case Name: Adding an alarm for an event.
Description: A user has an event already registered on the calendar, but the user wishes to be notified closer to the event’s deadline.

1) User navigates to the ”Calendar” tab and then clicks on “Events”.
2) User left-clicks on the event and clicks on “Notifications” then “Add Notification”.
3) User specifies the unit of time (days, weeks, months) and specifies the range of time (ex: 3 days, 2 weeks, 4 months, etc) that the program will need to alert the user of the event before its deadline.

Termination Outcome: The user has set an alert that will notify them corresponding to the time range the user has inputted.

Alternative: User modifies existing notification.

1) User navigates to the ”Calendar” tab and then clicks on “Events”.
2) User left-clicks on the event and clicks on “Notifications” and then right-clicks on the notification they want to modify and clicks on “Notification Details”.

3) User event notification is updated with a newly updated time unit and range.

Termination Outcome: The user now has an updated notification for an event that will alert the user based on their modification of time unit and range.

Alternative: User deletes an existing notification:

1) User navigates to the ”Calendar” tab and then clicks on “Events”.
2) User left-clicks on the event and clicks on “Notifications”.
3) User right-clicks on the desired notification.
4) User left-clicks on “Delete This Notification”.

Termination Outcome: The user has now deleted the desired notification and will not be alerted via an alarm to the original event.

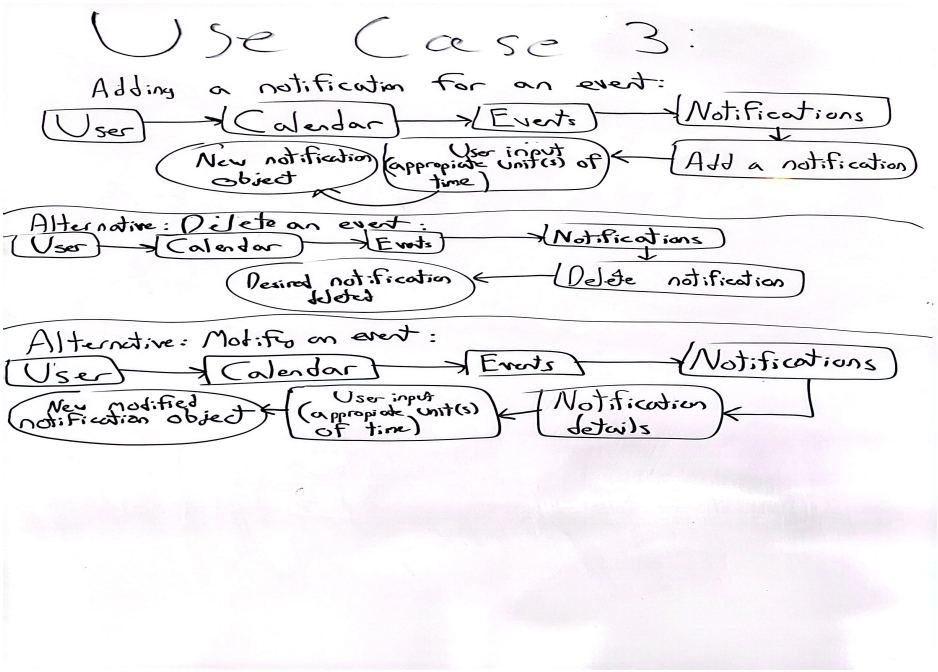

Figure 3. Use case 3

2.3. Interface Mock-ups

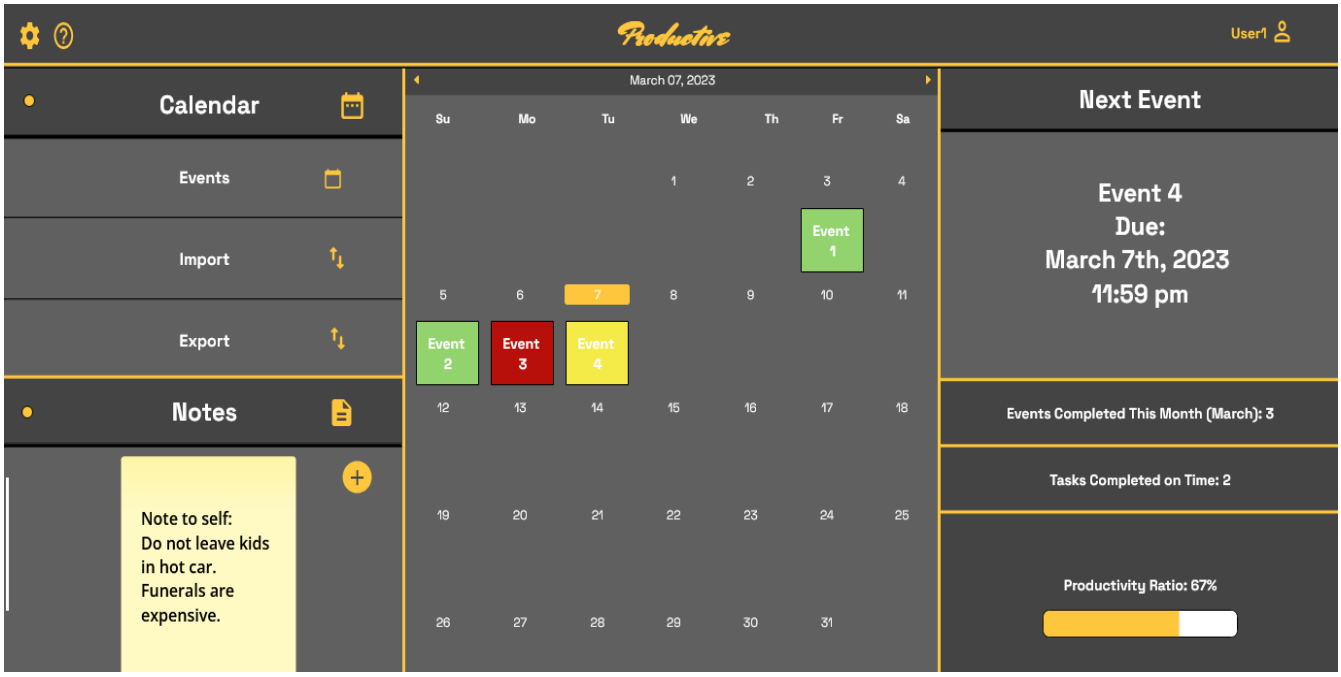

Figure 4. The home-screen/dashboard of our project

Figure 5. The home-screen/dashboard of our project

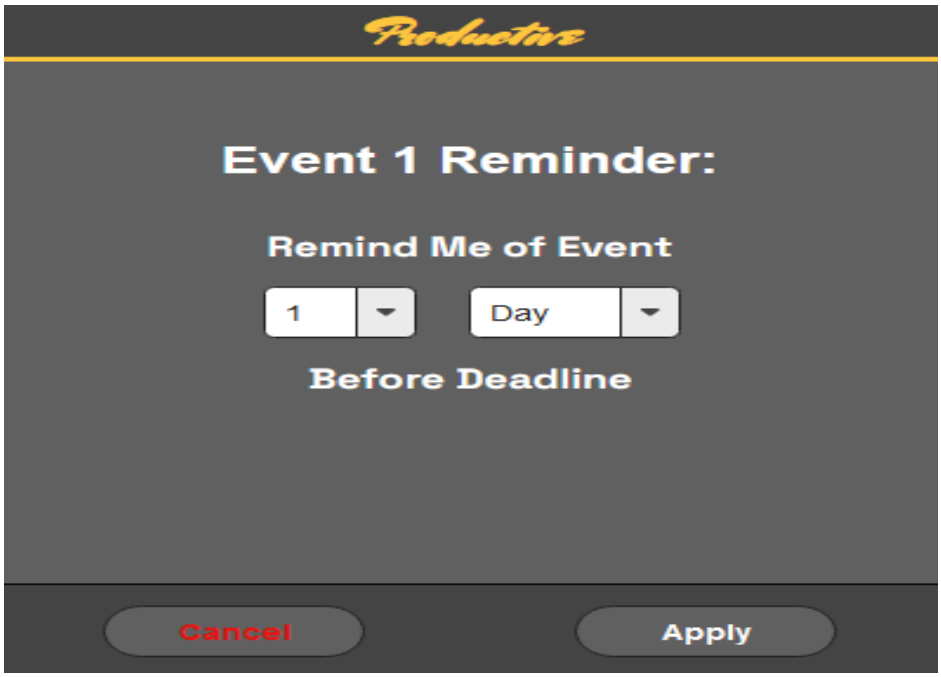

Figure 6. The reminder window of our project

3. Project Timeline

3.1. Current Timeline

- **February 4th** - The idea of implementing a project manager was decided for the assignment. Further consideration of the project's scope, challenges, and stretch goals moving forward were discussed and recorded. The first draft for the project's proposal was submitted.
- **February 10th** - Met and discussed previously stated project scope, challenges, and stretch goals with the instructor. Received criticism and feedback causing a balanced readjustment of the proposal and its characteristics.
- **March 4th** - Project use cases, functional and non-functional requirements, and whiteboard drawings of what the interface would look like were discussed, leading to an update of the project proposal.
- **March 8th** - Detailed mockups of the GUI design were added to the project repo.
- **March 9th** - Project identity communicated as well as interface mockups shown to the class. Received feedback from the instructor.
- **March 30th** - Implementation and design of the actual program has started.
- **April 1st to 5th** - Rough implementation of MainWindow, NextEvent section, Task Overview section, Productivity Ratio section, and Events subwindow.
- **March 30th** - Implementation and design of the actual program has started.
- **April 6th** - Presented a rough demo of the project to the class. Received feedback from the instructor.

3.2. Future Timeline

- **April 10th to 14th** - Update project proposal with timeline, project structure, and UML design. Fix Events subwindow, implement Add Note feature to ScrollViewer, implement Events button to display Events subwindow when selected.
- **April 17th to 21st** - Implement Reminders, Calendar views, and settings. Receive feedback and update the proposal accordingly. Make a draft of a PowerPoint presentation.
- **April 24th to 28th** - Make final implementations for the project. Update the proposal for the final time. Make a final presentation.

4. Project Structure

As what will be shown (See Figure 7) and discussed below, Productive will have many classes as it has quite a few features that require a subwindow to fully realize their implementation as they add new objects within the application. For the design patterns, we have chosen to only implement the minimum of 2 within our program. This section will be updated as we get closer to the final update.

4.1. UML Outline

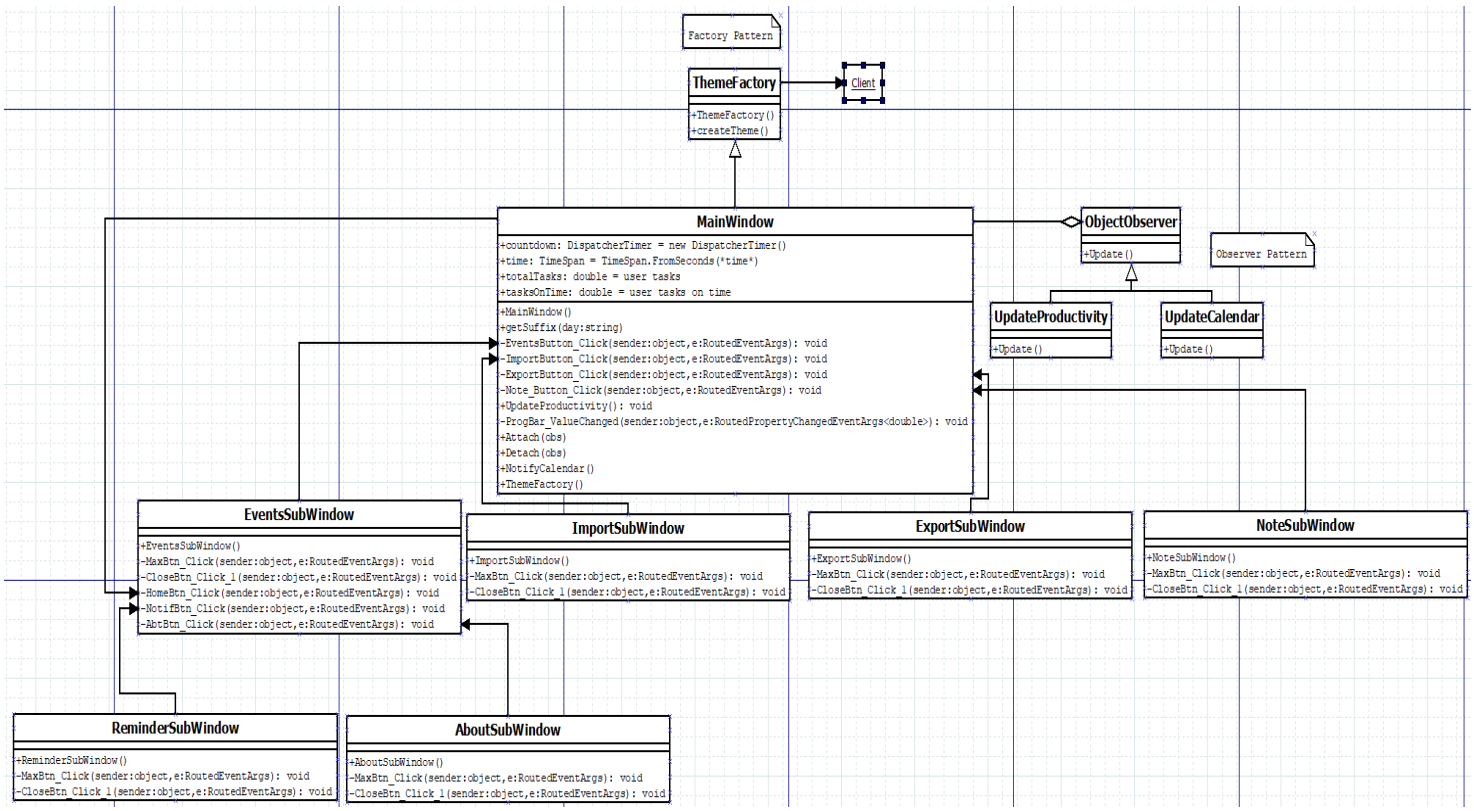

Figure 7. The UML diagram of our project.

As you can see in the UML diagram, the MainWindow is where most of the implementation is going to be. This is because of the project's purpose of being a project manager and how it should communicate information to the user. We asked what is the most important information that a project manager could show that a user would want to see and how we can get that information to the user as quickly as possible. This led to the productive ratio, calendar, notes, and next event sections being all placed in the MainWindow class along with the necessary operations to help during runtime. Many of those operations are for the buttons located on the main window as their purpose is to show a subwindow when clicked. Because of this, MainWindow has many instances of other subwindows as it needs a subwindow object to use the show() method.

The EventsSubWindow class is needed for the add event, modify event, delete event, and even reminder functionality, so it will have instances of ReminderSubWindow and AboutSubWindow(for the About section). The ImportSubWindow class is needed for the ability to import calendar events from other calendar apps to be used in Productive's calendar. The ExportSubWindow class is needed for the ability to export Productive's calendar events to other calendar apps. The NoteSubWindow class is neccessary as it will serve as a way to get user input to generate and add a note to the ScrollViewer on the MainWindow. The ReminderSubWindow class will be used as a window for customizing the time unit and range for when a user needs to be notified about an upcoming event. The AboutSubWindow class will just show information pertaining to the EventsSubWindow instructions of use and what it does.

4.2. Design Patterns Used

The first design pattern we decided to use was the Observer pattern. We want to use this pattern for its ability to observe certain subjects and update object state based on the subject's current state. This type of functionality will be useful in updating Productive's calendar, productivity ratio, and productivity bar to accurately display current information when events are added or deadlines are missed; even if the application is only running in the background. This way, users will always have access to the most recent information and data no matter what they are doing with the application.

The second design pattern we have decided to use in our project is the Factory pattern. We want to use this pattern for its ability to create new objects and pass those objects to the client interface. This will help with creating new instances of certain objects with different properties such as color, which can lead to the creation of a variety of new aesthetic themes for the application. With this, the application can host a variety of different themes to which users can switch back and forth to find their preferred look that they find the most aesthetically pleasing.