Remote Work Discipline App System Design

CS 3704 - Intermediate Software Design - Fall 2024

Raunak Chitre
Computer Science
Virginia Tech
Blacksburg, Virginia, USA
raunakc24@vt.edu

Jeriah Valencia
Computer Science
Virginia Tech
Blacksburg, Virginia, USA
jeriahv@vt.edu

Shane Matthews
Computer Science
Virginia Tech
Blacksburg, Virginia, USA
shanem64@vt.edu

Ben Sullivan
Computer Science
Virginia Tech
Blacksburg, Virginia, USA
bens21@vt.edu

Process Deliverable II - Extreme Programming:

From the previous milestone, it was demonstrated that providing hypothetical non functional requirements for the system requires 10 function points, providing hypothetical functional requirements for the system requires 10 function points, and writing five formal use cases for the system requires 20 function points, and use case and sequence diagrams require 15 function points, for a total of 55 function points. For this milestone, the high-level design tasks would be 10 function points because these only require a written explanation which is fairly easy. The low-level design would be slightly harder and should require 15 points because there is a pseudocode representation and class diagram that accompanies the explanation. The design sketch would be harder because the sketch requires using digital tools, so this task would be a higher number of 20 function points accordingly. The cumulative total of function points would be 55+10+15+20 = 100. For the next milestone, the Black Box Test Plan, I would estimate that this task would be 15 function points because our group is not planning on implementing our project, but the task may be slightly difficult because there needs to be 10 distinct test cases for the table.

High-Level Design:

The architectural pattern best for this system would be the Event-Based Architecture. The time-sensitive nature of the notifications produced by our system lends itself well to the production of events as a reaction to conditions being met. Events must be triggered in real time to deliver proper environment monitoring and timely notifications. Scalability is also a bonus, in the way that the number of prerequisites for an event to be triggered should be able to be changed based on the conditions of the user's work environment and schedule. Event-Based Architecture is commonly used for Mobile applications as well, so this seems like the natural best fit for the Remote Work Discipline App, which is intended to be a mobile app.

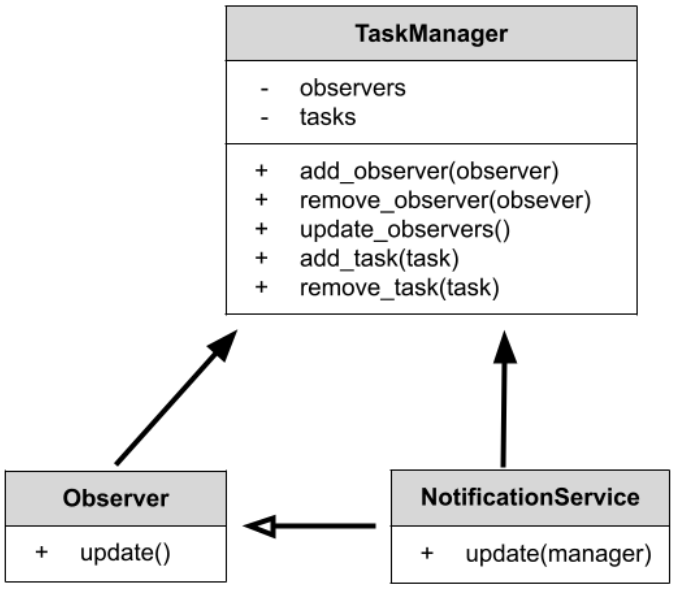
Low-Level Design:

The Behavioral design pattern family works well for the notification system, particularly the Observer pattern, which matches the idea of the app having to observe and communicate whenever changes occur in another part of the system. The Observer, as an entity separate from the task manager and the environment, allows the system to be easier to maintain

```
Class TaskManager:
    def __init__(self):
        self.observers = []
        self.tasks = []
    def add_observer(self, observer):
        self.observers.append(observer)
    def remove_observer(self, observer):
        self.observers.remove(observer)
    def update_observers(self):
        for observer in self.observers:
            observer.update()
    def add_task(self, task):
        self.tasks.append(task)
        update_observers()
    def remove_task(self, task):
        self.tasks.remove(task)
        update_observers()

Class Observer:
    def update(self):
        # Update status of observer

Class NotificationService(Observer):
    def update(self, manager):
        # Send notification logic; based on time, tasks, and
          current environment
```



Design Sketch:

The design of the app’s dashboard must include an overview of the user’s tasks, with options to add, edit, or remove tasks. Current environment status and a rating will be displayed, considering temperature, user’s posture, and other environmental factors/distractions. There will also be a way to access a user profile in the upper right, within the navigation bar, for notification customization settings and other preferences. The top navigation bar will also include tabs to go more in depth into each of these sections of user data. Through the app’s processes, notifications will be sent through the phone’s notification deck similar to any other mobile app notification. Generally, the dashboard provides an overview of the most essential information for the user to keep them organized, while allowing them to manually manage and customize their tasks in such a way that the app can meet the user’s specific needs. Environment monitoring and sending out notifications for breaks helps keep the user maintain a healthy and productive space to work, and being transparent about what the app requires of the user to meet these goals with distinct representations of the complex environmental data helps the user feel more rewarded for good practices. Just as well, displaying all of this information helps meet the requirements of the system we defined earlier in the project, and makes the inner workings a little evident and testable for developers who understand what goes into the data arrangement.