DESIGN

Maya Makked
Computer Science
Virginia Tech
Blacksburg VA U.S.
mayam22@vt.edu

Ethan Bourne
Computer Science
Virginia Tech
Blacksburg VA U.S.
ethanbourne25@vt.edu

Connor Graves
Computer Science
Virginia Tech
Blacksburg VA U.S.
cwgraves@vt.edu

Brandon Hoang
Computer Science
Virginia Tech
Blacksburg VA U.S.
brandonh03@vt.edu

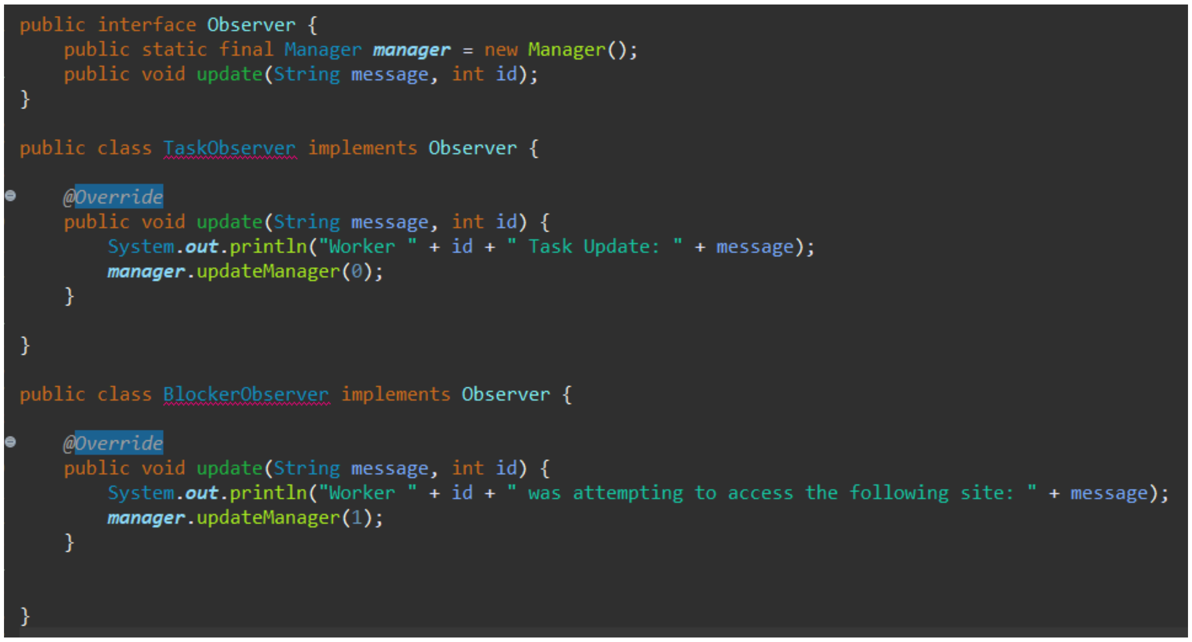
High-level Design

For our system we would use the event-based architectural pattern. This is because an event-based architectural pattern is reactive to real-time events. That is, when an event is triggered, say, an input or change in state, then we take an action. This works perfectly for TaskMaster, as most of the time is spent idle, waiting for users to interact with the system or for internal timers to trigger. TaskMaster will only give prompts when something happens.

Low-level Design

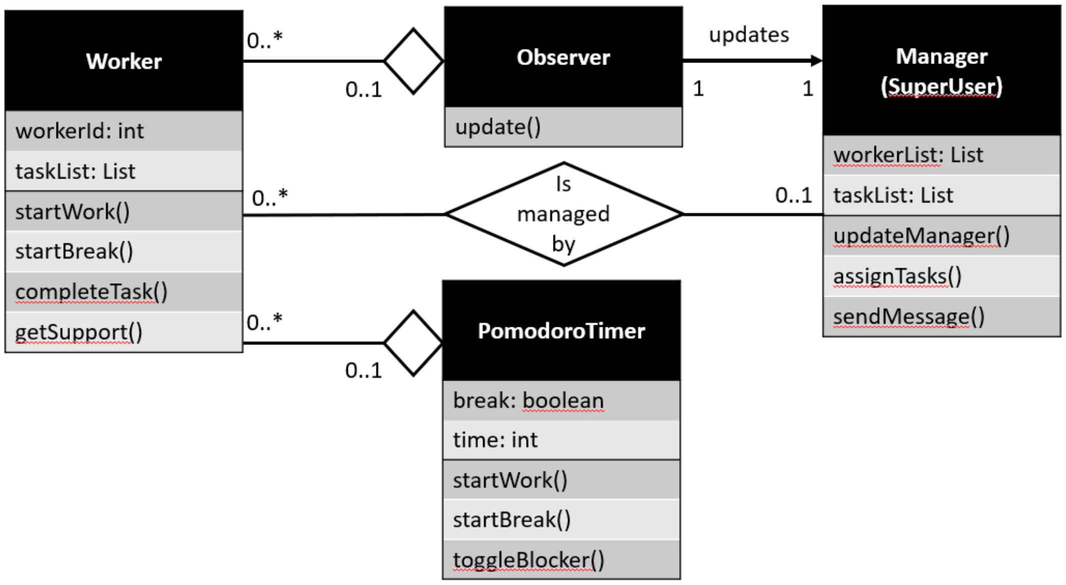
A design pattern family that would be helpful for our system would be behavioral. One design pattern from the behavioral family that would be useful is an observer. An observer is useful for the employees actions, whether they request a break, finish their tasks, or try to not work during work time. These examples of employee actions would ideally notify the manager and prompt the manager to interact with the system.

Pseudocode:

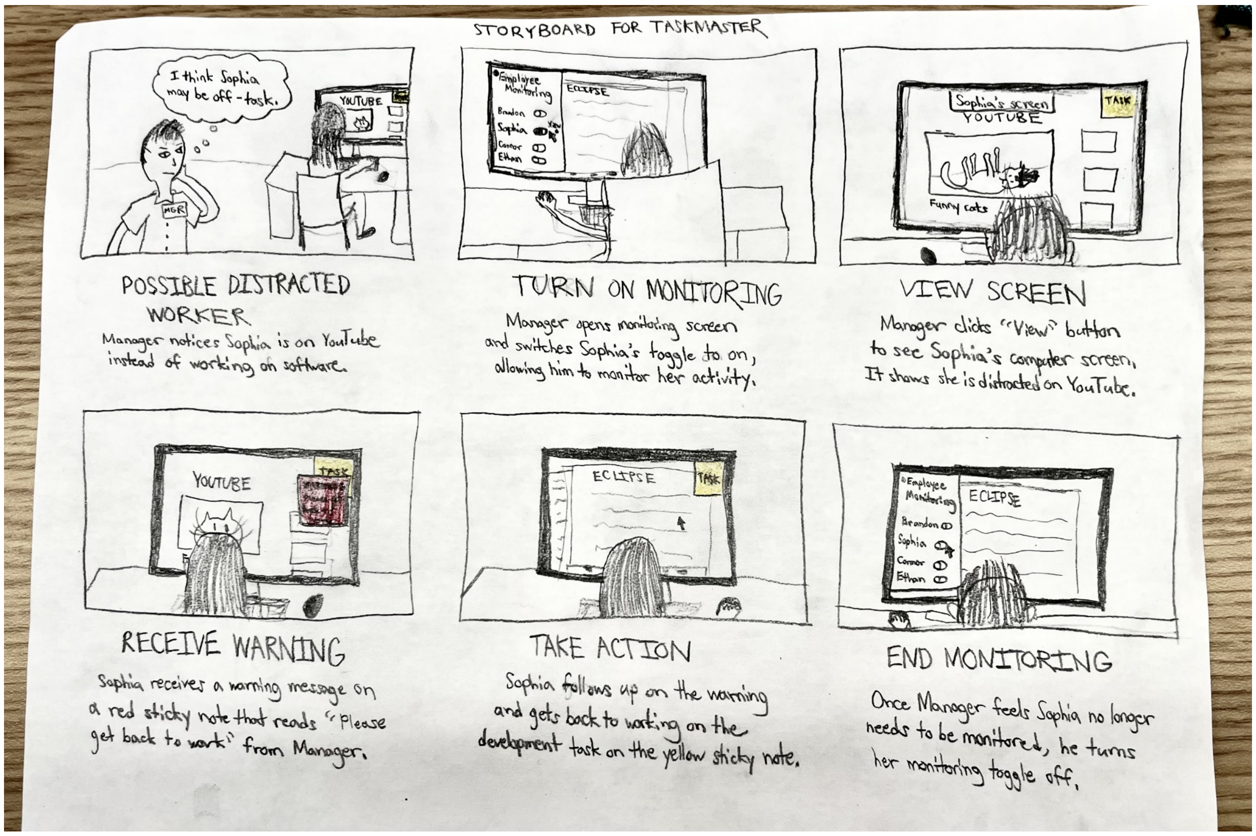


```
public interface Observer {
    public static final Manager manager = new Manager();
    public void update(String message, int id);
}

public class TaskObserver implements Observer {

    @Override
    public void update(String message, int id) {
        System.out.println("Worker " + id + " Task Update: " + message);
        manager.updateManager(0);
    }

}

public class BlockerObserver implements Observer {

    @Override
    public void update(String message, int id) {
        System.out.println("Worker " + id + " was attempting to access the following site: " + message);
        manager.updateManager(1);
    }


}
```

Class Diagram:



Design Sketch



STORYBOARD FOR TASKMASTER

POSSIBLE DISTRACTED WORKER
Manager notices Sophia is on YouTube instead of working on software.

TURN ON MONITORING
Manager opens monitoring screen and switches Sophia's toggle to on, allowing him to monitor her activity.

VIEW SCREEN
Manager clicks "View" button to see Sophia's computer screen. It shows she is distracted on YouTube.

RECEIVE WARNING
Sophia receives a warning message on a red sticky note that reads "Please get back to work" from Manager.

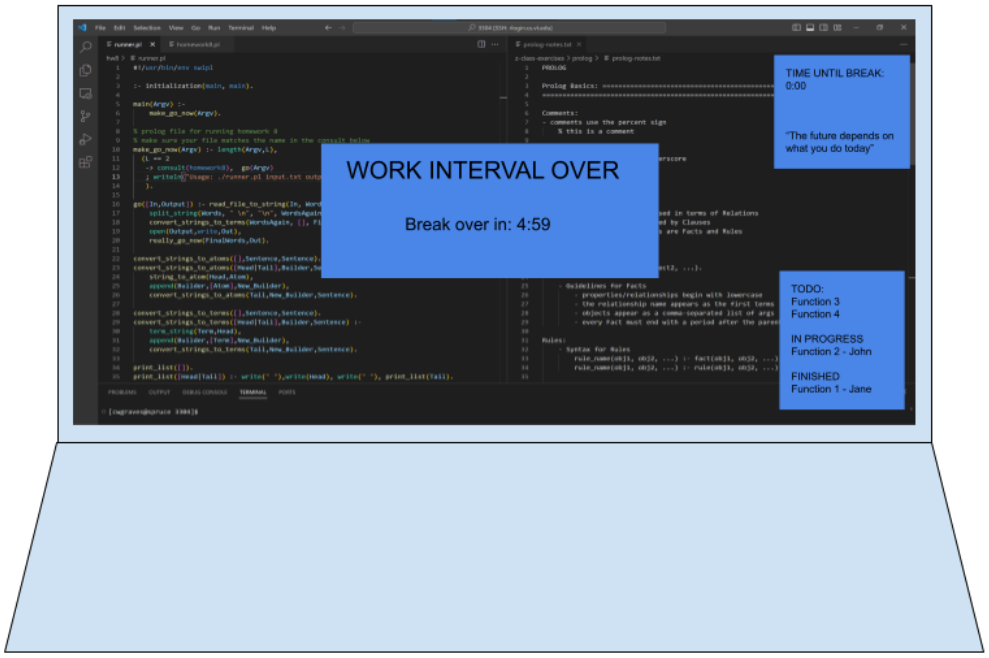
TAKE ACTION
Sophia follows up on the warning and gets back to working on the development task on the yellow sticky note.

END MONITORING
Once Manager feels Sophia no longer needs to be monitored, he turns her monitoring toggle off.

With TaskMaster, the task and warning sticky notes stay prolonged on the screen so that they can be emphasized to the user for extended periods of time. This way, the user does not need to click between screens to view the messages, which can be distracting. The manager can easily click a constantly-on-screen button to expand the employee monitoring screen, allowing them quicker access to the monitoring toggles. This places more of a feeling of control in the manager. Another reason TaskMaster is user friendly is because the button choice on the employee monitoring screen is intuitive. Toggles indicate that something is either on or off, which in this case is the monitoring for a certain listed employee. The labels, such as "Employee Monitoring" and "Sophia's screen," provide guidance to the user. TaskMaster displays consistency through the sticky note display style of different types of messages.

Prototype

Prototype showing the Pomodoro breaking display for a worker, in addition to a Kanban board for the team's progression across all current assignments.


TIME UNTIL BREAK:
0:00
"The future depends on what you do today"
WORK INTERVAL OVER
Break over in: 4:59
TODO:
Function 3
Function 4
IN PROGRESS
Function 2 - John
FINISHED
Function 1 - Jane