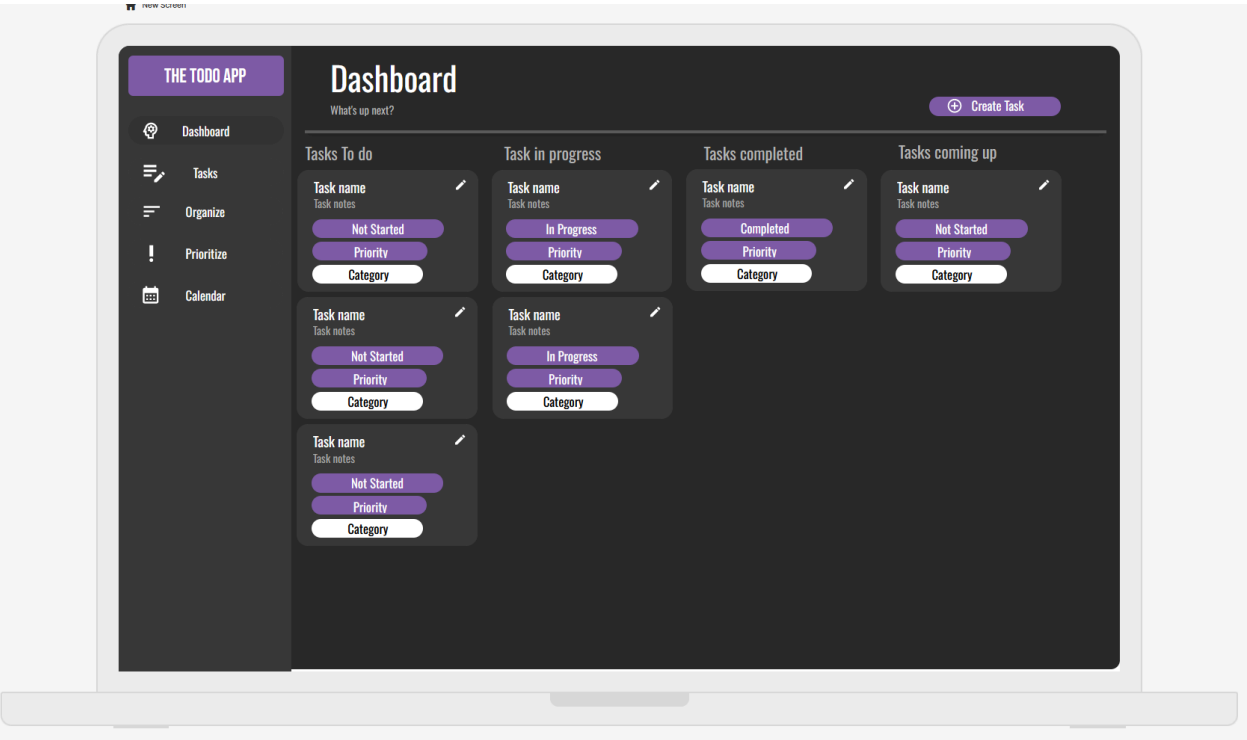
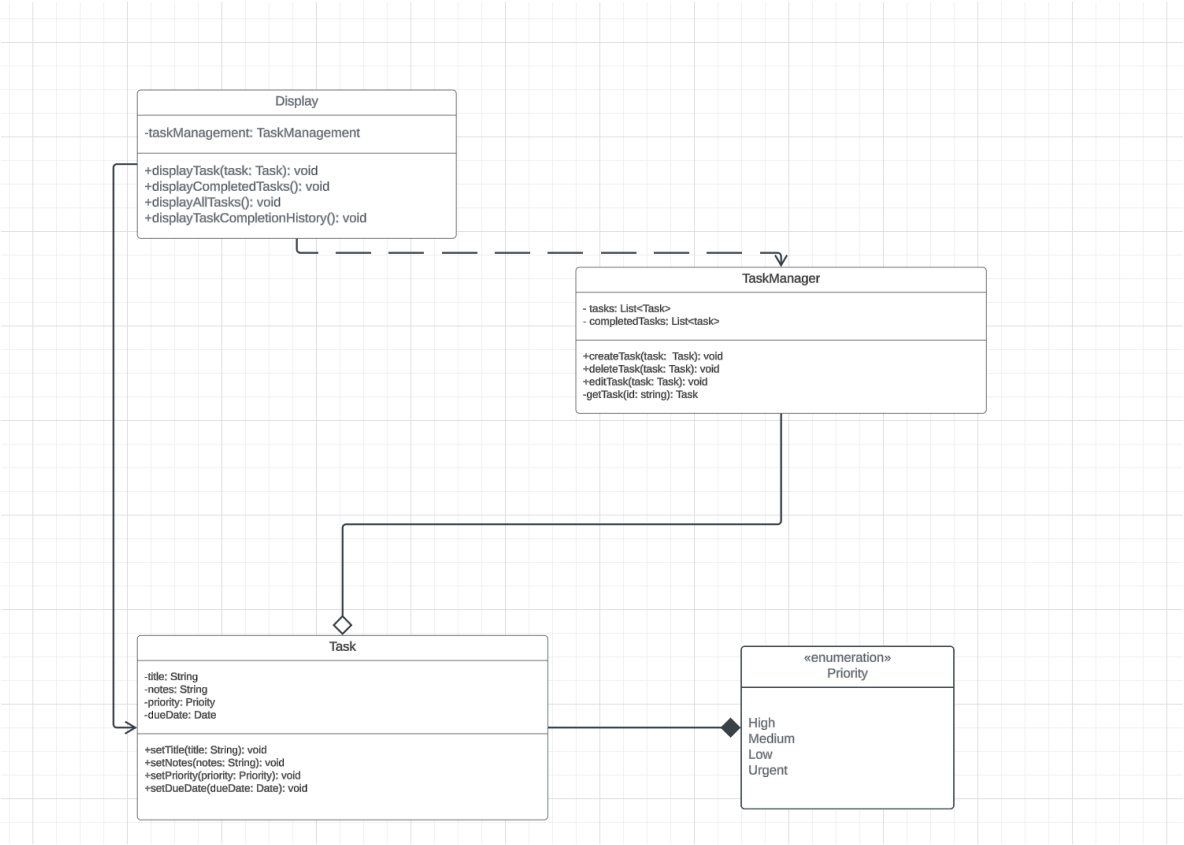
High Level Design:
We would choose to use an Event-Based architectural pattern for our TODO app. By using an event-based architectural pattern we will have support for flexibility and this type of pattern will be very useful for us as we are using a prototyping SE model where we will need to make changes on the fly and have multiple working iterations of our product. In a TODO app user actions can lead to asynchronous operations, such as notifications, reminders, and integration with external calendars or services. An Event-based architecture enables these features to be developed, tested, and refined independently from the core application logic. By having these as independent components this will reduce the complexity and speed up development.

Process Deliverable:





Low-level Design:

For our TO-DO app the best design family would be the structural patterns. Our code will be composed of different classes that will delegate work between them to provide functionality for the app. We will have a class that manages the creation, deletion, and modification of tasks in the To-Do app. As well as have a class that represents the task itself. This class will also work with a class that provides the GUI for the app.

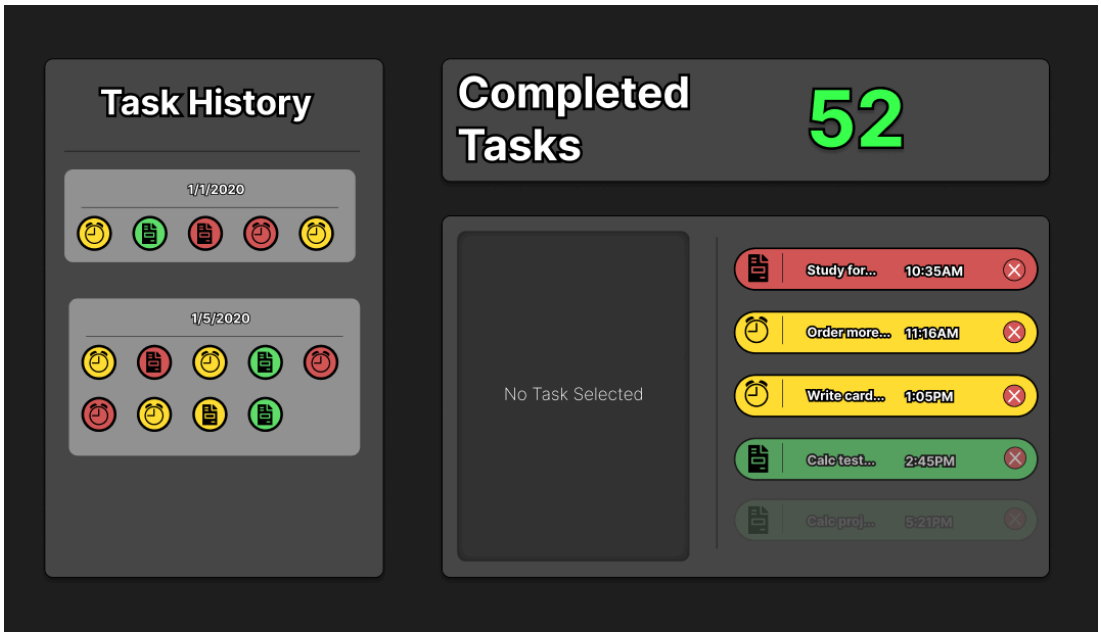
Here is some code represents the creation of a task:

```
Task hw = new Task();
hw.setTitle(“Home Work”);
hw.setPriority(High);
hw.setNotes(“Complete this homework assignment for calculus”);
hw.setDate(“4/10/24”);
```

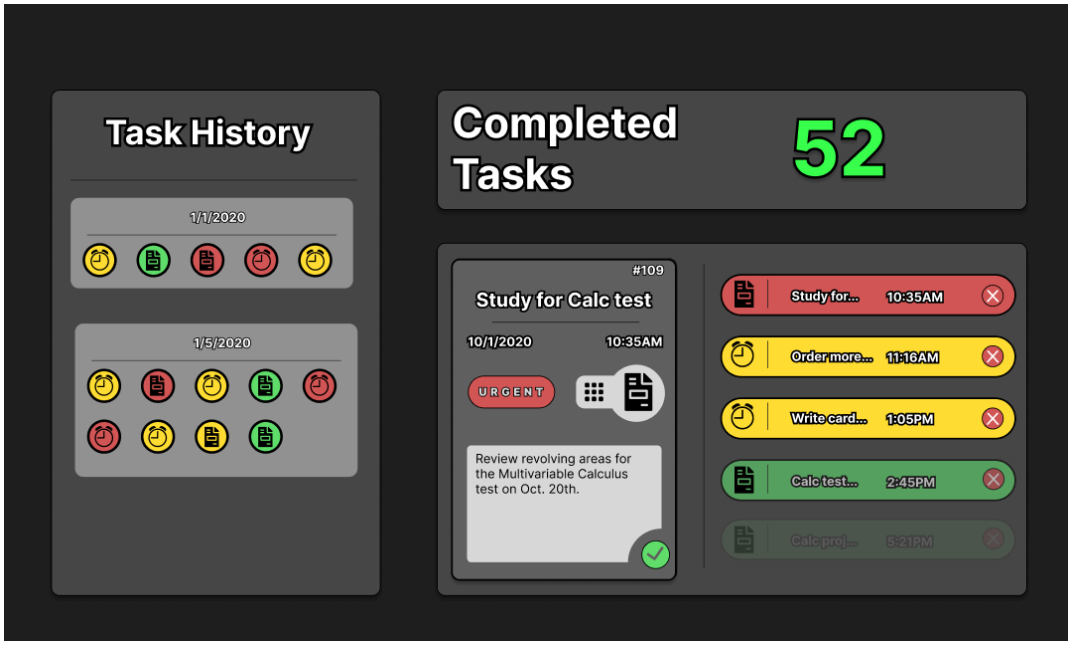
Design Sketch

Selecting a Task to Display

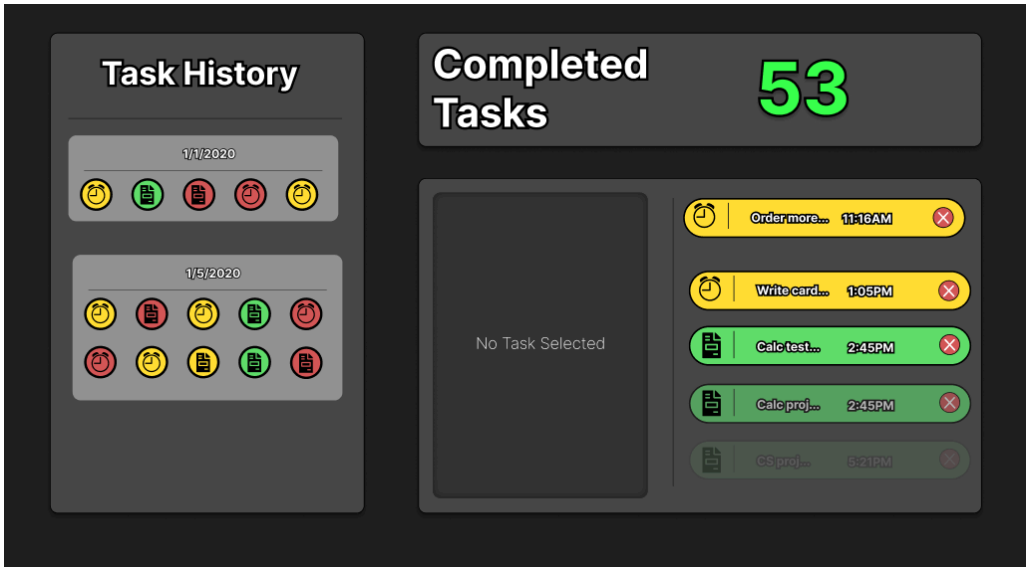
(No Task Selected)



Task Selected from queue on the bottom right



(User clicks on the green checkmark and the selected task is completed)



Design Explanation

The goal of this app is to both serve as a to-do app and motivate users to complete their tasks. The app's design accomplishes this in two ways, it gamifies tasks and it gives users a sense of progression. The gamification of tasks is mostly visible in the card system that displays tasks. Users can select tasks from their queue (list of tasks on the bottom right) which will display a task card for the selected task. When they are finished with the task, they can click on the green checkmark, and the card will be stashed into their task history (grid of task icons on the right). The “Completed Tasks” header serves both to gamify the tasks and demonstrate to users their growth to keep them engaged and motivated to complete tasks. Tasks also have a limited visual language. There are two visual components associated with each task to provide extra information. The first visual component, the icon, is set by users so that they can determine the type of task at a glance. In the demonstration above, the user used the paper icon for school-related tasks and the clock icon for miscellaneous tasks. The second visual component, color, is used to communicate the urgency of a task. Green is used for non-urgent tasks, yellow for mild urgency, and red for extremely urgent tasks. The user’s task queue (list on the bottom right) will sort tasks based on urgency. Currently, one flaw with the design is the color system which utilizes red and green as opposites in urgency. Users with red-green color blindness will not be able to benefit from the app’s visual language for urgency. We will have to do further research to find colors with similar meanings to red and green.