High-level Design (4%)

Describe which architectural pattern you would use to structure the system. Justify your answer.

We will use the event-based architecture to structure our system. Since this architecture promotes the production, detection, consumption of, and reaction to events, we think this would be a great structure for our task management system. This is because the event would be a creation of a task, and this architecture allows for deletion of them, adding reminders to them, adding journal entries, and more reactions.

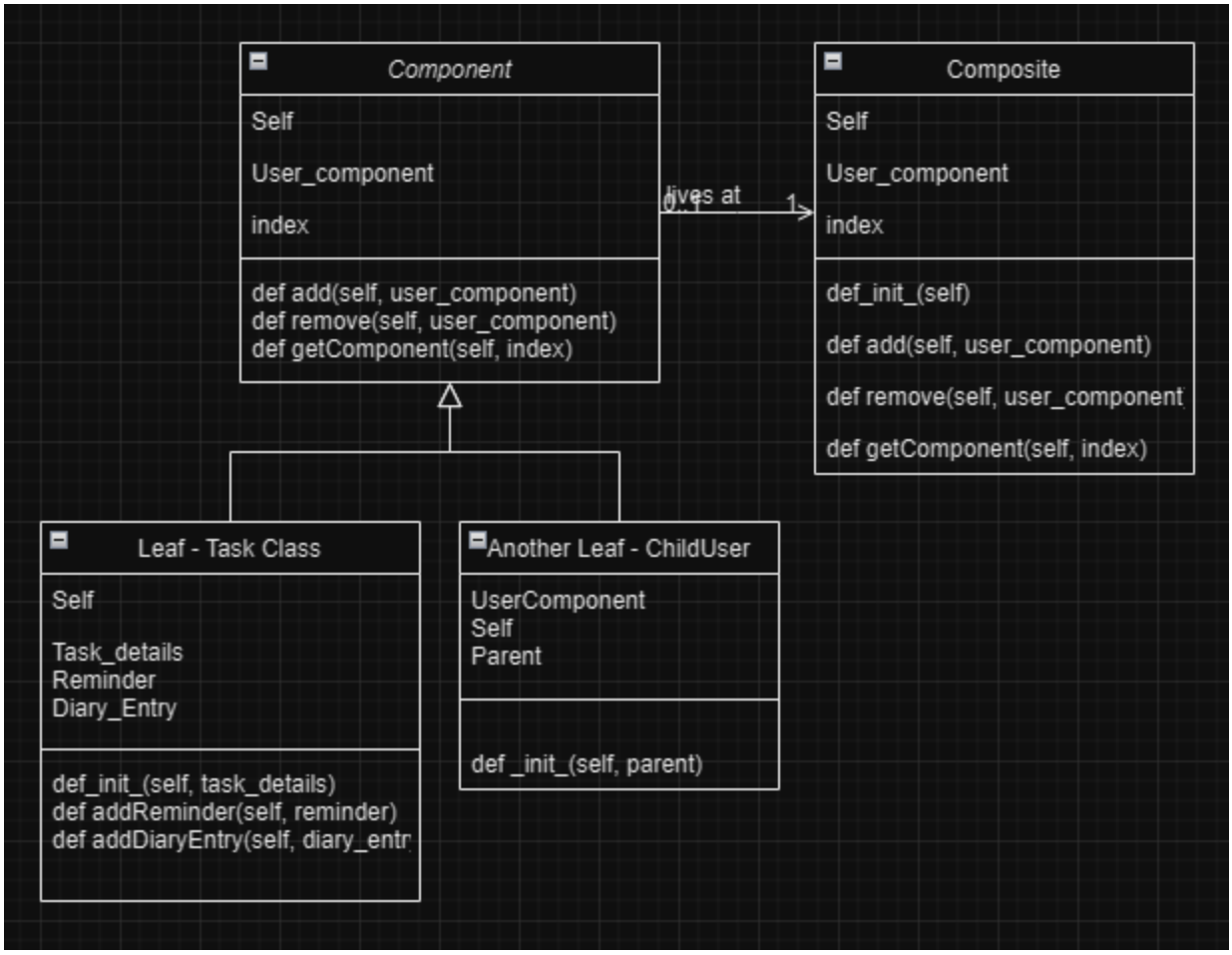
Low-level Design (4%)

Discuss which design pattern family might be helpful for implementing this project. Justify your answer, providing a code or pseudocode representation *and* an informal class diagram.

```
# Component

class :

    def add(self, user_component):

        pass


    def remove(self, user_component):

        pass


    def getComponent(self, index):

        pass
```

```
# Composite
class User(UserComponent):
    def __init__(self):
        self.components = []

    def add(self, user_component):
        self.components.append(user_component)

    def remove(self, user_component):
        self.components.remove(user_component)

    def getComponent(self, index):
        return self.components[index]

# Leaf
class Task(UserComponent):
    def __init__(self, task_details):
        self.task_details = task_details
        # Tasks can have reminders, diary entries, etc.
        self.reminders = []
        self.diary_entries = []

    def addReminder(self, reminder):
```

```
        self.reminders.append(reminder)

    def addDiaryEntry(self, diary_entry):
        self.diary_entries.append(diary_entry)

# Another Leaf
class ChildUser(UserComponent):
    def __init__(self, parent):
        self.parent = parent

# Client code
parent_user = User()
child_user = ChildUser(parent_user)
parent_user.add(child_user)

task = Task("Homework")
task.addReminder("Reminder: Do your math homework")
task.addDiaryEntry("Diary Entry: Started homework at 7 PM")

child_user.add(task)

# This structure allows us to treat both User and Task objects uniformly.
# A child may have a task that is linked to a parent user's account.
```



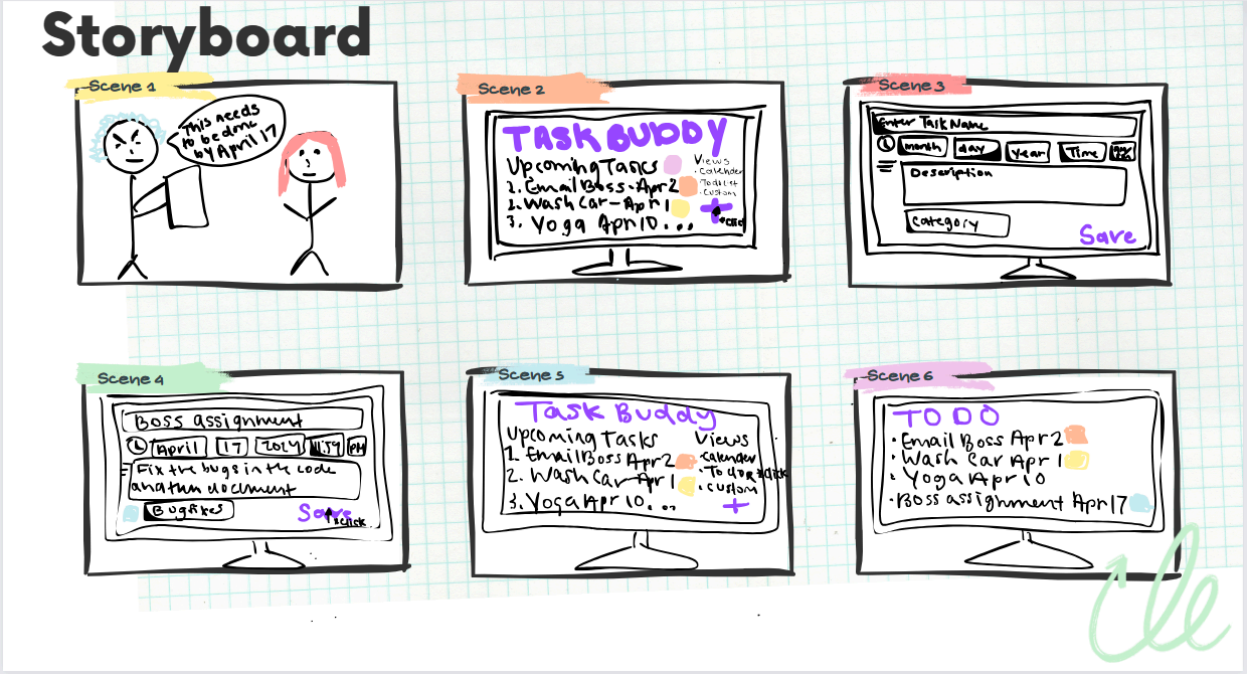
We think that the structural design pattern family would be helpful for implementing this project. The reason for this is that it will allow our team to organize all of our classes to form a large app since there are so many complex components. In addition, if we choose to write a new functionality in the future, the structural design pattern will easily allow us to integrate the new functionality into the existing program.

Design Sketch (4%)

Design sketches provide a visual overview of the look and feel of your project. This may include but is not limited to one of the following:

- **Creating a wireframe mockup of your project user interface in action.**
- **Creating a storyboard that illustrates a primary task that a user would complete with your project.**

Provide a brief rationale (no more than 1 paragraph) explaining some of the design decisions for your program based on the provided sketch.



We want to create an application that organizes different types of tasks. Anything from yoga to work assignments can all be seen in the same list. However, these different types of tasks can also be separated by category because sometimes you only want to see tasks of the same type instead of all of your tasks together. The storyboard shows how quickly a task can be added to the user's to-do list. Depending on what format the user prefers, they can look at their tasks in a to-do list, calendar, or custom style. The upcoming tasks on the home screen help the user see anything that is coming up soon without having to click on a specific view if they just need to check something really quickly.

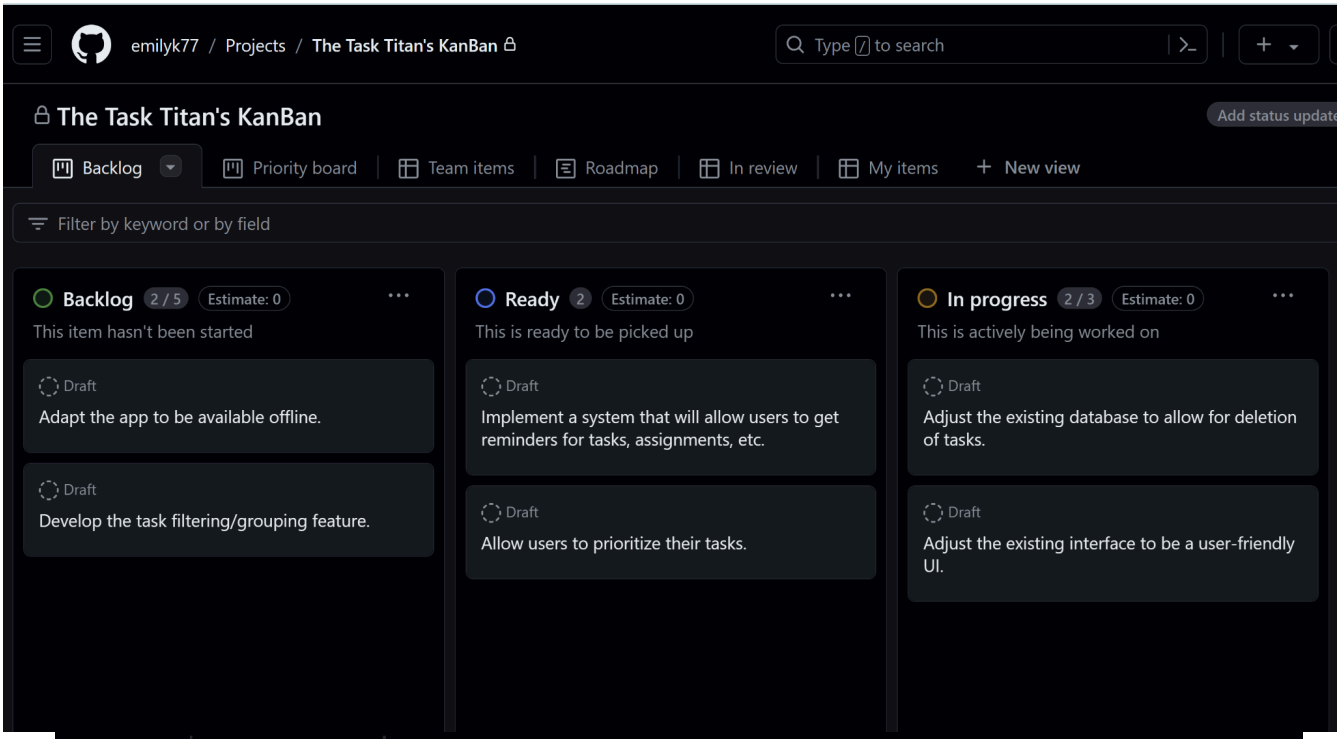
Project Check-In

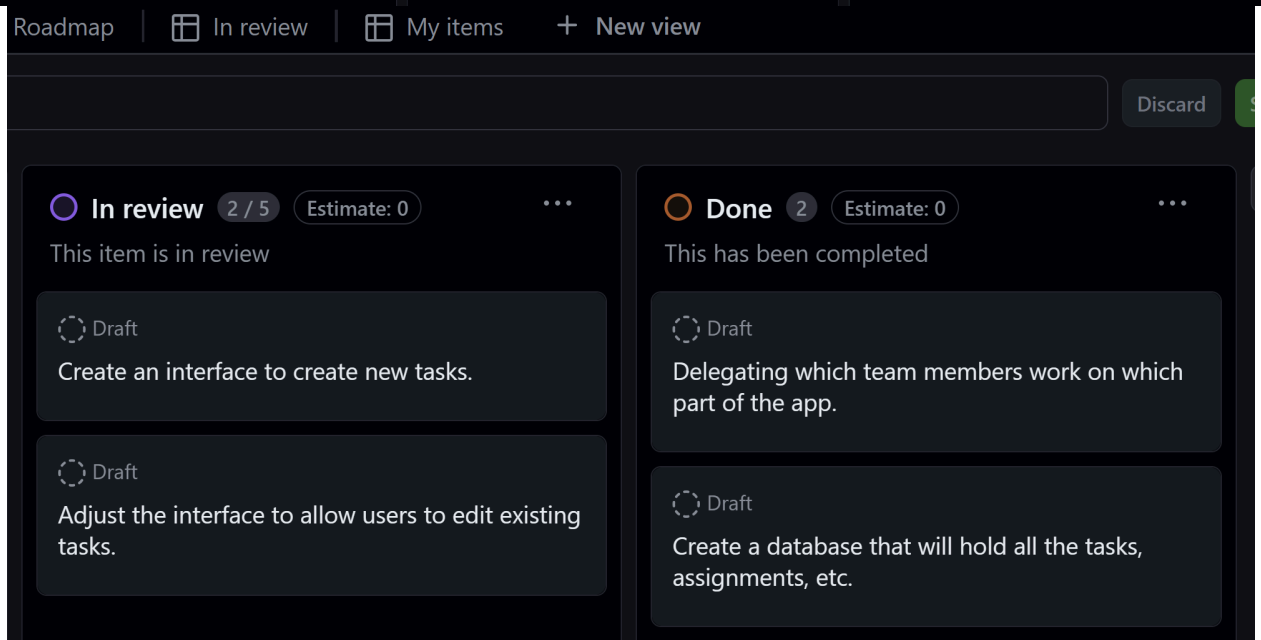
Complete this survey to provide an update on your team progress on the project for this semester. Only one team member needs to complete this for the group.

Process Deliverable (3%)

The submission for this deliverable will depend on the specific SE process model your team plans to use to complete the group project (as described in your project proposal). Example submissions for different processes include:

- **Prototyping: submit a prototype of your system**
- **Scrum: submit the notes (including each teammate) from your most recent scrum meeting**
- **Kanban: submit a list of prioritized tasks from your task management system**
- **Waterfall: submit supplementary planning documentation**
- **Extreme programming: submit acceptance test criteria**
- **Spiral: submit risk analysis**





- **... For other processes not listed above, the instructor will contact you with the exact submission requirements for this task.**