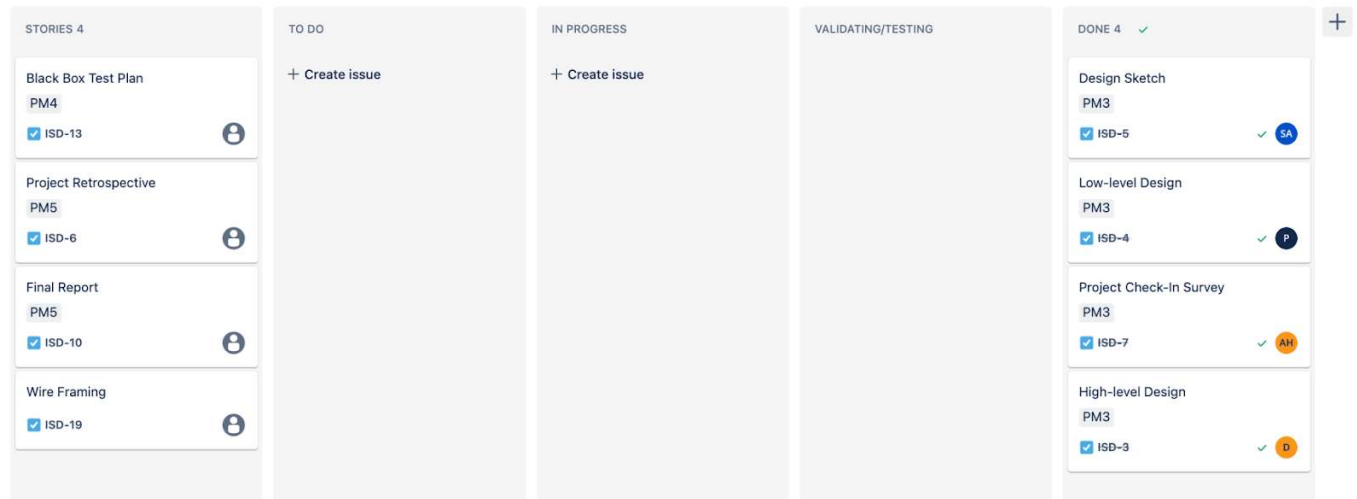


Names: Domenic Martin, Sam Austin, Ayda Haydarpour, and Peyton Ludwig

Kanban Board

Process Deliverable II

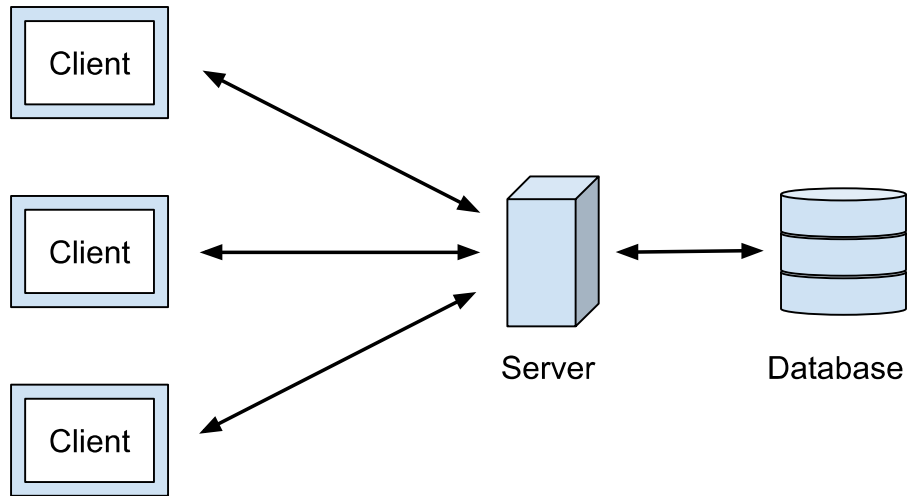


High-Level Design

We will use the client-server architecture pattern for our application. Our client side will contain the UIs for the Project Managers and Developers. The server side will store all of the data needed to run the application and generate the UI displays. When a client adds a task through their UI it will be sent to the server which will store the task and automatically assign a user to complete that task. When a task's data or status is updated in a user's UI it will be sent to the server and stored.

Justification: We choose this architecture pattern because it easily allows for collaboration between users. Using this pattern it will be easy to synchronize the data of each user's local kanban board containing their tasks as well as the global kanban board (contains all tasks). Any tasks added to the global board can also be easily assigned by the server which will have access to a database of user information and profiles.

Diagram:



Low-Level Design

The focus for this section will be the specific subtask of task assignment, both automatic and manual from the project manager. The **behavioral** design pattern family would be the most well suited for this implementation. This is because the behavioral design pattern family characterizes the interactions between various classes and objects, including distributing responsibility. Assigning tasks requires much interaction between objects, such as interactions between classes for tasks, employees, and task assigner logic.

Pseudocode for this subtask:

Employee

```
public void addTask()
{
    activeTasks += 1
}

public void removeTask()
{
    activeTasks -= 1
}

boolean isAvailable()
{
    return availability
}

public boolean hasSkill(String skill)
{
    return if skill in skillSet
}
```

Task

```
public void setAssignedEmployee (employee Employee)
{
    assignedEmployee = employee
    status = "assigned"
}

public void markComplete()
{
    status = "completed"
}
```

```

        assignedEmployee = null
    }

    public void updatePriority(int updatedPriority)
    {
        priority = updatedPriority
    }

```

WorkloadAssignmentStrategy implements AssignmentStrategy

```

    public Employee assignTask(task, employees)
    {
        define available employees
        sort by least workload
        return employee with least workload
    }

```

SkillAssignmentStrategy implements AssignmentStrategy

```

    public Employee assignTask(task, employees)
    {
        define available employees
        sort by skill set
        compare workloads
        return employee
    }

```

TaskAssigner

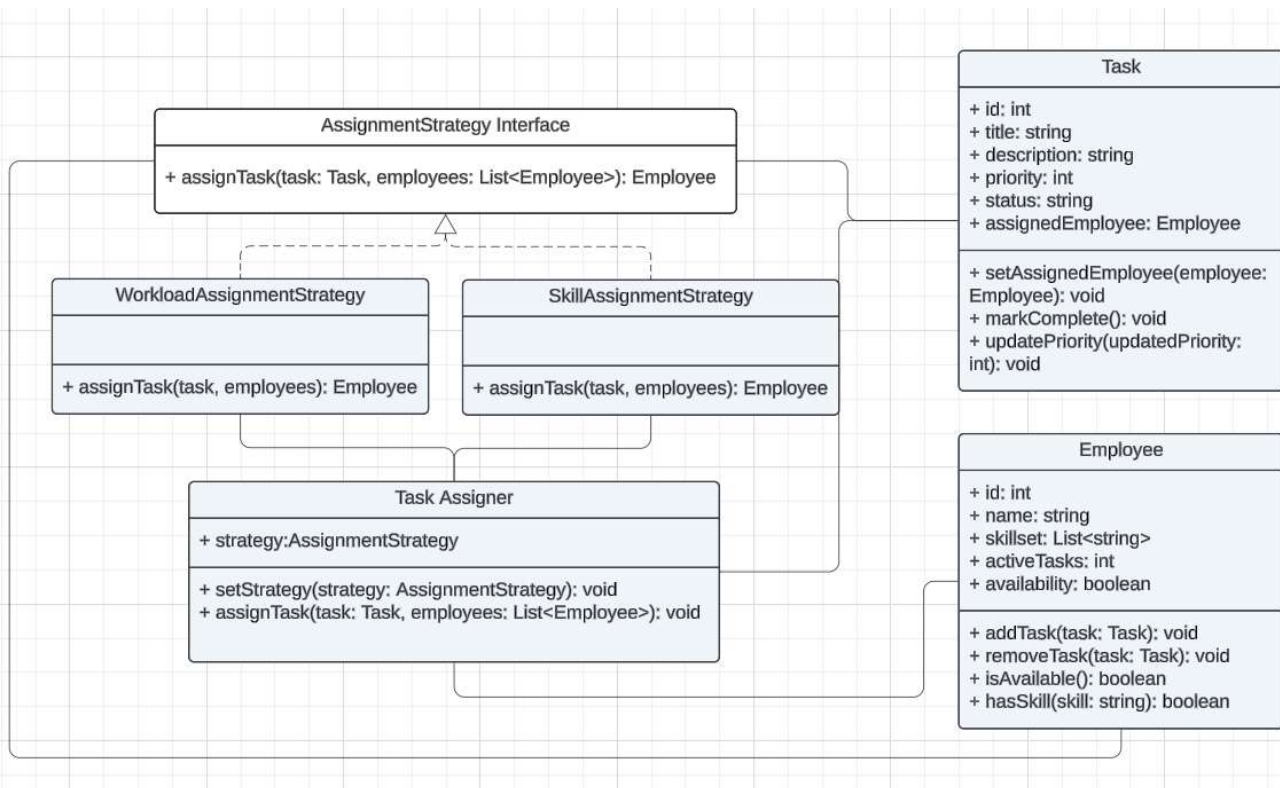
```

    public void setStrategy(strat: AssignmentStrategy)
    {
        strategy = strat;
    }

    public void assignTask(Task task, List<Employee> employees)
    {
        Check if strategy is null
        Call assignTask on Strategy object
        If employee is not null, update the assignedEmployee in the
        task object and the workload in the Employee object
    }

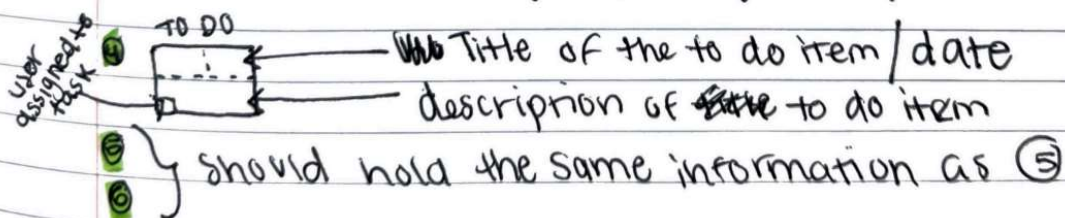
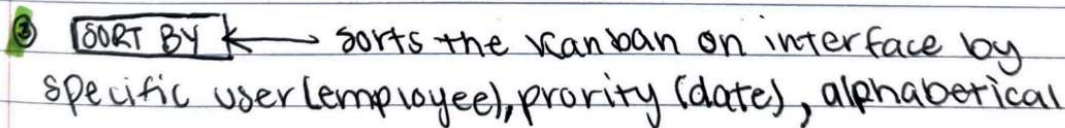
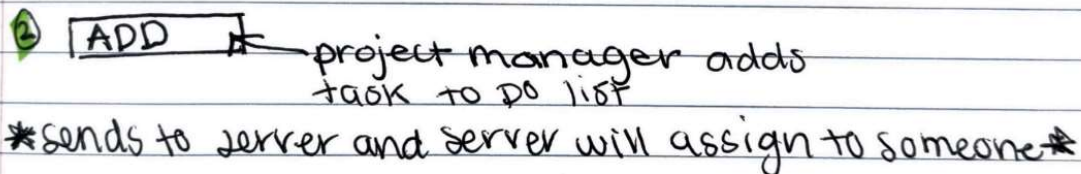
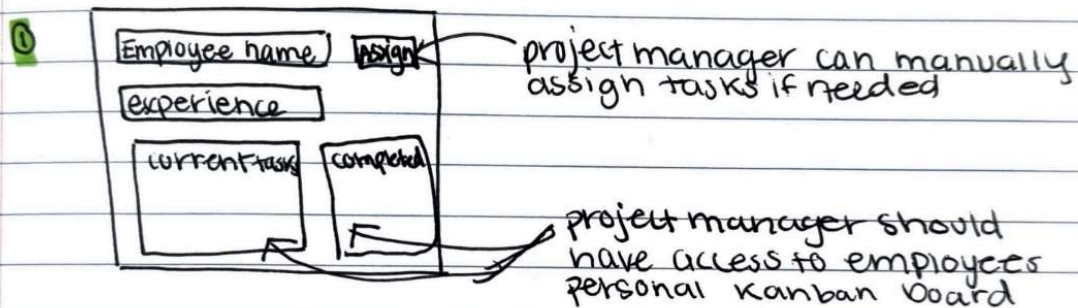
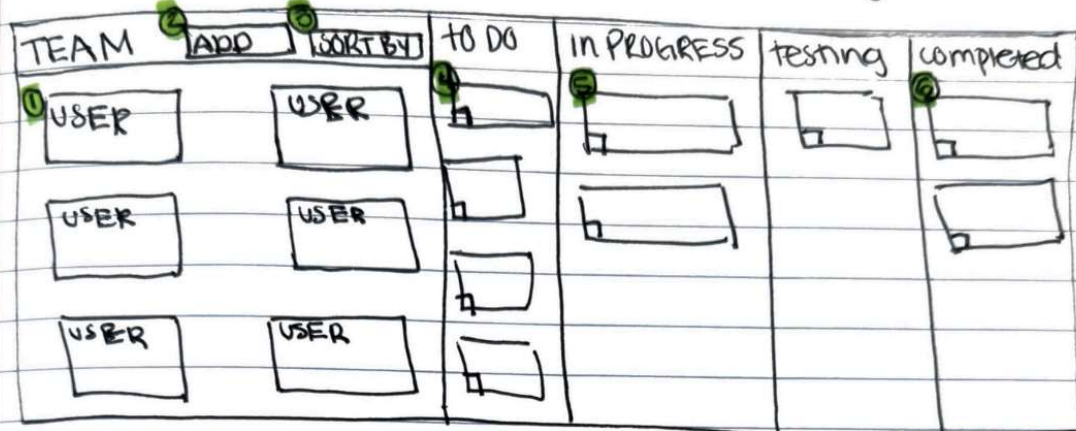
```

Class diagram for this subtask:

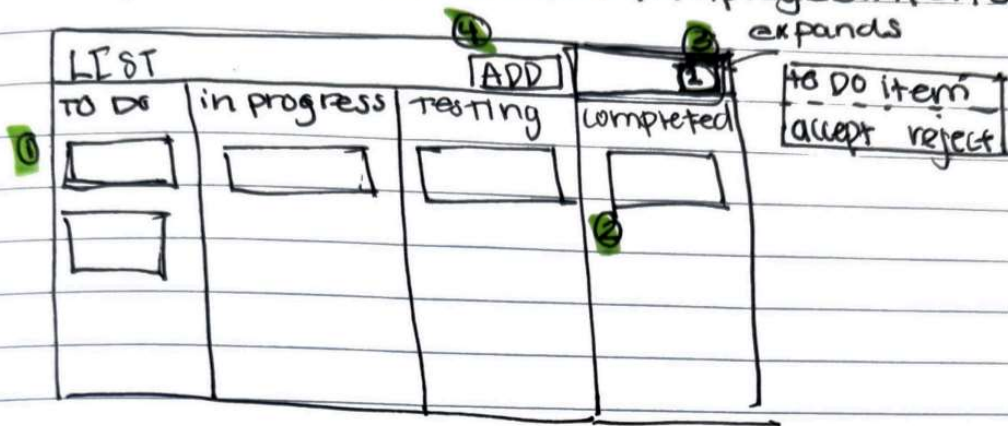


Design Sketch

Global Kanban Board (Project Manager Interface)



local ~~employee~~ KanBan Board (employee interface)



① title of to do item / date complete
description plus notes by

② still be same as ① but also include complete date

③ Notification of new items added
employee can accept or reject

④ ~~add more to personal list~~
~~disabled~~ employee can add a task
that needs to be added to project to do
list.