PM3

Team Name: **Taskboard Manager**
Nishesh Saikrishna (nisheshsai9@vt.edu)
Peter Murphy (pmurphy26@vt.edu)
Benjamin Hurt (benjaminhurt@vt.edu)
Joel Buba (joelb1@vt.edu)

High Level Design

We would use an event based architecture pattern for our system. This is because we have different events that we allow our users to interact with. We can interact with the taskboard by triggering different events with the buttons found across the taskboard. These events include adding and removing users, adding different groups working on different tasks together to the center panel, creating a task, and more. Even the act of creating a task can be broken down into different events, such as assigning the task name, assigning different users to the task, and adding any notes to the task. Since the act of creating, selecting, or editing a task would be treated as an event we would be able to easily produce new events, which is the main purpose of the taskboard. The option of easily adding new features will help our project, as we may have new options available for users to interact with tasks.
In this program, the GUI acts as the dispatcher which receives the user actions and other events, and then calls the appropriate functions to accordingly update itself and the taskboard object. In addition to our implementation design easily following the event based architecture design, there is no need for any of our components to have control over the order of execution so we do not have any drawbacks to consider when following the event based architecture pattern.

Low Level Design

We plan on following a Universal Design pattern for our taskboard. This is because we don't have a specific kind of user in mind and we want to allow the taskboard to be used for any type of project, specifically computer science ones though. This can include a university project where students are working in teams or in the workplace. In either case, team members can collaborate, evenly divide their tasks, and set deadlines for their tasks.

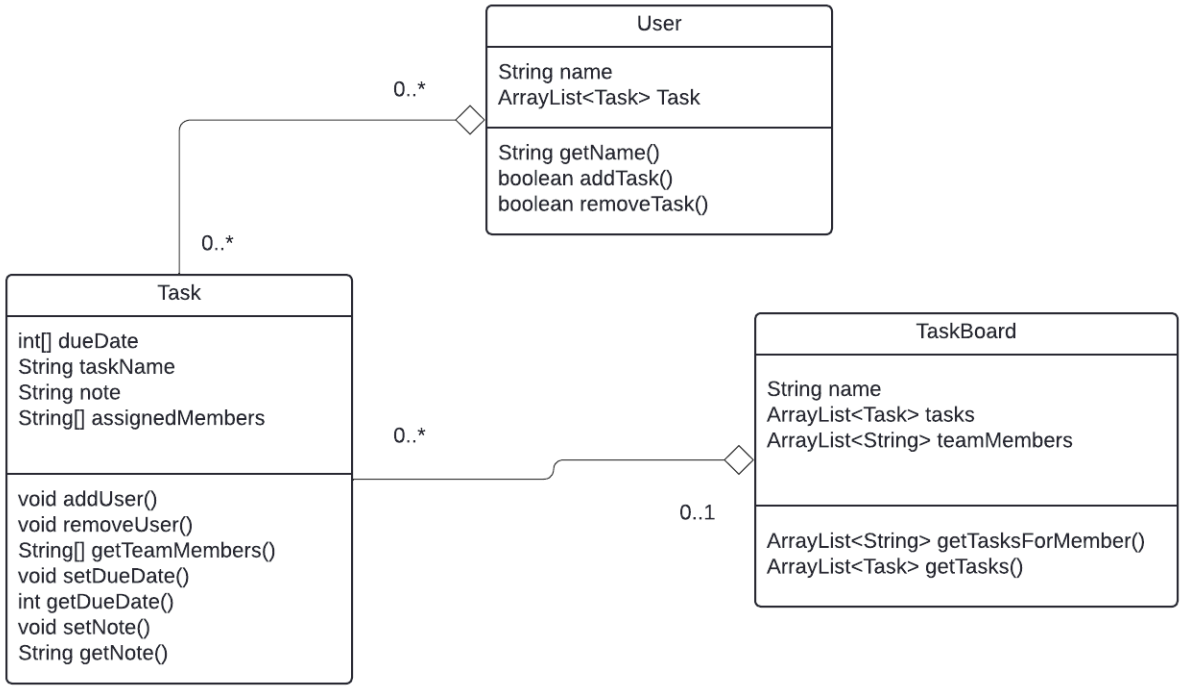
Here is some code for the Task class (Rest of the code available at : https://github.com/pmurphy26/Taskboard)

```
public class Task {
   private int[] dueDate;
   private String taskName;
```

```
private String note;
private TaskStatus status = TaskStatus.NOT_STARTED;

private Set<User> assignedMembers;

/**
 * creates a new Task object
 */
public Task() {
   taskName = "";
   dueDate = new int[]{0, 0, 0};
   note = "";
   assignedMembers = new HashSet<>();
}

/**
 * creates a new Task object
 *
 * @param taskName
 * @param dueDate
 * @param note
 */
public Task(String taskName, int[] dueDate, String note) {
   this.taskName = taskName;
   this.dueDate = dueDate;
   this.note = note;
}

/**
 * creates a new Task object
 *
 * @param taskName
 * @param dueDate
 * @param note
 */
public Task(String taskName, int[] dueDate, String note,
        Set<User> assignedMembers) {
   this.taskName = taskName;
   this.dueDate = dueDate;
   this.note = note;
   this.assignedMembers = assignedMembers;
}
```

From this code you can see the fields that make up the task class, and the class also contains all the getter and setter methods for these fields along with some other methods for functionality like a toString method, equality method, and a method that checks if a given set of users is the same as the set of users assigned to the task. This fits in with our approach of using Universal Testing because the class can handle basic things the tasks need to interact with the other objects in the taskboard, and contains fields for the basic features that any task for any project will have, as all tasks that are a part of the project completion will have a name/description, due date, additional notes about the task, and the assigned members working on the task. This can be a computer science project that relates more to either hardware or software, or can be a project for an entirely different field. Therefore, the universal design concept, set to accommodate as many users as possible, will be the best fit for our project.

Here is our class diagram:

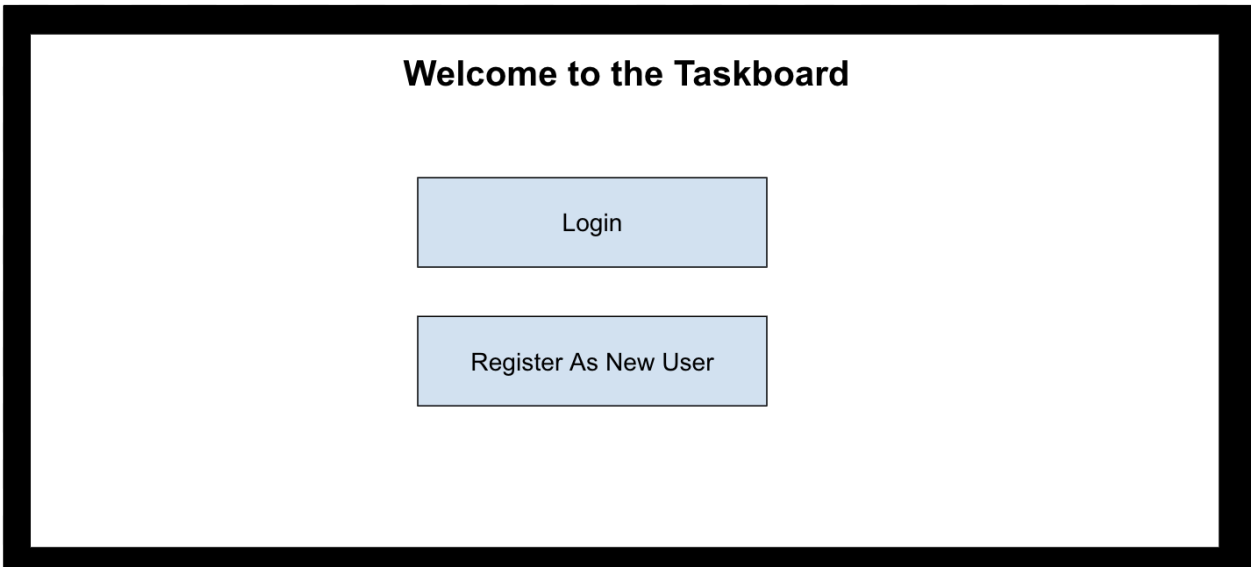


This is how the relationships were determined:

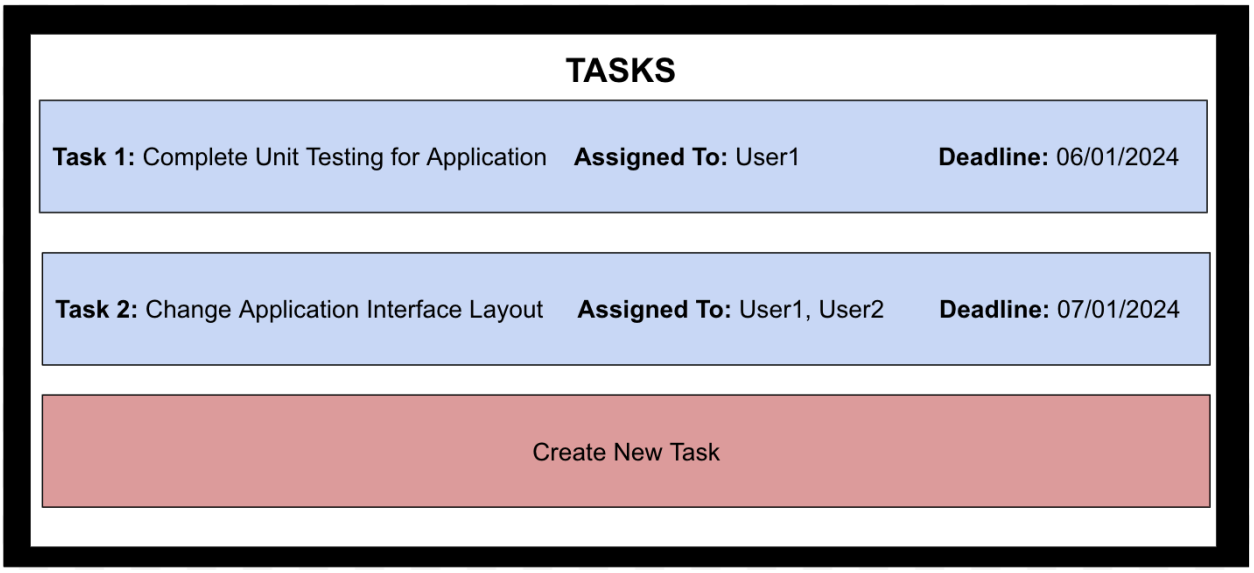
- Task
    - Wanted a task to have a list of all the users that were assigned to it
    - Contains all of the features needed to interact with a task, such as adding and removing users, setting due dates, setting notes, and noting how far along in a task a group is to a task along with getter and setter methods for these fields

    - Doesn’t need to know anything about the taskboard it is a part of
- User
    - wanted each user to have a list of all the tasks they were working on
    - Be able to add or remove tasks
    - Other fields need getter methods for its tasks and name
- Taskboard
    - Needs to know all the team members in the board and all the tasks in the board
    - Add, remove, and edit tasks and users/groups of users in the taskboard
    - Interacts with the GUI to make sure it displays correct information about tasks and users

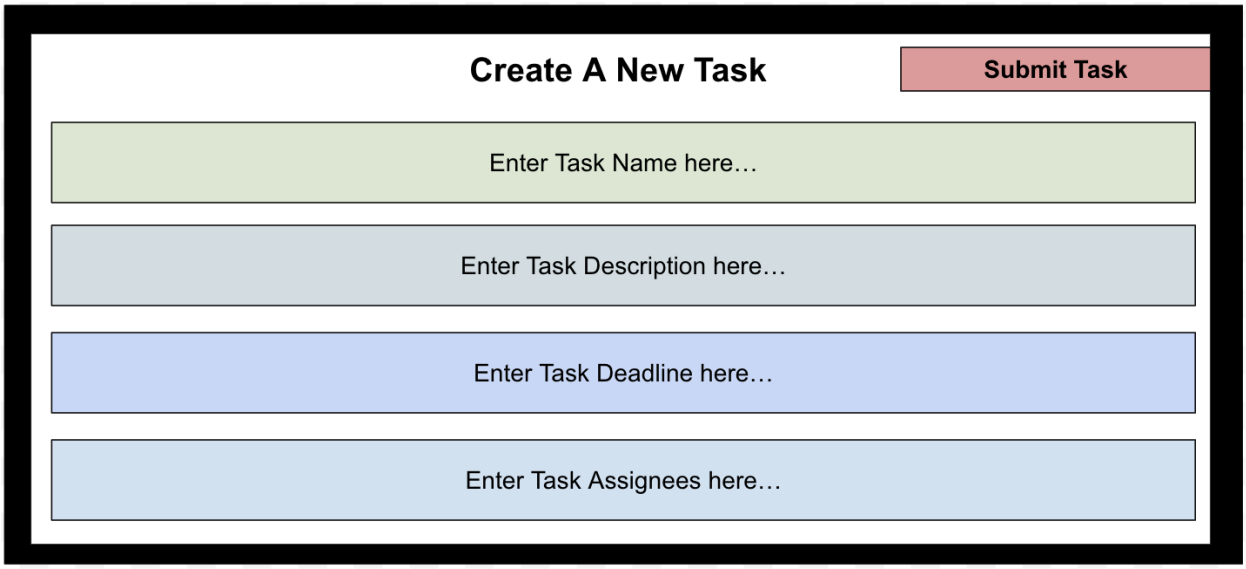
Design Sketch



This is the welcome screen of the taskboard. Here, users will have the option of logging in if they already have a set username and password. They can also register as a new user, where they will be asked to set their username and password.



This is the main viewing screen of our taskboard, here the user can view important information such as tasks, what users are assigned to the task, and the deadline for each task. When a user clicks on the create new task button, they will be prompted to enter information about a task.



When hitting the “Create A New Task” button, the user will be prompted to fill in necessary information about the new task. After filling out the necessary details for the new task they press the submit button and create the new task.

Our first design decision was to allow a user to log in or create a new account on the title screen before being able to gain access to the taskboard. On the home page, we wanted the user to be able to see all of their tasks because that is the first page that shows up, so they have easier access to all of the important information pertaining to a task. On this page, there is also an easily accessible button that can be used to create a new task. Once the button is clicked, a user is prompted for all of the details required for a task, so they can click on the text boxes and input the necessary details. Then, they can submit their task and view it on the home page. In the prototype we decided to implement a task viewer panel on the right side of the home page which allows you to select a given task and view and edit its features. This viewer has the task name at the top, and followed by four rows of text containing the task's due date, assigned members, status/completion, and additional notes along with an individual edit button for each feature. We found that this had many benefits, the most prominent being it gave us a simple way to edit the individual aspects of a task that our original plan didn't. It also made it possible to only display the task name on the homepage's list of tasks so that the center of the screen was less cluttered with text.

Process II Deliverable - prototype

https://github.com/pmurphy26/Taskboard