Web Based Scrum Board

Project Milestone 3

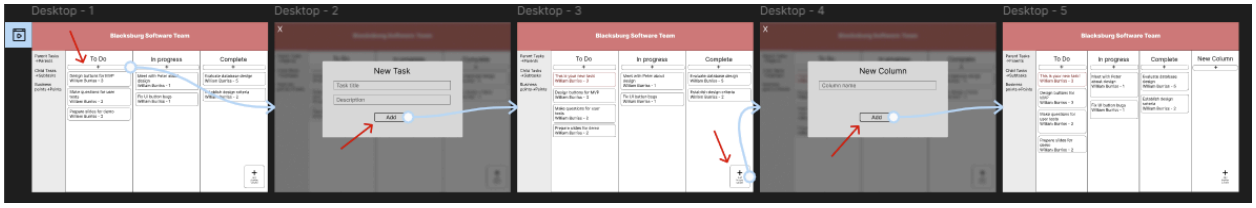
CS 3704 Fall 2024 (83365)

Aditya Rao
Aidan Carraretto
Akshara Gandrakota
William Burriss

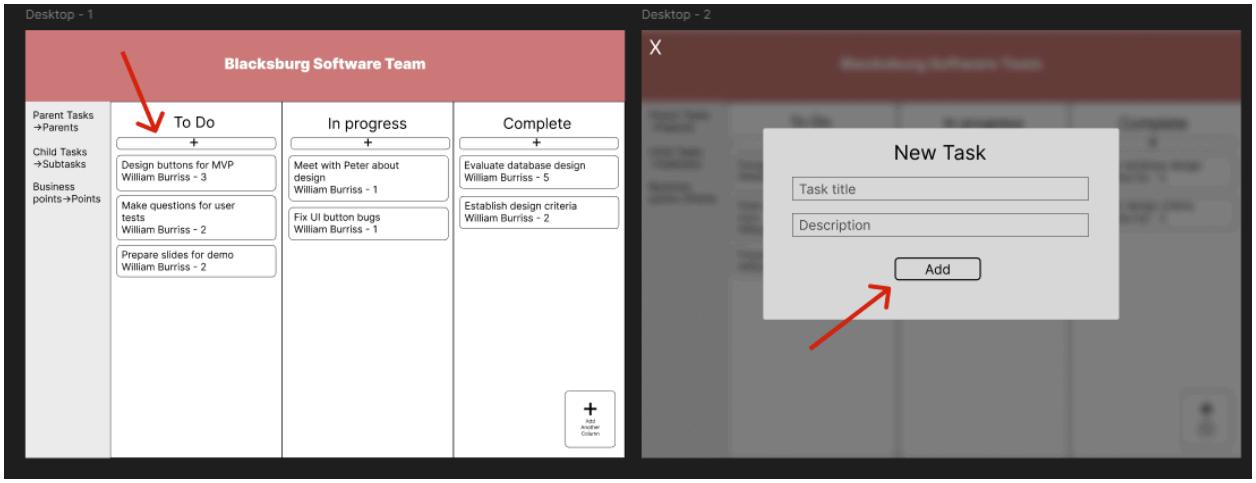
Process Deliverable II

Below is a link to access an interactive version of our figma prototype. You are guided by red arrows which will direct you to click the appropriate buttons for this example. To use this prototype follow the link and begin interacting with the interface.
https://www.figma.com/proto/w7ecleGuXyni8Kf9fP9pgr/Untitled?node-id=1-2&node-type=canvas&t=f7FWvtw4j7beCygX-0&scaling=min-zoom&content-scaling=fixed&page-id=0%3A1&starting-point-node-id=1%3A2

Below is an image of our layout for the figma prototype. In the image you will see how each of the screens are connected and how clicking a button will affect the state of the prototype. This image shows how the prototype will guide the user through our interactive example.



Below is another close up image of our prototype in figma. This image shows how two of the screens interact with each other. By clicking the buttons which are marked by the blue arrows you will be directed to the page at the end of the arrows path. The red arrows are again to guide the user through the prototype that we have created.



While our prototype is still limited we feel that we have made decent progress towards our final design. By creating an interactive prototype ,we have created a higher-fidelity version of our previous prototype and been able to evaluate how our design flows and identify any issues with the user interface. We have decided to keep our design simple for two reasons: Firstly, for the benefit of the user. If the user interface is too complicated, then the user might become frustrated or confused trying to navigate our design. To combat this, we have decided to create a simple design that is easy to understand and navigate through. Two, for the benefit of not over

complicating the design. The more things we design, the more things there are that can go wrong in the prototype and any further iterations. By keeping our design simple, we have the ability to make changes without having to worry about breaking too much for us to handle. Our plan for the future is to use this prototype as a blueprint to create a web application that will act in the same way. At the moment, we are not sure as to what specific technologies we will utilize. We will likely pick the technology that we have all used the most, such as plain JavaScript.

High-Level Design

The architectural pattern we would use is the model-view-controller architecture. The main reason why we chose this pattern is to be able to have a clean pipeline in our website between processing, storing, and displaying information. It is important for a scrum board to be able to maintain information, for example, and we can easily divide and conquer tasks performed on the website through the controller part of the architecture, which goes about manipulating the information stored within the scrum board through user input, and the model component, which holds data for the use and modification of any controller tasks. The bridge between the controller and view components, if the program is done properly, easily divides front- and back-end work. This will allow those of us who aren't as confident in back-end code to specialize on how the website and its data is actually presented to the user.

Initially we were considering using the event-based architecture, which could also work. However, a big advantage would be having a well-defined divider between access to that information, processing of that information, and retrieval/viewing of that information. Some considerations for this architectural pattern include possible performance degradation over time due to more moving parts. This also requires some extra setup on our end to build a stable backbone for the rest of development.

Low-Level Design

For our low-level design implementation example, we will implement a notification system through pseudocode. In order to implement the notification system within this project, we would use the observer design pattern. We'll use this because it allows dependency between objects, meaning that when one factor in one section changes, anything else dependent on that also changes. This is especially useful for the notification system, where the whole purpose of it is to notify users about any changes, comments, or additions that have been made to the scrum board. When one thing changes, the notification system, which is also dependent on that feature, will reflect those changes. The observer design pattern also allows for scalability, which means new notifications can be added, the amount of people notified can change, and the type of notification can be specified.

Here is the pseudocode for how this can be implemented:
Interfaces:
1.
Subject Interface:
    Method addPerson(person)
    Method removePerson(person)
    Method notifyPersons(message)
2.
Person Interface:
    Method update(message)

Class implementations:
1.
Class StatusNotifier Implements Subject:
    Private persons = empty list

    Method addPerson(person):
        Add person to persons

    Method removePerson(person):
        Remove person from persons

    Method notifyPersons(message):
        For each person in persons:
            Call person.update(message)

    Method setTaskStatus(status):
        Update the status of the task

```
    Call notifyPersons("Task status updated to: " + status)
```

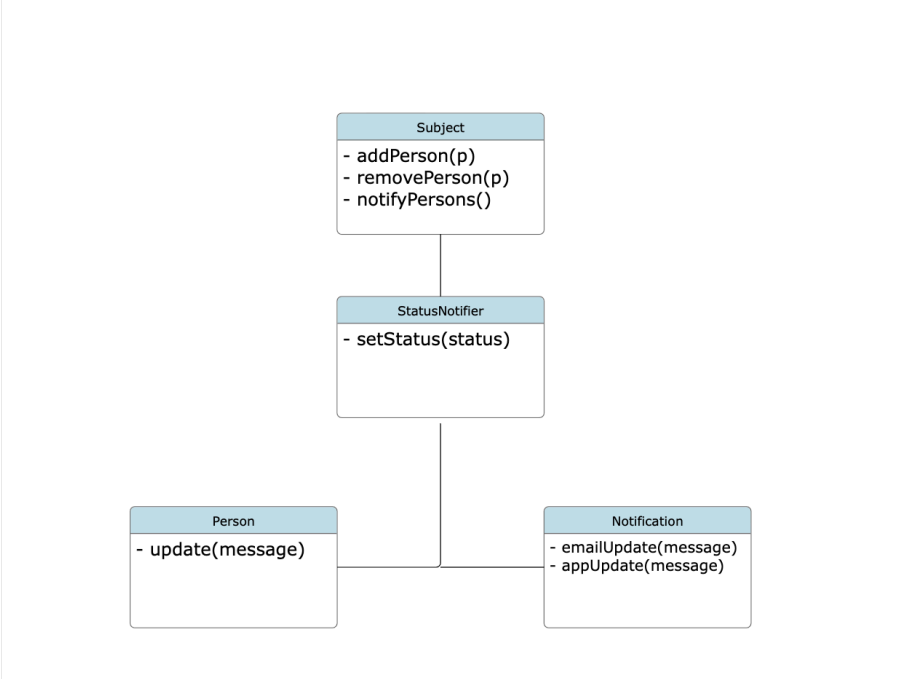
2.

```
Class NotificationPerson Implements Person:
    Private notificationType

    Constructor(notificationType):
        Set notificationType

    Method update(message):
        If notificationType == "Email":
            Display "Email notification: " + message
        Else If notificationType == "App":
            Display "App notification: " + message
```

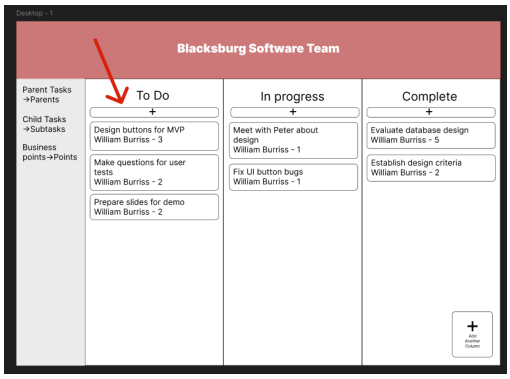
Here is the class diagram that reflects this:



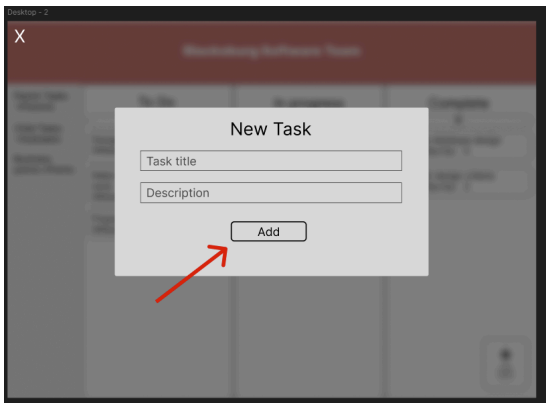
Design Sketch

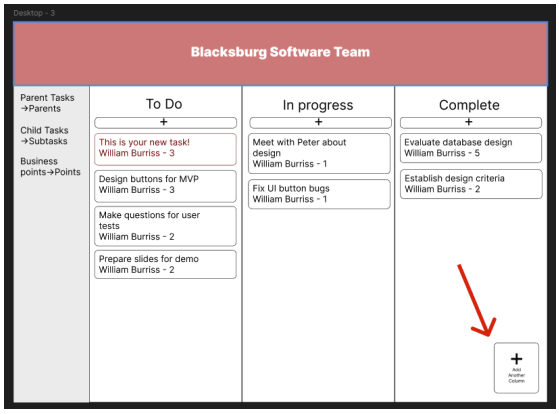
As displayed in our deliverable, we have created an interactive prototype in figma. In this section, we will go into creator detail on the design process. We will discuss how the user will use the prototype and why we designed the user interface in the way that we did.

First, if we look at the first page which can be seen below we see the same prototype we had last milestone. The red arrow is to guide the user when they are exploring our prototype. When it comes to explaining what each part does and how to interact with the prototype, we went into greater detail in the previous milestone. To recap the prototype's design: we have 3 columns with each having a button on the top to add more tasks. On the left side we see the ability to change the name of stories and epics. And in the bottom right we see a button to add a new column.

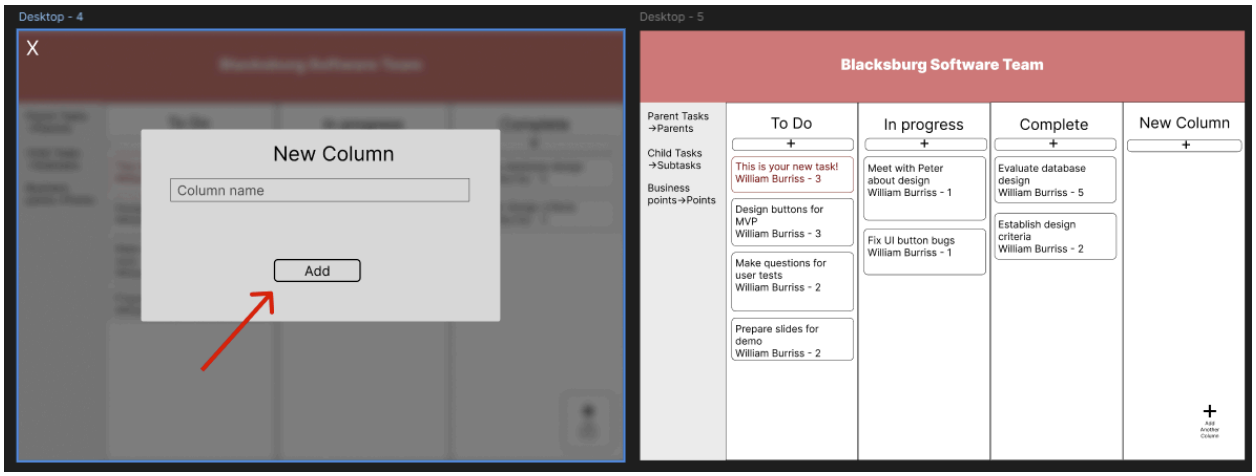


In the first image below, we see the screen that appears when the user clicks on the add task button for the todo column. Here they can enter a title and description and add the task. The second image is the screen after the task has been added. The new task is colored red to indicate that it has been added successfully. The red arrow moves on the main page to the add column button to show the user what to do if they want to add a new column.





Below are the screens when the user adds a new column. We can see they are prompted in the first image to enter a column name before clicking on the add button. Finally, the final screen shows that their new column has been added.



Below is our full prototype for this milestone. We aimed at creating a design that provides enough fidelity such that we and any other testers have a strong idea of how the actual program would operate in action. We elected to use figma to create an interactive prototype with the goal of making our prototype as immersive as possible. Below is the link to our prototype, which anyone should be able to access, along with the image showing how we used figma to make our design interactive.

https://www.figma.com/proto/w7ecleGuXyni8Kf9fP9pgr/Untitled?node-id=1-2&node-type=canvas&t=f7FWvtw4j7beCygX-0&scaling=min-zoom&content-scaling=fixed&page-id=0%3A1&starting-point-node-id=1%3A2