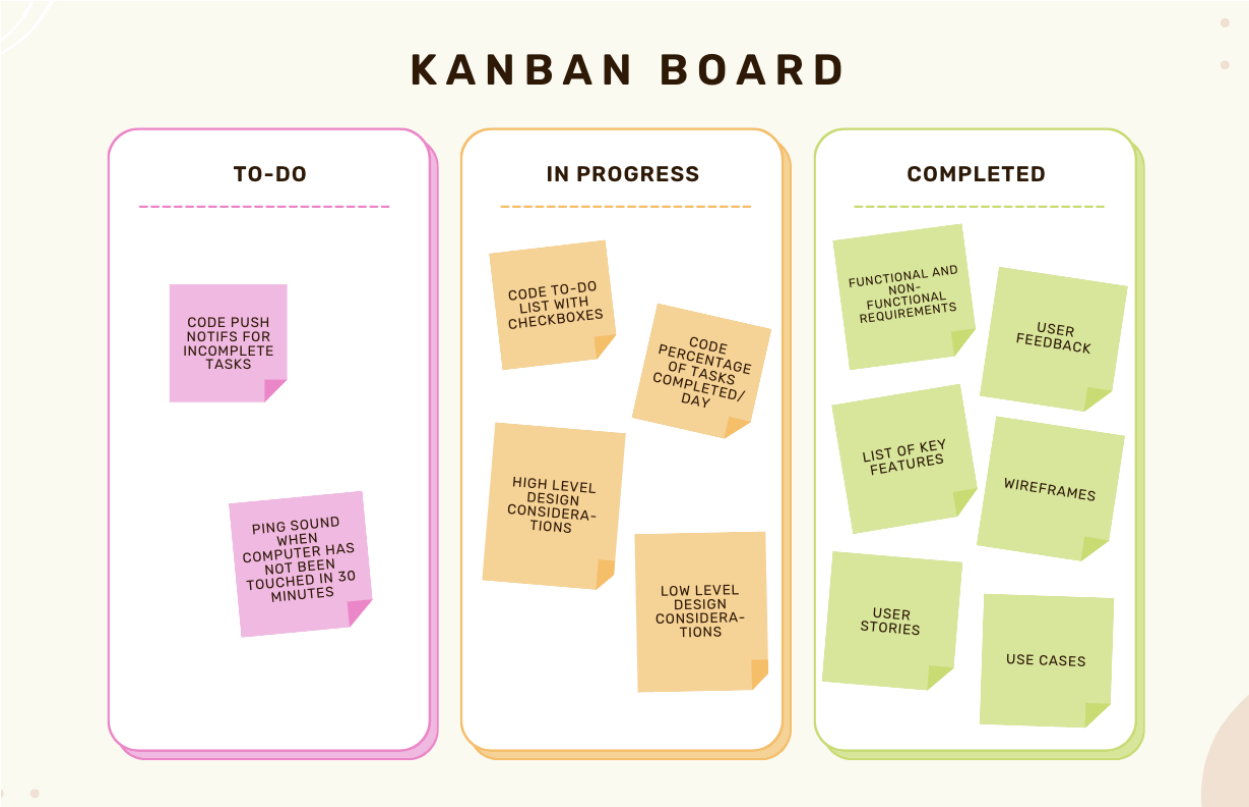
Project Milestone 3

Process Deliverable II (3%)

- Kanban: submit a depiction (i.e., screenshot, link, etc.) of completed and prioritized tasks in your team's Kanban board
    - To Do
        - Code push notifs for incomplete tasks
        - Ping sound when the computer has not been touched in 30 minutes
    - In Progress
        - Code to-do list with checkboxes
        - Code percentage of tasks completed/day
        - High-Level Design considerations
        - Low-Level Design considerations
    - Completed
        - Wireframes
        - User Stories
        - Use Cases



High-level Design (4%)

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

High-Level Design explains the system architecture used to develop a system. Software architecture is defined as the organization of a system. It includes components and their interactions between each other as well as the environment in which they operate. Software architecture designs the overall shape and structure of a system.

There are 3 main types of architecture patterns: Pipe and Filter, Event-Based, and Layered. Pipe and Filter contains the idea that the output of each element is the input of the next element. It is efficient for compiler optimization and signal processors. Event-Based architecture promotes the production, detection, consumption of, and reaction to events. It works especially well for graphical user interfaces, mobile apps, robotics, and video games. Layered architecture uses multiple layers that allocate responsibilities of a software product. There are two variants in layered architecture: 2-layered and 3-layered. Client-Server and Data-Centric architecture are 2-layered while Model-View-Controller (MVC) is 3-layered. Client-Server architecture partitions tasks or workloads between the providers and consumers of data. Data-Centric architecture uses a data store that resides at the center to be accessed frequently by agents. Examples of 2-layered architecture include distributed file systems and version control systems. MVC includes UI to interact with users and stores/retrieves information as needed. MVC works well for most modern web applications.

Focus-Bot is a program that is extremely dependent on internal data systems and user interaction. Some functionalities include pinging the computer when 30 minutes of inactivity has been realized, continuously updating a user-defined daily tasks list, presenting an incomplete task list before lunch and end of work day, and constantly displaying a progress bar of completed tasks at the top of the screen. An additional functionality for the project manager provides a comprehensive updated list of each employee's daily progress to more easily keep track of individual progress within a team environment.

Pipe and Filter would not apply well to FocusBot because the functionalities of FocusBot simply do not align with the way a Pipe and Filter process is set up. FocusBot requires a data center to retrieve values such as time, number of completed tasks, user name, and averaging of tasks.

Event-Based architecture could be a viable architecture pattern for FocusBot as a GUI is a large part of the functionality of the system. However, there is no aspect of a data center within event-based architecture. Additionally, a known con of event-based architecture is that components have no control over the order of execution. Users of FocusBot will be interacting with the system on a daily basis to update completed tasks

and add new ones. Having a pre-ordered operation for the system would not work well for FocusBot because it would not allow for continuous changes and updates to the task list or inactivity pinger.

This leaves Layered architecture. Layered seems to be the architecture pattern that aligns most with the functionalities of the system. Specifically, the 3-layer architecture in the form of MVC will be utilized for FocusBot. Some known pros of 3-layered architecture are “supporting increasing levels of abstraction during design” and “supporting reuse and enhancement”. FocusBot will involve multiple forms of abstraction. For instance, the data center simply holds data; it should not know about the progress bar, inactivity ping, or task list. The inactivity ping is an entirely separate entity from the progress bar; it is only concerned with how often the mouse is moved within a 30-minute window during the workday. MVC handles this increased level of abstraction. Users of FocusBot will interact with and reuse the system multiple times throughout the day. This feature aligns with the second pro of 3-layered architecture.

After considering the process, pros, and cons of each architecture pattern, it was determined that the layered, and more specifically, MVC, architecture pattern would apply best to the desired functionalities of FocuBot.

Low-level Design (4%)

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

For implementing a specific sub-task of our FocusBot, we believe that the “Behavioral Design Pattern” would be helpful regarding implementation. This is due to the fact that behavioral design patterns provide a chain of responsibility that would deal with the objects. For example, our subtask is the computer pinging a sound after 30 minutes of an activity is complete to keep you on track for your breaks.

Strategy is also a key value of behavioral design patterns. Strategy refers to the encapsulation of an algorithm inside a class. This works hand in hand with the above mentioned subtask. This is due to the fact that the algorithm that tracks your 30 minute breaks would fit well as an algorithm within a specific class of FocusBot. For example, below represents the pseudocode for the example of the computer pinging a sound after 30 minutes of inactivity:

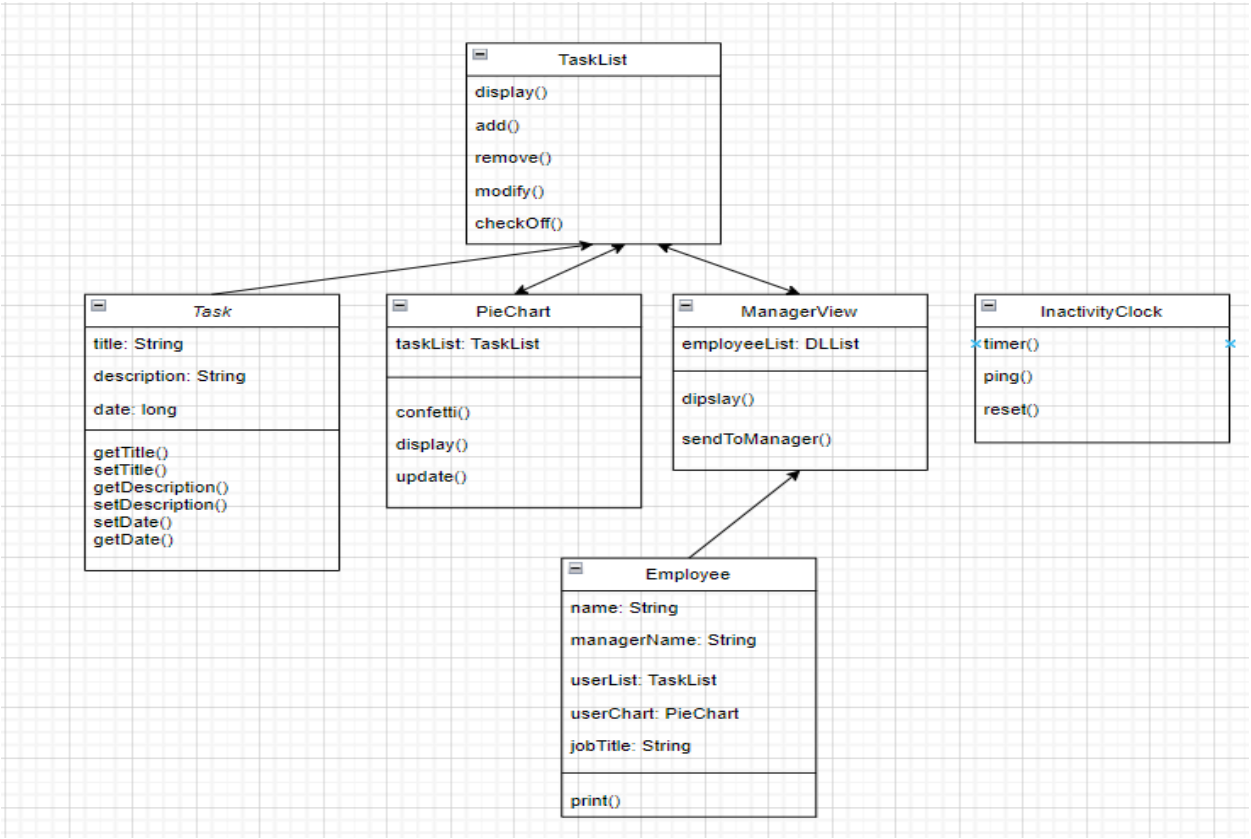
```
Initialize variable “Timer” = 0
Set variable “Time” = 30 minutes

Function StartActivity():
        Print "Starting time now."
        Call TrackTime()

Function TrackTime():
        While Timer < Time:
                Increment Timer by 1 second every second
        Call PlaySound()
        Reset Timer = 0
        Print"30 minutes have passed! Log back on."

Function PlaySound():
        Load sound file
        Play sound file

Main Program:
        Print "Welcome to the Focus Bot!"
        Ask the user: "Start a new focus session? (yes/no)"
        If user response == "yes":
                Call StartActivity()
        Else:
                Print "Session not started."
```



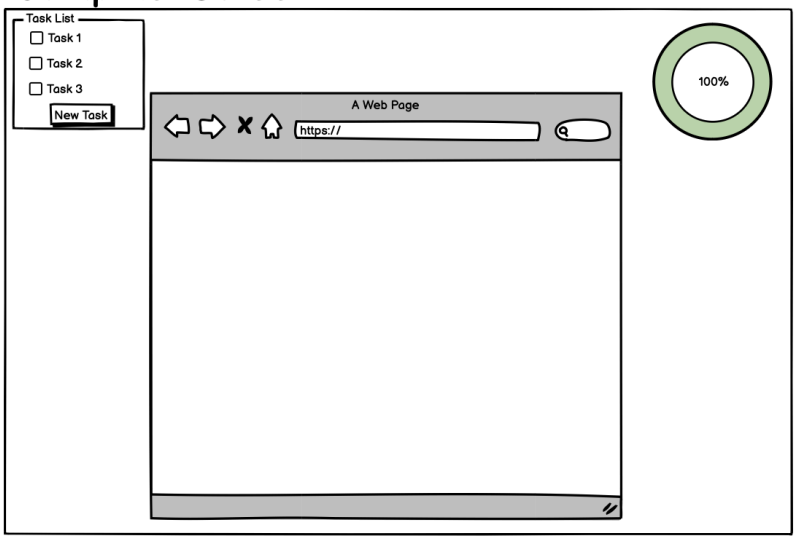
Design Sketch (4%)

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:

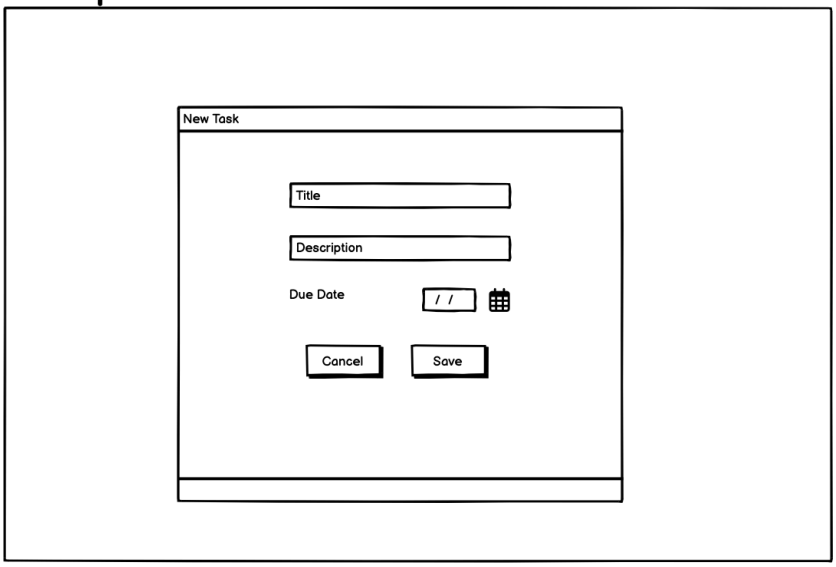
- Creating a wireframe mockup of your project user interface in action.
- Creating a storyboard that illustrates a primary task that a user would complete with your project.

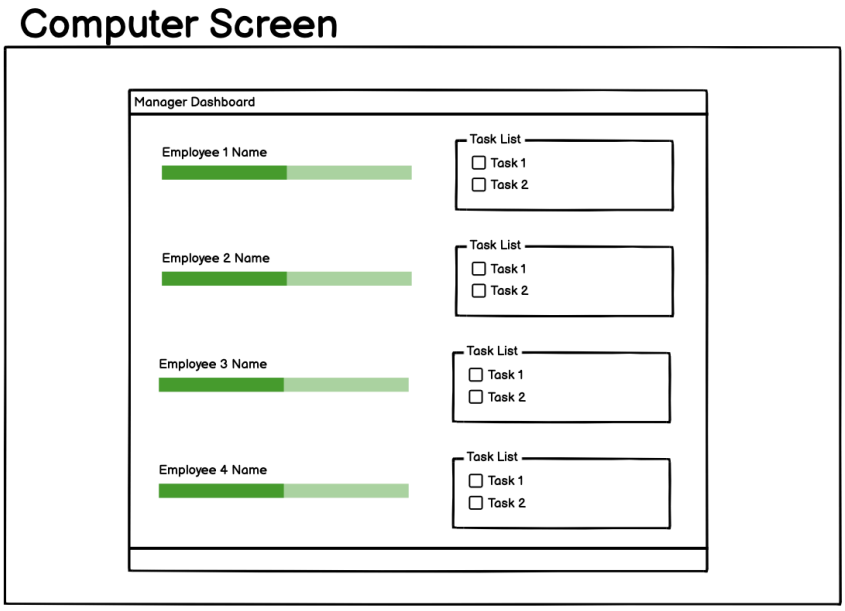
Provide a brief rationale (no more than 1 paragraph) explaining some of the design decisions for your program based on the provided sketch. Use concepts discussed in class to justify your design.

Computer Screen



Computer Screen





The design sketch above depicts a few aspects of the FocusBot program. The image on top is a view of the user's desktop with the pie chart and task list enabled and displayed. The pie chart lies at the top right of the screen and the task list at the top left. Our team focused on the 10 Usability Heuristics of UI Design when coming up with this wireframe. For the top image, we incorporated the "aesthetic and minimalist design" heuristic by making the pie chart and task list look simple and without frills. The middle image is what a working update page to the task list would look like. Here, we incorporated the "user control and freedom" heuristic by allowing the user to add, delete, and modify any existing or new task. Also included in this image is the "error prevention" heuristic. Any additions to the task list will not accept a date that is not in a specific format of HH/DD/MM/YYYY. If an input is outside of what is accepted, the task list will not be updated and an error message will be displayed asking the user to input the date in the correct format. This also falls under the "helps users recognize, diagnose, and recover from errors" heuristic. Lastly, the third image depicts the manager's point of view at the end of the day. A heuristic that applies to this wireframe is the "consistency and standards" because the list is displayed at the same time and in the same format at the end of each business day. All together, these three images depict most of the functionalities of our FocusBot program.

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.