CS3249 User Interface Development
Assignment 3: Custom UI WIdget

Due date:

This assignment is due **March 8, 2019, 23:59** (Friday of Week 7). The weightage of this assignment is **10%**.

Grading:

- Draw the inputs state diagram for a thermostat UI component (4%)
- Design and draw the software architecture as an MVP pattern (2%)
- Implement the designs in a single-page application (SPA) webpage (4%)

Description:

Assignment 3 is an **individual** assignment. The purpose of Assignment 3 is to apply the lessons you have learned about GUI design patterns (Lectures 4) and input/output (Lecture 5) to design a custom UI widget for a smart thermostat as shown in Figure 1. See the animation online or in lecture notes to get an idea of the UX. The thermostat is a round UI that changes color and number depending on what target temperature the user sets. The user sets temperature using buttons (right button to increase temperature, left button to decrease). The thermostat controls an air conditioner (AC) for cooling and a heater for heating. See Figure 2 for an example scenario of the behavior and controls.

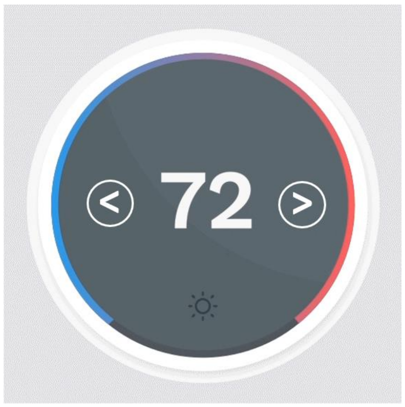

***Figure 1. Smart Thermostat UI Component*. See animation online or in lecture notes.**

Your goal is to design the model-view-presenter (MVP) architecture pattern and the inputs state diagram for this UX. With these designs implement the code in a Single-Page Application (SPA). The widget must be implemented using only plain Javascript, HTML and CSS only.

- **You must use HTML5 Canvas to draw the UI.**
- **Do not use external libraries or frameworks.**
- **Do not use fancy CSS (e.g., with CSS border-radius)**, since this is generally unintuitive and leads to code that is hard for others to understand.

Important Variables

You will need to understand and use the following variables in your program.

- Target (set point) temperature (T_t)
- Current (measured) temperature (T_c)
- Temperature margins
    - $\mathrm{d}T$: buffer regarding comfort range
    - $\mathrm{d}T_{cool}$, $\mathrm{d}T_{heat}$: buffer to delay AC/heater switching
- Mode: off, cooling, or heating

Thermostat Rules

- If $T_c > T_t + \mathrm{d}T + \mathrm{d}T_{cool}$, then mode=cooling
- If $T_c < T_t - \mathrm{d}T - \mathrm{d}T_{heat}$, then mode=heating
- If $T_t - (\mathrm{d}T - \mathrm{d}T_{heat}) < T_c < T_t + (\mathrm{d}T - \mathrm{d}T_{cool})$, then mode=off
- Else, then do not change mode

User Interface (UI) Rules

- If mode==off, then grey color
- Else If mode==heating, then red
- Else If mode==cooling, then blue

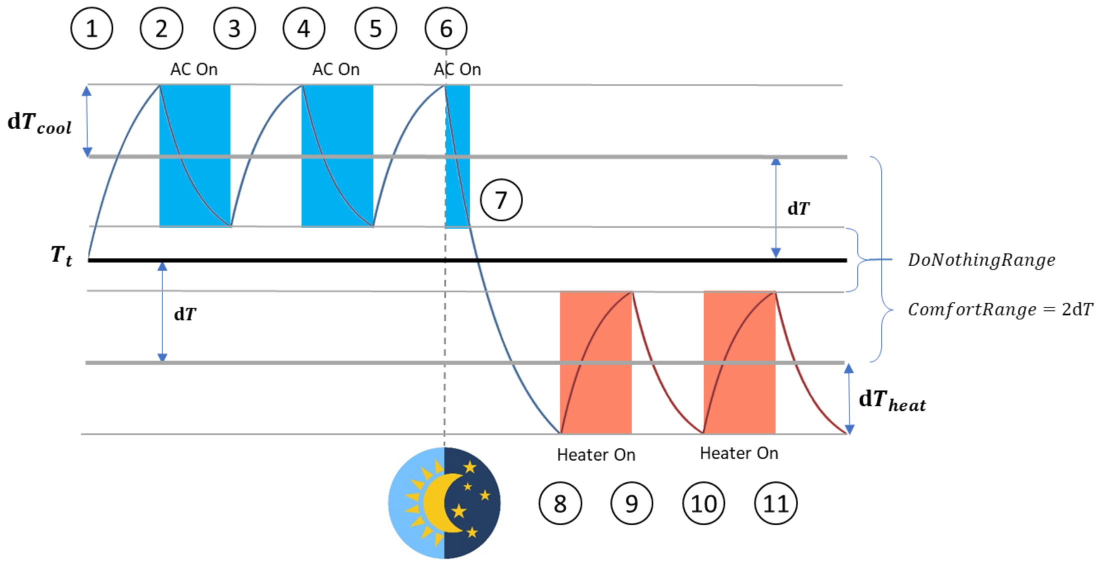

Figure 2. Example scenario from day time to night time showing how room temperature changes would cause the air conditioner (AC) or heater to turn on/off at various times. The curved zig-zag line indicates room temperature across time. Vertical axis is temperature, horizontal axis is time.

Example Scenario (see Figure 2)

1. Thermostat is off. Room temperature is rising.
2. Temperature rises and gets too hot with $T = T_t + \mathrm{d}T + \mathrm{d}T_{cool}$.
   - AC turns on (mode=cooling) and starts to cool the room. Temperature starts to drop.
3. Temperature drops until a little too cold, i.e., $T = T_t + \mathrm{d}T - \mathrm{d}T_{cool}$.
   - AC turns off (mode=off). Room starts to heat up again.
4. Repeat (2).
5. Repeat (3).
6. Repeat (4). This time the room cools down faster (colder outside, day → night)
7. Repeat (5). Temperature drops too low, i.e., $T = T_t + \mathrm{d}T - \mathrm{d}T_{cool}$.
   - AC turns off. But because it is too cold outside, room temperature continues to drop.
8. Temperature drops until too cold, i.e., $T = T_t - \mathrm{d}T - \mathrm{d}T_{heat}$.
   - Heater turns on (mode=heating). Room starts to warm up.
9. Temperature rises until a little too hot, i.e., $T = T_t - \mathrm{d}T + \mathrm{d}T_{heat}$.
   - Heater turns off (mode=off). Room starts to cool down again.
10. Repeat (8).
11. Repeat (9).

Some Hints

- State Diagram
  - Think about how to maintain the model data
  - Think about how to use databinding between sub-components
- MVC Architecture
  - If you cannot figure out the MVP architecture, it is ok to implement without any particular architecture design, but you will lose the allocated points.
- User Interface
  - Do not forget to include buttons to change target temperature
  - You do not need to implement smooth transition of colors
- Other details
  - Cite any tutorials, discussions or source code that you used to get inspiration. Plagiarism will be penalized.
  - Provide a **screen recording** of the interaction with the widget in the submitted files. This will be helpful if your code does not work or we do not understand its behavior.
  - You may want to include paragraph text to **describe and explain** your diagrams; this will help the graders understand your work.
  - You may include **screenshots** to point to certain UI components in the user interface.

Deliverables:

- **Report:** in both Word Doc and PDF formats containing the state diagram and the MVP pattern
- **Code:** Create a subfolder named code in your main submission folder and place the html, css and js files under it. Additionally, add a readme text file if any special steps are required to test the widget. You may also include details about your implementation that you think might be helpful to assist with grading.

Submitting Assignment 3

All deliverables should be archived into a .zip file with the following naming convention:

YOURNAME_MATRICNUMBER_Assignment3.zip

submitted to the IVLE Assignment 3 workbin before March 8, 2019, 23:59.