Project: Custom UI Dashboard

This assignment has four deadlines due on

Group Formation:	Thursday, 14 March 2019, 11:59pm (Week 7)
Report Draft 1:	Friday, 29 March 2019, 11:59pm (Week 10)
Report & Presentation Slides:	Friday, 17 April 2019, 11:59pm (Week 13)
Presentation:	Thursday, 18 April 2018, 10am-12pm (Week 13, during Lecture)

The weightage of this assignment is **25%**. Grading:

- Documentation (10%)
- System implementation (10%)
- Presentation (5%)

Learning Aims and Encouragement:

- We use assignments as another learning opportunity in the course beyond lectures and tutorials.
  - You apply the concepts learned in class with technical implementations that you can learn from online forums and documentation.
  - However, we will help you if you are struggling with a specific framework, library, or implementation. Please reach out to us on Slack for prompt replies.
- We challenge you to learn how to design and implement a full UI application *from scratch*.
  - This project (and assignments) are more representative of real-world projects to help you appreciate the challenges in real-world work assignments.
  - **Hint:** careful thinking about the designs and choice of technologies or libraries can help to save you difficulties in programming. Consider this trade-off carefully.
  - We do not provide scaffolding code which may bias your creative designs or limit your learning that there are many decisions you need to consider.
- We expect this to be **hard** project (sorry!) for every student and will curve the score accordingly (as we do for all assignments). Please start early!

Description:

The Project Assignment is a **paired group assignment**. The purpose of the Project is to apply the lessons you have learned on technical topics and practice on GUI design patterns (Lectures 4) and input/output (Lecture 5), web frameworks (Lecture 6), and N-tier Web Architectures (Lecture 7) to design a custom UI dashboard for a smart dormitory with temperature monitoring as shown in Figure 1. You need to implement all the UI components shown with the behaviors described in the next section.

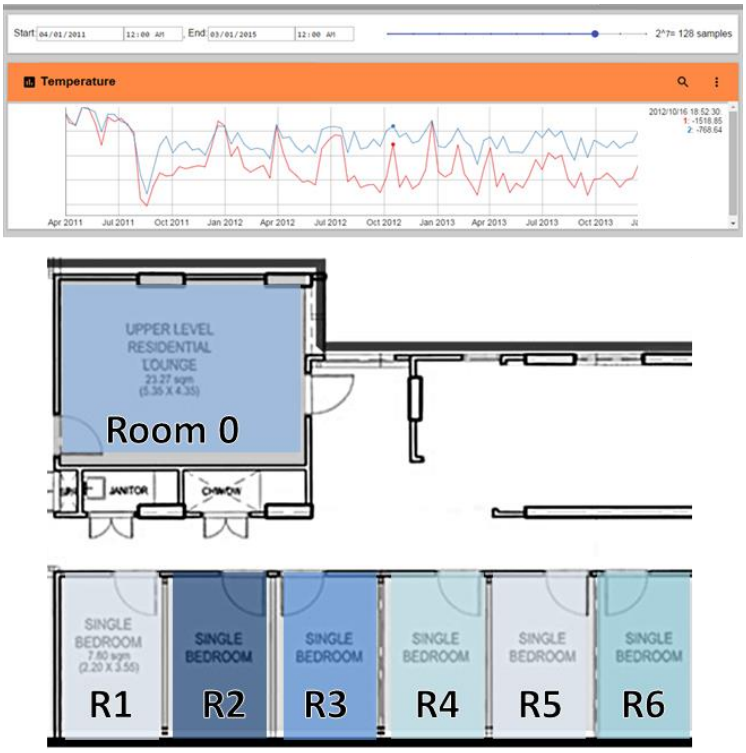

Figure 1. Smart Dormitory Temperature Monitoring Dashboard.

Dashboard Features

Try to implement all of the following interaction behaviors for the dashboard's UI components:

1. Start and End Time fields
    - For users to explicitly change start and stop times by typing
2. "Number of Samples" Slider
    - For users to change the *number of points per line* to draw in the graph
3. Temperature graph
    - Shows multiple lines
    - One line per room
4. Graph sync-ed with
    - Start/End times
    - Number of Samples (for each line in the graph)
5. Graph interaction
    - Zoom by clicking and dragging (like Dygraphs)
    - Pan with mouse drag (like Dygraphs)
6. Room colors change with
    - Average temperature of time period
    - Darker blue for colder temperature
7. Clicking on room toggles whether
    - Temperature graph for room is shown/hidden
    - Temperature color (rectangle) on room is shown/hidden
    - This helps to see your graphs more clearly and in a less cluttered way

Goal

Your goal is to design the software architectures for the dashboard and implement the design in an N-Tier architecture with the React and Meteor JS frameworks.

- Software design
  - Draw the component hierarchy (as a tree diagram) and indicate on the dashboard screenshot the various UI components in your dashboard.
  - Design and draw the software architecture as an MVP pattern
    - **Hint:** account for all components
    - **Hint:** this is only for the front-end components, not the back-end.
  - Draw the front/back-ends architecture diagram
  - Draw the UI inputs state diagram for the full dashboard UI
    - **Hint:** the state diagram for UIs is different than the application's state
  - Define the format (schema) in JSON for the temperature data
  - **Hint:** you may want to do some light prototyping as you come up with your architecture design. This can help you find early limitations.
- Software implementation
  - Front-end with React JS
    - For the graph, you can find an appropriate React component, or find a great JS library and wrap it as a React component.
      - **Hint:** do not draw your own graphs from scratch; it will be error prone and hard to make interactive
    - You need to draw the floorplan and room toggles. You may use Canvas or SVG.
      - **Hint:** you should not be pixel plotting everything and drawing *fully* in Canvas will be tedious
  - Full stack with Meteor JS
    - Data storage in MongoDB (and front-end Minmongo cache)
    - You need to ingest the temperature data into MongoDB (e.g., using the `insert` function).

Other Hints

- Cite any tutorials, discussions or source code that you used to get inspiration.
  - Plagiarism will be penalized.
- 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.

Tasks:

Group formation

1. Sign up for the group on IVLE by **Thursday, 14 March 2019, 11:59pm (Week 7)**
   Note, that there will be equal grading for both individuals in a group. It is your responsibility to talk to your teammate and share the workload equally. If you feel an exception is required, raise this with the teaching staff at least a week before the deadline.

Self-learning activities for implementation, but we will be happy to guide you if you need help. Please ask!

2. Complete React tutorial (Tutorial 4 exercise)
3. Study Thinking in React to help document your application design
4. Complete Meteor-React tutorial
5. Study and choose a time series chart library / component (see lecture notes for examples)
   - Use an existing React component library, or
   - Wrap a good library yourself into a React component
6. Learn about storing time series data in MongoDB (e.g., article)

Data preparation and Application development

7. Create private GitHub repository (e.g., tutorial)
8. Download temperature data from IVLE (room-temperatures.csv)
9. Develop the application with **Meteor+React**
   a. Ingest (insert) data into MongoDB

Deliverables

- **Documentation Report (10%):** in both Word Doc and PDF formats, describe with diagrams and explanatory text, specifically:
    1. Component hierarchy
    2. MVP architecture diagram (MVP front-end)
    3. N-Tier architecture diagram of the full system
    4. UI state diagram of application
    5. Data schema (JSON format) of the temperature data
    6. Describe any mathematical equations or calculations you performed
    7. Discuss weaknesses
        - E.g., in performance, reusability of code, testability
        - In design and what should be changed, e.g., different technology, different architecture
    8. Project folder structure, README/setup instructions, Team contribution (who did what)
- **Code (10%):** Create a subfolder named code in your main submission folder and place the html, css and js files under it. Include the csv file in a location such that it will load properly in your application. Include all library files that you have downloaded so that your application is self-contained and can run. Additionally, add a readme text file if any special steps are required to test the application. You may also include details about your implementation that you think might be helpful to assist with grading. Implement the following:
    - Front-end in React JS; implement the 7 dashboard features
    - Back-end with Meteor
    - Data on MongoDB (in Meteor)
    - Include: video of demo
- **Presentation (5%):** upload your presentation as PowerPoint slides the night before your presentation, that you will be presenting as a group.
    - Here are some guidelines for your slides
        1. Introduce the dashboard, its UI components, and component hierarchy
        2. Describe and justify the MVP architecture
        3. Describe the UI state diagram (and, optionally, additional overall application state diagram, if you wish)
        4. Describe the data schema
        5. Show a (live) demo of the application
        6. Discuss weaknesses in software design
        7. Share four key learning experiences about the design and/or implementation (2 per student)
    - Presenting timing (14 min total)
        1. 1-min (setup)
        2. 7-min (talk/demo)
        3. 6-min (Q&A)

Submitting Project Assignment

Draft 1

The first deliverable is the Document Report (draft 1) which should be archived into a .zip file with the following naming convention:

GROUPID_YOURNAME_MATRICNUMBER_ProjectAssignment_Draft1.zip

submitted to the IVLE "Project - Report Draft 1" workbin before **Friday, 29 March 2019, 11:59pm (Week 10)**. Each student should submit one for himself/herself even though this is a group submission, i.e., group members will submit the same files, but with different names.

No code is needed for Draft 1.

Final Submission

Subsequently, for the final submission, all deliverables should be archived into a .zip file with the following naming convention:

GROUPID_YOURNAME_MATRICNUMBER_ProjectAssignment.zip

submitted to the IVLE "Project" workbin before **Friday, 17 April 2019, 11:59pm (Week 13)**. Each student should submit one for himself/herself even though this is a group submission, i.e., group members will submit the same files, but with different names.