Project #2: Design of a Python Library for a Building Energy Performance Report

Background

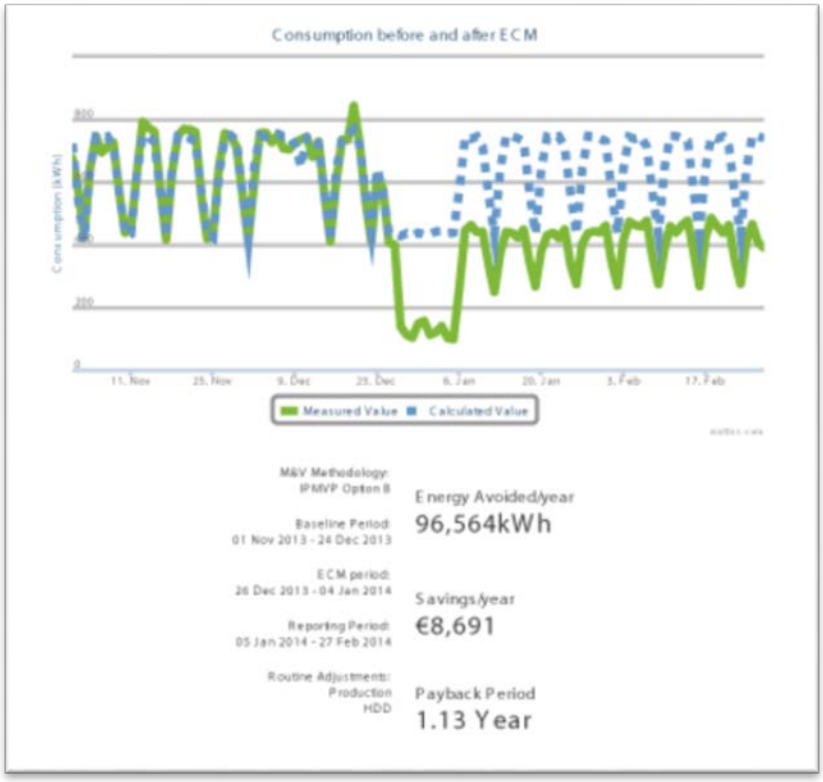
Buildings are responsible for 40% of CO_2 emissions in US. Most of these emissions come from the combustion of fossil fuels to provide heating, cooling and lighting, and to power appliances and electrical equipment. Most buildings waste a significant amount of energy on inefficient operation and UC Davis is no exception. However, UC Davis is committed to improving its operational energy efficiency and eliminating waste and so has commissioned a team in Facilities Management to identify and eliminate energy waste in its campus buildings. The Energy Conservation Office (ECO) is leading this effort and is looking for students interested in joining their team.
The ECO has developed the Campus Energy Education Dashboard, or CEED (http://ceed.ucdavis.edu), to help the campus community better understand energy use in buildings. They have also developed the TherMOOstat (http://thermoostat.ucdavis.edu/#/map) thermal feedback app for students, faculty and staff to send in their feedback on room comfort issues and take part in our energy investigations across campus. The ECO is now interested in developing additional online tools for displaying, reporting, and interacting with UC Davis' energy data.

Project Description

The ECO is looking for a team of students who want to help design and build a Python Library for a Building Energy Performance Report for their energy engineering team. This report will be used to present actual energy and cost savings to campus administration that is financing the project work.

The project team will design and build an Interactive Building Energy Performance Report:

- Develop a Python software library to evaluate savings
- Query a database of building operational data and metrics
- Develop user inputs for time periods representing:
    - baseline performance (before)
    - project dates
    - a training period
    - a comparison performance period (after)
- Develop the report output which will display building performance before and after a project implementation, with resulting savings indicated both graphically and numerically.



Outcome

- A working prototype of the report.
- The students will be expected to deliver the report along with source code of the library in Github.

If the project is successful, the ECO is looking to hire one or more students for an internship during summer 2017.

Skillset

- The ideal team will have experience with databases (retrieve the data) and Python (processing data), along with some HTML, CSS and Javascript (web development).

Notes from Energy Conservation Office IT:

First and foremost, we're only writing **web applications**. These are applications served on the internet through the web browser. We're not really skilled enough to maintain 'native' apps, e.g. ones that run on Windows or are downloadable through a smartphone's app store.

Secondly, we work exclusively in **Javascript**. We use **node.js** to run apps on the server side, and a client-side Javascript framework like **React** or **Angular** to create the user interface in the browser.

Node.js - https://en.wikipedia.org/wiki/Node.js
React - https://en.wikipedia.org/wiki/React_(JavaScript_library)
Angular - https://en.wikipedia.org/wiki/AngularJS

Node.js on the server side is essential. We're less strict about what they want to use on the client-side, so long as it uses Javascript/HTML.

I thought about giving you a wider range of web app code languages – such as PHP, Python, or even Java – but after some consideration, I think we would prefer to stick **exclusively with Javascript**. I think it would end up being more work on our end to learn and maintain a web app in a different language, than to just rewrite the whole thing in Javascript to keep with our existing technology stack.

We also prefer that apps be **mobile-responsive** – that is, they are designed to work equally well across mobile and desktop screen sizes. But, if we received an app that was desktop-only, I think we could adapt it to work on mobile screens without too much effort.

Beginner's guide to Mobile Responsive Design - http://www.studiopress.com/beginners-guide-responsive-design/

So, to summarize:

- Web applications only (no native apps)
- Node.js on the server
- Javascript/HTML on the client (web browser)
- Preferably mobile-responsive