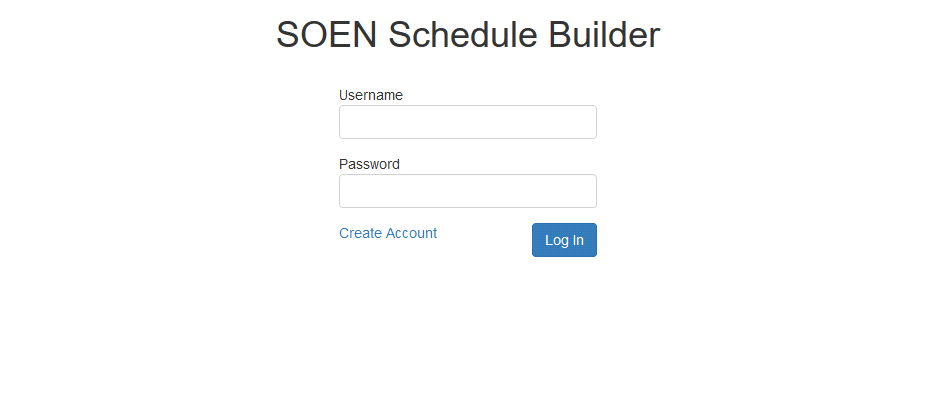
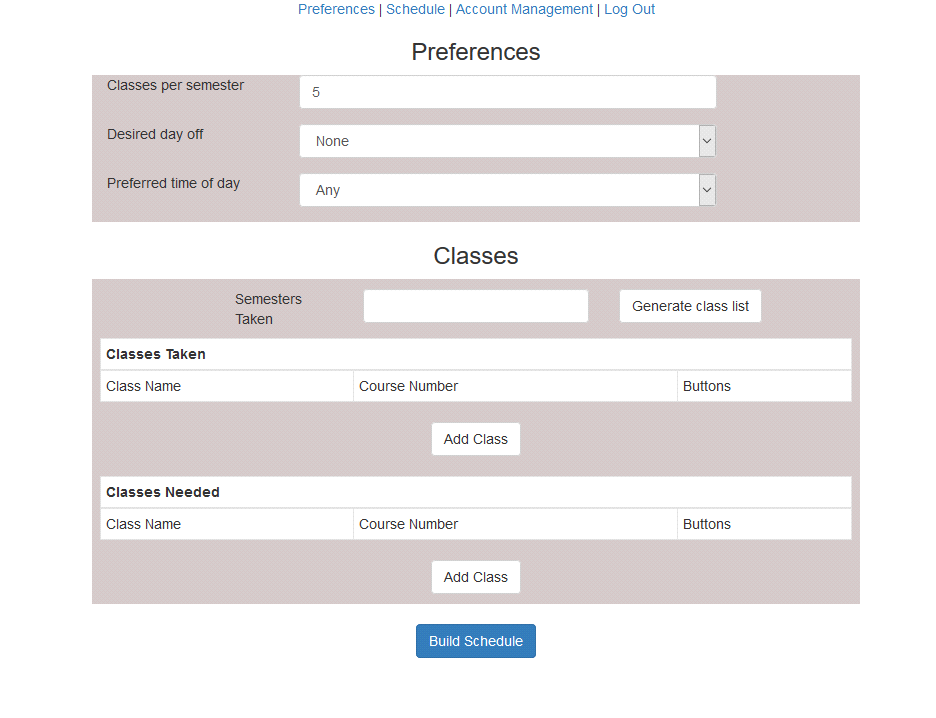
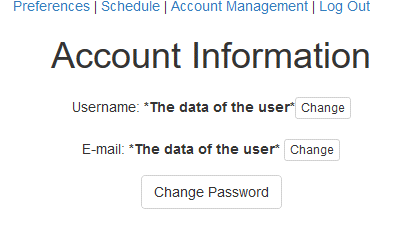
**7.1 Front-End Work**

All the major pages were completed for the prototype, except for the schedule page, which will be completed after the scheduling algorithm is in place

The login page



The preferences/classes page



The account page

The front-end work progressed as expected without any scope changes. React continues to be a simple and elegant solution for the front-end that only gets complicated when it comes to communicating with the back-end.

7.2 Back-End Work

For the prototype, the plan was to get a working interface that interacted with the database. As planned, one of the first steps was to design and set up the database that was to be used. A document explaining how to create and populate the Mysql database was also shared with the team. The second step was to work with the front-end team to link the set of pages with the databases through server calls. The register and the log in modules were working. However, they were not using the Laravel framework. Incorporating the current pages in the MVC model and the Laravel folder architecture as well as using the features of the framework (such as routing) were the last step of the prototype implementation. As a confirmation, the application was shared with the team and implemented on their end to test the modules that were implemented.

7.2.1 Front-end and back-end communication issues

There were a lot of issues with the usage of React and Laravel. React was initially designed for use with NodeJS, where the Javascript can be rendered on the server, but this is not an option when our back-end is running with a PHP framework. Since we are using React generate the views (this is operation is done client-side) and Laravel to handle the architecture, this makes it complicated for Laravel to manage the views for instance. While the original prototype did prove React renders the pages flawlessly, we had not yet implemented them with the framework and its architecture. Our initial prototype was merely testing log-in authorization and registration. Once Laravel came into play, this complicated the project. These issues did not affect our design decisions.

**7.3 Added Technology**

**7.3.1 Twitter Typeahead**

We have implemented Twitter’s typeahead component to improve the user experience. It is a jQuery open source text component that provides auto-completion suggestions as the user types (much like the google search bar). We will be using it to help the user add their needed and taken courses. The list of courses given in the auto-complete will be retrieved from the server. This will not only make a better user experience, but make it less likely for the user to input a course that doesn’t exist by accident.

