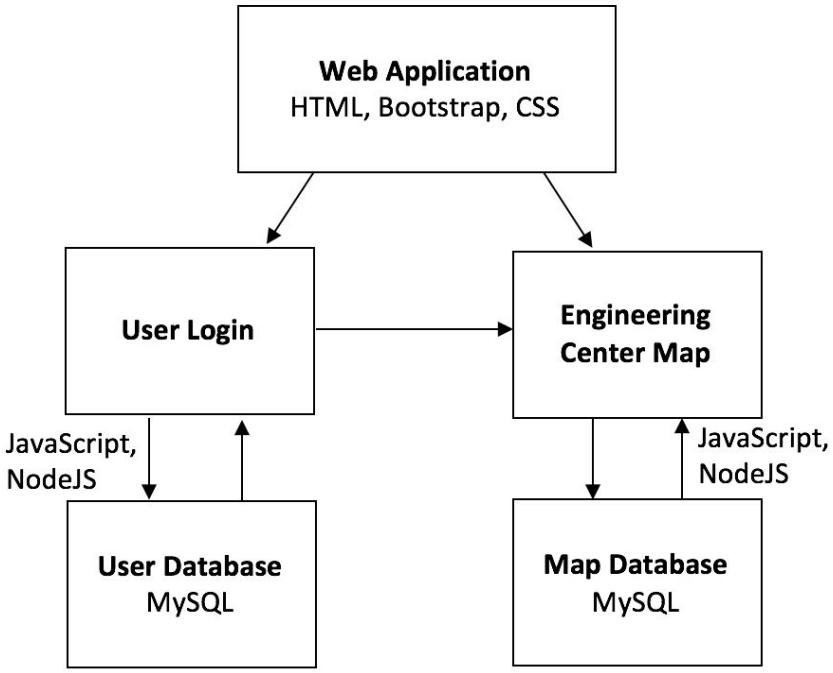
Team Number: Section 101 - Team 4
Team Name: International Justice League of Super Acquaintances (IJLSA)
Team Members: Taylor Ellis, Sam Fitzsimmons, Alyvia Hildebrand, Jason Hong, Steven McDonald, Robert Tennant

Revised List of Features: Our features have not changed dramatically from our initial list, but the priority of them have been updated as follows:

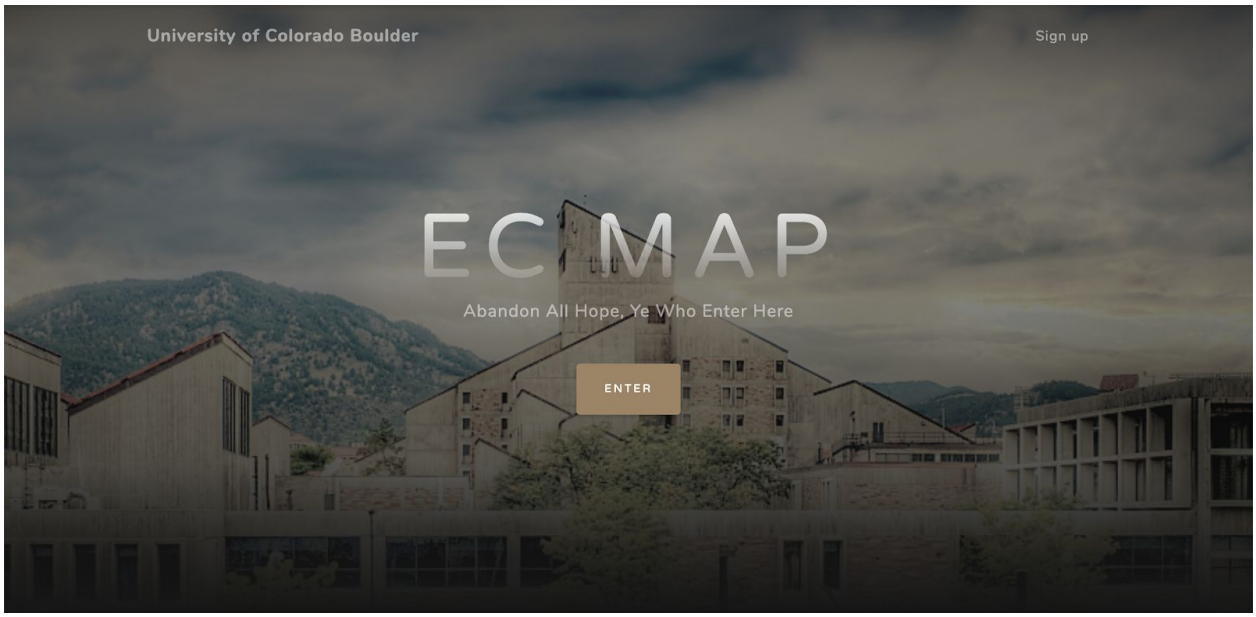
1. Classroom Locator: This is the key functionality for our application, and getting all the necessary components to work together is fundamental to the success of our project.
2. Navigable Map: Though our first priority is getting the functionality of the website setup, our next priority is to load in the actual graphics/data of the Engineering Center so that we can begin to test further features.
3. Room Legend: This is tied very closely with the two above features. Once we have the classroom database set up, it's simply a matter of implementing a filter function that returns all rooms of a given type.
4. Account: This is feature is required per our project description, and though it will be important to set up sooner rather than later, it is something that can be tackled after the website is up and running.
5. *New Feature:* Contact Us/Feedback page. Because we may not have enough time to test every single room, we thought it might be relevant to add a Feedback form such that users can send us a response in case they find a bug.
6. Taggable Locations: This feature does not seem inherently difficult to implement, but there are certain questions that arise when implementing this system that will require our attention (how long do the tags last, do they need to be moderated, where do we store the user tags, etc.). It makes more sense to focus on the key features first and save this one for later.
7. Classroom routing: While we are motivated to implement this feature, we know that this will take a dramatic level of work and effort to do properly. We believe that it still makes the most sense to focus on building the base application and then constructing the router afterwards so that we are guaranteed to have a finished product.

Architecture Diagram:

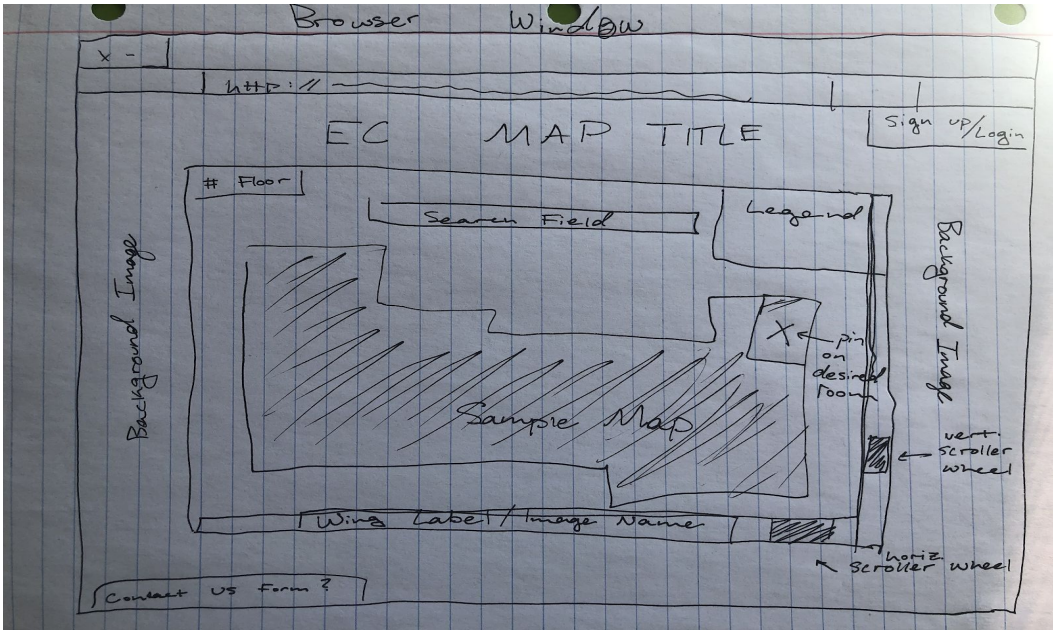


Front End Design:

Here is a sample image of our intended homepage. We will use similar HTML, CSS, Bootstrap, and Javascript elements across our website to maintain a similar theme and look.



We intend to have a Javascript applet pop-up upon clicking “Enter,” within which will be the primary window for navigating the map. Within this app, we will have fields for searching by room number and by wing, as well as our legend. The displayed map will allow for zooming and panning/scrolling so that users can have greater control over the map view. A rough sketch of the application’s main view in a web browser’s window is included below.



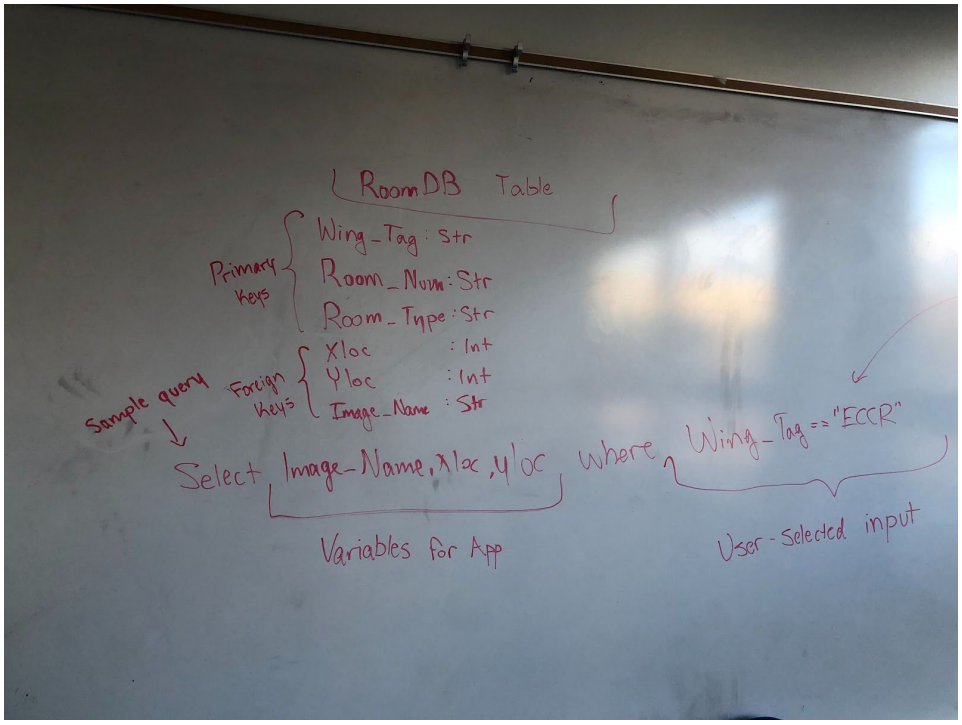
Web Service Design:

Our primary web hosting service is Reclaim Hosting, a service designed specifically for students and educators. This service is particularly useful, as it integrates directly with several of our other architectural components. For example, it supports Git for version control and Remote MySQL for accessing our database remotely. We have already secured our domain and are using this service as our primary means of joining front-end to back-end.

Currently we do not have plans for utilizing any other APIs, though we may look into using a Google Maps API (or something of the sort) as we make progress on our classroom routing feature.

Back End Design

Our project will primarily use Remote MySQL for its backend database storage. Our “Room Database” stores all the information for our map. This includes the wing_tag (ECCR, ITLL, etc.), the room_num, and the room_type (bathroom, lecture hall, etc.). These are all stored as strings and are our primary keys for user-input searching. Our returning/foreign keys are the pixel integer locations of the room (xloc and yloc) and the respective image_name for which map to bring up. A sample query can be seen in the figure below.



Here is a more directional relationship between our application and our back-end items. Most of our interactions between the application and the databases are facilitated by Node.js functions.