# Multitask Overlay

### Design Review for Code Shack

### 2019-03-07, version 1.0

## Document Revision History

**Rev. 1.0 2019-03-07: Initial Version**

## Summary

One of the main criticisms focuses on the system architecture. The model and controller components doesn’t accurately represent the model and controller in an MVC. Another focus in the review is how your program is able to access specific information such as courses and an active wisc email. The team should also revisit how they are testing their program as in what tools will be used to create these tests.

## Questions

1. Are there any inconsistencies in the design? That is, does the document contradict itself?

From our understanding, their project is based off of letting UW-Madison students access their project, but later on in the document they have a task to create a sign-up page. This task is to allow a user to create an account based off of their UW Google Account, is this not the same as logging in using their UW login?

1. Are there omissions in the design? That is, are there elements that are mentioned but never discussed, or obvious pieces that are missing?

The User profile information is mentioned with regards to the database, including courses, history, and favorite company but there are no methods, fields, or forms specified for how this information is set. The only field mentioned is email which is verified by firebase for signing up and signing in.

1. Are any parts of the design unclear? The standard should be that given the design document, a competent programmer can code the project. Note that “clear” does not mean that the document must be very detailed. We assume that a decent computer scientist can fill in missing details, provided the overall document is clear enough.

They mention using Redux to manage the state of their application at the System Architecture section. There isn’t any further explanation of what Redux exactly is. Redux is mentioned again in the Design Details section for their Frontend side of the software and they simply reiterate that Redux will be used to manage the state of their application with no further explanation of what Redux is. The other components for their Frontend are explained for what they are and what they are used for. Task 9 for their implementation plan mentions setting up Redux store, actions, and reducers, however, at this point we don’t know what Redux exactly is and we don’t know what Redux store, actions, and reducers specifically consist of.

1. Are there technical errors in the design? Is there any statement of fact that you know is false?

The email validation “algorithm” is really just one line of code. It can be implemented with the following snippet:

Const valid = str.match(/@wisc.edu$/) ? ‘yay’ : (function () {throw ‘☹’});

1. Has thought been given to testing? How would you test this design? What, if anything, could be done to make the design easier to test?

The testing could be better organized and more focused. Tools to aid in testing are hardly mentioned. Testing in the document is broken down per component, but not by type of test. The testing should be divided into unit testing, integration testing, and system testing. Think about what frameworks and libraries can be used for these types of tests. For example, Mocha is commonly used for structuring unit tests. Restify was mentioned, so that could be used to help set up integration tests. Finally, it will be necessary to utilize a UI automation framework in order to do system testing. Cucumber, coupled with WebdriverIO for UI automation, is a good choice for system testing. Cucumber allows tests to be behavior driven and are written out in language that most people can understand.

1. Does the design make realistic assumptions about the environment? That is, will the team have trouble getting access to important external components (e.g., specialized hardware) and are the systems the project needs to interact with suited to the purpose?

The design does not specify how the data from the UW System will be obtained, specifically course information. It also does not seem to include connecting to an external server, assuming that is how they are planning to access the UW System, they will need to incorporate a section that interprets the data from the UW System server.

1. Does the plan seem realistic? Are tasks at a reasonable level of granularity and is it clear what each task means? Do the time estimates seem appropriate? Do any parts of the plan seem risky in the sense that they are likely to become a bottleneck to further progress?

The plan as a whole seems realistic, with most tasks having a reasonable time estimates. The main problem is obtaining data from the UW System, being able to access the users course lists, (i.e such as current courses) seems like a risky task that will lead to a bottleneck. They do not mention how they are going to receive the permissions to access a user’s information in the document, and making progress in this project is going to rely on being able to access this information.

1. Any other comments?

The System Architecture diagram provided does not accurately reflect a Model View Controller architecture. Although implementation may vary, this diagram seems unnecessarily complicated. The Controller should be the medium tying the Model and the View together. In this diagram, the Controller does not directly interface with the View. It would be better to call the backend API from the Controller, while still interfacing with the View and the Model. The Dispatcher, Store, and Action Creator are also parts that could come under the umbrella of the Controller. If the architecture was viewed in this way it would be more maintainable. The Controller should be handling and orchestrating these different components.

When looking through the distribution of tasks, it seems some people have more units for tasks than others which means more work is expected from others. Also, it seems that some people aren’t included in all iterations. It may not be wise for someone to not look at the program code for that long.