

CS 320 Course Project Final Report

for

Keyswitch Lookup Application

Prepared by

Group Name: The Fellowship of the Keys

Elliott Long
Jonathan Meisner
Joseph Boothby
Kyle Stoneberg

11686651
11520302
01445002
011700532

elliott.long@wsu.edu
jonathan.meisner@wsu.edu
joseph.boothby@wsu.edu
k.stoneberg@wsu.edu

Date: 12/16/2020

CONTENTS	ii
1 INTRODUCTION	1
1.1 PROJECT OVERVIEW	1
1.2 DEFINITIONS, ACRONYMS AND ABBREVIATIONS	1
1.3 REFERENCES AND ACKNOWLEDGMENTS	2
2 DESIGN	3
2.1 SYSTEM MODELING	3
2.2 INTERFACE DESIGN	6
3 IMPLEMENTATION	9
3.1 DEVELOPMENT ENVIRONMENT	9
3.2 TASK DISTRIBUTION	9
3.3 CHALLENGES	9
4 TESTING	10
4.1 TESTING PLAN	10
4.2 TESTS FOR FUNCTIONAL REQUIREMENTS	10
4.3 TESTS FOR NON-FUNCTIONAL REQUIREMENTS	10
4.4 HARDWARE AND SOFTWARE REQUIREMENTS	10
5 ANALYSIS	11
6 CONCLUSION	12
APPENDIX A - GROUP LOG	13

1 Introduction

In the world of computer components, one of the most paramount and ubiquitous of them all is that of the keyboard. And within this world are numerous different varieties, from dome and optical, to that of the mechanical switch. Further still, within the world of the mechanical switch, there are innumerable variants, a somewhat daunting aspect that can leave users searching out mechanical keyboards with specific needs in mind confused and annoyed due to the lack of easy-to-parse sites of information. The project Fellowship of the Keys aims to eliminate that issue by providing a simple to use site in order to parse through differing varieties of keys, all depending on desired qualities, be that audibility of clicks, actuation strength required, or otherwise.

1.1 Project Overview

The keyswitch lookup application exists to fill a niche that is currently unfilled: the ability to organize and search through mechanical keyboard switches based on various criteria. Though other resources exist to organize switches in various ways, there did not, until the existence of this application, exist an easy way to quickly organize by attributes, only by switch type themselves, which requires prerequisite information on the attributes of said switches. For example, if you wanted to see a list of switches in descending order of how strong the actuation pressure required was, this would be impossible. It is, with the creation of this application, no longer impossible.

As individuals who own and regularly use mechanical keyboards, it was imperative that the lack of filters like this be rectified, leading us to choose this as a topic. In short, it was something we were passionate about, and a project that could solve an existing problem that had not been previously solved, rather than provide another version of an already existing solution. This web application was developed utilizing a combination of HTML, CSS, and JavaScript, which allowed us to both make an easy-to-parse application, but one that could be dynamically managed as well, with the presence of an administrative mode acting as a way to add and remove switches.

1.2 Definitions, Acronyms and Abbreviations

Admin - Administrator

Clicky - Loudly audible

CSS - Cascading Style Sheets

Dome Switches - A switch that uses a depressed rubber dome to register a keypress

HTML - HyperText Markup Language

Mechanical Keyboard - A keyboard featuring mechanical switches

Mechanical Switches - A switch that uses a mechanical mechanism to register a keypress

Modern - Maintained and updated from 2018 and on

MX Cherry - The leading manufacturer of mechanical keyboard switches

Optical Switches - Switches that uses a visual sensor to register a keypress

SDD - Software Design Document

SRS - Software Requirements Specification

Switch - Mechanical switch

UML - Unified Modeling Language

1.3 References and Acknowledgments

- [1] "The Ultimate Mechanical Keyboard Catalog" *mechanicalkeyboards.com* [Online]. Available: <https://www.mechanicalkeyboards.com>. [Accessed 11/01/20].

2 Design

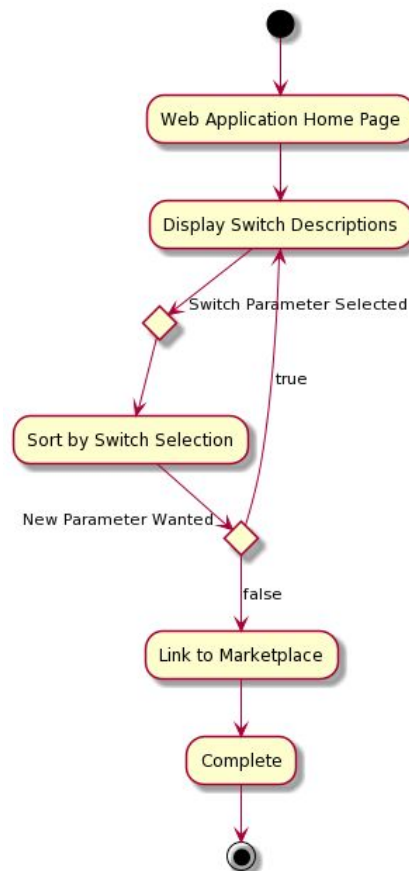
2.1 System Modeling

While our program follows the general design of the diagrams from milestone 2, some slight modifications have been made:

Activity Diagram - User Case:

This activity diagram covers the typical user use case of researching mechanical key switches, and then purchasing a keyboard. The user accesses the web application, then looks through different switch types until they decide on a specific switch. At this point, the user chooses if they would like to view keyboards sorted by these switch characteristics. Finally, they are provided with a link to the trusted marketplace that displays the correct keyboards and the correct pricing order. The start state is accessing our website, and the end state is being redirected to the marketplace.

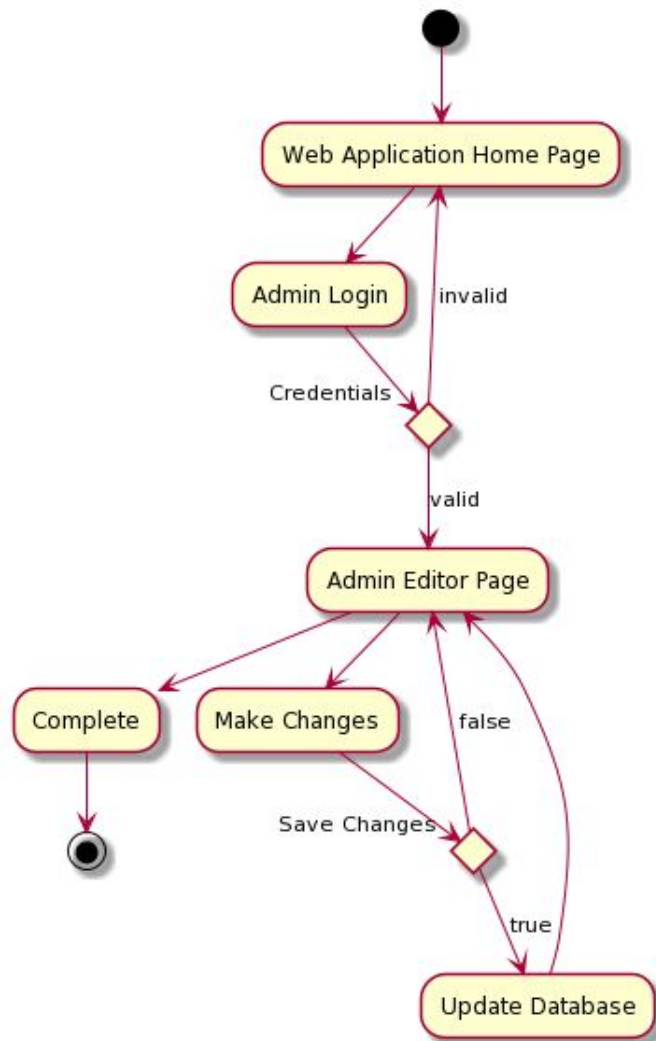
Fellowship of the Keys
Activity Diagram



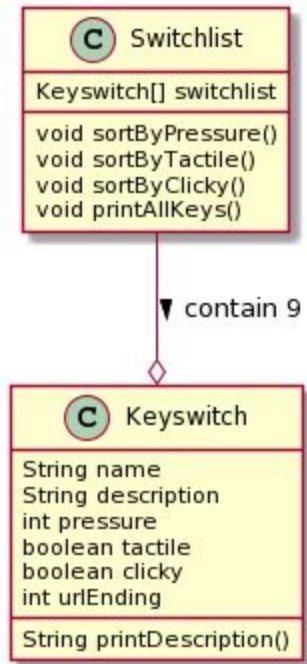
Activity Diagram - Admin Case:

This activity diagram covers the admin use case of our webapp. The administrator accesses the website home page, then logs in to an admin account via a username/password combination. This admin function allows the administrator to manipulate the data that is stored in the Switchlist and Keyswitch classes in order to add or remove Keyswitches and change the marketplace URL for the hyperlinks. The start state is accessing the homepage of the website, and the end state is successfully modifying the web app.

Fellowship of the Keys Activity Diagram



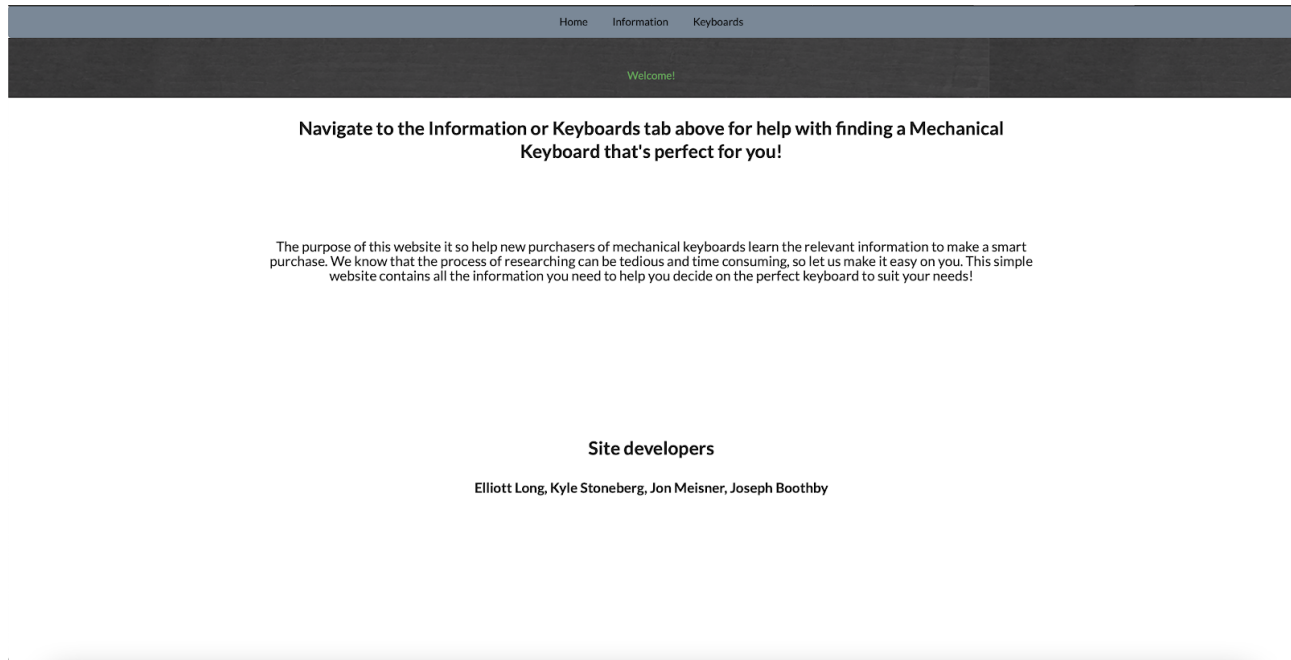
Class Diagram:



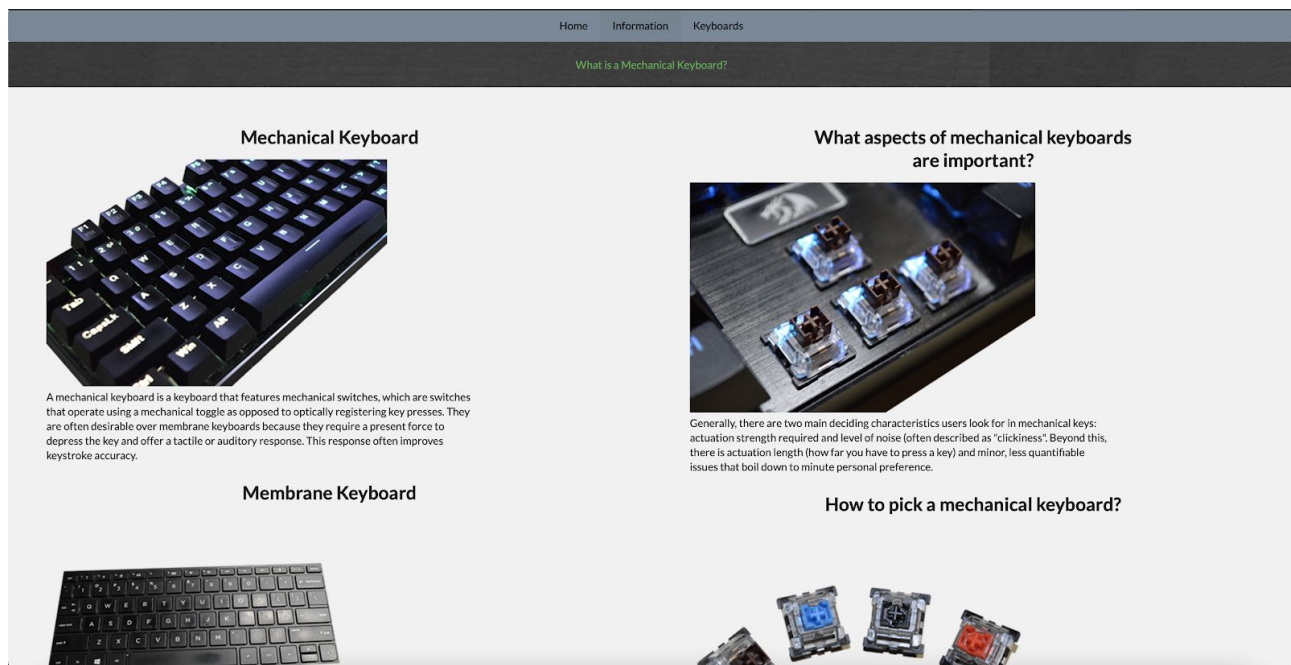
Switchlist: A class that contains all instances of **Keyswitch** and methods for sorting and printing. There will be one master instance of the **Switchlist** object, and it will be responsible for sorting the array of **Keyswitch** objects based on the attributes of the **Keyswitches**. The **printAllKeys** method of **Switchlist** will display the **Keyswitches** in the newly sorted order.

Keyswitch: A class that defines the attributes of a switch. These will be immutable objects hardcoded into the JavaScript file. Each **Keyswitch** is responsible for displaying its own attributes using the **printDescription** method.

2.2 Interface Design




Landing Page




Information Page (1/2)

How to pick a mechanical keyboard




Most computer and laptop keyboards fall into the membrane keyboard category. Under the key caps are small rubber domes. When the key is pressed, the rubber dome collapses and presses on a plastic sheet with small electrical connections. These connections then press against a base sheet with matching connections. When the two make contact, a key press is registered. This results in a squishy feeling with no tactile response.



The way to choose a mechanical keyboard is by what switch has the qualities you desire. Click [here](#) to get to our informative list of switch types.

Courtesy of
The Fellowship of the Keys



Information Page (2/2)

[Home](#)
[Information](#)
[Keyboards](#)

Looking for a Keyboard?
Check out our list below!

Sort By Attribute

Sort By

Name	Description	Pressure	Clicky	Tactile	URL
MX Black	Powerful and Direct. A linear switch with increased actuation pressure	60	No	No	Buy Here
MX Blue	Clicky and Noticeable. A Tactile switch that provides audible acoustic feedback	60	Yes	Yes	Buy Here
MX Brown	Focused and Noticeable. A Tactile switch with noticeable feedback	55	No	Yes	Buy Here
MX Clear	Focused, Powerful, and Noticeable. A Tactile switch with no click, and a sister model to the MX Brown and MX Grey	65	No	Yes	Buy Here
MX Gray	Focused, Robust, and Noticeable. A slightly modified version of the MX Brown. Offers increased actuation pressure	80	No	Yes	Buy Here
MX Green	Clicky, Robust, and Noticeable. A slightly modified version of the MX Blue. Offers increased actuation pressure	80	Yes	Yes	Buy Here
MX Red	Smooth and Direct. A linear switch with low actuation pressure. A good first choice for beginners	45	No	No	Buy Here
MX Silent Red	Smooth and Silent. A modified version of the MX red that uses damping to reliably minimize noise.	45	No	No	Buy Here
MX Speed Silver	Fast and Direct. The fastest model of CHERRY MX in full height. Linear with low actuation pressure	45	No	No	Buy Here

Keyboards Page (1/2)



Name	Description	Pressure	Clicky	Tactile	URL
MX Black	Powerful and Direct. A linear switch with increased actuation pressure	60	No	No	Buy Here
MX Blue	Clicky and Noticeable. A Tactile switch that provides audible acoustic feedback	60	Yes	Yes	Buy Here
MX Brown	Focused and Noticeable. A Tactile switch with noticeable feedback	55	No	Yes	Buy Here
MX Clear	Focused, Powerful, and Noticeable. A Tactile switch with no click, and a sister model to the MX Brown and MX Grey	65	No	Yes	Buy Here
MX Gray	Focused, Robust, and Noticeable. A slightly modified version of the MX Brown. Offers increased actuation pressure	80	No	Yes	Buy Here
MX Green	Clicky, Robust, and Noticeable. A slightly modified version of the MX Blue. Offers increased actuation pressure	80	Yes	Yes	Buy Here
MX Red	Smooth and Direct. A linear switch with low actuation pressure. A good first choice for beginners	45	No	No	Buy Here
MX Silent Red	Smooth and Silent. A modified version of the MX red that uses damping to reliably minimize noise.	45	No	No	Buy Here
MX Speed Silver	Fast and Direct. The fastest model of CHERRY MX in full height. Linear with low actuation pressure	45	No	No	Buy Here

Keyboards Page (2/2)

This is the admin page

Please enter the data you would like to add to the database

<input type="text" value="Switch Name"/>	<input type="text" value="Description"/>	<input type="text" value="Pressure"/>	<input type="text" value="Tactile(Yes/No)"/>	<input type="text" value="Clicky(Yes/No)"/>	<input type="text" value="URL"/>	<input type="button" value="Submit"/>
--	--	---------------------------------------	--	---	----------------------------------	---------------------------------------

Please enter the name of the item you would like to remove from the database

<input type="text" value="Name to Remove"/>	<input type="button" value="Remove"/>
---	---------------------------------------

Current Database Contents

Name	Description	Pressure	Clicky	Tactile	URL
MX Black	Powerful and Direct. A linear switch with increased actuation pressure	60	No	No	Buy Here
MX Blue	Clicky and Noticeable. A Tactile switch that provides audible acoustic feedback	60	Yes	Yes	Buy Here
MX Brown	Focused and Noticeable. A Tactile switch with noticeable feedback	55	No	Yes	Buy Here
MX Clear	Focused, Powerful, and Noticeable. A Tactile switch with no click, and a sister model to the MX Brown and MX Grey	65	No	Yes	Buy Here
MX Gray	Focused, Robust, and Noticeable. A slightly modified version of the MX Brown. Offers increased actuation pressure	80	No	Yes	Buy Here
MX Green	Clicky, Robust, and Noticeable. A slightly modified version of the MX Blue. Offers increased actuation pressure	80	Yes	Yes	Buy Here
MX Red	Smooth and Direct. A linear switch with low actuation pressure. A good first choice for beginners	45	No	No	Buy Here
MX Silent Red	Smooth and Silent. A modified version of the MX red that uses damping to reliably minimize noise.	45	No	No	Buy Here
MX Speed Silver	Fast and Direct. The fastest model of CHERRY MX in full height. Linear with low actuation pressure	45	No	No	Buy Here

Admin Page

3 Implementation

3.1 Development Environment

This project was completed using the IntelliJ IDE and implements HTML, CSS, and JavaScript for languages. Further, git was used for version control and GitHub was used for code hosting and collaboration.

3.2 Task Distribution

The work was more or less a group effort, though special note should be made to Joseph Boothby for the structural design of the data structure holding the keys and the dynamic “adding-of-key” functionality in the admin page, Jonathan Meisner for the overall project leading in regards to front-end development information (being that he was the most senior in regards to front-end knowledge), and to Elliott Long for his research in the information page and the meticulous checking of structure, syntax, and diction/spelling in both the code and the documents presented (SRS, SDD, and this document).

3.3 Challenges

The natural challenges that arose from this were that of balancing this project along with others, both in this class and in other classes. Further, the ongoing pandemic (COVID-19) created issue in the lack of ability to meet in person. Finally, though it ended up being of no consequence, the organization of time was paramount and was by definition a challenge.

4 Testing

The testing of the project went off without a hitch and, as such, a working product was verified to have been produced.

4.1 Testing Plan

To Test:

- All pages load and are accurate
- The database loads
- Admin login works
- The admin page functions in both adding and removing switches
- That each page and function loads in a reasonable (below one second) time frame

4.2 Tests for Functional Requirements

Tested:

- Database loads (PASSED)
- Landing page loads (PASSED)
- Information page loads (PASSED)
- Keyboards page loads (PASSED)
- Admin login works (PASSED)
- Admin page can add a switch (PASSED)
- Admin page can remove a switch (PASSED)

4.3 Tests for Non-functional Requirements

Tested:

- Landing page loads in <1 second (PASSED)
- Information page loads in <1 second (PASSED)
- Keyboards page loads in <1 second (PASSED)
- Admin login works in <1 second (PASSED)
- Admin page can add a switch in <1 second (PASSED)
- Admin page can remove a switch in <1 second (PASSED)

4.4 Hardware and Software Requirements

The hardware requirements used were a computer of modern (2015 or later) parts and a web browser. Notably, the app works on both MacOS and Windows (utilizing Big Sur and Windows 10, respectively), and both Safari and Chrome (the most modern versions of both used).

5 Analysis

In looking at the project in terms of milestones, the effort can be broken down thusly:

Milestone 1:

This was an entirely collaborative effort done by all members of the team simultaneously on a single document. As such, it can be considered a perfectly even split of effort on everyone, and was worked on for approximately 3 hours.

Milestone 2:

Milestone 2 was spearheaded by Joseph Boothby in a major way, with the UML diagrams being entirely created by him. These were looked over and approved by the remaining group members to verify accuracy. The split would be Joseph at 2 hours, and the remaining members at 1 hour.

Milestone 3:

This milestone was also spearheaded by Joseph, though it was still a collaborative effort on the part of the group. First and foremost, Jonathan Meisner acted as a project lead as he has the most front-end experience, and provided ample information on how to do various things, notably linking HTML, CSS, and JavaScript in a way that made them work well with one another. The majority of the code itself was written by Joseph. Elliott Long took the lead on testing the workability of the entire app after writing, correcting some minor diction and spelling mistakes, writing the final report, and fixing the SRS and SDD to match the final product. The estimated split would be Joseph Boothby at 5 hours, Jonathan Meisner at 5 hours, and Elliott Long at 5 hours.

The most difficult aspect of the project (as evident by the split in hours worked, making up roughly 47% of the total time worked) was milestone 3. This is an easy to understand thing, as the final section involved both writing the code and editing the various documents. It, by necessity, took the most effort, but produced the most overall change in what was completed.

6 Conclusion

The project worked as intended in regards to simulating a (admittedly low stress) collaborative work environment with peers in order to produce a cohesive end product. The deadlines forced collaborative efforts on sections and setting realistic timetables in order to accomplish the required goals. Further, the design aspects allowed for the free flow of ideas to be involved, further facilitating creativity and thoughtfulness in what was being created and why. This served to show the importance of not only coming up with an idea of an application to code, but to also think of why we are making it in the first place. This, more so than any other project in our computer science careers, created a realistic experience on a real-world coding situation in which we would be tasked with coming up with creative solutions to solve existing problems. In short, this project did not only teach us how to code, but also how to programmatically solve problems and, thus, think as computer scientists.

Appendix A - Group Log

To most efficiently explain the group log, a copy will be placed here with the understanding that constant communication via the messaging application Discord was also present.

9/30:

Catalog build to be object oriented javascript or java with additional modules or packages to be determined

Git Repo (keyboard-webapp) Created & Invited Group Members

10/2:

First Team Meeting

Team Arrangment Documentation Filled out.

Jonathan Meisner appointed as Initial Project Lead

Submitted Document via BB ~4:45pm 10/2/2020

10/26:

Met With Team

Discussed structure of project

~1.25 hour meeting time

11/4:

Met With Team

11/20:

Put together the Software Design Document with team via Zoom

Updated this README with the updated status of the overall project

~ 1.5 hour meeting time