Hi, this is Daniel Nelsen. My project is an SSH Organizer. The program is able to read a user’s SSH config file and then create a list of hosts that the user can connect to. When the program is start, the hosts are presented to the user and the user can choose which host to connect to. A terminal window will be opened and a connection will be made to the host the user chooses. Finally, the user is able to add new hosts to the SSH file. Each menu of the program also contains a help menu to aid the user if they have questions on a particular portion of the program.

The display of the SSH hosts is data from the microservice. You can see here how entering the SSH hosts menu causes a signal to the microservice to read, parse and send the SSH hosts information back to the main program.

I will demonstrate opening an SSH session with a host.

And I will demonstrate adding an SSH host to the SSH config file.

Inclusivity Heuristics

1. Benefits of using new and existing features – ***When the program is started the user is presented with an explanation of the program and how to interact with it.***
2. Costs of using new and existing features - ***The program indicates what the options will do on the main screen. This helps inform the user what will happen if the button is pushed.***
3. Let people gather as much information as they want, and no more than they want – ***The program has an option for a help window to be displayed. This allows the user to read or not read the help file.***
4. Keep familiar features available – ***The program has a familiar interface on each screen. For example, the option to quit or go back to the main screen are always the last option. This way the user knows where to look each time they need to quit or go to the main screen.***
5. Make undo/redo and backtracking available – ***The program offers a confirmation prompt when adding an SSH Host. This way the user can be certain that they have made the correct additions for a new host.***
6. Provide an explicit path through the task – ***The main screen indicates in the options menu what each of the options are for. This way the user is able to determine which option to choose for a given task.***
7. Provide ways to try out different approaches – ***The program has a built-in help file. This way the user can use the help menus if something isn’t working for them.***
8. Encourage tinkerers to tinker mindfully – ***The program offers a confirmation prompt when adding an SSH Host. This way the user is able to back out if they made a mistake and need to re-enter information for the new host.***

Show a quality attribute.

One Quality Attribute of the program is interoperability. The program is able to be ran on both Windows and Linux operating systems. Show Code line 166. The operating system is sent to the microservice which then can find the correct location of the user’s SSH config to parse and send back to the main program. Also when writing to the SSH file the code is able to write to the correct location based on either operating system.

Self-reflection:

Big Ideas.

1. For the first big idea, I would point out the inclusivity heuristics. At work I’ve put together a few programs that my team uses to interact with the pharmacy information systems. When I’ve been working on these programs I didn’t know about the inclusivity heuristics, but found through trial and error to include such things as a help file or an explicit path in the program. Now that I have a better understanding of these principles, I can make these programs more user-friendly and inclusive to the various learning and user styles that are present on my team.
2. The second big idea is code smells. Writing this project and the programs at work. I look back at the programs that I’ve written for work and think I wished I had known about code smells. Now that I’ve written about 8,000 lines of code for programs at work, I kind of dread having to go back through and refactor all this work. Starting the good habits of clean code would have prevented a large technology debt that I’ll have to work on over the next weeks.
3. The last big idea is the Agile software development. I’ve heard about this style of development before, but didn’t look into its actual meaning. When I first heard about sprints, I thought it was another way for the company to make the engineers be more productive and cost effective. However, after studying this during this course, I can now appreciate that Agile is more of a methodology for developing sustainable software than making the engineers work harder with less resources.

Two actions I plan to embrace from this course are code smells and using diagrams. I really appreciated the exercise to use diagrams to mock up the program at the beginning of the program. At first, I thought it was a waste of my time, but as I started actual work on my project I constantly referred back to the mockups for direction as the project progressed along. Finally, like I stated above, I need to really spend time reducing code smells. I feel that I need to come up with a system and really stick to the style that was started. My work programs are all over the board with different styles. I feel with time and practice, I’ll be able generate something that works for me and that is used as a standard across the software industry.

One major insight from this course is the use of project management resources. For example, Asana versus Hive versus Taskade. I found not all the systems are built the same. Hive wouldn’t resend a reset password after I forgot to put my password in my password manager. So after spending 30 minutes trying to reset my password, I just moved on. After evaluating three different systems, I see the importance of using a system that is reliable because I don’t a project system to cause development be become mired bug fixing the project system rather than working on the my software projects.