**The following instructions can be used to build Edward Johnson's CS499 Final Project locally.**

I recommend that you use the Docker build/instruction path, as that will handle any dependencies that the project relies on, you will not need to install any additional packages. I say this because I found that at least for my primary computer, running a Debian environment, I needed to have additional dependencies such as "pkg-config" installed in order to build the project from source using the standard local build command for rust "cargo build." If you are on Linux, cargo will let you know if you are missing a dependency, and it is as simple as just executing an install command using your distribution’s package manager, ex: apt install pkg-config. Windows systems should not require any additional steps beyond what is described below.

**BUILDING WITH DOCKER:**

Whether you are on Windows, or on Linux, the first step for this particular path is to install docker.

**LINUX DOCKER:**

For Linux, the most straight forward way is to use the package manager for your distribution, such as apt for Debian/Ubuntu. That command would look like "apt install docker", if you have any issues building the docker file on Linux with just standard docker, I would recommend adding "apt install docker-compose."

The official docker page for Linux is where I would recommend looking to find the install command for a specific distribution. That can be found here: https://docs.docker.com/desktop/install/linux-install/

**WINDOWS DOCKER:**

For windows, you will need to download a recent version of docker, and then install it. This can be done using Docker's web page here: https://docs.docker.com/desktop/install/windows-install/

Docker will ask you to log out / reboot if you are on windows, do so, then proceed on to building.

**BUILDING:**

The next step, is to download the source directory.

Extract the directory, open up a console / terminal emulator of your choice, and then cd into the extracted project directory.

Once you are in the project directory, for example '/home/user/Documents/SNHU/C4/CS499/FinalProject', you will then want to execute the following commands in the terminal, while you are in the project directory.

To build the image, run the following command. NOTE: on Linux, you may need to prepend a 'sudo' to each of these commands before execution if you are given an error regarding permissions

```

docker build -t final\_project .

```

Once the image has finished building, to start the container / application, use the following command:

```

docker run -it --rm final\_project

```

**BUILDING THE PROJECT LOCALLY WITH RUST / CARGO:**

The first step will be to download and install rust for your system.

**For LINUX/OSX:**

This installation is as simple as executing the rustup command from within your terminal:

The command can be found here:

https://www.rust-lang.org/tools/install

and it includes the following curl:

```

curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh

```

**For WINDOWS**

The command is similarly easy, download the rust installer from the following page:

https://forge.rust-lang.org/infra/other-installation-methods.html

A direct link to the x86\_64 version can be found below.

[x86\_64-pc-windows-msvc](https://static.rust-lang.org/rustup/dist/x86\_64-pc-windows-msvc/rustup-init.exe)

I would recommend using the default installation for both the Linux/OSX and Windows builds. Non-standard installations have not been tested, so there is no guarantee they will operate the same.

**BUILDING:**

Once you have finished the installation, open up your selected console / terminal emulator.

cd into the main project directory wherever it is downloaded to, ex: 'cd /home/user/Documents/SNHU/C4/CS499/FinalProject'.

The following commands are used to build the project on both Windows, and Linux.

To begin the building process, execute:

```

cargo build

```

When it is finished, you may run the built executable by then running:

```

cargo run

```

You may also directly run the executable that was built in the project directory, but I recommend allowing cargo to handle it.

*--- PROCEED TO DOCUMENTATION OR OPERATING THE APPLICATION*

**DOCUMENTATION:**

If you would like to build the documentation website / pages for the project, execute the following command while cd'd into the main directory for the project ex: '/home/user/Documents/SNHU/C4/CS499/FinalProject'

```

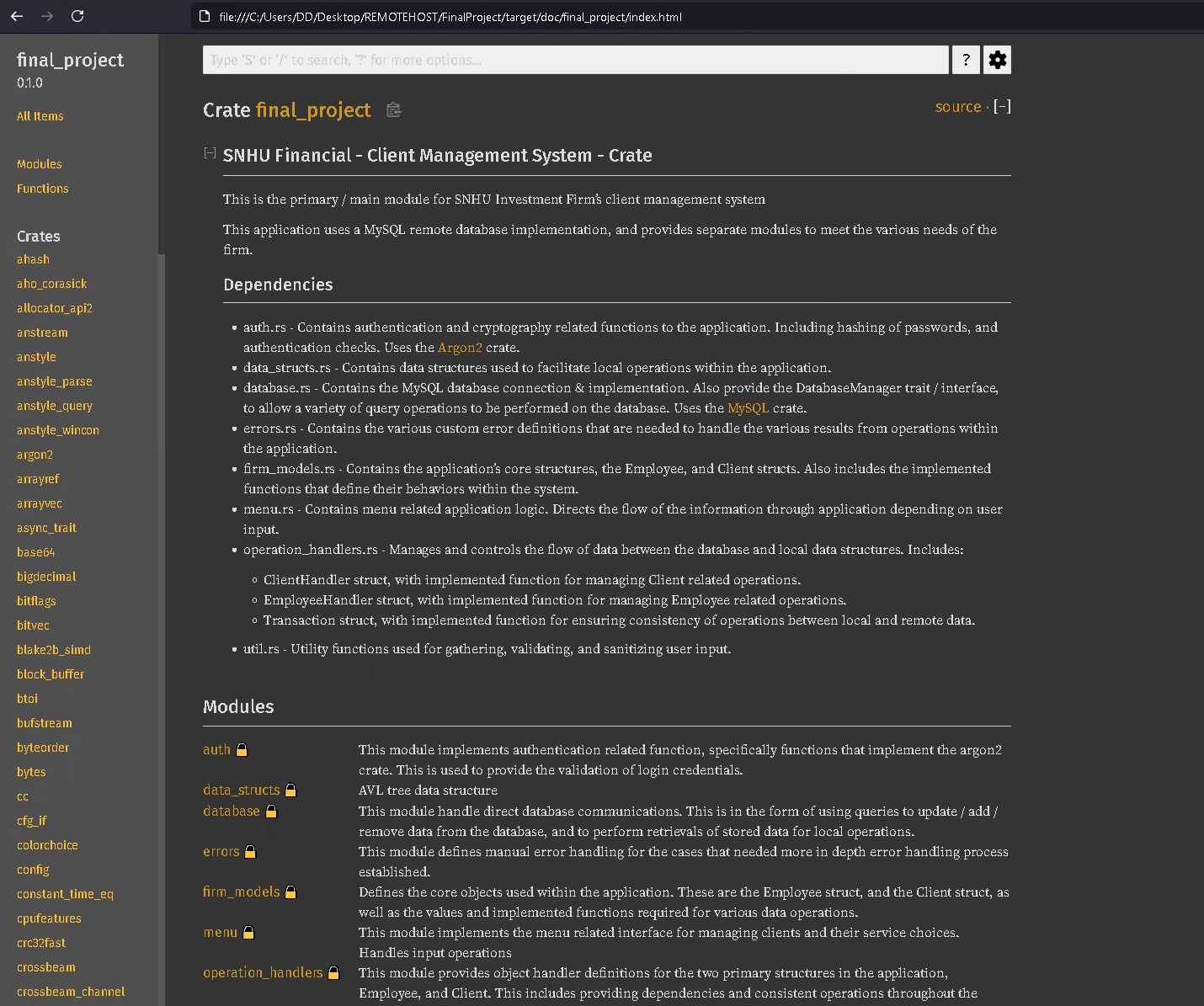
cargo doc

```

On the windows environment I used to test the cross-platform functionality, the following location is where the index page of the project's documentation wikipedia style html-docs are located. You can also open any of the generated document html pages, and use the search bar, along with one of the trait, enum, strucs, etc from the project, and it will provide you with a description as to what the application component is used for, and what options you have when working with it.

'file:///C:/Users/USERACCOUNT/Desktop/FinalProject/target/doc/final\_project/index.html'

*An example of these documentation pages for this project can be seen on the following page.*

****

**OPERATING THE APPLICATION:**

**FROM THE LOGIN SCREEN:**

The application will then connect to the login screen, at which time you will be able to provide an employee ID and the specific password for the employee of your choice. I have not implemented any user control policies for specific accounts, so they will all have the same access permissions, and will be able to work with the data.

The list of available name and passwords that can be used to log into the system are provided in the main project directory. Contained within the file auth-data.csv.

There are 10 accounts / employees that have client lists assigned to them. Use your selected employee id integer (1-10), and the corresponding password for that id in order to log into the system.

**USING THE SYSTEM:**

The application is currently limited to the following actions / interactions:

You may display a list of all clients assigned to a specific employee, by providing their employee\_id,

You may perform modifications to the services that a client is receiving by providing their client\_id,

You may perform modifications to the employee pairing for a client by providing their client\_id and the desired employee’s id value for which to switch the pairing to,

You may exit the application.

**NOTE:**

The back end for additional functionality has been implemented, and is working. It is what I used to seed the database with Client/Employee data. These options include adding/removing employee & clients, changing additional client/employee values. But there are currently no menu options available for many of these operations, as I have not had the time to design / implement a user account control system.