User Manual

Version 1.3
26 April 2021

Team DigiFolio
Dr. Andy Wang
Fabio Santos

Burbank, Logan(lead)
Braudaway, Jackson
Chen, Kailin
Marschel, Parker
Yang, Zhenyu

Table of Contents

1. Introduction **2**

2. Installation **3**

3. Configuration and Daily Operation **5**

4. Maintenance **7**

5. Troubleshooting **14**

6. Conclusion **16**

1. Introduction

We are happy that you have chosen to use the Engineering Career Network for helping provide for the students of NAU. The Engineering Career Network is a powerful application that has been built to fit your needs. Some of its key features include:

- Students can create accounts to then have a profile tracking their career milestones.
- The milestones can be manually added, or they can be added by pulling from their LinkedIn public profile.
- Any user or administrator can see the overall status of the student career data in an easy to filter dashboard.

The purpose of this user manual is to help you, our client, successfully install, administer, and maintain the Engineering Career Builder within the school's production going forward. Our aim is to make sure you are able to utilize and learn from our product for the coming years.

2. Installation

As part of the final delivery of the product, the Engineering Career Builder has been installed and is running on the NAU ITS servers. This includes both the backend server, as well as the backend database. Over time, however, you may want to move it to another web hosting service, or to reinstall the product on a different server within NAU's systems.

Accessing the System
To start, the system can currently be accessed using any web browser. The URL for the application in its current state is:

https://ac.nau.edu/egrcn/

This will allow you to access and use the app as a user/administrator.

Installing the Server
In order to install the application, you can obtain all of the files through the Github repository the code is installed on. We expect to transfer ownership of the repository to you, our client, and in doing so, the link where the repository will change based on the Github account it is transferred to. The command to install the repository on a new server is:

```
git clone https://github.com/<OwnerGithubName>/EngineeringCareerNetwork
```

Once this command is called, it will pull all of the files needed to run the backend application. The server it is running on must also have access to Node.js, which can usually either come preloaded or is installed using the distribution's package manager for the package 'nodejs'.
Once the server has the files and access to Node.js, you must then run the following command to install the Node package dependencies our app uses:

```
npm install
```

After doing so and downloading the packages, all that is needed to start it is to be in the Server directory and run the command:

```
node app.js
```

This will start running the application and will start listening on port 80. If the application instead needs to run in a Docker container, as it is currently running, then there is a slightly more complex method of running the application.

Running the Application in a Docker Container
As the application is currently running in a Docker container, it is useful to understand how to run it and install it within a container. When using the Podman system, as the NAU ITS servers use, the container image must first be built within the server. This can be done using the following command:

```
podman build -t <image-name> .
```

This command will use the Dockerfile within the Server directory to build the container image. Following this, the container image can then be turned into a container and run using the following command:

```
podman run -d --name <container-name> -p 9007:<port-used> <image-name>
```

Running this command will create the container and start running it in the background of the server. If at any point the server needs to be stopped or started again without wanting to rebuild the container, you can call the command:

```
podman start <container-name>          OR     podman stop <container-name>
```

Using these commands will help work with Docker/Podman containers, and it is helpful to understand as this is once again what NAU ITS servers use.

3. Configuration and Daily Operation

Now that the application can be installed and moved between different servers in the future if needed, it is time to go over the different configuration settings and tasks to operate the application.

Primary Admin Account Configuration
To begin, the application does not need too much in order to configure it. Currently, the primary configuration that must be put into place is your own account, which will become the primary administrator account. We currently have an administrator account setup that can verify and convert your account into the primary Admin account when you set one up within our system. This will then give you access to the administrator tools within the website, as well as give you the ability to create other administrators if and when you need more.

Dashboard Page
Once your administrator account is set up, you will be ready to start using the website as an administrator. The first page you are sent to upon logging in is the Dashboard, which has some functionality you can interact with. By default, the page will display a graph showing the progress of all students within CEIAS on obtaining any milestone. The filters can then be changed using the drop down selections, and clicking 'Generate Graph' will allow you to filter the graph to different groups and programs. Currently more programs can be added through adding to the database, and the details on this will be in the Maintenance section below.

User Search
After the Dashboard page, there is the User Search page. On this page, you will be able to do a good portion of the administrator functions. To start, you can search for a user by typing in a search term and clicking Search. As of right now, it will try to match for names. If you click Search without typing anything in, it will bring up a list of all users.

It is important to note that the user search will typically only bring up accounts set to PUBLIC by their owners, but as an administrator, you will be able to see all accounts and it will tell you their set visibility.

From the search page, you can then navigate to a user's page and see their page information, which will display their name, email, program, and milestones. Like with the search, user's profiles will usually only show PUBLIC milestones, but an administrator account can see all milestones. From here, an administrator has a few functions that a typical user does not:

- The administrator can delete user milestones from their table.
- The administrator can choose to scrape the student's LinkedIn profile. This will update the student's milestone table with their LinkedIn experiences.
- The administrator can set an account to an Alumni or an Administrator.

With these functions implemented, an administrator is able to easily search up a specific user and update their milestones, or update their role within the site.

Admin Tools

The final page that an administrator account has access to is the Admin Tools page. This page gives the administrator a place for some additional functions for monitoring the milestones being added to the website. Every milestone has a 'verified' characteristic within each table, and in the Admin Tools, any administrator can see all of the unverified milestones on the site. Any milestone added to the site is unverified by default, and a milestone is only verified by getting manual verification from an administrator on this page.

The other functionality of the Admin Tools page is the ability to scrape from all LinkedIn pages to update the site's database. This is currently set up to pull from all student profiles that have a LinkedIn URL set up.

User Profile

One page that is not available to an administrator account, as it is specifically for students, is the profile page. This page allows a student account to see their milestones, add to them manually, and set their LinkedIn URL. They are also able to activate the LinkedIn scraping to pull their career experiences from LinkedIn. All of this is done in a similar way to the user search page for an administrator, except it is only available to each student as their own profile.

4. Maintenance

Regarding the maintenance of the system, there luckily is not an incredible amount of maintenance needed to keep the application running. As it is, as long as the servers the backend and database are running on stay up, the website should continue to stay up and run. The main need for maintenance would likely revolve around the database itself, as the frontend website does not provide full control and modification of the database. As such, this section will go more into detail about accessing and managing the database on the server it is currently running on.

Database Access
The database is currently being stored and run from an ITS server within NAU. The domain for the database server is:

sweetleaf.nau.froot.nau.edu\gpdev

We are currently accessing the database through an account set up for the Engineering Career Builder, with the account details enclosed within the 'Accounts.txt' file included with the User Manual.

The last security measure for accessing the database is that you must be on NAU's wireless network, or connected to the wireless VPN through NAU. Otherwise the database connection will not work. Once this is all done, you can access the database as the ECN user. This user is only able to access and modify the database named:

engineering_career_network

From here, the database has been accessed and you are able to begin making changes or reading from it.

In our current setup, we use the software Microsoft SQL Server Management Studio to make this connection and manage the database within a GUI.

Modifying the Database
Once the database has been accessed, it can then be modified through common SQL queries. These are varied and plentiful, and we will not be going into the different queries here, but there are many resources where you can find the different query

structures for getting specific results or modifications, such as at https://www.w3schools.com/sql/.

Rather than using queries, if you did connect to the database through Microsoft SQL Server Management Studio as we typically do, you can right click on any table and click View Top 200 Rows to see the top rows of the database. There is also a Modify Top 200 Rows, and this gives an easy to use table of records that can be modified with simple box clicking to decide what to change.

Once the changes have been made, you can simply close the connection, or in our case, Microsoft SQL Server Management Studio, and the modification should take effect almost immediately.

Using Microsoft SQL Server Management Studio to Connect to and Modify the Database

In this section, I will go over all of the steps and details on how to use the software Microsoft SQL Server Management Studio to manage the database. It is a free software that is typically recommended for managing an SQL database as it makes many of the functions easy to do.

To start, when opening Microsoft SQL Server Management Studio, a window will pop up asking for the details of the database server you want to connect to. For the Server Type, set the dropdown selection to Database Engine. Next, the Server name is:

sweetleaf.nau.froot.nau.edu\gpdev

Following this, the Authentication field gets set to SQL Server Authentication, which is typically the default. Then the last two fields will use the login and password for our database access account, with the details for it once again given in the Accounts.txt file given along with this user manual.

As long as you are connected to the Internet through NAU's wifi, or using the school VPN, you will then connect to the database server. You will know it has connected when you see a list of databases show up on the left side. Look for the Engineering Career Network, and expand its Tables folder to see all of our tables. We currently have three tables set up: the User table, the Milestone table, and the Program table.

Note: Even though you will see all of the other databases, the account being used is only allowed to access and modify the Engineering Career Network database. Trying to access another, or accidentally clicking the wrong one will result in an error.

Once the tables can be seen, this is where you can either view, edit, or remove the data within the tables. In order to do this easily, right click on the table you wish to view and click Edit Top 200 Rows. This will bring up a display of the table and you can edit values similarly to editing an Excel spreadsheet. Right clicking a row will allow you to delete the row altogether, allowing you to remove records of milestones and students.

Once you are finished with the database, you can exit by closing the program. The connection to the database engine will be cut and you will have to reconnect when you open the application again.

Restarting the Backend Server
In this final section of troubleshooting, I will go over how to connect to the backend container and restart it. This will use commands similar to the ones that were seen in the Installation section, except with details on how to connect to the current server included.

Currently, the backend application is located on a server provided by NAU. You will have to be given permission to access the server by contacting the ITS department. Once your account does have access, you will first connect to the server by going to a command line and typing in:

```
ssh <nau_id>@webapps.ac.nau.edu
```

This will prompt you for your password, and once input, you will be connected remotely to the NAU server. The next step is to connect into the instance running our application, which can be done by typing:

```
ssh egrcn@localhost
```

This will not need a password, and you will be inside of the instance of the server where our server container is being run. From here, you will be able to use ‘podman’ commands to restart the container. First, you will want to be able to list all of the containers, including the running one, by typing:

```
podman ps -a
```

Once you can see the containers, and which one is running, you are able to see the name of the container on the very end of its details. At this point, you are able to stop the container by typing:

```
podman stop <container-name>
```

This step typically takes a while to execute, but it will give you control of the terminal when it is done executing after a few seconds. Once the container is stopped, there are multiple possibilities. One is to rebuild the container, and this is only needed if there have been changes to the code, and these commands are given in the Installation section. If you are instead just restarting the server, you can then run the following command to start the container back up:

```
podman run <container-name>
```

Once this is run, the server will be back up and running. To exit the ssh sessions, you just need to type ‘exit’. As you are currently two ssh sessions deep, you will likely need to exit twice.

Updating the Server with File Changes
The web server is currently running inside of a podman container. This means that when you edit files inside of the server after you SSH in, there will be no effect on the current site. This allows you to make changes without worrying about causing an issue in real time for users of the site.

Once changes have been made, you must rebuild the container image that the server container is built from. Luckily, this is just a few commands that have to be done in succession.

To start, you want to stop the current container that is running. This will mean you need to know the container’s name, and you can easily find out by using the command

```
podman ps
```

This command will list all running containers, and our server should only ever have one running. This list will show the container name, and you will then do the command:

```
podman stop <container-name>
```

This command will take a few seconds to execute, and once it is finished, it will display a random string of characters. After the container is stopped, the website will no longer be accessible until we rebuild the container.

Now you can remove the container if you for sure are not going to use it anymore, and this can be done with the following command:

```
podman rm <container-name>
```

Once this is done, you can remove the image that the container is built from. This image name can be found using the command:

```
podman images
```

Once the image name is found in this list, you can do the following command to remove it:

```
podman rmi <image-name>
```

Finally, you have removed the old container and image that used the files before the changes. If you want to keep the image and container and just not run them, you can also do so. You will just then need to name the new image and container something new.

To build the new image for the server container, you must be inside of the Server directory and run the command:

```
podman build -t <image-name> .
```

Running this command will then cause a 10 step process to start that will install all the necessary files for the container to use. It may take a while, but when it is done, it will return a long random string of characters. After the image is built, all that is needed is to build and run the container. Luckily, this can be done in one command:

```
podman run -d --name <container-name> -p 9007:9007 <image-name>
```

The container name is something you can set here for this new container, and the image name is the one that matches what you built in the previous command.

Updating the Server with File Changes (quick version)
Begin by stopping the current running container:

```
podman stop <container-name>
```

(Optional) Remove the container, and then the image if not needed for later use (old image and container will use old files before changes made):

```
podman rm <container-name>
podman rmi <image-name>
```

Build the new container image:

```
podman build -t <image-name> .
```

Build and run the new container in one command:

```
podman run -d --name <container-name> -p 9007:9007 <image-name>
```

The new container will be up and running, and the website should now be accessible.

Transfer files between Local Machine and Server
First you will want to SCP the files into the backend server using the following command:

```
scp -r server/directory/path <nau_id>@webapps.ac.nau.edu:Server
```

This will move all of the files into a directory in the backend server called Server. These files are not in the right place, however. Once they have been moved here, ssh into the backend server using:

```
ssh <nau_id>@webapps.ac.nau.edu
```

Once you are in, you will need to now SCP the files to the correct instance on the server. This can be done using the following command:

```
scp -r Server egrcn@localhost:Server
```

This will move all of the files into the egrcn instance, which is now where you can start the server. To take files out and put them on the local machine, these steps are done in reverse order, with the argument parameters swapped. So to get files out, you would start in the egrcn instance and do:

```
exit
scp -r egrcn@localhost:Server Server
exit
scp -r <nau_id>@webapps.ac.nau.edu server/directory/path
```

Once this is done, the files can be transferred in and out using SCP.

5. Troubleshooting

Troubleshooting, similarly to maintenance, is not expected to be incredibly complex. As our website is mostly focused on handling data, there are two main areas we had some trouble with ourselves that we had to work to solve: the database and the LinkedIn scraping. These two pieces are the key areas we would expect problems to arise if any were to come up, as these are the two pieces that are based on external sources from our actual website files.

Database Troubleshooting
First, if there are ever any issues on the website, the quickest place to check is the Dashboard. If the page loads up but the graph does not, it is likely that the system is not getting a connection to the database. This was only ever a problem when moving between databases, and it should not come up without warning, but it will likely happen in the case that the NAU ITS database server goes down.

In the case that the database is not connecting, the first step to try fixing it is to stop and start the backend application. This is because we want to ensure that the connection to the database has not timed out. If this does not work, check with the database administrators within NAU's ITS department to see if any of the login credentials for the database have changed. If this is the case, the file within our application that holds the database login information must be modified to be correct. Otherwise, it is likely there is an issue with the database server itself, and this is something handled by NAU's ITS department.

LinkedIn Troubleshooting
The LinkedIn part of our website is where we would expect the most trouble to arise. Currently, the tool works by signing into a hidden account and then pulling the HTML of a user's public page. It then parses this information for the experiences. Through extensive testing, we have found that the current system in place works very consistently. We expect this to continue, but there is always the possibility that LinkedIn changes their site layout in some way, or that the account being used needs to be checked in the case that it stops working. Both cases are laid out here.

- LinkedIn HTML structure changes

  If the LinkedIn HTML changes, this is not incredibly difficult to go about fixing. The primary section to fix is within our parser, as the rest of the system will still go about getting the HTML of the page. Within the parser, all that needs to be changed is what the class names of the HTML structure certain parts are in, and this is specified within the code of the parser itself.

- LinkedIn Account Trouble

  If the HTML is still parsing correctly, it is likely there is an account issue with the LinkedIn account our system uses. We have included the sign in details for the LinkedIn account, as well as the email it is attached to (both made for the purpose of this application). We recommend checking both accounts to see if there is some problem with the LinkedIn account, and try getting it resolved. If it cannot be resolved, another fix is to use a different LinkedIn account with only one change needed within the LinkedIn module of the code.

With these two problems out of the way, we do not expect the LinkedIn module to give any other kind of problems, as we did not run into any other types through our testing.

6. Conclusion

We hope you enjoy using the Engineering Career Network for many years to come. We understand that there are some pieces that could be added before it goes into full production, but we hope that the software along with our documentation will be enough to push this project further. It has been great being able to help get this product realized. With best wishes from your Engineering Career Network developers:

Logan Burbank	ljb298@nau.edu
Jackson Braudaway	jeb523@nau.edu
Kailin Chen	kc2637@nau.edu
Parker Marschel	pgm33@nau.edu
Zhenyu Yang	zy66@nau.edu

While we are all moving on to professional careers, we would be happy to answer any quick questions in the coming months to help you get the product deployed. Once again, we wish you the best with the Engineering Career Network and we thank you for the opportunity to develop it.