



# **C&I Doctoral Tracking Tool User Manual**

12/13/2022

**Project:**

C & I Doctoral Tracking Tool

**Project Sponsors:**

Gretchen McAllister & Michele Benedict

**Faculty Member:**

Michael Leverington

**Team Name:**

What's Up Doc

**Team Members:**

Adam Larson (Lead), Brandon Shaffer, and Eddie Lipan

**Team Mentor:**

Daniel Kramer

<b>Table of Contents:</b>	Pages
<b>1.0 Introduction</b>	3
<b>2.0 Front-End Installation</b>	4
<b>3.0 Front-End Configuration and Daily Operation</b>	13
<b>4.0 Front-End Maintenance</b>	19
<b>5.0 Front-End Trouble-shooting</b>	21
<b>6.0 Server</b>	23
<b>7.0 Database</b>	28
<b>8.0 Server/Database Troubleshooting</b>	30
<b>9.0 Conclusion</b>	31

# 1.0 Introduction

Team What's Up Doc would like to formally acknowledge and thank our NAU Capstone Sponsors for working with us: **Dr. Gretchen McAllister, Ph.D.**, Coordinator of the NAU Curriculum and Instruction Doctoral Program, and **Michele Benedict**, Administrative Services Assistant. We thank you for your time, dedication, and patience throughout this past year. We would also like to thank anyone coming to use our product who might need further instruction. We are pleased to provide you with an Acceptance Demo and a User Manual that we have tirelessly worked on and we hope you are pleased with the results.

Team What's Up Doc have created an intuitive website application that includes, but are not limited to, these key features:

- Student Home page that allows a Student user to track and view the required milestones for each phase of the graduate program (with color coding for completed/uncompleted milestones).
  - Students can upload a PDF file to the database for each milestone.
  - Students can download any PDF they have previously uploaded to the database.
  - Students can delete any milestone PDF they have added to the database.
- Admin home page that allows Administrators to access three distinct windows.
  - The Add/Remove Users window houses one of our awesome DataTables which displays relevant user data from the database to our admins.
    - Includes searching, sorting, pagination, copy to clipboard, export to csv, export to pdf, export to excel, new user button, edit user button, delete user button, and refresh table button.
  - The Phase Review window houses our second DataTable which displays the relevant Phase Review data for each user from the database.
    - Includes searching, sorting, pagination, copy to clipboard, export to csv, export to pdf, export to excel, and refresh table.
  - The last window is the Student Progress Review, which will allow the admin to view a Student's unique home page and access their uploaded files.

The purpose of this user manual is to instruct, inform, and aid you, our clients, in installing, administering and maintaining your Doctoral Tracking Tool Product. We also hope that this user manual will serve as a guide for anyone looking to refactor and expand upon this product in the future.

## **2.0 Front-End Installation**

### **2.1 GitHub Repo Handover**

Here Team What's Up Doc will detail who currently owns our GitHub repository, how to access our GitHub repository, and how to download our most up to date codebase from the repository.

Repo Link: <https://github.com/adlarson88/WhatsUpDoc>

Owner: adlarson88

Repository Privacy: Currently Private

### **2.2 User Installation**

Considering this is a website application, there are a few initial installation instructions that are required in order to access the product. Note: Users may access the website from any platform. The following requirements need to be met:

1. Users are required to have an internet browser installed, i.e. Google Chrome, FireFox, etc.
2. Users are required to have internet connection.
3. In future iterations of the product, users are required to login using a valid NAU associated gmail account, which.
4. For the Acceptance Demo and handover, both the student home page and the admin home page will be accessible, **by anyone**, from the initial login page via two redirect buttons in a green box at the bottom of the page.

Without these few initial requirements, a user will not be able to access the internet and our web page will not load for the user.

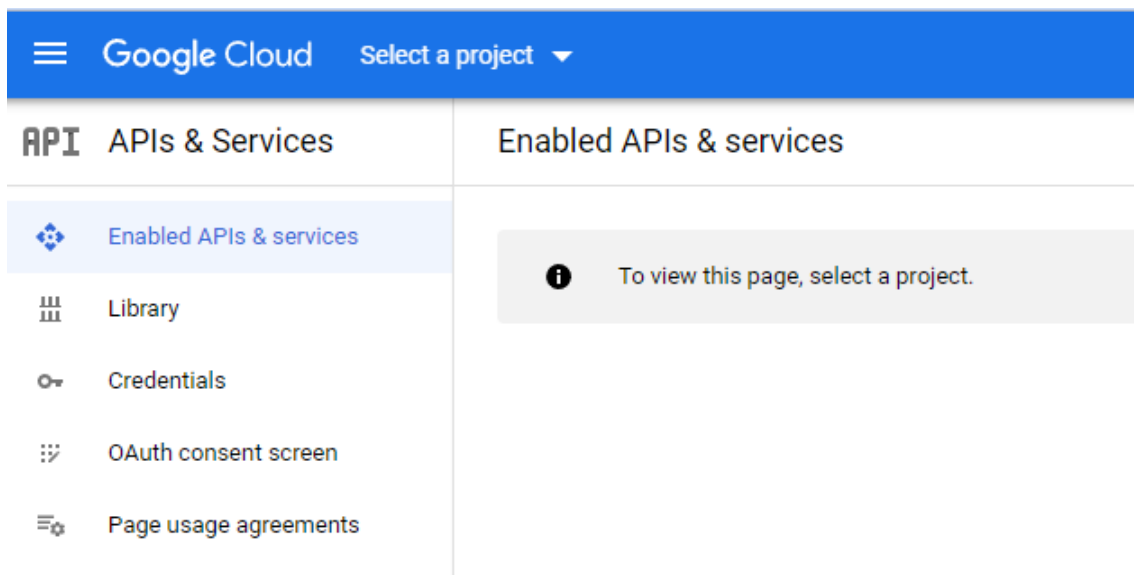
### **2.3 Developer Project Installation**

1. Download our most recent codebase from our GitHub repository: <https://github.com/adlarson88/WhatsUpDoc> OR, download it from the flash drive provided at the Final Acceptance Demo/handoff.
2. All front end website related code for the product can be found in the directory titled "C&I\_Doctoral\_Tracker".
  - a. In this first directory you will find:
    - i. all three HTML files
    - ii. our style sheet
    - iii. an images directory: holds any images used within the website application

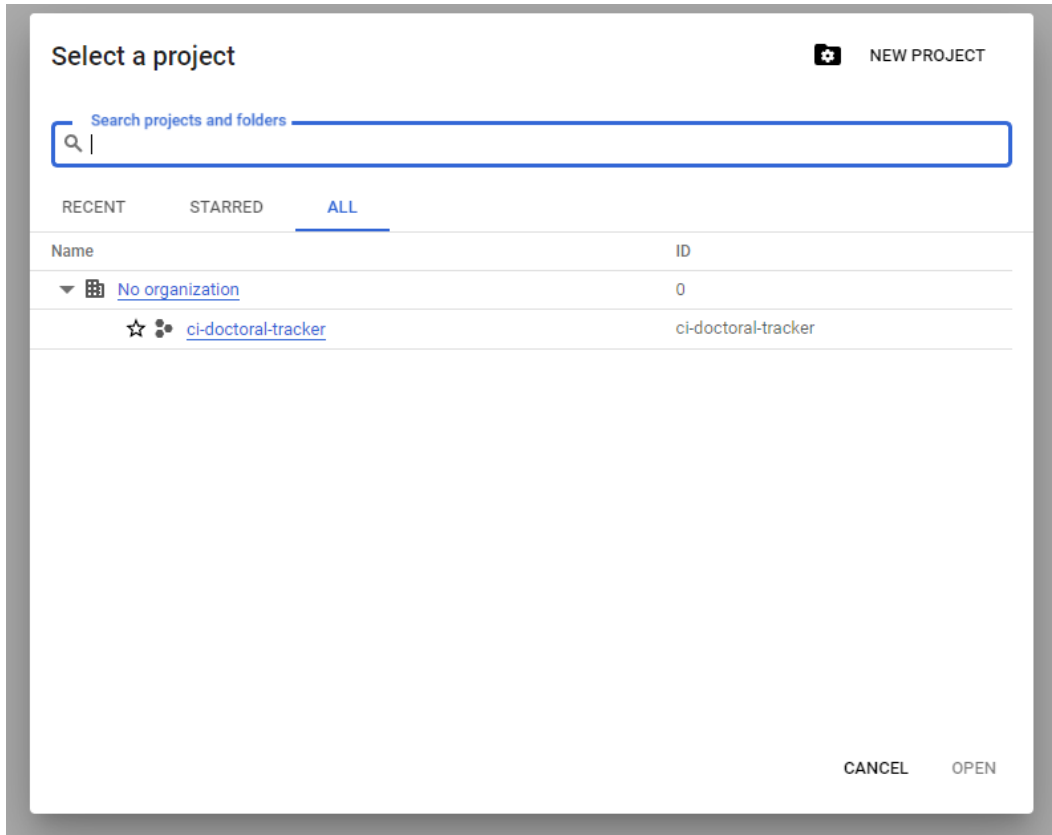
- iv. a scripts directory: holds the two relevant JavaScript files that are referenced in their corresponding HTML files
3. Database code can be found in the directory titled “PhDTracker”.
4. Team What’s Up Doc’s Team Website code can also be found in this main directory, titled “website”.
5. Then, the developer would need any IDE (such as VS Code) to access and write to the code.
6. Once changes are made to the developer’s local files, they must commit and create a pull request on the GitHub repository.
7. Once the pull request has been approved and merged, the product's main/live code can be updated and any code changes made to the product will be reflected on the live webpage.

## **2.4 Developer Google APIs, Services & Credentials Installation (For use with the “Sign In With Google” Button)**

1. First, go to the following website:  
<https://console.cloud.google.com/apis/dashboard>.
2. Then, the developer must log into an NAU associated gmail account.
3. At the top of the Google APIs & Services Dashboard, there is a navigation bar that contains a button near the left hand side of the page. This button is titled “Select a project” and allows the developer to select an existing project or create a new project.




4. Upon clicking the “Select a project” button, a popup window will appear that looks like this:




5. Click the “New Project” button.
6. The New Project page should look like this:



### New Project

 You have 11 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

Project name \*  

Project ID: linen-sunbeam-371512. It cannot be changed later. [EDIT](#)

Organization \*   

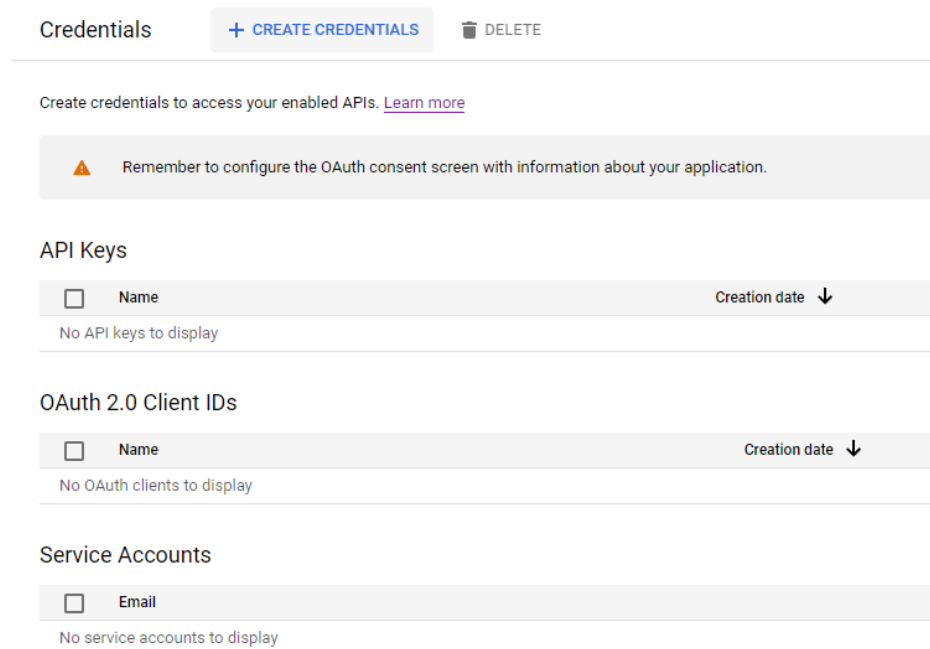
Select an organization to attach it to a project. This selection can't be changed later.

Location \*  [BROWSE](#)

Parent organization or folder

[CREATE](#) [CANCEL](#)

7. Here you will specify your project name, your project ID (automatically created, but can be manually edited by clicking the blue highlighted EDIT button), and your associated organization.
  - a. Note: This is why it is imperative that the developer has an active @nau.edu gmail account. If the project is not created on an nau associated gmail account, then the nau.edu organization will not pop up as an option when you are creating your project. This step is also very important if you would like the login process to include NAU CAS Authentication and DUO Mobile Two-Factor Authentication.
8. Click the blue “Create” button.
9. Now, look back to the top navigation bar, again click the button that allows you to select your project, and make sure your newly created NAU associated project is selected.
10. Now that you are within your newly created Google APIs & Services project, you can specify the API keys and any credentials your project might use.
11. First, I would click the “Enabled APIs & services” button from the left hand taskbar.
  - a. Here you can specify exactly which of Google’s APIs your project intends to use and call upon. Select the APIs that you will prompt your users to give consent to!
12. Then, I would direct you to click the “Credentials” button from the same left hand taskbar.
  - a. This page is arguably the most important page of the Google APIs & Services dashboard. It should look like this to start:



- b. Click “Create Credentials”, highlighted in blue.
- c. Begin by selecting and creating an “API Key” credential. A popup should appear with a long API Key code. You will be able to view this key again shortly. Exit the API Key code popup.
- d. Click your new API Key under the “API Keys” section, it should be highlighted blue with an underline.
  - i. Here you can specify your Key Restrictions, Application Restrictions, and API Restrictions for this particular API Key of your project.
- e. Click the blue back button to go back to the Credentials List page.
- f. Click the blue “Create Credentials” button once more.
- g. Now choose “OAuth client ID” as your credential to be created.
  - i. Since it is your first time trying to create an OAuth client, you must click the blue button on the right that says “Configure Consent Screen ”. From the two options available, choose “Internal ”, this will make it so that the project only accepts nau associated gmail accounts during login. Then accept and you should redirect to a page that looks like the image on the following page:



## Edit app registration

1 **OAuth consent screen** — 2 Scopes — 3 Summary

### App information

This shows in the consent screen, and helps end users know who you are and contact you

The name of the app asking for consent

For users to contact you with questions about their consent

[BROWSE](#)

Upload an image, not larger than 1MB on the consent screen that will help users recognize your app. Allowed image formats are JPG, PNG, and BMP. Logos should be square and 120px by 120px for the best results.

### App domain

To protect you and your users, Google only allows apps using OAuth to use Authorized Domains. The following information will be shown to your users on the consent screen.

Provide users a link to your home page

Provide users a link to your public privacy policy

Provide users a link to your public terms of service

### Authorized domains

When a domain is used on the consent screen or in an OAuth client's configuration, it must be pre-registered here. If your app needs to go through verification, please go to the [Google Search Console](#) to check if your domains are authorized. [Learn more](#) about the authorized domain limit.

[+ ADD DOMAIN](#)

### Developer contact information

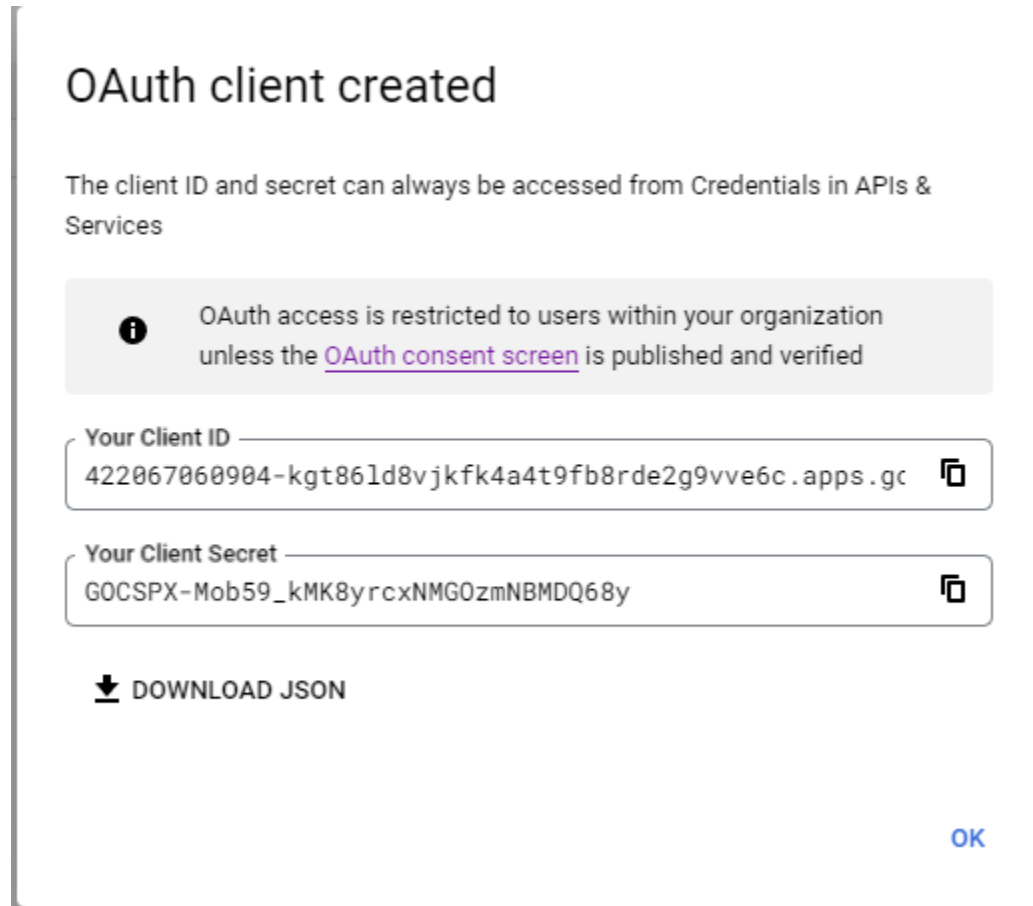
These email addresses are for Google to notify you about any changes to your project.

[SAVE AND CONTINUE](#)

[CANCEL](#)

- h. Enter in your project applications name.
  - i. Enter in a support email.
  - j. Upload a logo if you choose.
  - k. Under “App Domain”, provide the URL for your project's homepage. As of the Acceptance Demo, our domain is [doctracker.org](https://doctracker.org).
  - l. Include the link to your applications Privacy Policy document.
  - m. Include the link to your Applications Terms of Service document.
  - n. Under “Authorized Domains”, add any domains that your product might be running on, or redirecting to.
  - o. Include a developer's email and hit “Save and Continue”.
  - p. Now you are in the Scopes section of the OAuth Client Creation page.
    - i. Click the blue button titled “Add or Remove Scopes”.
    - ii. Here you can explicitly specify any of the Google APIs you might have chosen to add earlier in Step 11. It is best practice to only ask your users to consent to Google APIs you are actively using within the product. Add in more scopes as you add features, do not preemptively ask users to consent to a whole bunch of APIs.
    - iii. Once you have selected all your scopes, click the blue “Save and Continue” button at the bottom.
  - q. On the final page of the OAuth Client creation, you will see a summary of your OAuth consent screen with all the information you just disclosed, as well as a list of all the scopes you enabled.
  - r. Click the blue “Back to Dashboard” button.
13. Now that your “Configure Consent Screen” is completed, go ahead and click “Create Credentials” again, and once more select OAuth Client ID.
- a. Now you should be directed to a page that asks you to specify your application type. In the case of this project, select “Website Application”.
  - b. Name your OAuth 2.0 client.
  - c. Next, under “Authorized JavaScript Origins“, you will click the blue “Add URI” button and add any origin URIs needed for the project.
    - i. The website describes these origins as: “The HTTP origins that host your web application. This value can't contain wildcards or paths. If you use a port other than 80, you must specify it. For example: <https://example.com:8080>.”
  - d. Next, under “Authorized Redirect URIs”, click the blue “Add URI” button and add any URIs that might be used during a page redirect.

- e. Click create! A final popup screen will appear giving you your Client ID!



- f. Copy this Client ID to your clipboard as you will need it in the following step.
14. Finally, open your chosen IDE and open the index.html file within the C&I\_Doctoral\_Tracker project directory that you downloaded earlier. Within the index.html file, there will be two <divs> that specifically relate to the "Sign In With

Google” button. Here is a screenshot of the two Google button divs.

```
<!-- The data-client_id attribute is asking for the Client ID given to you when you
      create your Google API Credentials page. Products outside of the development stages
      should ideally hide the client ID so people can't see it in the HTML code with F12.

      The data-ux_mode attribute declares whether the Google sign in workflow will be
      carried out in a separate window or a popup window.

      The data-login_url attribute refers to the URL that you wish to redirect to
      if the user successfully logs in using an NAU gmail account. This URL/Domain needs to be
      explicitly specified on the Google API Credentials page as well. -->
<div id="g_id_onload"
      data-client_id="911436566286-36r6uc7oavbrmqcmn7b2ijo3q34mn1gi.apps.googleusercontent.com"
      data-auto_select="false"
      data-auto_prompt="false"
      data-context="signin"
      data-ux_mode="popup"
      data-login_uri="https://doctracker.org/home.html?"
      data-itp_support="true">
</div>

<!-- This div allows the user to customize their Google button. Refer to Google's
      documentation to better customize your Google button. -->
<div class="g_id_signin"
      data-type="standard"
      data-shape="pill"
      data-theme="filled_blue"
      data-text="signin_with"
      data-size="large"
      data-locale="en-US"
      data-logo_alignment="left"
      data-width="275">
</div>
```

15. The div we are interested in making changes to has the id “g\_id\_onload”. Inside this div, we are particularly interested in two attributes. The first one is the data-client\_id attribute. The second one is the data-login\_uri attribute.
  - a. The data-client\_id attribute is where we want to paste our recently saved OAuth Client ID. This will link our “Sign In with Google” button with the project and OAuth credentials we just created.
  - b. The data-login\_uri attribute is where we specify the redirect URI which is where the user will be sent upon successful login with an NAU gmail account.
16. Congratulations! Your recognizable Google button should be ready to go assuming your Redirect URI is a valid destination.

## 3.0 Front-End Configuration and Daily Operation

### 3.1 Initial Login/Landing Page View

Initial Set Up:

1. The only part of the project that Team What's Up Doc was not able to finish relates to the unique user login experience, unique student home pages, and (as a result of having no unique student pages) the Student Progress Viewer available to the admins only allows them to view a single default student home page instead of being able to search for numerous unique student home pages.
2. If a future developer intends to complete this part of the project, they would need to refactor the index.html code. The features that need to be added to make this fully functional includes:
  - a. Upon successful NAU user login, the developer must decode the JWT payload provided by Google.
  - b. Once decoded, this JSON Web Token payload contains all the users personal google information (email, name, DOB, etc.), as well as, two authentication tokens. The first token is the actual authentication token needed to associate a user with a unique home page. The second token is a refresh token in case the authentication token expires.
  - c. Once the JWT payload has been decoded, the developer must send a request to the database containing the userID used to sign in, the admin status of the user that signed in, and both the tokens provided to us by Google. On the back end, the database must save and verify both authentication tokens for that particular userID, if the token is a match, redirect student users to their UNIQUE student home page. If the user has admin status set to "true", then redirect the user to the admin home page.
  - d. Note: Team What's Up Doc fell short in this feature due to the fact that all four major features of the project (Login page, Student page, Admin page, and the MySQL database, would have required extensive changes to be made to the source code to allow for a cohesive authentication and authorization process. It just proved to be an overwhelming feature to complete on top of all of our other challenges.**
3. Since the Acceptance Demo will not feature a working Login window, we have implemented a Temporary Developer Window that contains two redirect buttons. One button redirects straight to the student home page, and the other button redirects straight to the admin home page. This allows users to test our working

database and web pages, despite not having the authentication and authorization features fully functional.

Daily Operation:

1. Upon reaching our landing page, you should see a clean login window with the recognizable “Sign in With Google” button. **NOTE:** A valid Google APIs & Services Client ID and valid redirect URIs are needed in order for the “Sign In with Google” button to function properly: if those requirements are met, then the Google Sign In workflow should look like this:
  - a. Click the “Sign in With Google” button.
  - b. Enter in your NAU gmail account.
  - c. NAU’s Central Authentication Service (CAS) window should popup.
  - d. Enter in your NAU credentials.
  - e. Choose your @nau.edu Google account.
  - f. If you have DUO push enabled, you will be prompted to authenticate your two factor DUO push app.
  - g. After hitting “Accept” in your DUO push application, you will be prompted to “Trust Browser”.
  - h. If this is your first time accessing the application using your NAU gmail account, you will be prompted to give consent to the Google APIs being used by the application.
  - i. Lastly, if the user is recognized as a student, they will be redirected to their unique student home page (unique student pages, still in development). If the user is recognized as having admin status, then they will be redirected to their
2. If you don’t have a working/valid redirect URI (like in the case of our Acceptance Demo), then you can use the green \*Temporary\* Developer Window at the bottom of the login page to redirect to our two different home pages.

### **3.2 Student View**

Initial Set Up:

Ensure the phase count and corresponding sub phases are accurate in the home.html.

- If more are needed, copy and paste prepared sections and update the phase\_x\_y labels with x being the phase and y being the sub phase.
- Then in home.js add function calls for the new phases in parseCompleteList(), prepare(), and pingDB()

Daily Operation:

1. Upon signing in, the student should have the phase 1 sub menu open by default.

2. By clicking a section of either the upper phase bar and/or the sub menu that drops down, the student can select the desired task to upload a file for.
3. From the selected task, a student can see a preview of an uploaded PDF file, and do any of the following:
  - a. Upload a file: click on the upload button to browse local files to upload to the database. This will replace a previously uploaded file under the same task if one exists.
  - b. Delete a file: click the delete button to remove selected file from the database.
  - c. Download a file: click the download button to download a copy of the file to the local machine.

### **3.3 Administrator View**

#### Initial Set Up:

1. The first thing admins will have to do on their admin home page is begin populating their empty database via the “Add/Remove Users” tab.
  - a. If both admins are not already in the database, add them to the table. Admins need to be in the database table in order to login in the future using the “Sign in With Google” button.
    - i. Go ahead and click the “New User” button twice and fill out the form twice. Once for Dr. Gretchen McAllister and once for Michele Benedict.
    - ii. For administrators, it is only necessary to specify their userID, last name, first name, and their admin status (set this to **TRUE**). Administrators are the only users within the database that should have the admin status set as “true”, all other student users should have their admin status set to “false” by default.
2. Once both administrators are in the database (and consequently being shown in the table), you can now continue to add all your users into the graduate program. This initial part of the admin set up will be a little tedious as the admins will have to manually enter in all the students of the program one by one.
3. At any point, the admins can select a cell within the table and click the “Edit User” button. This will open up another form window which will prompt the admin to enter in new user information. Upon submitting this “Edit User Form”, the user you had selected will have their information rewritten using your newly submitted information.

4. Once all the students and admins have been added to the table/database, users can be searched by keyword and each column can be individually sorted by clicking on the colored column name.

#### Daily Operation:

1. Upon opening the Administrators home page, the first thing the admins should see is a taskbar at the top of the page. This taskbar should contain three buttons at the time of creating this User Manual. The three buttons are as follows:
  - a. Add/Remove Users Table: This button opens up the tab containing the “Add/Remove Users” DataTable, and closes either of the other two tabs that might be open instead.
  - b. Phase Review Table: This button opens up the tab containing the “Phase Review” DataTable, and closes either of the other two tabs that might be open instead.
  - c. Student Progress Viewer: This button opens up the tab containing the search bar that allows the admins to search for students by userID. An embedded window of the student’s unique home page should appear upon submission. However, as we do not currently have token access or unique user pages, this tab only showcases a single student's home page, for reference.
2. **Looking at the “Add/Remove Users” Table:** Admins may search for specific users by keyword, or they can sort each column individually. Admins can also select how many table cells they want to appear on one page and they have access to pagination which means they can move between pages of the table.
  - a. Admins can click the “New User” button at any time to add a new user to the graduate program. This will open up another form window which will prompt the admin to enter in new user information. Upon submitting this “New User Form”, you should see your new user added to the table/database in real time.
    - i. If the admin did not mean to click the “New User” button or the “Edit User” button, they may click the X at the bottom of the popup window to exit the window and make no changes to the table/database.
  - b. Admins can select a cell within the table and click the “Edit User” button. This will open up another form window which will prompt the admin to enter in new user information. Upon submitting this “Edit User Form”, the user you had selected will have their information rewritten using your newly submitted information. Note: if the admin clicks the “Edit User” button after selecting a user, they **MUST** explicitly enter in all the information being asked for or the user will be submitted blank information.



- i. If the admin did not mean to click the “New User” button or the “Edit User” button, they may click the X at the bottom of the popup window to exit the window and make no changes to the table/database.
  - c. Admins can, again, select a user cell within the table and click the “Delete User” button to delete a user from the table AND the database. Note: the admin **MUST** press the “Refresh Table” button after **every user deletion**, in order for the deletion to appear on the table.
  - d. Admins can click the “Refresh Table” button at any time to get a fresh table reload.
  - e. Admins can click the “Copy” button which copies the entire table to your clipboard.
  - f. Admins can click the “CSV” button which exports the table to a CSV file available in your browser downloads.
  - g. Admins can click the “Excel” button which exports the table to an Excel file available in your browser downloads.
  - h. Admins can click the “PDF” button which exports the table to a PDF file available in your browser downloads.
3. **Looking at the “Phase Review” Table:** Admins may search for specific users or phases by keyword, or they can sort each column individually. Admins can also select how many table cells they want to appear on one page and they have access to pagination which means they can move between pages of the table. This “Phase Review” table is meant to be strictly analytical and as a result, the admins can not dynamically change anything directly from this table. *Note:* this is not to say that changes made to the “Add/Remove Users” table will not show up in the overlapping columns in the “Phase Review” table, as the changes will be apparent immediately in both tables.
- a. Admins can click the “Refresh Table” button at any time to get a fresh table reload. Note: the “Refresh Table” button on the “Phase Review” table simply reloads the entire web page. I was not able to find a way to reload only the div containing the “Phase Review” table.
  - b. Admins can click the “Copy” button which copies the entire table to your clipboard.
  - c. Admins can click the “CSV” button which exports the table to a CSV file available in your browser downloads.
  - d. Admins can click the “Excel” button which exports the table to an Excel file available in your browser downloads.
  - e. Admins can click the “PDF” button which exports the table to a PDF file available in your browser downloads.

4. Looking at the “Student Progress Viewer” window: The intention is for administrators to be able to search for student users by their unique userID and view an embedded view of the student’s unique home page. This will allow the admins to view exactly what was submitted by the searched student.
  - a. Note: This tab requires further development and refactoring. Since we do not currently have unique student pages implemented, the search bar is statically set to show the default student home page (set to show the home page of eml292 as of the Acceptance Demo).
5. Lastly, admins can click the “Sign Out” button in the top right corner and be redirected back to the landing page/login page.
  - a. Note: This sign out button DOES NOT sign the user out of their Google account. As of the time I coded the Sign Out button, Google did not have an easy way to explicitly sign the user out of their gmail account upon button click. Instead, if the user wishes to sign out of their Google gmail account, they must do so through their browser (Ideally, the Chrome browser).

## **4.0 Front-End Maintenance**

### ***4.1 Landing Page/Login Page View***

1. The little maintenance that might need to be done to the landing page involves the Google APIs & Services Credentials page outlined in section 2.4.
  - a. If the domain were to ever change, you would have to make those changes in the credentials OAuth 2.0 page.
2. The only other maintenance that might need to be done to the login page involves getting the Login Window fully functional (as described in Section 3.1).
  - a. Once the “Sign in With Google” button is doing its job properly and redirecting correctly, the future developers can remove the green “Developer Only” div box that contains the two student home page and admin home page redirect buttons.

### ***4.2 Student View***

#### **Change in number of Phases**

Refer to 3.1 initial set-up instructions

#### **Change in host server**

Should the host of the database change, the url for the requests will need to be changed in home.js.

- `uploadFile()`
- `pingDB()`
- `dbPreview()`
- `download()`
- `deleteFile()`

### ***4.3 Administrator View***

1. Admins will proactively be doing maintenance on both DataTables that are accessible. As they Add/Edit/Remove users from the program, they will be actively monitoring the table size and, as a result, the database size as well. We have coded these features with scalability in mind, but it is worth noting that the more users within the database, the slower fetch times will become. Therefore, it is vital to manage the table sizes accordingly.
2. In the event that the graduate program changes, or phases begin to include more milestones, a developer will need to access the admin.html page and the admin.js page in order to alter the text related to each Phase/Milestone.

3. If, for whatever reason, admins want to add additional relevant student information, they will need to add a table in the database first, and then come back to the admin.js and admin.html code and add the new database tables/rows to the DataTables initializations and JavaScript functions.

## **5.0 Front-End Troubleshooting**

### **5.1 Landing Page/Login Page View**

1. “Sign in With Google” button not working?
  - a. Check that your Google APIs & Services project is created with the correct API keys, and the correct OAuth 2.0 Client ID is created and correctly attached to your Google button code within the index.html code.
  - b. Check that your Redirect URIs and Domain URIs are correctly implemented in your OAuth 2.0 credentials page and in your Google button code within the index.html code. You must explicitly define all redirect domains and URIs, therefore if the redirect URI is not a part of the credentials page, then the redirect will prompt an error.

### **5.2 Student View**

#### **Database not responding**

Ensure that the database is up, and that requests are sent to the correct URL. Ensure that the internet connection is online as well.

#### **Error in Promise**

Check corresponding error code for details

### **5.3 Administrator View**

1. Help! Neither of the DataTables tables are showing up when I click on either of the two table tabs:
  - a. This is almost 100% always due to the server being down. Note: If the server is down, you will see a colored box with nothing inside of it in the middle of your screen (where the table normally loads).
    - i. Reset or make sure your server is running through its host.
2. Help! When I click the Student Progress Viewer tab, and I search for a specific userID, the userID’s unique home page is not showing up in place of the default student home page:
  - a. This is due to the fact that we have not fully implemented unique student homepages and as a result, we cannot reference unique student home pages yet, therefore the Student Progress Viewer is showing a single students (eml292) home page.
3. Help! When I click the “Sign Out” button, I am only redirected back to the landing page and I am not signed out of my Google account:
  - a. As stated above in section 3.3, list item 5: Note: This sign out button DOES NOT sign the user out of their Google account. As of the time I coded the Sign Out button, Google did not have an easy way to explicitly

sign the user out of their gmail account upon button click. Instead, if the user wishes to sign out of their Google gmail account, they must do so through their browser (Ideally, the Chrome browser).

4. Help! When I click the “Refresh Table” button within the “Phase Review Table”, it reloads the entire page instead of just the table I am currently viewing:
  - a. As stated above in section 3.3 list item 3: Note: the “Refresh Table” button on the “Phase Review” table simply reloads the entire web page. I was not able to find a way to reload only the div containing just the “Phase Review” table.

## **6.0 Server**

The server acts as the backbone of the product. All requests and responses are provided by this service, as well data management and processing.

### **6.1 Operation**

#### Controller

The controller provides a sender/receiver interface for requests to be made to. Requests are made to the IP address or domain where the server is housed using the designated port. At the time of writing, this program is requested through <https://doctracker.org:8443> as a base access point. Functionality is added through controller functions with specific HTTP request methods attached to them.

The access point <https://doctracker.org:8443/user> is currently the only controller in effect. Below sections will explain additional path information and data that needs to be available to perform the type of request

#### GET requests (no payload)

##### **/all**

Returns a list of each user currently in the database and their associated information in JSON format. Specific user information can be found in the Database section user table.

##### **/all/summary**

Returns a list of each user in the database with a summary of completed milestones in JSON format. JSON information is limited to userID and first/last name, and a sum of milestones in each phase (1 to 4).

##### **/id/info**

Returns a JSON of a specific user's information. Specific user information can be found in the Database section user table. To access a specific user, the userID must be known and input into the **{id}** section of the path (ex. /xyz789/info).

##### **/files/{uploadID}**

Returns either the requested file or a 404 response if the file does not exist. The **{uploadID}** must be known and is retrievable by the following request.

##### **/id/getFiles**

Returns a specific user's files in a list of JSON format. The userID must be known and input into the **{id}** section of the path. JSON data returned is the uploadID, what it's uploaded as, and the upload time.

**/preview/{uploadID}**

Returns a base64 encoded string of the byte data of a file. The **{uploadID}** must be known and can be retrieved with the above request.

DELETE requests (no payload)

**/delete/{uploadID}**

Returns a 200 if the file is removed or a 404 if the file does not exist. **{uploadID}** must be known to successfully remove the file and can be found by the "getFiles" GET request.

**/{id}/deleteStudent**

Returns either the deleted user information or a 404 if the student does not exist. **{id}** is the userID value of the user to be deleted. All associated files uploaded are also cleared upon deletion. Currently written in an early notation, id and student should be updated in a future iteration.

POST requests (create payload)

**/create**

Returns the created user information, 409 if the student already exists, or 417 if the user cannot be stored. Payload data is a JSON with all desired field information in the Database user table except 'userNo'. **REQUIRED FIELDS** are userID, first and last names.

**/{id}/upload/{uploadAs}**

Returns a 200 if the file was uploaded successfully or a 417 if an error occurred. **{id}** requires the userID and **{uploadAs}** requires the task the file is being uploaded under. Payload is a Multipart file as a variable called 'file' (ex. file= in the multipart request). This also sends an email out based on the userID input with a successful upload.

PUT requests (update payload)

**/{id}/update**

Returns the updated user JSON or a 404 if the student doesn't exist. **{id}** requires the userID to be modified. Payload is the fields under the Database user table as desired without the userID or userNo.



## 6.2 Classes and Primary Functions

### Controllers

The user controller is the only controller currently implemented. It is the core front-facing class. All requests connect to the controller methods based off of the requested path and handle processing as necessary. This class is connected to the entirety of the project which cannot function without it.

### Services

The services package includes the majority of internal functioning classes of the project. These include all the fields of the databases, the automatic email system, all exception handling, and the file transfer system. All of these sub-packages and classes are referenced by the user controller to perform functions.

### Repositories

Repositories are simple interfaces extending Spring's built in JPA repository systems. These require the proper class with matching database table fields and corresponding type to function properly. Two are included as the database only consists of two tables at time of writing.

### Security

Currently houses the security policy for the server which disables CSRF and adds CORS mapping for browser connectivity. Long term this will include permissions for users that are logged in, though this is currently excluded.

### Configuration

#### Application.yml

Houses a significant amount of config data:

- SSL and Port information for server operation
- JPA/Hibernate
- MySQL connection information
- File transfer size
- OAuth2
- Email service

#### Springboot.p12

SSL file necessary for encrypted data transfer. Necessary keystore information housed in yml file.

Pom.xml

Maven application class path. Includes **ALL** dependencies and versions for the project. Any changes require the project to be rebuilt and can be considered as the core of the project. If additional functionality is added to the program, dependencies must be included here for proper operation - these are typically Spring related but can be external.

### **6.3 How to Run**

Being a Maven based project, packaging and compiling into a JAR executable is automated through a simple command. To successfully create this, the following steps can be followed in a Unix-based environment from the command line based on the file structure stored on the provided flash drive.

1. cd WhatsUpDoc/PhDTracker
2. mvn clean package
  - a. Program will compile and show status during
  - b. If it does not compile successfully view the troubleshooting section at the end of this manual
3. cd target

Within this 'target' folder is the 'PhDTracker-0.0.1-SNAPSHOT.jar' executable file that launches the server when run. This file (the others in 'target' can be ignored) should be copied to the desired location for server operation. Once copied, enter the following commands to run the server.

1. java -jar PhDTracker-0.0.1-SNAPSHOT.jar
2. Wait 'started' message to show
3. Press 'control + Z'
4. jobs
  - a. note the application number, assumed job #1 for following steps
5. bg %1
6. Jobs
  - a. ensure this says 'Running'

The environment running the server is safe to be exited at this point and the server will continue to run.

If restarting the server or loading a new version, the old application needs to be stopped to save memory. The following steps will close the previous application before the above steps are used to restart it.

1. ps aux
  - a. Find PID that matches the command 'java -jar PhDTracker-0.0.1-SNAPSHOT.jar' assumed PID # 1234 for following

2. Kill 1234
  - a. If this does not stop the program view the troubleshooting section at the end of this manual
3. Restart the program

## 7.0 Database

MySQL is used as the database of choice for ease of use and extensibility. Two tables comprise this as the user and files. They are further described below.

### **7.1 Tables and Fields**

User

The user table stores all information regarding the users registered within the application. Fields are displayed in the following table:

Field	Type	Null	Key	Default	Extra
userNo	int	no	Primary	null	increment
userID	varchar(255)	no		null	
first_name	varchar(255)	no		null	
last_name	varchar(255)	no		null	
admin	boolean	yes		false	
advisor	varchar(100)	yes		null	
enrollment_staus	varchar(4)	yes		null	
term_activation	varchar(5)	yes		null	

Description of items:

- userNo is unused and simply numbers users as they are added
- userID, first/last names are all up to 255 characters and are required
- admin is a simple true/false for additional permissions
  - Defaults to false as students are the majority of new additions
- advisor, enrollment status, and term activation are optional on create/update but are limited to 100, 4, and 5 characters respectively

## Files

The files table stores all information related to uploaded files by users. Fields are displayed in the following table:

Field	Type	Null	Key	Default	Extra
uploadID	varchar(36)	no	Primary	null	
filename	varchar(256)	yes		null	
filetype	varchar(256)	yes		null	
data	longblob	yes		null	
upload_time	varchar(100)	yes		null	
userID	varchar(50)	yes	Foreign	null	
uploaded_as	varchar(100)	yes		null	

### Description of items:

- uploadID is a UUID generated by the server when a file is uploaded
  - It cannot be null and is how all files are accessed by the server
  - UUID values are a fixed 36 characters
- filename and file type is information associated with the uploaded file with a max of 256 characters
- data is where the contents of the file are stored
  - longblob type used for potentially large files
  - **AVOID** selecting this field if manually using the SELECT command in a mysql environment
- upload time is generated automatically by the server at the time of upload
- userID is the user that uploaded the file and tied to the user table for additional information
- uploaded\_as is the value assigned to the file for population purposes on the website and calculations within the server

# **8.0 Server/Database Troubleshooting**

## **8.1 Server**

### **Failed to compile**

Spring has color coded errors and significant stack trace reporting. There are a significant amount of errors that can cause compilation errors, though following the stack trace to recent changes is recommended. Before the stack trace it will show what has run and where it crashed.

Debugging is available by placing the breakpoint and running in debug mode as normal. It will run and await the end point http request which will then hit the breakpoint and debugging can be performed as normal.

### **Kill PID# command does not work**

It's possible the program hung or crashed when running or closing. If this happened use 'kill -9 {pid#}' to hard kill the application. This is not the recommended way to kill processes unless other options are not working.

### **Port already in use error**

If this error is received use the following steps on the command line:

- lsof -i
  - Find PID number of application, assumed 1234 for following
- kill 1234
  - If this doesn't work use the -9 flag as above

## **9.0 Conclusion**

On behalf of Team What's Up Doc, we thank you for reading through this extensive user manual and we hope it proves helpful in any and all situations that may arise in the future. Lastly, we wish our clients, Dr. Gretchen McAllister and Michele Benedict good fortune in the future iterations of the product and we hope to see it in use within the university as soon as the product is ready for production.

**Feel free to email any member of the development team with any questions regarding the product. Some of the development team may be staying on in Spring 2023 to see that the project reaches production value.**

With best wishes,

Team What's Up Doc:

Team Lead - Adam Larson ([al762@nau.edu](mailto:al762@nau.edu))

Brandon James Shaffer ([bjs397@nau.edu](mailto:bjs397@nau.edu))

Eddie Lipan ([eml292@nau.edu](mailto:eml292@nau.edu))

**Team Website:**

[https://ceias.nau.edu/capstone/projects/CS/2022/WhatsUpDoc\\_S22/](https://ceias.nau.edu/capstone/projects/CS/2022/WhatsUpDoc_S22/)