

[General](#)

[Overview](#)

[Debugging](#)

[Philosopher AI: Data Gen + Training](#)

[2.1 Setup](#)

[Tools \(Windows\)](#)

[2.2 Data Generation](#)

[2.3 Model Training](#)

[2.4 Conversion to GGUF, Quantization, Upload to HF](#)

[Rideshare](#)

[Plugin Customization Guide](#)

[Tools and References](#)

[Plugin Installation Instructions](#)

[Plugin Modification Instructions](#)

[If Updates are Blocked or Error Messages Appear](#)

[Overhauling the Plugin](#)

[Approach Attempted](#)

[Alternative Approach Used](#)

[Booking Calendar](#)

[Timeline](#)

[Major changes:](#)

[Booking Form](#)

[Timeline Integration](#)

[Navigation](#)

[Selecting Cells](#)

[Recurring Bookings](#)

[Areas for Improvement](#)

[Notable files](#)

[Login Information](#)

[Import Users](#)

[Login Security](#)

[Content Control](#)

General

Overview

-The Blount Compass site is a beta wordpress site located at <https://brahe.betatesting.as.ua.edu/>. It is restricted to on-campus IP addresses, or users on the UA VPN. It runs the latest version of WP, which is currently 6.7.1. PHP 8.2 and 8.3 are installed on the server. There's a MultiPHP manager within cPanel where you can switch versions. For any further questions regarding access to CPanel or the site hosting, our university contact is Amy Garner and can be contacted at amy.garner@ua.edu.

-There is a cPanel for the site located at <https://betatesting.as.ua.edu:2083>. This contains access to a number of different tools that are used to manage files, databases and more.

-The site is a multisite, so there is a separate admin page at <https://brahe.betatesting.as.ua.edu/wp-admin/network/>. This is where more plugins can be added and is separate from the regular admin page at <https://brahe.betatesting.as.ua.edu/wp-admin/>.

-From the main admin dashboard you can manage all the important aspects of the website including pages, plugins, users, settings, and more.

-The main plugins that are utilized throughout the site include Advanced Custom Fields PRO, Booking Calendar, Content Control, Frontend Post Submission Manager Lite, Import and Export Users and Customers, Limit Login Attempts Reloaded, Media Library Categories, OAuth Single Sign On - SSO (OAuth Client), User Role Editor, User Submitted Posts, WP Job Manager, and WPCode Lite.

-The site is broken down into four main sections: Academics, Involvement, News, and Tuscaloosa. You can always add more sections as needed with the addition of new modules. Each of these have their own main button on the home menu that has main navigation pages from there. This is where you can add buttons to go to new pages that fall under these categories. You should also add any new pages to the main navigation menu where they can be placed under the corresponding drop down.

-Much of the website theming was done in the addition css tab of Appearance > Customize. This Code simply adds on to the existing wildcat theme that was used and changes much of the look of the website. Additional CSS code can be added here if you want to make site wide changes to the theme. Inspect element is useful when making specific changes.

-Recommended VS Code extensions:

- GitLens by GitKraken
- IntelliPHP by DEVSENSE
- PHP by DEVSENSE
- PHP Profiler by DEVSENSE

Debugging

To turn debugging mode on, edit the wp-config file. The following four settings are relevant:

```
define('WP_DEBUG', false);  
define('WP_DEBUG_DISPLAY', false);  
define('WP_DEBUG_LOG', false);  
define('SAVEQUERIES', false);
```

To print to the debug.log, you can use the function `error_log(string $message)`. Using this in combination with [print_r\(mixed \\$value, bool \\$return = false\)](#) is useful for determining the states of objects or contents of arrays – when using `print_r`, `$return` must be set to `true` in order to print to the debug log. It is recommended to routinely download the debug log and then delete it from the site so that a new, empty debug log can be created; this makes it easier to track the desired output. A large debug log can also slow down the website.

Calls to `error_log` used by the previous team have been commented out and left in the plug-in files. Uncommenting these lines can provide insight into the order of function calls and structure of objects / arrays while getting started.

```
error_log( message: "bookings (243): " . print_r( value: $this->bookings, return: true ) );
```

To learn more about WordPress debugging, see [this page](#).

Philosopher AI: Data Gen + Training

2.1 Setup

Tools (Windows)

- **VS Code:** IDE with Jupyter, WSL, Python, C# extensions.
- **NVIDIA GPU Drivers**
 - You can download and install drivers for your graphics card from the “GeForce Experience” program or the NVIDIA website.
- **WSL :** Install and download WSL according to the official Microsoft guide (distribution: Ubuntu).
 - Installing the CUDA 12.1 SDK for WSL:

§ [Installation](#)



§

§ After running those 2 commands, run:

```
sudo apt-get update
sudo apt-get install cuda-toolkit
```

Python WSL

- Install Python 3.10

sudo apt-get install python3.10

- Install pandas, Jupyter and venv.

§ Use pip install or sudo apt-get install

- [Install PyTorch 2.3.0](#)

§

PyTorch Build	Stable (2.3.1)	Preview (Nightly)			
Your OS	Linux	Mac	Windows		
Package	Conda	Pip	LibTorch	Source	
Language	Python	C++ / Java			
Compute Platform	CUDA 11.8	CUDA 12.1	CUDA 12.4	ROCm 6.0	CPU
Run this Command:	pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu121				

- [Install Unsloth for PyTorch 2.3.0](#)

§

```
7. For Pytorch 2.3.0: Use the "ampere" path for newer RTX 30xx GPUs or higher.

pip install "unsloth[cu118-torch230] @ git+https://github.com/unslothai/unsloth.git"
pip install "unsloth[cu121-torch230] @ git+https://github.com/unslothai/unsloth.git"
pip install "unsloth[cu118-ampere-torch230] @ git+https://github.com/unslothai/unsloth.git"
pip install "unsloth[cu121-ampere-torch230] @ git+https://github.com/unslothai/unsloth.git"
```

Update PATH :

- Let's edit the bash_profile file so that these variables are added when WSL starts. Command to edit the file:

- `sudo nano ~/.bash_profile`

- Now add this text:

```
export PATH = " $PATH :/usr/local/cuda-12.1/bin"

export LD_LIBRARY_PATH = " $LD_LIBRARY_PATH :/usr/local/cuda-12.1/lib64"

export PATH = " $PATH :/home/USER/.local/bin"

alias jupyter-notebook = "~/local/bin/jupyter-notebook --no-browser"

alias python = python3
```

- And save with CTRL+X, Y, then Enter

2.2 Data Generation

1. Open WSL, then open VS Code.
 - a. You can open VS Code directly in WSL by typing 'code' in the terminal
 - b. You can also connect a VS Code instance running on Windows with some extensions
2. Open DataGenerator.ipynb
3. The second block of code contains most of the variables you should edit
 - a. Mainly, you'll change the output directory and file names. Set RESUME_INDEX to None when you first start on a dataset.

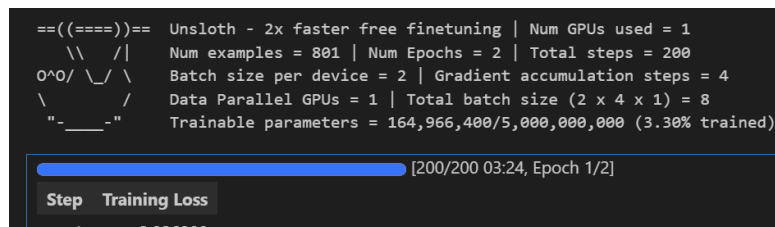
```
HF_DATASET = "yahma/alpaca-cleaned"
MODEL_NAME = "unsloth/llama-3-8b-Instruct"
OUTPUT_DIR = "Nietzsche_Training_Dataset"
OUTPUT_FILE = os.path.join(OUTPUT_DIR, "nietzsche_alpaca_" +
str(datetime.date.today()) + ".csv")
DEVICE = "cuda" if torch.cuda.is_available() else "cpu"
RANK = 300
ALPHA = 600
RESUME_INDEX = 801 # ← set to an int to force-start there,
or leave None for auto-detect
```

- b. The 5th block of code contains the prompt used to create the dataset.

4. It may take some time, but it's not necessary to process the whole original dataset to get a good training dataset. Be sure to tweak your prompt to ensure cleaner results.

2.3 Model Training

1. Open Train.ipynb in your WSL-powered VS Code.
2. Similarly, you'll find the variables that should be edited in the 2nd block of code.
3. The 5th block contains some other variables. The default of 2 training epochs produces good results.



```
==((====))== Unsloth - 2x faster free finetuning | Num GPUs used = 1
  \ \  / |   Num examples = 801 | Num Epochs = 2 | Total steps = 200
0^0/ \_/ \   Batch size per device = 2 | Gradient accumulation steps = 4
 \ \  /      Data Parallel GPUs = 1 | Total batch size (2 x 4 x 1) = 8
  "-__-"     Trainable parameters = 164,966,400/5,000,000,000 (3.30% trained)
```

[200/200 03:24, Epoch 1/2]

Step	Training Loss
------	---------------

a.

4. Once training is done, you'll see some stats, and the model will be saved in the safetensors format.

2.4 Conversion to GGUF, Quantization, Upload to HF

1. Once you've compiled llama.cpp:
2. Use the script 'convert_hf_to_gguf.py' to convert your safetensors model into a gguf.
3. Use the tool 'llama-quantize' to quantize the model. This will make it smaller and will speed up inference time.
4. Finally, you can upload the model to HuggingFace and add it to the list of Philosophers GPT (which is easily done by adding an entry to philosopherModels in the file philosopherLLM.html)

```

const philosopherModels = [
  {
    name: "Socrates - Qwen2 (0.5B)",
    prompt: "What is virtue?",
    model:
      "https://huggingface.co/Qwen/Qwen2-0.5B-Instruct-GGUF/resolve/main/qwen2-0.5b-instruct-q4_0.gguf",
    type: "GGUF_CPU",
    avatar: "res/Socrates.png"
  },
  {
    name: "Plato - TinyMistral",
    prompt: "<|im_start|>user\\nWhat is justice?<|im_end|>\\n<|im_start|>assistant\\n",
    model:
      "https://huggingface.co/rahuldshetty/llm.js/resolve/main/TinyMistral-248M-SFT-v4.Q8_0.gguf",
    type: "GGUF_CPU",
    avatar: "res/Plato.png"
  },
  {
    name: "Aristotle - LLaMa Lite",
    prompt: "### Instruction:\\nWhat defines a good life?\\n\\n### Response:\\n",
    model:
      "https://huggingface.co/rahuldshetty/llm.js/resolve/main/llama2_xs_468m_experimental_evol_instruct.q4_k_m.gguf",
    type: "GGUF_CPU",
    avatar: "res/Aristotle.png"
  },
  {
    name: "Nietzsche",
    prompt: "What is the will to power?\\n",
    model:
      "https://huggingface.co/Greenfire/Qwen2.5-0.5B-Nietzsche-alpaca/resolve/main/Qwen2.5-0.5B-Nietzsche-alpaca-Q4_K_M.gguf",
    type: "GGUF_CPU",
    avatar: "res/Nietzsche.png"
  }
];

```


Rideshare:

I used the WP Job Manager plugin, originally designed for job listings, and customized it to support a rideshare feature. By modifying text labels, template files, and saving custom ride-specific fields, I repurposed the plugin to allow users to post and manage ride offers instead of job postings.

Plugin Customization Guide

Tools and References

- **WP Job Manager Plugin:** <https://wpjobmanager.com/>
- **Official Documentation:** <https://wpjobmanager.com/documentation/>
- **Shortcode Reference:**
<https://wpjobmanager.com/document/advanced-usage/shortcode-reference/>
- **Template Overrides Guide:**
<https://wpjobmanager.com/document/developer-reference/themes/template-overrides/#top>

The Shortcode Reference and Template Overrides pages were particularly helpful for understanding the plugin's codebase. They helped determine which files to target when making specific customizations.

Plugin Installation Instructions

To install the plugin:

1. Visit the **Network Admin** site:
<https://brahe.betateesting.as.ua.edu/wp-admin/network/plugins.php>
2. Under the "Plugins" section, click **Add Plugin**.
3. Search for the desired plugin or click **Upload Plugin** if you have downloaded the file from the plugin's official website.
4. After uploading or finding the plugin, click **Network Activate** to activate it across the entire multisite network.

Plugin Modification Instructions

To modify the plugin's code:

1. Visit the **Network Admin** site:
<https://brahe.betateesting.as.ua.edu/wp-admin/network/plugins.php>
2. Under "Plugins," click **Plugin File Editor**.
3. In the dropdown menu next to **Select plugin to edit**, find and select **WP Job Manager**.
4. Click **Select** to open the plugin's code files for editing.

Important:

WordPress does not provide built-in source control for code. Always ensure that the current version of any file you intend to edit is backed up and stored in GitHub or another version control system before making changes directly in the Plugin File Editor.

5. After making changes, click **Update File** to save your edits.

If Updates are Blocked or Error Messages Appear

You may need to deactivate the plugin before you can edit its files.

Follow these steps:

To Deactivate and Network Deactivate:

1. In the **Network Admin** site, click **Installed Plugins**:
https://brahe.betatesting.as.ua.edu/wp-admin/network/plugins.php?plugin_status=active
2. Search for **WP Job Manager**.
3. Click **Network Deactivate**.
4. Go to the **Admin Site**, click **Plugins**, search for **WP Job Manager**, and click **Deactivate**.

After completing your edits, **reactivate** the plugin from both the Network Admin and Admin sites.

Overhauling the Plugin: Changing “Job” to “Ride”

Approach Attempted

Some sources suggest using **Poedit** to modify the plugin’s language files by generating **.po** and **.mo** files and uploading them to cPanel File Manager.

However, this method was not effective in this case and is not recommended based on experience.

Alternative Approach Used

The full transition from "Job" to "Ride" terminology in the WP Job Manager plugin was accomplished using **three methods**:

1. Direct File Editing (Outside of WordPress)

Some plugin files cannot be edited through the WordPress Plugin File Editor because they are treated as constant system files.

To modify these:

- Download the plugin’s ZIP file from the official WP Job Manager website.
- Extract the files locally on your computer.
- Use **Command + F** (or **Ctrl + F** on Windows) to search for every occurrence of "Job" or related terminology.
- Manually replace the terms as needed.
- Repackage or upload the modified files if necessary.

Note:

Direct editing can be overwritten by plugin updates. Always document your changes separately or maintain a custom version of the plugin.

2. Editing Through the Plugin File Editor

Some plugin templates and accessible PHP files can be edited directly within WordPress using the **Plugin File Editor**.

These edits include:

- Modifying output text
- Adjusting labels
- Minor template changes

Remember:

WordPress does not provide built-in source control for code. Always ensure that the current version of any file you intend to edit is backed up and stored in GitHub or another version control system before making changes directly in the Plugin File Editor.

3. Overriding Text Dynamically Through the Theme's `functions.php`

For text that is generated dynamically through translation functions (`gettext`), direct editing is not effective. Instead, we used WordPress hooks inside the active theme's `functions.php` file.

Theme Directory:

`public_html/wp-content/themes/as-wildcat/functions.php`

Purpose of the `functions.php` Modifications:

- Override specific plugin text at runtime (e.g., changing "job dashboard" to "ride dashboard," "Choose job type..." to "Choose ride type...").
- Add and save additional custom fields during ride listing creation (e.g., `ride_date`, `pickup_location`, `contact_method`).

Benefits of Using `functions.php`:

- Safe customization method without modifying plugin core files directly.
- Preserves changes even when the plugin is updated.
- Follows WordPress development best practices.

Booking Calendar

The timeline and booking form are based on the free version of the [WP Booking Calendar plug-in](#). The code behind the built-in [booking] and [timeline] shortcodes has been edited to provide the required additional functionality, but the basics on these shortcodes and their parameters can be viewed [here](#). The shortcode [bookingseries] is a custom shortcode added to the plug-in to allow admins to create recurring series of bookings and takes no parameters.

The plug-in is configured with all rooms being a single resource, so anywhere resource ID is required, the resource ID = 1.

Timeline

The timeline is used on the Room Schedules landing page as well as the Booking page. The parameters for type are to configure the timeline as a matrix during generation: all updates made to the timeline code were done to the timeline as a matrix and configuring it differently may result in errors or unintended behavior.

The bar to highlight the current time slot is done through assigning a cell with the class 'today_time'.

The primary functions of interest when generating the timeline are:

- wpbc_show_timeline_booking_row
 - Creates array with times slots in milliseconds and creates array of bookings within those times for the corresponding room
- show_day_cell
 - Determines the cell classes based on the booking array and time

Major changes:

- Display bookings by room instead of by day
- 30 minute intervals
- Correction of today_time calculation
- Addition of legend
- Interactivity with booking calendar

Booking Form

The booking form was initially configured through WordPress's built-in admin panel. Base settings can still be changed through this panel. Booking information is sent to two tables: `zindr_booking` and `zindr_bookingdates` on submission.

Note: with the timeline integration and auto-fill added in commit 12c6109, changes to the structure of the form may require edits to the source files.

Timeline Integration

Navigation

The timeline navigation function, `wpbc_flextimeline_nav`, is called when a date on the booking calendar is selected. This is in order to move the timeline to display that day's booking information. When called from the booking calendar, no parameter for `move_datepicker` is passed: we do not want to move the calendar's selected date since we are moving to the date that was selected from the calendar.

When the arrows are used to navigate with the timeline, the `move_datepicker` parameter is set to true so that the datepick function `_selectDay` is called. This is to move the selected day on the booking calendar with the movement of the timeline.

The functions for the datepick are in their own file, `jquery.datepick.wpbc.9.0.js`.

Selecting Cells

The booking form was changed to use the cell selected on the timeline to populate the dropdowns for selected time and selected room instead of allowing the user to set these values themselves. This was done to make time selection more straightforward on the user side, and to make time selection validation simpler from the development side.

During timeline generation, if a cell is not occupied by a booking or in the past, the class `available_time`, a class indicating the row number, and an onclick event called `start_booking` are added.

The `start_booking` function gets information associated with the selected cell to

1. Determine start time
2. Determine selected room
3. Check if the appointment time is valid based on the selected duration

4. If the time is valid, set the dropdown values

```
function start_booking(td) {
    const seminarRoomList = ["Main Hall", "OB 1006", "OB 1007", "OB 1012"];
    const parentElem = td.parentElement;
    const hourSelected = Number(parentElem.classList[1].slice(9)) / 2;
    const duration = document.getElementById("durationtime1").value.split(':');
    const roomSelected = seminarRoomList[Number(td.className.split(' ')[1])];

    const isValid = check_time_selected(td, parentElem, duration);
    if (!isValid) {
        alert("Invalid time. Please select different duration or start time.");
        return;
    }

    // Set dropdown values
    var startTimeVals = hourSelected.toString().split('.');
    const startTime = startTimeVals[0] + ':' + (startTimeVals[1] == '5' ? '30' : '00');

    document.getElementById("starttime1").value = startTime;
    document.getElementById("seminarroom1").value = roomSelected;
}
```

The helper function `check_time_selected` is also used with the `onchange` event for the duration dropdown to ensure a change to duration doesn't make an appointment time invalid.

`check_time_selected` is the function that adds the `user_selected` class, adding color to the cell, and the function that resets the dropdowns if the time is invalid.

Recurring Bookings

The page for creating a series of recurring bookings is not connected to the site navigation. It is published privately and available through the WordPress admin panel, or at <https://brahe.betateesting.as.ua.edu/recurring-bookings/>. All bookings created through this form are automatically approved. The date bookends for the series are inclusive; selecting a start date on 4/25/2025 for a series that occurs on Fridays will create a booking for 4/25/2025.

Submitting a new recurring series calls the function `ajax_WPBC_AJX_BOOKING_SERIES_CREATE` on the server side, which then runs a procedure in the database to create all occurrences of the new series.

Select Start Date*	Select Room*
<input type="text" value="04 / 25 / 2025"/>	<input type="text" value="Main Hall"/>
Select End Date*	Select Start Time*
<input type="text" value="mm / dd / yyyy"/>	<input type="text" value="8:00 AM"/>
Days Occuring*	Select Duration*
<input type="checkbox"/> Monday <input type="checkbox"/> Tuesday <input type="checkbox"/> Wednesday <input type="checkbox"/> Thursday <input type="checkbox"/> Friday <input type="checkbox"/> Saturday <input type="checkbox"/> Sunday	<input type="text" value="30 min"/>
<input type="button" value="Create"/>	

The procedure to create a recurring booking series is under the Routines tab of phpMyAdmin. It is called `create_recurring_booking`. It takes 11 parameters. It uses the helper function `gen_booking_hash` to create a unique hash for each new booking, based on an adapted version of the booking hash performed when a booking is created on the server.

Areas for Improvement

- Condensing the seminar room list into a single variable that can be used and modifying the admin-panel generation for the booking calendar to allow for editing rooms through the UI instead of through the code
 - Rooms available are currently static, and an array is hard-coded where they are needed.
- Select current day on the datepick when the datepick loads
 - Date must be manually selected from datepick or by navigating on the timeline before a booking is submitted, because no default date is selected. The date used in the booking submission comes from the datepick - this could be switched to use the date from the timeline since that is automatically set, if unable to set a default datepick value
 - Suggested file to start with: [calendar_load.php](#)
- Disable selectable cells on booking submission
 - Since interacting with the timeline after the booking is submitted will not mess with anything, this is simply for refinement and not important to functionality
 - Removing the onclick event and available_time class after sending the booking changed the values in the inspection window, but they reverted after the page finished loading. The reason why is unknown
- Fetch user first and last name from their log-in information
 - The email is automatically populated, but the function to pull the user information does not currently manage to pull the name
- Revisit current date / 'today_time' calculation
 - The date from the plug-in's formula was consistently 24 hours ahead. The fix applied simply subtracts 24 hours from their formula instead of correcting the formula itself.
- Make fields and shortcodes more dynamic
 - Examples:
 - Add parameter for whether the timeline should have clickable cells and don't add the available_time class to cells if not
 - Disable auto-filling fields from admin panel UI instead of static array
 - Loop when checking time validity instead of switch, to account for adding longer durations

Notable files

- Booking form/recurring series classes/generation: [wpdev-booking-class.php](#)
- Booking form/recurring series interactions: [client.js](#)
- Booking creation: [create_booking.php](#)
- Calendar interactions: [wpbc_all.js](#)
- Datepick: [jquery.datepick.wpbc.9.0.js](#)
- Shortcode parsing:
 - Booking form: [sh_tpl_booking_form.php](#)
 - Timeline: [sh_tpl_booking_timeline.php](#)
- Timeline class/generation: [wpbc-class-timeline_v2.php](#)
- Timeline interactions: [timeline_v2.js](#)

Login Information

The plugin used for OAuth capabilities is the MiniOrange OAuth Single Sign On located at <https://wordpress.org/plugins/miniorange-login-with-eve-online-google-facebook/> . This plugin inserts the sign in with microsoft option on the normal WordPress login screen and connects with microsoft azure app registration to connect crimson email logins using OAuth with users in the database. **The current client secret that is used in the configuration of OAuth is connected to a personal crimson email and expires 9/15/2025 and therefore the OAuth configuration must be redone before then.** The Instructions for OAuth configuration for this specific plugin can be found on the [MiniOrange setup guide](#). The plugin currently only is configured to allow office 365 login, but the plugin is capable of handling many different OAuth providers if that is desired.

The free version of the plugin has limited functionality, and therefore, edits were made to plugin code files. The main changes include allowing admins to login using the OAuth, only authorizing OAuth for already created users(whitelisting), and updating the attribute mapping. These changes were focused on the class-mooauth-widget.php file. Some keys to achieving these changes deal with \$username, \$username_attribute, if(\$user) else - for changing handling of unregistered users, and \$allow_admin_sso. It is also key to note that a lot of the important code is repeated twice in this file, so make sure changes made are consistent throughout.

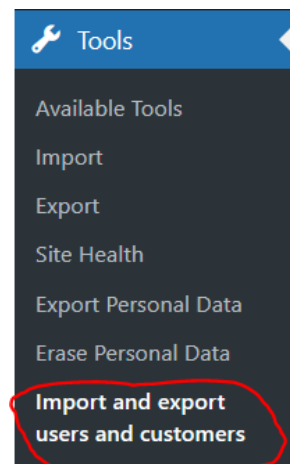
Because of the effective whitelist, in order to allow new students to login using MyBama, they should first be added as users in the database.

Import Users

This feature uses the import and export users and customers plugin. More information on the plugin itself can be found at <https://wordpress.org/plugins/import-users-from-csv-with-meta/> . There is more functionality to the plugin but what we want to use it for is to mass import new users into the database using a CSV file. Once the users are imported, they should be able to login to Microsoft OAuth using their MyBama credentials and crimson email.

To import new users into wordpress go to tools > import and export users and customers

Then, click browse... and select the corresponding CSV file. Please format the CSV files like so with the row having columns Username, Email, First_Name, Last_Name with Username being the crimson id and email being the crimson email:



	A	B	C	D
1	Username	Email	First_Name	Last_Name
2	piking	piking@crimson.ua.edu	Parker	King
3	slbird	slbird@crimson.ua.edu	Sue	Bird
4	jmsmith	jmsmith@crimson.ua.edu	Jane	Smith

All default values are fine, so then you can start importing. Now all the new users should be seen in the users tab.

Users can also be added one at a time in the users>add new user.

Student users that only have base access should be added under the subscriber role.

Login Security

Login security is currently handled by the [Limit Login Attempts Reloaded plugin](#). This plugin can be accessed in the admin panel in order to access data about login attempts as well as update the settings pertaining to login security.

WordPress' default login page is used and was edited to implement more secure input sanitization. The file is named wp-login.php. WordPress has different sanitization calls, but the one used for the username was `sanitize_user()`.

Content Control

A plugin named [Content Control](#) was used to limit certain pages to only logged in users. It can be accessed in the admin panel at Settings > Content Control. From here there is a restriction in place named student pages that creates a restriction on certain pages to only allow logged in users to access the pages. To add this restriction to more pages that are intended for students only you click edit on Student Pages, go to content, and add pages that you wish to restrict under the and condition. This plugin has capability for more complex content restriction if you wish, so feel free to play around with it if you want added functionality.

It is also worth noting that using the [WPCode](#) plugin, the admin bar was disabled across the site for subscriber user roles, the base role used for the students. The plugin can be accessed under the admin panel Code Snippets. Using this plugin you can definitely edit what the user sees based on role for more specialized content restriction. There is a much wider range of functionality that this plugin is capable of producing, so feel free to play around with it.