Blount Compass - Individual Contributions

Khushi Modi

Rideshare Feature

- Identified job board plugin to serve as the foundation for rideshare feature.
- Configured the job board plugin and made design decisions on how to flip and leverage its existing functionality for rideshare purposes.
- Analyzed job board plugin codebase
    - Large codebase of php files. Downloaded these files to our personal computers and stored them in GitHub. This allowed us to maintain and have source control over any changes and edits we make to the plugin's default code. Once we made any changes, we uploaded the updated version of the plugin to WordPress and activated it. Certain features in the plugin can not be directly accessed in the downloadable code file to customize, however it can be accessed through the file manager on cPanel. This requires an understanding of the plugin's file structure and PHP in order to navigate to the necessary files, make the correct changes, and write functions to override existing functionality to flip this job board plugin to suit rideshare needs. There were 2 main methods I attempted in order to do this. I researched and attempted both of these methods.
        - (1) Write a function to override strings.
            - Hook into the gettext filter. The gettext filter is applied to each string as it is retrieved, giving me the opportunity to modify it. I pass the plugin's text domain to the function and the string I want to modify. This string must appear in the function the exact same way it does in the plugin. Once the target string is detected, I replace it with my custom text and return it. If no changes are needed, I return the original text.
        - (2) Generate .mo and .po files to override the default language settings.
            - **.po Files (Portable Object Files):**
              These are human-readable text files where all the translatable strings (the original text and its corresponding translations) are stored. Translators use these files to update or modify the text as needed. They serve as the working document for translation.
            - **.mo Files (Machine Object Files):**
              Once the translations in the .po file are finalized, the file is compiled into a .mo file. This is a binary format that WordPress loads at runtime to quickly retrieve the translated strings. The .mo file is optimized for performance, making it faster for WordPress to process during page loads.
    - Created rideshare landing page using custom HTML and CSS
- Added function to send users an email notification that their listing has been approved by an administrator.
- Modified the subject of the auto-generated email that is created when you want to "Apply for Ride"
- Added custom fields to the job board plugin for rideshare needs

- Learned how data from custom fields needs to be saved and stored in a wordpress plugin in order to display them
- Ride listing displays date, preferred pickup location, and preferred contact method if the information was provided in the form.
- Added google maps hyperlink to preferred pickup location.

Project Website

- Revamped the project website (https://blountwebapp.github.io/)

Hannah Bray

- Set up WordPress debugging, added edited WordPress files to GitHub for version control
- Exported and trimmed WordPress .xml files for each snapshot
- Exported relevant SQL tables, procedures, and functions
- Identified booking calendar plug-in to serve as foundation for the timeline and booking system
- Analyzed booking calendar plug-in codebase
- Adapted timeline generation
    - Separated bookings by room instead of by day
    - Removed hour indicator from cells and created header row with hours
    - Divided timeline into 30-minute cells instead of hour cells
    - Customized CSS
    - Added timeline legend
    - Corrected calculation for current time (used for timeline bar)
- Configured booking plug-in and make design decisions about the booking form
    - Added timeline to booking page
    - Connected timeline navigation to booking calendar navigation
    - Added clickable timeline cells
        - Added cell class in timeline generation based on room availability and current date/time
        - Added JS function to auto-populate booking form fields when clicked
            - Colored cells for the selected time frame
        - Added availability validation on user changing appointment duration
        - Disabled user input for auto-populating booking form fields
- Set up recurring booking procedure and page
    - Adapted booking hash from the PHP in the plug-in files to be callable as a function on the mySQL server
    - Created mySQL procedure to create recurring bookings based on a set of parameters, between two given dates
    - Created custom booking form specifically for recurring bookings, to be accessed by admins only
        - Added new, custom shortcode for the form
        - HTML and CSS adaptations and customizations
        - JavaScript input parsing with input validation
        - Set up AJAX hook and function to create recurring booking

- Documented basic booking form and timeline functionality
    - Detailed changes that were made from the plug-in and why
    - Included potential areas for improvement

Andrés Aguilar Calderón

- Worked on RideShare plugin:
    - Designed and dictated plugin functionality and features.
    - Analyzed PHP codebased (and picked up the language) and provided base structure to Khushi
    - Edited shortcodes, variable names, and most of the UI
    - Other responsibilities shared with Khushi include keeping the github repo synced with WordPress and testing.
- Work on Philosopher AI Web app:
    - Created a philosopher-themed LLM web application for running local models using llm.js
        - Optimized for user flow and conversation readability
        - Designed the layout to evoke classical philosophical themes, using Garamond/Georgia typography and a warm beige/brown palette reminiscent of ancient scrolls and parchment
        - Ensured responsiveness of layout and proper handling of long model names in the dropdown
        - Added decorative Greek-style column imagery on both sides of the chat UI to reflect a “temple of wisdom” visual identity
    - Integrated front-end UI with actual backend functionality from llm.js, ensuring the web app loads and executes language models from Hugging Face directly in the browser
        - Instead of our sponsor paying for access to an LLM API such as ChatGPT, hosting an LLM on a powerful UA server or paying for an external server, the web app loads the language model directly onto a device’s browser cache and executes inference locally. This minimizes cost, and allows a ‘practically free’ AI service.
    - Implemented dropdown selector to dynamically choose between philosopher-themed models (Socrates, Plato, Aristotle, Nietzsche), each with unique prompt structures and personalized avatar support
    - Built a fully functional chat interface and appended messages from both user and assistant using HTML, JavaScript, and CSS
    - Created art assets: Philosopher pixel art icons, user icon, and greek columns
- (Pending) Work on Philosopher AI Training/Finetuning
    - Will use Python and Unsloth library under CUDA-enabled WSL2 to finetune a series of models imitating renowned philosophers
    - Small Language models (~500mb) aren’t as precise; to mitigate this I will use a large language model like Gemini or Deepseek to ‘teach’ the small model
        - Dataset will be generated by taking a series of inputs (regular human conversations) and generating outputs by feeding them to the large model and asking it to reply as ‘x’ philosopher.

        - Finetuned Models and datasets will be uploaded to huggingface

Nicholas King

- Updated OAuth login plugin code to include new login logic to whitelist users and grab user data to match specifications and add admin OAuth login capability
- Analyzing and troubleshooting site-wide error by manually disabling plugins files to bring the site back online
- Updating login screen security to implement sanitization of input as well as login attempt limit by IP and to include less specific messaging
- Inserted website wide snippets of code to give different views for non-admins
- Researched many different OAuth plugins and configured and tested final plugin that utilizes microsoft OAuth implemented using azure app registration
- Worked on website theme integration and design
    - added site wide CSS code to update color schemes, navigation, and removing unwanted features by hand
    - This required a deep dive to understand the css architecture for the ua wordpress template
- Created wordpress pages that were fully custom by using html and javascript code snippets in order create the pages exactly how imagined

Chandler Norman

- Wrote HTML and CSS for initial version of project website
        - Designed pages to host deliverables, group member bios, and front page
- Designed and maintained navigation links for WordPress site
        - Used a “quick links” side menu for easy navigation
- Kept in close communication with sponsor
        - Designated as point of contact for Dr. Whiting
        - Facilitated progress updates and feedback for every sprint and led discussions during meetings with sponsor
- Designed local and national news feeds
        - Researched, installed and utilized a plugin to handle user submitted articles
        - Set up queue for faculty approval of user submissions
        - Used WordPress post management to organize rotating feed of verified, curated articles for Blount students
        - This serves our sponsor’s primary request for tools that will help Blount students adjust to college life and the adult world
- Trained and tested a Socrates themed LLM
        - Wrote a training script in WSL2 for finetuning the Qwen2-0.5-instruct model from HuggingFace with the assistance of the Unsloth Python library and Llama.cpp quantification library.
        - The data used for this training was obtained from HuggingFace. It was cleaned and organized for this specific use case.
        - This process included several training sessions followed by loss analysis and spot checking using a test script.

        - The test script, written in Python, allowed me to load and run the model locally and thus interact with it in a manner similar to the end user. This way, I could test common questions the model might be asked.
    - Due to the fact that this model would be hosted in the user's cache, I needed to keep the model small. This required a fine balance between epoch count, warm-up steps, weight decay, and other hyperparameters during the training process
    - This model was uploaded to HuggingFace and then integrated into the Philosopher AI page hosted on GitHub.
- Used frames to attach externally hosted HTML Philosopher AI page to WordPress site
- Adjusted visual design for Philosopher AI HTML/CSS code