**BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY**

**ON-THE-JOB TRAINING**

**DAILY ACTIVITIES**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME: DOMINGO, JOHN KENNETH NICKO M.** | | **INTERSHIP SITE: WEAVERSOFT** | |
| **DATE** | **ACTIVITIES PERFORMED** | **PROBLEMS ENCOUNTERED** | **ACTION TAKEN** |
| February 24, 2025 | * Finished the multi-stage registration by catching error messages from the server side. * Fixed the task creation bug by updating the dropdown so that the output policy only displayed the output policy, not all policies. * Removed the validation for the required output policy, as it was not necessary. * Added a loading animation to the task creation button since the process took too long. * Applied Dyze's code to fix the appropriate payload for task creation, as it was not pushed on his end and was passed to me to prevent a merge conflict. * Fixed a bug in the test policy that caused a UI crash due to null values by adding more defensive checks. * Disabled the save button when no changes were made in the policy creation task. Since it was a step procedure, the disabled save was applied only to that step, but on the last page, it was enabled if any changes were made. * Cloned the Django server API for the React.js project. * Had a crash course with Sir Artem, where he walked us through the basics and the architecture of the system. * Used Docker to run the server, which included a global database and a user-exclusive database. * Learned that the URL pattern was set to /api/v1/ for future-proofing. * Discussed decorators, classes like permission classes and auth classes, models, and views. * Completed the first task on the server side by modifying the login response to include Django’s default first\_name and last\_name. | * Encountered an issue when initializing the app, which prevented it from running properly. | * Asked Sir Artem for assistance and found out that the database needed to be migrated first. * Performed database migration, created a superuser, created a tenant, and created a tenant for the test database. |
| February 25, 2025 | * Edited the Django edit profile functionality by adding the first name and last name. * Applied the changes to the client side (React.js) as part of my previous task. * Tested the functionality before submitting a pull request. * Reviewed several pull requests. * Fixed a returned pull request by ensuring the button was disabled after saving in a step. * Assisted Dyze in setting up the Django environment, as he had a slow internet connection during our crash course. | * N/A | * N/A |
| February 26, 2025 | * Edited the edit profile API to follow RESTful standards. * Initially kept the route as /api/v1/users/edit/ since it was already working, but changed it to /api/v1/users/ as per Sir Artem’s request to rely on standard HTTP verbs (GET, POST, PUT, DELETE). * Encountered limitations with Django’s ModelViewSet, which did not allow using PUT without an ID. * Overrode the function but later realized that modifying the original code and replacing it with a regular ViewSet provided more flexibility. * Used the ID from JWT instead of passing it as a parameter. * Optimized the code by removing unnecessary if statements for better readability. * Updated the client-side API call to use /api/v1/users/ instead of /api/v1/users/edit/. * Updated the register API by adding first name and last name to the request body. * Removed automated email verification since new users should be verified by admins. * Modified the send mail function to send the verification request to admins instead of the users. * Investigated errors in the edit profile API from development to production. * Discovered that an existing middleware required the new API to be registered, which caused it to work on localhost but fail in production. | * Faced issues overriding the PUT method in Django’s ModelViewSet without an ID. * The edit profile API worked locally but failed in production due to a missing middleware registration. | * Switched from ModelViewSet to a regular ViewSet for better flexibility and proper JWT-based authentication. * Updated the client-side API calls to align with the new API structure. * Adjusted the registration process to remove automated user verification and notify admins instead. * Identified the middleware issue and registered the API properly, ensuring it worked in both development and production. |
| February 27, 2025 | * Edited the Create New Task dialog. * Automatically opened the accordion after a file upload. * Ensured that if more than one file was uploaded, the first accordion opened by default. * Added a switch to allow users to copy and sync all reference data. * When enabled, all changes were applied to all files. * When disabled, reference data was not synced, meaning changes to one file were not applied to others. * When toggled from on to off, the reference data from the first file was copied to all files. * Fixed a bug that prevented additional file uploads after the first upload. * Fixed a bug where reference data was not cleared after changing or removing a policy. * Removed the display of the switch if there was no reference data. * Ensured that the newly uploaded file’s accordion opened automatically. * Edited the View Task component because JSX by default did not display boolean values. | * Data from the staging environment was cleared, preventing proper testing. | * Created a dummy JSON and replaced the payload to proceed with testing. |
| February 28, 2025 | * Set up the policy edit feature. * Fixed an issue where it displayed Rule No. instead of the field key name by analyzing the dynamic code and updating it. * Reviewed React components and their reusability. * Studied props and state management. * Explored React hooks such as useState, useEffect, and useContext. * Looked into React Router for handling navigation. * Revised conditional rendering techniques. * Reviewed serializers for converting querysets to JSON and vice versa. * Studied viewsets and how they simplify API development. * Looked into authentication and permissions in Django Rest Framework. * Explored query optimizations to improve API performance. * Revised middleware and request handling in Django. |  |  |