



Date	# Hours	Description of work
17.03.2025	3.067	<p>I implemented a detailed view for all listings, allowing users to access details for products, jobs, and projects with a click. Now, when a logged-in user clicks a listing, it correctly opens the detailed page. In contrast, an unauthenticated user is redirected to the login page before they can access any listing details.</p> <p>I also improved the marketplace page for all users to ensure a smoother experience when navigating between products, jobs, and projects. Previously, clicking the toggle buttons redirected users to the product details page because of the stretched-link class, which made the entire card clickable, including the buttons. To resolve this, I modified the clickable areas so that only the title or image of each listing is clickable.</p> <p>I was attempting to implement a "like" functionality that would allow users to like a product, with the number of likes being stored and updated in the database. However, despite numerous attempts to debug and fix the code, the functionality is not working as expected. The likes are not being saved in the database, and the user interface does not update correctly when a user clicks the like button. Even after verifying that the repository, controller, and JavaScript functions were properly configured, the issue still persists. At this point, I have decided to start from scratch and rebuild the like feature from the ground up to ensure it functions correctly.</p>
19.03.2025	2.75	<p>Today, I worked on implementing the purchase functionality for my project. I created a purchase page where users can buy products from entrepreneurs. This page includes seller information, product details, and a purchase form for users to enter their shipping and payment details. I ensured that the form properly handles user input, including selecting the quantity, filling in address fields, and adding payment information. Additionally, I integrated the form with the backend so that when a user submits it, the data is sent correctly.</p> <p>On the product details page, I added a buy button for all users and a "request custom amount" option specifically for business users. If the quantity of a product is zero, the buy button becomes hidden. I also corrected an issue on the homepage where the user who posted a product could see their own listing, which should not be the case.</p> <p>On the backend, I set up the PurchaseController to handle form submissions and process orders. I ensured that the data is saved correctly in the database and that the appropriate associations are made between the user, product, and seller. During this process, I debugged several issues related to missing fields and incorrect mappings in my entities. At one point, I encountered a SQL error because a required field was missing a default value, but I resolved it by updating my entity and database schema.</p>
20.03.2025	2.73	<p>Today, I developed a page that allows users to view their product orders and enables sellers to see who purchased their products. I created separate views for buyers and sellers. Users can access a "My Orders" page that displays a list of all the products they've purchased, including</p>

		<p>details such as quantity, product information, and shipping data. Sellers can access a "Buyers of My Products" page, which shows who bought each of their listed products. On the backend, I implemented the necessary methods in the "OrderController" to fetch the relevant transaction data based on the logged-in user's role.</p> <p>I also worked on implementing and debugging the save/unsave feature for products, job postings, and project postings on the NativeSpark platform. I built the logic for the save button using a toggle mechanism that updates the database through POST requests to endpoints such as /product/{id}/toggle-save, /job/{id}/toggle-save, and /project/{id}/toggle-save. Each listing entity has a savedByUsers field that maps a many-to-many relationship with the User entity, creating new tables in the database. This setup allows multiple users to save the same listing and enables users to manage their saved items.</p> <p>On the frontend, I added heart icons next to each listing card, which change color when clicked—red when saved and white when unsaved. The initial implementation visually toggled the heart correctly on click; however, I noticed that after reloading the page or navigating away and back to the homepage, the heart icon did not accurately reflect whether an item was saved.</p> <p>To address this issue, I improved the toggleSave function in the JavaScript code to read the server's text response (saved or unsaved) and update the heart icon to display  or  accordingly. This enhancement significantly improved the visual feedback after saving or unsaving an item.</p>
21.03.2025	1.55	<p>Today, I added the CustomProductRequest entity, along with the CustomProductRequestController, CustomProductRequestRepository, and the corresponding request_custom_amount.html page. This addition allows business users to request a custom amount of a product, and these requests are now saved in the database. I also included a success message on the page and added a "Back to Home" button to enhance the user experience.</p> <p>In addition, I worked on feedback to better distinguish job postings from project postings. I removed the "Contract" option from job postings to prevent any overlap. The feedback also emphasized the importance of fully implementing individual features before expanding them. As a result, I plan to revise the product purchase flow. Instead of allowing users to buy products directly, they will now be able to add products to a cart and purchase from there, making the functionality more complete and user-friendly.</p>

I implemented a detailed view for listings, enabling users to click and access product, job, and project details. Logged-in users can view details directly, while unauthenticated users are redirected to the login page.

I enhanced the marketplace by fixing the clickable areas so that only the title or image of each listing is clickable, avoiding previous issues with the toggle buttons redirecting users unnecessarily.

I attempted to add "like" functionality for products, but encountered persistent issues with saving likes to the database and updating the user interface. I've decided to rebuild this feature from scratch for better performance.

Today, I created a purchase page for users to buy products, which includes seller info and a purchase form. I made sure it correctly processes user input and integrates with the backend. On the product details page, I added a buy button and a "request custom amount" option for business users. I also fixed a bug that allowed users to see their own posted listings.

On the backend, I set up the PurchaseController to handle orders and ensure data is saved correctly. I resolved several issues, including SQL errors due to missing default values.

I added a "My Orders" page for users to track their purchases and a "Buyers of My Products" page for sellers to view buyers. I implemented necessary methods in the OrderController to fetch this data.

I also worked on a save/unsave feature for listings on the NativeSpark platform, creating a toggle mechanism that updates the database. I designed heart icons on the frontend that visually indicate saved items but realized they sometimes didn't update correctly after a page reload. I improved the JavaScript function to ensure the icons reflect the saved status accurately.

Lastly, I introduced the CustomProductRequest entity and associated components, allowing business users to request custom amounts for products. I added a success message and a "Back to Home" button for improved user experience.

The files/folders I have checked in the repo are as follows: Along with the developing the program different htmls, entities, and controllers were checked in.

ValeriiaN_Progress_Report.xlsx

ValeriiaN_Progress_Report3.docx