Date	# Hours	Description of work
03.03.2025	2.75	After logging into my account, I focused on displaying user-specific details based on different user types: Basic, Business, and Entrepreneur. Initially, I could only see the email and user type, but additional details like names, descriptions, and logos were absent. The issue stemmed from my AccountController, which was retrieving user-specific entities (e.g., BusinessUser, EntrepreneurUser, etc.), but the data was not being correctly passed to the Thymeleaf template. After debugging, I ensured that the correct user details were loaded and properly mapped. As a result, I was able to fix the missing information. However, the logo for business users still wasn't showing because the uploaded images were stored in the uploads directory rather than in the static directory, which prevented Thymeleaf from accessing them directly. I adjusted the image path handling to serve the uploaded images properly, and after that, everything displayed correctly. Additionally, I implemented a logout feature for all users. While users are logged in, they can navigate within the platform, including the "My Account" page, but once they log out, they cannot access the "My Account" page anymore.
04.03.2025	2.67	Today, I worked on implementing the subscription change feature within the user accounts of my web application. When users register, they receive a Free subscription by default. I wanted to give them the option to upgrade or downgrade their plan from their account settings. Initially, I encountered an issue where the subscription wasn’t updating in the database. I resolved this by ensuring the backend correctly retrieved the authenticated user and updated their subscription information in the database. Next, I faced a "bootstrap is not defined" error when trying to display the confirmation modal. This occurred because Bootstrap’s JavaScript file wasn’t loaded properly. I fixed this by adding the correct Bootstrap JS file just before the closing </body> tag. After that, I encountered another problem: the frontend failed with the error "Unexpected token 'S'" when saving the subscription. This happened because the backend was returning a plain text response instead of JSON. To fix this, I made the backend send a JSON response and updated the frontend to properly parse it.
05.03.2025	2.67	Today, I began by separating the controller logic for each user type: Business, Entrepreneur, and Basic. This allowed me to test and manually update the HTML form, ensuring that user data was being correctly updated. Initially, I entered the update paths manually in the HTML form for each user type to verify the data submission process. After confirming that the updates worked for each type individually, I combined all three user types into a single controller. This new setup would handle updates dynamically based on the logged-in user's type.

		However, I encountered an issue where the "Save" button did not appear when clicking "Edit," and the file input for logos/photos was not enabling for Entrepreneur and Basic users. I resolved this by adjusting the JavaScript to check the user type before enabling the appropriate fields, ensuring that only the relevant inputs were editable for each user type.
06.03.2025	1.88	I was working on enabling file uploads so that Business users could upload a logo, while Entrepreneur and Basic users could upload a profile photo. I also needed to ensure that these updates reflected on the account page. The main issue I encountered was that the file input fields were not activating correctly for different user types, and the uploaded images were not being saved to the database properly. I resolved this by adjusting the JavaScript `toggleEdit()` function to enable the correct input based on user type. Additionally, I updated the form to include both logo and photo inputs, and I modified the backend services and controller to correctly handle file uploads, saving them in the appropriate directories. I worked on implementing a transaction system to log subscription changes when users upgrade or downgrade their plans. Previously, changing a subscription only updated the Subscription table without retaining a record of past changes. To address this, I added a Transactions table to store the subscription type, cost, and timestamp whenever a user updates their plan. One challenge I faced was ensuring that transactions were recorded only for paid plans (Basic and Premium) while treating "Free" as a non-billable change. I resolved this by incorporating conditional logic in the backend to log only the relevant changes. Additionally, I tackled a UI issue where canceled profile edits still displayed modified (but unsaved) data. To fix this, I implemented a Cancel button that refreshes the page to reload the original database values. As a result, both subscription changes and profile edits now function smoothly, ensuring data accuracy and proper tracking.
10.03.2025	2.87	Renamed transations table to orders for future differentiation of transactions for the business and the customers. Also had an error when the table was named order. The error occurs because order is a reserved keyword in SQL. Since order is used for ordering results in queries, using it as a table name causes a syntax error. Today, I established the posting functionality for different user types. While the postings are not yet visible on the front end, I have successfully created the necessary tables in the database: job_postings, project_postings, and products. The controller is set up, allowing business users to manage job and project postings, while entrepreneur users can list products for sale. This functionality is role-specific, ensuring

		that businesses can add job and project postings, whereas entrepreneurs can add products along with images. I also implemented a toggle switch for business users, allowing them to easily switch between job and project postings while ensuring that the correct data is displayed. Additionally, I added a "Back to Account" button to enhance navigation. Access to "My Postings" has been restricted, so Basic users cannot see this option in the navigation menu. During this process, I encountered security role issues and repository mismatches, which I resolved by refining security configurations and adjusting database mappings.
11.03.2025	2.53	Today, I worked extensively on refining the functionalities for posting jobs, projects, and products in my Spring Boot application. I began by ensuring that business users can create job and project postings, while entrepreneurs can list their products for sale. I organized my Thymeleaf templates to properly display these listings, making sure that the images for product postings appear correctly and are aligned to the left. I encountered some issues during this process, so I improved the image upload logic for products. This ensures that uploaded images are stored correctly and displayed on the "My Postings" page.
12.03.2025	2.58	Today, I worked on implementing the delete functionality for job postings, project postings, and product postings. Initially, I encountered a "405 Method Not Allowed" error when attempting to delete a posting. This issue arose because the delete request was being blocked by CSRF protection in Spring Security. Additionally, using @DeleteMapping directly in the controller did not function properly with Thymeleaf forms, which default to sending POST requests. To resolve this, I modified the Thymeleaf delete buttons to include a hidden input field with _method=DELETE. This change allowed the form to properly send a DELETE request while still adhering to CSRF protection. I also updated the controllers to use @PostMapping for the delete endpoints instead of @DeleteMapping, ensuring that requests were correctly processed. I have implemented the Edit Posting functionality, making updates to both the frontend and backend of my application. On the frontend, I created an HTML page that maintains consistent styling with the job and project editing pages. Additionally, I added an image upload field, allowing users to update their product image while retaining the existing one if no new file is provided. On the backend, I made several important modifications to the ProductPostingController to support the editing feature. I introduced a GET mapping to load the existing product data into the form and a POST mapping to handle updates. The controller retrieves the product by its

		ID, ensuring that only authenticated entrepreneurs can modify their own postings. I also implemented logic to manage image file updates, so if a new image is uploaded, the old one is replaced; otherwise, the previous image remains unchanged.
13.03.2025	2.92	Today, I refined the home page functionality to ensure different types of users, including non-logged-in visitors, see the appropriate content. I solved an issue where unauthenticated users were wrongly redirected to the login page by adjusting the controller logic to keep the product list accessible to everyone. I also addressed a problem with product images not displaying for non-logged-in users due to incorrect image path references. I fixed this by updating the Thymeleaf syntax to properly serve images from the static resources directory. For entrepreneur users, I implemented a toggle bar to switch between products, job postings, and project postings. The challenge was keeping the clicked button visually active (highlighted in blue). I resolved this by using JavaScript to toggle Bootstrap classes, ensuring only the active button remained highlighted. Moreover, I fixed internal server errors (500) that occurred when entrepreneur users clicked "Home" in the navbar by updating the controllers to ensure the correct data was fetched for different user roles. Lastly, I improved the navbar behavior so that clicking "Home" directs users to the correct page based on their authentication status and role. Entrepreneurs can now see products, jobs, and projects, while basic users can only view products. These changes greatly enhanced the user experience and navigation consistency. I encountered an issue where removing some commented-out code caused the home button in the navbar to malfunction for unauthenticated users, redirecting them to the login page instead of displaying products. This was confusing since the home page should be accessible to everyone. Upon investigation, I found that the commented-out code contained crucial configurations that affected user authentication behavior. To fix the issue, I made sure products were visible to all users, allowing both authenticated and unauthenticated individuals to browse without needing to log in.

After logging in, I focused on displaying user-specific details for Basic, Business, and Entrepreneur users. Initially, only the email and user type were visible; however, I resolved the issue of missing details by correctly mapping user data in the AccountController. For Business users, logos were not displaying because they were stored outside the static directory, so I adjusted how the image paths were handled.

I implemented a logout feature to ensure that users couldn’t access the "My Account" section after logging out. Additionally, I made it possible for users to upgrade or downgrade their subscription plans. I fixed database update issues and addressed a "bootstrap is not defined" error to ensure smooth functionality. To resolve a frontend JSON parsing issue, I modified the backend to return a proper JSON response.

To improve account updates, I separated the controller logic for different user types, manually tested the update paths, and later merged them into a dynamic controller. I addressed issues with the "Save" button not appearing and file inputs not enabling correctly for Entrepreneur and Basic users by adjusting the JavaScript.

For file uploads, I enabled Business users to upload logos, while others could upload profile photos. I resolved input activation issues and ensured the backend properly handled the storage and display of images. I also implemented a transaction system to log subscription changes, renaming the "transactions" table to "orders" to avoid SQL errors associated with the reserved keyword "order."

I established posting functionality for different user types by creating database tables for job postings, project postings, and products. Business users can manage job and project postings, while Entrepreneurs can list products for sale. I added a toggle switch for Business users and fixed security role issues that were affecting access control.

I refined the features for job, project, and product postings, ensuring the proper organization of Thymeleaf templates and image uploads. I implemented deletion functionality, resolved CSRF protection issues, and modified Thymeleaf forms to correctly send DELETE requests.

For editing postings, I created a consistent user interface that supports image uploads and updated the backend to handle modifications. I ensured that old images remain unless they are replaced.

I improved the home page by making products visible to all users, fixing image path issues, and ensuring proper navigation for different user roles. A toggle bar for Entrepreneurs allows them to switch between products, jobs, and projects while keeping the active button visually highlighted. I also resolved an issue with the navbar where clicking "Home" caused internal server errors (500) and fixed an authentication-related problem where unauthenticated users were redirected to the login page instead of being shown the products.

By the end of these updates, I ensured that products were accessible to everyone while maintaining proper role-based navigation.

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_Report2.docx