**Project Report**

In this project, I focused on creating an application for monitoring factories (clothing and footwear). In the database, I created 8 tables (of which 2 are linking tables): Factory, Employee, Product, Material, Order, Client, OrderDetails, ProductDetails.

**Basic functionalities:**

I implemented a user-friendly page that allows viewing in table format and full control over data manipulation: by simply clicking on the value the user wants to change, they can rewrite and save values in the database. Additionally, at the end of each row in the table, there is a delete button that removes the respective row along with all fields in other tables that are directly linked to it. I also implemented an "Add" button that allows inserting a new empty row, which will then be populated with values by the user and saved.

**Extra functionalities:**

At the top of the page, I placed a search bar with a domain selector to choose the domain where we want to search for data. Once the search button is clicked or Enter is pressed, the table of the searched domain only displays the entities that contain (in any of the fields) the search query entered in the search bar. If no results are found after the search, an error message will appear instead of the table rows. When deleting an instance, a pop-up window asks for confirmation to delete by pressing one of the two buttons, "Yes" or "No." If an attempt is made to enter incorrect values in the table (e.g., a phone number that is not from Romania or an email without the "@" character, etc.), an error message alerts that one or more of the entered values are incorrect. Additionally, access to the site is done only through the "login" page, which is also connected to the database and allows the user to log in only if they are on the registered users list, have entered the correct password, and have administrator privileges. Otherwise, an error message will be displayed.

**Project structure:**

The project includes a "controllers" folder, which contains the two controllers for managing the "login.html" and "welcome.html" pages; an "entities" folder that contains all the database entities and other helper classes for the logic of these entities; a "repositories" folder containing one repository for each entity, with functions for querying the database; and finally a "services" folder, which contains only the "Verifications" class, holding all the logic behind validating the data entered by the user. The controller functions use annotations such as "@GetMapping" or "@PostMapping" followed by a form name from HTML for connecting with the HTML page and acquiring/returning data. After each function execution, the page is returned in full, with necessary changes added. The validation functions use regex expressions and are responsible for checking phone numbers, gender of employees/clients, email addresses, and also ensuring that the new entity introduced is linked to another valid existing entity (for example, a new employee cannot be assigned to a factory with ID 2 if this factory does not exist).

**Potential improvements:**

Regarding the login page interface, the option to create a user account is not available. New users need to be manually added to the database. Also, an interface that would not only be for monitoring purposes, but also for potential clients is needed (though this is a purely educational project).

For searching elements in tables, it would be useful to have a "dropdown" list below the search bar that updates in real-time as the user searches for something.

Another small issue, which time did not allow me to correct, is with validations. When a user tries to change multiple fields without saving them by pressing the Enter button or the "Apply Changes" button, and one of those fields happens to be invalid, the application will reset all the fields, including the correct ones.

Thank you for the interest in reading this report!