Course Project Specification

For the course project, I would like to develop a management system for a logistics firm.

The system’s ultimate goal is to supply employees with all necessary information on products, client complaints, trucks and vans, routes, and deliveries. It will allow authorized personnel to add, edit, delete and search for different entities within the system and control its security levels (permissions) for all employee types. Users will be able to browse through lists of various entities, sorting them according to their properties, including their ID number, description, and name. They can perform operations such as search by a keyword or select and deselect columns, thus applying the searching and sorting algorithms covered in class. Moreover, the program will allow searching through the list of employees by their respective permissions.

Considering a logistics company typically comprises of administrators and drivers, the following hierarchy suits this type of system the most:

1. The first level consists of an Employee class with an ID, a name, an email, a position, and a set of permissions. Each employee receives the ability to view and search through the list of all employees, as defined by the class’ polymorphic function setting the permissions property to a string vector of allowed actions.
2. The second level contains the Administrator and Driver classes that inherit Employee’s properties and override its polymorphic function to set the employee’s permissions depending on the selected type. The Administrator class overrides the function adding the ability to access and search through all lists. In contrast, the Driver class enables users to only view and interact with the lists relevant to the drivers, such as products, routes, and deliveries.
3. On the third level, the program incorporates specific types of administrators, divided into super admins, managers, clerks, and drivers, whose subcategories include truck drivers, van drivers, and supervisors. The application grants super admins the right to access and update all lists, define or alter user permissions, and manage employees. The polymorphic function will also reflect the changes enacted by the super admins when setting the allowed actions of the employee types. The function will enable managers to interact with and update all system categories, except the security levels. Clerks will be able to manage products, client complaints, and deliveries exclusively. Truck and van drivers will directly inherit their permissions from the Driver class, differing only in the type of vehicle list they can view. In other words, on top of the categories they can view by default, truck drivers could only open the list of trucks and van drivers - the list of vans. On the other hand, driver supervisors will serve both as a clerk and a type of driver, receiving all their rights plus those rights granted to them by a super admin.

To reflect user changes, the application will use a text file as a database containing the default permissions for the employees and their subtypes, together with an initial user in the role of a super admin and a set of allowed categories. Once the super admin “logs in,” the program will permit creating new users with various roles, security levels, and entities within each of the categories. The program will sort each entity by ID when accessing each list, utilizing one of the covered sorting algorithms. The user can then choose to either sort the list by another property, filter it by a chosen keyword with the help of one of the recommended pattern-matching algorithms, or select and deselect which properties to display (for instance, only output employee names). Authorized users can also add, edit, and delete entities from their respective lists, which will be reflected in the text file. The polymorphic function defining the permissions will decide whether the given user can open and manage each list. Once the user “logs out,” the program terminates, and in case it runs again, it will load the altered data from the input file.