**Health and Safety Reporting system - Multi-threaded TCP Server Application**

I have outlined the design decisions I made during the development of the Health & Safety Reporting System and I explain the functionality of the code. The project involves a client-server architecture, enabling employees to log in, create health and safety reports, retrieve assigned reports, and perform related operations.

System Structure: My project uses a client-server architecture because it allows multiple users to connect to a central server, ensuring scalability and centralized control.

Data Transfer: Java Object Serialization for communication between the client and server because it makes it easy to send complex data like reports and users without extra processing.

Data Storage: Text files for storing user and report information because text files are simple and lightweight, suitable for a small-scale system. They are also easy to inspect.

Unique Identifiers: Reports are given 8-digit numeric IDs because these IDs are easy for users to manage and ensure each report is uniquely identifiable.

Error Handling: Input validation and exception handling are built in, to ensure the system runs smoothly and provides helpful feedback for errors.

**Client Class:** Manages user interactions.

* run(): Displays menus and processes user inputs.
* sendMessage(String msg): Sends data to the server.

**ServerThread Class**: Handles client connections and processes requests.

* run(): Handles login, menu navigation, and actions like creating or retrieving reports.
* handleLogin(): Authenticates a user by verifying their email and password against the stored user data. If successful, grants access to the system; otherwise, prompts the user to try again.
* handleRegistration(): Allows a new user to register by collecting details such as name, employee ID, email, and password. Ensures no duplicate email or employee ID exists before adding the user to the system.
* RandomNumberGenerator: A class that generates a unique 8-digit numeric ID, used for report identification.
* createHealthAndSafetyReport(): Allows users to create new reports.
* retrieveAccidentReports(): Returns all accident reports.
* assignReport(): Assigns reports to employees by employeeID and updates their status.
* viewMyReports(): Displays all reports assigned to the logged-in user.
* updatePassword(): Allows a logged-in user to change their password by verifying their current password and updating the stored value if the input is correct.

**UserList Class**: Manages user data and authentication.

* loadUsersFromFile(): Reads all user data from a file (UserList.txt) during system initialization to populate the user list in memory.
* addUser(): Adds a new user to the list after ensuring their email or employee ID is not already registered and updates the file.
* updateUserPassword(): Updates a user’s password.
* saveUsersToFile(): Writes the current list of users back to the file, ensuring any changes (like a new registration or password update) are stored.
* findUserByEmailAndPassword(): Verifies login credentials.
* isEmailOrEmployeeIDExists(): Checks if an email or employee ID already exists in the user list to prevent duplicate entries during registration.

**HealthAndSafetyReportsList Class**: Manages the list of reports, including loading from and saving to a file.

* loadReportsFromFile(): Reads reports from the file.
* addReport(): Adds a new report to the list and saves it to the file.
* saveReportsToFile(): Updates the file with the latest report data.

**HealthAndSafetyReports Class**: Represents individual reports.

* Includes type, ID, assigned employee, and status.
* Tracks whether a report is open, assigned, or closed.