*Design Choices for my Operating Systems Project | Fionn McGoldrick | G0042249*

1. *User and UserManager Classes*

**Purpose:**

* The User class represents a user with attributes such as name, email, employee ID, department, and role​.
* The UserManager class handles operations such as registering users, searching for users, and managing employee IDs​.

**Design Decisions:**

* Encapsulation: Attributes in the User class are private, and getters/setters are provided to ensure controlled access and modifications.
* Reusability: Methods like registering and login in the UserManager class allow multiple User objects to use the same logic, reducing redundancy.
* Validation: Methods like emailExists and handleEmpID enforce rules for uniqueness and format, ensuring data consistency in the UserDatabase.txt file​.
* Separation of Concerns: User-related data and logic are separated into the User and UserManager classes, making the code modular and easier to maintain.

1. Report and ReportManager Classes

**Purpose:**

* The Report class encapsulates report attributes such as type, ID, date, and status​.
* The ReportManager class handles the storage, retrieval, and validation of reports​.

**Design Decisions:**

* Serializable Reports: The Report class implements Serializable, enabling objects to be transmitted between the server and client using streams.
* Validation: The ReportManager ensures that only valid reports are saved and supports options to handle different report types and statuses.
* File-Based Storage: Reports are saved in reports.txt for persistence, ensuring data is retained between sessions​.
* User-Specific Operations: The viewUserReports method retrieves reports based on the email of the creator, ensuring personalized data access​.

1. Client-Server Architecture

**Requester (Client):**

**Purpose:**

* Acts as the client, allowing users to interact with the server for operations like login, registration, and report management​.

**Design Decisions:**

* **User Prompts:** The Requester prompts users for input and validates it locally before sending it to the server, improving user experience and reducing server load.
* **Stream Handling:** Uses ObjectInputStream and ObjectOutputStream for structured communication with the server.

**Server and ServerThread:**

**Purpose:**

* The Server class listens for incoming connections, and the ServerThread handles client-specific operations concurrently​​.

**Design Decisions:**

* **Concurrency:** Each client connection spawns a new thread, ensuring simultaneous handling of multiple clients.
* **Switch-Based Menu:** The ServerThread uses a switch statement for handling commands like REGISTER, LOGIN, and VIEW ALL REPORTS, making the code extensible and readable​.
* **Centralized Logic:** The server delegates report operations to the ReportManager, keeping business logic separate from the networking layer.

1. File Management

**Purpose:**

* Files like UserDatabase.txt and reports.txt store user and report data persistently.

**Design Decisions:**

* **Buffered Streams:** Using BufferedReader and BufferedWriter improves file handling efficiency.
* **Validation:** Lines in files are validated to ensure correct format before processing, reducing the risk of runtime errors​​.
* **Recovery Utility:** Functions like repairDatabase can be implemented to handle malformed data or clean up files when errors are detected.

1. Error Handling and Logging

**Purpose:**

* Identify and recover from errors during runtime, ensuring stability and debugging support.

**Design Decisions:**

* **Error Logging:** Critical errors are logged using System.err, helping developers trace issues during debugging.
* **Graceful Handling:** Operations like viewUserReports and employeeIDExists skip malformed lines and continue processing valid data, ensuring robustness​​.

1. Future Improvements

* **Switch to a Database:** Transitioning from file-based storage to a relational database for scalability and complex queries.
* **Secure Data Handling:** Encrypting sensitive data like passwords for improved security.
* **Unit Testing:** Adding tests for critical components like UserManager and ReportManager to ensure reliability.