ENSE 375 EduTrack Project - Solution

3 Solution

We will describe each of the solutions that our team came up with to implement the EduTrack project. Some solutions may lack some desired features, others may not satisfy the constraints. Eventually, we will select a solution that we feel has all the features and satisfies all the constraints.

3.1 Solution 1

Description:

We brainstormed the first solution: a simple command-line application to serve all the basic functionalities of the EduTrack project using in-memory data structures. This solution will entail using Java collections like Lists and Maps in order to store and manage data for assignments, schedules, and reminders.

Why the team did not choose this solution:

- **Low Data Persistence:** Using in-memory means all data is lost on the application closure, not practical for users.
- **Potential Scalability Issues:** Managing large volumes of data in memory can become inefficient and problematic.
- **Security:** Using in-memory storage with no robust data security mechanism.
- **Constraint Satisfaction:** This solution does not meet all the basic persistence and data handling in the goals of the course and the project.

3.2 Solution 2

Description:

This is the improved form of the first solution. It is still a command-line kind of interface but will have a more structured way of handling data in memory. This will be done through Java objects and classes in data entity representation, namely, assignments, courses, and

schedules. It also allows basic encryption for sensitive data and more than basic testing frameworks.

Improvements over Solution 1:

- **Structured Data Management:** Using Java objects and classes allow for more organized and maintainable code.
- **Enhanced Security:** Basic encryption to the sensitive data stored in memory.
- **Focusing on Testing:** This solution is better aligned with the course focus on software testing and unit tests by allowing for comprehensive testing of the individual components.
- **Constraints:** The solution meets the project requirements without overcomplicating the development process.

Reasons why this solution will not be selected:

- **Limited Data Persistence:** The problem remains exactly the same as the first solution in that data will still be lost once the application is closed, which is again a significant drawback.
- **Scope Limitation:** It may be an improvement over the first solution, but there is no data persistence, and some of the features used in practical applications are missing.

3.3 Final Solution (Chosen)

Description:

This is the final chosen solution and a derivation of the second solution but done with a more focused idea of in-memory data management and a command line interface. The critical thing about this final solution is the prime necessity of the course regarding giving importance to software testing and unit tests, so no database or file storage is used. It has a well-structured approach to representing data entities, secure handling of sensitive information, and comprehensive test coverage.

Why this solution will be selected:

- **Subject Focus:** Aligns with the course's aim of understanding software testing and unit test concepts, and not the development of a solution.
- **Easy to Test:** It keeps data in memory and has a command-line interface, so we can write and execute unit tests without any hassle for each component.
- **More straightforward and Clear:** The lack of database and file storage simplifies the development process, and the whole focus remains on testing and validation.
- **Satisfies Constraint:** Since the main requirements and constraints are stated at the beginning of the document and it does not introduce anything that violates them, the solution will satisfy the stated constraints.

Features and Constraints Satisfied:

- **Functions:** Assignment tracking, study session scheduling, reminder setting, progress tracking, course organization, and task management.
- **Objectives:** User-friendly, efficient, reliable, customizable, informative, and secure.
- **Constraints:** Platform compatibility, data security, performance, usability, compliance, accessibility, scalability, and maintenance.