EECS 3311 Software Design
Winter 2024 Section N
Project Deliverable I (**100 points** *i.e. 10% of the overall grade*) Version 1
Release Date: Jan. 21, 2024

Due: 11:59 PM, Monday, Feb 12, 2024

In this project, you will learn to design software from scratch, i.e., analyze software requirements and use cases, produce sequence diagrams as a requirements specification, decompose a complex problem into sub-problems, start understanding and addressing software architecture design. In Deliverable-I, you are asked to analyze the raw user requirements from clients and create use cases, activity diagram, and sequence diagrams to illustrate the requirements with more detailed, after that you are further asked to create a class diagram that meets the requirements. Note, no design patterns are required at Deliverable-I, we will explore the effect of design patterns on your project in Deliverable-II. The estimated working time required for this deliverable is 5-10 hours.

Check the Amendments section of this document regularly for changes, fixes, and clarifications.

Ask questions on course slack channel or emails.

Policies

Your (submitted or un-submitted) solution to this assignment (which is not revealed to the public) remains the property of the EECS department. Do not distribute or share your code in any public media (e.g., a non-private Github repository) in any way, shape, or form. The department reserves the right to take necessary actions upon found violations of this policy.

- You are highly encouraged to **work in a group for this project deliverable**.
- When you submit your solution, you claim that it is solely your work. Therefore, it is considered as an violation of academic integrity if you copy or share any parts of your code or documentation.
- When assessing your submission, the instructor and TA may examine your doc/code, and suspicious submissions will be reported to the department/faculty if necessary. We do not tolerate academic dishonesty, so please obey this policy strictly.
- Emailing your solutions to the instruction or TAs will not be acceptable.

Submitting Your Work on eClass

Zip your submission named with the assigned team id of your group. **A 5% penalty will be applied if not follow this rule.**

Amendments

- so far so good

The "system-to-be": YorkU Library Management Java App

"YorkU Library Management Team is now seeking a new system that can help them provide better online services to their clients (e.g., students, faculty members, non-faculty staffs, and visitors). The system is supposed to be a GUI-based Java application. The basic requirements of the system (from an interview with their management teams) are as follows:

- **Req1:** Any client should be able to register as a user of the system with a unique/valid email and strong password (i.e., a combination of uppercase letters, lowercase letters, numbers, and symbols). The system currently allows four types of clients to be registered, i.e., students, faculty members, non-faculty staffs, and visitors, while it's open for new types in the future. If a client registers as a student, a faculty member or a non-faculty staff, her/his registration requires a further validation from the management teams.
- **Req2:** Using the system, any registered client can rent a physical item (i.e., books, magazines, CDs), open an online book, or subscribe to an online university-provided newsletter (e.g. NY Times), etc. Each physical item has 20 copies in the library. Penalty will be applied if a book is overdue (i.e., 0.5$ a day). A user can borrow up to 10 physical items and can keep a item for at most 1 month. All physical items borrowed from the library (books, magazines, CDs) count toward the total of 10 items. A user will lose his borrowing privileges if he has more than 3 items overdue. Books that are 15 days overdue will be considered lost.
- **Req3:** After login, the system should show a list of hardcover books that a user is currently renting and the due date for returning the books. It should also prompt warnings about any book that is not returned yet and it is approaching (less than 24 hours until the due date) or past the due date.
- **Req4:** The system should allow a user to subscribe and read a paid-for newsletter via its interface, such as the NY Times. This can be done by opening a frame within the system where the NY Times website can be loaded. A user can decide at any time to cancel a newsletter subscription.
- **Req5:** A user can search for a book using the application. For a book a user is searching, the app should also show recommendations of similar other books (based on the text similarity of book titles).
- **Req6:** If a user is a faculty, the app can keep track of the courses the user is teaching and the textbooks the user has previously used. The app then offers notifications to the user when a new edition of the textbook is available. If a textbook is not available, the app should notify the library management team of this, so that they could consult with the user to procure the book.
- **Req7:** Each item has a unique identification number and other details including its location in the library and whether the item can be purchased, which will help with the navigation for clients. Managers of the system can add, enable (can be rented), or disable (cannot be rented) an item.
- **Req8:** If a user is a student, the textbooks of a given course the student is taking, the app should make virtual copies of the textbooks available to the account of the user for the duration of the course. After that, the app should remove the virtual copies from the student account.
- **Req9:** A user can request for a new book. A request can be of two types, i.e., textbooks for course teaching and self-improvement, etc. Depending on the type, the request will need to be prioritized by the app and the user should be notified of the priority accordingly. Often, textbooks for course teaching will be given higher priority.
- **Req10:** The system could also offer discounted purchases of items via its special agreements with publishers, whose books/DVD are not normally freely available via the usual library management system. For this, the system needs to provide payment options like debit, credit, mobile wallet, etc.
- **Req11:** System data are stored in database, we will use Csv/Excel files to simulate this process.

Part I: Requirements Eliciting and Modeling (75 points)

Information from the interview might not cover all the requirements, you can make your own assumptions if necessary.

Please draw the diagrams carefully and present as many details as possible. A real-world developer should be able to develop the system based on these diagrams.

Task1: Use Case Diagram (25 points)

Your first task is to identify the main actors in the system. Please also indicate the sources of your use cases (i.e., from which Reqs). In your report, you are expected to clarify how each Reqs can be achieved with the use casesyou designed. Use a UML tool of your choice to draw a use case diagram based on the identified actors and use cases.

Task2: Activity Diagram (25 points)

Based on the use case diagram drawn in **Task1,** please select five use cases and for each of them draw an activity diagram accordingly.

Task3: Sequence Diagram (25 points)

Based on the use case diagram drawn in **Task1,** please select five use cases and for each of them draw a sequence diagram accordingly.

Part II: Design (25 points)

The diagrams you drew in **Part I** can help developers understand the basic requirements and functionalities of the 'system-to-be'. In this part, we will further explore how to design the 'system-to-be'.

Task4: Design (Class Diagram)

Before we start the implementation task, you are asked to design the system via drawing a detailed class diagram. Please identify all the possible interfaces, abstract classes, classes (including their attributes, methods, and relations among classes) and further draw the class diagram of the 'system-to-be'. Make sure you specify the multiplicity of relationships when applicable. You must also make sure the possible interfaces, abstract classes and classes of your class diagram are connected using the most suitable relationships. In case of potential modeling ambiguity, please use UML comments to clarify such ambiguities.

If you see any opportunity to apply SOLID principles, well, seize it!

Note that using design patterns at current stage is not mandatory (i.e. is optional). We will explore how we can use design patterns to improve your designs in Project Deliverable II.