Arian Ghazanfariyan
March 18, 2024
EECS3311
Deliverable 2

- **Singleton Pattern:**
  This pattern ensures that certain classes have only one instance throughout the program. For example, in our library system, we might use it to make sure there's only one manager handling the library's **database**.

- **Facade Pattern:**
  Think of this as a receptionist at a hotel. The **UserManager** acts like that receptionist, hiding the complexity of handling user-related tasks behind a simple interface. It makes it easy for clients (users) to interact with the system without worrying about the details of user management.

- **Factory Method Pattern:**
  Imagine a factory that produces different types of products. Similarly, in our system, the **LibraryItemFactory** is like that factory, responsible for creating various types of library items. Concrete factories, like **BookFactory** or **MagazineFactory**, handle the creation of specific types of items, like books or magazines.

- **Observer Pattern:**
  Picture a watchful guardian who alerts you when something important happens. In our system, the **BorrowingSystem** acts like that guardian, keeping an eye on borrowed items. When an item becomes overdue, it notifies the user and the management team.

- **Strategy Pattern:**
  This pattern is like having different strategies for tackling a problem. In our system, the **PaymentProcessor** might use different strategies to handle various payment methods, like credit cards, debit cards, or mobile wallets, giving users different options to pay for items.

- **Dependency Injection Pattern:**
  Imagine you need a tool to complete a task, and someone hands you exactly what you need. In our system, classes like **ConcreteLibraryItemFactory** depend on other classes, like Item, to do their job. Think of it as getting the right tools (dependencies) to get the job done.

Component Decomposition:

1. **User Interface Component:**
   - Responsible for handling user interactions and presenting information to the users.
   - Includes user-facing features such as registration, login, browsing library items, borrowing items, etc.
2. **Business Logic Component:**
   - Contains the core business logic of the library management system.
   - Includes classes such as ClientManager, BorrowingSystem, BookRecommendationSystem, CourseTextbookManager, ItemPurchaseManager, PaymentProcessor, etc.

- Handles user management, borrowing system, book recommendations, course textbook management, item purchases, payment processing, etc.

3. **Data Management Component:**
    - Responsible for managing data storage and retrieval.
    - Includes the DatabaseManager class for reading and writing data to SQLite files.
    - Manages user information, library items, borrowing records, etc.

Interactions:

- The User Interface Component interacts with the Business Logic Component to perform various user-related actions such as registration, login, browsing items, etc.
- The Business Logic Component interacts with the Data Management Component to retrieve and store data related to users, library items, borrowing records, etc.
- The Business Logic Component orchestrates the overall functionality of the system by coordinating interactions between different components.
- Components communicate with each other through well-defined interfaces, ensuring loose coupling and modularity.

Justification of Decomposition:

- **Separation of Concerns:** Decomposing the system into separate components based on user interface, business logic, and data management helps in separating different concerns, making the system more maintainable and understandable.
- **Modularity:** Each component encapsulates a specific set of functionalities, making it easier to understand and modify without affecting other parts of the system. This modularity enhances scalability and facilitates future changes and updates.
- **Flexibility:** By decomposing the system into components, it becomes easier to replace or upgrade individual components without impacting the entire system. For example, if we need to switch to a different data storage mechanism, we can update the Data Management Component without affecting other parts of the system.
- **Ease of Testing:** The modular architecture allows for easier testing of individual components in isolation, leading to more effective testing and debugging processes.

Req1: User Registration and Authentication:

- **Justification:** The UserManager class, within the Business Logic Component, handles user registration and authentication functionalities. When a user registers, the UserManager validates the provided email and password, creates a new user account, and stores it in the database. During authentication, the UserManager verifies the user's credentials against the stored information in the database.

Req2: Item Borrowing and Return:

- **Justification:** The BorrowingSystem class manages the borrowing and returning of items. When a user requests to borrow an item, the BorrowingSystem checks the availability and eligibility of the user based on borrowing limits and overdue items. If eligible, it updates the database to reflect the borrowing status. Similarly, when a user returns an item, the BorrowingSystem updates the database accordingly.

Req3: Display Borrowed Items and Overdue Warnings:

- **Justification:** Upon user login, the User Interface Component interacts with the BorrowingSystem to retrieve the list of borrowed items for the user. The BorrowingSystem also checks for any overdue items and generates warnings, which are displayed to the user through the user interface.

Req4: Newsletter Subscription and Cancellation:

- **Justification:** The UserManager handles newsletter subscription and cancellation. Users can subscribe to newsletters through the user interface, and UserManager updates the database accordingly. Similarly, users can cancel subscriptions, and UserManager updates the database to reflect the changes.

Req5: Book Search and Recommendations:

- **Justification:** The BookRecommendationSystem class provides book search and recommendation functionalities. When a user searches for a book, the BookRecommendationSystem queries the database for relevant titles and displays them to the user. Additionally, it uses text similarity algorithms to recommend similar books based on the searched title.

Req6: Course Textbook Tracking and Notifications:

- **Justification:** The CourseTextbookManager class manages course textbook tracking and notifications. For faculty users, it keeps track of courses taught and textbooks used. When a new edition of a textbook becomes available, CourseTextbookManager notifies the user about it, facilitating textbook procurement if needed.

Req7: Item Management and Navigation:

- **Justification:** The Item class represents individual library items, and the ConcreteLibraryItemFactory creates instances of specific item types. Managers can add, enable, or disable items through the Business Logic Component, updating the database accordingly. Users can navigate items through the user interface, which interacts with the Item class to display item details and availability.

Req8: Virtual Textbook Copies for Students:

- **Justification:** The Item class and UserManager collaborate to provide virtual textbook copies for students. When a student registers for a course, the UserManager associates the required textbooks with the student's account. The User Interface Component displays virtual copies of textbooks to the student for the duration of the course.

Req9: Book Request Prioritization:

- **Justification:** The ItemPurchaseManager class handles book requests and prioritization. Depending on the type of request (e.g., textbook for course teaching or self-improvement), ItemPurchaseManager prioritizes requests accordingly. Notifications about priority are sent to users, and requests are managed based on priority levels.

Req10: Discounted Purchases with Payment Options:

- **Justification:** The ItemPurchaseManager facilitates discounted purchases with special agreements with publishers. Users can choose items for purchase through the user interface, and ItemPurchaseManager handles payment processing using various payment options such as debit, credit, or mobile wallet.

Req11: Database Storage using SQLite:

- **Justification:** The DatabaseManager class manages data storage using SQLite. It reads data from these files during system initialization and writes data back to them when updates occur.