CITS 4401/3301

Take home test #2

03 May 2023

Student Name: Pritam SUWAL SHRESTHA

Student Number: 23771397

**Instructions:**

1. To be submitted as a single PDF or docx file by 04 May 2023, 4pm.
2. Please note this is an individual test. You can consult your notes but do not consult other students.
3. Provide your answers just below the questions in this file.
4. You can take as much space as required for your answers.
5. Do not change the order of your answers.
6. Use a **black fon**t for providing the answers.
7. You can use other colours for your diagrams if you wish to, but please make sure that the diagrams are readable.
8. There are **ten points** for following these instructions.
9. If I am not able to open your file, then you will get zero points.
10. Make sure to **submit** the file.

**Case study**

In a small town in the “Utopian land”, there is a Townhall. The Townhall provides multiple services to the residents. They would like to automate some of the services including:

1) *management of the library*, 2) *a small cafeteria* (online orders can be made through an app) that they have inside the town hall, and 3) the *marriage register*.

Each new user in the system will be given a username and a password.

The town hall consists of 10 employees. The library consists of about 3000 books.

For getting a book, users can self-scan a book. Upon returning the book, the librarian will scan the book to update the system.

For the cafeteria, a small inventory of current items will be managed, which will be visible to users on the app. Once someone buys an item (either online or in-person), the cashier will scan the item, which will help update the inventory records.

***Please state your assumptions while attempting the questions***

**Questions (4 questions)**

1. Design and draw a UML class diagram for the Townhall system. Include the main classes and their relationships for modelling the system. Maximum 6 classes. Only display the classes, associations and their multiplicities. [10 marks]

**Assumptions**:

* The library will have a single librarian to manage the books.
* The cafeteria will have a single cashier to manage the inventory and transactions.
* A user can register the marriage only once.

**Constraints**:

* A user can borrow a book only if it available (Constraint)
* A user can borrow item form cafeteria only if there is item (constraint)

The UML class diagram includes the following classes:

* **User**: Represents a user in the system, who can access the library, cafeteria and marriage register services by providing their unique username and password.
* **Employee**: Employee can be librarian or cashier
* **Librarian**: Represents the librarian in the library service, who can scan books to update the system.
* **Cashier**: Represents the cashier in the cafeteria service, who can scan items to update the inventory records.
* **Resident**: A townsfolk who is going to use the Townhall system
* **Book**: Represents a book in the library, which can have a unique book ID, title, author, and status (e.g., available, borrowed, etc.).
* **BookRecord**: A record of book borrowed by a user which is updated either when user borrow the book or librarian scan the book for return.
* **Item**: Represents an item in the cafeteria item, which can have a unique item name and quantity.
* **OrderRecord**: A record of order by a user which is updated either when user order the item from cafeteria or when the cashier scan the item for sale.
* **MarriageRegister**: Represents the marriage register service provided by the town hall.

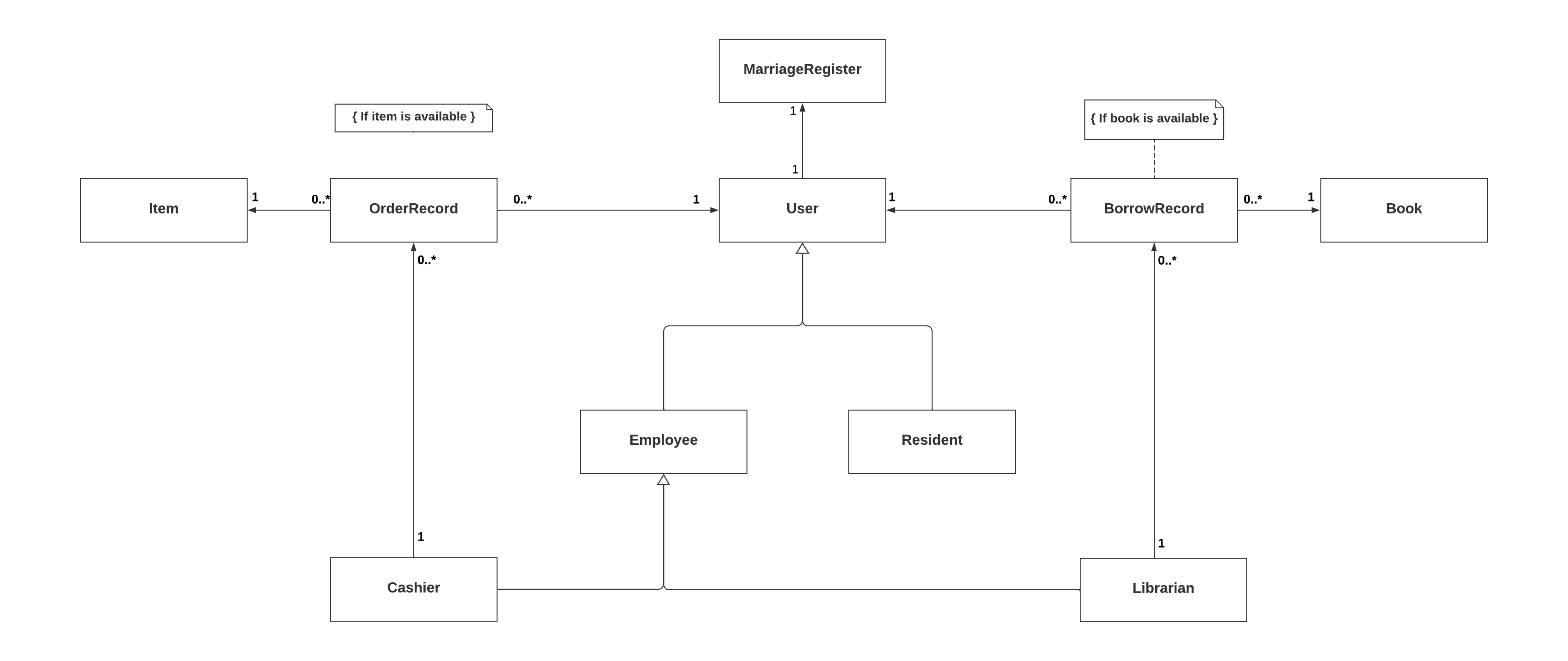


Fig: UML class diagram for Townhall system

The diagram includes the following multiplicities:

* A user can borrow one or more books, with each borrowing recorded in a BorrowRecord.
* Each user can borrow only one book at a time.
* The same book can be borrowed by multiple users and can exist even if no one has borrowed it.
* A librarian can update zero or more BookRecord objects.
* Each user can register for marriage only once, so a MarriageRegister can belong to a single user.
* A user can order one or more items.
* The same item can be ordered by zero or more users.
* Each order is recorded in an OrderRecord, and the same item can appear in one or more OrderRecords.
* A cashier can update zero or more OrderRecords after scanning the items.

Although the diagram above captures most of the details, the question requires identifying only **six main classes**, so I will select the following classes.

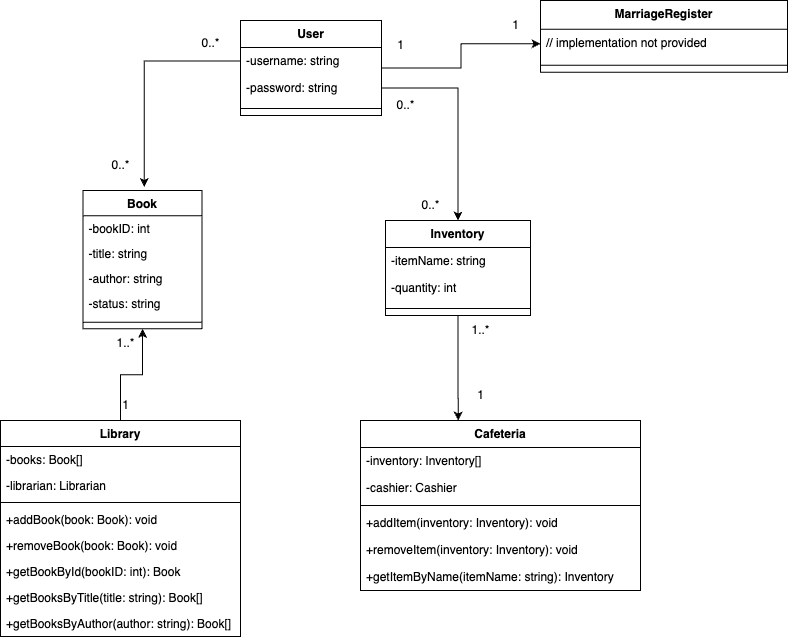


Fig: UML Class diagram for the Townhall system (6 main classes)

When trying to abstract the classes furthermore down, we get a UML class diagram something like one right above. In that scenario, Library and Cafeteria are represented as class as well.

2. Draw a UML sequence diagram for the online process of ordering a sandwich. [10 marks]

**Assumption**:

1. The sequence diagram assumes that user authentication is successful and does not include steps for handling incorrect user credentials to keep the diagram simple and readable.

2. The sequence diagram assumes that payment is always successful and does not include steps for handling payment failures.

3. The sequence diagram assumes that the order process is completed successfully without any errors or exceptions that would cause the order to be cancelled midway.

A picture containing screenshot, diagram, design

Description automatically generatedFig: Sequence diagram for the online process of ordering a sandwich

The above sequence diagram shows the interaction between the user, user authentication system, Online ordering system (cafeteria), and payment gateway during an online sandwich ordering process.

The sequence starts with:

1. A user entering their login credentials and the user authentication system verifying the user's credentials.
2. Upon successful authentication, the user selects the sandwich type and toppings in the online ordering system.
3. Then, the ordering system creates an order based on the user's choices and initiates the payment process through the payment gateway.
4. The payment gateway verifies the payment details and confirms the payment to the ordering system.
5. Finally, the sandwich ordering system sends a confirmation message and order details to the user, completing the sequence.

3. A first step of system design is to clarify the requirements, and then break down the solution domain into subsystems. A subsystem is a collection of classes, associations, operations, events and constraints that are interrelated. [10 marks]

(1) Identify a list of subsystems for the Townhall system. Keep your subsystem decomposition simple and abstract at this stage. Make a list describing a maximum of 5 subsystems.

The potential subsystems for the Townhall system are as follows:

1. **User Management Subsystem**: This subsystem would handle user authentication and management, including creating and deleting user accounts, managing usernames and passwords, and enforcing access control policies.
2. **Library Management Subsystem**: This subsystem will be responsible for managing the library, including adding and removing books, updating book information, and keeping track of book loans and returns.
3. **Cafeteria Management Subsystem**: This subsystem would handle the management of the small cafeteria, including managing the inventory of available items, processing customer orders (both online and in-person), and updating inventory records based on customer purchases.
4. **Marriage Register Subsystem**: This subsystem will be responsible for managing the marriage register, including creating new marriage records, updating existing records, and deleting records including maintaining accurate and up-to-date records of marriages and processing requests for marriage certificates.
5. **Barcode/QR Scanning App**: This subsystem could be responsible for providing the functionality to scan books and cafeteria items. This could include developing a user interface for the app, integrating with the library and cafeteria subsystems to access information about available books and items, and communicating with the scanner hardware to read the barcodes or QR codes on the items.

A picture containing text, screenshot, font, white

Description automatically generated

Fig: Potential Subsystems for the Townhall system

(2) What is the coupling between your subsystems? Is the level of coupling and cohesion in your system high or low? Explain your answer.

The Townhall system's subsystems have varying levels of cohesion and coupling. Cohesion refers to how closely related elements within a subsystem work together towards a common goal, while coupling refers to the interdependence between different subsystems.

1. The User Management Subsystem has high cohesion and low coupling, as all its elements work together to manage user accounts and access control, without depending on other subsystems.

2. The Library Management Subsystem has high cohesion and tight coupling with the User Management and Barcode Scanning App subsystems, as they provide critical functionality for authentication, access control, and updating borrow and return of books.

3. The Cafeteria Management Subsystem has high cohesion and tight coupling with the User Management and Barcode Scanning App subsystems, as they manage inventory and process orders.

4. The Marriage Register Subsystem has high cohesion and tight coupling with the User Management subsystem, as it fails to operate if User Management System fails.

5. The Barcode/QR Scanning App Subsystem has high cohesion and no coupling, as it provides scanning functionality without depending on other systems.

While high coupling can make a system less flexible and harder to maintain, in this case, it may be necessary for the subsystems to work together and share information to provide a cohesive system for managing the town hall's resources.

4. Select one of your chosen subsystems and identify 2 contrasting designs for implementing that subsystem. Use design rationale to document the best design decision considering these 2 options. Your answer should clearly identify: Issue, Proposals (your two design ideas), Criteria, Arguments and Resolution. Also mention any unresolved questions or assumptions you identify. [10 marks]

**Assumptions:**

Since, the townhall is trying to automate the services it provides to residents, I am assuming that the services are sought often regularly. This employs that the town has a large population.

**Sub-System**: Library Management Subsystem

**Issue**: How to implement the Library Management Subsystem such that it can provide the ability for users to browse and search the library catalogue and check out and return books.

**Proposals**:

1. **Proposal 1**: Design the Library Management Subsystem as a monolithic system. In this design, all the functionality for browsing the catalogue, checking out and returning books, and managing user accounts would be contained within a single system. This system would be responsible for all the logic for managing the library and would have direct access to the database of books, user accounts, and activity logs.
2. **Proposal 2**: Design the Library Management Subsystem as a microservices-based system. In this design, the functionality for browsing the catalogue, checking out and returning books, and managing user accounts would be divided into separate microservices. Each microservice would be responsible for a specific set of tasks and would communicate with other microservices through a messaging system. For example, one microservice might be responsible for managing the book catalogue, while another microservice might be responsible for checking out and returning books.

**Criteria**:

- Scalability: The system should be able to handle many users and books.

- Maintainability: The system should be easy to modify and update as new requirements arise.

- Reliability: The system should be able to handle errors and failures gracefully.

- Performance: The system should be able to respond quickly to user requests.

**Arguments**:

1. **Proposal 1**: Design the Library Management Subsystem as a monolithic system.

**Pros:**

* Simple and easier to build, since all the functionality is contained within a single system.
* Easier to test, since there are no dependencies between different components of the system.

**Cons:**

* Can become difficult to manage as the system grows in complexity.
* May suffer from performance issues, since all requests must go through a single system.

1. **Proposal 2**:

**Pros**:

* The microservices design has the advantage of modularity.
* Each microservice can be developed and maintained independently, which makes it easier to scale and modify the system.
* It is also more fault-tolerant, since failures in one microservice will not necessarily affect the rest of the system.

**Cons:**

* Can be more difficult to develop and test, since it requires more complex communication between different microservices.
* May suffer from performance issues, since there is overhead involved in communicating between different microservices.

**Resolution**:

Based on the criteria and arguments, I have decided to implement the Library Management Subsystem as a microservices-based system **(Proposal 2)**. This design offers greater flexibility and scalability, which will be important as the system grows in complexity and size. While there may be some additional overhead involved in communicating between microservices, we believe that the benefits of modularity and fault tolerance outweigh the potential performance costs.

**Unresolved Questions/Assumptions**:

* How will the messaging system be implemented?
* Will each microservice have its own database, or will they share a single database?