Ans1. **Main Actors:**

* User
* Author
* Administrator

**Main Objects:**

* Book

**Functionalities:**

* **User(extends Userbase):**

Attributes: user\_id, username, password, email, address

Functionalities: login(), logout(), register(), updateProfile()

* **Author (extends Userbase):**

Attributes: Inherits attributes from User class

Functionalities: addBook(Book), updateBook(Book), deleteBook(Book)

* **Administrator (extends Userbase):**

Attributes: Inherits attributes from User class

Functionalities: manageUsers(), manageBooks(), monitorSales()

* **Book:**

Attributes: book\_id, title, author, price, isbn, category

Functionalities: addBook(), updateBook(), deleteBook(), searchBook()

Ans3. **Cardinalities:**

User-Book Interaction (Purchase): Many-to-Many (Users can purchase multiple books, and books can be purchased by multiple users). Author-Book Relationship (Write): One-to-Many (An author writes multiple books, but each book has one author). Administrator Responsibilities (Manage): Many-to-Many (Administrators can manage multiple users and books).

Ans4. **Design Choices:**

Used class inheritance: User, Author, and Administrator inherit from a common UserBase class to capture shared attributes and methods. Introduced associative classes Purchase, Write, and Manage to represent complex relationships. Included relevant attributes and methods for each class to encapsulate their functionalities. Utilized aggregation (e.g., Manage contains a reference to User and Book) to represent the administrator's responsibilities.

Ans5. **Challenges and Trade-offs:**

Complexity of relationships: Representing the many-to-many relationships using associative classes might add complexity to the system. However, this approach allows for more flexibility and extensibility in the long run.

Scalability: As the system grows, managing the associations and ensuring data integrity can become challenging. Proper database design and indexing should be considered to address potential scalability issues.

Security: The diagram does not address security features like authentication and authorization explicitly. These aspects should be implemented in the system but are not represented in the class diagram for simplicity.

Concurrency and transactions: Handling concurrent access to book quantities during purchases and ensuring atomic transactions should be addressed in the system's implementation but are not shown in the diagram.