**Synopsis for Online Bookstore Management System**

**NAME = SOURABH THAPLIYAL**

**ANUDIP ID = AF0414599**

**BATCH CODE**

**ANP-C8363**

**BATCH COORDINATOR**

**ANUJ KUMAR**

**Online Bookstore Management System**

**2. Project Category:**

**Desktop Application (Command-Line Interface)**

**3. Requirement Analysis:**

**The Online Bookstore Management System aims to provide users with a platform to buy books, manage their profiles, place orders, and leave reviews. The system also provides functionalities for administrators to manage the bookstore inventory, user accounts, and order histories. The key requirements include:**

* **User Features: Registration, login, profile management, placing orders, adding reviews.**
* **Admin Features: Book and author management, user management, and order tracking.**
* **System Requirements:**
  + **Persistent storage of books, users, orders, and reviews using a relational database.**
  + **Command-line interface for interaction.**
  + **Secure user authentication and role-based access control for admins.**

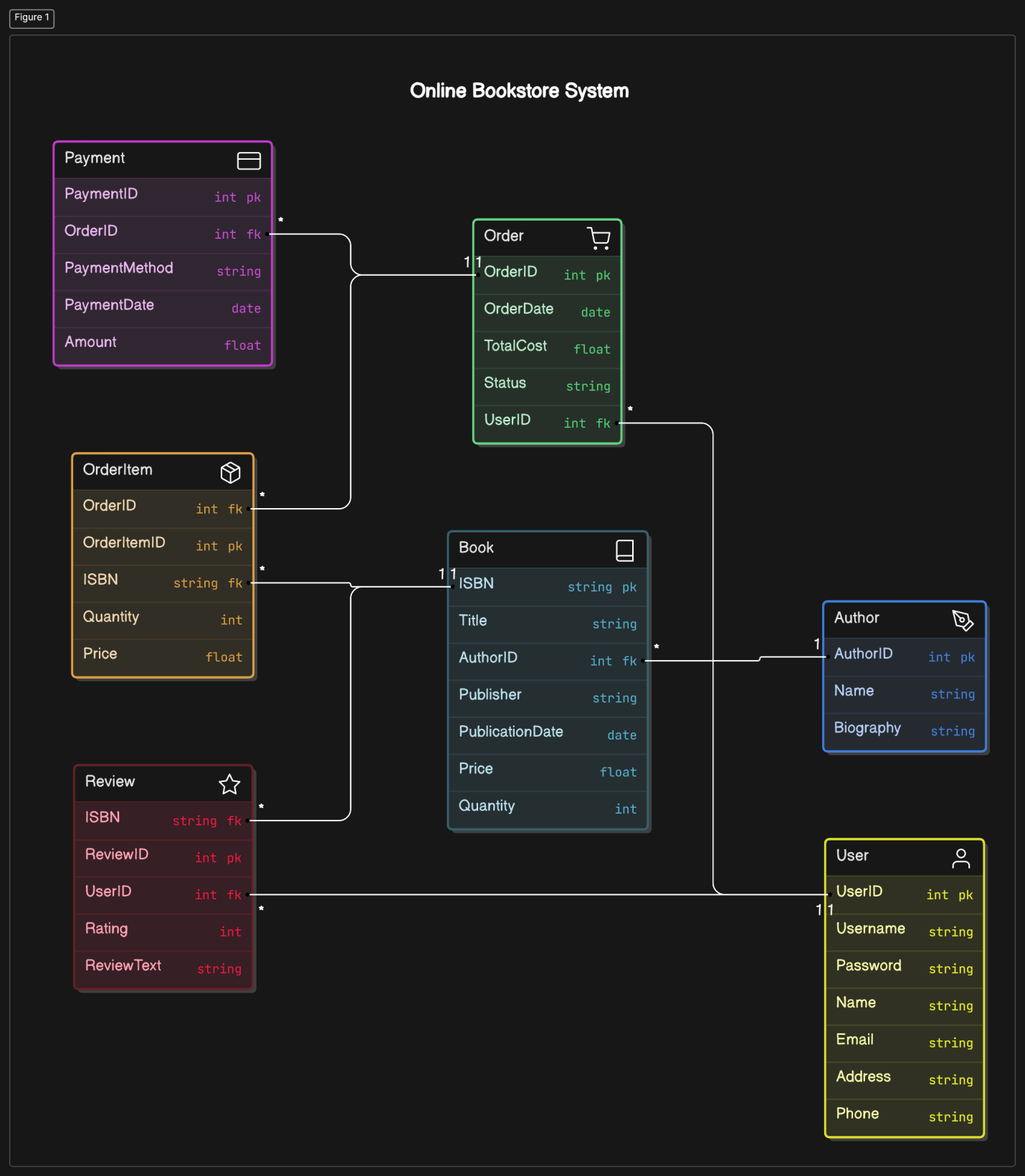
**4. Technology Used:**

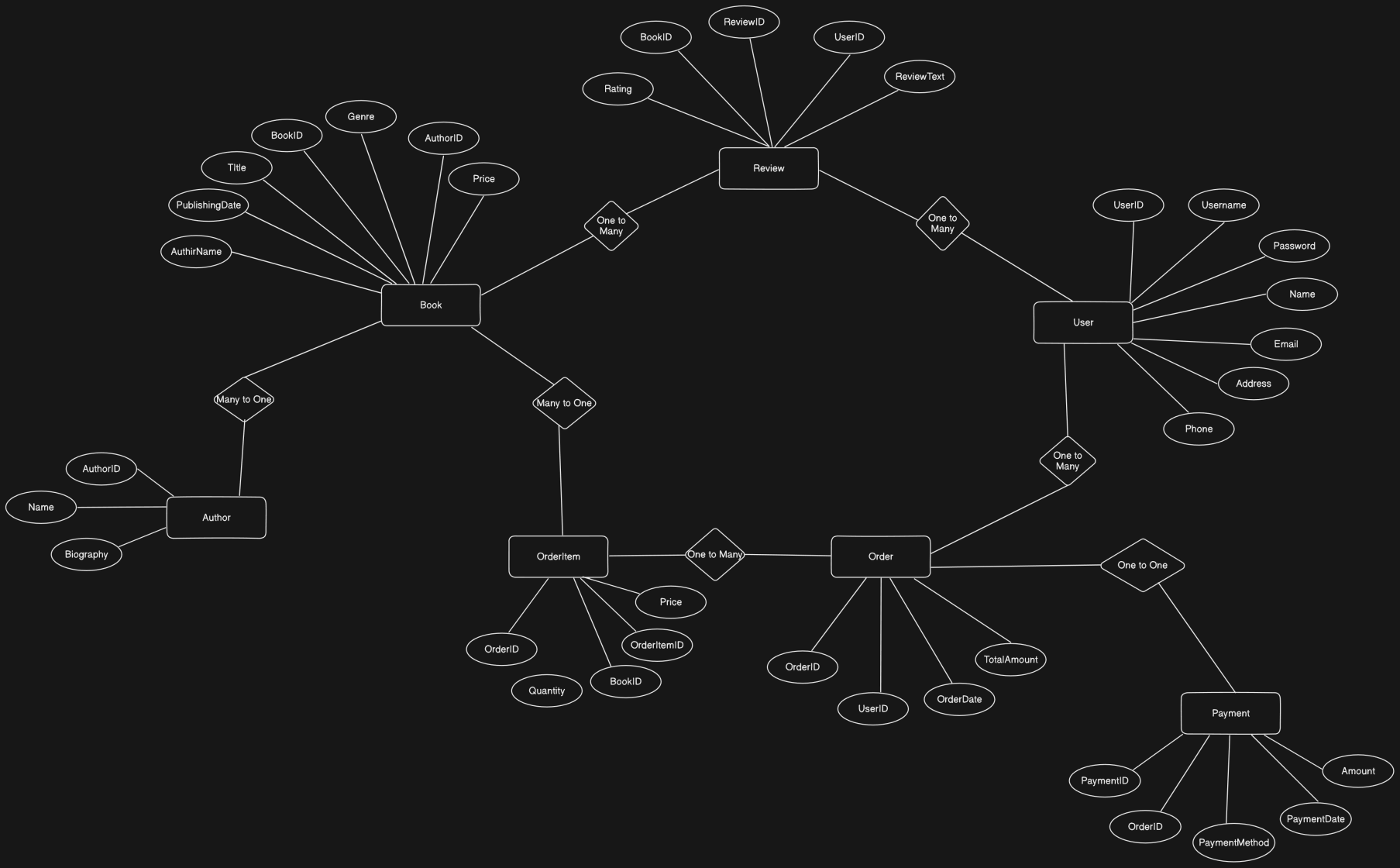
* **Programming Language: Java**
* **ORM Framework: Hibernate (for database interaction)**
* **Database: MySQL**
* **Development Environment: Eclipse**
* **Version Control: Github**

**5. Modules Description:**

* **User Management Module:**
  + **Functions: User registration, login, profile management, viewing order history.**
  + **Classes:**
    - **UserController: Handles user-related operations.**
    - **UserService and UserServiceImplementation: Business logic layer between the controller and DAO.**
    - **UserDAO and UserDAOImplementation: DAO layer responsible for database interactions for users.**
    - **User: Entity class representing user data.**
* **Admin Management Module:**
  + **Functions: Admin login, adding, updating, and deleting books, viewing user data, managing orders.**
  + **Classes:**
    - **AdminController: Handles admin-related operations.**
    - **BookController: Manages book-related functions.**
    - **AdminService, AdminServiceImplementation: Service layer for admin operations.**
    - **BookService, BookServiceImplementation: Service layer for book operations.**
* **Order Management Module:**
  + **Functions: Place orders, view order history, and admin management of orders.**
  + **Classes:**
    - **OrderController: Handles order-related operations.**
    - **OrderItemController: Manages individual order items.**
    - **OrderService, OrderServiceImplementation: Service layer for managing orders.**
    - **OrderDAO and OrderDAOImplementation: DAO layer responsible for database interactions for orders.**
    - **Order: Entity class representing order data.**
    - **OrderItem: Entity class representing individual items in an order.**
* **Book and Author Management Module:**
  + **Functions: Add, update, delete books, associate books with authors, and manage author information.**
  + **Classes:**
    - **BookController: Manages book-related functions and interactions.**
    - **AuthorController: Handles author-related functions.**
    - **BookService and BookServiceImplementation: Service layer for book operations.**
    - **AuthorService and AuthorServiceImplementation: Service layer for author operations.**
    - **BookDAO and BookDAOImplementation: DAO layer for managing book data.**
    - **AuthorDAO and AuthorDAOImplementation: DAO layer for managing author data.**
    - **Book: Entity class representing book data.**
    - **Author: Entity class representing author data.**
* **Review Management Module:**
  + **Functions: Add, update, and delete book reviews.**
  + **Classes:**
    - **ReviewController: Manages user reviews of books.**
    - **ReviewService, ReviewServiceImplementation: Service layer for review-related operations.**
    - **ReviewDAO and ReviewDAOImplementation: DAO layer for database interactions related to reviews.**
    - **Review: Entity class representing review data.**

**6. Database Design:**

****

**7. ER Diagram:**

**8. Testing:**

* **Unit Testing: JUnit was used to test individual components like controllers, services and DAOs. Mocking techniques were used to isolate database operations during testing.**
* **Integration Testing: The interaction between the user, order, and book management modules was tested.**
* **Manual Testing: Extensive manual testing was performed through the CLI, ensuring all functionalities worked as expected.**

**9. Future Scope:**

* **Enhanced CLI: Additional features like a search filter, book recommendations, or a wishlist system can be implemented.**
* **GUI Development: Transition to a graphical user interface (GUI) to improve usability.**
* **Performance Optimization: Use Hibernate caching mechanisms (e.g., second-level caching) to improve performance for frequently accessed data like books and authors.**
* **Mobile Application: Develop a mobile version of the system for better accessibility.**

**End of Synopsis**

**This synopsis reflects the core functionalities of my CLI-based Online Bookstore Management System with Hibernate integration.**