Mid Term Project

Online Book Shop Micro-Service Project

This is the micro-service version of the previous **Online Book Shop Application**. This version will have 6 different micro services.

Service 01: API Gateway

You need to follow API gateway architecture here. We will not directly communicate with the individual micro-services. All the communication will happen via the API gateway. In the API gateway users (**ADMIN** and **USER**) need to authenticate themselves before accessing the other micro-services.

Service 02: Discover Server

Along with API gateway, you need to use discovery server as a service. This discover service will be responsible for identifying the available service instances of our application.

Service 03: Cloud Config Server

The configuration properties of all the micro-services will be stored in the cloud (GitHub repo). We will fetch the configuration properties via this cloud config server.

Service 04: Authentication Service

This service will be responsible for authenticating the user and generate access token. Later this token will be used to access other micro services. **ADMIN** and **USER** both can use this service to authenticate them.selves This service will have the below APIs:

/auth-server/register: Both **USER** and **ADMIN** will use this API to register them.

/auth-server/login: Both **USER** and **ADMIN** will use this API for login. After login, this API will return them a JWT token. Later which will be used in the API gateway for accessing the micro-services.

Service 05: Book Service

ADMIN and **USER** both can use this service. This service will have the below APIs:

/book-service/create: Only **ADMIN** will use this API to insert new book information to the DB. While calling this API, **ADMIN** will pass *book name*, *author name*, *genre*, *price* and *quantity* in the request body. Among of these fields, *book name*, *author name* and *genre* will be saved to the DB by this service. To store *price* and *quantity*, you need to call **Service 06: Book Inventory Service**.

/book-service/update: Only **ADMIN** will use this API to update book information to the DB. This API will also call **Service 06: Book Inventory Service** to update *price* and *quantity* information.

/book-service/delete: Only **ADMIN** will use this API to delete book from the DB. This API will call **Service 06: Book Inventory Service** to delete *price* and *quantity* information for that particular book.

/book-service/book/all: Both **ADMIN** and **USER** can use this API to fetch all the book data from DB. This API will call **Service 06: Book Inventory Service** to fetch *price* and *quantity* information for the books.

/book-service/book/{id}: Both **ADMIN** and **USER** can use this API to fetch a single book data from DB. This API will also call **Service 06: Book Inventory Service** to fetch *price* and *quantity* information for that book.

/book-service/book/buy: Only **USER** can use this API to buy books from the online book store. In the request body **USER** need to pass *book id* and *quantity* for the desired book that he/she wants to purchase. After successful purchase, this API will call **Service 06: Book Inventory Service** to update the *quantity* information for that book.

Service 06: Book Inventory

This micro-service will not be directly called by users. We will call it from **Service 05: Book Service**. This micro-service will have the below APIs:

/book-inventory/update/book-id: This API will be used to **UPDATE** the *price* and *quantity* information of a particular book.

/book-inventory/book-id: This API will be used to **FETCH** the *price* and *quantity* information of a particular book.

/book-inventory: This API will take a list of book ids, and then **FETCH** the *price* and *quantity* information for those books.

/book-inventory/delete/book-id: This API will be used to **DELETE** the *price* and *quantity* information of a particular book.

Note: Analyze the requirements carefully. And identify the classes and database tables needed to design the application. Also try to follow the proper design principals (i.e, proper exception handling, DB best practices etc.)