**Assignment Overview:**

You will create three microservices:

1. Create Eureka Server using spring boot running on port number 8761
2. **User Service**: Manages user data stored in MySQL running on port number 9191
3. **Product Service**: Manages product data stored in MongoDB running on port number 9292
4. **Order Service**: Manages order data stored in H2 running on port number 9393
5. **API gate way** : Responsible to provide route all 3 micro service running on port number 8080

The microservices will communicate with each other using RESTful APIs. For instance, the Order Service will need to fetch user and product details from the User and Product Services respectively.

**User Service (MySQL):**

* **Database**: MySQL
* **Entity**: User with fields id, username, email, createdAt.
* **API Endpoints**:
  + GET /users/{id}: Get user details by ID.
  + POST /users: Create a new user.
  + GET /users: List all users.
* **Database Interaction**: Spring Data JPA

**Product Service (MongoDB):**

* **Database**: MongoDB
* **Entity**: Product with fields id, name, description, price, createdAt.
* **API Endpoints**:
  + GET /products/{id}: Get product details by ID.
  + POST /products: Create a new product.
  + GET /products: List all products.
* **Database Interaction**: Spring Data MongoDB

**Order Service (H2):**

* **Database**: H2 (in-memory)
* **Entity**: Order with fields id, userId, productId, quantity, orderDate.
* **API Endpoints**:
  + GET /orders/{id}: Get order details by ID.
  + POST /orders: Create a new order. Oid auto increment or you can pass manually, username, product name, orderdate(system date).
  + GET /orders: List all orders.
* **Database Interaction**: Spring Data JPA

**Microservice Communication:**

1. **Order Service → User Service**:
   * When creating or placing an order, the Order Service will call the User Service to validate the user by fetching user details. Passing user id get the user name. You need to use RestTemplate with load balance features to get user name. if we pass wrong user id then don’t place the order return the message as user not exits.
2. **Order Service → Product Service**:
   * Similarly, the Order Service will call the Product Service to fetch product details before placing an order. Passing product id to get product details. This service need to use Feign Client concept to call Product Service . if product not present don’t place the order.

**Set Up Each Microservice:**

* Use Spring Initializr to create three separate Spring Boot projects.
* Add necessary dependencies:
  + **User Service**: Spring Web, Spring Data JPA, MySQL Driver.
  + **Product Service**: Spring Web, Spring Data MongoDB.
  + **Order Service**: Spring Web, Spring Data JPA, H2 Database.

**Define Entities and Repositories:**

* **User Service**: Create User entity and UserRepository.
* **Product Service**: Create Product entity and ProductRepository.
* **Order Service**: Create Order entity and OrderRepository.

**Implement Service and Controller Layers:**

* Implement CRUD operations in each service.
* Expose REST endpoints through controllers.

**Set Up Inter-Service Communication:**

* Use RestTemplate and Feign client in the Order Service to call User and Product services.

**Configure Databases:**

* **User Service**: Configure MySQL in application.properties.
* **Product Service**: Configure MongoDB in application.properties.
* **Order Service**: Configure H2 in application.propertie