**Tailoring Management System: End-to-End Approach and Documentation**

**Project Overview**

Small tailoring businesses face challenges with manual order handling, custom measurement tracking, and timely delivery. This project aims to streamline these operations using a **Spring Boot, Angular, and MySQL-based microservices architecture** integrated with a **public API** for enhanced functionality.

**Development Tools and Technologies**

1. **Backend**: Spring Boot (Java)
2. **Frontend**: Angular
3. **Database**: MySQL
4. **Microservices Communication**: Spring Cloud OpenFeign, RestTemplate
5. **Message Queue**: RabbitMQ or Kafka (for asynchronous events)
6. **Security**: JWT (JSON Web Tokens)
7. **Public API**: Example – a delivery service API or a government MSME API

**Step-by-Step Approach**

**1. Planning and Requirement Analysis**

* Define user roles:
  + **Customer**: Places orders, provides measurements, tracks progress.
  + **Tailor**: Manages tasks, updates order status.
* Identify key features:
  + Online order placement.
  + Measurement tracking.
  + Task assignment for tailors.
  + Notifications for order status updates.
* **Microservices**: Separate each feature into distinct services for scalability.

**2. Microservices Design**

**Microservice 1: User Management Service**

**Responsibilities**:

* User registration, login, and role-based access control.
* JWT-based authentication.

**Endpoints**:

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /register | Register a new user |
| POST | /login | Authenticate user |
| GET | /users/{id} | Get user details |

**Database Schema**: user\_management\_db

CREATE TABLE users (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255),

email VARCHAR(255) UNIQUE,

password VARCHAR(255),

role ENUM('CUSTOMER', 'TAILOR'),

phone\_number VARCHAR(15),

created\_at DATETIME

);

**Microservice 2: Order Management Service**

**Responsibilities**:

* Order creation, tracking, and status updates.

**Endpoints**:

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /orders | Place a new order |
| GET | /orders/{id} | Get order details |
| PUT | /orders/{id}/status | Update order status |

**Database Schema**: order\_management\_db

CREATE TABLE orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT,

tailor\_id INT,

order\_date DATE,

status ENUM('Pending', 'In Progress', 'Completed'),

delivery\_date DATE,

invoice\_id INT

FOREIGN KEY (customer\_id) REFERENCES users(user\_id),

FOREIGN KEY (tailor\_id) REFERENCES users(user\_id)

);

**Microservice 3: Measurement Service**

**Responsibilities**:

* Store and manage custom measurements.

**Endpoints**:

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /measurements | Add new measurements |
| GET | /measurements/{id} | Get measurements for an order |

**Database Schema**: measurement\_db

CREATE TABLE measurements (

measurement\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT,

gender ENUM(‘Male’, ‘Female’),

category varchar,

design varchar,

measurements varchar

order\_id Long

FOREIGN KEY (customer\_id) REFERENCES users(user\_id)

);

**Microservice 4: Tailor Workflow Service**

**Responsibilities**:

* Assign and manage tasks for tailors.

**Endpoints**:

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /tasks | Assign a new task |
| GET | /tasks/{tailorId} | Get tasks for a tailor |

**Database Schema**: workflow\_db

CREATE TABLE tasks (

task\_id INT AUTO\_INCREMENT PRIMARY KEY,

tailor\_id INT,

order\_id INT,

due\_date DATE,

status ENUM('Pending', 'Completed'),

FOREIGN KEY (tailor\_id) REFERENCES users(user\_id),

FOREIGN KEY (order\_id) REFERENCES orders(order\_id)

);

**Microservice 5: Notification Service**

**Responsibilities**:

* Notify users of order status updates.

**Endpoints**:

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /notifications | Send a notification |

**Database Schema**: notification\_db

CREATE TABLE notifications (

notification\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

message VARCHAR(255),

status ENUM('Sent', 'Pending'),

created\_at DATETIME,

FOREIGN KEY (user\_id) REFERENCES users(user\_id)

);

**3. Public API Integration**

Integrate a government or delivery service API for tracking or MSME support.

* Example API: Delivery Tracking API.

**4. Asynchronous Communication**

* Use **RabbitMQ** or **Kafka** for communication between Order Management and Notification services.

**5. Security**

* Use JWT tokens for securing APIs. Implement a Gateway service to handle authentication and route requests.

**6. Implementation Steps**

1. **Set Up Spring Boot Projects** for each microservice.
2. **Create Angular Frontend** with components for order creation, tracking, and measurement entry.
3. **Implement REST Endpoints** in each service.
4. **Configure MySQL Databases** for each service.
5. **Implement Feign Clients or RestTemplate** for communication.
6. **Set Up Message Queue** for status updates.
7. **Secure APIs** with JWT.
8. **Integrate Public API**.
9. **Test Each Service** independently and as a whole.
10. **Deploy Services** using Docker or Kubernetes for scalability.