**Tailor Management System Documentation**

**1. Project Overview**

**Project Title**: Tailor Management System

**Description**: A microservices-based application designed to manage tailor business operations, including user management, order tracking, measurement storage, workflow progression, and email notifications.

**Technologies Used**:

* **Frontend**: Angular
* **Backend**: Spring Boot (Microservices)
* **Database**: MySQL

**2. System Architecture**

**High-Level Architecture Diagram**:

+---------------------+ +---------------------------+

| Angular Frontend | <-----> | API Gateway (Optional) |

+---------------------+ +---------------------------+

|

+-----------------------------------+-----------------------------------+

| | | | | |

+----------------+ +----------------+ +----------------+ +---------------+ +--------------+

| User Service | | Order Service | | Measurement | | Workflow | | Email Service|

+----------------+ +----------------+ | Service | | Service | +--------------+

+----------------+ +---------------+

\ \ /

\ \ /

-----------------------------------------------

MySQL Databases

**Microservices**:

| **Service Name** | **Responsibilities** |
| --- | --- |
| **User Management** | Handles users (tailors, customers, admins), user authentication, and profile management. |
| **Order Management** | Manages tailoring orders (creation, updating, viewing order status). |
| **Measurement** | Stores and manages customer measurements linked to specific orders. |
| **Workflow Management** | Tracks the status of tailoring jobs (pending, stitching, completed). |
| **Email Service** | Sends notifications and updates to customers and tailors. |

**3. Database Design**

**ER Diagram**:

[User] (1) -------- (0..\*) [Order]

[Order] (1) ------- (1) [Measurement]

[Order] (1) ------- (0..\*) [Workflow]

[Notification] ----- (Email Microservice)

**Entity Descriptions**:

| **Entity** | **Attributes** |
| --- | --- |
| **User** | user\_id (PK), name, email, password, role (tailor/admin/customer) |
| **Order** | order\_id (PK), user\_id (FK), status, creation\_date, due\_date |
| **Measurement** | measurement\_id (PK), order\_id (FK), size, length, other details |
| **Workflow** | workflow\_id (PK), order\_id (FK), stage, description |
| **Notification** | notification\_id (PK), email, subject, message, sent\_date |

**4. API Documentation**

| **Service** | **Endpoint** | **Method** | **Description** |
| --- | --- | --- | --- |
| User Management | /users | GET | Fetch all users |
|  | /users/{id} | GET | Fetch a user by ID |
|  | /users | POST | Create a new user |
| Order Management | /orders | GET | Fetch all orders |
|  | /orders/{id} | GET | Fetch an order by ID |
|  | /orders | POST | Create a new order |
| Measurement | /measurements | POST | Add a measurement |
| Workflow Management | /workflows | POST | Add a workflow step |
| Email Service | /emails/send | POST | Send email notification |

**5. Flow Diagram**

1. User logs into the Angular frontend.

2. Frontend calls respective microservice endpoints.

3. User management handles login and registration.

4. Order service creates new tailoring orders.

5. Measurement service stores customer measurement details.

6. Workflow service updates tailoring progress.

7. Email service sends status updates to customers.

**6. Frontend Modules**

* **User Module**: Manages registration and authentication.
* **Order Module**: Displays orders and allows CRUD operations.
* **Measurement Module**: Interfaces for measurement input.
* **Workflow Module**: Tracks order status.

**Angular Component Structure**:

| **Component** | **Description** |
| --- | --- |
| user-login | User login form. |
| order-list | Displays list of orders. |
| order-details | Shows individual order details. |
| measurement-form | Form for adding/editing measurements. |
| workflow-status | Displays workflow progress of orders. |

**7. Advanced Features**

* **JWT Authentication**: Secure API communication.
* **Centralized Configuration** (Optional): Manage microservices properties from a single point.
* **Circuit Breaker (Resilience4j)**: Handle service failures gracefully.

**8. Deployment Instructions**

1. **Run MySQL Database**:
   * Use the provided schema to create tables.
2. **Spring Boot Microservices**:
   * Start each service independently.
3. **Frontend (Angular)**:
   * ng build --prod for production build.
   * ng serve for local testing.

This documentation can be expanded further with UML diagrams, screenshots, and additional configurations as required. Let me know if more sections are needed!