🎨 ArtistHub - Artist Booking Application

1. Project Overview

ArtistHub is a comprehensive web application built with Spring Boot for connecting artists with customers. It provides a platform where artists can showcase their profiles, upload media, receive bookings, and manage their schedules. Customers can search for artists, make bookings, and leave reviews.

Core Interaction Flow

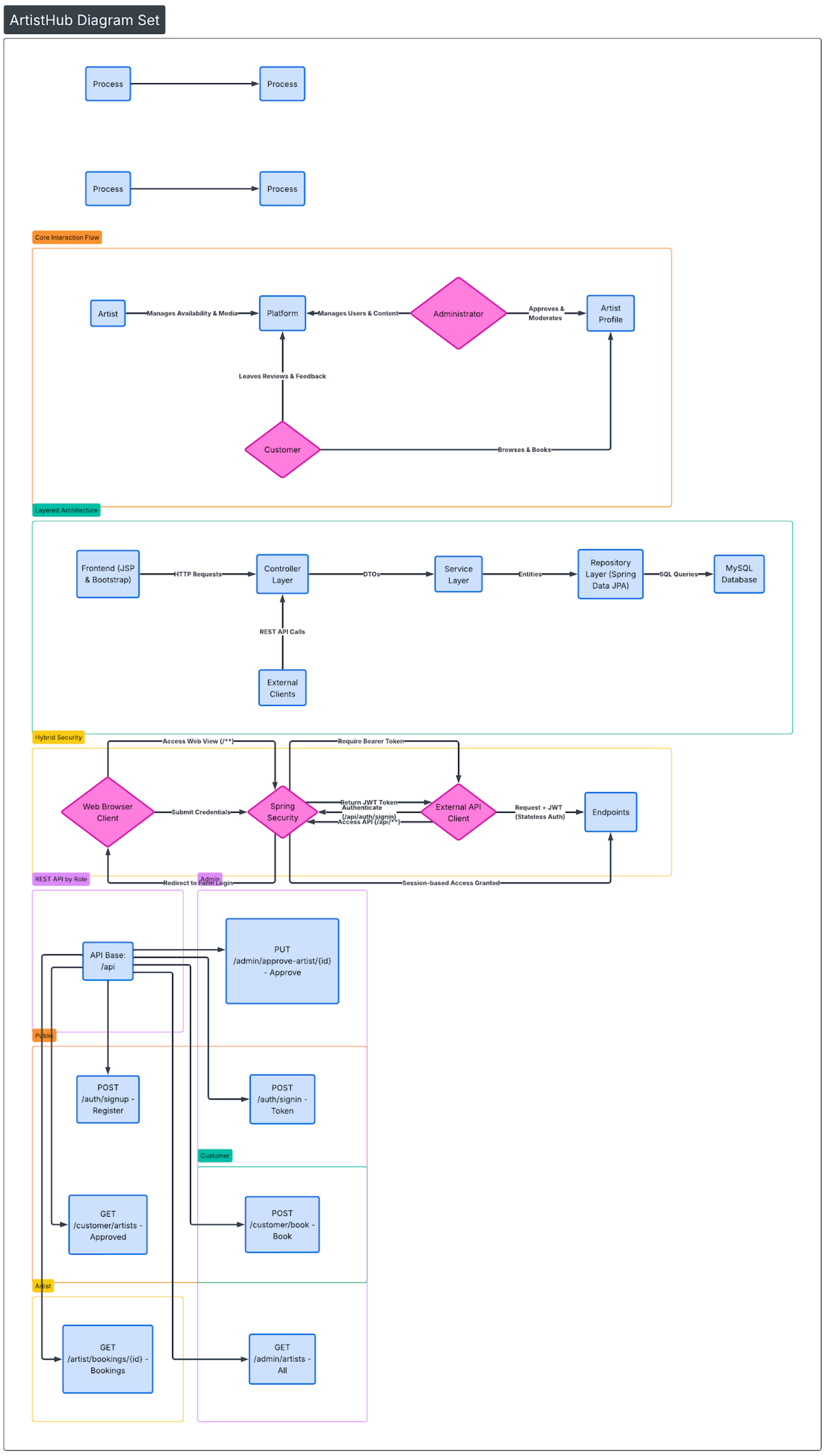
Code snippet

```
graph LR
    C[Customer] -->|Browses & Books| A(Artist Profile)
    C -->|Leaves Reviews & Feedback| P{Platform}
    A -->|Manages Availability & Media| P
    AD[Administrator] -->|Approves & Moderates| A
    AD -->|Manages Users & Content| P
```

---

2. System Architecture & Design

The application follows a standard layered architecture to separate business logic, request handling, and data management.



Technology Stack

- **Backend:** Java 17, Spring Boot 3.2.2
- **Database:** MySQL 8.0
- **ORM:** Hibernate / Spring Data JPA

- **Security:** Spring Security 6
- **Frontend:** JSP (JavaServer Pages), Bootstrap 5
- **Build Tool:** Maven
- **Packaging:** WAR (Web Application Archive)

---

3. Security Implementation

ArtistHub uses a **Hybrid Security Approach** to accommodate both the web interface and external API consumers.

Code snippet

```
sequenceDiagram
  participant Client
  participant Spring Security
  participant Endpoints

  Note over Client,Endpoints: Web Browser Access
  Client->>Spring Security: Access Web View (/**)
  Spring Security->>Client: Redirect to Form Login
  Client->>Spring Security: Submit Credentials
  Spring Security->>Endpoints: Session-based Access Granted

  Note over Client,Endpoints: External API Access
  Client->>Spring Security: Access API (/api/**)
  Spring Security->>Client: Require Bearer Token
  Client->>Spring Security: Authenticate (/api/auth/signin)
  Spring Security->>Client: Return JWT Token
  Client->>Endpoints: Request + JWT (Stateless Auth)
```

---

4. User Roles & Credentials

Role	**Default Access**	**Key Capabilities**	**Dashboard URL**
Admin	admin@artisthub.com / admin123	Full control over users, approvals, and platform content.	/admin/dashboard
Artist	Registered via /register	Update profile, upload media, manage availability, view bookings.	/artist/dashboard

Customer	Registered via /register	Browse artists, book events, view history, leave reviews.	/customer/dashboard

5. Installation & Setup Guide

Prerequisites

- **JDK 17** or higher
- **Maven 3.6+**
- **MySQL Server** running on port 3306

Database Configuration

Run the following SQL to create the database:

SQL
```
CREATE DATABASE artisthub;
```

Ensure your src/main/resources/application.properties matches your local setup:

Properties
```
spring.datasource.url=jdbc:mysql://localhost:3306/artisthub?createDatabaseIfNotExist=true&useSSL=false&allowPublicKeyRetrieval=true
spring.datasource.username=root
spring.datasource.password=root
```

Running the Application (CLI)

1. **Navigate:** cd c:\Users\HP\Desktop\springbootproject
2. **Build:** mvn clean package
3. **Run:** mvn spring-boot:run OR java -jar target/ArtistHub-0.0.1-SNAPSHOT.war
4. **Access:** Open http://localhost:8080/

Eclipse IDE Setup

1. **Install Lombok:** Download lombok.jar, run it, install to Eclipse, and restart. *(Required for getters/setters)*.
2. **Import:** File > Import > Maven > Existing Maven Projects > Browse to project folder.
3. **Update:** Right-click project > Maven > Update Project... > Check "Force Update".
4. **Run:** Right-click ArtistHubApplication.java > Run As > Spring Boot App.

6. REST API Documentation

- **Base URL:** http://localhost:8080/api
- **Authentication:** POST /api/auth/signin returns a Bearer Token.

Method	**Endpoint**	**Description**	**Role Required**
POST	/api/auth/signup	Register new user	Public
GET	/api/customer/artists	Get approved artists	Public
POST	/api/customer/book	Book an artist	Customer
GET	/api/artist/bookings/{id}	Get artist bookings	Artist
GET	/api/admin/artists	Get all artists	Admin
PUT	/api/admin/approve-artist/{id}	Approve artist	Admin

---

7. Troubleshooting Common Issues

- **ERR_TOO_MANY_REDIRECTS**: Spring Security interception loop. Ensure `DispatcherType.FORWARD` and `DispatcherType.ERROR` are permitted in `SecurityConfig.java`.
- **404 Not Found on Pages**: Ensure `pom.xml` uses `<packaging>war</packaging>` and includes the `tomcat-embed-jasper` dependency.
- **500 Internal Server Error (sec:authorize)**: Missing `spring-security-taglibs` library in your `pom.xml`.