# Artistry Hub Project Documentation

## I. Project Requirements

### Purpose and Scope

Artistry Hub is envisioned as a collaborative and interactive platform specifically designed for artists and musicians to digitally manage, share, and showcase their work. The application allows users to:

- Register for an account and manage their profile.
- Create personalized virtual rooms for displaying their artwork or musical pieces.
- Upload and organize their files and folders in a structured manner.

### Expected Outcomes

The successful implementation of Artistry Hub will result in:

- A centralized and user-friendly platform for artistic content management.
- Increased visibility for artists and musicians through digital exposure.
- An organized and accessible repository of creative works.

### Constraints and Limitations

- The platform is optimized for modern web browsers and may not offer full functionality on older versions.
- File upload size and format may be limited by server configurations.
- Real-time collaboration features are not supported in the current version.

## II. Project Plan

The project plan section should detail the project's lifecycle, from conception to deployment, including milestones, resource allocation, risk assessment, and maintenance strategies.

## III. Source Code

### Repository Access

The complete source code for Artistry Hub can be accessed and cloned from GitHub:

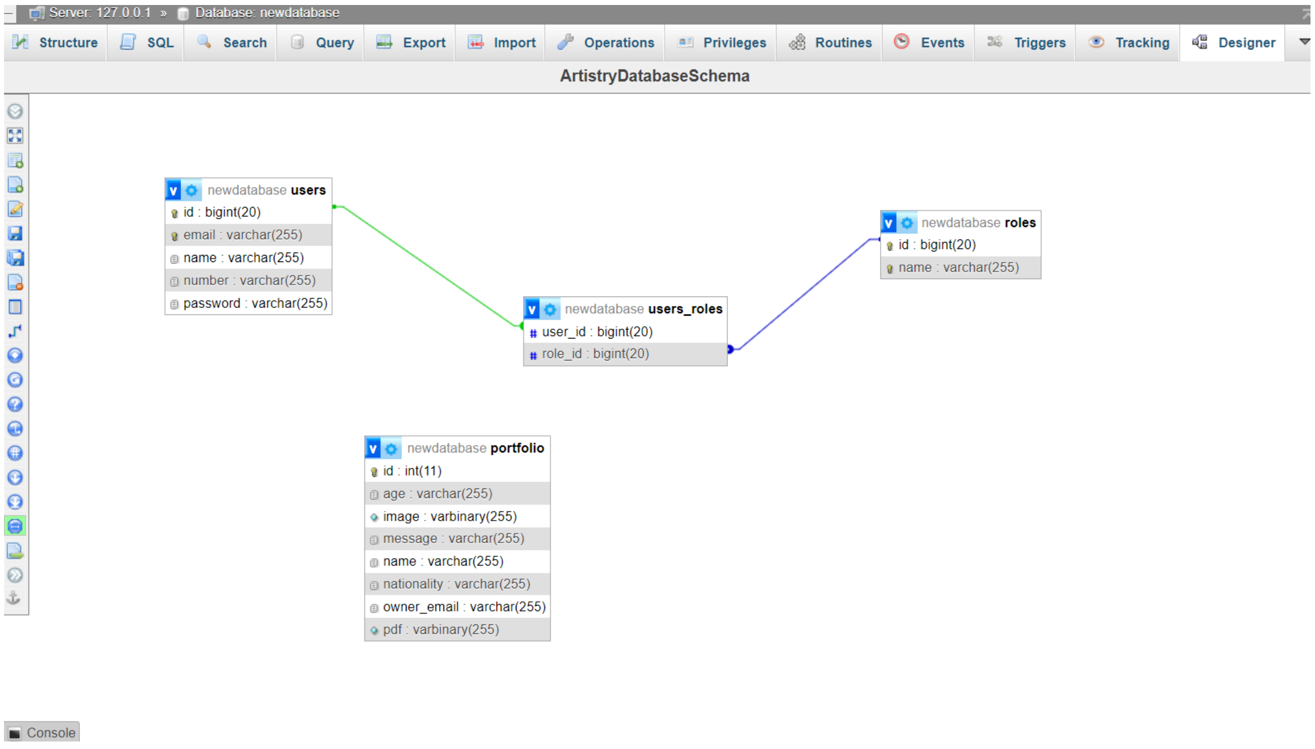
**git clone https://github.com/bienheureuseuwineza/ArtistryHub.git**

### Setup and Configuration

To set up the development environment, follow these instructions:
- Install Java Development Kit (JDK) and Maven.
- Navigate to the project directory and run `mvn install` to install dependencies.
- Execute `mvn spring-boot:run` to start the application.
-Use any IDE like Intellij or VS Code

Include any additional environment setup instructions, scripts, or configuration details here.

## IV. Database Schema



newdatabase **users**
- id : bigint(20)
- email : varchar(255)
- name : varchar(255)
- number : varchar(255)
- password : varchar(255)

newdatabase **users_roles**
- user_id : bigint(20)
- role_id : bigint(20)

newdatabase **roles**
- id : bigint(20)
- name : varchar(255)

newdatabase **portfolio**
- id : int(11)
- age : varchar(255)
- image : varbinary(255)
- message : varchar(255)
- name : varchar(255)
- nationality : varchar(255)
- owner_email : varchar(255)
- pdf : varbinary(255)

## V. User Documentation

### Getting Started

- **Registration:** Visit the sign-up page and fill in the required details to create an account.
- **Personalization:** Customize your room settings to reflect your artistic style.
- **File Management:** Upload your artwork or music files using the 'Add New Item' button.

## Navigation Guide for Artistry Hub

Welcome to Artistry Hub, your personal digital space for artistic expression and management. This guide will walk you through the main features of our platform and help you get the most out of your Artistry Hub experience.

### Step 1: Registration and Login

**1. Homepage**: Visit the Artistry Hub homepage.
   - Click on the **Sign Up** button in the top right corner to create a new account.
   - Fill in your details on the registration form and submit.

**2. Confirmation:** You will receive a confirmation email. Click on the verification link to activate your account.

3**. Login:** Return to the homepage and click on the **Login** button.
   - Enter your username and password to access your Artistry Hub dashboard.

**Step 2: Dashboard Overview**

**1. Dashboard:** Once logged in, you will be directed to your personal dashboard.
- Here, you can view a summary of your rooms and recent activity.

**2. Navigation Menu:** Use the navigation bar to access different parts of the application.
- The menu includes links to **Rooms, Uploads, Settings, and Help.**

**Step 3: Creating and Customizing Rooms**

**1. Create Room:** Click on the **Create Room** button.
- Choose a name for your room and select a theme.

**2. Customize Room:** Access the customization settings to personalize your room.
- Upload cover images, adjust layouts, and choose color schemes.

**Step 4: Managing Artistic Content**

**1. Upload Files:** Click on the **Upload** button to add new artistic content.
- Select files from your device or drag and drop them into the upload area.

**2. Organize Files:** Once uploaded, you can move files into specific rooms or folders.
- Use the drag-and-drop feature or right-click for more options.

**Step 5: Interacting with the Community**

**1. Explore:** Discover other artists and musicians by visiting their rooms.
- Use the **Explore** feature to browse curated lists or search for specific users.

**2. Follow and Connect:** Follow other users to keep up with their latest uploads.
- Leave comments or share feedback on their artwork or music tracks.

**Step 6: Account Settings and Support**

**1. Profile Settings:** Update your profile information, change your password, and manage your account settings.

**2. Support:** If you need help or have any questions, visit the **Help** section.
- Access FAQs, user guides, or contact the support team for assistance.

**Step 7: Logging Out**

**1. Logout:** When you are finished, securely log out of your account.
- Click on your profile icon in the top right corner and select **Logout.**

By following this guide, you should now be able to navigate the Artistry Hub platform confidently and make full use of its features. Enjoy your creative journey!

## VI. Technical Documentation

### Architecture

Artistry Hub is architected as a web application with a Java-based backend and a frontend utilizing web technologies for a responsive user experience.

### Libraries and Frameworks

**- Java:** Core programming language for backend logic.
**- Spring Boot:** Simplifies the setup and development of new Spring applications.
- **HTML/CSS/JavaScript:** Structures and styles the frontend, and manages user interactions.
- **Thymeleaf**: Integrates with Spring for dynamic web pages.

## API Documentation

Detail any RESTful APIs or backend services and their endpoints, request/response formats, and usage examples.

## System Dependencies

List all system dependencies and their versions, and provide instructions on how to install them.

1. Lombok: A Java library that helps reduce boilerplate code.
  - To install, add the Lombok dependency in your `pom.xml` or `build.gradle` file. For Maven, use:
```xml
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.x</version> <!-- Use the latest version -->
    <scope>provided</scope>
</dependency>
```
  - Additionally, install the Lombok plugin in IntelliJ for seamless integration.

2. Spring Security: A powerful authentication and access-control framework.
  - Add its dependency in your project's build configuration file:
```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

3. MySQL: A relational database management system.
  - Add the MySQL driver dependency:
```xml
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.x</version> <!-- Use the latest version -->
</dependency>
```
  - Ensure MySQL Server is installed on your system. You can download it from the official MySQL website.

4. Spring Web: For building web applications.
  - Include it via:
```xml
<dependency>
```

```
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
```

5. Thymeleaf: A modern server-side Java template engine.
   - To use Thymeleaf, add:
    ```xml
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    ```

6. API Development Tools: If you are creating APIs, you might also need:
   - Spring Boot Starter Data JPA, for database interaction.
   - Spring Boot Starter Validation, for validating request data.
   - Jackson for JSON parsing.

Replace `1.18.x` and `8.0.x` with the latest versions available at the time of your project setup. Ensure that IntelliJ IDEA is configured with the correct JDK version compatible with these dependencies.

## Artistry Hub API Documentation

This documentation provides an overview of the RESTful APIs available in the Artistry Hub application. Below you will find detailed descriptions of available endpoints, their functionalities, and examples of how to use them.

### Authentication

#### Register User

- POST `/api/register`
  - Description: Register a new user in the system.
  - Request Body:
    ```json
    {
      "username": "new_user",
      "email": "user@example.com",
      "password": "securepassword"
    }
    ```
  - Response: `201 Created`
    ```json
    {
      "message": "User registered successfully."
    }
    ```

**Login User**
- POST `/api/login`
  - Description: Authenticate a user and retrieve a token.
  - Request Body:
    ```json
    {
      "username": "existing_user",
      "password": "userpassword"
    }
    ```
  - Response: `200 OK`
    ```json
    {
      "token": "jwt-token-string"
    }
    ```

**User Profile**

**Get User Information**
- GET `/api/user/{userId}`
  - Description: Retrieve information for a specific user.
  - Response: `200 OK`
    ```json
    {
      "userId": "userId",
      "username": "existing_user",
      "email": "user@example.com"
      // Other user details
    }
    ```

**Update User Information**
- **PUT** `/api/user/{userId}`
  - Description: Update user information.
  - Request Body:
    ```json
    {
      "email": "newemail@example.com"
      // Other fields to update
    }
    ```
  - Response: `200 OK`
    ```json
    {
      "message": "User updated successfully."
    }
    ```

## Portfolio Management

### Create Portfolio

- **POST** `/api/portfolio`
  - Description: Create a new portfolio for storing artworks.
  - Request Body:
    ```json
    {
      "name": "My Art Collection",
      "description": "A collection of my best artworks."
      // Other portfolio details
    }
    ```
  - Response: `201 Created`
    ```json
    {
      "portfolioId": "new_portfolio_id",
      "name": "My Art Collection"
      // Other portfolio details
    }

### Retrieve Portfolios

- GET `/api/portfolios/{userId}`
  - Description: Retrieve all portfolios for a user.
  - Response: `200 OK`    [
      {
        "portfolioId": "portfolio_id",
        "name": "My Art Collection"
        // Other portfolio details
      },
    ]

## File Management

### Upload Artwork

- POST`/api/portfolio/{portfolioId}/artwork`
  - Description: Upload a new artwork to a portfolio.
  - Request Body: `FormData` with file and metadata.
  - Response: `201 Created`

    {
      "artworkId": "new_artwork_id",
      "fileName": "artwork.jpg"
       Other artwork details
    }

### Retrieve Artwork

- GET`/api/portfolio/{portfolioId}/artwork/{artworkId}`
  - Description: Retrieve details of a specific artwork.
    ```

- Response: `200 OK`

```
{
  "artworkId": "artwork_id",
  "fileName": "artwork.jpg"    }
```

Fill in the request and response examples based on your actual implementation. Make sure to replace the placeholder URLs and JSON with the actual paths and structures used in your application. If your services require specific headers, such as `Content-Type` or authorization tokens, include those details as well. This template should be adapted to match the functionality provided by the `PdfService`, `PortfolioService`, `PortfolioServiceImpl`, `UserService`, and `UserServiceImpl` in your project.

## VIII. Contact Information

For further inquiries or contributions, contact:
- Uwineza Bienheureuse: bienheureuseuwineza@gmail.com