

# VIRTUAL ART GALLERY – CASE STUDY

By Sara Pricilla

## Table of contents

S.No	Title	Pg.no
1.	Introduction	3
2.	Objective of the Project	3
3.	Overview	4
4.	Scope of the Project	5
5.	Purpose of the Project	5
6.	Software Requirements	6
7.	Implementation	7
8.	Coding Part	13
9.	Unit Testing	68
10.	Future Enhancements	74
11.	Conclusion	75

# 1.INTRODUCTION

The Virtual Art Gallery is a software application that provides a digital platform to explore, manage, and showcase various artworks. This system simulates an online art exhibition environment where users can view artworks, add them to their favorites, create galleries, and explore artist profiles. The goal is to combine software development principles with the aesthetic and management needs of an art gallery.

Built using object-oriented programming concepts and connected to a SQL database, the application ensures a structured, scalable, and interactive user experience. It also includes robust exception handling and unit testing to validate its functionalities.

## 2.OBJECTIVES OF THE PROJECT

The main objective of the Virtual Art Gallery project is to develop a user-friendly and interactive platform that allows users to explore, manage, and favorite various artworks in a digital format. The project is designed to replicate the functionality of a real-world art gallery by applying object-oriented programming (OOP) principles such as encapsulation, inheritance, and abstraction, ensuring a well-structured and maintainable codebase. A key goal is to implement full CRUD (Create, Read, Update, Delete) operations for essential entities like artworks, users, galleries, and artists, providing comprehensive data management capabilities.

The system is integrated with a relational SQL database (such as SQLite or MySQL) to handle the storage and retrieval of data efficiently. Another important objective is to implement robust error handling using custom exception classes, enabling the application to manage common runtime issues gracefully and improve overall user experience. The code is organized into a modular structure, dividing it into separate components such as entity, dao, util, exception, and main, which enhances code reusability, scalability, and clarity. Overall, the project aims to demonstrate core software development principles while offering a strong

foundation for future upgrades like graphical interfaces, web integration, or smart recommendation features.

### 3.OVERVIEW

The Virtual Art Gallery project is a console-based software application designed to provide users with a digital experience of browsing and managing artworks, artists, and galleries. It simulates the environment of a real-world art gallery by allowing users to interact with various elements of the system through a structured, menu-driven interface. The project emphasizes the use of Python programming along with object-oriented principles, ensuring the system is modular, maintainable, and easy to understand.

At its core, the application manages data related to artworks, users, galleries, and artists, all stored in a SQL-based relational database (such as SQLite or MySQL). The database integration enables efficient storage, retrieval, and manipulation of data. The backend logic handles CRUD (Create, Read, Update, Delete) operations, providing users with full control over the management of gallery content.

- The architecture of the project is cleanly divided into multiple modules:
- The entity module contains the core data models (like Artwork, User, Gallery).
- The dao (Data Access Object) module handles direct interaction with the database.
- The util module includes utility functions like database connections.
- The exception module defines custom exceptions for better error handling.
- The main module serves as the application's entry point, offering a menu-driven interface for smooth navigation.
- Through its simple console interface, users can perform a variety of actions such as viewing all available artworks, searching by artist or gallery, and managing user profiles. Administrators can update or delete records as needed. This project not only highlights the fundamentals of software

engineering but also lays a strong foundation for future expansions, such as transitioning to a graphical user interface or integrating web technologies.

## 4.SCOPE OF THE PROJECT

The scope of the Virtual Art Gallery system covers:

- **Artwork Management:** Adding, viewing, updating, and deleting artworks.
- **User Interaction:** Allowing users to mark and manage favorite artworks.
- **Gallery Creation:** Users or admins can create galleries and assign artists as curators.
- **Artist Management:** Artists can be added and managed within the system.
- **Object-Oriented Design:** The application uses real-world modeling through classes and objects.
- **Database Integration:** Artworks, users, and galleries are persisted using a relational database.
- **Testing and Exception Handling:** Ensures error-free operation and accurate functionality validation.

## 5.PURPOSE OF THE PROJECT

The primary purpose of the Virtual Art Gallery project is to create a digital platform that allows users to explore, manage, and experience artworks in an organized and interactive way. The system aims to simulate the functionality of a real-world art gallery while leveraging the benefits of software automation, such as accessibility, scalability, and user personalization.

This project is designed with educational and practical goals in mind. It showcases the effective use of object-oriented programming principles, database integration, exception handling, and modular architecture. The platform provides an opportunity for developers to understand how to build a fully functional backend system that handles real-life entities and operations.

### Key Objectives:

- **Digital Accessibility:** Make art galleries accessible to users anytime and from anywhere through a software-based system.
- **Efficient Art Management:** Allow users or curators to manage details of artworks, artists, galleries, and users easily through Create, Read, Update, and Delete (CRUD) functionalities.
- **User Engagement:** Enable users to interact with the system by favoriting artworks, exploring galleries, and discovering artist profiles.
- **Learning Platform:** Serve as a comprehensive learning project to demonstrate software engineering concepts such as:
  - Modular programming
  - OOP design patterns
  - SQL database interaction via pymysql
  - Custom exception handling
- **Scalability and Extensibility:** Lay the foundation for future expansion such as web integration, 3D walkthroughs, or AI-based artwork suggestions.

## 6.SOFTWARE REQUIREMENTS

### 1. Programming Language:

Python 3.x

Used for implementing the entire backend logic, following OOP principles and modular architecture.

### 2. Database:

MySQL A relational database is used to store and manage data related to users, artworks, galleries, and artists.

### 3. Development Environment / IDE:

VS Code, PyCharm, or any Python-compatible text editor

Provides code editing, debugging, and project navigation features.

#### 4. Terminal/Command Line Interface:

Required to run the menu-driven application and interact with the system.

## 7. IMPLEMENTATION

The application follows a layered architecture using the following packages:

- ``entity``: Contains pure POJO classes representing database tables.
- ``dao``: Contains service interfaces and their implementations for business logic and DB operations.
- ``exception``: Contains custom exception classes.
- ``util``: Contains utility classes for DB connection and property file handling.
- ``main``: Contains the entry point of the application and menu-driven UI logic

#### Entity Layer (``entity`` package)

Created the following classes with private variables, default and parameterized constructors, and standard getter/setter methods:

- `Artwork``
- `Artist``
- ``User``
- ``Gallery``
- `UserFavoriteArtwork`

#### DAO Layer (``dao`` package)

- Service interface contains the abstract classes
- Service Implementation: Implemented this interface in ``implementation.py``
- Use SQL queries with ``PreparedStatement``.
- Fetch connection using ``DBConnUtil.getConnection()``.

#### Exception Layer (``exception`` package)

Created the following custom exceptions:

- `ArtWorkNotFoundException`

- `UserNotFoundException`

Thrown these exceptions appropriately in the DAO methods when data is not found.

Utility Layer (`util` package)

DBPropertyUtil

- Method: `public static String getPropertyString(String fileName)`
- Reads property file and returns the DB connection string.

DBConnUtil

- Method: `public static Connection getConnection(String propertyFileName)`
- Uses `DBPropertyUtil` to fetch DB connection string and returns `Connection` object.

Main Application (`main` package)

- Create `MainModule` class with `main()` method.
- Display a menu-driven interface to:
  - Add, update, delete, and search artworks.
  - Add/remove favorites for a user.
  - View user's favorite artworks.
- Handle all exceptions using try-catch blocks.

Gallery Management ☐ (Handled by Me)

Files Involved:

Entity/gallery.py, dao/implementation.py, dao/interface.py



### Responsibilities:

- Designed and implemented the data model for galleries.
- Developed DAO methods to perform CRUD operations on gallery data.
- Implemented business logic for linking galleries with artworks and artists.
- Ensured data consistency and integrity across related entities.
- Added exception handling using custom exceptions for gallery operations.

### Artwork Management (Handled by Teammate)

#### Files Involved:

entity/artwork.py

The system is divided into multiple modules including Artwork Management and Gallery Management. I was responsible for the Gallery Management module. This involved defining the gallery entity, implementing DAO operations, and handling logic to associate galleries with artworks and artists. Exception handling and utility support were also part of my contribution. The Artwork Management module was handled by another team member.

#### 1. add\_gallery(gallery: Gallery)

Purpose: Adds a new gallery to the database.

##### Implementation:

- Validates the Gallery object (e.g., name, location).
- Inserts the gallery into the gallery table.
- Checks for duplicates before insertion.

#### 2. update\_gallery(gallery: Gallery)

Purpose: Updates existing gallery information.

##### Implementation:

- Finds the gallery by ID.

- Updates fields like name, address, or associated artwork count.  
Ensures data consistency by validating inputs.

### 3. search\_gallery\_by\_name(name\_keyword)

Purpose: Allows searching for galleries using partial or full names.

Implementation:

- Uses SQL LIKE queries to match gallery names.
- Returns a list of gallery objects matching the keyword.

### 4. view\_all\_galleries()

Purpose: Retrieves all galleries stored in the system.

Implementation:

- Fetches all rows from the gallery table.
- Converts them into Gallery objects for display.

### 5. search\_gallery\_by\_id(gallery\_id)

Purpose: Retrieves details of a specific gallery using its ID.

Implementation:

- Performs a direct fetch using the gallery ID.
- Returns a full object or raises an exception if not found.

### 6. remove\_gallery(gallery\_id)

Purpose: Deletes a gallery from the system.

Implementation:

- Checks if the gallery exists and whether it is associated with any artworks.
- Removes the gallery and its links safely (e.g., from any junction tables).
- Handles exceptions if deletion isn't allowed due to constraints.

### 7. gallery\_artist\_impact\_report()

Purpose: Generates a report of artists and their impact on galleries.

Implementation:

- Joins gallery and artist tables.
- Calculates metrics like number of exhibitions, artworks displayed, etc.
- Returns a formatted report or list of statistics.

#### 8. `get_artworks_in_gallery(gallery_id)`

Purpose: Lists all artworks displayed in a given gallery.

Implementation:

- Fetches artwork IDs linked to the gallery.
- Retrieves full artwork details and returns them.

#### 9. `get_galleries_for_artwork(artwork_id)`

Purpose: Shows in which galleries a specific artwork is/was exhibited.

Implementation:

- Looks up a mapping/junction table.
- Returns gallery names or objects associated with the artwork.

#### 10. `get_top_exhibited_artworks(top_n)`

Purpose: Displays the top N artworks based on how many galleries they've been shown in.

Implementation:

- Counts gallery associations for each artwork.
- Sorts and returns the top N based on count.

Exceptions Handled:

Custom Exception: `GalleryNotFoundException` and `GalleryNotAddedException`

Location: exception/exceptions.py

Base Class: Inherits from ArtGalleryException

Purpose: GalleryNotFoundException is a custom exception specifically designed to handle scenarios where a requested gallery does not exist in the system.

When It's Used:

- When a gallery ID provided for search, update, or delete does not match any entry in the database.
- When attempting to retrieve gallery details and the system fails to locate the corresponding record.

Benefits:

- Improves error handling by providing a clear and specific exception.
- Enhances code readability and maintainability.
- Helps in debugging by isolating gallery-specific errors.

Database Utility: DBConnection

Location: util/db\_utils.py

- Purpose: Provides a centralized and reusable way to establish a connection to the MySQL database used by the Virtual Art Gallery system.

Class: DBConnection

Method:

@staticmethod connect()

Establishes and returns a connection to the MySQL database using pymysql.

Credentials (user, password, host, database) are hardcoded for simplicity but can be externalized for production environments.

Error Handling:

- Catches pymysql.Error during connection attempts.
- Prints a descriptive error message if the connection fails.

- Terminates the program using `sys.exit(1)` to prevent further execution without a valid DB connection.

## 8.CODING PART

### **a.Database design:**

Tables:

- Artwork(ArtworkID, Title, Description, CreationDate, Medium, ImageURL, ArtistID)
- Artist(ArtistID, Name, Biography, BirthDate, Nationality, Website, ContactInfo)
- User(UserID, Username, Password, Email, FirstName, LastName, DOB, ProfilePic)
- Gallery(GalleryID, Name, Description, Location, CuratorID, OpeningHours)
- Artwork\_Gallery(ArtworkID, GalleryID)
- User\_Favorite\_Artwork(UserID, ArtworkID)

Relationships:

- Artwork - Artist (Many-to-One)
- Artwork - Gallery (Many-to-Many)
- User - Favorite Artwork (Many-to-Many)
- Artist - Gallery (One-to-Many)

DATABASE DESIGN:

```
create table Artwork(ArtworkID int primary key AUTO_INCREMENT,Title
varchar(100),Description varchar(250),CreationDate DATE,Medium
text,ImageURL varchar(500));
```

```
CREATE TABLE Artist(ArtistID int primary key,  
Name varchar(20),  
Biography varchar(200),  
BirthDate Date,  
Nationality varchar(20),  
Website varchar(100),  
ContactInformation varchar(500));
```

```
CREATE TABLE Users (  
    UserID INT AUTO_INCREMENT PRIMARY KEY,  
    Username VARCHAR(50) UNIQUE,  
    Password VARCHAR(200) ,  
    Email VARCHAR(100) UNIQUE,  
    FirstName VARCHAR(50),  
    LastName VARCHAR(50),  
    DateOfBirth DATE,  
    ProfilePicture VARCHAR(200),  
    FavoriteArtwork INT,  
    FOREIGN KEY (FavoriteArtwork) REFERENCES Artwork(ArtworkID)  
);
```

```
CREATE TABLE Gallery (  
    GalleryID INT PRIMARY KEY,
```

```

Name VARCHAR(100) ,
Description TEXT,
Location VARCHAR(200),
Curator INT,
OpeningHours VARCHAR(100),
FOREIGN KEY (Curator) REFERENCES Artist(ArtistID)
);

INSERT INTO Artwork (Title, Description, CreationDate, Medium, ImageURL)
VALUES

('Emperia Images',
'Emperia Art Gallery is an immersive, futuristic digital space showcasing
innovative and interactive virtual artworks.',
'2023-01-15',
'Digital Painting',
'https://emperiavr.com/wp-content/uploads/2021/03/ArtGalleries-1.jpg'),

('Art meets AR',
'Where art meets augmented reality, creating immersive, interactive, and
dynamic experiences that blend the physical and digital worlds.',
'2022-11-20',
'Augmented Reality (AR) Art',
'https://www.pluginr.com/augmented-reality/wp-content/uploads/2022/09/AR-
art-1024x536.jpg'),

```

('DIGITAL MUSEUM',

'A virtual art gallery within a digital museum, showcasing immersive 3D artworks, interactive exhibits, and multimedia experiences.',

'2021-05-10',

'Multimedia Installation',

'<https://www.invaluable.com/blog/wp-content/uploads/sites/77/2019/05/01-Kremer-Museum.jpg>'),

('Human Metamorphosis',

'An immersive virtual exhibit exploring human metamorphosis through dynamic digital transformations and evolving 3D forms.',

'2023-03-08',

'Motion Graphics',

'[https://img.freepik.com/premium-vector/natural-metamorphosis-human-head-tranquil-landscape-merge-art-prints\\_961038-5507.jpg?w=740](https://img.freepik.com/premium-vector/natural-metamorphosis-human-head-tranquil-landscape-merge-art-prints_961038-5507.jpg?w=740)'),

('Time meets Virtual Art',

'A captivating fusion where the concept of time intertwines with virtual art, creating immersive, ever-evolving digital experiences.',

'2023-07-01',

'3D Digital Art',

'<https://static.tnn.in/thumb/msid-117633497,thumbsize-1366252,width-1280,height-720,resizemode-75/117633497.jpg>'),

('Kids Favourite Art',



'A joyful collection of vibrant, playful, and whimsical digital artworks designed to spark imagination and delight young minds.',

'2022-12-05',

'Hand Painting',

'<https://familyforeverychild.org/wp-content/uploads/2022/01/69824735.jpeg>',

('Galaxy meets Virtual Art',

'An awe-inspiring virtual experience where cosmic wonders collide with digital artistry, immersing viewers in a galaxy of surreal visuals and interstellar creativity.',

'2024-01-10',

'Animation',

'<https://c02.purpledshub.com/uploads/sites/48/2021/06/solar-system-planets-2dff1b1.jpg?w=940&webp=1>',

('Delicious Explains the Art',

'A mouthwatering visual feast where vibrant colors, textures, and digital artistry come together to create deliciously captivating artworks.',

'2021-09-22',

'Canva Painting',

'[https://cdn11.bigcommerce.com/s-x49po/images/stencil/800w/products/127776/294765/prints%2Fdownscaled%2Fp\\_rp771ai9bxp\\_2000x2000\\_\\_65157.1713421001.jpg?c=2](https://cdn11.bigcommerce.com/s-x49po/images/stencil/800w/products/127776/294765/prints%2Fdownscaled%2Fp_rp771ai9bxp_2000x2000__65157.1713421001.jpg?c=2)',

('Exploring Simple and Wow Nature',

'A serene yet stunning virtual showcase capturing the simplicity and grandeur of nature through breathtaking digital landscapes and immersive art.',

'2024-02-28',

'Digital Painting',

'https://imgcdn.stablediffusionweb.com/2024/4/10/27d94784-0cd7-4096-ad0f-3b9a4cf53904.jpg'),

('Music Therapy',

'A harmonious blend of digital art and soothing soundscapes, offering a therapeutic, immersive experience that calms the mind and uplifts the spirit.',

'2023-10-18',

'Digital Art',

'https://i0.wp.com/london-post.co.uk/wp-content/uploads/2023/12/image002.jpeg?fit=649%2C433&ssl=1');

```
mysql> select * from artwork;
```

ArtworkID	Title	Medium	ImageURL	Description	CreatedOnDate
1	Emperia Images			Emperia Art Gallery is an immersive, futuristic digital space showcasing innovative and interactive virtual artworks.	2023-10-18
1-15	Digital Painting		https://emperiarvr.com/wp-content/uploads/2021/03/ArtGalleries-1.jpg	Where art meets augmented reality, creating immersive, interactive, and dynamic experiences that blend the physical and digital worlds.	2022-10-15
1-20	Augmented Reality (AR) Art		https://www.pluginr.com/augmented-reality/wp-content/uploads/2022/09/AR-art-1024x536.jpg	A virtual art gallery within a digital museum, showcasing immersive 3D artworks, interactive exhibits, and multimedia experiences.	2021-09-15
3	DIGITAL MUSEUM		https://www.invaluable.com/blog/wp-content/uploads/sites/77/2019/05/01-Kremer-Museum.jpg	An immersive virtual exhibit exploring human metamorphosis through dynamic digital transformations and evolving 3D forms.	2023-03-08
4	Human Metamorphosis		https://img.freepik.com/premium-vector/natural-metamorphosis-human-head-tranquil-landscape-merge-art-prints_961838-5507.jpg?w=740	A captivating fusion where the concept of time intertwines with virtual art, creating immersive, ever-evolving digital experiences.	2023-03-08
5	Motion Graphics		https://img.freepik.com/premium-vector/natural-metamorphosis-human-head-tranquil-landscape-merge-art-prints_961838-5507.jpg?w=740	A joyful collection of vibrant, playful, and whimsical digital artworks designed to spark imagination and delight young minds.	2022-10-15
5-01	3D Digital Art		https://static.tnn.in/thumb/msid-117633497,thumbsize-136252,width-1280,height-720,resizemode-75/117633497.jpg	An awe-inspiring virtual experience where cosmic wonders collide with digital artistry, immersing viewers in a galaxy of surreal visuals and interstellar creativity.	2024-02-28
6	Kids Favourite Art		https://familyforeverychild.org/wp-content/uploads/2022/01/69824735.jpeg		2025-03-08
7	Hand Painting		https://c92.purpledshub.com/uploads/sites/48/2021/06/solar-system-planets-20ff1b1.jpg?w=948&webp=1		2025-03-08
7-01	Galaxy meets Virtual Art		https://c92.purpledshub.com/uploads/sites/48/2021/06/solar-system-planets-20ff1b1.jpg?w=948&webp=1		2025-03-08
8	Animation		https://gntt.com		2025-03-08
8-07	zzz				2025-03-08
9	Exploring Simple and Wow Nature			A serene yet stunning virtual showcase capturing the simplicity and grandeur of nature through breathtaking digital landscapes and immersive art.	2024-02-28
2-28	Digital Painting		https://imgcdn.stablediffusionweb.com/2024/4/10/27d94784-0cd7-4096-ad0f-3b9a4cf53904.jpg		2024-02-28
10	Music Therapy			A harmonious blend of digital art and soothing soundscapes, offering a therapeutic, immersive experience that calms the mind and uplifts the spirit.	2023-10-18
0-18	Digital Art		https://i0.wp.com/london-post.co.uk/wp-content/uploads/2023/12/image002.jpeg?fit=649%2C433&ssl=1		2023-10-18
11	zzz				2025-03-08

INSERT INTO Artist (ArtistID, Name, Biography, BirthDate, Nationality, Website, ContactInformation)

VALUES

(101, 'Arjun Mehta', 'Contemporary digital artist specializing in virtual art galleries.', '1990-04-15', 'Indian', 'https://www.emperiavr.com', 'arjun.mehta@example.com'),

(102, 'Alex Rivera', 'Mixed media artist known for augmented reality exhibits.', '1987-09-21', 'American', 'https://www.artivive.com', 'alex.rivera@example.com'),

(103, 'Ishita Sharma', 'Abstract artist whose works are featured in major global tours.', '1993-07-03', 'Indian', 'https://www.theguardian.com/travel/2020/mar/23', 'ishita.sharma@example.com'),

(104, 'Taylor Morgan', 'Sculptor blending traditional and AR techniques.', '1985-02-10', 'Canadian', 'https://www.artstation.com/artwork/wrJm5L', 'taylor.morgan@example.com'),

(105, 'Rohan Patel', 'Painter focusing on nature and landscape themes.', '1992-05-27', 'Indian', 'https://www.shine.cn/feature/art-culture/2311227602/', 'rohan.patel@example.com'),

(106, 'Jordan Lee', 'Photographer and digital artist specializing in virtual spaces.', '1990-03-19', 'Australian', 'https://thinkplaycreate.org/explore/virtual-tours/', 'jordan.lee@example.com'),

(107, 'Ananya Iyer', 'Installation artist with a focus on virtual reality environments.', '1995-06-11', 'Indian', 'https://www.twinkl.cz/blog/twinkl-virtual-earth-day-art-gallery', 'ananya.iyer@example.com'),

(108, 'Casey Bennett', 'Artist known for interactive digital installations.', '1988-12-22', 'British', 'https://huwmessie.com/2020/03/04/the-virtual-artifact-gallery/', 'casey.bennett@example.com'),

(109, 'Kunal Desai', 'Oil painter blending traditional and digital techniques.', '1994-01-18', 'Indian', 'https://www.fizdi.com/trees-and-house-art\_4551\_27736-handpainted-art-painting-20in-x-20in/', 'kunal.desai@example.com'),

(110, 'Charlie Dawson', 'Multimedia artist specializing in music and art fusion.', '1986-08-09', 'American', 'https://harmonyarts.com/products/guitars-musical-instruments-paintings-artworks', 'charlie.dawson@example.com');

```
mysql> select * from artist;
```

ArtistID	Name	Biography	BirthDate	Nationality	Website	ContactInformation
101	Arjun Mehta	Contemporary digital artist specializing in virtual art galleries.	1990-04-15	Indian	https://www.emperiavr.com	un.mehta@example.com
102	Alex Rivera	Mixed media artist known for augmented reality exhibits.	1987-09-21	American	https://www.artivive.com	x.rivera@example.com
103	Ishita Sharma	Abstract artist whose works are featured in major global tours.	1993-07-03	Indian	https://www.theguardian.com/travel/2020/mar/23	ita.sharma@example.com
104	Taylor Morgan	Sculptor blending traditional and AR techniques.	1985-02-10	Canadian	https://www.artstation.com/artwork/wrJmSL	lor.morgan@example.com
105	Rohan Patel	Painter focusing on nature and landscape themes.	1992-05-27	Indian	https://www.shine.cn/feature/art-culture/2311227602/	an.patel@example.com
106	Jordan Lee	Photographer and digital artist specializing in virtual spaces.	1990-03-19	Australian	https://thinkplaycreate.org/explore/virtual-tours/	dan.lee@example.com
107	Ananya Iyer	Installation artist with a focus on virtual reality environments.	1995-06-11	Indian	https://www.twinkl.cz/blog/twinkl-virtual-earth-day-art-gallery	nya.iyer@example.com
108	Casey Bennett	Artist known for interactive digital installations.	1988-12-22	British	https://humessie.com/2020/03/04/the-virtual-artifact-gallery/	ey.bennett@example.com
109	Runal Desai	Oil painter blending traditional and digital techniques.	1994-01-18	Indian	https://www.fizdi.com/trees-and-house-art_4551_27736-handpainted-art-painting-20in-x-20in/	al.desai@example.com
110	Charlie Dawson	Multimedia artist specializing in music and art fusion.	1986-08-09	American	https://harmonyarts.com/products/guitars-musical-instruments-paintings-artworks	rlie.dawson@example.com
111	cricket	sport	2023-09-09	indian	https://rt.com	542896

INSERT INTO users (UserID, Username, Password, Email, FirstName, LastName, DateOfBirth, ProfilePicture, FavoriteArtwork)

VALUES

(001, 'priya', 'Priya@123', 'priya@example.com', 'Priya', 'Sharma', '1995-04-10', 'https://example.com/profiles/priya.jpg',4 ),

(002, 'charlie', 'Charlie@456', 'charlie@example.com', 'Charlie', 'Dawson', '1990-08-09', 'https://example.com/profiles/charlie.jpg',6 ),

(003, 'ram', 'Ram@789', 'ram@example.com', 'Ram', 'Patel', '1992-12-01', 'https://example.com/profiles/ram.jpg',5 ),

(004, 'sai', 'Sai@234', 'sai@example.com', 'Sai', 'Krishnan', '1998-06-15', 'https://example.com/profiles/sai.jpg',4 ),

(005, 'suba', 'Suba@567', 'suba@example.com', 'Suba', 'Iyer', '1994-11-23', 'https://example.com/profiles/suba.jpg',2),

(006, 'sudeesh', 'Sudeesh@890', 'sudeesh@example.com', 'Sudeesh', 'Menon', '1991-03-05', 'https://example.com/profiles/sudeesh.jpg', 5),

(007, 'pradeep', 'Pradeep@321', 'pradeep@example.com', 'Pradeep', 'Verma', '1989-07-11', 'https://example.com/profiles/pradeep.jpg', 6),

(008, 'alice', 'Alice@654', 'alice@example.com', 'Alice', 'Bennett', '1993-05-30', 'https://example.com/profiles/alice.jpg',1 ),

(009, 'bob', 'Bob@987', 'bob@example.com', 'Bob', 'Lee', '1996-01-17',  
'https://example.com/profiles/bob.jpg', 9),

(010, 'rocky', 'Rocky@543', 'rocky@example.com', 'Rocky', 'Desai', '1997-09-29',  
'https://example.com/profiles/rocky.jpg', 1);

```
mysql> select * from users;
```

UserID	Username	Password	Email	FirstName	LastName	DateOfBirth	ProfilePicture	FavoriteArtwork
1	priya	Priya@123	priya@example.com	Priya	Sharma	1995-04-10	https://example.com/profiles/priya.jpg	4
2	charlie	Charlie@456	charlie@example.com	Charlie	Dawson	1990-08-09	https://example.com/profiles/charlie.jpg	6
3	ram	Ram@789	ram@example.com	Ram	Patel	1992-12-01	https://example.com/profiles/ram.jpg	5
4	sai	Sai@234	sai@example.com	Sai	Krishnan	1998-06-15	https://example.com/profiles/sai.jpg	4
5	suba	Suba@567	suba@example.com	Suba	Iyer	1994-11-23	https://example.com/profiles/suba.jpg	2
6	sudeesh	Sudeesh@890	sudeesh@example.com	Sudeesh	Menon	1991-03-05	https://example.com/profiles/sudeesh.jpg	5
7	pradeep	Pradeep@321	pradeep@example.com	Pradeep	Verma	1989-07-11	https://example.com/profiles/pradeep.jpg	6
8	alice	Alice@654	alice@example.com	Alice	Bennett	1993-05-30	https://example.com/profiles/alice.jpg	1
9	bob	Bob@987	bob@example.com	Bob	Lee	1996-01-17	https://example.com/profiles/bob.jpg	9
10	rocky	Rocky@543	rocky@example.com	Rocky	Desai	1997-09-29	https://example.com/profiles/rocky.jpg	1

10 rows in set (0.00 sec)

INSERT INTO Gallery (GalleryID, Name, Description, Location, Curator,  
OpeningHours)

VALUES

(10, 'art sphere', 'Art unites everyone together', 'india', 104, '9 am to 1 pm'),

(20, 'zumba art', 'experiencing each and every art brings peace', 'indonesia', 106, '8  
am to 10 am'),

(30, 'flying colours', 'life becomes flying colours when you work hard', 'singapore',  
102, '6pm to 8 pm'),

(40, 'wondering sculpture', 'mind blowing ideas and pictures', 'malaysia', 101, '10  
am to 12 pm'),

(50, 'beauty of art', 'beauty lies in the way we design an art', 'indore', 102, '1 pm to  
4 pm'),

(60, 'epics of art', 'whole universe is made of art', 'bhutan', 104, '2 pm to 4 pm');

```
mysql> select * from gallery;
```

GalleryID	Name	Description	Location	Curator	OpeningHours
10	art sphere	Art unites everyone together	india	104	9 am to 1 pm
20	zumba art	experiencing each and every art brings peace	indonesia	106	8 am to 10 am
30	album	explore	france	104	6 am to 11 pm
40	wondering sculpture	mind blowing ideas and pictures	malaysia	101	10 am to 12 pm
50	beauty of art	beauty lies in the way we design an art	indore	102	1 pm to 4 pm
60	epics of art	whole universe is made of art	bhutan	104	2 pm to 4 pm
61	sara	hello	chennai	105	6 am to 10 pm

7 rows in set (0.00 sec)

```
CREATE TABLE User_Favorite_Artwork (
```

```
    UserID INT,
```

```
    ArtworkID INT,
```

```
    PRIMARY KEY (UserID, ArtworkID),
```

```
    FOREIGN KEY (UserID) REFERENCES Users(UserID),
```

```
    FOREIGN KEY (ArtworkID) REFERENCES Artwork(ArtworkID)
```

```
);
```

```
INSERT INTO User_Favorite_Artwork (UserID, ArtworkID) VALUES
```

```
(1, 1), (1, 2),(1,3),
```

```
(2, 1), (2, 4),
```

```
(3, 2), (3, 6),
```

```
(4, 3), (4,5),
```

```
(5, 1), (5, 5), (5, 8),
```

```
(6, 9), (6,7),
```

```
(7, 9), (7, 2),
```

```
(8, 1),(8,2),
```

(9, 6),

(10, 9), (10, 8);

```
mysql> select * from user_favorite_artwork;
```

UserID	ArtworkID
1	2
3	2
7	2
8	2
1	3
4	3
2	4
4	5
5	5
6	7
10	8
6	9
7	9
10	9
7	12

15 rows in set (0.01 sec)

```
CREATE TABLE Artwork_Gallery (  
    ArtworkID INT,  
    GalleryID INT,  
    PRIMARY KEY (ArtworkID, GalleryID),  
    FOREIGN KEY (ArtworkID) REFERENCES Artwork(ArtworkID),  
    FOREIGN KEY (GalleryID) REFERENCES Gallery(GalleryID)  
);
```

```
INSERT INTO Artwork_Gallery (ArtworkID, GalleryID) VALUES  
(1, 10), (1, 20),  
(2, 60),  
(3, 40), (3, 50),
```

(4, 20),  
(5, 30), (5, 10),  
(6, 60),  
(7, 40), (7, 60) ,  
(8, 10),  
(9, 40), (9, 60),  
(10, 10);

```
mysql> select * from artwork_gallery;
```

ArtworkID	GalleryID
1	10
5	10
8	10
10	10
1	20
4	20
5	30
3	40
7	40
9	40
3	50
2	60
6	60
7	60
9	60

15 rows in set (0.01 sec)

alter table artwork add column ArtistID int, add constraint ar\_foreign foreign key (ArtistID) references artist(ArtistID);

insert into artwork(artistid)  
values(101),(102),(103),(102),(105),(104),(106),(107),(108),(109);

alter table artwork drop constraint ar\_foreign;



```
alter table artwork drop column artistid;
```

## **b. Python modules:**

project-root

```
|— dao
|  |— __init__.py
|  |— implementation.py
|  |— interface.py
|— entity
|  |— artist.py
|  |— artwork.py
|  |— gallery.py
|  |— userfavoriteartwork.py
|— exception
|  |— exceptions.py
|— util
|  |— db_utils.py
```

Main.py

## **Artist.py:**

```
class Artist:
    def __init__(self, artist_id, name, biography, birth_date, nationality,
website, contact_info):
        self._artist_id = artist_id
```

```

        self._name = name
        self._biography = biography
        self._birth_date = birth_date
        self._nationality = nationality
        self._website = website
        self._contact_info = contact_info

# Getters
def get_artist_id(self):
    return self._artist_id

def get_name(self):
    return self._name

def get_biography(self):
    return self._biography

def get_birth_date(self):
    return self._birth_date

def get_nationality(self):
    return self._nationality

def get_website(self):
    return self._website

def get_contact_info(self):
    return self._contact_info

# Setters
def set_artist_id(self, artist_id):
    self._artist_id = artist_id

def set_name(self, name):
    self._name = name

def set_biography(self, biography):
    self._biography = biography

def set_birth_date(self, birth_date):
    self._birth_date = birth_date

def set_nationality(self, nationality):
    self._nationality = nationality

```

```
def set_website(self, website):  
    self._website = website  
  
def set_contact_info(self, contact_info):  
    self._contact_info = contact_info
```

## artwork.py:

```
class Artwork:  
    def __init__(self, artwork_id, title, description, creation_date, medium,  
image_url):  
        self._artwork_id = artwork_id  
        self._title = title  
        self._description = description  
        self._creation_date = creation_date  
        self._medium = medium  
        self._image_url = image_url  
  
    # Getters  
    def get_artwork_id(self):  
        return self._artwork_id  
  
    def get_title(self):  
        return self._title  
  
    def get_description(self):  
        return self._description  
  
    def get_creation_date(self):  
        return self._creation_date  
  
    def get_medium(self):  
        return self._medium  
  
    def get_image_url(self):  
        return self._image_url  
  
    # Setters  
    def set_artwork_id(self, artwork_id):  
        self._artwork_id = artwork_id  
  
    def set_title(self, title):
```

```
self._title = title

def set_description(self, description):
    self._description = description

def set_creation_date(self, creation_date):
    self._creation_date = creation_date

def set_medium(self, medium):
    self._medium = medium

def set_image_url(self, image_url):
    self._image_url = image_url
```

## **gallery.py:**

```
class Artwork:
    def __init__(self, artwork_id, title, description, creation_date, medium,
image_url):
        self._artwork_id = artwork_id
        self._title = title
        self._description = description
        self._creation_date = creation_date
        self._medium = medium
        self._image_url = image_url

    # Getters
    def get_artwork_id(self):
        return self._artwork_id

    def get_title(self):
        return self._title

    def get_description(self):
        return self._description

    def get_creation_date(self):
        return self._creation_date

    def get_medium(self):
        return self._medium
```

```

def get_image_url(self):
    return self._image_url

# Setters
def set_artwork_id(self, artwork_id):
    self._artwork_id = artwork_id

def set_title(self, title):
    self._title = title

def set_description(self, description):
    self._description = description

def set_creation_date(self, creation_date):
    self._creation_date = creation_date

def set_medium(self, medium):
    self._medium = medium

def set_image_url(self, image_url):
    self._image_url = image_url

```

## userfavoriteartwork.py:

```

class UserFavoriteArtwork:
    def __init__(self, user_id: int = 0, artwork_id: int = 0):
        self._user_id = user_id
        self._artwork_id = artwork_id

    def get_user_id(self):
        return self._user_id

    def set_user_id(self, value):
        self._user_id = value

    def get_artwork_id(self):
        return self._artwork_id

    def set_artwork_id(self, value):
        self._artwork_id = value

```

## exceptions.py

```
class ArtGalleryException(Exception):
    pass

class ArtworkNotFoundException(ArtGalleryException):
    pass

class ArtistNotFoundException(ArtGalleryException):
    pass

class GalleryNotAddedException(ArtGalleryException):
    pass

class FavoriteNotFoundException(ArtGalleryException):
    pass

class UserNotFoundException(ArtGalleryException):
    pass

class GalleryNotFoundException(ArtGalleryException):
    pass
```

## interface.py:

```
from abc import ABC, abstractmethod
from entity.gallery import Gallery
from entity.userfavoriteartwork import UserFavoriteArtwork

class Interface(ABC):

    @abstractmethod
    def view_artworks(self):
        pass

    @abstractmethod
    def add_artwork(self, title, description, medium, image_url):
        pass

    @abstractmethod
    def update_artwork(self, artwork):
        pass
```

```

@abstractmethod
def remove_artwork(self, identifier):
    pass

@abstractmethod
def get_artwork_by_id(self, artwork_id):
    pass

@abstractmethod
def search_artworks(self, keyword):
    pass

@abstractmethod
def add_artwork_to_favorite(self, favorite: UserFavoriteArtwork):
    pass

@abstractmethod
def remove_artwork_from_favorite(self, favorite: UserFavoriteArtwork):
    pass

@abstractmethod
def get_user_favorite_artworks(self, user_id):
    pass

@abstractmethod
def add_artist(self, name, biography, birth_date, nationality, website,
contact_info):
    pass

@abstractmethod
def add_gallery(self, gallery: Gallery):
    pass

@abstractmethod
def update_gallery(self, gallery: Gallery):
    pass

@abstractmethod
def search_gallery_by_name(self, name_keyword):
    pass

@abstractmethod

```

```

def view_all_galleries(self):
    pass

@abstractmethod
def search_gallery_by_id(self, gallery_id):
    pass

@abstractmethod
def remove_gallery(self, gallery_id):
    pass

@abstractmethod
def gallery_artist_impact_report(self):
    pass

@abstractmethod
def get_artworks_in_gallery(self, gallery_id):
    pass

@abstractmethod
def get_galleries_for_artwork(self, artwork_id):
    pass

@abstractmethod
def get_top_exhibited_artworks(self, top_n):
    pass

```

## **implementation.py:**

```

import pymysql
from dao.interface import Interface
from tabulate import tabulate
from datetime import datetime
from entity.artwork import Artwork
from entity.gallery import Gallery
from entity.userfavoriteartwork import UserFavoriteArtwork
from exception.exceptions import (

```



```

    ArtworkNotFoundException,
    FavoriteNotFoundException,
    UserNotFoundException,
    GalleryNotAddedException,
    GalleryNotFoundException
)

class VirtualArtGalleryDAO(Interface):
    def __init__(self, conn):
        self.conn = conn
        self.cursor = conn.cursor()
    def view_artworks(self):
        try:
            self.cursor.execute("SELECT * FROM artwork")
            rows = self.cursor.fetchall()
            if not rows:
                raise ArtworkNotFoundException("No artworks found.")
            print()
            print("\nArtworks:")
            for row in rows:
                print(f"Artwork ID: {row[0]}")
                print(f"Title: {row[1]}")
                print(f"Medium: {row[2]}")
                print(f"Artist ID: {row[3]}")
                print(f"Year: {row[4]}")
                print("-" * 30) # Separator for better readability
            except pymysql.Error as e:
                print(f"Database Error: {e}")

    def add_artwork(self, title, description, medium, image_url):
        try:
            creation_date = datetime.now().strftime('%Y-%m-%d') # current date in
            YYYY-MM-DD format
            query = """
                INSERT INTO artwork (Title, Description, CreationDate, Medium,
            ImageURL)
                VALUES (%s, %s, %s, %s, %s)
            """
            self.cursor.execute(query, (title, description, creation_date, medium,
            image_url))

```

```

        self.conn.commit()
        print()
        print("/n Artwork added successfully!")

        self.cursor.execute("SELECT * FROM artwork WHERE ArtworkID =
LAST_INSERT_ID()")
        inserted_artwork = self.cursor.fetchone()

        if inserted_artwork:
            artwork_id, title, description, creation_date, medium, image_url =
inserted_artwork
            print("\n--- Inserted Artwork ---")
            print(f"Artwork ID   : {artwork_id}")
            print(f"Title       : {title}")
            print(f"Description  : {description}")
            print(f"Creation Date : {creation_date}")
            print(f"Medium       : {medium}")
            print(f"Image URL    : {image_url}")
            print("-" * 40)
        except pymysql.Error as e:
            print(f"Error Adding Artwork: {e}")

```

```
C:\Users\admin\Downloads\virtual-art-gallery>python main.py
```

```
=====
🎨 WELCOME TO THE VIRTUAL ART GALLERY 🎨
=====
```

- a. Artwork Management
- b. Gallery Management
- c. Exit

Enter your choice (a/b/c): a

Virtual Art Gallery

- 1. Create Account for artist
  - 2. View Artworks
  - 3. Add Artwork
  - 4. Update Artwork
  - 5. Remove Artwork
  - 6. View Your Artwork with ID
  - 7. Search Artworks
  - 8. Add Artwork to Favorites
  - 9. Remove Artwork from Favorites
  - 10. View User's Favorite Artworks
  - 11. Back to main menu
- Enter your choice: 2

Artworks:

Artwork ID: 1

Title: Emperia Images

Medium: Emperia Art Gallery is an immersive, futuristic digital space showcasing innovative and

Artist ID: 2023-01-15

Year: Digital Painting

Artwork ID: 2

Title: Art meets AR

Medium: Where art meets augmented reality, creating immersive, interactive, and dynamic experie

Artist ID: 2022-11-20

Year: Augmented Reality (AR) Art

Artwork ID: 3

Title: DIGITAL MUSEUM

Medium: A virtual art gallery within a digital museum, showcasing immersive 3D artworks, intera

Artist ID: 2021-05-10

Year: Multimedia Installation

Artwork ID: 4

Title: premire

Medium: nothing

Artist ID: 2025-04-07

Year: digital art

Artwork ID: 5

Title: Time meets Virtual Art

Medium: A captivating fusion where the concept of time intertwines with virtual art, creating

Artist ID: 2023-07-01

```
def update_artwork(self, artwork):
```

```
    try:
```

```
        creation_date = datetime.now().strftime('%Y-%m-%d')
```

```
        query = """
```

```
            UPDATE artwork
```

```

        SET Title=%s, Description=%s, CreationDate=%s, Medium=%s,
ImageURL=%s
        WHERE ArtworkID=%s
        """
    self.cursor.execute(query, (
        artwork.get_title(),
        artwork.get_description(),
        creation_date,
        artwork.get_medium(),
        artwork.get_image_url(),
        artwork.get_artwork_id()
    ))
    self.conn.commit()

    if self.cursor.rowcount == 0:
        print("Artwork ID not found.")
        return
    print()
    print("\nArtwork updated successfully!")

    # Display updated artwork
    self.cursor.execute("SELECT * FROM artwork WHERE ArtworkID =
%s", (artwork.get_artwork_id(),))
    updated_artwork = self.cursor.fetchone()

    if updated_artwork:
        artwork_id, title, description, creation_date, medium, image_url =
updated_artwork
        print("\n--- Updated Artwork ---")
        print(f"Artwork ID   : {artwork_id}")
        print(f>Title       : {title}")
        print(f>Description  : {description}")
        print(f"Creation Date : {creation_date}")
        print(f"Medium       : {medium}")
        print(f"Image URL    : {image_url}")
        print("-" * 40)

    except pymysql.Error as e:
        print(f"Error Updating Artwork: {e}")

```

```

🖌 Update Artwork
Enter Artwork ID to update: 4
Title: premire
Description: nothing
Medium: digital art
Image URL: https://picture.com

Artwork updated successfully!

--- Updated Artwork ---
Artwork ID      : 4
Title           : premire
Description      : nothing
Creation Date   : 2025-04-07
Medium          : digital art
Image URL       : https://picture.com
-----

```

```

def add_artwork(self, title, description, medium, image_url):
    try:
        creation_date = datetime.now().strftime('%Y-%m-%d') # current date in
        YYYY-MM-DD format
        query = """
            INSERT INTO artwork (Title, Description, CreationDate, Medium,
            ImageURL)
            VALUES (%s, %s, %s, %s, %s)
            """
        self.cursor.execute(query, (title, description, creation_date, medium,
            image_url))
        self.conn.commit()
        print()
        print("/n Artwork added successfully!")

        self.cursor.execute("SELECT * FROM artwork WHERE ArtworkID =
        LAST_INSERT_ID()")
        inserted_artwork = self.cursor.fetchone()

        if inserted_artwork:
            artwork_id, title, description, creation_date, medium, image_url =
            inserted_artwork
            print("\n--- Inserted Artwork ---")
            print(f"Artwork ID   : {artwork_id}")
            print(f"Title       : {title}")
            print(f"Description  : {description}")

```

```

        print(f"Creation Date : {creation_date}")
        print(f"Medium      : {medium}")
        print(f"Image URL   : {image_url}")
        print("-" * 40)
    except pymysql.Error as e:
        print(f"Error Adding Artwork: {e}")

```

```

Virtual Art Gallery
1. Create Account for artist
2. View Artworks
3. Add Artwork
4. Update Artwork
5. Remove Artwork
6. View Your Artwork with ID
7. Search Artworks
8. Add Artwork to Favorites
9. Remove Artwork from Favorites
10. View User's Favorite Artworks
11. Back to main menu
Enter your choice: 3
Title: hexaware
Description: u can explore
Medium: online painting
Image URL: https://picture.jpg

/n Artwork added successfully!

--- Inserted Artwork ---
Artwork ID      : 38
Title           : hexaware
Description      : u can explore
Creation Date   : 2025-04-07
Medium          : online painting
Image URL       : https://picture.jpg
-----

```

```

def remove_artwork(self, identifier):
    try:
        query = "DELETE FROM artwork WHERE ArtworkID = %s OR Title = %s"
        self.cursor.execute(query, (identifier, str(identifier)))
        self.conn.commit()

        if self.cursor.rowcount == 0:
            raise ArtworkNotFoundException("Artwork not found.")
        print()
        print("❑ Artwork removed successfully!")
        print("❑ We're always evolving! Feel free to add your next masterpiece anytime.")

```

```
print("🎨 Visit us again for more artistic inspiration!")
```

```
except pymysql.Error as e:  
    print(f"Error Removing Artwork: {e}")
```

```
Virtual Art Gallery  
1. Create Account for artist  
2. View Artworks  
3. Add Artwork  
4. Update Artwork  
5. Remove Artwork  
6. View Your Artwork with ID  
7. Search Artworks  
8. Add Artwork to Favorites  
9. Remove Artwork from Favorites  
10. View User's Favorite Artworks  
11. Back to main menu  
Enter your choice: 5  
Enter Artwork ID or Title to remove: 38  
  
✅ Artwork removed successfully!  
🚀 We're always evolving! Feel free to add your next masterpiece anytime.  
🎨 Visit us again for more artistic inspiration!
```

```
def get_artwork_by_id(self, artwork_id):  
    try:  
        query = "SELECT * FROM artwork WHERE ArtworkID = %s"  
        self.cursor.execute(query, (artwork_id,))  
        row = self.cursor.fetchone()  
  
        if not row:  
            raise ArtworkNotFoundException("Artwork not found.")  
  
        # Create Artwork object  
        artwork = Artwork(  
            artwork_id=row[0],  
            title=row[1],  
            description=row[2],  
            creation_date=row[3],  
            medium=row[3],  
            image_url=row[4]  
        )  
  
        # Print details with separator  
        print("\n" + "-"*40)
```

```

print("🎨 Artwork Details")
print("-"*40)
print(f"Artwork ID : {artwork.get_artwork_id()}")
print(f"Title      : {artwork.get_title()}")
print(f"Description : {artwork.get_description()}")
print(f"Medium      : {artwork.get_medium()}")
print(f"Image URL   : {artwork.get_image_url()}")
print("-"*40 + "\n")

```

```

return artwork

```

```

except pymysql.Error as e:
    print(f"Error Fetching Artwork: {e}")

```

```

Virtual Art Gallery
1. Create Account for artist
2. View Artworks
3. Add Artwork
4. Update Artwork
5. Remove Artwork
6. View Your Artwork with ID
7. Search Artworks
8. Add Artwork to Favorites
9. Remove Artwork from Favorites
10. View User's Favorite Artworks
11. Back to main menu
Enter your choice: 6
Enter Artwork ID to view: 4

```

```

-----
🎨 Artwork Details
-----

```

```

Artwork ID   : 4
Title        : premire
Description   : nothing
Medium       : 2025-04-07
Image URL    : digital art
-----

```

```

def search_artworks(self, keyword):
    try:
        query = "SELECT * FROM artwork WHERE Title LIKE %s"
        self.cursor.execute(query, (f"% {keyword} %",))

        rows = self.cursor.fetchall()

        if not rows:

```



```

        raise ArtworkNotFoundException("No matching artworks found.")

    print("\n❏ Search Results:")
    print("-" * 40)
    for row in rows:
        artwork = Artwork(
            artwork_id=row[0],
            title=row[1],
            description=row[2],
            creation_date=None,
            medium=row[3],
            image_url=row[4]
        )
        print(f"Artwork ID : {artwork.get_artwork_id()}")
        print(f"Title      : {artwork.get_title()}")
        print(f"Description : {artwork.get_description()}")
        print(f"Medium      : {artwork.get_medium()}")
        print(f"Image URL   : {artwork.get_image_url()}")
        print("-" * 40)

    except pymysql.Error as e:
        print(f"Error Searching Artworks: {e}")

```

```

Virtual Art Gallery
1. Create Account for artist
2. View Artworks
3. Add Artwork
4. Update Artwork
5. Remove Artwork
6. View Your Artwork with ID
7. Search Artworks
8. Add Artwork to Favorites
9. Remove Artwork from Favorites
10. View User's Favorite Artworks
11. Back to main menu
Enter your choice: 7
Enter keyword to search in artwork titles: Time

🔍 Search Results:
-----
Artwork ID   : 5
Title        : Time meets Virtual Art
Description   : A captivating fusion where the concept of time intertwines with virt
Medium       : 2023-07-01
Image URL    : 3D Digital Art
-----

```

```

def add_artwork_to_favorite(self, favorite: UserFavoriteArtwork):
    try:
        user_id = favorite.get_user_id()

```

```

artwork_id = favorite.get_artwork_id()

# 1. Check if user exists
check_user_query = "SELECT 1 FROM users WHERE UserID = %s"
self.cursor.execute(check_user_query, (user_id,))
if not self.cursor.fetchone():
    raise UserNotFoundException("User not found")

# 2. Check if artwork exists
check_artwork_query = "SELECT 1 FROM artwork WHERE ArtworkID =
%s"
self.cursor.execute(check_artwork_query, (artwork_id,))
if not self.cursor.fetchone():
    raise ArtworkNotFoundException("Artwork not found")

# 3. Insert into user_favorite_artwork table
query = "INSERT INTO user_favorite_artwork (UserID, ArtworkID)
VALUES (%s, %s)"
self.cursor.execute(query, (user_id, artwork_id))
self.conn.commit()
print()
print(f"👍 Favorited! Artwork [{artwork_id}] is now saved for User
[{user_id}].")

except UserNotFoundException as e:
    print(f"User Error: {e}")
except ArtworkNotFoundException as e:
    print(f"Artwork Error: {e}")
except pymysql.IntegrityError:
    print("This favorite already exists.")
except pymysql.Error as e:
    print(f"Database Error: {e}")

```

```

Virtual Art Gallery
1. Create Account for artist
2. View Artworks
3. Add Artwork
4. Update Artwork
5. Remove Artwork
6. View Your Artwork with ID
7. Search Artworks
8. Add Artwork to Favorites
9. Remove Artwork from Favorites
10. View User's Favorite Artworks
11. Back to main menu
Enter your choice: 8
Enter User ID: 7
Enter Artwork ID: 3

👍 Favorited! Artwork [3] is now saved for User [7].

```

```

def remove_artwork_from_favorite(self, favorite: UserFavoriteArtwork):
    try:
        artwork_id = favorite.get_artwork_id()

        query = "DELETE FROM user_favorite_artwork WHERE ArtworkID =
%s"
        self.cursor.execute(query, (artwork_id,))
        self.conn.commit()

        if self.cursor.rowcount == 0:
            raise FavoriteNotFoundException("Favorite not found.")

        print(f"❤️ Farewell, Artwork #{artwork_id} — you've been unfavorited
with style.")

    except FavoriteNotFoundException as e:
        print(f"❌ {e}")

    except pymysql.Error as e:
        print(f"❌ Error Removing from Favorites: {e}")

```

```

Virtual Art Gallery
1. Create Account for artist
2. View Artworks
3. Add Artwork
4. Update Artwork
5. Remove Artwork
6. View Your Artwork with ID
7. Search Artworks
8. Add Artwork to Favorites
9. Remove Artwork from Favorites
10. View User's Favorite Artworks
11. Back to main menu
Enter your choice: 9
Enter Artwork ID to remove from favorites: 3

❤️ Farewell, Artwork #3 - you've been unfavorited with style.
Virtual Art Gallery

```

```

def get_user_favorite_artworks(self, user_id):
    try:
        query = """
        SELECT a.ArtworkID, a.Title, a.Description, a.CreationDate, a.Medium,
a.ImageURL
        FROM artwork a
        JOIN user_favorite_artwork ufa ON a.ArtworkID = ufa.ArtworkID
        WHERE ufa.UserID = %s
        """

```

```

self.cursor.execute(query, (user_id,))
rows = self.cursor.fetchall()

if not rows:
    raise FavoriteNotFoundException("No favorite artworks found.")

print("\n□ User's Favorite Artworks:")
for row in rows:
    print(f"□ Artwork ID: {row[0]}")
    print(f"□ Title: {row[1]}")
    print(f"□ Description: {row[2]}")
    print(f"□ Created On: {row[3]}")
    print(f"□ Medium: {row[4]}")
    print(f"□ Image URL: {row[5]}")
    print("-" * 40)

except FavoriteNotFoundException as e:
    print(f"□ {e}")

except pymysql.Error as e:
    print(f"□ Error Fetching Favorites: {e}")

```

```

Virtual Art Gallery
1. Create Account for artist
2. View Artworks
3. Add Artwork
4. Update Artwork
5. Remove Artwork
6. View Your Artwork with ID
7. Search Artworks
8. Add Artwork to Favorites
9. Remove Artwork from Favorites
10. View User's Favorite Artworks
11. Back to main menu
Enter your choice: 10
Enter User ID to view favorite artworks: 4

🎨 User's Favorite Artworks:
🆔 Artwork ID: 5
🖌️ Title: Time meets Virtual Art
📄 Description: A captivating fusion where the concept of time intertwines with virtual art,
📅 Created On: 2023-07-01
🎨 Medium: 3D Digital Art
🖼️ Image URL: https://static.tnn.in/thumb/msid-117633497,thumbsize-1366252,width-1280,height-
-----

```

```

def add_artist(self, name, biography, birth_date, nationality, website,
contact_info):
    try:
        query = """
        INSERT INTO Artist (Name, Biography, BirthDate, Nationality,
Website, ContactInformation)
        VALUES (%s, %s, %s, %s, %s, %s)
        """

        self.cursor.execute(query, (name, biography, birth_date, nationality,
website, contact_info))
        self.conn.commit()

        # Fetch the auto-generated ArtistID
        artist_id = self.cursor.lastrowid

        print("\nArtist added successfully!")
        print(f"Artist ID: {artist_id}")
        print(f"Name: {name}")
        print(f"Biography: {biography}")
        print(f"Birth Date: {birth_date}")
        print(f"Nationality: {nationality}")
        print(f"Website: {website}")
        print(f"Contact Information: {contact_info}")
    except pymysql.Error as e:
        print(f"Error Adding Artist: {e}")

```

```

=====
🎨 WELCOME TO THE VIRTUAL ART GALLERY 🎨
=====
a. Artwork Management
b. Gallery Management
c. Exit

Enter your choice (a/b/c): a

Virtual Art Gallery
1. Create Account for artist
2. View Artworks
3. Add Artwork
4. Update Artwork
5. Remove Artwork
6. View Your Artwork with ID
7. Search Artworks
8. Add Artwork to Favorites
9. Remove Artwork from Favorites
10. View User's Favorite Artworks
11. Back to main menu
Enter your choice: 1
Name: charles
Biography: work
Birth Date (YYYY-MM-DD): 2024-09-07
Nationality: Indian
Website: https://wwoi.com
Contact Information: 8765433290

Artist added successfully!
Artist ID: 112
Name: charles
Biography: work
Birth Date: 2024-09-07
Nationality: Indian
Website: https://wwoi.com
Contact Information: 8765433290

```

```

def add_gallery(self, gallery: Gallery):
    try:
        query = """
            INSERT INTO gallery (Name, Description, Location, Curator,
OpeningHours)
            VALUES (%s, %s, %s, %s, %s)
            """
        self.cursor.execute(query, (
            gallery.get_name(),
            gallery.get_description(),
            gallery.get_location(),
            gallery.get_curator(),
            gallery.get_opening_hours()
        ))
        self.conn.commit()

        gallery_id = self.cursor.lastrowid
        print("\n❑ Gallery Successfully Added!")
        print("=" * 40)
        print(f"❑ Gallery ID    : {gallery_id}")
        print(f"❑ Name          : {gallery.get_name()}")
        print(f"❑ Description   : {gallery.get_description()}")
        print(f"❑ Location     : {gallery.get_location()}")
        print(f"❑ Curator ID    : {gallery.get_curator()}")
        print(f"❑ Opening Hours : {gallery.get_opening_hours()}")

```

```

print("=" * 40 + "\n")

except pymysql.Error as e:
    raise GalleryNotAddedException(f"❌ Error Adding Gallery: {e}")

```

```

-----
11. view all Galleries
12. Create Gallery
13. Update Gallery
14. Search Gallery By Name
15. Search by ID
16. Remove gallery
17. Gallery Artist Impact Report
18. Get Artworks From Gallery
19. Get Artworks From Gallery
20. Top Exhibited Artworks
21. Back to main menu
Enter Your Choice:12
Enter Gallery Name: hexaware
Enter Description: nothing
Enter Location: chennai
Enter Curator ID (leave blank if none): 103
Enter Opening Hours: 4 pm to 8 pm

🏛️ Gallery Successfully Added!
=====
🎨 Gallery ID      : 65
🖼️ Name           : hexaware
📄 Description     : nothing
📍 Location        : chennai
👤 Curator ID      : 103
🕒 Opening Hours   : 4 pm to 8 pm
=====

```

```

def update_gallery(self, gallery: Gallery):
    try:
        # Check if the gallery exists
        self.cursor.execute("SELECT 1 FROM gallery WHERE GalleryID = %s",
(gallery.get_gallery_id(),))
        if not self.cursor.fetchone():
            raise GalleryNotFoundException(f"Gallery with ID
{gallery.get_gallery_id()} not found.")

        # Perform the update
        query = """
            UPDATE gallery

```

```

        SET Name = %s, Description = %s, Location = %s, Curator = %s,
OpeningHours = %s
        WHERE GalleryID = %s
        """
    self.cursor.execute(query, (
        gallery.get_name(),
        gallery.get_description(),
        gallery.get_location(),
        gallery.get_curator(),
        gallery.get_opening_hours(),
        gallery.get_gallery_id()
    ))
    self.conn.commit()
    print(f"\n□ Gallery ID {gallery.get_gallery_id()} has been successfully
updated!\n")

    # Display the updated gallery
    self.cursor.execute("SELECT * FROM gallery WHERE GalleryID = %s",
(gallery.get_gallery_id(),))
    updated = self.cursor.fetchone()

    if updated:
        print("□ Updated Gallery Details:")
        print("=" * 50)
        print(f"Gallery ID    : {updated[0]}")
        print(f"Name          : {updated[1]}")
        print(f"Description   : {updated[2]}")
        print(f"Location     : {updated[3]}")
        print(f"Curator ID   : {updated[4]}")
        print(f"Opening Hours : {updated[5]}")
        print("=" * 50)

    except GalleryNotFoundException as e:
        print(f"□ {e}")
    except pymysql.Error as e:
        print(f"□ Database Error: {e}")

```



```

11. view all Galleries
12. Create Gallery
13. Update Gallery
14. Search Gallery By Name
15. Search by ID
16. Remove gallery
17. Gallery Artist Impact Report
18. Get Artworks From Gallery
19. Get Artworks From Gallery
20. Top Exhibited Artworks
21. Back to main menu
Enter Your Choice:13

✳ Update Gallery
Enter Gallery ID to update: 65
Enter New Name: hexa4
Enter New Description: learning
Enter New Location: banglore
Enter New Curator ID: 103
Enter New Opening Hours: 5 pm to 6 pm

✅ Gallery ID 65 has been successfully updated!

🔍 Updated Gallery Details:
=====
Gallery ID      : 65
Name            : hexa4
Description     : learning
Location        : banglore
Curator ID     : 103
Opening Hours   : 5 pm to 6 pm
=====

```

```

def search_gallery_by_name(self, name_keyword):
    try:
        query = "SELECT * FROM gallery WHERE Name LIKE %s"
        self.cursor.execute(query, ('%' + name_keyword + '%',))
        results = self.cursor.fetchall()

        if not results:
            raise GalleryNotFoundException(f"No galleries found matching name:
'{name_keyword}'.")

        print(f"\n❑ Search Results for '{name_keyword}':")
        print("=" * 60)

        for result in results:
            # Create Gallery object
            gallery = Gallery(
                result[0], # GalleryID

```

```

        result[1], # Name
        result[2], # Description
        result[3], # Location
        result[4], # Curator
        result[5] # OpeningHours
    )

    print(f"□ Gallery ID    : {gallery.get_gallery_id()}")
    print(f"□ Name          : {gallery.get_name()}")
    print(f"□ Description      : {gallery.get_description()}")
    print(f"□ Location         : {gallery.get_location()}")
    print(f"□ Curator ID       : {gallery.get_curator()}")
    print(f"□ Opening Hours    : {gallery.get_opening_hours()}")
    print("-" * 60)

except GalleryNotFoundException as e:
    print(f"□ {e}")
except pymysql.Error as e:
    print(f"□ Database Error: {e}")

```

```

11. view all Galleries
12. Create Gallery
13. Update Gallery
14. Search Gallery By Name
15. Search by ID
16. Remove gallery
17. Gallery Artist Impact Report
18. Get Artworks From Gallery
19. Get Artworks From Gallery
20. Top Exhibited Artworks
21. Back to main menu
Enter Your Choice:14
Enter gallery name keyword to search: hexa

```

```

🔍 Search Results for 'hexa':

```

```

=====
🏛 Gallery ID      : 65
🎨 Name           : hexa4
📄 Description    : learning
📍 Location       : banglore
👤 Curator ID     : 103
🕒 Opening Hours  : 5 pm to 6 pm
=====

```

```

11. view all Galleries

```

```

def view_all_galleries(self):
    try:
        query = "SELECT * FROM gallery"
        self.cursor.execute(query)
        results = self.cursor.fetchall()

        if not results:
            raise GalleryNotFoundException("No galleries found in the database.")

        print("\n□ All Galleries:")
        print("=" * 60)

        for result in results:
            gallery = Gallery(
                result[0], # GalleryID
                result[1], # Name
                result[2], # Description
                result[3], # Location
                result[4], # Curator
                result[5] # OpeningHours
            )

            print(f"□ Gallery ID    : {gallery.get_gallery_id()}")
            print(f"□ Name          : {gallery.get_name()}")
            print(f"□ Description   : {gallery.get_description()}")
            print(f"□ Location     : {gallery.get_location()}")
            print(f"□ Curator ID    : {gallery.get_curator()}")
            print(f"□ Opening Hours : {gallery.get_opening_hours()}")
            print("-" * 60)

    except GalleryNotFoundException as e:
        print(f"□ {e}")
    except pymysql.Error as e:
        print(f"□ Database Error: {e}")

```

```
👤 WELCOME TO THE VIRTUAL ART GALLERY 👤
=====
a. Artwork Management
b. Gallery Management
c. Exit

Enter your choice (a/b/c): b
11. view all Galleries
12. Create Gallery
13. Update Gallery
14. Search Gallery By Name
15. Search by ID
16. Remove gallery
17. Gallery Artist Impact Report
18. Get Artworks From Gallery
19. Get Artworks From Gallery
20. Top Exhibited Artworks
21. Back to main menu
Enter Your Choice:11

📁 Viewing All Galleries...

📁 All Galleries:
=====
🏠 Gallery ID : 10
👤 Name : art sphere
📄 Description : Art unites everyone together
📍 Location : india
👤 Curator ID : 104
🕒 Opening Hours : 9 am to 1 pm
=====
🏠 Gallery ID : 20
👤 Name : zumba art
📄 Description : experiencing each and every art brings peace
📍 Location : indonesia
👤 Curator ID : 106
🕒 Opening Hours : 8 am to 10 am
=====
🏠 Gallery ID : 30
👤 Name : album
📄 Description : explore
📍 Location : france
👤 Curator ID : 104
🕒 Opening Hours : 6 am to 11 pm
=====
🏠 Gallery ID : 40
👤 Name : wondering sculpture
📄 Description : mind blowing ideas and pictures
📍 Location : malaysia
👤 Curator ID : 101
🕒 Opening Hours : 10 am to 12 pm
=====
🏠 Gallery ID : 50
👤 Name : beauty of art
📄 Description : beauty lies in the way we design an art
📍 Location : indore
👤 Curator ID : 102
🕒 Opening Hours : 1 pm to 4 pm
=====
```

```
def search_gallery_by_id(self, gallery_id):
    try:
        query = "SELECT * FROM gallery WHERE GalleryID = %s"
        self.cursor.execute(query, (gallery_id,))
```

```

result = self.cursor.fetchone()

if not result:
    raise GalleryNotFoundException(f"Gallery with ID {gallery_id} not
found.")

# Create Gallery object
gallery = Gallery(
    result[0], # GalleryID
    result[1], # Name
    result[2], # Description
    result[3], # Location
    result[4], # Curator
    result[5] # OpeningHours
)

print("\n□ Gallery Details:")
print("=" * 60)
print(f"□ Gallery ID    : {gallery.get_gallery_id()}")
print(f"□ Name          : {gallery.get_name()}")
print(f"□ Description     : {gallery.get_description()}")
print(f"□ Location       : {gallery.get_location()}")
print(f"□ Curator ID      : {gallery.get_curator()}")
print(f"□ Opening Hours   : {gallery.get_opening_hours()}")
print("-" * 60)

except GalleryNotFoundException as e:
    print(f"□ {e}")
except pymysql.Error as e:
    print(f"□ Database Error: {e}")

```

```

11. view all Galleries
12. Create Gallery
13. Update Gallery
14. Search Gallery By Name
15. Search by ID
16. Remove gallery
17. Gallery Artist Impact Report
18. Get Artworks From Gallery
19. Get Artworks From Gallery
20. Top Exhibited Artworks
21. Back to main menu
Enter Your Choice:15
Enter Gallery ID to search: 40

```

#### Gallery Details:

```

=====
🏛️ Gallery ID      : 40
🎨 Name           : wondering sculpture
🖼️ Description    : mind blowing ideas and pictures
📍 Location       : malaysia
👤 Curator ID     : 101
🕒 Opening Hours  : 10 am to 12 pm
=====

```

```

def remove_gallery(self, gallery_id):
    try:
        # Check if gallery exists
        self.cursor.execute("SELECT * FROM gallery WHERE GalleryID = %s",
(gallery_id,))
        result = self.cursor.fetchone()
        if not result:
            raise GalleryNotFoundException(f"Gallery with ID {gallery_id} not
found.")

        # Delete gallery
        query = "DELETE FROM gallery WHERE GalleryID = %s"
        self.cursor.execute(query, (gallery_id,))
        self.conn.commit()

        print(f"📦 Gallery with ID {gallery_id} has been successfully removed.\n")

    except GalleryNotFoundException as e:
        print(f"📦 {e}")
    except pymysql.Error as e:
        print(f"📦 Database Error while removing gallery: {e}")

```

```

11. view all Galleries
12. Create Gallery
13. Update Gallery
14. Search Gallery By Name
15. Search by ID
16. Remove gallery
17. Gallery Artist Impact Report
18. Get Artworks From Gallery
19. Get Artworks From Gallery
20. Top Exhibited Artworks
21. Back to main menu
Enter Your Choice:16
Enter Gallery ID to remove: 64

```

❌ Gallery with ID 64 not found.

```

11. view all Galleries
12. Create Gallery
13. Update Gallery
14. Search Gallery By Name
15. Search by ID
16. Remove gallery
17. Gallery Artist Impact Report
18. Get Artworks From Gallery
19. Get Artworks From Gallery
20. Top Exhibited Artworks
21. Back to main menu
Enter Your Choice:16
Enter Gallery ID to remove: 65

```

✅ Gallery with ID 65 has been successfully removed.

```

def gallery_artist_impact_report(self):
    try:
        query = """
            SELECT ar.ArtistID, ar.Name, COUNT(g.GalleryID) AS
TotalGalleriesCurated
            FROM artist ar
            LEFT JOIN gallery g ON ar.ArtistID = g.Curator
            GROUP BY ar.ArtistID, ar.Name
            ORDER BY TotalGalleriesCurated DESC
        """
        self.cursor.execute(query)
        rows = self.cursor.fetchall()

        print("\n❑ Gallery Artist Impact Report ❑")
        print("-" * 40)
        for row in rows:
            print(f"Artist ID   : {row[0]}")
            print(f"Name       : {row[1]}")
            print(f"Galleries  : {row[2]}")
            print("-" * 40)

    except pymysql.Error as e:
        print(f"❑ Error generating report: {e}")

```

```

11. view all Galleries
12. Create Gallery
13. Update Gallery
14. Search Gallery By Name
15. Search by ID
16. Remove gallery
17. Gallery Artist Impact Report
18. Get Artworks From Gallery
19. Get Artworks From Gallery
20. Top Exhibited Artworks
21. Back to main menu
Enter Your Choice:17

```

```

🔴 Gallery Artist Impact Report 🔴
-----
Artist ID   : 104
Name        : Taylor Morgan
Galleries   : 3
-----
Artist ID   : 101
Name        : Arjun Mehta
Galleries   : 1
-----
Artist ID   : 102
Name        : Alex Rivera
Galleries   : 1
-----
Artist ID   : 105
Name        : Rohan Patel
Galleries   : 1
-----
Artist ID   : 106
Name        : Jordan Lee
Galleries   : 1
-----
Artist ID   : 103
Name        : Ishita Sharma
Galleries   : 0
-----

```

```

def get_artworks_in_gallery(self, gallery_id):
    try:
        query = """
            SELECT a.ArtworkID, a.Title, a.Description, a.CreationDate, a.Medium,
            a.ImageURL
            FROM artwork a
            JOIN artwork_gallery ag ON a.ArtworkID = ag.ArtworkID
            WHERE ag.GalleryID = %s
            """
        self.cursor.execute(query, (gallery_id,))
        rows = self.cursor.fetchall()

        if not rows:
            raise ArtworkNotFoundException(f"❑ No artworks found in Gallery ID {gallery_id}.")

        print(f"\n❑ Artworks in Gallery ID {gallery_id}:")
        for row in rows:
            artwork = Artwork(

```



```

        artwork_id=row[0],
        title=row[1],
        description=row[2],
        creation_date=row[3],
        medium=row[4],
        image_url=row[5]
    )

    print(f"Artwork ID: {artwork.get_artwork_id()}")
    print(f"Title: {artwork.get_title()}")
    print(f"Description: {artwork.get_description()}")
    print(f"Created On: {artwork.get_creation_date()}")
    print(f"Medium: {artwork.get_medium()}")
    print(f"Image URL: {artwork.get_image_url()}")
    print("-" * 30)

except ArtworkNotFoundException as e:
    print(e)

except pymysql.Error as e:
    print(f"❌ Error fetching artworks: {e}")

```

```

-----
11. view all Galleries
12. Create Gallery
13. Update Gallery
14. Search Gallery By Name
15. Search by ID
16. Remove gallery
17. Gallery Artist Impact Report
18. Get Artworks From Gallery
19. Get Artworks From Gallery
20. Top Exhibited Artworks
21. Back to main menu
Enter Your Choice:18

🎨 View Artworks in a Gallery
Enter Gallery ID: 50

🖼️ Artworks in Gallery ID 50:
Artwork ID: 3
Title: DIGITAL MUSEUM
Description: A virtual art gallery within a digital museum, sh
Created On: 2021-05-10
Medium: Multimedia Installation
Image URL: https://www.invaluable.com/blog/wp-content/uploads/
-----

```

```

def get_galleries_for_artwork(self, artwork_id):
    try:
        query = """
            SELECT g.GalleryID, g.Name, g.Description, g.Location, g.Curator,
g.OpeningHours
            FROM gallery g
            JOIN artwork_gallery ag ON g.GalleryID = ag.GalleryID
            WHERE ag.ArtworkID = %s
        """
        self.cursor.execute(query, (artwork_id,))
        rows = self.cursor.fetchall()

        if not rows:
            raise GalleryNotFoundException(f"□ No galleries found for Artwork ID {artwork_id}.")

        print(f"\n□ Galleries for Artwork ID {artwork_id}:")
        for row in rows:
            gallery = Gallery(*row)
            print(f"Gallery ID: {gallery.get_gallery_id()}")
            print(f"Name: {gallery.get_name()}")
            print(f"Location: {gallery.get_location()}")
            print(f"Curator ID: {gallery.get_curator()}")
            print(f"Opening Hours: {gallery.get_opening_hours()}")
            print("-" * 30)

    except GalleryNotFoundException as e:
        print(e)

    except pymysql.Error as e:
        print(f"□ Database error: {e}")

```

```

-----
11. view all Galleries
12. Create Gallery
13. Update Gallery
14. Search Gallery By Name
15. Search by ID
16. Remove gallery
17. Gallery Artist Impact Report
18. Get Artworks From Gallery
19. Get Artworks From Gallery
20. Top Exhibited Artworks
21. Back to main menu
Enter Your Choice:19

🏛️View Galleries Displaying an Artwork
Enter Artwork ID: 6

📖 Galleries for Artwork ID 6:
Gallery ID: 60
Name: epics of art
Location: bhutan
Curator ID: 104
Opening Hours: 2 pm to 4 pm
-----

```

```

def get_top_exhibited_artworks(self, top_n):
    try:
        query = """
            SELECT a.ArtworkID, a.Title, COUNT(ag.GalleryID) as gallery_count
            FROM artwork a
            JOIN artwork_gallery ag ON a.ArtworkID = ag.ArtworkID
            GROUP BY a.ArtworkID, a.Title
            ORDER BY gallery_count DESC
            LIMIT %s
        """
        self.cursor.execute(query, (top_n,))
        rows = self.cursor.fetchall()

        if not rows:
            raise ArtworkNotFoundException("❑ No artwork exhibition data available.")

        print(f"\n❑ Top {top_n} Most Exhibited Artworks:")
        for row in rows:
            print(f"Artwork ID: {row[0]}, Title: {row[1]}, Exhibited in {row[2]} galleries")
    
```

```
except ArtworkNotFoundException as e:  
    print(e)
```

```
except pymysql.Error as e:  
    print(f"❌ Database error: {e}")
```

```
11. view all Galleries  
12. Create Gallery  
13. Update Gallery  
14. Search Gallery By Name  
15. Search by ID  
16. Remove gallery  
17. Gallery Artist Impact Report  
18. Get Artworks From Gallery  
19. Get Artworks From Gallery  
20. Top Exhibited Artworks  
21. Back to main menu  
Enter Your Choice:20  
  
🖼️ View Top Exhibited Artworks  
Enter how many top artworks you want to view: 3  
  
🏆 Top 3 Most Exhibited Artworks:  
Artwork ID: 1, Title: Emperia Images, Exhibited in 2 galleries  
Artwork ID: 5, Title: Time meets Virtual Art, Exhibited in 2 galleries  
Artwork ID: 3, Title: DIGITAL MUSEUM, Exhibited in 2 galleries
```

## main.py:

```
from util.db_utils import DBConnection  
from dao.implementation import VirtualArtGalleryDAO  
from entity.artwork import Artwork  
from entity.gallery import Gallery  
from entity.userfavoriteartwork import UserFavoriteArtwork  
  
def main():  
    conn = DBConnection.connect()  
    dao = VirtualArtGalleryDAO(conn)  
  
    while True:  
  
        print("\t")  
        print("="*50)  
        print("🎨 WELCOME TO THE VIRTUAL ART GALLERY 🎨".center(50))  
        print("="*50)
```

```

print("      a. Artwork Management")
print("      b. Gallery Management")
print("      c. Exit")

main_choice = input("\nEnter your choice (a/b/c): ").lower()

if main_choice == 'a':

    while True:
        print("\nVirtual Art Gallery")
        print("1. Create Account for artist") #
Moved to the top
        print("2. View Artworks")
        print("3. Add Artwork")
        print("4. Update Artwork")
        print("5. Remove Artwork")
        print("6. View Your Artwork with ID")
        print("7. Search Artworks")
        print("8. Add Artwork to Favorites")
        print("9. Remove Artwork from Favorites")
        print("10. View User's Favorite Artworks")
        print("11. Back to main menu")

        choice = input("Enter your choice: ")

        if choice == '1': # Add artist functionality
            name = input("Name: ")
            biography = input("Biography: ")
            birth_date = input("Birth Date (YYYY-MM-DD): ")
            nationality = input("Nationality: ")
            website = input("Website: ")
            contact_info = input("Contact Information: ")
            dao.add_artist(name, biography, birth_date, nationality,
website, contact_info)

        if choice == '2':
            dao.view_artworks()

        elif choice == '3':
            title = input("Title: ")
            description = input("Description: ")
            medium = input("Medium: ")
            image_url = input("Image URL: ")
            dao.add_artwork(title, description, medium, image_url)

```

```

elif choice == '4':
    print("\n☞ Update Artwork")
    artwork_id = int(input("Enter Artwork ID to update: "))
    title = input("Title: ")
    description = input("Description: ")
    medium = input("Medium: ")
    image_url = input("Image URL: ")

    artwork = Artwork(
        artwork_id=artwork_id,
        title=title,
        description=description,
        creation_date=None, # or pass a dummy value since
it's auto-updated
        medium=medium,
        image_url=image_url
    )

    dao.update_artwork(artwork)

elif choice == '5':
    identifier = input("Enter Artwork ID or Title to remove: ")

    # Convert to int if possible (assume it's an ID)
    if identifier.isdigit():
        identifier = int(identifier)

    dao.remove_artwork(identifier)

elif choice == '6':
    artwork_id = int(input("Enter Artwork ID to view: "))
    dao.get_artwork_by_id(artwork_id)

elif choice == '7':
    keyword = input("Enter keyword to search in artwork
titles: ")

    dao.search_artworks(keyword)

elif choice == '8':
    user_id = int(input("Enter User ID: "))
    artwork_id = int(input("Enter Artwork ID: "))

```

```

        favorite = UserFavoriteArtwork()
        favorite.set_user_id(user_id)
        favorite.set_artwork_id(artwork_id)

        dao.add_artwork_to_favorite(favorite)

    elif choice == '9':
        artwork_id = int(input("Enter Artwork ID to remove from
favorites: "))

        print() # One-line gap

        favorite = UserFavoriteArtwork()
        favorite.set_artwork_id(artwork_id)

        dao.remove_artwork_from_favorite(favorite)

    elif choice == '10':
        user_id = int(input("Enter User ID to view favorite
artworks: "))

        print() # Adds a one-line gap

        dao.get_user_favorite_artworks(user_id)

    elif choice == '11':
        break

    else:
        print("valid option")

elif main_choice == 'b':
    while True:
        print("11. view all Galleries")
        print("12. Create Gallery")
        print("13. Update Gallery")
        print("14. Search Gallery By Name")
        print("15. Search by ID ")
        print("16. Remove gallery")
        print("17. Gallery Artist Impact Report")
        print("18. Get Artworks From Gallery")
        print("19. Get Artworks From Gallery")
        print("20. Top Exhibited Artworks")
        print("21. Back to main menu")

        choice=input("Enter Your Choice:")

```

```

        if choice == '11':
            print("\n👁 Viewing All Galleries...\n")
            dao.view_all_galleries()

        elif choice == '12':
            name = input("Enter Gallery Name: ")
            description = input("Enter Description: ")
            location = input("Enter Location: ")
            curator = input("Enter Curator ID (leave blank if none): ")

            opening_hours = input("Enter Opening Hours: ")
            print()

            curator = int(curator) if curator.strip() else None

            # Pass gallery_id=None explicitly since it's auto-
incremented

            gallery = Gallery(
                gallery_id=None,
                name=name,
                description=description,
                location=location,
                curator=curator,
                opening_hours=opening_hours
            )

            dao.add_gallery(gallery)

        elif choice == '13':
            print("\n🔄 Update Gallery")
            gallery_id = int(input("Enter Gallery ID to update: "))
            name = input("Enter New Name: ")
            description = input("Enter New Description: ")
            location = input("Enter New Location: ")
            curator = input("Enter New Curator ID: ")
            opening_hours = input("Enter New Opening Hours: ")
            print()

            updated_gallery = Gallery(gallery_id, name, description,
location, curator, opening_hours)
            dao.update_gallery(updated_gallery)

```



```

        elif choice == '14':
            keyword = input("Enter gallery name keyword to search:
").strip()

            print()
            dao.search_gallery_by_name(keyword)

        elif choice == '15':
            gallery_id = input("Enter Gallery ID to search:
").strip()

            print()
            dao.search_gallery_by_id(gallery_id)

        elif choice == '16':
            gallery_id = input("Enter Gallery ID to remove:
").strip()

            print()
            dao.remove_gallery(gallery_id)

        elif choice == '17':
            print()
            dao.gallery_artist_impact_report()

        elif choice == '18':
            print("\n👁 View Artworks in a Gallery")
            gallery_id = int(input("Enter Gallery ID: "))
            print()
            dao.get_artworks_in_gallery(gallery_id)

        elif choice == '19':
            print("\n👁 View Galleries Displaying an Artwork")
            artwork_id = int(input("Enter Artwork ID: "))
            print()
            dao.get_galleries_for_artwork(artwork_id)

        elif choice == '20':
            print("\n👁 View Top Exhibited Artworks")
            top_n = int(input("Enter how many top artworks you want
to view: "))

            print()
            dao.get_top_exhibited_artworks(top_n)

```

```

        elif choice == '21':
            break

        else:
            print("Enter valid option")

    elif main_choice == 'c':
        break

    else:
        print("Invalid Option")

conn.close()

if __name__ == "__main__":
    main()

```

### **DBUtil.py:**

```

import pymysql
import sys

class DBConnection:
    @staticmethod
    def connect():
        try:
            return pymysql.connect(
                user="root",
                password="root",
                host="127.0.0.1",
                database="virtualartgallery"
            )
        except pymysql.Error as e:
            print(f"Database Connection Error: {e}")
            sys.exit(1)

```

## Output:

```
=====
🎨 WELCOME TO THE VIRTUAL ART GALLERY 🎨
=====

a. Artwork Management
b. Gallery Management
c. Exit

Enter your choice (a/b/c): |
```

```
=====
🎨 WELCOME TO THE VIRTUAL ART GALLERY 🎨
=====

a. Artwork Management
b. Gallery Management
c. Exit

Enter your choice (a/b/c): a

Virtual Art Gallery
1. Create Account for artist
2. View Artworks
3. Add Artwork
4. Update Artwork
5. Remove Artwork
6. View Your Artwork with ID
7. Search Artworks
8. Add Artwork to Favorites
9. Remove Artwork from Favorites
10. View User's Favorite Artworks
11. Back to main menu
Enter your choice: |
```

```
=====
🎨 WELCOME TO THE VIRTUAL ART GALLERY 🎨
=====

a. Artwork Management
b. Gallery Management
c. Exit

Enter your choice (a/b/c): b
11. view all Galleries
12. Create Gallery
13. Update Gallery
14. Search Gallery By Name
15. Search by ID
16. Remove gallery
17. Gallery Artist Impact Report
18. Get Artworks From Gallery
19. Get Artworks From Gallery
20. Top Exhibited Artworks
21. Back to main menu
Enter Your Choice:|
```

## 9.UNIT TESTING

### Test\_artwork.py:

```
import pytest
from unittest.mock import MagicMock
from service.artwork_manager import ArtworkManager
from entity.artwork import Artwork
from exception.exceptions import ArtworkNotFoundException

@pytest.fixture
def mock_db():
    mock_conn = MagicMock()
    mock_cursor = MagicMock()
    mock_conn.cursor.return_value = mock_cursor
    return mock_conn, mock_cursor

def test_add_artwork(mock_db):
    mock_conn, mock_cursor = mock_db
```

```

manager = ArtworkManager(mock_conn)

# Setup mock behavior
mock_cursor.lastrowid = 1

result = manager.add_artwork(
    title='Mona Lisa',
    description='A classic painting',
    creation_date='2024-04-09',
    medium='Oil',
    image_url='http://image.com/mona.jpg'
)

assert isinstance(result, Artwork)
assert result.get_title() == 'Mona Lisa'
mock_cursor.execute.assert_called_once()
mock_conn.commit.assert_called_once()

def test_update_artwork_success(mock_db):
    mock_conn, mock_cursor = mock_db
    manager = ArtworkManager(mock_conn)

    mock_cursor.rowcount = 1

    artwork = Artwork(
        artwork_id=1,
        title='Updated',
        description='Updated Desc',
        creation_date='2024-04-09',
        medium='Acrylic',
        image_url='http://image.com/updated.jpg'
    )

    manager.update_artwork(artwork)

    assert mock_cursor.execute.called
    assert mock_conn.commit.called

def test_update_artwork_not_found(mock_db):
    mock_conn, mock_cursor = mock_db

```

```

manager = ArtworkManager(mock_conn)

mock_cursor.rowcount = 0

artwork = Artwork(
    artwork_id=99,
    title='Missing',
    description='Not there',
    creation_date='2024-01-01',
    medium='Digital',
    image_url='http://image.com/missing.jpg'
)

with pytest.raises(ArtworkNotFoundException):
    manager.update_artwork(artwork)

def test_remove_artwork_success(mock_db):
    mock_conn, mock_cursor = mock_db
    manager = ArtworkManager(mock_conn)

    mock_cursor.rowcount = 1

    manager.remove_artwork(1)

    mock_cursor.execute.assert_called_once()
    mock_conn.commit.assert_called_once()

def test_remove_artwork_not_found(mock_db):
    mock_conn, mock_cursor = mock_db
    manager = ArtworkManager(mock_conn)

    mock_cursor.rowcount = 0

    with pytest.raises(ArtworkNotFoundException):
        manager.remove_artwork(999)

def test_get_artwork_by_id_success(mock_db):
    mock_conn, mock_cursor = mock_db
    manager = ArtworkManager(mock_conn)

```

```

mock_cursor.fetchone.return_value = (
    1, 'Starry Night', 'Night sky',
    '2024-04-09', 'Oil',
    'http://image.com/starry.jpg'
)

artwork = manager.get_artwork_by_id(1)

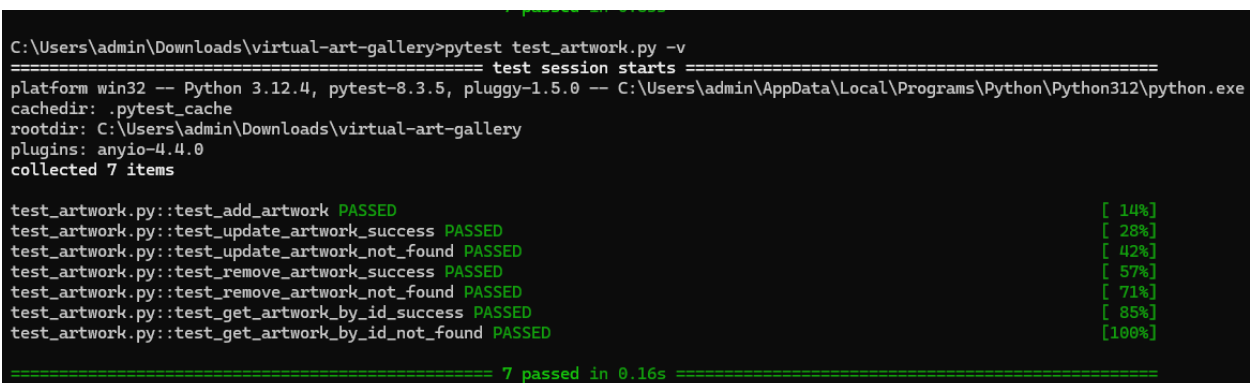
assert isinstance(artwork, Artwork)
assert artwork.get_title() == 'Starry Night'
assert artwork.get_medium() == 'Oil'

def test_get_artwork_by_id_not_found(mock_db):
    mock_conn, mock_cursor = mock_db
    manager = ArtworkManager(mock_conn)

    mock_cursor.fetchone.return_value = None

    with pytest.raises(ArtworkNotFoundException):
        manager.get_artwork_by_id(999)

```



```

C:\Users\admin\Downloads\virtual-art-gallery>pytest test_artwork.py -v
===== test session starts =====
platform win32 -- Python 3.12.4, pytest-8.3.5, pluggy-1.5.0 -- C:\Users\admin\AppData\Local\Programs\Python\Python312\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\admin\Downloads\virtual-art-gallery
plugins: anyio-4.4.0
collected 7 items

test_artwork.py::test_add_artwork PASSED [ 14%]
test_artwork.py::test_update_artwork_success PASSED [ 28%]
test_artwork.py::test_update_artwork_not_found PASSED [ 42%]
test_artwork.py::test_remove_artwork_success PASSED [ 57%]
test_artwork.py::test_remove_artwork_not_found PASSED [ 71%]
test_artwork.py::test_get_artwork_by_id_success PASSED [ 85%]
test_artwork.py::test_get_artwork_by_id_not_found PASSED [100%]

===== 7 passed in 0.16s =====

```

## test\_gallery.py

```

import pytest
from unittest.mock import MagicMock
from entity.gallery import Gallery
from service.gallery_manager import GalleryManager
from exception.exceptions import GalleryNotFoundException,
GalleryNotAddedException

```

```

@pytest.fixture
def mock_db():
    mock_conn = MagicMock()
    mock_cursor = MagicMock()
    mock_conn.cursor.return_value = mock_cursor
    return mock_conn, mock_cursor

def test_add_gallery_success(mock_db):
    mock_conn, mock_cursor = mock_db
    manager = GalleryManager(mock_conn)
    gallery = Gallery(1, "Modern Art House", "Contemporary collections", "NYC",
"Curator01", "10AM-5PM")

    manager.add_gallery(gallery)

    assert mock_cursor.execute.called
    assert mock_conn.commit.called

def test_update_gallery_success(mock_db):
    mock_conn, mock_cursor = mock_db
    manager = GalleryManager(mock_conn)
    gallery = Gallery(1, "Updated Gallery", "Updated desc", "Paris", "CuratorX",
"9AM-6PM")

    mock_cursor.fetchone.return_value = (1,)
    manager.update_gallery(gallery)

    assert mock_cursor.execute.call_count >= 2
    assert mock_conn.commit.called

def test_update_gallery_not_found(mock_db):
    mock_conn, mock_cursor = mock_db
    manager = GalleryManager(mock_conn)
    gallery = Gallery(99, "Ghost Gallery", "Nowhere", "Hidden", "NoOne",
"Never")

    mock_cursor.fetchone.return_value = None
    with pytest.raises(GalleryNotFoundException):
        manager.update_gallery(gallery)

```



```

def test_remove_gallery_success(mock_db):
    mock_conn, mock_cursor = mock_db
    manager = GalleryManager(mock_conn)

    mock_cursor.fetchone.return_value = ("some gallery",)
    manager.remove_gallery(1)

    assert mock_cursor.execute.call_count >= 2
    assert mock_conn.commit.called

def test_remove_gallery_not_found(mock_db):
    mock_conn, mock_cursor = mock_db
    manager = GalleryManager(mock_conn)

    mock_cursor.fetchone.return_value = None
    with pytest.raises(GalleryNotFoundException):
        manager.remove_gallery(404)

def test_search_gallery_by_name_success(mock_db):
    mock_conn, mock_cursor = mock_db
    manager = GalleryManager(mock_conn)

    mock_cursor.fetchall.return_value = [(1, "Gallery A", "Desc", "Loc", "Curator",
"9-5")]
    results = manager.search_gallery_by_name("Gallery")

    assert results[0][1] == "Gallery A"

def test_search_gallery_by_name_not_found(mock_db):
    mock_conn, mock_cursor = mock_db
    manager = GalleryManager(mock_conn)

    mock_cursor.fetchall.return_value = []
    with pytest.raises(GalleryNotFoundException):
        manager.search_gallery_by_name("DoesNotExist")

```

```

C:\Users\admin\Downloads\virtual-art-gallery>pytest test_gallery.py -v
===== test session starts =====
platform win32 -- Python 3.12.4, pytest-8.3.5, pluggy-1.5.0 -- C:\Users\admin\AppData\Local\Programs\Python\Python312\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\admin\Downloads\virtual-art-gallery
plugins: anyio-4.4.0
collected 7 items

test_gallery.py::test_add_gallery_success PASSED [ 14%]
test_gallery.py::test_update_gallery_success PASSED [ 28%]
test_gallery.py::test_update_gallery_not_found PASSED [ 42%]
test_gallery.py::test_remove_gallery_success PASSED [ 57%]
test_gallery.py::test_remove_gallery_not_found PASSED [ 71%]
test_gallery.py::test_search_gallery_by_name_success PASSED [ 85%]
test_gallery.py::test_search_gallery_by_name_not_found PASSED [100%]

===== 7 passed in 0.21s =====

```

## 10.FUTURE ENHANCEMENT

The Virtual Art Gallery project, developed using Python and MySQL, can be further enhanced to provide a richer and more interactive experience for its users. One major enhancement would be integrating a web-based interface using a Python web framework like Flask or Django, allowing users to browse artworks, view artist profiles, and manage their personal galleries through a visually appealing and user-friendly platform. Additional features like 3D virtual tours and artwork search filters based on artist, medium, or style can make navigation more intuitive. Enabling users to register, log in securely, and curate their favorite collections using authentication methods like JWT or OAuth will also enhance personalization and security.

Another future improvement could include support for online artwork sales or auctions, with integration of secure payment systems. This would allow users to not just view but also purchase or bid on artworks, turning the gallery into a functional marketplace. Artists could be given the ability to upload their own works, manage portfolios, and view engagement statistics. Implementing community features like comments and ratings can encourage interaction, while admin dashboards can provide insights into user behavior and artwork popularity. Finally, ensuring robustness through unit testing, exception handling, and database integrity checks will make the system more reliable and scalable for future use.

## **11.CONCLUSION**

The Virtual Art Gallery Management System focuses on efficiently organizing and managing artists, artworks, galleries, and user preferences. The system is structured with separate layers for entities, data access, exceptions, and utilities to keep the code clean and easy to maintain. Gallery-related operations such as adding, updating, searching, and reporting are handled in a dedicated module, ensuring smooth interaction between galleries, artworks, and artists. Custom error handling improves reliability, while a reusable database connection setup ensures consistent access to data. Overall, the system provides a solid backend foundation for managing digital art collections.