

Virtual Art Gallery - Final Project Report

Submitted To: Hexaware Technologies (Hexavarsity)

Trainer: Karthika Thangaraj

Team Members:

- Dhamini Machireddy
- Elekkiya

Date: June 2025

Abstract

This Virtual Art Gallery project is a console-based application developed using Python and MySQL to manage and display information about artworks, artists, users, and galleries. Instead of a graphical interface, the system uses simple text-based menus and commands to interact with users, making it easy to run on any computer without needing complex setups.

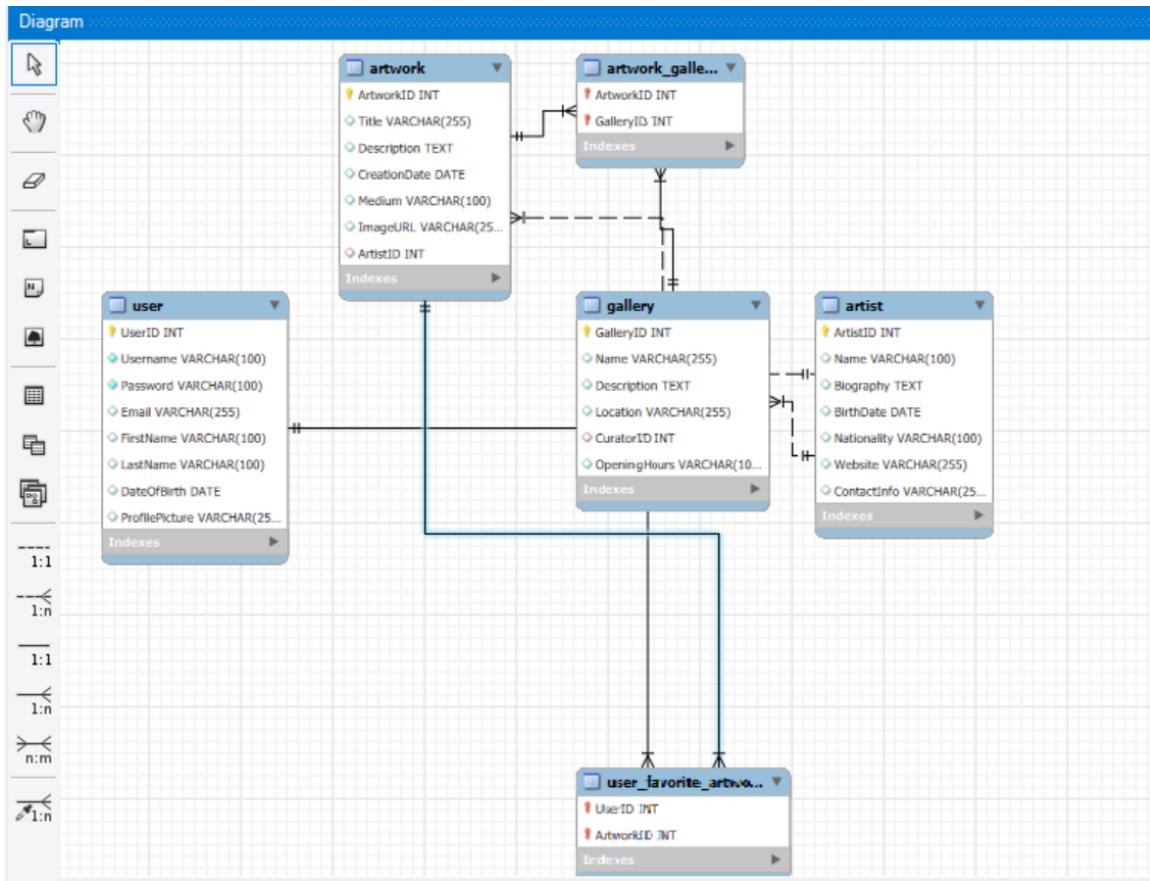
The project models key entities like Artwork, User, Artist, and Gallery using Python classes, which organize the data and actions related to each entity. The main functionalities include adding, viewing, and managing artwork details through console inputs, with all data stored securely in a MySQL database for easy retrieval and update.

By separating the program into different modules for database connection, business logic, and entity representation, the design remains clear and easy to maintain. Although the current version primarily focuses on artwork management, it lays a strong foundation to add more features for users, artists, and galleries in the future.

This console-based Virtual Art Gallery is a practical example of how programming and databases work together to build functional applications without a graphical interface. It allows users to manage and explore art collections through straightforward commands and outputs, providing a useful learning experience in database handling and object-oriented programming. Future improvements may include enhanced user interaction, data validation, and expanded features to better simulate a full art gallery system.

Entity-Relationship (ER) Diagram

Entities: Artist, Artwork, User, Gallery, User_Favorite_Artwork, Artwork_Gallery



Entity Descriptions

Artwork: ArtworkID, Title, Description, CreationDate, Medium, ImageURL, ArtistID

Artist: ArtistID, Name, Biography, BirthDate, Nationality, Website, ContactInfo

User: UserID, Username, Password, Email, FirstName, LastName, DateOfBirth, ProfilePicture

Gallery: GalleryID, Name, Description, Location, Curator, OpeningHours

User_Favorite_Artwork: UserID, ArtworkID (Composite PK)

Artwork_Gallery: ArtworkID, GalleryID (Composite PK)

Module Breakdown

entity/: Model classes only (no logic).

dao/: Interface and implementation for service methods.

exception/: Custom exceptions for user and artwork not found.

util/: DB connection and configuration utility.

main/: CLI menu interface.

Business Logic

- addArtwork(): Validates artist, inserts record.
- updateArtwork(): Validates ID, updates fields.
- removeArtwork(): Deletes record and references.
- getArtworkById(): Fetches by ID or raises an exception.
- addArtworkToFavorite(): Checks user/artwork, inserts into favorites.
- removeArtworkFromFavorite(): Removes entry.
- getUserFavoriteArtworks(): Joins tables and displays artworks.
- Exception handling is done using try...except in CLI.

Technologies Used

Python 3.11, MySQL, mysql-connector-python, VS Code, unittest

Creation of database schema :

```
CREATE DATABASE IF NOT EXISTS art_gallery;
```

```
USE art_gallery;
```

```
CREATE TABLE Artist (
```

```
    ArtistID INT AUTO_INCREMENT PRIMARY KEY,
```

```
    Name VARCHAR(100),
```

```
    Biography TEXT,
```

```
    BirthDate DATE,
```

```
Nationality VARCHAR(100),  
Website VARCHAR(255),  
ContactInfo VARCHAR(255)  
);  
  
CREATE TABLE Artwork (  
    ArtworkID INT AUTO_INCREMENT PRIMARY KEY,  
    Title VARCHAR(255),  
    Description TEXT,  
    CreationDate DATE,  
    Medium VARCHAR(100),  
    ImageURL VARCHAR(255),  
    ArtistID INT,  
    FOREIGN KEY (ArtistID) REFERENCES Artist(ArtistID)  
);  
  
USE art_gallery;  
  
CREATE TABLE Gallery (  
    GalleryID INT AUTO_INCREMENT PRIMARY KEY,  
    Name VARCHAR(255),  
    Description TEXT,  
    Location VARCHAR(255),  
    CuratorID INT,  
    OpeningHours VARCHAR(100),  
    FOREIGN KEY (CuratorID) REFERENCES Artist(ArtistID)  
);
```

```
CREATE TABLE `User` (  
    UserID INT AUTO_INCREMENT PRIMARY KEY,
```

```
Username VARCHAR(100) NOT NULL,  
Password VARCHAR(100) NOT NULL,  
Email VARCHAR(255),  
FirstName VARCHAR(100),  
LastName VARCHAR(100),  
DateOfBirth DATE,  
ProfilePicture VARCHAR(255)  
);  
  
CREATE TABLE User_Favorite_Artwork (  
    UserID INT,  
    ArtworkID INT,  
    PRIMARY KEY (UserID, ArtworkID),  
    FOREIGN KEY (UserID) REFERENCES `User`(UserID),  
    FOREIGN KEY (ArtworkID) REFERENCES Artwork(ArtworkID)  
);  
  
CREATE TABLE Artwork_Gallery (  
    ArtworkID INT,  
    GalleryID INT,  
    PRIMARY KEY (ArtworkID, GalleryID),  
    FOREIGN KEY (ArtworkID) REFERENCES Artwork(ArtworkID),  
    FOREIGN KEY (GalleryID) REFERENCES Gallery(GalleryID)  
);  
  
SET FOREIGN_KEY_CHECKS = 0;
```

```
TRUNCATE TABLE user_favorite_artwork;  
TRUNCATE TABLE artwork;  
TRUNCATE TABLE `user`;  
TRUNCATE TABLE artist;
```

```
SET FOREIGN_KEY_CHECKS = 1;

INSERT INTO artist (Name, Biography, BirthDate, Nationality, Website, ContactInfo) VALUES
('Vincent van Gogh', 'Dutch post-impressionist painter.', '1853-03-30', 'Dutch', 'https://vangogh.com', 'vangogh@art.com'),
('Pablo Picasso', 'Spanish painter and sculptor.', '1881-10-25', 'Spanish', 'https://picasso.com', 'picasso@art.com'),
('Leonardo da Vinci', 'Renaissance polymath.', '1452-04-15', 'Italian', 'https://davinci.com', 'davinci@art.com'),
('Frida Kahlo', 'Mexican self-portrait artist.', '1907-07-06', 'Mexican', 'https://kahlo.com', 'kahlo@art.com'),
('Claude Monet', 'French impressionist.', '1840-11-14', 'French', 'https://monet.com', 'monet@art.com'),
('Georgia O'Keeffe', 'Modernist artist.', '1887-11-15', 'American', 'https://okeeffe.com', 'okeeffe@art.com'),
('Salvador Dali', 'Surrealist painter.', '1904-05-11', 'Spanish', 'https://dali.com', 'dali@art.com'),
('Andy Warhol', 'Pop art pioneer.', '1928-08-06', 'American', 'https://warhol.com', 'warhol@art.com'),
('Rembrandt', 'Dutch Golden Age artist.', '1606-07-15', 'Dutch', 'https://rembrandt.com', 'rembrandt@art.com'),
('Michelangelo', 'Sculptor and painter.', '1475-03-06', 'Italian', 'https://michelangelo.com', 'mike@art.com');
```

```
INSERT INTO `user` (Username, Password, Email, FirstName, LastName, DateOfBirth, ProfilePicture)
VALUES
('anna01', 'pass123', 'anna01@mail.com', 'Anna', 'Smith', '1995-01-20', 'anna.jpg'),
('john_d', 'john@123', 'john.d@mail.com', 'John', 'Doe', '1992-03-15', 'john.jpg'),
('lisa_g', 'lisa456', 'lisa.g@mail.com', 'Lisa', 'Gray', '1988-07-12', 'lisa.jpg'),
('mike92', 'mikepass', 'mike92@mail.com', 'Mike', 'Brown', '1990-11-22', 'mike.jpg'),
('susanT', 'susan789', 'susanT@mail.com', 'Susan', 'Taylor', '1993-06-10', 'susan.jpg'),
('robert_b', 'rob123', 'robert@mail.com', 'Robert', 'Black', '1985-12-01', 'robert.jpg'),
('emmaK', 'emm@456', 'emma.k@mail.com', 'Emma', 'King', '1998-04-04', 'emma.jpg');
```

```
('alexH', 'alex321', 'alex.h@mail.com', 'Alex', 'Hunt', '1991-09-29', 'alex.jpg'),  
('ninaW', 'nina999', 'nina.w@mail.com', 'Nina', 'White', '1989-08-18', 'nina.jpg'),  
('rajK', 'rajk007', 'raj.k@mail.com', 'Raj', 'Kumar', '1996-02-25', 'raj.jpg');
```

```
INSERT INTO artwork (Title, Description, CreationDate, Medium, ImageURL, ArtistID) VALUES  
('Starry Night', 'A famous post-impressionist painting.', '1889-06-01', 'Oil on canvas', 'starry1.jpg', 1),  
('Starry Night', 'Iconic swirling sky.', '1889-06-02', 'Oil on canvas', 'starry2.jpg', 1),  
('Starry Night', 'Recognized worldwide.', '1889-06-03', 'Oil on canvas', 'starry3.jpg', 1),  
('Starry Night', 'Emotional masterpiece.', '1889-06-04', 'Oil on canvas', 'starry4.jpg', 1),  
('Starry Night', 'Van Gogh's greatest work.', '1889-06-05', 'Oil on canvas', 'starry5.jpg', 1),  
('Guernica', 'Powerful anti-war painting.', '1937-06-01', 'Oil on canvas', 'guernica.jpg', 2),  
('Mona Lisa', 'Portrait of Lisa Gherardini.', '1503-10-01', 'Oil on poplar', 'monalisa.jpg', 3),  
('The Two Fridas', 'Dual self-portrait.', '1939-01-01', 'Oil on canvas', 'fridas.jpg', 4),  
('Water Lilies', 'Impressionist water.', '1906-05-01', 'Oil on canvas', 'lilies.jpg', 5),  
('Black Iris III', 'Abstract flower painting.', '1926-04-01', 'Oil on canvas', 'blackiris.jpg', 6),  
('The Persistence of Memory', 'Melting clocks.', '1931-01-01', 'Oil on canvas', 'memory.jpg', 7),  
('Campbell\\'s Soup Cans', 'Pop art series.', '1962-01-01', 'Synthetic on canvas', 'soup.jpg', 8),  
('The Night Watch', 'Dutch militia.', '1642-01-01', 'Oil on canvas', 'nightwatch.jpg', 9),  
('The Creation of Adam', 'Part of Sistine Chapel.', '1512-01-01', 'Fresco', 'creation.jpg', 10),  
('Red Canna', 'OKeeffe floral work.', '1924-01-01', 'Oil on canvas', 'redcanna.jpg', 6);
```

```
INSERT INTO user_favorite_artwork (UserID, ArtworkID) VALUES  
(1, 1),  
(2, 6),  
(3, 7),  
(4, 8),  
(10, 14),  
(5, 9),
```

(6, 10),

(7, 11),

(8, 12),

(9, 13),

The screenshot shows a MySQL Workbench interface with a query editor containing the following SQL code:

```
1 ● CREATE DATABASE IF NOT EXISTS art_gallery;
2 ● USE art_gallery;
3 ● CREATE TABLE Artist (
4     ArtistID INT AUTO_INCREMENT PRIMARY KEY,
5     Name VARCHAR(100),
6     Biography TEXT,
7     BirthDate DATE,
8     Nationality VARCHAR(100),
9     Website VARCHAR(255),
10    ContactInfo VARCHAR(255)
11 );
12 ● CREATE TABLE Artwork (
13     ArtworkID INT AUTO_INCREMENT PRIMARY KEY,
14     Title VARCHAR(255),
15     Description TEXT,
16     CreationDate DATE,
17     Medium VARCHAR(100),
18     ImageURL VARCHAR(255),
19     ArtistID INT,
20     FOREIGN KEY (ArtistID) REFERENCES Artist(ArtistID)
21 );
22 ● USE art_gallery;
23
24 ● CREATE TABLE Gallery (
25     GalleryID INT AUTO_INCREMENT PRIMARY KEY,
26     Name VARCHAR(255),
27     Description TEXT,
28     Location VARCHAR(255),
29     CuratorID INT,
30     OpeningHours VARCHAR(100),
31     FOREIGN KEY (CuratorID) REFERENCES Artist(ArtistID)
32 );
```

The code creates a database named 'art_gallery' and uses it. It then defines three tables: 'Artist', 'Artwork', and 'Gallery'. The 'Artist' table has columns for ArtistID (primary key, auto-increment), Name, Biography, BirthDate, Nationality, Website, and ContactInfo. The 'Artwork' table has columns for ArtworkID (primary key, auto-increment), Title, Description, CreationDate, Medium, ImageURL, and ArtistID (with a foreign key reference to the Artist table). The 'Gallery' table has columns for GalleryID (primary key, auto-increment), Name, Description, Location, CuratorID, and OpeningHours (with a foreign key reference to the Artist table).

```

34 ● CREATE TABLE `User` (
35     UserID INT AUTO_INCREMENT PRIMARY KEY,
36     Username VARCHAR(100) NOT NULL,
37     Password VARCHAR(100) NOT NULL,
38     Email VARCHAR(255),
39     FirstName VARCHAR(100),
40     LastName VARCHAR(100),
41     DateOfBirth DATE,
42     ProfilePicture VARCHAR(255)
43 );
44 ● CREATE TABLE User_Favorite_Artwork (
45     UserID INT,
46     ArtworkID INT,
47     PRIMARY KEY (UserID, ArtworkID),
48     FOREIGN KEY (UserID) REFERENCES `User`(UserID),
49     FOREIGN KEY (ArtworkID) REFERENCES Artwork(ArtworkID)
50 );
51 ● CREATE TABLE Artwork_Gallery (
52     ArtworkID INT,
53     GalleryID INT,
54     PRIMARY KEY (ArtworkID, GalleryID),
55     FOREIGN KEY (ArtworkID) REFERENCES Artwork(ArtworkID),
56     FOREIGN KEY (GalleryID) REFERENCES Gallery(GalleryID)
57 );

```

```

66 ● INSERT INTO artist (Name, Biography, BirthDate, Nationality, Website, ContactInfo) VALUES
67     ('Vincent van Gogh', 'Dutch post-impressionist painter.', '1853-03-30', 'Dutch', 'https://vangogh.com', 'vangogh@art.com'),
68     ('Pablo Picasso', 'Spanish painter and sculptor.', '1881-10-25', 'Spanish', 'https://picasso.com', 'picasso@art.com'),
69     ('Leonardo da Vinci', 'Renaissance polymath.', '1452-04-15', 'Italian', 'https://davinci.com', 'davincig@art.com'),
70     ('Frida Kahlo', 'Mexican self-portrait artist.', '1907-07-06', 'Mexican', 'https://kahlo.com', 'kahlo@art.com'),
71     ('Claude Monet', 'French impressionist.', '1840-11-14', 'French', 'https://monet.com', 'monet@art.com'),
72     ('Georgia O'Keeffe', 'Modernist artist.', '1887-11-15', 'American', 'https://okeeffe.com', 'okeeffe@art.com'),
73     ('Salvador Dali', 'Surrealist painter.', '1904-05-11', 'Spanish', 'https://dali.com', 'dali@art.com'),
74     ('Andy Warhol', 'Pop art pioneer.', '1928-08-06', 'American', 'https://warhol.com', 'warhol@art.com'),
75     ('Rembrandt', 'Dutch Golden Age artist.', '1606-07-15', 'Dutch', 'https://rembrandt.com', 'rembrandt@art.com'),
76     ('Michelangelo', 'Sculptor and painter.', '1475-03-06', 'Italian', 'https://michelangelo.com', 'mike@art.com'),
77
78 ● INSERT INTO `user` (Username, Password, Email, FirstName, LastName, DateOfBirth, ProfilePicture) VALUES
79     ('anna01', 'pass123', 'anna01@mail.com', 'Anna', 'Smith', '1995-01-20', 'anna.jpg'),
80     ('john_d', 'john@123', 'john.d@mail.com', 'John', 'Doe', '1992-03-15', 'john.jpg'),
81     ('lisa_g', 'lisa.g@mail.com', 'Lisa', 'Gray', '1988-07-12', 'lisa.jpg'),
82     ('mike92', 'mikepass', 'mike92@mail.com', 'Mike', 'Brown', '1990-11-22', 'mike.jpg'),
83     ('susanT', 'susan789', 'susan@mail.com', 'Susan', 'Taylor', '1993-06-18', 'susan.jpg'),
84     ('robert_b', 'rob123', 'robert@mail.com', 'Robert', 'Black', '1985-12-01', 'robert.jpg'),
85     ('emmaK', 'emmp456', 'emma.k@mail.com', 'Emma', 'King', '1998-04-04', 'emma.jpg'),
86     ('alexH', 'alex321', 'alex.h@mail.com', 'Alex', 'Hunt', '1991-09-29', 'alex.jpg'),
87     ('ninaW', 'nina999', 'nina.w@mail.com', 'Nina', 'White', '1989-08-18', 'nina.jpg'),
88     ('rajK', 'rajk007', 'raj.k@mail.com', 'Raj', 'Kumar', '1996-02-25', 'raj.jpg');
89
90 ● INSERT INTO artwork (Title, Description, CreationDate, Medium, ImageURL, ArtistID) VALUES
91     ('Starry Night', 'A famous post-impressionist painting.', '1889-06-01', 'Oil on canvas', 'starry1.jpg', 1),
92     ('Starry Night', 'Iconic swirling sky.', '1889-06-02', 'Oil on canvas', 'starry2.jpg', 1),
93     ('Starry Night', 'Recognized worldwide.', '1889-06-03', 'Oil on canvas', 'starry3.jpg', 1),
94     ('Starry Night', 'Emotional masterpiece.', '1889-06-04', 'Oil on canvas', 'starry4.jpg', 1),
95     ('Starry Night', 'Van Gogh's greatest work.', '1889-06-05', 'Oil on canvas', 'starry5.jpg', 1),
96     ('Guernica', 'Powerful anti-war painting.', '1937-06-01', 'Oil on canvas', 'guernica.jpg', 2),
97     ('Mona Lisa', 'Portrait of Lisa Gherardini.', '1503-10-01', 'Oil on poplar', 'monalisa.jpg', 3),
98

```

```

107 •    INSERT INTO user_favorite_artwork (UserID, ArtworkID) VALUES
108      (1, 1),
109      (2, 6),
110      (3, 7),
111      (4, 8),
112      (10, 14),
113      (5, 9),
114      (6, 10),
115      (7, 11),
116      (8, 12),
117      (9, 13),
118
119

```

Creation of classes :

1. entity/ - Data Model Classes

Entity folder :

This package contains the data representation classes for the Virtual Art Gallery system. Each class corresponds to a real-world entity and matches a table in the MySQL database. These classes have only private variables, constructors, and getter/setter methods — no business logic.

Include Screenshots of:

- Artwork, Artist, User, Gallery, UserFavoriteArtwork, ArtworkGallery

Artwork :

class Artwork:

```

def __init__(self, artwork_id, title, description, creation_date, medium, image_url, artist_id):
    self.artwork_id = artwork_id
    self.title = title
    self.description = description
    self.creation_date = creation_date
    self.medium = medium
    self.image_url = image_url
    self.artist_id = artist_id

def __str__(self):
    return f"Artwork({self.artwork_id}, {self.title}, {self.medium}, by Artist {self.artist_id})"

```

```

1  class Artwork:
2      def __init__(self, artwork_id, title, description, creation_date, medium, image_url, artist_id):
3          self.artwork_id = artwork_id
4          self.title = title
5          self.description = description
6          self.creation_date = creation_date
7          self.medium = medium
8          self.image_url = image_url
9          self.artist_id = artist_id
10
11     def __str__(self):
12         return f"Artwork({self.artwork_id}, {self.title}, {self.medium}, by Artist {self.artist_id})"

```

Artist:

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

```

def __str__(self):

```
    return f"Artist({self.artist_id}, {self.name})"
```

```

entity > artist.py > Artist
1  class Artist:
2      def __init__(self, artist_id, name, biography, birth_date, nationality, website, contact_info):
3          self.artist_id = artist_id
4          self.name = name
5          self.biography = biography
6          self.birth_date = birth_date
7          self.nationality = nationality
8          self.website = website
9          self.contact_info = contact_info
10
11     def __str__(self):
12         return f"Artist({self.artist_id}, {self.name})"

```

User:

```
class User:

    def __init__(self, user_id, username, password, email, first_name, last_name, date_of_birth,
profile_picture):

        self.user_id = user_id

        self.username = username

        self.password = password

        self.email = email

        self.first_name = first_name

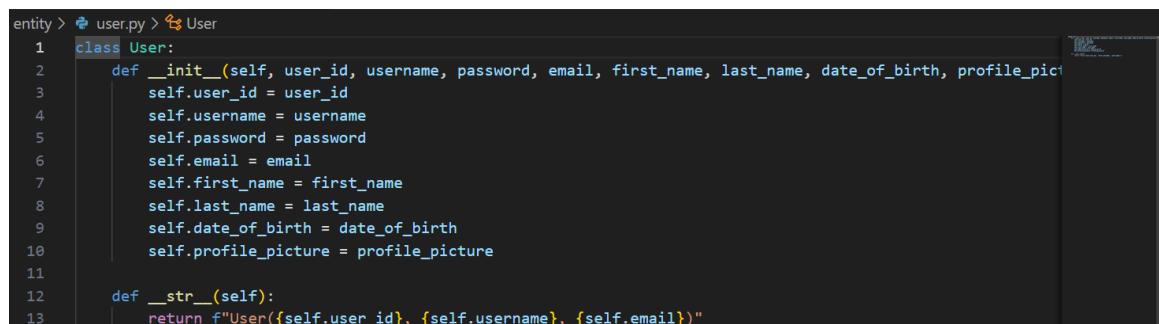
        self.last_name = last_name

        self.date_of_birth = date_of_birth

        self.profile_picture = profile_picture

def __str__(self):

    return f"User({self.user_id}, {self.username}, {self.email})"
```



The screenshot shows a code editor window with the file 'user.py' open. The code defines a class 'User' with an __init__ method that initializes various attributes: user_id, username, password, email, first_name, last_name, date_of_birth, and profile_picture. It also includes a __str__ method that returns a string representation of the user object using f-strings.

```
entity > user.py > User
1  class User:
2      def __init__(self, user_id, username, password, email, first_name, last_name, date_of_birth, profile_picture):
3          self.user_id = user_id
4          self.username = username
5          self.password = password
6          self.email = email
7          self.first_name = first_name
8          self.last_name = last_name
9          self.date_of_birth = date_of_birth
10         self.profile_picture = profile_picture
11
12     def __str__(self):
13         return f"User({self.user_id}, {self.username}, {self.email})"
```

Gallery :

```
class Gallery:

    def __init__(self, gallery_id, name, description, location, curator_id, opening_hours):

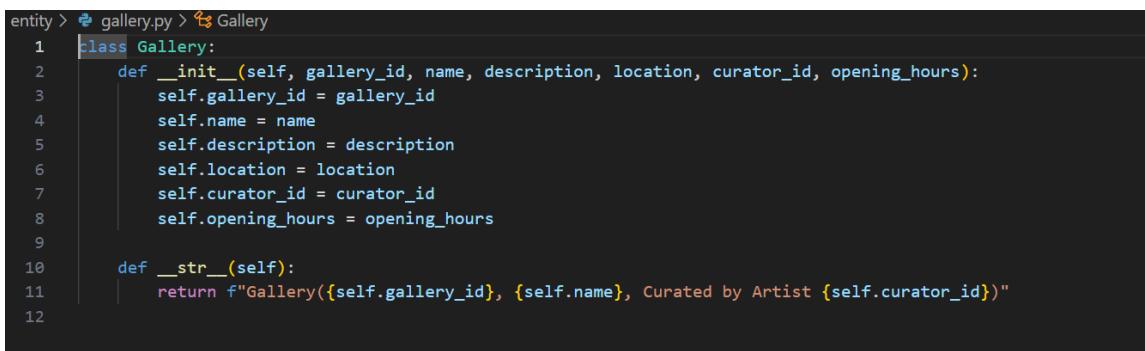
        self.gallery_id = gallery_id
```

```

self.name = name
self.description = description
self.location = location
self.curator_id = curator_id
self.opening_hours = opening_hours

def __str__(self):
    return f"Gallery({self.gallery_id}, {self.name}, Curated by Artist {self.curator_id})"

```



The screenshot shows a code editor window with the following code:

```

entity > gallery.py > Gallery
1  class Gallery:
2      def __init__(self, gallery_id, name, description, location, curator_id, opening_hours):
3          self.gallery_id = gallery_id
4          self.name = name
5          self.description = description
6          self.location = location
7          self.curator_id = curator_id
8          self.opening_hours = opening_hours
9
10     def __str__(self):
11         return f"Gallery({self.gallery_id}, {self.name}, Curated by Artist {self.curator_id})"
12

```

UserFavoriteArtwork:

```

class UserFavoriteArtwork:

    def __init__(self, user_id: int, artwork_id: int):
        self.user_id = user_id
        self.artwork_id = artwork_id

    def __str__(self):
        return f"UserFavoriteArtwork(UserID={self.user_id}, ArtworkID={self.artwork_id})"

```

```
entity > user_fav_artwork.py > UserFavoriteArtwork
1  class UserFavoriteArtwork:
2      def __init__(self, user_id: int, artwork_id: int):
3          self.user_id = user_id
4          self.artwork_id = artwork_id
5
6      def __str__(self):
7          return f"UserFavoriteArtwork(UserID={self.user_id}, ArtworkID={self.artwork_id})"
8
```

ArtworkGallery:

```
class ArtworkGallery:

    def __init__(self, artwork_id: int, gallery_id: int):
        self.artwork_id = artwork_id
        self.gallery_id = gallery_id

    def __str__(self):
        return f"ArtworkID: {self.artwork_id}, GalleryID: {self.gallery_id}"
```

```
entity > artwork_gallery.py > ...
1  class ArtworkGallery:
2      def __init__(self, artwork_id: int, gallery_id: int):
3          self.artwork_id = artwork_id
4          self.gallery_id = gallery_id
5
6      def __str__(self):
7          return f"ArtworkID: {self.artwork_id}, GalleryID: {self.gallery_id}"
```

2. dao / - Service Interface and Implementation

The dao (Data Access Object) package contains the service interface `gallery_service_interface.py`, which defines the functionalities the system offers. The `gallery_service_impl.py` class implements all the methods and handles actual SQL operations (add, update, delete, select) with the MySQL database.

Include Screenshots of:

- Interface with all method signatures
- Implementation showing logic for:
 - Adding artworks
 - Searching artworks
 - Adding to favorites
 - Raising exceptions

gallery_service_interface.py :

```
from abc import ABC, abstractmethod

from entity.artwork import Artwork


class GalleryServiceInterface(ABC):

    @abstractmethod
    def add_artwork(self, artwork: Artwork) -> bool:
        pass

    @abstractmethod
    def update_artwork(self, artwork: Artwork) -> bool:
        pass

    @abstractmethod
    def remove_artwork(self, artwork_id: int) -> bool:
        pass

    @abstractmethod
```

```
def get_artwork_by_id(self, artwork_id: int) -> Artwork:  
    pass
```

```
@abstractmethod  
def search_artworks(self, keyword: str) -> list:  
    pass
```

```
@abstractmethod  
def add_artwork_to_favorite(self, user_id: int, artwork_id: int) -> bool:  
    pass
```

```
@abstractmethod  
def remove_artwork_from_favorite(self, user_id: int, artwork_id: int) -> bool:  
    pass
```

```
@abstractmethod  
def get_user_favorite_artworks(self, user_id: int) -> list:  
    pass
```

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure under "VIRTUALLARTGALLERY".
- Open Editors:** Displays four files:
 - artwork_gallery.py
 - gallery_service_impl.py
 - gallery_service_interface.py (highlighted)
 - artist.py
- Code Editor:** The current file is "gallery_service_interface.py". The code defines an abstract class `GalleryServiceInterface` with several abstract methods (add_artwork, update_artwork, remove_artwork, get_artwork_by_id, search_artworks, add_artwork_to_favorite, remove_artwork_from_favorite) each containing a `pass` statement.
- Status Bar:** Shows the file is 3.11.5 ('venv': venv).

```
32     |     pass
33
34     @abstractmethod
35     def get_user_favorite_artworks(self, user_id: int) -> list:
36     |     pass
37
```

gallery_service_impl.py :

```
import mysql.connector

from entity.artwork import Artwork

from exception.artwork_not_found_exception import ArtworkNotFoundException

from dao.gallery_service_interface import GalleryServiceInterface
```

```
from util.db_conn_util import get_connection

class GalleryServiceImpl(GalleryServiceInterface):

    def __init__(self, db_config_path="db_config.properties"):

        self.connection = get_connection(db_config_path)

        self.cursor = self.connection.cursor(dictionary=True)
```

```
def add_artwork(self, artwork: Artwork) -> bool:  
    query = """  
        INSERT INTO Artwork (Title, Description, CreationDate, Medium, ImageURL,  
        ArtistID)  
        VALUES (%s, %s, %s, %s, %s, %s)  
        """  
  
    values = (  
        artwork.title,  
        artwork.description,  
        artwork.creation_date,  
        artwork.medium,  
        artwork.image_url,  
        artwork.artist_id  
    )  
  
    try:  
        self.cursor.execute(query, values)  
        self.connection.commit()  
        return True  
  
    except Exception as e:  
        print(f"Error while adding artwork: {e}")  
        return False
```

```
def update_artwork(self, artwork: Artwork) -> bool:  
    query = """  
        UPDATE Artwork SET Title=%s, Description=%s, CreationDate=%s,  
        """
```

```
Medium=%s, ImageURL=%s, ArtistID=%s WHERE ArtworkID=%s
"""

values = (
    artwork.title,
    artwork.description,
    artwork.creation_date,
    artwork.medium,
    artwork.image_url,
    artwork.artist_id,
    artwork.artwork_id
)

try:
    self.cursor.execute(query, values)
    if self.cursor.rowcount == 0:
        raise ArtworkNotFoundException(f"No artwork with ID {artwork.artwork_id}")
    self.connection.commit()
    return True
except ArtworkNotFoundException as ae:
    print(ae)
    return False
except Exception as e:
    print(f"Error while updating artwork: {e}")
    return False
```

```
def remove_artwork(self, artwork_id: int) -> bool:  
    query = "DELETE FROM Artwork WHERE ArtworkID = %s"  
  
    try:  
  
        self.cursor.execute(query, (artwork_id,))  
  
        if self.cursor.rowcount == 0:  
  
            raise ArtworkNotFoundException(f"No artwork found with ID  
{artwork_id}")  
  
        self.connection.commit()  
  
    return True  
  
except ArtworkNotFoundException as ae:  
  
    print(ae)  
  
    return False  
  
except Exception as e:  
  
    print(f"Error while deleting artwork: {e}")  
  
    return False
```

```
def get_artwork_by_id(self, artwork_id: int) -> Artwork:  
    query = "SELECT * FROM Artwork WHERE ArtworkID = %s"  
  
    try:  
  
        self.cursor.execute(query, (artwork_id,))  
  
        result = self.cursor.fetchone()  
  
        if not result:  
  
            raise ArtworkNotFoundException(f"No artwork found with ID  
{artwork_id}")  
  
        return Artwork(
```

```
        result['ArtworkID'], result['Title'], result['Description'],
        result['CreationDate'], result['Medium'], result['ImageURL'],
result['ArtistID']

    )

except ArtworkNotFoundException as ae:

    print(ae)

except Exception as e:

    print(f"Error fetching artwork: {e}")

def search_artworks(self, keyword: str) -> list:

    query = """
SELECT * FROM Artwork

WHERE Title LIKE %s OR Medium LIKE %s
"""

    try:

        like_keyword = f"%{keyword}%""
        self.cursor.execute(query, (like_keyword, like_keyword))
        results = self.cursor.fetchall()

        artworks = []
        for row in results:
            artwork = Artwork(
                row['ArtworkID'], row['Title'], row['Description'],
                row['CreationDate'], row['Medium'], row['ImageURL'], row['ArtistID']
            )
            artworks.append(artwork)
    except Exception as e:
        print(f"Error searching for artwork: {e}")
        return []
    return artworks
```

```
        artworks.append(artwork)

    return artworks

except Exception as e:
    print(f"Error while searching artworks: {e}")

    return []

def add_artwork_to_favorite(self, user_id: int, artwork_id: int) -> bool:
    query = """
    INSERT INTO User_Favorite_Artwork (UserID, ArtworkID)
    VALUES (%s, %s)
    """

    try:
        self.cursor.execute(query, (user_id, artwork_id))
        self.connection.commit()

        return True
    except mysql.connector.IntegrityError:
        print("This artwork is already in the user's favorites.")

        return False
    except Exception as e:
        print(f"Error while adding to favorites: {e}")

        return False

def remove_artwork_from_favorite(self, user_id: int, artwork_id: int) -> bool:
    query = """
```

```
DELETE FROM User_Favorite_Artwork
WHERE UserID = %s AND ArtworkID = %s
"""

try:
    self.cursor.execute(query, (user_id, artwork_id))
    if self.cursor.rowcount == 0:
        print("Favorite entry not found.")
        return False
    self.connection.commit()
    return True
except Exception as e:
    print(f"Error while removing from favorites: {e}")
    return False
```

```
def remove_artwork_from_favorite(self, user_id: int, artwork_id: int) -> bool:
    query = """
DELETE FROM User_Favorite_Artwork
WHERE UserID = %s AND ArtworkID = %s
"""

    try:
        self.cursor.execute(query, (user_id, artwork_id))
        if self.cursor.rowcount == 0:
            print("Favorite entry not found.")
            return False
        self.connection.commit()
```

```
        return True

    except Exception as e:
        print(f"Error while removing from favorites: {e}")
        return False

def get_user_favorite_artworks(self, user_id: int) -> list:
    query = """
        SELECT a.* FROM Artwork a
        JOIN User_Favorite_Artwork ufa ON a.ArtworkID = ufa.ArtworkID
        WHERE ufa.UserID = %s
    """

    try:
        self.cursor.execute(query, (user_id,))
        results = self.cursor.fetchall()
        artworks = []
        for row in results:
            artwork = Artwork(
                row['ArtworkID'], row['Title'], row['Description'],
                row['CreationDate'], row['Medium'], row['ImageURL'], row['ArtistID']
            )
            artworks.append(artwork)
    except Exception as e:
        print(f"Error while fetching favorite artworks: {e}")
        return []
```

```
def assign_artwork_to_gallery(self, artwork_id: int, gallery_id: int) -> bool:
    query = """
        INSERT INTO Artwork_Gallery (ArtworkID, GalleryID)
        VALUES (%s, %s)
    """

    try:
        self.cursor.execute(query, (artwork_id, gallery_id))
        self.connection.commit()
        return True
    except Exception as e:
        print(f"Error assigning artwork to gallery: {e}")
        return False
```

```
def view_artworks_in_galleries(self) -> list:
    query = """
        SELECT ag.ArtworkID, g.Name AS GalleryName, a.Title AS ArtworkTitle
        FROM Artwork_Gallery ag
        JOIN Artwork a ON ag.ArtworkID = a.ArtworkID
        JOIN Gallery g ON ag.GalleryID = g.GalleryID
    """

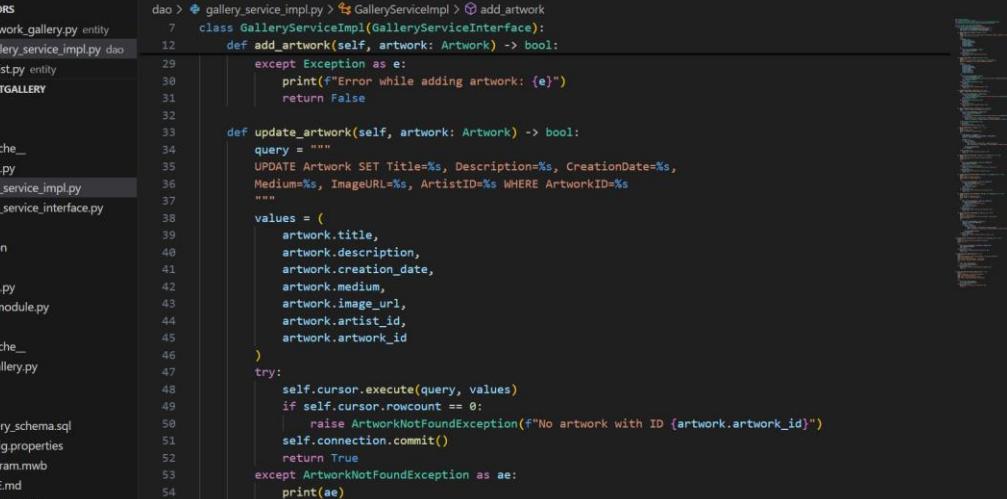
    try:
        self.cursor.execute(query)
        return self.cursor.fetchall()
    except Exception as e:
```

```
print(f"Error fetching artwork-gallery mapping: {e}")  
return []  
  
  
def view_user_favorite_artworks_details(self) -> list:  
    query = """  
        SELECT u.Username, a.Title AS ArtworkTitle  
        FROM User_Favorite_Artwork ufa  
        JOIN `User` u ON ufa.UserID = u.UserID  
        JOIN Artwork a ON ufa.ArtworkID = a.ArtworkID  
        """  
  
    try:  
        self.cursor.execute(query)  
        return self.cursor.fetchall()  
    except Exception as e:  
        print(f"Error fetching user favorites: {e}")  
    return []
```

The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows the project structure with files like `artwork_gallery.py`, `gallery_service_impl.py`, and `artist.py`.
- Editor:** Displays the `gallery_service_impl.py` file containing Python code for a gallery service implementation.
- Terminal:** Shows the command `poetry run flask run` being executed.
- Status Bar:** Shows the current file is `gallery_service_impl.py`, line 20, column 15, with 311 lines of code in total.

```
file:///C:/Users/.../VirtualArtGallery/gallery_service_impl.py
1 import mysql.connector
2 from entity.artwork import Artwork
3 from exception.artwork_not_found_exception import ArtworkNotFoundException
4 from dao.gallery_service_interface import GalleryServiceInterface
5
6 from util.db_conn_util import get_connection
7 class GalleryServiceImpl(GalleryServiceInterface):
8     def __init__(self, db_config_path="db_config.properties"):
9         self.connection = get_connection(db_config_path)
10        self.cursor = self.connection.cursor(dictionary=True)
11
12    def add_artwork(self, artwork: Artwork) -> bool:
13        query = """
14            INSERT INTO Artwork (Title, Description, CreationDate, Medium, ImageURL, ArtistID)
15            VALUES (%s, %s, %s, %s, %s, %s)
16        """
17
18        values = [
19            artwork.title,
20            artwork.description,
21            artwork.creation_date,
22            artwork.medium,
23            artwork.image_url,
24            artwork.artist_id
25        ]
26
27        try:
28            self.cursor.execute(query, values)
29            self.connection.commit()
30        except:
31            return False
32
33        return True
```



The screenshot shows a Python code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Toolbar:** Includes icons for file operations like Open, Save, Find, and Run.
- Left Sidebar (EXPLORER):** Shows the project structure:
 - OPEN EDITORS: artwork_gallery.py, gallery_service_impl.py, artist.py
 - VIRTUALARTGALLERY: .vscode, dao, _pycache_, __init__.py, gallery_service_impl.py, gallery_service_interface.py, exception, main, tests, _pycache_, test_gallery.py, util, venv, art_gallery_schema.sql, db.config.properties, eer.diagram.mwb, README.md, requirements.txt
 - OUTLINE and TIMELINE sections.
- Code Editor:** Displays the `gallery_service_impl.py` file content. The code implements a `GalleryServiceImpl` class that interacts with a DAO to add and update artworks. It includes error handling and logging.
- Right Sidebar:** Shows a vertical stack of code snippets or history.
- Bottom Status Bar:** Shows the current file (VirtualArtGallery), line number (Ln 20), and other status information (Col 35, Spaces: 4, UTF-8, CRLF, Python, 3.11.5 (venv:venv)).

File Edit Selection View Go Run ... ↶ ↷ 🔍 VirtualArtGallery

EXPLORER OPEN EDITORS VIRTUALARTGALLERY .vscode dao _pycache_ __init__.py gallery_service_impl.py dao entity exception main __init__.py main_module.py tests __pycache__ test_gallery.py util venv art_gallery_schema.sql db_config.properties eer_diagram.mwb README.md requirements.txt tool_db_connection.py OUTLINE TIMELINE

```
    class GalleryServiceImpl(GalleryServiceInterface):
        def remove_artwork(self, artwork_id: int) -> bool:
            query = "DELETE FROM Artwork WHERE ArtworkID = %s"
            try:
                self.cursor.execute(query, (artwork_id,))
                if self.cursor.rowcount == 0:
                    raise ArtworkNotFoundException(f"No artwork found with ID {artwork_id}")
                self.connection.commit()
            return True
        except ArtworkNotFoundException as ae:
            print(ae)
            return False
        except Exception as e:
            print(f"Error while deleting artwork: {e}")
            return False

        def get_artwork_by_id(self, artwork_id: int) -> Artwork:
            query = "SELECT * FROM Artwork WHERE ArtworkID = %s"
            try:
                self.cursor.execute(query, (artwork_id,))
                result = self.cursor.fetchone()
                if not result:
                    raise ArtworkNotFoundException("No artwork found with ID {artwork_id}")
                return Artwork(
                    result['ArtworkID'], result['Title'], result['Description'],
                    result['CreationDate'], result['Medium'], result['ImageURL'], result['ArtistID']
                )
            except ArtworkNotFoundException as ae:
                print(ae)
            except Exception as e:
                print(f"Error fetching artwork: {e}")
```

Ln 20, Col 35 Spaces: 4 UTF-8 CRLF () Python 3.11.5 (venv: venv)

File Edit Selection View Go Run ... ↶ ↷ 🔍 VirtualArtGallery

EXPLORER OPEN EDITORS VIRTUALARTGALLERY .vscode dao _pycache_ __init__.py gallery_service_impl.py dao entity exception main __init__.py main_module.py tests __pycache__ test_gallery.py util venv art_gallery_schema.sql db_config.properties eer_diagram.mwb README.md requirements.txt tool_db_connection.py OUTLINE TIMELINE

```
    class GalleryServiceImpl(GalleryServiceInterface):
        def get_artwork_by_id(self, artwork_id: int) -> Artwork:
            print(f"Error fetching artwork: {e}")

        def search_artworks(self, keyword: str) -> list:
            query = """
            SELECT * FROM Artwork
            WHERE Title LIKE %s OR Medium LIKE %
            """
            try:
                like_keyword = f"%{keyword}%""
                self.cursor.execute(query, (like_keyword, like_keyword))
                results = self.cursor.fetchall()

                artworks = []
                for row in results:
                    artwork = Artwork(
                        row['ArtworkID'], row['Title'], row['Description'],
                        row['CreationDate'], row['Medium'], row['ImageURL'], row['ArtistID']
                    )
                    artworks.append(artwork)
                return artworks
            except Exception as e:
                print(f"Error while searching artworks: {e}")
                return []

        def add_artwork_to_favorite(self, user_id: int, artwork_id: int) -> bool:
            query = """
            INSERT INTO User_Favorite_Artwork (UserID, ArtworkID)
            VALUES (%s, %s)
            """
            try:
```

Ln 20, Col 35 Spaces: 4 UTF-8 CRLF () Python 3.11.5 (venv: venv)

```
File Edit Selection View Go Run ... artword_gallery.py gallery_service_impl.py artist.py

EXPLORER artwork_gallery.py ...
OPEN EDITORS artwork_gallery.py entity
  gallery_service_impl.py dao
    artist.py entity
VIRTUAUTOGALLERY .vscode
  dao
    _pycache_
    __init__.py
    gallery_service_impl.py
    gallery_service_interface.py
  entity
  exception
  main
    __init__.py
    main_module.py
  tests
    _pycache_
    test_gallery.py
  util
  venv
  art_gallery_schema.sql
  db_config.properties
  eer diagram.mwb
  README.md
  requirements.txt
  .tox/dlib_connection.py
  OUTLINE
  TIMELINE

dao > gallery_service_impl.py > GalleryServiceImpl > add_artwork
114     class GalleryServiceImpl(GalleryServiceInterface):
115         def add_artwork_to_favorite(self, user_id: int, artwork_id: int) -> bool:
116             try:
117                 self.cursor.execute(query, (user_id, artwork_id))
118                 self.connection.commit()
119                 return True
120             except mysql.connector.IntegrityError:
121                 print("This artwork is already in the user's favorites.")
122                 return False
123             except Exception as e:
124                 print(f"Error while adding to favorites: {e}")
125                 return False
126
127         def remove_artwork_from_favorite(self, user_id: int, artwork_id: int) -> bool:
128             query = """
129             DELETE FROM User_Favorite_Artwork
130             WHERE UserID = %s AND ArtworkID = %s
131             """
132
133             try:
134                 self.cursor.execute(query, (user_id, artwork_id))
135                 if self.cursor.rowcount == 0:
136                     print("Favorite entry not found.")
137                     return False
138                 self.connection.commit()
139                 return True
140             except Exception as e:
141                 print(f"Error while removing from favorites: {e}")
142                 return False
143
144         def remove_artwork_from_favorite(self, user_id: int, artwork_id: int) -> bool:
145             query = """
146             DELETE FROM User_Favorite_Artwork
147             """

Ln 20, Col 35 Spaces:4 UTF-8 CRLF () Python 3.11.5 (venv:venv)
```

The screenshot shows a Python code editor interface with the following details:

- File Explorer (Left):** Shows the project structure with files like `artwork_gallery.py`, `gallery_service_impl.py`, `artist.py`, `.vscode`, `dao`, `_pycache_`, `__init__.py`, `gallery_service_impl.py`, `gallery_service_interface.py`, `entity`, `exception`, `main`, `tests`, `venv`, `art_gallery_schema.sql`, `db_config.properties`, `er diagram.mwb`, `README.md`, `requirements.txt`, and `outline`.
- Code Editor (Right):** Displays the `gallery_service_impl.py` file content. The code implements a `GalleryServiceImpl` class that interacts with a database to manage artwork favorites.

```
class GalleryServiceImpl(GalleryServiceInterface):
    def remove_artwork_from_favorite(self, user_id: int, artwork_id: int) -> bool:
        DELETE FROM User_Favorite_Artwork
        WHERE UserID = %s AND ArtworkID = %
        """
        try:
            self.cursor.execute(query, (user_id, artwork_id))
            if self.cursor.rowcount == 0:
                print("Favorite entry not found.")
                return False
            self.connection.commit()
            return True
        except Exception as e:
            print(f"Error while removing from favorites: {e}")
            return False

    def get_user_favorite_artworks(self, user_id: int) -> list:
        query = """
        SELECT a.* FROM Artwork a
        JOIN User_Favorite_Artwork ufa ON a.ArtworkID = ufa.ArtworkID
        WHERE ufa.UserID = %
        """
        try:
            self.cursor.execute(query, (user_id,))
            results = self.cursor.fetchall()
            artworks = []
            for row in results:
                artwork = Artwork(
                    row['ArtworkID'], row['Title'], row['Description'],
                    row['CreationDate'], row['Medium'], row['ImageURL'], row['ArtistID']
                )
                artworks.append(artwork)
        
```

VirtualArtGallery

```
artwork_gallery.py gallery_service.impl.py artist.py
dao > gallery_service.impl.py > GalleryServiceImpl > add_artwork
      7 class GalleryServiceImpl(GalleryServiceInterface):
      8     def get_user_favorite_artworks(self, user_id: int) -> list:
      9         return artworks
     10     except Exception as e:
     11         print(f"Error while fetching favorite artworks: {e}")
     12         return []
     13
     14     def assign_artwork_to_gallery(self, artwork_id: int, gallery_id: int) -> bool:
     15         query = """
     16             INSERT INTO Artwork_Gallery (ArtworkID, GalleryID)
     17             VALUES (%s, %s)
     18         """
     19
     20         try:
     21             self.cursor.execute(query, (artwork_id, gallery_id))
     22             self.connection.commit()
     23             return True
     24         except Exception as e:
     25             print(f"Error assigning artwork to gallery: {e}")
     26             return False
     27
     28     def view_artworks_in_galleries(self) -> list:
     29         query = """
     30             SELECT ag.ArtworkID, g.Name AS GalleryName, a.Title AS ArtworkTitle
     31             FROM Artwork_Gallery ag
     32             JOIN Artwork a ON ag.ArtworkID = a.ArtworkID
     33             JOIN Gallery g ON ag.GalleryID = g.GalleryID
     34         """
     35
     36         try:
     37             self.cursor.execute(query)
     38             return self.cursor.fetchall()
     39         except Exception as e:
     40             print(f"Error fetching artwork-gallery mapping: {e}")
     41             return []
     42
     43
     44
     45
     46
     47
     48
     49
     50
     51
     52
     53
     54
     55
     56
     57
     58
     59
     60
     61
     62
     63
     64
     65
     66
     67
     68
     69
     70
     71
     72
     73
     74
     75
     76
     77
     78
     79
     80
     81
     82
     83
     84
     85
     86
     87
     88
     89
     90
     91
     92
     93
     94
     95
     96
     97
     98
     99
     100
     101
     102
     103
     104
     105
     106
     107
     108
     109
     110
     111
     112
     113
     114
     115
     116
     117
     118
     119
     120
     121
     122
     123
     124
     125
     126
     127
     128
     129
     130
     131
     132
     133
     134
     135
     136
     137
     138
     139
     140
     141
     142
     143
     144
     145
     146
     147
     148
     149
     150
     151
     152
     153
     154
     155
     156
     157
     158
     159
     160
     161
     162
     163
     164
     165
     166
     167
     168
     169
     170
     171
     172
     173
     174
     175
     176
     177
     178
     179
     180
     181
     182
     183
     184
     185
     186
     187
     188
     189
     190
     191
     192
     193
     194
     195
     196
     197
     198
     199
     200
     201
     202
     203
     204
     205
     206
     207
     208
     209
     210
     211
     212
     213
     214
     215
     216
     217
     218
     219
     220
     221
     222
     223
     224
     225
     226
     227
     228
     229
     230
     231
     232
     233
     234
     235
     236
     237
     238
     239
     240
     241
     242
     243
     244
     245
     246
     247
     248
     249
     250
     251
     252
     253
     254
     255
     256
     257
     258
     259
     260
     261
     262
     263
     264
     265
     266
     267
     268
     269
     270
     271
     272
     273
     274
     275
     276
     277
     278
     279
     280
     281
     282
     283
     284
     285
     286
     287
     288
     289
     290
     291
     292
     293
     294
     295
     296
     297
     298
     299
     300
     301
     302
     303
     304
     305
     306
     307
     308
     309
     310
     311
     312
     313
     314
     315
     316
     317
     318
     319
     320
     321
     322
     323
     324
     325
     326
     327
     328
     329
     330
     331
     332
     333
     334
     335
     336
     337
     338
     339
     340
     341
     342
     343
     344
     345
     346
     347
     348
     349
     350
     351
     352
     353
     354
     355
     356
     357
     358
     359
     360
     361
     362
     363
     364
     365
     366
     367
     368
     369
     370
     371
     372
     373
     374
     375
     376
     377
     378
     379
     380
     381
     382
     383
     384
     385
     386
     387
     388
     389
     390
     391
     392
     393
     394
     395
     396
     397
     398
     399
     400
     401
     402
     403
     404
     405
     406
     407
     408
     409
     410
     411
     412
     413
     414
     415
     416
     417
     418
     419
     420
     421
     422
     423
     424
     425
     426
     427
     428
     429
     430
     431
     432
     433
     434
     435
     436
     437
     438
     439
     440
     441
     442
     443
     444
     445
     446
     447
     448
     449
     450
     451
     452
     453
     454
     455
     456
     457
     458
     459
     460
     461
     462
     463
     464
     465
     466
     467
     468
     469
     470
     471
     472
     473
     474
     475
     476
     477
     478
     479
     480
     481
     482
     483
     484
     485
     486
     487
     488
     489
     490
     491
     492
     493
     494
     495
     496
     497
     498
     499
     500
     501
     502
     503
     504
     505
     506
     507
     508
     509
     510
     511
     512
     513
     514
     515
     516
     517
     518
     519
     520
     521
     522
     523
     524
     525
     526
     527
     528
     529
     530
     531
     532
     533
     534
     535
     536
     537
     538
     539
     540
     541
     542
     543
     544
     545
     546
     547
     548
     549
     550
     551
     552
     553
     554
     555
     556
     557
     558
     559
     560
     561
     562
     563
     564
     565
     566
     567
     568
     569
     570
     571
     572
     573
     574
     575
     576
     577
     578
     579
     580
     581
     582
     583
     584
     585
     586
     587
     588
     589
     590
     591
     592
     593
     594
     595
     596
     597
     598
     599
     600
     601
     602
     603
     604
     605
     606
     607
     608
     609
     610
     611
     612
     613
     614
     615
     616
     617
     618
     619
     620
     621
     622
     623
     624
     625
     626
     627
     628
     629
     630
     631
     632
     633
     634
     635
     636
     637
     638
     639
     640
     641
     642
     643
     644
     645
     646
     647
     648
     649
     650
     651
     652
     653
     654
     655
     656
     657
     658
     659
     660
     661
     662
     663
     664
     665
     666
     667
     668
     669
     670
     671
     672
     673
     674
     675
     676
     677
     678
     679
     680
     681
     682
     683
     684
     685
     686
     687
     688
     689
     690
     691
     692
     693
     694
     695
     696
     697
     698
     699
     700
     701
     702
     703
     704
     705
     706
     707
     708
     709
     710
     711
     712
     713
     714
     715
     716
     717
     718
     719
     720
     721
     722
     723
     724
     725
     726
     727
     728
     729
     730
     731
     732
     733
     734
     735
     736
     737
     738
     739
     740
     741
     742
     743
     744
     745
     746
     747
     748
     749
     750
     751
     752
     753
     754
     755
     756
     757
     758
     759
     760
     761
     762
     763
     764
     765
     766
     767
     768
     769
     770
     771
     772
     773
     774
     775
     776
     777
     778
     779
     780
     781
     782
     783
     784
     785
     786
     787
     788
     789
     790
     791
     792
     793
     794
     795
     796
     797
     798
     799
     800
     801
     802
     803
     804
     805
     806
     807
     808
     809
     810
     811
     812
     813
     814
     815
     816
     817
     818
     819
     820
     821
     822
     823
     824
     825
     826
     827
     828
     829
     830
     831
     832
     833
     834
     835
     836
     837
     838
     839
     840
     841
     842
     843
     844
     845
     846
     847
     848
     849
     850
     851
     852
     853
     854
     855
     856
     857
     858
     859
     860
     861
     862
     863
     864
     865
     866
     867
     868
     869
     870
     871
     872
     873
     874
     875
     876
     877
     878
     879
     880
     881
     882
     883
     884
     885
     886
     887
     888
     889
     890
     891
     892
     893
     894
     895
     896
     897
     898
     899
     900
     901
     902
     903
     904
     905
     906
     907
     908
     909
     910
     911
     912
     913
     914
     915
     916
     917
     918
     919
     920
     921
     922
     923
     924
     925
     926
     927
     928
     929
     930
     931
     932
     933
     934
     935
     936
     937
     938
     939
     940
     941
     942
     943
     944
     945
     946
     947
     948
     949
     950
     951
     952
     953
     954
     955
     956
     957
     958
     959
     960
     961
     962
     963
     964
     965
     966
     967
     968
     969
     970
     971
     972
     973
     974
     975
     976
     977
     978
     979
     980
     981
     982
     983
     984
     985
     986
     987
     988
     989
     990
     991
     992
     993
     994
     995
     996
     997
     998
     999
     1000
     1001
     1002
     1003
     1004
     1005
     1006
     1007
     1008
     1009
     1010
     1011
     1012
     1013
     1014
     1015
     1016
     1017
     1018
     1019
     1020
     1021
     1022
     1023
     1024
     1025
     1026
     1027
     1028
     1029
     1030
     1031
     1032
     1033
     1034
     1035
     1036
     1037
     1038
     1039
     1040
     1041
     1042
     1043
     1044
     1045
     1046
     1047
     1048
     1049
     1050
     1051
     1052
     1053
     1054
     1055
     1056
     1057
     1058
     1059
     1060
     1061
     1062
     1063
     1064
     1065
     1066
     1067
     1068
     1069
     1070
     1071
     1072
     1073
     1074
     1075
     1076
     1077
     1078
     1079
     1080
     1081
     1082
     1083
     1084
     1085
     1086
     1087
     1088
     1089
     1090
     1091
     1092
     1093
     1094
     1095
     1096
     1097
     1098
     1099
     1100
     1101
     1102
     1103
     1104
     1105
     1106
     1107
     1108
     1109
     1110
     1111
     1112
     1113
     1114
     1115
     1116
     1117
     1118
     1119
     1120
     1121
     1122
     1123
     1124
     1125
     1126
     1127
     1128
     1129
     1130
     1131
     1132
     1133
     1134
     1135
     1136
     1137
     1138
     1139
     1140
     1141
     1142
     1143
     1144
     1145
     1146
     1147
     1148
     1149
     1150
     1151
     1152
     1153
     1154
     1155
     1156
     1157
     1158
     1159
     1160
     1161
     1162
     1163
     1164
     1165
     1166
     1167
     1168
     1169
     1170
     1171
     1172
     1173
     1174
     1175
     1176
     1177
     1178
     1179
     1180
     1181
     1182
     1183
     1184
     1185
     1186
     1187
     1188
     1189
     1190
     1191
     1192
     1193
     1194
     1195
     1196
     1197
     1198
     1199
     1200
     1201
     1202
     1203
     1204
     1205
     1206
     1207
     1208
     1209
     1210
     1211
     1212
     1213
     1214
     1215
     1216
     1217
     1218
     1219
     1220
     1221
     1222
     1223
     1224
     1225
     1226
     1227
     1228
     1229
     1230
     1231
     1232
     1233
     1234
     1235
     1236
     1237
     1238
     1239
     1240
     1241
     1242
     1243
     1244
     1245
     1246
     1247
     1248
     1249
     1250
     1251
     1252
     1253
     1254
     1255
     1256
     1257
     1258
     1259
     1260
     1261
     1262
     1263
     1264
     1265
     1266
     1267
     1268
     1269
     1270
     1271
     1272
     1273
     1274
     1275
     1276
     1277
     1278
     1279
     1280
     1281
     1282
     1283
     1284
     1285
     1286
     1287
     1288
     1289
     1290
     1291
     1292
     1293
     1294
     1295
     1296
     1297
     1298
     1299
     1300
     1301
     1302
     1303
     1304
     1305
     1306
     1307
     1308
     1309
     1310
     1311
     1312
     1313
     1314
     1315
     1316
     1317
     1318
     1319
     1320
     1321
     1322
     1323
     1324
     1325
     1326
     1327
     1328
     1329
     1330
     1331
     1332
     1333
     1334
     1335
     1336
     1337
     1338
     1339
     1340
     1341
     1342
     1343
     1344
     1345
     1346
     1347
     1348
     1349
     1350
     1351
     1352
     1353
     1354
     1355
     1356
     1357
     1358
     1359
     1360
     1361
     1362
     1363
     1364
     1365
     1366
     1367
     1368
     1369
     1370
     1371
     1372
     1373
     1374
     1375
     1376
     1377
     1378
     1379
     1380
     1381
     1382
     1383
     1384
     1385
     1386
     1387
     1388
     1389
     1390
     1391
     1392
     1393
     1394
     1395
     1396
     1397
     1398
     1399
     1400
     1401
     1402
     1403
     1404
     1405
     1406
     1407
     1408
     1409
     1410
     1411
     1412
     1413
     1414
     1415
     1416
     1417
     1418
     1419
     1420
     1421
     1422
     1423
     1424
     1425
     1426
     1427
     1428
     1429
     1430
     1431
     1432
     1433
     1434
     1435
     1436
     1437
     1438
     1439
     1440
     1441
     1442
     1443
     1444
     1445
     1446
     1447
     1448
     1449
     1450
     1451
     1452
     1453
     1454
     1455
     1456
     1457
     1458
     1459
     1460
     1461
     1462
     1463
     1464
     1465
     1466
     1467
     1468
     1469
     1470
     1471
     1472
     1473
     1474
     1475
     1476
     1477
     1478
     1479
     1480
     1481
     1482
     1483
     1484
     1485
     1486
     1487
     1488
     1489
     1490
     1491
     1492
     1493
     1494
     1495
     1496
     1497
     1498
     1499
     1500
     1501
     1502
     1503
     1504
     1505
     1506
     1507
     1508
     1509
     1510
     1511
     1512
     1513
     1514
     1515
     1516
     1517
     1518
     1519
     1520
     1521
     1522
     1523
     1524
     1525
     1526
     1527
     1528
     1529
     1530
     1531
     1532
     1533
     1534
     1535
     1536
     1537
     1538
     1539
     1540
     1541
     1542
     1543
     1544
     1545
     1546
     1547
     1548
     1549
     1550
     1551
     1552
     1553
     1554
     1555
     1556
     1557
     1558
     1559
     1560
     1561
     1562
     1563
     1564
     1565
     1566
     1567
     1568
     1569
     1570
     1571
     1572
     1573
     1574
     1575
     1576
     1577
     1578
     1579
     1580
     1581
     1582
     1583
     1584
     1585
     1586
     1587
     1588
     1589
     1590
     1591
     1592
     1593
     1594
     1595
     1596
     1597
     1598
     1599
     1600
     1601
     1602
     1603
     1604
     1605
     1606
     1607
     1608
     1609
     1610
     1611
     1612
     1613
     1614
     1615
     1616
     1617
     1618
     1619
     1620
     1621
     1622
     1623
     1624
     1625
     1626
     1627
     1628
     1629
     1630
     1631
     1632
     1633
     1634
     1635
     1636
     1637
     1638
     1639
     1640
     1641
     1642
     1643
     1644
     1645
     1646
     1647
     1648
     1649
     1650
     1651
     1652
     1653
     1654
     1655
     1656
     1657
     1658
     1659
     1660
     1661
     1662
     1663
     1664
     1665
     1666
     1667
     1668
     1669
     1670
     1671
     1672
     1673
     1674
     1675
     1676
     1677
     1678
     1679
     1680
     1681
     1682
     1683
     1684
     1685
     1686
     1687
     1688
     1689
     1690
     1691
     1692
     1693
     1694
     1695
     1696
     1697
     1698
     1699
     1700
     1701
     1702
     1703
     1704
     1705
     1706
     1707
     1708
     1709
     1710
     1711
     1712
     1713
     1714
     1715
     1716
     1717
     1718
     1719
     1720
     1721
     1722
     1723
     1724
     1725
     1726
     1727
     1728
     1729
     1730
     1731
     1732
     1733
     1734
     1735
     1736
     1737
     1738
     1739
     1740
     1741
     1742

```

3. util/ – Database Utility Layer:

Description:

This package is responsible for database configuration and connection. The DBPropertyUtil class reads database credentials from a .properties file. The DBConnUtil class uses this information to establish a connection using mysql.connector.

Include Screenshots of:

- db_config.properties (credentials like host, username, dbname)
- db_property_util.py showing file read logic
- db_conn_util.py showing connection logic

1 db.property_util.py

```
def get_property_string(file_path):  
    props = {}  
  
    with open(file_path, 'r') as f:  
  
        for line in f:  
  
            if '=' in line and not line.startswith('#'):  
  
                key, value = line.strip().split('=', 1)  
  
                props[key.strip()] = value.strip()  
  
    return props
```

The screenshot shows the VS Code interface with the 'VirtualArtGallery' workspace open. The Explorer sidebar on the left lists files and folders, including 'artwork_gallery.py', 'gallery_service_impl.py', 'gallery_service_interface.py', 'db_conn_util.py', 'db_property_util.py', and 'artist.py'. The 'util' folder contains '_pycache_/_init_.py', 'db_conn_util.py', and 'db_property_util.py'. The 'db_property_util.py' file is currently selected and shown in the main editor area. The code in the editor is:

```
util > db_property_util.py > get_property_string
1 def get_property_string(file_path):
2     props = {}
3     with open(file_path, 'r') as f:
4         for line in f:
5             if '=' in line and not line.startswith('#'):
6                 key, value = line.strip().split('=', 1)
7                 props[key.strip()] = value.strip()
8
9 return props
```

The status bar at the bottom indicates the file is at Line 1, Column 1, with 4 spaces, using UTF-8 encoding, and is a Python 3.11.5 ('venv': venv) file.

db_conn_util.py

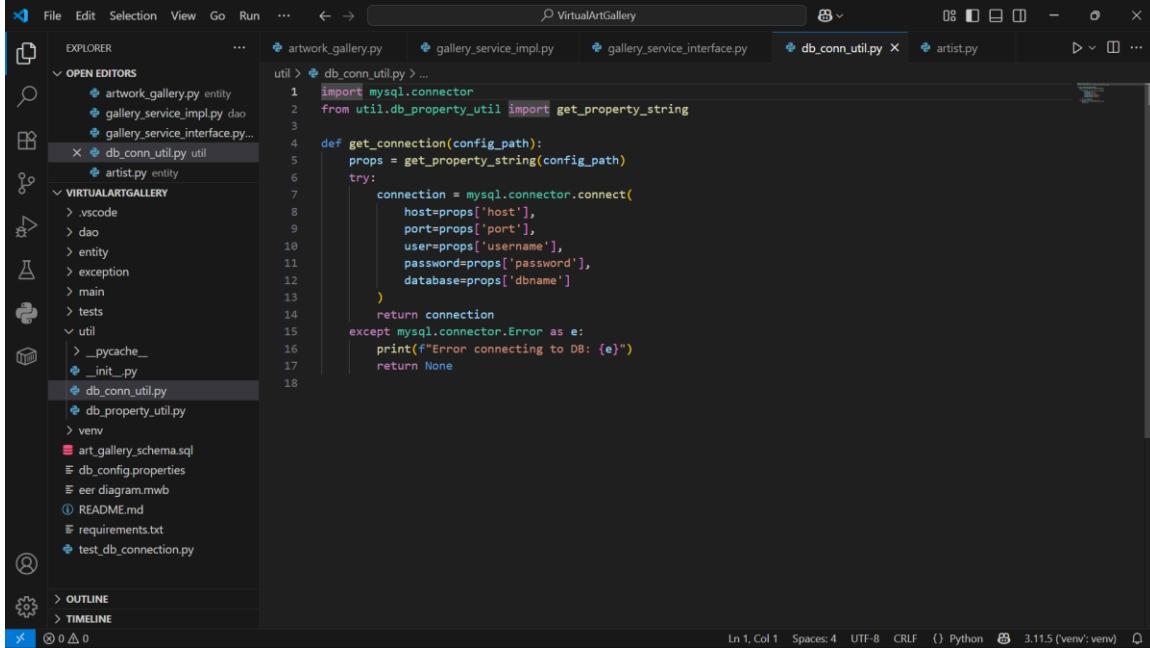
```
import mysql.connector

from util.db_property_util import get_property_string

def get_connection(config_path):
    props = get_property_string(config_path)
    try:
        connection = mysql.connector.connect(
            host=props['host'],
            port=props['port'],
            user=props['username'],
            password=props['password'],
            database=props['dbname']
        )
        return connection
    except mysql.connector.Error as e:
```

```
print(f"Error connecting to DB: {e}")
```

```
return None
```



The screenshot shows a code editor interface with the title "VirtualArtGallery". The left sidebar displays a file tree for a project named "VIRTUALARTGALLERY". The "db_conn_util.py" file is open in the editor, showing the following code:

```
util > db_conn_util.py ...
1 import mysql.connector
2 from util.db_property_util import get_property_string
3
4 def get_connection(config_path):
5     props = get_property_string(config_path)
6     try:
7         connection = mysql.connector.connect(
8             host=props['host'],
9             port=props['port'],
10            user=props['username'],
11            password=props['password'],
12            database=props['dbname']
13        )
14     return connection
15 except mysql.connector.Error as e:
16     print(f"Error connecting to DB: {e}")
17     return None
18
```

The status bar at the bottom indicates the code is in Python 3.11.5 environment.

4. exception/ - Custom Exceptions

Description:

This package contains user-defined exceptions used to handle errors gracefully in the system. For example, if a user searches for an invalid artwork ID, ArtworkNotFoundException is raised and caught in the CLI menu, showing a user-friendly message.

Include Screenshots of:

- **ArtworkNotFoundException class**

```
• class ArtworkNotFoundException(Exception):
•     def __init__(self, message="Artwork not found with the given
      ID."):
•         super().__init__(message)
```

- **UserNotFoundException class**

```

class UserNotFoundException(Exception):
    def __init__(self, message="User not found with the given ID."):
        super().__init__(message)

```

- Code in VirtualArtGalleryImpl that raises exceptions

```

exception > user_not_found_exception.py > UserNotFoundException
1 class UserNotFoundException(Exception):
2     def __init__(self, message="User not found with the given ID."):
3         super().__init__(message)
4

```

```

artwork_not_found_exception.py X
exception > artwork_not_found_exception.py > ArtworkNotFoundException
1 class ArtworkNotFoundException(Exception):
2     def __init__(self, message="Artwork not found with the given ID."):
3         super().__init__(message)

```

```

artwork_not_found_exception.py ArtworkAlreadyFavoritedException.py DatabaseConnectionException.py
exception > ArtworkAlreadyFavoritedException.py > ...
1 class ArtworkAlreadyFavoritedException(Exception):
2     def __init__(self, message="This artwork is already in the user's favorites."):
3         super().__init__(message)
4

```

```

DatabaseConnectionException.py X
exception > DatabaseConnectionException.py > ...
1 class DatabaseConnectionException(Exception):
2     def __init__(self, message="Unable to connect to the database."):
3         super().__init__(message)
4

```

```

DatabaseConnectionException.py duplicate_artwork_exception.py X
exception > duplicate_artwork_exception.py > ...
1 class DuplicateArtworkException(Exception):
2     def __init__(self, message="Artwork already exists."):
3         super().__init__(message)

```

```

DatabaseConnectionException.py duplicate_artwork_exception.py empty_field_exception.py X
exception > empty_field_exception.py > EmptyFieldException > __init__
1 class EmptyFieldException(Exception):
2     def __init__(self, message="Required field cannot be empty."):
3         super().__init__(message)

```

```
exception > 🗂️ GalleryNotFoundException.py > ...
1   class GalleryNotFoundException(Exception):
2       def __init__(self, message="Gallery not found."):
3           super().__init__(message)
4
```

```
exception > 🗂️ InvalidMediumException.py > ...
1   class InvalidMediumException(Exception):
2       def __init__(self, message="Invalid artwork medium. Please choose from a supported list."):
3           super().__init__(message)
4
```

```
exception > 🗂️ NoFavoriteArtworksException > ...
1   class NoFavoriteArtworksException(Exception):
2       def __init__(self, message="No favorite artworks found for this user."):
3           super().__init__(message)
4
```

5. main/ - CLI Menu (User Interaction)

Description:

The main package holds the CLI menu interface (main_module.py) that simulates the interactive system. From here, users can create artworks, search, add to favorites, and more. Exception handling is also performed here to ensure smooth user experience.

Include Screenshots of:

- Menu interface running in terminal
- Adding new artwork
- Searching artwork
- Adding to favorites
- Exception messages

Main.module

```
• import sys
• import os
• sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
•
• from dao.gallery_service_impl import GalleryServiceImpl
• from entity.artwork import Artwork
```

```
•     from datetime import datetime
•
•
•     def main():
•         service = GalleryServiceImpl("db_config.properties")
•
•
•         while True:
•             print("\nWELCOME TO VIRTUAL ART GALLERY")
•             print("=====")
•             print("1. Add Artwork")
•             print("2. Update Artwork")
•             print("3. Remove Artwork")
•             print("4. Get Artwork by ID")
•             print("5. Search Artworks")
•             print("6. Add to Favorites")
•             print("7. Remove from Favorites")
•             print("8. View Favorite Artworks")
•             print("9. Exit")
•             print("=====")
•
•
•             choice = input("Enter your choice: ")
•
•
•             if choice == '1':
•                 title = input("Enter title: ")
•                 desc = input("Enter description: ")
•                 date = input("Enter creation date (YYYY-MM-DD): ")
•                 medium = input("Enter medium: ")
•                 image = input("Enter image URL: ")
•                 artist_id = int(input("Enter artist ID: "))
•                 artwork = Artwork(0, title, desc, date, medium, image,
•                                   artist_id)
•                 if service.add_artwork(artwork):
•                     print("Artwork added successfully.")
•                 else:
•                     print("Failed to add artwork.")
•
•
•             elif choice == '2':
•                 artwork_id = int(input("Enter Artwork ID to update: "))
•                 title = input("New title: ")
•                 desc = input("New description: ")
•                 date = input("New creation date (YYYY-MM-DD): ")
•                 medium = input("New medium: ")
•                 image = input("New image URL: ")
•                 artist_id = int(input("New artist ID: "))
•                 updated = Artwork(artwork_id, title, desc, date, medium,
•                                   image, artist_id)
```

```
•         if service.update_artwork(updated):
•             print("Artwork updated successfully.")
•         else:
•             print("Failed to update artwork.")
•
•     elif choice == '3':
•         artwork_id = int(input("Enter Artwork ID to remove: "))
•         if service.remove_artwork(artwork_id):
•             print("Artwork removed successfully.")
•         else:
•             print("Artwork not found or not removed.")
•
•     elif choice == '4':
•         artwork_id = int(input("Enter Artwork ID to view: "))
•         artwork = service.get_artwork_by_id(artwork_id)
•         if artwork:
•             print(f"\nArtwork Details:\nTitle: {artwork.title}\nDescription: {artwork.description}\nDate: {artwork.creation_date}")
•         else:
•             print("Artwork not found.")
•
•     elif choice == '5':
•         keyword = input("Enter keyword to search: ")
•         results = service.search_artworks(keyword)
•         if results:
•             print(f"\nFound {len(results)} artworks:")
•             for a in results:
•                 print(f" - {a.title} ({a.medium})")
•         else:
•             print("No artworks found matching that keyword.")
•
•     elif choice == '6':
•         user_id = int(input("Enter User ID: "))
•         artwork_id = int(input("Enter Artwork ID to add to favorites: "))
•         if service.add_artwork_to_favorite(user_id, artwork_id):
•             print("Artwork added to favorites.")
•         else:
•             print("Could not add to favorites.")
•
•     elif choice == '7':
•         user_id = int(input("Enter User ID: "))
•         artwork_id = int(input("Enter Artwork ID to remove from favorites: "))
```

```
•             if service.remove_artwork_from_favorite(user_id,
• artwork_id):
•                 print("Artwork removed from favorites.")
•             else:
•                 print("Could not remove from favorites.")
•
•             elif choice == '8':
•                 user_id = int(input("Enter User ID: "))
•                 favs = service.get_user_favorite_artworks(user_id)
•                 if favs:
•                     print(f"\nFavorite Artworks ({len(favs)}):")
•                     for f in favs:
•                         print(f" - {f.title} by Artist ID
{f.artist_id}")
•                 else:
•                     print("No favorites found.")
•
•             elif choice == '9':
•                 print("Exiting... Goodbye.")
•                 break
•
•             else:
•                 print("Invalid choice. Please try again.")
•
•         if __name__ == "__main__":
•             main()
```

VirtualArtGallery

```
File Edit Selection View Go Run ... ← → ⌘ VirtualArtGallery ⌘ VirtualArtGallery ⌘ main_module.py X ⌘ ...
```

EXPLORER OPEN EDITORS VIRTUALARTGALLERY exception main

```
artwork_not_found_exception.py user_not_found_exception.py main_module.py main
.main > main_module.py > main
1 import sys
2 import os
3 sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
4
5 from dao.gallery_service_impl import GalleryServiceImpl
6 from entity.artwork import Artwork
7 from datetime import datetime
8
9 def main():
10     service = GalleryServiceImpl("db_config.properties")
11
12     while True:
13         print("\nWELCOME TO VIRTUAL ART GALLERY")
14         print("=====")
15         print("1. Add Artwork")
16         print("2. Update Artwork")
17         print("3. Remove Artwork")
18         print("4. Get Artwork by ID")
19         print("5. Search Artworks")
20         print("6. Add to Favorites")
21         print("7. Remove from Favorites")
22         print("8. View Favorite Artworks")
23         print("9. Exit")
24         print("====")
25
26         choice = input("Enter your choice: ")
27
28         if choice == '1':
29             title = input("Enter title: ")
30             desc = input("Enter description: ")
31             date = input("Enter creation date (YYYY-MM-DD): ")
32             medium = input("Enter medium: ")
33
34             date = input("Enter creation date (YYYY-MM-DD): ")
35             medium = input("Enter medium: ")
36             image = input("Enter image URL: ")
37             artist_id = int(input("Enter artist ID: "))
38             artwork = Artwork(0, title, desc, date, medium, image, artist_id)
39             if service.add_artwork(artwork):
40                 print("Artwork added successfully.")
41             else:
42                 print("Failed to add artwork.")
43
44             elif choice == '2':
45                 artwork_id = int(input("Enter Artwork ID to update: "))
46                 title = input("New title: ")
47                 desc = input("New description: ")
48                 date = input("New creation date (YYYY-MM-DD): ")
49                 medium = input("New medium: ")
50                 image = input("New image URL: ")
51                 artist_id = int(input("New artist ID: "))
52                 updated = Artwork(artwork_id, title, desc, date, medium, image, artist_id)
53                 if service.update_artwork(updated):
54                     print("Artwork updated successfully.")
55                 else:
56                     print("Failed to update artwork.")
57
58             elif choice == '3':
59                 artwork_id = int(input("Enter Artwork ID to remove: "))
60                 if service.remove_artwork(artwork_id):
61                     print("Artwork removed successfully.")
62                 else:
63                     print("Artwork not found or not removed.")
```

Ln 18, Col 38 Spaces: 4 UTF-8 CRLF {} Python 3.11.5 (venv: venv) ⌘

VirtualArtGallery

```
File Edit Selection View Go Run ... ← → ⌘ VirtualArtGallery ⌘ VirtualArtGallery ⌘ main_module.py X ⌘ ...
```

EXPLORER OPEN EDITORS VIRTUALARTGALLERY exception main

```
artwork_not_found_exception.py user_not_found_exception.py main_module.py main
.main > main_module.py > main
1 import sys
2 import os
3 sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
4
5 from dao.gallery_service_impl import GalleryServiceImpl
6 from entity.artwork import Artwork
7 from datetime import datetime
8
9 def main():
10     service = GalleryServiceImpl("db_config.properties")
11
12     while True:
13         print("\nWELCOME TO VIRTUAL ART GALLERY")
14         print("=====")
15         print("1. Add Artwork")
16         print("2. Update Artwork")
17         print("3. Remove Artwork")
18         print("4. Get Artwork by ID")
19         print("5. Search Artworks")
20         print("6. Add to Favorites")
21         print("7. Remove from Favorites")
22         print("8. View Favorite Artworks")
23         print("9. Exit")
24         print("====")
25
26         choice = input("Enter your choice: ")
27
28         if choice == '1':
29             title = input("Enter title: ")
30             desc = input("Enter description: ")
31             date = input("Enter creation date (YYYY-MM-DD): ")
32             medium = input("Enter medium: ")
33
34             date = input("Enter creation date (YYYY-MM-DD): ")
35             medium = input("Enter medium: ")
36             image = input("Enter image URL: ")
37             artist_id = int(input("Enter artist ID: "))
38             artwork = Artwork(0, title, desc, date, medium, image, artist_id)
39             if service.add_artwork(artwork):
40                 print("Artwork added successfully.")
41             else:
42                 print("Failed to add artwork.")
43
44             elif choice == '2':
45                 artwork_id = int(input("Enter Artwork ID to update: "))
46                 title = input("New title: ")
47                 desc = input("New description: ")
48                 date = input("New creation date (YYYY-MM-DD): ")
49                 medium = input("New medium: ")
50                 image = input("New image URL: ")
51                 artist_id = int(input("New artist ID: "))
52                 updated = Artwork(artwork_id, title, desc, date, medium, image, artist_id)
53                 if service.update_artwork(updated):
54                     print("Artwork updated successfully.")
55                 else:
56                     print("Failed to update artwork.")
57
58             elif choice == '3':
59                 artwork_id = int(input("Enter Artwork ID to remove: "))
60                 if service.remove_artwork(artwork_id):
61                     print("Artwork removed successfully.")
62                 else:
63                     print("Artwork not found or not removed.")
```

Ln 18, Col 38 Spaces: 4 UTF-8 CRLF {} Python 3.11.5 (venv: venv) ⌘

```
File Edit Selection View Go Run ... ← → ⌘ VirtualArtGallery
EXPLORER OPEN EDITORS
artwork_not_found_exception.py user_not_found_exception.py main_module.py X
main > main_module.py > main
main():
    elif choice == '4':
        artwork_id = int(input("Enter Artwork ID to view: "))
        artwork = service.get_artwork_by_id(artwork_id)
        if artwork:
            print(f"\nArtwork Details:\nTitle: {artwork.title}\nDescription: {artwork.description}\nDate: {artwork.date}")
        else:
            print("Artwork not found.")

    elif choice == '5':
        keyword = input("Enter keyword to search: ")
        results = service.search_artworks(keyword)
        if results:
            print(f"\nFound {len(results)} artworks:")
            for a in results:
                print(f" - {a.title} ({a.medium})")
        else:
            print("No artworks found matching that keyword.")

    elif choice == '6':
        user_id = int(input("Enter User ID: "))
        artwork_id = int(input("Enter Artwork ID to add to favorites: "))
        if service.add_artwork_to_favorite(user_id, artwork_id):
            print("Artwork added to favorites.")
        else:
            print("Could not add to favorites.")

    elif choice == '7':
        user_id = int(input("Enter User ID: "))
        artwork_id = int(input("Enter Artwork ID to remove from favorites: "))
        if service.remove_artwork_from_favorite(user_id, artwork_id):
            print("Artwork removed from favorites.")
        else:
            print("Could not remove from favorites.")

    elif choice == '8':
        user_id = int(input("Enter User ID: "))
        favs = service.get_user_favorite_artworks(user_id)
        if favs:
            print(f"\nFavorite Artworks ({len(favs)}):")
            for f in favs:
                print(f" - {f.title} by Artist ID {f.artist_id}")
        else:
            print("No favorites found.")

    elif choice == '9':
        print("Exiting... Goodbye.")
        break

    else:
        print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

Ln 18, Col 38 Spaces:4 UTF-8 CRLF {} Python 3.11.5 (venv: venv) Q

```
File Edit Selection View Go Run ... ← → ⌘ VirtualArtGallery
EXPLORER OPEN EDITORS
artwork_not_found_exception.py user_not_found_exception.py main_module.py X
main > main_module.py > main
main():
    elif choice == '4':
        artwork_id = int(input("Enter Artwork ID to view: "))
        artwork = service.get_artwork_by_id(artwork_id)
        if artwork:
            print(f"\nArtwork Details:\nTitle: {artwork.title}\nDescription: {artwork.description}\nDate: {artwork.date}")
        else:
            print("Artwork not found.")

    elif choice == '5':
        keyword = input("Enter keyword to search: ")
        results = service.search_artworks(keyword)
        if results:
            print(f"\nFound {len(results)} artworks:")
            for a in results:
                print(f" - {a.title} ({a.medium})")
        else:
            print("No artworks found matching that keyword.")

    elif choice == '6':
        user_id = int(input("Enter User ID: "))
        artwork_id = int(input("Enter Artwork ID to add to favorites: "))
        if service.add_artwork_to_favorite(user_id, artwork_id):
            print("Artwork added to favorites.")
        else:
            print("Could not add to favorites.")

    elif choice == '7':
        user_id = int(input("Enter User ID: "))
        artwork_id = int(input("Enter Artwork ID to remove from favorites: "))
        if service.remove_artwork_from_favorite(user_id, artwork_id):
            print("Artwork removed from favorites.")
        else:
            print("Could not remove from favorites.")

    elif choice == '8':
        user_id = int(input("Enter User ID: "))
        favs = service.get_user_favorite_artworks(user_id)
        if favs:
            print(f"\nFavorite Artworks ({len(favs)}):")
            for f in favs:
                print(f" - {f.title} by Artist ID {f.artist_id}")
        else:
            print("No favorites found.")

    elif choice == '9':
        print("Exiting... Goodbye.")
        break

    else:
        print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

Ln 18, Col 38 Spaces:4 UTF-8 CRLF {} Python 3.11.5 (venv: venv) Q

6. tests/ - Unit Testing:

description:

The tests folder contains unit test files written using Python's unittest framework. These validate the functionality of service methods such as retrieving artworks, handling missing records, and adding favorites.

Include Screenshots of:

- test_gallery.py
- Terminal output of test results

```
• import unittest
• from dao.gallery_service_impl import GalleryServiceImpl
• from entity.artwork import Artwork
•
• class TestGalleryService(unittest.TestCase):
•     @classmethod
•     def setUpClass(cls):
•         cls.service = GalleryServiceImpl("db_config.properties")
•
•     def test_add_artwork(self):
•         artwork = Artwork(
•             0, "Test Title", "Test Description", "2024-01-01",
•             "Oil", "test.jpg", 1
•         )
•         result = self.service.add_artwork(artwork)
•         self.assertTrue(result)
•
•     def test_get_artwork_by_id(self):
•         artwork = self.service.get_artwork_by_id(1)
•         self.assertIsNotNone(artwork)
•         self.assertEqual(artwork.artwork_id, 1)
•
•     def test_search_artworks(self):
•         results = self.service.search_artworks("Starry")
•         self.assertIsInstance(results, list)
•         self.assertGreaterEqual(len(results), 1)
•
• if __name__ == '__main__':
•     unittest.main()
```

The screenshot shows a Python development environment with the following details:

- File Explorer:** Shows files like `artwork_not_found_exception.py`, `user_not_found_exception.py`, `main_module.py`, and `test_gallery.py`.
- Virtual Environment:** `VIRTUALARTGALLERY` is selected.
- Code Editor:** The file `test_gallery.py` is open, containing test cases for a `GalleryService`. The code includes imports for `unittest`, `GalleryServiceImpl`, and `Artwork`. It defines three test methods: `test_add_artwork`, `test_get_artwork_by_id`, and `test_search_artworks`. The `__main__` block at the bottom runs `unittest.main()`.
- Status Bar:** Shows "Ln 1, Col 1", "Spaces: 4", "UTF-8", "CRLF", "Python", "3.11.5 (venv: venv)", and a terminal icon.

7. Db connectivity check:

Db_connection.py :

```
from util.db_conn_util import get_connection
```

```
def test_connection():

    conn = get_connection("db_config.properties")

    if conn:

        print(" Database connection successful!")

        conn.close()

    else:

        print(" Failed to connect to the database.")

if __name__ == "__main__":
    test_connection()
```

```

File Edit Selection View Go Run ... ← → ⌘ VirtualArtGallery
OPEN EDITORS py test_gallery.py art_gallery_schema.sql db_config.properties requirements.txt test_db_connection.py ...
EXPLORER artwork_not_found_excepti... user_not_found_exception.p... main_module.py main test_gallery.py tests art_gallery_schema.sql db_config.properties requirements.txt
VIRTUALARTGALLERY > .vscode > dao > entity > exception > main > tests > util > venv art_gallery_schema.sql db_config.properties eer_diagram.mwb README.md requirements.txt test_db_connection.py
OUTLINE > TIMELINE
POWERLEVEL9K_VCS_MODIFIED_GAUGE 0 △ 0 Java: Ready
OUTPUT DEBUG CONSOLE PROBLEMS TERMINAL PORTS
=====
Enter your choice: []
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF () Python 3.11.5 (venv: venv) Q

```

7. Output Evidence (MySQL + CLI)

1. Add the artwork

```

File Edit Selection View Go Run ... ← → ⌘ VirtualArtGallery
OPEN EDITORS main_module.py main
EXPLORER < main_module.py > main
VIRTUALARTGALLERY > .vscode > dao > entity > exception > main > tests > util > venv art_gallery_schema.sql db_config.properties eer_diagram.mwb README.md requirements.txt test_db_connection.py
OUTLINE > TIMELINE
POWERLEVEL9K_VCS_MODIFIED_GAUGE 0 △ 0 Java: Ready
OUTPUT DEBUG CONSOLE PROBLEMS TERMINAL PORTS
=====
Error while adding artwork: 1452 (23000): Cannot add or update a child row: a foreign key constraint fails ('art_gallery`.`artwork', CONSTRAINT `artwork_ibfk_1` FOREIGN KEY (`ArtistID`) REFERENCES `artist` (`ArtistID`))
Failed to add artwork.

WELCOME TO VIRTUAL ART GALLERY
=====
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add to Favorites
7. Remove from Favorites
8. View Favorite Artworks
9. Exit
=====
Enter your choice: 1
Enter title: Starry Night
Enter description: racers
Enter creation date (YYYY-MM-DD): 2025-06-08
Enter medium: Oil on canvas
Enter image URL: starry.jpg
Enter artist ID: 3
Artwork added successfully.
Ln 18, Col 38 Spaces: 4 UTF-8 CRLF () Python 3.11.5 (venv: venv) Q

```

2. update the artwork

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following text:

```
WELCOME TO VIRTUAL ART GALLERY
=====
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add to Favorites
7. Remove from Favorites
8. View Favorite Artworks
9. Exit
=====
Enter your choice: 2
Enter Artwork ID to update: 2
New title: jersycow
New description: cow painting
New creation date (YYYY-MM-DD): 2025-09-08
New medium: oil on canva
New image URL: cow.jpg
New artist ID: 6
Artwork updated successfully.
```

3. remove artwork

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following text:

```
WELCOME TO VIRTUAL ART GALLERY
=====
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add to Favorites
7. Remove from Favorites
8. View Favorite Artworks
9. Exit
=====
Enter your choice: 3
Enter Artwork ID to remove: 4
Artwork removed successfully.
```

4. get artworks by id

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following text:

```
Enter Artwork ID to remove: 4
Artwork removed successfully.

WELCOME TO VIRTUAL ART GALLERY
=====
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add to Favorites
7. Remove from Favorites
8. View Favorite Artworks
9. Exit
=====
Enter your choice: 4
Enter Artwork ID to view: 5

Artwork Details:
Title: Starry Night
Description: Van Gogh's greatest work.
Date: 1889-06-05
```

5. add to favourites

The terminal window shows the directory structure of a Python project. The current working directory is 'main'. Inside 'main', there are files '_init_.py' and 'main_module.py', which are highlighted. Other files like 'README.md' and 'requirements.txt' are also listed. To the right of the terminal, the output of the application is displayed:

```
Description: van gogh's greatest work.
Date: 1889-06-05

WELCOME TO VIRTUAL ART GALLERY
=====
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add to Favorites
7. Remove from Favorites
8. View Favorite Artworks
9. Exit
=====

Enter your choice: 6
Enter User ID: 5
Enter Artwork ID to add to favorites: 6
Artwork added to favorites.
```

6. search artworks

The terminal window shows the directory structure of a Python project. The current working directory is 'main'. Inside 'main', there are files '_init_.py' and 'main_module.py', which are highlighted. Other files like 'README.md' and 'requirements.txt' are also listed. To the right of the terminal, the output of the application is displayed:

```
All artwork added to favorites.

WELCOME TO VIRTUAL ART GALLERY
=====
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add to Favorites
7. Remove from Favorites
8. View Favorite Artworks
9. Exit
=====

Enter your choice: 5
Enter keyword to search: starry

Found 5 artworks:
- Starry Night (Oil on canvas)
```

7. remove from favourites

The terminal window shows the directory structure of a Python project. The current working directory is 'main'. Inside 'main', there are files '_init_.py' and 'main_module.py', which are highlighted. Other files like 'README.md' and 'requirements.txt' are also listed. To the right of the terminal, the output of the application is displayed:

```
WELCOME TO VIRTUAL ART GALLERY
=====
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add to Favorites
7. Remove from Favorites
8. View Favorite Artworks
9. Exit
=====

Enter your choice: 7
Enter User ID: 5
Enter Artwork ID to remove from favorites: 6
Artwork removed from favorites.
```

8. view favourite artworks

```
> .vscode
> dao
> entity
> exception
< main
  + _init_.py
  + main_module.py
> tests
> util
> venv
art_gallery_schema.sql
db_config.properties
eer diagram.mwb
 README.md
requirements.txt

=====
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add to Favorites
7. Remove from Favorites
8. View Favorite Artworks
9. Exit
=====

Enter your choice: 8
Enter User ID: 5
=====

Favorite Artworks (2):
  - Guernica by Artist ID 2
  - Water Lilies by Artist ID 5
```

9. exit

```
③

> OUTLINE
> TIMELINE
⊗ 0 ⚠ 0 ⚡ Java: Ready

❶ README.md
⠁ requirements.txt
❷ test_db_connection.py

WELCOME TO VIRTUAL ART GALLERY
=====
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add to Favorites
7. Remove from Favorites
8. View Favorite Artworks
9. Exit
=====
Enter your choice: 9
Exiting... Goodbye.
PS C:\Users\machi\Documents\VirtualArtGallery>
```

Testing Summary

Tested artwork operations, user favorites, exception triggers, and SQL consistency. Unit tests were written for major DAO functions.

Screenshots to Include

- CLI Menu
 - Adding artwork
 - Searching artworks
 - Adding to favorites
 - Viewing favorites

- Exceptions (artwork/user not found)

Conclusion

The Virtual Art Gallery provides a functional simulation of an online art platform. With real-world entity mapping, structured modules, and MySQL integration, this project demonstrates the practical application of OOP and database-driven programming.

Team Member Contributions

- Dhamini Machireddy: Artwork module, exception handling, testing, DB schema
- Elekkiya: User favorites, gallery logic, search module, unit testing