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.

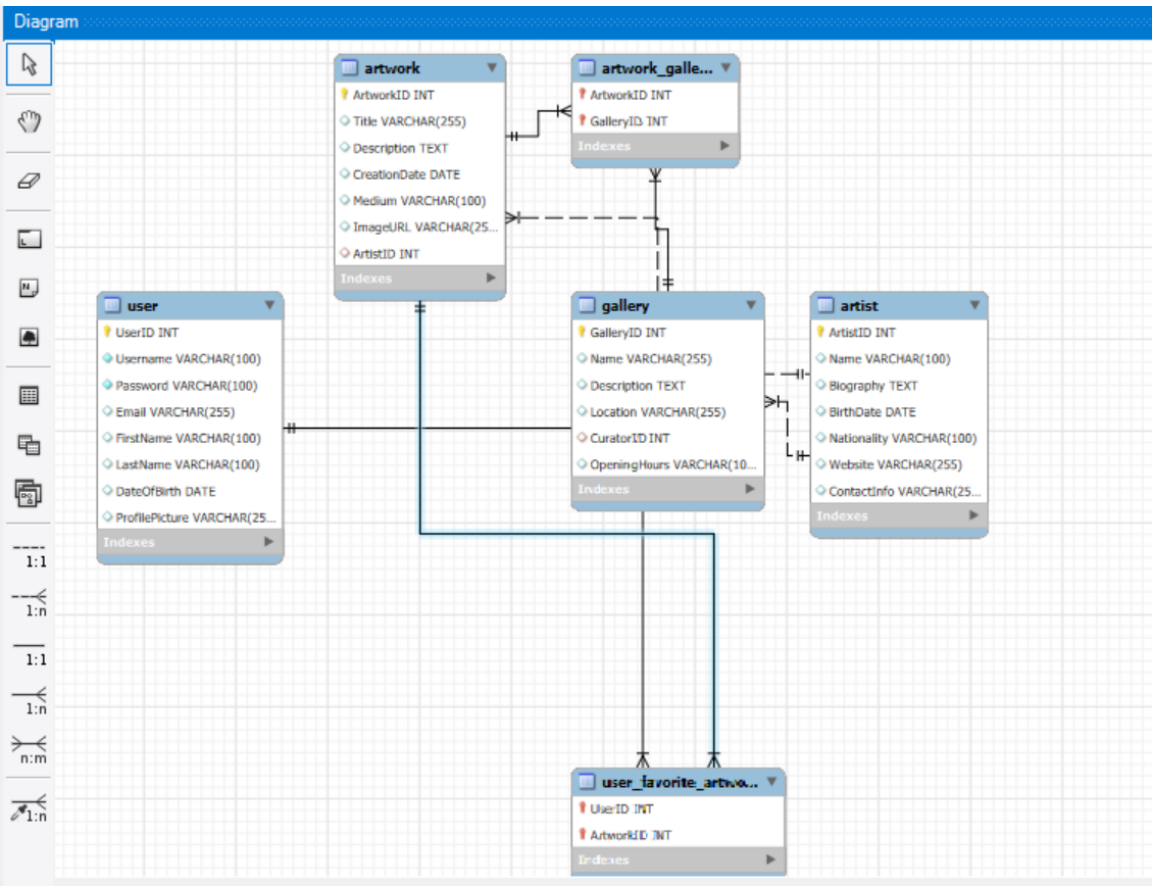
The project models key entities like Artwork, User, Artist, and Gallery using Python classes, which organize the data and actions related to each entity. The main functionalities include adding, viewing, and managing artwork details through console inputs, with all data stored securely in a MySQL database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



Relationships:
- Artist → Artwork: One-to-Many
- Artist → Gallery: One-to-Many
- Artwork ↔ Gallery: Many-to-Many
- User ↔ Favorite Artworks: Many-to-Many

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 Dalí', '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),

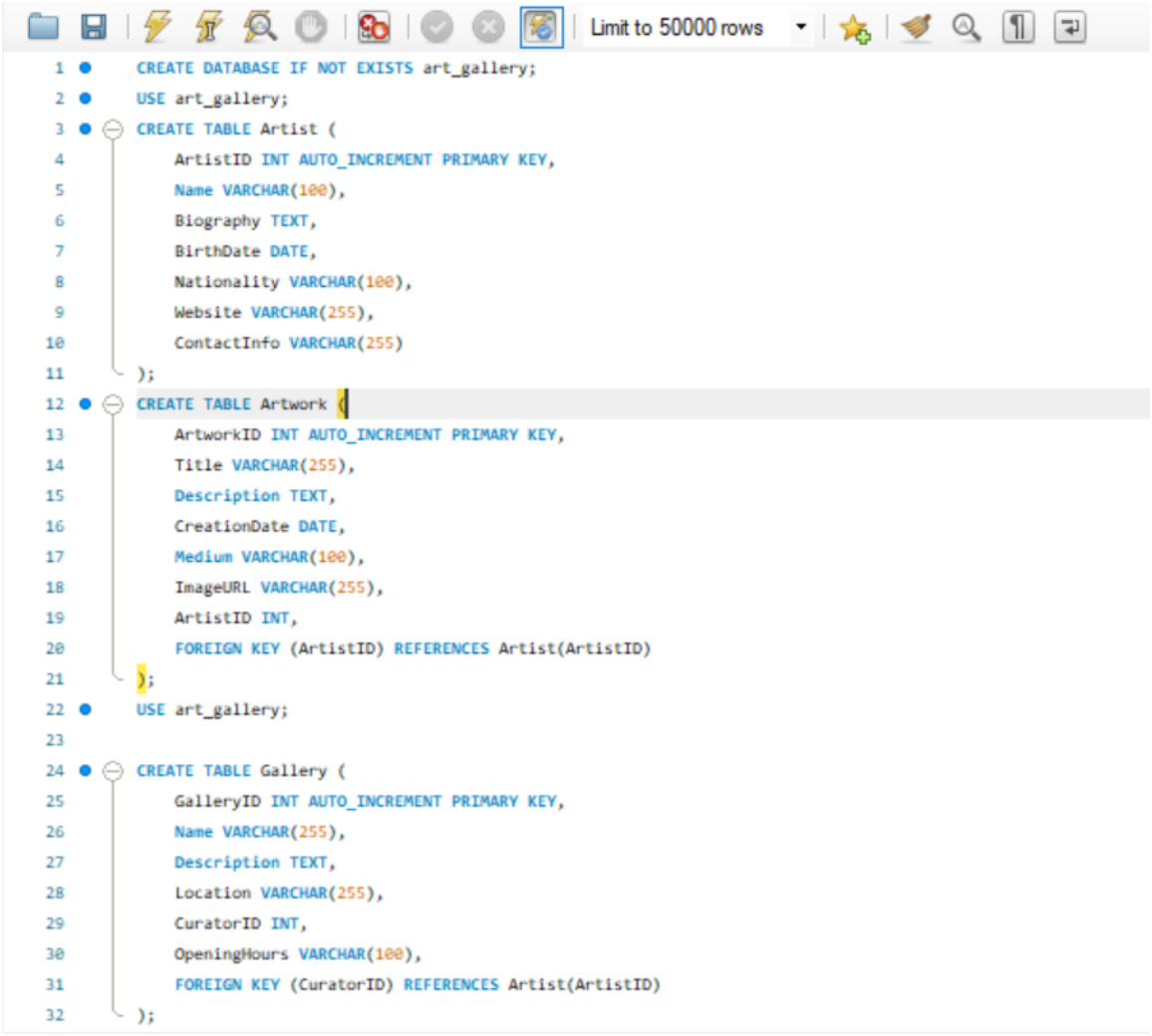
(5, 9),

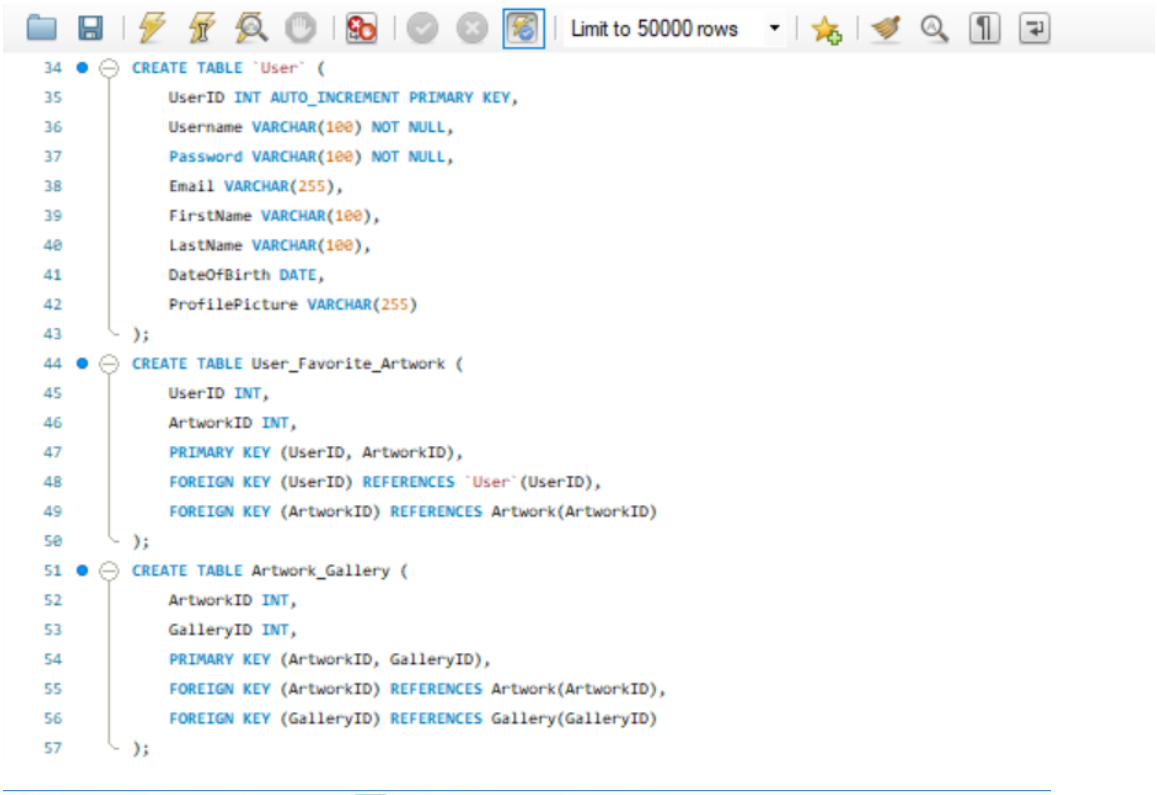
(6, 10),

(7, 11),

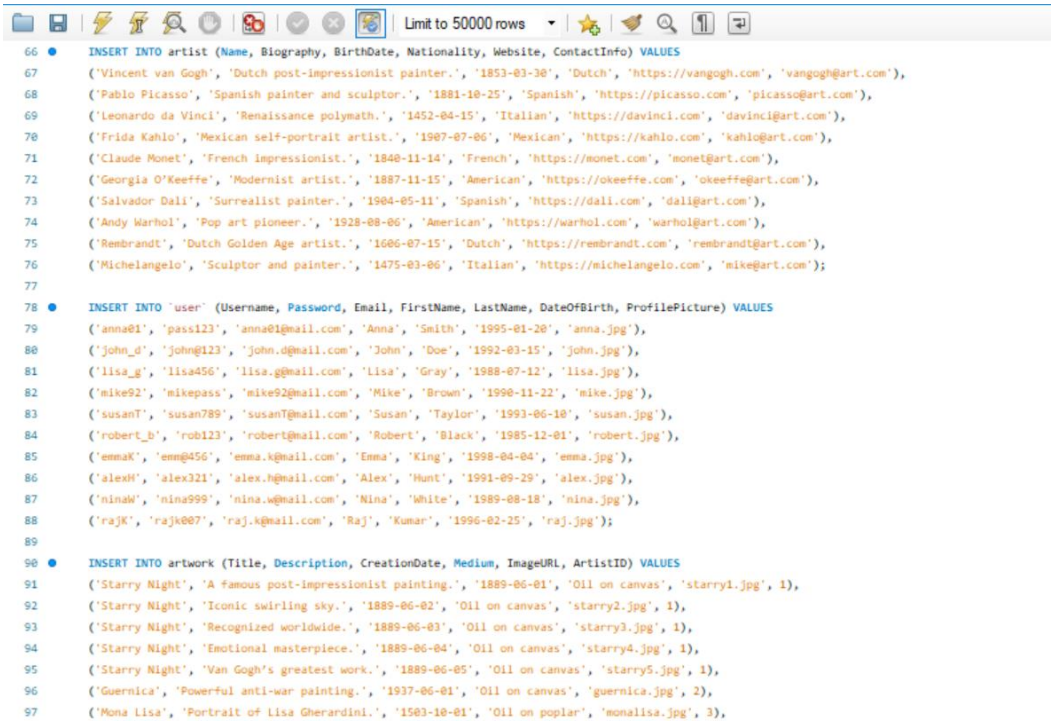
(8, 12),

(9, 13),





```sql
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)
);
```



```sql
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),
```

```sql
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),
```

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})"
```

```python
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})"
```

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

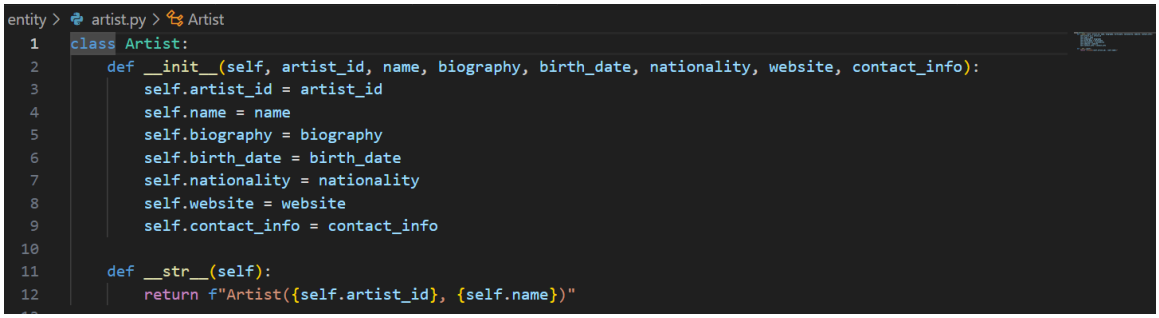
self.nationality = nationality

self.website = website

self.contact_info = contact_info

def __str__(self):

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



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

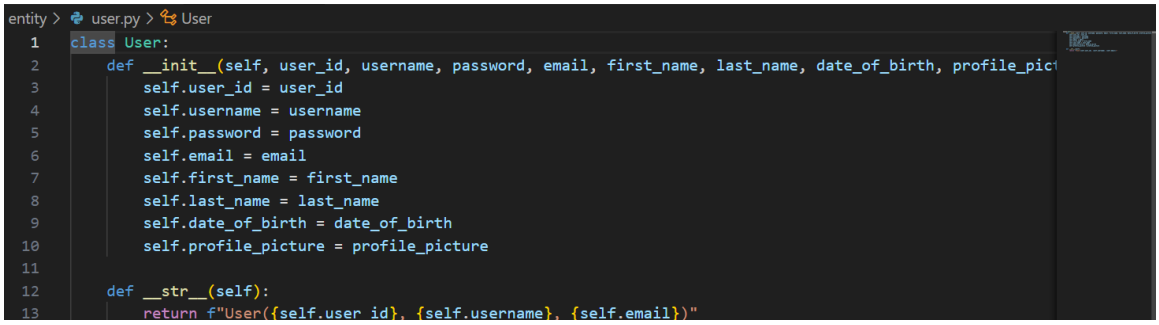
self.last_name = last_name

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})"



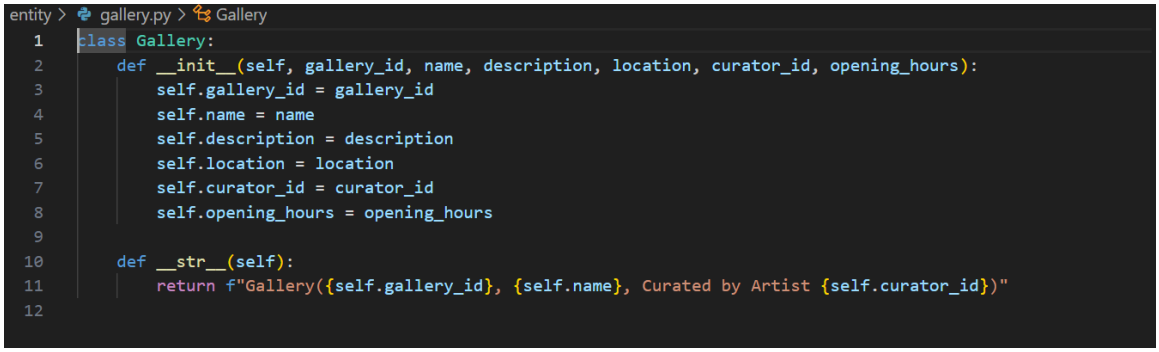
Gallery :

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})"
```



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})"
```

ArtworkGallery:

class ArtworkGallery:

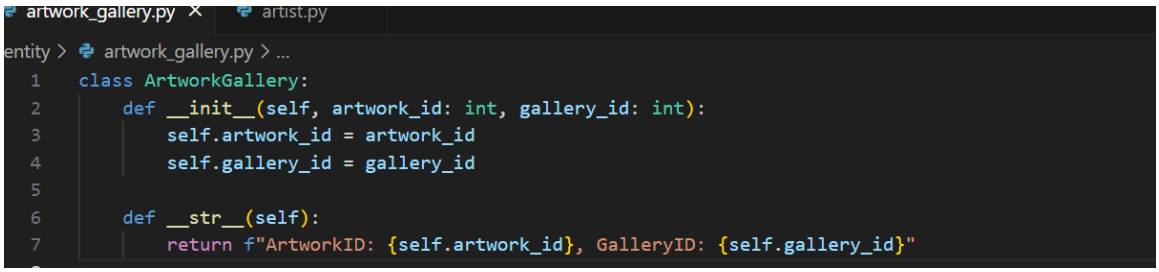
def __init__(self, artwork_id: int, gallery_id: int):

self.artwork_id = artwork_id

self.gallery_id = gallery_id

def __str__(self):

return f"ArtworkID: {self.artwork_id}, GalleryID: {self.gallery_id}"



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.

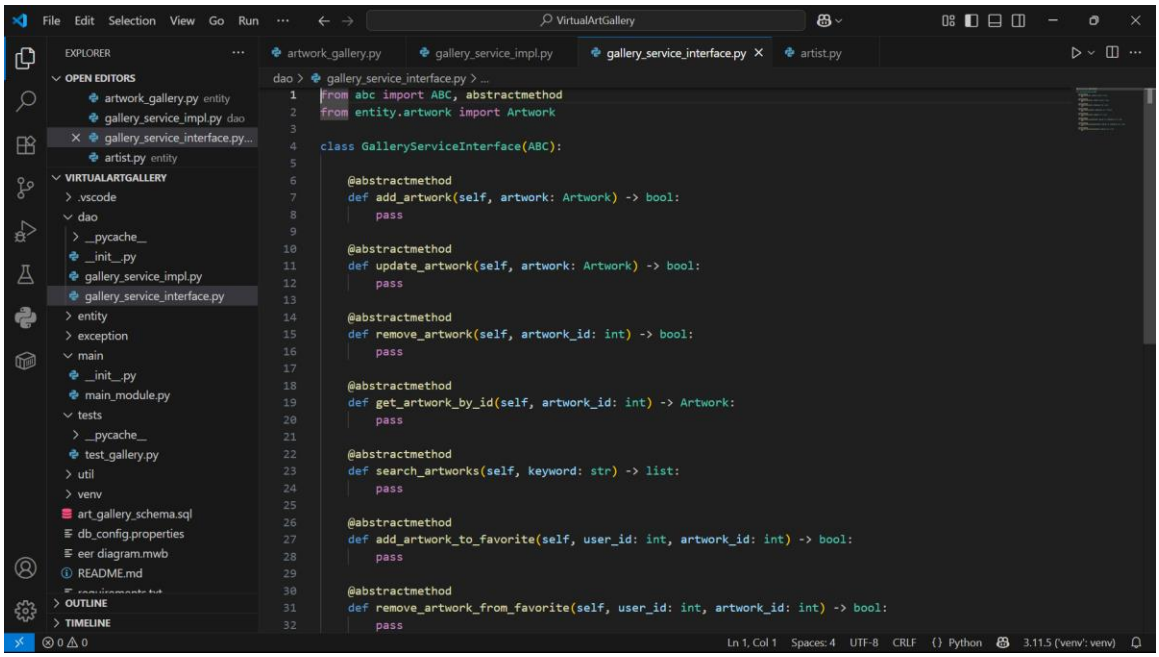
Include Screenshots of:

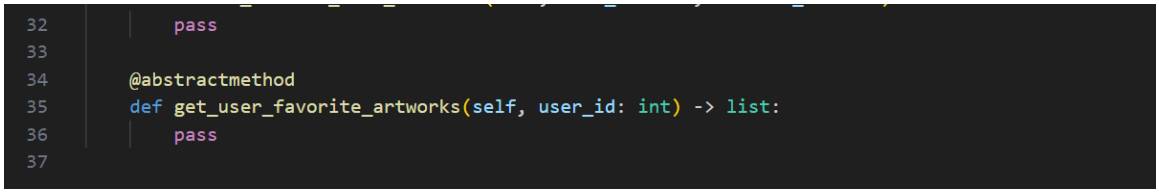
- Interface with all method signatures
- Implementation showing logic for:
    - Adding artworks
    - Searching artworks
    - Adding to favorites
    - Raising exceptions

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
```





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']
            )
```

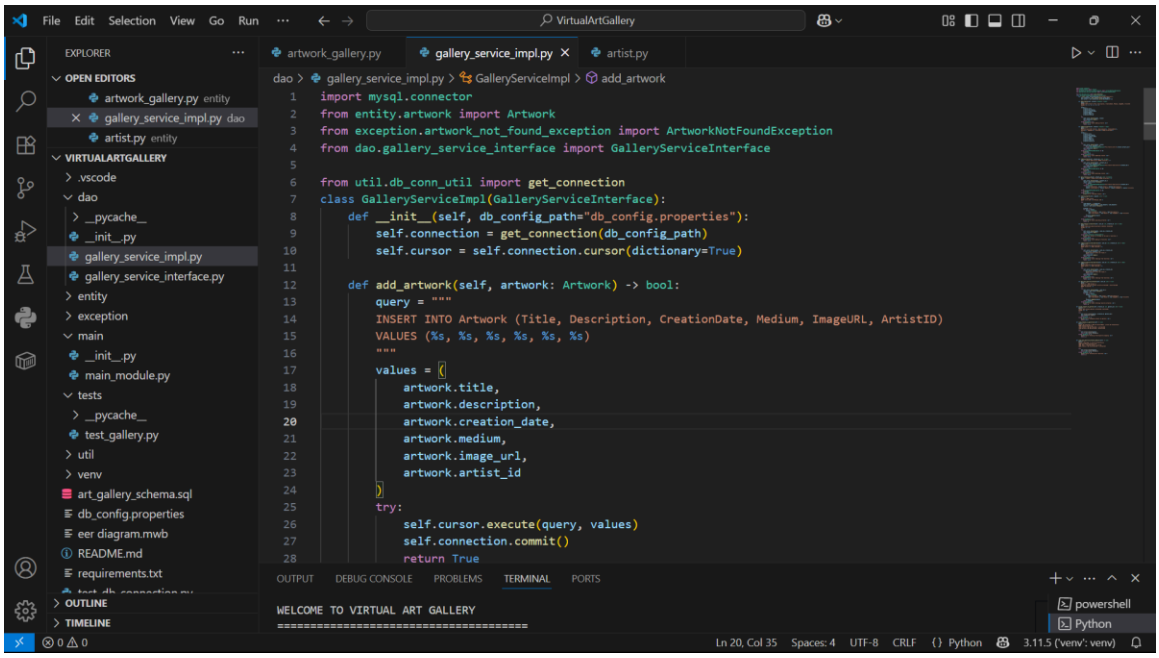
```
            artworks.append(artwork)
        return artworks

    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 = """
```

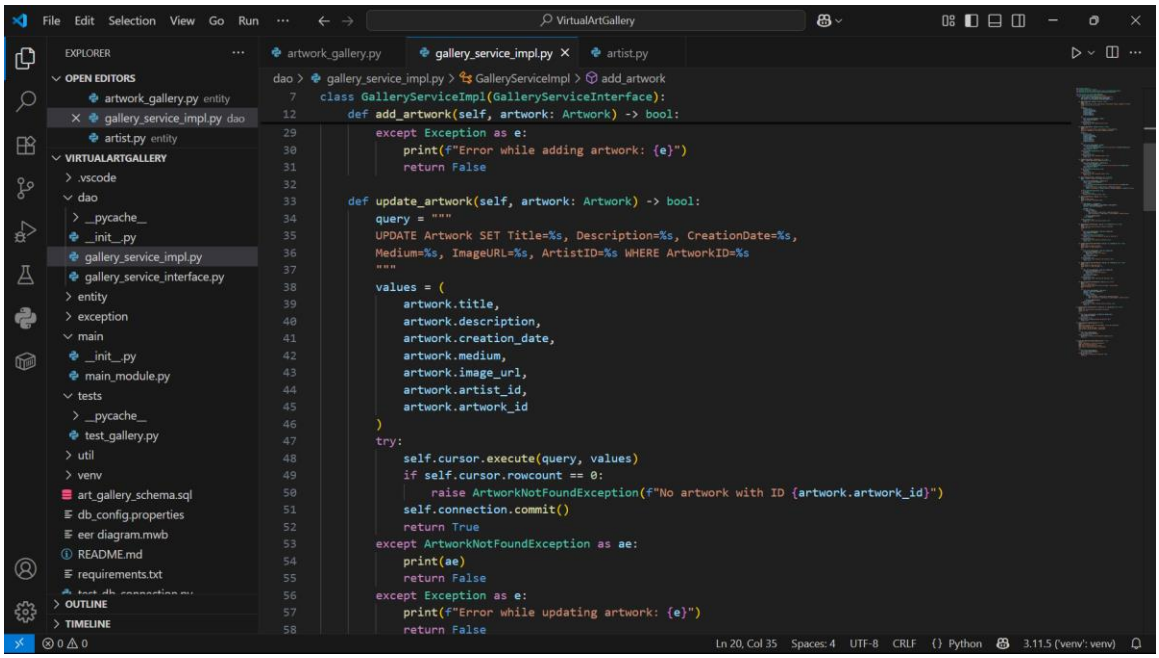
```
    DELETE FROM User_Favorite_Artwork
    WHERE UserID = %s AND ArtworkID = %s
    """
    try:
        self.cursor.execute(query, (user_id, artwork_id))
        if self.cursor.rowcount == 0:
            print("Favorite entry not found.")
            return False
        self.connection.commit()
        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()
```

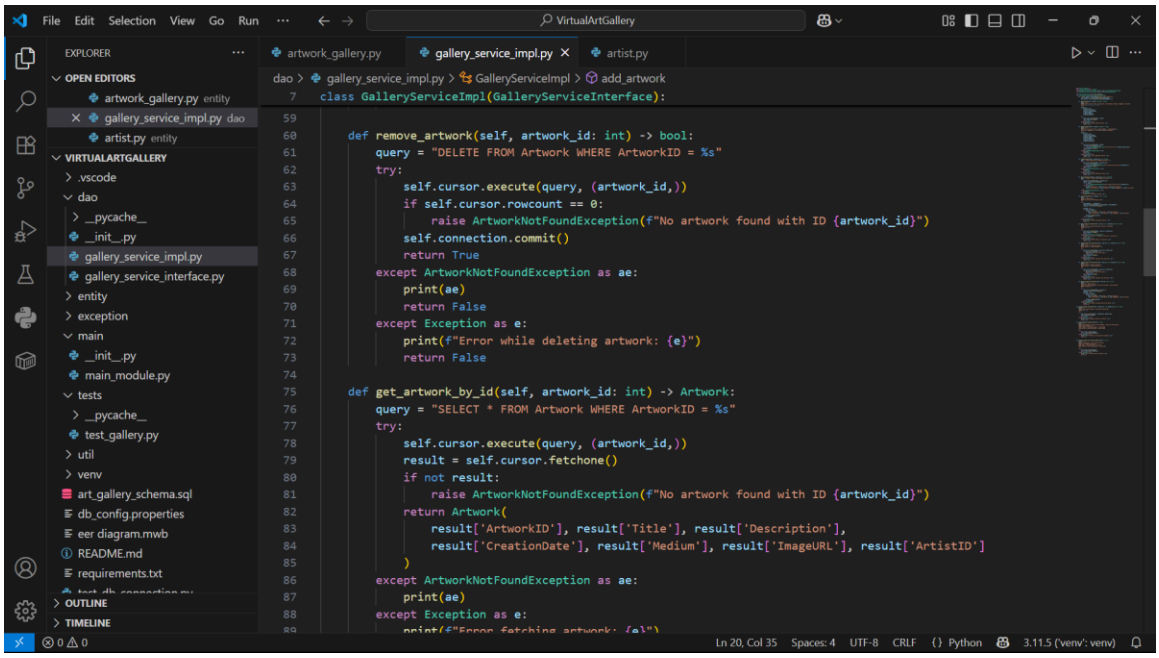
```
        return True
    except Exception as e:
        print(f"Error while removing from favorites: {e}")
        return False

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)
        return artworks
    except Exception as e:
        print(f"Error while fetching favorite artworks: {e}")
        return []
```

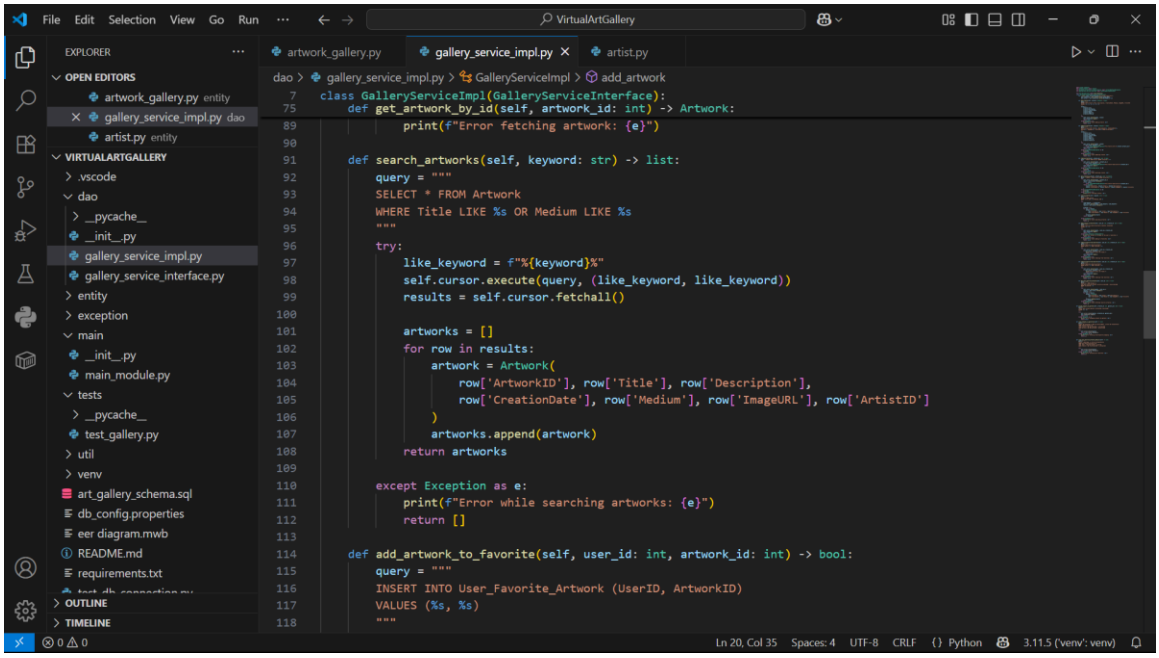
```
def assign_artwork_to_gallery(self, artwork_id: int, gallery_id: int) -> bool:
    query = """
    INSERT INTO Artwork_Gallery (ArtworkID, GalleryID)
    VALUES (%s, %s)
    """
    try:
        self.cursor.execute(query, (artwork_id, gallery_id))
        self.connection.commit()
        return True
    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:
```

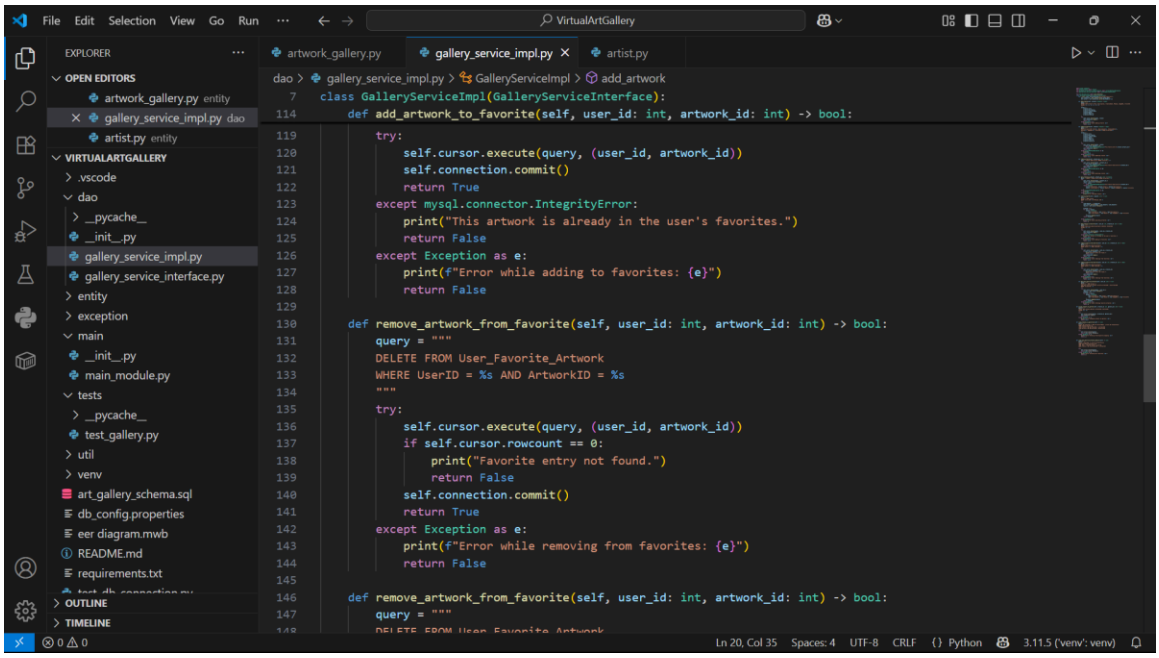
```
        print(f"Error fetching artwork-gallery mapping: {e}")
        return []


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 []
```

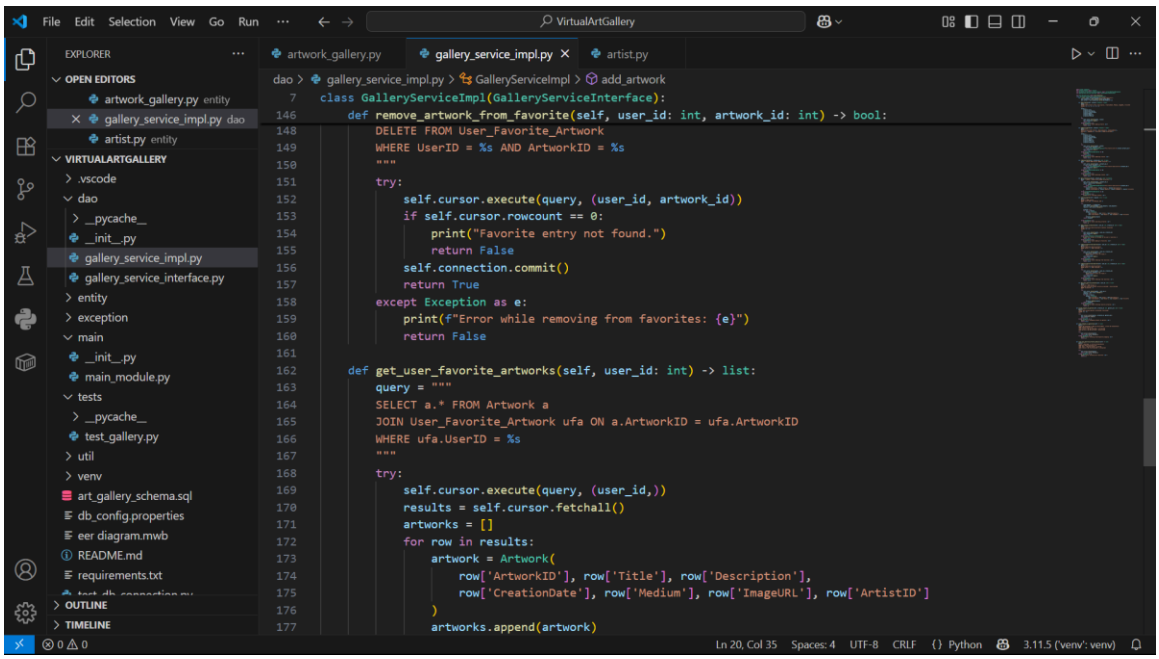

File Edit Selection View Go Run
VirtualArtGallery
EXPLORER
OPEN EDITORS
artwork_gallery.py entity
gallery_service_impl.py dao
artist.py entity
VIRTUALARTGALLERY
.vscode
dao
__pycache__
__init__.py
gallery_service_impl.py
gallery_service_interface.py
entity
exception
main
__init__.py
main_module.py
tests
__pycache__
test_gallery.py
util
venv
art_gallery_schema.sql
db_config.properties
eer diagram.mwb
README.md
requirements.txt
OUTLINE
TIMELINE
artwork_gallery.py
gallery_service_impl.py
artist.py
dao > gallery_service_impl.py > GalleryServiceImpl > add_artwork
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
OUTPUT DEBUG CONSOLE PROBLEMS TERMINAL PORTS
WELCOME TO VIRTUAL ART GALLERY
powershell
Python
Ln 20, Col 35 Spaces: 4 UTF-8 CRLF {} Python 3.11.5 ('venv': venv)

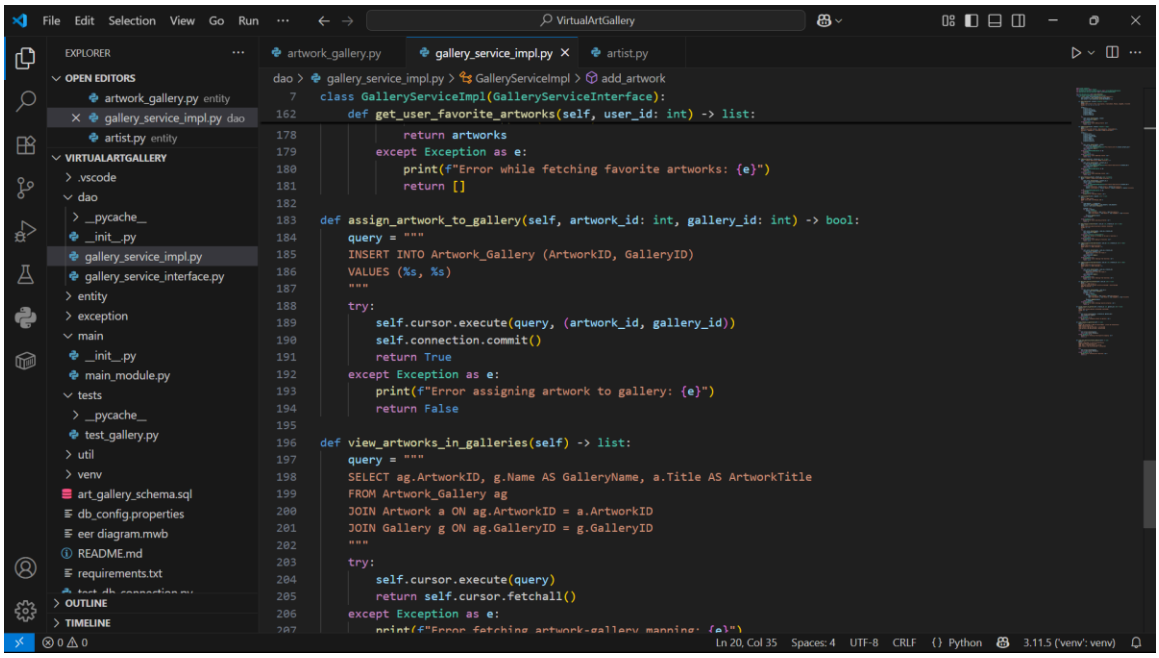

File Edit Selection View Go Run
VirtualArtGallery
EXPLORER
OPEN EDITORS
artwork_gallery.py entity
gallery_service_impl.py dao
artist.py entity
VIRTUALARTGALLERY
.vscode
dao
__pycache__
__init__.py
gallery_service_impl.py
gallery_service_interface.py
entity
exception
main
__init__.py
main_module.py
tests
__pycache__
test_gallery.py
util
venv
art_gallery_schema.sql
db_config.properties
eer diagram.mwb
README.md
requirements.txt
OUTLINE
TIMELINE
artwork_gallery.py
gallery_service_impl.py
artist.py
dao > gallery_service_impl.py > GalleryServiceImpl > add_artwork
class GalleryServiceImpl(GalleryServiceInterface):
def add_artwork(self, artwork: Artwork) -> bool:
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
Ln 20, Col 35 Spaces: 4 UTF-8 CRLF {} Python 3.11.5 ('venv': venv)

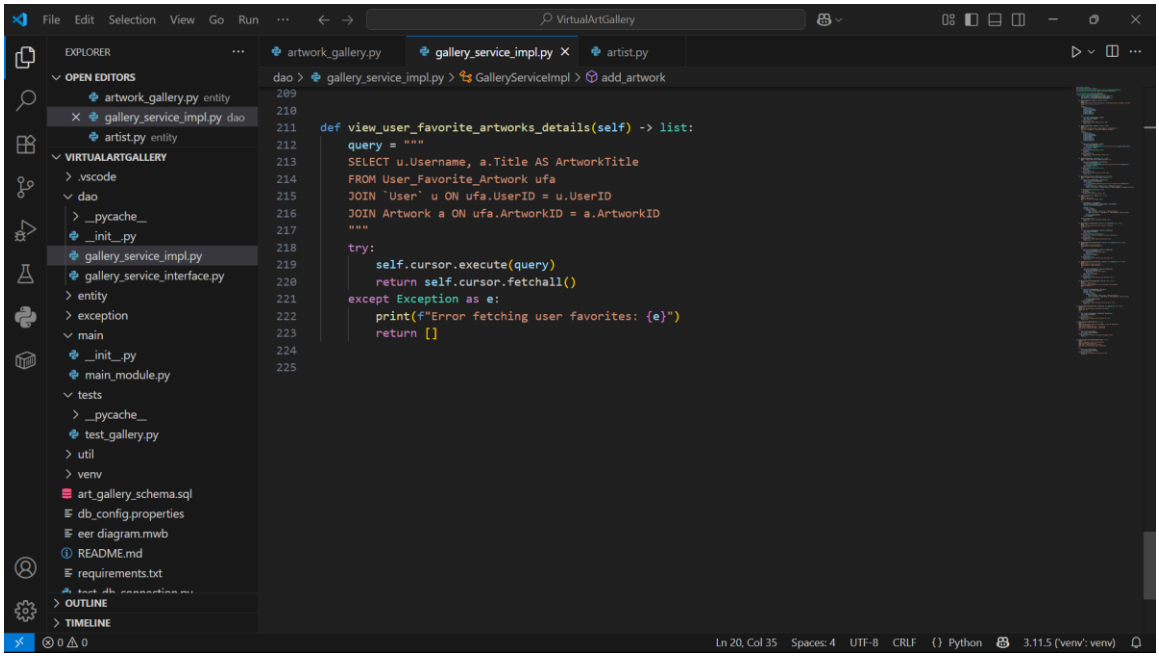

File Edit Selection View Go Run
VirtualArtGallery
EXPLORER
OPEN EDITORS
artwork_gallery.py entity
gallery_service_impl.py dao
artist.py entity
VIRTUALARTGALLERY
.vscode
dao
__pycache__
__init__.py
gallery_service_impl.py
gallery_service_interface.py
entity
exception
main
__init__.py
main_module.py
tests
__pycache__
test_gallery.py
util
venv
art_gallery_schema.sql
db_config.properties
eer diagram.mwb
README.md
requirements.txt
OUTLINE
TIMELINE
artwork_gallery.py
gallery_service_impl.py
artist.py
dao > gallery_service_impl.py > GalleryServiceImpl > add_artwork
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(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:
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
artwork_gallery.py entity
gallery_service_impl.py dao
artist.py entity
VIRTUALARTGALLERY
.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
OUTLINE
TIMELINE
artwork_gallery.py
gallery_service_impl.py
artist.py
dao > gallery_service_impl.py > GalleryServiceImpl > add_artwork
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 %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)
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)
"""
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
artwork_gallery.py entity
gallery_service_impl.py dao
artist.py entity
VIRTUALARTGALLERY
.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
OUTLINE
TIMELINE
artwork_gallery.py
gallery_service_impl.py
artist.py
dao > gallery_service_impl.py > GalleryServiceImpl > add_artwork
class GalleryServiceImpl(GalleryServiceInterface):
def add_artwork_to_favorite(self, user_id: int, artwork_id: int) -> bool:
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 = """
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
artwork_gallery.py entity
gallery_service_impl.py dao
artist.py entity
VIRTUALARTGALLERY
.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
OUTLINE
TIMELINE
artwork_gallery.py
gallery_service_impl.py
artist.py
dao > gallery_service_impl.py > GalleryServiceImpl > add_artwork
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 = %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)
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
artwork_gallery.py entity
gallery_service_impl.py dao
artist.py entity
VIRTUALARTGALLERY
.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
OUTLINE
TIMELINE
artwork_gallery.py
gallery_service_impl.py
artist.py
dao > gallery_service_impl.py > GalleryServiceImpl > add_artwork
class GalleryServiceImpl(GalleryServiceInterface):
def get_user_favorite_artworks(self, user_id: int) -> list:
return artworks
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:
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
artwork_gallery.py entity
gallery_service_impl.py dao
artist.py entity
VIRTUALARTGALLERY
.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
OUTLINE
TIMELINE
artwork_gallery.py
gallery_service_impl.py
artist.py
dao > gallery_service_impl.py > GalleryServiceImpl > add_artwork
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 []
Ln 20, Col 35 Spaces: 4 UTF-8 CRLF {} Python 3.11.5 ('venv': venv)

3. util/ – Database Utility Layer:

Description:

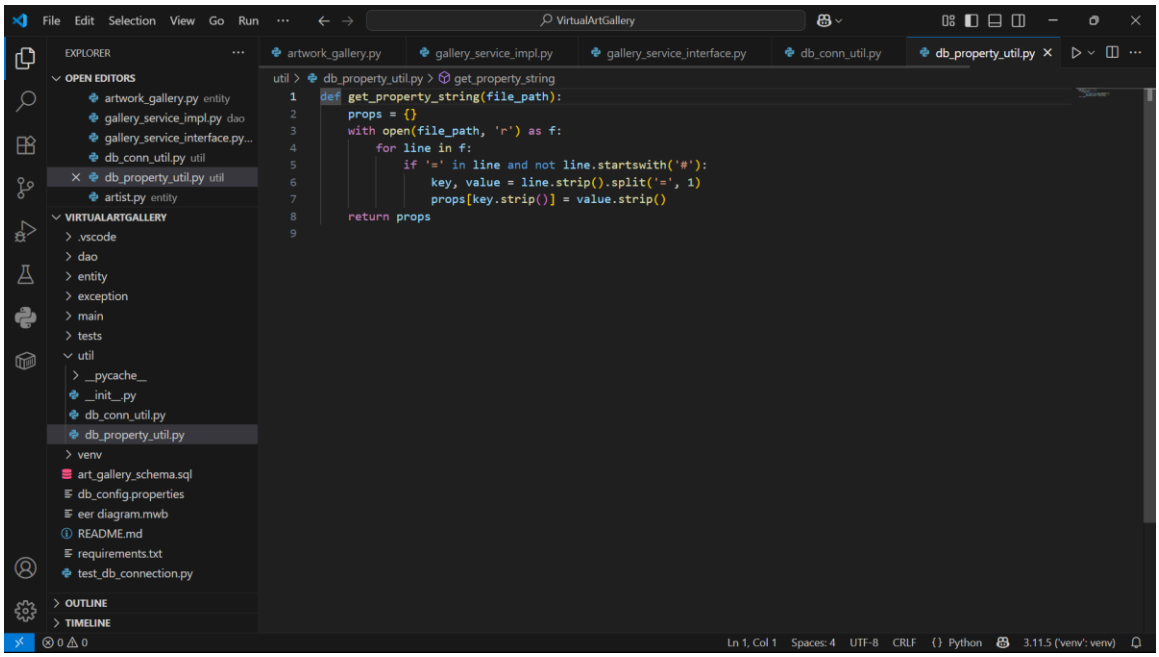
This package is responsible for database configuration and connection. The DBPropertyUtil class reads database credentials from a .properties file. The DBConnUtil class uses this information to establish a connection using mysql.connector.

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
```

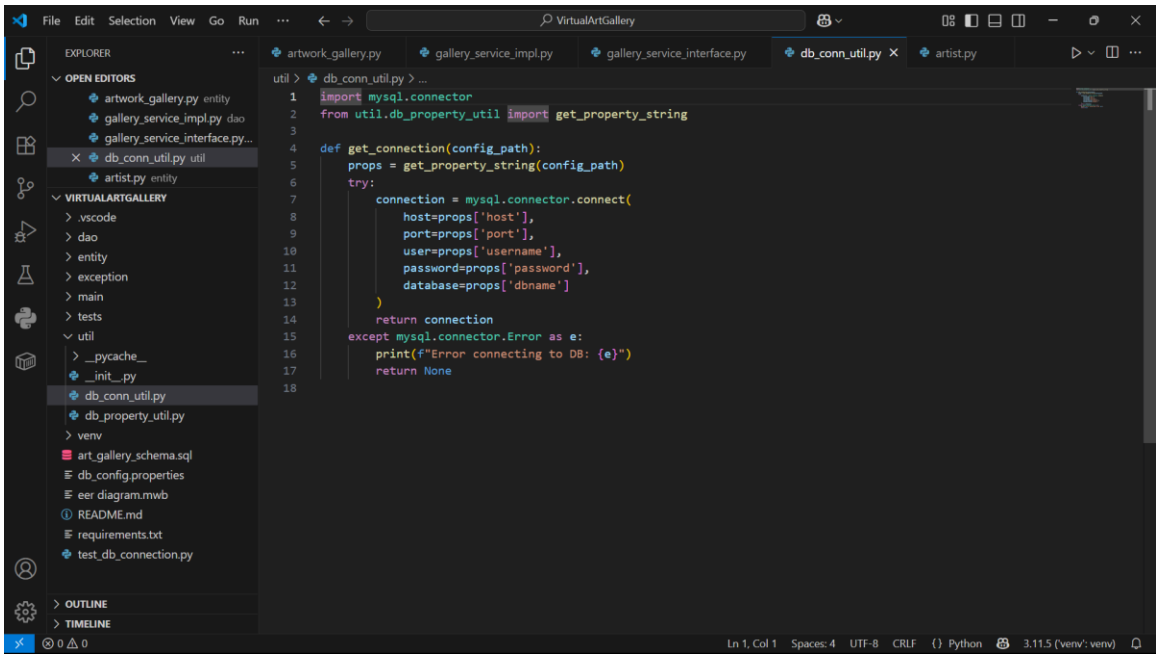


db_conn_util.py

```
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



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:

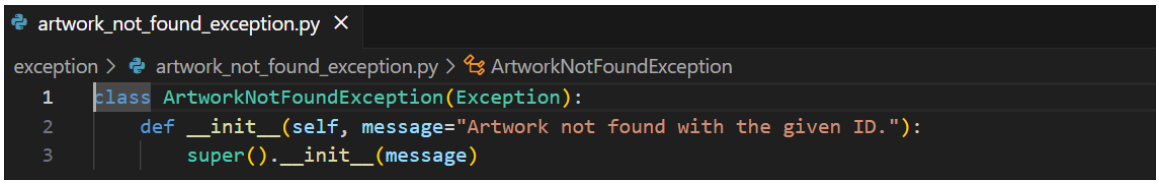
- **ArtworkNotFoundExceptionclass**

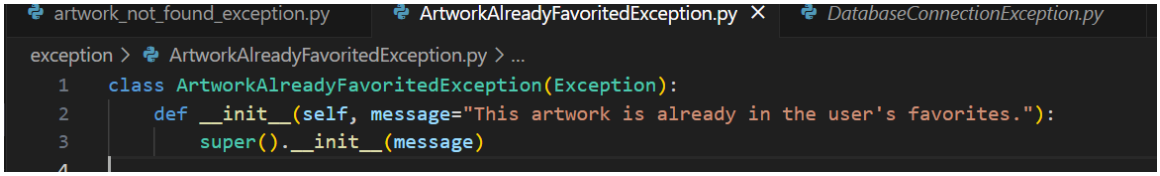
```
class ArtworkNotFoundException(Exception):
    def __init__(self, message="Artwork not found with the given ID."):
        super().__init__(message)
```

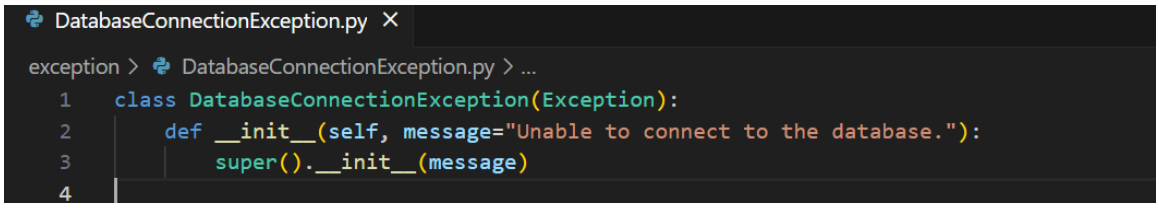
- **UserNotFoundException class**

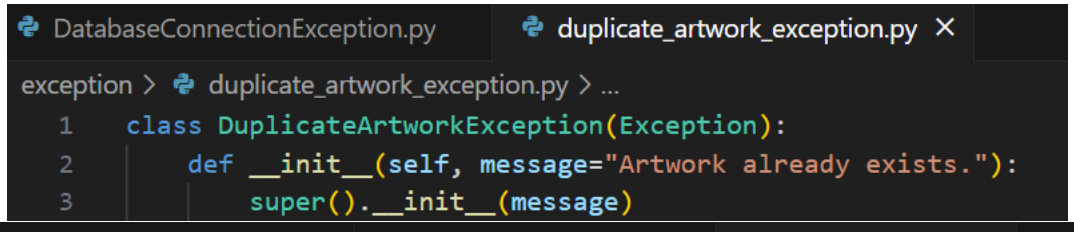
```
class UserNotFoundException(Exception):
    def __init__(self, message="User not found with the given ID."):
        super().__init__(message)
```

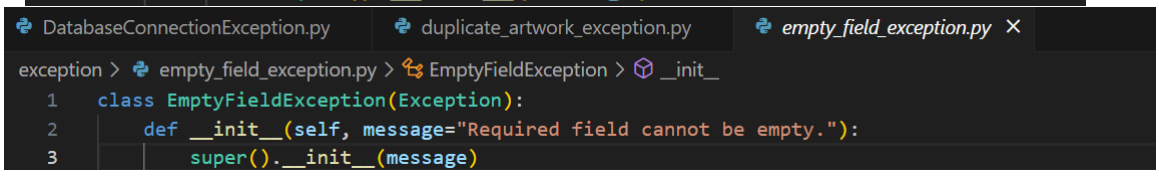
- Code in VirtualArtGalleryImpl that raises exceptions











exception > GalleryNotFoundException.py > ...

```python
class GalleryNotFoundException(Exception):
    def __init__(self, message="Gallery not found."):
        super().__init__(message)

```

exception > InvalidMediumException.py > ...

```python
class InvalidMediumException(Exception):
    def __init__(self, message="Invalid artwork medium. Please choose from a supported list."):
        super().__init__(message)

```

exception > NoFavoriteArtworksException > ...

```python
class NoFavoriteArtworksException(Exception):
    def __init__(self, message="No favorite artworks found for this user."):
        super().__init__(message)

```

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

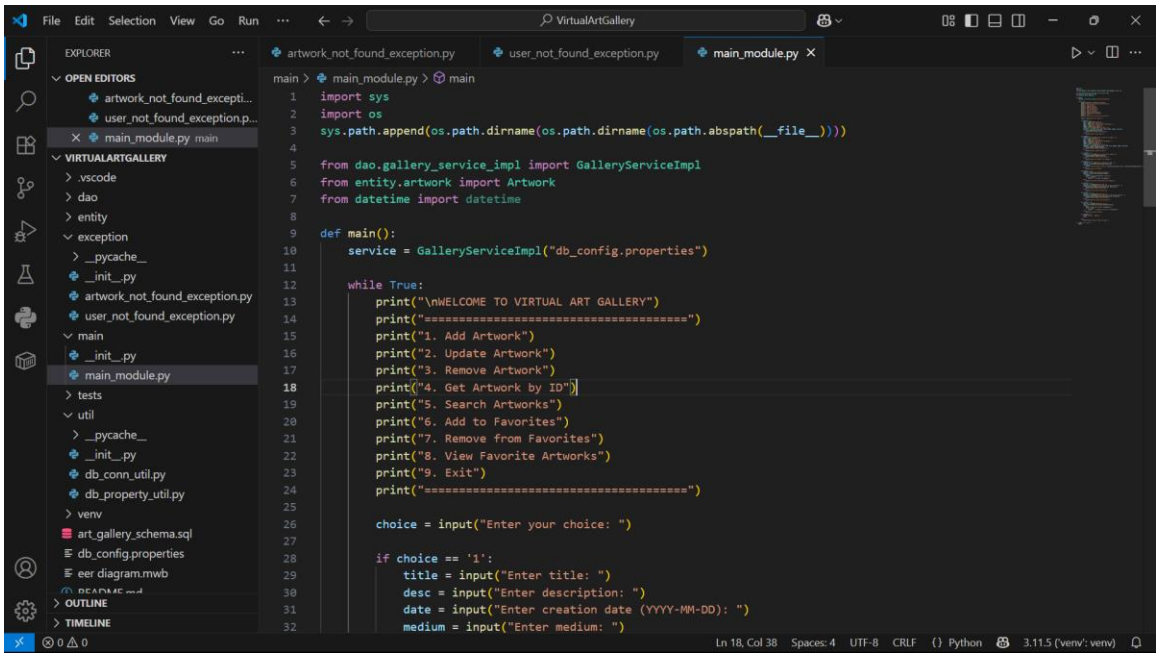
Main.module

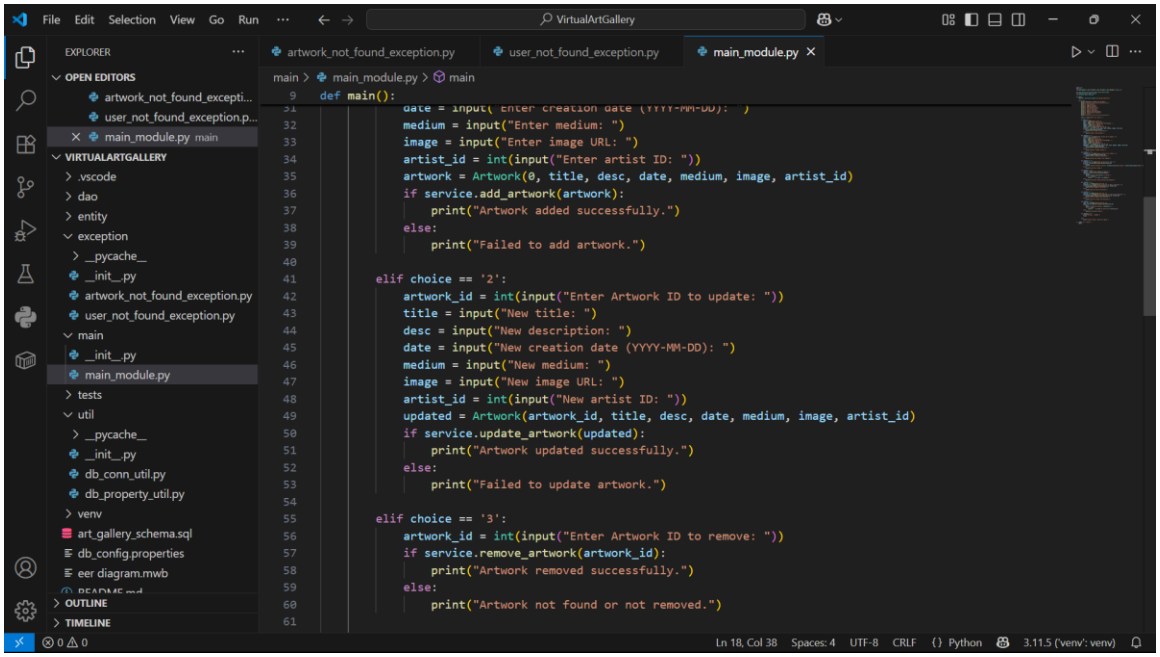
```python
import sys
import os
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))

from dao.gallery_service_impl import GalleryServiceImpl
from entity.artwork import Artwork
```

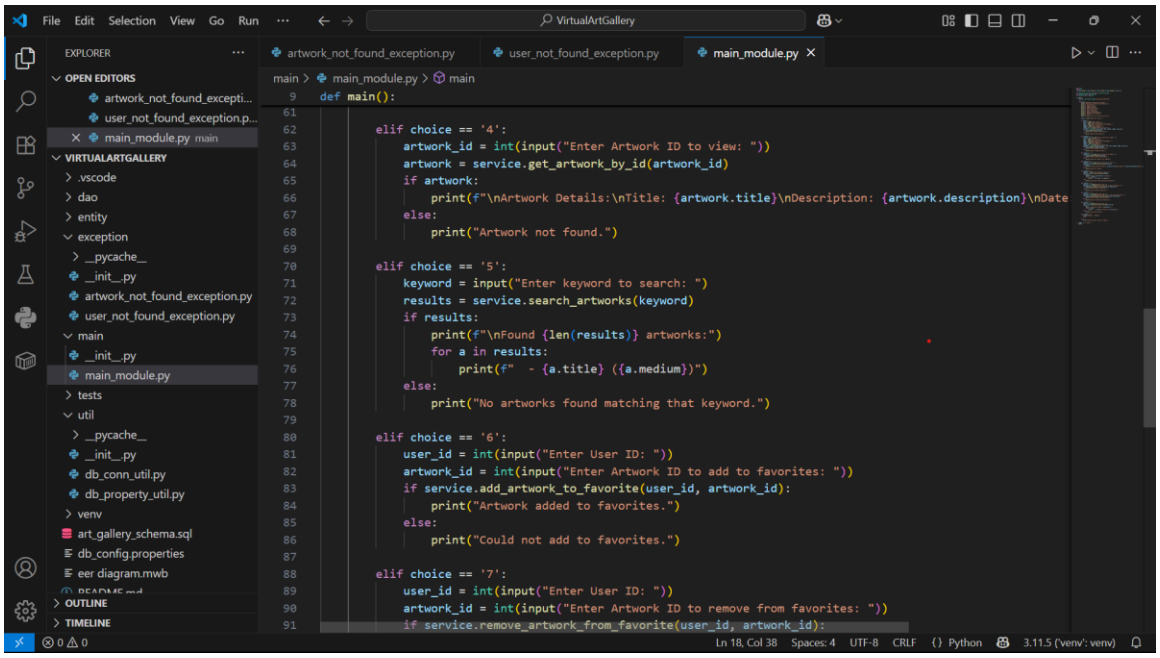
```python
from datetime import datetime

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)
```

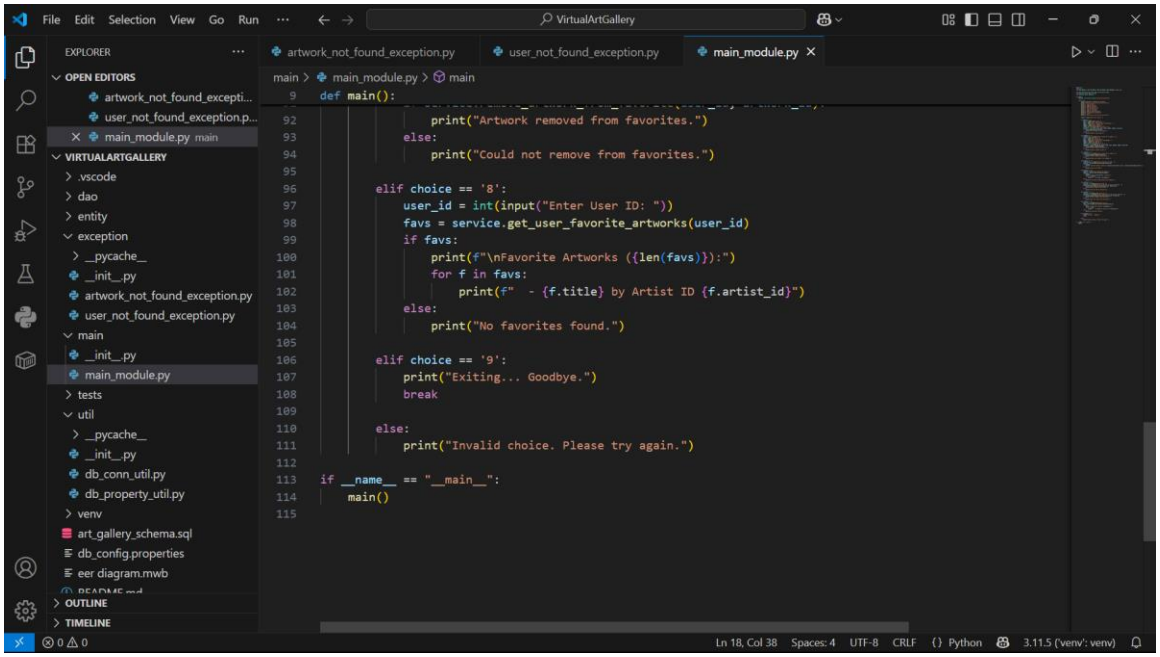
```
            if service.update_artwork(updated):
                print("Artwork updated successfully.")
            else:
                print("Failed to update artwork.")

        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()

```


File Edit Selection View Go Run
VirtualArtGallery
EXPLORER
OPEN EDITORS
artwork_not_found_excepti...
user_not_found_exception.p...
main_module.py main
VIRTUALARTGALLERY
.vscode
dao
entity
exception
__pycache__
__init__.py
artwork_not_found_exception.py
user_not_found_exception.py
main
__init__.py
main_module.py
tests
util
__pycache__
__init__.py
db_conn_util.py
db_property_util.py
venv
art_gallery_schema.sql
db_config.properties
eer diagram.mwb
OUTLINE
TIMELINE
artwork_not_found_exception.py
user_not_found_exception.py
main_module.py
main > main_module.py > main
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: ")
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_excepti...
user_not_found_exception.p...
main_module.py main
VIRTUALARTGALLERY
.vscode
dao
entity
exception
__pycache__
__init__.py
artwork_not_found_exception.py
user_not_found_exception.py
main
__init__.py
main_module.py
tests
util
__pycache__
__init__.py
db_conn_util.py
db_property_util.py
venv
art_gallery_schema.sql
db_config.properties
eer diagram.mwb
OUTLINE
TIMELINE
artwork_not_found_exception.py
user_not_found_exception.py
main_module.py
main > main_module.py > main
def main():
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.")
Ln 18, Col 38 Spaces: 4 UTF-8 CRLF {} Python 3.11.5 ('venv': venv)





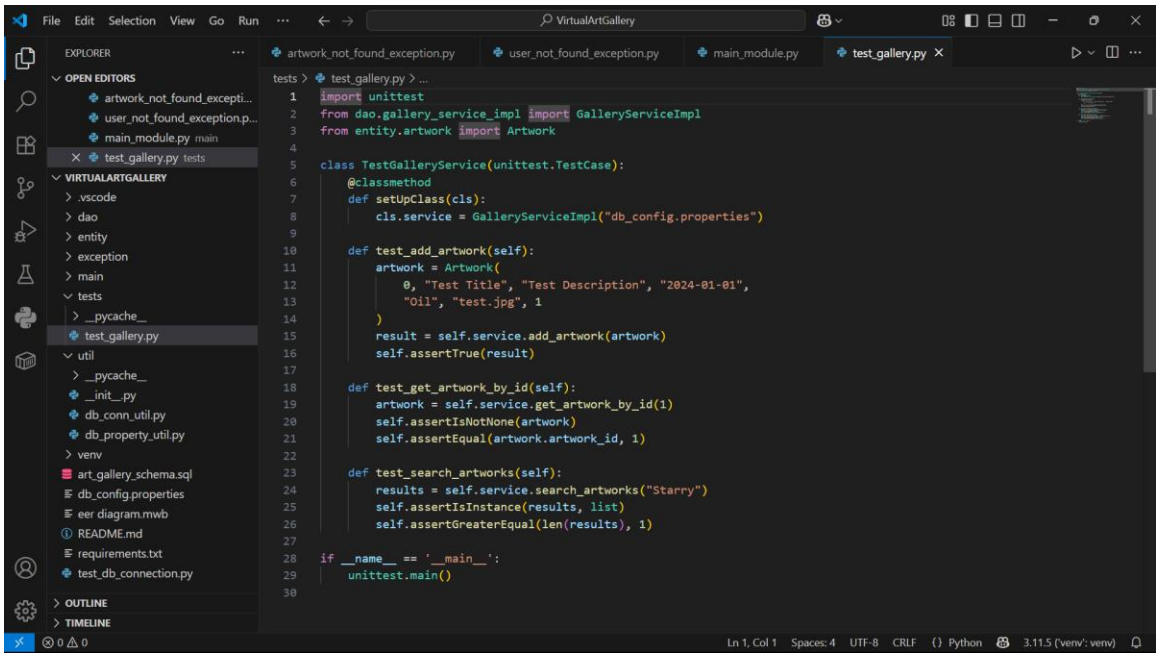
6. tests/ – Unit Testing:

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

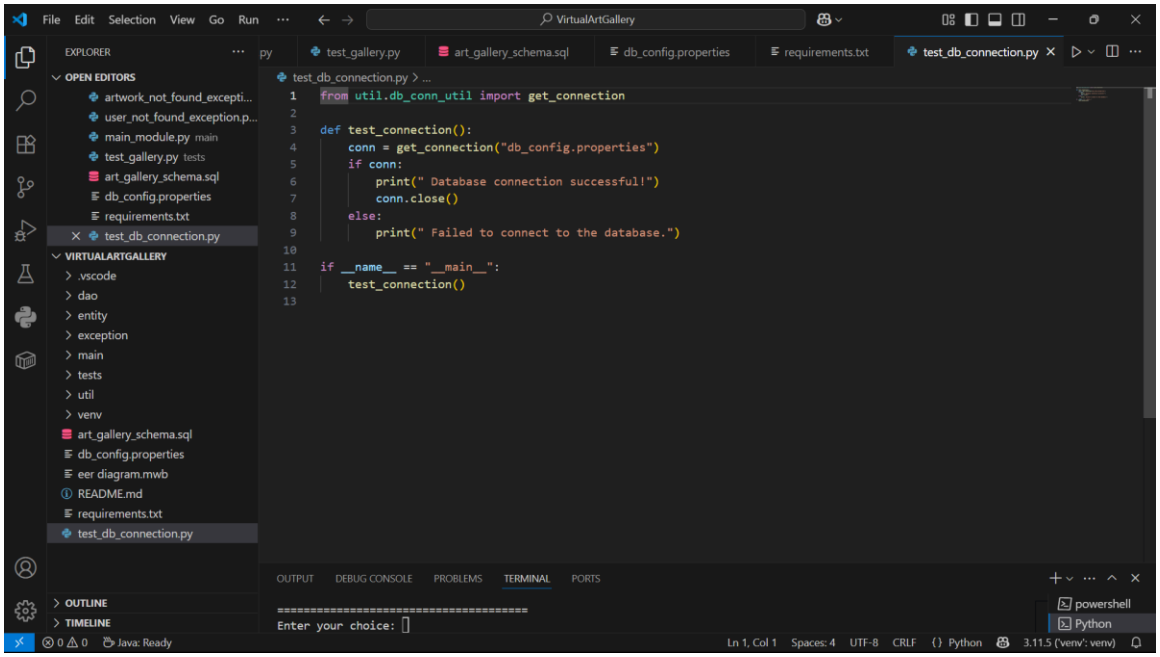
```
import unittest
from dao.gallery_service_impl import GalleryServiceImpl
from entity.artwork import Artwork

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()
```



7. Db connectivity check:

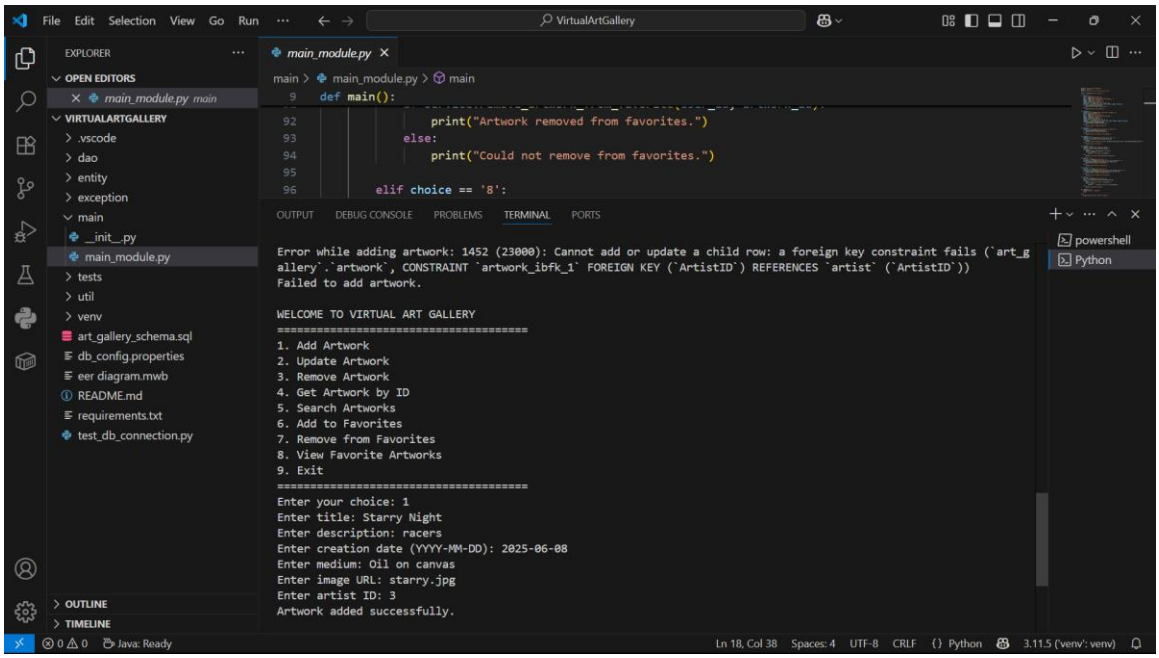
Db_connection.py :

```
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()
```

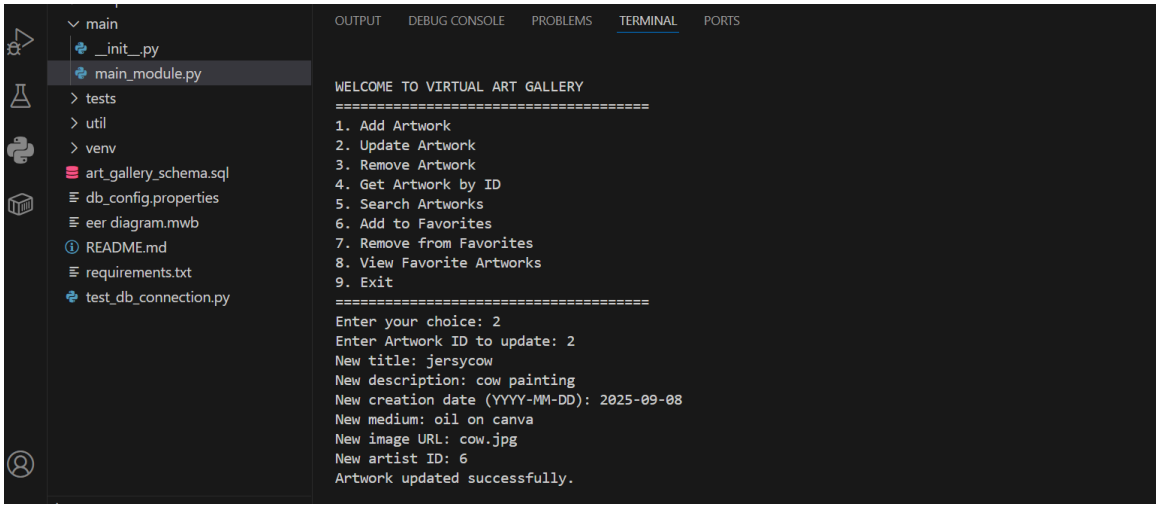


7. Output Evidence (MySQL + CLI)

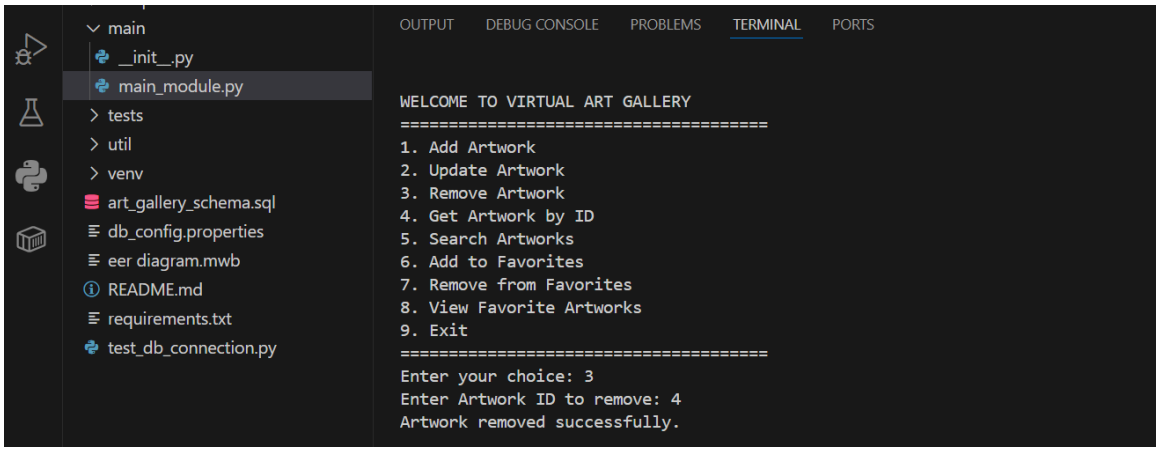
1. Add the artwork



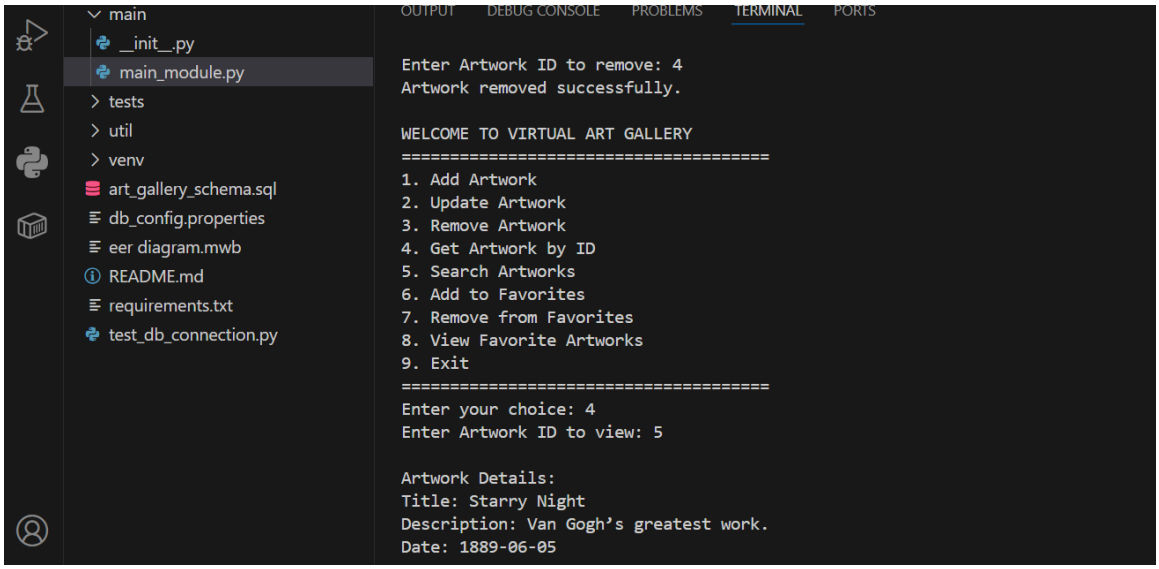
2. update the artwork



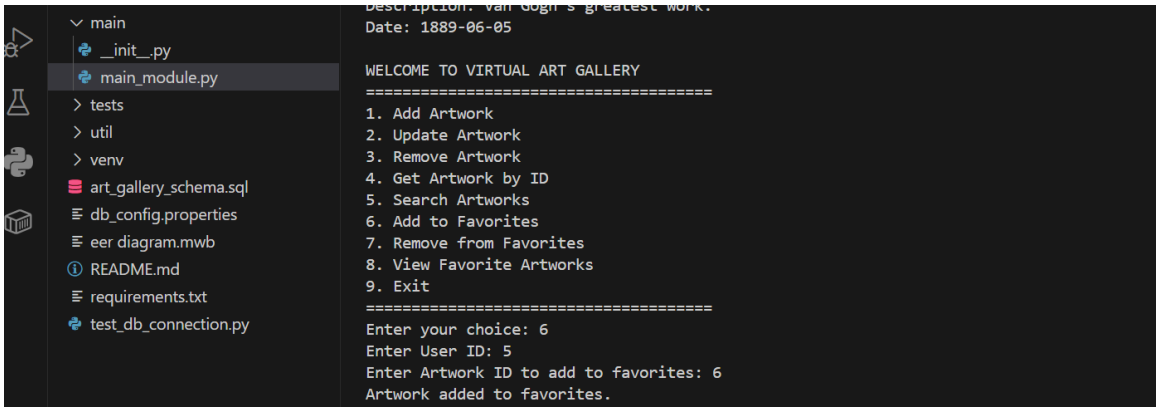
3. remove artwork



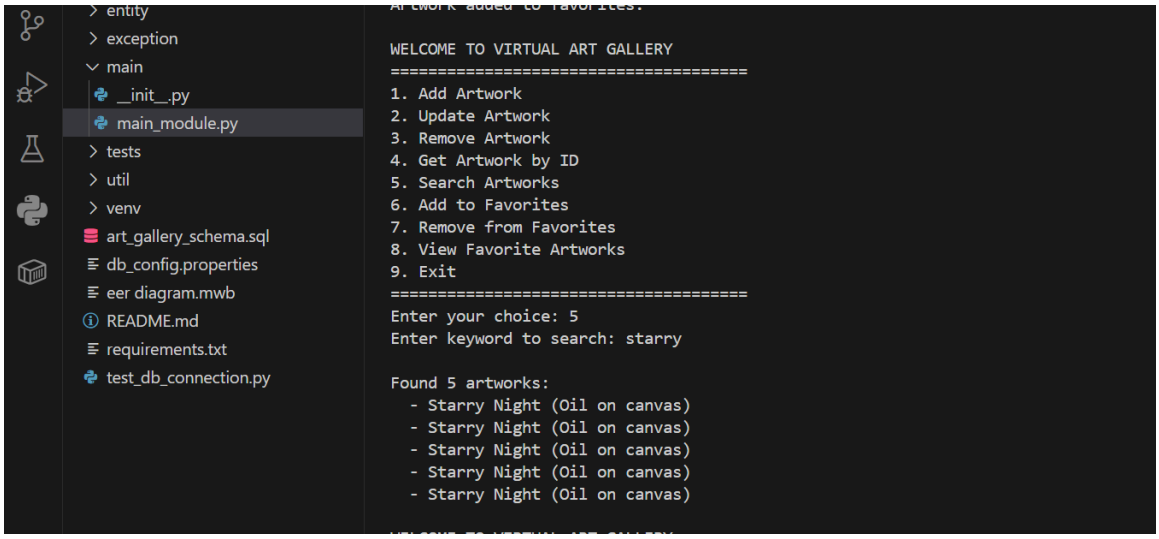
4. get artworks by id



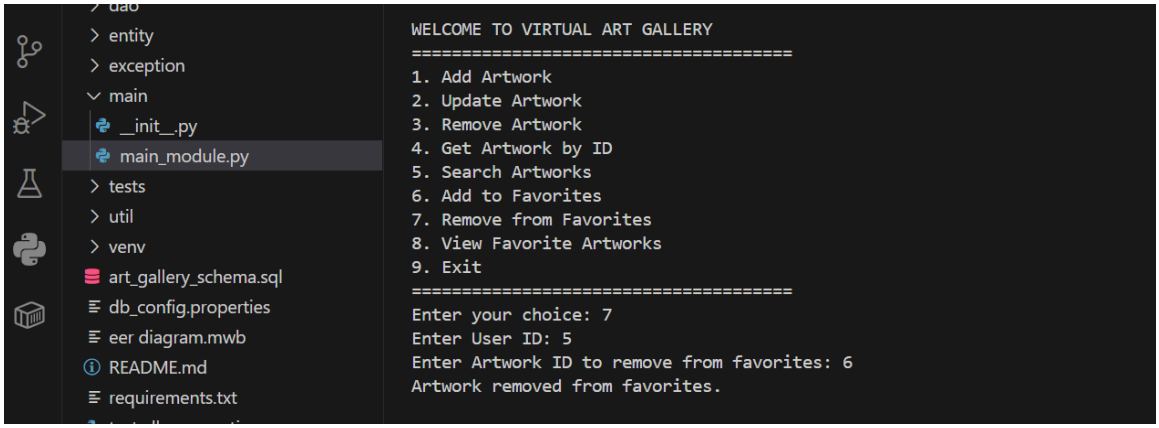
5. add to favourites



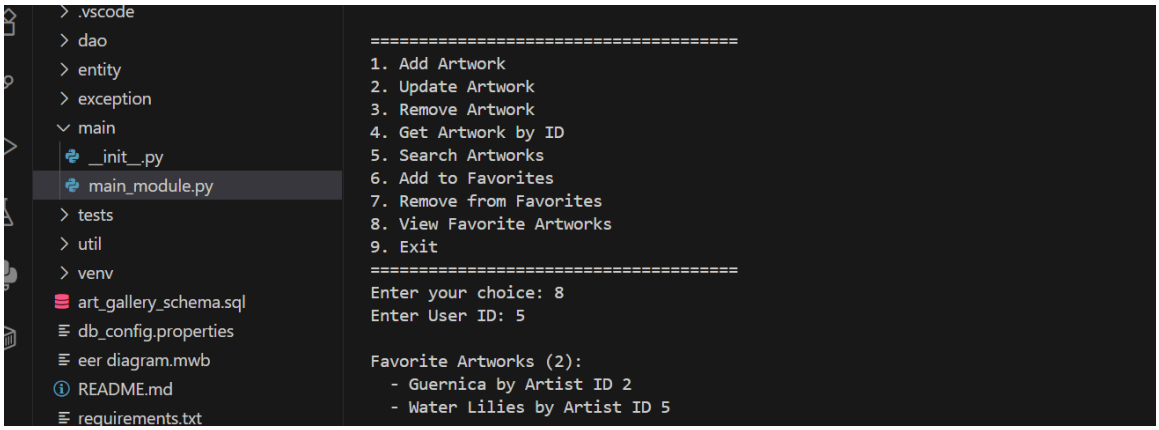
6. search artworks



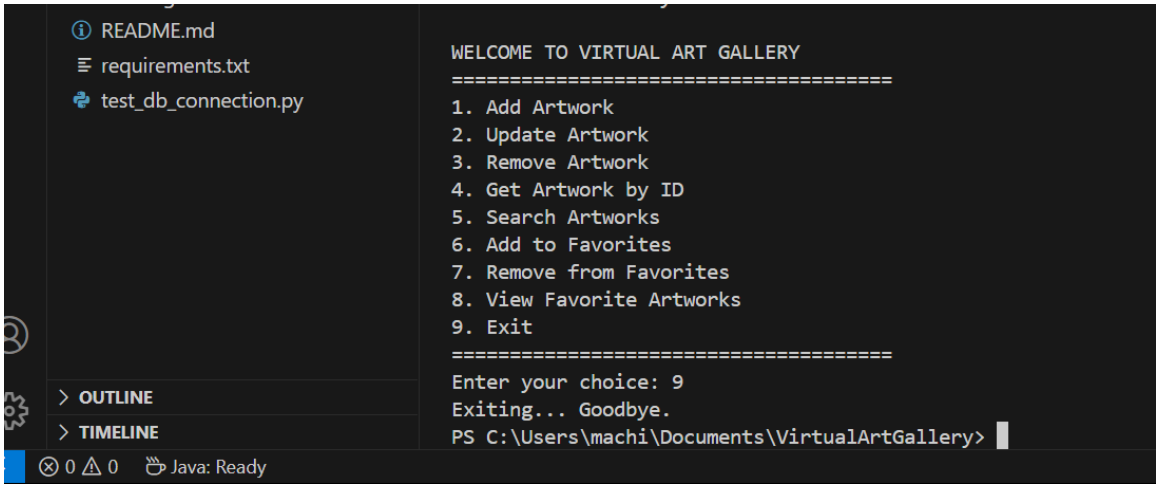
7. remove from favourites



8. view favourite artworks



9. exit



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