CASE STUDY – Visual Art Gallery

Database Schema:

Artwork Table:

- ArtworkID (Primary Key)
- Title
- Description
- CreationDate
- Medium
- ImageURL
- ArtistID (Foreign Key referencing Artist.ArtistID)

Artist Table:

- ArtistID (Primary Key)
- Name
- Biography
- BirthDate
- Nationality
- Website
- Contact Information

User Table:

- UserID (Primary Key)
- Username
- Password
- Email
- First Name
- Last Name
- Date of Birth
- Profile Picture

Gallery Table:

- GalleryID (Primary Key)
- Name
- Description
- Location
- Curator (Foreign Key referencing Artist.ArtistID)
- OpeningHours

User_Favorite_Artwork Table (Many-to-Many Junction Table):

- UserID (Foreign Key referencing User.UserID)
- ArtworkID (Foreign Key referencing Artwork.ArtworkID)

Artwork_Gallery Table (Many-to-Many Junction Table):

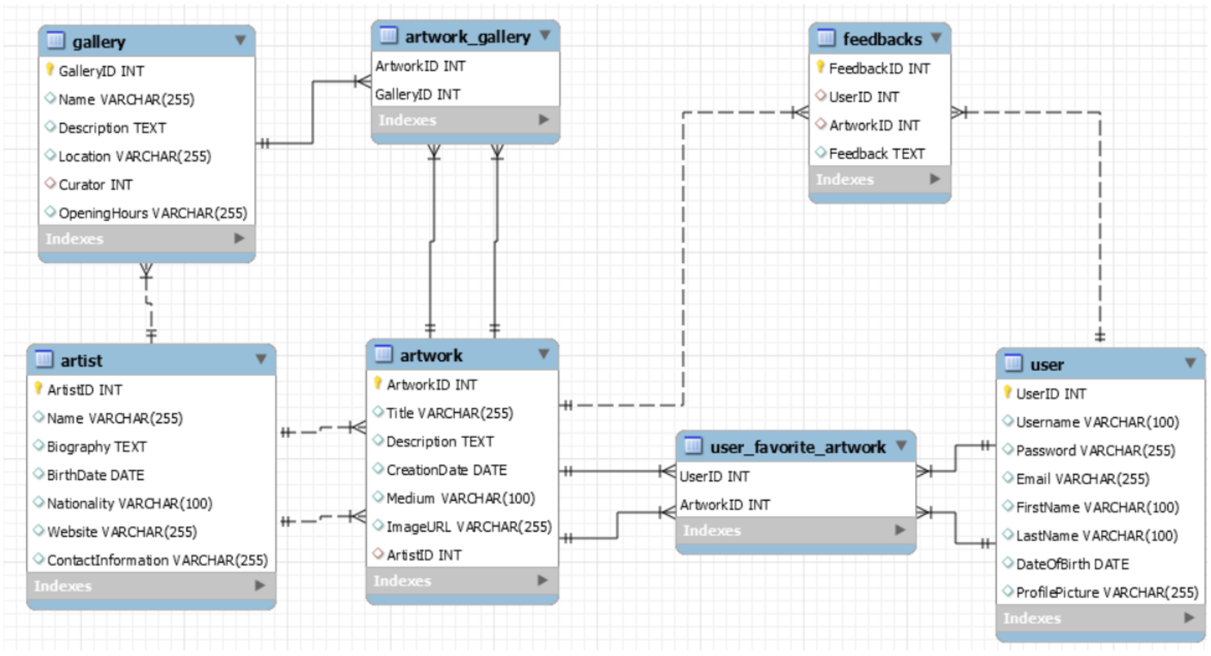
- ArtworkID (Foreign Key referencing Artwork.ArtworkID)
- GalleryID (Foreign Key referencing Gallery.GalleryID)

Feedbacks Table

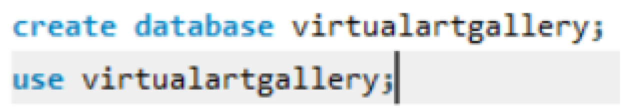
- UserID (Foreign Key referencing User.UserID)
- ArtworkID (Foreign Key referencing Artwork.ArtworkID)
- Feedback_id(primary key)
- feedback

Database Name: virtualartgallery

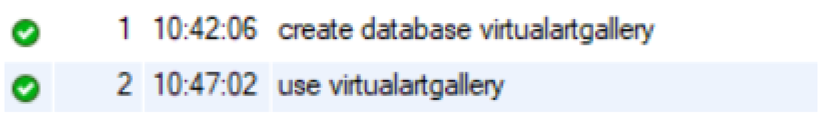
ERD:



Database creation:

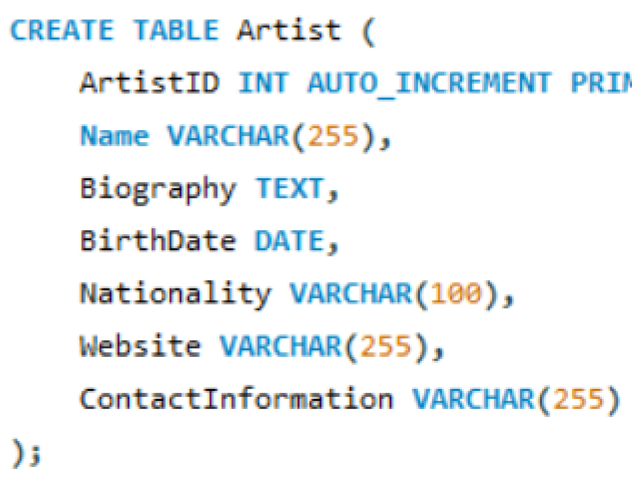


```
create database virtualartgallery;
use virtualartgallery;
```

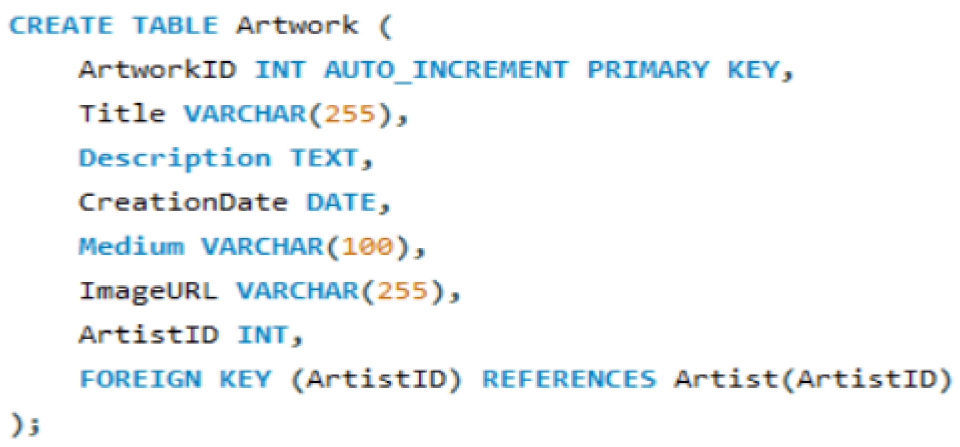


```
1  10:42:06  create database virtualartgallery
2  10:47:02  use virtualartgallery
```

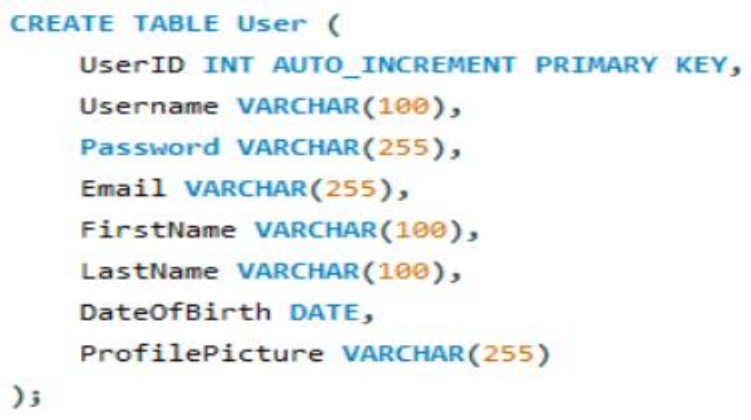
Table Creation:



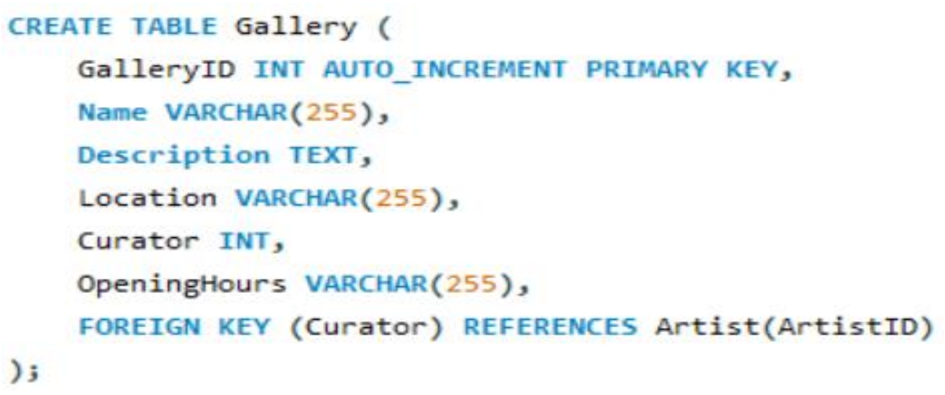
```
CREATE TABLE Artist (
    ArtistID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255),
    Biography TEXT,
    BirthDate DATE,
    Nationality VARCHAR(100),
    Website VARCHAR(255),
    ContactInformation VARCHAR(255)
);
```



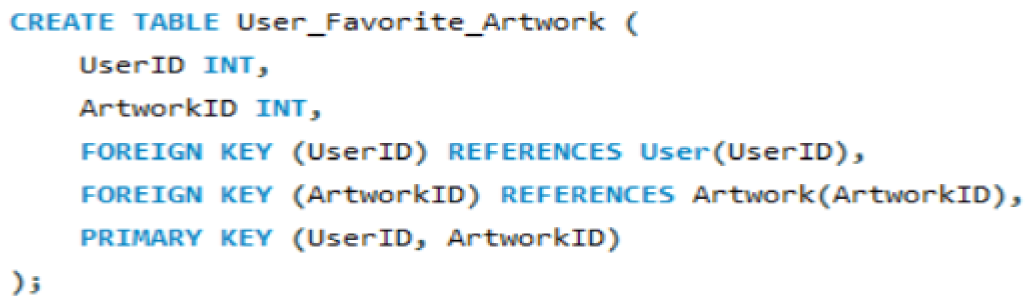
```
CREATE TABLE Artwork (
    ArtworkID INT AUTO_INCREMENT PRIMARY KEY,
    Title VARCHAR(255),
    Description TEXT,
    CreationDate DATE,
    Medium VARCHAR(100),
    ImageURL VARCHAR(255),
    ArtistID INT,
    FOREIGN KEY (ArtistID) REFERENCES Artist(ArtistID)
);
```



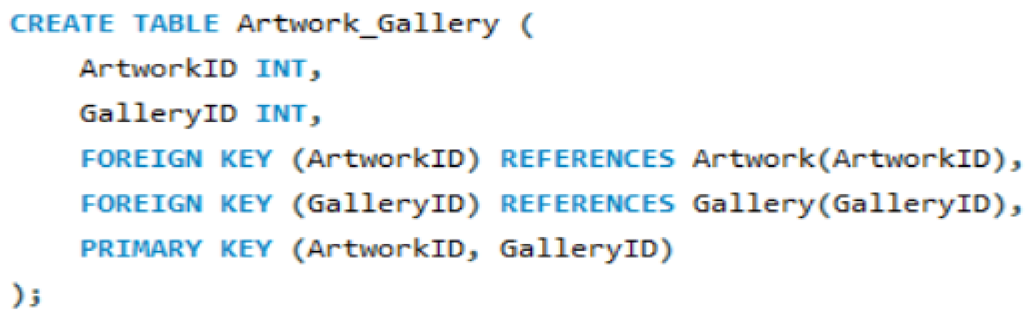
```
CREATE TABLE User (
    UserID INT AUTO_INCREMENT PRIMARY KEY,
    Username VARCHAR(100),
    Password VARCHAR(255),
    Email VARCHAR(255),
    FirstName VARCHAR(100),
    LastName VARCHAR(100),
    DateOfBirth DATE,
    ProfilePicture VARCHAR(255)
);
```



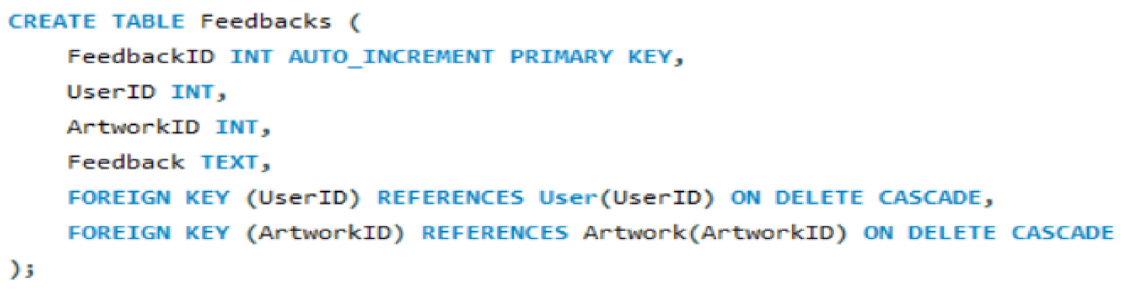
```sql
CREATE TABLE Gallery (
    GalleryID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255),
    Description TEXT,
    Location VARCHAR(255),
    Curator INT,
    OpeningHours VARCHAR(255),
    FOREIGN KEY (Curator) REFERENCES Artist(ArtistID)
);
```



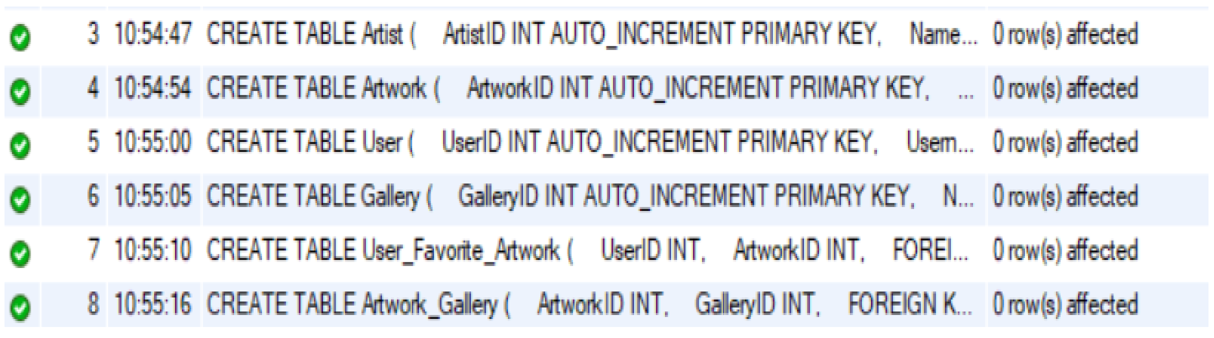
```sql
CREATE TABLE User_Favorite_Artwork (
    UserID INT,
    ArtworkID INT,
    FOREIGN KEY (UserID) REFERENCES User(UserID),
    FOREIGN KEY (ArtworkID) REFERENCES Artwork(ArtworkID),
    PRIMARY KEY (UserID, ArtworkID)
);
```



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

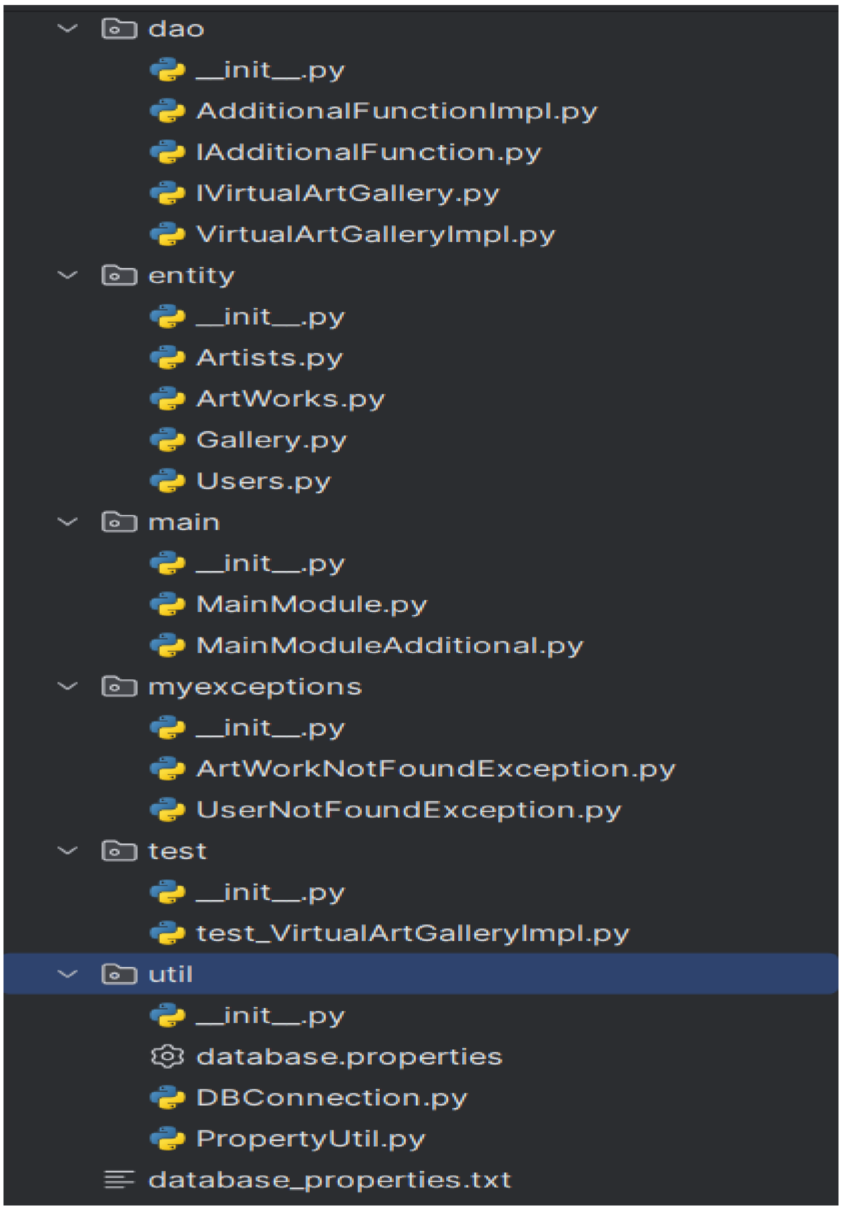


```sql
CREATE TABLE Feedbacks (
    FeedbackID INT AUTO_INCREMENT PRIMARY KEY,
    UserID INT,
    ArtworkID INT,
    Feedback TEXT,
    FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE,
    FOREIGN KEY (ArtworkID) REFERENCES Artwork(ArtworkID) ON DELETE CASCADE
);
```



```
✓  3  10:54:47  CREATE TABLE Artist (    ArtistID INT AUTO_INCREMENT PRIMARY KEY,    Name...      0 row(s) affected
✓  4  10:54:54  CREATE TABLE Artwork (    ArtworkID INT AUTO_INCREMENT PRIMARY KEY,    ...      0 row(s) affected
✓  5  10:55:00  CREATE TABLE User (    UserID INT AUTO_INCREMENT PRIMARY KEY,    Usern...       0 row(s) affected
✓  6  10:55:05  CREATE TABLE Gallery (    GalleryID INT AUTO_INCREMENT PRIMARY KEY,    N...     0 row(s) affected
✓  7  10:55:10  CREATE TABLE User_Favorite_Artwork (    UserID INT,    ArtworkID INT,    FOREI... 0 row(s) affected
✓  8  10:55:16  CREATE TABLE Artwork_Gallery (    ArtworkID INT,    GalleryID INT,    FOREIGN K... 0 row(s) affected
```

File Structure:



```
dao
    __init__.py
    AdditionalFunctionImpl.py
    IAdditionalFunction.py
    IVirtualArtGallery.py
    VirtualArtGalleryImpl.py
entity
    __init__.py
    Artists.py
    ArtWorks.py
    Gallery.py
    Users.py
main
    __init__.py
    MainModule.py
    MainModuleAdditional.py
myexceptions
    __init__.py
    ArtWorkNotFoundException.py
    UserNotFoundException.py
test
    __init__.py
    test_VirtualArtGalleryImpl.py
util
    __init__.py
    database.properties
    DBConnection.py
    PropertyUtil.py
database_properties.txt
```

Coding works:

Artist Class:

```
class Artist:
    def __init__(self, artist_id=None, name=None, biography=None,
birth_date=None, nationality=None, website=None,
                 contact_information=None):
        self.__artist_id = artist_id
        self.__name = name
        self.__biography = biography
        self.__birth_date = birth_date
        self.__nationality = nationality
        self.__website = website
        self.__contact_information = contact_information

    # Getter methods
    def get_artist_id(self):
        return self.__artist_id

    def get_name(self):
        return self.__name

    def get_biography(self):
        return self.__biography

    def get_birth_date(self):
        return self.__birth_date

    def get_nationality(self):
        return self.__nationality

    def get_website(self):
        return self.__website

    def get_contact_information(self):
        return self.__contact_information

    # Setter methods
    def set_name(self, name):
        self.__name = name

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

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

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

    def set_website(self, website):
        self.__website = website

    def set_contact_information(self, contact_information):
        self.__contact_information = contact_information
```

Artwork class:

```
class Artwork:
    def __init__(self, artwork_id=None, title=None, description=None,
creation_date=None, medium=None, image_url=None,
                 artist_id=None):
        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

    # Getter methods
    def get_artwork_id(self):
        return self.__artwork_id
    def get_title(self):
        return self.__title

    def get_description(self):
        return self.__description

    def get_creation_date(self):
        return self.__creation_date

    def get_medium(self):
        return self.__medium

    def get_image_url(self):
        return self.__image_url

    def get_artist_id(self):
        return self.__artist_id

    # Setter methods
    def set_title(self, title):
        self.__title = title

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

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

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

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

    def set_artist_id(self, artist_id):
        self.__artist_id = artist_id
```

Gallery class:

```
class Gallery:
    def __init__(self, gallery_id=None, name=None, description=None,
location=None, curator=None, opening_hours=None):
        self.__gallery_id = gallery_id
        self.__name = name
        self.__description = description
        self.__location = location
        self.__curator = curator
        self.__opening_hours = opening_hours

    # Getter methods
    def get_gallery_id(self):
        return self.__gallery_id

    def get_name(self):
        return self.__name
```

```python
    def get_description(self):
        return self.__description

    def get_location(self):
        return self.__location

    def get_curator(self):
        return self.__curator

    def get_opening_hours(self):
        return self.__opening_hours

    # Setter methods
    def set_name(self, name):
        self.__name = name

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

    def set_location(self, location):
        self.__location = location

    def set_curator(self, curator):
        self.__curator = curator

    def set_opening_hours(self, opening_hours):
        self.__opening_hours = opening_hours
```

User class:

```python
class User:
    def __init__(self, user_id=None, username=None, password=None,
email=None, first_name=None, last_name=None,
                 date_of_birth=None, profile_picture=None):
        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

    # Getter methods
    def get_user_id(self):
        return self.__user_id

    def get_username(self):
        return self.__username

    def get_password(self):
        return self.__password

    def get_email(self):
        return self.__email

    def get_first_name(self):
        return self.__first_name
    def get_last_name(self):
        return self.__last_name

    def get_date_of_birth(self):
        return self.__date_of_birth

    def get_profile_picture(self):
        return self.__profile_picture

    # Setter methods
    def set_username(self, username):
        self.__username = username

    def set_password(self, password):
        self.__password = password

    def set_email(self, email):
        self.__email = email

    def set_first_name(self, first_name):
        self.__first_name = first_name

    def set_last_name(self, last_name):
        self.__last_name = last_name

    def set_date_of_birth(self, date_of_birth):
        self.__date_of_birth = date_of_birth

    def set_profile_picture(self, profile_picture):
        self.__profile_picture = profile_picture
```

Interfaces:

IVirtualArtGalery interface:

```
from abc import ABC, abstractmethod


class IVirtualArtGallery(ABC):
    @abstractmethod
    def addArtwork(self, artwork):
        pass

    @abstractmethod
    def updateArtwork(self, artwork):
        pass

    @abstractmethod
    def removeArtwork(self, artworkID):
        pass

    @abstractmethod
    def getArtworkById(self, artworkID):
        pass

    @abstractmethod
    def searchArtworks(self, keyword):
        pass
    @abstractmethod
    def showAllArtworks(self):
        pass
    @abstractmethod
    def addArtworkToFavorite(self, userID, artworkID):
        pass

    @abstractmethod
    def removeArtworkFromFavorite(self, userID, artworkID):
        pass

    @abstractmethod
    def getUserFavoriteArtworks(self, userID):
        pass
    @abstractmethod
    def feedback(self, user_id, artwork_id, feedback_text):
        pass

    @abstractmethod
    def getFeedbacksByArtworkId(self, artwork_id):
        pass
```

Added a few more functions to IAdditionalFunctions:

```
from abc import ABC, abstractmethod


class IAdditionalFunction(ABC):
    @abstractmethod
    def addGallery(self, gallery):
        pass

    @abstractmethod
    def updateGallery(self, gallery):
        pass

    @abstractmethod
    def removeGallery(self, gallery_id):
        pass

    @abstractmethod
    def getGalleryById(self, gallery_id):
        pass

    @abstractmethod
    def searchGalleries(self, keyword):
        pass

    # Artwork_gallery Management
    @abstractmethod
    def addArtworkToGallery(self, gallery_id, artwork_id):
        pass

    @abstractmethod
    def removeArtworkFromGallery(self, gallery_id, artwork_id):
        pass

    @abstractmethod
    def getArtworksOfGallery(self, gallery_id):
        pass

    # Artist management
    @abstractmethod
    def createArtist(self, artist):
        pass

    @abstractmethod
    def updateArtist(self, artist):
        pass

    @abstractmethod
    def deleteArtist(self, artist_id):
        pass

    @abstractmethod
    def getArtistById(self, artist_id):
        pass
```

Implementation of interface:

VirtualArtGalleryImpl:

```
from mysql.connector import Error

from dao.IVirtualArtGallery import IVirtualArtGallery
from entity.ArtWorks import Artwork
from myexceptions.ArtWorkNotFoundException import ArtWorkNotFoundException
from myexceptions.UserNotFoundException import UserNotFoundException
from util.DBConnection import DBConnection


class VirtualArtGalleryImpl(IVirtualArtGallery):
    def addArtwork(self, artwork):
        # print("inside addartwork")
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()

            sql = ("INSERT INTO Artwork (Title, Description, CreationDate,
Medium, ImageURL, ArtistID) VALUES (%s, %s, "
                   "%s, %s, %s, %s)")
            values = (artwork.get_title(), artwork.get_description(),
artwork.get_creation_date(), artwork.get_medium(),
                      artwork.get_image_url(),artwork.get_artist_id())
            cursor.execute(sql, values)

            connection.commit()
            new_artwork_id = cursor.lastrowid
            print("Artwork ID:", new_artwork_id)
            cursor.close()
            print("---------------Artwork added successfully!--------------
-")
            return True

        except Exception as e:
            # print("Error adding artwork:", e)
            print("Your artist id is invalid.Only artists can add
artwork.")
            return False

    def updateArtwork(self, artwork):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()

            sql = ("UPDATE Artwork SET Title=%s, Description=%s,
CreationDate=%s, Medium=%s, ImageURL=%s WHERE "
                   "ArtworkID=%s")
            values = (artwork.get_title(), artwork.get_description(),
artwork.get_creation_date(), artwork.get_medium(),
                      artwork.get_image_url(), artwork.get_artwork_id())
            cursor.execute(sql, values)

            connection.commit()
            cursor.close()
            print("---------------Artwork updated successfully!------------
---")

            return True
        except Exception as e:
            print("Error updating artwork:", e)
            return False

    def removeArtwork(self, artwork_id):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()

            sql_check = "SELECT * FROM Artwork WHERE ArtworkID=%s"
            cursor.execute(sql_check, (artwork_id,))
            artwork = cursor.fetchone()
            if artwork is None:
                raise ArtWorkNotFoundException(artwork_id)

            sql = "DELETE FROM Artwork WHERE ArtworkID=%s"
            cursor.execute(sql, (artwork_id,))

            connection.commit()
            cursor.close()
            # print("Artwork removed successfully!")
            return True
        except ArtWorkNotFoundException as e:
            return False
            # print(e)
        except Exception as e:
            print("Error removing artwork:", e)
            return False

    def getArtworkById(self, artwork_id):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()

            sql = "SELECT * FROM Artwork WHERE ArtworkID = %s"
            cursor.execute(sql, (artwork_id,))
            artwork_data = cursor.fetchone()
            if artwork_data:
                artwork = Artwork(artwork_data[0], artwork_data[1],
artwork_data[2], artwork_data[3], artwork_data[4],
                                  artwork_data[5],artwork_data[6])
                return artwork
            else:
                # print("Artwork with ID", artwork_id, "not found.")
                raise ArtWorkNotFoundException(artwork_id)

        except ArtWorkNotFoundException as e:
            # print(e)
            return None
        except Exception as e:
            print("Error getting artwork by ID:", e)
            return None

    def searchArtworks(self, keyword):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()

            sql = "SELECT * FROM Artwork WHERE Title LIKE %s OR Description
LIKE %s"
            keyword_like = f"%{keyword}%"
            cursor.execute(sql, (keyword_like, keyword_like))
            artworks_data = cursor.fetchall()

            artworks = []

            if not artworks_data:
                print("No artworks found matching the keyword:", keyword)
            else:
                for artwork_data in artworks_data:
                    artwork = Artwork(artwork_data[0], artwork_data[1],
artwork_data[2], artwork_data[3],
                                      artwork_data[4],
                                      artwork_data[5], artwork_data[6])
                    artworks.append(artwork)

                cursor.close()

            return artworks
        except Exception as e:
            print("Error searching artworks:", e)
            return []

    def showAllArtworks(self):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()
            cursor.execute("SELECT * FROM artwork")
            artwork = cursor.fetchall()
            cursor.close()
            print("All artworks Available:")
            for artwork in artwork:
                print(artwork)
            print()

        except Error as e:
            print("Error fetching artwork details:", e)
    def addArtworkToFavorite(self, user_id, artwork_id):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()

            sql_check_user = "SELECT * FROM User WHERE UserID = %s"
            cursor.execute(sql_check_user, (user_id,))
            user_data = cursor.fetchone()
            if user_data is None:
                raise UserNotFoundException(user_id)

            sql_check_artwork = "SELECT * FROM Artwork WHERE ArtworkID =
%s"
            cursor.execute(sql_check_artwork, (artwork_id,))
            artwork_data = cursor.fetchone()
            if artwork_data is None:
                raise ArtWorkNotFoundException(artwork_id)
            sql_check_favorites = "SELECT * FROM User_Favorite_Artwork
WHERE UserID = %s AND ArtworkID = %s"
            cursor.execute(sql_check_favorites, (user_id, artwork_id))
            existing_favorite = cursor.fetchone()
            if existing_favorite:
                print("Artwork is already in the user's favorites.")
            else:
                sql_add_favorite = "INSERT INTO User_Favorite_Artwork
(UserID, ArtworkID) VALUES (%s, %s)"
                cursor.execute(sql_add_favorite, (user_id, artwork_id))
                connection.commit()
                print("Artwork added to favorites successfully!")
                cursor.close()
                return True
            cursor.close()
            return False
        except UserNotFoundException as e:
            print(e)
        except ArtWorkNotFoundException as e:
            print(e)
        except Exception as e:
            print("Error adding artwork to favorites:", e)

    def removeArtworkFromFavorite(self, user_id, artwork_id):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()

            sql_check_user = "SELECT * FROM User WHERE UserID = %s"
            cursor.execute(sql_check_user, (user_id,))
            user_data = cursor.fetchone()
            if user_data is None:
                raise UserNotFoundException(user_id)

            sql_check_artwork = "SELECT * FROM Artwork WHERE ArtworkID =
%s"
            cursor.execute(sql_check_artwork, (artwork_id,))
            artwork_data = cursor.fetchone()
            if artwork_data is None:
                raise ArtWorkNotFoundException(artwork_id)

            sql_check_favorites = "SELECT * FROM User_Favorite_Artwork
WHERE UserID = %s AND ArtworkID = %s"
            cursor.execute(sql_check_favorites, (user_id, artwork_id))
            existing_favorite = cursor.fetchone()
            if existing_favorite:
                sql_remove_favorite = "DELETE FROM User_Favorite_Artwork
WHERE UserID = %s AND ArtworkID = %s"
                cursor.execute(sql_remove_favorite, (user_id, artwork_id))
                connection.commit()
                cursor.close()
                print("Artwork removed from favorites successfully!")
                return True
            else:
                print("Artwork is not in the user's favorites.")
                cursor.close()

                return False
        except UserNotFoundException as e:
            print(e)
        except ArtWorkNotFoundException as e:
            print(e)
        except Exception as e:
            print("Error removing artwork from favorites:", e)

    def getUserFavoriteArtworks(self, user_id):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()

            sql_check_user = "SELECT * FROM User WHERE UserID = %s"
            cursor.execute(sql_check_user, (user_id,))
            user_data = cursor.fetchone()
            if user_data is None:
                raise UserNotFoundException(user_id)

            sql_get_favorites = ("SELECT A.* FROM Artwork A JOIN
User_Favorite_Artwork UFA ON A.ArtworkID = "
                                 "UFA.ArtworkID WHERE UFA.UserID = %s")
            cursor.execute(sql_get_favorites, (user_id,))
            favorite_artworks = cursor.fetchall()

            if not favorite_artworks:
                print("User has no favorite artworks.")
            else:
                print("Favorite artworks for User ID:", user_id)
                for artwork_data in favorite_artworks:
                    print("------------------------------------")
                    print("Artwork ID:      ", artwork_data[0])
                    print("Title:           ", artwork_data[1])
                    print("Description:     ", artwork_data[2])
                    print("Creation Date:   ", artwork_data[3])
                    print("Medium:          ", artwork_data[4])
                    print("Image URL:       ", artwork_data[5])
                    print("------------------------------------")
                cursor.close()
                return True
            cursor.close()
            return False
        except UserNotFoundException as e:
            print(e)
        except Exception as e:
            print("Error getting user's favorite artworks:", e)
```

AdditionalFunctionImpl:

```
from dao.IAdditionalFunction import IAdditionalFunction
from entity.ArtWorks import Artwork
from entity.Artists import Artist
from entity.Gallery import Gallery
from util.DBConnection import DBConnection


class AdditionalFunctionImpl(IAdditionalFunction):
    # gallery management
    def addGallery(self, gallery):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()

            sql = "INSERT INTO Gallery (Name, Description, Location,
Curator, OpeningHours) VALUES (%s, %s, %s, %s, %s)"
            values = (gallery.get_name(), gallery.get_description(),
gallery.get_location(), gallery.get_curator(),
                      gallery.get_opening_hours())
            cursor.execute(sql, values)
            connection.commit()
            cursor.close()
            print("Gallery created successfully.")
            return True

        except Exception as e:
            print("Error creating gallery:", e)
            return False

    def updateGallery(self, gallery):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()

            sql = ("UPDATE Gallery SET Name = %s, Description = %s,
Location = %s, Curator = %s, OpeningHours = %s "
                   "WHERE GalleryID = %s")
            values = (gallery.get_name(), gallery.get_description(),
gallery.get_location(), gallery.get_curator(),
                      gallery.get_opening_hours(),
gallery.get_gallery_id())
            cursor.execute(sql, values)
            connection.commit()
            cursor.close()

            print("Gallery updated successfully.")
            return True

        except Exception as e:
            print("Error updating gallery:", e)
            return False

    def removeGallery(self, gallery_id):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()

            sql_check = "SELECT * FROM Gallery WHERE GalleryID = %s"
            cursor.execute(sql_check, (gallery_id,))
            gallery_data = cursor.fetchone()

            if gallery_data:
                sql_delete = "DELETE FROM Gallery WHERE GalleryID = %s"
                cursor.execute(sql_delete, (gallery_id,))
                connection.commit()
                cursor.close()
                print("Gallery removed successfully.")
                return True
            else:
                print("Gallery with ID", gallery_id, "does not exist.")
                return False

        except Exception as e:
            print("Error removing gallery:", e)
            return False

    def searchGalleries(self, keyword):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()

            sql = "SELECT * FROM Gallery WHERE Name LIKE %s OR Description
LIKE %s"
            keyword_like = f"%{keyword}%"
            cursor.execute(sql, (keyword_like, keyword_like))
            galleries_data = cursor.fetchall()

            galleries = []
            for gallery_data in galleries_data:
                gallery = Gallery(gallery_data[0], gallery_data[1],
gallery_data[2], gallery_data[3], gallery_data[4],
                                  gallery_data[5])
                galleries.append(gallery)

            return galleries

        except Exception as e:
            print("Error searching galleries:", e)

    def getGalleryById(self, gallery_id):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()
            sql = "SELECT * FROM Gallery WHERE GalleryID = %s"
            cursor.execute(sql, (gallery_id,))
            gallery_data = cursor.fetchone()

            if gallery_data:
                gallery = Gallery(gallery_data[0], gallery_data[1],
gallery_data[2], gallery_data[3], gallery_data[4],
                                  gallery_data[5])
                return gallery
            else:
                print("Gallery with ID", gallery_id, "not found.")
                return None

        except Exception as e:
            print("Error retrieving gallery by ID:", e)
            return None
    # Artwork_gallery Management
    def addArtworkToGallery(self, gallery_id, artwork_id):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()


            sql_check = "SELECT * FROM Artwork WHERE ArtworkID = %s"
            cursor.execute(sql_check, (artwork_id,))
            artwork_data = cursor.fetchone()

            sql_check = "SELECT * FROM Gallery WHERE GalleryID = %s"
            cursor.execute(sql_check, (gallery_id,))
            gallery_data = cursor.fetchone()

            if artwork_data and gallery_data:
                sql_insert = "INSERT INTO Artwork_Gallery (ArtworkID,
GalleryID) VALUES (%s, %s)"
                cursor.execute(sql_insert, (artwork_id, gallery_id))
                connection.commit()
                print("Artwork added to gallery successfully.")
                cursor.close()
                return True
            else:
                print("Artwork or gallery does not exist.")
                return False
        except Exception as e:
            print("Error adding artwork to gallery:", e)

    def removeArtworkFromGallery(self, gallery_id, artwork_id):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()
            sql_check = "SELECT * FROM Artwork_Gallery WHERE ArtworkID = %s
AND GalleryID = %s"
            cursor.execute(sql_check, (artwork_id, gallery_id))
            link_data = cursor.fetchone()

            if link_data:
                sql_delete = "DELETE FROM Artwork_Gallery WHERE ArtworkID =
%s AND GalleryID = %s"
                cursor.execute(sql_delete, (artwork_id, gallery_id))
                connection.commit()
                cursor.close()
                print("Artwork removed from gallery successfully.")
                return True
            else:
                print("Artwork is not linked to the gallery.")

                return False
        except Exception as e:
            print("Error removing artwork from gallery:", e)

    def getArtworksOfGallery(self, gallery_id):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()
            sql = ("SELECT a.* FROM Artwork a INNER JOIN Artwork_Gallery ag
ON a.ArtworkID = ag.ArtworkID WHERE "
                   "ag.GalleryID = %s")
            cursor.execute(sql, (gallery_id,))
            artworks_data = cursor.fetchall()

            artworks = []
            for artwork_data in artworks_data:
                artwork = Artwork(artwork_data[0], artwork_data[1],
artwork_data[2], artwork_data[3], artwork_data[4],
                                  artwork_data[5], artwork_data[6])
                artworks.append(artwork)

            return artworks

        except Exception as e:
            print("Error retrieving artworks of gallery:", e)

    # Artist management
    def createArtist(self, artist):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()

            sql_insert = ("INSERT INTO Artist (Name, Biography, BirthDate,
Nationality, Website, ContactInformation) "
                          "VALUES (%s, %s, %s, %s, %s, %s)")
            values = (artist.get_name(), artist.get_biography(),
artist.get_birth_date(), artist.get_nationality(),
                      artist.get_website(),
artist.get_contact_information())
            cursor.execute(sql_insert, values)
            connection.commit()
            print("Artist added successfully.")
            artist_id = cursor.lastrowid
            print("Your Artist ID:", artist_id)
            cursor.close()
            return True

        except Exception as e:
            print("Error adding artist:", e)
            return False

    def updateArtist(self, artist):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()
            sql_update = ("UPDATE Artist SET Name = %s, Biography = %s,
BirthDate = %s, Nationality = %s, Website = "
                          "%s, ContactInformation = %s WHERE ArtistID =
%s")
            value = (artist.get_name(), artist.get_biography(),
artist.get_birth_date(), artist.get_nationality(),
                     artist.get_website(),
artist.get_contact_information(), artist.get_artist_id())
            cursor.execute(sql_update, value)
            connection.commit()
            cursor.close()
            print("Artist updated successfully.")
            return True

        except Exception as e:
            print("Error updating artist:", e)

    def deleteArtist(self, artist_id):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()

            sql_check = "SELECT * FROM Artist WHERE ArtistID = %s"
            cursor.execute(sql_check, (artist_id,))
            gallery_data = cursor.fetchone()

            if gallery_data:
                sql_delete = "DELETE FROM Artist WHERE ArtistID = %s"
                cursor.execute(sql_delete, (artist_id,))
                connection.commit()
                cursor.close()
                print("Artist removed successfully.")
                return True
            else:
                print("No artist found.")
        except Exception as e:
            print("Error removing artist:", e)
        return False

    def getArtistById(self, artist_id):
        try:
            connection = DBConnection.getConnection()
            cursor = connection.cursor()

            sql = "SELECT * FROM Gallery WHERE ArtistID = %s"
            cursor.execute(sql, (artist_id,))
            artist_data = cursor.fetchone()

            if artist_data:
                artist = Artist(artist_data[0], artist_data[1],
artist_data[2], artist_data[3], artist_data[4],
                                artist_data[5], artist_data[6])
                return artist
            else:
                print("Artist with ID", artist_id, "not found.")
                return None

        except Exception as e:
            print("Error retrieving gallery by ID:", e)
        return None

def feedback(self, user_id, artwork_id, feedback_text):
    try:
        connection = DBConnection.getConnection()
        cursor = connection.cursor()

        sql_check_user = "SELECT * FROM User WHERE UserID = %s"
        cursor.execute(sql_check_user, (user_id,))
        user_data = cursor.fetchone()
        if user_data is None:
            raise UserNotFoundException(user_id)

        sql_check_artwork = "SELECT * FROM Artwork WHERE ArtworkID =
%s"
        cursor.execute(sql_check_artwork, (artwork_id,))
        artwork_data = cursor.fetchone()
        if artwork_data is None:
            raise ArtWorkNotFoundException(artwork_id)
        if not feedback_text:
            print("Feedback cannot be empty.")
            return False

        query = "SELECT * FROM Feedbacks WHERE UserID = %s AND
ArtworkID = %s"
        cursor.execute(query, (user_id, artwork_id))
        existing_feedback = cursor.fetchone()

        if existing_feedback:
            update_query = "UPDATE Feedbacks SET Feedback = %s WHERE
UserID = %s AND ArtworkID = %s"
            cursor.execute(update_query, (feedback_text, user_id,
artwork_id))
            connection.commit()
            cursor.close()
            print("Feedback updated successfully!")
            return True
        else:
            sql = "INSERT INTO Feedbacks (UserID, ArtworkID, Feedback)
VALUES (%s, %s, %s)"
            values = (user_id, artwork_id, feedback_text)
            cursor.execute(sql, values)
            connection.commit()
            cursor.close()
            print("Feedback added successfully!")
            return True

    except Exception as e:
        print("Error adding feedback:", e)
        return False

def getFeedbacksByArtworkId(self,artwork_id):
    try:
        connection = DBConnection.getConnection()
        cursor = connection.cursor(dictionary=True)

        cursor.execute("SELECT * FROM Artwork WHERE ArtworkID = %s",
(artwork_id,))
        artwork = cursor.fetchone()
        if not artwork:
            print(f"Artwork with ID {artwork_id} does not exist.")
            return []
        cursor.execute("SELECT UserID, Feedback FROM Feedbacks WHERE
ArtworkID = %s", (artwork_id,))
        feedbacks = cursor.fetchall()

        cursor.close()
        return feedbacks

    except Exception as e:
        print("Error retrieving feedbacks:", e)
        return []
```

Main module to Implement all the methods:

MainModule class:

```
from dao.VirtualArtGalleryImpl import VirtualArtGalleryImpl
from entity.ArtWorks import Artwork


class MainModule:
    @staticmethod
    def display_menu():
        print("-------------------------------------")
        print("Virtual Art Gallery Menu")
        print("1. Add Artwork")
        print("2. Update Artwork")
        print("3. Remove Artwork")
        print("4. Get Artwork by ID")
        print("5. Search Artworks keyword")
        print("6. Add Artwork to Favorites")
        print("7. Remove Artwork from Favorites")
        print("8. Get User Favorite Artworks")
        print("9. Give Feedback")
        print("10. Show Feedbacks for a Artwork")
        print("0. Exit")
        print("-------------------------------------")

    @staticmethod
    def add_artwork(virtual_gallery):
        title = input("Enter title : ")
        description = input("Enter description: ")
        creation_date = input("Enter creation_date(in yyyy-mm-dd format):
")
        medium = input("Enter medium: ")
        image_url = input("Enter image_url: ")
        artist_id = input("Enter artist_id: ")
        artwork = Artwork(None, title, description, creation_date, medium,
image_url, artist_id)
        virtual_gallery.addArtwork(artwork)

    @staticmethod
    def update_artwork(virtual_gallery):
        artwork_id = input("Enter the ID of the artwork you want to update:
")
        artwork = virtual_gallery.getArtworkById(artwork_id)
        if artwork is None:
            print("Artwork with ID", artwork_id, "not found.")
            return

        title = input("Enter the new title for the artwork (press Enter to
keep current title): ")
        description = input("Enter the new description for the artwork
(press Enter to keep current description): ")
        creation_date = input("Enter the new creation date for the artwork
(press Enter to keep current date): ")
        medium = input("Enter the new medium for the artwork (press Enter
to keep current medium): ")
        image_url = input("Enter the new image URL for the artwork (press
Enter to keep current URL): ")

        if title:
            artwork.set_title(title)
        if description:
            artwork.set_description(description)
        if creation_date:
            artwork.set_creation_date(creation_date)
        if medium:
            artwork.set_medium(medium)
        if image_url:
            artwork.set_image_url(image_url)

        virtual_gallery.updateArtwork(artwork)
        #print("Artwork updated successfully!")

    @staticmethod
    def remove_artwork(virtual_gallery):
        artwork_id = input("Enter the ID of the artwork you want to remove:
")

        removed = virtual_gallery.removeArtwork(artwork_id)
        if removed:
            print("---------------Artwork removed successfully!------------
---")
        else:
            print("Failed to remove artwork. Please check the artwork ID.")

    @staticmethod
    def get_artwork_by_id(virtual_gallery):
        artwork_id = input("Enter the ID of the artwork you want to
retrieve: ")
        artwork = virtual_gallery.getArtworkById(artwork_id)
        if artwork:
            print("Artwork details:")
            print("ID:", artwork.get_artwork_id())
            print("Title:", artwork.get_title())
            print("Description:", artwork.get_description())
            print("Creation Date:", artwork.get_creation_date())
            print("Medium:", artwork.get_medium())
            print("Image URL:", artwork.get_image_url())
            print("Artist ID:", artwork.get_artist_id())

        else:
            print("Artwork with ID", artwork_id, "not found.")

    @staticmethod
    def search_artworks(virtual_gallery):
        keyword = input("Enter the keyword to search artworks: ")

        artworks = virtual_gallery.searchArtworks(keyword)
        if artworks:
            print("Matching artworks:")

            for artwork in artworks:
                print("---------------------------------------")
                print("ID:              ", artwork.get_artwork_id())
                print("Title:           ", artwork.get_title())
                print("Description:     ", artwork.get_description())
                print("Creation Date:   ", artwork.get_creation_date())
                print("Medium:          ", artwork.get_medium())
                print("Image URL:       ", artwork.get_image_url())
                print("---------------------------------------")
        else:
            print("No artworks found matching the keyword:", keyword)

    @staticmethod
    def add_artwork_to_favorites(virtual_gallery):

        user_id = input("Enter your user ID: ")
        virtual_gallery.showAllArtworks()
        artwork_id = input("Enter the ID of the artwork you want to add to
favorites: ")

        virtual_gallery.addArtworkToFavorite(user_id, artwork_id)

    @staticmethod
    def remove_artwork_from_favorites(virtual_gallery):
        user_id = input("Enter your user ID: ")
        artwork_id = input("Enter the ID of the artwork you want to remove
from favorites: ")
        virtual_gallery.removeArtworkFromFavorite(user_id, artwork_id)

    @staticmethod
    def get_user_favorite_artworks(virtual_gallery):
        user_id = input("Enter your user ID: ")
        virtual_gallery.getUserFavoriteArtworks(user_id)
    @staticmethod
    def give_feedback(virtual_gallery):
        user_id = input("Enter your user ID: ")
        virtual_gallery.showAllArtworks()
        artwork_id = input("Enter the ID of the artwork for which you
want to give feedback: ")
        feedback_text = input("Enter your feedback: ")
        virtual_gallery.feedback(user_id, artwork_id, feedback_text)

    @staticmethod
    def show_feedbacks(virtual_gallery):
        artwork_id = input("Enter the ID of the artwork: ")
        feedbacks = virtual_gallery.getFeedbacksByArtworkId(artwork_id)
        if feedbacks:
                print(f"Feedbacks for Artwork ID {artwork_id}:")
                for feedback in feedbacks:
                #print(feedback)
                print("-----------------------------------------------------
------------")
                print(f"User ID: {feedback['UserID']}")
                print(f"Feedback: {feedback['Feedback']}")
                print("-----------------------------------------------------
------------")


    @staticmethod
    def main():
        virtual_gallery = VirtualArtGalleryImpl()
        while True:
            MainModule.display_menu()
            choice = input("Enter your choice: ")

            if choice == "1":
                MainModule.add_artwork(virtual_gallery)
            elif choice == "2":
                MainModule.update_artwork(virtual_gallery)
            elif choice == "3":
                MainModule.remove_artwork(virtual_gallery)
            elif choice == "4":
                MainModule.get_artwork_by_id(virtual_gallery)
            elif choice == "5":
                MainModule.search_artworks(virtual_gallery)
            elif choice == "6":
                MainModule.add_artwork_to_favorites(virtual_gallery)
            elif choice == "7":
                MainModule.remove_artwork_from_favorites(virtual_gallery)
            elif choice == "8":
                MainModule.get_user_favorite_artworks(virtual_gallery)
            elif choice == "9":
                MainModule.give_feedback(virtual_gallery)
            elif choice == "10":
                MainModule.show_feedbacks(virtual_gallery)

            elif choice == "0":
                print("Exiting the program...")
                break
            else:
                print("Invalid choice. Please enter a valid option.")


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

Implementations of additional Functions:

```
from dao.AdditionalFunctionImpl import AdditionalFunctionImpl
from entity.Artists import Artist
from entity.ArtWorks import Artwork
from entity.Gallery import Gallery


class MainDriver:
    @staticmethod
    def display_menu():
        print("-----------------------------------")
        print("Virtual Art Gallery Menu")
        print("1. Add Gallery")
        print("2. Update Gallery")
        print("3. Remove Gallery")
        print("4. Search Galleries")
        print("5. Get Gallery by ID")
        print("6. Add Artwork to Gallery")
        print("7. Remove Artwork from Gallery")
        print("8. Get Artworks of Gallery")
        print("9. Create Artist")
        print("10. Update Artist")
        print("11. Delete Artist")
        print("12. Get Artist by ID")
        print("0. Exit")
        print("-----------------------------------")

    @staticmethod
    def main():
        additional_functions = AdditionalFunctionImpl()

        while True:
            MainDriver.display_menu()
            choice = input("Enter your choice: ")

            if choice == "1":
                name = input("Enter gallery name: ")
                description = input("Enter gallery description: ")
                location = input("Enter gallery location: ")
                curator = input("Enter gallery Artist ID: ")
                opening_hours = input("Enter gallery opening hours: ")
                gallery = Gallery(None, name, description, location,
curator, opening_hours)
                additional_functions.addGallery(gallery)

            elif choice == "2":
                gallery_id = input("Enter the ID of the gallery you want to
update: ")

                gallery = additional_functions.getGalleryById(gallery_id)

                if gallery:
                    new_name = input(f"Enter new name for gallery (current:
{gallery.name})(press Enter to keep "
                                     f"current title): ")
                    new_description = input(f"Enter new description for
gallery (current: {gallery.description})("
                                            f"press Enter to keep current
title): ")
                    new_location = input(f"Enter new location for gallery
(current: {gallery.location})(press Enter "
                                         f"to keep current title): ")
                    new_curator = input(f"Enter new curator for gallery
(current: {gallery.curator})(press Enter to "
                                        f"keep current title): ")
                    new_opening_hours = input(
                        f"Enter new opening hours for gallery (current:
{gallery.opening_hours})(press Enter to keep "
                        f"current title): ")

                    if new_name:
                        gallery.set_name(new_name)
                    if new_description:
                        gallery.set_description(new_description)
                    if new_location:
                        gallery.set_location(new_location)
                    if new_curator:
                        gallery.set_curator(new_curator)
                    if new_opening_hours:
                        gallery.set_opening_hours(new_opening_hours)

                    additional_functions.updateGallery(gallery)
                    print("Gallery updated successfully!")
                else:
                    print("Gallery not found.")

            elif choice == "3":
                gallery_id = input("Enter gallery ID to remove: ")
                additional_functions.removeGallery(gallery_id)

            elif choice == "4":

                keyword = input("Enter keyword to search galleries: ")
                additional_functions.searchGalleries(keyword)
            elif choice == "5":
                gallery_id = input("Enter gallery ID: ")
                gallery = additional_functions.getGalleryById(gallery_id)
                print("Gallery name:", gallery.get_name(), "\nGallery
opening hours:", gallery.get_opening_hours())

            elif choice == "6":
                gallery_id = input("Enter gallery ID: ")
                artwork_id = input("Enter artwork ID: ")
                additional_functions.addArtworkToGallery(gallery_id,
artwork_id)

            elif choice == "7":
                gallery_id = input("Enter gallery ID: ")
                artwork_id = input("Enter artwork ID: ")
                additional_functions.removeArtworkFromGallery(gallery_id,
artwork_id)

            elif choice == "8":
                gallery_id = input("Enter gallery ID: ")
                additional_functions.getArtworksOfGallery(gallery_id)

            elif choice == "9":
                name = input("Enter artist name: ")
                biography = input("Enter artist biography: ")
                birth_date = input("Enter artist birth date (YYYY-MM-DD):
")
                nationality = input("Enter artist nationality: ")
                website = input("Enter artist website: ")
                contact_information = input("Enter artist contact
information: ")
                artist = Artist(None, name, biography, birth_date,
nationality, website, contact_information)

                additional_functions.createArtist(artist)

            elif choice == "10":
                artist_id = input("Enter the ID of the artist you want to
update: ")
                artist = additional_functions.getArtistById(artist_id)
                if artist is None:
                    print("Artist with ID", artist_id, "not found.")
                else:
                    new_name = input("Enter the new name for the artist
(press Enter to keep current name): ")
                    new_biography = input(
                        "Enter the new biography for the artist (press
Enter to keep current biography): ")
                    new_birth_date = input(
                        "Enter the new birth date for the artist (press
Enter to keep current birth date): ")
                    new_nationality = input(
                        "Enter the new nationality for the artist (press
Enter to keep current nationality): ")
                    new_website = input("Enter the new website for the
artist (press Enter to keep current website): ")
                    new_contact_information = input(
                        "Enter the new contact information for the artist
(press Enter to keep current contact "
                        "information): ")
                    if new_name:
                        artist.set_name(new_name)
                    if new_biography:
                        artist.set_biography(new_biography)
                    if new_birth_date:
                        artist.set_birth_date(new_birth_date)
                    if new_nationality:
                        artist.set_nationality(new_nationality)
                    if new_website:
                        artist.set_website(new_website)
                    if new_contact_information:

artist.set_contact_information(new_contact_information)

                    if additional_functions.updateArtist(artist):
                        print("Artist updated successfully!")
                    else:
                        print("Failed to update artist. Please try again.")

            elif choice == "11":
                artist_id = input("Enter the ID of the artist you want to
delete: ")
                if additional_functions.deleteArtist(artist_id):
                    print("Artist deleted successfully!")
                else:
                    print("Failed to delete artist. Please check the artist
ID.")

            elif choice == "12":

                artist_id = input("Enter artist ID: ")
                artist = additional_functions.getArtistById(artist_id)
                print("Artist name: ", artist.get_name(), "\nArtist
Website:", artist.get_website())


            elif choice == "0":
                print("Exiting the program...")
                break

            else:
                print("Invalid choice. Please enter a valid option.")


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

Exceptions:

ArtWorkNotFoundException:

```
class ArtWorkNotFoundException(Exception):
    def __init__(self, artwork_id):
        self.artwork_id = artwork_id

    def __str__(self):
        return f"Artwork with ID {self.artwork_id} not found in the
database"
```

UserNotFoundException:

```
class UserNotFoundException(Exception):
    def __init__(self, user_id):
        self.user_id = user_id

    def __str__(self):
        return f"User with ID {self.user_id} not found in the database"
```

Test_VirtualArtGallery:

```
import pytest
from unittest.mock import patch, Mock

from dao.AdditionalFunctionImpl import AdditionalFunctionImpl
from dao.VirtualArtGalleryImpl import VirtualArtGalleryImpl
from entity.ArtWorks import Artwork
from entity.Gallery import Gallery


def test_add_artwork():
    virtual_gallery = VirtualArtGalleryImpl()
    artwork = Artwork(None, "Test Artwork", "Test Description", "2024-02-
10", "Oil on canvas", "test_image_url", 1)
    result = virtual_gallery.addArtwork(artwork)
    assert result == True


def test_update_artwork():
    virtual_gallery = VirtualArtGalleryImpl()
    artwork = Artwork(6, "Test Artwork", "Test Description new", "2024-02-
10", "Oil on canvas", "test_image_url", 1)
    result = virtual_gallery.updateArtwork(artwork)
    assert result == True


def test_remove_artwork():
    virtual_gallery = VirtualArtGalleryImpl()
    removed = virtual_gallery.removeArtwork(6)
    assert removed == True


def test_search_artwork():
    virtual_gallery = VirtualArtGalleryImpl()
    result = virtual_gallery.searchArtworks(1)
    assert result is not None


def test_create_gallery():
    additional_functions = AdditionalFunctionImpl()
    gallery = Gallery(None, 'Test gallery Name', 'Test description', 'Test
Location', 'Test Curator',
                      '10:00 AM - 6:00 PM')
    result = additional_functions.addGallery(gallery)
    assert result == True
```

```python
def test_update_gallery():
    additional_functions = AdditionalFunctionImpl()
    gallery = Gallery(None, 'Test gallery Name', 'Test description', 'Test
Location', 'Test Curator',
                      '10:00 AM - 6:00 PM')
    result = additional_functions.updateGallery(gallery)
    assert result == True


def test_remove_gallery():
    additional_functions = AdditionalFunctionImpl()
    result = additional_functions.removeGallery(6)
    assert result == True


def test_search_gallery():
    additional_functions = AdditionalFunctionImpl()
    result = additional_functions.getGalleryById(1)
    assert result is not None
```

Util classes:

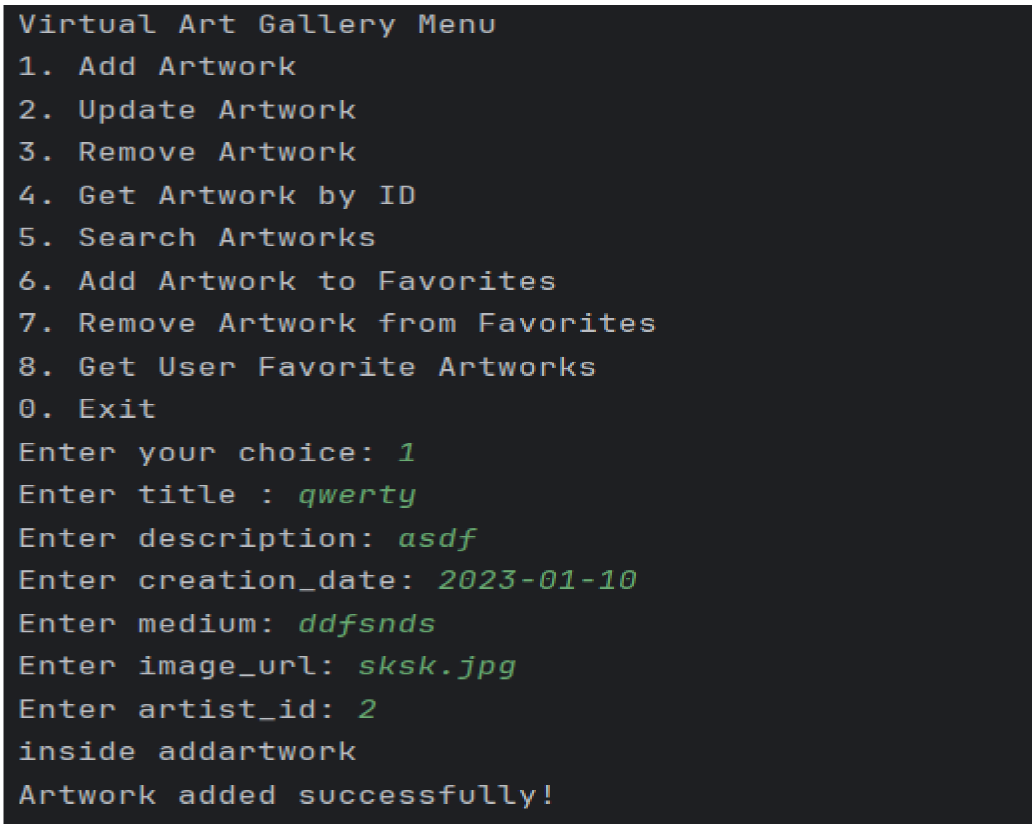
DBConnection:

```python
import mysql.connector

from util.PropertyUtil import PropertyUtil


class DBConnection:
    connection = None

    @staticmethod
    def getConnection():
        if DBConnection.connection is None:
            connection_parameters = PropertyUtil.getPropertyString()
            DBConnection.connection =
mysql.connector.connect(**connection_parameters)
            # if DBConnection.connection.is_connected():
            #     print("Connected to MySQL database")
        return DBConnection.connection
```

PropertyUtil class:

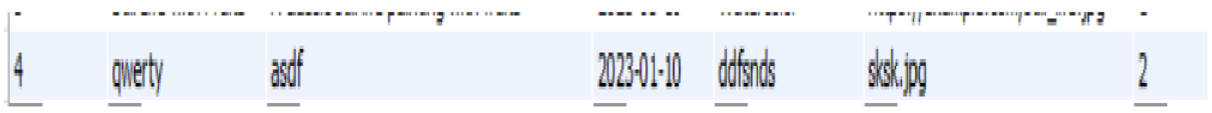
```python
class PropertyUtil:
    @staticmethod
    def getPropertyString():
        with open("../database_properties.txt", "r") as file:
            properties = {}
            for line in file:
                key, value = line.strip().split("=")
                properties[key.strip()] = value.strip()
        return properties


"""print(PropertyUtil.getPropertyString())"""
```

Outputs of Working System:

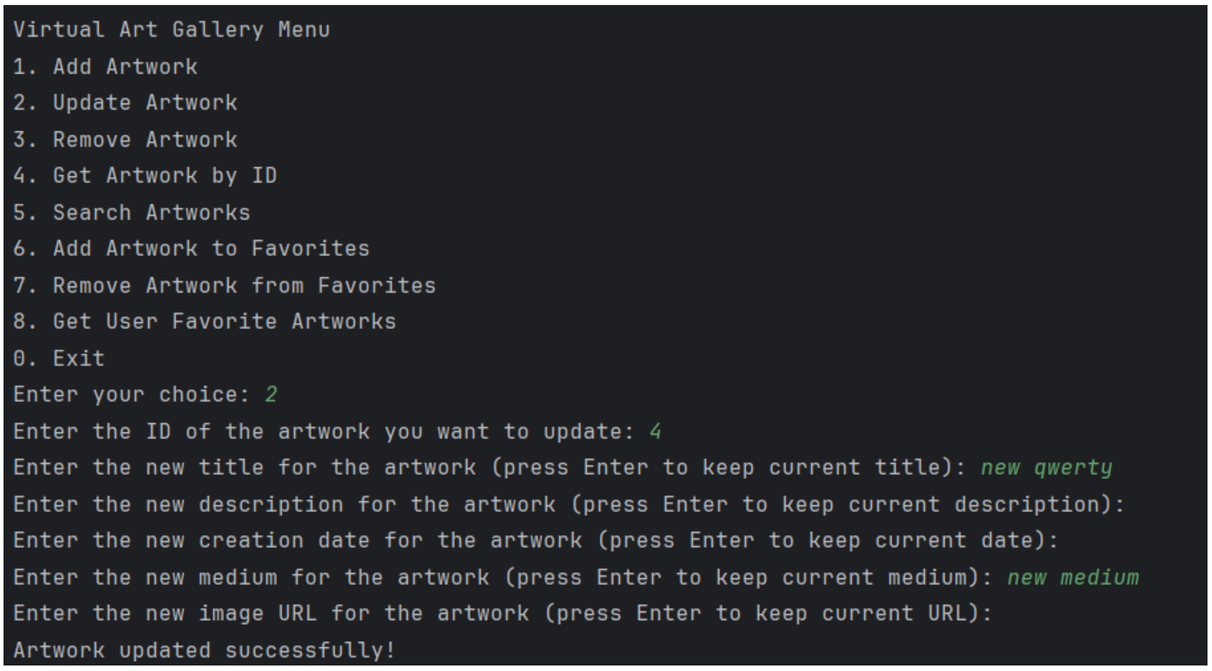


```
Virtual Art Gallery Menu
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. Get User Favorite Artworks
0. Exit
Enter your choice: 1
Enter title : qwerty
Enter description: asdf
Enter creation_date: 2023-01-10
Enter medium: ddfsnds
Enter image_url: sksk.jpg
Enter artist_id: 2
inside addartwork
Artwork added successfully!
```

Updated database

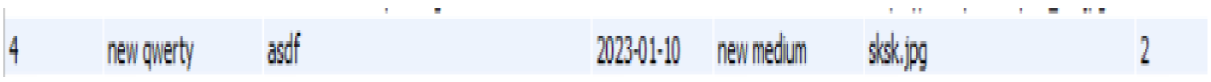


4	qwerty	asdf	2023-01-10	ddfsnds	sksk.jpg	2

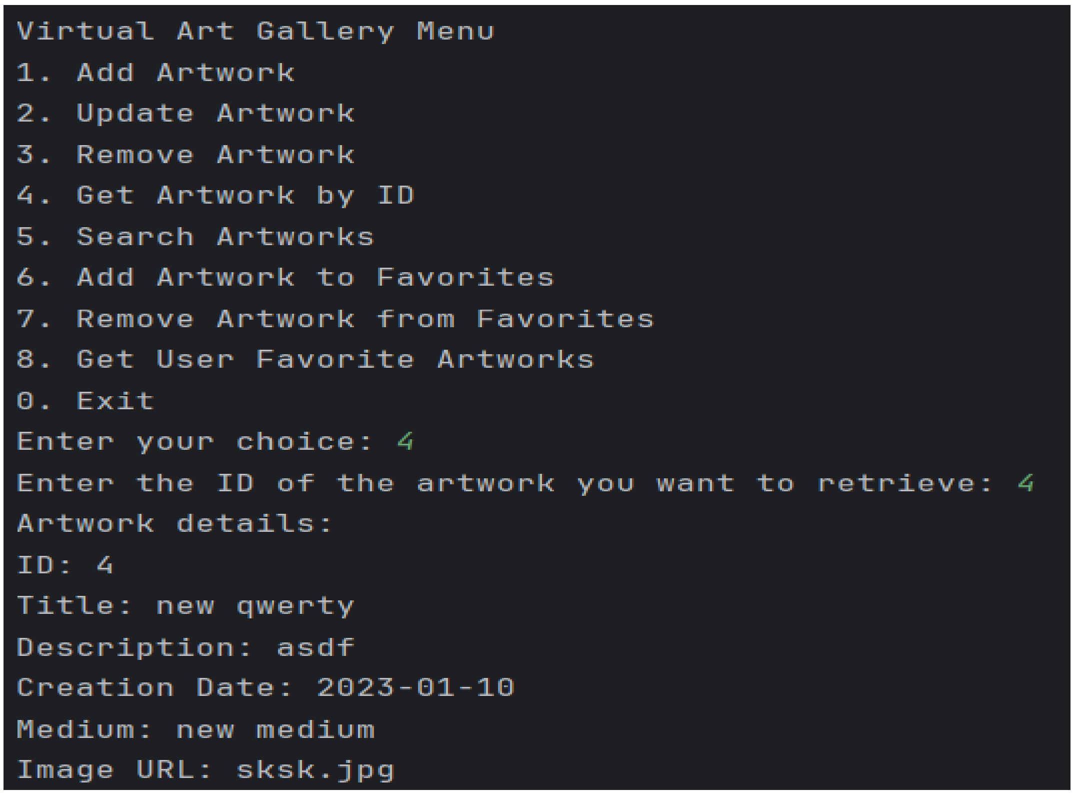


```
Virtual Art Gallery Menu
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. Get User Favorite Artworks
0. Exit
Enter your choice: 2
Enter the ID of the artwork you want to update: 4
Enter the new title for the artwork (press Enter to keep current title): new qwerty
Enter the new description for the artwork (press Enter to keep current description):
Enter the new creation date for the artwork (press Enter to keep current date):
Enter the new medium for the artwork (press Enter to keep current medium): new medium
Enter the new image URL for the artwork (press Enter to keep current URL):
Artwork updated successfully!
```

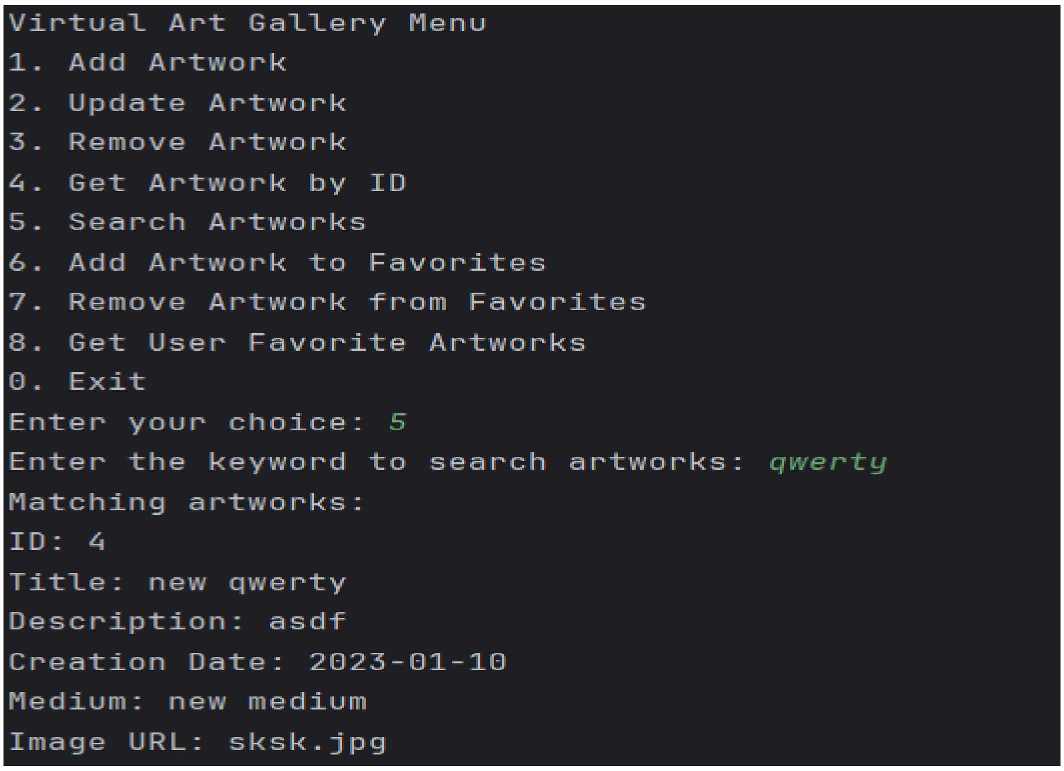
Updated database:



4	new qwerty	asdf	2023-01-10	new medium	sksk.jpg	2

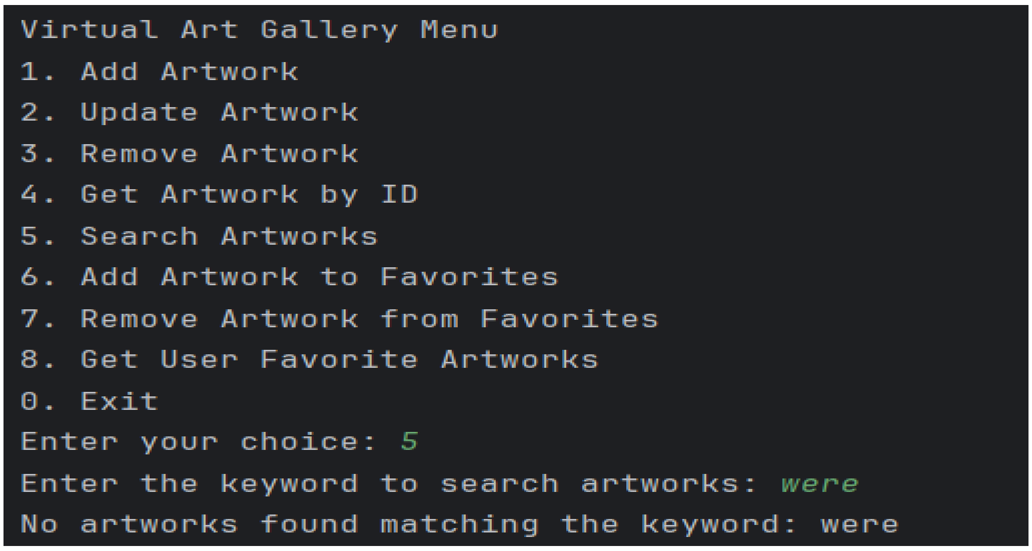


```
Virtual Art Gallery Menu
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. Get User Favorite Artworks
0. Exit
Enter your choice: 4
Enter the ID of the artwork you want to retrieve: 4
Artwork details:
ID: 4
Title: new qwerty
Description: asdf
Creation Date: 2023-01-10
Medium: new medium
Image URL: sksk.jpg
```

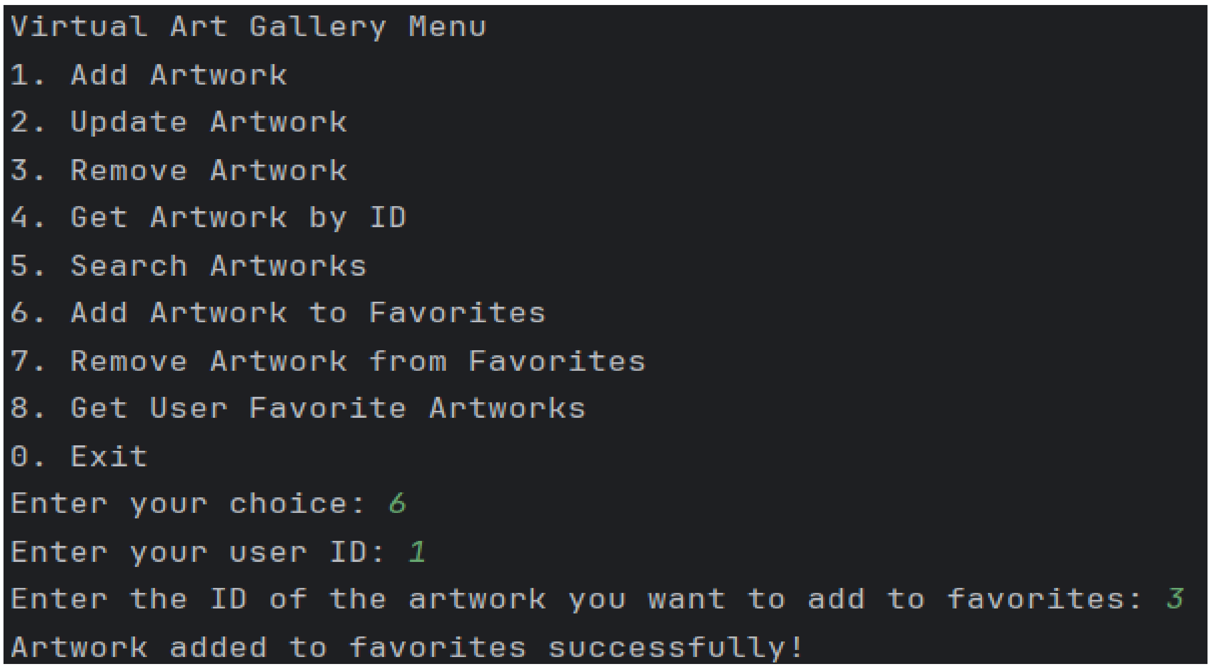


```
Virtual Art Gallery Menu
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. Get User Favorite Artworks
0. Exit
Enter your choice: 5
Enter the keyword to search artworks: qwerty
Matching artworks:
ID: 4
Title: new qwerty
Description: asdf
Creation Date: 2023-01-10
Medium: new medium
Image URL: sksk.jpg
```

If no such artwork

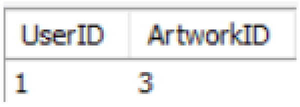


```
Virtual Art Gallery Menu
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. Get User Favorite Artworks
0. Exit
Enter your choice: 5
Enter the keyword to search artworks: were
No artworks found matching the keyword: were
```

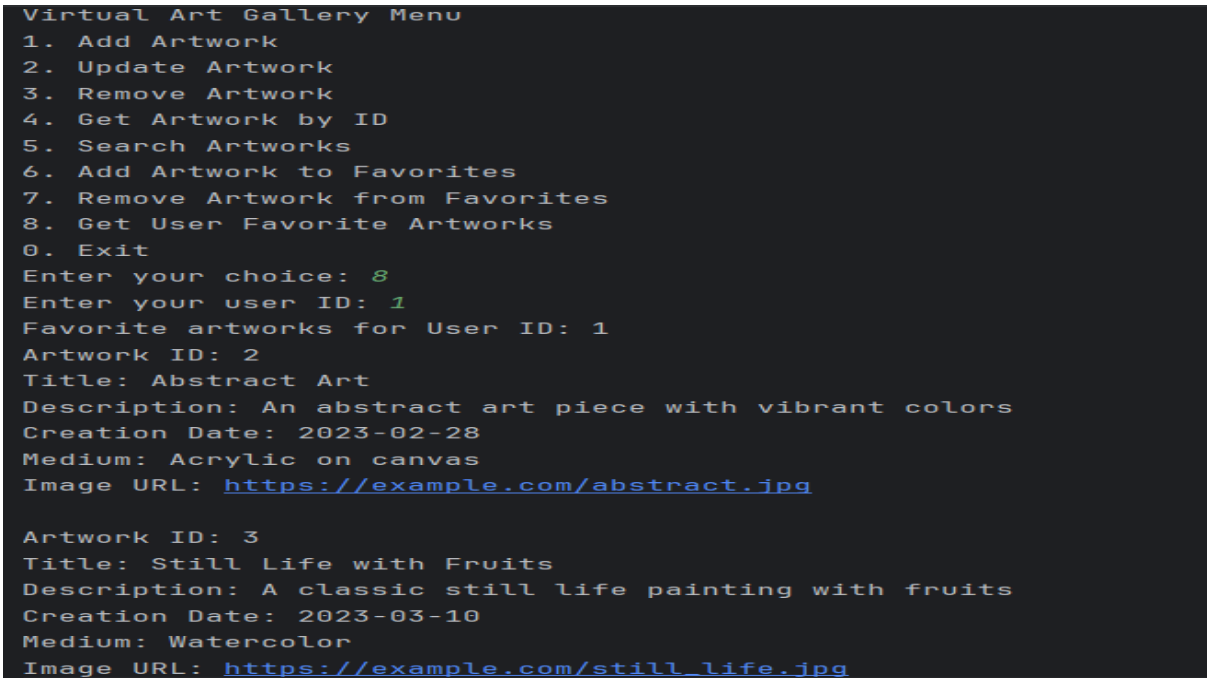


```
Virtual Art Gallery Menu
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. Get User Favorite Artworks
0. Exit
Enter your choice: 6
Enter your user ID: 1
Enter the ID of the artwork you want to add to favorites: 3
Artwork added to favorites successfully!
```

Updated database user favorite artworks

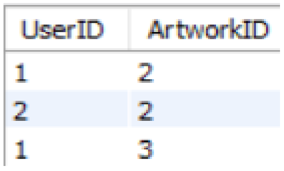


UserID	ArtworkID
1	3



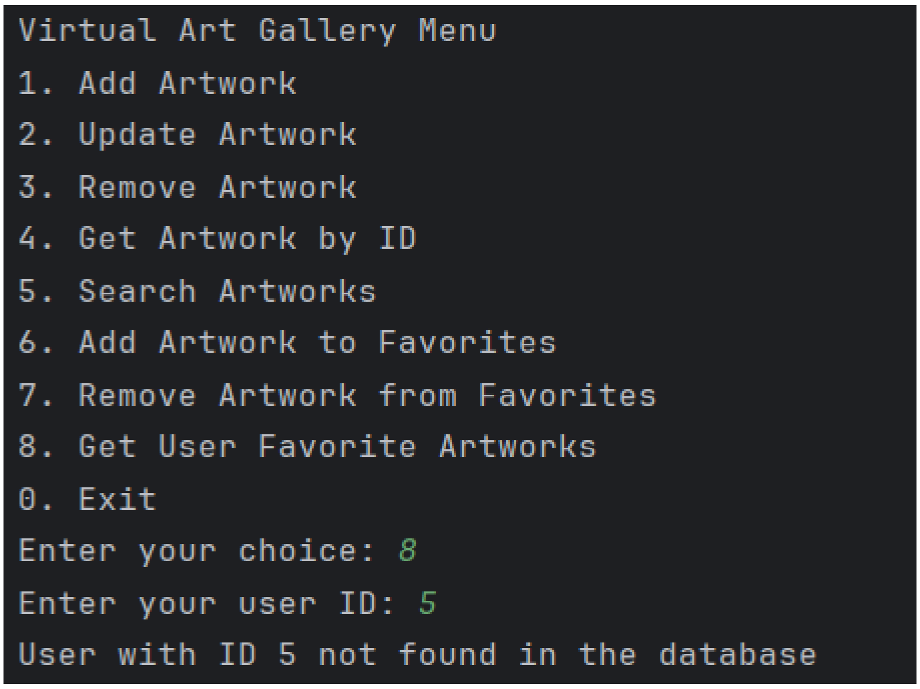
```
Virtual Art Gallery Menu
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. Get User Favorite Artworks
0. Exit
Enter your choice: 8
Enter your user ID: 1
Favorite artworks for User ID: 1
Artwork ID: 2
Title: Abstract Art
Description: An abstract art piece with vibrant colors
Creation Date: 2023-02-28
Medium: Acrylic on canvas
Image URL: https://example.com/abstract.jpg

Artwork ID: 3
Title: Still Life with Fruits
Description: A classic still life painting with fruits
Creation Date: 2023-03-10
Medium: Watercolor
Image URL: https://example.com/still_life.jpg
```

Data in user favorite artworks table

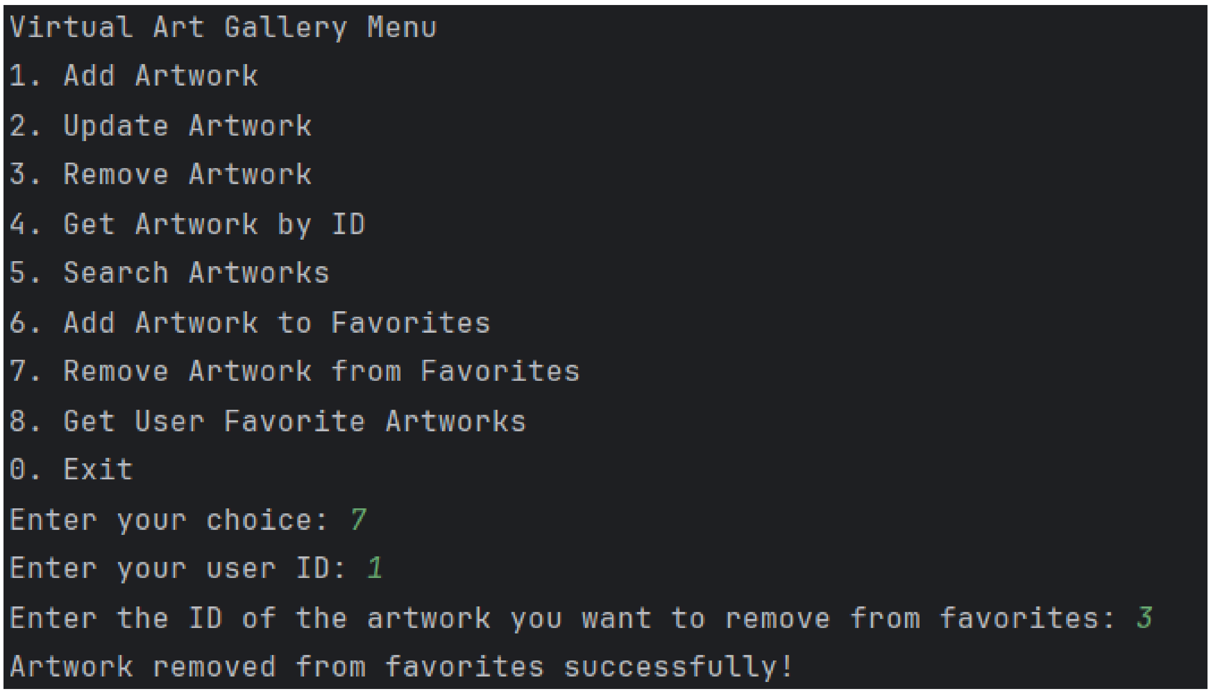


UserID	ArtworkID
1	2
2	2
1	3

If user don't exists

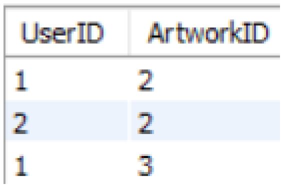


```
Virtual Art Gallery Menu
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. Get User Favorite Artworks
0. Exit
Enter your choice: 8
Enter your user ID: 5
User with ID 5 not found in the database
```



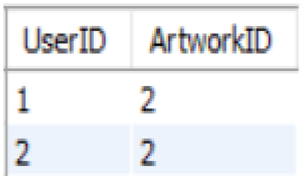
```
Virtual Art Gallery Menu
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. Get User Favorite Artworks
0. Exit
Enter your choice: 7
Enter your user ID: 1
Enter the ID of the artwork you want to remove from favorites: 3
Artwork removed from favorites successfully!
```

Database before:



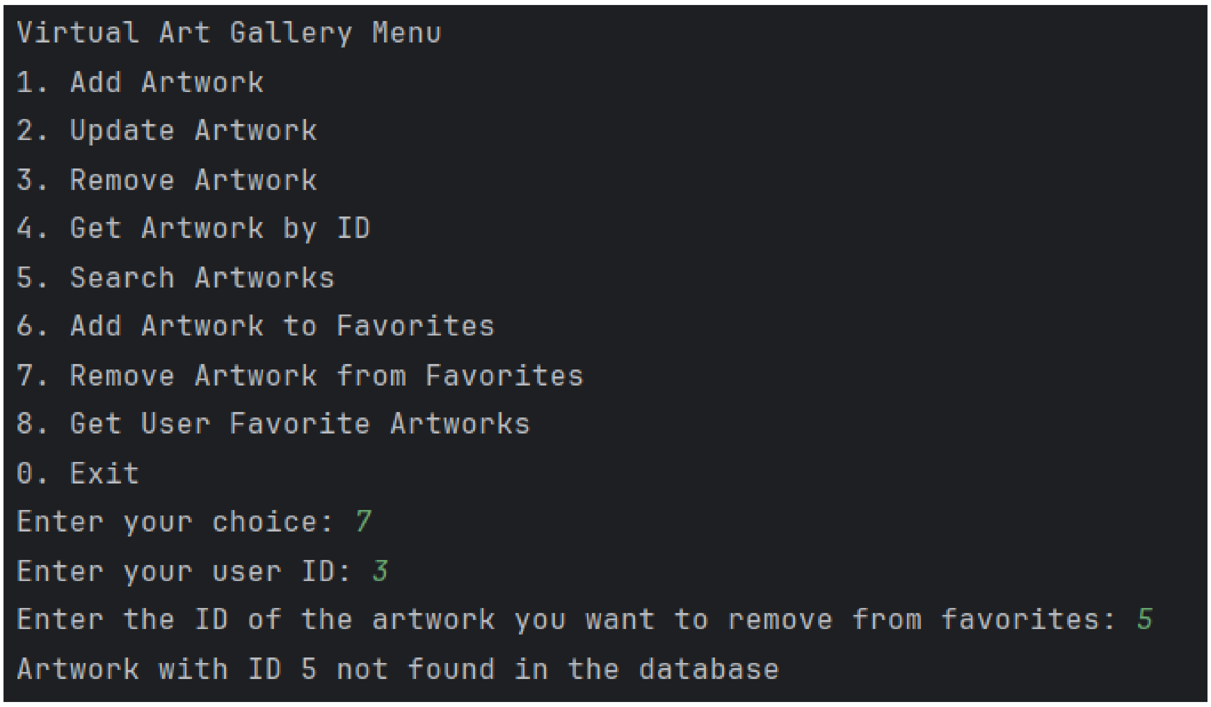
UserID	ArtworkID
1	2
2	2
1	3

Database after execution of code:



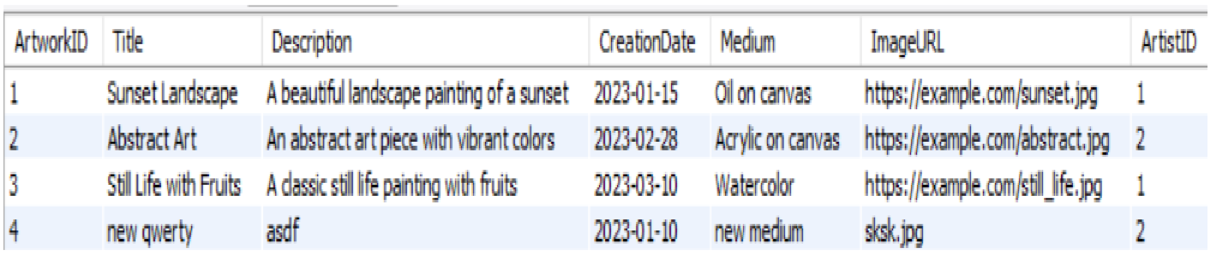
UserID	ArtworkID
1	2
2	2

When given random inputs to remove artwork from favorites:

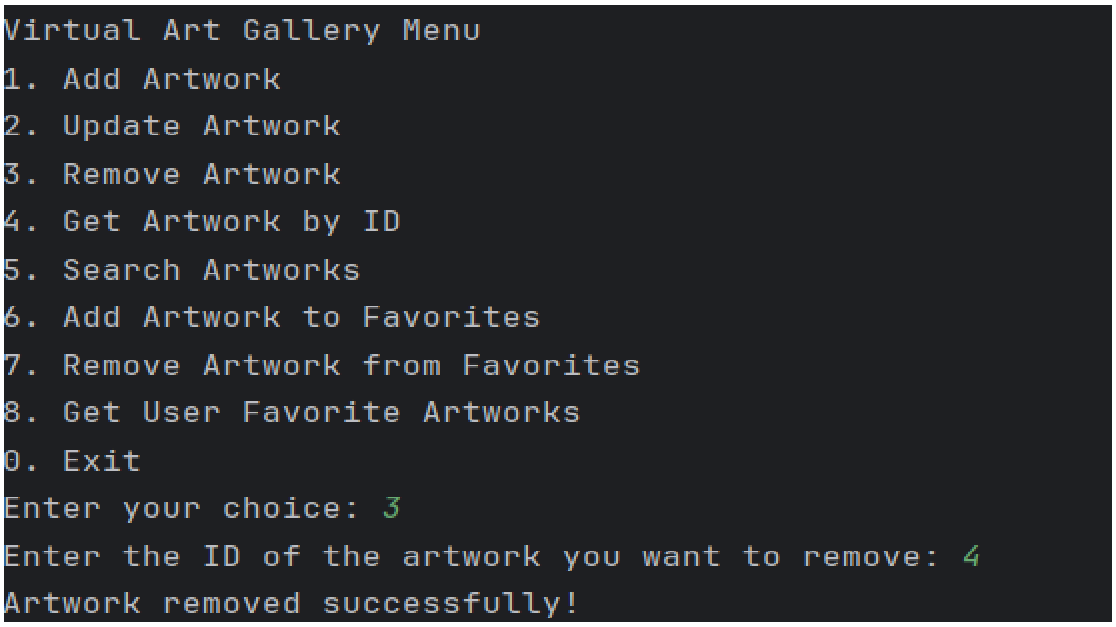


```
Virtual Art Gallery Menu
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. Get User Favorite Artworks
0. Exit
Enter your choice: 7
Enter your user ID: 3
Enter the ID of the artwork you want to remove from favorites: 5
Artwork with ID 5 not found in the database
```

Artwork table before deletion

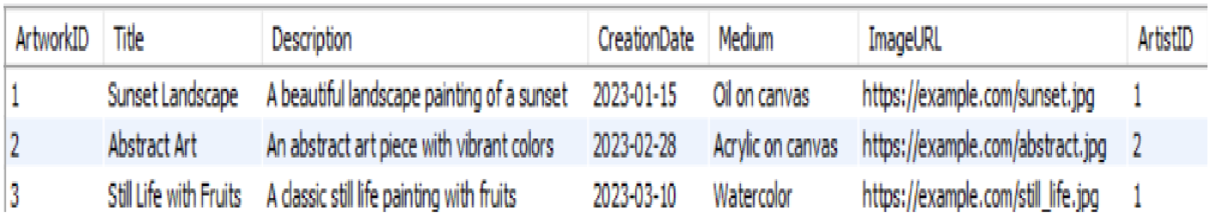


ArtworkID	Title	Description	CreationDate	Medium	ImageURL	ArtistID
1	Sunset Landscape	A beautiful landscape painting of a sunset	2023-01-15	Oil on canvas	https://example.com/sunset.jpg	1
2	Abstract Art	An abstract art piece with vibrant colors	2023-02-28	Acrylic on canvas	https://example.com/abstract.jpg	2
3	Still Life with Fruits	A classic still life painting with fruits	2023-03-10	Watercolor	https://example.com/still_life.jpg	1
4	new qwerty	asdf	2023-01-10	new medium	sksk.jpg	2



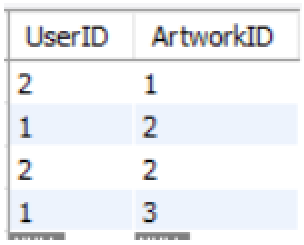
```
Virtual Art Gallery Menu
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. Get User Favorite Artworks
0. Exit
Enter your choice: 3
Enter the ID of the artwork you want to remove: 4
Artwork removed successfully!
```

After deletion database:

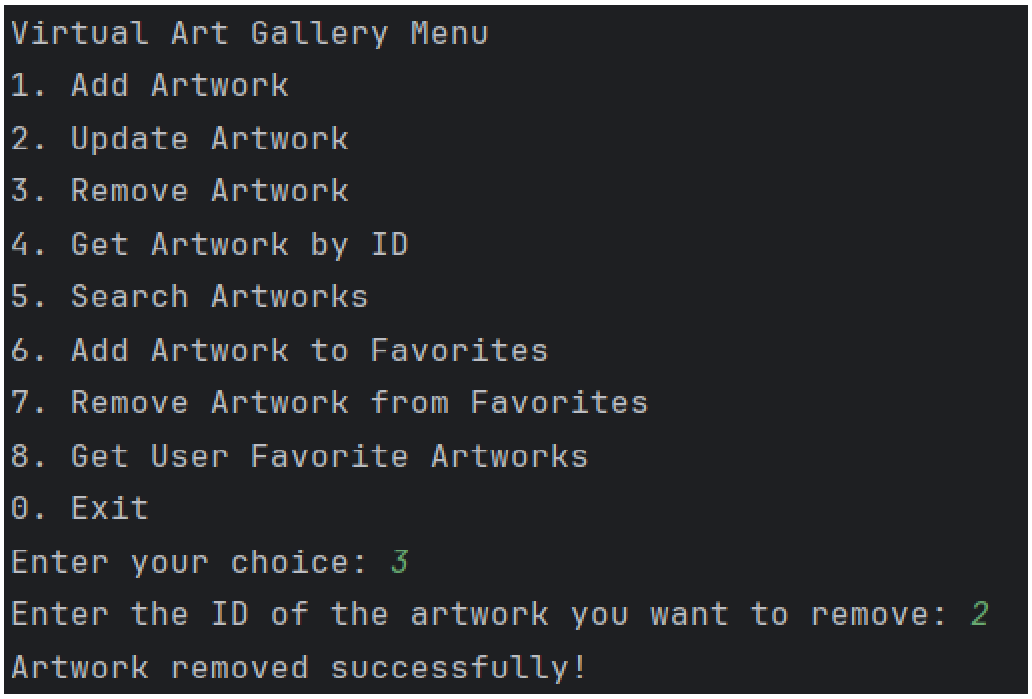


ArtworkID	Title	Description	CreationDate	Medium	ImageURL	ArtistID
1	Sunset Landscape	A beautiful landscape painting of a sunset	2023-01-15	Oil on canvas	https://example.com/sunset.jpg	1
2	Abstract Art	An abstract art piece with vibrant colors	2023-02-28	Acrylic on canvas	https://example.com/abstract.jpg	2
3	Still Life with Fruits	A classic still life painting with fruits	2023-03-10	Watercolor	https://example.com/still_life.jpg	1

User favorite artwork table before deleting an artwork:

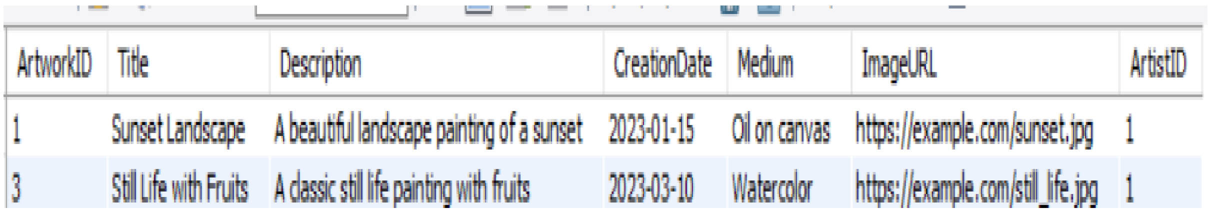


UserID	ArtworkID
2	1
1	2
2	2
1	3



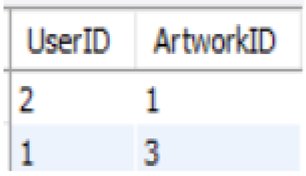
```
Virtual Art Gallery Menu
1. Add Artwork
2. Update Artwork
3. Remove Artwork
4. Get Artwork by ID
5. Search Artworks
6. Add Artwork to Favorites
7. Remove Artwork from Favorites
8. Get User Favorite Artworks
0. Exit
Enter your choice: 3
Enter the ID of the artwork you want to remove: 2
Artwork removed successfully!
```

User favorite artwork table after deleting an artwork:



ArtworkID	Title	Description	CreationDate	Medium	ImageURL	ArtistID
1	Sunset Landscape	A beautiful landscape painting of a sunset	2023-01-15	Oil on canvas	https://example.com/sunset.jpg	1
3	Still Life with Fruits	A classic still life painting with fruits	2023-03-10	Watercolor	https://example.com/still_life.jpg	1

When an artwork is removed then all its references also get deleted



UserID	ArtworkID
2	1
1	3

Submitted By – Asutosh Mishra