Name: Gayathri G

Case Study: Virtual Art Gallery

Schema design:

Entities:

- Designing the schema for a Virtual Art Gallery involves creating a structured representation of the database that will store information about artworks, artists, users, galleries, and various relationships between them. Below is a schema design for a Virtual Art Gallery database:
- **Entities and Attributes:**
- **Artwork**

  ArtworkID (Primary Key)

  Title

  Description

  CreationDate

  Medium

  ImageURL (or any reference to the digital representation)
- **Artist**

  ArtistID (Primary Key)

  Name

  Biography

  BirthDate

  Nationality

  Website

  Contact Information
- **User**

  UserID (Primary Key)

  Username

  Password

  Email

First Name

Last Name

Date of Birth

Profile Picture

FavoriteArtworks (a list of references to ArtworkIDs)

- **Gallery**

GalleryID (Primary Key)

Name

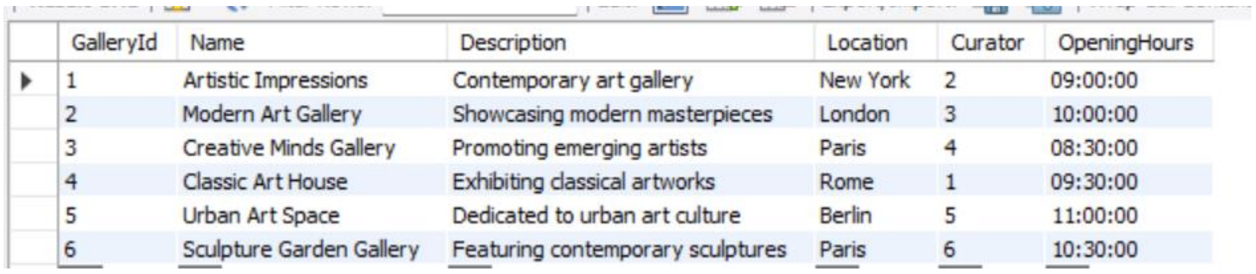
Description

Location

Curator (Reference to ArtistID)

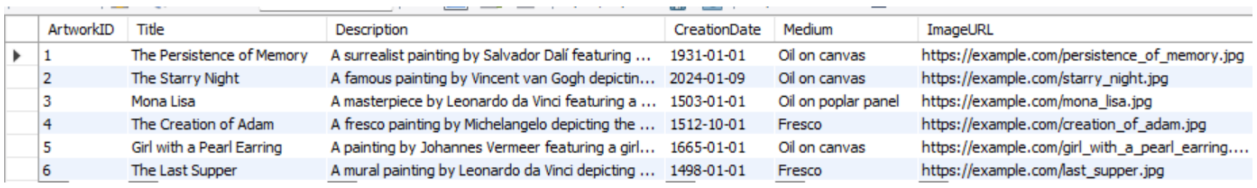
OpeningHours

TABLE Gallery



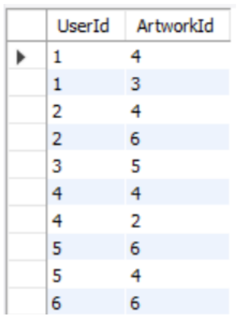
GalleryId	Name	Description	Location	Curator	OpeningHours
1	Artistic Impressions	Contemporary art gallery	New York	2	09:00:00
2	Modern Art Gallery	Showcasing modern masterpieces	London	3	10:00:00
3	Creative Minds Gallery	Promoting emerging artists	Paris	4	08:30:00
4	Classic Art House	Exhibiting classical artworks	Rome	1	09:30:00
5	Urban Art Space	Dedicated to urban art culture	Berlin	5	11:00:00
6	Sculpture Garden Gallery	Featuring contemporary sculptures	Paris	6	10:30:00

TABLE Artwork



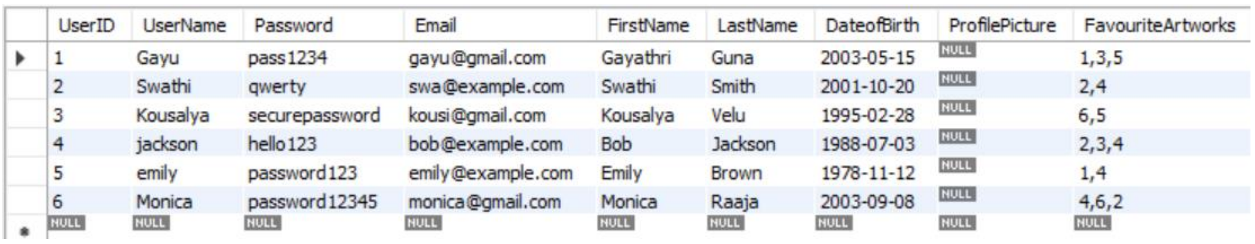
ArtworkID	Title	Description	CreationDate	Medium	ImageURL
1	The Persistence of Memory	A surrealist painting by Salvador Dalí featuring ...	1931-01-01	Oil on canvas	https://example.com/persistence_of_memory.jpg
2	The Starry Night	A famous painting by Vincent van Gogh depictin...	2024-01-09	Oil on canvas	https://example.com/starry_night.jpg
3	Mona Lisa	A masterpiece by Leonardo da Vinci featuring a ...	1503-01-01	Oil on poplar panel	https://example.com/mona_lisa.jpg
4	The Creation of Adam	A fresco painting by Michelangelo depicting the ...	1512-10-01	Fresco	https://example.com/creation_of_adam.jpg
5	Girl with a Pearl Earring	A painting by Johannes Vermeer featuring a girl...	1665-01-01	Oil on canvas	https://example.com/girl_with_a_pearl_earring....
6	The Last Supper	A mural painting by Leonardo da Vinci depicting ...	1498-01-01	Fresco	https://example.com/last_supper.jpg

TABLE User_artwork_favorite:



UserId	ArtworkId
1	4
1	3
2	4
2	6
3	5
4	4
4	2
5	6
5	4
6	6

TABLE User:



UserID	UserName	Password	Email	FirstName	LastName	DateofBirth	ProfilePicture	FavouriteArtworks
1	Gayu	pass1234	gayu@gmail.com	Gayathri	Guna	2003-05-15	NULL	1,3,5
2	Swathi	qwerty	swa@example.com	Swathi	Smith	2001-10-20	NULL	2,4
3	Kousalya	securepassword	kousi@gmail.com	Kousalya	Velu	1995-02-28	NULL	6,5
4	jackson	hello123	bob@example.com	Bob	Jackson	1988-07-03	NULL	2,3,4
5	emily	password123	emily@example.com	Emily	Brown	1978-11-12	NULL	1,4
6	Monica	password12345	monica@gmail.com	Monica	Raaja	2003-09-08	NULL	4,6,2
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Coding

Create the model/entity classes corresponding to the schema **within package** entity **with variables declared private, constructors(default and parametrized) and getters,setters)**

Entity package:

Artist.py

```
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

    @property
    def artist_id(self):
        return self.__artist_id

    @artist_id.setter
    def artist_id(self, value):
        self.__artist_id = value

    @property
    def name(self):
        return self.__name
```

```python
    @name.setter
    def name(self, value):
        self.__name = value

    @property
    def biography(self):
        return self.__biography

    @biography.setter
    def biography(self, value):
        self.__biography = value

    @property
    def birth_date(self):
        return self.__birth_date

    @birth_date.setter
    def birth_date(self, value):
        self.__birth_date = value

    @property
    def nationality(self):
        return self.__nationality

    @nationality.setter
    def nationality(self, value):
        self.__nationality = value

    @property
    def website(self):
        return self.__website

    @website.setter
    def website(self, value):
        self.__website = value

    @property
    def contact_information(self):
        return self.__contact_information

    @contact_information.setter
    def contact_information(self, value):
        self.__contact_information = value
```

Artwork.py

```python
class Artwork:
    def __init__(self, ArtworkId=None, Title=None, Description=None,
CreationDate=None, Medium=None, ImageUrl=None):
        self.__ArtworkId = ArtworkId
        self.__Title = Title
        self.__Description = Description
        self.__CreationDate = CreationDate
        self.__Medium = Medium
        self.__ImageUrl = ImageUrl

    def __str__(self):
        return f"Artwork ID: {self.__ArtworkId}\nTitle:
{self.Title}\nDescription: {self.Description}\nCreation Date:
{self.CreationDate}\nMedium: {self.Medium}\nImage URL: {self.ImageUrl}\n"


    @property
    def ArtworkId(self):
        return self.__ArtworkId

    @ArtworkId.setter
    def ArtworkId(self, value):
        self.__ArtworkId = value

    @property
    def Title(self):
        return self.__Title

    @Title.setter
    def Title(self, value):
        self.__Title = value

    @property
    def Description(self):
        return self.__Description

    @Description.setter
    def Description(self, value):
        self.__Description = value

    @property
    def CreationDate(self):
        return self.__CreationDate

    @CreationDate.setter
    def CreationDate(self, value):
        self.__CreationDate = value

    @property
    def Medium(self):
        return self.__Medium

    @Medium.setter
    def Medium(self, value):
        self.__Medium = value

    @property
    def ImageUrl(self):
        return self.__ImageUrl

    @ImageUrl.setter
    def ImageUrl(self, value):
        self.__ImageUrl = value
```

Gallery.py

```
class Gallery:
    def __init__(self, GalleryId=None, Name=None, Description=None,
Location=None, Curator=None, OpeningHours=None):
        self.__GalleryId = GalleryId
        self.__Name = Name
        self.__Description = Description
        self.__Location = Location
        self.__Curator = Curator
        self.__OpeningHours = OpeningHours

    def __str__(self):
        return f"Galler ID: {self.__GalleryId}\nName:
{self.__Name}\nDescription: {self.Description}\nLocation:
{self.__Location}\nCurator: {self.__Curator}\nOpening Hours:
{self.__OpeningHours}\n"



    @property
    def GalleryId(self):
        return self.__GalleryId

    @GalleryId.setter
    def GalleryId(self, GalleryId):
        self.__GalleryId = GalleryId

    @property
    def Name(self):
        return self.__Name

    @Name.setter
    def Name(self, Name):
        self.__Name = Name

    @property
    def Description(self):
        return self.__Description

    @Description.setter
    def Description(self, Description):
        self.__Description = Description

    @property
    def Location(self):
        return self.__Location

    @Location.setter
    def Location(self, Location):
        self.__Location = Location

    @property
    def Curator(self):
        return self.__Curator

    @Curator.setter
    def Curator(self, Curator):
        self.__Curator = Curator

    @property
    def OpeningHours(self):
        return self.__OpeningHours

    @OpeningHours.setter
    def OpeningHours(self, OpeningHours):
        self.__OpeningHours = OpeningHours
```

User.py

```
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, favorite_artworks=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
        self.__favorite_artworks = favorite_artworks

    @property
    def user_id(self):
        return self.__user_id

    @user_id.setter
    def user_id(self, value):
        self.__user_id = value

    @property
    def username(self):
        return self.__username

    @username.setter
    def username(self, value):
        self.__username = value

    @property
    def password(self):
        return self.__password

    @password.setter
    def password(self, value):
        self.__password = value

    @property
    def email(self):
        return self.__email

    @email.setter
    def email(self, value):
        self.__email = value

    @property
    def first_name(self):
        return self.__first_name

    @first_name.setter
    def first_name(self, value):
        self.__first_name = value

    @property
    def last_name(self):
        return self.__last_name

    @last_name.setter
    def last_name(self, value):
        self.__last_name = value

    @property
    def date_of_birth(self):
        return self.__date_of_birth

    @date_of_birth.setter
    def date_of_birth(self, value):
        self.__date_of_birth = value

    @property
    def profile_picture(self):
        return self.__profile_picture

    @profile_picture.setter
    def profile_picture(self, value):
        self.__profile_picture = value

    @property
    def favorite_artworks(self):
        return self.__favorite_artworks

    @favorite_artworks.setter
    def favorite_artworks(self, value):
        self.__favorite_artworks = value
```

UserFavoriteArtwork.py

```
class UserFavoriteArtwork:
    def __init__(self, user_id=None, artwork_id=None):
        self.__user_id = user_id
        self.__artwork_id = artwork_id

    @property
    def user_id(self):
        return self.__user_id

    @user_id.setter
    def user_id(self, value):
        self.__user_id = value

    @property
    def artwork_id(self):
        return self.__artwork_id

    @artwork_id.setter
    def artwork_id(self, value):
        self.__artwork_id = value
```

Service Provider Interface/Abstract class

Keep the interfaces and implementation classes in package dao

Create **IVirtualArtGallery** Interface/abstract class with the following methods

// Artwork Management
addArtwork(); parameters-
Artwork object return type
Boolean **updateArtwork**();
parameters- Artwork object
return type Boolean

removeArtwork()
parameters-artworkID
return type Boolean
getArtworkById();
parameters-artworkID
return type Artwork
searchArtworks()
searchArtworks();
parameters- keyword
return type list of Artwork Object

// User Favorites
addArtworkToFavorite();

parameters- userId, artworkId
return type boolean

removeArtworkFromFavorite()
parameters- userId, artworkId
return type boolean

getUserFavoriteArtworks()
parameters- userId
return type boolean

dao package:

IVirtualArtGallery.py:

```
from abc import ABC, abstractmethod
from typing import List
from entity import Artwork


class IVirtualArtGallery(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[Artwork]:
        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[Artwork]:
        pass

    @abstractmethod
    def create_new_gallery(self,gallery):
        pass

    @abstractmethod
    def update_gallery(self,gallery):
        pass

    @abstractmethod
    def remove_gallery(self,gallery):
        pass

    @abstractmethod
    def search_gallery(self,keyword):
        pass
```

7: Connect your application to the SQL database:

1. Write code to establish a connection to your SQL database.

Create a utility class **DBConnection** in a package **util** with a static variable **connection** of Type **Connection** and a static method **getConnection()** which returns connection.

Connection properties supplied in the connection string should be read from a property file.

Create a utility class **PropertyUtil** which contains a static method named **getPropertyString()** which reads a property fie containing connection details like hostname, dbname, username, password, port number and returns a connection string.

util package:

DBConnection.py:

```
import mysql.connector
from util.PropertyUtil import PropertyUtil


class DBConnection:
    connection = None

    def getConnection(self):
        if DBConnection.connection is None:
            connection_string = PropertyUtil.getPropertyString()
            try:
                DBConnection.connection =
mysql.connector.connect(**connection_string)
                print("Connection successful")
            except mysql.connector.Error as error:
                print("Error while connecting to MySQL", error)
        return DBConnection.connection
```

PropertyUtil.py:

```

class PropertyUtil:
    @staticmethod
    def getPropertyString():
        connection_string = {
            'host': "localhost",
            'database': "virtualartgallery",
            'user': "root",
            'password': "root",
            'port': "3306"
        }
        return connection_string
```

8: Service implementation

1. Create a Service class **VirtualArtGalleryImpl**
2. Provide implementation for all the methods in the interface.

dao package:

VirtualArtGalleryImpl.py:

```
from datetime import date, datetime, timedelta, time

import mysql.connector
from typing import List, Tuple, Set, Dict

from _decimal import Decimal

from entity.Artwork import Artwork
from dao.IVirtualArtGallery import IVirtualArtGallery
from util.DBConnection import DBConnection
from myexceptions.ArtWorkNotFoundException import ArtworkNotFoundException
from myexceptions.UserNotFoundException import UserNotFoundException
from entity.User import User
from entity.Gallery import Gallery


class VirtualArtGalleryImpl(IVirtualArtGallery):
    def __init__(self):
        print("here con")
        self.connection = DBConnection.getConnection(self)
        # self.connection = mysql.connector.connect(
        #     host='localhost',
        #     user='root',
        #     password='root',
        #     port="3306",
        #     database='virtualartgallery'
        # )
        self.mycursor = self.connection.cursor()

    def add_artwork(self, artwork: Artwork) -> bool:
        query = (
            "INSERT INTO artwork (ArtworkID,Title, Description, CreationDate,
Medium, ImageURL) VALUES (%s, %s, %s, %s, %s, %s)")
        values = (
            artwork.ArtworkId, artwork.Title, artwork.Description,
artwork.CreationDate, artwork.Medium,
            artwork.ImageUrl)
        self.mycursor.execute(query, values)
        self.connection.commit()
        return True

    def update_artwork(self, artwork: Artwork) -> bool:
        query = ("UPDATE artwork SET ArtworkId = %s,Title = %s, Description =
%s, CreationDate = %s, Medium = %s, "
                 "ImageUrl = %s WHERE ArtworkId = %s")
        values = (
            artwork.ArtworkId, artwork.Title, artwork.Description,
artwork.CreationDate, artwork.Medium,
            artwork.ImageUrl, artwork.ArtworkId
        )
        self.mycursor.execute(query, values)
        self.connection.commit()
        return True

    def remove_artwork(self, artwork_id: int) -> bool:
        query = "DELETE FROM artwork WHERE ArtworkId = %s"
        self.mycursor.execute(query, (artwork_id,))
        self.connection.commit()
        return True

    def get_artwork_by_id(self, artwork_id: int) -> Artwork:
        query = "SELECT * FROM artwork WHERE ArtworkID = %s"
        self.mycursor.execute(query, (artwork_id,))
        result = self.mycursor.fetchone()
        if result:
            artwork = Artwork()
            artwork.ArtworkId = result[0]
            artwork.Title = result[1]
            artwork.Description = result[2]
            artwork.CreationDate = result[3]
            artwork.Medium = result[4]
            artwork.ImageUrl = result[5]
            return artwork
        else:
            raise ArtworkNotFoundException(artwork_id)
    def search_artworks(self, keyword: str) -> List[Artwork]:
        query = "SELECT * FROM artwork WHERE Title LIKE %s OR Description
LIKE %s"
        self.mycursor.execute(query, ("%" + keyword + "%", "%" + keyword +
"%"))
        results = self.mycursor.fetchall()
        print(results)
        artworks = []
        for result in results:
            artwork = Artwork()
            artwork.ArtworkId = result[0]
            artwork.Title = result[1]
            artwork.Description = result[2]
            artwork.CreationDate = result[3]
            artwork.Medium = result[4]
            artwork.ImageUrl = result[5]
            artworks.append(artwork)
        return artworks

    def add_artwork_to_favorite(self, user_id: int, artwork_id: int) -> bool:
        query = "INSERT INTO user_favorite_artwork (UserId, ArtworkId) VALUES
(%s, %s)"
        values = (user_id, artwork_id)
        self.mycursor.execute(query, values)
        self.connection.commit()
        return True

    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"
        values = (user_id, artwork_id)
        self.mycursor.execute(query, values)
        self.connection.commit()
        return True

    def get_user_favorite_artworks(self, user_id: int) -> List[Artwork]:
        try:
            self.get_user_by_id(user_id)
        except UserNotFoundException as e:
            print(e)
            return []
        query = ("SELECT * FROM artwork WHERE ArtworkId IN (SELECT ArtworkId
FROM user_favorite_artwork WHERE UserId = "
                 "%s)")
        self.mycursor.execute(query, (user_id,))
        results = self.mycursor.fetchall()
        artworks = []
        for result in results:
            artwork = Artwork()
            artwork.ArtworkId = result[0]
            artwork.Title = result[1]
            artwork.Description = result[2]
            artwork.CreationDate = result[3]
            artwork.Medium = result[4]
            artwork.ImageUrl = result[5]
            artworks.append(artwork)
        return artworks

    def get_user_by_id(self, id) -> User:
        query = "select * from user where UserID= %s"
        values = (id,)
        self.mycursor.execute(query, values)
        result = self.mycursor.fetchone()
        if result:
            return result
        else:
            raise UserNotFoundException(id)

    def create_new_gallery(self, gallery):
        query = "Insert into
Gallery(GalleryId,Name,Description,Location,Curator,OpeningHours) values
(%s,%s,%s,%s,%s,%s)"
        values = (
        gallery.gallery_id, gallery.name, gallery.description,
gallery.location, gallery.curator, gallery.opening_hours)
        self.mycursor.execute(query, values)
        self.connection.commit()
        return True

    def get_gallery_by_id(self, gallery_id)-> Gallery | None:
        query = "select * from gallery where GalleryId= %s"
        values = (gallery_id,)
        self.mycursor.execute(query, values)
        result = self.mycursor.fetchone()
        if result:

gallery=Gallery(result[0],result[1],result[2],result[3],result[4],result[5])
            return gallery
        else:
            return None

    def update_gallery(self, gallery):
        query = ("UPDATE gallery SET Name = %s, Description = %s, Location =
%s, Curator = %s, Openinghours=%s WHERE GalleryId = %s")
        values = (
               gallery.Name, gallery.Description, gallery.Location,
gallery.Curator,
               gallery.OpeningHours,gallery.GalleryId)
        self.mycursor.execute(query,  values)
        self.connection.commit()
        return True

    def remove_gallery(self, gallery_id: int) -> bool:
        query = "DELETE FROM gallery WHERE GalleryId = %s"
        self.mycursor.execute(query, (gallery_id,))
        self.connection.commit()
        return True

    def search_gallery(self,keyword):
        query = "SELECT * FROM gallery WHERE Name LIKE %s OR Description LIKE
```

```
%s OR Location LIKE %s"
        self.mycursor.execute(query, ("%" + keyword + "%", "%" + keyword +
"%","%" + keyword + "%"))
        results = self.mycursor.fetchall()
        print(results)
        galleries = []
        for result in results:
            gallery =
Gallery(result[0],result[1],result[2],result[3],result[4],result[5])
            galleries.append(gallery)
        return galleries
```

9: Exception Handling

Create the exceptions in package **myexceptions**

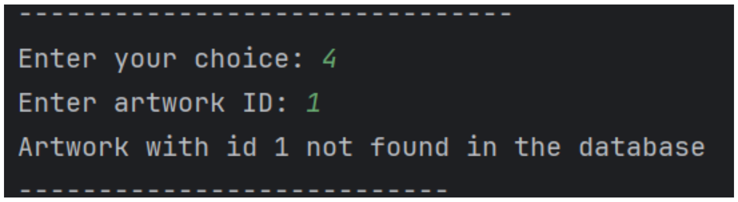
Define the following custom exceptions and throw them in methods whenever needed. Handle all the exceptions in main method,

1. **ArtWorkNotFoundException** :throw this exception when user enters an invalid id which doesn't exist in db
2. **UserNotFoundException** :throw this exception when user enters an invalid id which doesn't exist in db

Myexceptions package:

ArtWorkNotFoundException.py:

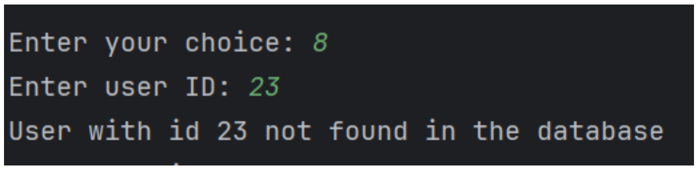
```
class ArtworkNotFoundException(Exception):
    def __init__(self, artwork_id):
        self.artwork_id = artwork_id
        super().__init__(f"Artwork with id {artwork_id} not found in the
database")
```



```
-------------------------------
Enter your choice: 4
Enter artwork ID: 1
Artwork with id 1 not found in the database
---------------------------
```

UserNotFoundException.py:

```
class UserNotFoundException(Exception):
    def __init__(self, user_id):
        self.user_id = user_id
        super().__init__(f"User with id {user_id} not found in the database")
```


```
Enter your choice: 8
Enter user ID: 23
User with id 23 not found in the database
```

9. **Main Method**

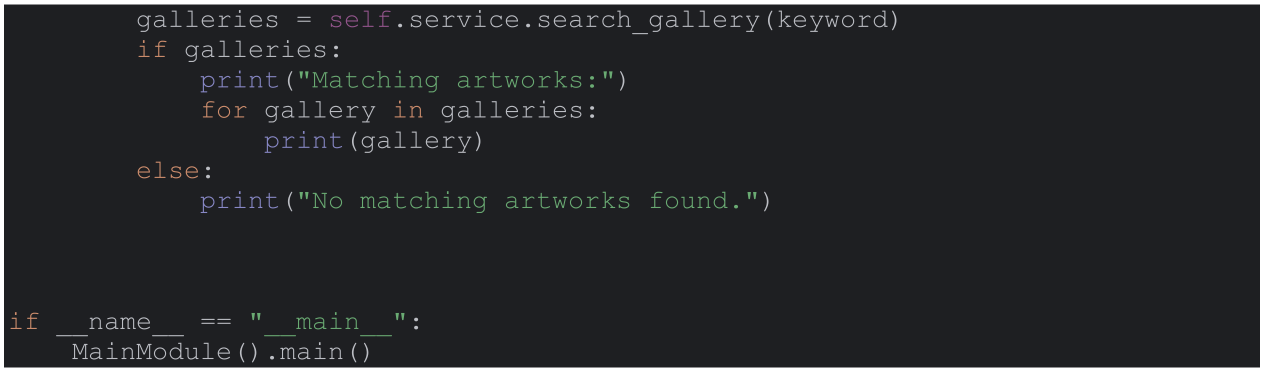
Create class named MainModule with main method in main package. Trigger all the methods in service implementation class.

main package:

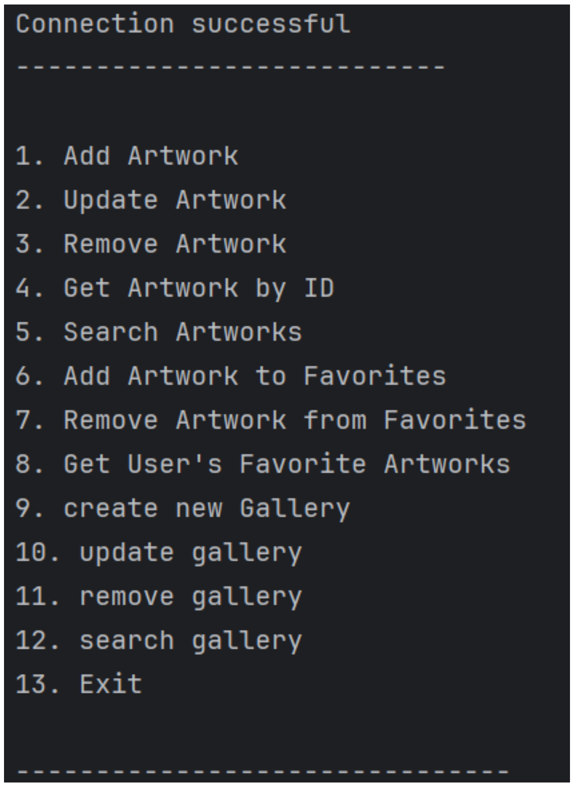
MainModule.py:

```
from dao.VirtualArtGalleryImpl import VirtualArtGalleryImpl
from myexceptions.ArtWorkNotFoundException import ArtworkNotFoundException
from myexceptions.UserNotFoundException import UserNotFoundException
from entity.Artwork import Artwork
from entity.Gallery import Gallery


class MainModule:
    def __init__(self):
        self.service = VirtualArtGalleryImpl()

    def main(self):
        while True:
            print("----------------------------")
            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 Artwork to Favorites")
            print("7. Remove Artwork from Favorites")
            print("8. Get User's Favorite Artworks")
            print("9. create new Gallery")
            print("10. update gallery ")
            print("11. remove gallery")
            print("12. search gallery")
            print("13. Exit")
            print()
            print("--------------------------------")

            choice = input("Enter your choice: ")

            if choice == "1":
                self.add_artwork()
            elif choice == "2":
                self.update_artwork()
            elif choice == "3":
                self.remove_artwork()
            elif choice == "4":
                self.get_artwork_by_id()
            elif choice == "5":
                self.search_artworks()
            elif choice == "6":
                self.add_artwork_to_favorites()
            elif choice == "7":
                self.remove_artwork_from_favorites()
            elif choice == "8":
                self.get_user_favorite_artworks()
            elif choice == "9":
                self.create_new_gallery()
            elif choice == "10":
                self.update_gallery()
            elif choice == "11":
                self.remove_gallery()
            elif choice == "12":
                self.search_gallery()
                break
            else:
                print("Invalid choice. Please enter a valid option.")

    def add_artwork(self):
        artworkid = int(input("Enter artwork id: "))
        title = input("Enter artwork title: ")
        description = input("Enter artwork description: ")
        creation_date = input("Enter artwork creation date: ")
        medium = input("Enter artwork medium: ")
        image_url = input("Enter artwork image URL: ")
        artwork = Artwork(ArtworkId=artworkid, Title=title,
Description=description, CreationDate=creation_date,
                          Medium=medium,
                          ImageUrl=image_url)
        print(artwork)
        self.service.add_artwork(artwork)
        print("Artwork added successfully.")

    def update_artwork(self):
        artwork_id = int(input("Enter artwork ID: "))
        artwork = self.service.get_artwork_by_id(artwork_id)
        if artwork:
            title = input("Enter new title (press Enter to keep current
title): ")
            description = input("Enter new description (press Enter to keep
current description): ")
            creation_date = input("Enter new creation date (press Enter to
keep current creation date): ")
            medium = input("Enter new medium (press Enter to keep current
medium): ")
            image_url = input("Enter new image URL (press Enter to keep
current image URL): ")
            if title:
                artwork.Title = title
            if description:
                artwork.Description = description
            if creation_date:
                artwork.CreationDate = creation_date
```

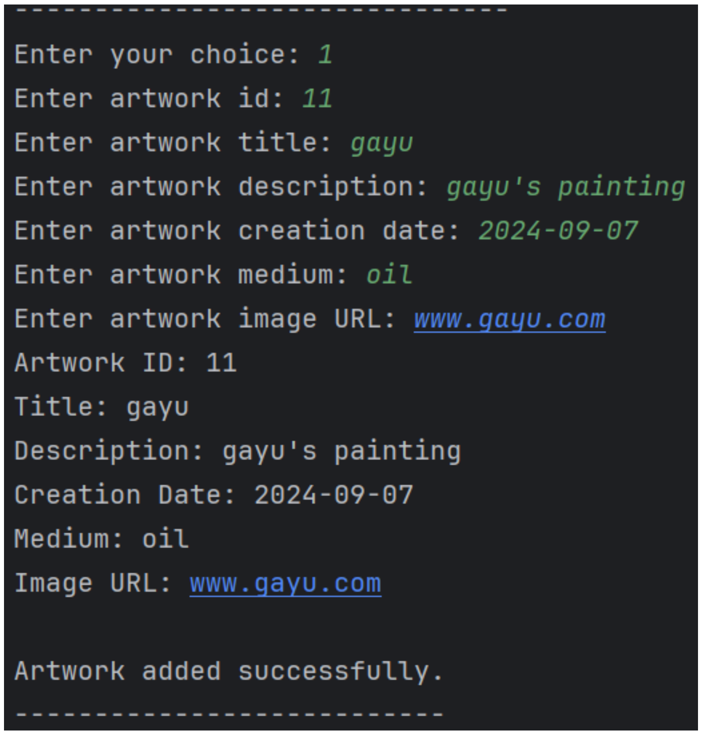
```python
            if medium:
                artwork.Medium = medium
            if image_url:
                artwork.ImageUrl = image_url
            self.service.update_artwork(artwork)
            print("Artwork updated successfully.")
        else:
            print("Artwork not found.")

    def remove_artwork(self):
        artwork_id = int(input("Enter artwork ID: "))
        try:
            self.service.remove_artwork(artwork_id)
            print("Artwork removed successfully.")
        except ArtworkNotFoundException:
            print("Artwork not found.")

    def get_artwork_by_id(self):
        artwork_id = int(input("Enter artwork ID: "))
        try:
            artwork = self.service.get_artwork_by_id(artwork_id)
            print(artwork)
            if artwork:
                print("Artwork details:")
                print(artwork)
            else:
                print("Artwork not found.")
        except ArtworkNotFoundException as e:
            print(e)

    def search_artworks(self):
        keyword = input("Enter keyword to search: ")
        artworks = self.service.search_artworks(keyword)
        if artworks:
            print("Matching artworks:")
            for artwork in artworks:
                print(artwork)
        else:
            print("No matching artworks found.")

    def add_artwork_to_favorites(self):
        user_id = int(input("Enter user ID: "))
        artwork_id = int(input("Enter artwork ID: "))
        self.service.add_artwork_to_favorite(user_id, artwork_id)
        print("Artwork added to favorites.")

    def remove_artwork_from_favorites(self):
        user_id = int(input("Enter user ID: "))
        artwork_id = int(input("Enter artwork ID: "))
        self.service.remove_artwork_from_favorite(user_id, artwork_id)
        print("Artwork removed from favorites.")

    def get_user_favorite_artworks(self):
        user_id = int(input("Enter user ID: "))
        artwork = self.service.get_user_favorite_artworks(user_id)
        if artwork:
            print("User Favorite details:")
            for i in artwork:
                print(i)
        else:
            print("User Favorite not found.")

    def create_new_gallery(self):
        galleryid = int(input("Enter gallery id: "))
        name = input("Enter the gallery name: ")
        description = input("Enter description: ")
        location = input("Enter location: ")
        curator = int(input("Enter curator: "))
        openinghours = input("Enter Opening hours: ")
        gallery = Gallery(GalleryId=galleryid, Name=name,
Description=description, Location=location, Curator=curator,
                          OpeningHours=openinghours)
        self.service.create_new_gallery(gallery)
        print("Gallery created successfully")

    def update_gallery(self):
        gallery_id = int(input("Enter galleryid: "))
        gallery = self.service.get_gallery_by_id(gallery_id)
        print(gallery)
        if gallery:
            name = input("Enter new name (press Enter to keep current name):
")
            description = input("Enter new description (press Enter to keep
current description): ")
            location = input("Enter new location  (press Enter to keep
location): ")
            curator = input("Enter new curator (press Enter to keep current
curator): ")
            openinghours = input("Enter new openinghours  (press Enter to
keep current openinghours): ")
            if name:
                gallery.Name = name
            if description:
                gallery.Description = description
            if location:
                gallery.Location = location
            if curator:
                gallery.Curator = curator
            if openinghours:
                gallery.OpeningHours = openinghours
            self.service.update_gallery(gallery)
            print("Gallery updated successfully.")
        else:
            print("Gallery not found.")

    def remove_gallery(self):
        gallery_id = int(input("Enter gallery ID: "))
        self.service.remove_gallery(gallery_id)
        print("Gallery removed successfully.")

    def search_gallery(self):
        keyword = input("Enter keyword to search: ")
```



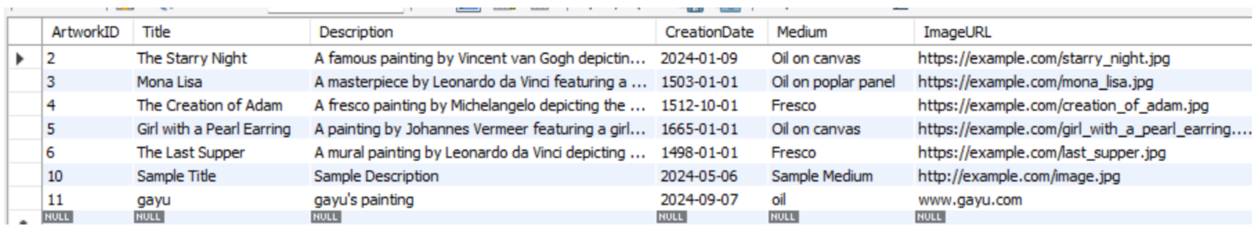
```python
        galleries = self.service.search_gallery(keyword)
        if galleries:
            print("Matching artworks:")
            for gallery in galleries:
                print(gallery)
        else:
            print("No matching artworks found.")


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



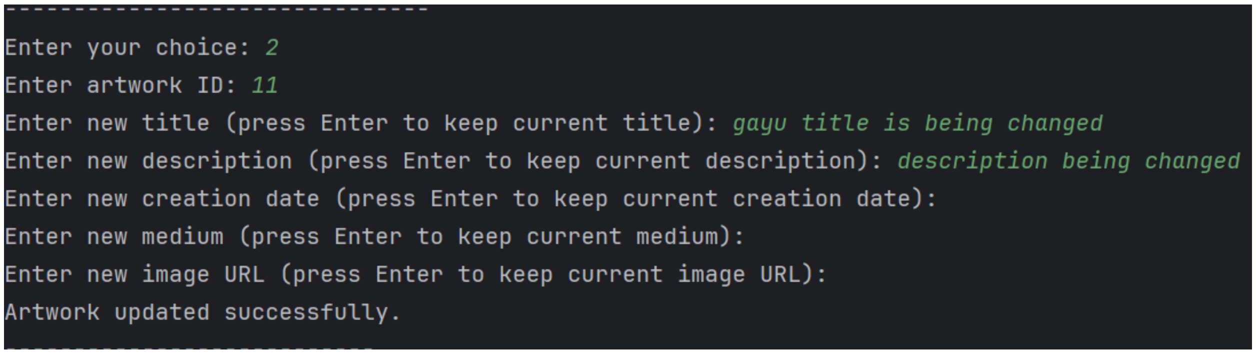
```
Connection successful
---------------------------


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's Favorite Artworks
9. create new Gallery
10. update gallery
11. remove gallery
12. search gallery
13. Exit


-------------------------------
```



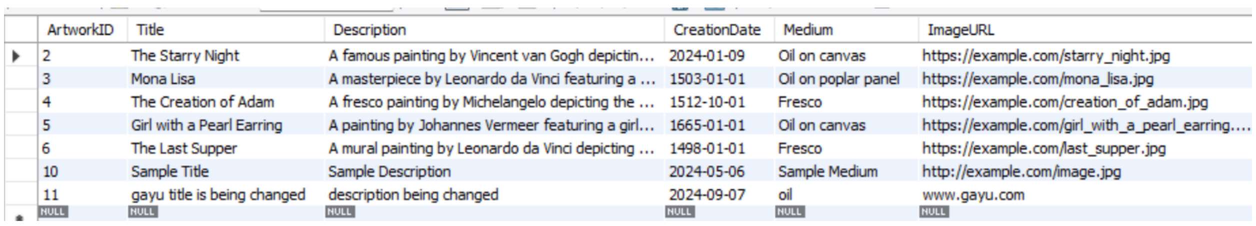
```
-------------------------------
Enter your choice: 1
Enter artwork id: 11
Enter artwork title: gayu
Enter artwork description: gayu's painting
Enter artwork creation date: 2024-09-07
Enter artwork medium: oil
Enter artwork image URL: www.gayu.com
Artwork ID: 11
Title: gayu
Description: gayu's painting
Creation Date: 2024-09-07
Medium: oil
Image URL: www.gayu.com

Artwork added successfully.
---------------------------
```



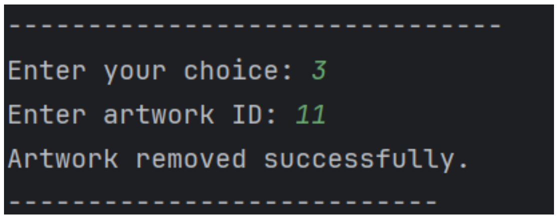
ArtworkID	Title	Description	CreationDate	Medium	ImageURL
2	The Starry Night	A famous painting by Vincent van Gogh depictin...	2024-01-09	Oil on canvas	https://example.com/starry_night.jpg
3	Mona Lisa	A masterpiece by Leonardo da Vinci featuring a ...	1503-01-01	Oil on poplar panel	https://example.com/mona_lisa.jpg
4	The Creation of Adam	A fresco painting by Michelangelo depicting the ...	1512-10-01	Fresco	https://example.com/creation_of_adam.jpg
5	Girl with a Pearl Earring	A painting by Johannes Vermeer featuring a girl...	1665-01-01	Oil on canvas	https://example.com/girl_with_a_pearl_earring....
6	The Last Supper	A mural painting by Leonardo da Vinci depicting ...	1498-01-01	Fresco	https://example.com/last_supper.jpg
10	Sample Title	Sample Description	2024-05-06	Sample Medium	http://example.com/image.jpg
11	gayu	gayu's painting	2024-09-07	oil	www.gayu.com
NULL	NULL	NULL	NULL	NULL	NULL



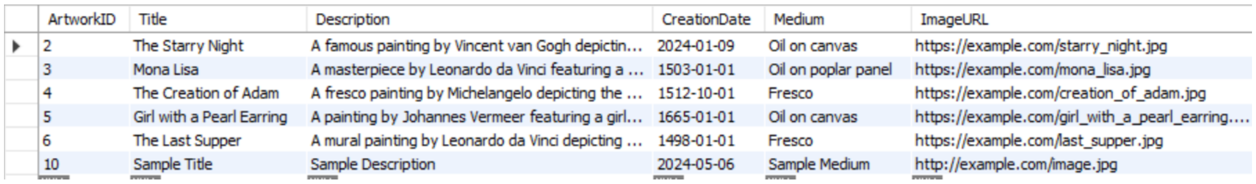
```
-------------------------------
Enter your choice: 2
Enter artwork ID: 11
Enter new title (press Enter to keep current title): gayu title is being changed
Enter new description (press Enter to keep current description): description being changed
Enter new creation date (press Enter to keep current creation date):
Enter new medium (press Enter to keep current medium):
Enter new image URL (press Enter to keep current image URL):
Artwork updated successfully.
---------------------------
```



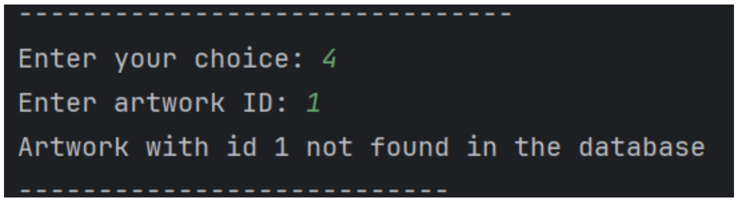
ArtworkID	Title	Description	CreationDate	Medium	ImageURL
2	The Starry Night	A famous painting by Vincent van Gogh depictin...	2024-01-09	Oil on canvas	https://example.com/starry_night.jpg
3	Mona Lisa	A masterpiece by Leonardo da Vinci featuring a ...	1503-01-01	Oil on poplar panel	https://example.com/mona_lisa.jpg
4	The Creation of Adam	A fresco painting by Michelangelo depicting the ...	1512-10-01	Fresco	https://example.com/creation_of_adam.jpg
5	Girl with a Pearl Earring	A painting by Johannes Vermeer featuring a girl...	1665-01-01	Oil on canvas	https://example.com/girl_with_a_pearl_earring....
6	The Last Supper	A mural painting by Leonardo da Vinci depicting ...	1498-01-01	Fresco	https://example.com/last_supper.jpg
10	Sample Title	Sample Description	2024-05-06	Sample Medium	http://example.com/image.jpg
11	gayu title is being changed	description being changed	2024-09-07	oil	www.gayu.com
NULL	NULL	NULL	NULL	NULL	NULL



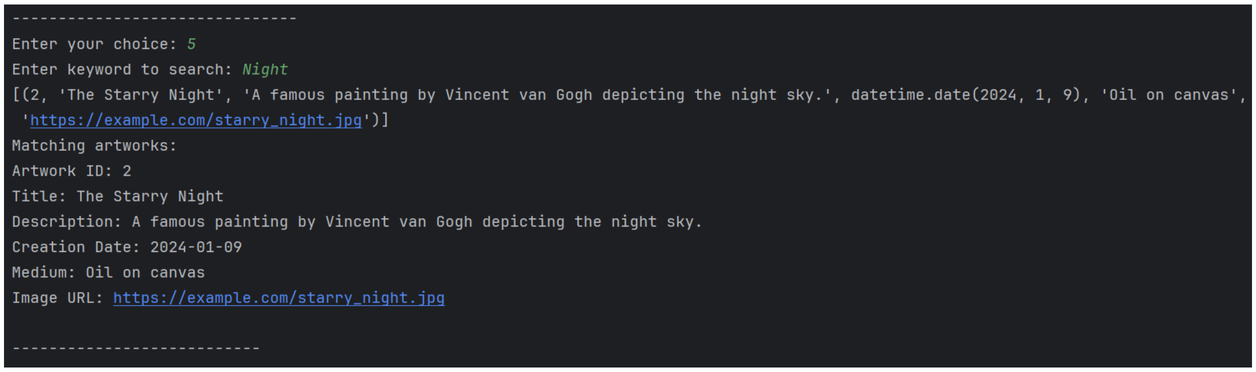
```
-------------------------------
Enter your choice: 3
Enter artwork ID: 11
Artwork removed successfully.
--------------------------
```



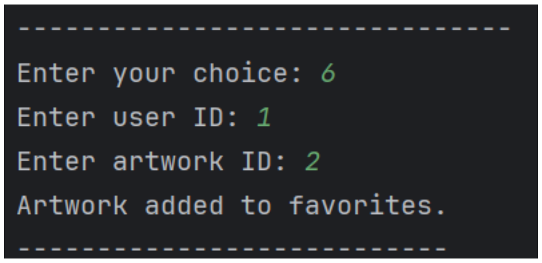
	ArtworkID	Title	Description	CreationDate	Medium	ImageURL
▶	2	The Starry Night	A famous painting by Vincent van Gogh depictin...	2024-01-09	Oil on canvas	https://example.com/starry_night.jpg
	3	Mona Lisa	A masterpiece by Leonardo da Vinci featuring a ...	1503-01-01	Oil on poplar panel	https://example.com/mona_lisa.jpg
	4	The Creation of Adam	A fresco painting by Michelangelo depicting the ...	1512-10-01	Fresco	https://example.com/creation_of_adam.jpg
	5	Girl with a Pearl Earring	A painting by Johannes Vermeer featuring a girl...	1665-01-01	Oil on canvas	https://example.com/girl_with_a_pearl_earring....
	6	The Last Supper	A mural painting by Leonardo da Vinci depicting ...	1498-01-01	Fresco	https://example.com/last_supper.jpg
	10	Sample Title	Sample Description	2024-05-06	Sample Medium	http://example.com/image.jpg



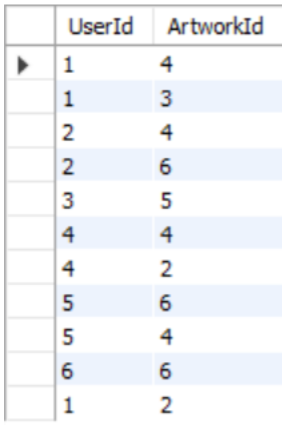
```
-------------------------------
Enter your choice: 4
Enter artwork ID: 1
Artwork with id 1 not found in the database
--------------------------
```



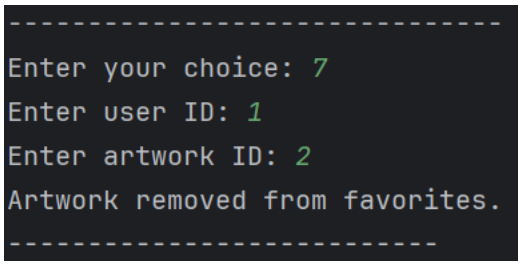
```
------------------------------
Enter your choice: 5
Enter keyword to search: Night
[(2, 'The Starry Night', 'A famous painting by Vincent van Gogh depicting the night sky.', datetime.date(2024, 1, 9), 'Oil on canvas',
 'https://example.com/starry_night.jpg')]
Matching artworks:
Artwork ID: 2
Title: The Starry Night
Description: A famous painting by Vincent van Gogh depicting the night sky.
Creation Date: 2024-01-09
Medium: Oil on canvas
Image URL: https://example.com/starry_night.jpg

--------------------------
```



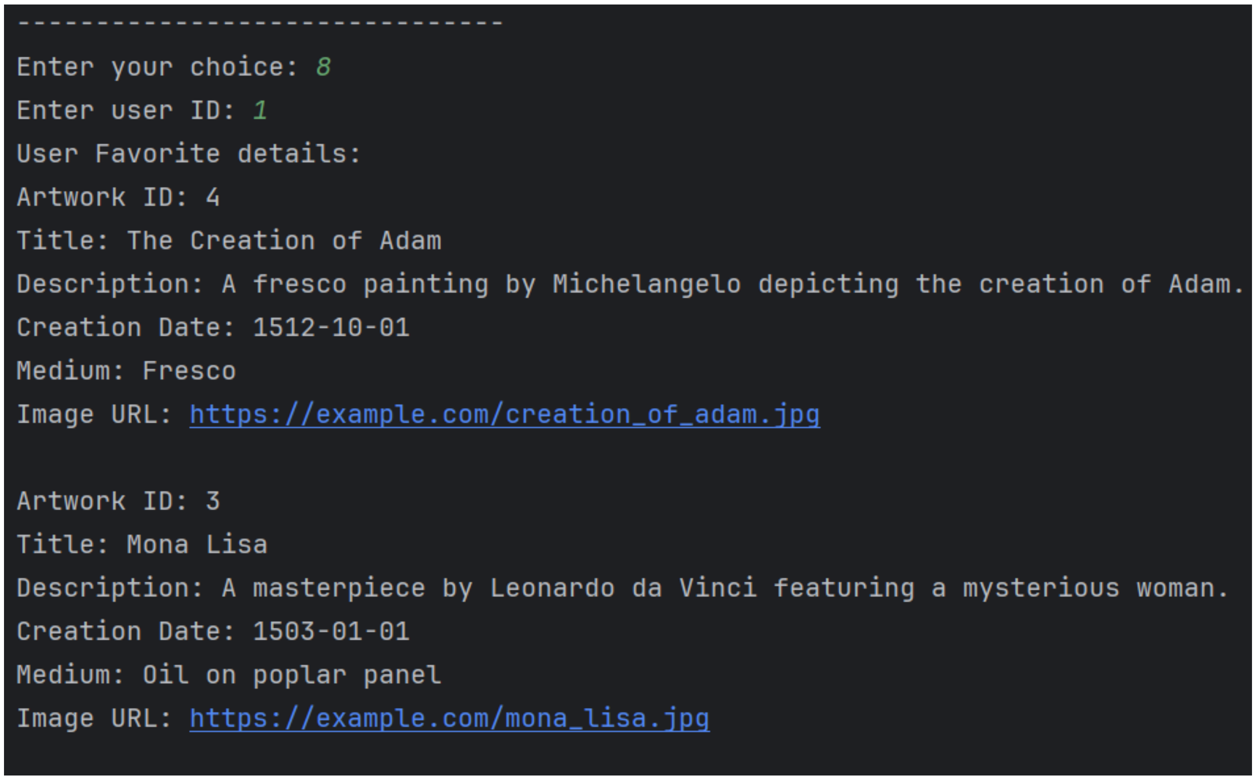
```
-------------------------------
Enter your choice: 6
Enter user ID: 1
Enter artwork ID: 2
Artwork added to favorites.
--------------------------
```



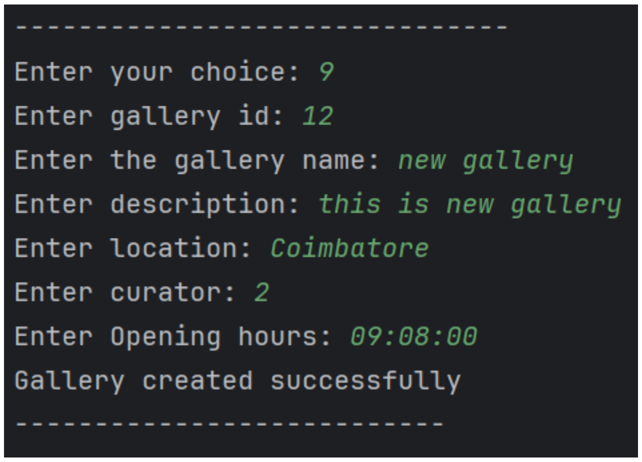
UserId	ArtworkId
1	4
1	3
2	4
2	6
3	5
4	4
4	2
5	6
5	4
6	6
1	2



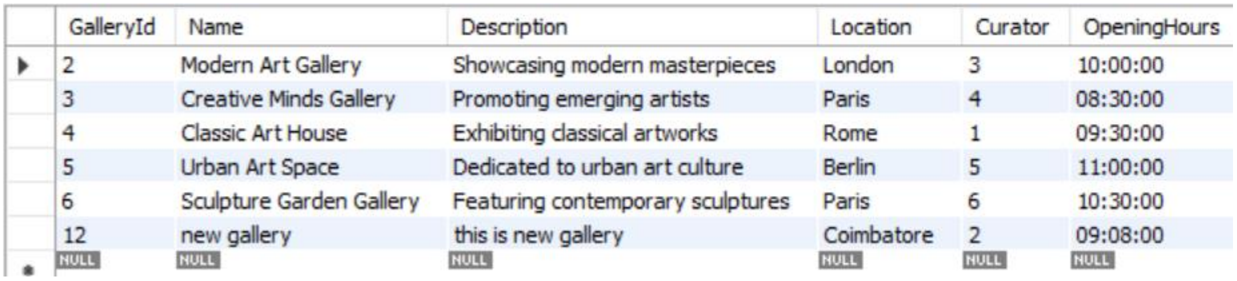
```
--------------------------------
Enter your choice: 7
Enter user ID: 1
Enter artwork ID: 2
Artwork removed from favorites.
----------------------------
```



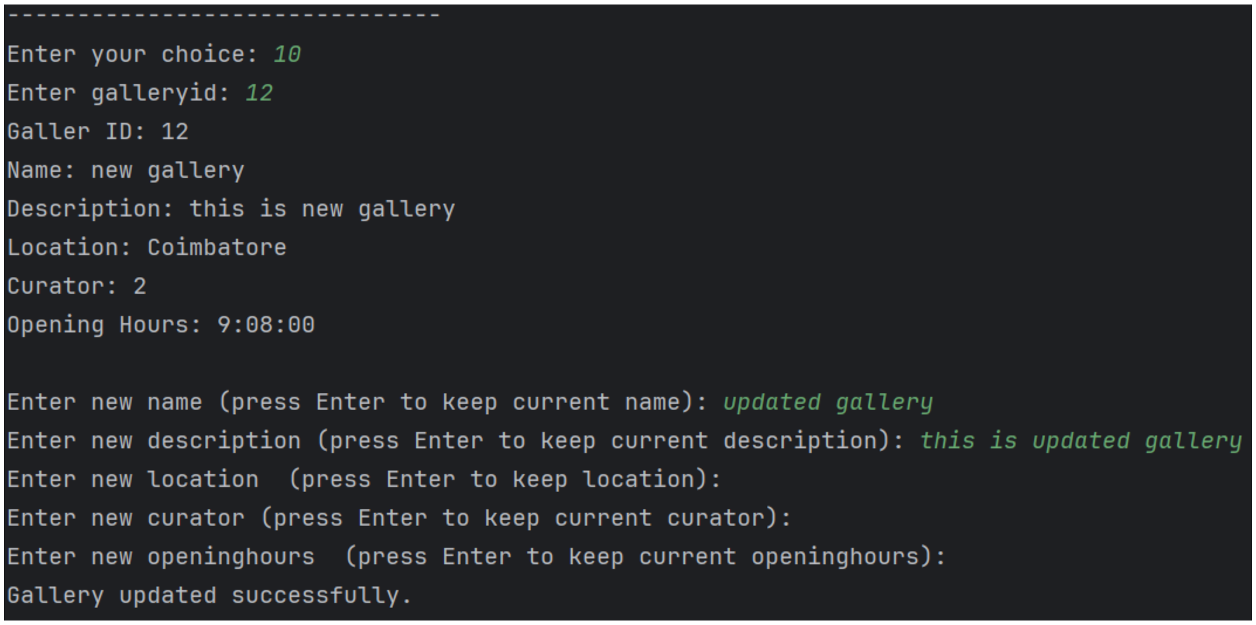
```
-------------------------------
Enter your choice: 8
Enter user ID: 1
User Favorite details:
Artwork ID: 4
Title: The Creation of Adam
Description: A fresco painting by Michelangelo depicting the creation of Adam.
Creation Date: 1512-10-01
Medium: Fresco
Image URL: https://example.com/creation_of_adam.jpg

Artwork ID: 3
Title: Mona Lisa
Description: A masterpiece by Leonardo da Vinci featuring a mysterious woman.
Creation Date: 1503-01-01
Medium: Oil on poplar panel
Image URL: https://example.com/mona_lisa.jpg
```



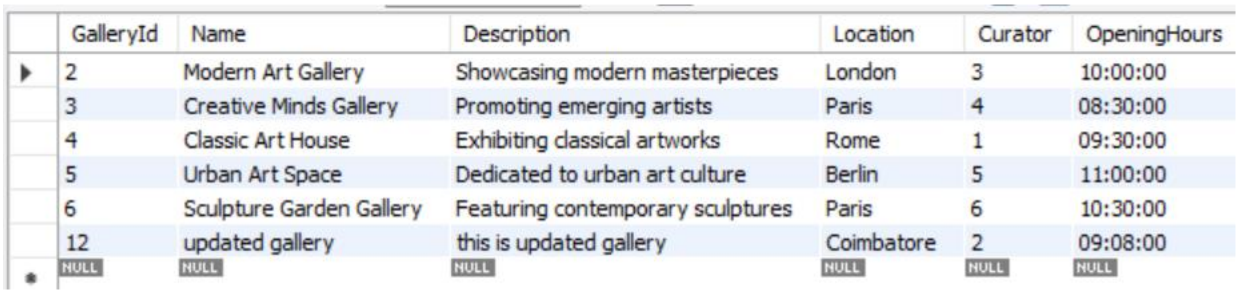
```
-------------------------------
Enter your choice: 9
Enter gallery id: 12
Enter the gallery name: new gallery
Enter description: this is new gallery
Enter location: Coimbatore
Enter curator: 2
Enter Opening hours: 09:08:00
Gallery created successfully
---------------------------
```



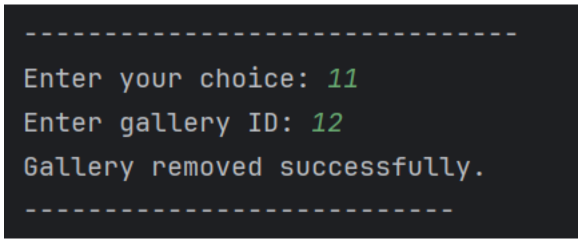
GalleryId	Name	Description	Location	Curator	OpeningHours
2	Modern Art Gallery	Showcasing modern masterpieces	London	3	10:00:00
3	Creative Minds Gallery	Promoting emerging artists	Paris	4	08:30:00
4	Classic Art House	Exhibiting classical artworks	Rome	1	09:30:00
5	Urban Art Space	Dedicated to urban art culture	Berlin	5	11:00:00
6	Sculpture Garden Gallery	Featuring contemporary sculptures	Paris	6	10:30:00
12	new gallery	this is new gallery	Coimbatore	2	09:08:00
NULL	NULL	NULL	NULL	NULL	NULL



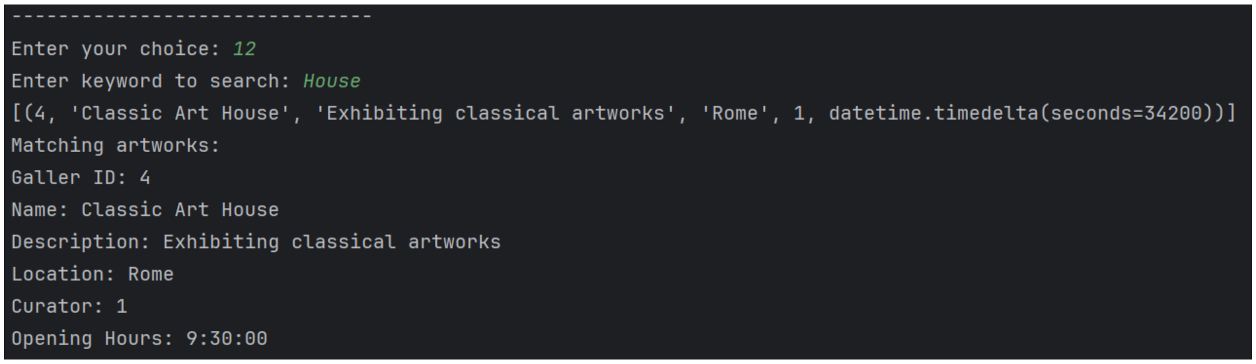
```
-------------------------------
Enter your choice: 10
Enter galleryid: 12
Galler ID: 12
Name: new gallery
Description: this is new gallery
Location: Coimbatore
Curator: 2
Opening Hours: 9:08:00

Enter new name (press Enter to keep current name): updated gallery
Enter new description (press Enter to keep current description): this is updated gallery
Enter new location  (press Enter to keep location):
Enter new curator (press Enter to keep current curator):
Enter new openinghours  (press Enter to keep current openinghours):
Gallery updated successfully.
```



	GalleryId	Name	Description	Location	Curator	OpeningHours
▶	2	Modern Art Gallery	Showcasing modern masterpieces	London	3	10:00:00
	3	Creative Minds Gallery	Promoting emerging artists	Paris	4	08:30:00
	4	Classic Art House	Exhibiting classical artworks	Rome	1	09:30:00
	5	Urban Art Space	Dedicated to urban art culture	Berlin	5	11:00:00
	6	Sculpture Garden Gallery	Featuring contemporary sculptures	Paris	6	10:30:00
	12	updated gallery	this is updated gallery	Coimbatore	2	09:08:00
*	NULL	NULL	NULL	NULL	NULL	NULL



```
------------------------------
Enter your choice: 11
Enter gallery ID: 12
Gallery removed successfully.
---------------------------
```



```
------------------------------
Enter your choice: 12
Enter keyword to search: House
[(4, 'Classic Art House', 'Exhibiting classical artworks', 'Rome', 1, datetime.timedelta(seconds=34200))]
Matching artworks:
Galler ID: 4
Name: Classic Art House
Description: Exhibiting classical artworks
Location: Rome
Curator: 1
Opening Hours: 9:30:00
```

10. Unit Testing

Creating Unit test cases for a Virtual Art Gallery system is essential to ensure that the system functions correctly. Below are sample test case questions that can serve as a starting point for your JUnit test suite:

1. **Artwork Management:**

a. Test the ability to upload a new artwork to the gallery.

b. Verify that updating artwork details works correctly.

c. Test removing an artwork from the gallery.

d. Check if searching for artworks returns the expected results.

2. **Gallery Management:**

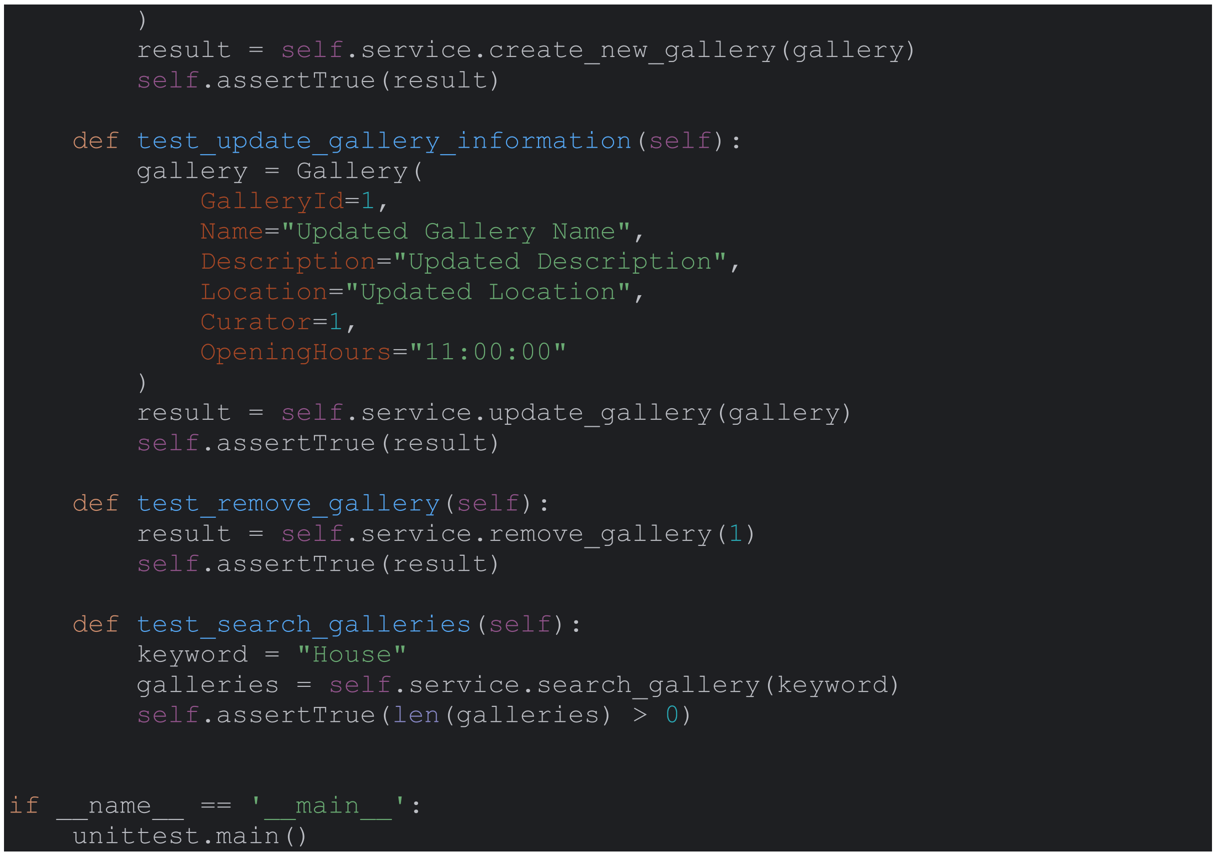
a. Test creating a new gallery.

b. Verify that updating gallery information works correctly.

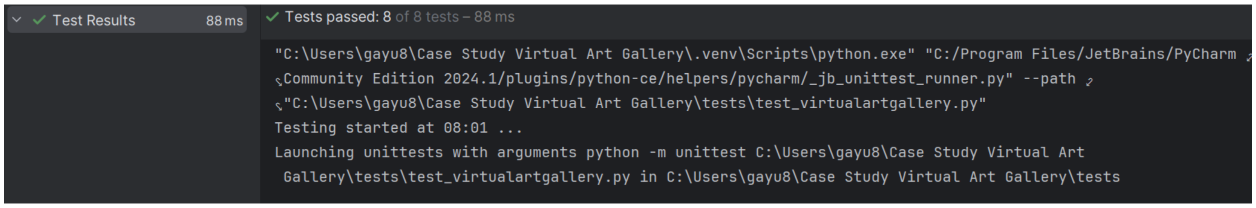
c. Test removing a gallery from the system.

d. Check if searching for galleries returns the expected results.

```python
import unittest
from dao.VirtualArtGalleryImpl import VirtualArtGalleryImpl
from entity.Gallery import Gallery
from entity.Artwork import Artwork


class TestVirtualArtGallery(unittest.TestCase):
    def setUp(self):
        self.service = VirtualArtGalleryImpl()

    def test_upload_new_artwork(self):
        artwork = Artwork(
            ArtworkId=90,
            Title="Sample Title",
            Description="Sample Description",
            CreationDate="2024-05-06",
            Medium="Sample Medium",
            ImageUrl="http://example.com/image.jpg"
        )
        result = self.service.add_artwork(artwork)
        self.assertTrue(result)

    def test_update_artwork_details(self):
        artwork = Artwork(
            ArtworkId=1,
            Title="Updated Title",
            Description="Updated Description",
            CreationDate="2024-05-06",
            Medium="Updated Medium",
            ImageUrl="http://example.com/updated_image.jpg"
        )
        result = self.service.update_artwork(artwork)
        self.assertTrue(result)

    def test_remove_artwork(self):
        result = self.service.remove_artwork(1)
        self.assertTrue(result)

    def test_search_artworks(self):
        keyword = "Sample"
        artworks = self.service.search_artworks(keyword)
        self.assertTrue(len(artworks) > 0)

    def test_create_new_gallery(self):
        gallery = Gallery(
            GalleryId=1,
            Name="Sample Gallery",
            Description="Sample Description",
            Location="Sample Location",
            Curator=1,
            OpeningHours="10:00:00"
```



```
        )
        result = self.service.create_new_gallery(gallery)
        self.assertTrue(result)

    def test_update_gallery_information(self):
        gallery = Gallery(
            GalleryId=1,
            Name="Updated Gallery Name",
            Description="Updated Description",
            Location="Updated Location",
            Curator=1,
            OpeningHours="11:00:00"
        )
        result = self.service.update_gallery(gallery)
        self.assertTrue(result)

    def test_remove_gallery(self):
        result = self.service.remove_gallery(1)
        self.assertTrue(result)

    def test_search_galleries(self):
        keyword = "House"
        galleries = self.service.search_gallery(keyword)
        self.assertTrue(len(galleries) > 0)


if __name__ == '__main__':
    unittest.main()
```



```
✓ Test Results  88 ms    ✓ Tests passed: 8 of 8 tests – 88 ms

"C:\Users\gayu8\Case Study Virtual Art Gallery\.venv\Scripts\python.exe" "C:/Program Files/JetBrains/PyCharm
Community Edition 2024.1/plugins/python-ce/helpers/pycharm/_jb_unittest_runner.py" --path
"C:\Users\gayu8\Case Study Virtual Art Gallery\tests\test_virtualartgallery.py"
Testing started at 08:01 ...
Launching unittests with arguments python -m unittest C:\Users\gayu8\Case Study Virtual Art
 Gallery\tests\test_virtualartgallery.py in C:\Users\gayu8\Case Study Virtual Art Gallery\tests
```