Case Study: Virtual Art Gallery

-Elakkiya M

Abstract

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

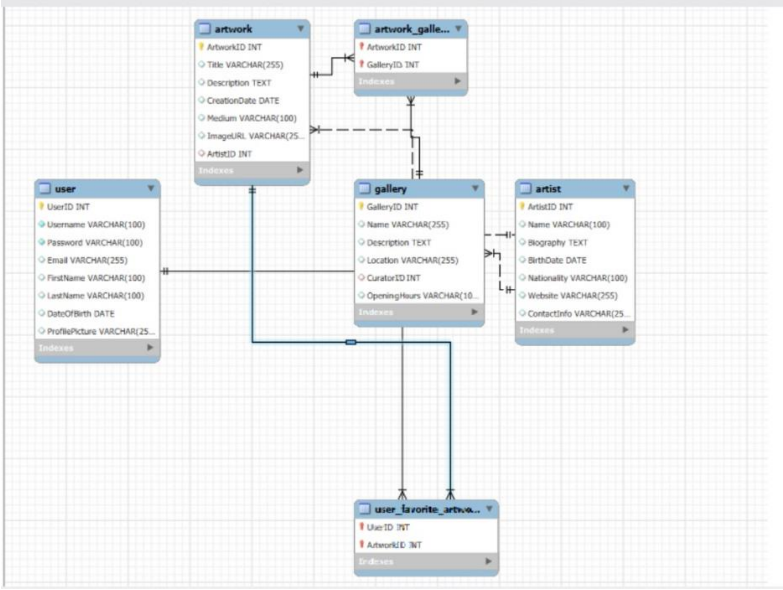
The project models key entities like Artwork, User, Artist, and Gallery using Python classes, which organize the data and actions related to each entity. The main functionalities include adding, viewing, and managing artwork details through console inputs, with all data stored securely in a MySQL database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

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

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

Schema design:

- ER diagram:



- Created a dedicated database: VIRTUALARTGALLERY.

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| courier_management |
| elakkiya           |
| hospital_management |
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| virtualartgallery  |
| world              |
+--------------------+
10 rows in set (0.00 sec)
```

- All required tables were created with proper data types, keys, and relationships:

```
mysql> USE VIRTUALARTGALLERY;
Database changed
mysql> SHOW TABLES;
+-----------------------------+
| Tables_in_virtualartgallery |
+-----------------------------+
| artist                      |
| artwork                     |
| artwork_gallery             |
| gallery                     |
| user                        |
| user_favorite_artwork       |
+-----------------------------+
6 rows in set (0.00 sec)
```

Table Name	Description
ARTIST	Stores artist details
ARTWORK	Stores artwork information with reference to the artist
USER	Stores user profile details
GALLERY	Stores gallery data, each linked to a curator (artist)
USER_FAVORITE_ARTWORK	**Junction table** for many-to-many relationship between users and artworks
ARTWORK_GALLERY	**Junction table** for many-to-many relationship between artworks and galleries

All tables include primary keys, and necessary foreign key constraints

- Relationship Mapping (via Foreign Keys):

  One-to-Many: ARTWORK → ARTIST, GALLERY → ARTIST

  Many-to-Many: Handled via:

  USER_FAVORITE_ARTWORK (USERID, ARTWORKID)

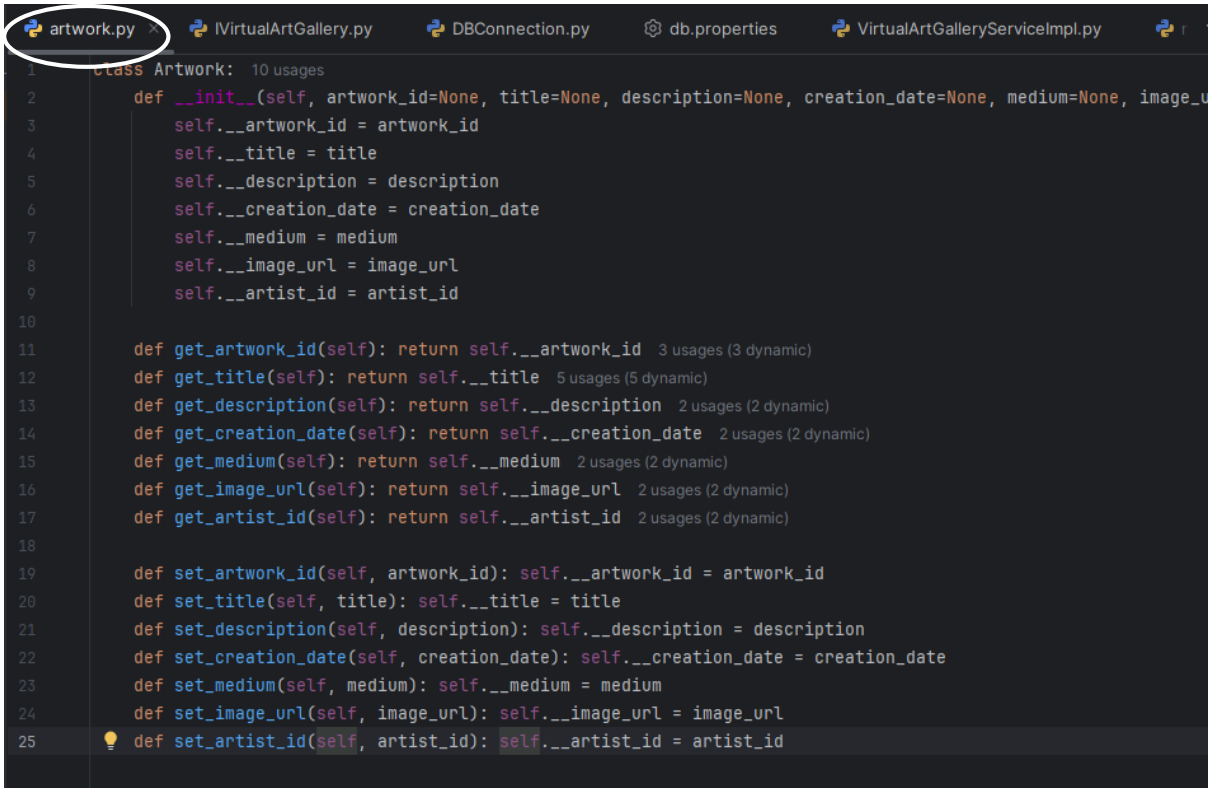
  ARTWORK_GALLERY (ARTWORKID, GALLERYID)

- Sample data inserted.

Coding

1. **Entity Classes**

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



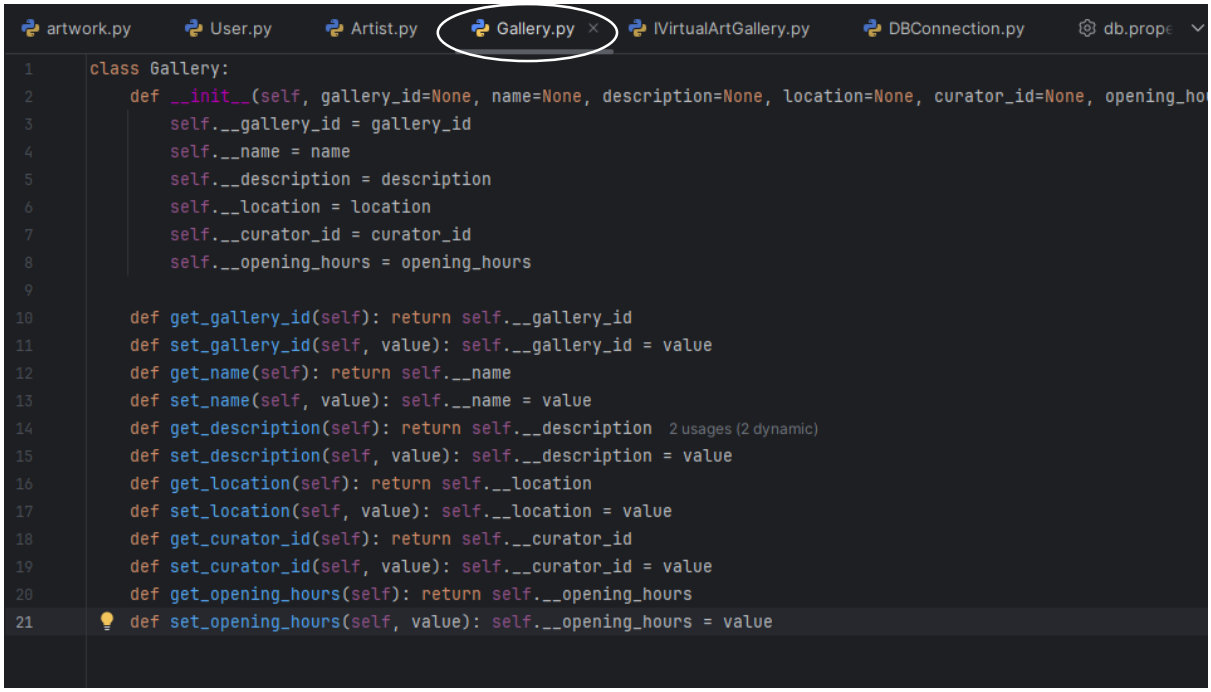
Even though we use only artwork file for artwork management we also created other files in entity as per schema which can be used for future update

artwork.py | User.py | Artist.py | Gallery.py | IVirtualArtGallery.py | DBConnection.py | db.prope

```python
class User:
    def __init__(self, user_id=None, username=None, password=None, email=None, first_name=None, last_name=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.__dob = dob
        self.__profile_picture = profile_picture

    def get_user_id(self): return self.__user_id
    def set_user_id(self, value): self.__user_id = value
    def get_username(self): return self.__username
    def set_username(self, value): self.__username = value
    def get_password(self): return self.__password
    def set_password(self, value): self.__password = value
    def get_email(self): return self.__email
    def set_email(self, value): self.__email = value
    def get_first_name(self): return self.__first_name
    def set_first_name(self, value): self.__first_name = value
    def get_last_name(self): return self.__last_name
    def set_last_name(self, value): self.__last_name = value
    def get_dob(self): return self.__dob
    def set_dob(self, value): self.__dob = value
    def get_profile_picture(self): return self.__profile_picture
    def set_profile_picture(self, value): self.__profile_picture = value
```

artwork.py | User.py | Artist.py | Gallery.py | IVirtualArtGallery.py | DBConnection.py | db.prop

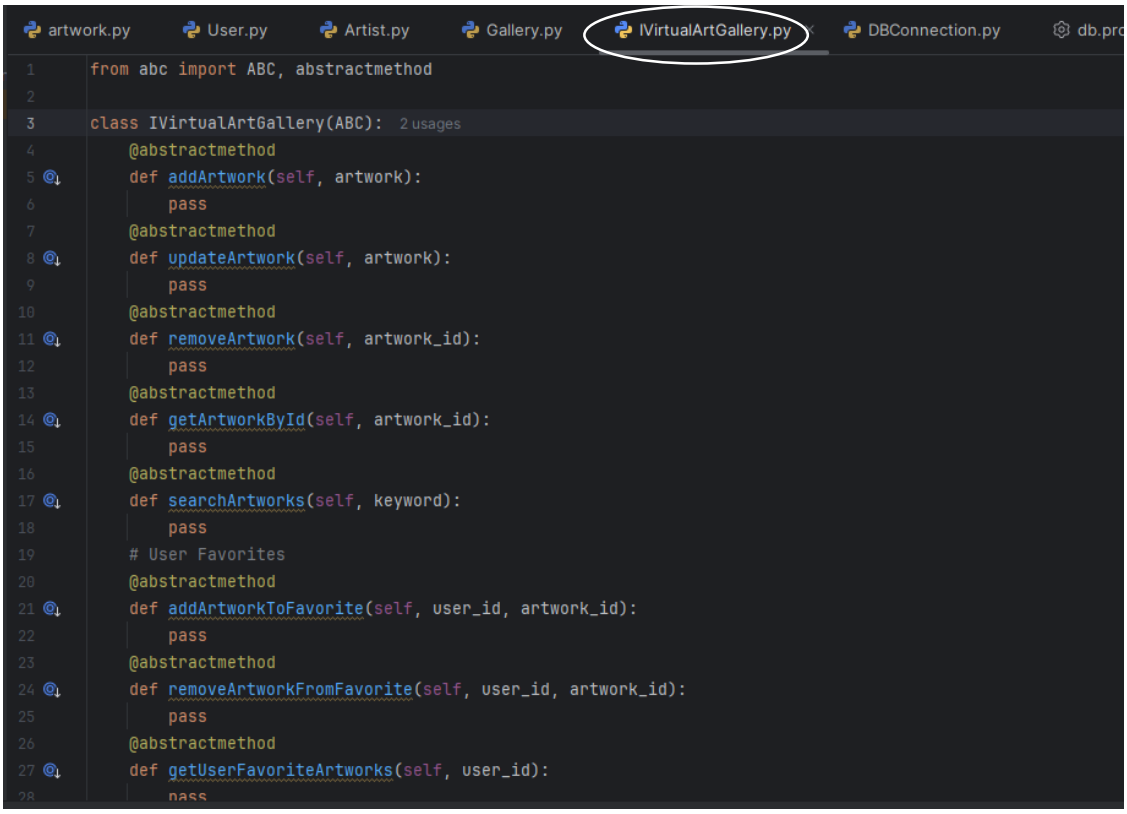
```python
class Artist:
    def __init__(self, artist_id=None, name=None, biography=None, birth_date=None, nationality=None, websit
        self.__artist_id = artist_id
        self.__name = name
        self.__biography = biography
        self.__birth_date = birth_date
        self.__nationality = nationality
        self.__website = website
        self.__contact_info = contact_info

    def get_artist_id(self): return self.__artist_id  2 usages (2 dynamic)
    def set_artist_id(self, value): self.__artist_id = value
    def get_name(self): return self.__name
    def set_name(self, value): self.__name = value
    def get_biography(self): return self.__biography
    def set_biography(self, value): self.__biography = value
    def get_birth_date(self): return self.__birth_date
    def set_birth_date(self, value): self.__birth_date = value
    def get_nationality(self): return self.__nationality
    def set_nationality(self, value): self.__nationality = value
    def get_website(self): return self.__website
    def set_website(self, value): self.__website = value
    def get_contact_info(self): return self.__contact_info
    def set_contact_info(self, value): self.__contact_info = value

```



2. Interface Layer (dao/IVirtualArtGallery.py)

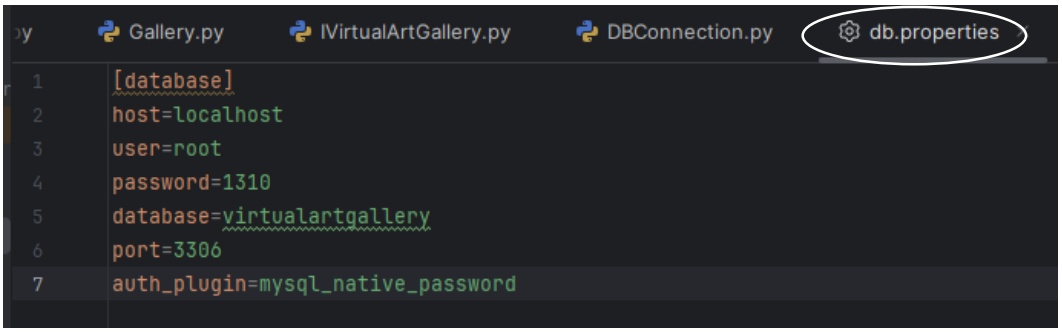
Defines abstract methods for:

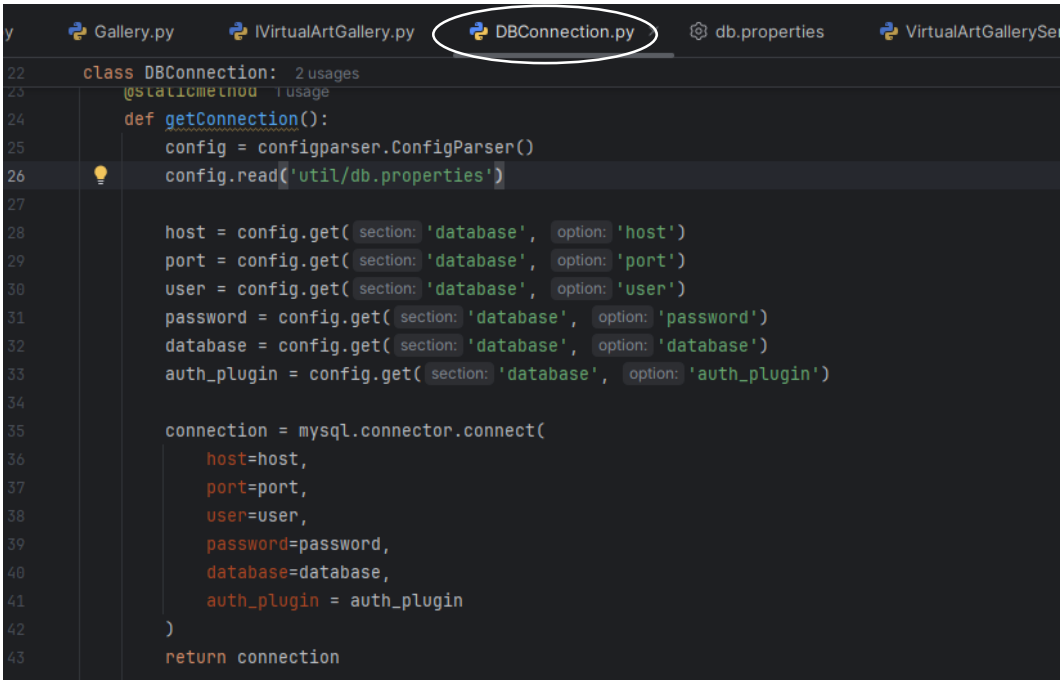
- Artwork management (add, update, delete, search)
- User favorite management (add/remove favorites, fetch list)



3. Database Utility (util/DBConnection.py)

- Reads MySQL credentials from db.properties
- Establishes and returns a MySQLConnection object

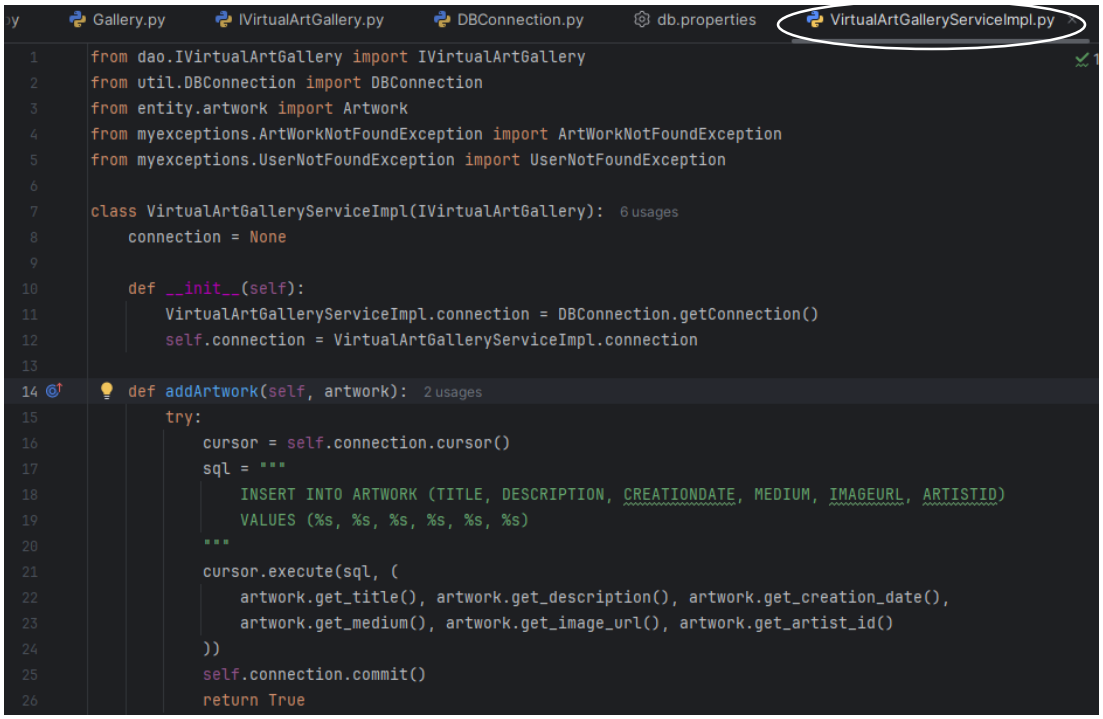




4. Service Implementation (dao/VirtualArtGalleryServiceImpl.py)

Implements all interface methods with actual database operations:

- SQL queries (INSERT, SELECT, UPDATE, DELETE)
- Interacts with MySQL through a unified connection utility



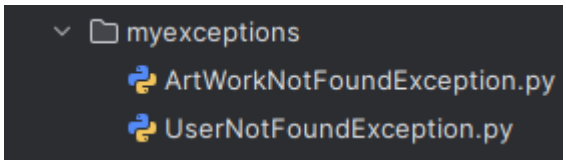

```python
from dao.IVirtualArtGallery import IVirtualArtGallery
from util.DBConnection import DBConnection
from entity.artwork import Artwork
from myexceptions.ArtWorkNotFoundException import ArtWorkNotFoundException
from myexceptions.UserNotFoundException import UserNotFoundException

class VirtualArtGalleryServiceImpl(IVirtualArtGallery):  6 usages
    connection = None

    def __init__(self):
        VirtualArtGalleryServiceImpl.connection = DBConnection.getConnection()
        self.connection = VirtualArtGalleryServiceImpl.connection

    def addArtwork(self, artwork):  2 usages
        try:
            cursor = self.connection.cursor()
            sql = """
                INSERT INTO ARTWORK (TITLE, DESCRIPTION, CREATIONDATE, MEDIUM, IMAGEURL, ARTISTID)
                VALUES (%s, %s, %s, %s, %s, %s)
            """
            cursor.execute(sql, (
                artwork.get_title(), artwork.get_description(), artwork.get_creation_date(),
                artwork.get_medium(), artwork.get_image_url(), artwork.get_artist_id()
            ))
            self.connection.commit()
            return True
        except Exception as e:
            print("Error adding artwork:", e)
            return False

    def updateArtwork(self, artwork):  2 usages
        try:
            cursor = self.connection.cursor()
            sql = """
                UPDATE ARTWORK SET TITLE=%s, DESCRIPTION=%s, CREATIONDATE=%s, MEDIUM=%s, IMAGEURL=%s, ARTISTID=%s
                WHERE ARTWORKID=%s
            """
            cursor.execute(sql, (
                artwork.get_title(), artwork.get_description(), artwork.get_creation_date(),
                artwork.get_medium(), artwork.get_image_url(), artwork.get_artist_id(), artwork.get_artwork_id()
            ))
            self.connection.commit()
            return True
        except Exception as e:
            print("Error updating artwork:", e)
            return False
```

```
    def removeArtwork(self, artwork_id):  2 usages
        try:
            cursor = self.connection.cursor()
            cursor.execute("DELETE FROM ARTWORK WHERE ARTWORKID = %s", (artwork_id,))
            if cursor.rowcount == 0:
                raise ArtWorkNotFoundException()
            self.connection.commit()
            return True
        except Exception as e:
            print("Error removing artwork:", e)
            return False

    def getArtworkById(self, artwork_id):  2 usages
        try:
            cursor = self.connection.cursor()
            cursor.execute("SELECT * FROM ARTWORK WHERE ARTWORKID = %s", (artwork_id,))
            row = cursor.fetchone()
            if row:
                return Artwork(*row)
            else:
                raise ArtWorkNotFoundException()
        except Exception as e:
            print("Error fetching artwork:", e)
            raise
```

```
    def searchArtworks(self, keyword):  2 usages
        try:
            cursor = self.connection.cursor()
            cursor.execute("SELECT * FROM ARTWORK WHERE TITLE LIKE %s", (f"%{keyword}%",))
            rows = cursor.fetchall()
            return [Artwork(*row) for row in rows]
        except Exception as e:
            print("Error searching artworks:", e)
            return []

    def addArtworkToFavorite(self, user_id, artwork_id):  3 usages
        try:
            cursor = self.connection.cursor()
            cursor.execute("INSERT INTO USER_FAVORITE_ARTWORK (USERID, ARTWORKID) VALUES (%s, %s)", (user_id, artw
            self.connection.commit()
            return True
        except Exception as e:
            print("Error adding to favorites:", e)
            return False

    def removeArtworkFromFavorite(self, user_id, artwork_id):  2 usages
        try:
            cursor = self.connection.cursor()
            cursor.execute("DELETE FROM USER_FAVORITE_ARTWORK WHERE USERID = %s AND ARTWORKID = %s", (user_id, a
            if cursor.rowcount == 0:
                raise ArtWorkNotFoundException("Artwork not found in user favorites")
            self.connection.commit()
            return True
        except Exception as e:
            print("Error removing from favorites:", e)
            return False

    def getUserFavoriteArtworks(self, user_id):  2 usages
        try:
            cursor = self.connection.cursor()
            cursor.execute("SELECT * FROM USER WHERE USERID = %s", (user_id,))
            if not cursor.fetchone():
                raise UserNotFoundException()
```

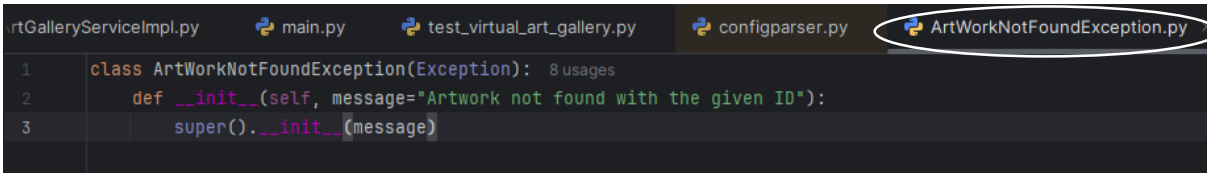
```
            cursor.execute("""
                SELECT A.* FROM ARTWORK A
                JOIN USER_FAVORITE_ARTWORK UFA ON A.ARTWORKID = UFA.ARTWORKID
                WHERE UFA.USERID = %s
            """, (user_id,))
            rows = cursor.fetchall()
            return [Artwork(*row) for row in rows]
        except Exception as e:
            print("Error getting favorite artworks:", e)
            return []
```

5. Exception Handling (myexceptions)

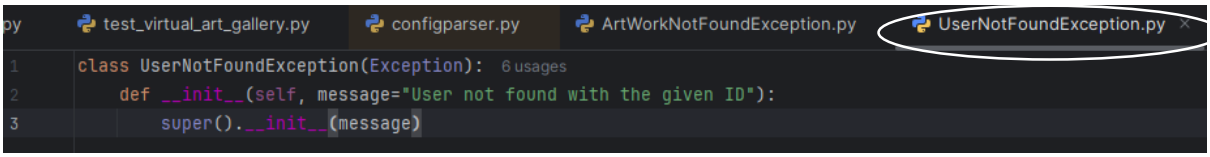


Custom exceptions:

- ArtWorkNotFoundException



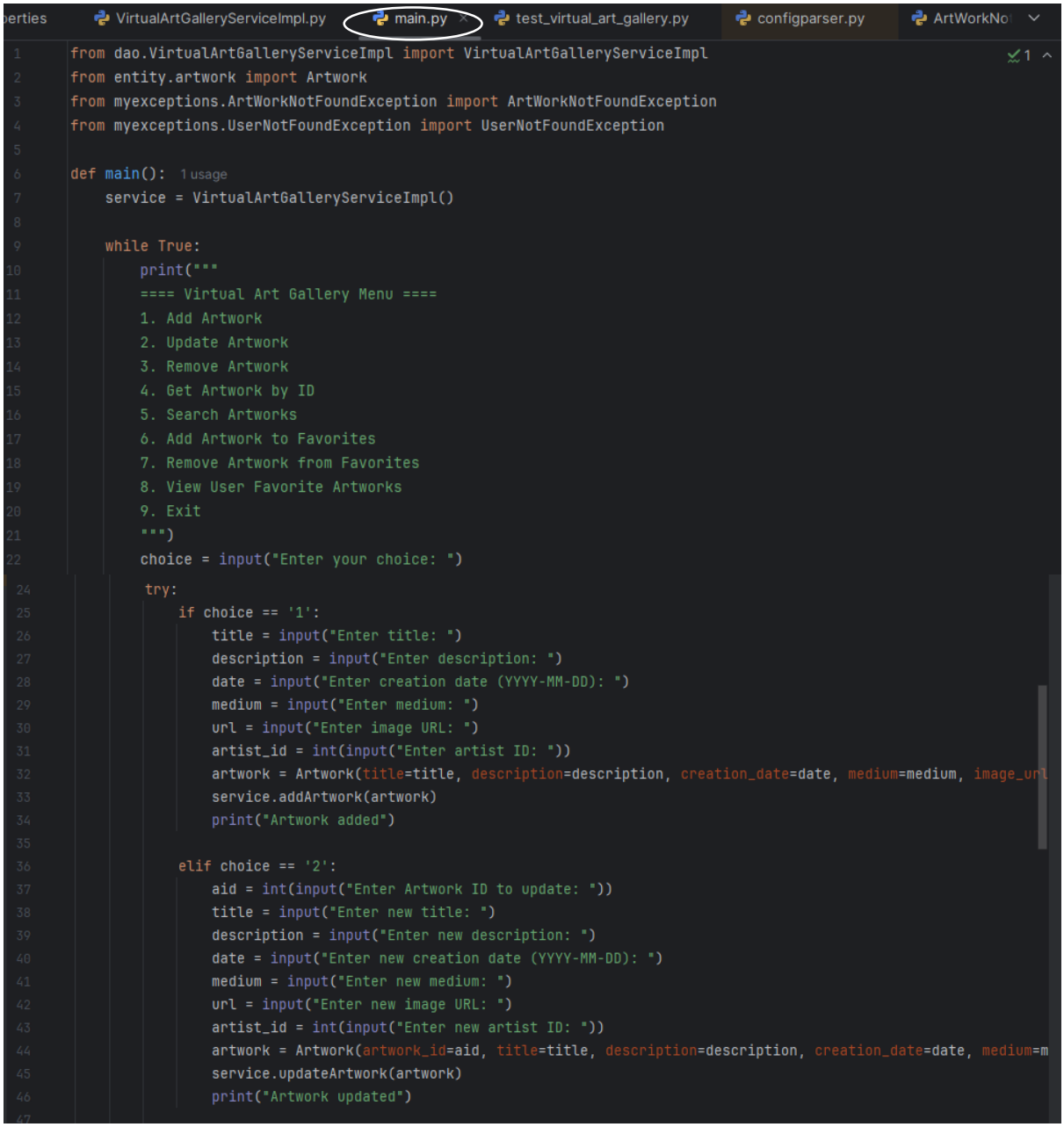
- UserNotFoundException



6. Main Module (main/MainModule.py)

Menu-driven console program to:

- Trigger all services via user input
- Display results and error messages
- Loops back for continuous interaction



```
            elif choice == '3':
                aid = int(input("Enter Artwork ID to remove: "))
                service.removeArtwork(aid)
                print("Artwork removed")

            elif choice == '4':
                aid = int(input("Enter Artwork ID: "))
                try:
                    art = service.getArtworkById(aid)
                    print("Artwork Title:", art.get_title())
                except ArtWorkNotFoundException as ae:
                    print(ae)

            elif choice == '5':
                key = input("Enter keyword to search: ")
                results = service.searchArtworks(key)
                for art in results:
                    print("ID:", art.get_artwork_id(), "Title:", art.get_title())

            elif choice == '6':
                uid = int(input("Enter User ID: "))
                aid = int(input("Enter Artwork ID: "))
                service.addArtworkToFavorite(uid, aid)
                print("Favorite added")

             elif choice == '7':
                 uid = int(input("Enter User ID: "))
                 aid = int(input("Enter Artwork ID: "))
                 service.removeArtworkFromFavorite(uid, aid)
                 print("Favorite removed")

             elif choice == '8':
                 uid = int(input("Enter User ID: "))
                 try:
                     favs = service.getUserFavoriteArtworks(uid)
                     for art in favs:
                         print("ID:", art.get_artwork_id(), "Title:", art.get_title())
                 except UserNotFoundException as ue:
                     print(ue)

             elif choice == '9':
                 print("Exiting...")
                 break
             else:
                 print("Invalid choice")
         except Exception as e:
             print("Error:", e)
main()
```

Console output:

```
Run  main
C:\Users\Admin\PycharmProjects\VIRTUAL_ART_GALLERY\.venv\Scripts\python.exe C:\Users\Admin\PycharmProjects\VIRTUAL_ART_GALLERY\main.py

        ==== 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. View User Favorite Artworks
        9. Exit

Enter your choice:
```

- Add artwork:

```
Enter your choice: 1
Enter title: kutrallam
Enter description: beauty of kutrallam falls
Enter creation date (YYYY-MM-DD): 2025-05-30
Enter medium: oil
Enter image URL: https://gallery.com/images/kutrallam.jpg
Enter artist ID: 4
Artwork added
```

```
mysql> SELECT * FROM ARTWORK;
+-----------+-----------------------+-----------------------------+--------------+-----------+-----------------------------------------+----------+
| ARTWORKID | TITLE                 | DESCRIPTION                 | CREATIONDATE | MEDIUM    | IMAGEURL                                | ARTISTID |
+-----------+-----------------------+-----------------------------+--------------+-----------+-----------------------------------------+----------+
|         1 | tanjai periya kovil   | tanjaiyin azhagu            | 2024-12-18   | oil       | https://gallery.com/images/tanjai.jpg   |        2 |
|         2 | Madurai Market        | Flower market morning scene | 2021-07-15   | Watercolor | http://gallery.com/images/madurai.jpg   |        1 |
|         3 | Chennai Marina        | Evening beach view          | 2023-03-22   | Acrylic   | http://gallery.com/images/marina.jpg    |        3 |
|         4 | Coimbatore Hills      | Western Ghats scenery       | 2020-08-05   | Oil       | http://gallery.com/images/coimbatore.jpg |        4 |
|         5 | Kumbakonam Streets    | Temple town street view     | 2023-02-18   | Sketch    | http://gallery.com/images/kumbakonam.jpg |        5 |
|         6 | Salem Fields          | Sunset over rice fields     | 2022-09-09   | Pastel    | http://gallery.com/images/salem.jpg     |        6 |
|         7 | Tirunelveli Traditions | Folk traditions of Tirunelveli | 2021-11-11 | Ink       | http://gallery.com/images/tirunelveli.jpg |        7 |
|        11 | kutrallam             | beauty of kutrallam falls   | 2025-05-30   | oil       | https://gallery.com/images/kutrallam.jpg |        4 |
+-----------+-----------------------+-----------------------------+--------------+-----------+-----------------------------------------+----------+
8 rows in set (0.00 sec)
```

- Update artwork:

```
Enter your choice: 2
Enter Artwork ID to update: 11
Enter new title: kutrallam falls
Enter new description: beauty of kutrallam
Enter new creation date (YYYY-MM-DD): 2025-05-30
Enter new medium: oil
Enter new image URL: https://gallery.com/images/kutrallam.jpg
Enter new artist ID: 4
Artwork updated
```

The tile and description is updated

```
mysql> SELECT * FROM ARTWORK;
+-----------+-----------------------+-----------------------------+--------------+-----------+-----------------------------------------+----------+
| ARTWORKID | TITLE                 | DESCRIPTION                 | CREATIONDATE | MEDIUM    | IMAGEURL                                | ARTISTID |
+-----------+-----------------------+-----------------------------+--------------+-----------+-----------------------------------------+----------+
|         1 | tanjai periya kovil   | tanjaiyin azhagu            | 2024-12-18   | oil       | https://gallery.com/images/tanjai.jpg   |        2 |
|         2 | Madurai Market        | Flower market morning scene | 2021-07-15   | Watercolor | http://gallery.com/images/madurai.jpg   |        1 |
|         3 | Chennai Marina        | Evening beach view          | 2023-03-22   | Acrylic   | http://gallery.com/images/marina.jpg    |        3 |
|         4 | Coimbatore Hills      | Western Ghats scenery       | 2020-08-05   | Oil       | http://gallery.com/images/coimbatore.jpg |        4 |
|         5 | Kumbakonam Streets    | Temple town street view     | 2023-02-18   | Sketch    | http://gallery.com/images/kumbakonam.jpg |        5 |
|         6 | Salem Fields          | Sunset over rice fields     | 2022-09-09   | Pastel    | http://gallery.com/images/salem.jpg     |        6 |
|         7 | Tirunelveli Traditions | Folk traditions of Tirunelveli | 2021-11-11 | Ink       | http://gallery.com/images/tirunelveli.jpg |        7 |
|        11 | kutrallam falls       | beauty of kutrallam         | 2025-05-30   | oil       | https://gallery.com/images/kutrallam.jpg |        4 |
+-----------+-----------------------+-----------------------------+--------------+-----------+-----------------------------------------+----------+
8 rows in set (0.00 sec)
```

- Remove artwork:

```
Enter your choice: 3
Enter Artwork ID to remove: 11
Artwork removed
```

```
mysql> SELECT * FROM ARTWORK;
+-----------+-----------------------+-------------------------------+--------------+-----------+-----------------------------------------+----------+
| ARTWORKID | TITLE                 | DESCRIPTION                   | CREATIONDATE | MEDIUM    | IMAGEURL                                | ARTISTID |
+-----------+-----------------------+-------------------------------+--------------+-----------+-----------------------------------------+----------+
|         1 | tanjai periya kovil   | tanjaiyin azhagu              | 2024-12-18   | oil       | https://gallery.com/images/tanjai.jpg   |        2 |
|         2 | Madurai Market        | Flower market morning scene   | 2021-07-15   | Watercolor | http://gallery.com/images/madurai.jpg   |        1 |
|         3 | Chennai Marina        | Evening beach view            | 2023-03-22   | Acrylic   | http://gallery.com/images/marina.jpg    |        3 |
|         4 | Coimbatore Hills      | Western Ghats scenery         | 2020-08-05   | Oil       | http://gallery.com/images/coimbatore.jpg |        4 |
|         5 | Kumbakonam Streets    | Temple town street view       | 2023-02-18   | Sketch    | http://gallery.com/images/kumbakonam.jpg |        5 |
|         6 | Salem Fields          | Sunset over rice fields       | 2022-09-09   | Pastel    | http://gallery.com/images/salem.jpg     |        6 |
|         7 | Tirunelveli Traditions | Folk traditions of Tirunelveli | 2021-11-11   | Ink       | http://gallery.com/images/tirunelveli.jpg |        7 |
+-----------+-----------------------+-------------------------------+--------------+-----------+-----------------------------------------+----------+
7 rows in set (0.00 sec)
```

The artwork ID 11 is removed

Exceptions:

```
Enter your choice: 3
Enter Artwork ID to remove: 8
Error removing artwork: Artwork not found with the given ID
```

- Get artwork by ID:

```
Enter your choice: 4
Enter Artwork ID: 2
Artwork Title: Madurai Market
```

- Search artwork:

```
Enter your choice: 5
Enter keyword to search: chennai
ID: 3 Title: Chennai Marina
```

- Add artwork to favourites:

```
Enter your choice: 6
Enter User ID: 1
Enter Artwork ID: 6
Favorite added
```

```
mysql> SELECT * FROM user_favorite_artwork;
+--------+-----------+
| USERID | ARTWORKID |
+--------+-----------+
|      2 |         1 |
|      1 |         2 |
|      3 |         2 |
|      1 |         3 |
|      3 |         3 |
|      4 |         4 |
|      2 |         5 |
|      5 |         5 |
|      1 |         6 |
|      6 |         6 |
|      7 |         7 |
+--------+-----------+
11 rows in set (0.00 sec)
```

- Remove artwork from favourite:

```
Enter your choice: 7
Enter User ID: 1
Enter Artwork ID: 6

Favorite removed
```

```
mysql> SELECT * FROM user_favorite_artwork;
+--------+-----------+
| USERID | ARTWORKID |
+--------+-----------+
|      2 |         1 |
|      1 |         2 |
|      3 |         2 |
|      1 |         3 |
|      3 |         3 |
|      4 |         4 |
|      2 |         5 |
|      5 |         5 |
|      6 |         6 |
|      7 |         7 |
+--------+-----------+
10 rows in set (0.00 sec)
```

The artwork is removed from favourite.

- View user favourite artworks:

```
Enter your choice: 8
Enter User ID: 1
ID: 2 Title: Madurai Market
ID: 3 Title: Chennai Marina
```

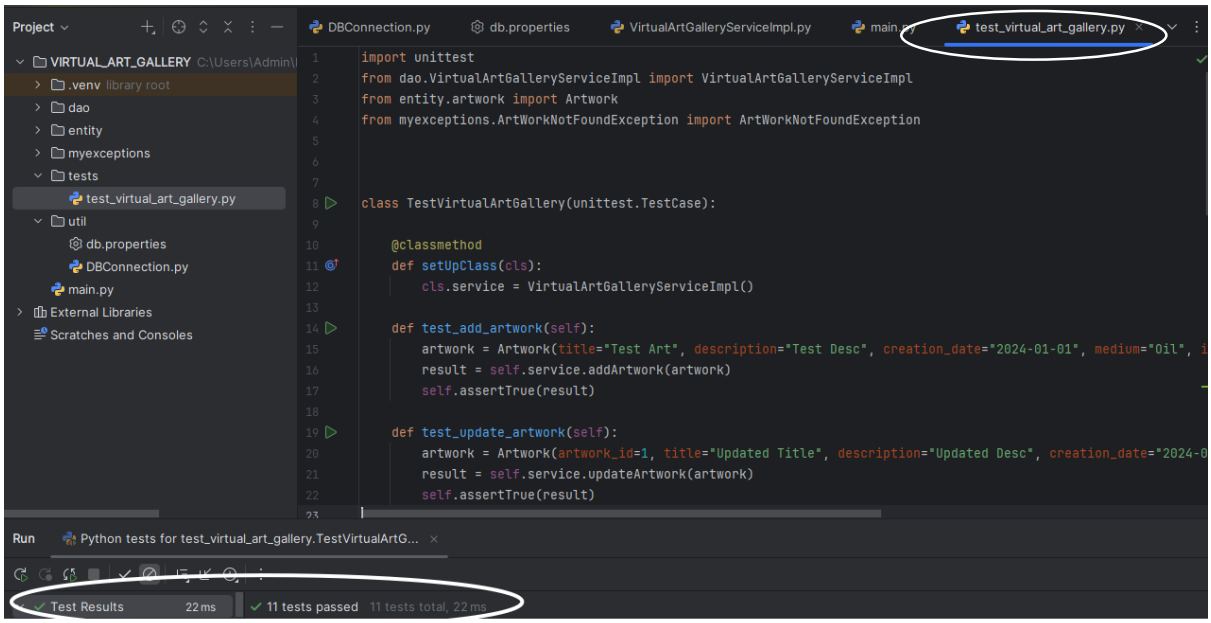
- Exiting:

```
==== 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. View User Favorite Artworks
9. Exit

Enter your choice: 9
Exiting...

Process finished with exit code 0
```

7. Unit Testing (tests)

Tested key functionalities like:

- Adding/updating/removing artwork
- Handling invalid IDs
- Managing user favourites
  Uses Python's built-in unittest module.



Conclusion:

The Virtual Art Gallery project successfully demonstrates how Python can be used to build a console-based application for managing and exploring artworks, artists, users, and galleries. By applying object-oriented programming principles, the system models real-world entities in a structured and reusable way. Integration with a MySQL database allows for reliable data storage and retrieval.

Though the project does not include a graphical user interface, it effectively performs all core operations through a user-friendly command-line interface. Key functionalities such as adding, updating, removing, and searching artworks are implemented and tested. The system also supports features like managing user favourites, showcasing how basic user interaction can be handled using simple console outputs.

Unit testing ensures that the core services work correctly, while a clean separation of concerns in the code base (entities, services, exceptions, and utilities) makes the project maintainable and scalable. This project serves as a solid foundation for building more advanced applications in the future, such as a web-based or GUI-based virtual art gallery.

Overall, the project meets its primary goal of simulating a virtual art collection management system and provides valuable experience in Python programming, database connectivity and exception handling.