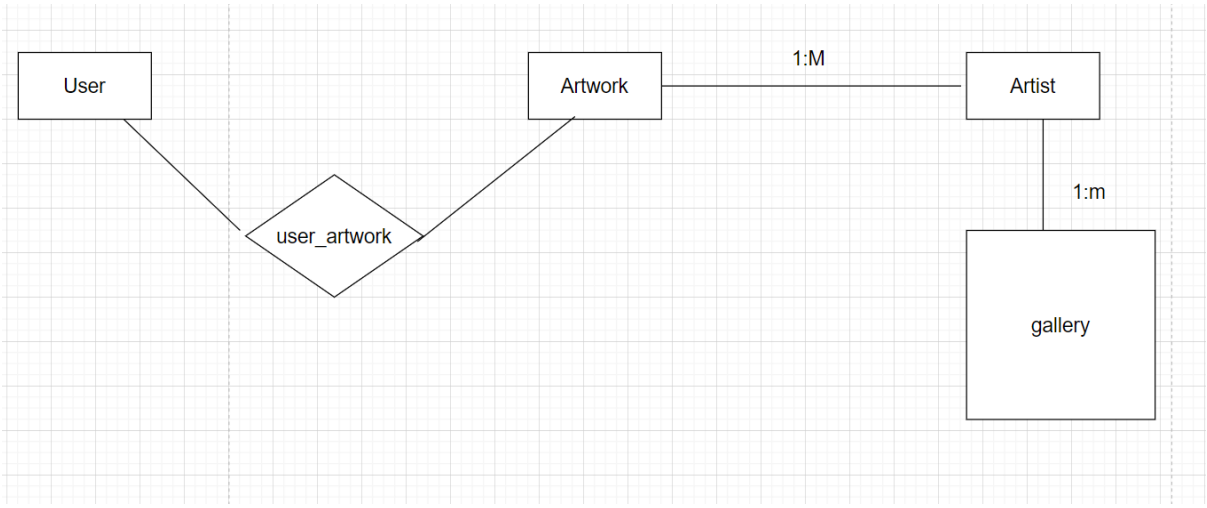
Virtual Art Gallery



Case Study: Virtual Art Gallery

Instructions

- Project submissions should be done through the partcipants' Github repository and the link should be shared with trainers and Hexavarsity.
- Each section builds upon the previous one, and by the end, you will have a comprehensive Virtual Art Gallery implemented with a strong focus on SQL, control flow statements, loops, arrays, collections, exception handling, database interaction and Unit Testing.
- Follow object-oriented principles throughout the project. Use classes and objects to model real-world entities, encapsulate data and behavior, and ensure code reusability.
- Throw user defined exceptions from corresponding methods and handled.

- The following Directory structure is to be followed in the application.
  - **entity**
    - Create entity classes in this package. All entity class should not have any business logic.
  - **dao**
    - Create Service Provider interface to showcase functionalities.
    - Create the implementation class for the above interface with db interaction.
  - **exception**
    - Create user defined exceptions in this package and handle exceptions whenever needed.
  - **util**
    - Create a DBPropertyUtil class with a static function which takes property file name as parameter and returns connection string.
    - Create a DBConnUtil class which holds static method which takes connection string as parameter file and returns connection object(Use method defined in DBPropertyUtil class to get the connection String).
  - **main**
    - Create a class MainModule and demonstrate the functionalities in a menu driven application.

Key Functionalities:

Artwork management The Virtual Art Gallery System aims to provide an immersive and interactive experience for art enthusiasts to explore, view, and appreciate a diverse collection of artworks online.

Personal Galleries: Enable users to create their virtual galleries and curate their collections.

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

// 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

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

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.

- **Relationships:**
- **Artwork - Artist (Many-to-One)**

  An artwork is created by one artist.

  Artwork.ArtistID (Foreign Key) references Artist.ArtistID.
- **User - Favorite Artwork (Many-to-Many)**

  A user can have many favorite artworks, and an artwork can be a favorite of multiple users.

  User_Favorite_Artwork (junction table):

  UserID (Foreign Key) references User.UserID.

  ArtworkID (Foreign Key) references Artwork.ArtworkID.
- **Artist - Gallery (One-to-Many)**

  An artist can be associated with multiple galleries, but a gallery can have only one curator (artist).

  Gallery.ArtistID (Foreign Key) references Artist.ArtistID.
- **Artwork - Gallery (Many-to-Many)**

  An artwork can be displayed in multiple galleries, and a gallery can have multiple artworks.

  Artwork_Gallery (junction table):

  ArtworkID (Foreign Key) references Artwork.ArtworkID.

  GalleryID (Foreign Key) references Gallery.GalleryID.