**Hexaware Case Study:**

**Virtual Art Gallery**

**Key Functionalities:**

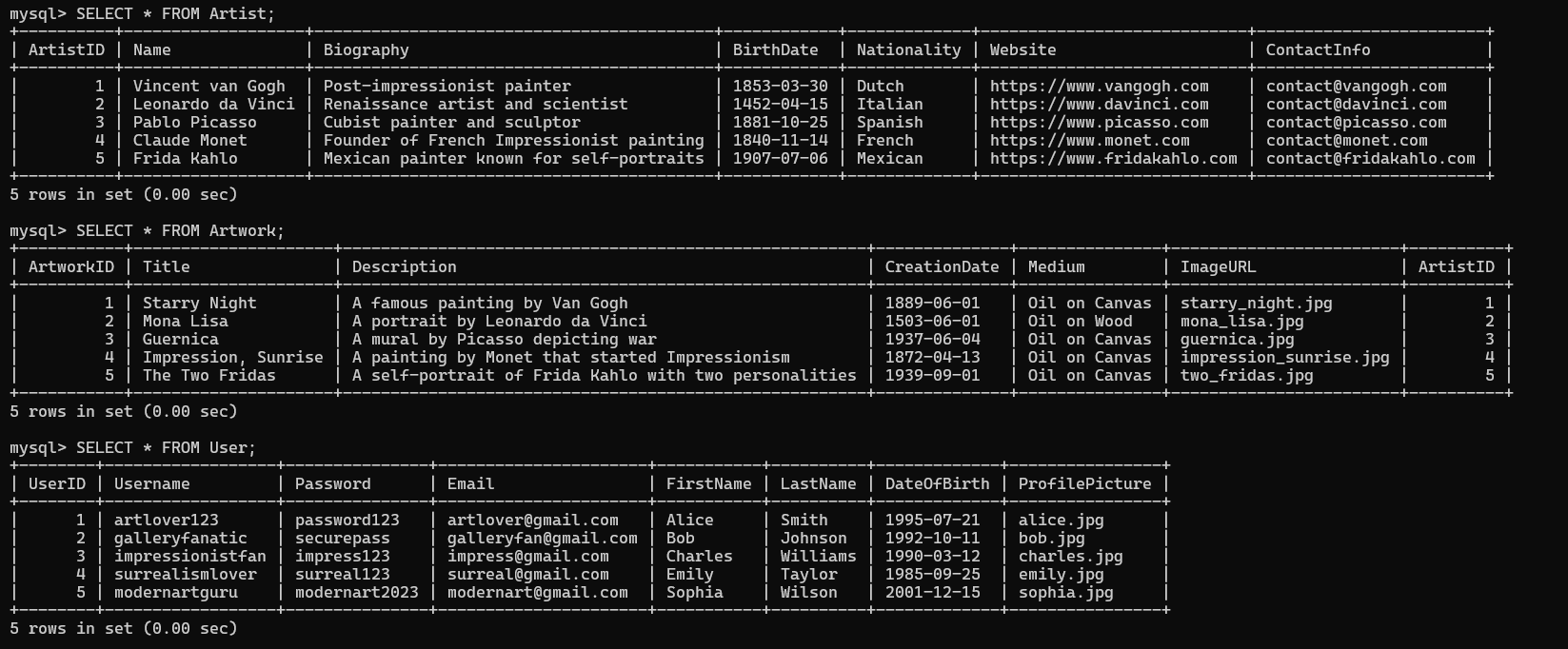
**Artwork management The Virtual Art Gallery System aims to provide an immersive and**

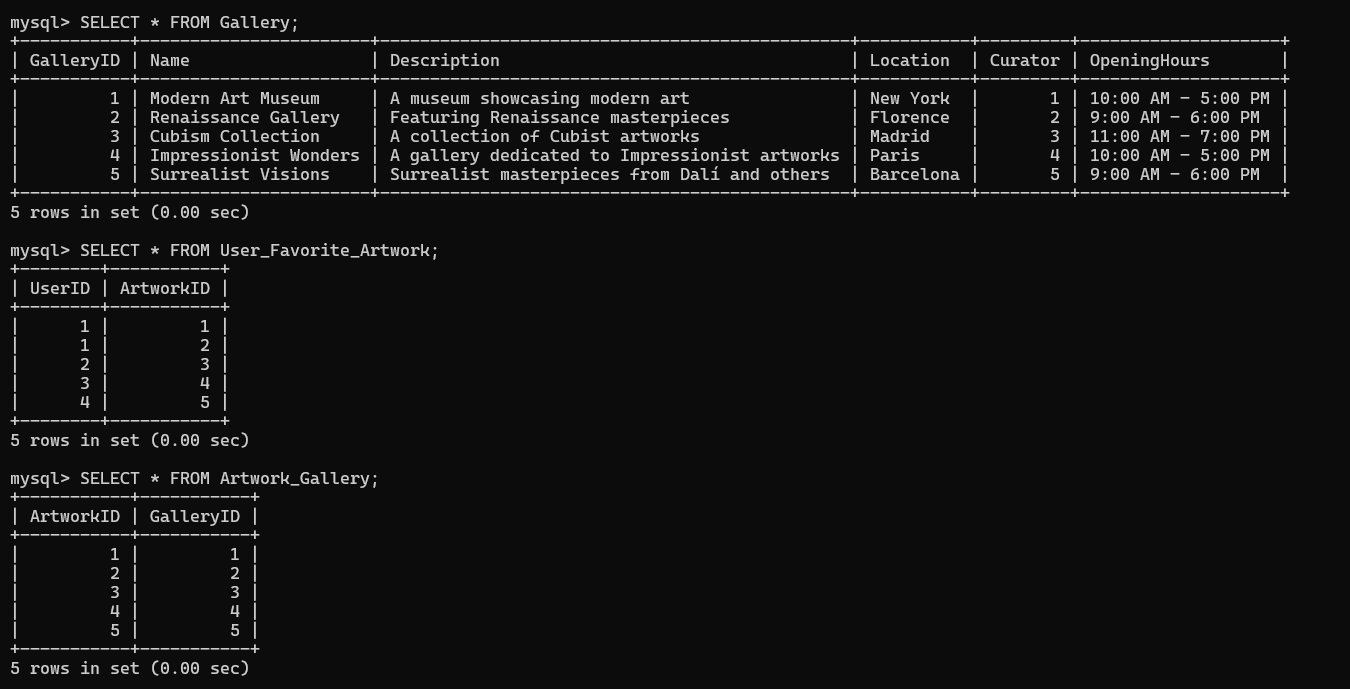
**interactive experience for art enthusiasts to explore, view, and appreciate a diverse collection of**

**artworks online.**

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







**Coding**

**Create the model/entity classes corresponding to the schema within package entity with**

**variables declared private, constructors(default and parametrized) and getters,setters )**

**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**

**}**

|  |
| --- |
| **package** dao;  **import** entity.Artwork;  **import** myexceptions.ArtWorkNotFoundException;  **import** myexceptions.UserNotFoundException;  **import** java.util.List;  **public** **interface** IVirtualArtGallery {  **boolean** addArtwork(Artwork artwork);  **boolean** updateArtwork(Artwork artwork);  **boolean** removeArtwork(**int** artworkId);  Artwork getArtworkById(**int** artworkId) **throws** ArtWorkNotFoundException;  List<Artwork> searchArtworks(String keyword);  **boolean** addArtworkToFavorite(**int** userId, **int** artworkId) **throws** ArtWorkNotFoundException, UserNotFoundException;  **boolean** removeArtworkFromFavorite(**int** userId, **int** artworkId) **throws** ArtWorkNotFoundException, UserNotFoundException;  List<Artwork> getUserFavoriteArtworks(**int** userId) **throws** UserNotFoundException;  } |

**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.**

**DBConnection**

|  |
| --- |
| package util;  import java.io.IOException;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.SQLException;  public class DBConnection {  private static final String fileName = "src/db.properties";  public static Connection getDbConnection() {  Connection con = null;  String connString=null;  try {  connString = PropertyUtil.getConnectionString(fileName);  } catch (IOException e) {  System.out.println("Could not load DB properties file.");  e.printStackTrace();  }  if(connString!=null) {  try {  con=DriverManager.getConnection(connString);  }  catch (SQLException e) {  System.out.println("Database connection failed.");  e.printStackTrace();  }  }  return con;  }  } |

**PropertyUtil**

|  |
| --- |
| package util;  import java.io.FileInputStream;  import java.io.IOException;  import java.util.Properties;  public class PropertyUtil {  public static String getConnectionString(String fileName) throws IOException {  Properties props = new Properties();  FileInputStream fis = new FileInputStream(fileName);  props.load(fis);  String url = props.getProperty("db.url");  String user = props.getProperty("db.user");  String password = props.getProperty("db.password");  return url + "?user=" + user + "&password=" + password;  }  public static String getDriverClass(String fileName) throws IOException {  Properties props = new Properties();  FileInputStream fis = new FileInputStream(fileName);  props.load(fis);  return props.getProperty("db.driver");  }  } |

**db.properties**

|  |
| --- |
| db.driver=com.mysql.cj.jdbc.Driver  db.url=jdbc:mysql://localhost:3306/virtualartgallery  db.user=root  db.password=cats |

**8: Service implementation**

**1. Create a Service class CrimeAnalysisServiceImpl in dao with a static variable named connection**

**of type Connection which can be assigned in the constructor by invoking the getConnection()**

**method in DBConnection class**

**2. Provide implementation for all the methods in the interface.**

|  |
| --- |
| package dao;  import entity.Artwork;  import entity.Artist;  import entity.Gallery;  import util.DBConnection;  import java.sql.\*;  import java.util.ArrayList;  import java.util.List;  public class VirtualArtGalleryImpl {  public boolean addArtwork(Artwork artwork) {  try (Connection conn = DBConnection.getDbConnection()) {  String sql = "INSERT INTO Artwork (Title, Description, CreationDate, Medium, ImageURL, ArtistID) VALUES (?, ?, ?, ?, ?, ?)";  PreparedStatement ps = conn.prepareStatement(sql);  ps.setString(1, artwork.getTitle());  ps.setString(2, artwork.getDescription());  ps.setString(3, artwork.getCreationDate());  ps.setString(4, artwork.getMedium());  ps.setString(5, artwork.getImageUrl());  ps.setInt(6, artwork.getArtistId());  int rows = ps.executeUpdate();  return rows > 0;  } catch (SQLException e) {  System.out.println("Failed to add artwork: " + e.getMessage());  return false;  }  }  public boolean updateArtwork(Artwork artwork) {  try (Connection conn = DBConnection.getDbConnection()) {  String sql = "UPDATE Artwork SET Title=?, Description=?, CreationDate=?, Medium=?, ImageURL=?, ArtistID=? WHERE ArtworkID=?";  PreparedStatement ps = conn.prepareStatement(sql);  ps.setString(1, artwork.getTitle());  ps.setString(2, artwork.getDescription());  ps.setString(3, artwork.getCreationDate());  ps.setString(4, artwork.getMedium());  ps.setString(5, artwork.getImageUrl());  ps.setInt(6, artwork.getArtistId());  ps.setInt(7, artwork.getArtworkId());  return ps.executeUpdate() > 0;  } catch (SQLException e) {  System.out.println("Failed to update artwork: " + e.getMessage());  return false;  }  }  public boolean removeArtwork(int artworkId) {  try (Connection conn = DBConnection.getDbConnection()) {  String sql = "DELETE FROM Artwork WHERE ArtworkID = ?";  PreparedStatement ps = conn.prepareStatement(sql);  ps.setInt(1, artworkId);  return ps.executeUpdate() > 0;  } catch (SQLException e) {  System.out.println("Failed to remove artwork: " + e.getMessage());  return false;  }  }  public Artwork getArtworkById(int artworkId) {  try (Connection conn = DBConnection.getDbConnection()) {  String sql = "SELECT \* FROM Artwork WHERE ArtworkID = ?";  PreparedStatement ps = conn.prepareStatement(sql);  ps.setInt(1, artworkId);  ResultSet rs = ps.executeQuery();  if (rs.next()) {  Artwork art = new Artwork();  art.setArtworkId(rs.getInt("ArtworkID"));  art.setTitle(rs.getString("Title"));  art.setDescription(rs.getString("Description"));  art.setCreationDate(rs.getString("CreationDate"));  art.setMedium(rs.getString("Medium"));  art.setImageUrl(rs.getString("ImageURL"));  art.setArtistId(rs.getInt("ArtistID"));  return art;  }  } catch (SQLException e) {  System.out.println("Error fetching artwork by ID: " + e.getMessage());  }  return null;  }  public List<Artwork> searchArtworks(String keyword) {  List<Artwork> results = new ArrayList<>();  try (Connection conn = DBConnection.getDbConnection()) {  String sql = "SELECT \* FROM Artwork WHERE Title LIKE ?";  PreparedStatement ps = conn.prepareStatement(sql);  ps.setString(1, "%" + keyword + "%");  ResultSet rs = ps.executeQuery();  while (rs.next()) {  Artwork art = new Artwork();  art.setArtworkId(rs.getInt("ArtworkID"));  art.setTitle(rs.getString("Title"));  art.setDescription(rs.getString("Description"));  art.setCreationDate(rs.getString("CreationDate"));  art.setMedium(rs.getString("Medium"));  art.setImageUrl(rs.getString("ImageURL"));  art.setArtistId(rs.getInt("ArtistID"));  results.add(art);  }  } catch (SQLException e) {  System.out.println("Error searching artworks: " + e.getMessage());  }  return results;  }  public boolean addArtworkToFavorite(int userId, int artworkId) {  try (Connection conn = DBConnection.getDbConnection()) {  String sql = "INSERT INTO User\_Favorite\_Artwork (UserID, ArtworkID) VALUES (?, ?)";  PreparedStatement ps = conn.prepareStatement(sql);  ps.setInt(1, userId);  ps.setInt(2, artworkId);  return ps.executeUpdate() > 0;  } catch (SQLException e) {  System.out.println("Failed to add to favorites: " + e.getMessage());  return false;  }  }  public boolean removeArtworkFromFavorite(int userId, int artworkId) {  try (Connection conn = DBConnection.getDbConnection()) {  String sql = "DELETE FROM User\_Favorite\_Artwork WHERE UserID = ? AND ArtworkID = ?";  PreparedStatement ps = conn.prepareStatement(sql);  ps.setInt(1, userId);  ps.setInt(2, artworkId);  return ps.executeUpdate() > 0;  } catch (SQLException e) {  System.out.println("Failed to remove from favorites: " + e.getMessage());  return false;  }  }  public List<Artwork> getUserFavoriteArtworks(int userId) {  List<Artwork> favorites = new ArrayList<>();  try (Connection conn = DBConnection.getDbConnection()) {  String sql = "SELECT a.\* FROM Artwork a JOIN User\_Favorite\_Artwork ufa ON a.ArtworkID = ufa.ArtworkID WHERE ufa.UserID = ?";  PreparedStatement ps = conn.prepareStatement(sql);  ps.setInt(1, userId);  ResultSet rs = ps.executeQuery();  while (rs.next()) {  Artwork art = new Artwork();  art.setArtworkId(rs.getInt("ArtworkID"));  art.setTitle(rs.getString("Title"));  art.setDescription(rs.getString("Description"));  art.setCreationDate(rs.getString("CreationDate"));  art.setMedium(rs.getString("Medium"));  art.setImageUrl(rs.getString("ImageURL"));  art.setArtistId(rs.getInt("ArtistID"));  favorites.add(art);  }  } catch (SQLException e) {  System.out.println("Error fetching favorites: " + e.getMessage());  }  return favorites;  }    private List<Gallery> galleries = new ArrayList<>();  private List<Artist> artists = new ArrayList<>();  private int galleryIdCounter = 1;  private int artistIdCounter = 1;  public boolean addGallery(Gallery gallery) {  gallery.setGalleryID(galleryIdCounter++);  return galleries.add(gallery);  }  public List<Gallery> getAllGalleries() {  return new ArrayList<>(galleries);  }  public Gallery getGalleryById(int id) {  for (Gallery g : galleries) {  if (g.getGalleryID() == id) {  return g;  }  }  return null;  }  public boolean updateGallery(Gallery gallery) {  for (int i = 0; i < galleries.size(); i++) {  if (galleries.get(i).getGalleryID() == gallery.getGalleryID()) {  galleries.set(i, gallery);  return true;  }  }  return false;  }  public boolean removeGallery(int id) {  return galleries.removeIf(g -> g.getGalleryID() == id);  }  public boolean addArtist(Artist artist) {  artist.setArtistId(artistIdCounter++);  return artists.add(artist);  }  public Artist getArtistById(int id) {  for (Artist a : artists) {  if (a.getArtistId() == id) {  return a;  }  }  return null;  }  } |

**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**

|  |
| --- |
| **package** exception;  **public** **class** ArtworkNotFoundException **extends** Exception {  **private** **static** **final** **long** ***serialVersionUID*** = 1L;  **public** ArtworkNotFoundException(String message) {  **super**(message);  }  } |

**2. UserNotFoundException :throw this exception when user enters an invalid id which doesn’t**

**exist in db**

|  |
| --- |
| **package** exception;  **public** **class** UserAuthenticationException **extends** Exception {  **private** **static** **final** **long** ***serialVersionUID*** = 1L;  **public** UserAuthenticationException(String message) {  **super**(message);  }  } |

**9. Main Method**

**Create class named MainModule with main method in main package.**

**Trigger all the methods in service implementation class.**

|  |
| --- |
| package main;  import dao.VirtualArtGalleryImpl;  import entity.Artwork;  import myexceptions.ArtWorkNotFoundException;  import myexceptions.UserNotFoundException;  import util.DBConnection;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.SQLException;  import java.util.List;  import java.util.Scanner;  public class MainModule {  public static void main(String[] args) throws UserNotFoundException, ArtWorkNotFoundException, SQLException, Exception {  Scanner sc = new Scanner(System.in);  VirtualArtGalleryImpl service = new VirtualArtGalleryImpl();  Connection connection = DBConnection.getDbConnection();    try {  String artistInsert = "INSERT INTO Artist (Name, Biography, BirthDate, Nationality, Website, ContactInformation) VALUES (?, ?, ?, ?, ?, ?)";  PreparedStatement artistPs = connection.prepareStatement(artistInsert);  artistPs.setString(1, "Vincent van Gogh");  artistPs.setString(2, "Dutch post-impressionist painter known for Starry Night.");  artistPs.setString(3, "1853-03-30");  artistPs.setString(4, "Dutch");  artistPs.setString(5, "https://vangoghgallery.com");  artistPs.setString(6, "vincent@artmail.com");  artistPs.executeUpdate();  artistPs.setString(1, "Leonardo da Vinci");  artistPs.setString(2, "Italian Renaissance artist known for Mona Lisa and The Last Supper.");  artistPs.setString(3, "1452-04-15");  artistPs.setString(4, "Italian");  artistPs.setString(5, "https://davinciart.com");  artistPs.setString(6, "leo@renaissance.com");  artistPs.executeUpdate();  String artworkInsert = "INSERT INTO Artwork (Title, Description, CreationDate, Medium, ImageURL, ArtistID) VALUES (?, ?, ?, ?, ?, ?)";  PreparedStatement artworkPs = connection.prepareStatement(artworkInsert);  artworkPs.setString(1, "Starry Night");  artworkPs.setString(2, "One of Van Gogh's most famous works, painted in 1889.");  artworkPs.setString(3, "1889-06-01");  artworkPs.setString(4, "Oil on canvas");  artworkPs.setString(5, "https://example.com/starrynight.jpg");  artworkPs.setInt(6, 1);  artworkPs.executeUpdate();  artworkPs.setString(1, "Mona Lisa");  artworkPs.setString(2, "A portrait painting by Leonardo da Vinci.");  artworkPs.setString(3, "1503-01-01");  artworkPs.setString(4, "Oil on poplar panel");  artworkPs.setString(5, "https://example.com/monalisa.jpg");  artworkPs.setInt(6, 2);  artworkPs.executeUpdate();  System.out.println("Sample artists and artworks inserted successfully!");  } catch (SQLException e) {  System.out.println("Data seeding skipped or failed: " + e.getMessage());  }  while (true) {  System.out.println("\n Virtual Art Gallery Menu ");  System.out.println("1. Add Artist");  System.out.println("2. Add Artwork");  System.out.println("3. Update Artwork");  System.out.println("4. Remove Artwork");  System.out.println("5. Get Artwork by ID");  System.out.println("6. Search Artworks");  System.out.println("7. Add Artwork to Favorites");  System.out.println("8. Remove Artwork from Favorites");  System.out.println("9. Get User Favorite Artworks");  System.out.println("10. Exit");  System.out.print("Enter your choice: ");  int choice = Integer.parseInt(sc.nextLine());  try {  switch (choice) {  case 1:  System.out.print("Enter Artist Name: ");  String name = sc.nextLine();  System.out.print("Enter Biography: ");  String bio = sc.nextLine();  System.out.print("Enter Birth Date (YYYY-MM-DD): ");  String dob = sc.nextLine();  System.out.print("Enter Nationality: ");  String nationality = sc.nextLine();  System.out.print("Enter Website: ");  String website = sc.nextLine();  System.out.print("Enter Contact Information: ");  String contact = sc.nextLine();  PreparedStatement artistStmt = connection.prepareStatement(  "INSERT INTO Artist (Name, Biography, BirthDate, Nationality, Website, ContactInformation) VALUES (?, ?, ?, ?, ?, ?)");  artistStmt.setString(1, name);  artistStmt.setString(2, bio);  artistStmt.setString(3, dob);  artistStmt.setString(4, nationality);  artistStmt.setString(5, website);  artistStmt.setString(6, contact);  int rows = artistStmt.executeUpdate();  System.out.println(rows > 0 ? " Artist added!" : " Failed to add artist.");  break;  case 2:  Artwork a1 = new Artwork();  System.out.print("Enter Title: ");  a1.setTitle(sc.nextLine());  System.out.print("Enter Description: ");  a1.setDescription(sc.nextLine());  System.out.print("Enter Creation Date (YYYY-MM-DD): ");  a1.setCreationDate(sc.nextLine());  System.out.print("Enter Medium: ");  a1.setMedium(sc.nextLine());  System.out.print("Enter Image URL: ");  a1.setImageUrl(sc.nextLine());  System.out.print("Enter Artist ID: ");  a1.setArtistId(Integer.parseInt(sc.nextLine()));  System.out.println(service.addArtwork(a1) ? " Artwork added!" : " Failed to add artwork.");  break;  case 3:  Artwork a2 = new Artwork();  System.out.print("Enter Artwork ID to update: ");  a2.setArtworkId(Integer.parseInt(sc.nextLine()));  System.out.print("Enter Title: ");  a2.setTitle(sc.nextLine());  System.out.print("Enter Description: ");  a2.setDescription(sc.nextLine());  System.out.print("Enter Creation Date (YYYY-MM-DD): ");  a2.setCreationDate(sc.nextLine());  System.out.print("Enter Medium: ");  a2.setMedium(sc.nextLine());  System.out.print("Enter Image URL: ");  a2.setImageUrl(sc.nextLine());  System.out.print("Enter Artist ID: ");  a2.setArtistId(Integer.parseInt(sc.nextLine()));  System.out.println(service.updateArtwork(a2) ? "Artwork updated!" : " Update failed.");  break;  case 4:  System.out.print("Enter Artwork ID to remove: ");  int rid = Integer.parseInt(sc.nextLine());  System.out.println(service.removeArtwork(rid) ? "Artwork removed!" : " Failed to remove.");  break;  case 5:  System.out.print("Enter Artwork ID: ");  int artworkId = Integer.parseInt(sc.nextLine());  Artwork art = service.getArtworkById(artworkId);  if (art == null) {  throw new ArtWorkNotFoundException("Invalid Artwork ID: " + artworkId);  }  System.out.println(" Artwork Found:\nTitle: " + art.getTitle() + "\nMedium: " + art.getMedium());  break;  case 6:  System.out.print("Enter search keyword: ");  String keyword = sc.nextLine();  List<Artwork> found = service.searchArtworks(keyword);  System.out.println("Found " + found.size() + " artworks:");  for (Artwork aw : found) {  System.out.println("- " + aw.getTitle() + " [" + aw.getMedium() + "]");  }  break;  case 7:  System.out.print("Enter User ID: ");  int uid1 = Integer.parseInt(sc.nextLine());  if (!userExists(connection, uid1)) {  throw new UserNotFoundException("Invalid User ID: " + uid1);  }  System.out.print("Enter Artwork ID: ");  int aid1 = Integer.parseInt(sc.nextLine());  System.out.println(service.addArtworkToFavorite(uid1, aid1) ? " Added to favorites!" : "Failed.");  break;  case 8:  System.out.print("Enter User ID: ");  int uid2 = Integer.parseInt(sc.nextLine());  System.out.print("Enter Artwork ID: ");  int aid2 = Integer.parseInt(sc.nextLine());  System.out.println(service.removeArtworkFromFavorite(uid2, aid2) ? "Removed from favorites!" : " Failed.");  break;  case 9:  System.out.print("Enter User ID: ");  int uid3 = Integer.parseInt(sc.nextLine());  List<Artwork> favs = service.getUserFavoriteArtworks(uid3);  if (favs == null || favs.isEmpty()) {  System.out.println("No favorites found.");  } else {  System.out.println("Favorite Artworks:");  for (Artwork fav : favs) {  System.out.println("- " + fav.getTitle() + " (" + fav.getMedium() + ")");  }  }  break;  case 10:  System.out.println(" Exiting Virtual Art Gallery. Have a nice day!");  sc.close();  System.exit(0);  default:  System.out.println("Invalid choice. Try again.");  }  } catch (Exception e) {  throw new RuntimeException(e);  }  }  }  private static boolean userExists(Connection connection, int uid1) {  String query = "SELECT COUNT(\*) FROM User WHERE UserID = ?";  **try** (PreparedStatement stmt = connection.prepareStatement(query)) {  stmt.setInt(1, userId);  **try** (ResultSet rs = stmt.executeQuery()) {  **if** (rs.next()) {  **return** rs.getInt(1) > 0;  }  }  } **catch** (SQLException e) {  e.printStackTrace();  }  return false;  }  } |

**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.**

|  |
| --- |
| package test;  import dao.VirtualArtGalleryImpl;  import entity.Artist;  import entity.Artwork;  import org.junit.jupiter.api.\*;  import java.util.List;  import static org.junit.jupiter.api.Assertions.\*;  public class ArtworkServiceTest {  static VirtualArtGalleryImpl service;  static int testArtistId;  @BeforeAll  public static void setup() {  service = new VirtualArtGalleryImpl();    Artist artist = new Artist();  artist.setName("Test Artist");  service.addArtist(artist);  testArtistId = artist.getArtistId();  }  @Test  @DisplayName("a. Test the ability to upload a new artwork")  public void testAddArtwork() {  Artwork art = new Artwork();  art.setTitle("Test Upload");  art.setDescription("Uploaded artwork");  art.setCreationDate("2024-01-01");  art.setMedium("Oil");  art.setImageUrl("http://image.url");  art.setArtistId(testArtistId);  boolean result = service.addArtwork(art);  assertTrue(result, "Artwork should be uploaded successfully");  }  @Test  @DisplayName("b. Verify that updating artwork details works correctly")  public void testUpdateArtwork() {  Artwork art = new Artwork();  art.setTitle("Old Title");  art.setDescription("Old Description");  art.setCreationDate("2023-01-01");  art.setMedium("Watercolor");  art.setImageUrl("http://old.image");  art.setArtistId(testArtistId);  service.addArtwork(art);  Artwork existing = service.searchArtworks("Old Title").get(0);  existing.setTitle("Updated Title");  existing.setDescription("Updated Description");  boolean updated = service.updateArtwork(existing);  assertTrue(updated, "Artwork should be updated");  Artwork result = service.searchArtworks("Updated Title").get(0);  assertEquals("Updated Description", result.getDescription());  }  @Test  @DisplayName("c. Test removing an artwork from the gallery")  public void testRemoveArtwork() {  Artwork art = new Artwork();  art.setTitle("To Remove");  art.setDescription("Temporary");  art.setCreationDate("2022-10-10");  art.setMedium("Digital");  art.setImageUrl("http://remove.url");  art.setArtistId(testArtistId);  service.addArtwork(art);  int id = service.searchArtworks("To Remove").get(0).getArtworkId();  boolean removed = service.removeArtwork(id);  assertTrue(removed, "Artwork should be removed");  assertTrue(service.searchArtworks("To Remove").isEmpty(), "Artwork should no longer exist");  }  @Test  @DisplayName("d. Check if searching for artworks returns the expected results")  public void testSearchArtwork() {  Artwork art = new Artwork();  art.setTitle("Galaxy Art");  art.setDescription("Stars and nebulae");  art.setCreationDate("2023-05-05");  art.setMedium("Acrylic");  art.setImageUrl("http://galaxy.art");  art.setArtistId(testArtistId);  service.addArtwork(art);  List<Artwork> results = service.searchArtworks("Galaxy");  assertFalse(results.isEmpty(), "Artwork should be found");  assertEquals("Galaxy Art", results.get(0).getTitle());  }  } |

**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.**

|  |
| --- |
| package test;  import static org.junit.jupiter.api.Assertions.\*;  import java.util.List;  import org.junit.jupiter.api.BeforeEach;  import org.junit.jupiter.api.Test;  import dao.VirtualArtGalleryImpl;  import entity.Gallery;  public class GalleryServiceTest {  private VirtualArtGalleryImpl service;  @BeforeEach  public void setup() {  service = new VirtualArtGalleryImpl();  }  // a. Test creating a new gallery  @Test  public void testAddGallery() {  Gallery gallery = new Gallery();  gallery.setName("Nature Gallery");  gallery.setLocation("Mumbai");  gallery.setDescription("Gallery of Nature Art");  boolean added = service.addGallery(gallery);  assertTrue(added, "Gallery should be added successfully");  List<Gallery> galleries = service.getAllGalleries();  assertEquals(1, galleries.size(), "There should be one gallery in the system");  assertEquals("Nature Gallery", galleries.get(0).getName());  }  // b. Verify updating gallery information  @Test  public void testUpdateGallery() {  Gallery gallery = new Gallery();  gallery.setName("Abstract Art");  gallery.setLocation("Delhi");  gallery.setDescription("Abstract Collection");  service.addGallery(gallery);  Gallery added = service.getAllGalleries().get(0);  added.setName("Modern Abstract Art");  boolean updated = service.updateGallery(added);  assertTrue(updated, "Gallery should be updated");  assertEquals("Modern Abstract Art", service.getAllGalleries().get(0).getName());  }  // c. Test removing a gallery  @Test  public void testRemoveGallery() {  Gallery gallery = new Gallery();  gallery.setName("Classic Gallery");  gallery.setLocation("Chennai");  gallery.setDescription("Old Masterpieces");  service.addGallery(gallery);  Gallery added = service.getAllGalleries().get(0);  boolean removed = service.removeGallery(added.getGalleryID());  assertTrue(removed, "Gallery should be removed successfully");  assertTrue(service.getAllGalleries().isEmpty(), "Galleries list should be empty");  }  // d. Check search functionality  @Test  public void testSearchGalleryById() {  Gallery gallery = new Gallery();  gallery.setName("Sci-Fi Art");  gallery.setLocation("Pune");  gallery.setDescription("Space and Future");  service.addGallery(gallery);  int id = service.getAllGalleries().get(0).getGalleryID();  Gallery found = service.getGalleryById(id);  assertNotNull(found, "Gallery should be found");  assertEquals("Sci-Fi Art", found.getName());  }  } |

**Output:**

|  |
| --- |
| Sample artists and artworks inserted successfully!  Virtual Art Gallery Menu  1. Add Artist  2. Add Artwork  3. Update Artwork  4. Remove Artwork  5. Get Artwork by ID  6. Search Artworks  7. Add Artwork to Favorites  8. Remove Artwork from Favorites  9. Get User Favorite Artworks  10. Exit  Enter your choice: 1  Enter Artist Name: New Artist  Enter Biography: Contemporary painter  Enter Birth Date (YYYY-MM-DD): 1990-04-11  Enter Nationality: Canadian  Enter Website: https://newartist.com  Enter Contact Information: Info: contact@newartist.com  Artist added!  Virtual Art Gallery Menu  1. Add Artist  2. Add Artwork  3. Update Artwork  4. Remove Artwork  5. Get Artwork by ID  6. Search Artworks  7. Add Artwork to Favorites  8. Remove Artwork from Favorites  9. Get User Favorite Artworks  10. Exit  Enter your choice: 2  Enter Title: Ocean Breeze  Enter Description: Calming ocean view  Enter Creation Date (YYYY-MM-DD): 2022-06-15  Enter Medium: Acrylic on Canvas  Enter Image URL: ocean\_breeze.jpg  Enter Artist ID: 1  Artwork added!  Virtual Art Gallery Menu  1. Add Artist  2. Add Artwork  3. Update Artwork  4. Remove Artwork  5. Get Artwork by ID  6. Search Artworks  7. Add Artwork to Favorites  8. Remove Artwork from Favorites  9. Get User Favorite Artworks  10. Exit  Enter your choice: 3  Enter Artwork ID to update: 1  Enter Title: Starry Night Updated  Enter Description: Modified description  Enter Creation Date (YYYY-MM-DD): 1889-06-01  Enter Medium: Oil on Canvas  Enter Image URL: starry\_night\_updated.jpg  Enter Artist ID: 1  Artwork updated!  Virtual Art Gallery Menu  1. Add Artist  2. Add Artwork  3. Update Artwork  4. Remove Artwork  5. Get Artwork by ID  6. Search Artworks  7. Add Artwork to Favorites  8. Remove Artwork from Favorites  9. Get User Favorite Artworks  10. Exit  Enter your choice: 4  Enter Artwork ID to remove: 3  Artwork removed!  Virtual Art Gallery Menu  1. Add Artist  2. Add Artwork  3. Update Artwork  4. Remove Artwork  5. Get Artwork by ID  6. Search Artworks  7. Add Artwork to Favorites  8. Remove Artwork from Favorites  9. Get User Favorite Artworks  10. Exit  Enter your choice: 5  Enter Artwork ID: 1  Artwork Found:  Title: Starry Night Updated  Medium: Oil on Canvas  Virtual Art Gallery Menu  1. Add Artist  2. Add Artwork  3. Update Artwork  4. Remove Artwork  5. Get Artwork by ID  6. Search Artworks  7. Add Artwork to Favorites  8. Remove Artwork from Favorites  9. Get User Favorite Artworks  10. Exit  Enter your choice: 6  Enter search keyword: ocean  Found 1 artworks:  - Ocean Breeze [Acrylic on Canvas]  Virtual Art Gallery Menu  1. Add Artist  2. Add Artwork  3. Update Artwork  4. Remove Artwork  5. Get Artwork by ID  6. Search Artworks  7. Add Artwork to Favorites  8. Remove Artwork from Favorites  9. Get User Favorite Artworks  10. Exit  Enter your choice: 7  Enter User ID: 5  Enter Artwork ID: 5  Added to favorites!  Virtual Art Gallery Menu  1. Add Artist  2. Add Artwork  3. Update Artwork  4. Remove Artwork  5. Get Artwork by ID  6. Search Artworks  7. Add Artwork to Favorites  8. Remove Artwork from Favorites  9. Get User Favorite Artworks  10. Exit  Enter your choice: 8  Enter User ID: 5  Enter Artwork ID: 5  Removed from favorites!  Virtual Art Gallery Menu  1. Add Artist  2. Add Artwork  3. Update Artwork  4. Remove Artwork  5. Get Artwork by ID  6. Search Artworks  7. Add Artwork to Favorites  8. Remove Artwork from Favorites  9. Get User Favorite Artworks  10. Exit  Enter your choice: 9  Enter User ID: 1  Favorite Artworks:  - Starry Night Updated (Oil on Canvas)  Virtual Art Gallery Menu  1. Add Artist  2. Add Artwork  3. Update Artwork  4. Remove Artwork  5. Get Artwork by ID  6. Search Artworks  7. Add Artwork to Favorites  8. Remove Artwork from Favorites  9. Get User Favorite Artworks  10. Exit  Enter your choice: 10  Exiting Virtual Art Gallery. Have a nice day! |

**Exception Handling:**

|  |
| --- |
| Sample artists and artworks inserted successfully!  Virtual Art Gallery Menu  1. Add Artist  2. Add Artwork  3. Update Artwork  4. Remove Artwork  5. Get Artwork by ID  6. Search Artworks  7. Add Artwork to Favorites  8. Remove Artwork from Favorites  9. Get User Favorite Artworks  10. Exit  Enter your choice: 5  Enter Artwork ID: 100  Exception in thread "main" java.lang.RuntimeException: myexceptions.ArtWorkNotFoundException: Invalid Artwork ID: 100  at main.MainModule.main(MainModule.java:218)  Caused by: myexceptions.ArtWorkNotFoundException: Invalid Artwork ID: 100  at main.MainModule.main(MainModule.java:159) |

|  |
| --- |
| Sample artists and artworks inserted successfully!  Virtual Art Gallery Menu  1. Add Artist  2. Add Artwork  3. Update Artwork  4. Remove Artwork  5. Get Artwork by ID  6. Search Artworks  7. Add Artwork to Favorites  8. Remove Artwork from Favorites  9. Get User Favorite Artworks  10. Exit  Enter your choice: 7  Enter User ID: 7  Exception in thread "main" java.lang.RuntimeException: myexceptions.UserNotFoundException: Invalid User ID: 7  at main.MainModule.main(MainModule.java:219)  Caused by: myexceptions.UserNotFoundException: Invalid User ID: 7  at main.MainModule.main(MainModule.java:181) |