MUSEUM ART MANAGEMENT SYSTEM

A MINI-PROJECT REPORT

Submitted by

BALAJI S	**240701069**
SAI SARAN	**240701455**

in partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2025

BONAFIDE CERTIFICATE

Certified that this project **“MUSEUM ART MANAGEMENT SYSTEM”** is the bonafide work of **“BALAJI S, SAI SARAN”** who carried out the project work under my supervision.

SIGNATURE

MRs . S . VINOTHINI

ASSISTANT PROFESSOR

Dept. of Computer Science and Engg,
Rajalakshmi Engineering College
Chennai

This mini project report is submitted for the viva voce examination to be held on

____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

Art museums and galleries often showcase timeless masterpieces, but few offer a digital platform for users to engage with these works beyond passive viewing. This project introduces a Museum Art Bidding System — a desktop application built using Java Swing and Oracle SQL — that allows users to explore curated artworks, save favourites, submit purchase requests, and provide feedback. The system is designed to streamline both user interaction and administrative oversight, offering a modular and visually intuitive interface.

Users can browse a collection of iconic artworks, request to buy pieces, and track the status of their requests. Administrators, on the other hand, can view all incoming requests, accept or decline them, and monitor user feedback through a dedicated dashboard. The application emphasizes clarity, responsiveness, and maintainability, with a focus on clean UI design and efficient data handling.

By digitizing the artwork request process and enhancing user engagement, this system empowers museums and galleries to modernize their operations and offer a more interactive experience to art enthusiasts.

.

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honorable chairman **MR. S. MEGANATHAN** and the chairperson **DR. M.THANGAM MEGANATHAN** for their timely support and encouragement.

We are greatly indebted to our respected and honorable principal **Dr. S.N. MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by our Head Of The Department **Dr. E.M. MALATHY** and our Deputy Head Of The Department **Dr. J. MANORANJINI** for being ever supporting force during our project work

We also extend our sincere and hearty thanks to our internal guide **Mrs. S. VINOTHINI,** for her valuable guidance and motivation during the completion of this project.

Our sincere thanks to our family members, friends and other staff members of computer science engineering.

1. BALAJI S

2. SAI SARAN

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Museum Art Bidding System is designed to provide users with an interactive platform to explore, request, and engage with iconic artworks showcased by museums or galleries. The system helps users access essential information about each artwork, track their request status, and submit feedback — all through a clean and intuitive interface. It also enables administrators to manage incoming requests efficiently, making the overall experience seamless for both users and curators.

1.2 SCOPE OF THE WORK

This system aims to modernize the way museums and galleries interact with visitors by offering a digital platform for artwork engagement. It allows users to browse curated collections, save favorites, and request to buy specific pieces. For administrators, it provides tools to view, accept, or decline requests and monitor user feedback. The system is built to support modular UI design, maintainable backend logic, and a user-centric experience that can scale with future enhancements like bidding, payment integration, or exhibition scheduling.

1.3 PROBLEM STATEMENT

While museums and galleries hold vast collections of valuable art, most lack a digital interface that allows users to actively engage with these works beyond passive viewing. Existing systems are either too generic or inaccessible to local institutions. This creates a gap in user interaction, request management, and feedback collection. The absence of a streamlined platform limits the ability of museums to offer personalized services, track interest, and compete with more digitally advanced institutions.

1.4 AIM AND OBJECTIVES OF THE PROJECT

The main objective of this project is to build a user-friendly system that allows users to request artworks based on interest and track the status of those requests. The system maintains detailed records of each artwork, including title, artist, description, and request status. It also enables administrators to manage requests and feedback efficiently. By digitizing the artwork request process, the system helps museums and galleries improve user engagement, streamline operations, and stay competitive in a modern, tech-driven cultural landscape.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Component	Specification
Processor	Intel i5 or higher
RAM	8 GB Minimum
Display	1366*768 or higher

2.2 SOFTWARE SPECIFICATIONS

Operating System	:	WINDOWS 10
Front – End	:	Java Swing
Back - End	:	OracleSQL
Programming Language	:	Java(JDK 17+)
Tools Used	:	Visual Studios Code, SQL Developer
Libraries	:	JDBC Connectors, Swing Components

CHAPTER 3

MODULE DESCRIPTION

This application consists of two modules. When the program runs, it will ask for a confirmation to the login window. The person who interacts can login as an Administrator or as a User. The description of the modules are as follows:

1. **Admin login**

   When the person who interacts tries to login as Admin then he needs to login with his password. The administrator only has the power to change and manipulate the data in the database.

2. **User login**

   When the person tries to login as a user then he/she will be prompted to enter the number of symptoms and the final result will be viewed.

3. **Dashboard Module**

   After login, users can view data's and can request them to buy or they can save for later. It includes buttons to navigate request, analyze content, and view request status.
4. **Request Module**

   It shows the status of the data in an unclickable button that changes into accepted, declined or pending when admin changes them respectively.
5. **Database Module**

   Handles Storing:
   - User Credential
   - Uploading arts and details
   - Stores the data

   Uses relational design:

   - users (user_id, username, password, email)
   - artworks (art_id, title, artist, description, image_url)
   - contact_messages (message_id, name, email, message)
   - saved_artworks (username, art_id , title, artist, description, image_url)
   - requested_artworks (username, art_id, title, artist, description, image_url)

6. **History and Report Module**

   In requests tab of the admin, all the previous requests and the new ones are displayed along with the status with detailed data. Also allows admin to change the status of the request.

CHAPTER 4

4.1 Database Connection Code:

```
package db;

import java.sql.*;

public class DBConnection {

    private static final String URL = "jdbc:oracle:thin:@localhost:1521/museum_art";

    private static final String USER = "root";

    private static final String PASS = "";

    public static Connection getConnection() throws SQLException {

        return DriverManager.getConnection(URL, USER, PASS);

    }

}
```

4.2 Login Page:

```
package ui;

import javax.swing.*;

import java.awt.*;

public class LoginPage extends JFrame {

    public LoginPage() {

        setTitle("Login");

        setSize(300, 200);

        setLayout(new GridLayout(3, 1));

        JButton userBtn = new JButton("User Login");
        JButton signupBtn = new JButton("User Signup");
        JButton adminBtn = new JButton("Admin Login");
        userBtn.addActionListener(e -> new UserLoginPage());
        signupBtn.addActionListener(e -> new SignupPage());
        adminBtn.addActionListener(e -> new AdminLoginPage());
        add(userBtn);
        add(signupBtn);
        add(adminBtn);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setVisible(true);
    }}
```

4.3 User Authentication:

```
 package db;
import java.sql.*;
import java.util.*;
public class UserDAO {
    private Connection conn;
    public UserDAO() throws Exception {
        conn = DriverManager.getConnection(
            "jdbc:oracle:thin:@localhost:1521/museum_art",
            "root",
            """
        );
    }
    public void registerUser(String username, String email, String password) throws
Exception {
        PreparedStatement ps = conn.prepareStatement( "INSERT INTO users
(username, email, password) VALUES (?, ?, ?)" );
        ps.setString(1, username);
        ps.setString(2, email);
        ps.setString(3, password);
        ps.executeUpdate();
    }
    public boolean validateLogin(String username, String password) throws
Exception {
        PreparedStatement ps = conn.prepareStatement( "SELECT * FROM users
WHERE username = ? AND password = ?" );
        ps.setString(1, username);
        ps.setString(2, password);
        ResultSet rs = ps.executeQuery();
        return rs.next(); // true if match found
    }
    public List<String[]> getAllUsers() throws Exception {
        List<String[]> users = new ArrayList<>();
        PreparedStatement ps = conn.prepareStatement( "SELECT username, email
FROM users" );
```

```
        ResultSet rs = ps.executeQuery();
        while (rs.next()) {
            String[] user = new String[2];
            user[0] = rs.getString("username");
            user[1] = rs.getString("email");
            users.add(user);
        }
        return users;
    }
    public boolean userExists(String username) throws Exception {
        PreparedStatement ps = conn.prepareStatement(
            "SELECT username FROM users WHERE username = ?"
        );
        ps.setString(1, username);
        ResultSet rs = ps.executeQuery();
        return rs.next();
    }
}
```

4.4 Admin Dashboard:

```
package ui;
import java.awt.*;
import javax.swing.*;
```

```
public class AdminDashboard extends JFrame {

    public AdminDashboard() {

        setTitle("Admin Dashboard");

        setSize(400, 300);

        setLayout(new BorderLayout());

        JLabel welcome = new JLabel("Welcome, Admin!",
SwingConstants.CENTER);

        welcome.setFont(new Font("SansSerif", Font.BOLD, 18));

        add(welcome, BorderLayout.NORTH);

        JPanel options = new JPanel();

        options.setLayout(new GridLayout(0, 1, 10, 10));

        options.setBorder(BorderFactory.createEmptyBorder(20, 40, 20, 40));

        JButton requestsBtn = new JButton("Requests");

        JButton feedbacksBtn = new JButton("Feedbacks");

        JButton logoutBtn = new JButton("Logout");

        requestsBtn.addActionListener(e -> {

            dispose();

            new AdminRequestPage();

        });

        options.add(requestsBtn);

        options.add(feedbacksBtn);

        options.add(logoutBtn);

        add(options, BorderLayout.CENTER);
        feedbacksBtn.addActionListener(e -> {
            JOptionPane.showMessageDialog(this, "Feedbacks panel coming soon!");
        });
        logoutBtn.addActionListener(e -> {
            dispose();
            new LoginPage();
        });
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setVisible(true);
    }
}
```

4.5 Admin Request Page:

```
package ui;
import db.RequestDAO;
import java.awt.*;
import java.util.List;
import javax.swing.*;
import model.Artwork;
public class AdminRequestPage extends JFrame {
JPanel mainPanel = new JPanel();
 public AdminRequestPage() {
```

```
setTitle("Artwork Requests");
setSize(800, 600);
setLayout(new BorderLayout());
mainPanel.setLayout(new BoxLayout(mainPanel, BoxLayout.Y_AXIS));
 mainPanel.setBackground(Color.WHITE);
 JScrollPane scrollPane = new JScrollPane(mainPanel);
  add(scrollPane, BorderLayout.CENTER);
   JButton backBtn = new JButton("← Back to Admin Dashboard");
    backBtn.addActionListener(e -> {
           dispose();
           new AdminDashboard();});
       JPanel topPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
       topPanel.setBackground(Color.WHITE);
       topPanel.add(backBtn);
       add(topPanel, BorderLayout.NORTH);
       loadRequests();
       setDefaultCloseOperation(EXIT_ON_CLOSE);
       setLocationRelativeTo(null);
       setVisible(true);
   }
   private void loadRequests() {
       mainPanel.removeAll();
```

```
        try {
            RequestDAO dao = new RequestDAO();
            List<Artwork> requests = dao.getAllRequests(); // includes status
            for (Artwork art : requests) {
                mainPanel.add(createRequestCard(art));
                mainPanel.add(Box.createVerticalStrut(10));
            }
        } catch (Exception e) {
            e.printStackTrace();
            JLabel error = new JLabel("Failed to load requests.");
            error.setForeground(Color.RED);
            mainPanel.add(error);
        }
        mainPanel.revalidate();
        mainPanel.repaint();
    }
    private JPanel createRequestCard(Artwork art) {
        JPanel card = new JPanel();
        card.setLayout(new BoxLayout(card, BoxLayout.Y_AXIS));
        card.setBorder(BorderFactory.createLineBorder(Color.GRAY));
        card.setBackground(Color.WHITE);
        card.setPreferredSize(new Dimension(700, 120));
```

```
JLabel title = new JLabel("🎨 " + art.title);
JLabel artist = new JLabel("👤 " + art.artist);
JLabel desc = new JLabel("📝 " + art.description);
JButton statusBtn = new JButton(art.status);
statusBtn.setEnabled(false);
statusBtn.setBackground(getStatusColor(art.status));
statusBtn.setForeground(Color.WHITE);
JButton acceptBtn = new JButton("Accept");
JButton denyBtn = new JButton("Decline");
acceptBtn.addActionListener(e -> {
    new Thread(() -> {
        try {
            new RequestDAO().updateRequestStatus(art.id, "Accepted");
            SwingUtilities.invokeLater(() -> {
                statusBtn.setText("Accepted");
                statusBtn.setBackground(getStatusColor("Accepted"));
            });
        } catch (Exception ex) {
            ex.printStackTrace();
            SwingUtilities.invokeLater(() ->
                JOptionPane.showMessageDialog(null, "Failed to update status."));}
    }).start();
    });
    denyBtn.addActionListener(e -> {
        try {
            new RequestDAO().updateRequestStatus(art.id, "Declined");
            statusBtn.setText("Declined");
            statusBtn.setBackground(getStatusColor("Declined"));
        } catch (Exception ex) {
            ex.printStackTrace();
        }});
    JPanel buttonPanel = new JPanel();
    buttonPanel.add(acceptBtn);
    buttonPanel.add(denyBtn);
    card.add(title);
    card.add(artist);
    card.add(desc);
    card.add(statusBtn);
    card.add(buttonPanel);
    return card; }
private Color getStatusColor(String status) {
    switch (status.toLowerCase()) {
        case "accepted": return new Color(0, 128, 0);
        case "declined": return new Color(200, 0, 0);
    default: return new Color(100, 100, 100);

}}}
```

CHAPTER 5

SCREEN SHOTS

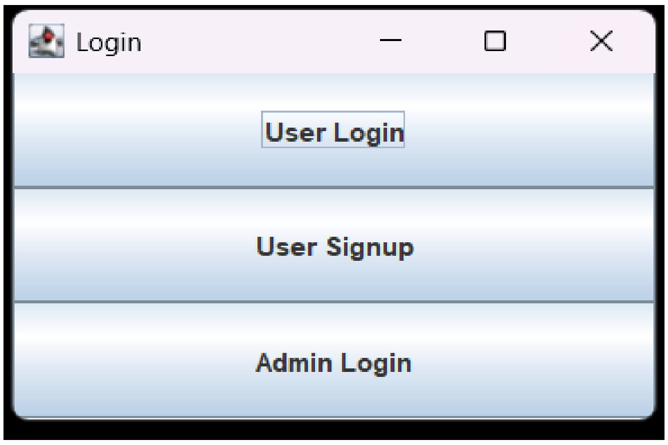
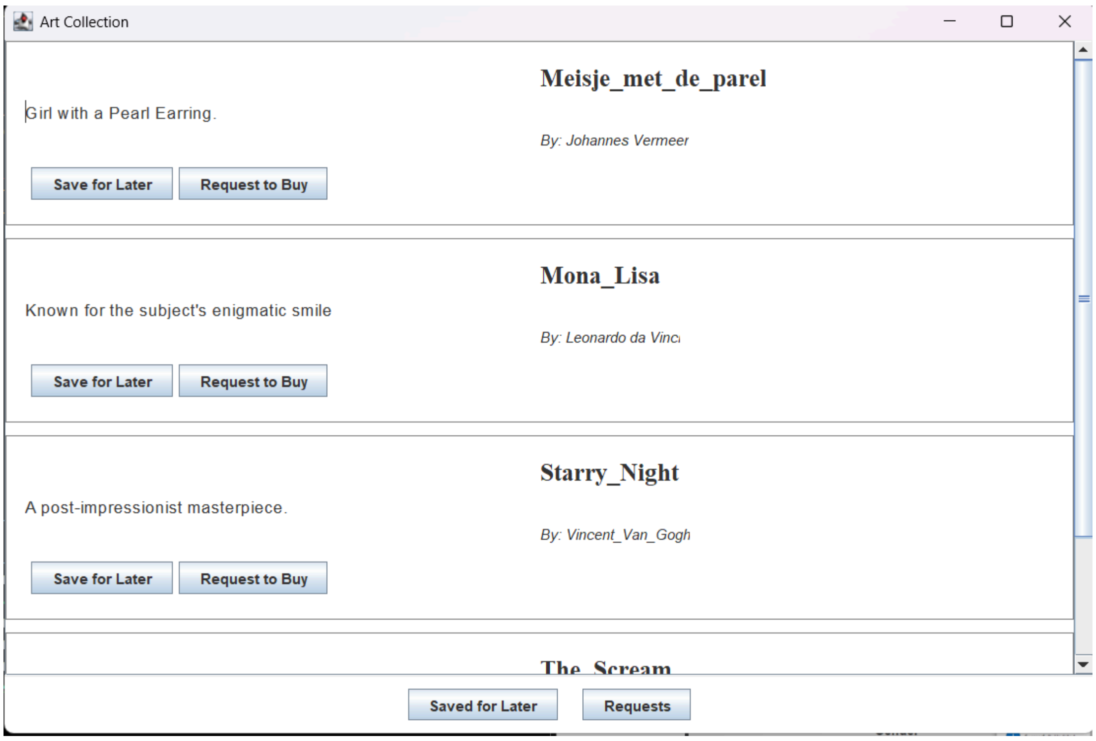


Fig 5.1 Login Page



Meisje_met_de_parel

Girl with a Pearl Earring.

By: Johannes Vermeer

Mona_Lisa

Known for the subject's enigmatic smile

By: Leonardo da Vinci

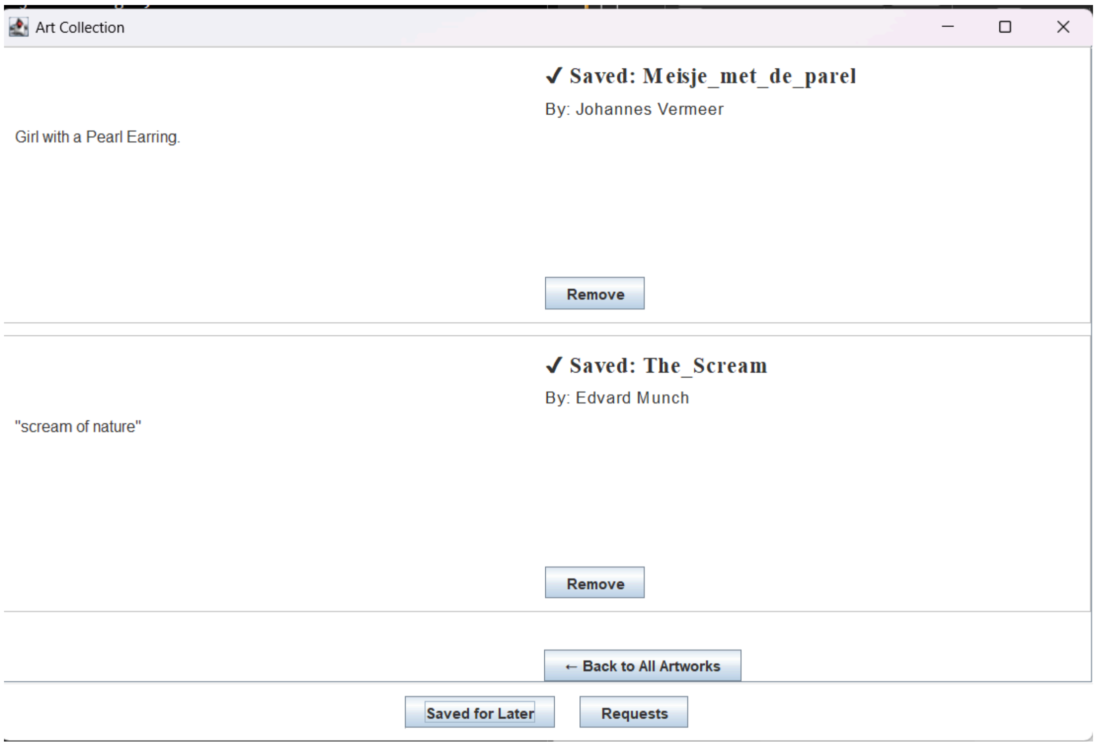
Starry_Night

A post-impressionist masterpiece.

By: Vincent_Van_Gogh

The_Scream

Fig 5.2 User Home Page


Art Collection

✓ Saved: Meisje_met_de_parel

By: Johannes Vermeer

Girl with a Pearl Earring.

Remove

✓ Saved: The_Scream

By: Edvard Munch

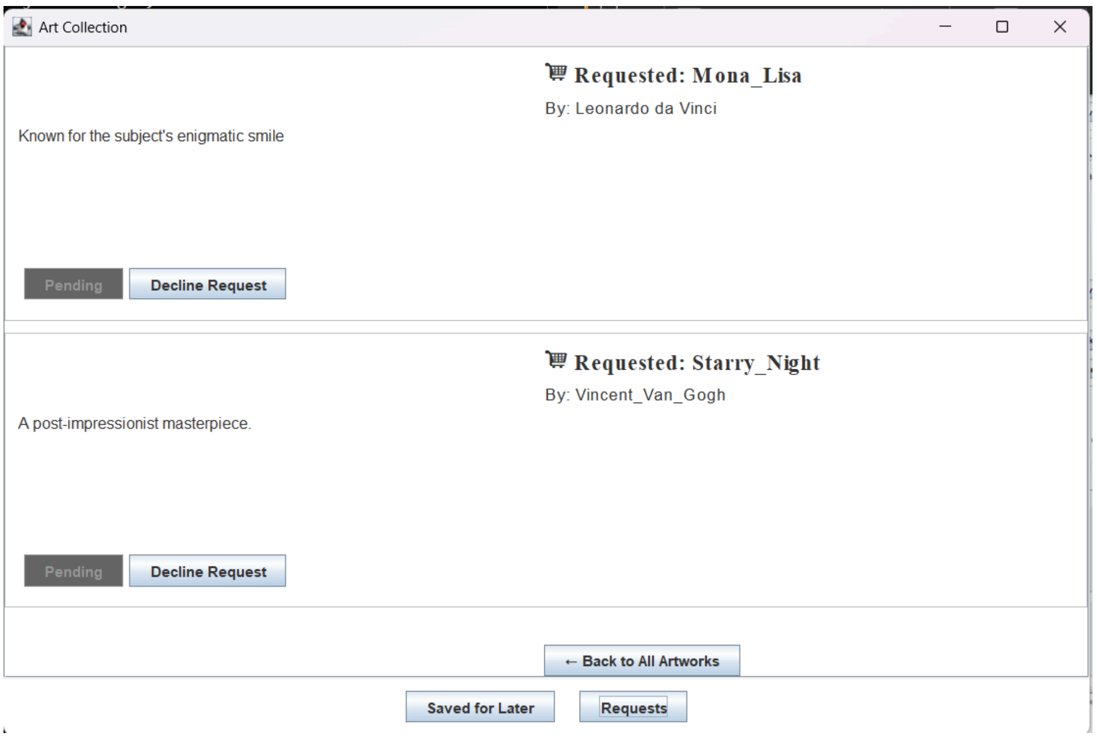
"scream of nature"

Remove

← Back to All Artworks

Saved for Later   Requests

Fig 5.3 Saved For Later Page


Art Collection

🛒 Requested: Mona_Lisa

By: Leonardo da Vinci

Known for the subject's enigmatic smile

Pending   Decline Request

🛒 Requested: Starry_Night

By: Vincent_Van_Gogh

A post-impressionist masterpiece.

Pending   Decline Request

← Back to All Artworks

Saved for Later   Requests

Fig 5.4 Request Given Page

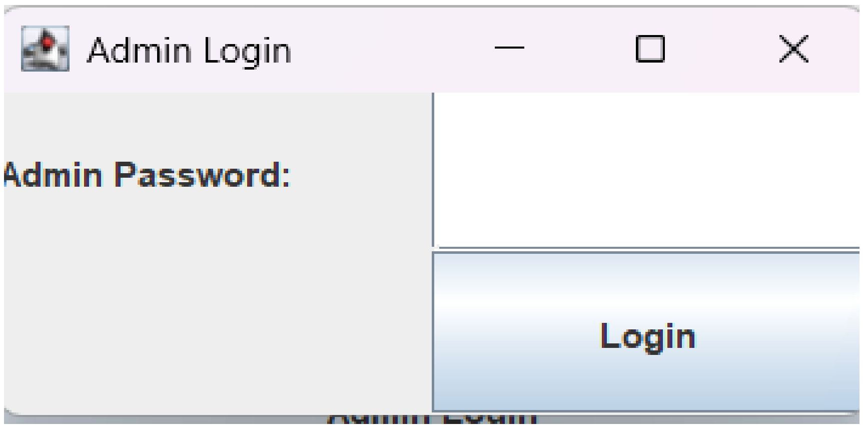


Fig 5.5 Admin Login

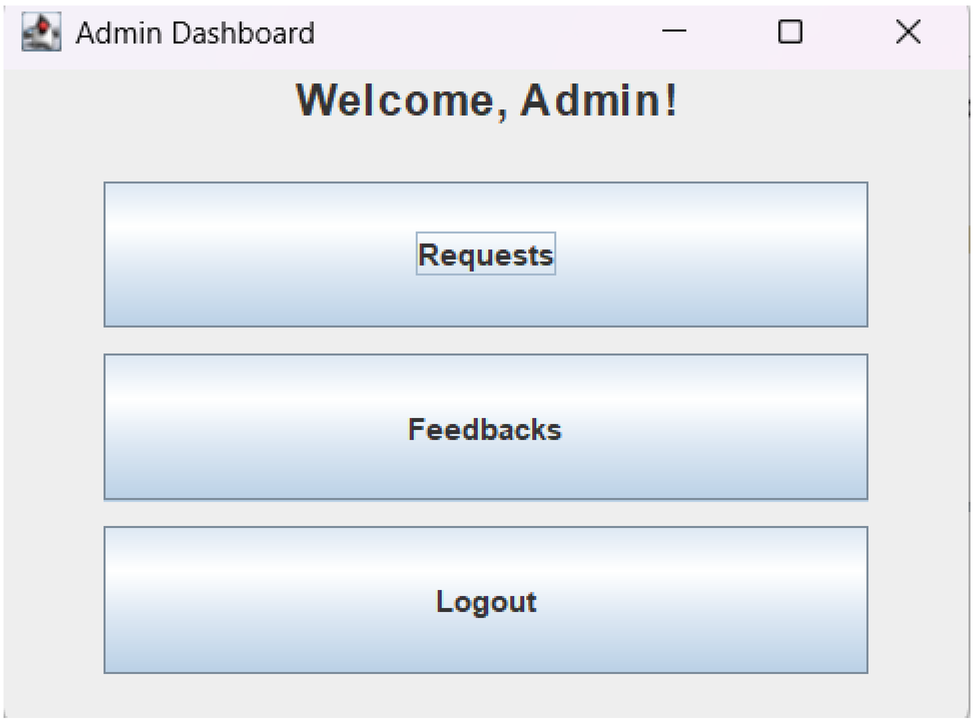


Fig 5.6 Admin Home Page

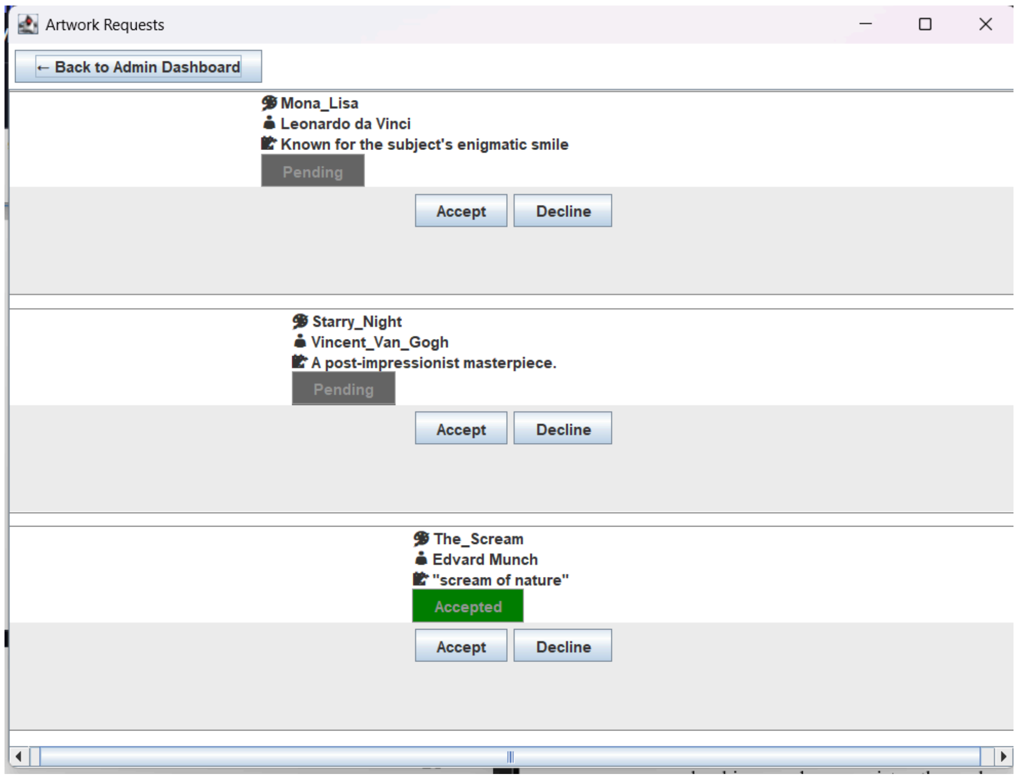


Fig 5.7 Admin Request Page

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

The Museum Art Bidding System provides a simple and interactive way for users to explore artworks, request purchases, and share feedback. It helps museums manage requests efficiently and improves user engagement through a clean and organized interface.

Future Enhancement:

- Add bidding and payment features
- Enable user notifications for request updates
- Include search and filter options for artworks
- Build a mobile-friendly version for wider access

REFERENCES

1. https://docs.oracle.com/javase/8/docs/api/javax/swing/package summary.html

2. https://docs.oracle.com/javase/tutorial/jdbc/index.html

3. https://dev.mysql.com/doc/refman/8.0/en/

4. https://dev.mysql.com/doc/connector-j/8.0/en/

5. https://www.oracle.com/java/technologies/data-access-object.html