Project Report

Art Sale Management System

USING

PHP & MYSQL

NAME: PRERNA ANAND

REGISTRATION NO.: 12218114

SUBJECT CODE: CAP 512 (OPEN-SOURCE WEB APPLICATION DEVELOPMENT)

TOPIC: Art Sale Management System

SUBMITTED TO: MR. SAHIL SIR

CONTENTS

1. OVERVIEW OF ORGANIZATION
2. PREFACE
3. INTRODUCTION
4. NEED OF ONLINE SHOPPING PORTAL
5. PROFILE OF THE PROBLEM
6. STRUCTURE OF THE PROJECT
7. SOFTWARE DEVELOPMENT LIFE CYCLE
8. PROBLEM ANALYSIS
9. PROJECT PLAN
10. AVTIVITIES DURING SOFTWARE PROJECT PLANNING
11. HARDWARE AND SOFTWARE REQUIREMENT
12. FRONT-END DETAILS
13. BACK-END DETAILS
14. SYSTEM DESIGN
15. DESIGN NOTATIONS
16. PRODUCT FUNCTION
17. DETAILED DESIGN
18. E-R DIAGRAMS

19. DATA FLOW DIAGRAMS

20. TESTING

21. METHODOLOGY USED FOR TESTING

22. IMPLEMENTATION

23. A DATABASE DRIVEN ONLINE SHOPPING PORTAL

24. ABOUT THE CURRENT SYSTEM

25. ADVANTAGES

26. DEFFICIENCY OF MANUAL SYSTEM

27. GOALS OF PROPOSED SYSTEM

28. USER REQUIREMENTS

29. LIMITATIONS

30. PROJECT LEGACY

31. USER MANUAL

32. ABBREVIATIONS/ ACRONYMS

33. BIBLOGRAPHY

34. SCREEN-SHORTS OF THE PROJECT

35. DATABASE INTERGRATION

1. Overview of Organization

This project built the main website for **"The Art Gallery"**. Think of The Art Gallery as an online place created to help artists connect with people who want to buy art.

What The Art Gallery Does:
The main goal is to make buying and selling art easier using the internet. The gallery shows and sells different kinds of art, like paintings, sculptures, and digital creations. It wants everyone to be able to easily see and buy art they like. It also gives artists, whether they are new or already known, a good place to show their work to people all over the world.
How It Works Online:
The Art Gallery works mainly through its website. This means it doesn't have the limits of a physical shop, like opening hours or a specific location. People can visit the website anytime, from anywhere. This project built that website, which is the heart of The Art Gallery's online operations.
Why This Project is Important:
The website created in this project is how The Art Gallery does its business online. It's where people can sign up or log in, look at art, learn about artists, put art in a shopping cart, go through a practice checkout, and contact the gallery (artists might also submit their artwork here for consideration). This website makes the idea of The Art Gallery real and usable online.

2. Preface

This document serves as the official report for the **Art Sale Management System** project. Its main purpose is to provide a clear and detailed explanation of the entire project, from the initial idea to the final implementation. We aim to describe the goals we set out to achieve, the problems this system solves, the steps taken during its development, and the technologies used to build it.

The core idea behind this project was to create a functional and user-friendly online platform for "The Art Gallery". In today's world, having an online presence is essential, especially for connecting artists with a wider audience. This system provides the necessary tools for users to explore art, manage their selections through a shopping cart, and interact with the gallery, while also ensuring secure access through user registration and login.

This report walks through the different stages of the project, including planning, design, coding, and testing. It details the specific features implemented, such as the gallery display, the client-side shopping cart, the user authentication system using PHP and MySQL, and the contact submission viewer. We also discuss the technical requirements, the overall system architecture, and potential areas for future improvement. This document is intended for anyone interested in understanding the project's scope, development process, and final outcome, including instructors, evaluators, or future developers who might build upon this work.

3. Introduction

Welcome to the introduction for the **Art Sale Management System**. This project is all about creating a special kind of website – an online art gallery. In a world that's becoming more and more digital, the way people find, appreciate, and buy art is also changing. This project tackles that change by building a website where artists can show their work and art lovers can browse and purchase pieces they like, all online.

The main problem this website aims to solve is connecting artists directly with potential buyers, no matter where they are located. Traditional art galleries are great, but they can be limited by location and opening times. An online gallery, like the one built here, is open 24/7 and accessible from anywhere with an internet connection. This makes it easier for people to discover new art and for artists to reach a much larger audience than they might otherwise.

To make this happen, the website includes several important parts. There's a secure system for users to create accounts and log in, ensuring only registered users can access certain features. A gallery section displays various artworks, allowing users to browse different pieces. For those interested in buying, there's an easy-to-use shopping cart where they can add items, change quantities, or remove pieces. The site also includes a simulated checkout process to show how a purchase might work. Finally, there's a way for users to contact the gallery, and these messages are stored using a database (MySQL) managed by PHP scripts on the server. This project combines visual design (HTML, CSS) with interactive elements (JavaScript) and server-side power (PHP, MySQL) to create a foundational platform for online art sales and interaction.

4. Need for an Online Shopping Portal

Creating an online shopping portal, specifically for an art gallery, is more important today than ever before. While physical galleries have their charm, an online platform like the one developed in this project offers significant advantages that meet the needs of modern artists and buyers.

- **Reaching More People:** A physical gallery is limited to visitors who can travel to its location. An online portal breaks down these walls. The Art Gallery website can

be visited by anyone with internet access, anywhere in the world. This means artists showing their work can reach a vastly larger audience, increasing their chances of making sales and gaining recognition far beyond their local area.
- **Always Available:** Unlike a physical shop with set opening and closing times, the online gallery is open 24 hours a day, 7 days a week. People can browse art, learn about artists, and even make purchase decisions late at night, early in the morning, or whenever it suits them best. This constant availability is incredibly convenient for busy potential buyers.
- **Shopping from Home:** The convenience factor is huge. Users don't need to travel or dedicate specific time slots to visit. They can explore the entire collection from the comfort of their own homes, using their computers or mobile devices. They can take their time, revisit favourite pieces, and easily compare different artworks without pressure.
- **More Opportunities for Artists:** Physical galleries have limited wall space. An online portal can showcase a much larger number of artworks and artists simultaneously. This gives more creators, especially those just starting, a valuable opportunity to display and sell their work where they might not have had a chance otherwise.
- **Easier Searching and Discovery:** Finding a specific style or artist in a large physical gallery can sometimes be challenging. An online portal can include features like categories, filters (as seen in the gallery page of this project), and potentially search functions. This makes it much easier for buyers to find exactly what they are looking for or discover new art that matches their tastes.
- **Streamlined Interaction:** Online tools can make communication more efficient. Instead of relying solely on phone calls or in-person visits, users can use contact forms (like the one implied in this project) to send inquiries anytime, and these can be stored centrally for the gallery to manage.

In summary, an online shopping portal for art addresses the limitations of physical-only galleries. It provides greater reach for artists, unmatched convenience and accessibility for buyers, and efficient ways to manage interactions, making it an essential tool for a modern art gallery.

5. Profile of the Problem

Before building this online system, several specific problems existed in the traditional way of managing art sales and interactions, which this project aimed to solve:

- **Limited Artist Exposure:** Many talented artists, particularly those who are emerging or live outside major art hubs, struggle to get their work noticed.

Physical gallery representation is competitive and often geographically limited. This means a lot of great art may never reach potential buyers simply due to a lack of visibility. The problem is finding an effective way to showcase a wide range of artists to a broad audience.

- **Buyer Inconvenience and Limited Choice:** For potential buyers, finding art often involves visiting multiple physical galleries, which takes time and effort and is restricted to gallery opening hours. The selection at any single gallery is limited by its physical space and curation choices. Buyers might find it difficult to compare pieces across different locations or easily find works in specific styles or price ranges they are interested in.
- **Inefficient Communication and Management:** Handling inquiries, tracking customer interest, managing artist details, and processing sales manually can be very time-consuming for gallery staff or individual artists. Relying on phone calls, emails without a central system, or paper records can lead to delays, lost information, and administrative overhead. There was a need for a more organized and efficient way to handle these interactions.
- **Geographical Restrictions:** The art market can feel very localized. Buyers in one city might have little access to artists in another region or country, and vice-versa. This geographical barrier limits opportunities for both sides. The problem was how to create a space where location is no longer a major obstacle to connecting artists and buyers.
- **Lack of a Dedicated Online Hub:** For "The Art Gallery" specifically, there wasn't a single, integrated online platform where users could perform all key actions – browse art, learn about artists, manage potential purchases (cart), and contact the gallery securely. Information might have been scattered or interactions handled through less efficient means.

This Art Sale Management System project was designed to directly tackle these issues by providing a centralized, accessible, and feature-rich online platform.

6. Structure of the Project

The Art Sale Management System is organized into several distinct parts, mostly represented by different PHP files. Each file handles a specific function or displays a particular section of the website. This modular structure helps keep the code organized and easier to manage. Here's a breakdown of the main components:

- **Configuration (config.php):** This is a crucial background file. Its only job is to set up and manage the connection to the MySQL database, making the database available to other parts of the project that need it.

- **User Authentication (login.php, register.php, logout.php):** This group of files handles everything related to user accounts.
  - register.php: Provides the form for new users to sign up and processes the registration, securely saving user details to the database.
  - login.php: Provides the form for existing users to log in and verifies their credentials against the database. It starts the user's session upon success.
  - logout.php: Ends the user's session, effectively logging them out.
- **Core User Interface (homepage.php):** This acts as the main dashboard or landing page for users after they log in. It typically includes navigation to other sections and might feature highlights like the image slider. It checks if the user is logged in before displaying content.
- **Content Display (gallerypage.php, artistpage.php):** These files are responsible for showing the art and artist information.
  - gallerypage.php: Displays the artworks, includes filtering options, and has the "Add to Cart" buttons that interact with the JavaScript cart functionality.
  - artistpage.php: Displays information about the artists featured in the gallery.
- **E-commerce Functionality (cart.php, checkout.php, success.php):** These files manage the shopping aspect.
  - cart.php: Shows the items the user has added to their cart (using JavaScript and localStorage), allowing them to adjust quantities or remove items.
  - checkout.php: Takes the cart information (passed via sessionStorage), displays a summary, and simulates the payment selection step.
  - success.php: A simple page shown after the simulated checkout is complete.
- **Contact & Submissions (contact.php [Implied], view_submission.php):**
  - contact.php (Implied): A page with a form for users to send messages or submit artwork inquiries to the gallery (the form processing logic would save data to the database).
  - view_submission.php: A page designed to display the messages and file information submitted through the contact form, retrieved from the database. (*Note: Currently lacks security controls*).

Within these PHP files, HTML is used for the structure, CSS for the visual styling, and JavaScript is embedded to handle client-side actions like the image slider, cart operations, and gallery filtering, making the website interactive.

7. Software Development Life Cycle (SDLC)

To build the Art Sale Management System, a structured approach was helpful, even for a smaller project. We can think of the process using a simple model like the **Waterfall Model**. This model involves completing one phase before moving on to the next, like water flowing down steps. Here's how it applied to this project:

1. **Requirement Gathering and Analysis:**
   - **What happened:** This was the first step, where we figured out what the website needed to do. We looked at the problems (like artists needing more visibility and buyers needing convenience) and decided on the essential features: user accounts (login/register), a gallery to show art, a way to select items (shopping cart), a checkout process (even if simulated), and a contact method.
   - **Outcome:** A list of features and understanding of what the final website should achieve.
2. **System Design:**
   - **What happened:** Once we knew *what* was needed, we planned *how* to build it. This involved deciding on the technologies (PHP, MySQL, HTML, CSS, JS), planning the structure of the website (which pages are needed and how they link together), and designing the database tables (users, contact_submissions). We also planned the look and feel of the website and how users would navigate it.
   - **Outcome:** A blueprint for the website, including page layouts, database structure, and technology choices.
3. **Implementation (Coding):**
   - **What happened:** This was the main building phase. Based on the design, the actual code was written. This included creating the database tables in MySQL, writing the PHP scripts for user login/registration and database interactions (config.php), building the HTML structure for each page, applying CSS for styling, and adding JavaScript for features like the homepage slider and the shopping cart logic (localStorage).
   - **Outcome:** The working code for the website.
4. **Testing:**
   - **What happened:** After coding, we needed to check if everything worked correctly. This involved testing each feature individually (e.g., Does login work with correct/incorrect passwords? Can items be added to the cart? Do the gallery filters function?). We also tested if the pages linked together properly and looked for any obvious errors or bugs.
   - **Outcome:** Identifying and fixing bugs to ensure the website functions as

intended.

5. **Deployment:**
   - **What happened:** For this project, deployment likely meant setting it up on a local web server environment like XAMPP, WAMP, or MAMP. This involves placing the project files in the correct directory (e.g., htdocs) and ensuring the database is set up, making the website accessible via a web browser on the local machine (http://localhost/...).
   - **Outcome:** The website running in a test environment.
6. **Maintenance:**
   - **What happened:** This phase happens after the main project is delivered. It involves ongoing activities like fixing any bugs discovered later, making small updates, or potentially adding minor new features based on feedback.
   - **Outcome:** A website that continues to function correctly over time.

Using this step-by-step approach helped ensure that all necessary parts were considered and built in a logical order, leading to the final Art Sale Management System.

8. Problem Analysis

Looking closer at the problems mentioned earlier (in Section 5), we can analyze *why* they were significant issues that needed addressing:

- **Analyzing Limited Artist Exposure:** The difficulty for artists to gain visibility isn't just frustrating for them; it also means the art world potentially misses out on unique talents and perspectives. If only established or geographically advantaged artists get seen, the market becomes less diverse. Furthermore, artists struggling for exposure might find it harder to sustain their careers financially. This project aimed to lower that barrier by providing an accessible online showcase.
- **Analyzing Buyer Inconvenience:** When buying art is inconvenient (limited hours, travel required, difficult comparisons), people are less likely to engage with it regularly or make spontaneous discoveries. This can slow down the art market. Buyers might stick to known options rather than exploring new artists. The lack of easy comparison tools makes informed decision-making harder. This project sought to make browsing and discovering art as easy as any other online shopping experience.
- **Analyzing Inefficient Processes:** Manual management of inquiries and sales doesn't just waste time; it can also lead to errors. A missed email or a lost phone message could mean a lost sale or a frustrated customer. For artists managing

their own sales, this administrative burden takes time away from creating art. For a gallery, it increases operational costs and can impact customer service quality. The database storage for contact submissions in this project is a first step towards solving this.

- **Analyzing Geographical Restrictions:** Limiting art sales primarily to local markets restricts growth potential significantly. An artist in one region might have potential buyers worldwide, but no way to reach them easily. Similarly, buyers are limited to the art available locally. This hinders the natural flow of art and culture across borders. An online platform directly combats this by creating a potentially global marketplace.
- **Analyzing Lack of Online Hub:** Without a central, professional website, "The Art Gallery" would struggle to build a recognizable online brand. Customers expect businesses, including galleries, to have a functional website. Lacking one can appear unprofessional and makes it harder to communicate updates, feature artists, or provide a consistent experience. This project provides that essential online "storefront" and operational base.

Understanding these underlying issues and their consequences reinforces the value of creating the Art Sale Management System. It wasn't just about building a website, but about solving real problems faced by artists, buyers, and the gallery itself in the modern digital landscape.

9. Project Plan

To ensure the Art Sale Management System was developed effectively, a clear plan was followed. This plan broke down the work into logical steps, guiding the project from start to finish.

1. **Phase 1: Understanding and Defining (Requirements & Analysis)**
   - **Activity:** Identify the main goals (online gallery, sales platform, user accounts). Analyze the problems faced by artists and buyers (as detailed in Section 5 and 8). Define the specific features needed to solve these problems (e.g., secure login, gallery display with filters, shopping cart, contact form, submission viewer).
   - **Goal:** Have a clear list of what the website must do (functional requirements) and how it should perform (e.g., be secure, easy to use).
2. **Phase 2: Choosing the Tools (Technology Selection)**
   - **Activity:** Decide on the programming languages and tools. PHP was chosen for server-side logic due to its wide use in web development and good database integration. MySQL was selected for the database to store user and

submission data. HTML, CSS, and JavaScript were chosen for the front-end structure, styling, and user interactions (like the cart).
    - **Goal:** Select appropriate and familiar technologies to build the system effectively.
3. **Phase 3: Blueprinting the System (Design)**
    - **Activity:** Plan the overall structure of the website. This involved deciding which pages were needed (login.php, gallerypage.php, cart.php, etc.) and how they would link together (navigation). Design the database schema, specifying the tables (users, contact_submissions) and their columns. Create basic wireframes or mockups for page layouts and user interface elements.
    - **Goal:** Create a detailed plan for developers to follow during coding, covering architecture, database, and UI/UX.
4. **Phase 4: Building the Website (Implementation)**
    - **Activity:** Write the actual code based on the design specifications. This involved setting up the database tables in MySQL, coding the PHP scripts for backend logic (user authentication, database queries), building the HMTL structure for each page, applying CSS styles, and implementing JavaScript features (slider, cart functions, filters). The config.php file for database connection was created.
    - **Goal:** Develop all the planned features and components into a working website.
5. **Phase 5: Checking the Work (Testing)**
    - **Activity:** Test the implemented features to ensure they work correctly and find any bugs. This included testing login with valid/invalid credentials, adding/removing items from the cart, checking gallery filters, ensuring page navigation works, and viewing submissions (with awareness of its security flaw). Basic checks across different browsers might have been performed.
    - **Goal:** Identify and fix errors to ensure the website is functional and reliable.
6. **Phase 6: Setup and Documentation (Deployment & Reporting)**
    - **Activity:** Deploy the project files to a working environment (likely a local server like XAMPP) for final testing and demonstration. Prepare the project documentation (this report) explaining the project's details, process, and outcomes.
    - **Goal:** Make the project runnable and provide a comprehensive report explaining its development.

Following these planned phases helped organize the workflow and ensure all essential aspects of the project were addressed systematically.

10. Activities During Software Project Planning

The project planning stage, which happens early in the development lifecycle, involved several key activities to set the foundation for the Art Sale Management System. This wasn't just about deciding *what* to build, but also *how* to approach it effectively.

- **Defining Scope and Objectives:** This was perhaps the most critical planning activity. It involved clearly outlining the main goal: to create an online platform for The Art Gallery. More importantly, it meant defining the boundaries – what features *must* be included in this version (like user login, gallery view, basic cart, contact submission storage) and what features would be considered out of scope for now (like real payment processing, artist upload portals, advanced admin dashboards). This helps prevent the project from becoming too complex or delayed.
- **Detailed Feature Identification (Requirements Gathering):** Building on the scope, planning involved listing out the specific functions needed. This meant going beyond "user accounts" to specify "user registration with password hashing," "user login with session management," and "user logout." For the gallery, it meant "display artworks," "filter by category," and "add to cart button." This detailed list formed the basis for the design and implementation phases.
- **Technology Selection and Assessment:** A conscious decision was made during planning to use PHP for the backend, MySQL for the database, and HTML/CSS/JavaScript for the frontend. This planning step considered factors like the common use of these technologies for web development, the availability of learning resources and community support, their suitability for building dynamic websites with database interaction, and likely the developer's existing familiarity with them. Choosing standard, well-supported technologies reduces risks during development.
- **Resource Identification:** Planning also involved identifying the necessary resources. This included the software tools (code editor, web server package like XAMPP, web browser) and hardware (a computer capable of running these tools). It also implicitly included the main resource: the developer's time and effort.
- **Initial Time and Effort Estimation:** Although perhaps informal, planning involved estimating how much work each part might take and considering any deadlines (especially if it was an academic project). This helps in allocating time effectively during the implementation phase.
- **Basic Risk Consideration:** Good planning involves thinking about potential

problems. For example, considering the security implications of user login led to planning the use of password hashing. Choosing familiar technologies helped reduce the risk of getting stuck on complex technical issues. While not explicitly documented as a formal risk assessment, these considerations are part of effective planning.

These planning activities collectively created a roadmap for the project. They ensured that everyone involved understood the goals, the features to be built, the tools to be used, and the general approach, making the subsequent design, implementation, and testing phases much more focused and manageable.

11. Hardware and Software Requirement

To develop and run the Art Sale Management System website, specific hardware and software components are needed. These requirements cover both the development process and what an end-user would need to access the site.

A. Hardware Requirements:

- **Development Machine:**
  - A standard desktop computer or laptop is required for development.
  - **Processor:** A reasonably modern processor (e.g., Intel Core i3/i5/i7, AMD Ryzen equivalent, or newer) is sufficient.
  - **RAM (Memory):** Minimum 4GB is recommended, but 8GB or more provides a smoother experience when running development tools simultaneously (code editor, web server, browser, database client).
  - **Storage:** Enough free hard drive space (e.g., 10GB+) to install the necessary software (XAMPP, code editor) and store the project files and database. An SSD (Solid State Drive) improves performance significantly.
  - **Peripherals:** Standard keyboard, mouse, and monitor.
- **Web Server (for Deployment):**
  - For local testing (as done in this project), the development machine acts as the server using software like XAMPP.
  - If deployed online, a dedicated hosting server (physical or virtual) with appropriate specifications (CPU, RAM, storage) would be needed to handle website traffic.
- **End-User Device:**
  - Any computer (desktop, laptop), tablet, or smartphone capable of running a modern web browser and connecting to the internet.

B. Software Requirements:

- **Operating System (OS):**
  - **Development:** Windows (7, 8, 10, 11), macOS, or a Linux distribution (like Ubuntu). The choice depends on developer preference.
  - **Server (Deployment):** Often Linux-based (like Ubuntu Server, CentOS) for hosting, but Windows Server is also possible.
  - **End-User:** Any standard OS (Windows, macOS, Linux, iOS, Android).
- **Web Server Software:**
  - **Apache:** This is required to process the PHP files and serve the website content. It's a standard choice for PHP applications and is included in packages like XAMPP. (Alternatives like Nginx could also be used).
- **Database Server:**
  - **MySQL (or MariaDB):** Needed to store and manage the data for user accounts and contact submissions. MariaDB is a common drop-in replacement often bundled with XAMPP.
- **Programming Language Interpreter:**
  - **PHP:** A specific version of PHP (e.g., PHP 7.x or 8.x) needs to be installed on the server (or local development machine) to execute the backend scripts.
- **Web Browser:**
  - **Development:** Modern browsers like Google Chrome, Mozilla Firefox, or Microsoft Edge are needed for testing the website's appearance and functionality. Developer tools within these browsers are essential for debugging.
  - **End-User:** Any up-to-date web browser.
- **Code Editor / IDE:**
  - A text editor designed for coding (e.g., Visual Studio Code, Sublime Text, Notepad++, Atom) is needed to write and edit the project's source code (PHP, HTML, CSS, JS).
- **Local Server Environment Package (Optional but Recommended for Development):**
  - Software like **XAMPP**, **WAMP** (Windows), or **MAMP** (macOS) bundles Apache, MySQL/MariaDB, and PHP together, making the local development setup much easier.

These requirements represent a fairly standard setup for developing and running a typical PHP/MySQL web application like the Art Sale Management System.

12. Front-End Details

The front-end of the Art Sale Management System is what the user directly sees and interacts with in their web browser. It's built using a combination of standard web technologies that work together to create the visual presentation and interactive experience.

- **HTML (HyperText Markup Language):**
  - **Role:** HTML forms the fundamental structure and content of every page on the website. It defines elements like headings, paragraphs, lists, images, links, forms, and buttons.
  - **Usage:** In this project, HTML was used extensively to structure the login and registration forms (login.php, register.php), lay out the navigation bar and footer (header, footer elements used across pages like homepage.php), define the grid for displaying artworks (gallerypage.php), structure the artist profiles (artistpage.php), build the table-like display for the shopping cart (cart.php), and create the order summary section (checkout.php). Semantic HTML elements were used where appropriate to improve structure and accessibility.
- **CSS (Cascading Style Sheets):**
  - **Role:** CSS is responsible for the entire visual presentation and styling of the HTML structure. It controls layout, colors, fonts, spacing, borders, backgrounds, and responsive behavior.
  - **Usage:** CSS was used to create the overall look and feel, including the gradient headers, button styles (like the "Explore More" and "Add to Cart" buttons), background images (on login/register pages and the homepage hero section), blurred container effects, and consistent typography (using 'Poppins' font). It defines the layout using techniques like Flexbox (evident in the header and footer). Importantly, CSS media queries were used to make the website responsive, ensuring it adapts reasonably well to different screen sizes (desktops, tablets, mobiles). External libraries like Font Awesome were linked via CSS to provide icons. Styles were primarily embedded within <style> tags in the PHP files.
- **JavaScript (JS):**
  - **Role:** JavaScript adds interactivity and dynamic behavior to the website, allowing things to happen in the user's browser without necessarily reloading the page from the server.
  - **Usage:** JavaScript plays several key roles in this project:
    - **Homepage Slider:** Controls the image transitions, handling both automatic sliding (setInterval) and manual navigation via previous/next

        buttons.
        - **Gallery Filtering:** Allows users to click category buttons (filterArt function) to instantly show or hide artworks based on their category without a page refresh.
        - **Shopping Cart Management:** This is a major use of JS. It handles adding items to the cart (addToCart function), reading/writing cart data to the browser's localStorage (making the cart persistent between page loads), updating item quantities, removing items (updateQuantity, removeItem functions), and dynamically updating the cart display on cart.php and the cart item counter in the header (updateCartCount).
        - **Checkout Interaction:** On checkout.php, JavaScript is used to show or hide payment details based on the user's selection from the dropdown menu.
        - **Client-Side Data Transfer:** Uses sessionStorage to pass the finalized cart data from cart.php to checkout.php.
        - **User Feedback:** Used to display simple alert() messages for actions like login success/failure or registration status (though more integrated feedback methods are generally preferred).

Together, HTML provides the skeleton, CSS provides the visual design and layout, and JavaScript provides the interactive features and dynamic updates, creating the complete front-end experience for the user.

13. Back-End Details

The back-end is the part of the Art Sale Management System that runs on the web server, handling tasks that the user doesn't directly see but are essential for the website to function correctly. It manages data, security, and the core logic of the application.

- **PHP (Hypertext Preprocessor):**
    - **Role:** PHP is the primary server-side scripting language used in this project. It runs on the web server and is responsible for processing user input, interacting with the database, managing user sessions, and generating the HTML content that gets sent back to the user's browser.
    - **Usage:**
        - **Form Processing:** PHP scripts in login.php and register.php receive the data submitted from the HTML forms (username, password).
        - **User Authentication Logic:** It handles the core security tasks: verifying usernames against the database, securely comparing submitted

passwords with stored hashes using password_verify(), and hashing new passwords with password_hash() before saving them during registration.

        - **Session Management:** PHP's session functions (session_start(), $_SESSION, session_destroy()) are used to keep users logged in across different pages and to log them out securely. It also controls access to protected pages like homepage.php by checking for session variables.
        - **Database Interaction:** PHP connects to the MySQL database (using the mysqli extension and connection details from config.php). It constructs and executes SQL queries (using prepared statements for security in login/register) to fetch user data, insert new users, and retrieve contact submissions (view_submission.php).
        - **Dynamic Content Generation:** Although much of the content display (like the gallery) uses hardcoded data in this version, PHP *could* be used to dynamically generate HTML based on data retrieved from the database (e.g., displaying artworks stored in a database table).
- **MySQL (Database Server):**
    - **Role:** MySQL serves as the relational database management system (RDBMS). Its job is to store the application's data in an organized way (using tables, rows, and columns) and allow PHP to efficiently retrieve or modify that data using SQL queries.
    - **Usage:**
        - **users** Table: Stores essential user account information, most importantly the username and the securely hashed password.
        - **contact_submissions** Table: Stores data submitted through the contact form, including the sender's details, message, and the filename of any uploaded file. This allows the information to be viewed later via view_submission.php.
        - **Data Persistence:** The database provides long-term storage for data, unlike browser storage (localStorage, sessionStorage) which can be cleared by the user.
- **Web Server (Apache):**
    - **Role:** Apache is the software that listens for incoming requests from users' web browsers. When a request for a PHP page comes in, Apache passes it to the PHP interpreter. Once PHP finishes processing and generates HTML output, Apache sends that output back to the user's browser.
    - **Usage:** It acts as the intermediary between the user's browser and the PHP scripts, managing the request-response cycle. It's typically run locally using

packages like XAMPP for development.

- **Configuration File (config.php):**
    - **Role:** This separate PHP file holds the database connection credentials (host, username, password, database name).
    - **Usage:** By using require 'config.php'; in other PHP files (login.php, register.php, view_submission.php), the project avoids repeating connection code and makes it easy to update database details in one central place.

These back-end components work together seamlessly. The web server receives a request, PHP processes it (often interacting with the MySQL database via the config file), and the server sends the final result back to the user's browser (the front-end).

14. System Design

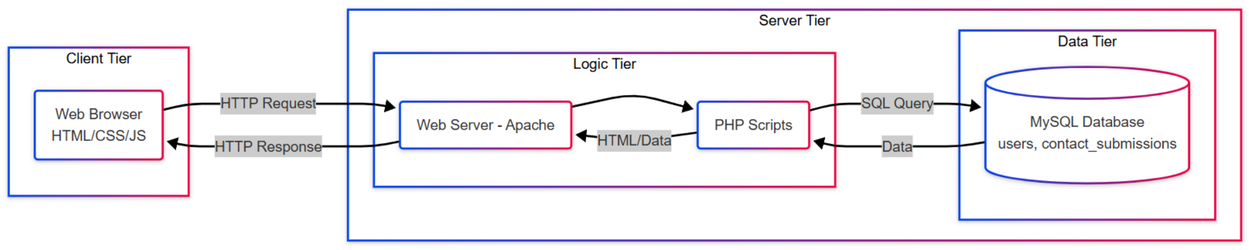
The system design outlines the overall architecture or blueprint of the Art Sale Management System, explaining how its different parts are organized and interact with each other. This project uses a common and effective design pattern for web applications.

A. Architecture:

- **Client-Server Model:** The system operates on a client-server basis. The **Client** is the user's web browser (like Chrome or Firefox) which requests information or sends data. The **Server** is the computer where the website's files and database are hosted (likely a local machine using XAMPP during development). The server processes requests from the client and sends back responses (web pages, data).
- **3-Tier Architecture:** We can also think of the design in three logical layers or tiers, as illustrated below:
    1. **Presentation Tier (Front-End):** This is what the user interacts with – the web pages displayed in the browser. It's built using HTML (structure), CSS (styling), and JavaScript (interactivity like the cart and slider). Its main job is to display information and collect input from the user.
    2. **Logic Tier (Back-End - Application Server):** This is where the "thinking" happens. The PHP scripts running on the web server (Apache) form this tier. It handles the core application logic, such as validating user logins, processing registrations, managing sessions, deciding what data to fetch from the database, and preparing the response to send back to the browser.
    3. **Data Tier (Back-End - Database Server):** This tier consists of the MySQL database server. Its sole purpose is to store and manage the application's data persistently and securely (user accounts, contact

submissions). The Logic Tier (PHP) communicates with this tier to read or write data as needed.

Conceptual Architecture Diagram:



- **Explanation:** The diagram shows the User interacting with the Browser (Client Tier). The Browser sends requests to the Web Server (part of the Server Tier's Logic Tier). The Web Server uses PHP scripts (Logic Tier) which in turn interact with the MySQL Database (Data Tier) to fetch or store data. PHP generates the response, which the Web Server sends back to the Browser.

B. Component Interaction & Data Flow:

The different parts work together in a typical request-response cycle:

1. **Request:** A user performs an action in their browser (e.g., clicks login, adds to cart, visits the gallery page). The browser sends an HTTP request to the web server.
2. **Processing (Server):**
    - The Web Server (Apache) receives the request and identifies the requested PHP file.
    - Apache passes the request to the PHP interpreter.
    - The PHP script executes. This might involve:
        - Reading request data (e.g., form inputs from POST).
        - Interacting with the session (`$_SESSION`).
        - Connecting to the MySQL database (via `config.php`).
        - Querying the database (e.g., SELECT user, INSERT submission).
        - Performing application logic (e.g., verifying password, calculating totals - though cart totals are client-side here).

        - Generating HTML output.
3. **Response:** The web server sends the generated HTML (along with any CSS and JavaScript references) back to the user's browser as an HTTP response.
4. **Rendering (Client):** The browser receives the response, interprets the HTML and CSS to display the page, and executes any JavaScript for interactivity (like initializing the cart display or the slider).

C. Modularity:

As described in Section 6 (Structure of the Project), the system is designed with modularity in mind. Different functionalities (login, gallery, cart) are handled by separate PHP files. This makes the codebase easier to understand, develop, and maintain.

D. State Management:

The system manages different types of "state" (information that needs to be remembered):

- **User Login State:** Managed server-side using PHP Sessions. A session variable (`$_SESSION['logged_in_user']`) is set on login and checked on protected pages.
- **Shopping Cart State:** Managed client-side using the browser's `localStorage`. This allows the cart contents to persist even if the user closes the browser tab and returns later.
- **Checkout Data Transfer:** Managed temporarily using the browser's `sessionStorage` to pass the cart contents from the cart page to the checkout page. This data is cleared when the browser tab is closed.

This layered, modular design provides a clear separation of concerns (presentation, logic, data) and utilizes appropriate mechanisms for managing different types of application state.

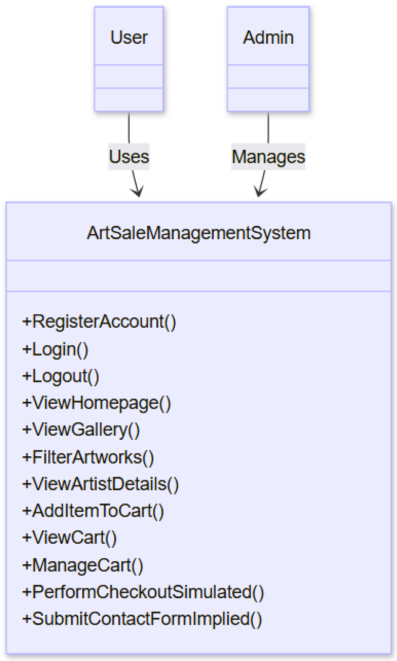
15. Design Notations

During the planning and design phases of software development, various standard diagrams and notations are often used to visualize and communicate the system's structure and behavior. While formal, detailed diagrams might not have been extensively created for every aspect of this specific project, understanding these notations helps clarify the design concepts used.

- **Entity-Relationship (ER) Diagrams:**
    - **What they show:** ER diagrams are used to model the structure of a database. They show the main data objects (Entities, like User, ContactSubmission) and the relationships between them (e.g., one-to-many,

    many-to-many). They also typically show the key pieces of information (Attributes) stored for each entity.
  - **Relevance to this project:** An ER diagram would visually represent the users and contact_submissions tables in the MySQL database. It would clearly show the columns in each table (like username, passwordHash in users; name, email, message, filename in contact_submissions) and identify primary keys (like userId, submissionId). It helps ensure the database design is logical and efficient before creating the actual tables. (A conceptual ER diagram was described in Section 18).
- **Data Flow Diagrams (DFDs):**
  - **What they show:** DFDs illustrate how data moves through a system. They show where data comes from (sources), where it goes (destinations), what processes transform the data, and where data is stored. They focus on the flow of information rather than processing logic.
  - **Relevance to this project:** DFDs could be used to map out key processes. For example, a DFD for user registration would show data flowing from the user (via the form) into a "Process Registration" step, which interacts with the "Users" data store (database) and potentially sends confirmation data back to the user. Similarly, a DFD could show how contact form data flows into the "Contact Submissions" data store.
- **Flowcharts:**
  - **What they show:** Flowcharts are used to represent the step-by-step logic or algorithm of a specific process or function. They use standard symbols (like rectangles for processes, diamonds for decisions, ovals for start/end) connected by arrows to show the sequence of operations.
  - **Relevance to this project:** Flowcharts are very useful for detailing the logic within specific PHP functions or JavaScript code blocks. For instance, a flowchart could clearly illustrate the steps involved in login.php (get input -> query DB -> check if user exists -> verify password -> set session/show error). Another could map out the addToCart JavaScript function's logic (get item details -> check localStorage -> update quantity or add new item -> save to localStorage -> update counter). (Conceptual flowcharts were described in Section 19).
- **Use Case Diagrams (Part of UML - Unified Modeling Language):**
  - **What they show:** These diagrams illustrate how users (Actors) interact with the system to achieve specific goals (Use Cases). They provide a high-level view of the system's functionality from a user's perspective.

- **Relevance to this project:** A simple Use Case diagram could show Actors like "Registered User" and "Administrator" (implied). The "Registered User" could interact with Use Cases like "Login," "View Gallery," "Add to Cart," "Checkout," "Logout." The "Administrator" might interact with "View Submissions."
-



Explanation:

- Actors: User represents any visitor or logged-in user interacting with the public/member features. Admin represents the intended user for viewing submissions (currently implied as the page lacks specific admin login).
- System Boundary: The rectangle encloses all the functions (Use Cases) provided by the Art Sale Management System.
- Use Cases: The ovals represent major functions like (Login), (View Gallery), (Add Item to Cart), etc.
- Associations: The lines connect actors to the use cases they perform. For example, the User can perform Login, View Gallery, Add Item to Cart, etc. The Admin can perform View Submissions.
- Understanding these standard notations provides a valuable framework for thinking about, communicating, and documenting the system's design effectively.

While this project might have relied more on informal design planning, understanding these standard notations provides a valuable framework for thinking about, communicating, and documenting the system's design effectively.

16. Product Function

This section outlines the specific functions and features that the Art Sale Management System offers to its users. These functions collectively define what the website can do.

A. User Account Management:

- **User Registration:** Allows new visitors to create a personal account by providing a unique username and a password. The system securely hashes the password before storing it in the database (register.php).
- **User Login:** Enables registered users to securely access their accounts by entering their username and password. The system verifies the credentials against the database using secure methods (login.php).
- **Session Management:** Once logged in, the system maintains a user session, allowing the user to navigate protected pages (like homepage.php) without needing to log in repeatedly.
- **User Logout:** Provides a clear way for logged-in users to securely end their session and log out of the system (logout.php).
- **Access Control:** Ensures that only logged-in users can access specific parts of the website, such as the main homepage.

B. Art and Artist Discovery:

- **Gallery Viewing:** Displays a collection of artworks, typically showing an image, name, and price for each piece (gallerypage.php).
- **Artwork Filtering:** Allows users to filter the displayed artworks based on categories (e.g., Painting, Sculpture, Digital Art), making it easier to find specific types of art (gallerypage.php).
- **Artist Information:** Presents details about the artists featured in the gallery, such as their name and biography (artistpage.php).
- **Homepage Display:** Offers a welcoming page for logged-in users, potentially featuring highlights, promotions, or featured artworks (like the image slider on homepage.php).

C. E-commerce / Shopping Features:

- **Add to Cart:** Enables users browsing the gallery to select artworks they are interested in and add them to a personal shopping cart (gallerypage.php, using JavaScript).

- **View Shopping Cart:** Displays all the items currently added to the user's cart, showing details like the item image, name, price, and quantity (cart.php).
- **Manage Cart Contents:** Allows users to modify the quantity of items in their cart or remove items completely (cart.php, using JavaScript). The cart state is remembered using localStorage.
- **Simulated Checkout Process:** Guides the user through the steps of finalizing a purchase, including viewing an order summary and selecting a payment method (checkout.php). (Note: Payment processing is simulated).
- **Order Confirmation:** Displays a success message to the user after they complete the simulated checkout process (success.php).

D. Communication and Administration:

- **Contact Form (Implied):** Provides a mechanism for users to send inquiries or messages to the gallery administrators. The submitted data is stored in the database.
- **View Contact Submissions:** Allows an authorized user (intended for administrators, but currently lacks security) to view the messages, contact details, and potentially download files submitted through the contact form (view_submission.php).

These functions work together to provide a comprehensive online experience for users interested in browsing, potentially purchasing, and interacting with The Art Gallery and its artists.

17. Detailed Design

This section provides more specific details about the design and implementation choices made for key components of the Art Sale Management System.

A. Database Design:

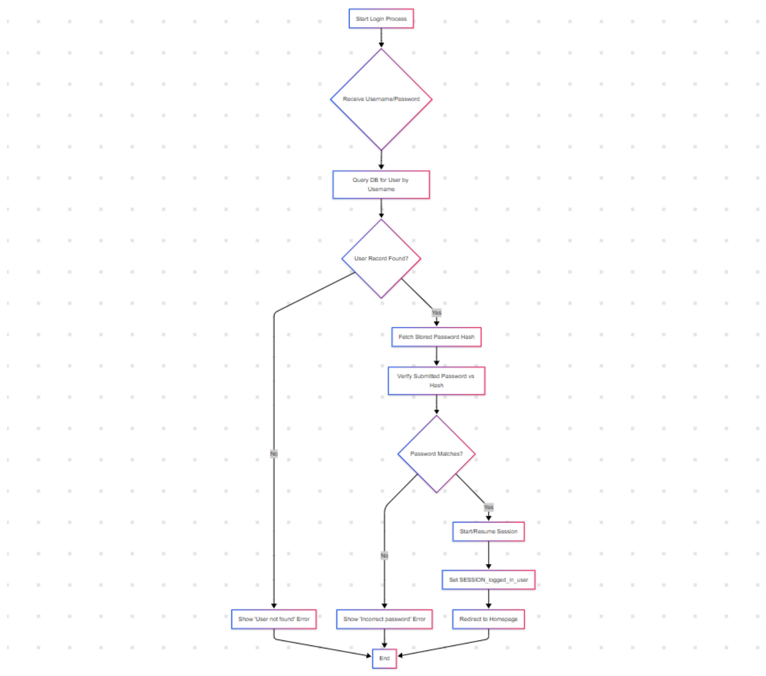
- **users** Table: This table is central to authentication.
  - It includes columns like userId (likely an auto-incrementing primary key, though not explicitly shown in code), username (which should have a UNIQUE constraint in the database to prevent duplicates), and password.
  - Crucially, the password column stores **hashed** passwords generated using PHP's password_hash() function with PASSWORD_DEFAULT. This ensures that plain-text passwords are never stored, significantly enhancing security.

- **contact_submissions** Table: This table captures data from user inquiries.
    - Columns include submissionId (primary key), name, email, phone, message (likely a TEXT type to accommodate longer messages), file (storing the filename if one was uploaded), and submitted_at (a TIMESTAMP or DATETIME column to record when the submission occurred).
    - The design allows for optional file uploads associated with submissions.
- **Connection Management:** The config.php file centralizes database connection parameters (host, username, password, dbname), promoting consistency and making it easier to update credentials if needed. The mysqli extension is used for database operations.

B. User Authentication Design:

- **Registration (register.php):**
    1. Receives username/password via POST.
    2. Performs a database query (using a prepared statement) to check if the username already exists.
    3. If unique, it hashes the password using password_hash().
    4. Inserts the username and the *hashed* password into the users table (using a prepared statement).
- **Login (login.php):**
    1. Receives username/password via POST.
    2. Queries the database for the user record matching the username (using a prepared statement).
    3. If a user is found, it uses password_verify() to compare the submitted password against the stored hash.
    4. If verification succeeds, it starts a session (session_start()) and stores an identifier (e.g., the username) in the $_SESSION array (e.g., $_SESSION['logged_in_user'] = $username;).
- **Session Handling:** Standard PHP sessions are used. session_start() is called at the beginning of pages that need session access. Protected pages (like homepage.php) check if isset($_SESSION['logged_in_user']). logout.php calls session_destroy() to clear the session data.

Login Verification Flowchart :


Start Login Process
Receive Username/Password
Query DB for User by Username
User Record Found?
Yes
No
Fetch Stored Password Hash
Verify Submitted Password vs Hash
Password Matches?
Yes
No
Start/Resume Session
Set SESSION_logged_in_user
Show 'User not found' Error
Show 'Incorrect password' Error
Redirect to Homepage
End

- **Explanation:** This flowchart details the steps within `login.php` after the user submits the form. It shows querying the database, checking if the user exists, verifying the password hash, setting the session on success, and showing appropriate errors otherwise.

C. Shopping Cart Design (Client-Side):

- **Storage:** The browser's localStorage was chosen to store the cart contents. This provides persistence – the cart remains even if the user closes the browser or navigates away and returns later.
- **Data Structure:** The cart is stored as a JSON string representing an array of JavaScript objects. Each object typically contains keys like name, image, price, and quantity.
- **JavaScript Logic:**
  - addToCart(): Retrieves the current cart from localStorage, parses the JSON string into an array, checks if the item already exists (based on image URL or a potential ID), either increments the quantity or adds a new item object to the array, converts the array back to a JSON string, and saves it back to localStorage.
  - updateQuantity() / removeItem(): Retrieve the cart, parse it, modify the array (update quantity or use splice() to remove), stringify, and save back to localStorage.

    - loadCart(): Retrieves the cart from localStorage, parses it, and dynamically generates the HTML to display the cart items on the cart.php page.
    - updateCartCount(): Retrieves the cart, calculates the total quantity of items, and updates the counter element in the header.

D. Checkout Data Transfer Design:

- **Storage:** The browser's sessionStorage is used to pass the cart data from cart.php to checkout.php.
- **Rationale:** sessionStorage was chosen because the checkout data is only needed for that specific browsing session (until the tab is closed). Once the checkout is complete or abandoned, the data is no longer required in sessionStorage. localStorage was used for the main cart persistence.
- **Process:** The proceedToCheckout() function in cart.php reads the final cart from localStorage, potentially sanitizes it, converts it to a JSON string, and stores it in sessionStorage before redirecting to checkout.php. The script in checkout.php then reads this data from sessionStorage.

E. Page Structure and Styling:

- The project uses a modular approach with separate PHP files for distinct sections.
- Styling is primarily handled via embedded CSS within <style> tags in each file, providing specific looks for different pages (e.g., login/register vs. main site pages). Responsive design techniques (media queries) are included to adapt the layout for smaller screens.

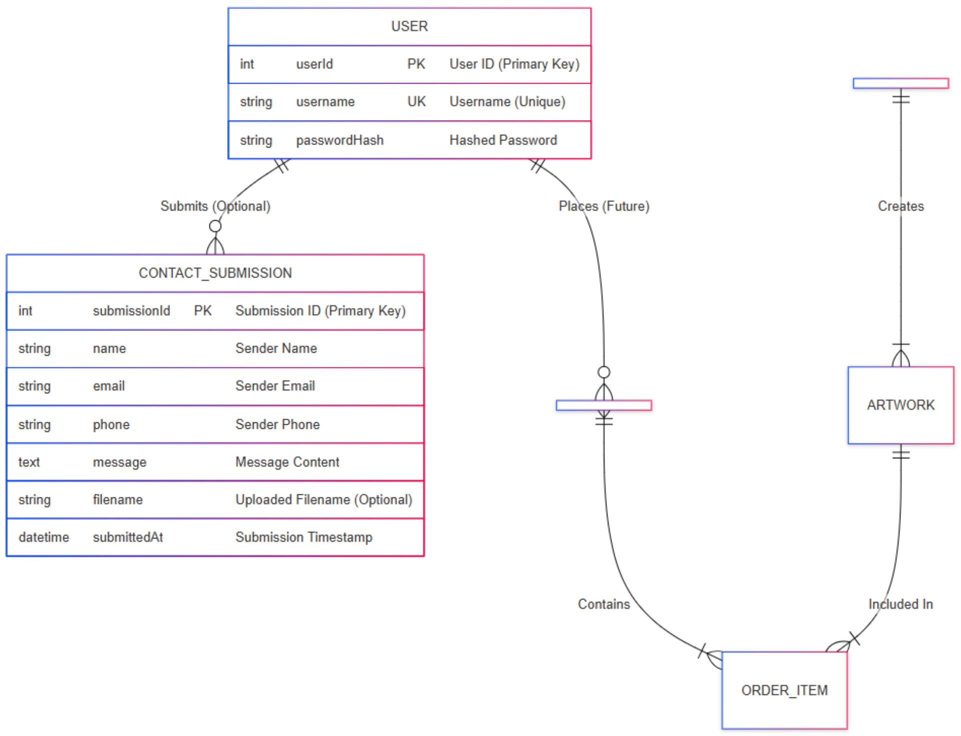
This detailed design approach combines server-side PHP/MySQL for secure data management and authentication with client-side JavaScript/Browser Storage for interactive features like the shopping cart.

18. E-R DIAGRAMS

An Entity-Relationship (ER) diagram is a visual tool used in database design. It helps show the main pieces of information (entities) that the system needs to store and how those pieces of information are related to each other. For the Art Sale Management System, an ER diagram helps us understand the structure of the MySQL database.

Below is a representation of the ER diagram using Mermaid syntax. This includes the tables that were implemented (USER, CONTACT_SUBMISSION) and also conceptual tables (ARTIST, ARTWORK, ORDER, ORDER_ITEM) that would be needed for a fully database-driven gallery and ordering system.

Representation:



USER			
int	userId	PK	User ID (Primary Key)
string	username	UK	Username (Unique)
string	passwordHash		Hashed Password

CONTACT_SUBMISSION			
int	submissionId	PK	Submission ID (Primary Key)
string	name		Sender Name
string	email		Sender Email
string	phone		Sender Phone
text	message		Message Content
string	filename		Uploaded Filename (Optional)
datetime	submittedAt		Submission Timestamp

Explanation of Diagram Components:

- **Entities:** The main boxes (USER, CONTACT_SUBMISSION, ARTWORK, ARTIST, ORDER, ORDER_ITEM) represent the key types of information the system manages or would manage. USER and CONTACT_SUBMISSION correspond to the tables implemented in this project's database. ARTWORK, ARTIST, ORDER, and ORDER_ITEM are shown conceptually to illustrate how the system could be expanded to handle gallery content and actual orders within the database.
- **Attributes:** Inside each box, the listed items are the attributes or specific details stored for that entity. For example, the USER entity stores userId, username, and passwordHash. Explanations are included in quotes. PK marks the Primary Key (unique identifier for each record), UK marks a Unique Key, FK marks a Foreign Key (linking to another table's Primary Key), and NULL indicates an attribute can be empty.
- **Relationships:** The lines connecting the boxes show how these entities relate:
    - USER ||--o{ CONTACT_SUBMISSION: One User can submit zero or more

Contact Submissions.
    - ARTIST ||--|{ ARTWORK: One Artist can create one or more Artworks. (Conceptual)
    - USER ||--o{ ORDER: One User can place zero or more Orders. (Conceptual/Future)
    - ORDER ||--|{ ORDER_ITEM: One Order contains one or more Order Items. (Conceptual/Future)
    - ARTWORK ||--|{ ORDER_ITEM: One Artwork can be included in one or more Order Items. (Conceptual/Future)

This ER diagram provides a clear visual map of the database structure, showing both the currently implemented parts and a potential structure for future enhancements, aiding in understanding data relationships and planning database development.

19. DATA FLOW DIAGRAMS

Data Flow Diagrams (DFDs) are used to show how information moves through the Art Sale Management System. Unlike ER diagrams that show data structure, DFDs focus on the *flow* – where data comes from, how it gets processed or changed, where it's stored, and where it goes. They help visualize the system's processes from an information perspective.

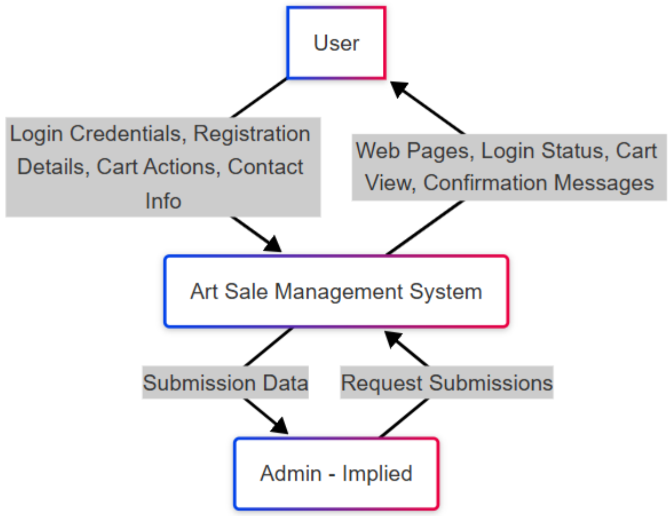
Common symbols in DFDs include:

- **Circles/Ovals:** Represent processes that transform data (e.g., "Verify Login", "Register User").
- **Rectangles:** Represent external entities that interact with the system (e.g., "User", "Admin").
- **Arrows:** Show the direction of data flow (e.g., "Login Credentials", "User Details").
- **Open-Ended Rectangles or Parallel Lines:** Represent data stores where information is kept (e.g., "Users Database Table", "Submissions Database Table", "Browser LocalStorage").

Here are conceptual DFDs for key processes, represented using Mermaid syntax:

A. Level 0 DFD (Context Diagram)

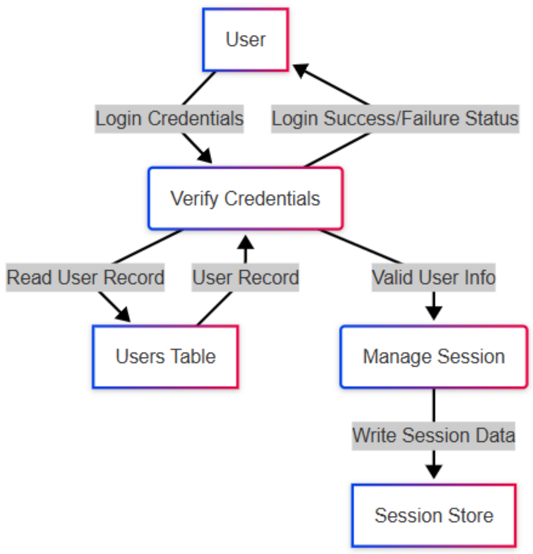
This diagram shows the entire system as a single process and its interaction with external entities.


User
Login Credentials, Registration Details, Cart Actions, Contact Info
Web Pages, Login Status, Cart View, Confirmation Messages
Art Sale Management System
Submission Data
Request Submissions
Admin - Implied

- **Explanation:** The User sends various inputs (login info, registration data, cart actions, contact messages) to the System. The System sends back web pages and status messages. An Admin (implied role) can request submission data and receive it from the System.

B. Level 1 DFD (User Login Process)

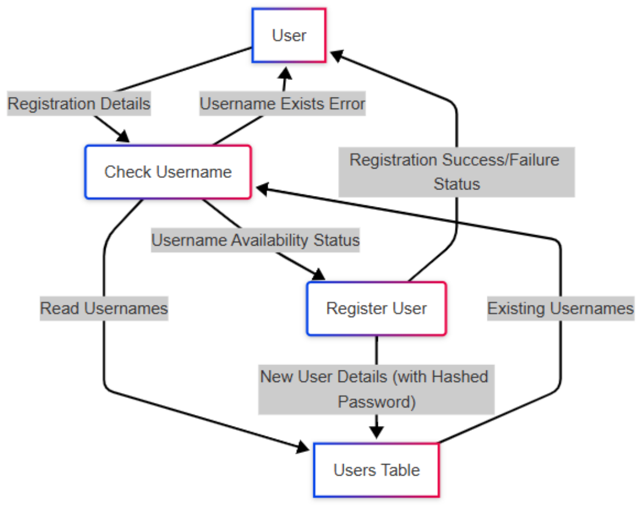
This zooms into the login functionality.



- **Explanation:** The User sends credentials to the "Verify Credentials" process. This process reads data from the "Users Table" data store. Based on the check, it sends a success or failure status back to the User. If successful, it sends user info to the "Manage Session" process, which writes data to the "Session Store".

C. Level 1 DFD (User Registration Process)

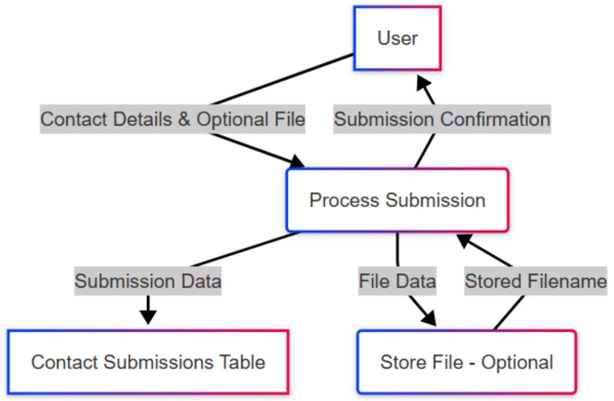
This zooms into the registration functionality.



- **Explanation:** The User sends registration details to the "Check Username" process. This process reads from the "Users Table". If the username is available, it passes control to the "Register User" process, which writes the new user details (including the hashed password) back to the "Users Table". Success or error messages are sent back to the User.

D. Level 1 DFD (Contact Submission Process - Implied)

This shows how a contact submission might be handled.



- **Explanation:** The User sends contact details (and maybe a file) to the "Process Submission" process. This process saves the text data into the "Contact Submissions Table". If there's a file, it might interact with a "Store File" process (which saves the file to the server, perhaps in an uploads/ folder) and get the filename back to store with the submission data. A confirmation is sent back to the User.

These DFDs help visualize the movement and transformation of data within the Art Sale Management System for key operations, complementing the static structure shown in the ER diagram.

20. TESTING

Testing is a vital phase in software development to ensure the application works correctly, meets requirements, and provides a good user experience. For the Art Sale Management System, various testing activities were likely performed, primarily manually, to identify and fix issues.

A. Functional Testing:

- **Purpose:** To verify that each function of the website operates as expected according to the defined requirements.
- **Activities:**
  - **User Authentication:** Tested registration with new usernames, attempted registration with existing usernames, tested login with correct and incorrect

passwords, tested the logout functionality, and verified that protected pages (like homepage.php) correctly redirect non-logged-in users to the login page.
    - **Gallery & Artist Pages:** Checked if artworks and artist details display correctly. Tested the gallery filtering buttons to ensure they show/hide the appropriate items based on category.
    - **Shopping Cart:** Tested adding items from the gallery, verified that the cart counter updates, checked adding the same item multiple times (increments quantity), tested changing quantities (+/- buttons) on the cart page, tested removing items from the cart, and ensured the cart persists (localStorage) when navigating between pages or closing/reopening the browser tab.
    - **Checkout Process:** Tested the flow from the cart page to the checkout page, verified that the order summary correctly reflects the cart contents, tested the payment method selection dropdown, and confirmed redirection to the success page upon clicking the final button.
    - **Submission Viewing:** Checked if the view_submission.php page successfully retrieves and displays data from the contact_submissions table and if file download links work (where applicable).

B. Usability Testing:

- **Purpose:** To evaluate how easy and intuitive the website is to use from an end-user's perspective.
- **Activities:** This was likely informal testing performed by the developer(s). It involved navigating the site, checking if the layout is logical, if buttons and links are clear, if forms are easy to understand and fill out, and if the overall user journey (e.g., finding art, adding to cart, checking out) feels smooth.

C. Integration Testing:

- **Purpose:** To verify that different modules or components of the system work together correctly.
- **Activities:** Tested the flow between different pages (e.g., Register -> Login -> Homepage -> Gallery -> Cart -> Checkout -> Success). Checked if data passed between components works as expected (e.g., user session established on login is recognized on the homepage, cart data saved in localStorage is correctly displayed on cart.php and passed via sessionStorage to checkout.php).

D. Compatibility Testing (Basic):

- **Purpose:** To ensure the website displays and functions reasonably well across different common web browsers.

- **Activities:** Likely involved opening the website in browsers like Google Chrome and Mozilla Firefox to check for major layout issues or functional differences.

E. Security Testing (Basic):

- **Purpose:** To check the implemented security measures.
- **Activities:** Verified that password hashing is used during registration and password_verify during login. Confirmed that sensitive pages require login. Checked if basic input sanitization (like htmlspecialchars on view_submission.php) is used to prevent simple XSS. *(Note: More rigorous security testing, like penetration testing or specific vulnerability scanning, was likely outside the scope but would be essential for a production system)*.

Overall Approach:
Testing was primarily conducted manually by interacting with the website through the browser interface, observing the behavior, and comparing it against expected outcomes. This iterative process of coding and testing helped identify bugs (like incorrect calculations, broken links, or unexpected behavior) which were then fixed, leading to the final version of the project.

21. METHODOLOGY USED FOR TESTING

The process of testing the Art Sale Management System involved several approaches, or methodologies, to ensure different aspects of the website were checked effectively. The primary methodologies used were:

A. Manual Testing:

- **Description:** This was the core methodology. It involved a human tester (likely the developer) interacting with the website directly through a web browser, mimicking the actions a real user would take. This included clicking links, filling out forms (login, registration, contact), adding items to the cart, navigating between pages, and visually inspecting the results.
- **Application:** Manual testing was used extensively for almost all types of testing mentioned in the previous section, including functional testing (checking if features work), usability testing (assessing ease of use), and integration testing (ensuring pages flow correctly).
- **Advantages/Disadvantages:** It's great for finding usability issues and exploring unexpected user paths. However, it can be time-consuming, repetitive, and prone to human error (missing certain checks).

B. Black-Box Testing:

- **Description:** In this approach, the tester focuses only on the inputs and outputs of a feature or the system as a whole, without considering the internal code structure. The system is treated like a "black box."

- **Application:** This was applied during functional testing. For example, when testing the login (login.php):
    - *Input:* Enter a valid username and password. *Expected Output:* Successful login and redirection to homepage.php.
    - *Input:* Enter a valid username but an incorrect password. *Expected Output:* An error message indicating incorrect password.
    - Input: Enter a username that doesn't exist. Expected Output: An error message indicating user not found.
      The tester doesn't need to know how the PHP code verifies the password, only that it produces the correct outcome for given inputs. Similarly, testing the "Add to Cart" button by clicking it and checking if the cart updates is black-box testing.
- **Advantages:** Focuses on user requirements and functionality from the user's perspective. Doesn't require deep coding knowledge.

C. White-Box Testing:

- **Description:** This approach involves looking inside the "box" – examining the actual source code (PHP, JavaScript) to design test cases. The tester uses their knowledge of the code's logic, paths, and conditions to ensure specific parts are working correctly.
- **Application:** This was likely performed by the developer during the coding and debugging process. Examples include:
    - Checking the register.php code to ensure password_hash() is correctly used before inserting data.
    - Verifying that prepared statements are used correctly in login.php and register.php to prevent SQL injection.
    - Inspecting the JavaScript functions for the cart (addToCart, updateQuantity) to ensure the logic for updating quantities or adding new items to the localStorage array is correct.
    - Checking the session handling logic (session_start, isset($_SESSION[...]), session_destroy) in relevant PHP files.
- **Advantages:** Can find errors in specific code paths, conditions, or logic that might be missed by black-box testing. Helps ensure code quality and security practices are followed.

D. Incremental Testing:

- **Description:** Rather than waiting until the entire website was built, testing was likely performed incrementally. As individual modules or features were developed (e.g., login page, then gallery page), they were tested individually (like unit testing). As more features were added, integration testing was performed to ensure the new parts worked correctly with the existing ones.
- **Application:** For example, the login function was likely tested once login.php was created. Later, when homepage.php was built with its access control check, integration testing ensured that only users successfully logged in via login.php could access homepage.php.
- **Advantages:** Helps find and fix bugs earlier in the development process when they are typically easier and cheaper to resolve.

In essence, the testing methodology was a practical combination of manual exploration, checking functions from a user's perspective (black-box), inspecting the code for correctness (white-box), and testing parts as they were built (incremental).

22. IMPLEMENTATION

The implementation phase is where the designs and plans were turned into the actual, working Art Sale Management System website. This involved setting up the necessary environment and writing the code using the chosen technologies (PHP, MySQL, HTML, CSS, JavaScript).

A. Environment Setup:

- **Local Server:** A local development environment was set up, likely using a package like XAMPP, WAMP, or MAMP. This package conveniently provides the Apache web server, the MySQL database server, and the PHP interpreter needed to run the project on a personal computer.
- **Database Creation:** The art_gallery database was created within the MySQL server.
- **Table Creation:** The necessary database tables, as designed earlier, were created using SQL commands. This included:
  - The users table with columns for user ID, username, and the hashed password. Constraints like a primary key on user ID and a unique key on username were likely set.
  - The contact_submissions table with columns for submission ID (primary key), name, email, phone, message, filename, and submission timestamp.
- **Project Folder:** A folder for the project was created within the web server's document root (e.g., htdocs for XAMPP).

B. Back-End Implementation (PHP & MySQL):

- **Database Connection (config.php):** The config.php file was created to store the database connection details and establish the connection using mysqli. This file was then included (require) in other PHP scripts needing database access.
- **User Authentication Scripts:**
  - register.php: PHP code was written to handle POST requests from the registration form. It included logic to check if the username already exists (querying the DB), hash the password using password_hash(), and insert the new user record into the users table using a prepared statement for security.
  - login.php: PHP code was implemented to handle login form submissions. It queried the users table for the entered username (using a prepared statement), fetched the stored hash, and used password_verify() to compare it with the submitted password. Upon success, it initiated a session using session_start() and set the $_SESSION['logged_in_user'] variable.
  - logout.php: A simple script was created to start the session, destroy it using session_destroy(), and redirect the user to the login page.
- **Page Access Control:** PHP code was added at the beginning of protected pages like homepage.php to check if $_SESSION['logged_in_user'] is set, redirecting to login.php if not.
- **Submission Viewing (view_submission.php):** PHP code was written to connect to the database (should use config.php), execute a SELECT query on the contact_submissions table, fetch the results, and loop through them to display the data in an HTML table. htmlspecialchars() was used for secure output.

C. Front-End Implementation (HTML, CSS, JavaScript):

- **HTML Structure:** The basic HTML structure (including forms, headings, paragraphs, divs, images, links, navigation, footer) was created for each PHP file (login.php, register.php, homepage.php, gallerypage.php, artistpage.php, cart.php, checkout.php, success.php, view_submission.php).
- **CSS Styling:** CSS rules were written (primarily embedded within <style> tags) to style the HTML elements, implementing the visual design, layout (using Flexbox, etc.), colors, fonts, background images, and responsive adjustments using media queries.
- **JavaScript Functionality:**
  - **Slider:** JavaScript code was added to homepage.php to handle the image slider logic (changing slides automatically with setInterval and manually with button clicks).

    - **Gallery Filters:** The filterArt() JavaScript function was implemented in gallerypage.php to show/hide art items based on CSS classes when filter buttons are clicked.
    - **Shopping Cart:** JavaScript functions (addToCart, loadCart, updateQuantity, removeItem, updateCartCount, proceedToCheckout) were written and included in relevant pages (gallerypage.php, cart.php, checkout.php). These functions handle all client-side cart operations using localStorage for storage and sessionStorage for checkout transfer, and dynamically update the HTML of the cart page and header counter.
    - **Checkout UI:** JavaScript was added to checkout.php to dynamically update the payment information display based on the dropdown selection.

D. Integration:
Throughout the process, the front-end and back-end components were integrated. HTML forms were set up to POST data to the correct PHP scripts. PHP scripts generated HTML dynamically where needed (though limited in this version) or included HTML structures. JavaScript functions were triggered by user actions on HTML elements (like button clicks) and interacted with the DOM (Document Object Model) to update the page display.
This implementation phase brought together the planned design and chosen technologies to create the functional Art Sale Management System application.

23. A DATABASE DRIVEN ONLINE SHOPPING PORTAL

The Art Sale Management System functions as an online portal where key aspects are driven by a database, specifically MySQL. While some features like the current shopping cart rely heavily on client-side browser storage (localStorage), the integration of the database for core functionalities makes the system robust, persistent, and scalable in critical areas.

How the Database Drives the Portal:

- **User Account Persistence:** The most significant database-driven feature is user management. Without the MySQL database storing user information in the users table, the concepts of registration and login would not be possible. The database provides a central, secure place to:
    - Store usernames.
    - Store securely hashed passwords, preventing plain-text exposure.
    - Verify user credentials during login attempts.
      This persistent storage ensures that user accounts exist independently of any single browser session.

- **Centralized Contact Submissions:** The contact_submissions table acts as a central repository for all inquiries or messages sent through the website's contact form (implied). Using the database means:
    - Submissions are not lost if an email fails or is accidentally deleted.
    - Information can be easily retrieved and viewed later (view_submission.php).
    - Data like submission time (submittedAt) is automatically recorded.
    - File uploads associated with submissions can be tracked via the stored filename. This provides a much more organized and reliable way to handle user communication compared to simple email forms.
- **Foundation for Dynamic Content:** Although the current gallery and artist pages might use hardcoded data, the underlying structure (using PHP and having a database connection via config.php) makes the portal *ready* to be fully database-driven. In future versions:
    - Artwork details (name, description, price, image path, category, artist link) could be stored in an artworks table. The gallerypage.php could then dynamically query and display this data, making updates easy without touching the code.
    - Artist details could be stored in an artists table and displayed dynamically on artistpage.php.
      This capability for dynamic content, enabled by the database connection, is a hallmark of database-driven websites.
- **Scalability:** Storing user and submission data in a database allows the system to handle a growing amount of information much more efficiently than storing it in flat files or relying solely on client-side solutions. Databases are designed for efficient querying and management of large datasets.

Contrast with Client-Side Elements:

It's important to note the shopping cart currently uses localStorage. This makes the cart user-specific and persistent on their browser but is not centrally stored or managed by the database. While functional for this project's scope, a fully database-driven shopping portal would typically store cart contents and especially finalized orders in database tables, linked to user accounts. This enhances reliability, allows users to access their cart/orders from different devices, and is essential for secure server-side processing of payments and order fulfillment.

Conclusion:

In summary, the Art Sale Management System leverages a MySQL database for essential functions like secure user authentication and persistent storage of contact submissions. This database integration makes it more powerful and reliable than a purely static website and provides the necessary foundation for future expansion into a fully dynamic, database-driven e-commerce platform for art.

24. ABOUT THE CURRENT SYSTEM

This section provides a summary of the Art Sale Management System as it stands upon completion of this phase of development. It highlights the features that are currently functional and acknowledges areas that were implemented using simplified approaches or have limitations.

Implemented Features:

- **Secure User Authentication:** The system includes fully functional user registration (register.php) and login (login.php) capabilities. User passwords are securely hashed using password_hash() and verified using password_verify(). PHP sessions are used to manage login state, and access control is implemented on the main homepage (homepage.php) to ensure only logged-in users can view it. A logout (logout.php) function correctly destroys the user session.
- **Content Display Pages:**
  - A homepage (homepage.php) serves as a landing page for logged-in users, featuring navigation and an interactive image slider.
  - A gallery page (gallerypage.php) displays artworks (currently from a hardcoded PHP array) in a grid layout. It includes working category filters implemented with JavaScript.
  - An artist page (artistpage.php) displays artist information (also from a hardcoded array).
- **Client-Side Shopping Cart:** A functional shopping cart system is implemented using JavaScript and the browser's localStorage. Users can add items from the gallery, view the cart (cart.php), change item quantities, and remove items. The cart contents persist across page loads and browser sessions for that specific user/browser. The cart counter in the header updates dynamically.
- **Simulated Checkout:** A checkout process (checkout.php) is implemented. It retrieves cart data from sessionStorage (passed from cart.php), displays an order summary, allows the user to select a (simulated) payment method, and redirects to a success page (success.php).
- **Contact Submission Storage & Viewing:** A database table (contact_submissions) exists to store data from a contact form (the form itself

and its processing script are implied). A page (view_submission.php) allows viewing these stored submissions, including links to download associated files (if uploaded).

Technology Stack Summary:
The system is built using standard web technologies: PHP for server-side logic and database interaction, MySQL for data storage (users, submissions), HTML for structure, CSS for styling (embedded), and JavaScript for client-side interactivity (slider, filters, cart).

Current Limitations:

- **Static Content:** Artwork and artist data are hardcoded within the PHP files, not dynamically loaded from the database. This makes updating content require code changes.
- **Client-Side Cart Vulnerabilities:** The shopping cart logic and pricing information reside entirely in the user's browser (localStorage). This is convenient but insecure for real transactions, as prices could potentially be manipulated by a savvy user.
- **Simulated Checkout:** The checkout process does not involve actual payment processing or server-side order recording/validation.
- **Submission Viewer Security:** The view_submission.php page currently lacks any authentication, making stored contact information potentially accessible to anyone who knows the URL.
- **Basic Feedback:** User feedback primarily relies on JavaScript alert() boxes, which can be intrusive.

Conclusion on Current State:
The Art Sale Management System, in its current form, is a functional web application demonstrating core features like user management, content display, and a client-side e-commerce flow simulation. It successfully integrates front-end and back-end technologies and provides a solid foundation, while also highlighting specific areas (like database-driven content, server-side cart/checkout, and enhanced security) for future development to create a production-ready system.

25. ADVANTAGES

The implemented Art Sale Management System, even in its current state, offers several significant advantages compared to having no online presence or relying solely on manual methods:

- **Increased Accessibility and Reach:** The website is accessible 24/7 from anywhere with an internet connection. This breaks down geographical barriers, allowing potential buyers from different locations to discover the gallery and its artists. It also provides convenience for users who can browse at any time that suits them, unlike a physical gallery with fixed hours.

- **Centralized Information Hub:** The website acts as a single point of information for users. They can find details about the gallery, browse artworks, learn about artists (even if currently basic), manage items they're interested in (via the cart), and find contact information, all in one place. This presents a more professional and organized image.
- **Enhanced User Experience:** Features like the homepage image slider, the interactive gallery filters, and the dynamic shopping cart provide a more engaging and user-friendly experience than static web pages or traditional browsing methods. Users can easily filter content and manage their selections.
- **Secure User Authentication:** The implementation of user registration and login with secure password hashing (password_hash, password_verify) and PHP sessions provides a fundamental level of security for user accounts. This protects user information and allows for features restricted to logged-in members.
- **Organized Communication Channel:** Storing contact form submissions in the contact_submissions database table offers a more reliable and organized way to manage inquiries compared to potentially losing track of emails. Having a timestamp (submittedAt) also helps track communication history.
- **Persistent Shopping Cart (Client-Side):** The use of localStorage for the shopping cart allows users to add items, leave the site, and find their items still in the cart when they return (using the same browser). This convenience can encourage users to complete purchases later.
- **Foundation for Scalability and Growth:** The modular structure (separate PHP files) and the use of standard technologies (PHP, MySQL) create a solid foundation. It's easier to add new features, integrate more database-driven content (like artworks/artists), or implement a full server-side e-commerce system in the future compared to starting from scratch or using less structured approaches.
- **Cost-Effectiveness (Compared to Physical Expansion):** Establishing an online presence is generally more cost-effective than opening additional physical gallery locations to achieve similar reach.

While acknowledging the current limitations (like the client-side cart and static content), these advantages demonstrate the value the Art Sale Management System brings as an initial online platform for The Art Gallery.

26. DEFICIENCY OF MANUAL SYSTEM

Comparing the implemented online Art Sale Management System to a purely manual system (e.g., a physical gallery relying only on walk-ins, phone calls, and paper records) highlights several significant deficiencies of the manual approach:

- **Limited Accessibility and Reach:** A manual system is inherently restricted by physical location and operating hours. Potential buyers must physically visit the gallery during specific times, limiting the customer base significantly. There's no easy way to reach audiences in different cities or countries, hindering growth and artist exposure.
- **Inefficient Inventory Management:** Keeping track of artworks, their details (artist, price, availability), and sales status manually using logbooks or basic spreadsheets is prone to errors. It's easy to make mistakes like selling an already sold piece, misquoting prices, or losing track of inventory, leading to customer dissatisfaction and administrative headaches.
- **Slow and Disorganized Communication:** Handling customer inquiries via phone or mail is slow and difficult to manage centrally. There's a higher risk of missed calls, lost messages, or delayed responses. Tracking communication history with specific customers or about particular artworks becomes very challenging.
- **Poor Information Discovery for Buyers:** Buyers visiting a physical gallery have limited ways to quickly find specific items. They cannot easily filter by category, price range, or artist without assistance. Comparing pieces requires physically moving around and relying on memory or notes. Accessing detailed artist information might require asking staff directly.
- **Lack of Data for Insights:** Manual systems make it extremely difficult to gather and analyze data about customer behavior, popular artists or styles, sales trends, or peak visiting times. This lack of data hinders informed decision-making regarding inventory, marketing, and gallery strategy.
- **Higher Potential for Errors in Transactions:** Manual calculation of totals, handling of payments (if only cash/check), and recording sales increases the likelihood of human error compared to automated (even simulated) online calculations and future potential for integrated payment systems.
- **Scalability Issues:** As the gallery grows (more artists, more artworks, more customers), a manual system becomes increasingly difficult and costly to manage effectively. The paperwork, communication overhead, and tracking requirements can quickly become overwhelming.

These deficiencies demonstrate that while physical galleries have their place, relying solely on manual systems in today's digital age puts the gallery at a disadvantage in terms of reach, efficiency, customer experience, and growth potential. The online system directly addresses many of these shortcomings.

27. GOALS OF PROPOSED SYSTEM

The development of the Art Sale Management System was driven by a set of clear goals aimed at addressing the identified problems and leveraging the advantages of an online platform. The primary objectives were:

- **Establish an Accessible Online Presence:** To create a professional and functional website for "The Art Gallery" that is available 24/7 to users worldwide, overcoming the limitations of a physical-only location.
- **Implement Secure User Management:** To build a reliable system for users to register for accounts and log in securely. This involved using industry-standard password hashing techniques to protect user credentials and employing PHP sessions to manage user login status across the site.
- **Facilitate Art Discovery and Browsing:** To provide a user-friendly interface where visitors can easily view the gallery's collection of artworks (gallerypage.php). This included implementing features like categorization and filtering to help users find pieces that match their interests. Displaying basic artist information (artistpage.php) was also a goal.
- **Provide E-commerce Functionality (Simulation):** To create a working shopping cart mechanism allowing users to select artworks, manage quantities, and remove items. The goal was also to simulate a checkout process (cart.php, checkout.php), providing users with an order summary and payment options, culminating in a success confirmation (success.php), even without real transaction processing in this phase.
- **Streamline Communication:** To offer an online method for users to contact the gallery (implied contact.php) and to store these submissions reliably in a database (contact_submissions table) for easier management and review (view_submission.php).
- **Build a Maintainable and Scalable Foundation:** To develop the system using standard web technologies (PHP, MySQL, HTML, CSS, JS) and a modular structure (separate files for different functions). This approach aimed to make the current system easier to understand and maintain, while also providing a solid base for adding more complex features or scaling the application in the future.

Achieving these goals was intended to result in a valuable online tool for The Art Gallery, enhancing its reach, improving user experience for both buyers and artists (implicitly), and creating a more efficient operational model compared to purely manual methods.

28. USER REQUIREMENTS

User requirements define what the Art Sale Management System needs to do to be useful and effective for the people using it (both regular users/buyers and potential administrators). These can be broken down into what the system *does* (Functional) and *how well* it does it (Non-Functional).

A. Functional Requirements (What the system must DO):

- **User Account Management:**
  - The system **must** allow new users to register with a unique username and password.
  - The system **must** allow registered users to log in using their correct credentials.
  - The system **must** prevent access to protected pages (like the homepage) for users who are not logged in.
  - The system **must** provide a way for logged-in users to log out.
- **Browsing and Viewing:**
  - The system **must** display a gallery of artworks with relevant details (image, name, price).
  - The system **must** allow users to filter the gallery based on artwork category.
  - The system **must** display information about artists associated with the gallery.
- **Shopping Cart and Checkout:**
  - The system **must** allow users to add selected artworks to a shopping cart.
  - The system **must** display the current contents of the shopping cart.
  - The system **must** allow users to change the quantity of items in the cart.
  - The system **must** allow users to remove items from the cart.
  - The system **must** persist the cart contents for the user (achieved via localStorage).
  - The system **must** present an order summary based on the cart contents during checkout.
  - The system **must** allow users to select a payment method during checkout (simulated).
  - The system **must** display a confirmation message upon completion of the simulated checkout.
- **Communication:**
  - The system **must** provide a mechanism for users to submit inquiries or messages (implied contact form).

    - The system **must** store these submitted messages and associated details in the database.
    - The system **must** allow an authorized administrator to view the stored submissions (view contact_submissions table).

B. Non-Functional Requirements (Qualities the system should HAVE):

- **Security:**
    - User passwords **must** be stored securely using hashing techniques, not plain text.
    - The login process **should** be protected against common attacks (e.g., using prepared statements helps prevent SQL injection).
    - Access to sensitive information (like contact submissions) **should** be restricted to authorized users (*Note: This needs improvement in view_submission.php*).
- **Usability:**
    - The website **should** be easy to navigate and understand for typical users.
    - Forms (login, register) **should** be straightforward to complete.
    - Buttons and links **should** be clearly labeled and function as expected.
- **Performance:**
    - Web pages **should** load within a reasonable time frame.
    - Client-side interactions (like filtering, cart updates) **should** feel responsive.
- **Reliability:**
    - The system **should** operate consistently without frequent errors or crashes during normal use.
- **Compatibility:**
    - The website **should** display and function correctly on major, up-to-date web browsers (e.g., Chrome, Firefox, Edge).
- **Responsiveness:**
    - The website layout **should** adapt appropriately to different screen sizes (desktops, tablets, smartphones) for a good viewing experience on various devices.

These requirements served as a checklist during development and testing to ensure the final system met the intended goals and provided a functional, usable, and reasonably secure experience within the project's scope.

29. LIMITATIONS

While the Art Sale Management System successfully implements its core features, it's important to acknowledge the limitations present in this version of the project. These limitations represent areas where the system could be improved or expanded in future development phases.

- **Client-Side Cart and Pricing:** The most significant limitation relates to the shopping cart. Because the cart data, including item prices and quantities, is stored and managed entirely within the user's browser (localStorage), it is potentially vulnerable. A user with technical knowledge could theoretically modify this data before checkout, potentially altering prices. This makes the current cart implementation unsuitable for processing real financial transactions where price integrity is paramount.
- **Simulated Checkout Process:** The current checkout flow (checkout.php) is a simulation. It displays an order summary and allows payment method selection, but it does not connect to any real payment gateway, nor does it validate the order details server-side or store the completed order in the database. It simply redirects to a success page. Therefore, no actual sales are processed or recorded centrally.
- **Static Gallery and Artist Content:** The artworks displayed in the gallery (gallerypage.php) and the information about artists (artistpage.php) are currently hardcoded directly into the PHP files as arrays. This means that adding, removing, or modifying artworks or artists requires direct code changes by a developer. The system lacks an administrative interface or database integration for managing this content dynamically, limiting the ease of updating the gallery's offerings.
- **Insecure Submission Viewing:** The view_submission.php page, which displays potentially sensitive contact information and messages submitted through the (implied) contact form, currently has no authentication or authorization checks. This means anyone who knows the URL could potentially access all submitted data, posing a significant privacy and security risk. Access should be restricted to authorized administrators only.
- **Basic User Feedback:** The system primarily uses JavaScript alert() popups for feedback (e.g., login errors, registration success). While functional, this method can be disruptive to the user experience compared to displaying messages integrated directly within the web page layout.
- **Lack of Administrative Features:** Beyond the insecure submission viewing page, there is no dedicated administrative panel. Features for managing users (e.g., deleting accounts, resetting passwords), managing gallery content (if it were dynamic), or viewing site statistics are not included in the current scope.

- **Limited Error Handling:** While basic checks exist (e.g., database connection), the error handling throughout the application could be more robust to gracefully handle unexpected situations like database query failures or invalid user inputs beyond basic form requirements.

Acknowledging these limitations is important for understanding the current state of the project and identifying key areas for enhancement in future iterations to build a more secure, robust, and fully-featured online art gallery platform.

30. PROJECT LEGACY (Future Scope)

The current Art Sale Management System serves as a strong foundation, but there are many potential ways it could be enhanced and expanded in the future to create an even more powerful and complete online gallery platform. This section outlines some key areas for future development, representing the project's potential legacy.

- **Database-Driven Content Management:**
  - **Suggestion:** Move all artwork and artist information from hardcoded PHP arrays into dedicated MySQL database tables (e.g., artworks, artists). Modify gallerypage.php and artistpage.php to dynamically query and display this data.
  - **Benefit:** This would make the gallery content easily updatable without code changes, likely via an admin panel, and allow for a much larger, more manageable collection.
- **Server-Side Shopping Cart and Order Processing:**
  - **Suggestion:** Re-implement the shopping cart logic on the server-side, perhaps storing cart contents in the user's session or a dedicated database table linked to the user ID. Crucially, implement server-side validation of item prices during the checkout process. Store completed orders (including user details, items purchased, quantities, prices, total amount, order status) in dedicated orders and order_items tables in the database.
  - **Benefit:** This drastically improves security by preventing client-side price manipulation and provides a reliable, central record of all placed orders. It also allows users to potentially view their order history.
- **Payment Gateway Integration:**
  - **Suggestion:** Integrate a real payment gateway (like Stripe, PayPal, Razorpay, or others) into the checkout.php process. This would involve handling secure payment authorization, processing transactions, and updating the order status in the database based on payment success or failure.
  - **Benefit:** Enables the website to handle actual financial transactions securely,

turning the simulation into a real e-commerce platform.

- **Secure Admin Dashboard:**
  - **Suggestion:** Develop a comprehensive, password-protected administrative area. This dashboard should require a separate admin login and provide tools to:
    - Manage gallery content (add/edit/delete artworks and artists - requires database-driven content).
    - Manage users (view registered users, potentially disable accounts or reset passwords).
    - Securely view and manage contact submissions (replacing the insecure view_submission.php).
    - View and manage orders (once server-side orders are implemented).
  - **Benefit:** Provides essential tools for managing the website and its data securely and efficiently.
- **Artist Portal:**
  - **Suggestion:** Introduce functionality allowing registered artists to submit their own artwork for inclusion in the gallery. This could involve forms for uploading images, adding descriptions, setting prices, and managing their profiles, potentially requiring admin approval before items go live.
  - **Benefit:** Empowers artists, reduces administrative workload for gallery staff, and potentially increases the variety of art available.
- **Enhanced User Experience (UX):**
  - **Suggestion:** Replace JavaScript alert() messages with more integrated, user-friendly notifications displayed directly on the page. Implement features like artwork search, advanced filtering/sorting options, user profiles with order history, and potentially a wishlist feature. Improve error handling and provide clearer messages to users.
  - **Benefit:** Creates a smoother, more professional, and more engaging experience for website visitors.
- **Improved Security:**
  - **Suggestion:** Conduct thorough security audits. Implement measures against common web vulnerabilities beyond basic password hashing and prepared statements (e.g., Cross-Site Request Forgery - CSRF protection, more robust input validation everywhere, secure file upload handling). Ensure the view_submission.php functionality is integrated into the secure admin dashboard.
  - **Benefit:** Protects user data, gallery data, and the integrity of the platform.

Implementing these future enhancements would build upon the current project's legacy, transforming it from a functional prototype into a robust, secure, and fully-featured online art sale management system.

31. USER MANUAL

This manual provides basic instructions for using the key features of the Art Sale Management System website.

A. Accessing the Site:

1. Open a modern web browser (like Chrome, Firefox, Edge).
2. Navigate to the website's URL (e.g., http://localhost/your_project_folder/ if running locally, starting typically with login.php or potentially register.php).

B. Registering a New Account:

1. If you are on the Login page (login.php), click the "Register" link.
2. On the Registration page (register.php), enter your desired unique username in the "Username" field.
3. Enter your desired password in the "Password" field.
4. Click the "Register" button.
5. You should see a confirmation message (currently an alert). You will likely be redirected to the Login page.

C. Logging In:

1. Navigate to the Login page (login.php).
2. Enter your registered username in the "Username" field.
3. Enter your password in the "Password" field.
4. Click the "Login" button.
5. If successful, you will be redirected to the Homepage (homepage.php). If unsuccessful, you will likely see an error message (currently an alert).

D. Browsing Artworks:

1. Once logged in, use the navigation menu (usually in the header) and click on the "Gallery" link to go to gallerypage.php.
2. Scroll through the page to view the artworks displayed in a grid. Each item shows an image, name, and price.
3. Use the filter buttons at the top (e.g., "All", "Paintings", "Sculptures", "Digital Art") to view only artworks belonging to that category.

E. Viewing Artist Information:

1. Use the navigation menu and click on the "Artists" link to go to artistpage.php.
2. View the information presented about the different artists.

F. Using the Shopping Cart:

1. **Adding Items:** While on the Gallery page (gallerypage.php), click the "Add to Cart" button located below an artwork you wish to purchase. The cart counter in the header should update.
2. **Viewing the Cart:** Click the "Cart" link in the navigation menu to go to cart.php. Here you will see a list of all items you have added.
3. **Changing Quantity:** On the Cart page, use the + button next to an item to increase its quantity and the - button to decrease it. The quantity display will update.
4. **Removing Items:** Click the "Remove" button next to an item you no longer want in your cart. The item will be removed from the list.

G. Checking Out (Simulation):

1. When you are ready to proceed from the Cart page (cart.php), click the "Next" (or similarly named) button.
2. You will be taken to the Checkout page (checkout.php). Review the order summary which lists the items and total cost based on your cart.
3. Select a payment method (e.g., "Cash on Delivery", "UPI / Razorpay") from the dropdown list. Note that this is a simulation, and no real payment will be processed.
4. Click the "Proceed to Payment" or "Confirm Order" button.
5. You should be redirected to the Success page (success.php), confirming your simulated order.

H. Viewing Contact Submissions (Admin Task - Requires Caution):

1. **Note:** This page (view_submission.php) currently lacks security. Access should be restricted in a production environment.
2. Navigate directly to the view_submission.php URL in your browser (e.g., http://localhost/your_project_folder/view_submission.php).
3. The page will display a table listing all messages submitted through the contact form, including sender details and message content.
4. If a file was uploaded with a submission, a "Download File" link will be available. Clicking this link will download the associated file.

I. Logging Out:

1. While logged in, click the "Logout" link in the header navigation menu.
2. You will be logged out of your session and redirected back to the Login page (login.php).

These steps cover the primary ways to interact with the current version of the Art Sale Management System.

32. ABBREVIATIONS / ACRONYMS

This section defines common abbreviations and acronyms used within this report and related to the project's technologies.

- **CSS:** Cascading Style Sheets - *Used for styling the appearance of web pages.*
- **DB:** Database - *An organized collection of data.*
- **DFD:** Data Flow Diagram - *A diagram showing how data moves through a system.*
- **DOM:** Document Object Model - *The browser's internal representation of an HTML page, which JavaScript interacts with.*
- **ER / ERD:** Entity-Relationship / Entity-Relationship Diagram - *A diagram showing database tables (entities) and their relationships.*
- **FK:** Foreign Key - *A key in a database table used to link to the primary key of another table.*
- **HTML:** HyperText Markup Language - *The standard language for creating the structure of web pages.*
- **HTTP:** HyperText Transfer Protocol - *The set of rules for transferring files (text, images, sound, video, etc.) on the World Wide Web.*
- **IDE:** Integrated Development Environment - *Software application that provides comprehensive facilities to computer programmers for software development (e.g., VS Code).*
- **JS:** JavaScript - *A programming language used to make web pages interactive and dynamic.*
- **MySQL:** My Structured Query Language - *A popular open-source relational database management system.*
- **OS:** Operating System - *Software that manages computer hardware and software resources (e.g., Windows, macOS, Linux).*
- **PHP:** Hypertext Preprocessor - *A popular server-side scripting language used for web development.*
- **PK:** Primary Key - *A unique identifier for each record (row) in a database table.*
- **RAM:** Random Access Memory - *The computer's short-term memory.*
- **RDBMS:** Relational Database Management System - *A type of database system that stores data in related tables (like MySQL).*

- **SDLC:** Software Development Life Cycle - *The process or methodology followed for developing software.*
- **SQL:** Structured Query Language - *The standard language for interacting with databases.*
- **SSD:** Solid State Drive - *A type of storage device often faster than traditional hard drives.*
- **UI:** User Interface - *The visual part of a system that the user interacts with.*
- **UK:** Unique Key - *A constraint in a database table ensuring that all values in a column (or set of columns) are unique.*
- **UML:** Unified Modeling Language - *A standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.*
- **URL:** Uniform Resource Locator - *The address of a resource on the internet (a web address).*
- **UX:** User Experience - *The overall experience a person has when using a product, such as a website.*
- **WAMP:** Windows, Apache, MySQL, PHP - *A software stack bundle for Windows.*
- **XAMPP:** Cross-Platform, Apache, MariaDB/MySQL, PHP, Perl - *A popular, free, and open-source cross-platform web server solution stack package.*

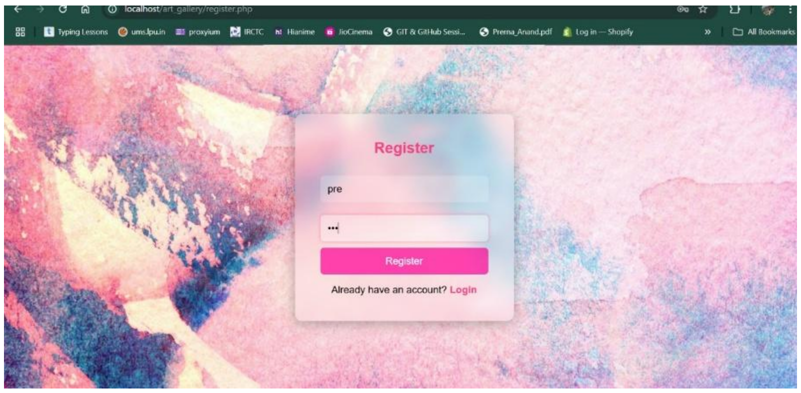
33. BIBLIOGRAPHY

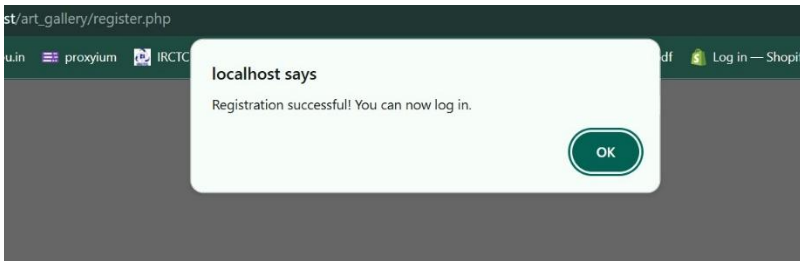
This section lists resources that are commonly used for learning and reference when developing web applications with the technologies used in this project. While specific books or articles used might vary, these online resources are standard references:

- **PHP:**
  - **PHP Official Manual:** The most comprehensive and accurate source for all PHP functions and language features.
    - URL: https://www.php.net/manual/en/
  - **W3Schools PHP Tutorial:** Beginner-friendly tutorials and examples covering core PHP concepts.
    - URL: https://www.w3schools.com/php/
- **MySQL:**
  - **MySQL Official Documentation:** The complete reference manual for MySQL database features and SQL syntax.
    - URL: https://dev.mysql.com/doc/
  - **W3Schools SQL Tutorial:** Covers fundamental SQL commands applicable to MySQL and other relational databases.
    - URL: https://www.w3schools.com/sql/

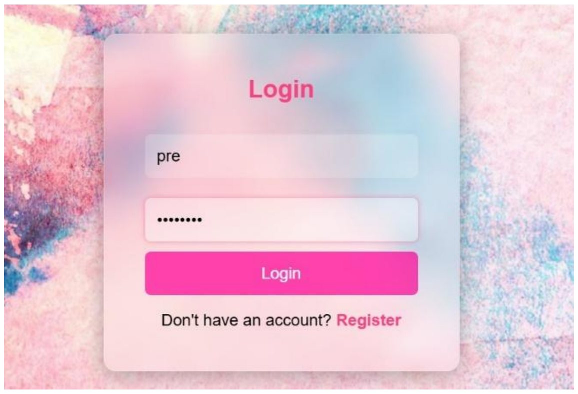
- **HTML:**
    - **MDN Web Docs - HTML:** Detailed documentation and guides on HTML elements and best practices from Mozilla.
        - URL: https://developer.mozilla.org/en-US/docs/Web/HTML
    - **W3Schools HTML Tutorial:** Interactive tutorials for learning HTML structure and elements.
        - URL: https://www.w3schools.com/html/
- **CSS:**
    - **MDN Web Docs - CSS:** Comprehensive reference for CSS properties, selectors, and concepts like Flexbox and Grid.
        - URL: https://developer.mozilla.org/en-US/docs/Web/CSS
    - **W3Schools CSS Tutorial:** Step-by-step tutorials covering CSS basics to advanced topics.
        - URL: https://www.w3schools.com/css/
    - **CSS-Tricks:** A popular website with articles, guides, and tips on modern CSS techniques and problem-solving.
        - URL: https://css-tricks.com/
- **JavaScript:**
    - **MDN Web Docs - JavaScript:** The definitive guide to the JavaScript language, including DOM manipulation and browser APIs like localStorage and sessionStorage.
        - URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript
    - **W3Schools JavaScript Tutorial:** Easy-to-follow tutorials for learning JavaScript fundamentals and practical examples.
        - URL: https://www.w3schools.com/js/
    - **JavaScript.info:** A detailed, modern tutorial covering JavaScript from basics to advanced topics.
        - URL: https://javascript.info/
- **General Development & Q&A:**
    - **Stack Overflow:** A large community question-and-answer site for programmers. An invaluable resource for finding solutions to specific coding problems and errors.
        - URL: https://stackoverflow.com/
- **Icon Library (if used):**
    - **Font Awesome Documentation:** Official guide for using the Font Awesome icon library (if it was used for icons in the project).
        - URL: https://fontawesome.com/docs

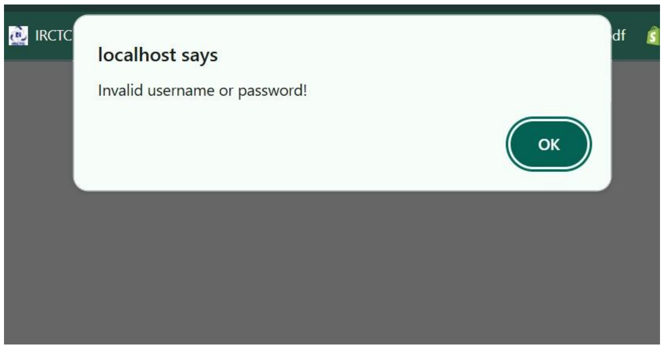
34. SNAP-SHOTS OF THE PROJECT


localhost/art_gallery/register.php
Register
pre
Register
Already have an account? Login

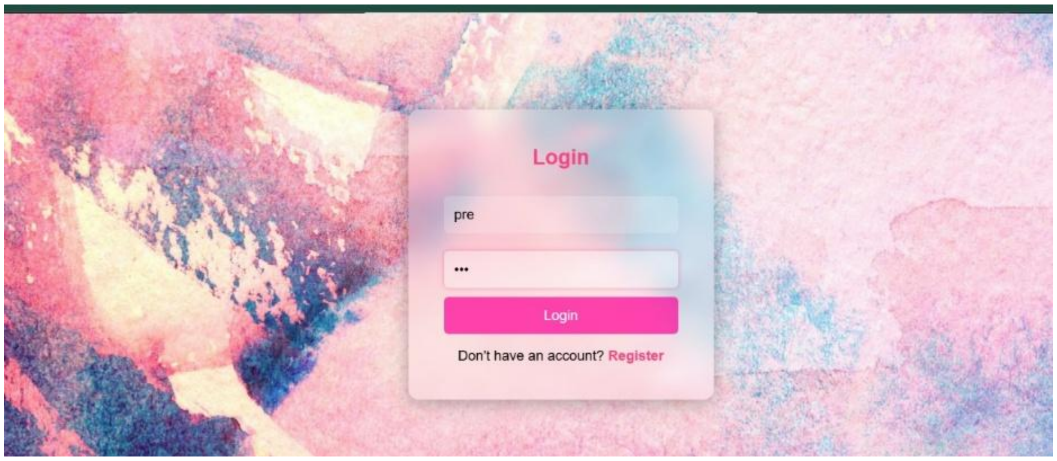

localhost says

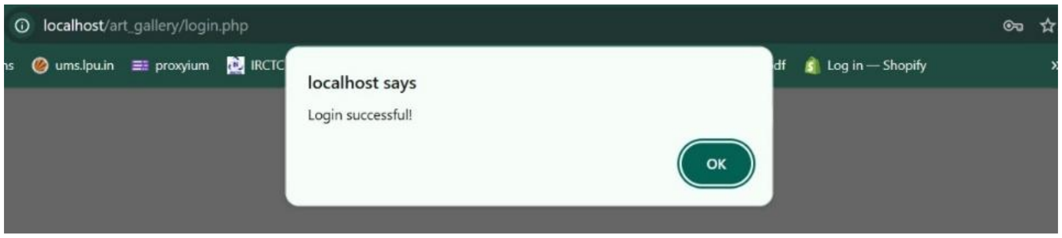
Registration successful! You can now log in.




localhost says

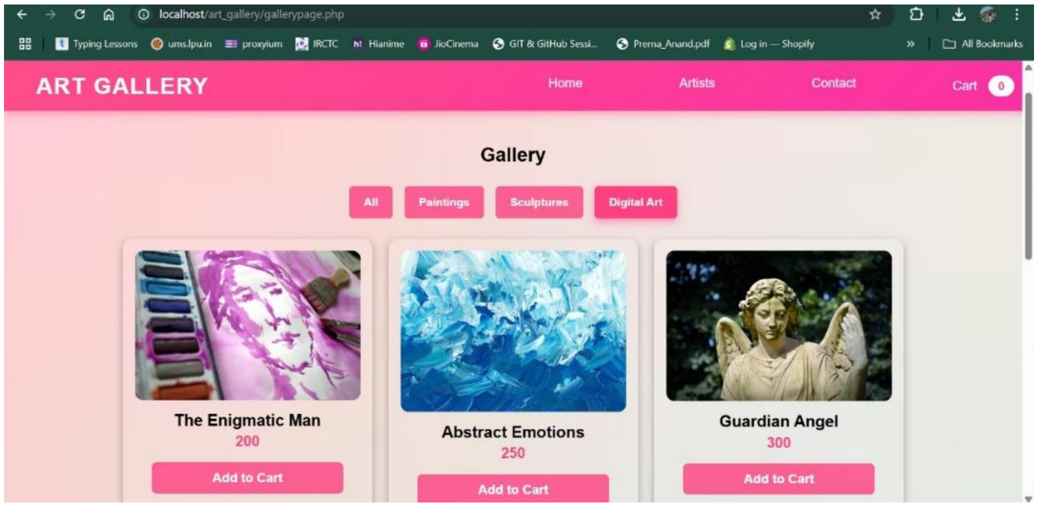
Invalid username or password!

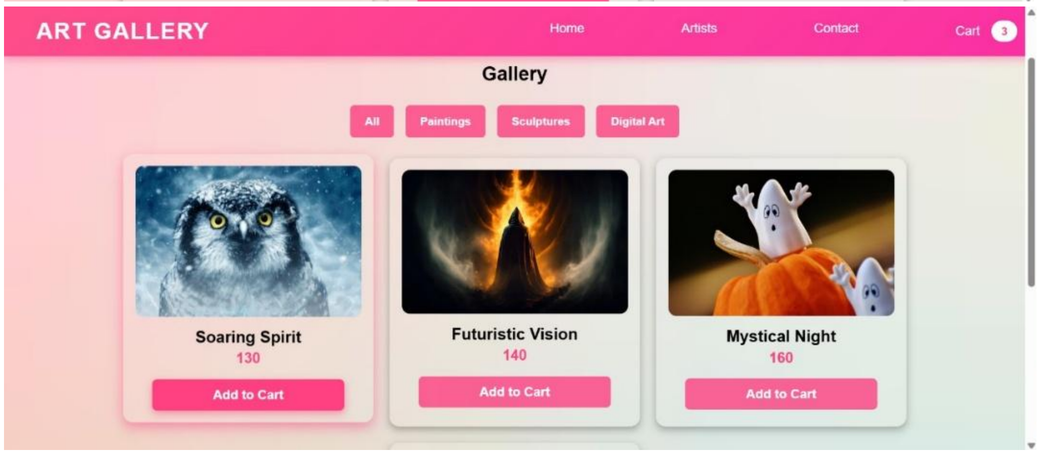


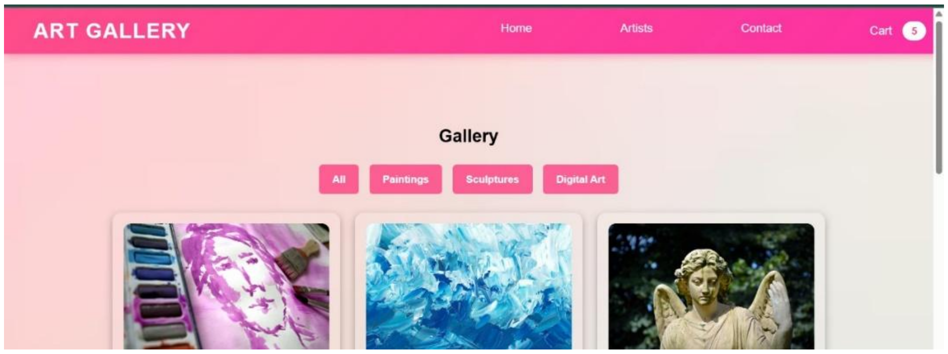


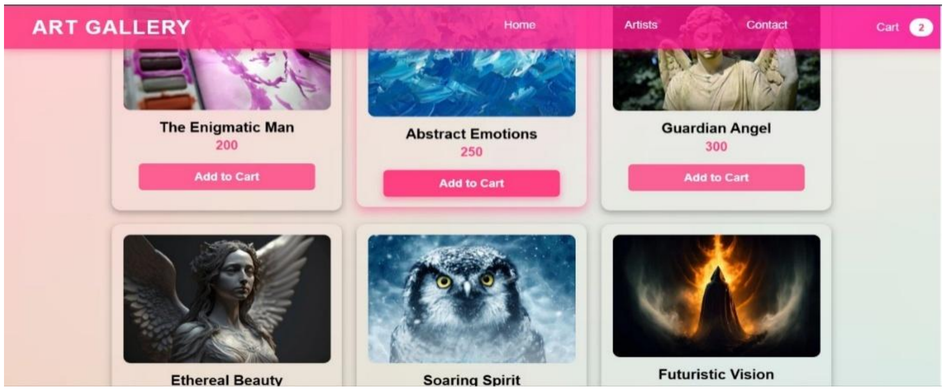
localhost says

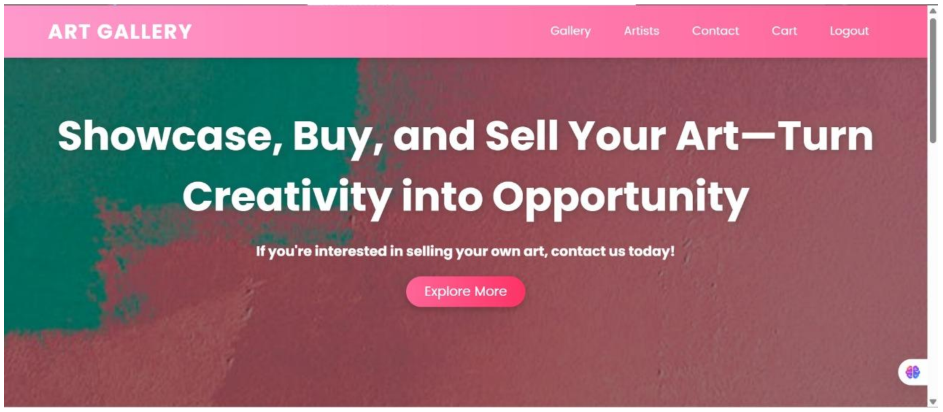
Login successful!





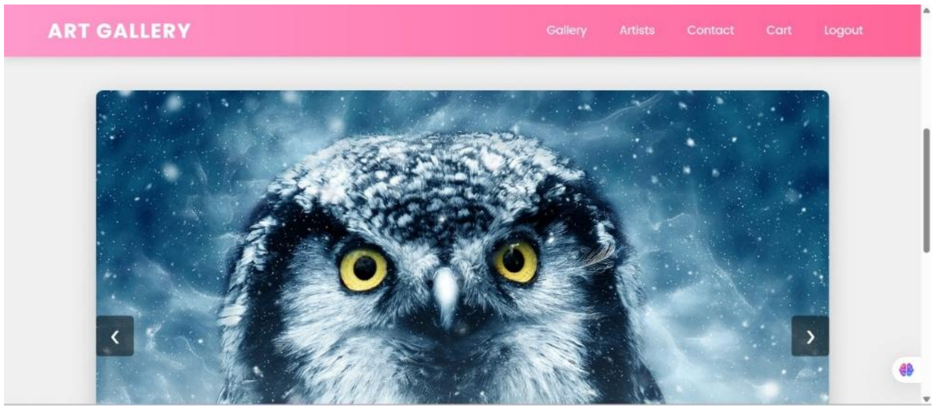


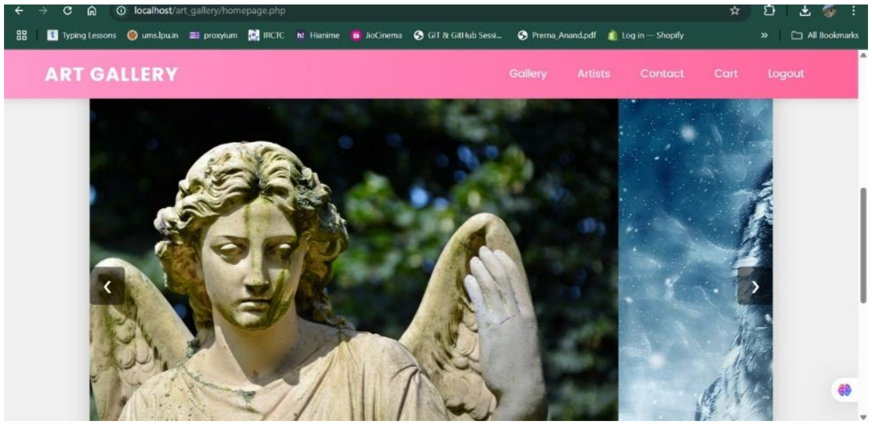


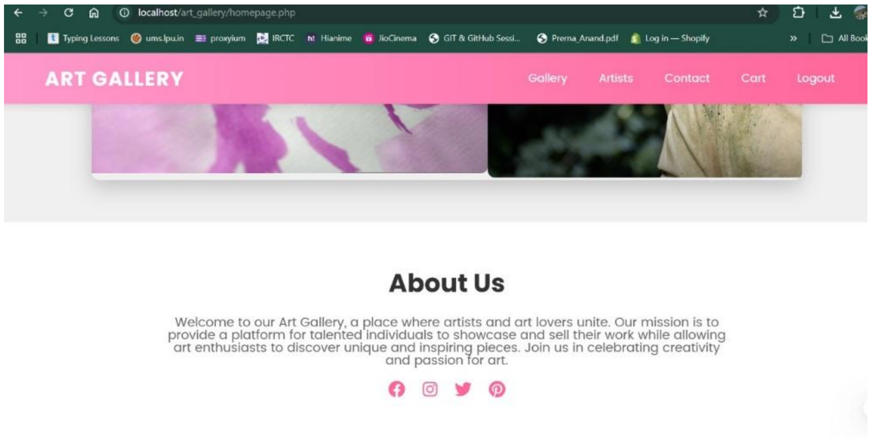


Showcase, Buy, and Sell Your Art—Turn Creativity into Opportunity

If you're interested in selling your own art, contact us today!

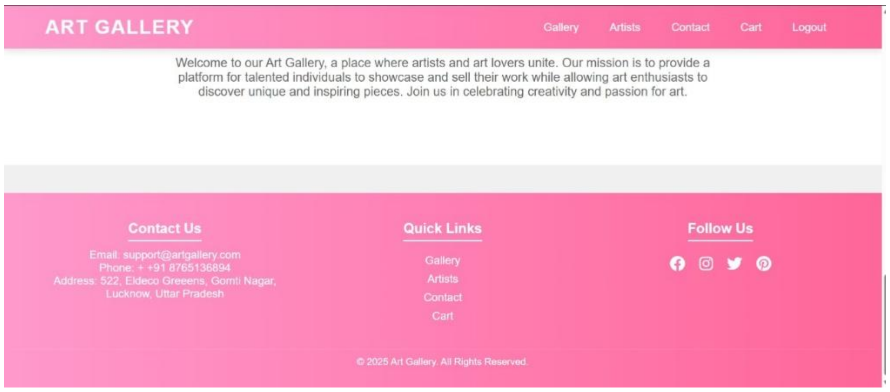






About Us

Welcome to our Art Gallery, a place where artists and art lovers unite. Our mission is to provide a platform for talented individuals to showcase and sell their work while allowing art enthusiasts to discover unique and inspiring pieces. Join us in celebrating creativity and passion for art.



Welcome to our Art Gallery, a place where artists and art lovers unite. Our mission is to provide a platform for talented individuals to showcase and sell their work while allowing art enthusiasts to discover unique and inspiring pieces. Join us in celebrating creativity and passion for art.

Contact Us

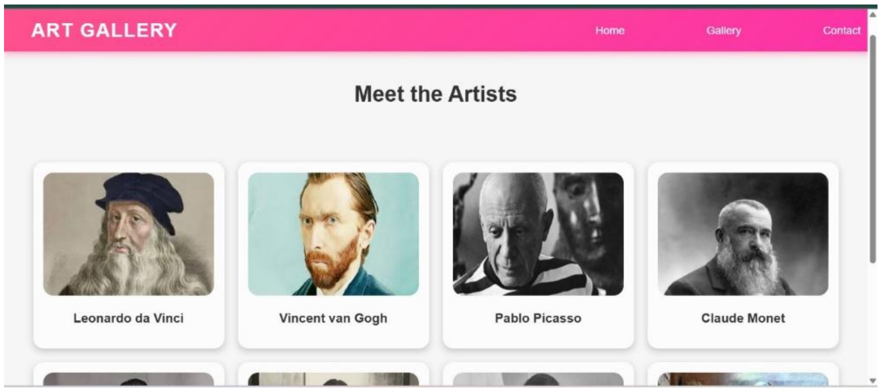
Email: support@artgallery.com
Phone: + +91 8765136894
Address: 522, Eldeco Greeens, Gomti Nagar, Lucknow, Uttar Pradesh

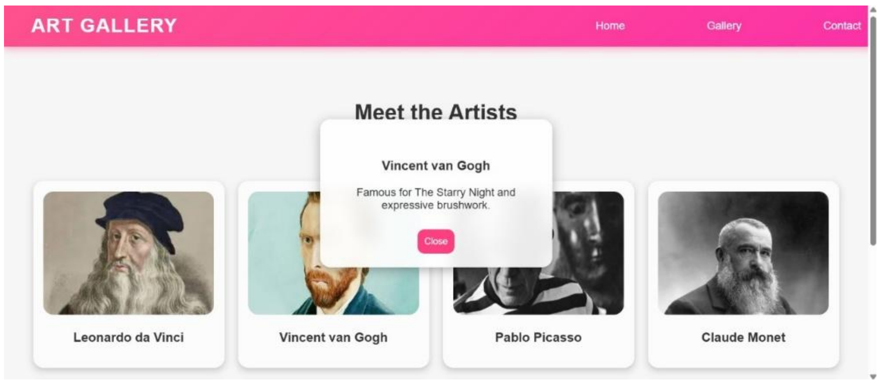
Quick Links

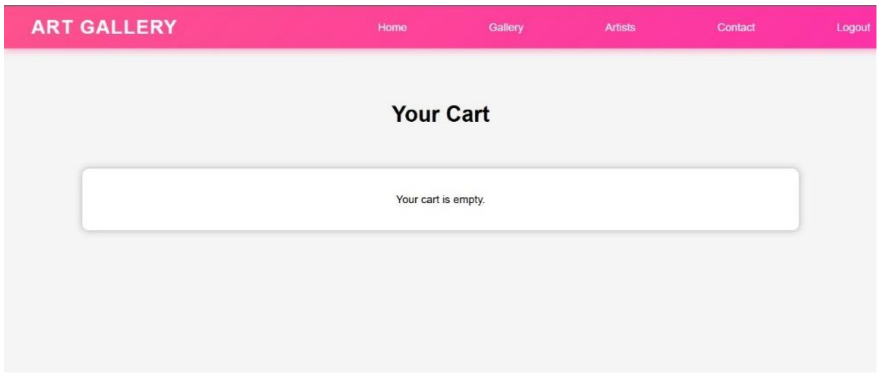
Gallery
Artists
Contact
Cart

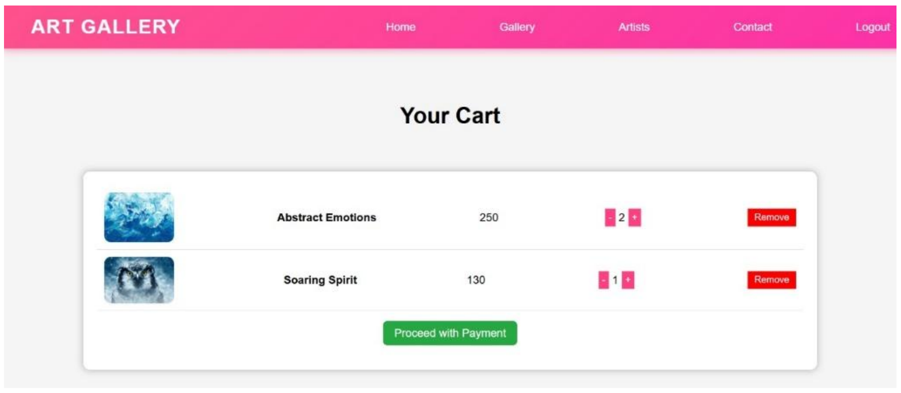
Follow Us

© 2025 Art Gallery. All Rights Reserved.

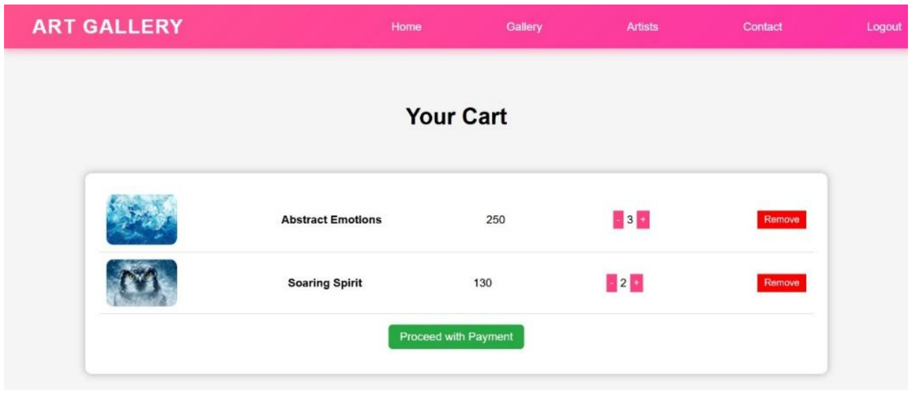






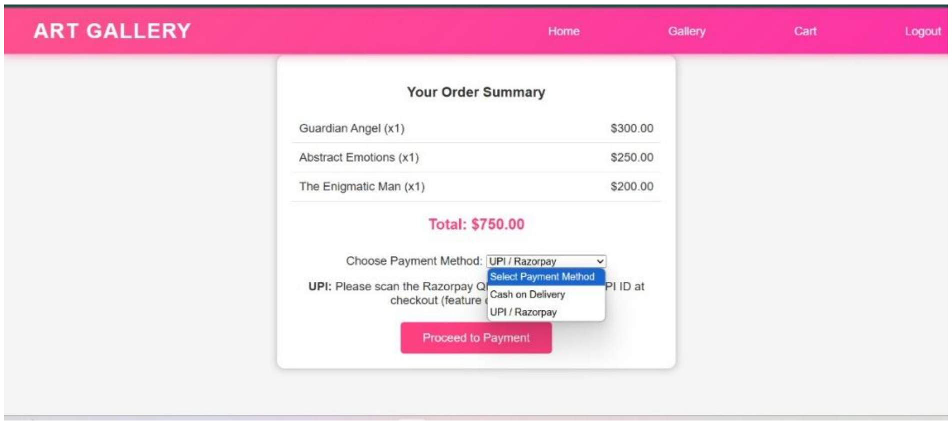


	Abstract Emotions	250	- 2 +	Remove
	Soaring Spirit	130	- 1 +	Remove



Your Cart

Abstract Emotions	250	- 3 +	Remove
Soaring Spirit	130	- 2 +	Remove



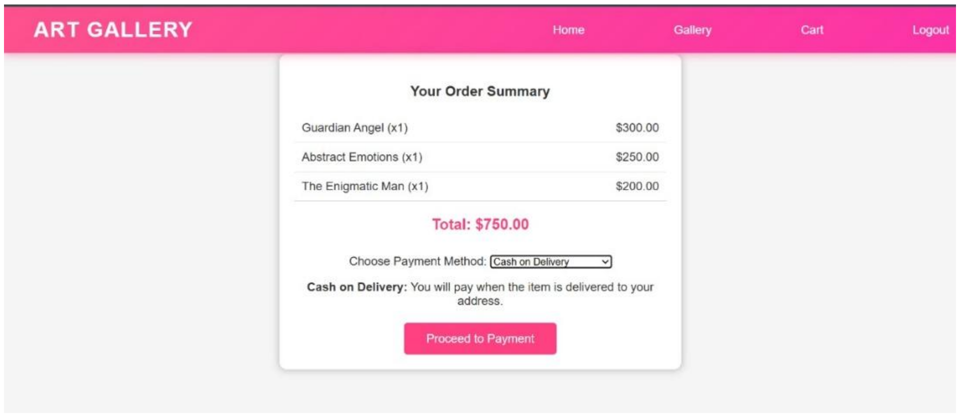
Your Order Summary

Guardian Angel (x1)	$300.00
Abstract Emotions (x1)	$250.00
The Enigmatic Man (x1)	$200.00

Total: $750.00

Choose Payment Method: UPI / Razorpay

UPI: Please scan the Razorpay Q PI ID at checkout (feature



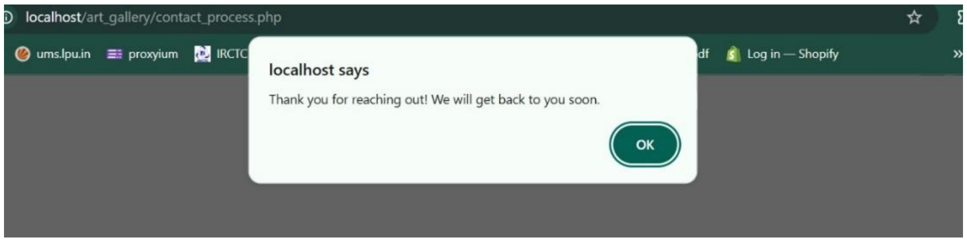
Your Order Summary

Guardian Angel (x1)	$300.00
Abstract Emotions (x1)	$250.00
The Enigmatic Man (x1)	$200.00

Total: $750.00

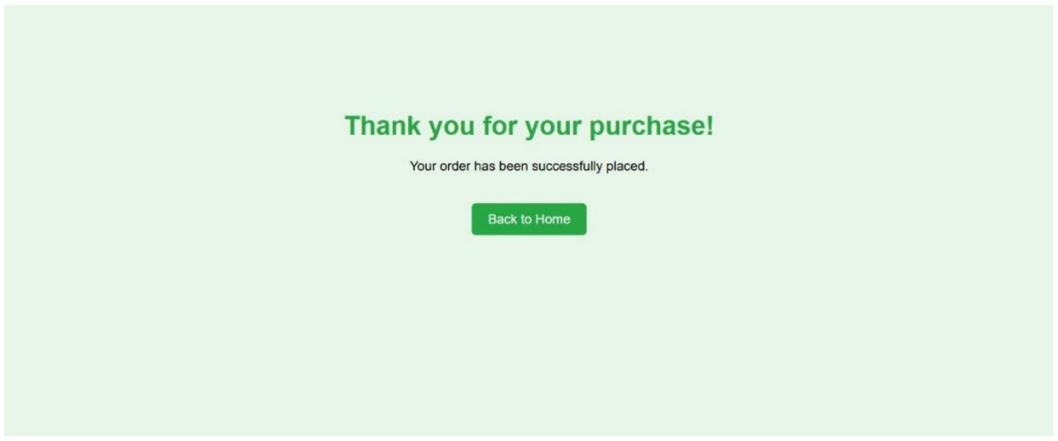
Choose Payment Method: Cash on Delivery

Cash on Delivery: You will pay when the item is delivered to your address.


localhost/art_gallery/contact_process.php

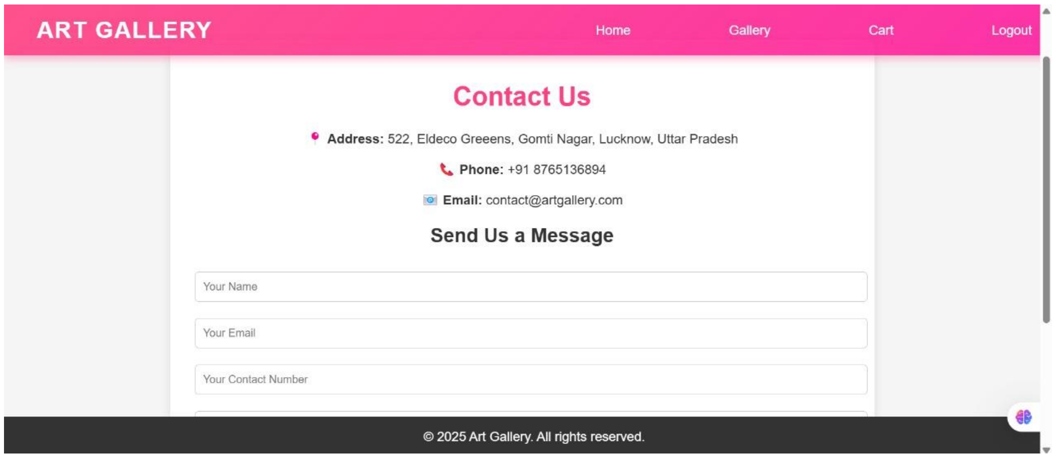
localhost says

Thank you for reaching out! We will get back to you soon.



Thank you for your purchase!

Your order has been successfully placed.



Contact Us

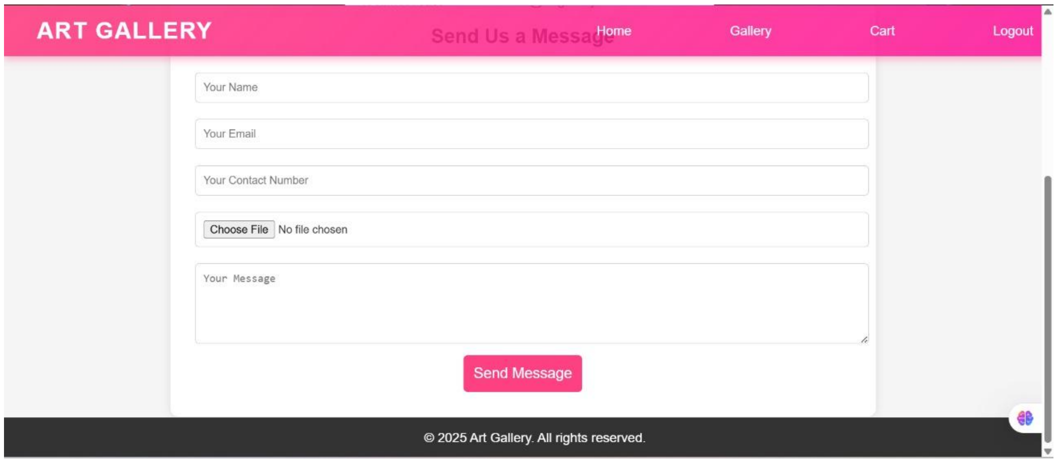
Address: 522, Eldeco Greeens, Gomti Nagar, Lucknow, Uttar Pradesh

Phone: +91 8765136894

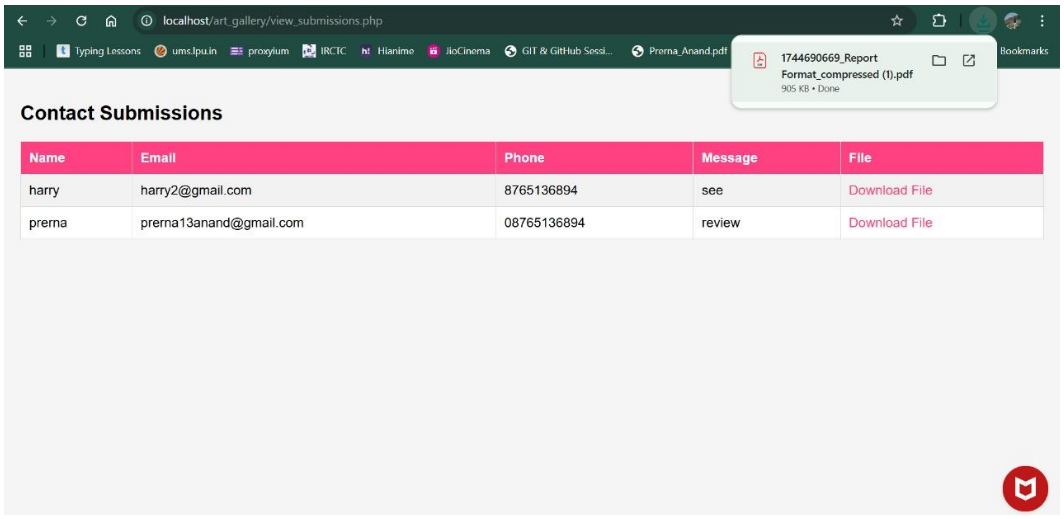
Email: contact@artgallery.com

Send Us a Message

© 2025 Art Gallery. All rights reserved.



© 2025 Art Gallery. All rights reserved.



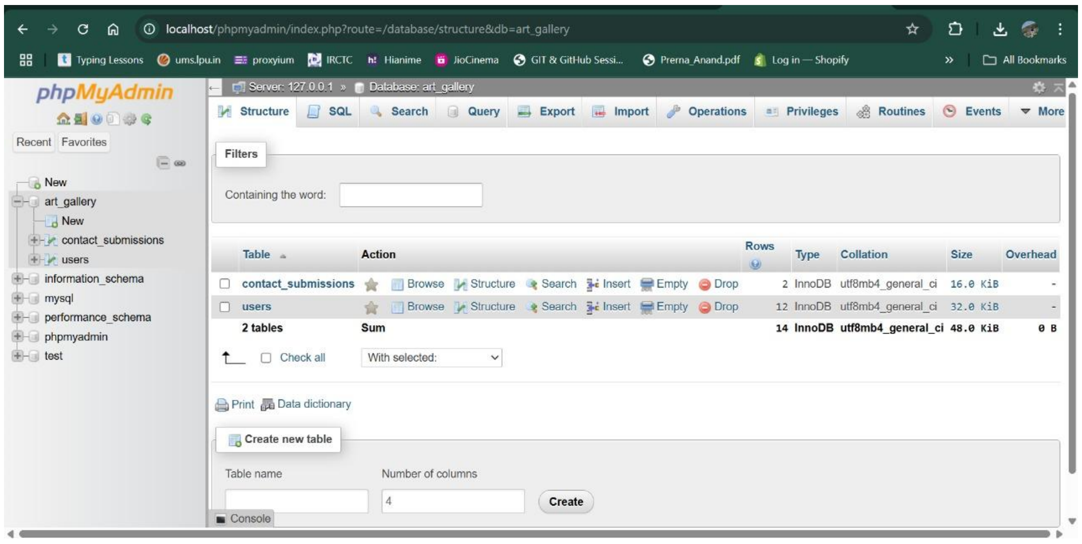
localhost/art_gallery/view_submissions.php

1744690669_Report Format_compressed (1).pdf
905 KB • Done

Contact Submissions

Name	Email	Phone	Message	File
harry	harry2@gmail.com	8765136894	see	Download File
prerna	prerna13anand@gmail.com	08765136894	review	Download File

35. DATABASE INTEGRATION



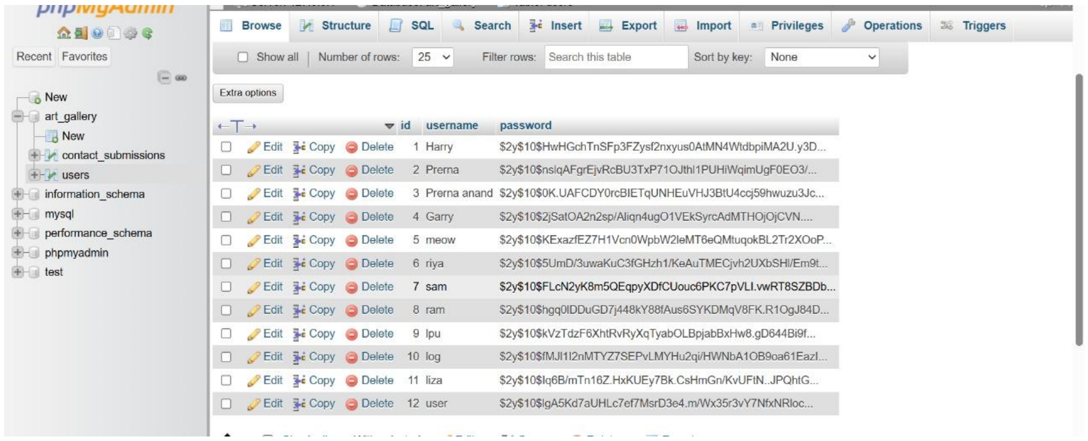
localhost/phpmyadmin/index.php?route=/database/structure&db=art_gallery

Server: 127.0.0.1 » Database: art_gallery

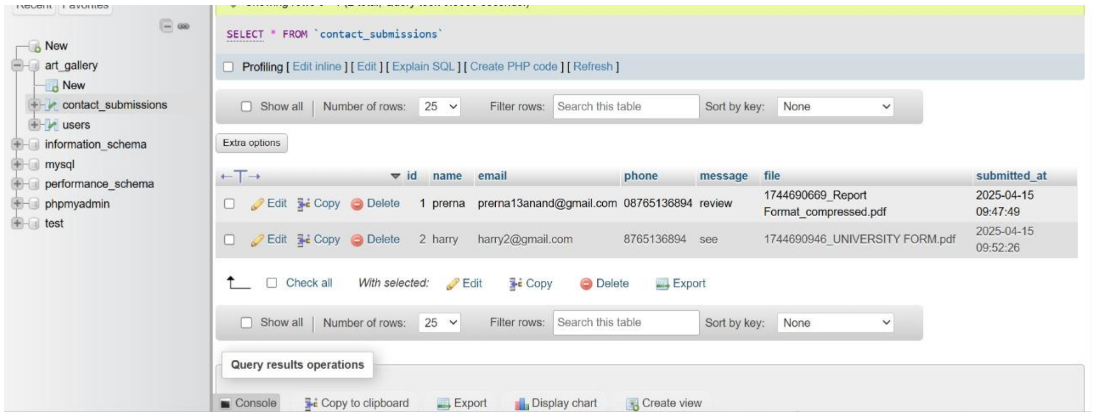
Filters

Containing the word:

	Table	Action	Rows	Type	Collation	Size	Overhead
☐	contact_submissions	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	users	Browse Structure Search Insert Empty Drop	12	InnoDB	utf8mb4_general_ci	32.0 KiB	-
	2 tables	Sum	14	InnoDB	utf8mb4_general_ci	48.0 KiB	0 B



id	username	password
1	Harry	$2y$10$HwHGchTnSFp3FZysf2nxyus0AtMN4WtdbpiMA2U.y3D...
2	Prerna	$2y$10$nslqAFgrEjvRcBU3TxP71OJthl1PUHiWqimUgF0EO3/...
3	Prerna anand	$2y$10$0K.UAFCDY0rcBIETqUNHEuVHJ3BtU4ccj59hwuzu3Jc...
4	Garry	$2y$10$2jSatOA2n2sp/Aliqn4ugO1VEkSyrcAdMTHOjOjCVN....
5	meow	$2y$10$KExazfEZ7H1Vcn0WpbW2leMT6eQMtuqokBL2Tr2XOoP...
6	riya	$2y$10$5UmD/3uwaKuC3fGHzh1/KeAuTMECjvh2UXbSHl/Em9t...
7	sam	$2y$10$FLcN2yK8m5QEqpyXDfCUouc6PKC7pVLI.vwRT8SZBDb...
8	ram	$2y$10$hgq0lDDuGD7j448kY88fAus6SYKDMqV8FK.R1OgJ84D...
9	lpu	$2y$10$kVzTdzF6XhtRvRyXqTyabOLBpjabBxHw8.gD644Bi9f...
10	log	$2y$10$fMJl1l2nMTYZ7SEPvLMYHu2qi/HWNbA1OB9oa61Eazl...
11	liza	$2y$10$lq6B/mTn16Z.HxKUEy7Bk.CsHmGn/KvUFtN..JPQhtG...
12	user	$2y$10$lgA5Kd7aUHLc7ef7MsrD3e4.m/Wx35r3vY7NfxNRloc...



```
SELECT * FROM `contact_submissions`
```

id	name	email	phone	message	file	submitted_at
1	prerna	prerna13anand@gmail.com	08765136894	review	1744690669_Report Format_compressed.pdf	2025-04-15 09:47:49
2	harry	harry2@gmail.com	8765136894	see	1744690946_UNIVERSITY FORM.pdf	2025-04-15 09:52:26

*****