Software Requirements Specification Document For Art Gallery

1. Introduction

1.1 Purpose

This document outlines the requirements for developing an avant-garde online art gallery platform/website. The purpose is to define the essential features and non-functional aspects required for a seamless consumer and seller experience. The platform provides a space for artists to showcase and sell their artwork while offering viewers the opportunity to explore and acquire art.

1.2 Scope

This platform aims to assist artists in augmenting artwork sales, streamlining the submission process, and efficiently managing their art for customers. It equips both customers and artists/sellers with personalized pages to organize and review their records/art. The platform encompasses features such as user registration, User Authentication and Profiles, artwork submission, Order History, Social Media Integration, Notifications, User Reviews and Ratings, Artwork Details, online gallery, and seamless payment processing.

1.3 Objective

- Design and implement a user-friendly interface ensuring seamless navigation for both artists and viewers.
- Develop a feature enabling artists to effortlessly showcase their artwork through high-quality images and detailed descriptions.
- Implement filter functionalities aiding viewers in the facile discovery of artworks based on categories, styles, artists, and other pertinent criteria.
- Incorporate a transparent feedback and rating system fostering trust within the art community.
- Establish a robust security system to safeguard user data, transactions, and sensitive information.

2. System Overview:

2.1 System Description

The online art gallery platform/website serves as a marketplace for artists to exhibit and sell their artwork, and for viewers to explore and purchase art. It encompasses features such as user registration and authentication, artwork search and browsing, artist profiles, product pages, shopping cart, checkout process, and secure payment integration.

2.2 System Architecture

2.2.1 Presentation Layer

- Web Interface: User-friendly web pages for artwork browsing, shopping cart, and user/admin profiles, responsively designed for various devices.
- User Interface Framework: HTML, CSS, and JavaScript for client-side interactions, employing frameworks like React or Angular for dynamic and responsive UI.
- Web Server: Utilizing Apache to handle HTTP requests with SSL/TLS for secure communication.

2.2.2 Application Layer

- Application Server: Implementing Node.js to manage application logic, encompassing user authentication, shopping cart operations, and transaction processing.
- Business Logic: Modules for user authentication, artwork management, and transaction processing, including integration with payment gateways for secure transactions.
- Restful API: Defining communication between the front-end and back-end to enable data retrieval for artwork details, user profiles, etc.
- Middleware: Connecting the application to the database, handling server-side operations, and communicating with the database layer.

2.2.3 Data Layer

- Database: Employing MySQL, PostgreSQL, or Mongo DB to store user data, artwork details, and transaction history, with tables for users, artists, artworks, orders, etc.
- Database Server: Efficiently managing database operations to ensure data integrity and security.

3. Functional Requirements

3.1 User Registration and Authentication

- Users can create accounts with unique usernames and emails. Passwords must meet security criteria. Registered users can securely log in, and a password reset process is available.

3.2 Artwork Search and Browsing

- Users can search for artwork using keywords, artists, or categories. An advanced search option includes filters like medium, style, and price range. Artworks are logically categorized, and users can browse within specific categories. Each artwork links to the artist's profile.

3.3 Product Pages and Shopping Cart

- Artwork pages include detailed information, an "Add to Cart" button, quantity selection, reviews, and availability status. The shopping cart allows users to add/remove items, displays a summary, enables quantity adjustment, and updates prices dynamically.

3.4 Checkout Process and Payment Integration

- A prominent "Checkout" button initiates the payment process. Payment integration involves a secure gateway supporting various methods, ensuring smooth transaction processing.

3.5 User and Artist Profile Management

- Users create accounts with essential information. Artists register with details, manage portfolios, submit artworks for approval, integrate social media profiles, and receive notifications.

4. Non Functional Requirements

4.1 Performance

- Response Time: Define acceptable response times for various operations (e.g., loading a page, processing a transaction).
- Scalability: The system should handle an increasing number of users, artworks, and transactions without significant performance degradation.
- Throughput: Specify the number of transactions or operations the system should handle within a given time frame.

4.2 Reliability

- Availability: Specify the acceptable system uptime, ensuring that the platform is available for users.
- Fault Tolérance: Define how the system should handle errors, crashes, or other failures without losing data or compromising user experience.

4.3 Security

- Authentication and Authorization: Clearly define how user authentication and authorization should be handled to ensure secure access to sensitive information.
- Payment Security: Ensure compliance with industry standards for secure payment processing, such as PCI DSS.

4.4 Compatibility

- Browser Compatibility: Specify which browsers the system should be compatible with to ensure a consistent user experience.
- Device Compatibility: Ensure that the platform works seamlessly on various devices such as smartphones, tablets, and desktops.

4.5 Maintainability

- Code Maintainability: Specify coding standards and practices to ensure that the code base is easy to understand, update, and maintain.
- Documentation: Require comprehensive documentation for developers, administrators, and end-users to facilitate system maintenance.

4.6 Performance

- Load Testing: Conduct thorough load testing to ensure the system can gracefully handle the expected user load without performance issues.
- Caching Mechanism: Implement an effective caching mechanism to optimize the performance of frequently accessed data, enhancing overall system responsiveness.