ART INVASION PROJECT

System Design Specification (SDS) for Art Invasion

---

1. Introduction

1.1 Purpose

The purpose of this document is to outline the design of the Art Invasion platform, describing its architecture, components, and how it will fulfill the requirements specified in the System Requirements Specification (SRS). It serves as a blueprint for the development and implementation of the platform.

1.2 Scope

This system will allow artists to create profiles, list artworks, and facilitate secure transactions with buyers. It will address challenges such as limited market access for local artists, lack of global exposure, and issues of trust between buyers and sellers. The platform will operate initially in Kenya and expand globally.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS:** System Requirements Specification
- **UI:** User Interface
- **DBMS:** Database Management System
- **API:** Application Programming Interface

---

2. System Architecture

The system architecture for Art Invasion is designed to be a scalable and secure platform with the following components:

- **Frontend (Client-side):**
  - Developed using **ReactJS** for dynamic, responsive web pages.
  - For future purposes, Mobile support will be added using **Java** and **Swift** for future mobile app development.
- **Backend (Server-side):**
  - Built using **PHP**, enabling efficient handling of multiple requests and real-time features.
  - **MySQL** will be used as the database management system for storing user and artwork data.

---

3. Module Design

The system is divided into the following key modules:

3.1 User Module

- **Artists:**
  - **Account Creation:** Artists will create profiles and upload artworks, including details like category, price, description, and images.
  - **Artwork Management:** Artists can edit, delete, or update their listings at any time.
  - **View Transactions:** Artists can view their transaction history and monitor sales.
- **Buyers:**
  - **Account Creation:** Buyers will create profiles to manage their purchases and preferences.
  - **Artwork Discovery:** Buyers will search and filter artworks based on various parameters such as price, category, artist, and location.

3.2 Authentication Module

- **Artwork Verification:** Use image recognition APIs for verifying the authenticity of artworks before they are listed.
- **Profile Verification:** Authentication of user identities to ensure credibility.

3.3 Recommendation Module

- **Personalized Recommendations:** Ad-driven system that suggests artworks to buyers based on their past behavior, interests, and preferences.

3.4 Admin Module

- **Dashboard:** Admins can monitor user activity, review transactions, approve artwork listings, and manage users.
- **Reporting:** Admins will have access to platform performance and transaction reports.

---

4. Database Design

The database design for Art Invasion includes the following main entities:

1. tblArtist

i. ArtistID (int): Primary Key
ii. Name (varchar(250))
iii. Email (varchar(250))
iv. Password (varchar(250))
v. Role (enum('Artist', 'Admin'))
vi. ProfilePic (varchar(250))
vii. Education (mediumtext)
viii. Award (mediumtext)
ix. PaymentStatus (enum('Pending', 'Processed'))
x. Amount (decimal(10,2))
xi. Date (timestamp)
xii. CreationDate (timestamp)

2. tblArtwork

- ArtworkID (int): Primary Key
- Title (varchar(250))
- Dimension (varchar(250))
- Orientation (enum('Landscape', 'Portrait'))
- Size (enum('Medium', 'Large'))
- ArtistID (int): Foreign Key referencing tblArtist.ArtistID
- SellingPrice (decimal(10,2))
- Description (mediumtext)
- Image (varchar(250))

- CreationDate (timestamp)

3. tblUser

- UserID (int): Primary Key
- Name (varchar(250))
- Email (varchar(250))
- Password (varchar(250))
- Role (enum('Artist', 'Buyer', 'Admin'))
- ProfilePic (varchar(250))
- CreationDate (timestamp)

4. tblTransaction

- TransactionID (int): Primary Key
- BuyerID (int): Foreign Key referencing tblUser.UserID
- ArtworkID (int): Foreign Key referencing tblArtwork.ArtworkID
- Amount (decimal(10,2))
- Date (timestamp)

5. tblVerification

- VerificationID (int): Primary Key
- ArtworkID (int): Foreign Key referencing tblArtwork.ArtworkID
- Status (enum('Pending', 'Approved', 'Rejected'))
- Reason (text)
- CreationDate (timestamp)

Relationships:

- **One-to-many:** One artist can have many artworks. (tblArtist to tblArtwork)
- **One-to-many:** One buyer can have many transactions. (tblUser to tblTransaction)
- **One-to-many:** One artwork can be involved in many transactions (e.g., if resold, reproduced, or relisted). (tblArtwork to tblTransaction)

---

5. User Interface Design

The system will feature an intuitive, clean, and responsive user interface:

5.1 Admin Dashboard

- **Profile Management:** Allows artists to update their profiles and manage their artworks.
- **Artwork Listings:** A list of artworks they have uploaded with the option to update or delete.

5.2 Buyer Homepage

- **Artwork Discovery:** A clean interface for browsing and filtering artworks by category, price, and artist.
- **Purchase Flow:** Simple, clear steps for purchasing artwork with payment options.
- **Recommendation Section:** AI-powered recommendations based on user preferences.

---

6. Security Design

6.1 Authentication and Authorization

- **Login/Signup:** Secure user login using email and password.
- **Role-based Access:** Different access levels for artists, buyers, and administrators.

6.2 Data Encryption

- All sensitive data (e.g., passwords, payment information) will be encrypted using industry-standard algorithms like **AES** and **SHA-256**.

6.3 Payment Security

- Payments will be handled securely, through the artist directly, which ensures robust security measures, including impersonation and fraud detection.

---

7. Performance Considerations

- **Scalability:** The platform will support at least 10,000 simultaneous users initially, with future scalability to handle increased traffic.
- **Load Testing:** Performance testing to ensure the platform responds in under 3 seconds under normal load.
- **Reliability:** The platform will ensure 99.9% uptime, leveraging a robust hosting provider (TrueHost/Safaricom).

---

8. Error Handling and Logging

- **Error Management:** Graceful handling of errors with appropriate error messages shown to the user.
- **Logging:** All transactions and errors will be logged for debugging and auditing purposes.

---

9. Testing Strategy

- **Unit Testing:** Ensure individual modules work as expected.
- **Integration Testing:** Test the interactions between modules and third-party integrations.
- **System Testing:** Ensure the entire platform functions smoothly and meets all requirements.

---

10. Deployment and Maintenance

- **Hosting:** The platform will be hosted on **TrueHost/Safaricom**, with provisions for scaling if needed.
- **Maintenance:** Regular updates and patches will be applied to ensure system stability and security.
