<Project Name>

Technical Design Document

	Prepared By / Last Updated By	Reviewed By	Approved By
Name			
Role			
Signature			
Date	22-06-2023		

Table of Contents

Introduction

Purpose of this document

The purpose of this document is to document the technical design, component details and Database design. This will also capture the scope, assumptions, risk, dependencies of this project.

Project overview

The Art Gallery Management System is a state-of-the-art platform that revolutionizes the art industry by showcasing a wide range of art pieces from artists worldwide. This comprehensive system provides art lovers, exhibition organizers, museums, and events with a seamless and immersive experience. With its extensive collection, efficient loan management, secure payment features, interactive communication, and rating functionality, the Art Gallery Management System aims to connect art enthusiasts and promote cultural appreciation on a global scale.

Solution Summary

Scope

<This sub-section shall state the scope of this detailed design document>

This detailed design document encompasses the following key aspects and functionalities of the Art Gallery Management System:

1. Authentication and Authorization:

- The application leverages Okta for user authentication and authorization.
- It supports different roles, including admin and user.
- Admins have full control over the system and access to administrative functions.
- Users have restricted access but can interact with the art collection and perform specific actions.

1.Art Management by Admin:

- The system empowers the admin with full control to manage the art collection.
- The admin can add new artworks, providing essential details such as title, author, category, description, quantity, and art image.
- The admin can modify the quantity of artworks as needed.
- Prompt response to user queries and inquiries is facilitated by the admin.

2. User Loans History and Shelf Page:

- Users, upon authentication, can loan artworks by adding them to their cart and proceeding to checkout.
- The system allows users to view their loan history, which displays the details of artworks they have previously checked out.
- A dedicated shelf page is provided for each user, where they can easily track and manage the artworks they have borrowed.
- Users can keep the artworks for the specified duration of time, as indicated during checkout.

3.User-Friendly UI and Enhanced Search Functionality:

- The application offers a visually appealing and intuitive user interface, enhancing the overall user experience.
- Advanced search features are implemented, including category-based and name-based searches.
- Pagination is implemented to optimize the display of search results, improving navigation efficiency.

4.Reviews, Queries, and Quantity Display:

- Users can view and contribute reviews and ratings for each artwork, enabling informed decision-making.
- A messaging system allows users to raise queries and seek additional information, with admin responsiveness ensured.
- The system provides real-time display of the available quantity for each artwork, enhancing user awareness of availability.

5. Artwork Checkout and Loan Management:

- Users can loan artworks by proceeding to checkout.
- After checkout, users can view the loan duration and can indicate whether they will return the artwork on or before the specified return date or choose to renew the loan.
- For extended loan durations, users are required to make securpayments through a payment gateway.

6. Secure Payment Integration:

- The system integrates a secure payment gateway, such as Stripe, to facilitate safe and reliable transactions.
- User card details are processed securely to ensure the confidentiality of financial information.
- Real-time payment transactions are generated and confirmation is sent through email notifications utilizing the capabilities of the Stripe application.

7.Guest Access:

- The application allows non-logged-in users to access certain features and functionalities.

- Guests can explore the art collection, browse artwork details, read reviews, and view the available quantity of each artwork.

- However, certain actions, such as checkout, loan history, and specific user-related functionalities, are restricted to authenticated users.

Assumptions

1.Network Connectivity:

- Users and administrators will have stable and reliable internet connectivity to access the Art Gallery Management System and its features.

2.Third-Party Integrations:

- The system assumes successful integration with third-party services, such as Okta for authentication and authorization and Stripe for secure payment processing.

3.System Scalability:

- The system assumes the ability to handle increased user activity and growing data volume without significant performance degradation.

4.System Compatibility:

- The Art Gallery Management System will be compatible with commonly used web browsers and devices, providing a seamless user experience across different platforms.

- The system assumes the availability of compatible software components, including operating systems, web servers, database management systems, and necessary frameworks or libraries.

- ## Dependencies

1. For Spring Boot:

- Okta Spring Boot Starter
- stripe-java
- JUnit
- SLF4J with Logback or Log4j2
- MySQL Connector/J
- Lombok

2. Database:

- SQL Workbench: You can use SQL Workbench/J as your database development and analysis tool.

3. For React:

- React (v18.2.0)
- @okta/okta-react
- stripe
- react-router-dom (v6.13.0)

- ## Risks

1. **Technical Risks:**
   - Hardware or network failures causing data loss or system downtime.

2. Scope Risks:

- Misunderstandings or discrepancies in the project requirements, leading to scope creep.

Schematic Diagram

A schematic, or schematic diagram, is a representation of the elements of a system using abstract, graphic symbols rather than realistic pictures. It gives an overview of overall system

System Design

Proposed design

- The proposed design for this project follows a client-server architecture, with React handling the frontend user interface and Spring Boot providing the backend services. The system aims to create a web-based library management application.

Frontend Design:

- The frontend is developed using React, a popular JavaScript library for building user interfaces. It provides an interactive and responsive user interface for managing library operations.

- The frontend consists of the following components:

1.Auth: This folder contains components and utilities related to authentication and user authorization, including login forms, registration forms, and authentication-related APIs.

2.Images: The "Images" folder stores images and other media files required by your React components or application.

3. Layouts:It is comprised of following files:-

- HomePage: This layout component represents the home page of the application. It provides relevant information and possibly includes navigation options to other sections of the application.
- ManageArtPage: This layout component is dedicated to managing art-related content. It may include functionalities for creating, editing, or deleting art items.
- MessagesPage: This layout component is responsible for managing and displaying messages or notifications.
- NavbarAndFooter: This layout component contains the navigation bar and footer sections, ensuring consistent navigation and branding across multiple pages.
- PaymentPage: This layout component represents a page where users can make payments for services or products.
- ProductCheckoutPage: This layout component allows users to review and finalize their product selections before making a purchase.
- SearchProductPage: This layout component enables users to search for specific products or items within the application.
- ShelfPage: This layout component represents a page dedicated to displaying a collection or shelf of products or items.

4.Models:

The "Models" folder contains data models or structures used throughout the application. These models define the shape and properties of different entities, such as user models, product models, or message models.

Backend Design:

- The backend services are developed using Spring Boot, a Java-based framework for creating robust and scalable web applications.The design includes the following components:

1.Config:

- Contains configuration files and settings for the backend application, such as database configurations, security configurations, and other application-specific configurations.

2.Controller:

- Handles incoming requests from the frontend, maps them to appropriate methods for processing, and returns the response.

3. Entity:

- Represents the data model for books and users, mapping to corresponding tables in the database.

4.GlobalExceptionsHandler:

Handles global exceptions and provides centralized error handling for the backend application.

5.Repository (DAO - Data Access Object):

- Provides an interface for accessing the database and performs CRUD (Create, Read, Update, Delete) operations on book and user data.

6.Request Models:

- Define the structure and validation rules for incoming request payloads.

7.Response Models:

- Define the structure of response payloads sent back to the frontend.

8.Service:

- Implements the business logic, coordinates data access and manipulation, and interacts with repositories.

Database:

The system utilizes a MySQL database to store book and user data. The database schema is designed to efficiently represent the relationships between entities and support the required operations.

External Dependencies:

Okta: The system integrates with Okta for authentication and authorization, allowing users to securely log in and access their library accounts.

Stripe: The system also integrates with Stripe, a popular payment processing platform. Stripe enables users to make online payments for services, such as fines or book purchases, securely and efficiently. The integration with Stripe includes handling payment transactions, managing customer information, and ensuring the security of financial data.

• Component inventory

frontend and backend components for the project:

FRONTEND:

Components:

Auth

Layouts

LoginWidget.jsx

OktaSignInWidget.jsx

Credentials:

OktaConfig.ts

Images

Models:

AddProductRequest.ts

AdminMessageRequest.ts

HistoryModel.ts

MessageModel.ts

PaymentInfoRequest.ts

ProductModel.ts

ReviewModel.ts

ReviewRequestModel.ts

ShelfCurrentLoans.ts

App.tsx

BACKEND:

Config:

MyDataRestConfig.java

Controllers:

SecurityConfiguration.java

AdminController.java

MessagesController.java

PaymentController.java

ProductController.java

ReviewController.java

Entities:

Checkout.java

History.java

Message.java

Payment.java

Product.java

Review.java

Repositories:

CheckoutRepository.java

HistoryRepository.java

MessageRepository.java

PaymentRepository.java

ProductRepository.java

ReviewRepository.java

Request models:

AddProductRequest.java

AdminQuestionRequest.java

PaymentInfoRequest.java

ReviewRequest.java

Response model:

ShelfCurrentLoansResponse.java

Services:

AdminService.java

AdminServiceImpl.java

MessageService.java

MessagesService.java

MessagesServiceImpl.java

PaymentService.java

PaymentServiceImpl.java

ProductService.java

ProductServiceImpl.java

ReviewService.java

ReviewServiceImpl.java

Utils:

ExtractJWT.java

Global Exceptions Handler:

GlobalExceptionHandler.java

MethodNotAllowedException.java

NoMessageFoundException.java

NoPaymentInfoFoundException.java

NoProductFoundException.java

NoProductFoundOrNotCheckedException.java

OutOfStockException.java

ReturnProductException.java

ReviewException.java

UserEmailNotFoundException.java

Database Design

Data Model

<This sub section will give the schematic view of the database design>

Tables Structure

<This sub section will describe the table structure>

Column Name	Data Type	Length	Nulls

Appendices

Glossary

Acronyms	Definitions

- Other

Terms & Conditions

Disclaimer: Please do not circulate or distribute this document outside of Cognizant Network, We have a Zero Tolerance Policy. Kindly adhere to 100% Compliance at all times.

Change Log

Please note that this table needs to be maintained even if a Configuration Management tool is used.

Version Number	**Changes made**			
V<n.n>	*<If the change details are not explicitly documented in the table below, reference should be provided here>*			
	Page no	Changed by	Effective date	Changes effected