NFT Marketplace Documentation

Introduction

This document provides an overview of the NFT marketplace, detailing the project's objectives and scope, focusing on digital asset trading using MENA (React & Node.js) .

1. Business Requirements

User Registration and Authentication: Secure account creation and login using JWT.

Product Catalog Management: Users can manage NFT listings with details such as name, description, and images using Cloudinary.

Shopping Cart and Checkout Process: Support for cart functionalities and a secure checkout process.

Order Management: Capability to handle orders, update statuses.

Responsive Design: Ensure compatibility across different devices.

Search and Filter: Enhanced product discovery features.

Timeline and Milestones

Phase 1: Setup of the backend environment and database.

Phase 2: Development of user authentication and NFT management.

Phase 3: Implementation of frontend interface and integration with backend.

2. Technical Requirements

Common Setup and Architecture

Scalable Project Structure: Organization of code into modules and test folders.

NPM Project Setup: Proper management of `package.json` for dependencies and scripts.

Version Control: Use Git for source control with well-managed branches and commits.

Functionality

JWT Authentication: Implementation of JWT for secure and efficient user authentication.

Database Setup: Use MongoDB, connected via Mongoose, with encrypted user data.

CRUD Endpoints: Well-organized routes for user, NFT, and order management.

3. Backend Functionality

Express in Node.js: Setup and configuration of Express server with essential middlewares.

API Endpoints: Defined for users, NFTs, orders, categories, and profiles.

Security Practices: Use of bcrypt for password encryption and dotenv for environment variables.

4. Frontend Functionality

Dashboard and Profile Management: Features for user interaction and profile management.

Component-Based Architecture: Use of modular and reusable React components.

Responsive Design: Implementation using Tailwind CSS for a flexible UI.

Client-Side Routing: Utilize React Router for SPA navigation.

Form Handling: Integration of Formik for robust form management.

State Management: Use of React state management techniques.

Error Handling: Comprehensive error management to enhance user experience.

Functional Requirements Document (FRD)

Key Functions

1. User Authentication: Users can register, log in, and manage their profiles.
2. NFT Management: Users can create, view, update, and delete NFT listings.

Requirements Prioritization and Dependencies

Primary: User authentication must be established before enabling NFT management.

Secondary: NFT transactions depend on user authentication and profile management.

. Technical Requirements Document (TRD)

Architecture and Components

Frontend: Developed using React with Vite and styled with Tailwind CSS.

Backend: Node.js with Express framework handles API requests and business logic.

Authentication: JWT for secure access control.

Infrastructure Requirements

Server: Node.js server environment.

Database: MongoDB for storing user and transaction data.

Third-party Services: Cloudinary for image management.

Development Tools and Libraries

Node.js, Express, React, Vite, Tailwind CSS, Axios, Mongoose, dotenv.

APIs and Data Formats

- Integration with external systems via RESTful APIs.
- Data exchanged in JSON format.

Database Schema Documentation

Users: Stores user data including email, password, and profile information.

NFTs: Records details of NFTs such as title, description, and owner.

Orders: Manages purchase orders for NFTs.

Categories: Classifies NFTs into different categories.

5. API Documentation

User API

Endpoint: `/api/users`

- `POST /register`: Registers a new user.
- `POST /login`: Authenticates a user.
- `GET /all`: Retrieves all users (admin only).
- More endpoints as per the routes defined.

NFT API

Endpoint: `/api/nfts`

- `POST /`: Creates a new NFT listing.
- `GET /`: Retrieves all NFT listings.

- More endpoints as per the routes defined.

Order API

Endpoint: `/api/orders`

- `GET /`: Lists all orders.
- `POST /`: Places a new order.
- More endpoints as per the routes defined.

Category API

Endpoint: `/api/categories`

- `POST /`: Creates a new category.
- `GET /`: Lists all categories.
- More endpoints as per the routes defined.

Profile API

Endpoint: `/api/profile`

- `PUT /profile/:id`: Updates user profile.
- `GET /profile/:id`: Retrieves a user profile.

. Environment Variables and Security

- `.env` configuration to manage sensitive data securely.
- Use of JWT for handling authentication and securing API endpoints.

6. README & Setup Instructions

Installation Guide: Step-by-step instructions to set up the project.

Usage Guidelines: How to use the application with examples.

7. Testing and Deployment

Testing: Implement unit and integration tests using Jest.

Deployment: Guidelines for deploying the application to a production environment.