# Project Report

## Abstract

This project implements a blockchain-based digital asset marketplace using Solidity for smart contract development and a React-based frontend with Bootstrap and Iconsax. The primary objective is to enable users to register, manage, and trade digital assets securely. The project integrates Ethers.js, Hardhat, Waffle, and Truffle for development and testing, with Remix IDE used for contract verification. While functional, this marketplace is a small-scale project and does not meet enterprise security standards.

## Introduction

The protection and ownership verification of digital assets has become crucial in the era of Web3 and decentralized finance. Blockchain technology offers immutable, transparent, and secure solutions to register and trade assets. This project aims to demonstrate how a smart contract can be leveraged to create a digital asset marketplace, allowing users to register assets via the payToMint function, retrieve asset information (hashes, timestamps, and metadata URIs), and trade assets between users. However, due to the use of setApprovalForAll, all users are automatically approved, presenting security risks.

## Methodology

### Design Decisions

- **Smart Contract:** Developed using Solidity, with OpenZeppelin libraries for security and token management.
- **Frontend:** Built with React, Bootstrap, and Iconsax for an intuitive user experience.
- **Blockchain Interaction:** Utilizes Ethers.js for seamless communication with the Ethereum network.
- **Testing Frameworks:** Hardhat, Waffle, and Truffle ensure contract functionality and security.

## Architecture

1. **Frontend (React + Bootstrap)**

    - User interface for asset registration, viewing, and transactions.
    - Integrates Ethers.js for blockchain interactions.
2. **Smart Contract (Solidity)**
    - Implements ERC-721 tokens to represent digital assets.
    - Tracks asset ownership, metadata, and pricing.
    - Handles asset trading through the buyToken function.
3. **Blockchain Transactions**
    - Users interact with smart contracts to mint and trade assets.
    - Transactions are recorded on the Ethereum blockchain for transparency.

# Implementation

## Smart Contract Structure

- payToMint: Mints a new NFT when a user pays the required fee.
- setTokenPrice: Allows users to set a price for their assets.
- buyToken: Enables users to purchase assets from others.
- getTokenHash: Retrieves the hash of a specific token.
- getTokenTimestamp: Fetches the timestamp of the asset's last transaction.

## Frontend UI Overview

- **Asset Registration Page:** Users submit metadata to mint NFTs.
- **Asset Marketplace:** Displays available assets for purchase.
- **Transaction History:** Logs past purchases and transfers.

## Diagrams & Screenshots

(Screenshots of the smart contract deployment, frontend interface, and blockchain transactions should be inserted here.)

# GitHub Repository

GitHub Repo Link

# Limitations & Security Considerations

- The project is a small-scale implementation and lacks enterprise-level security.

- All users are approved via setApprovalForAll, making unauthorized transfers possible.
- No advanced authentication or security measures are enforced.
- It should not be used in a real-world production environment without security improvements.