



华南理工大学  
South China University of Technology

# 本科毕业设计（论文）

基于区块链技术的数字时尚 NFT 交易平台

辅修学院      计算机科学与工程

辅修专业      计算机科学与技术

学生姓名      黄泽君

学生学号      202130670058

指导教师      解晓萌

提交日期      年    月    日



## 摘要

论文阐述了一个数字时尚 NFT 交易平台的构建过程，涵盖了从设计到实现的各个方面。该平台利用 React、Next.js 和 Tailwind CSS 构建了现代化的用户界面，结合以太坊区块链和智能合约实现了后端功能。通过使用 React Context API 进行全局状态管理，平台能够高效地处理用户信息和交易数据。实用函数的引入简化了与智能合约的交互，包括合约实例化、交易记录的管理和地址格式化等功能。

经过本地测试确认所有功能正常后，项目在 Vercel 平台完成了部署。此平台展示了其在安全性、透明度和用户体验方面的优势，为数字时尚领域的 NFT 交易提供了一个创新的解决方案。

**关键词：**数字时尚；非可同质化代币；以太坊区块链；智能合约；React 前端开发；全局状态管理

# Abstract

The paper outlines the construction process of a digital fashion NFT trading platform, covering various aspects from design to implementation. The platform utilizes React, Next.js, and Tailwind CSS to build a modern user interface, while leveraging the Ethereum blockchain and smart contracts to achieve backend functionality. By employing the React Context API for global state management, the platform efficiently handles user information and transaction data. The introduction of utility functions has simplified interactions with smart contracts, including contract instantiation, transaction record management, and address formatting.

After confirming the functionality through local testing, the project was deployed on the Vercel platform. This platform demonstrates its advantages in security, transparency, and user experience, offering an innovative solution for NFT trading in the digital fashion domain.

**Keywords:** Digital Fashion; Non-Fungible Tokens; Ethereum Blockchain; Smart Contracts; React Front-End Development; Global State Management

# 目 录

摘要.....	I
<b>Abstract.....</b>	<b>II</b>
目录.....	III
<b>第一章 绪论 .....</b>	<b>1</b>
1.1 引言.....	1
1.2 研究背景.....	1
1.3 研究现状.....	2
1.4 论文结构.....	2
<b>第二章 区块链的基础知识 .....</b>	<b>3</b>
2.1 区块链的结构.....	3
2.1.1 块区块.....	3
2.1.2 区块体.....	4
2.2 数字签名.....	4
2.3 区块链的要素.....	5
2.4 智能合约与 Solidity 语言 .....	5
<b>第三章 项目设置与合约构建 .....</b>	<b>8</b>
3.1 项目设置.....	8
3.2 合约构建.....	9
3.2.1 合约中的接口定义.....	9
3.2.2 NFT 市场合约 .....	10
3.2.3 其他合约.....	13
3.3 合约测试.....	14
3.4 部署配置.....	16
3.4.1 配置 Hardhat.....	16
3.4.2 配置部署脚本.....	16
<b>第四章 实用函数与上下文管理.....</b>	<b>17</b>

4.1	实用函数.....	17
4.1.1	合约交互.....	17
4.1.2	其他实用函数.....	18
4.2	Context 管理 .....	18
<b>第五章</b>	<b>前端开发 .....</b>	<b>21</b>
5.1	前端框架.....	21
5.1.1	框架介绍 .....	21
5.1.2	具体配置 .....	21
5.2	React 组件 .....	21
5.3	React 页面 .....	24
<b>第六章</b>	<b>最终测试及部署 .....</b>	<b>26</b>
6.1	最终测试.....	26
6.2	Vercel 部署 .....	28
<b>结论</b>	<b>.....</b>	<b>30</b>
1.	论文工作总结 .....	30
2.	工作展望 .....	30
<b>参考文献</b>	<b>.....</b>	<b>32</b>
<b>致谢</b>	<b>.....</b>	<b>34</b>

# 第一章 绪论

## 1.1 引言

元宇宙（Metaverse）作为一个融合了虚拟现实（VR）、增强现实（AR）、区块链、人工智能（AI）等多种前沿技术的虚拟共享空间，被认为是互联网的下一次重大变革，具有巨大的发展潜力和商业机会<sup>[1]</sup>。科技公司以及各国政府都在积极投入资源，推动元宇宙的技术发展和应用普及。此外，Web 3.0 的概念也在推动这一变革，强调去中心化、用户控制和智能化，为元宇宙的进一步发展提供了技术支持和新的应用场景。

## 1.2 研究背景

元宇宙的经济系统通常基于区块链技术，其中虚拟货币和非可同质化代币（NFT）占据着关键地位。这两者对应到传统经济中就是货币和所有权认证，在经济系统中的地位至关重要<sup>[2]</sup>。它们不仅是经济活动的基础，也是确保市场运作、交易安全和资源配置效率的核心要素。

在未明确所有权之前，地球上的自然资源、土地、黄金和石油等，尽管本身具有内在价值，但其市场价值无法充分体现。所有权的确立通过法律和经济机制赋予资源明确的控制权，使其具备市场交易的基础。因此，市场交易的核心是所有权的转移，这使得资源能够在经济系统中进行有效分配和利用。

同样地，数字世界中的所有权确认也至关重要。通过区块链技术，数字资产（如虚拟土地、数字艺术品）的所有权可以明确和验证。NFT 在此过程中发挥关键作用，因为它们提供了不可篡改的所有权记录和交易历史。这不仅确保了数字资产的唯一性和稀缺性，还为其创造了市场价值，使得这些数字资产能够在元宇宙中自由交易和流通。

时尚行业天然具备创新和自我表达的特性，在元宇宙中这一特性得到了更充分的发挥，成为元宇宙生态系统中的重要组成部分。数字时尚品牌和设计师借助先进的 3D 建模和区块链技术，创作并销售数字服装，这些服装通过 NFT 进行所有权认证和交易。这不仅为消费者提供了个性化和创新的时尚体验，也为时尚产业带来了新的商业模式和收入来源。例如，阿姆斯特丹的电子时装公司 The Fabricant 在 2019 年以 9500 美元的价格拍卖了第一件采用区块链技术制作的数字高级定制服装“Iridescence 彩虹连衣裙”<sup>[3]</sup>。

### 1.3 研究现状

NFT 利用区块链技术来实现其核心功能。区块链是一种去中心化的分布式账本技术，通过密码学和共识算法，确保数据的安全和透明<sup>[4]</sup>。区块链的关键特征是分布式数据库<sup>[5]</sup>。区块链数据库存在于去中心化网络中，网络中的每台计算机都会接收到区块链的副本，所有新交易的数据会传播到所有节点。

智能合约是 NFT 交易平台的核心，负责处理 NFT 的铸造、交易和所有权转移。确保智能合约的安全性和高效性至关重要。在安全性方面，文章[6]研究了以太坊和比特币等主流区块链系统在实际应用场景中的安全威胁。文章[7]探讨了区块链技术在云计算系统中的安全影响，重点分析了块隐瞒（Block Withholding, BWH）攻击的机制及其对区块链云系统的影响。

在效率方面，文章[8]展示了一种新的区块链系统在处理交易时的显著进展。该系统采用了一种新的拜占庭协议（Byzantine Agreement, BA），使用户能够快速达成共识，在扩展到更多用户时几乎没有性能损失。文章[9]研究链下交易和二层扩展以减少对主链的依赖，Layer-2 协议不仅提高了交易处理效率，还解决了传统区块链系统中存在的扩展性问题。

除此之外，也有一些研究探讨了区块链的可扩展性、跨链兼容性、智能合约的升级以及合规性与法律问题。Web 3.0 的出现进一步推动了这些研究，强调了去中心化、用户控制和智能化，这不仅为区块链技术的发展提供了新的方向，也为元宇宙的实际应用奠定了坚实的基础。

### 1.4 论文结构

本文分为六章。其中第一章简述了数字时尚 NFT 交易平台的研究背景和意义以及区块链的基础知识等。第二章节从区块链的结构、数字签名、区块链的要素、智能合约与 Solidity 四方面详细讲述了区块链的基础知识。第三、四、五章分别构建数字时尚 NFT 交易平台的智能合约、上下文管理及前端部分。在实际开发过程中，这三个模块通常由三个不同的团队负责。第六章节是数字时尚 NFT 交易平台的最终测试及部署。

## 第二章 区块链的基础知识

### 2.1 区块链的结构

区块链由多个“区块”按时间顺序链接而成，每个区块包含若干笔交易数据，并与前一个区块相连，形成一个线性链条<sup>[10]</sup>。这些块相互连接，每个块都引用链中的前一个块。链中的第一个区块称为创世区块。区块链被设想为一个垂直堆栈，区块相互堆叠，创世区块是堆栈的底部，如图 1-1 所示。

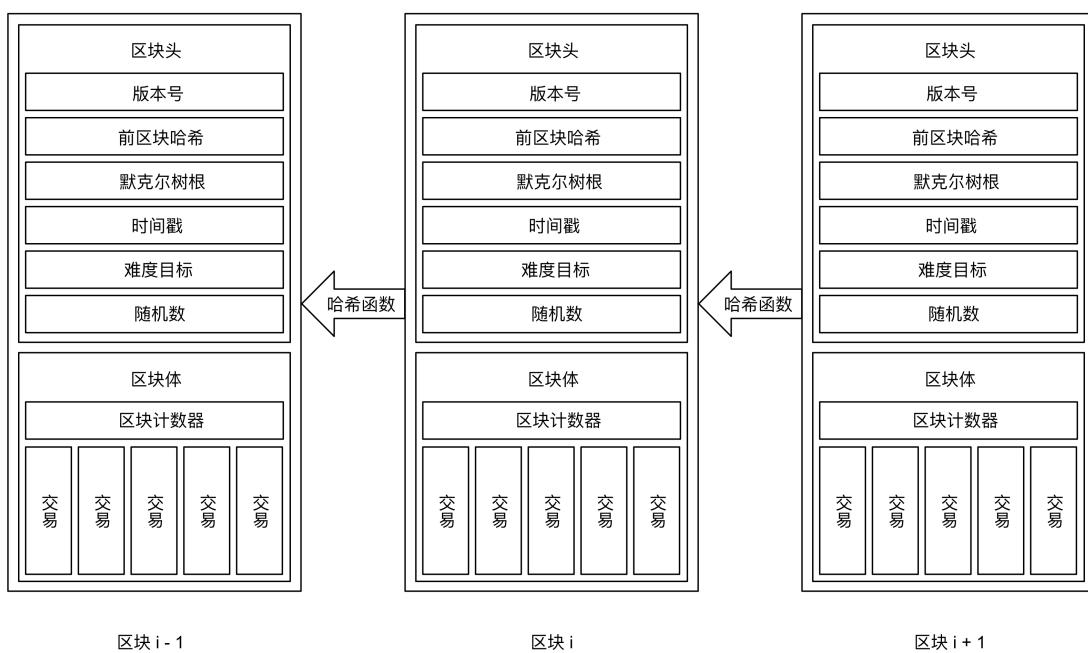


图 1-1 区块链的结构

#### 2.1.1 块区块

区块头是区块的重要部分，它包含多个字段，包括通过 SHA256 算法生成的前一个区块的哈希值、默克尔树根、时间戳、难度目标以及一个随机数（nonce）等。具体结构如下：

1. 版本号（Version）：表示区块的版本信息，确保节点能够解析区块结构的变化
2. 前区块哈希（Previous Block Hash）：当前区块链接到的前一个区块的哈希值，确保区块链的连贯性和不可篡改性。

3. 默克尔树根 (Merkle Root): 所有交易数据的哈希值通过哈希运算生成的根哈希值，用于快速验证区块内交易的完整性。
4. 时间戳 (Timestamp): 记录区块生成的时间，以确保区块链的时间顺序和可追溯性。
5. 难度目标 (Difficulty Target): 当前区块链网络设定的挖矿难度，用于控制新区块生成的速率。
6. 随机数 (Nonce): 挖矿过程中用来找到符合难度要求的哈希值，确保工作量证明 (PoW) 的有效性。

### 2.1.2 区块体

区块体包含区块计数器和该区块中记录的所有交易数据。具体结构如下：

1. 区块计数器 (Transaction Counter): 用于记录该区块中包含的交易数量，便于快速索引和检索。
2. 交易列表 (Transaction List): 每笔交易由交易输入和交易输出组成。交易输入包括引用上一笔交易的输出以及解锁脚本，用于验证和解锁资金；交易输出包括接收地址和锁定脚本，指定资金的接收者和使用条件。交易列表确保了区块中所有交易的完整性和可追溯性，是区块链数据结构的核心组成部分。

## 2.2 数字签名

数字签名是区块链技术中确保数据完整性和真实性的重要机制之一。它利用公钥密码算法 (Public Key Cryptography) 来验证交易的来源和内容，防止数据在传输过程中被篡改。公钥用于对要发送的消息进行签名和加密，指定的接收者可以使用其私钥解密消息。公钥和私钥在数学上是相互关联的。由于所使用的数学数学的复杂性，几乎不可能猜出这些密钥，这使得交易更难被破解<sup>[11]</sup>。

在区块链技术中，目前较为常用的是椭圆曲线数字签名算法 (ECDSA)<sup>[11]</sup>。ECDSA 使用较小的密钥长度即可提供比较好的的安全强度。例如，256 位的 ECDSA 密钥可以提供与 3072 位的 RSA 密钥相同的安全级别。此外，由于椭圆曲线的数学特性，ECDSA 在签名生成和验证时会更快。较小的密钥和签名长度意味着区块链中的交易数据更小，

降低了存储和传输的成本。

## 2.3 区块链的要素

区块链技术的要素可以分为多个方面，其中最重要的八个要素包括去中心化、共识模型、透明性、开源性、身份与访问控制、自主性、不变性和匿名性<sup>[11]</sup>。

1. 去中心化指的是功能和控制权的分散，在区块链中没有中央权威，每个用户（矿工）都拥有一份交易账本的副本，新区块通过矿工的验证后被添加到区块链中。
2. 共识算法是区块链技术中的关键组成部分，用于在不可信节点之间达成一致<sup>[13]</sup>。这一问题类似于拜占庭将军问题。在区块链网络中，没有中央节点来确保分布式节点上的账本一致性，因此需要共识算法来确保不同节点上的账本保持一致。常见的共识算法包括工作量证明（PoW）<sup>[14]</sup>、权益证明（PoS）<sup>[15]</sup>、实用拜占庭容错（PBFT）<sup>[16]</sup>等。
3. 透明性体现在区块链网络定期自我审计的能力上，确保了交易的透明和不可篡改。
4. 开源性意味着区块链应用程序的代码对外开放，增加了用户对应用程序的信任，特别是在涉及接收、保存或转移资金时。
5. 涉及区块链的公开性或私有性，公共区块链允许任何有互联网访问权限的用户加入网络并参与交易验证，而私有区块链限制用户的权限，适合传统商业模式。
6. 自主性强调区块链技术通过去中心化的方式，消除了对中央权威的依赖。分布式自治组织（DAO）是区块链技术的一个例子，通过智能合约来执行预设规则。
7. 不变性指的是区块链中存储的数据一旦写入，便无法被篡改，这对数据审计尤其重要。
8. 匿名性则体现在用户仅需提供区块链地址即可参与交易，从而解决了信任问题并保证了参与者的隐私。

## 2.4 智能合约与 Solidity 语言

智能合约（Smart Contract）是一种在区块链网络上运行的自执行合约。Solidity 是一种高级编程语言，用于在以太坊区块链上编写智能合约。它由 Gavin Wood 在 2014 年设计，并由以太坊项目团队开发和维护。Solidity 是一种静态类型的编程语言，具有类似于 JavaScript、Python 和 C++ 的语法，使得开发者能够快速上手并编写安全可靠的智能合约。

智能合约的安全性是 Solidity 开发的重中之重。智能合约一旦部署到区块链上，其代码和数据就无法轻易修改。此外，合约代码和经济价值的直接耦合，已部署合约中的错误可能会产生严重后果，因此必须在开发过程中仔细检查和测试代码<sup>[18]</sup>。

在以太坊中部署智能合约，需要执行一个特殊的创建事务，该事务将合约引入区块链。在此过程中，合约被分配一个唯一的地址，以 160 位标识符的形式，其代码被上传到区块链。一旦成功创建，智能合约就由合约地址、合约余额、预定义的可执行代码和状态组成。然后，不同的参与方可以通过向已知的合约地址发送调用合约的交易来与特定合约进行交互。

代码 2-1 展示了如何使用 Solidity 编写一个简单的酒店预订系统。合约定义了一个 Booking 结构体来存储每个预订的相关信息，使用 enum 定义 Pending、Confirmed、CheckedIn、CheckedOut 和 Canceled 五种状态，用于跟踪预订的当前进展。合约通过 bookRoom 函数允许用户创建新预订，自动分配唯一 ID 并存储在 bookings 映射中。cancelBooking 函数确保只有预订持有人可以取消预订，并在成功取消后将状态更新为 Canceled。

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

contract SimpleHotelBooking {

    enum Status { Pending, Confirmed, CheckedIn, CheckedOut, Canceled }

    struct Booking {
        uint256 id;
        address guest;
        uint256 checkInDate;
        uint256 checkOutDate;
        Status status;
    }

    uint256 public nextBookingId = 1;
    mapping(uint256 => Booking) public bookings;

    function bookRoom(uint256 checkInDate, uint256 checkOutDate) external {
        bookings[nextBookingId++] = Booking(nextBookingId, msg.sender, checkInDate,
checkOutDate, Status.Pending);
    }
}
```

```
function cancelBooking(uint256 bookingId) external {
    require(msg.sender == bookings[bookingId].guest, "Not your booking");
    bookings[bookingId].status = Status.Canceled;
}
```

代码 2-1 酒店预定合同

以太坊提供了一个去中心化的虚拟机作为运行时环境来执行智能合约，称为以太坊虚拟机（EVM）。EVM 可以被认为是一台全球去中心化计算机，所有智能合约都在该计算机上运行。处理智能合约执行的所有交易都是在网络的每个节点上本地进行的，并且以相对同步的方式进行处理。EVM 执行的每条指令都有一个与之相关的成本，以 Gas 为单位。需要更多计算资源的操作比需要较少计算资源的操作消耗更多的 Gas。这确保了系统不会受到拒绝服务攻击的干扰。

## 第三章 项目设置与合约构建

### 3.1 项目设置

在命令行输入 npx create-next-app 快速创建一个新的 Next.js 应用程序，删除不必要的文件并清空 CSS 样式得到简化和清理后的 Next.js 项目。在 package.json 中定义项目所需的各种依赖包如代码 3-1 所示<sup>[18]</sup>。

```

"dependencies": {
  "@formspree/react": "^2.5.1",           // 与 Formspree 集成的 React 组件, 处理表单提交
  "@nomiclabs/hardhat-ethers": "^2.0.5",   // 与 Hardhat 框架集成的以太坊 JavaScript 库
  "@nomiclabs/hardhat-waffle": "^2.0.3",    // 与 Hardhat 集成的 Solidity 测试库
  "@openzeppelin/contracts": "^4.5.0",      // OpenZeppelin 提供的经过审计的智能合约库
  "axios": "^0.26.0",                     // 用于发起 HTTP 请求的库
  "bignumber.js": "^9.1.1",                // 用于处理大数的 JavaScript 库
  "ethers": "^5.5.4",                     // 用于与以太坊区块链交互的 JavaScript 库
  "hardhat": "^2.9.0",                    // 构建智能合约开发环境的工具
  "ipfs-http-client": "^56.0.1",          // 与 IPFS 进行通信的客户端
  "next": "12.1.0",                      // Next.js 框架
  "next-themes": "^0.1.1",                // 实现主题切换功能的库
  "react": "17.0.2",                      // React 库, 用于构建用户界面
  "react-countdown": "^2.3.5",            // React 倒计时组件
  "react-dom": "17.0.2",                 // React DOM 操作库
  "react-dropzone": "^12.0.4",            // 文件拖放上传的 React 组件
  "react-hot-toast": "^2.3.0",             // 创建通知弹窗的 React 库
  "react-tailwindcss-datepicker": "^1.6.6", // 与 Tailwind CSS 集成的 React 日期选择器
  "web3modal": "^1.9.5"                  // 用于连接各种 Web3 钱包的模块
},
"devDependencies": {
  "autoprefixer": "^10.4.2",              // 自动为 CSS 添加浏览器前缀的工具
  "eslint": "^8.10.0",                    // JavaScript 代码检查工具
  "eslint-config-airbnb": "^19.0.4",       // Airbnb 的 ESLint 代码风格配置
  "eslint-config-next": "12.1.0",           // Next.js 提供的 ESLint 配置
  "eslint-plugin-import": "^2.25.4",        // 支持 ES6 模块的导入/导出检查
  "eslint-plugin-jsx-a11y": "^6.5.1",       // 检查 JSX 中的可访问性问题
  "eslint-plugin-react": "^7.29.3",         // React 代码风格检查插件
  "eslint-plugin-react-hooks": "^4.3.0",     // 用于 React Hooks 的 ESLint 插件
  "postcss": "^8.4.7",                   // CSS 处理工具
  "tailwindcss": "^3.0.23"                // CSS 框架, 用于快速构建自定义设计的用户界面
}

```

## 代码 3-1 项目所需的依赖包

在命令行输入 `npm install` 安装需要的依赖包，在项目的根目录下创建`.eslintrc.js`、`hardhat.config.js`、`postcss.config.js`、`tailwind.config.js` 定制项目的开发环境，并创建`components`、`context`、`scripts`、`utils`、`assets` 文件夹分别存放项目的第一种 React 组件、Context 对象、脚本文件、实用函数和一些静态资源。

## 3.2 合约构建

### 3.2.1 合约中的接口定义

ERC20 是以太坊网络上的一个代币标准，用于定义一种可互换的代币<sup>[19]</sup>。这一标准由以太坊社区定义，使得不同的代币可以在各种应用和服务之间进行互操作。ERC20 标准确保了代币合约遵循一致的接口，使得它们能够被广泛接受和支持。构建一个符合 ERC20 标准的智能合约接口如代码 3-2 所示。

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.4;
interface IERC20 {
    function totalSupply() external view returns(uint);
    //返回代币的总供应量。
    function balanceOf(address account) external view returns(uint);
    //查询指定地址的代币余额。
    function transfer(address recipient, uint amount) external view returns(bool);
    //将一定数量的代币从调用者的账户转移到另一个地址。
    function allowance(address owner, address spender) external view returns(uint);
    //查询一个地址授权给另一个地址的代币数量。
    function approve(address spender, uint amount ) external returns(bool);
    //授权某个地址（spender）可以从调用者的账户中支取一定数量的代币。
    function transferFrom(address sender, address recipient, uint amount) external
    returns(bool);
    //从一个地址（sender）转移代币到另一个地址（recipient），前提是已经获得了授权。
    event Transfer(address indexed from, address indexed to, uint value);
    //代币转移事件。当代币成功转移时触发此事件。
    event Approval(address indexed owner, address indexed spender, uint value);
    //授权事件。当调用 approve 函数时触发此事件。
}
```

代码 3-2 符合 ERC20 标准的智能合约接口

### 3.2.2 NFT 市场合约

NFT 市场合约继承了 OpenZeppelin 的 ERC721URIStorage 和 ReentrancyGuard。合约通过 ERC721 协议进行扩展，使用 Counters 库跟踪 Token ID 和售出商品数量。合约所有者在部署合约时被设置为合约的初始所有者。合约实现以下内容：

1. 铸造和上市 NFT：用户可以通过指定令牌 URI 和价格铸造新的 NFT 并将其列在市场上。
2. 购买和转售 NFT：用户可以购买上市的 NFT 并在市场上转售。
3. 拍卖 NFT：用户可以列出 NFT 进行拍卖，允许其他人出价。拍卖结束时，出价最高的人获胜。
4. 管理投标：合同安全地处理投标放置、撤回和拍卖完成。

在设计 NFT 市场合约时，安全性是最重要的考量之一。合约使用了 ReentrancyGuard 来防止重入攻击，这是一个常见的安全漏洞，尤其是在涉及资金转移的操作中。通过在关键函数中添加 `onlyOwner` 修饰符，确保只有合约的所有者才能调用特定的函数。此外，合约在转移资金时使用了 `require` 语句和检查目标地址，以确保资金的安全转移。

合约的主要模块有数据结构、事件和函数如下所示：

#### 3.2.2.1 数据结构 (Structure)

合约中的主要数据结构是 `MarketItem` 和 `userBidding`，如表 3-1 所示。

<code>MarketItem</code>	数据结构存储 NFT 的详细信息，包括 NFT 的 ID、卖家地址、所有者地址、价格、拍卖价格、是否在售、拍卖状态、起止时间等。
<code>userBidding</code>	数据结构存储用户的竞拍信息，包含 NFT 的 ID、竞拍 ID、竞拍金额、竞拍人地址等。这些数据结构通过映射存储在合约中，以便快速查找和更新相关信息。

表 3-1 NFT 市场合约的数据结构

这些数据结构通过映射存储在合约中，以便快速查找和更新相关信息。

#### 3.2.2.2 事件 (Event)

合约定义了一系列事件，用于记录和触发合约中的重要操作。事件数据存储在区块链的日志（logs）中，这些日志不直接影响合约的状态，查询时效率更高。事件不仅独

对开发者透明，也为用户和审计者提供了可靠的交易记录和状态追踪手段。如果你将数据存储在合约的状态变量中，比如数组或映射，每次更新数据时都需要支付较高的 Gas 费用。合约定义的事件如表 3-2 所示。

MarketItemCreated	事件在创建市场项目时触发，记录 NFT 的 ID、卖家、所有者、价格及售出状态。
AuctionCreated	事件在创建拍卖时触发，记录卖家、价格、NFT 的 ID、起止时间。
BidCreated	事件在竞拍时触发，记录竞拍 ID、竞拍人地址、竞拍金额。
AuctionCompleted	事件在拍卖完成时触发，记录竞拍 ID、卖家、竞拍人、竞拍金额。
WithdrawBid	事件在竞拍人撤回竞拍时触发，记录竞拍 ID、竞拍人地址、竞拍金额。

列表 3-2 NFT 市场合约的事件

这些事件帮助用户和开发者追踪合约操作，并提供透明度。

### 3.2.2.3 函数（Function）

合约构建了一系列函数如表 3-3 所示。

updateListingPrice	函数允许合约所有者更新挂牌价格。函数接受一个新的挂牌价格作为参数，并更新 listingPrice 变量。
getListingPrice	函数返回当前的挂牌价格。
createMarketItem	内部函数用于创建新的市场项目，并将其存储在 idToMarketItem 映射中。函数检查价格是否大于零，并验证支付的金额是否等于挂牌价格。
createToken	函数允许用户铸造新的 NFT，并将其添加到市场中。函数首先调用 _mint 和 _setTokenURI 创建新的 NFT，然后调用 createMarketItem 将其添加到市场项目中。
resellToken	函数允许 NFT 所有者重新挂牌出售 NFT。函数更新市场价格和所有者信息，并将 NFT 转移回市场合约。
createMarketSale	函数允许用户购买市场中的 NFT。函数验证支付金额是否等于 NFT 的价格，并将所有者信息更新为买家。
fetchMarketItems	函数返回所有未售出的市场项目。函数遍历 idToMarketItem 映射，筛选出所有未售出的项目。
fetchMyNFTs	函数返回用户购买的所有 NFT。函数遍历 idToMarketItem 映射，筛选

	出所有所有者为当前用户的项目。
fetchItemsListed	函数返回用户列出的所有 NFT。函数遍历 idToMarketItem 映射，筛选出所有卖家为当前用户的项目。
getContractBalance	函数返回合约当前的余额。
withdraw	函数允许合约所有者提取合约中的所有资金。
buyNFTWithToken	函数允许用户使用 ERC20 代币购买 NFT。函数更新市场项目的所有者信息，并将 NFT 转移给买家。
createAuctionListing	函数允许用户创建拍卖列表。函数设置拍卖价格、起始时间和结束时间，并触发 AuctionCreated 事件。
bid	函数允许用户竞拍 NFT。函数检查拍卖是否仍然开放，并验证竞拍金额是否高于当前价格。
completeAuction	函数允许卖家或竞拍赢家完成拍卖。
withdrawBid	函数允许竞拍人撤回竞拍。函数验证竞拍人不是最高竞拍者，并将竞拍金额退还给竞拍人。
isAuctionOpen	函数返回拍卖是否仍然开放。
isAuctionExpired	函数返回拍卖是否已经结束。
_transferFund	内部函数用于将资金转移给指定地址。
getHighestBidder	函数返回指定 Token ID 的最高竞拍人地址。
fetchMarketAuctionItems	函数返回市场中的所有拍卖项目。
fetchItemsAuctionListed	函数返回用户列出的所有拍卖项目。
getBidders	函数返回所有竞拍人的地址列表。
fetchUserBids	函数返回用户参与的所有竞拍。
fetchNFTBids	函数返回指定 NFT 的所有竞拍。

表 3-3 NFT 市场合约的函数

以 fetchMarketItems 函数为例，如代码 3-3 所示，函数在 for 循环中遍历所有已铸造 NFT。在每次循环中检查当前 NFT 的所有者是否为合约本身（即地址 address(this)）。如果条件成立，则表明该 NFT 尚未售出。将该 NFT 存储在 items 数组的当前位置，随后 currentIndex 递增以存储下一个未售出的 NFT。最后，函数返回该 items 数组，该数组包

含了市场上所有未售出的 NFT 的详细信息。

同样地，fetchMyNFTs 函数的条件就是判断当前 NFT 的所有者 (`idToMarketItem[i + 1].owner`) 与调用函数的用户地址 (`msg.sender`) 是否相同，fetchItemsListed 函数的条件就是判断当前 NFT 的卖家 (`idToMarketItem[i + 1].seller`) 与调用函数的用户地址 (`msg.sender`) 是否相同。

```
//GET ALL UNSOLD NFT
function fetchMarketItems() public view returns(MarketItem[] memory){
    uint itemCount = _tokenIds.current();
    uint unsoldItemCount = _tokenIds.current() - _itemsSold.current();
    uint currentIndex = 0;

    MarketItem[] memory items = new MarketItem[](unsoldItemCount);
    for(uint i = 0; i < itemCount; i++){
        if(idToMarketItem[i + 1].owner == address(this)){
            uint currentId = i + 1;
            MarketItem storage currentItem = idToMarketItem[currentId];
            items[currentIndex] = currentItem;
            currentIndex += 1;
        }
    }
    return items;
}
```

代码 3-3 NFT 市场合约的 `fetchMarketItems` 函数

### 3.2.3 其他合约

#### 3.2.3.1 原生代币合约

NFT 交易市场不仅可以接受以太坊（ETH）的付款，还可以接受平台自己的原生代币的付款。这种灵活性通过原生代币合约实现，允许用户使用 ETH 或平台特定令牌支付交易费用或购买商品和服务。

原生代币合约实现了代币的基本功能，包括转账、批准转账、查询余额和持有者信息等。它适用于 ERC20 代币标准的基本实现，可以在区块链上进行代币的管理和交易。

#### 3.2.3.2 代币销售合约

代币销售合约导入了原生代币合约，通过原生代币合约访问代币的相关方法。实现

了基本的代币销售平台，允许用户用以太币购买代币，并确保管理员可以安全地结束销售并收回未售出的代币和收益。

### 3.2.3.3 捐赠合约

捐赠合约提供了基本的捐赠系统，允许用户捐赠以太币，记录每次捐赠的详细信息包括地址、金额和时间戳，并更新合约的总捐赠金额。捐赠合约允许合约拥有者调用 `withdraw` 函数提取捐赠款。合约维护了一个捐赠者列表，存储所有捐赠的详细信息，包括捐赠者地址、金额和时间戳。

### 3.2.3.4 转账合约

转账合约实现了的资金转账功能，同时记录每次转账的详细信息包括发送者地址、接收者地址、金额、时间戳、名称和描述，并允许用户查询自己的转账历史记录或查看合约中所有的转账记录。

### 3.2.3.5 支持合约

合约实现了一个简单的用户反馈系统，允许用户通过 `sendMessage` 函数发送消息并记录这些消息的详细信息。用户可以查看自己发送的消息记录，管理员可以查看合约中所有的消息记录。

### 3.2.3.6 社区合约

社区合约实现了一个简单的社交网络平台，允许用户创建账户、添加好友、发送和接收消息。所有的用户信息、好友关系、消息记录都保存在区块链上，确保数据的安全性和透明性。

## 3.3 合约测试

Remix 是一个基于浏览器的开发环境，专为以太坊智能合约开发而设计。它支持 Solidity 编程语言，可以用于编写、测试、调试和部署智能合约。Remix 提供了一个用户友好的界面，并集成了多种工具，使开发者能够轻松进行智能合约开发。

复制 NFT 市场合约在 Remix 中进行部署，调用 `createToken` 函数并传入 `tokenURI`、`price` 和 `tokenPrice` 参数以创建 NFT，如图 3-1 所示。调用 `fetchMarketItems` 函数，显示创建的 4 个 NFT 如图 3-2 所示。测试结果表明，合约可以正常创建 NFT，并通过

fetchMarketItems 函数查看所有未售出的 NFT，在 Remix 环境中正常运行。

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

import "@openzeppelin/contracts/utils/Counters.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/security/ReentrancyGuard.sol";
import "hardhat/console.sol";

contract NFTMarketplace is ERC721URIStorage, ReentrancyGuard{
    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;
    Counters.Counter private _itemsSold;

    //BIDDING COUNT
    Counters.Counter public _numberOfBid;

    uint256 listingPrice = 0.015 ether;
    address payable owner;

    mapping(uint256 => MarketItem) private idToMarketItem;
    struct MarketItem {
        uint256 tokenId;
        address payable seller;
    }

    function createToken(string memory tokenURI, uint256 price, uint256 tokenPrice) public payable returns(uint256){
        _tokenIds.increment();

        uint256 newTokenId = _tokenIds.current();

        _mint(msg.sender, newTokenId);
        _setTokenURI(newTokenId, tokenURI);
        createMarketItem(newTokenId, price, tokenPrice);
        return newTokenId;
    }

    //CREATEMARKETITEM INTERNAL FUNCTION
    function createMarketItem(uint256 tokenId, uint256 price, uint256 tokenPrice) private {
        require(price > 0, "Price must be at least 1 wei");
        require(msg.value == listingPrice, "Price must be equal to listing price");

        idToMarketItem[tokenId] = MarketItem(
            tokenId,
            payable(msg.sender),
            payable(address(this)),
            price,
            price,
            false
        );
    }
}
```

图 3-1 调用 createToken 函数创建 NFT

```
function getListingPrice() public view returns(uint256){ return listingPrice; }

//MINTS NFT TO THE MARKETPLACE
function createToken(string memory tokenURI, uint256 price, uint256 tokenPrice) public payable returns(uint256){
    _tokenIds.increment();

    uint256 newTokenId = _tokenIds.current();

    _mint(msg.sender, newTokenId);
    _setTokenURI(newTokenId, tokenURI);
    createMarketItem(newTokenId, price, tokenPrice);
    return newTokenId;
}

//CREATEMARKETITEM INTERNAL FUNCTION
function createMarketItem(uint256 tokenId, uint256 price, uint256 tokenPrice) private {
    require(price > 0, "Price must be at least 1 wei");
    require(msg.value == listingPrice, "Price must be equal to listing price");

    idToMarketItem[tokenId] = MarketItem(
        tokenId,
        payable(msg.sender),
        payable(address(this)),
        price,
        price,
        false
    );
}
```

图 3-2 调用 fetchMarketItems 函数显示市场上的 NFT

## 3.4 部署配置

### 3.4.1 配置 Hardhat

完成合约构建后需要配置 Hardhat 用于开发和部署智能合约的环境。定义一个常量存储 Polygon Amoy 测试网络的 RPC URL，这里以 <https://rpc-amoy.polygon.technology/> 为例，此外也可以选择 Alchemy 或 Infura 等服务提供商提供的 RPC 端点。定义一个常量存储用户的私钥。这用于签署交易的密钥，因此非常重要。

配置 Solidity 编译器的版本以及相关设置，启用了优化器，指定优化的运行次数为 1000。优化器将尝试优化合约代码以减少其体积或提高执行效率。配置默认网络为 matic (Polygon 网络的简称)。定义 Hardhat 使用多个网络配置包括 Hardhat 自带的本地开发网络和 polygon\_amoy。

### 3.4.2 配置部署脚本

在合约部署脚本引入了 Hardhat 的运行环境模块。当前的脚本设计旨在为多个智能合约的部署做好准备。这些合约包括原生代币合约、NFT 市场合约、代币销售合约、社区合约、转账合约、支持合约和捐赠合约。代码编写阶段已经完成，下一步将依次在以太坊网络上部署这些合约，并在每次部署后输出合约地址。

在终端输入 `npx hardhat run scripts/deploy.js --network polygon_amoy` 完成部署，将合约的 JSON 文件从 `artifacts/contracts` 转移到 `context` 以便在项目中更方便地引用和使用这些合约的 ABI 和地址。在 `context/constants.js` 引入这些智能合约的 ABI，并定义部署后生成的 7 个合约的地址。将所有合约的 ABI 和地址集中在一个文件中，有利于方便管理和修改，减少冗余和出错的机会。

## 第四章 实用函数与上下文管理

### 4.1 实用函数

实用函数是指在代码中用于执行常见或重复性任务的小型、通用函数。它们通常被设计为简洁、独立，并且可以在多个地方复用，帮助简化代码和提高代码的可读性和可维护性。NFT 交易市场包括合约交互相关的实用函数及一些其他实用函数。

#### 4.1.1 合约交互

用户可以与以太坊区块链交互，如连接钱包、切换网络、获取用户余额，以及与智能合约交互。以代码 4-1 为例，FETCH\_CONTRACT 是一个辅助函数，负责实例化一个以太坊合约的接口。CALLING\_CONTRACT 是一个主函数，用于与用户钱包建立连接，并返回一个与智能合约交互的实例对象。这个函数主要用于确保用户的连接是安全且有效的，然后使用提供者（PROVIDER）来与合约进行交互。

```
//FETCH CONTRACT
const FETCH_CONTRACT = (CONTRACT_ADDRESS, CONTRACT_ABI, PROVIDER) =>
  new ethers.Contract(CONTRACT_ADDRESS, CONTRACT_ABI, PROVIDER);

//CONNECTING WITH CONTRACT
export const CALLING_CONTRACT = async (CONTRACT_ADDRESS, CONTRACT_ABI) => {
  try {
    const web3modal = new Web3Modal();
    const connection = await web3modal.connect();
    const provider = new ethers.providers.Web3Provider(connection);
    const PROVIDER = provider.getSigner();

    const contract = FETCH_CONTRACT(CONTRACT_ADDRESS, CONTRACT_ABI, PROVIDER);

    return contract;
  } catch (error) {
    console.log(error);
  }
};
```

代码 4-1 utils/contract.js 部分代码

## 4.1.2 其他实用函数

### 4.1.2.1 获取卖家总销售额

从一个包含交易或物品信息的数组中，计算每个卖家的总销售额，并返回一个包含卖家和他们总销售额的数组。

### 4.1.2.2 生成随机字符串

函数生成一个指定长度的随机字符串，字符集包括大写字母、小写字母和数字。

### 4.1.2.3 缩短地址

函数接收一个 address 作为参数，返回一个格式化后的字符串，其中只显示地址的前 5 个字符和最后 4 个字符，中间用 ... 表示省略部分。

### 4.1.2.4 时间转换

函数将输入的时间(字符串、时间戳或其他有效时间格式)转换为 JavaScript 的 Date 对象，并返回这个 Date 对象。

### 4.1.2.5 转账记录

这些函数主要用于在区块链应用中展示转账历史，用户可以查看所有的转账记录或仅查看与自己相关的转账记录。

### 4.1.2.6 社区

这些函数用于区块链的社区平台，让用户可以查看社区中的所有用户、自己的好友列表，以及与其他用户的消息记录。

## 4.2 上下文管理

useContext 是 React 中的一个 Hook，用于在函数组件中访问由 Context 提供的全局数据或状态。Context 是 React 提供的一种机制，允许我们在组件树中共享数据，而不需要通过每层组件手动传递 props。代码 4-2、4-3 和 4-4 展示了如何使用 React 的 Context API 来管理和共享全局状态，并在不同组件中访问它。

```
import React, { useState, useEffect } from "react";

export const NFTContext = React.createContext();
```

```

export const NFTProvider = ({ children }) => {
  const NFT_MARKETPLACE = "DFashn DF";

  return (
    <NFTContext.Provider value={{ NFT_MARKETPLACE }}>
      {children}
    </NFTContext.Provider>
  );
};

```

代码 4-2 context/NFTContext.js 部分代码

```

import "../styles/globals.css";

// INTERNAL IMPORT
import { NFTProvider } from "../context/NFTContext";

const Marketplace = ({ Component, pageProps }) => {
  return (
    <NFTProvider>
      <Component {...pageProps} />
    </NFTProvider>
  );
};

export default Marketplace;

```

代码 4-3 pages/\_app.js 部分代码

```

import React, { useContext } from "react";

// INTERNAL IMPORT
import { NFTContext } from "../context/NFTContext";

const index = () => {
  const { NFT_MARKETPLACE } = useContext(NFTContext);
  return (
    <div>
      <h1>{NFT_MARKETPLACE}</h1>
    </div>
  );
};

export default index;

```

代码 4-4 pages/index.js 部分代码

在 NFTContext.js 中通过 React 的 useState 和 useEffect 钩子创建并管理一系列全局状态变量。这些变量有当前用户的以太坊账户、账户余额、NFT 市场的拍卖信息、转账历史记录、支持消息、代币销售状态、捐赠合同余额等。useEffect 钩子用于在组件挂载时初始化 NFT 市场、代币销售和捐赠合约的状态，并检查用户是否已连接他们的以太坊钱包。

Context API 在这里被用作共享状态的机制，使得这些全局状态和功能能够在整个 React 应用程序中被不同的组件访问。NFTContext 和 NFTProvider 定义了一个全局的 Context，并通过 NFTProvider 组件将 Context 中的状态和函数暴露给其子组件。

在 NFTContext 中创建合约实例，以便与智能合约进行交互。通过这些实例，合约中的各种函数可以被调用，直接在区块链上执行操作。此外，引入多个实用函数，为 NFT 交易市场、转账历史、支持信息和社区消息等功能提供数据和交互支持，简化了与合约的交互流程，并增强了平台的功能性和用户体验。

# 第五章 前端开发

## 5.1 前端框架

### 5.1.1 框架介绍

使用 React、Next.js 和 Tailwind CSS 三个前端框架来构建数字时尚 NFT 交易平台的用户界面。首先，NFT 交易平台使用 React 作为基础库。React 是一个由 Facebook 开发并开源的 JavaScript 库，专门用于构建用户界面。它允许开发者创建可重用的 UI 组件，并通过组件的组合来构建复杂的用户界面。

其次，Next.js 作为应用框架。Next.js 由 Vercel 开发并开源，是基于 React 的框架，旨在简化构建 React 应用的过程，特别是涉及服务端渲染 (SSR) 和静态网站生成 (SSG) 的应用。我们利用 Next.js 的文件路由和 API 路由功能，实现了前后端的无缝集成，确保了平台的高效运作。

最后，Tailwind CSS 被用作 CSS 框架。Tailwind CSS 是一个实用优先的 CSS 框架，它通过提供一组预定义的类，简化了样式的编写。Tailwind CSS 使得我们能够快速构建响应式、现代化的用户界面，并保持代码的简洁性。

### 5.1.2 具体配置

在本项目的开发过程中，通过配置 tailwind.config.js 文件对 Tailwind CSS 进行了定制化配置，以满足具体的设计需求。在 styles/globals.css 文件中引入字体及 Tailwind 核心样式，使项目实现了全局样式的统一管理与应用。

## 5.2 React 组件

在本项目的开发过程中，创建了多个 React 组件，这些组件被分类为全局组件、主页组件、账户组件、管理组件、表格组件、弹窗组件以及 SVG 图标组件。

以主页中的 3D 模型卡片为例，如代码 5-1 和 5-2 所示。首先，创建了一个名为 ModelViewer 的 React 组件，该组件利用@google/model-viewer 库来实现 3D 模型的渲染。组件接收多个属性，包括 src (3D 模型的源地址)、alt (替代文本)、cameraControls (用

于控制摄像机的布尔值)、autoRotate(是否自动旋转)以及rotationSpeed(旋转速度)。根据传入的属性,组件将自动应用这些设置,并返回一个model-viewer元素来显示3D模型。

其次,创建了名为NFTCard的React组件用于显示NFT的卡片信息。组件使用了useContext钩子从NFTContext获取当前的加密货币符号,并使用useState钩子管理3D模型的自动旋转状态。在该组件中,通过onMouseEnter和onMouseLeave事件切换autoRotate状态,从而实现当鼠标悬停在卡片上时3D模型自动旋转的效果。此外,组件还展示了NFT的名称、价格以及卖家地址,并使用shortenAddress函数对卖家地址进行了简化显示。

```
import React from 'react';
import '@google/model-viewer';

const ModelViewer = ({ src, alt, cameraControls, autoRotate, rotationSpeed }) => {
  return (
    <model-viewer
      src={src}
      alt={alt}
      environment-image="neutral"
      style={{ width: '100%', height: '100%' }}
      {...(cameraControls && { 'camera-controls': true })}
      {...(autoRotate && { 'auto-rotate': true, 'auto-rotate-delay': '0',
        'rotation-per-second': rotationSpeed || '50deg' })}
    ></model-viewer>
  );
};

export default ModelViewer;
```

代码 5-1 ModelViewer.jsx 代码

```
import React, { useState, useContext } from "react";
import Image from "next/image";
import Link from "next/link";
import dynamic from 'next/dynamic';

//INTERNAL IMPORT
import images from "../../assets";
import { NFTContext } from "../../context/NFTContext";
import { shortenAddress } from "../../utils/shortenAddress";
```

```
// 动态加载 model-viewer 组件
const ModelViewer = dynamic(() => import('./ModelViewer'), { ssr: false });

const NFTCard = ({ nft, onProfilePage }) => {
  const { nftCurrency } = useContext(NFTContext);
  const [autoRotate, setAutoRotate] = useState(false);

  return (
    <Link href={{ pathname: "/nft-details", query: nft }}>
      <div
        className="flex-1 min-w-215 max-w-max xs:max-w-none sm:w-full sm:min-w-155
minmd:min-w-256 minlg:min-w-327 dark:bg-nft-black-3 bg-white rounded-2xl p-4 m-4
minlg:m-8 sm:mx-2 cursor-pointer shadow-md"
        onMouseEnter={() => setAutoRotate(true)}
        onMouseLeave={() => setAutoRotate(false)}
      >
        <div className="relative w-full h-52 sm:h-36 xs:h-56 minmd:h-60 minlg:h-300
rounded-2xl overflow-hidden">
          {nft.category === "Image" ? (
            <Image
              src={nft.image || images[`nft${nft.i}`]}
              layout="fill"
              objectFit="cover"
              alt="nft01"
            />
          ) : nft.category === "3D Model" ? (
            <ModelViewer
              src={nft.image}
              alt="3D Model"
              cameraControls={false}
              autoRotate={autoRotate}
            />
          ) : (
            <video
              src={nft.image}
              width={"560"}
              height={"315"}
              color="#705dfc"
              className="createPageVideo"
              controls
            ></video>
          )}
        </div>
      </div>
    </Link>
  );
}

export default NFTCard;
```

```
//省略了显示NFT名称、价格的代码
</div>
</Link>
);
};

export default NFTCard;
```

代码 5-2 NFTCard.jsx 代码

### 5.3 React 页面

项目创建了一系列页面，包括主页、社区、创建 NFT、管理员等等，在这些页面中需要首先设置 `_app.js`。`_app.js` 是 Next.js 框架中的一个特殊文件，用于自定义应用程序的根组件。通过 `_app.js`，可以在所有页面中引入全局样式表、设置全局布局、以及保持某些组件在页面之间的一致性。它的主要作用是为整个应用提供统一的结构和功能，而无需在每个页面中重复代码。

在 `_app.js` 中，如代码 5-3 所示，引入了全局样式并导入 `Navbar`, `Footer`, `Donate`, `DonateModal` 这些全局组件，确保它们在应用的各个页面中都可以被使用。`Navbar` 组件位于页面的顶部。`Component {...pageProps}` 是 Next.js 的一种模式，表示渲染当前页面的内容，这样每个页面都有相应的内容，而公共组件则在页面的头尾部分。

```
import React, { useState } from "react";
import Script from "next/script";
import { ThemeProvider } from "next-themes";
import toast, { Toaster } from "react-hot-toast";

// INTERNAL IMPORT
import "../styles/globals.css";
import { Navbar, Footer, Donate, DonateModal } from "../components/index";
import { NFTProvider } from "../context/NFTContext";

const Marketplace = ({ Component, pageProps }) => {
  const [openDonation, setOpenDonation] = useState(false); // 控制 Donation 模态框的显示
  const [openFooter, setOpenFooter] = useState(true); // 控制 Donate 底栏的显示

  return (
    <NFTProvider>
      <ThemeProvider attribute="class">
```

```
<div className="dark:bg-nft-dark bg-white min-h-screen">
  <Navbar />

  <div className="pt-65">
    <Component {...pageProps} />
  </div>
  <Footer />
  {openFooter && (
    <Donate
      setOpenDonation={setOpenDonation}
      setOpenFooter={setOpenFooter}
    />
  )}
  <Toaster />

  {openDonation && <DonateModal setOpenDonation={setOpenDonation} />}
</div>

<Script src="https://kit.fontawesome.com/d45b25ceeb.js" />
</ThemeProvider>
</NFTProvider>
);
};

export default Marketplace;
```

代码 5-3 代码

## 第六章 最终测试及部署

### 6.1 最终测试

在终端中输入 `npm run dev`，在浏览器中访问 `http://localhost:3000`，可以看到数字时尚 NFT 交易平台成功运行。点击导航栏的“create”按钮，选择“3D Model”类型并上传一个 3D 模型，可以看到上传的 3D 模型在界面中缓缓旋转，如图 6-1 所示。输入 NFT 的名字、描述、价格和原生代币价格后，点击“Create Item”按钮。此时，MetaMask 会自动弹出，并提示支付相应的 Gas 费用，支付完成后会自动跳转回主页。

此时，可以看到刚刚上传的 NFT 已成功显示在主页上，当鼠标悬浮在 NFT 卡片上时，对应的 NFT 会自动旋转，如图 6-2 所示。切换 MetaMask 账户并刷新网页后，点击刚才新建的 NFT，进入 NFT 详情页面。在 NFT 详情页面可以控制 NFT 旋转，并进行购买操作。此外，交易平台也有黑暗模式，如图 6-3 所示。

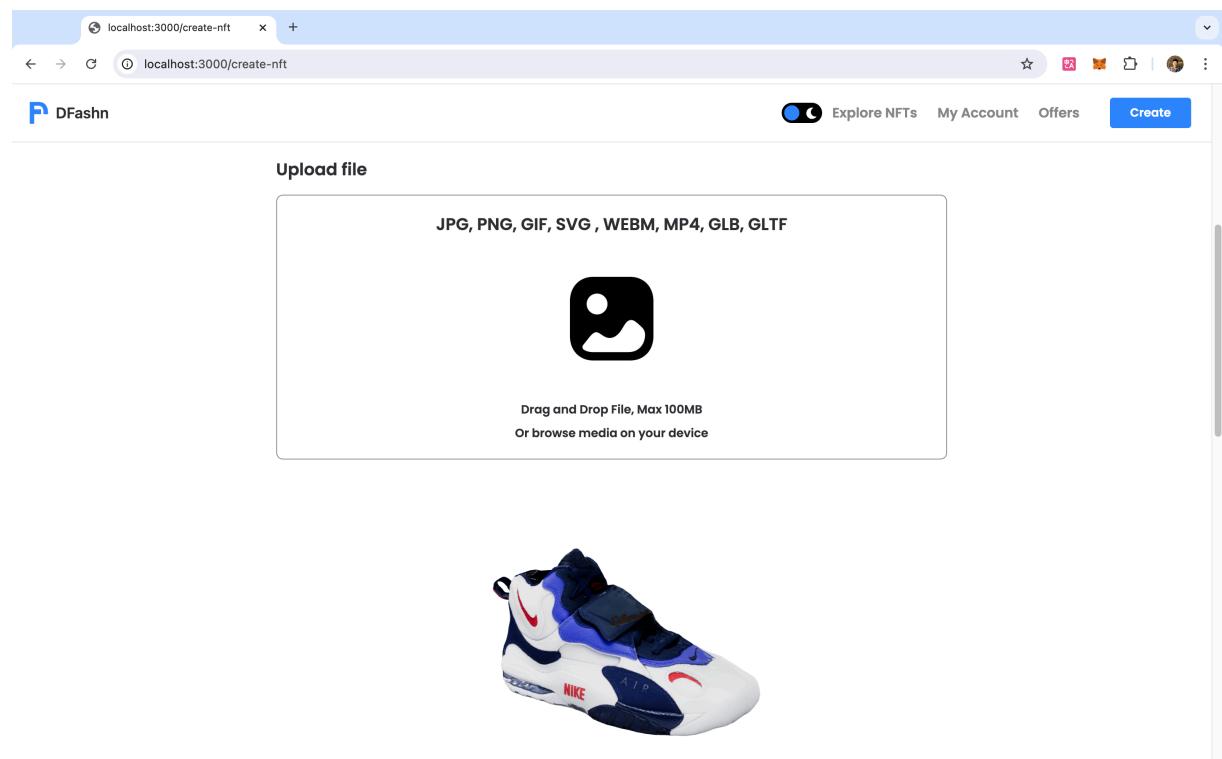


图 6-1 上传 NFT

## 结论

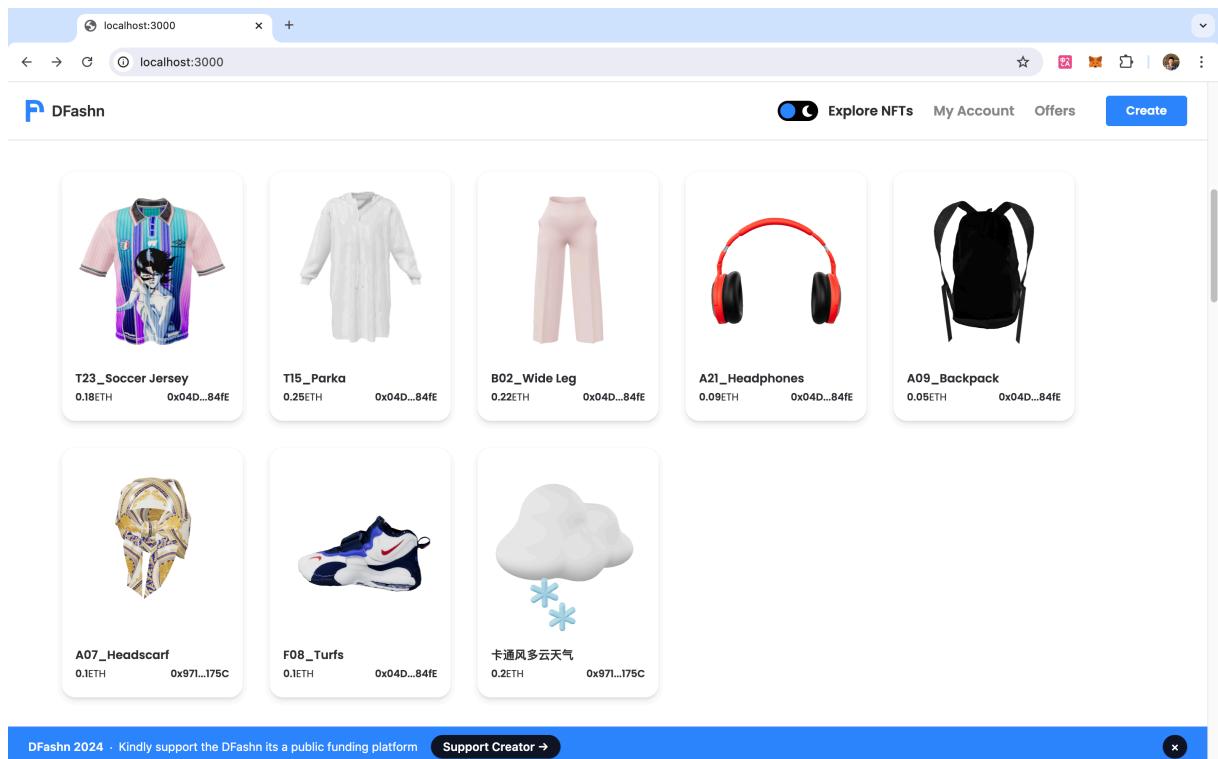


图 6-2 主页上的 NFT

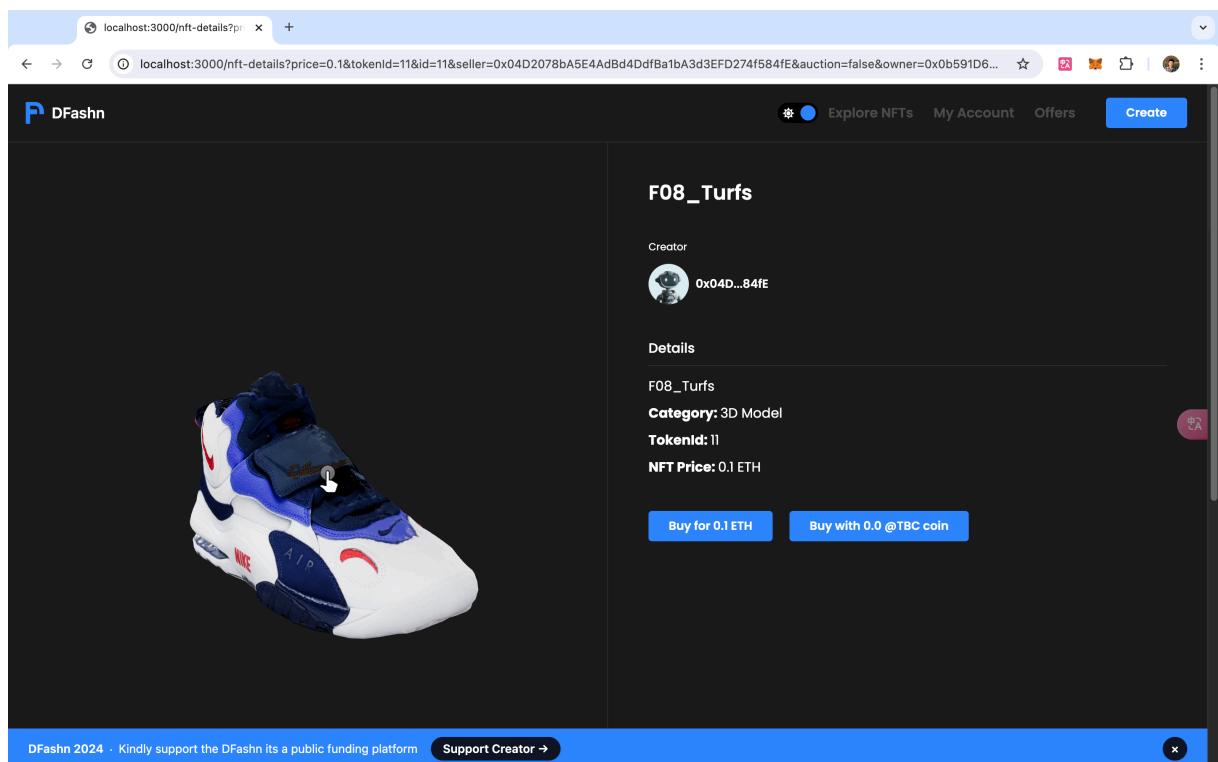


图 6-3 NFT 详情

## 6.2 Vercel 部署

测试所有功能没有问题后，在 Github 新建一个私人仓库并上传所有代码，如图 6-4 所示。代码上传完成后，前往 Vercel 平台进行项目部署。Vercel 是一款支持前端应用快速部署的云平台，尤其适用于 Next.js 等现代 Web 框架。在 Vercel 仪表板中选择“New Project”选项，系统会提示连接 GitHub 账户。授权后，Vercel 将显示 GitHub 上的所有仓库，选择刚才创建的私人仓库以开始部署，如图 6-5 所示。

在选择仓库后，Vercel 会自动检测项目的配置文件，并识别使用的框架。对于 Next.js 项目，Vercel 通常能够自动识别并配置构建命令和输出目录。如果需要自定义构建设置，例如环境变量、特殊构建命令或自定义域名等，可在 Vercel 的“Project Settings”页面进行相应配置。

确认所有配置无误后，点击“Deploy”按钮以开始部署过程。Vercel 将首先拉取代码并进行依赖安装，然后执行构建命令，生成静态文件及必要的服务端代码。整个构建过程通常只需几分钟时间。构建完成后，Vercel 会自动将项目部署至其全球 CDN，并生成一个唯一的域名，用于访问部署好的应用。

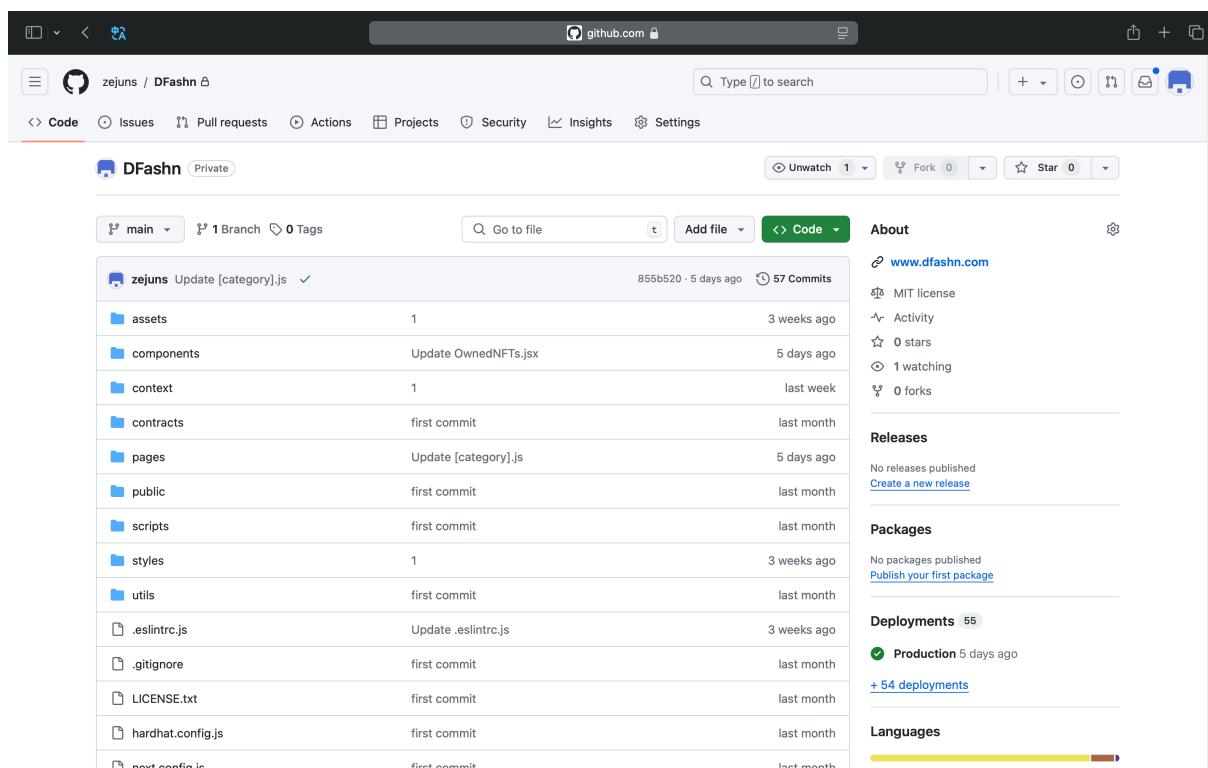


图 6-4 新建 Github 仓库并上传代码

## 结论

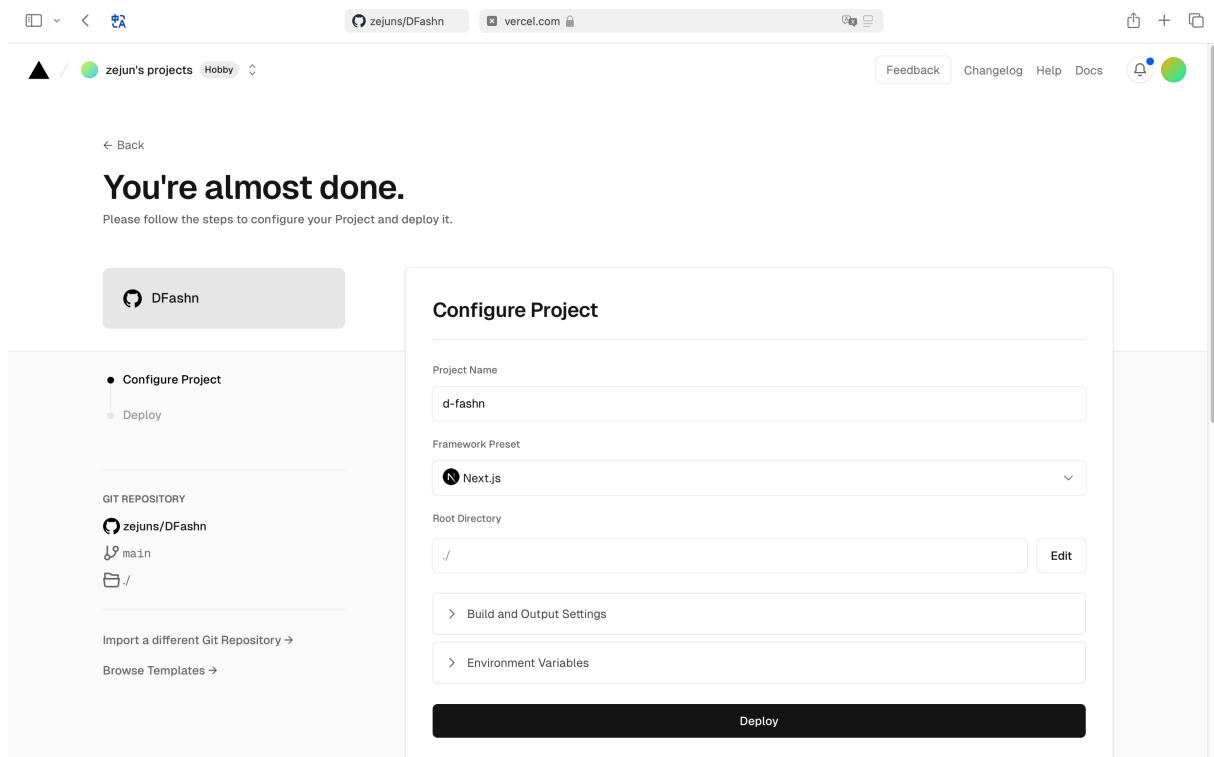


图 6-4 Vercel 部署

## 结论

### 1. 论文工作总结

本论文介绍了数字时尚 NFT 交易平台的设计与实现。项目旨在创建一个集成区块链技术和智能合约的数字时尚平台，提供安全、透明的 NFT 交易环境。系统设计涵盖前端和后端两部分，前端使用 React、Next.js 和 Tailwind CSS 构建用户界面，后端依托以太坊区块链实现业务逻辑。论文详细描述了智能合约的设计与实现，涵盖合约的结构、功能和安全性测试。

论文还探讨了实用函数和上下文管理的实现方法，介绍了 React 的 Context API 在全局状态管理中的应用，并提供了多种实用函数用于简化常见任务的实现。前端开发部分展示了如何利用选定的框架和组件构建用户界面，包括 3D 模型展示和 NFT 卡片的交互效果。最后，论文总结了项目的最终测试和部署过程，通过本地测试和 Vercel 平台部署，确保了平台的稳定性和可访问性。

基于区块链技术的 NFT 交易平台不仅提升了数字时尚交易的安全性和透明度，还通过高效的前端和智能合约设计提供了良好的用户体验。

### 2. 工作展望

论文提出的数字时尚 NFT 交易平台为用户提供了一个创新的交易和展示虚拟时尚产品的空间。然而，随着技术的不断发展，平台在未来还可以引入更多前沿功能，以进一步提升用户体验和市场竞争力。

首先，未来的工作将集中在利用人工智能技术，通过图像生成 3D 模型的功能。用户只需上传一张图片，系统便可自动生成与之对应的 3D 模型，这一功能将极大简化虚拟时尚产品的创建过程。其次，平台将探索在虚拟现实（VR）中实现真实穿戴体验的可能性，用户可以通过 VR 设备直观地体验所购买的 NFT 产品，增强互动感和沉浸感。此外，平台计划支持用户上传自己的照片，并通过 AI 技术将购买的 NFT 产品渲染到用户的照片上，提供个性化的展示效果。

除了交易功能外，平台还将扩展其社交和游戏功能，构建一个集交流、娱乐与购物为一体的综合性数字时尚生态系统。通过社交功能，用户可以在平台上分享自己的虚拟

## 结论

---

时尚作品，与他人互动；通过游戏功能，用户可以在虚拟世界中参与到时尚相关的游戏活动中，进一步增强用户粘性和平台活跃度。

这些功能的引入将有助于平台在数字时尚领域中保持领先地位，并为用户带来更加丰富和多样化的数字体验。

## 参考文献

- [1] Wang H, Ning H, Lin Y, et al. A survey on the metaverse: The state-of-the-art, technologies, applications, and challenges[J]. IEEE Internet of Things Journal, 2023, 10(16): 14671-14688.
- [2] Hoffmann J F. Property Rights in Money (and Cryptocurrencies)[J]. European Property Law Journal, 2022, 11(3): 133-157.
- [3] 李佳珈,田景宏,宋雨涵,等.中国虚拟服装的发展现状研究[J].纺织报告,2024,43(05):41-43.
- [4] Han M, Yan M, Li J, et al. Generating uncertain networks based on historical network snapshots[C]//Computing and Combinatorics: 19th International Conference, COCOON 2013, Hangzhou, China, June 21-23, 2013. Proceedings 19. Springer Berlin Heidelberg, 2013: 747-758.
- [5] Duan Z, Yan M, Cai Z, et al. Truthful incentive mechanisms for social cost minimization in mobile crowdsourcing systems[J]. Sensors, 2016, 16(4): 481.
- [6] Li X, Jiang P, Chen T, et al. A survey on the security of blockchain systems[J]. Future generation computer systems, 2020, 107: 841-853.
- [7] Tosh D K, Shetty S, Liang X, et al. Security implications of blockchain cloud with analysis of block withholding attack[C]//2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). IEEE, 2017: 458-467.
- [8] Gilad Y, Hemo R, Micali S, et al. Algorand: Scaling byzantine agreements for cryptocurrencies[C]//Proceedings of the 26th symposium on operating systems principles. 2017: 51-68.
- [9] Gangwal A, Gangavalli H R, Thirupathi A. A survey of Layer-two blockchain protocols[J]. Journal of Network and Computer Applications, 2023, 209: 103539.
- [10]Zheng Z, Xie S, Dai H, et al. An overview of blockchain technology: Architecture, consensus, and future trends[C]//2017 IEEE international congress on big data (BigData congress). Ieee, 2017: 557-564.
- [11]Joshi A P, Han M, Wang Y. A survey on security and privacy issues of blockchain technology[J]. Mathematical foundations of computing, 2018, 1(2).

- [12]Johnson D, Menezes A, Vanstone S. The elliptic curve digital signature algorithm (ECDSA)[J]. International journal of information security, 2001, 1: 36-63.
- [13]Swanson T. Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems[J]. Report, available online, 2015, 28.
- [14]Wood G. Ethereum: A secure decentralised generalised transaction ledger[J]. Ethereum project yellow paper, 2014, 151(2014): 1-32.
- [15]Zamfir V. Introducing casper the friendly ghost[J]. Ethereum Blog, 2015.
- [16]Castro M, Liskov B. Practical byzantine fault tolerance[C]//OsDI. 1999, 99(1999): 173-186.
- [17]Wohrer M, Zdun U. Smart contracts: security patterns in the ethereum ecosystem and solidity[C]//2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE). IEEE, 2018: 2-8.
- [18]Daulat Hussain. Pro NFT Marketplace[EB/OL]. theblockchaincoders, n.d. [2024-08-12].  
<https://www.theblockchaincoders.com/pro-nft-marketplace>.
- [19]VOGELSTELLER F. ERC20: Ethereum Token Standard[EB/OL].  
<https://eips.ethereum.org/EIPS/eip-20>, 2015-11-19.

## 致谢

在这篇论文的完成过程中，我要特别感谢我的指导老师解老师。他在论文选题、外文翻译以及论文方向的把控上给予了我无私的帮助和耐心指导，使我能够在每一个关键节点上得到及时的引导和支持。正是因为解老师的悉心教导，我才能顺利完成这项研究，并且在学术思维和专业素养上得到了极大的提升。

此外，我还要感谢 Daulat Hussain 的 Pro NFT Marketplace 课程。这是一门让我迈入区块链与 NFT 领域的入门课程。在学习过程中，我不仅学到了关于智能合约和 NFT 交易市场的基础知识，更加深了我对数字艺术与区块链技术结合的理解。从 Solidity 到 React/Next Js，从 Tailwinds 到 Hardhat，以及 MetaMask 和 Web3 Provider 等知识，涵盖了 NFT 市场开发所需的各个方面。这门课程为我的研究打下了坚实的基础，并激发了我对这一新兴领域的浓厚兴趣。

最后，我要深深感谢我的父母和亲人，正是他们无尽的支持与鼓励，给了我坚持到底的动力。他们在我学术道路上的关爱和理解，是我克服困难、取得进步的重要支柱。在此，我也感谢所有在论文写作过程中给予帮助和支持的师长、朋友，以及所有提供过建议和鼓励的人士。正是有了你们的帮助，我才能顺利完成这篇论文，并在这个过程中不断成长。