分类号：TP315.5　　U D C：D10621-408-(2025)1924-0
密　级：公 开　　编 号：2021131028

成 都 信 息 工 程 大 学
学 位 论 文

基于链上随机数和时间锁的彩票系统设计与实现

论文作者姓名：翁忠旭
申请学位专业：区块链工程
申请学位类别：工学学士
指导教师姓名(职称)：梁培利（讲师）
论文提交日期：2025 年 05 月 10 日

基于链上随机数和时间锁的彩票系统设计与实现

摘要：传统彩票系统存在中心化运营、透明度不足、易受人为操控等问题，本课题研究结合链上随机数生成方案和时间锁机制，设计并实现一个基于区块链的彩票系统，旨在解决传统彩票系统的弊端，提高彩票的公平性和透明度，增强用户信任。本项目通过与成熟的分布式随机数生成网络在区块链网络上实现链上交互获取可信的、可验证的随机数，并利用区块时间戳实现时间锁功能来确保系统的公平性与不可预测性。系统开发过程中，为了方便智能合约代码开发，我选择了Hardhat 测试框架并采用测试驱动开发模式确保合约代码的质量，前端采用 react 框架完成与智能合约的业务交互逻辑处理与用户展示页面的开发。彩票生命流程中除创建彩票阶段之外，一切皆由智能合约自动管理后续流程的进行，通过该系统，用户可购买具有足够的公正性和透明度的彩票。

关键词： 彩票系统；智能合约；随机数；时间锁

Design and Implementation of Lottery System Based on Random Number and Time Lock on Blockchain

Abstract: The traditional lottery system has problems such as centralized operation, insufficient transparency, and vulnerability to human manipulation. This research combines the random number generation scheme on the chain and the time lock mechanism to design and implement a blockchain-based lottery system. The purpose is to solve the drawbacks of the traditional lottery system, improve the fairness and transparency of the lottery, and enhance user trust. This project obtains credible and verifiable random numbers by interacting with the mature distributed random number generation network on the blockchain network, and uses the block timestamp to implement the time lock function to ensure the fairness and unpredictability of the system. In the process of system development, in order to facilitate the development of smart contract code, I chose the Hardhat test framework and adopted the test-driven development mode to ensure the quality of the contract code. The front-end uses the react framework to complete the business interaction logic processing with the smart contract and the development of the user display page. In addition to the lottery creation stage in the lottery life process, everything is automatically managed by the smart contract. Through this system, users can purchase lottery tickets with sufficient fairness and transparency.

Key words： Lottery System ; Smart Contract ; Random Number ; Time Lock

目 录

论文总页数：48 页

1 引言.....1
1.1 课题背景.....1
1.2 国内外研究现状.....1
1.3 基于链上随机数和时间锁的彩票系统项目研究的意义.....2
2 基于链上随机数和时间锁的彩票系统需求分析.....3
2.1 系统总体需求分析.....3
2.2 系统功能需求分析.....3
2.2.1 彩票发行与购买.....3
2.2.2 链上随机数生成.....4
2.2.3 时间锁机制.....5
2.2.4 开奖与派奖.....5
2.2.5 奖金管理.....6
2.3 系统业务流程分析.....7
2.4 系统非功能需求分析.....7
3 基于链上随机数和时间锁的彩票系统设计.....8
3.1 系统总体结构设计.....8
3.2 系统开发运行环境设计.....9
3.3 系统功能设计.....9
3.4 系统数据存储结构设计.....12
3.4.1 E-R 模型设计.....12
3.4.2 数据表信息.....14
3.4.3 存储过程信息.....17
3.5 系统关键类设计.....19
4 基于链上随机数和时间锁的彩票系统系统的实现.....21
4.1 彩票发行功能.....21
4.2 开奖功能.....25
4.3 奖金管理功能.....29
4.4 交易记录查询功能.....31
4.5 系统实现小结.....33
5 基于链上随机数和时间锁的彩票系统系统测试.....35
5.1 测试环境搭建.....35
5.2 功能测试.....35
5.2.1 彩票发行测试.....35
5.2.2 开奖功能测试.....38
5.2.3 奖金管理功能测试.....39
5.2.4 历史记录查询功能测试.....41
5.3 性能测试.....43
5.4 测试总结.....44

结 论.....45
参考文献.....46
致 谢.....47
声 明.....48

1 引言

1.1 课题背景

随着信息技术的发展，尤其是在大数据、云计算等新兴技术逐步应用于彩票行业的背景下，传统的彩票系统在技术和信任机制方面的局限日益显现。传统彩票系统依赖中心化的运营机构掌控发行、销售、开奖等各个环节，虽然具备一定的运营效率，但也因其高度中心化而频频暴露出透明度低、公平性差、容易受到人为干预等问题[1]。这些问题严重削弱了公众对彩票系统的信任，进而影响了整个行业的健康发展。

近年来，区块链技术的迅猛发展为解决上述问题提供了新的可能。区块链以其去中心化、不可篡改、可追溯等技术特性，天然适合构建一个更加公开、公平、透明的彩票系统[2] [3][3]。在区块链环境下，所有交易数据及规则皆可上链记录，任何人都可查验开奖过程与结果，从而有效提升系统的公信力与透明度。智能合约作为区块链应用的核心组成部分，可实现彩票规则的自动执行，杜绝人为干预，进一步提升系统的自动化程度与运行效率[4]。

然而，在区块链上构建一个真正安全、可靠且具备实用价值的彩票系统仍面临诸多技术挑战。其中最关键的问题之一是如何生成安全、公正且不可预测的随机数[5] [6]。由于区块链节点之间的公开性，链上生成随机数若处理不当，容易受到攻击者预测或干扰。此外，如何在彩票系统中合理引入时间锁机制以控制关键流程的执行时机，保障用户利益与系统稳定性，也是当前亟需研究的重要方向[7]。

通过将链上随机数生成机制与时间锁机制有机结合，彩票系统可以实现从购买彩票到开奖全过程的自动化与可信化。相较传统系统，这种新型架构能够有效增强用户的参与信任，提高操作透明度，并显著降低因人为失误或恶意行为带来的系统风险。

1.2 国内外研究现状

在国际范围内，基于区块链的彩票系统已经有了较为成熟的探索和实践。以以太坊平台为代表，不少项目尝试利用智能合约实现彩票逻辑的自动执行，其中"PoolTogether"是较为典型的项目之一[8]。该项目采用“无损彩票”模式，即用户购买彩票的资金不会被消耗，而是用于参与 DeFi 产品获得收益，最终以智能合约分配收益者，从而创新了传统彩票的盈利与激励模型。此外，一些项目采用了 Chainlink 的去中心化随机数服务（VRF）作为随机数生成方案，以保障随机性的安全性和不可预测性[9] [10]。

相较之下，国内在区块链彩票系统方面的研究尚处于起步阶段。部分学者已关注区块链在彩票领域的应用潜力，并提出了理论模型。例如，罗志坚等人探讨

了区块链技术在彩票系统中的实际应用前景，并设计了一种初步的区块链彩票模型[11]。张涛等人则从系统架构角度出发，重点分析了如何通过智能合约保障系统公平性、透明性及抗篡改能力。

在随机数生成方面，尽管链上环境天然公开透明，但这也带来了生成机制易被预测或攻击的问题。目前常见的解决方案包括结合链下预言机（如 Chainlink VRF、Drand 等）提供的随机性服务，实现链上与链下数据的安全融合[12] [13]。时间锁机制方面，区块链系统通常通过区块高度或时间戳来控制特定合约函数的执行时间，从而确保流程的可控性与时效性[14]。

从开发方法论来看，当前区块链系统开发趋向于采用敏捷开发与测试驱动开发（TDD）模式，以提升开发效率与系统稳定性。本课题中所构建的彩票系统以以太坊为基础，采用 Solidity 语言编写核心智能合约逻辑，借助 OpenZeppelin 提供的安全合约库保障合约安全性，并使用 Hardhat 作为开发测试框架进行持续集成和部署验证。前端采用 React 框架构建用户界面，借助 Web3.js 实现前后端与区块链的交互通信，从而实现良好的用户交互体验。

1.3 基于链上随机数和时间锁的彩票系统项目研究的意义

本课题旨在设计与实现一个基于链上随机数和时间锁机制的彩票系统，依托区块链技术提升系统的透明度、公平性和自动化水平。通过引入可信的随机数生成机制，确保开奖过程不被人为干预[15]；通过设置时间锁机制，有效规避信息不对称带来的提前泄露与作弊风险，从根本上提升系统的安全性与可靠性。

本系统不仅在应用层面探索了区块链技术在传统行业中的落地方式，也在理论层面对链上随机数生成与时间控制机制进行了深入研究。通过将这些关键技术应用于实际的彩票系统设计中，为后续区块链应用的创新与实践提供了技术基础与设计参考。此外，系统的开发过程采用模块化设计思路，结合现代前端框架与开发工具，有效提升了系统的可维护性与扩展能力。

从社会意义来看，区块链彩票系统有望通过技术手段打破原有信任壁垒，引导用户理性参与，降低因不信任带来的参与门槛，从而促进彩票行业健康、规范、有序的发展。更重要的是，本系统的构建与研究不仅仅是对现有技术的简单应用，更是一次对技术信任机制重塑路径的有益尝试，对推动区块链技术在更广泛领域的应用具有一定的现实意义与前瞻价值。

2 基于链上随机数和时间锁的彩票系统需求分析

2.1 系统总体需求分析

该系统主要功能包括彩票发行、购买、开奖及奖金分配。开奖过程通过区块链的随机数技术和时间锁机制可确保公平性与不可预测性。因区块链部署特性，智能合约需保障资金安全、确保开奖透明以及系统稳定运行。

用户可通过界面连接或退出加密钱包，用户也可查看可购买彩票和已购彩票信息，其中包括购买记录和票号等。用户也能自主选择购票数量，购票后系统会自动记录相关信息并更新奖池总额。

在开奖时间到达后，系统管理员须创建开奖交易，然后智能合约自动完成链上随机数生成过程，即通过外部随机数提供商（即预言机）从分布式随机数生成网络中获取安全可靠的随机数，利用随机数通过 Fisher-Yates 洗牌算法洗混已购买的票号的顺序，根据中奖人数将排列在存储票号数组前面的序号作为中奖者。开奖前时间锁机制能确保彩票只能在预定时间后进行，防止提前开奖。开奖后，智能合约会自动计算中奖金额同时实现中奖者的奖金分配。中奖用户可以浏览器前端页面上查询开奖结果和领取奖金。

每一期彩票结束后，系统管理员能创建新一期彩票，包括设置彩票的价格和奖项结构。同时管理员还需确保彩票能在开奖时间到达后创建开奖交易触发开奖，并监控系统运行状态。同时管理员需要确保系统安全稳定运行，及时处理异常情况，保障用户资金安全和系统公平性。

2.2 系统功能需求分析

本系统的主要用户群体是区块链彩票参与者，包括彩票购买用户和系统管理员。这些用户具备基本的区块链交互知识，能够使用加密钱包进行交易。系统设计基于智能合约的运行机制，满足彩票发行、购买、开奖和奖金分配的全流程需求，同时保证随机性和公平性。

2.2.1 彩票发行与购买

彩票发行与购买模块包括彩票信息管理、用户购票管理和奖池资金管理等，彩票发行与购买用例图如图 2.1 所示。

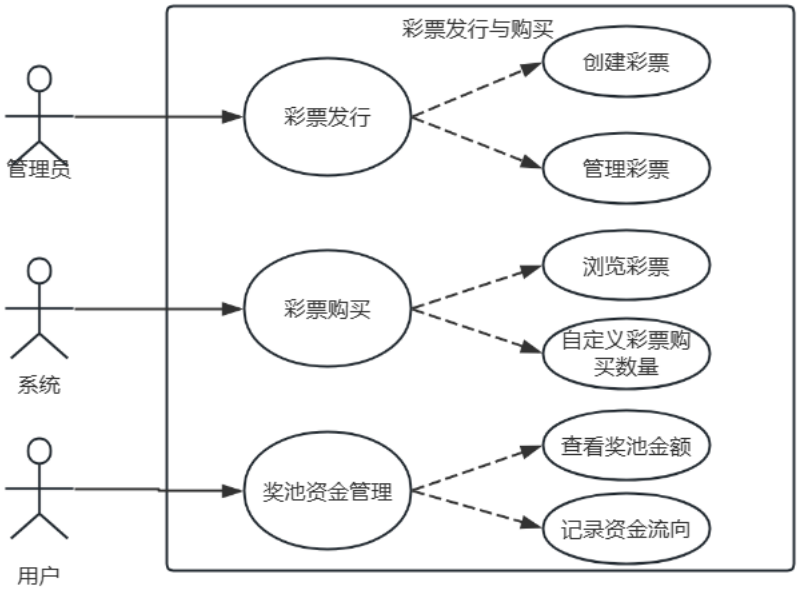


图 2.1 彩票发行与购买用例图

管理员能够实现对彩票信息的管理，功能包括创建新一期彩票、设置彩票单价、配置开奖时间、定义彩票总量、管理彩票销售状态。用户能够实现对彩票购买的管理，功能包括浏览当前可用彩票、查看彩票详情（价格、开奖时间、奖池金额）、选择购买数量、使用钱包支付购买费用、获取购票凭证。系统能够实现对奖池资金的管理，功能包括累计用户购票资金、查看奖池金额、记录资金流向。

2.2.2 链上随机数生成

链上随机数生成模块包括随机源获取、随机数验证和随机结果管理等，链上随机数生成用例图如图 2.2 所示。

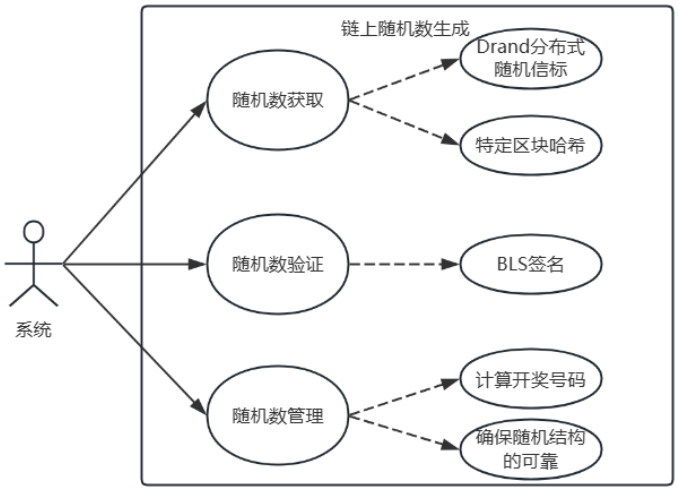


图 2.2 链上随机数生成用例图

系统能够实现对随机源的获取，功能包括接入 Drand 分布式随机信标、利用 api 获取可靠的随机数、读取特定区块哈希、综合多随机源最终生成最终随机种子。系统能够实现对随机结果的管理，功能包括将随机种子转换为开奖号码和确保随机结果不可预测和不可操控。

2.2.3 时间锁机制

时间锁机制模块包括时间锁设置、锁定状态管理和解锁触发管理等，时间锁机制用例图如图 2.3 所示。

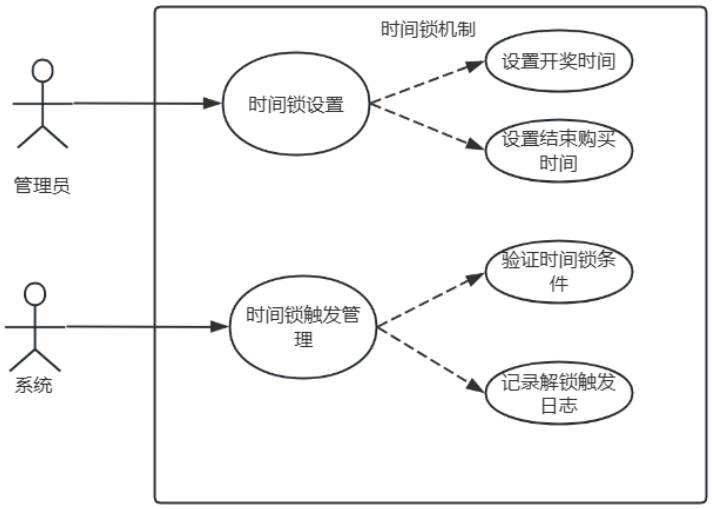


图 2.3 时间锁机制用例图

管理员能够实现对时间锁的设置，功能包括配置彩票销售截止时间和设定开奖时间点。系统能够实现对锁定状态的管理，功能包括销售截止后锁定彩票信息、锁定随机数参数、冻结奖池资金。系统能够实现对解锁触发的管理，功能包括验证当前时间是否达到开奖条件和记录解锁触发证明。

2.2.4 开奖与派奖

开奖与派奖模块包括中奖号码生成、中奖结果验证和奖金分配管理等，开奖与派奖用例图如图 2.4 所示。

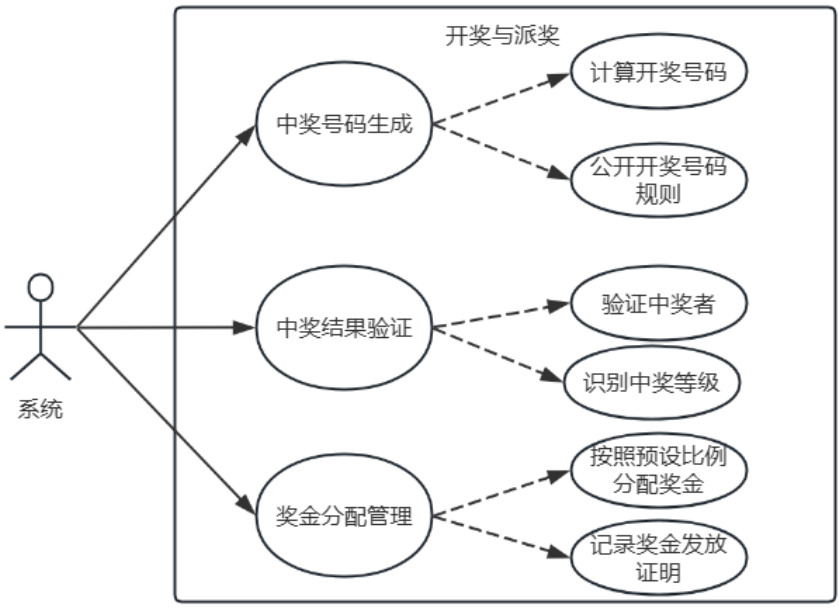


图 2.4 开奖与派奖用例图

系统能够实现对中奖号码的生成，功能包括根据最终随机种子计算开奖号码、公开号码生成规则、广播开奖结果。系统能够实现对中奖结果的验证，功能包括匹配用户彩票与中奖号码、识别中奖等级、生成中奖明细报告。系统能够实现对奖金分配的管理，功能包括按照预设比例分配奖金、计算每位中奖者应得奖金、记录奖金发放证明（交易哈希存储）。

2.2.5 奖金管理

奖金管理模块包括奖池管理和奖金查询管理等，奖金管理用例图如图 2.5 所示。

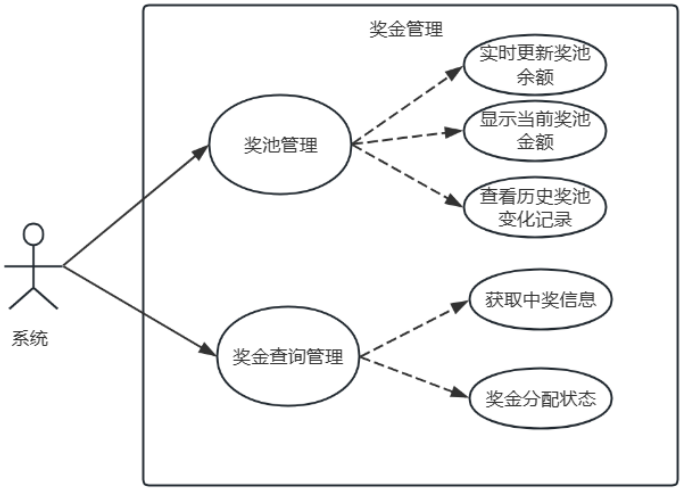


图 2.5 奖金管理用例图

系统能够实现对奖池的管理，功能包括实时查看历史奖池变化记录、更新奖池余额和显示当前累积金额。系统能够实现对资金流向的管理，功能包括记录每笔购票资金入账、追踪奖金发放流向、确保资金安全。用户能够实现对奖金查询的管理，功能包括获取中奖信息和奖金分配状态。

2.3 系统业务流程分析

用户首先通过钱包登录系统，可查看最新一期的彩票信息和历史彩票信息，若当前彩票可购买，用户可选择自定义购买彩票的数量。用户也可在个人中心查询之前购买的彩票票号，若中奖则可领取奖金。系统管理员负责设置每一期彩票的价格、日期和奖项结果，并在每一期彩票到达开奖时间后触发开奖。智能合约负责每一期彩票创建之后的记录交易记录、负责开奖和奖金分配等流程。系统业务流程图如图 2.6 所示

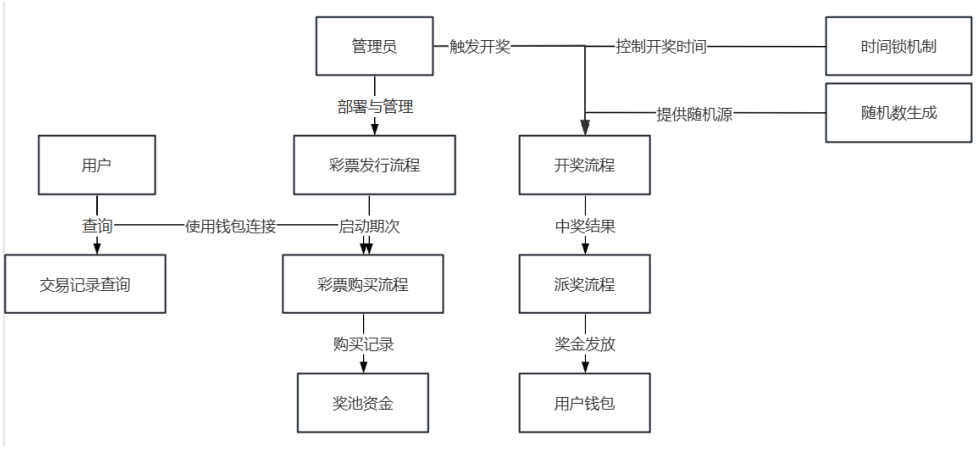


图 2.6 系统业务流程图

2.4 系统非功能需求分析

（1）性能需求：本系统因需处理彩票创建、购买、开奖等交易，因此需支持每秒至少 25 笔链上交易来满足需求。部署智能合约的区块链网络的交易确认时间应控制在 15-25 秒内。需至少支持 50 个并发用户操作，特别是在购买彩票金额查询交易记录时。同时智能合约需优化 gas 费用，降低用户支付成本

（2）安全性需求：由于本系统涉及到大量资金转移，因此安全性至关重要。首先需保证权限的处理，防止越权情况的发生。同时需避免常见的攻击漏洞，以防合约资金被转移。最后需确保每一笔关于资金的交易都能正确完整的执行。

（3）稳定性需求：区块链彩票系统作为一种全天候服务的金融应用，其稳定性是赢得用户信任与保持市场竞争力的关键因素。因此系统需保证即使是在高峰期，也需要保证每笔交易的正确处理，不出现任何错误或故障。

3 基于链上随机数和时间锁的彩票系统设计

3.1 系统总体结构设计

系统采用区块链技术、智能合约与前端框架结合的架构模式，从底层到顶层分为数据层、数据持久层、业务逻辑层和表现层。系统总体结构如图 3.1 所示

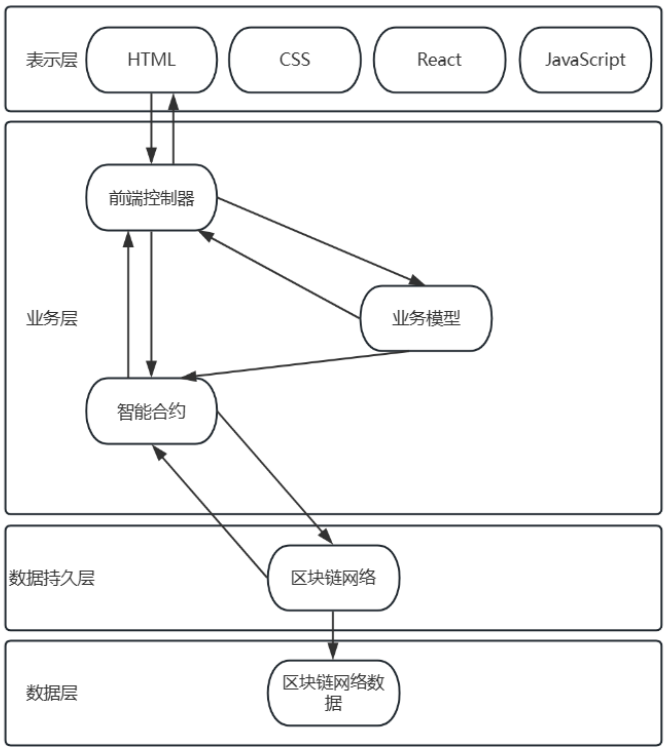


图 3.1 系统总体结构设计图

（1） 表现层：

该层主要采用 HTML、CSS、JavaScript 以及 React 框架作为前端的主要技术，给用户提供了一个干净、整洁、易于操作的前端页面，该页面功能丰富，UI 设计美观，给予了用户美好的观感。

（2） 业务层：

业务层是系统的决策中枢，负责连接用户与智能合约。前端控制器负责引导用户请求，调用智能合约。其中模型组件定义数据结构，确保信息有序流动。智能合约作为系统中枢，负责核心业务逻辑。

（3） 数据持久层和数据层：

负责记录交易记录，智能合约将交易记录存储在区块链网络上，同时保存彩

票信息、购奖信息等系统关键信息，以便于数据随时检索，确保数据的真实性与不可篡改。

综上，本系统通过使用各种先进的技术和理念，为用户提供了便捷、高效且安全的彩票参与环境。

3.2 系统开发运行环境设计

（1）硬件环境：

对于本彩票系统，选择阿里云 ECS 作为主要服务器环境。采用计算型实例规格（ecs.c6.4xlarge），配置为 8 核 vCPU、16GB 内存，以满足区块链节点运行和智能合约处理的高计算需求。存储选择高性能 SSD 云盘，初始容量为 1.5TB，其中 150GB 用于系统盘，1350GB 用于数据盘，保证区块链数据的高速读写和完整存储。操作系统采用 Ubuntu 20.04 LTS 服务器版，该版本具有长期支持特性，确保系统稳定性和安全更新。

（2）软件环境：

本系统采用 Solidity 语言开发智能合约，JavaScript 语言开发前端页面，所以需安装 Solidity 语言和 JavaScript 语言的最新稳定版。为测试系统，安装 Edge 或 Chrome 浏览器并安装 Metamask 钱包插件用于系统的登录。

（3）网络环境；

因需要与区块链网络连接和应对可能出现的大规模的用户访问，需建立高速稳定的网络连接以提供尽可能大的带宽，同时需配置防火墙等安全设备，来防止可能出现的 DDoS 等攻击。

（4）开发工具：

采用 Hardhat 测试框架辅助智能合约的开发，采用 React 前端框架开发前端页面，利用 VSCode 作为代码编辑器。同时利用 Git 管理代码版本历史记录，方便代码的管理。

3.3 系统功能设计

为实现彩票系统的所有功能，系统分为以下模块：彩票发行、开奖、奖金管理和交易记录查询功能模块。详见表 3.1。

表 3.1 功能模块列表

模块编号	模块名称	对应需求功能编号	所对应需求功能	实现优先级
DS_BLSRT01	彩票发行	SRS_SYS1.01	彩票发行	高
DS_BLSRT02	开奖	SRS_SYS2.01, SRS_SYS2.02	开奖随机数生成，派奖执行	高

续表 3.1 功能模块列表

模块编号	模块名称	对应需求功能编号	所对应需求功能	实现优先级
DS_BLSRT03	奖金管理	SRS_SYS3.01, SRS_SYS3.02	奖金池管理，奖金分配记录	高
DS_BLSRT04	交易记录查询	SRS_SYS4.01, SRS_SYS4.02	购彩记录查询，中奖记录查询	高

以下是对各功能模块的详细描述

（1）彩票发行模块：

管理员账号登录系统后，导航栏会出现管理中心，管理员在管理中心负责每一期彩票的发行，填写彩票的开始时间、结束时间、开奖时间、彩票单价和奖项结构。用户可购买彩票获得票号。

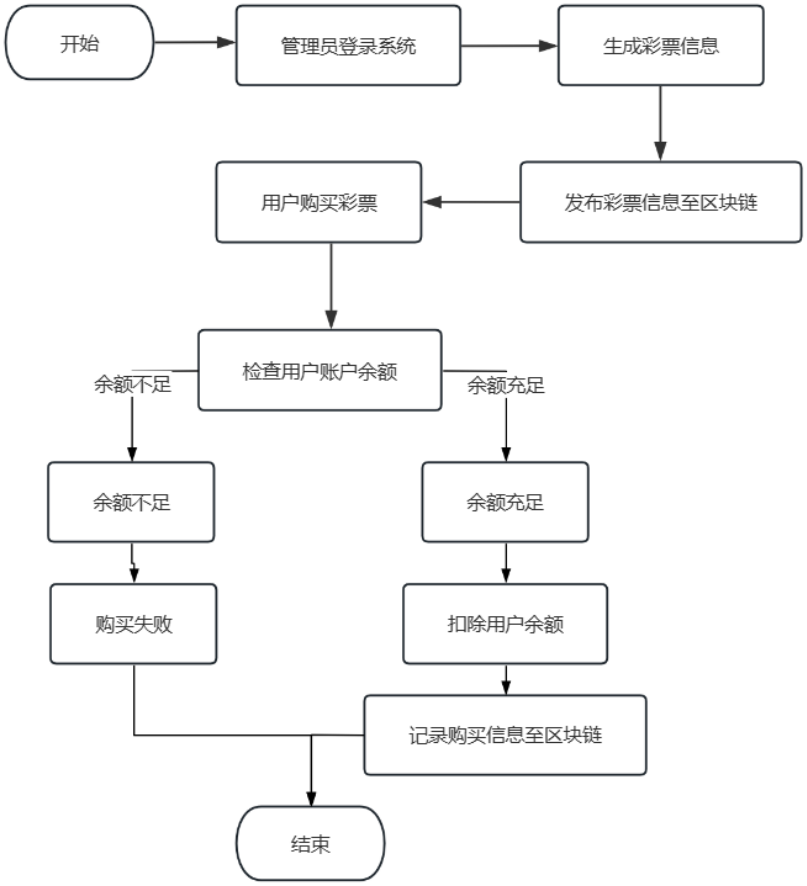


图 3.2 彩票发行模块业务流程图

（2）开奖模块：

彩票到达开奖时间后，管理员进入管理中心页面点击当前一期彩票的开奖按

钮与智能合约发起交易，智能合约自动检测当前时间是否在开奖时间之后，以及彩票购买量是否达到开奖要求，若通过，则与随机数提供商合约交互获取可靠的随机数，再利用随机数对票号利用 Fisher-Yates 洗牌算法获取中奖彩票号，找出中奖者。

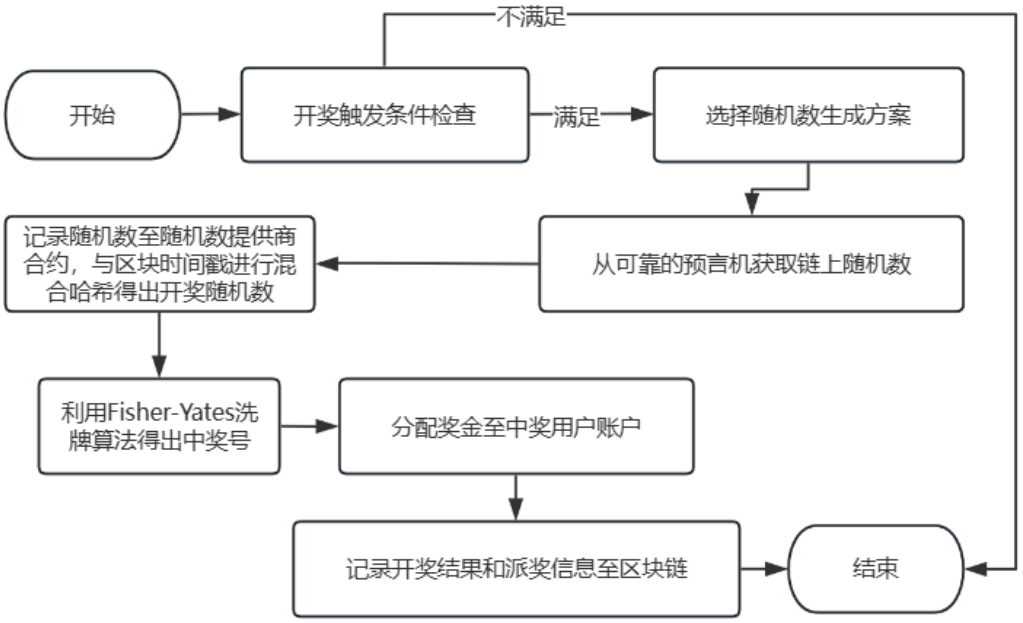


图 3.3 开奖模块业务流程图

（3）奖金管理模块：

该模块负责奖金的分发与奖池信息的更新。当开奖过程中出现彩票购买量没达到开奖要求会自动启动退款处理，当开奖完成后会自动根据奖项等级分配给中奖者奖金。有人购买彩票时负责实时更新奖池金额。

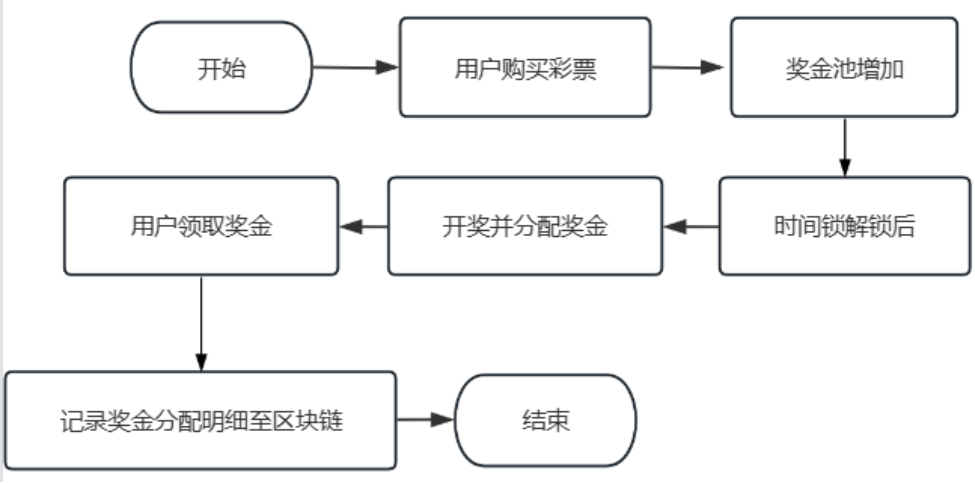


图 3.4 奖金管理模块业务流程图

（4）交易记录查询模块：

该模块的主要功能是提供每期彩票的交易记录查询，能让用户知道自己购买彩票的交易记录以及自己是否中奖。并向所有人提供历史彩票的信息，例如奖池、中奖票号、发行数量等。

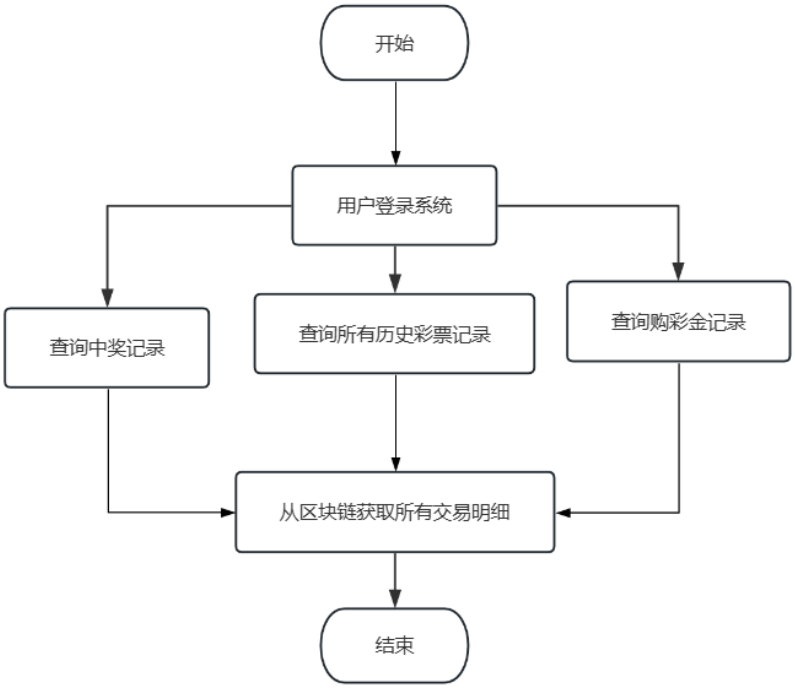


图 3.5 交易记录查询模块业务流程图

3.4 系统数据存储结构设计

因本系统数据采用链上存储，因此无传统数据库设计，但需要设计数据存储结构用于映射区块链网络事件来存储和检索数据，这些数据存储结构根据实际业务情况分布于智能合约和前端代码中。

3.4.1 E-R 模型设计

系统包括以下主要实体：彩票(Lottery)、用户(User)、彩票购买记录(Ticket)、奖项结构(PrizeStructure)、中奖记录(Winner)、奖金池(PrizePool)、交易记录(Transaction)，各实体及其属性分别如图 3.6 E-R 图所示。

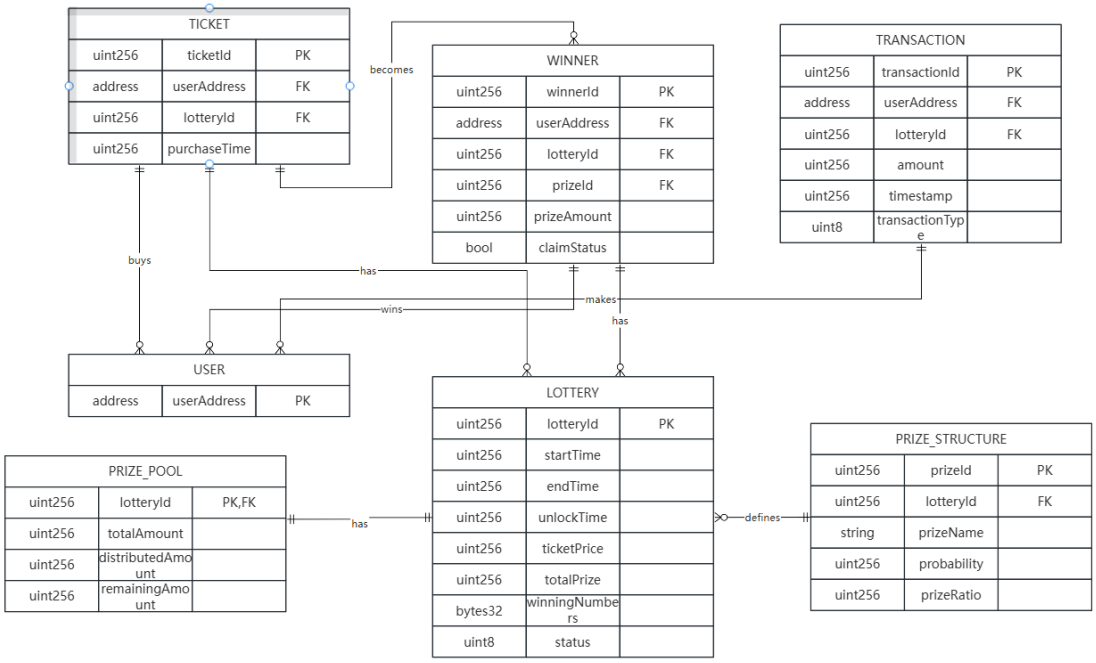


图 3.6 E-R 图

以下是对各实体和关系的讲解：

（1） Lottery：

彩票是整个系统的中心实体，表示已创建的彩票场次。每个彩票场次具有唯一标识符（lotteryId），包含开始时间、结束时间和解锁时间（即开奖时间）三个关键时间点，形成彩票生命周期的时间轴。彩票的状态（status）反映其当前所处阶段，如开放购买、已关闭、已开奖等。

（2） USER：

用户实体代表系统的参与者，以区块链钱包地址（userAddress）作为唯一标识。在区块链彩票系统中，用户无需传统的注册流程，仅通过钱包地址即可识别身份。用户是购买彩票、获得奖励和发起交易的主体。

（3） TICKET：

票据是用户购买彩票的凭证，每张票据关联特定用户和特定彩票场次。票据记录了购买时间（purchaseTime），这对于确认票据的有效性至关重要，特别是在彩票设有购买截止时间的情况下。票据是参与开奖和获得奖励的基本单位。

（4） PRIZE_STRUCTURE：

奖项结构定义了彩票的各个奖项，包括奖项名称、中奖人数和奖金比例。不同等级的奖项可能有不同的中奖人数和奖金分配，形成完整的奖励梯度。这种结构允许系统灵活配置各类彩票游戏的奖励机制。

（5） WINNER：

中奖记录实体记录了谁在哪场彩票中赢得了什么奖项，以及奖金金额和领取

状态。claimStatus 字段标记用户是否已领取奖金，有助于管理奖金发放。

（6） PRIZE_POOL:

奖金池实体管理彩票的资金流向，包括总金额、已分配金额和剩余金额。奖金池与单个彩票场次一一对应，记录该场彩票的资金状态。确保了系统中资金流转的透明性。

（7）TRANSACTION：

交易记录实体跟踪系统中的所有资金操作，包括购票支付、奖金领取等。每笔交易都记录金额、时间戳和交易类型，提供完整的资金审计链。对于系统的金融透明度和用户信任至关重要。

（8）实体间关系细析：

彩票系统的核心业务流程围绕着彩票场次与用户参与展开。系统设计中，每场彩票活动允许多位用户购买，形成典型的一对多关系。当用户参与某场彩票时，系统会生成独立的票据记录，这些票据既是用户参与的证明，也是后续开奖的依据。值得注意的是，一位用户可多次购买同一场彩票，每次操作都会产生新的票据编号，增加中奖机会。彩票活动的设计十分灵活，管理员可针对不同场次定制多样化的奖项结构。比如，周末特别场可能设置更高的头奖比例，而日常彩票则可能提供更多的小奖项，使奖金分配更加均匀。这种差异化配置满足了不同用户群体的期望，也丰富了系统的玩法。开奖环节是整个流程的关键点。系统会根据预设规则确定哪些票据满足中奖条件，并为这些幸运票据创建中奖记录。这一过程体现了票据与中奖记录间的条件性映射关系—只有被系统判定为中奖的票据才会关联相应的奖金信息。随后，系统自动标记用户的奖金领取状态，方便后续追踪。资金管理方面，每场彩票都对应唯一的奖金池，清晰记录总额、已分配和剩余金额。这种一对一设计确保了资金流向的透明性，避免了跨期彩票间的资金混淆。整个过程中，用户的购票支付和奖金领取都会生成交易记录，构成完整的财务轨迹，既方便用户查询个人历史，也为系统审计提供了可靠依据。

3.4.2 数据表信息

表 3. 2 表列表

表名	说明	对应合约结构
Lotteries	彩票信息表	mapping(uint256 => LotteryInfo)
PrizeStructures	奖项结构表	mapping(uint256 => PrizeStructure[])
PrizeWinners	中奖记录表	mapping(uint256 => PrizeWinner[])
UserTickets	用户彩票表	mapping(uint256 => mapping(address => uint256[]))
UserPrizes	用户奖金表	mapping(uint256 => mapping(address => uint256))

续表 3.2 表列表

表名	说明	对应合约结构
PrizeClaimStatus	奖金领取状态表	mapping(uint256 => mapping(address => bool))
UserTransactions	用户交易表	mapping(address => Transaction[])
UserWinningHistory	用户中奖历史表	mapping(address => uint256[])
TotalTickets	彩票总数表	mapping(uint256 => uint256)
LotteryBuyers	彩票购买者表	mapping(uint256 => address[])
HasParticipated	用户参与状态表	mapping(uint256 => mapping(address => bool))
LotteryIdByRequestId	随机数请求映射表	mapping(uint256 => uint256)

由于数据存储在链上，因此表 3.2 展示的是对应合约结构，各表详细说明如下：

表 3.3 彩票信息表

序号	中文名称	属性名	数据类型	描述
1	彩票 ID	lotteryId	uint256	彩票的唯一标识符
2	开始时间	startTime	uint256	彩票销售开始时间戳
3	结束时间	endTime	uint256	彩票销售结束时间戳
4	开奖时间	unlockTime	uint256	彩票开奖时间戳
5	票价	ticketPrice	uint256	单张彩票的价格（wei）
6	奖池总额	totalPrize	uint256	彩票的奖金池总额（wei）
7	中奖号码	winningNumbers	bytes32	开奖后的中奖号码
8	状态	status	LotteryState	彩票当前状态（枚举）

彩票信息表是整个系统的核心数据结构，记录了每场彩票活动的全生命周期信息。表中通过 lotteryId 字段唯一标识每场彩票，三个时间戳字段（startTime、endTime 和 unlockTime）精确控制彩票的销售周期和开奖时间点。其中 startTime 和 endTime 构成了用户可购买彩票的有效时间窗口，而 unlockTime 则设定了系统执行开奖操作的确切时刻，这种设计确保了开奖过程的透明性和不可操控性。ticketPrice 字段定义了参与门槛，以 wei 为单位记录彩票价格。随着用户购买行为的累积，totalPrize 字段动态更新，反映当前奖池资金总量。开奖后，系统将生

成的中奖号码存入 winningNumbers 字段，并采用 bytes32 类型以支持多种开奖模式的灵活实现。整个彩票流程中，status 字段追踪彩票当前所处阶段，如未开始、销售中、已结束待开奖、已开奖等状态，引导系统执行对应阶段的业务逻辑。

表 3.4 奖项结构表

序号	中文名称	属性名	数据类型	描述
1	奖项 ID	prizeId	uint8	奖项的唯一标识符
2	彩票 ID	lotteryId	uint256	关联的彩票 ID
3	奖项名称	name	string	奖项的名称
4	中奖人数	numbers	uint256	中奖人数
5	奖金比例	prizeRatio	uint256	奖金占奖池的比例（百分比）

奖项结构表定义了彩票中各个奖项的详细配置，支持系统实现多级奖励机制。每条记录由 prizeId 和 lotteryId 共同索引，前者区分同一彩票中的不同奖级，后者关联到特定彩票场次。奖项结构灵活配置了不同级别奖项的基本属性：通过 name 字段提供直观的奖项名称（如"一等奖"、"二等奖"），number 字段设定了该奖项的中奖人数，而 prizeRatio 则规定了该奖项在总奖池中的分配比例。这种设计使得不同彩票场次可以配置完全不同的奖项结构，例如周末特别场可能设置更高额的头奖，而日常彩票则可能提供更多的小额奖项。系统在开奖时会根据这些预设规则，结合随机生成的中奖号码，确定各个奖项的具体获奖者，并计算相应的奖金金额。

表 3.5 中奖记录表

序号	中文名称	属性名	数据类型	描述
1	中奖 ID	winnerId	uint256	中奖记录的唯一标识符
2	用户地址	winner	address	中奖者的区块链地址
3	彩票 ID	lotteryId	uint256	关联的彩票 ID
4	奖项 ID	prizeRank	uint8	中奖的奖项 ID
5	奖金金额	amount	uint256	中奖金额（wei）
6	中奖时间	winTime	uint256	中奖时间戳

中奖记录表保存了系统中所有中奖信息，是奖金分配的权威记录。每条中奖记录通过 winnerId 字段唯一标识，同时记录了中奖者地址（winner 字段）、关联

的彩票 ID（lotteryId 字段）及获得的奖项级别（prizeRank 字段）。amount 字段精确记录了中奖者应获得的具体奖金金额，该数值根据奖项结构中的比例规则和当期奖池总额计算得出。另外，系统还记录了中奖时间（winTime 字段），为后续的数据分析和审计提供时间维度的支持。当系统执行开奖操作时，会根据预设规则和随机生成的中奖号码，筛选出符合条件的参与者，并在该表中为每位中奖者创建记录。这些记录既是系统自动执行奖金分配的依据，也是用户查询个人中奖历史的数据来源。

表 3.6 用户交易表

序号	中文名称	属性名	数据类型	描述
1	用户地址	user	address	用户的区块链地址
2	彩票 ID	lotteryId	uint256	关联的彩票 ID
3	交易金额	amount	uint256	交易金额（wei）
4	交易时间	timestamp	uint256	交易时间戳
5	交易类型	txType	TransactionType	交易类型（枚举）

用户交易表跟踪记录了系统中所有资金流动，构建了完整的财务审计链。表中每笔交易都关联到特定用户地址（user 字段）和特定彩票场次（lotteryId 字段），记录交易发生的具体金额（amount 字段）和时间点（timestamp 字段）。交易类型（txType 字段）区分了不同业务场景下的资金操作，如购买彩票、领取奖金等。这种设计使系统能够清晰追踪每位用户的所有财务活动，无论是支出（购买彩票）还是收入（领取奖金）。交易记录不仅方便用户查询个人历史，也为系统管理员提供了全面的资金流向视图。在区块链环境中，这些交易记录与链上实际转账交易一一对应，确保了数据的真实性和不可篡改性，大大增强了系统的公信力。

其余表结构较为简单且易于理解，这里就不再详细描述。

3.4.3 存储过程信息

由于区块链智能合约的特性，传统数据库中的存储过程在本系统中以合约函数的形式实现。系统存储过程说明如下表 3.7 所示。

表 3.7 存储过程表

序号	名称	参数	功能说明
1	createLottery	uint256 startTime uint256 _endTime uint256 _unlockTime uint256_ticketPrice PrizeStructure[] calldata _prizes	创建新彩票，设置基本信息并初始化奖项结构，同时递增 currentLotteryId
2	buyTicket	uint256 lotteryId	记录用户购买彩票信息，更新用户票据映射、总票数和彩票总奖金，同时记录参与用户
3	drawLottery	uint256 lotteryId	将彩票状态更改为计算中，记录随机数请求信息，建立随机数请求 ID 与彩票 ID 的映射关系
4	receiveRandomness	uint256 requestId bytes32 randomness	接收随机数并存储为中奖号码，更新彩票状态，收取费用并触发奖金分配
5	distributePrizes	uint256 lotteryId	内部函数，生成中奖票据并分配奖金，更新中奖记录和奖金池统计信息
6	assignPrize	uint256 _lotteryId address winner uint256 amount	内部函数，将奖金分配给中奖者，记录用户奖金信息和中奖历史
7	recordTransaction	address _user uint256 _lotteryId uint256 _amount, TransactionType _type	内部函数，记录用户交易信息到 userTransactions 映射
8	claimPrize	uint256 lotteryId	处理用户领奖请求，更新领奖状态和奖金池统计，记录领奖交易
9	updatePrizePoolStats	uint256 _lotteryId	内部函数，更新奖金池统计信息，包括总奖金、已支付奖金和剩余奖金

续表 3.7 存储过程表

序号	名称	参数	功能说明
10	buyBatchTickets	uint256 lotteryId uint256 _quantity	批量购买多张彩票，更新用户票据映射、总票数和彩票总奖金
11	setPrizeStructure	uint256lotteryId PrizeStructure[] calldata _prizes	内部函数，设置彩票的奖项结构，包括奖项等级、比例和获奖人数
12	setFeeRate	uint256 feeRate	更新系统手续费率，仅合约拥有者可调用
13	enableRefund	uint256 lotteryId	将彩票状态更改为退款状态，允许用户申请退款
14	claimRefund	uint256 lotteryId	处理用户退款请求，更新退款状态并返还购票金额
15	startLottery	uint256 lotteryId	将彩票状态设置为开放状态，允许用户购买彩票

3.5 系统关键类设计

区块链彩票系统的功能实现包括彩票发行、开奖、奖金管理和交易记录查询，主要功能集中在开奖功能。图 3.7 给出了系统实现的主要类图，图 3.8 是各类之间流程的说明。

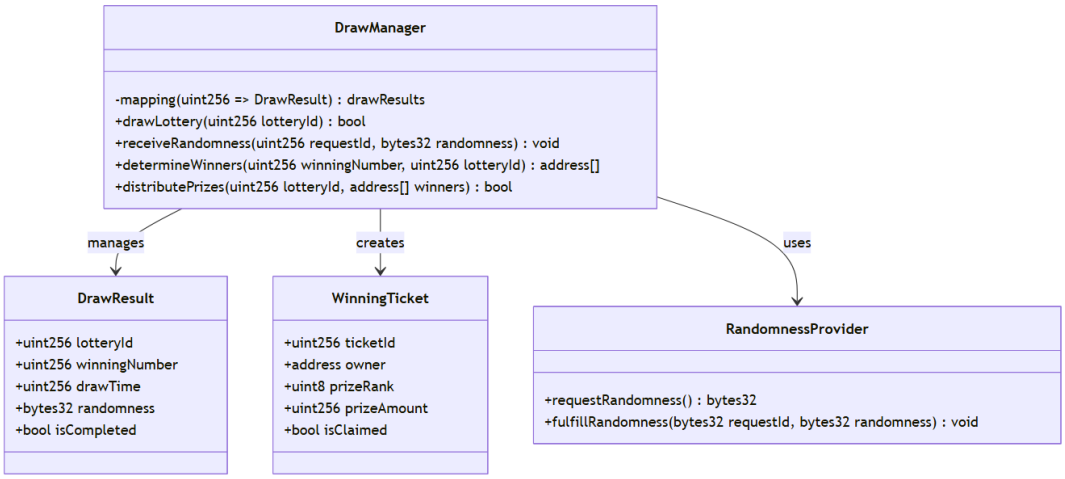


图 3.7 类图

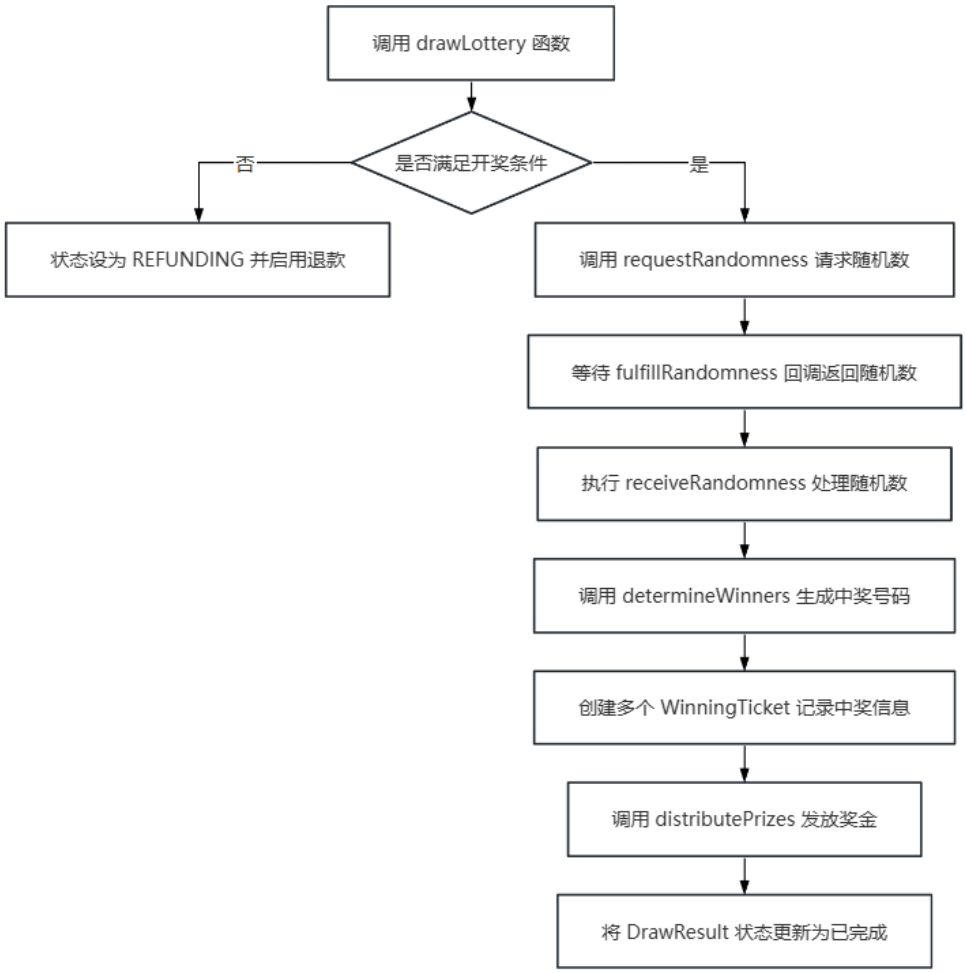


图 3.8 类之间的流程图

类图中展示了彩票系统开奖模块的核心结构，核心部分由四个协同工作的类组成。流程图中展示了各类图中的方法执行逻辑。DrawManager 类是整个开奖流程的控制中心，负责协调从随机数请求到奖金分配的完整流程，通过 drawLottery 方法触发开奖流程后，当彩票销售结束且时间锁到期后，该方法会验证参与人数是否满足奖项设置要求，满足要求后请求随机数。每次开奖过程中，DrawManager 都会创建并维护一个 DrawResult 实例，记录关键开奖数据。这个结构包含彩票 ID、生成的中奖号码、开奖时间戳、原始随机数和开奖完成状态。开奖核心依赖 RandomnessProvider 提供的可验证随机性。这个类封装了链上随机数生成的复杂性，提供 requestRandomness 方法发起随机数请求，并通过 fulfillRandomness 回调将生成的随机数返回给 DrawManager。

当确定中奖号码后，DrawManager 创建多个 WinningTicket 实例记录中奖详情。每个实例关联特定票号、所有者地址、奖项等级、奖金金额和领取状态。

图 3.7 中主要展示了系统实现的关键实体类的类图，实际应用中，还涉及到其他管理类和控制类的应用，但不再详细列举。

4 基于链上随机数和时间锁的彩票系统系统的实现

本系统包括用户操作子系统和智能合约。智能合约为整个系统的核心内容，它负责整个系统的核心功能的实现，即彩票发行、开奖、奖金管理和交易记录查询。在用户操作子系统中，钱包连接、易操作的彩票系统平台为该子系统的重点实现内容。以下是对各重点实现内容的实现的说明。

4.1 彩票发行功能

彩票发行功能实现了彩票创建与购买，其中彩票创建定义彩票单价、奖项结构、开始时间、结束时间和开奖时间等信息，管理员只需利用钱包登录管理员账户地址，在管理面面板下的创建彩票页面填写信息后发起交易，等待智能合约交易成功后就完成了彩票的创建。用户只需利用钱包登录就能购买这期彩票

（1）彩票发行功能的程序结构

管理员发起彩票创建交易在 AdminPanel.jsx 页面进行，管理员发起交易后会调用智能合约的 createLottery()方法，等待交易确认后智能合约会将信息存储在区块 Lotteries 映射中通过 getLotteryInfo()方法将信息返回给前端页面进行展示。用户通过当前彩票页面查看可购买的彩票，用户发起购买交易后会调用智能合约的 buyBatchTickets()方法完成购买

（2）彩票发行功能的界面设计

管理员在主页点击连接钱包，然后点击导航栏的管理面板进入管理员界面，选择创建彩票，填写相关信息，点击创建彩票并设置奖项结构，跳转出钱包页面后点击确认发送交易，如图 4.1 彩票创建界面所示。

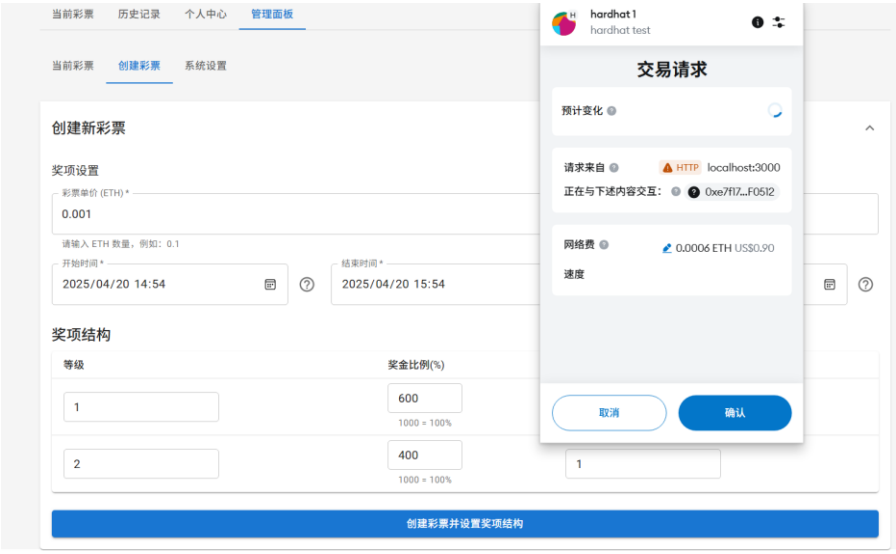


图 4.1 彩票创建界面

用户连接钱包后在当前彩票页面查看彩票信息，输入想要购买的彩票数量，点击购买后跳转钱包页面，确认划扣金额无误后发起交易，如图 4.2 彩票购买界面所示。

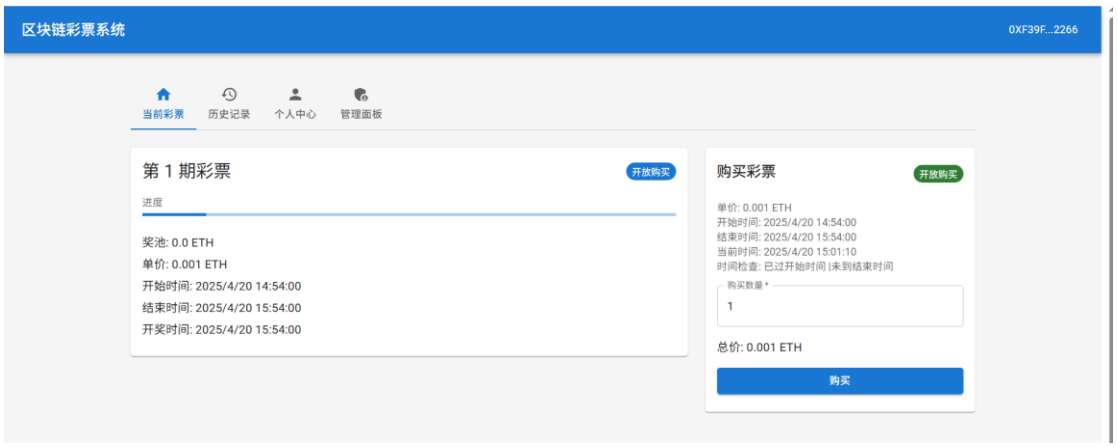


图 4. 2 彩票购买界面

（3）彩票发行功能的业务逻辑

以下关键代码是创建彩票的核心逻辑，通过 createLottery()方法创建彩票并将信息存储在智能合约 Lotteries 映射中。

```
 function createLottery(uint256 _startTime,uint256 _endTime,uint256，
_unlockTime,uint256 _ticketPrice,PrizeStructure[] calldata _prizes)
external onlyOwner {
    require(_startTime > block.timestamp, "Start time must be in future");
    require(_endTime > _startTime, "End time must be after start time");
    require(_ticketPrice > 0, "Ticket price must be greater than 0");
        currentLotteryId++;
        lotteries[currentLotteryId] = LotteryInfo({
            lotteryId: currentLotteryId,
            startTime: _startTime,
            endTime: _endTime,
            unlockTime: _unlockTime,
            ticketPrice: _ticketPrice,
            totalPrize: 0,
            winningNumbers: bytes32(0),
            status: LotteryState.OPEN   // 直接设置为 OPEN 状态
        });
        setPrizeStructure(currentLotteryId, _prizes);
        emit LotteryCreated(currentLotteryId, _startTime, _endTime,
_unlockTime, _ticketPrice);
    }
```

代码流程图如图 4.3 所示，该函数首先确保只有合约拥有者可以调用，并对传入的参数进行严格校验：要求彩票的开始时间必须晚于当前区块时间，结束时间要在开始时间之后，同时票价也必须大于零。通过这些校验后，系统会为新的彩票分配一个唯一的 ID，并将所有相关信息（如起止时间、解锁时间、票价等）存储到彩票信息结构体中。随后，函数会根据传入的奖品结构设置当前彩票的奖品分布，最后通过事件机制对外发布新的彩票创建信息，整个流程环环相扣，确保了彩票创建的规范性和安全性。

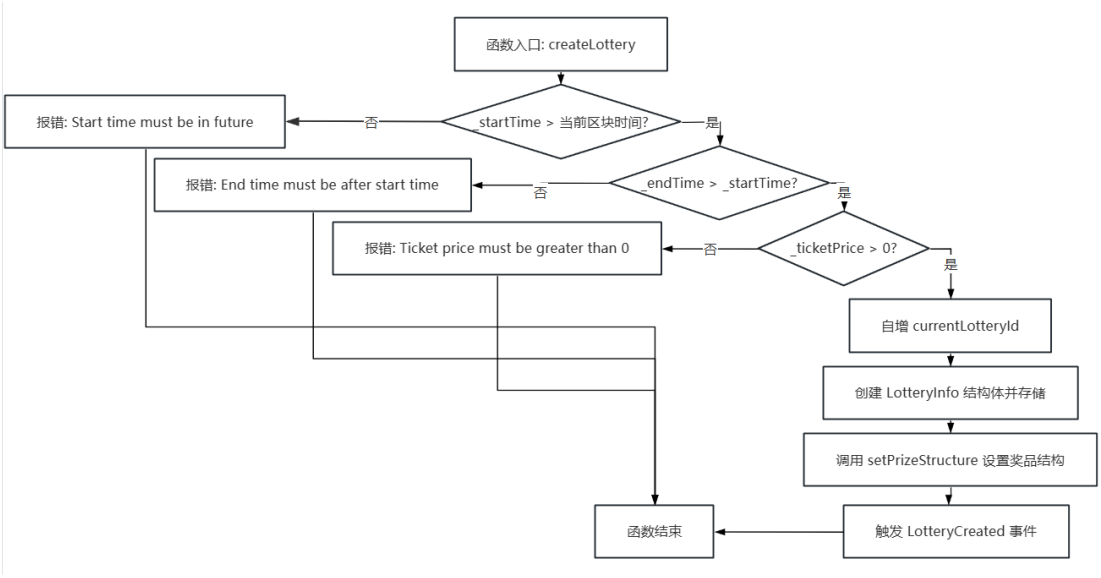


图 4.3 创建彩票代码流程图

以下关键代码是彩票购买的核心逻辑，用户通过 buyBatchTicket()方法购买彩票后，通过已售出的彩票的数量生成票号，并通过 userTickets 映射记录购买信息，通过内部函数_recordTransaction()方法记录交易。

```
// 添加批量购买函数
    function buyBatchTickets(uint256 _lotteryId, uint256 _quantity) external payable
nonReentrant {
        require(_quantity > 0, "Quantity must be greater than 0");
        LotteryInfo storage lottery = lotteries[_lotteryId];
        require(lottery.status == LotteryState.OPEN, "Lottery not open");
        require(block.timestamp >= lottery.startTime, "Lottery not started");
        require(block.timestamp <= lottery.endTime, "Lottery ended");
        require(msg.value == lottery.ticketPrice * _quantity, "Incorrect total price");
        uint256[] memory ticketIds = new uint256[](_quantity);
        // 批量生成票号
        for(uint256 i = 0; i < _quantity; i++) {
            uint256 ticketId = totalTickets[_lotteryId] + i;
            userTickets[_lotteryId][msg.sender].push(ticketId);
            ticketIds[i] = ticketId;
        }
        // 更新总票数
        totalTickets[_lotteryId] += _quantity;
        // 记录购买者(如果是首次购买)
        if (!hasParticipated[_lotteryId][msg.sender]) {
            lotteryBuyers[_lotteryId].push(msg.sender);
            hasParticipated[_lotteryId][msg.sender] = true;
        }
        lottery.totalPrize += msg.value;
        // 记录购买交易
        _recordTransaction(
            msg.sender,
            _lotteryId,
            msg.value,
            TransactionType.PURCHASE
        );
        emit BatchTicketsPurchased(_lotteryId, msg.sender, _quantity, ticketIds);
    }
```

代码流程图如图 4.4 所示，当用户调用 buyBatchTickets 函数时，首先会验证购买数量是否大于 0，随后判断对应的彩票是否处于开放状态，并检查当前时间是否在允许的购票时间段内。同时，还会校验用户转账金额是否与购票总价相符。若所有条件均满足，合约将开始批量生成票号，并将票号存入用户名下，更新该彩票的总票数。如果该用户是第一次参与当前彩票，还会记录其身份。接着，系统将累加奖池总金额，记录此次购买行为，并触发一个批量购票成功的事件。整个流程确保交易的原子性和一致性，适配多人并发购买的场景。

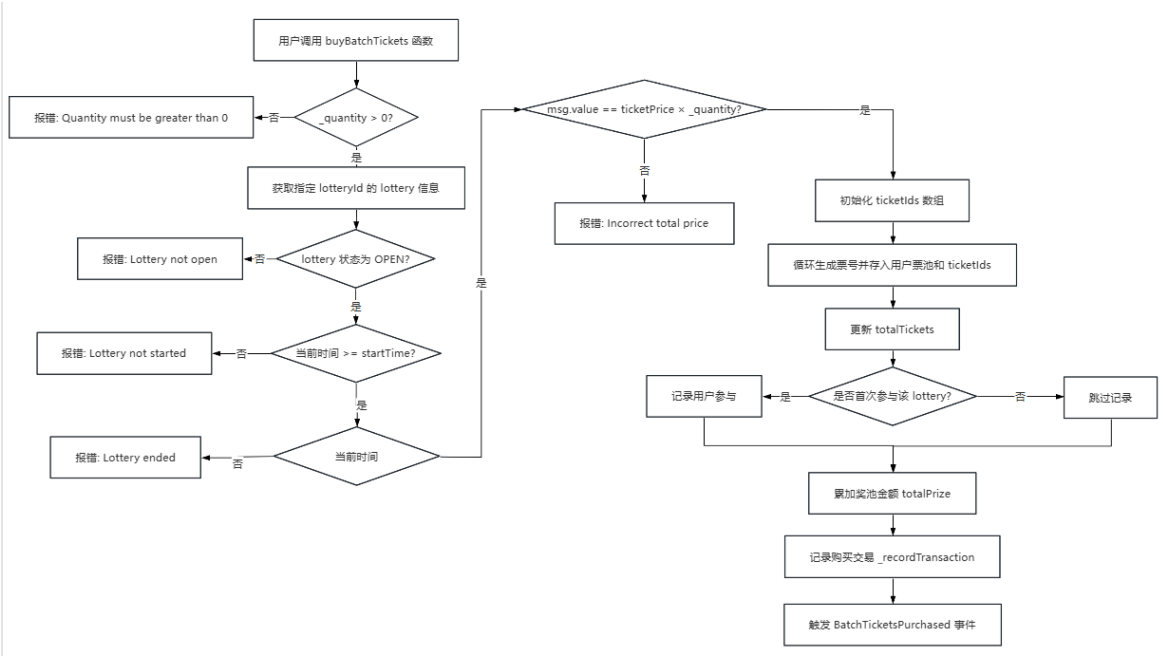


图 4.4 购买彩票代码流程图

4.2 开奖功能

开奖功能实现了彩票的自动开奖与结果公布，主要包括开奖时间验证、随机数生成和中奖号码确定三个环节。管理员在彩票销售结束且达到开奖时间后，通过管理面板触发开奖流程，系统会通过链上随机数生成机制产生公平、不可预测的开奖结果，并自动确定中奖用户及其奖金金额。

（1）开奖功能的程序结构

开奖流程在 AdminPanel.jsx 的开奖管理页面进行，管理员点击开奖按钮后会调用智能合约的 drawLottery()方法，触发随机数请求，从 Drand 分布式网络节点中获取随机数后智能合约会通过 receiveRandomness()方法接收随机数提供者返回的随机数，利用随机数调用 distributePrizes()方法进行奖金分配，最终将开奖结果存储在 lotteryWinners 映射中。前端通过 getLotteryWinners()方法获取开奖结果并在结果页面展示。

（2）开奖功能的界面设计

管理员在管理面板选择开奖管理，可以看到所有可开奖的彩票列表，包括彩票 ID、销售状态和开奖时间等信息。管理员点击开奖按钮后，系统会验证当前时间是否已达到开奖时间，如果条件满足，将弹出确认窗口。确认后，页面会显示开奖进度，随后展示开奖结果，如图 4.5 开奖界面所示。

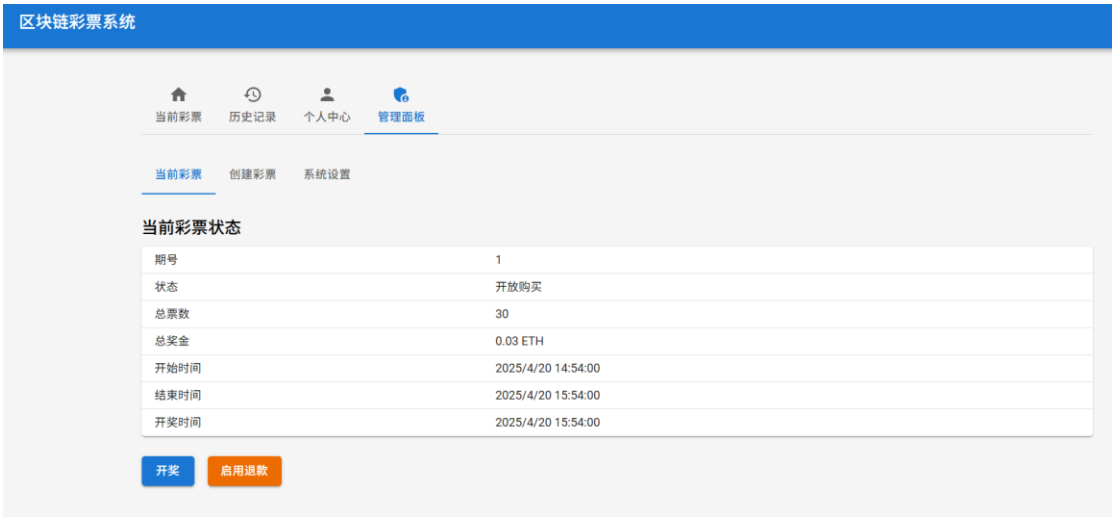


图 4.5 开奖界面

开奖结果页面会显示中奖号码、各奖项获奖情况和奖金分配明细，用户可以在此查看自己是否中奖，如图 4.6 开奖结果界面所示。

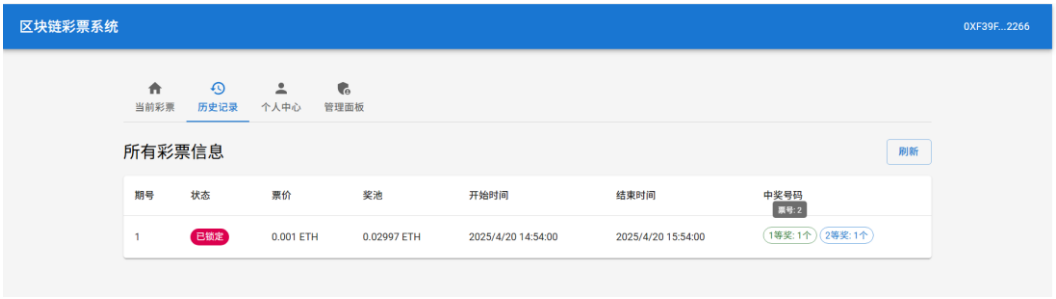


图 4.6 开奖结果界面

（3）开奖功能的业务逻辑

以下关键代码是开奖的核心逻辑，通过 drawLottery()方法触发开奖流程，然后从 Drand 分布式网络节点获取可靠的随机数，传递给随机数提供合约，随机数提供合约通过回调 receiveRandomness()方法让彩票系统接收随机数并进行奖金分配。

```
    // 请求随机数进行开奖
    function drawLottery(uint256 _lotteryId) external onlyOwner {
        LotteryInfo storage lottery = lotteries[_lotteryId];
        // require(lottery.status == LotteryState.OPEN, "Lottery not open");
        // require(block.timestamp > lottery.endTime, "Lottery not ended");
        // require(block.timestamp >= lottery.unlockTime, "Lottery locked");
        // 如果没有票，直接返回
        if (totalTickets[_lotteryId] == 0) {
            revert("No tickets sold");
        }
        // 检查是否有足够的参与者
        uint256 totalWinners = getTotalWinners(_lotteryId);
        if (totalTickets[_lotteryId] < totalWinners) {
            lottery.status = LotteryState.REFUNDING;
            emit RefundEnabled(_lotteryId);
            return;
        }
        lottery.status = LotteryState.CALCULATING;
        currentRequestId =
IRandomnessProvider(RANDOMNESS_PROVIDER).requestRandomness();
        lotteryIdByRequestId[currentRequestId] = _lotteryId;
        emit RandomnessRequested(_lotteryId, currentRequestId, lottery.unlockTime);
    }
```

代码流程图如图 4.7 所示，该函数用于管理员触发彩票开奖流程。函数开始时会获取对应彩票的信息，如果当前未售出任何票，则直接中止并抛出错误。接着系统计算应有的中奖人数，并检查实际售票是否满足开奖条件。如果售票数量不足以选出所有中奖者，则转入退款流程，修改彩票状态为 REFUNDING 并触发退款事件；否则进入开奖准备阶段，将状态更新为 CALCULATING，并向随机数服务请求一个随机数，同时记录此次请求与彩票编号的关联，最终触发随机数请求事件，为后续开奖做准备。

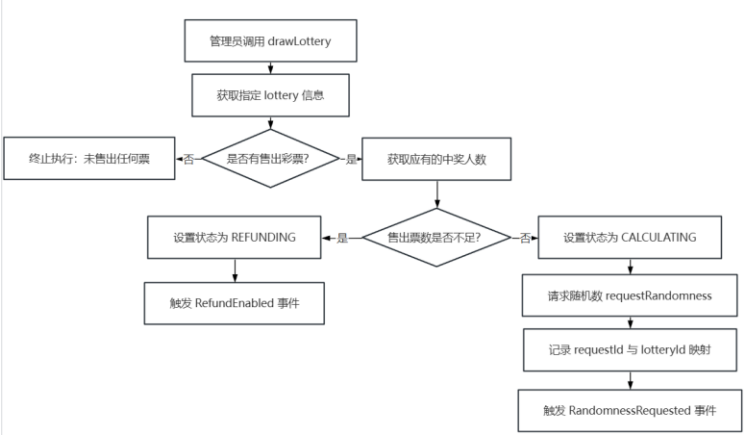


图 4.7 开奖代码流程图

获取随机数后通过 Fisher-Yates 洗牌算法得出中奖票号，然后根据中奖票号的等级进行奖金分配，随机数生成核心逻辑代码如下。

```
    // 生成中奖号码
    function generateWinningTickets(uint256 _lotteryId) internal view returns
(uint256[] memory) {
        uint256[] memory winningTickets = new
uint256[](getTotalWinners(_lotteryId));
        uint256 ticketCount = totalTickets[_lotteryId];
        require(ticketCount > 0, "No tickets sold");
        // 获取总需要的中奖者数量
        uint256 totalWinners = getTotalWinners(_lotteryId);
        require(ticketCount >= totalWinners, "Not enough tickets sold");
        bytes32 randomSeed = lotteries[_lotteryId].winningNumbers;
        // 使用 Fisher-Yates 洗牌算法
        uint256[] memory allTickets = new uint256[](ticketCount);
        for(uint256 i = 0; i < ticketCount; i++) {
            allTickets[i] = i;
        }
        // 只需要洗牌前 totalWinners 个位置
        for(uint256 i = 0; i < totalWinners; i++) {
            randomSeed = keccak256(abi.encodePacked(randomSeed, i));
            uint256 j = i + uint256(randomSeed) % (ticketCount - i);
            // 交换位置 i 和 j 的票号
            uint256 temp = allTickets[i];
            allTickets[i] = allTickets[j];
            allTickets[j] = temp;
            winningTickets[i] = allTickets[i];
        }
        return winningTickets;
    }
```

代码流程图如图 4.8 所示，该函数用于从所有购票者中公平地选出中奖票号。流程开始时首先验证是否有票售出，并确保实际票数不少于应中奖人数。随后函数构建从 0 到票数减 1 的所有票号数组 allTickets，并借助 Fisher-Yates 洗牌算法从中随机挑选出指定数量的中奖票号。在每次循环中，通过哈希计算出一个随机索引 j，然后将第 i 张票和第 j 张票交换，最终前 totalWinners 个票号即为中奖结果。整个过程保证了中奖结果的随机性与公平性，不依赖外部排序或重复抽取操作。

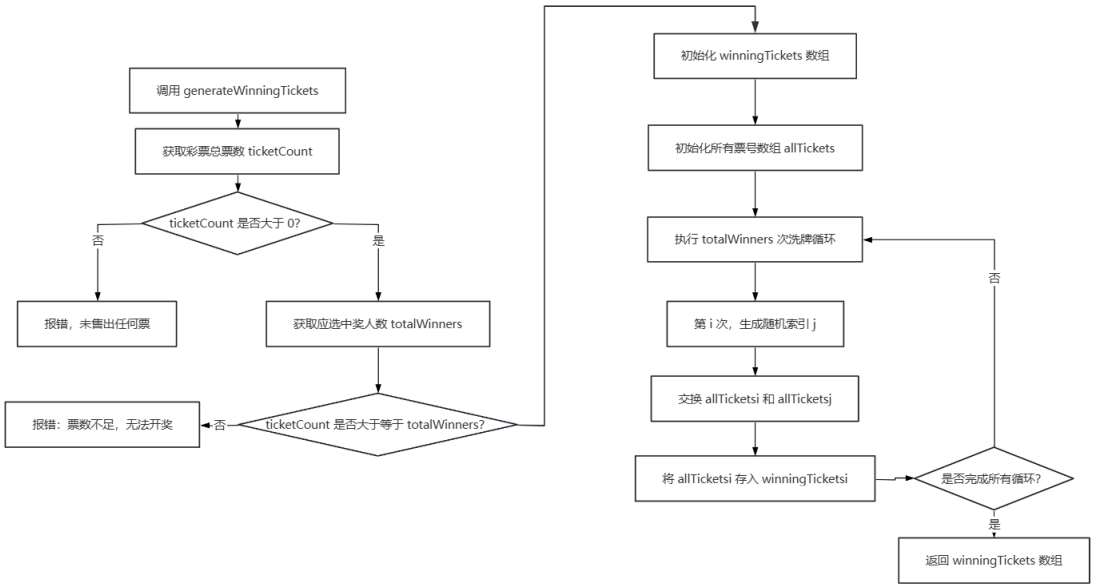


图 4.8 生成中奖号码代码流程图

4.3 奖金管理功能

奖金管理功能实现了中奖奖金的计算、分配和领取，确保奖金分配透明公正。系统根据预设的奖项结构自动计算各奖项金额，中奖用户可以随时查看并领取自己的奖金，管理员可以监控奖金池状态和分配情况。

（1）奖金管理功能的程序结构

奖金管理主要通过智能合约的 distributePrizes()方法实现奖金分配，用户通过 UserCenter.jsx 页面的奖金领取功能领取奖金。后端使用 lotteryWinners 映射存储中奖信息，使用 userPrizes 映射记录用户奖金，prizeClaimStatus 映射跟踪领取状态。前端通过 getUserWinningHistory()和 getUserPrize()方法获取中奖信息，并在用户中心页面展示，用户点击领取按钮后调用 claimPrize()方法领取奖金。

（2）奖金管理功能的界面设计

用户在中奖后，可以在用户中心查看自己的中奖记录，包括彩票 ID、奖项等级、奖金金额和领取状态。对于未领取的奖金，系统会显示领取按钮，用户点击后确认交易即可完成领取，如图 4.9 奖金领取界面所示。

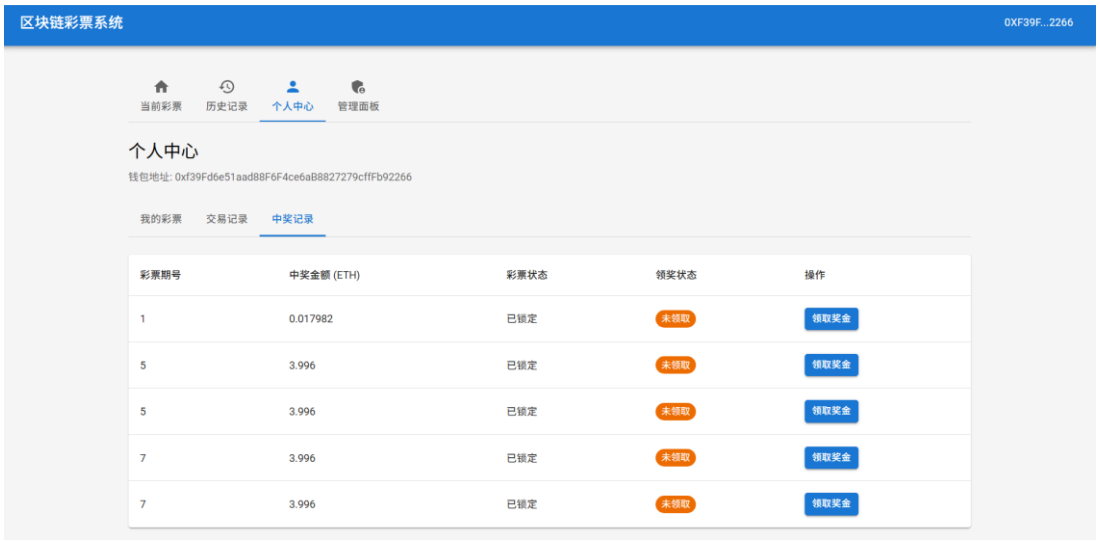


图 4.9 奖金领取界面

（3）奖金管理功能的业务逻辑

以下是奖金分配的核心代码。

```
function distributeWinningPrizes(uint256 _lotteryId, uint256[] memory
winningTickets) internal { LotteryInfo storage lottery = lotteries[_lotteryId];
        PrizeStructure[] storage prizeStructure = lotteryPrizes[_lotteryId];
        delete lotteryWinners[_lotteryId];
        uint256 currentWinner = 0;
        for(uint256 i = 0; i < prizeStructure.length; i++) {
    uint256 prize Amount = (lottery.totalPrize*prizeStructure[i].percentage) / 1000;
            uint256 perWinnerAmount = prizeAmount / prizeStructure[i].winners;
            for(uint256 j = 0; j < prizeStructure[i].winners; j++) {
                address winner = findTicketOwner(_lotteryId,
winningTickets[currentWinner]);
                require(winner != address(0), "Winner not found");
                _assignPrize(_lotteryId, winner, perWinnerAmount);
                lotteryWinners[_lotteryId].push(PrizeWinner({
                    winner: winner,
                    rank: prizeStructure[i].rank,
                    amount: perWinnerAmount,
                    ticketId: winningTickets[currentWinner]
                }));
                currentWinner++;
```

代码流程图见图 4.10，该函数用于根据预设的奖项结构将总奖池按比例分发给对应的中奖者。首先清空原有的中奖者列表并初始化计数器，然后对每个奖项等级进行处理。对于每一等级，先按比例计算应分配的总奖金，再除以该等级中奖人数得到每位中奖者应得金额。随后通过中奖票号查找对应地址，校验其有效

性后分配奖金，并将信息记录到中奖者列表中。

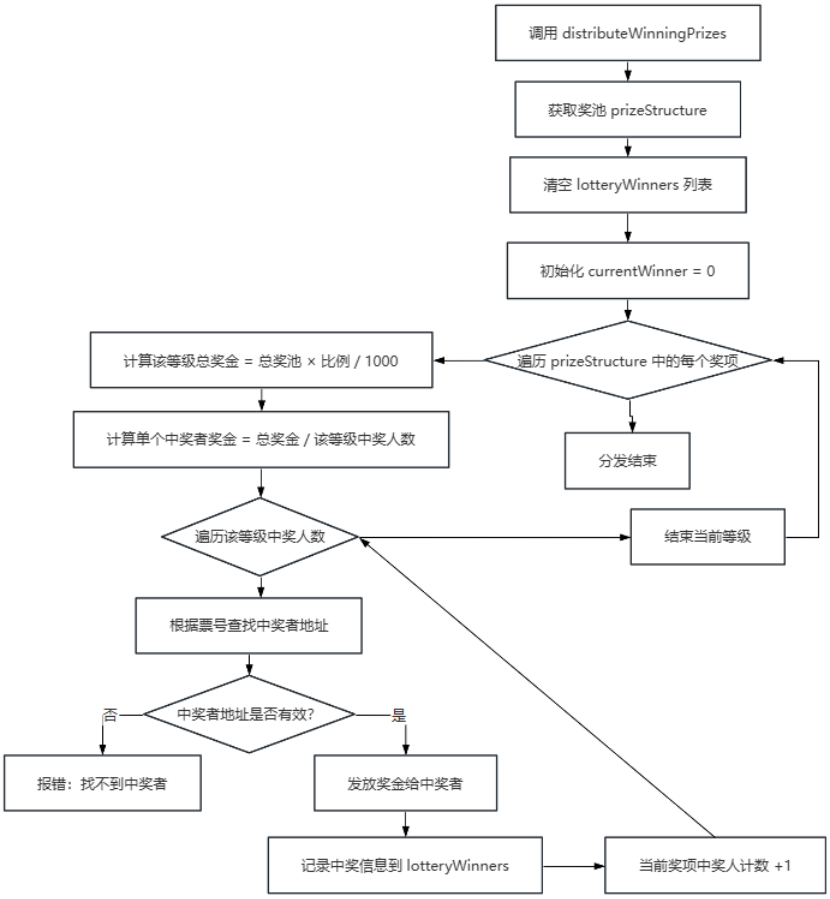


图 4.10 奖金分配代码流程图

4.4 交易记录查询功能

交易记录查询功能实现了用户交易历史的透明记录和查询，包括购票记录、中奖记录和领奖记录等。用户可以通过该功能查看自己的全部交易历史，便于跟踪参与情况和资金流向。

（1）交易记录查询功能的程序结构

交易记录查询主要在用户中心的交易历史页面实现，前端通过 UserTransactions.jsx 组件展示用户交易记录。后端使用 userTransactions 映射存储用户的所有交易信息，每次购票或领奖操作都会通过_recordTransaction()方法记录交易详情。前端通过 getUserTransactions()方法获取交易记录并在页面上展示，支持按交易类型和时间筛选。

（2）交易记录查询功能的界面设计

用户在用户中心点击交易历史标签，可以查看自己的所有交易记录，包括交易类型、彩票 ID、交易金额和交易时间等信息。界面支持按交易类型筛选和时间排序，以及详情查看功能，如图 4.11 交易记录查询界面所示。

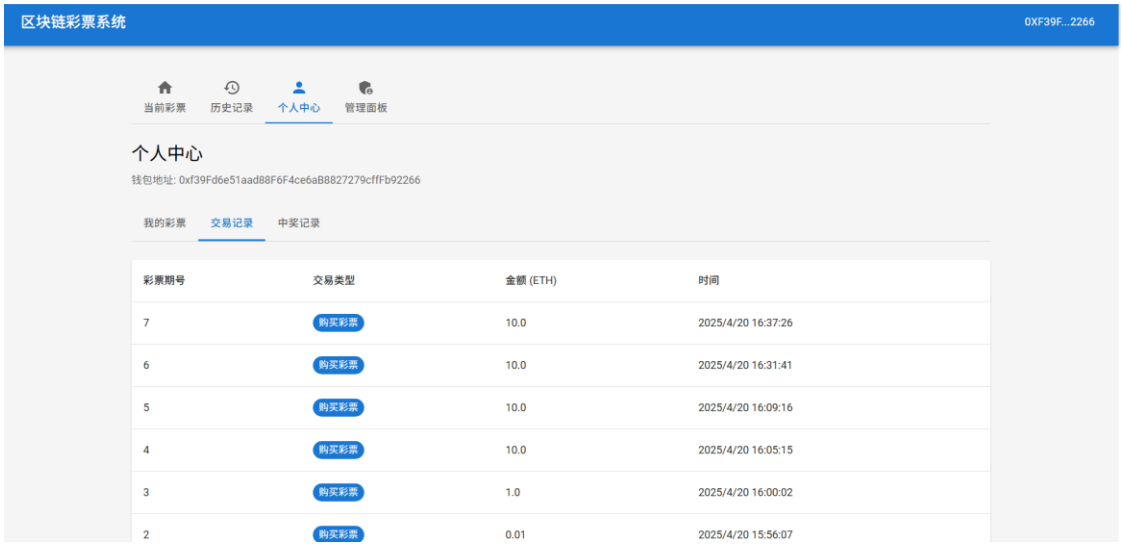


图 4. 11 交易记录查询界面

（3）交易记录查询功能的业务逻辑

以下是交易记录的核心代码，系统通过_recordTransaction()方法记录每笔交易，getUserTransactions()方法负责获取用户相关交易。

```
    // 记录交易
        function _recordTransaction(address _user,uint256 _lotteryId,uint256
_amount,
    TransactionType _type) internal {
            userTransactions[_user].push(Transaction({
                user: _user,
                lotteryId: _lotteryId,
                amount: _amount,
                timestamp: block.timestamp,
                txType: _type
            }));
        }
         function getUserTransactions(address _user) external view returns
(Transaction[] memory) {
```

代码流程图如图 4.12 所示，该记录交易的内部函数 _recordTransaction 会在用户参与活动（如购票、兑奖等）时被调用。它接收用户地址、彩票编号、金额及交易类型作为输入参数，并将这些信息封装成一个 Transaction 结构体。随后，这个结构体会被添加到对应用户的交易记录数组中，方便后续查询。同时，外部可通过 getUserTransactions 查看用户所有历史交易记录，实现完整的交易数据

追踪与溯源。

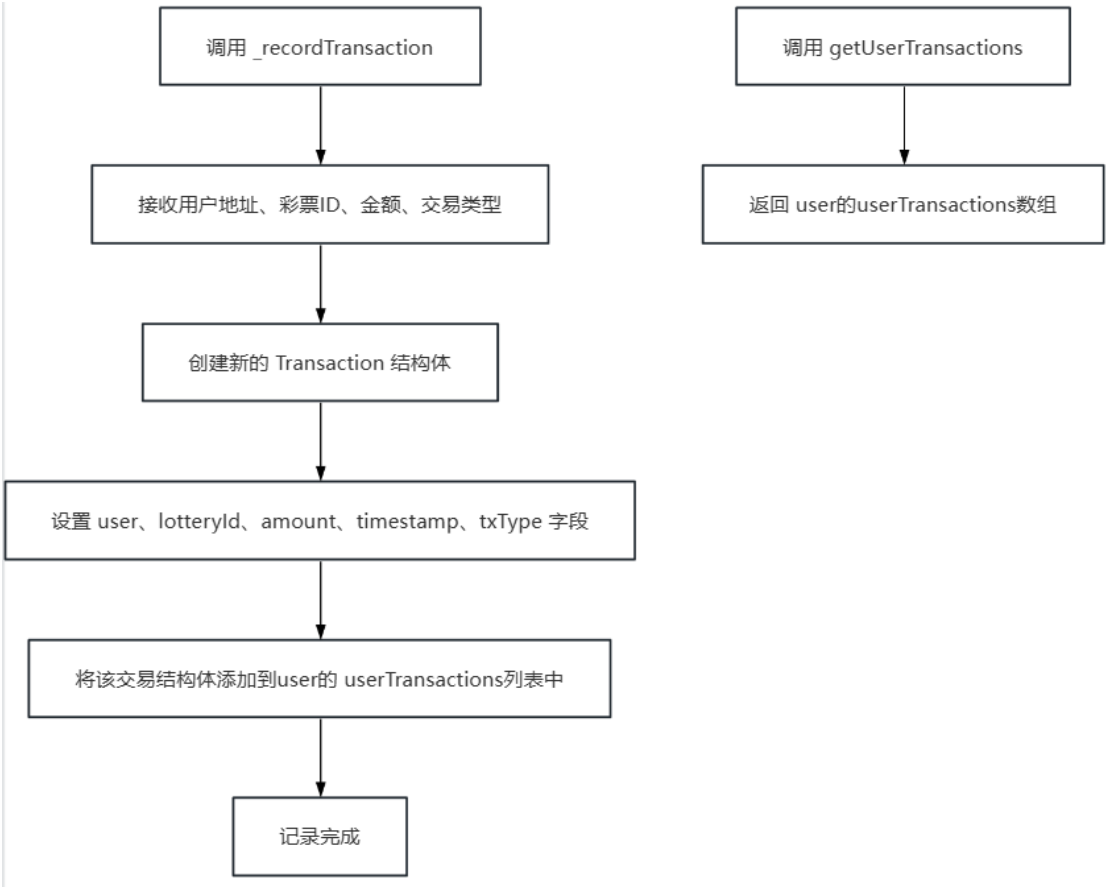


图 4.12 交易管理代码流程图

4.5 系统实现小结

本章概述了基于链上随机数和时间锁的彩票系统从发行到开奖、奖金管理再到交易记录查询各个关键功能的实现思路与流程。系统由用户操作子系统和智能合约两部分组成，核心逻辑均在智能合约中完成。

首先，彩票发行功能包括彩票的创建与购买两大环节。管理员通过管理面板填写价格、奖项结构、销售与开奖时间等信息，调用合约的 createLottery() 方法将新彩票信息保存在链上；用户则在前端界面查看当前可售彩票，输入购买数量后发起交易，由合约的 buyBatchTickets() 方法完成购票，系统会自动生成并记录票号，同时更新售出总数和奖池金额。

其次，开奖功能由管理员在销售结束且到达开奖时间后触发，前端调用合约的 drawLottery() 方法，合约向 Drand 分布式网络请求随机数，收到随机数后通过 Fisher–Yates 洗牌算法公平选出中奖票号。若售票不足以开出全部奖项，则系统会进入退款流程并触发相应事件；正常情况下，合约将中奖结果存入链上映射，并触发事件供前端展示。

随后，奖金管理功能利用合约的 distributePrizes() 方法根据预设奖项结构计算并分发奖金。合约先根据各奖级比例计算总奖金，再均分给对应级别的中奖者，最后更新用户可领取状态。用户可在前端 “用户中心” 页面查看中奖详情并调用 claimPrize() 方法领取奖金，合约会更新领取状态并转账。

最后，交易记录查询功能确保系统操作透明。每次购票或领奖时，合约内部函数 _recordTransaction() 会将交易类型、彩票编号、金额、时间等信息封装存入用户交易映射。用户在 “交易历史” 页面可通过 getUserTransactions() 方法查看、筛选并排序自己的所有交易记录，方便追溯资金流向和参与记录。

整体来看，本系统强调链上存储与事件触发机制，保证了发行、开奖、奖金发放和交易查询的透明性与安全性，同时为用户提供了直观易用的前端界面。

5 基于链上随机数和时间锁的彩票系统系统测试

5.1 测试环境搭建

测试环境包括服务器环境、客户端环境、网络环境。服务器环境主要用于前端页面的部署，客户端环境包括浏览器环境和操作系统的测试，网络环境指区块链网络环境，分两种情况测试，本地区块链网络测试和区块链测试网网络测试。

测试机与服务器的软硬件配置如表 5.1 所示

表 5.1 软硬件配置表

设备名称	硬件配置	软件配置
服务器	处理器：AMD 5800H 或以上，内存：16GB 或以上，存储：1.5TB SSD 或以上，网络：千兆以太网连接	操作系统：Ubuntu 20.04 LTS，区块链客户端：Geth / OpenEthereum，开发环境：Hardhat，通信协议：HTTP/HTTPS、WebSocket
开发工作站	处理器：双核及以上 ，内存：8GB 或以上，存储：256GB SSD 或以上，显示器：1920×1080 分辨率	操作系统：Windows 10/11 或 MacOS，IDE：VS Code、Remix，前端框架：React.js，区块链工具：Web3.js、Truffle
测试客户端	处理器：双核及以上 内存：4GB 及以上，硬盘容量：50GB 及以上，网络：稳定的互联网连接	操作系统：Windows / MacOS / Linux，浏览器：Edge 最新版，钱包插件：MetaMask，网络工具：Postman

5.2 功能测试

5.2.1 彩票发行测试

主要对彩票发行功能进行测试，测试是否能成功创建彩票，系统能否正确显示新建彩票信息。

首先，对管理员调用 createLottery 创建新彩票的流程进行了验证：在输入合法的开始时间、结束时间、开奖时间和票价，并设置完奖项结构后，合约能成功返回并触发 LotteryCreated 事件，随后通过 getLotteryInfo 能够准确读回新建的彩

票信息。紧接着，通过将票价设为零，验证合约会因不满足“票价必须大于 0”的要求而执行失败；而当使用非管理员账户调用同样接口时，也会因权限不足而被拒绝。

在购票环节，单张购票（buyTicket）和批量购票（buyBatchTickets）两种常规操作均能正确触发相应事件（TicketPurchased/BatchTicketsPurchased）、更新用户票据映射，并将奖池总额按实际购票金额同步增加。对于未开始或已结束的彩票，系统亦会分别抛出 “Lottery not started” 及 “Lottery ended” 错误，确保购票时序受到严格控制。

测试用例及测试结果如下所示：

表 5.2 彩票发行测试表

用例编号	测试标题	前置条件	输入及操作说明	预期结果	测试结果
T001	管理员创建彩票	1. 部署合约 2. 使用管理员账户登录	1. 调用 createLottery 方法 2. 输入开始时间、结束时间、开奖时间、票价 3. 设置奖项结构	1. 返回成功 2.触发 LotteryCreated 事件 3. 通过 getLotteryInfo 获取到新创建的彩票信息	通过
T002	创建彩票票价为零	1. 部署合约 2. 使用管理员账户登录	1. 调用 createLottery 方法 2. 设置票价为 0	合约执行失败，抛出 "Ticket price must be greater than 0" 错误	通过
T003	非管理员创建彩票	1. 部署合约 2. 使用非管理员账户登录	1. 调用 createLottery 方法 2. 输入合法参数	合约执行失败，抛出权限错误	通过
T004	用户购买单张彩票	1. 存在活跃彩票 2. 用户余额充足	1. 调用 buyTicket 方法 2. 发送与票价相等的 ETH	1. 返回成功 2. 触发 TicketPurchased 事件 3. 用户票据映射更新 4. 奖池总额增加	通过

续表 5.2 彩票发行测试表

用例编号	测试标题	前置条件	输入及操作说明	预期结果	测试结果
T005	用户批量购买彩票	1. 存在活跃彩票 2. 用户余额充足	1. 调用 buyBatchTickets 方法 2. 设置购买数量为 5 3. 发送等于 5 倍票价的 ETH	1. 返回成功 2. 触发 BatchTicketsPurchased 事件 3. 用户获得 5 张票据 4. 奖池总额增加相应金额	通过
T006	彩票未开始时购买	1. 存在未开始的彩票 2. 当前时间 < 开始时间	调用 buyTicket 方法尝试购买	合约执行失败，抛出 "Lottery not started" 错误	通过
T007	彩票已结束时购买	1. 存在已结束的彩票 2. 当前时间 > 结束时间	调用 buyTicket 方法尝试购买	合约执行失败，抛出 "Lottery ended" 错误	通过

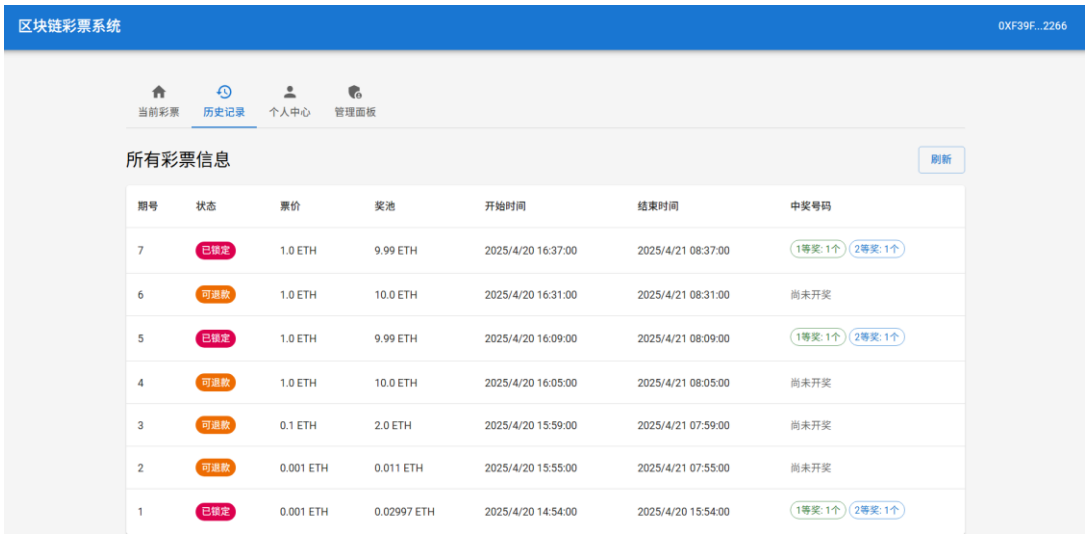


图 5.1 彩票发行测试记录

5.2.2 开奖功能测试

这项测试主要是为了测试开奖流程是否能正常完整的完成，随机数和中奖者能否正常的产生。

开奖流程分为请求随机数与处理回调两步：在所有票据销售结束且参与人数充足的前提下，管理员调用 drawLottery 后，彩票状态自动切换到 CALCULATING，并触发 RandomnessRequested 事件；随机数提供者回调 receiveRandomness 后，状态进一步进入 LOCKED，中奖号码被确定且 LotteryDrawn 事件发出，同时奖金按既定比例分配。

异常场景也都被覆盖——当无任何票据售出时，直接失败并报“ No tickets sold”；若总参与人数少于设定的获奖名额，则彩票状态转为 REFUNDING，并触发 RefundEnabled 事件；同样，非管理员尝试发起开奖也会因权限校验而失败。最后，通过查询 lotteryWinners 映射及奖金池统计，验证所有中奖者信息和分配金额都与预设比例完全一致。

测试用例和测试结果如下所示：

表 5.3 开奖功能测试表

用例编号	测试标题	前置条件	输入及操作说明	预期结果	测试结果
T101	正常开奖流程	1. 彩票销售已结束 2. 有足够的参与者 3. 使用管理员账户	1. 调用 drawLottery 方法 2. 传入有效彩票 ID	1. 彩票状态变为 CALCULATING 2. 触发 RandomnessRequested 事件 3. 随机数提供者被调用	通过
T102	随机数回调处理	1. 彩票状态为 CALCULATING 2. 随机数回调触发	随机数提供者调用 receiveRandomness 方法	1. 彩票状态变为 LOCKED 2. 中奖号码被设置 3. 触发 LotteryDrawn 事件 4. 奖金分配完成	通过
T103	无票销售的开奖	1. 彩票销售已结束 2. 总票数为 0 3. 使用管理员账户	调用 drawLottery 方法	合约执行失败，抛出 "No tickets sold" 错误	通过

续表 5.3 开奖功能测试表

用例编号	测试标题	前置条件	输入及操作说明	预期结果	测试结果
T104	参与人数不足开奖	1. 彩票销售已结束 2. 参与人数少于总获奖名额	调用 drawLottery 方法	1. 彩票状态变为 REFUNDING 2. 触发 RefundEnabled 事件	通过
T105	非管理员触发开奖	1. 彩票销售已结束 2. 使用非管理员账户	调用 drawLottery 方法	合约执行失败，抛出权限错误	通过
T106	奖金分配验证	1. 开奖已完成 2. 奖金已分配	查询中奖信息和奖金池统计	1. lotteryWinners 映射包含所有中奖者信息 2. 分配的总奖金等于设定比例 3. 奖金池统计数据已更新	通过

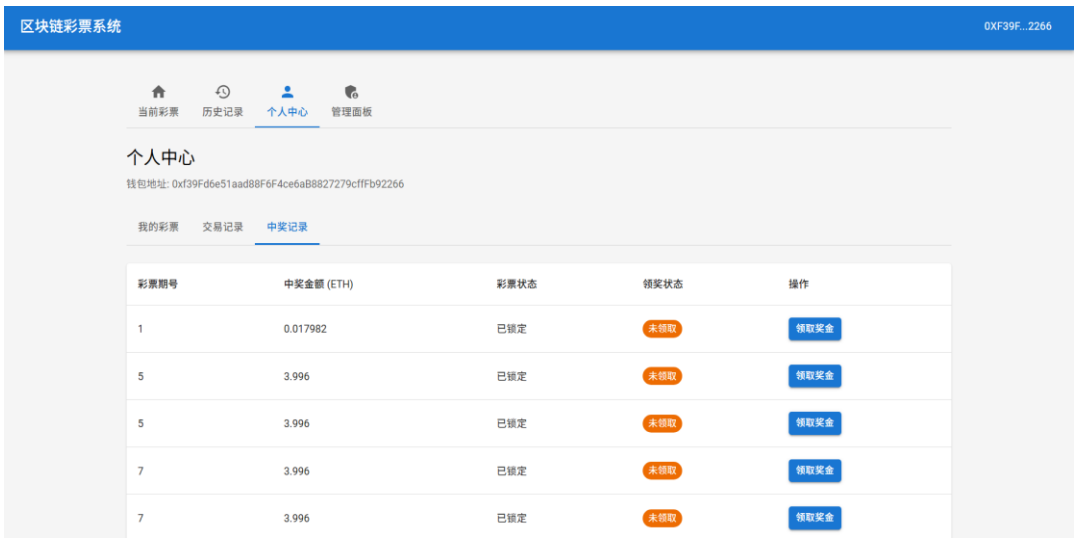


图 5.2 开奖功能测试记录

5.2.3 奖金管理功能测试

该测试是为了确保用户能正常查看每期彩票的信息，能正常领取奖金，如果有退款流程时能正常领取奖金。

在奖金领取环节，拥有未领取奖金的用户可调用 claimPrize 并获得转账，同时合约触发 PrizeClaimApproved 事件，用户领奖状态变为“已领取”，而奖金池

中的 totalPaidOut 与 currentBalance 也会据此更新。针对重复领取或非中奖用户再次调用同一方法，合约分别抛出 “Prize already claimed” 和 “No prize to claim” 的错误消息，防止资金被滥取。除此之外，调用 getUserPrize 接口能准确返回用户应得奖金数额。每次领取完成后，检索奖金池统计，发现 totalPaidOut 增加、currentBalance 减少且 lastUpdateTime 更新至当前时间，验证了统计信息的实时性与准确性。

测试用例和测试结果如下图所示：

表 5. 4 奖金管理功能测试表

用例编号	测试标题	前置条件	输入及操作说明	预期结果	测试结果
T201	用户领取奖金	1. 用户有未领取的奖金 2. 合约余额充足	1. 用户调用 claimPrize 方法 2. 传入对应彩票 ID	1. 用户收到奖金转账 2. 触发 PrizeClaimApproved 事件 3. 用户领奖状态变为“已领取” 4. 奖金池统计更新	通过
T202	重复领取奖金	1. 用户已领取奖金 2. 尝试再次领取	用户调用 claimPrize 方法	合约执行失败，抛出 "Prize already claimed" 错误	通过
T203	非中奖用户领奖	用户没有中奖记录	用户调用 claimPrize 方法	合约执行失败，抛出 "No prize to claim" 错误	通过
T204	查询用户奖金	用户有中奖记录	调用 getUserPrize 方法	返回正确的用户应得奖金金额	通过
T205	奖金池统计更新	已有用户领取奖金	查询奖金池统计信息	1. totalPaidOut 增加对应金额 2.currentBalance 减少对应金额 3. lastUpdateTime 更新为当前时间	通过

续表 5. 4 奖金管理功能测试表

用例编号	测试标题	前置条件	输入及操作说明	预期结果	测试结果
T206	奖金分配验证	1. 开奖已完成 2. 奖金已分配	查询中奖信息和奖金池统计	1. lotteryWinners 映射包含所有中奖者信息 2. 分配的总奖金等于设定比例 3. 奖金池统计数据已更新	通过

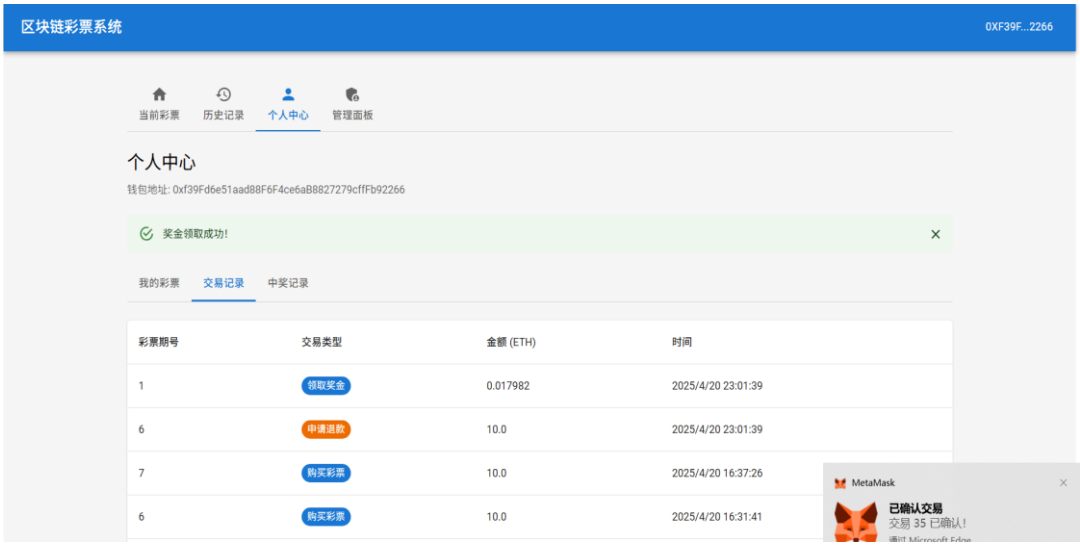


图 5. 3 奖金管理功能测试记录

5.2.4 历史记录查询功能测试

本次测试是为了查看用户能否正常查看与自己有关的交易记录，历史彩票信息能否正常显示。

最后一部分测试聚焦于各种只读接口的正确性——用户可通过 getUserTickets 和 getUserWinningHistory 分别查询所持票据与中奖历史，并获得对应的 ID 列表与奖金金额；通过 getLotteryInfo、getLotteryTicketCount、getLotteryBuyers 可分别获取单期彩票的详情、总售票数及所有参与者地址；getUserTransactions 接口则能列出用户的购票与领奖等交易记录；getLotteryWinners 返回每期的彩票中奖者名单、奖级与奖金金额；getLockStatus 可查当前彩票是否处于时间锁中及剩余时长。对不存在的彩票 ID 调用

getLotteryInfo，系统会返回一个空结构体，确保在无效输入下仍有稳定而友好的表现。

测试用例和测试结果如下图所示：

表 5.5 历史记录查询功能测试表

用例编号	测试标题	前置条件	输入及操作说明	预期结果	测试结果
T301	查询用户票据	用户已购买彩票	1. 调用 getUserTickets 方法 2. 传入用户地址和彩票 ID	返回用户拥有的所有票据 ID 数组	通过
T302	查询彩票信息	彩票已创建	调用 getLotteryInfo 方法	返回包含彩票所有信息的 LotteryInfo 结构	通过
T303	查询用户中奖历史	用户有中奖记录	调用 getUserWinningHistory 方法	1. 返回用户所有中奖的彩票 ID 数组 2. 返回对应的奖金金额数组	通过
T304	查询用户交易历史	用户有交易记录	调用 getUserTransactions 方法	返回用户所有交易记录的数组，包括购票和领奖记录	通过
T305	查询彩票中奖者	彩票已开奖	调用 getLotteryWinners 方法	返回彩票的所有中奖信息，包括中奖者、奖项等级和奖金金额	通过
T306	查询彩票票数	彩票有销售记录	调用 getLotteryTicketCount 方法	返回彩票的总销售票数	通过
T307	查询彩票参与者	彩票有参与用户	调用 getLotteryBuyers 方法	返回所有参与彩票的用户地址数组	通过
T308	查询彩票时间锁状态	彩票已创建	调用 getLockStatus 方法	返回彩票的锁定状态和剩余锁定时间	通过

续表 5.5 历史记录查询功能测试表

用例编号	测试标题	前置条件	输入及操作说明	预期结果	测试结果
T309	查询非存在彩票	使用不存在的彩票 ID	调用 getLotteryInfo 方法	返回默认值（空结构体）	通过

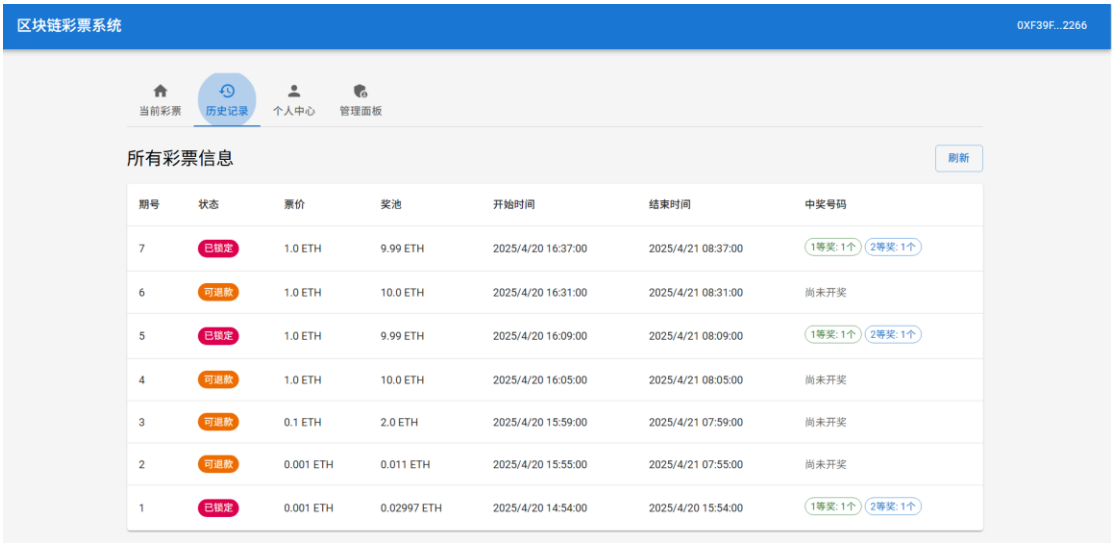


图 5.4 历史记录查询功能测试记录

5.3 性能测试

性能测试采用与以太坊主网类似的以太坊测试网，其中智能合约测试由 Hardhat Gas Reporter 完成。主要测试合约核心功能彩票发行、彩票开奖、领取奖金的 gas 消耗测试，批量购票函数相较于单次购票函数单张票据的 Gas 消耗降低了约 54%，大幅提高了用户批量操作的效率，结果如下：

表 5.6 gas 消耗测试

函数名称	最小 Gas 消耗	最大 Gas 消耗	平均 ETH 成本（约估）
createLottery	187,456	192,321	0.000937 ETH
buyTicket	102,573	105,980	0.000513 ETH
buyBatchTickets(5 张)	231,425	245,678	0.001157 ETH
drawLottery	156,734	162,452	0.000784 ETH
claimPrize	94,562	98,754	0.000473 ETH

模拟不同数量的并发用户同时购买彩票，记录系统响应情况如下表 5.7：

表 5.7 并发测试

并发用户数	平均响应时间（秒）	交易成功率	系统 CPU 使用率	内存占用（GB）
10	16.2	100%	15%	1.2
25	17.5	100%	28%	1.5
50	19.3	98%	46%	2.1
100	25.7	92%	75%	3.3
200	38.4	85%	93%	4.7

系统在 50 并发用户时表现良好，达到 100 用户时开始出现响应延迟，200 用户时成功率下降明显。

5.4 测试总结

整体来看，系统在功能正确性、异常处理、统计展示等方面表现均已满足设计预期；性能测试也验证了批量操作的高效性以及系统在中等并发下的稳健性。但在超高并发场景下出现的成功率下降与响应延迟，后期考虑优化合约逻辑来提升用户体验，或采用引入 Layer2 扩展方案（如 Arbitrum / zkSync）将购票与领奖等高频操作迁移至二层网络，降低 Gas 成本，提高 TPS，改善高并发处理能力。同时因为与 Drand 随机数生成分布式网络连接时可能会出现网络延迟导致随机数生成迟缓，后期考虑采用同时集成 Chainlink VRF + Drand 双重预言机，确保随机数生成的稳定性。

结 论

本文成功设计并实现了基于链上随机数和时间锁机制的彩票系统，解决了传统彩票系统中的信任问题。通过将区块链技术与彩票业务相结合，我们构建了一套去中心化、公开透明且安全可靠的彩票系统，实现了从彩票发行、购买到开奖、派奖的完整业务流程。

系统的核心价值在于其公平性与透明度。通过引入链上随机数生成机制，我打破了传统彩票随机性难以验证的困境，成功实现了可靠的随机数生成过程。尤其是在结合 Drand 分布式随机数网络后，随机数的可靠性与不可预测性得到了显著提升。同时，时间锁机制确保了开奖过程只能在预定时间点后进行，有效防止了人为干预的可能性。

从技术上看，本系统充分发挥了区块链技术的优势。首先，我们遵循"代码即法律"的理念，智能合约完全控制了彩票发行后的核心流程，包括销售记录、开奖和奖金分配等，无需任何中间机构参与。其次，采用 Fisher-Yates 洗牌算法生成中奖票号，保证了中奖过程的公平性与随机性。此外，系统的批量购票功能显著降低了 Gas 消耗（单张票据节省约 54%），大幅提升了用户体验。

通过一系列的功能测试和性能测试，系统展现出了良好的稳定性和可靠性。功能测试验证了系统各模块的正常运行，包括彩票发行、开奖、奖金管理和交易记录查询等。性能测试表明系统可同时支持 50 个并发用户时表现良好，满足了初期应用场景的需求。不过，当并发用户数超过 100 时，系统响应时间明显延长，成功率也有所下降，这一问题需要在未来版本中进一步优化。

从实际应用角度看，本系统展示了区块链技术在彩票和游戏领域的巨大潜力。传统彩票行业长期以来受到公信力不足、透明度低的困扰，而区块链技术恰恰能解决这些痛点。通过代码开源和交易公开，参与者能够全程监督彩票的发行、销售和开奖过程，彻底改变了"靠信任维系"的旧模式。

未来，系统仍有多方面的发展空间。首先是性能优化，需解决高并发下的响应延迟问题；其次是扩展玩法，可增加多种彩票类型以满足不同用户需求；最后，可探索与传统彩票系统的融合模式，促进区块链技术的落地应用。随着技术的不断迭代和完善，基于区块链的彩票系统将为彩票行业带来革命性的变革，重塑用户信任，创造更加公平透明的参与环境。

通过本项目的研究与实践，我不仅深入理解了区块链技术在实际应用中的潜力和挑战，也为区块链技术在游戏娱乐领域的应用贡献了一份实践案例，希望能为区块链技术的落地与普及提供有益参考。

参考文献

[1] De Angelis S, Lombardi F, Zanfino G, et al. Security and dependability analysis of blockchain systems in partially synchronous networks with Byzantine faults[J]. International Journal of Parallel, Emergent and Distributed Systems, 2023: 1-21. DOI: 10.1080/17445760.2023.2272777.

[2] Homoliak I, Venugopalan S, Reijsbergen D, et al. The security reference architecture for blockchains: Toward a standardized model for studying vulnerabilities, threats, and defenses[J]. IEEE Communications Surveys & Tutorials, 2020, 23(1): 341-390. DOI: 10.1109/COMST.2020.3031064. https://www.researchgate.net/publication/388890428.

[3] Soofiyan S, Karami A. Ethereum Smart Contracts: A Hierarchical Analysis of Vulnerability Challenges and Mitigation Strategies[J]. Cluster Computing, 2025. https://www.researchgate.net/publication/388890428.

[4] Haouari W, Hafid A S, Fokaefs M. Vulnerabilities of smart contracts and mitigation schemes: A Comprehensive Survey[J]. arXiv preprint arXiv:2403.19805, 2024. https://arxiv.org/pdf/2403.19805.

[5] Hire A, Lanjewar H, Haridas P, et al. Decentralized lottery using blockchain[C]//2023 3rd International Conference on Pervasive Computing and Social Networking (ICPCSN). IEEE, 2023: 1035-1041. DOI: 10.1109/ICSSIT57734.2023.10266035.

[6] Alleman P. Randomness and Games on Ethereum[EB/OL]. University of Bern , (2021-12). https://crypto.unibe.ch/archive/theses/2021.msc.peter.allemann.pdf.

[7] Vu N M. Tackle the Oracle Problem: Enhancing Trust and Functionality for Smart Contracts Through Blockchain Oracle[D]. University of Wollongong, 2024. https://ro.uow.edu.au/ndownloader/files/50575734/1.

[8] Chen E, Liang J, Huang R, et al. Building Random, Fair, and Verifiable Games on Blockchain. Raffle smart contract designs on Sui Network[J]. arXiv preprint arXiv:2310.12305, 2023. https://arxiv.org/pdf/2310.12305.

[9] Šimunić S, Bernaca D, Lenac K. Verifiable computing applications in blockchain[J]. IEEE access, 2021, 9: 156729-156745. DOI: 10.1109/ACCESS.2021.3134865.

[10] Alp E C. Towards General-Purpose Decentralized Computing with Permissionless Extensibility[D]. EPFL, 2024. https://infoscience.epfl.ch/record/307240/files/EPFL_TH8858.pdf.

[11] Li J, Zhang Z, Li M. BanFEL: A blockchain based smart contract for fair and efficient lottery scheme[C]//2019 IEEE conference on dependable and secure computing (DSC). IEEE, 2019: 1-8.

[12] Arafat S M. A Study of Blockchain Consensus Protocols[J]. Cryptology ePrint Archive, 2025. https://eprint.iacr.org/2025/637.pdf.

[13] Liao J. Towards Reliable and Efficient Blockchain Applications[D]. Wayne State University, 2024. https://search.proquest.com/openview/b35f7fcee88c3161bea85de5511b47dc/1.

[14] Dong J, Song C, Sun Y, et al. Daon: A decentralized autonomous oracle network to provide secure data for smart contracts[J]. IEEE Transactions on Information Forensics and Security, 2023, 18: 5920-5935. DOI: 10.1109/TDSC.2023.3297134.

[15] Lys L. Security and reliability of cross-chain exchanges[D]. Sorbonne Université, 2022. https://theses.hal.science/tel-03847642/file/LYS_Leonard_2022.pdf.

致 谢

在论文编写过程中，感谢梁培利老师自论文开题报告到如今论文编写结束的辛勤指导， 老师严谨治学的态度和不厌其烦的指导，让我受益匪浅。老师的建议让我在系统功能设计和数据验证等方面有了更深入的思考。尤其是在链上随机数生成的安全性分析中,提出的参考思路为我的实现方案提供了重要参考。

此外，还要感谢本课题答辩过程中评审老师提出的修改建议，这些建议帮助我更准确地完善了论文的细节，使内容更加严谨、逻辑更加清晰。

感谢我的王晨室友，在项目开发和测试阶段协助我进行系统联调和代码复查，他们认真负责的态度令我感动。每一次模拟测试中，他们的配合都让我更加坚定地走完毕业设计的每一步。

最后，感谢学院为我们提供良好的学术环境和实践平台，让我有机会在一个真实项目的语境下完成这次研究。这次毕业设计不仅提升了我独立思考和项目实践能力，也让我更深入地理解了区块链技术在现实中的应用价值。

本次设计虽已完成，但在我心中，它更像是一段旅程的总结，也是我学术探索的一个崭新起点。愿今后能继续秉承老师们传授的严谨精神，不忘初心，砥砺前行。

作者简介：成都信息工程大学人工智能学院区块链工程专业 21 级学生

姓　　名：翁忠旭　　性别：　男

出生年月：1999 年 2 月 15 日　　民族：　汉

E-mail:1834435282@qq.com

声 明

本论文的工作是2024年10月至2025年5月在成都信息工程大学人工智能学院（区块链产业学院）完成的。文中除了特别加以标注地方外，不包含他人已经发表或撰写过的研究成果，也不包含为获得成都信息工程大学或其他教学机构的学位或证书而使用过的材料。

此课题仅做区块链技术研究，所设计的产品不用于实际的虚拟货币交易。

关于学位论文使用权和研究成果知识产权的说明：

本人完全了解成都信息工程大学有关保管使用学位论文的规定，其中包括：

（1）学校有权保管并向有关部门递交学位论文的原件与复印件。

（2）学校可以采用影印、缩印或其他复制方式保存学位论文。

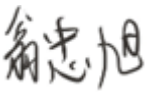
（3）学校可以学术交流为目的复制、赠送和交换学位论文。

（4）学校可允许学位论文被查阅或借阅。

（5）学校可以公布学位论文的全部或部分内容（保密学位论文在解密后遵守此规定）。

除非另有科研合同和其他法律文书的制约，本论文的科研成果属于成都信息工程大学。

特此声明！

作者签名： 

2025年5月5日