



Decentralized Book Rental Platform

TEAM CRYPTOSPHERE



Team Members

- ▶ Ayush Kumar – 230001012
- ▶ Hemsai – 230001079
- ▶ Shubham Prajapati – 230005047
- ▶ Raja Reddy – 230001054
- ▶ Vikas – 230002023
- ▶ Sudheendra – 230001076



Introduction

- ▶ Blockchain-based platform enabling users to list, rent, and return books in a trustless environment
- ▶ Leverages Ethereum smart contracts for rental agreements and payments
- ▶ Uses IPFS for decentralized storage of book cover images
- ▶ Creates a transparent and secure marketplace for book rentals



Key Features

- ▶ Decentralized book listing and rental system
- ▶ Secure deposit and refund mechanism
- ▶ Automated rental fee calculations
- ▶ IPFS integration for cover images
- ▶ Modern, responsive user interface
- ▶ No intermediaries or central authority



Technical Stack

- ▶ Smart Contract: Solidity 0.8.19
- ▶ Blockchain Development: Truffle & Hardhat
- ▶ Frontend: React.js with Web3.js and Ethers.js
- ▶ File Storage: IPFS via Pinata
- ▶ Testing: Mocha, Hardhat and Chai
- ▶ UI Framework: Bootstrap with custom CSS

System Components

- ▶ Smart Contract Layer: BookRental.sol and OpenZeppelin modules
- ▶ Frontend Layer: React components with Web3 integration
- ▶ Storage Layer: On-chain (rental data) and Off-chain (IPFS book images)

Key Smart Contract Functions

- ▶ listItem: Book listing with validation
- ▶ rentItem: Rental processing with deposit handling
- ▶ returnItem: Return processing with refund calculation
- ▶ getBookDetails: Retrieval of book information
- ▶ getUserBooks: Listing books owned by a user
- ▶ getUserRentals: Listing books rented by a user



Security Considerations

- ▶ Reentrancy Protection using OpenZeppelin Guards
- ▶ Access Control for owner-only functions
- ▶ Secure Payment Handling with Check-Effects-Interactions pattern
- ▶ Privacy-focused data storage approach
- ▶ Transaction validation to prevent fraud



Optimizations

- ▶ Gas Optimization: Efficient data types, minimal on-chain storage, optimized struct packing
- ▶ Performance Enhancements: Efficient React rendering, optimized Web3 calls, cached IPFS content



Testing Strategy

- ▶ Comprehensive Test Suite:
 - ▶ - Unit tests for contract functions
 - ▶ - Integration tests for user flows
 - ▶ - Security-focused tests
 - ▶ - Performance benchmarks

Testcases:

BookRental Contract

- ✓ Should calculate refunds accurately for different rental durations
- ✓ Should require additional payment when rental fees exceed deposit (49ms)
- ✓ Should reject returns from non-renters

Basic Book Rental Functionality

- ✓ Should allow listing a book
- ✓ Should reject listing with empty title
- ✓ Should reject listing with empty author
- ✓ Should allow renting a book
- ✓ Should reject renting an unavailable book
- ✓ Should reject insufficient payment
- ✓ Should reject renting by the owner
- ✓ Should allow returning a book

11 passing (1s)

Reentrancy Protection

BookRental Reentrancy Protection

Reentrancy Protection Tests

- ✓ Should set up books for testing
- ✓ Should allow normal rental and return operations (74ms)
- ✓ Should allow attacker contract to rent a book

Attack correctly failed with error: VM Exception while processing transaction: reverted with reason string 'Refund transfer failed'

- ✓ Should prevent reentrancy attack on returnItem (50ms)

4 passing (3s)

Key Frontend Components

- ▶ Marketplace.js: Book browsing, rental initiation, search/filtering
- ▶ MyRentals.js: Rental management, return handling, transaction status
- ▶ ListBook.js: Book listing, IPFS upload, validation



User Interface Features

- ▶ Responsive design for all devices
- ▶ Real-time updates for blockchain events
- ▶ Transaction notifications and status tracking
- ▶ Interactive elements for improved UX
- ▶ Seamless wallet integration



User Journey

- ▶ Connect wallet to the platform
- ▶ Browse available books
- ▶ Rent a book with deposit and rental fee
- ▶ Return the book when finished
- ▶ Receive refund automatically based on rental period



Future Enhancements

- ▶ Technical: Layer 2 scaling, enhanced metadata, improved gas optimization
- ▶ User: Rating system, social features, advanced search, multiple token support



Scalability Considerations

- ▶ Sharding compatibility for Ethereum upgrades
- ▶ State channel integration for reduced gas costs
- ▶ IPFS optimization for content delivery
- ▶ Cross-chain interoperability



Conclusion

- ▶ Enhanced security through smart contracts
- ▶ Reduced costs by eliminating intermediaries
- ▶ Improved transparency in rental process
- ▶ Decentralized storage for digital content



Thank You

▶ Team CryptoSphere