ICP Collateral Lending Protocol 🚀

Internet Computer Protocol

Built with Rust

DeFi Lending

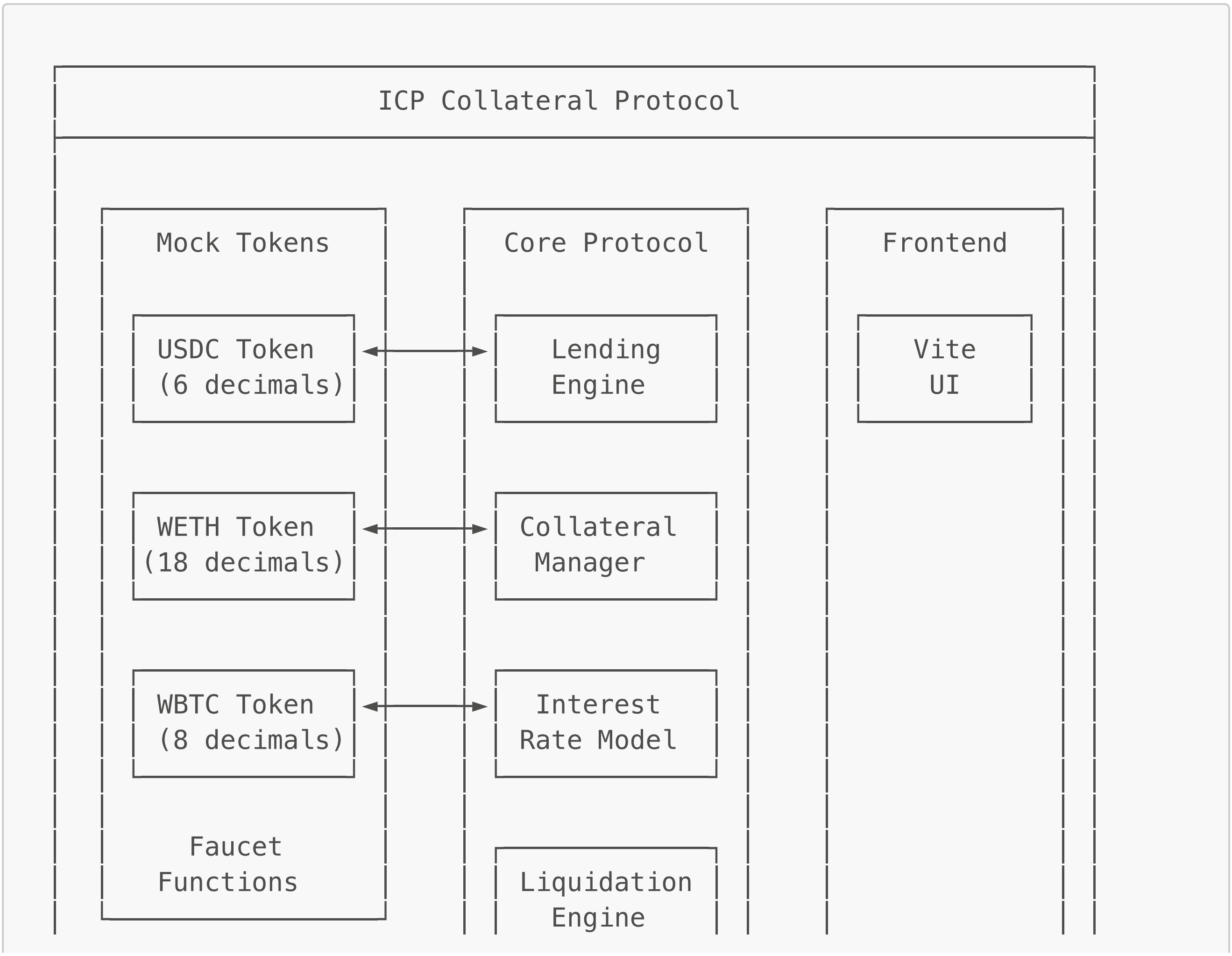
Overview

A decentralized lending and borrowing protocol built on the Internet Computer Protocol (ICP) that enables users to supply liquidity, deposit collateral, and borrow assets with dynamic interest rates and token locking mechanisms.

🎯 Key Features

- **Multi-Asset Collateral**: Support for WETH and WBTC as collateral
- **Dynamic Interest Rates**: Rate adjustment based on pool utilization
- **Token Locking**: Lock tokens for bonus rewards (1%-10% APY)
- **Liquidation Protection**: Health factor monitoring and automated liquidation
- **Mock Tokens**: Complete testing environment with faucet functionality

🏗️ System Architecture



```
│                         │ └─────────────┘ │    │              │ │
│                         │                 │    │              │ │
│                         │ ┌─────────────┐ │    │              │ │
│                         │ │    Lock     │ │    │              │ │
│                         │ │  Mechanism  │ │    │              │ │
│                         │ └─────────────┘ │    │              │ │
│                         └─────────────────┘    └──────────────┘ │
└─────────────────────────────────────────────────────────────────┘
```

📦 Canister Architecture

```
graph TB
    subgraph "ICP Network"
        subgraph "Mock Token Canisters"
            USDC[USDC Token<br/>6 decimals<br/>$1.00]
            WETH[WETH Token<br/>18 decimals<br/>$3,000]
            WBTC[WBTC Token<br/>8 decimals<br/>$45,000]
        end

        subgraph "Core Protocol"
            BACKEND[icp_collateral_backend<br/>Main Lending Protocol]

            subgraph "Protocol Modules"
                LENDING[Lending Engine]
                COLLAT[Collateral Manager]
                RATES[Interest Rate Model]
                LIQUID[Liquidation Engine]
                LOCK[Token Lock Mechanism]
            end
        end

        subgraph "Frontend"
            UI[Vite Frontend<br/>User Interface]
        end
    end

    USDC --> BACKEND
    WETH --> BACKEND
    WBTC --> BACKEND

    BACKEND --> LENDING
    BACKEND --> COLLAT
    BACKEND --> RATES
    BACKEND --> LIQUID
    BACKEND --> LOCK

    UI --> BACKEND
    UI --> USDC
    UI --> WETH
    UI --> WBTC
```

🔄 User Flow Diagrams

1. Lender Flow (Supply Liquidity)

```
graph TD
    A[User has USDC] --> B[Call faucet to get USDC]
    B --> C[Supply USDC to Pool]
    C --> D[Earn Dynamic Interest]
    D --> E[Monitor Pool Utilization]
    E --> F[Withdraw USDC + Interest]

    style A fill:#e1f5fe
    style C fill:#c8e6c9
    style D fill:#fff9c4
    style F fill:#ffcdd2
```

2. Borrower Flow

```
graph TD
    A[User has WETH/WBTC] --> B[Get Tokens from Faucet]
    B --> C[Deposit as Collateral]
    C --> D[Check Borrowing Power]
    D --> E{Sufficient Collateral?}
    E -->|Yes| F[Borrow USDC]
    E -->|No| C
    F --> G[Monitor Health Factor]
    G --> H{Health Factor < 100%?}
    H -->|Yes| I[Risk of Liquidation]
    H -->|No| J[Safe Position]
    J --> K[Repay Debt]
    K --> L[Withdraw Collateral]
    I --> M[Add Collateral or Repay]
    M --> G

    style A fill:#e1f5fe
    style C fill:#c8e6c9
    style F fill:#fff9c4
    style I fill:#ffcdd2
    style L fill:#d4edda
```

3. Token Locking Flow

```
graph TD
    A[User has Collateral Tokens] --> B[Choose Lock Duration]
    B --> C{Lock Period}
    C -->|1-30 days| D[1% Bonus]
    C -->|31-90 days| E[2% Bonus]
    C -->|91-180 days| F[3% Bonus]
    C -->|181-365 days| G[5% Bonus]
    C -->|>365 days| H[10% Bonus]

    D --> I[Tokens Locked]
    E --> I
    F --> I
    G --> I
    H --> I

    I --> J[Wait for Unlock Period]
    J --> K[Claim Bonus Rewards]
    K --> L[Tokens Unlocked]

    style A fill:#e1f5fe
    style I fill:#fff9c4
    style K fill:#c8e6c9
    style L fill:#d4edda
```

⚙️ System Flow

Interest Rate Calculation

```
graph LR
    A[Pool Utilization] --> B[Calculate Rate]
    B --> C[Base Rate: 2%]
    C --> D[+ Utilization × 15%]
    D --> E[Final Interest Rate]

    subgraph Examples
        F[40% Utilization<br/>= 2% + 6% = 8%]
        G[80% Utilization<br/>= 2% + 12% = 14%]
    end

    E --> F
    E --> G
```

Health Factor Monitoring

```
graph TD
    A[User Position] --> B[Calculate Collateral Value]
    B --> C[Apply Liquidation Threshold]
    C --> D[Calculate Debt Value]
    D --> E[Health Factor = Collateral×Threshold/Debt×100]
    E --> F{Health Factor ≥ 100%?}
    F -->|Yes| G[Position Safe]
    F -->|No| H[Position at Risk]
    H --> I[Trigger Liquidation]
    I --> J[Liquidator Receives 5% Bonus]

    style G fill:#c8e6c9
    style H fill:#ffcdd2
    style I fill:#ff8a80
    style J fill:#fff9c4
```

Liquidation Process

```
sequenceDiagram
    participant U as User (Borrower)
    participant P as Protocol
    participant L as Liquidator

    U->>P: Health Factor drops below 100%
    P->>P: Mark position as liquidatable
    L->>P: Call liquidate function
    P->>P: Verify health factor < 100%
    P->>P: Calculate collateral to seize
    P->>U: Reduce debt position
    P->>U: Reduce collateral position
    P->>L: Transfer seized collateral + 5% bonus
    P->>P: Update pool state
```

🛠️ Technical Specifications

Token Configuration

Token	Symbol	Decimals	Price (USD)	Collateral Factor	Liquidation Threshold	Can Borrow	Can Collateralize
USDC	USDC	6	$1.00	0%	0%	✅	❌
WETH	WETH	18	$3,000	80%	85%	❌	✅
WBTC	WBTC	8	$45,000	75%	80%	❌	✅

Interest Rate Model

```
Interest Rate = Base Rate + (Utilization Rate × Multiplier)

Where:
- Base Rate = 2% APY
- Multiplier = 15%
- Utilization Rate = Total Borrowed / Total Liquidity
```

Lock Bonus Structure

Duration	Bonus Rate
1-30 days	1% APY
31-90 days	2% APY
91-180 days	3% APY
181-365 days	5% APY
>365 days	10% APY

🚀 Quick Start

Prerequisites

```
# Install dfx
sh -ci "$(curl -fsSL https://internetcomputer.org/install.sh)"

# Start local IC replica
dfx start --background
```

Deploy the Protocol

```
# Clone repository
git clone <repository-url>
cd icp_collateral

# Deploy all canisters
dfx deploy --with-cycles 1000000000000
```

Get Test Tokens

```
# Claim free tokens from faucets
dfx canister call usdc_token faucet    # Get 1,000 USDC
dfx canister call weth_token faucet    # Get 1,000 WETH
dfx canister call wbtc_token faucet    # Get 1,000 WBTC
```

Basic Usage Example

```
# 1. Supply liquidity to earn interest
dfx canister call icp_collateral_backend supply_liquidity '(variant {
USDC }, 2000000000)'
```

```
# 2. Deposit collateral
dfx canister call icp_collateral_backend deposit_collateral '(variant {
WETH }, 1000000000000000000)'

# 3. Check borrowing power
dfx canister call icp_collateral_backend get_borrowing_power '(null)'

# 4. Borrow USDC
dfx canister call icp_collateral_backend borrow '(variant { USDC },
1000000000)'

# 5. Check health factor
dfx canister call icp_collateral_backend get_user_health_factor '(null)'
```

📊 Monitoring & Analytics

Check Pool Status

```
# View all pools
dfx canister call icp_collateral_backend get_all_pools

# View specific pool
dfx canister call icp_collateral_backend get_pool_info '(variant { USDC
})'
```

Account Management

```
# View account details
dfx canister call icp_collateral_backend get_account_info '(null)'

# Check lock positions
dfx canister call icp_collateral_backend get_lock_positions '(null)'

# Monitor health factor
dfx canister call icp_collateral_backend get_user_health_factor '(null)'
```

Token Information

```
# Get token configuration
dfx canister call icp_collateral_backend get_token_info '(variant { WETH
})'

# Check token balances
USER_PRINCIPAL=$(dfx identity get-principal)
```

```
dfx canister call weth_token balance_of "(principal
\"$USER_PRINCIPAL\")"
```

🧪 Testing

Run the comprehensive test suite:

```
# Make test script executable
chmod +x test_script.sh

# Run all tests
./test_script.sh
```

🔐 Security Features

- **Health Factor Monitoring**: Continuous position health tracking
- **Liquidation Protection**: Automated liquidation when health factor < 100%
- **Interest Rate Caps**: Maximum interest rate limits
- **Admin Controls**: Emergency pause/unpause functionality
- **Token Locking**: Prevent unauthorized withdrawals during lock period

🌐 Network Information

Local Development

- **Network**: Local IC Replica
- **Port**: 4943
- **Candid UI**: http://127.0.0.1:4943/?canisterId=

Canister URLs

After deployment, access canisters via:

- **Backend**: `http://127.0.0.1:4943/?canisterId=<candid-ui-canister-id>&id=<backend-canister-id>`
- **Frontend**: `http://<frontend-canister-id>.localhost:4943/`

📝 API Reference

Core Functions

Function	Type	Description
`supply_liquidity`	Update	Supply USDC to earn interest
`deposit_collateral`	Update	Deposit WETH/WBTC as collateral
`borrow`	Update	Borrow USDC against collateral

Function	Type	Description
`repay`	Update	Repay outstanding debt
`withdraw_collateral`	Update	Withdraw collateral if healthy
`lock_tokens`	Update	Lock tokens for bonus rewards
`liquidate`	Update	Liquidate unhealthy positions

Query Functions

Function	Type	Description
`get_account_info`	Query	Get user account details
`get_pool_info`	Query	Get specific pool information
`get_all_pools`	Query	Get all pool information
`get_user_health_factor`	Query	Get position health factor
`get_borrowing_power`	Query	Get maximum borrow capacity
`get_lock_positions`	Query	Get token lock positions

🛣️ Roadmap

- [ ] Oracle integration for real-time prices
- [ ] Governance token and DAO
- [ ] Flash loan functionality
- [ ] Cross-chain bridges
- [ ] Mobile application
- [ ] Advanced analytics dashboard
- [ ] Automated liquidation bots
- [ ] Insurance pool integration

🤝 Contributing

Contributions are welcome! Please feel free to submit a Pull Request.

📄 License

This project is licensed under the MIT License - see the LICENSE file for details.

🔗 Links

- Internet Computer
- IC SDK Documentation
- Candid
- Rust CDK

Built with ❤️ on the Internet Computer