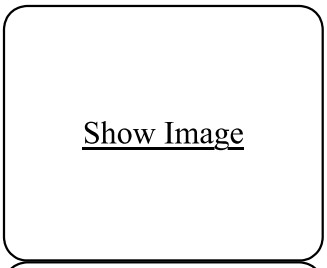
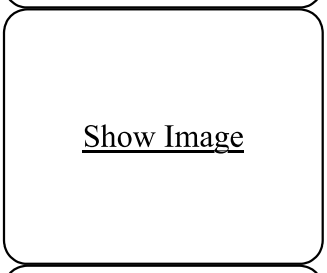
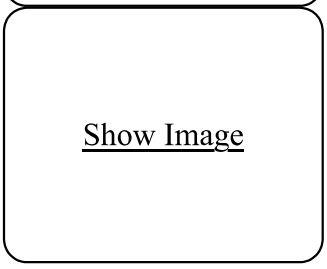
DailyApp V11 - Secured Smart Contract







> Production-ready task-based gamification platform with NFT tiers and sponsorships - Fully secured and audited

🌟 Features

- ✅ **Task System** - Complete tasks to earn points and rewards
- 🎯 **NFT Tiers** - 5 tiers (Bronze → Diamond) with different benefits
- 💰 **Sponsorship Program** - Brands can sponsor tasks with tokens
- 🔐 **Soulbound NFTs** - Non-transferable membership tokens
- 👥 **Referral System** - Earn rewards by inviting friends
- 🛡️ **Security Hardened** - Reentrancy protection, SafeERC20, input validation
- ⏱️ **Price Timelock** - 24-hour delay on price changes
- 🚫 **Blacklist System** - Protection against malicious users
- ⏸️ **Emergency Pause** - Quick response to incidents

📋 Table of Contents

🔒 Security Audit

This contract has been thoroughly tested and includes fixes for:

- ✅ Reentrancy attacks
- ✅ Integer overflow/underflow
- ✅ Access control vulnerabilities
- ✅ Front-running attacks
- ✅ DoS attacks
- ✅ Price manipulation
- ✅ Supply cap bypass
- ✅ Input validation issues

⚠️ **IMPORTANT:** Before mainnet deployment, get a professional audit from:

- ConsenSys Diligence
- Trail of Bits
- OpenZeppelin

📦 Installation

Prerequisites

- Node.js >= 16.0.0
- npm or yarn
- Git

Setup

```bash
# Clone repository
git clone https://github.com/yourproject/dailyapp-v11.git
cd dailyapp-v11

# Install dependencies
npm install

# Copy environment variables
cp .env.example .env

# Edit .env with your configuration
nano .env
```

⚙️ Configuration

Environment Variables

Edit `.env` file:

```bash
# Deployer private key (KEEP SECRET!)
PRIVATE_KEY=your_private_key_here

# Token address (ERC20 for sponsorships)
TOKEN_ADDRESS=0x...

# Initial owner (use multisig for production!)
INITIAL_OWNER=0x...

# RPC endpoints
ALCHEMY_API_KEY=your_alchemy_key
SEPOLIA_RPC_URL=https://eth-sepolia.g.alchemy.com/v2/${ALCHEMY_API_KEY}
MAINNET_RPC_URL=https://eth-mainnet.g.alchemy.com/v2/${ALCHEMY_API_KEY}

# Etherscan API (for verification)
ETHERSCAN_API_KEY=your_etherscan_key
```

Contract Parameters

Default configuration in constructor:

```solidity
// NFT Tier Configs
BRONZE:   1,000 points,   0.001 ETH, 1.1x multiplier
SILVER:   5,000 points,   0.005 ETH, 1.2x multiplier
GOLD:    20,000 points,   0.02  ETH, 1.5x multiplier
PLATINUM:100,000 points,  0.1   ETH, 2.0x multiplier
DIAMOND: 500,000 points,  0.5   ETH, 3.0x multiplier

// Sponsorship Packages (USD pegged)
BRONZE: $10
SILVER: $50
GOLD:   $100

// Token Price
Default: $0.01 per token
```

🚀 Deployment

Step 1: Compile Contracts

```bash
npm run compile
```

Step 2: Run Tests

```bash
# Run all tests
npm test

# With gas reporting
npm run test:gas

# Coverage report
npm run test:coverage
```

Step 3: Deploy to Testnet

```bash
# Sepolia testnet
npm run deploy:sepolia

# Verify contract
npm run verify:sepolia -- DEPLOYED_ADDRESS TOKEN_ADDRESS OWNER_ADDRESS
```

Step 4: Deploy to Mainnet

```bash
# ⚠️ ONLY after thorough testing and audit!
npm run deploy:mainnet

# Verify
npm run verify:mainnet -- DEPLOYED_ADDRESS TOKEN_ADDRESS OWNER_ADDRESS
```

Step 5: Post-Deployment Setup

```javascript
```

```
const dailyApp = await ethers.getContractAt("DailyAppV11Secured", DEPLOYED_ADDRESS);

// 1. Set tier URIs (IPFS metadata)
await dailyApp.setTierURI(1, "ipfs://QmBronze...");
await dailyApp.setTierURI(2, "ipfs://QmSilver...");
await dailyApp.setTierURI(3, "ipfs://QmGold...");
await dailyApp.setTierURI(4, "ipfs://QmPlatinum...");
await dailyApp.setTierURI(5, "ipfs://QmDiamond...");

// 2. Add tasks
await dailyApp.addTask(
  200,                // base reward
  86400,              // 24h cooldown
  0,                  // no tier requirement
  "Follow Twitter",
  "https://twitter.com/yourproject"
);

// 3. Transfer to multisig (CRITICAL!)
await dailyApp.transferOwnership(GNOSIS_SAFE_ADDRESS);
```

🧪 Testing

Run Full Test Suite

```
bash

# All tests
npm test

# Specific test file
npx hardhat test test/DailyApp.test.js

# With gas reporting
REPORT_GAS=true npm test

# Coverage
npm run test:coverage
```

Test Categories

- ✅ Unit tests (individual functions)

- ✅ Integration tests (complete workflows)
- ✅ Security tests (attack scenarios)
- ✅ Gas optimization tests

📚 Contract Overview

Main Functions

User Functions

solidity

```
// Complete task and earn rewards
function doTask(uint256 taskId, address referrer) external

// Mint NFT for tier
function mintNFT(NFTTier tier) external payable

// Upgrade to next tier
function upgradeNFT() external payable

// Buy sponsorship
function buySponsorshipWithToken(
    SponsorLevel level,
    string title,
    string link,
    string email
) external
```

Admin Functions

solidity

```solidity
// Add new task
function addTask(
    uint256 baseReward,
    uint256 cooldown,
    NFTTier minTier,
    string title,
    string link
) external onlyOwner

// Schedule price change (24h timelock)
function scheduleTokenPriceUpdate(uint256 newPrice) external onlyOwner

// Execute price change (after timelock)
function executePriceChange() external

// Approve/reject sponsorship
function approveSponsorship(uint256 reqId) external onlyOwner
function rejectSponsorship(uint256 reqId, string reason) external onlyOwner

// Emergency controls
function pause() external onlyOwner
function unpause() external onlyOwner
function setUserBlacklist(address user, bool status) external onlyOwner
```

View Functions

solidity

```
// Get user statistics
function getUserStats(address user) external view returns (UserStats)

// Calculate task reward
function calculateTaskReward(address user, uint256 taskId)
    external view returns (uint256 base, uint256 final, uint256 multiplier)

// Check if user can do task
function canDoTask(address user, uint256 taskId)
    external view returns (bool, string reason)

// Get contract statistics
function getContractStats() external view returns (...)

// Get users (paginated)
function getUsers(uint256 offset, uint256 limit)
    external view returns (address[] users, uint256 total)
```

💡 Usage Examples

For Users

javascript

```
// 1. Join and complete task
await dailyApp.doTask(1, referrerAddress);

// 2. Check your stats
const stats = await dailyApp.getUserStats(userAddress);
console.log("Points:", stats.points.toString());
console.log("Tasks completed:", stats.totalTasksCompleted.toString());

// 3. Mint NFT when you have enough points
const canMint = stats.points >= 1000; // BRONZE requirement
if (canMint) {
  await dailyApp.mintNFT(1, { value: ethers.parseEther("0.001") });
}

// 4. Request sponsorship
const cost = await dailyApp.getCostInTokens(0); // BRONZE package
await token.approve(dailyAppAddress, cost);
await dailyApp.buySponsorshipWithToken(
  0, // BRONZE
  "My Amazing DApp",
  "https://myproject.com",
  "sponsor@myproject.com"
);
```

For Admins

javascript

```
// 1. Add new task
await dailyApp.addTask(
    300,          // 300 points reward
    86400,        // 24h cooldown
    2,            // SILVER tier required
    "Join Discord",
    "https://discord.gg/yourserver"
);

// 2. Schedule price change
await dailyApp.scheduleTokenPriceUpdate(ethers.parseEther("0.02")); // $0.02

// Wait 24 hours, then execute
await dailyApp.executePriceChange();

// 3. Review and approve sponsorship
const request = await dailyApp.getSponsorRequest(1);
console.log("Sponsor:", request.sponsor);
console.log("Title:", request.title);

await dailyApp.approveSponsorship(1);

// 4. Emergency pause if needed
await dailyApp.pause();

// After issue resolved
await dailyApp.unpause();
```

Security Features

1. Reentrancy Protection

All external calls use `nonReentrant` modifier:

solidity

```
function doTask(...) external nonReentrant {
    // State changes BEFORE external calls
    stats.points += reward;

    // External call (if any)
    // ...
}
```

2. SafeERC20

All token transfers use SafeERC20:

```solidity
creatorToken.safeTransferFrom(msg.sender, address(this), amount);
creatorToken.safeTransfer(recipient, amount);
```

3. Input Validation

Comprehensive validation on all inputs:

```solidity
require(bytes(_title).length > 0 && bytes(_title).length <= 100, "Invalid title");
require(_baseReward > 0 && _baseReward <= MAX_TASK_REWARD, "Invalid reward");
```

4. Price Change Timelock

24-hour delay on price changes:

```solidity
// Day 0: Schedule
await dailyApp.scheduleTokenPriceUpdate(newPrice);

// Day 1+: Execute
await dailyApp.executePriceChange();
```

5. Supply Caps

Enforced on all NFT tiers:

```solidity
require(config.currentSupply < config.maxSupply, "Tier sold out");
```

6. Soulbound NFTs

NFTs cannot be transferred:

```solidity
```

```
function _update(...) internal override {
    require(from == address(0) || to == address(0), "Soulbound");
}
```

⛽ Gas Optimization

Optimizations Implemented

- ✅ Storage packing
- ✅ Immutable variables
- ✅ Calldata instead of memory
- ✅ Short-circuit evaluation
- ✅ Batch operations
- ✅ Efficient loops

Gas Estimates

Function	Gas Used	USD (30 gwei)
doTask (first time)	~150,000	$4.50
doTask (repeat)	~80,000	$2.40
mintNFT	~200,000	$6.00
upgradeNFT	~100,000	$3.00
buySponsorshipWithToken	~180,000	$5.40

Run gas report:

```bash
REPORT_GAS=true npm test
```

🐛 Troubleshooting

Common Issues

"Insufficient points"

```javascript
// Check user points
const stats = await dailyApp.getUserStats(userAddress);
console.log("Current points:", stats.points.toString());

// Check requirement
const config = await dailyApp.nftConfigs(tierNumber);
console.log("Required:", config.pointsRequired.toString());
```

"Cooldown active"

```javascript
// Check when task can be done again
const lastTime = await dailyApp.lastTaskTime(userAddress, taskId);
const task = await dailyApp.getTask(taskId);
const nextAvailable = lastTime.add(task.cooldown);
console.log("Available at:", new Date(nextAvailable * 1000));
```

"User blacklisted"

```javascript
// Contact admin - account flagged for abuse
const stats = await dailyApp.getUserStats(userAddress);
console.log("Blacklisted:", stats.isBlacklisted);
```

🤝 Contributing

We welcome contributions! Please follow these guidelines:

1. Fork the repository
2. Create feature branch (`git checkout -b feature/amazing-feature`)
3. Commit changes (`git commit -m 'Add amazing feature'`)
4. Push to branch (`git push origin feature/amazing-feature`)
5. Open Pull Request

Development Standards

- Follow Solidity style guide
- Add tests for new features

- Update documentation
- Run linter before commit

```bash
npm run lint
npm run format
npm test
```

📄 License

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

📞 Support

- **Email**: security@dailyapp.io
- **Discord**: Join our server
- **Docs**: docs.dailyapp.io
- **Bug Bounty**: bounty.dailyapp.io

⚠️ Disclaimers

- This is experimental software
- Use at your own risk
- Get professional audit before mainnet
- No warranty provided
- Follow local regulations

🙏 Acknowledgments

- OpenZeppelin for secure contract libraries
- Hardhat for development framework
- Community for testing and feedback

Built with ❤️ by DailyApp Team

Last updated: 2024