Faylotto Whitepaper

v 0.3 update 25 july 2025

Abstract

Faylotto is a decentralized lottery and social web3 platform built on EVM multi blockchain, utilizing (smart contract) technology to ensure transparency, security, and fairness throughout the lottery and other betting game processes. The platform allows users to place bets, interact socially, track player statistics, and participate in a ranking-based reward system. This document outlines the architecture, functionality, and economic model of Faylotto smart contracts, highlighting an innovative approach to the world of decentralized betting games.

1. Introduction

Conventional lotteries often face various problems such as lack of transparency, centralized control, and late payment of prizes. Faylotto is here as a solution to these problems by implementing blockchain technology to form a lottery system that does not require third party trust (trustless) and is completely decentralized.

Objective

* Provides a transparent and fair gamebet lottery system.

* Enables social interaction on the platform.

* Rewards active participants through a ranking system.

* Ensures secure and verifiable prize payouts through multi blockchain EVM.

2. Philosophy

Where human fingers dance on screens

chasing a fleeting gleam,

too many are burned by fire.

We were born slowly.

We are not fireworks,

that explode for a moment and then disappear into the night sky.

We are not furnaces,

that burn slowly but also slowly go out.

We are embers in the system,

honestly guarded by lines of code.

We are flames that do not burn,

but light the way for those

who want to play,

not to be destroyed.

Faylotto is not just a betting place.

But like a house:

that in the vast and fluid world of Web3,

there is still room for fairness

that cannot be manipulated,

there is still hope

that is not exchanged for loss.

We do not promise instant riches.

We offer a fair chance,

and a system that favors no one —

except transparency.

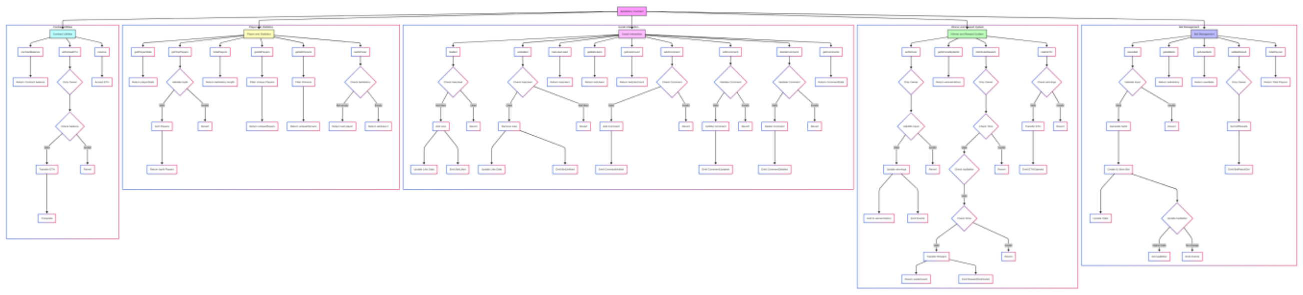
Here, luck is not determined by an invisible hand, but by trust written eternally in a chain of immovable blocks.

3. System Architecture

Faylotto smart contracts are written in Solidity (version ^0.8.25) and are MIT licensed. They are designed to handle the betting process, social interactions, winner selection, and prize distribution.

3.1 Main Components

Below is an integrated chart diagram that covers all the functions in the contract.



Explanation of Chart Structure

Integration of All Functions:

- The main chart starts from the Contract node which branches into five categories: Bet Management, Winner and Reward System, Social Interaction, Player and Statistics, and Contract Utilities.

- Each category is a subgraph containing all related functions, covering more than 20 functions in the contract.

State Variables

• **Owner:** The party who deploys the contract, responsible for administrative functions such as determining winners and distributing rewards.

• **Players:** An array that records all unique participants.

Betting Parameters

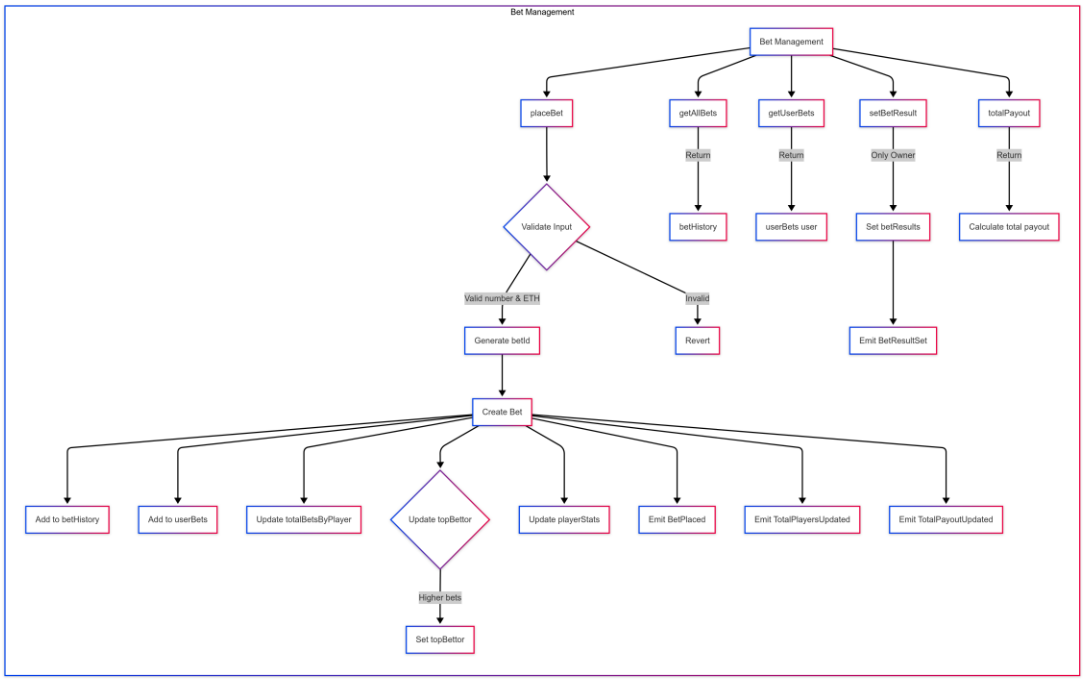
`minBet`: Minimum bet value (1).

`maxBet`: Maximum bet value (99).

`betPrice`: This fixed fee per bet (dynamic) depends on the blockchain used.

A. Bet Management

- Related functions: `placeBet`, `getAllBets`, `getUserBets`, `setBetResult`, `totalPayout`



Explanation:

`placeBet` includes input validation, bet creation, statistics updates, and leaderboard checking.

`getAllBets` and `getUserBets` is a view function to access betting history.

`setBetResult` can only be carried out by the owner to determine the betting results.

`totalPayout` calculate total payout based on bets.

Data Structures

`Bet`: Stores details of each bet (player address, bet ID, number, amount, time, etc.).

`PlayerStats`: Track individual player statistics (number of bets, total value, number of wins, payouts).

`LeaderboardReward`: Record players with the highest bets for periodic prizes.

`WinnerData`: Record winner data (address, prize amount, winning numbers, time).

`LikeData` & `CommentData`: Manage social interactions (likes and comments).

`BetResult`: Saves lottery results (ID, winning numbers, processing status)

Mapping

`userBets`: Linking a player's address to his betting history.

`betResults`: Linking the draw ID with the results.

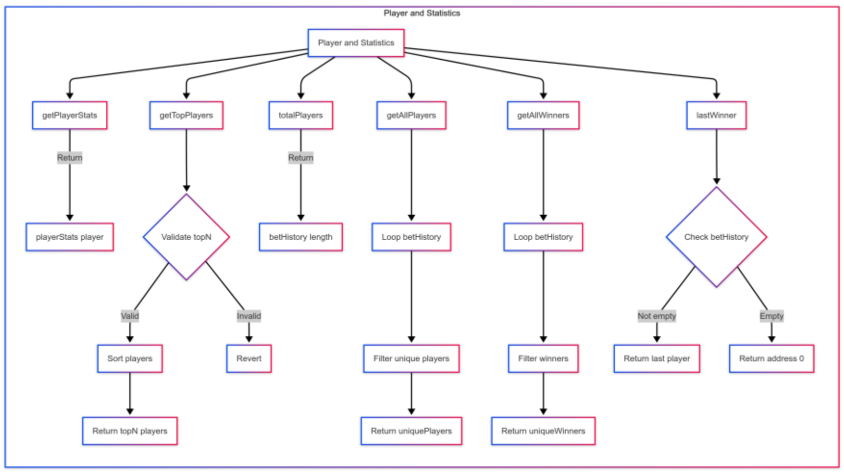
`betLikes` & `betComments`: Record likes and comments on bets.

`winnings`: Tracks the number of prizes a player can claim.

`playerStats`: Stores cumulative statistics of each player.

B. Player and Statistics

• Related functions: `getPlayerStats`, `getTopPlayers`, `totalPlayers`, `getAllPlayers`, `getAllWinners`, `lastWinner`



Explanation:

`getPlayerStats` returns player statistics such as total bets and wins.

`getTopPlayers` sort players by betting activity for the leaderboard.

`totalPlayers`, `getAllPlayers`, `getAllWinners`, and `lastWinner` provide information about participation and winners.

Event

`BetPlaced`: When the bet is made.

`WinnerAnnounced`, `WinnerSet`: When the winner is determined.

`BetLiked`, `BetUnliked`, `CommentAdded`: When social interaction occurs.

`RewardDistributed`: When the leaderboard prizes are distributed.

`ETHClaimed`: When a player claims a prize

3.2 Core Functionality

Bet

`placeBet(string _number, uint256 _times, bool _isETH, uint256 _payoutAmount)`
Allows players to bet on specific numbers (2–4 digits) with a bet multiplier. (_times). Total cost: _times × betPrice.

Condition:

Number must be 2–4 digits.

The cost sent must be appropriate (_msg.value == *times × betPrice*).

Effect:

A unique betting ID is created with `keccak256`.

Betting history, statistics and *leaderboard* updated.

Related *events* are published.

C. Social Interaction

`likeBet(bytes32 betId, string likeType)`

`unlikeBet(bytes32 betId)`

`addComment(bytes32 betId, string _comment)`

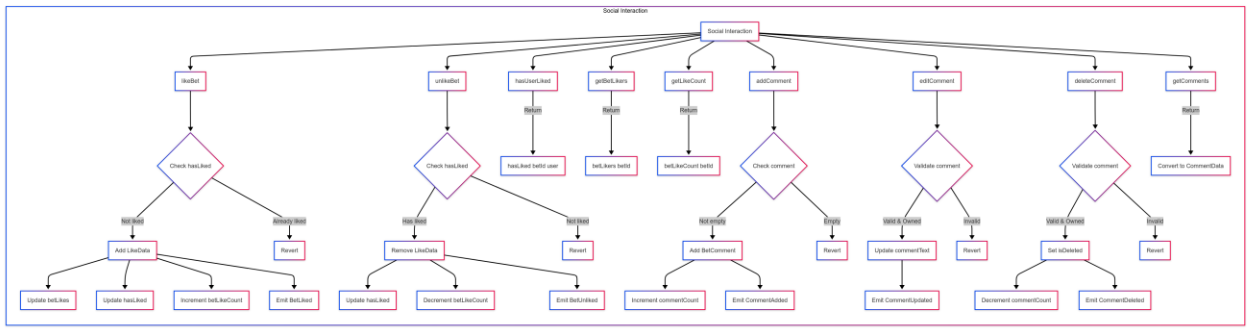
`editComment(bytes32 betId, uint256 commentIndex, string newComment)`

`deleteComment(bytes32 betId, uint256 commentIndex)`

Players can like single bets and add comments. View function is also available:

`getLikeCount`, `getBetLikers`, `getComments`

• Related functions: `likeBet`, `unlikeBet`, `hasUserLiked`, `getBetLikers`, `getLikeCount`, `addComment`, `editComment`, `deleteComment`, `getComments`



Explanation:

`likeBet` and `unlikeBet` manage like interactions on bets, with checks to prevent duplication or deletion of non-existent likes.

`addComment`, `editComment`, and `deleteComment` manage comments, with ownership validation and deletion status.

View functions such as hasUserLiked, getBetLikers, getLikeCount, and getComments provide access to social interaction data.

Winner Determination

`setWinner(address _winner, uint256 _amount, uint256 _betId, uint256 _number)`
Can only be done by the contract owner. Adding winning data and updating the player's prize balance.

Prize Distribution

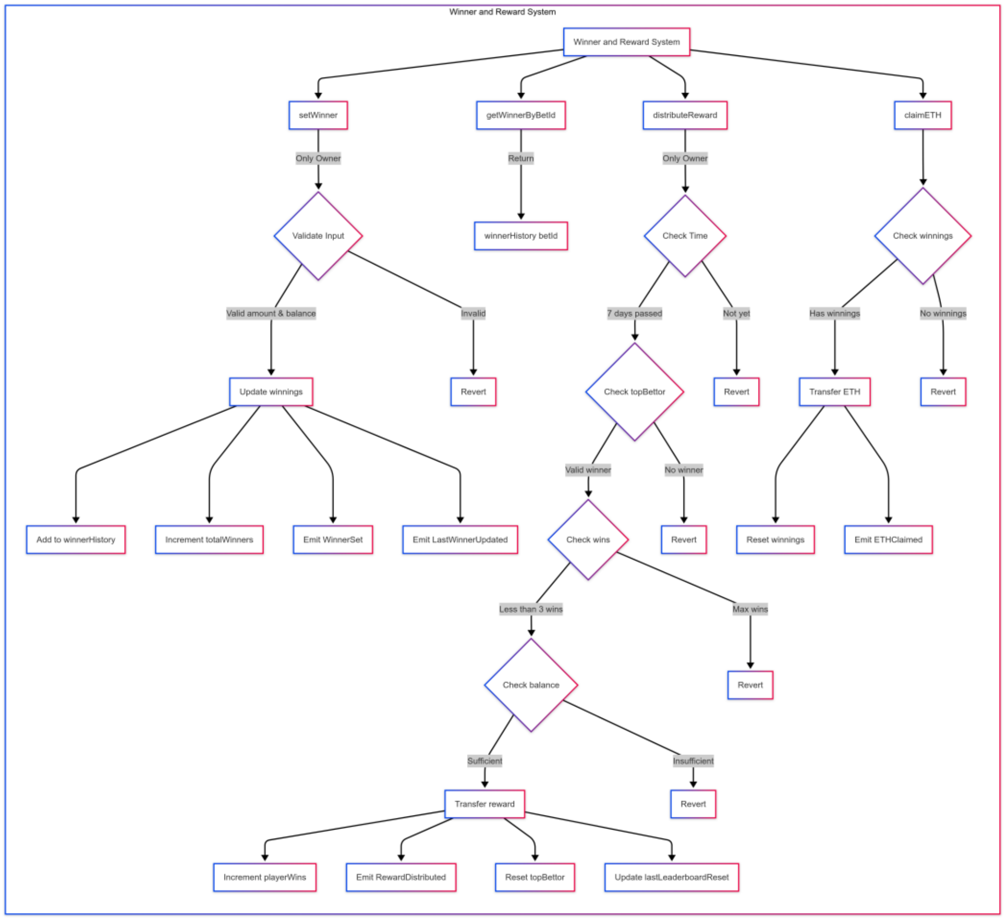
`distributeReward()`
On the same day, in a 1x12 hour process.

Prize Withdrawal

`claimETH()`
Players can withdraw their winning balance through the claim function listed in the contract.

D. Player and Statistics

• Related functions: `getPlayerStats`, `getTopPlayers`, `totalPlayers`, `getAllPlayers`, `getAllWinners`, `lastWinner`



Explanation:

`getPlayerStats` returns player statistics such as total bets and wins.

`getTopPlayers` sort players by betting activity for leaderboard.

`totalPlayers`, `getAllPlayers`, `getAllWinners`, and `lastWinner` provide information about participation and winners.

Administrative Functions

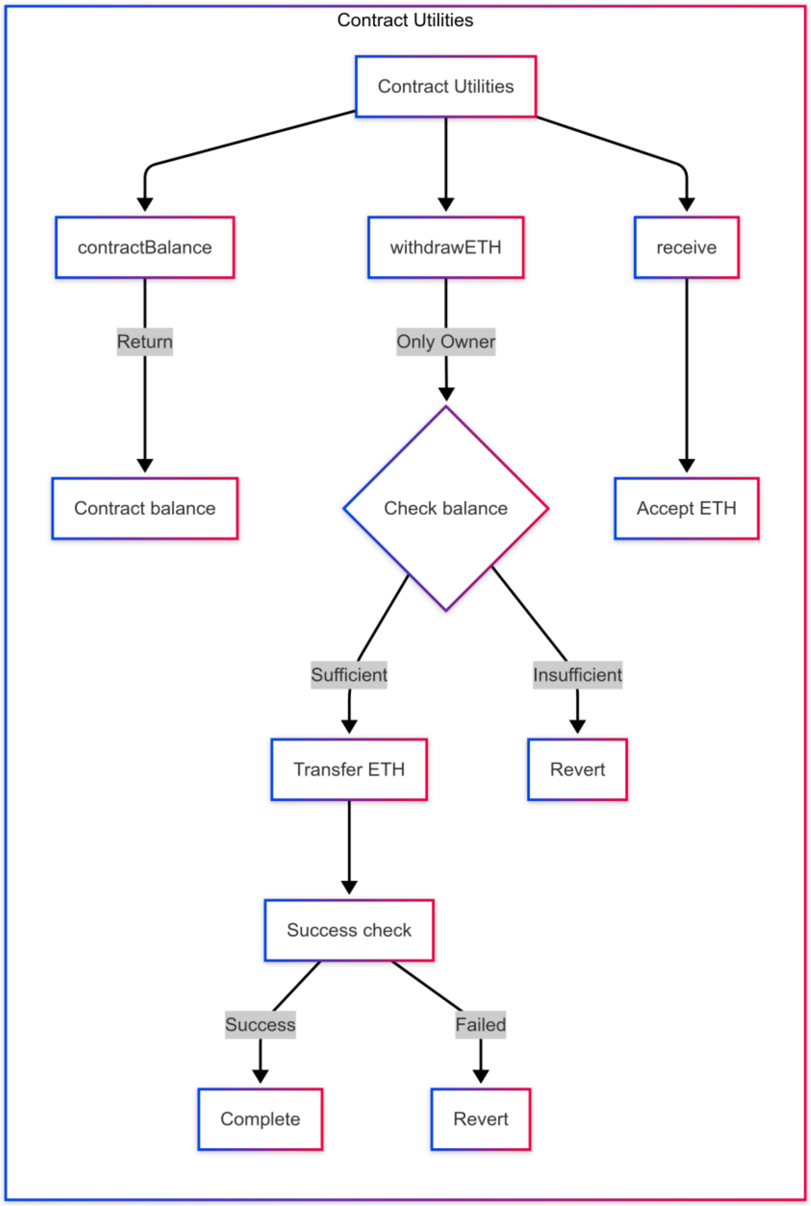
`setBetResult(uint256 _drawId, uint256 _winningNumber)`

`withdrawETH(uint256 _amount)`

Additional display function:
`getPlayerStats`, `getTopPlayers`, `getWinnerByBetId`, `getAllBets`, `getUserBets`, and etc.

E. Contract Utilities

Related functions: `contractBalance`, `withdrawETH`, `receive`



Explanation:

`contractBalance` return the contract balance.

`withdrawETH` allows owners to withdraw ETH by checking the balance.

`receive` is a fallback function to receive ETH.

4. Economic Models

4.1 Income

Betting Fees: Each bet is charged a dynamic fee depending on the Blockchain chain used. Players can increase the bet amount through the multiplier parameter.

Contract Balance: All coins from betting go to the contract balance, used to pay prizes and other distributions.

4.2 Payment

Victory: Determined by the contract owner via setWinner, and claimed by the player via claimETH.

Leaderboard Prizes: Each week, the highest betting player receives 5% of their total bets.

3.3 Incentive

Social Engagement: The like and comment feature encourages interaction between users.

Leaderboard: Encourage active competition between players.

5. Security Considerations

Access Control: Critical functions can only be accessed by the contract owner (onlyOwner).

Input Validation: Bet amount, coin amount, and comments are validated to prevent unauthorized input.

Reentry Protection: Coin transfers are done with a call method and are equipped with a success check.

Transparency: All activity is recorded on-chain and can be audited via blockchain explorers.

6. Use Cases

Players: Can bet, interact socially, and compete in the rankings.

Community: Can interact with each other, increasing social engagement in the game.

Administrators: Manage the draw, determine winners, and distribute prizes.

7. Future Development Plans

Random Number Generator: Integration of decentralized oracles (e.g. Chainlink VRF) for fairness of winner selection.

Decentralized Governance: Replacing owner decisions with DAOs or automated mechanisms.

Advanced Social Features: Adding comment replies and social incentives.

Multi-Chain Expansion: Expanding to other EVM networks for gas cost efficiency, and offering more choices.

User Interface: Creating a user-friendly UI for easier interaction.

Faylotto Brand, Logo, and Source Code Protection Policy.

v 0.3 update 25 july 2025

8. Intellectual Property Policy

Faylotto is a decentralized smart contract platform built on blockchain technology, designed to support digital lotteries, gamification incentives, and decentralized financial services. The brand, logo, and source code of **Faylotto** are core assets representing the platform's identity, trust, and technological innovation. This policy governs the global protection, usage, and enforcement of these assets, independent of governmental jurisdictions, to ensure integrity, consistency, and economic value within the decentralized ecosystem.

Objective

1. Protect the brand, logo, and source code as intellectual property worldwide.
2. Regulate usage by the community, developers, and partners to maintain platform reputation and functionality.
3. Provide a fair, transparent, and decentralized dispute resolution mechanism.
4. Promote innovation and adoption through a strong brand identity and protected code.

A. Definitions

In this Policy, the following terms are defined:

1. **Faylotto Brand**: A graphical mark, including the name “Faylotto,” logo, color scheme (#f86abc, #347af3, #2f7af4, #f36abd), slogans (**Decentralized Betting Web3 Social**), or other visual elements distinguishing the platform’s services in the blockchain ecosystem.

2. **Faylotto Logo**: The official graphic design of the platform, combining text, imagery, and colors, serving as the primary visual identity.

3. **Faylotto Source Code**: All software code, including smart contracts, decentralized applications (dApps), user interfaces, and technical documentation, developed for the platform, protected as a copyrighted work.

4. **Asset Owner**: The decentralized entity (Faylotto DAO) holding exclusive rights to the brand, logo, and source code, as determined by community governance.

5. **User**: Any individual, entity, or developer interacting with the platform, including those using smart contracts, dApps, or related services.

6. **Asset License**: Permission granted via smart contract or written agreement by the Asset Owner to use the brand, logo, or source code within a specified scope.

7. **Asset Violation**: Unauthorized use of the brand, logo, or source code, including copying, modification, or distribution harmful to the platform.

8. **Copyright**: Exclusive rights over the logo (as artwork) and source code (as literary/technical work), recognized automatically under the Berne Convention 1886.

9. **Faylotto DAO**: The decentralized autonomous organization managing platform governance, including asset-related decisions.

10. **License Smart Contract**: A blockchain-based smart contract governing the issuance, management, and termination of asset licenses.

11. **Decentralized Arbitration**: A dispute resolution mechanism via platforms like Kleros or Aragon Court for handling asset violations.

B. Scope & Principles

b.1 Scope

This Policy applies to:

- Use of the **Faylotto** brand, logo, and source code in digital platforms, blockchains, and NFT markets.
- Protection of assets in the global ecosystem, including decentralized finance and digital lotteries.
- Management of licenses via smart contracts or written agreements.
- Enforcement of sanctions and dispute resolution through decentralized mechanisms.
- Protection of the logo and source code as copyrighted works under international law.

b.2 Principles of Protection

- **Decentralization**: Assets are managed via DAO and smart contracts, independent of centralized authorities.
- **Transparency**: Decisions, licenses, and sanctions are recorded on the blockchain for public audit.
- **Integrity**: Assets must be used in alignment with **Faylotto**'s mission (innovation, trust, gamification).
- **Absolute Rights**: The Asset Owner holds exclusive rights, with the source code protected as absolute copyright.
- **Global Compatibility**: Aligned with the Paris Convention, TRIPS Agreement, Madrid Protocol, and Berne Convention.

C. Asset Protection

c.1 Protection Status

1. **Faylotto Brand**:
   - Protected as a distinguishing mark under **first-to-use** (evidenced on blockchain).
   - Includes the name "Faylotto," typographic variations, and related visual elements.

2. **Faylotto Logo**:
   - Protected as part of the brand and as artwork under the Berne Convention.
   - Includes 2D/3D designs, colors (#f86abc, #347af3, #2f7af4, #f36abd),

3. **Faylotto Source Code**:
   - Protected as a literary/technical work under the Berne Convention, with absolute copyright owned by the Asset Owner.
   - Includes smart contracts (Solidity), dApps (JavaScript, React), backend, and technical documentation.
   - Prohibited from copying, modification, or distribution without explicit license.

4. Ownership proof is recorded on the blockchain (e.g., code hash on IPFS, smart contract deployment timestamp).

c.2 Brand Registration

1. The Asset Owner may register the brand via the **Madrid Protocol** through WIPO for protection in over 130 countries.
2. Registration is optional, as **first-to-use** protection is recognized in blockchain ecosystems.
3. Registration costs are covered by the DAO, funded by the **Faylotto** token treasury.
4. Source code is not registered but protected automatically as copyright upon creation.

c.3 Rights of the Asset Owner

1. Use the brand, logo, and source code for platform operations.
2. Grant licenses via smart contracts or written agreements.
3. Prohibit unauthorized use through DAO governance.
4. Initiate disputes via decentralized arbitration.
5. Receive royalties from licenses, managed via smart contracts.

D. Asset Usage

d.1 Permitted Usage

1. **Brand and Logo**:
   - Used by the Asset Owner or authorized licensees.
   - Examples: Logo on dApps, "Faylotto" in whitepapers, branding in NFTs.
   - Must include ™ (unregistered) or ® (registered) symbols.

2. **Source Code**:
   - Restricted to the Asset Owner or explicit licensees.
   - Limited to platform operations or licensed projects.
   - Examples: Official smart contract deployment, dApp integration by partners.

3. Non-commercial community use (e.g., tutorials) is allowed with attribution: "© Faylotto, all rights reserved."

d.2 Asset Licensing

1. **License Types**:
   - **Brand and Logo**: For branding (e.g., partner dApps).
   - **Source Code**: Limited licenses for code use or modification (e.g., forks for licensed projects).

2. **License Mechanisms**:
   - **License Smart Contract**: Automates rights, duration, and royalties.
     Example smart contract function for granting a license.

```
function grantLicense(address user, uint256 duration, bytes32
assetType) public onlyDAO {
    licenses[user] = License(user, assetType, block.timestamp,
duration);
    emit LicenseGranted(user, assetType, duration); }
```

3.
- **Written Agreement**: For strategic partners, digitally signed and hashed on the blockchain.

4. **License Terms**:
- Scope: Virtual regions, service types (e.g., lotteries, DeFi).
- Duration: E.g., 1 year, renewable via DAO.
- Royalties: 5-10% of revenue, paid USDT stablecoins.
- Prohibitions: Modifying logo/code without approval, use for illegal purposes.

5. **License Revocation**: Automatic via smart contract for:
- Failure to pay royalties.
- Violation of guidelines (e.g., altering logo colors).
- Harm to platform reputation.

d.3 Usage Guidelines

1. **Brand and Logo**:
- Colors: (#f86abc, #347af3, #2f7af4, #f36abd).
- Resolution: 300 dpi (digital), 1200 dpi (print).
- Prohibited: Altering design, adding elements, use in fraudulent contexts.

2. **Source Code**:
- Must follow official documentation (e.g., smart contract ABI).
- Modifications only for licensees, with DAO approval.
- Attribution required: “Source code © Faylotto”.

3. Full guidelines are accessible via IPFS or the official **Faylotto** website.

d.4 Non-Commercial Use

1. Community may use brand/logo for education (e.g., blogs, videos) with attribution.
2. Source code is not permitted for non-commercial use unless open-sourced under a specific license (e.g., MIT, approved by DAO).
3. Harmful non-commercial use (e.g., FUD, phishing) is considered a violation.

E. Violations & Sanctions

e.1 Types of Violations

1. **Brand**:
- Unauthorized use of “Faylotto” name.
- Use of similar marks (e.g., “FaylottoX”).

2. **Logo**:
   - Copying or modifying logo for other projects.
   - Distributing logo in NFTs without permission.

3. **Source Code**:
   - Copying smart contracts/dApps without a license.
   - Modifying code for fraudulent purposes (e.g., scam contracts).
   - Distributing code without attribution.

4. **Reputation**: Using assets for fraud, phishing, or illegal activities.

e.2 Enforcement Mechanisms

1. **Warning**: DAO issues a warning via blockchain, email, or Discord, with a 7-day period to cease violations.

2. **Smart Contract Sanctions**:
   - Block violator's access to dApps/smart contracts.
   - Seize related tokens (e.g., staking rewards).
   - Disable license via revokeLicense(address user).
   Example smart contract function for revoking a license.

```
function revokeLicense(address user) public onlyDAO {
    delete licenses[user];
    emit LicenseRevoked(user);
}
```

3. **Decentralized Arbitration**: Disputes are handled by Kleros or Aragon Court, with on-chain rulings.

4. **Violator Registry**: Offenders are listed on a public blockchain registry, impacting their on-chain reputation.

e.3 Contractual Sanctions

1. **Fines**:
   - Brand/Logo: 5,000-25,000 USDT.
   - Source Code: 25,000-100,000, USDT based on damage.

2. **License Revocation**: Automatic via smart contract.

3. **Blacklist**: Ban from the **Faylotto** ecosystem.

4. **Damages**: Compensation for material (e.g., lost revenue) and immaterial (e.g., reputational) losses.

e.4 Copyright Sanctions (Logo and Source Code)

1. Logo copying: Fine up to USD 50,000 USDT
2. Source code copying: Fine up to 250,000, USDT plus cessation of distribution.
3. Offenders must destroy digital copies (e.g., NFTs, code repositories).**F. Dispute Resolution**

f.1 Mechanisms

1. **Community Mediation**: Conducted via DAO forums with a neutral facilitator.
2. **Decentralized Arbitration**: Handled by Kleros or Aragon Court, with binding on-chain rulings.
3. **Conventional Courts (Optional)**: If arbitration is infeasible, parties may use a neutral jurisdiction (e.g., Singapore), with DAO approval.

f.2 Costs

1. Arbitration costs are borne by the losing party, paid in USDT.
2. Estimated: 1,000-5,000 USDT per case, depending on complexity.

f.3 On-Chain Evidence

1. Evidence (e.g., code hashes, screenshots, transactions) is stored on IPFS or blockchain.
2. DAO verification tools validate evidence authenticity.

G. Policy Revision and Evaluation

- This policy can be revised through a DAO community vote.
- Any revisions must be recorded on the blockchain.
- Evaluations are conducted at least every 12 months or as needed by the community.

H. Document Metadata

- **Document Number:** LC-POL/ASSET/2025-01
- **Effective Date:** May 18, 2025
- **Authorized By:** Lotto Chain Decentralezed Betting Web3 Social
- **Version:** 0.1
- **Status:** Valid
- **Official Contact:** mailto:buz@faylotto.xyz| X / Twitter: @FaylottoXyz
- **Legal Basis:** Berne Convention, Madrid Protocol, TRIPS Agreement, Paris Convention

---

9. Closing

Faylotto presents a decentralized lottery system with integrated social interaction, combining betting transparency, community engagement, and a competitive reward system. While currently the winner selection is still controlled by the owner, future iterations are planned to be closer to decentralization and fairness. With a solid architecture and attractive incentives, the platform has strong potential as a blockchain-based gaming solution.

---

10. Reference

- Web Site: https://faylotto.xyz
- E-mail: mailto:service@faylotto.xyz
- X: https://x.com/FaylottoXyz

Disclaimer: This document is prepared for informational purposes only. Users are advised to interact with the Faylotto contract wisely and understand the official Faylotto regulatory laws in accordance with applicable laws, and realize that blockchain-based applications are transparent and permanent.