Whitepaper
FlashARB: Flash Automated RealTime Bet

A Decentralized Prediction Market Protocol for Short-Term Cryptocurrency Price Bets

Table of Contents

1. Introduction

Flash Automated Realtime Bet (**FlashARB**) is a fully decentralized prediction market protocol. It enables permissionless, short-term bets on cryptocurrency price movements. Users can create or join pools based on their prediction (**Up or Down**). Resolution is transparent via on-chain price oracles.

FlashARB is optimized for security, gas efficiency, and trustlessness. It offers a fair and open betting experience.

2. Features

- **User-defined Pools:** Creator selects token, duration, participant limits, and minimum stake.
- **Open Participation:** Any user may join until the pool is full.
- **Pyth Oracle Integration:** Trustless on-chain price feeds are used.
- **Automated Distribution:** Rewards are redistributed pro-rata to winners.
- **Resolution Grace Period:** A flexible settlement window is provided.
- **Fully Permissionless:** No admin intervention is required.
- **Gas Optimized:** Immutable variables and efficient data layout are used.
- **Security Hardened:** Reentrancy guards, validation checks, and controlled access are in place.
- **Detailed Event Logs:** Indexed events are available for all key operations.

3. Architecture & Contracts

3.1 BetFactory

- Deploys new Bet pools.
- Tracks active and settled pools.
- Contains core validation logic and immutable variables.
- Emits events for creation and resolution.

3.2 Bet

- An isolated contract per prediction pool.
- Handles join, start, resolve, and claim logic.
- Stores participants, pricing, status, and pool configuration.
- Enforces one-entry-per-user.
- Applies reentrancy protection and data integrity checks.

4. How It Works

4.1 Create a Bet

`function createBet(address token, uint256 durationSeconds, uint8 minParticipants, uint256 minStakeWei) external returns (uint256 betId);`

- Deploys a new Bet instance.
- The caller becomes the pool creator (for metadata).
- No tokens/ETH are required to create.

4.2 Join a Bet

`function joinBet(uint256 betId, bool direction, uint256 stakeWei) external payable;`

- User selects direction and stakes amount >= minStake.
- Records position and adds to participants.

4.3 Bet Start

`function startBet(uint256 betId) external;`

- Triggered when the participant minimum is reached.
- Fetches and stores initial price from Pyth.

4.4 Resolution & Settlement

`function resolveBet(uint256 betId) external;`

- After time expiry, fetches the final price.
- Winners are determined based on the price delta.
- A draw condition is handled.

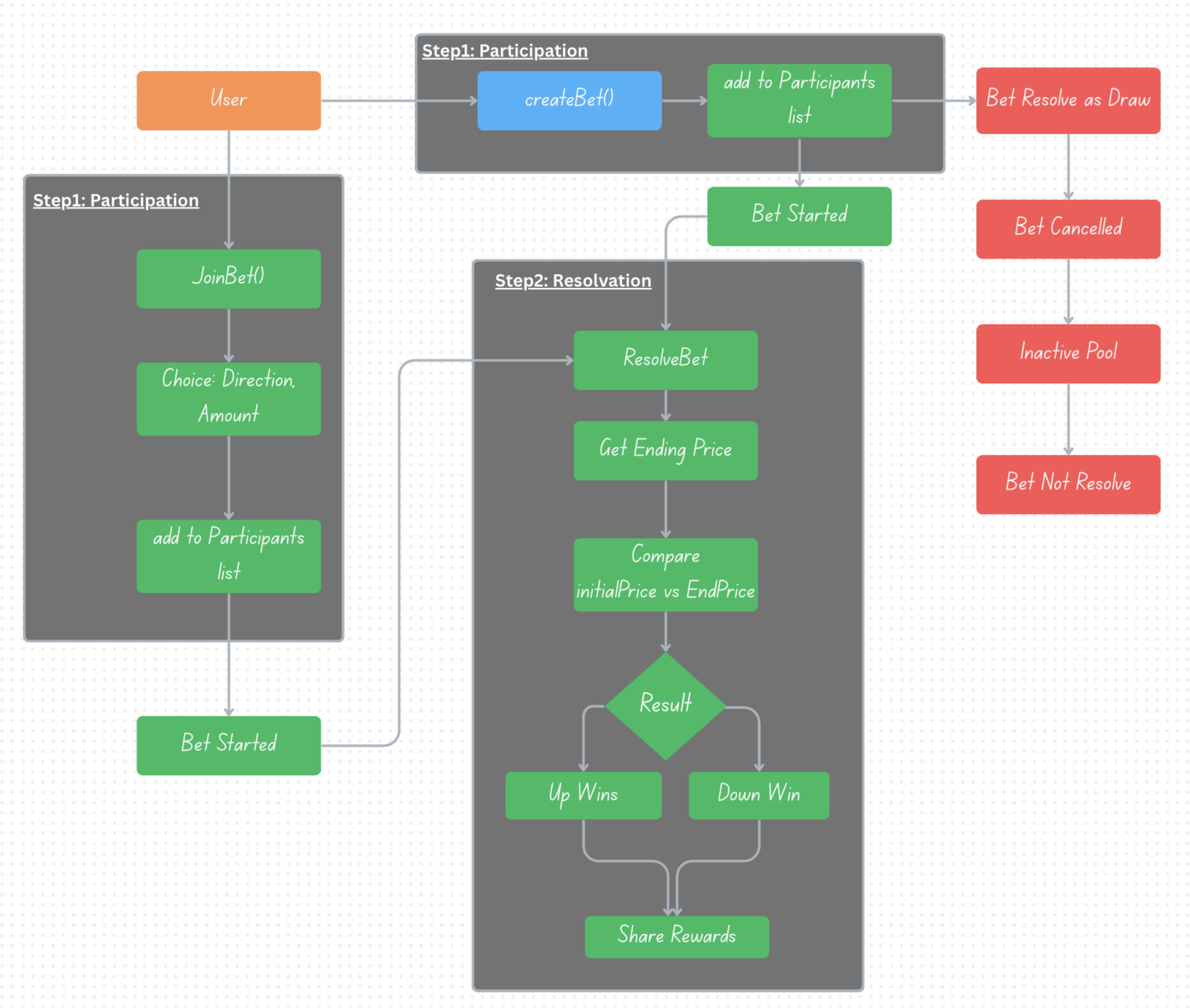
4.5 Claim Rewards

`function claim(uint256 betId) external;`

- Winners claim pro-rata from the reward pool.
- A draw returns the full stake.

5. Core Contract Methods

Function	Description	Security Features
`createBet(...)`	Deploy a new prediction pool	Input validation, reentrancy guard
`joinBet(...)`	Join an existing bet	Checks, duplicate prevention
`startBet(...)`	Record initial price from oracle	Oracle read, start timestamp logging
`resolveBet(...)`	Finalize the bet outcome	Oracle read, reward logic
`claim(...)`	Withdraw reward or refund	Prevents double claims
`forceResolveDraw()`	Draw resolution on timeout	Safety fallback for stuck bets



6. Data Structures

```
`enum BetStatus { Created, InProgress, Resolved }`
struct Participant {
   address user;
   direction direction;
   uint256 stakeUSDC;
   bool claimed;
}

struct BetInfo {
   address creator;
   address priceFeed;
   address usdcToken;
   uint256 durationSeconds;
   uint8 maxParticipants;
   uint256 minStakeUSDC;
   uint256 initialPrice;
   uint256 endPrice;
   uint256 startTimestamp;
   uint256 rewardPoolUSDC;
   uint256 feeETH;
   BetStatus status;
   Participant[] participants;
}
```

7. Oracle Integration

- Uses **Pyth Network** for asset pricing.
- Real-time price is fetched at:
    - `startBet()` for `initialPrice`
    - `resolveBet()` for `endPrice`
- On-chain pricing ensures trustless resolution.

8. Security & Edge Cases

- **One-entry rule:** Per-user entry is enforced.
- **Reentrancy protection:** Applied on all external mutative calls.
- **Zero-address and input checks:** On all sensitive operations.
- **Graceful fallback:** Allows resolution as draw if unresolved.
- **NonReentrant & CEI Pattern:** Ensures transfer safety.
- **Robust event system:** Enables accurate off-chain tracking.
- **Draw safety:** Refunds all participants.
- **Oracle read errors:** Handled gracefully.