Guess Application Documentation

Overview

Guess is a decentralized game application deployed on the **Sepolia testnet**. It has a simple user interface and a single Solidity smart contract backend. The game allows users to **participate in bets by guessing outcomes**, where the user whose guess is closest to the actual result wins a reward. The system supports ERC20 tokens (configured as a USD stablecoin like USDT) and native ETH.

There is **no traditional backend**; all logic and data storage are fully on-chain.

Key Features

- On-chain game logic (no backend).
- Automatic gift (ETH & USD tokens) on user registration.
- Admins can create and close bets with configurable parameters.
- Players guess an outcome before a deadline.
- Closest guesser wins the pooled stake minus a fee.
- Secure claim mechanism for winners and fee receiver.

Architecture

- **Frontend**: Simple UI (not covered in this doc).
- **Blockchain**: Smart contract deployed on Sepolia.
- **Tokens**: Interacts with an ERC20 token (assumed as a USD stablecoin like USDT).

Roles

Role	Abilities
Owner	Full control over contract settings and funds
Admin	Can create/close bets, but not change ownership or withdraw funds
User	Can register, place guesses, and claim rewards

Getting Started

1. Register as a User

- Call `register(string userName)`
- If the contract has sufficient balance, you receive:
    - `10^9` USD tokens
    - `0.05 ETH` native tokens

2. View Available Bets

- Use `getNextBetId()` and `getBetInformation(betId)` to list open bets.

3. Place a Guess

- Use `betOn(betId, guess)`
- You must:
    - Be a registered user
    - Have approved enough USD tokens for transfer

- Bet before the participation deadline

4. Admin Closes the Bet

- After the due date, the admin or owner calls:
  - `closeBet(betId, outcome)`
- Determines the closest prediction, calculates winnings, and updates state.

5. Claim Reward

- Winner calls `claimReward(betId)` to receive the prize.
- Fee (if any) is transferred to the designated fee receiver.

Smart Contract Functionalities

User Functions

Function	Description
`register(userName)`	Registers a new user and gifts them tokens
`getUserInformation()`	Retrieves your own user profile
`betOn(betId, guess)`	Participate in a bet with your guess
`claimReward(betId)`	Claim winnings if you are the winner
`getBetInformation(betId)`	View a specific bet's details
`getNextBetId()`	Get the ID of the next bet (useful for listing bets)

Admin/Owner Functions

Function	Description
`addBet(...)`	Create a new bet with configuration

`closeBet(betId, outcome)`	Close a bet after due date and determine winner
`changeAdmin(newAdminAddress)`	Change admin address
`changeOwner(newOwnerAddress)`	Transfer ownership
`changeFeeReceiverAddress(newAddress)`	Change who receives the collected fee
`notifyUsers(message)`	Set a public owner message for users

Owner-Only Fund Management

Function	Description
`cashOutUSD(amount)`	Withdraw unused USD tokens (preserves locked amounts)
`cashOutEth(amount)`	Withdraw ETH balance

Security and Validations

- Users cannot register twice.
- Only active users can place bets.
- ETH and USD gifts require minimum contract balance thresholds.
- Bets can only be closed after their due time.
- Users cannot claim others' rewards.
- Locked USD balances prevent overspending.

Technical Details

Token and Reward Logic

- Uses a predefined **USD token address** on Sepolia (`USD_TOKEN_ADDRESS`).

- Bets are staked in `baseStakeUnit` of USD tokens.
- On bet closure:
    - A **feePercentage** is deducted.
    - The rest goes to the **closest guesser**.
    - If the remaining prize is less than `baseStakeUnit`, the fee is skipped.

Structures

- `UserInformation`: Stores user profile data.
- `Bet`: Metadata and configuration of each bet.
- `Prediction`: Each user's guess on a bet.

Storage Mappings

Mapping	Description
`users[address]`	User info
`betIdToBet[uint256]`	All bets by ID
`betIdToPredictions[uint256]`	Predictions per bet

Testing & Deployment

- Deployed on **Sepolia testnet**
- Uses **IERC20** interface for token interactions