Web3 Number Guessing Game - Technical Documentation

Project Overview

The Web3 Number Guessing Game is a blockchain-based application built with Flutter and Solidity where players guess numbers between 0-100 and earn GUESS tokens based on their accuracy. The game leverages Ethereum smart contracts to handle game logic and token distribution in a transparent and decentralized manner. **The game is completely free-to-play** - players only receive rewards when they win, and there are no entry fees.

Table of Contents

Architecture

The project follows a client-server architecture with:

- **Client**: Flutter mobile application (guess_game)
- **Backend**: Ethereum blockchain with smart contracts
- **Integration**: Web3Dart library to connect the Flutter app with the blockchain
- **Network**: Deployed on Sepolia Testnet for testing

High-Level Architecture Diagram

```
+------------------------+       +------------------------+
|                        |       |                        |
|  Flutter Application   |<----->|  Ethereum Blockchain   |
|     (guess_game)       |       |   (Sepolia Testnet)    |
+------------------------+       +------------------------+
        |                                 |
        v                                 v
+------------------------+       +------------------------+
|                        |       |                        |
|    Local Storage       |       |   Smart Contracts      |
|  (SharedPreferences)   |       | (Game Logic & Tokens)  |
|                        |       |                        |
+------------------------+       +------------------------+
```

Frontend (Flutter)

Project Structure

```
lib/
├──── constants/          # App constants and contract addresses
│   ├──── app_constants.dart      # Main app constants
│   └──── constants.dart          # Export file
├──── contracts/          # ABI definitions and contract config
│   ├──── contract_config.dart  # Contract addresses and network config
│   ├──── erc20_abi.dart        # ERC-20 token ABI
│   ├──── game_contract_abi.dart # Game contract ABI
│   └──── contracts.dart        # Export file
├──── models/             # Data models
│   ├──── game_result.dart      # Game result data structure
│   └──── models.dart           # Export file
├──── providers/          # State management
│   ├──── app_provider.dart     # Main app state provider
│   └──── providers.dart        # Export file
├──── screens/            # UI screens
│   ├──── home_screen.dart      # Main game interface
│   └──── screens.dart          # Export file
├──── services/           # Business logic services
│   ├──── web3_service.dart     # Blockchain interaction service
│   ├──── storage_service.dart  # Local storage management
│   └──── services.dart         # Export file
├──── lib.dart            # Main library export
└──── main.dart           # App entry point
```

Key Components

1. **Main Application** (main.dart):
   - Entry point for the Flutter application
   - Configures Material Design 3 theme (light/dark mode)
   - Sets up Provider state management
   - Initializes the application
2. **Home Screen** (screens/home_screen.dart):
   - Primary user interface for the game
   - Handles wallet connection state
   - Game play interface with number input
   - Real-time result display with performance indicators
   - Statistics display (games played, total rewards, accuracy)
3. **Game Result Model** (models/game_result.dart):
   - Represents the outcome of a single game
   - Stores target number, user guess, difference, reward amount, and timestamp

State Management

The app uses the Provider pattern for centralized state management:

- **AppProvider** (providers/app_provider.dart):
  - Manages wallet connection state and user address

    - Handles game state (idle, playing, showing results)
    - Coordinates with Web3Service for blockchain interactions
    - Manages loading states and error handling
    - Stores game statistics and history

UI/UX Design

- **Material Design 3**: Modern design system with Material You theming
- **Responsive Layout**: Adapts to different screen sizes and orientations
- **Color-coded Results**: Performance indicators with intuitive color schemes
- **Loading States**: Smooth loading animations during blockchain transactions
- **Error Handling**: User-friendly error messages and recovery options
- **Dark/Light Theme**: Automatic theme switching based on system preferences

Backend (Smart Contracts)

Token Contract

GuessToken.sol - ERC-20 token contract with enhanced features:

- **Token Details**:
    - Name: Guess Token
    - Symbol: GUESS
    - Decimals: 18
    - Max Supply: 1,000,000 tokens
    - Initial Owner Supply: 100,000 tokens
- **Key Features**:
    - **Minting System**: Role-based minting with owner control
    - **Access Control**: Minter role management (add/remove minters)
    - **Pausable**: Emergency pause functionality
    - **Burning**: Token holders can burn their tokens
    - **Supply Cap**: Hard cap at 1 million tokens
- **Security Features**:
    - OpenZeppelin standard implementation
    - Pause functionality for emergency situations
    - Role-based access control

Game Contract

NumberGuessingGame.sol - Main game logic contract:

- **Game Mechanics**:
    - **Free-to-Play**: No entry fees, players only receive rewards for winning
    - **Random Number Generation**: Pseudo-random (use Chainlink VRF for production)
    - **Winning Condition**: Guesses within 20 points of target are considered wins
    - **Automatic Rewards**: Winners receive tokens automatically
- **Key Functions**:

    - playGame(uint256 guess): Main game function (free to play)
    - getUserGameHistory(address user): Retrieves complete game history
    - getLatestGameResult(address user): Gets the most recent game
    - getUserTotalRewards(address user): Total rewards earned
    - getUserTotalGames(address user): Total games played
    - getUserAverageAccuracy(address user): Average guess accuracy

- **Reward Structure**:
    - Perfect Guess (0 difference): 50 GUESS tokens (10 base + 40 bonus)
    - Excellent (≤5 difference): 17.5 GUESS tokens (10 + 75% bonus)
    - Very Good (≤10 difference): 15 GUESS tokens (10 + 50% bonus)
    - Good (≤20 difference): 12.5 GUESS tokens (10 + 25% bonus)
    - Poor (>20 difference): 0 GUESS tokens (loss, but free to play)

Security Considerations

- **OpenZeppelin Libraries**: Uses audited, battle-tested contract libraries
- **Reentrancy Protection**: ReentrancyGuard on all state-changing functions
- **Access Control**: Owner-only functions for contract administration
- **Pausable Contracts**: Emergency pause functionality
- **Input Validation**: Strict validation of all user inputs
- **Safe Math**: Built-in overflow protection in Solidity 0.8+

Integration Layer

Web3 Service

web3_service.dart handles all blockchain interactions:

- **Connection Management**: Initializes Web3 client with Sepolia testnet
- **Contract Interaction**: Loads and interacts with deployed smart contracts
- **Wallet Integration**: Manages wallet connection state
- **Transaction Processing**: Handles game transactions and confirmations
- **Error Handling**: Comprehensive error handling for blockchain operations
- **Gas Management**: Appropriate gas limits for contract interactions

Storage Service

storage_service.dart manages local data persistence:

- **Wallet Persistence**: Stores connected wallet address
- **User Preferences**: Saves app settings and preferences
- **Session Management**: Handles user session state
- **Data Clearing**: Clean data removal when disconnecting wallet

Deployment Information

🌐 Live Deployment Details

The Web3 Number Guessing Game is currently deployed on **Sepolia Testnet** with the following configuration:

Contract Addresses

Contract	Address	Purpose
GuessToken (ERC-20)	0x2AC923843d160A63877b83EC7bC69027C97bc45e	GUESS token rewards
NumberGuessingGame	0x2a7081a264DDF15f9e43B237967F3599D743B0f5	Main game logic

Network Configuration

Parameter	Value
Network Name	Sepolia Testnet
Chain ID	11155111
RPC URL	https://ethereum-sepolia-rpc.publicnode.com
Currency Symbol	ETH
Block Explorer	https://sepolia.etherscan.io (https://sepolia.etherscan.io)

View Contracts on Block Explorer

- **GuessToken Contract**: https://sepolia.etherscan.io/address/0x2AC923843d160A63877b83EC7bC69027C97bc45e (https://sepolia.etherscan.io/address/0x2AC923843d160A63877b83EC7bC69027C97bc45e)
- **Game Contract**: https://sepolia.etherscan.io/address/0x2a7081a264DDF15f9e43B237967F3599D743B0f5 (https://sepolia.etherscan.io/address/0x2a7081a264DDF15f9e43B237967F3599D743B0f5)

Get Testnet Tokens

To play the game, you need Sepolia ETH for gas fees:

Faucet	URL	Daily Limit
Sepolia Faucet	https://sepoliafaucet.com (https://sepoliafaucet.com)	0.5 ETH
Alchemy Faucet	https://sepoliafaucet.net (https://sepoliafaucet.net)	0.5 ETH
QuickNode Faucet	https://faucet.quicknode.com/ethereum/sepolia (https://faucet.quicknode.com/ethereum/sepolia)	0.1 ETH

Add Sepolia Network to MetaMask

To connect to the game, add Sepolia testnet to your wallet:

```
{
  "networkName": "Sepolia Testnet",
  "rpcUrl": "https://ethereum-sepolia-rpc.publicnode.com",
  "chainId": "11155111",
  "symbol": "ETH",
  "explorerUrl": "https://sepolia.etherscan.io"
}
```

Quick Add Button: Add Sepolia to MetaMask (https://chainlist.org/chain/11155111)

Ready to Play?

1. Add Sepolia network to your wallet
2. Get some Sepolia ETH from faucets above
3. Download the Flutter app
4. Connect your wallet and start guessing!

Getting Started

Prerequisites

- **Flutter SDK**: 3.7.0 or higher
- **Node.js**: 16.0 or higher
- **Git**: For version control
- **Ethereum Wallet**: MetaMask or compatible Web3 wallet
- **Sepolia ETH**: For testing transactions (free from faucets above)

Installation

1. **Clone the repository**

```
git clone <repository-url>
cd quiz_app
```

2. **Install Flutter dependencies**

```
flutter pub get
```

3. **Install smart contract dependencies**

```
cd smart-contracts
npm install
cd ..
```

Deployment

1. **Configure Environment** (edit smart-contracts/hardhat.config.js)

```
networks: {
  sepolia: {
    url: "YOUR_SEPOLIA_RPC_URL",
    accounts: ["YOUR_PRIVATE_KEY"]
  }
}
```

2. **Deploy Contracts**

```
cd smart-contracts
npx hardhat run scripts/deploy-testnet.js --network sepolia
```

3. **Update Contract Addresses** in lib/constants/app_constants.dart:

```
static const String guessTokenContractAddress = 'NEW_TOKEN_ADDRESS';
static const String gameContractAddress = 'NEW_GAME_ADDRESS';
```

4. **Generate ABI Files**

```
cd smart-contracts
npx hardhat run scripts/generate-abi.js
```

5. **Run the App**

```
flutter run
```

Game Mechanics

Gameplay

1. **Wallet Connection**: User connects Web3 wallet (no registration required)
2. **Game Start**: User initiates a new game (completely free)
3. **Number Input**: User enters a guess between 0-100
4. **Blockchain Processing**: Smart contract generates random number and calculates results
5. **Reward Distribution**: Winners automatically receive GUESS tokens
6. **Result Display**: Game shows target number, difference, and reward earned

Reward Structure

The game uses a tiered reward system based on guess accuracy:

Performance Level	Difference Range	Reward Amount	Description
Perfect	0	50 GUESS	Exact match - maximum reward
Excellent	1-5	17.5 GUESS	Very close guess
Very Good	6-10	15 GUESS	Close guess
Good	11-20	12.5 GUESS	Moderate accuracy
Loss	21+	0 GUESS	No reward, but free to play

Example Scenarios

Target Number: 42

Player	Guess	Difference	Performance	Reward
Alice	42	0	Perfect	50 GUESS
Bob	46	4	Excellent	17.5 GUESS
Carol	51	9	Very Good	15 GUESS
Dave	60	18	Good	12.5 GUESS
Eve	72	30	Loss	0 GUESS

Development Scripts

The project includes essential development scripts in smart-contracts/scripts/:

Essential Scripts (Kept)

1. **deploy-testnet.js**:
   - Main deployment script for testnet deployment

    - Deploys both GuessToken and NumberGuessingGame contracts
    - Sets up initial token approvals
    - Provides comprehensive deployment information and next steps

2. **generate-abi.js**:

    - Generates ABI files for Flutter integration
    - Creates filtered ABIs with only necessary functions
    - Outputs Dart files for contract interaction

3. **check-balance.js**:

    - Simple utility to check account balance
    - Useful for verifying wallet funding before deployment

Removed Scripts

The following development and testing scripts have been removed to keep the codebase clean:

- All test-*.js files (18 test scripts)
- Debug scripts (debug-*.js, check-transactions.js)
- Demo scripts (demo-*.js)
- Fix scripts (fix-*.js)
- Old deployment scripts (deploy-updated-contract.js)
- Utility scripts (transfer-tokens.js, show-test-addresses.js)

Testing

Frontend Testing

- **Unit Tests**: Test individual components and services
- **Widget Tests**: Test UI components and user interactions
- **Integration Tests**: Test complete user flows

Smart Contract Testing

- **Hardhat Tests**: Comprehensive contract testing
- **Network Testing**: Live testing on Sepolia testnet
- **Security Testing**: Audit contract security features

Manual Testing

1. Connect different wallet types
2. Test various guess scenarios
3. Verify reward calculations
4. Test error handling

Known Issues & Limitations

Smart Contract Limitations

- **Pseudo-Random Numbers**: Current implementation uses block-based randomness (not production-ready)

- **Gas Costs**: Transaction fees apply for each game (use layer 2 for lower costs)
- **Centralized Rewards**: Owner must fund the contract with tokens for rewards

Frontend Limitations

- **Mobile Focus**: UI optimized primarily for mobile devices
- **Wallet Support**: Limited to Web3-compatible wallets
- **Network Dependency**: Requires stable internet connection

General Limitations

- **Testnet Only**: Currently deployed on Sepolia testnet
- **Token Distribution**: Manual token distribution to contract for rewards

Future Improvements

Short-term Improvements

- **Chainlink VRF Integration**: Implement truly random number generation
- **Layer 2 Deployment**: Deploy on Polygon or Arbitrum for lower gas costs
- **Improved UI**: Enhanced mobile and web responsiveness

Medium-term Features

- **Multiplayer Games**: Real-time multiplayer guessing competitions
- **Leaderboards**: Global and weekly leaderboards
- **Social Features**: Share results and challenge friends
- **Achievement System**: Badges and achievements for milestones

Long-term Vision

- **Tournament System**: Organized tournaments with bigger rewards
- **NFT Integration**: Special NFT rewards for top performers
- **Cross-chain Support**: Multi-chain deployment for broader accessibility
- **Advanced Analytics**: Detailed player statistics and performance tracking
- **Governance Token**: Community governance for game parameters