FUNCTIONAL REQUIREMENTS FOR THE FREEBLING WEB APP & SMART CONTRACT

Manuel . J

Jan 18, 2024

INTRODUCTION

This document outlines the functional requirements for the FreeBling WebSite and smart contract. Web Apps will include React.js / Thrid party libraries strategies, The smart contract is a fungible, self-executing contract based on the BEP20contract-compatible standard of the Binance Smart Chain network.

THE FREEBLING WEBSITE

We are aiming for versions 1.0 and 2.0 to be production-ready and bug-free. It is currently in progress and requires blockchain developers to advance to version 2.
The work in progress is already being worked on by full-stack developers.

Responsibilities:

- Review and understand the existing Scrum Board status, BugHerd tasks, and bugs.
- Debug and fix existing issues as identified in the BugHerd.
- Implement new features as per the project backlog.
- Ensure all features are functional, fully responsive, and cross-browser compatible.
- Work with the existing database fields and develop new entry fields.
- Implement UI/UX changes as per provided design assets, ensuring high fidelity to the design.
- Finalize third-party authentications, e.g., Facebook, Twitter, and MetaMask.
- Improve SEO by making profile URLs SEO-friendly.
- Test the web application to identify and resolve any functionality or performance issues.

Here's some information about the FreeBling online site:
https://app.freebling.io/

Here is Figma mockup:
https://www.figma.com/file/FtyziR4eHSB68FIarRZw7Y/FreeBling-v2.1.3?type=design&node-id=0-1&mode=design&t=1ZK9SShRXZYhWlrF-0

the estimated initial budget is 8 - 10K for the first month, and in progress we will increase

The timeline would be 6 months, possible to extend to Full time.

Leaderboard is static, needs to be dynamic.
Prizes needs ability to add multiple, currently only accepts one,
Main image is currently set to prize image, needs a new database field for main image
YouTube and Spider tanks tasks need to be added
Ability to add social icon based on task type
Integrate MetaMask wallet signing/auth to the user's firebase/firestore database account

THE FREEBLING UTILITY COIN TOKEN (FBC)

FreeBling Utility Coin Token is a decentralized finance (DeFi) BEP20 token that can be used to champion causes for the greater good, especially in regions where aid is most needed. FBC plans to develop a donation based crowdfunding platform application for aid and assistance projects to ensure that the most dire needs are recognized and addressed in the most transparent manner using blockchain technology.

FUNCTIONAL SPECIFICATIONS

These functional specification cover requirements for the following categories:

1. General Implementation
2. Token Distribution & Tokenomics
3. Presale & Sale
4. Charities & Business
5. Costs
6. Access Control
7. Security & Testing
8. Deployment & Release Management
9. Advanced Features

General Token Implementation

GI-1	The contract should be a BEP20 token. It should implement this standard
GI-2	The contract should be able to dynamically change the fee by calling a method
GI-3	The contract should be able to restore the original fee rate set at initial deployment of the contract
GI-4	The contract should be able to determine the allowance of a specified address (Allowance is a reference to a required method in the BEP20 interface specification here. It is used to determine how much an address can spend from the total supply of tokens)

GI-5	The contract should be able to determine the available amount of tokens
GI-6	The contract should be able to determine the current transaction fees rate
GI-7	The contract should be able to exclude certain addresses from paying transfer fees
GI-8	The contract should be able to determine its name
GI-9	The contract should be able to determine its symbol
GI-10	The contract should implement safe transaction operations i.e safe mathematical calculations. In the event that an audited library does not yet exist to provide safe operations for BEP20 tokens the way that OpenZeppelin does for ERC20 tokens, then the contract should implement a local interface for safe transaction operations
GI-11	The contract should ensure that transactions fail if there is an insufficient balance
GI-12	The contract should ensure that transactions succeed if all conditions for the transactions are met

Token Distribution & Tokenomics

TDT-1	The contract should have an initial max supply of 1 Quadrillion tokens (1,000,000,000,000,000).
TDT-2	The liquidity pool of the contract should equal 40% of the max supply of tokens.
TDT-3	The dev & marketing wallet should equal 10% of the max supply of tokens.
TDT-4	The presale wallet should equal 10% of the max supply of tokens.
TDT-5	The contract should implement the following transaction fees: 1. ANTI-RUG, TRANSACTION FEE: 20% Transaction fee for any token not held for at least 1 year in a wallet. To promote HODL. 2. NORMAL TRANSACTION FEE: 12% Transaction fee a. 1% to Marketing and Operations b. 5% to Charity Wallets c. 3.5% to Liquidity Pool d. 2.5% to a monthly lottery program 3. AUTO DONATION FEATURE: 50% of transaction fees will be automatically distributed to Charity Wallet selected by Wallet Owner.
TDT-6	The charity wallets should equal 20% of the max supply of available tokens
TDT-7	The founders wallet should equal 20% of the max supply of tokens.

TDT-8	The contract should implement a system of progressive taxation to prevent sell-runs from bots (See email discussion)
TDT-9	Buy/sell orders have a maximum of 1% of the total supply so that no whale can pull the rug on investors in a single move.
TDT-10	The contract should be able to increase/decrease allowances for a specific address (Allowance is a reference to a required method in the BEP20 interface specification here. It is used to determine how much an address can spend from the total supply of tokens)
TDT-11	The contract should be able to lock charity wallets for a period of up to 2 years, and to release this lock early as required.

Presale & Sale

PS-1	DxSale protocol should be used to lock creator wallets for some specified amounts of time
PS-2	Some locks will be gradually released over time so the token needs to account for that, and for different time lock release schedules per address if necessary (See email discussion)
PS-3	Tokens should be listed in DxSale for fair launch. Fair Launch tokens are tokens that have a publicly announced launch without any form of pre-mine. (see definition here of fair launch)

Charities & Business

CB-1	The contract should be able to send tokens directly to the address of a charity organization
CB-2	The contract should be able to remove fees associated with a particular transaction for special circumstances
CB-3	The contract should be able to integrate with internal and 3rd party APIs. For instance, it could integrate with an API for online based stores like our market place and OpenSea to facilitate the automatic transfer of transaction fees to a charity selected by the business owner

CB-4	The contact should be able to exclude charity organizations that register with FreeBling Utility Coin from paying transaction fees

Costs

COS-1	The contract should employ best practices for solidity i.e optimal usage of storage and memory, arithmetic operations, mapping over arrays to eliminate the need for loops etc, in order to keep costs down
COS-2	The contract should be able to determine the current balance of a specified address
COS-3	The contract should be able to charge arbitrary fees. This might mean circumventing the recommended transfer utilities for one that can safely transfer tokens while charging the required transaction fee
COS-4	The contract should be able to revert failed transactions completely without any cost to the user

Access Control

AC-1	Contracts should be Ownable. The owner of a contract should be the account that deployed it
AC-2	The contract should be able to transfer ownership of the contract to a different account. Calling this function should be restricted to only the current owner of the contract being transferred
AC-3	The contract should be able to renounce ownership of the contract. Calling this function should be restricted to only the current owner of the contract being transferred.

Security & Testing

ST-1	There should be adequate unit testing to ensure that the contract works as expected
ST-2	There should be unit test that attempt to replicate network conditions like latency
ST-3	There should be unit tests to verify restrictions in the contract like access control

ST-4	The contract should use reusable modifiers to implement requirements for its restrictions
ST-5	The contract should be able to determine if it is communicating with another contract, an address or some other API
ST-6	The contract along with the rest of the project should follow security best practices i.e not exposing sensitive information in the contracts or the source code repository, using the most restrictive access modifiers possible for methods and variables in the contract(s) and taking all necessary precautions when integrating with 3rd party APIs etc. This requirement is especially important and must be included in the test suites/cases for the project.

Deployment & Release Management

DRM-1	The contract should be verifiable on BSCScan
DRM-2	The contract should be compatible with Defi exchange DApps like Pancake swap and Uniswap

Advanced Features

AF-1	The contract should be able to implement a daily lottery system that will reward holders with AirDrops. Holders should be notified if they win. The more of the token that is assigned to an address and the longer the tokens are held, the more chance the holder has of winning. 1% collected across all transactions for a day is given to a random address once a day
AF-2	The token should be able to facilitate the initialization of an ERC 721 NFT token for each donation to a charity organization.
AF-3	The ERC 721 token will be a completely different contract that the token will eventually communicate with
AF-4	The NFTs identifier will be a unique tokenID (ERC-721) that will be decryptable via a dApp. This dApp will be able to tell from the TokenID - the Time when the donation was made, the amount donated in Dollars, and the charity wallet that received the donation

AF-5	The contract should be able to burn tokens, making them unusable. Burning tokens will involve sending the tokens to an address where they are completely unreachable or locking the tokens away permanently within the contract. Over a period of 10 years the liquidity pool will shrink from 40% of max supply to 30% on the following schedule: i. 2% max supply burn in year 1 – half of it in May, half of it in November ii. 2% max supply burn in year 2– half of it in May, half of it in November iii. 2% max supply burn in year 3– half of it in May, half of it in November iv. 1% max supply burn in year 4– half of it in May, half of it in November v. 0.5% max supply burn in year 5– half of it in May, half of it in November vi. 0.5% max supply burn in year 6– half of it in May, half of it in November vii. 0.5% max supply burn in year 7– half of it in May, half of it in November viii. 0.5% max supply burn in year 8– half of it in May, half of it in November ix. 0.5% max supply burn in year 9– half of it in May, half of it in November x. 0.5% max supply burn in year 10– half of it in May, half of it in November