

STAKE-GAME DAPP **Breakdown and Estimations**

**Overview:**

Following is an estimate document of the staking Dapp discussed with the client. It will follow a deflationary concept of token(s) and will operate strictly on the BSC network. The staking game will be a contract only wherein the deposited tokens, after a certain amount of tax withdrawn will wait for the other party to withdraw first, early withdrawal will be subject to loss as well as tax and the remaining will get the staked amount + reward.

* Authentication through wallet
* Select opponent with whom you want to stake
* Submit the tokens you want to stake
* Set the Losing Tolerance
* Stake tokens - Winning Scenario:
* Seeing Winning Amounts
* Unstake
* Losing Scenario:
* Check Amounts
* Unstake
* Open a new competition
* Submit the tokens you want to stake

**Disclaimer:**

* All chain endpoint integrations are provided by the client beforehand.
* Contract Documentations will be given by us.
* This is valid for BSC for now.
* All fees and deployment addresses will be provided by the client for the mainnet deployment.
* Client will provide wireframes, we will handle the contract, react and integrations.
* Client will give us the designs and frontend, we will analyze the code for bugs and or missing features and in that case, the appropriate time will be estimated and added. - This will be based on the Resource Augmentation Model

**Breakdown & Estimates**

|  |  |  |  |
| --- | --- | --- | --- |
| **Features** | | **Estimated Hours** | |
| **Blockchain** | | | |
| **Pre-requisites**   * Setup web3 * Setup Solidity env * Setup dependencies * Setup RPC and WSS * Configure Builds and Version on Server | |  | |
| **Frontend (Assuming it’s 5 pages)**   * Sanity Check * Validate All Buttons / Connections * Validate Routes * Edit/Add | |  | |
| **React**   * React Conversion * React Routing * Inject web3 * Integrate Metamask * Integrate WalletConnect | |  | |
| **Contracts**   * BEP20 Contract * Taxable * Define Supply, Symbol, Decimals * Burnable * Staking Contract * Inherit BEP20 Token Address * Inherit Wallet for Staking Reward Distribution * Rebase Defined * Add Taxes * Define ‘Loss Tolerance’ Tier * Logic for STAKING/UNSTAKING * Map Addresses * Withdraw with Rewards to Unstaked Address * Withdraw Limiter * Get Staked and Mapped Address   (Leaderboard)   * Deploy Contract(s) * Verify on Explorers | |  | |
| **ABI**   * Set Exposal ABI * Emit Functions | |  | |
| **DApp Interaction**   * Web3 Call (routed by moralis) * Web3 Payloads (interacted by metamask and walletconnect) * Populate Leaderboard Module Get Addresses from Contract | |  | |
| **Quality Assurance**   * Document any bug(s) * Testnet Testing * Document Smart Contract Tests * Sanity Check * ABI Documentation * Token and Contract Documentation | |  | |
| **Development Estimated hrs** |  | |
| **Quality Assurance Estimated hrs** |  | |
| **Total Estimated hrs** |  | |
|  |  | |

**Augmented Personnel Specific Engagement**

|  |  |
| --- | --- |
| **Task** | **Resources** |
| Pre-requisites | MERN Developer, DevOps Engineer |
| Frontend | Frontend Developer |
| React | MERN Developer |
| Contracts | Solidity Developer |
| ABI | Solidity Developer |
| Dapp Interaction | MERN Developer |
| QA | QA Engineer |

**Resources Engagement**

|  |  |
| --- | --- |
| **Resources** | **Expertise** |
| 1 | MERN Developer |
| 1 | Solidity Developer |
| 1 | DevOps Engineer |
| 1 | Frontend Developer |
| 1 | QA Engineer |

**Technology Stacks**

|  |  |
| --- | --- |
| **Operation** | **Stack** |
| Connectivity | Web3 |
| Smart Contract | Solidity |
| Functionality | MERN |

**Milestones**

|  |  |  |
| --- | --- | --- |
| **Milestones** | **Timeline** | |
| 1- Prerequisites and FrontEnd | | 2 Weeks |
| 2- Contract Development and ABI | | 3 Weeks |
| 3- REACT Conversion and DAPP Interaction | | 2 Weeks |
| 4- Quality Assurance | | 2 Weeks |