User-Friendly Nomination

Problem Statement

Nominating KSM (and DOTs in the future) is a complex process:

1. Beginners:
    a. Create Controller Account
    b. Transfer funds to controller account

    c. Bond funds
    d. Choose Validators:
        i. Reputation
        ii. Likelihood of staying in the Validator Pool
        iii. Profitability
2. Advanced users
    a. All of the above
    b. Figure out the strategy
3. Maintaining profitable nominations – manual calculations, plus all of the above

Proposed Solution

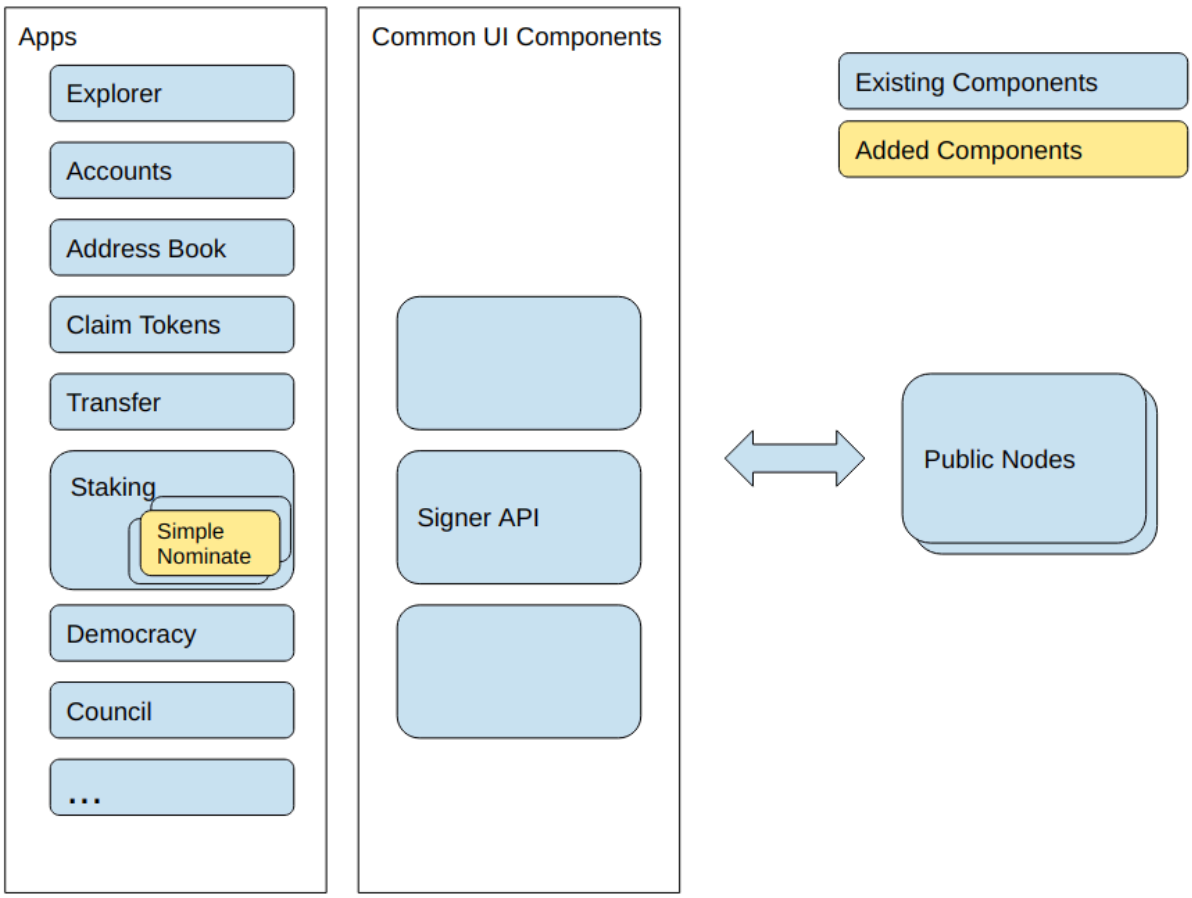
The roadmap for the solution contains three steps:

1. Implement a very simple, beginners-friendly 2 Nomination process for people who own KSMs but do not know the technical details (and don't want to know)
2. Create a update solution – evaluate profitability of existing Nominations and alternatives, re-shuffle automatically
3. Start creating customization tools: Conservative/Aggressive slider, Exclude/Include, etc.

Bellow is the spec for the 1st step which covers the majority of potential users. Specs for the steps 2 and 3 will follow.

Frontend Architecture

Simple Nomination will be implemented as a new tab (called Nomination) in Staking UI app according to the frontend system architecture below.



Simple Nomination

Common Notes

All screens will have a question mark “Help” icon, which leads to fairly simple, but detailed description of the Nomination process, rules and links to other resources for further info.

Usecase 1: Nominating

Pre-Conditions

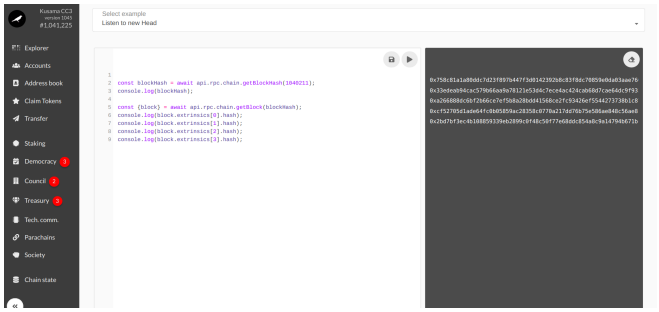
We have a user (person) with a browser UI that stores encrypted private key for a Stash Address. User knows the password to decrypt that private key. Stash Address has some KSM (DOT) balance. Alternatively, the user may use Polkadot Browser Extension (https://github.com/polkadot-js/extension) to store the private key and sign transactions, but this situation is not in scope for this proposal, we are only going to handle nomination when Stash Address (and accordingly, Controller Address - see below) is stored in the Browser UI.

The user wants to use his/her KSMs (DOTs) to participate in mining by nomination, but does not want to learn a lot of details about the Polkadot nomination process.

Our task is to facilitate the nomination process for this user in the Polkadot UI as much as reasonably possible.

Implementation Details

1. We plan to work in the fork of this repository: https://github.com/polkadot-js/apps/tree/master/packages
2. Create a tab in the Staking App in the UI:



The tab is called “Nominate”. This tab should have three input controls: (1) select the stash address, if there is more than one, (2) input the amount of KSW (DOT) to use in nomination, and (3) one button “Nominate”.

   a. Also user’s balances are displayed:
      i. Total balance
      ii. Staked balance (a.k.a. Locked or Bonded)
      iii. Unstaked (or Free) balance

      The amount of KSW (DOT) is limited so that after bonding the user has enough freeBalance (unstaked) left in order to run other transactions such as “Stop Nomination” or “Change Nominees”. The default number of such transactions is yet to be decided, but it will be configurable in the UI configuration files.
      The amount field will be pre-populated with maximum possible amount.

When the Nominate button is pressed, a pop-up is displayed that explains that funds will be locked for the Nomination duration and will remain locked for at least for *staking.bondingDuration* eras (number of eras is read from chain state) after Nomination is stopped (and show how many days approximately this takes), and asks the user to confirm. If

the user confirms, a series of actions is taken, see below.

3. Nomination Controller address is created (if it does not yet exist).

    a. The first step is to check if there is currently locked balance from the selected stash address. If there is, we read the controller address and check if it exists among the user's accounts. If it does, we automatically select it and go to the next step. If it does not, we also go to the next step, but select the "Create New Controller Address" option.
    b. In this step, we ask the user to select the Controller address from the drop-down list of existing addresses found in the UI data, create a new Controller address, or import a Controller address into the UI from JSON file.

4. Read transaction fee from the node and existential amount to estimate how much KSW (DOT) we need to reserve for updating nominees or stopping nomination. Transfer the needed amount of KSW (DOT) to the controller address so that:
    a. Controller address can perform at least two transactions: Nominate and Stop Nominating
    b. Controller address has at least the existential amount after performing these two transactions

    Note: The user will have to input their stash account password at this stage.

5. Send Staking.Bond transaction from Stash Address so that funds are bonded to the Controller account. The UI will check that Stash Address has enough freeBalance to send Staking.unbond transaction as well.

    Starting from this point, the funds are controlled by the Controller address.

    Note: The user will have to input their stash account password at this stage again.

6. Choose the optimal set of nominated validators. See the Selection Algorithm below. At this point UI will request the list of all current and waiting validators, their parameters (such as commission, steak, Identity, current era points), and their history, like it is done on these pages:

    - Validators and their properties: https://polkadot.js.org/apps/#/staking
    - Validators history: https://polkadot.js.org/apps/#/staking/query

7. Send transaction staking.nominate from Controller address that will nominate the optimally selected validator set.

    Note: The user will have to input their Controller account password at this stage.

Usecase 2: See Nomination Status

Pre-Conditions

User has utilized Simple Nomination. It does not have to be current, user may have used “Stop Nomination” after that.

Implementation Details

1. If nomination is current: Read the list of validators currently nominated by user’s controller address and display in the “Simple Nomination” tab as a table. Each row will contain:
    a. Clickable validator address that will take user to validator stats if clicked
    b. Nomination status: Active, non-active (Nominators may be grouped by this status like it is done now)
    c. Validator parameters such as stake, era points, Identity, etc.
2. In any case: Display the date and time until when the bonded funds are locked. Display each portion of locked funds in a separate line because each will have separate unlock time and amount. Explain that funds will remain locked until that time even if the user stops nomination.

Usecase 3: Update Nomination

Pre-Conditions

User has utilized Simple Nomination and it is current.

Implementation Details

1. Read the current balance of Controller address and only enable this feature if freeBalance is sufficient to send three transactions: (1) staking.nominate transaction, (2) staking.chill transaction, (3) staking.unbond, and remain above existential amount. Otherwise, display a warning and explain what to do.
2. If feature is enabled, the button “Update Nomination” will appear on the “Simple Nomination” tab. When clicked, all actions from the “Nominate” use case will repeat (except creating the Controller address and sending funds to it).

Usecase 4: Stop Nomination

Pre-Conditions

User has utilized Simple Nomination and it is current.

Implementation Details

1. The “Stop Nomination” button will be displayed on the “Simple Nomination” tab. When pressed, staking.chill() transaction will be issued from the Controller account, and staking.unbond() transaction will be issued from the Stash account.
2. All remaining balance will be transferred from the Controller account to the Stash account.
3. The “Simple Nominate” screen will update to display the time when funds will be unlocked, and will remain in this state until funds are unlocked.

Optimal Validator Set Selection Algorithm

1. Maximal mathematical expectation of return. Calculate expected return as it is currently done here: https://polkadot.js.org/apps/#/staking/returns and sort validators by their expected return.
2. Remove validators without verified identity.
   Identity of a validator with address *valAddress* is verified if at least one of the conditions is met:

   A. indentity.identityOf(valAddress) returns a non null Registration structure that has non-empty judgements member array. For example:

```
{
    "judgements":[[0,{"Reasonable":null}]],
    "deposit":10000000000000,
    "info":
    {
        "additional":[],
        "display": {"Raw":"0x5374616b6572205370616365"},
        "legal": {"None":null},
        "web": {"Raw":"0x68747470733a2f2f7374616b65722e7370616365"},
        "riot": {"Raw":"0x40676e6f737369656e6c693a6d61747269782e6f7267"},
        "email": {"Raw":"0x68656c6c6f407374616b65722e7370616365"},
        "pgpFingerprint":null,
        "image":{"None":null},
        "twitter":{"None":null}
    }
}
```


B. Condition A is true for the address returned by identity.superOf(valAddress)

3. Read the list of nodes waiting to validate like it is done here: https://polkadot.js.org/apps/#/staking/waiting, take the top waiting account, and remove all validators that have current steak (own plus nominations) below the top waiting validator.
4. Select top 16 (`MAX_NOMINATIONS`) validators from the remaining list.The number 16 is hardcoded in Staking frame (module) and there is no way to read it on the client side, so we also hardcode it.

Milestones

Milestone 1 (done)

Funding required: 2800 KSM

- Substrate node is deployed locally with 42 validator addresses, out of which 41 has verified identity. Validator count is set to 40 in the staking module. Validator without verified identity is within top 10 by estimated nomination return (has enough steak amount to be in the list of 40, but low enough to be close to the bottom of this list).
- Use case 1: User can Nominate
    - User opens Simple Nominate tab, selects stash address, selects “Create New Controller Address”, inputs nomination amount, and clicks “Nominate”. Series of actions are taken as described in use case 1, which result in running nomination of 16 addresses.
    - Same as previous, but using existing Controller address
    - Same as previous, but using an option to import Controller address

Milestone 2 (2-3 weeks)

Funding required: $7000 (in KSM)

- Use case 2: User can see nomination status
    - User clicks on “Simple Nominate” tab
    - List of nominated validators is displayed
    - Unlock time is displayed with explanation about bonding period
- Use case 3: Update Nomination
    - User clicks on “Simple Nominate” tab and clicks “Update Nomination” button
    - Series of actions is taken, which results in a new list of 16 nominated validators
- Docker image for running deliverables for acceptance
    - Dockerfile to run substrate node
    - Dockerfile to run the UI

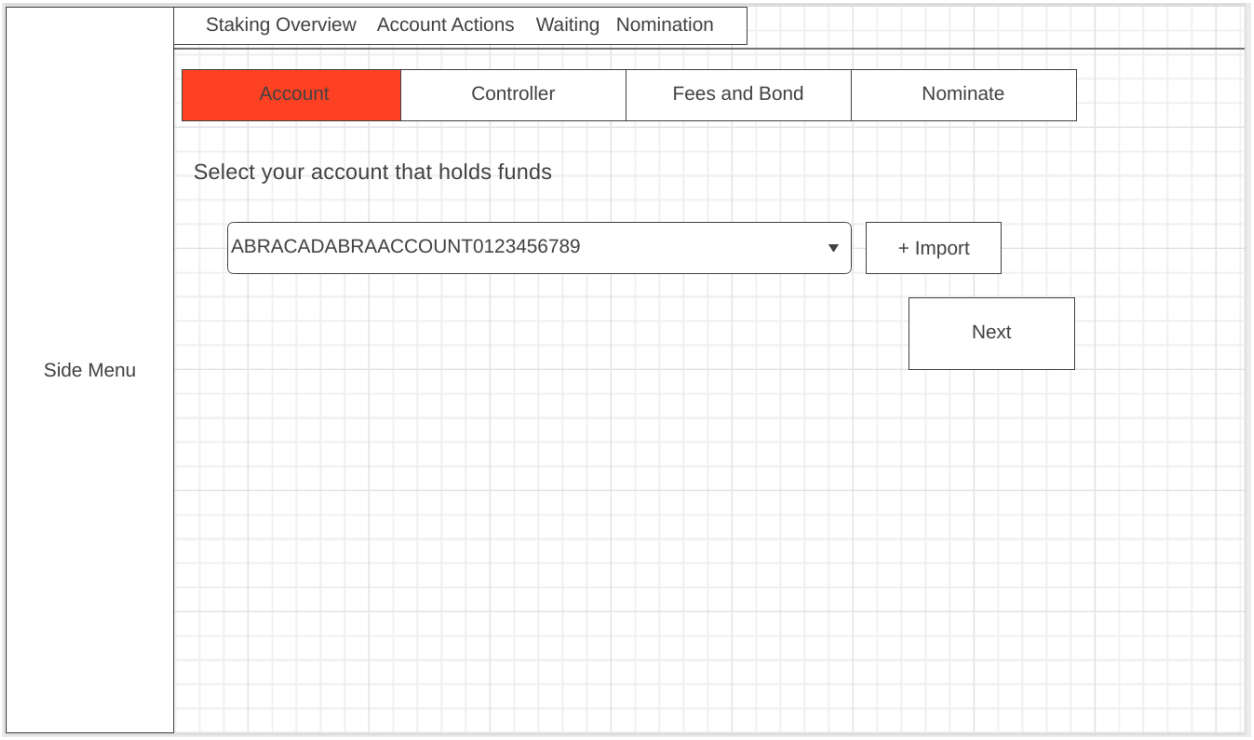
Team
Team behind this project is 1 FT Dev, 1 Architect part time and 1 PM part time
www.usetech.com
For our work so far on Substrate and other blockchains see our GitHub:
https://github.com/usetech-llc?tab=repositories

Preliminary UX

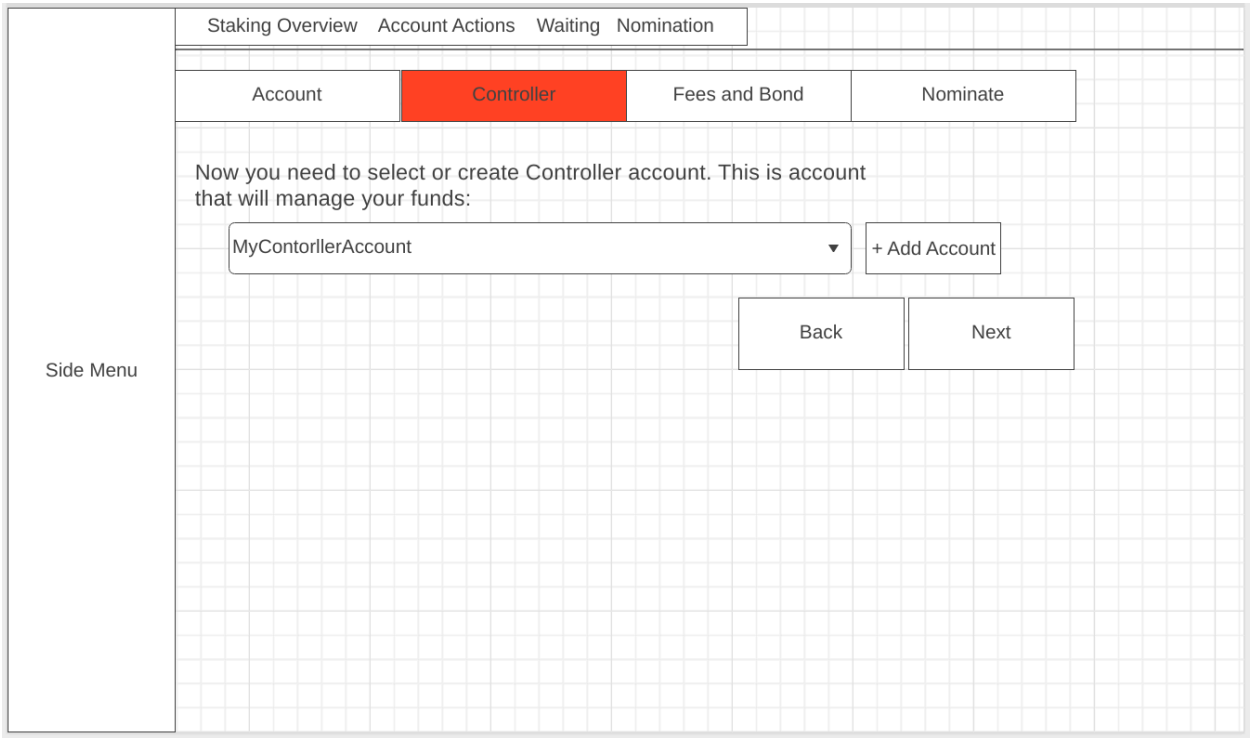
All screens will have a progress indicator that shows what currently happens.

Some screens may be skipped if not needed.

Screen #1 - Account: Select stash address, click “Next”

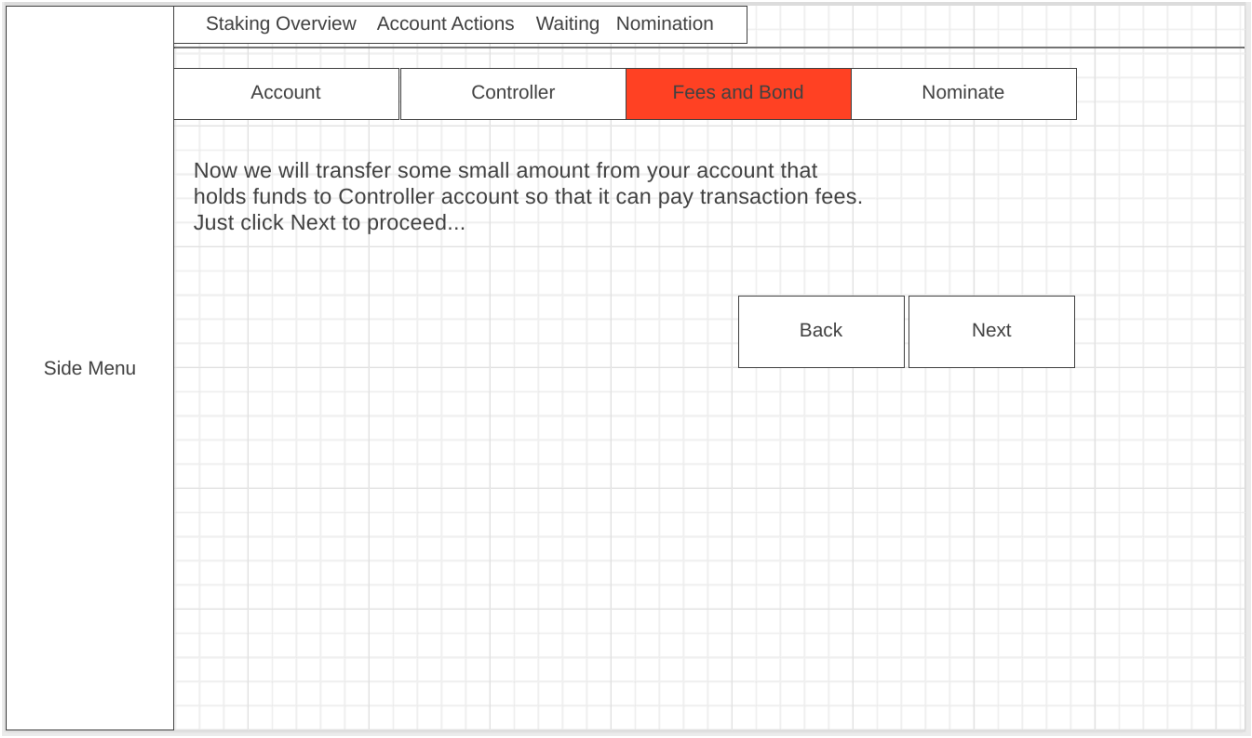


Screen #2 - Controller: Create/Import/ or Select controller address, click “Next”. When controller address is created, the name will be prefilled as “MyControllerAccount”.

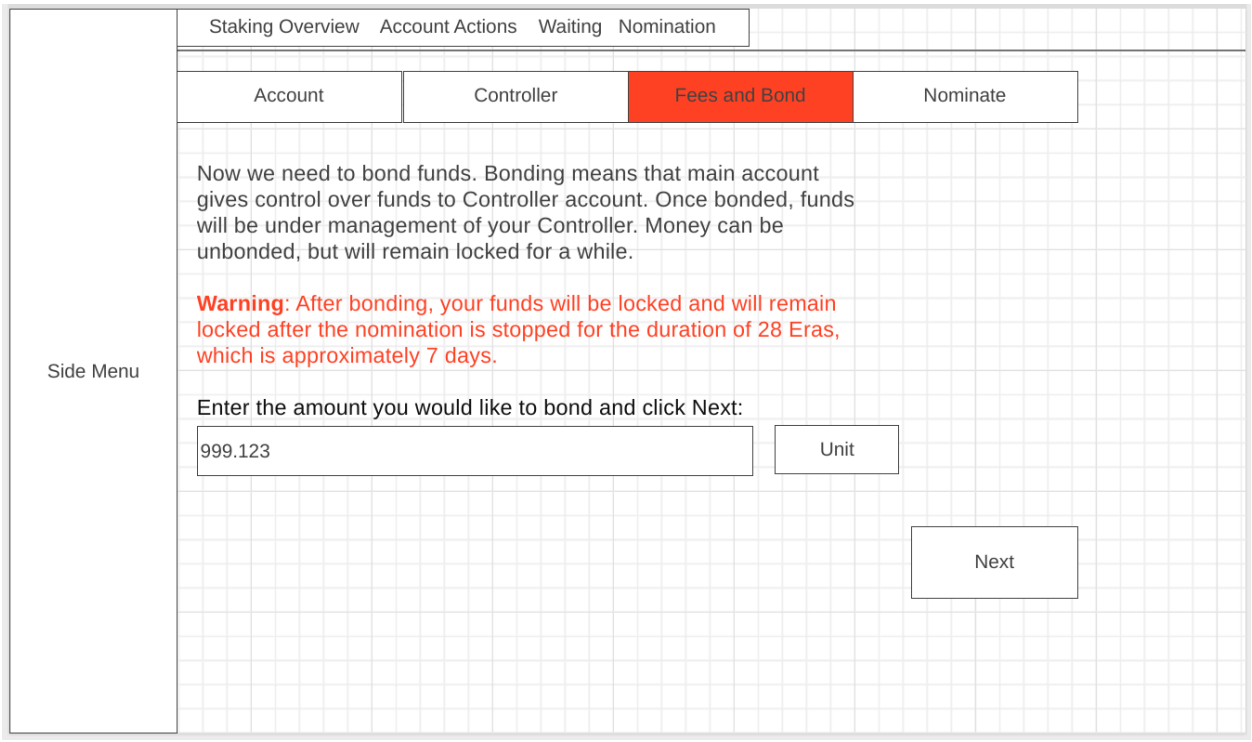


Screen #3 - Feed and Bond:

a. Enter Stash password to transfer DOTs (KSWs) to Controller so that it can pay fees if needed



b. Input amount to bond, click “Next”



c. Enter password for Stash account to send bond transaction

Screen #4 - Nominate:

a. Click Nominate, enter password for Controller account to send nominate transaction