Interactive Account Labelling feature

This simple feature could take the form of a standalone API and barebones frontend that allows anyone to tag an account in our database with any type of label. The first iteration would be a simple html page with two text boxes that is consuming a public REST API that interacts with the azure blob to store accounts and their associated labels.

This table, which we have previously referred to in the abstract as the “Translation Table”, would have a unique key for every account-label, where the account is the account hash and the label is the type of class we are labelling. The reason we call it the translation table is that one of its purposes would be to translate a query for a certain label value (such as exchange = T, or fraud = T) in the main Alethio frontend into a list of accounts having a label with that value, which would then be consumable by APIs on the backend, like the ranking API, or machine learning pipelines.

Each set of unique account-labels has an associated value. It would be ideal to have a fourth column in the Translation Table indicating whether or not the label is binary (which is itself a binary field – no NULLs allowed).

There is only one API call that is crucial to getting started:

**addLabel(account, label, value = TRUE, overwrite=TRUE, binary = T)**

functionality:

* Add a record to the translation table keyed by the account and label, with the specified value (or default value), and a flag for whether or not the label is binary.

Parameter/Error Handling:

* For binary labels, the value parameter is optional, and the assumption would be that the label is a “positive” for that type of classification, unless the client specifies value=FALSE. (All accounts that have not been labeled are assumed to be negative).
* If the user specifies binary = F, the value parameter must be specified. If it is not, respond with an error.
* If overwrite=F and that account-label already exists in the translation table, respond with an error including the existing value of the label, and asking whether to overwrite.
* If overwrite = T, respond with success and overwrite the label.
* If the user specifies binary = T (F) during an overwrite for a variable that was previously set to binary=F (T) (i.e. tries to change the multiplicity of the label), respond with an error, suggesting that the user create a new label with a different name
* If the user specifies binary = T and then attempts to specify any value other than T/F, return an error.
* If that account-label does not exist, it could be because that account has never been labeled at all, or because it has only been labeled with different “types”, or because the account does not even exist in the overall blob database (i.e. it is invalid or hasn’t been scraped yet).
  + In either of the first two cases, simply add a new record for that account-label (multiple records for a single account are fine, as long as they have distinct labels – we just don’t want two records for the same account-label).
  + In the last case, where the account does not even exist in our database, return an error saying that.

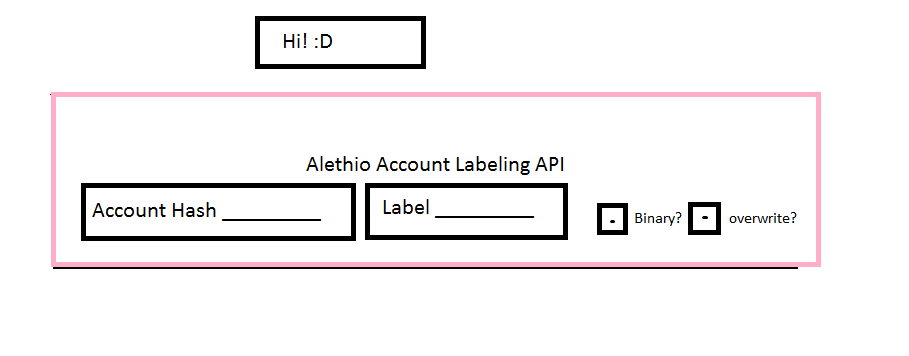
It would also be good to have a function for removing a label, which would be very simple:

**removeLabel(account,label)**

functionality:

Remove the account-label from the translation table.

Frontend:



Additional features we will want in the future:

* A private API with database entries that are user-specific
* A feature for labelling specific transactions or relationships

Questions: What assumption do we make about the value of a non-binary variable for accounts that have not yet been labeled? There is no obvious default value in some cases. This is probably going to be decided on a case-by-case basis.