Transaction Reconciliation Application

Category: Technical

Version: 1.0

Date: 2 Dec 2020 - 6 Dec 2020

Motive

To match transactions between Tutuka and Client files.

Layout and Design

Angular framework is used in this application for building single-page client applications using HTML and TypeScript. It implements core and optional functionality as a set of TypeScript libraries that we import into out app.

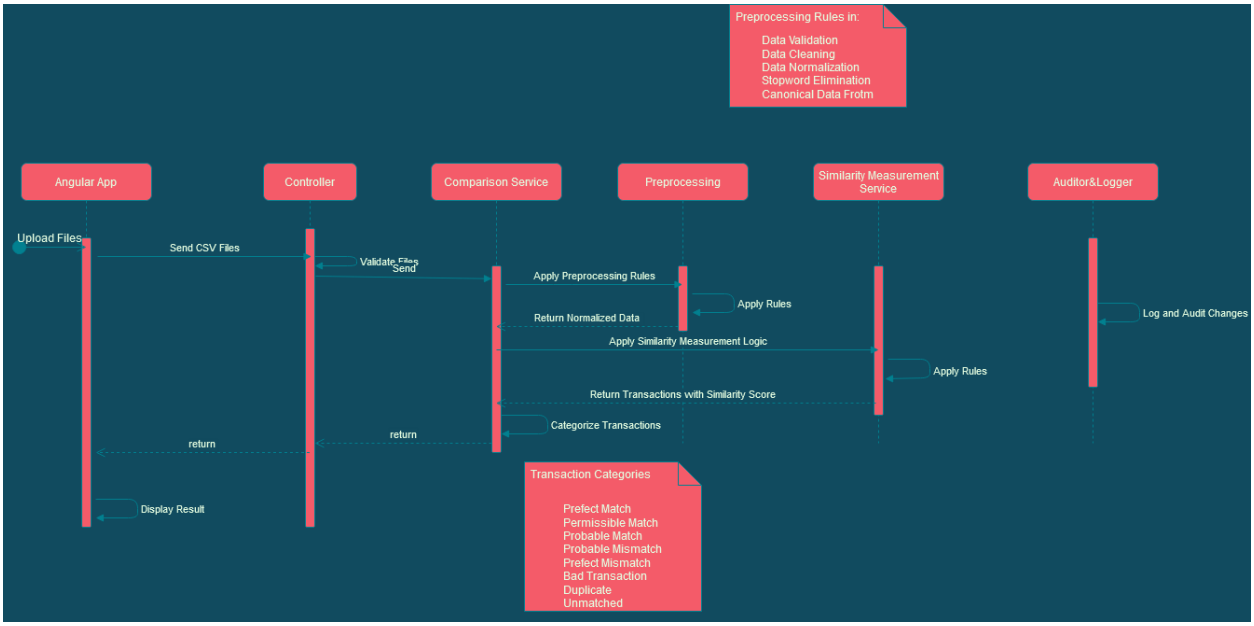
The architecture of an Angular application relies on certain fundamental concepts. The basic building blocks of the Angular framework are Angular components that are organized into NgModules. NgModules collect related code into functional sets; an Angular app is defined by a set of NgModules. An app always has at least a root module that enables bootstrapping, and typically has many more feature modules.

In this application, the following guidelines from experts have been considered towards creating a good layout and design :

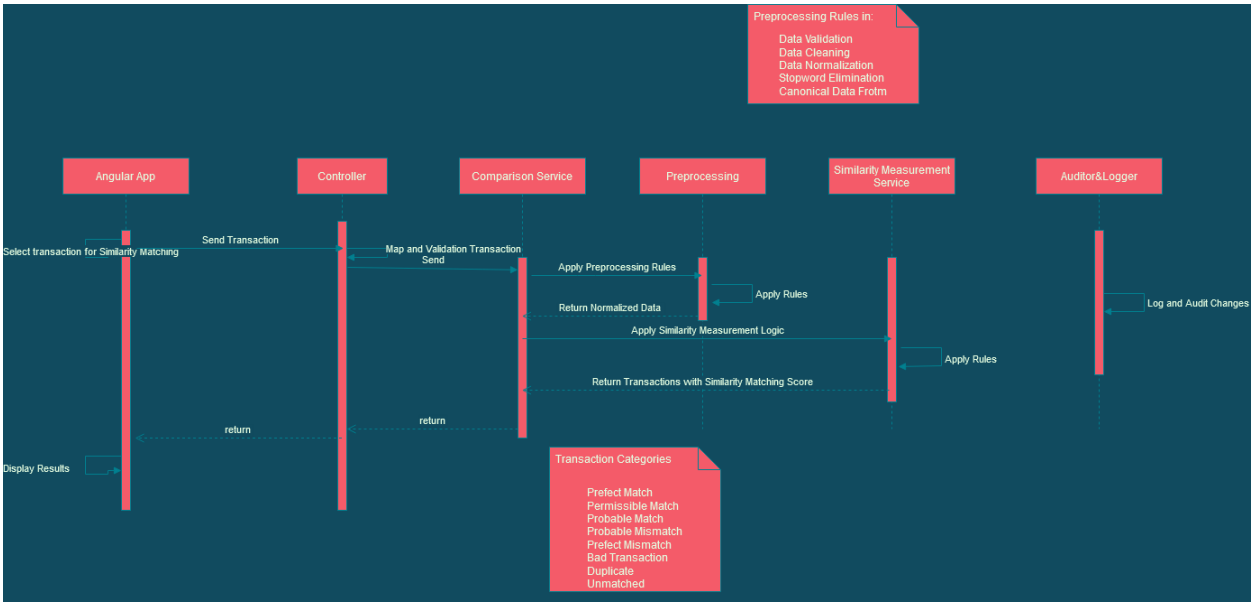
- Responsive Design
- Modular Design (Component Based)
- Lazy Loading Design
- Attractive Design
- Efficient Design
- Forgiving Design
- Clear Design
- Concise Design
- Consistent Design

Algorithm

Transaction Comparison with CSV Files Sequence Diagram:



Transaction Similarity Matching Sequence Diagram:



Texting Matching with Cousine Similarity

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that “measures the cosine of the angle between them.

So Cosine Similarity determines the dot product between the vectors of two documents/sentences to find the angle and cosine of that angle to derive the similarity.

In our case, Cosine Similarity tends to determine how similar two TransactionNarrative are.

Transaction Matching with Fuzzy Matching and Cousine Similarity

Fuzzy matching is a method that provides an improved ability to process word-based matching queries to find matching records or sentences from a database or any other file.

Both Fuzzy Matching and Cousine Similarity are used to Compare Transactions and find the PERFECT_MATCH,PERFECT_MISMATCH,PROBABLE_MATCH,PERMISSIBLE_MATCH, PROBABLE_MISMATCH, DUPLICATE, BAD_TRANSACTION, and UNMATCHED transactions

Stopword Identification and Elimination

Stopwords are the English words which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence.

There is a englishStopWords.txt which contains a list of English Common Stopwords. Stopwords are filtered out before the processing and comparison.

TDD/BDD

I emphasize developing features based on a user story and writing code that provides a solution to real problems. I like to following Behavior Driven Development (BDD) practices. (BDD)

In this specific case, I tracked all software development by repeatedly testing the software against all test cases. This is opposed to software being developed first and test cases created later. (TDD)

Exception Handling

To separate error-handling code from regular code, propagate errors up the call stack, and group and differentiate error types, the following pactices for exception-handling has been considered:

- No Ignoring of Exceptions
- Preference to Specific Exceptions
- Clean Up Resources in a Finally Block or Use a Try-With-Resource Statement
- Throw Exceptions With Descriptive Messages
- Catch the Most Specific Exception First
- No Catching of Throwable
- Wrap the Exception Without Consuming It

Speed of Reconciliation

The comparison operation takes 618 MilliSeconds for the files with following record counts:

- Tutuka File: 306 transactions
- Client File: 306 transactions

Transaction Similar Matching takes 297 MilliSeconds

PC Specifications:

- Memory: 16GM
- CPU: Intel Core i7 2.60GHZ 2.9GHZ

Code Modularity

Modular code is code which is separated into independent modules. The idea is that internal details of individual modules should be hidden behind a public interface, making each module easier to understand, test and refactor independently of others

To provide understandability, continuity, protection, and decomposibility for application, the multi-module maven application with Java modules have been implemented. The project consists of the following modules:

- lib-lang : includes shared functionalities between all other modules
- lib-api: includes utility functionalities for build a robust API
- lib-logger: includes logging utility functionalities
- lib-audit: includes audit utility functionalities
- lib-json: includes JSON management utility functionalities
- reconcile-core: includes infrastructure and transaction comparison implementations
- reconcile-web: Angular application (UI) for transaction comparison

Code Neatness

Writing clean code is important and will help save headaches later. I have been following practices to improve code cleanliness and neatness.

- Using Naming Standard
- Avoid Large Functions
- Use Proper Indentation
- Use Explanatory Comments
- Don’t Overuse Comments

Self-Documenting Code

To wride self-documented code, the following practices have been considered in developmend:

- Codes are properly grouped.
- Standard variable, method, and class naming is followed
- No use of magic numbers
- Codes are moved into functions
- Expressions are replaced with variables
- Class and module interfaces
- Prefer to use named constants
- Adding Sudo-Codes and Logics as Comments

Match Categories

Transaction comparison result is categorized in the following categories:

- PERFECT_MATCH
- PERFECT_MISMATCH
- PROBABLE_MATCH
- PERMISSIBLE_MATCH
- PROBABLE_MISMATCH
- DUPLICATE
- BAD_TRANSACTION
- UNMATCHED

Assumptions

As requirements are fairly vague, the following assumption has been made during development process:

- Similarity matching of one transaction is compared with transactions in another file
- Transactions are not stored in any database
- Comparison and similarity matching has been requested in one session.
- Comma is assumed to be the CSV field delimiter.
- Duplicate transactions might exist.
- Bad transactions might exist.
- If transactionID or transactionDescription is Null or Zero, than it is a bad transaction and will not be matched for similarity measurement with other files.
- Transaction unique key (TUKey) is the composite of transactionID or transactionDescription. transactionID is unique but it is the same in the case of transaction reversal.

- If TUKey of a transaction is present in the another file, then it is considered as a matched transaction.
- If TUKey of a transaction is not present in the another file, then it is considered as a unmatched transaction.
- Only if TUKey matches, it qualifies for a matching.
- transactionNarrative field is an address field and will have to be cousine similarity and fuzzy matching logics. As almost NONE of the values from the files [TutukaMarkoffFile20140113.csv & ClientMarkoffFile20140113.csv] resolved to a valid location, geo-location fuzzy matching was out of the equation.
- transactionDate values are at the same time zone.
- For those transactions that have different transactionID's but other transactions do match mostly, are still considered as UNMATCHED as transactionID is an unique ID and a null value of which is considered a BAD_TRANSACTION. Reason for this is transactionID should be a generated value for every transaction and cannot be the same for two different transactions.
- If possible matches of a unmatched transaction is requested, then calculate similarity score and list matched transactions with more than 80% similarity.

User Manual

Module UI Access

(1) Open http://localhost:4200

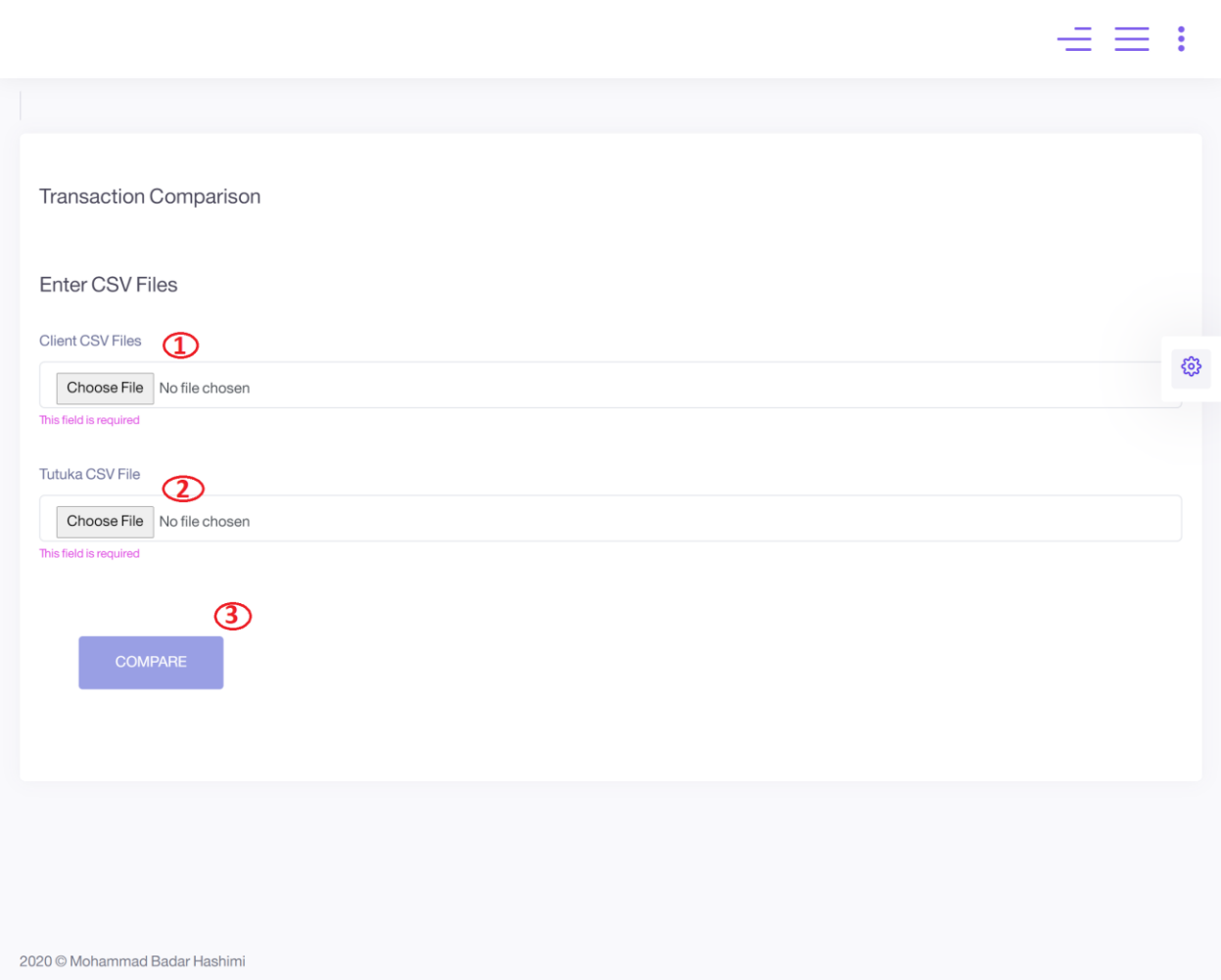
(2) Go to Transaction Comparison > Transaction Comparison

File Upload:

(1) Select Client CSV File

(2) Select Tutuka CSV File

(3) Click on COMPARE Button

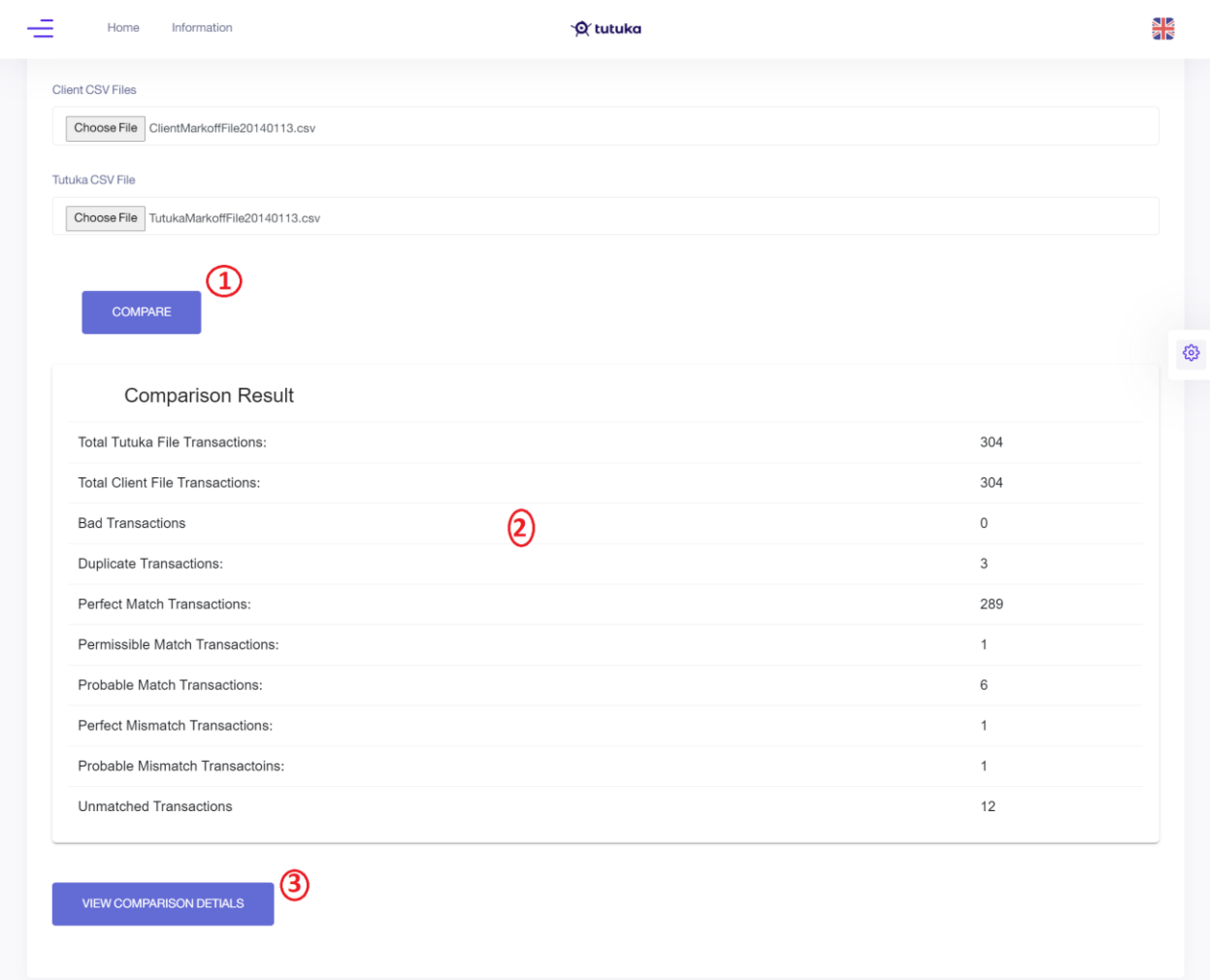


Comparison Result:

(1) COMPARE Button will active once you selected files. Click to View the Result

(2) Shows the Result of Comparison

(3) Click on VIEW_COMPARISON_DETAILS Button To See Comparison Report in a Tabular Format



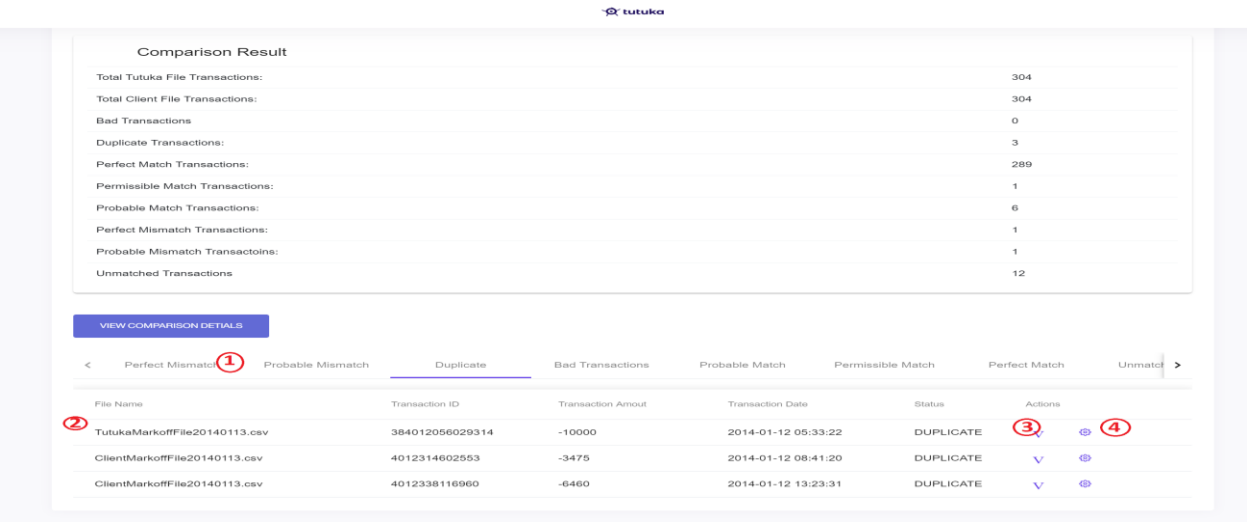
Comparison Report in Tabular Format:

(1) Each tab Shows a table of categorized transactions like mismatch, probable_mismatch, perfect_match, and so on. Click to View a List of Transations in that Tab

(2) Shows a Transaction

(3) Click on View Icon To View Details of that Transaction

(4) Click on Settings Icon To View Similar Matching Transactions of that Transaction

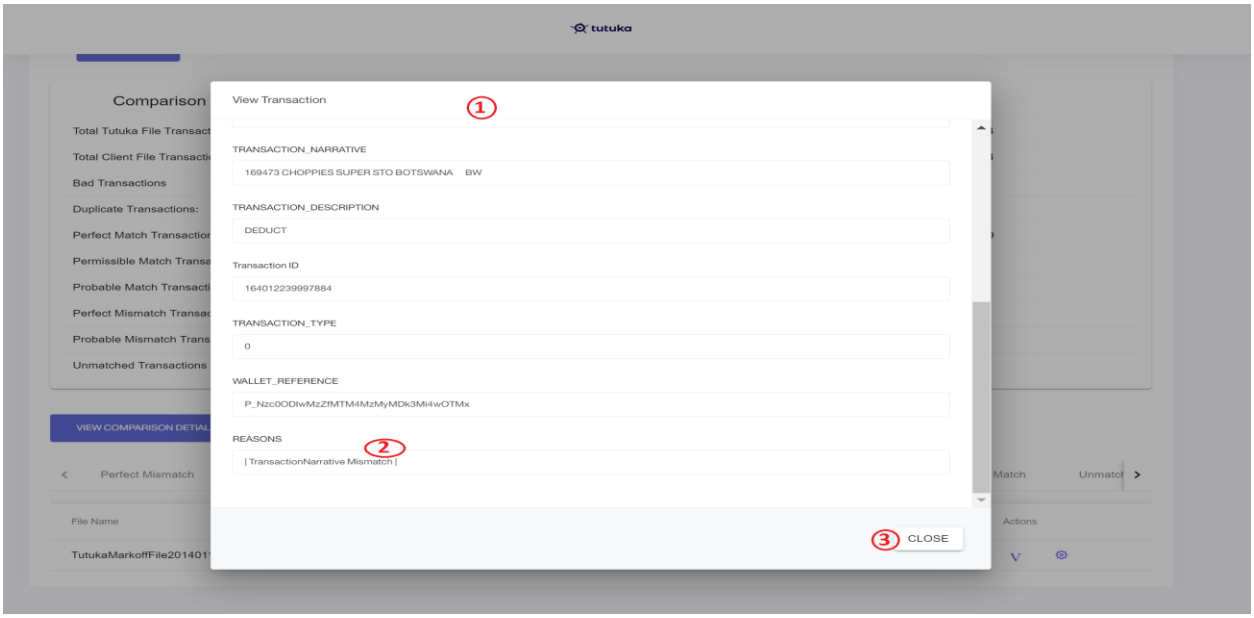


View Transaction:

(1) Shows All Details of the Selected Transaction

(2) Reason of this Transaction is categorized in this transaction category

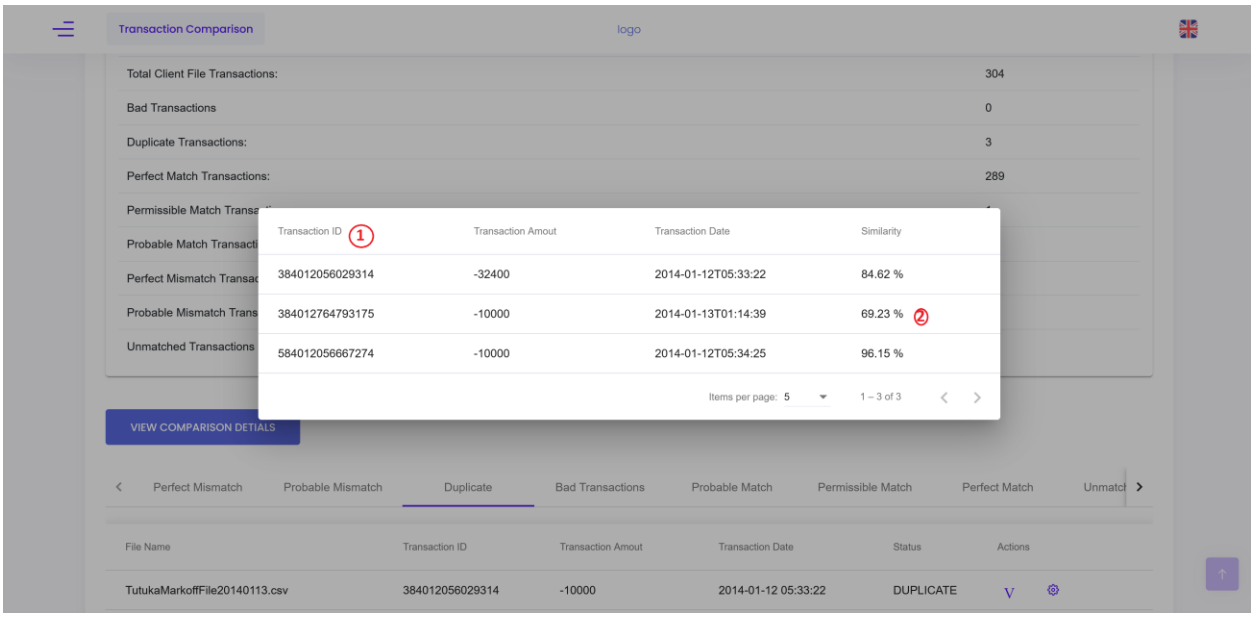
(3) Click on CLOSE Button To Close the Model



Similar Matching Transaction:

(1) Shows List of Similar Matching Transactions

(2) Percentage of Transactions Similarity


Transaction Comparison
logo
Total Client File Transactions:
304
Bad Transactions
0
Duplicate Transactions:
3
Perfect Match Transactions:
289
Permissible Match Transa
Probable Match Transacti
Perfect Mismatch Transac
Probable Mismatch Trans
Unmatched Transactions
Transaction ID
1
Transaction Amout
Transaction Date
Similarity
384012056029314
-32400
2014-01-12T05:33:22
84.62 %
384012764793175
-10000
2014-01-13T01:14:39
69.23 %
2
584012056667274
-10000
2014-01-12T05:34:25
96.15 %
Items per page: 5
1 – 3 of 3
VIEW COMPARISON DETIALS
Perfect Mismatch
Probable Mismatch
Duplicate
Bad Transactions
Probable Match
Permissible Match
Perfect Match
Unmatcl
File Name
Transaction ID
Transaction Amout
Transaction Date
Status
Actions
TutukaMarkoffFile20140113.csv
384012056029314
-10000
2014-01-12 05:33:22
DUPLICATE