MTEch KE-4102 (ISBA) Project report

# “Catch me…if you can”

# 1.0 eXECUTIVe SUMMARY

**teAM MEMBERS**

Bharat Nagaraju

Muthuraman Ramasamy

Laxman Singh

Gopa Sen

Louis Tham

Edwin Tam Wei Ching

MASTER OF TECHNOLOGY IN   
KNOWLEDGE ENGINEERING

BATCH KE-30(2018)

Is your money as safe as the banks say it is?

We’re a little skeptical.

In 2015, Financial Fraud Action UK reported 26% increase in banking fraud activity – resulting in a loss of £755m from consumers and financial institutions1. In 2016, USD 181Bn was lost to conmen and fraudsters, with Wire Fraud taking most of the cake at USD 50Bn (27.6%)2.

Often account holders find out far too late.

So what can we do if our retirement, slush, save-the-world, business, or other funds aren’t as safe as we thought?

**Find out faster. Stop the outflow quicker. Catch the baddies, of course.**

“Catch me… if you can.” is a recommendation/identification system that simulates how compliance, Money Laundering and fraud detection works in banks.

It red flags suspicious banking transactions made by (or in place of) customers by using a combination of transaction frequency, value, and type, alongside customer risk, which it matches against a list of known fraud scenarios.

The results are displayed for compliance officers to take action – be it with the bank or escalate to regulatory bodies.

So “Catch me… if you can.”

1. *Banking scams push up UK financial fraud 'by more than 25%'. Src:* [*The Guardian*](https://www.theguardian.com/money/2016/mar/17/banking-scams-uk-financial-fraud)
2. *Top 10 Fraud Types Based on Losses. Src:* [*Frank on Fraud*](http://frankonfraud.com/fraud-reporting/top-10-fraud-losses-for-2016-and-where-they-are-headed-now/)

# 2.0 problem description

It started with curiosity… and three questions: “Just how do we know if bank transactions are fraudulent? How do we know if a bank customer is a money launderer? Or if they’re the victims?”

Just how do banks do it? That’s what we set out to find and do.

**2.1 PROJECT OBJECTIVE**

Build an identification system to find customers at risk of fraud. This system does so by pattern-matching transaction and customer data against identified fraud scenarios.

This system should do three things:

1. Take in a list of transactions and pre-loaded customer database
2. Match transactions against known fraud scenarios and assign risks (aka likelihood) of fraud
3. Tell the compliance officer which customers are at risk for fraud and contributing transactions

**How it works**

Identifying fraud is done by matching transaction patterns against common fraud transaction patterns (aka scenarios) or highlighting transaction patterns that break the norm. The system’s sensitivity to fraud is determined by user-defined parameters (such as period of time, maximum transaction values, maximum number of transactions).

These scenarios have multiple possible permutations. They could be any combination of transaction values, number of transactions, transaction denominations, time periods, recipients and senders, geographies, or transaction methods.

For example, a series of small credit transactions that amount to a large value over a short period of time (e.g. 1 day) could be considered fraud; or 1 big cheque transaction when small ATMs transactions are the norm might be fraud. It really depends on the system sensitivity as mentioned earlier.

As you can imagine, there are quite a number of fraud scenarios (confirmed by SME interview and research). To make it manageable, we focused on identifying 6 fraud scenarios and only for cash transactions.

**Benefits of this system**

1. Early-warning system: if we could identify possible fraud early, we can raise the alarm before it’s too late to take action.
2. Helps compliance officers prioritise fraud cases by ascertaining the likelihood that a customer might be a victim.
3. Churns and matches large amounts of transaction data
4. Designed to “plug” into existing fraud identification systems

# 3.0 KNowledge modeling

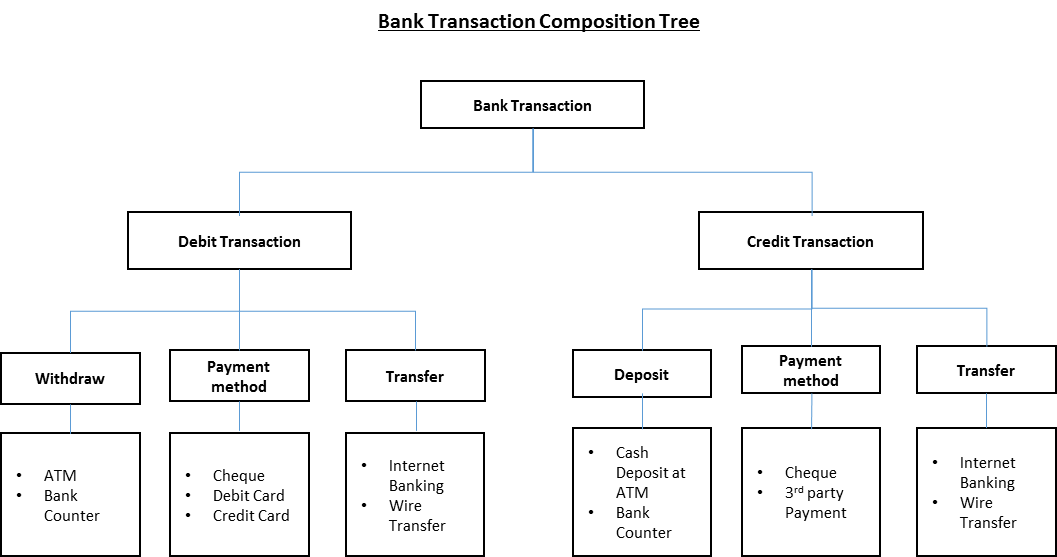
Fraud detection depends on knowing the customer type and his transaction behaviours. Hence we generated 3 trees:

1. Bank Transaction Composition Tree: What makes up Bank Transactions?
2. Customer (Bank Account) Composition Tree: What makes up a customer?
3. Fraud Detection Attribute Tree: What types of attributes and values in the transaction do experts look at when identifying fraud?
4. Cash Transaction Fraud Scenario Decision Tree: How do experts decide what type of fraud has happened?

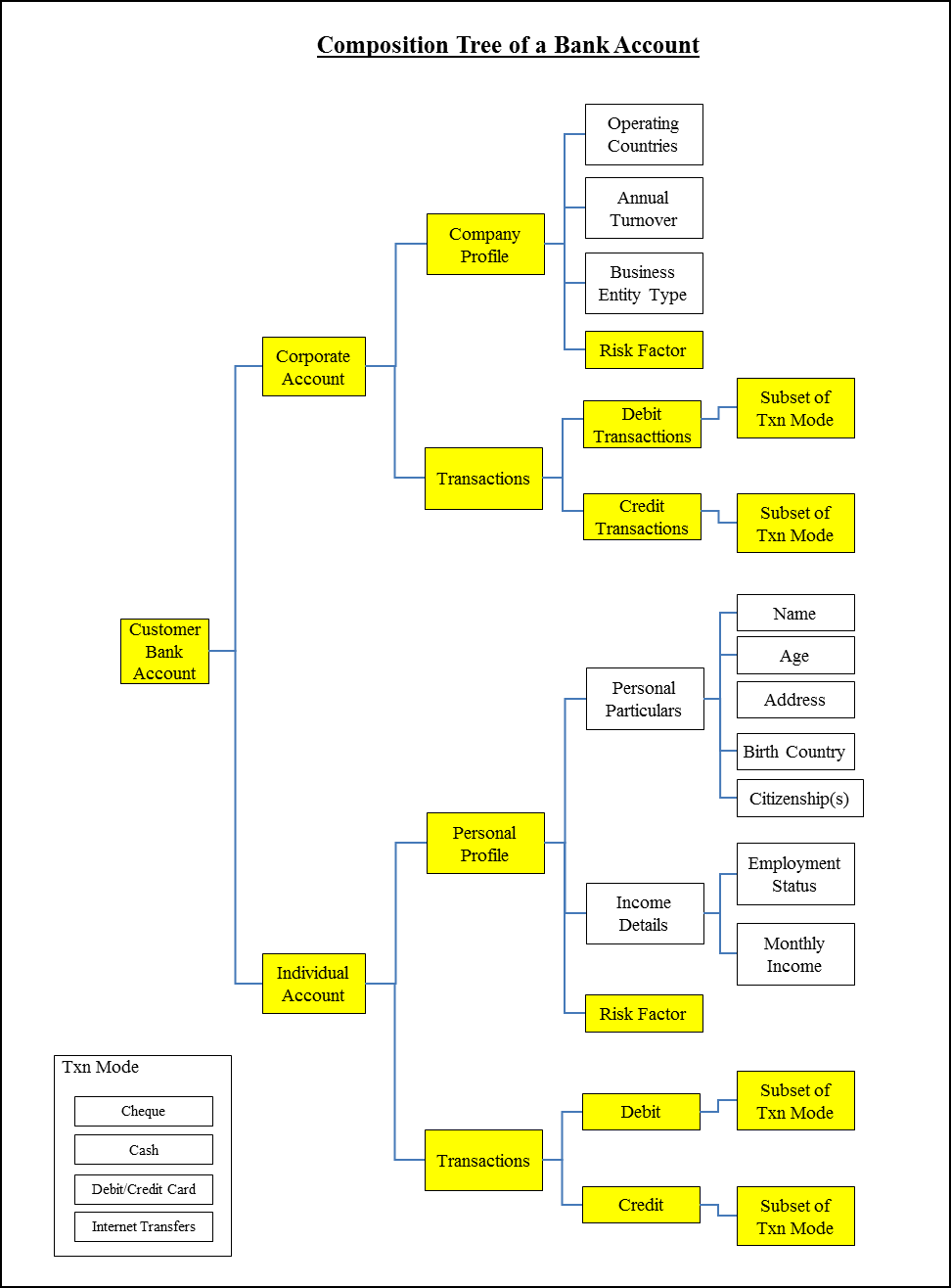
Trees 1 & 2 inform our dataset structure to upload. Tree 3 & 4 informs the rules and parameters of the system. We have highlighted the knowledge that we will use in the system.

Our SME (a Compliance Programmer; see *SME Interview – Transcript.docx* & *Interviews* videos) is our primary source of knowledge. However, we have supplemented knowledge extracted from our SME with common frauds (see *References*).

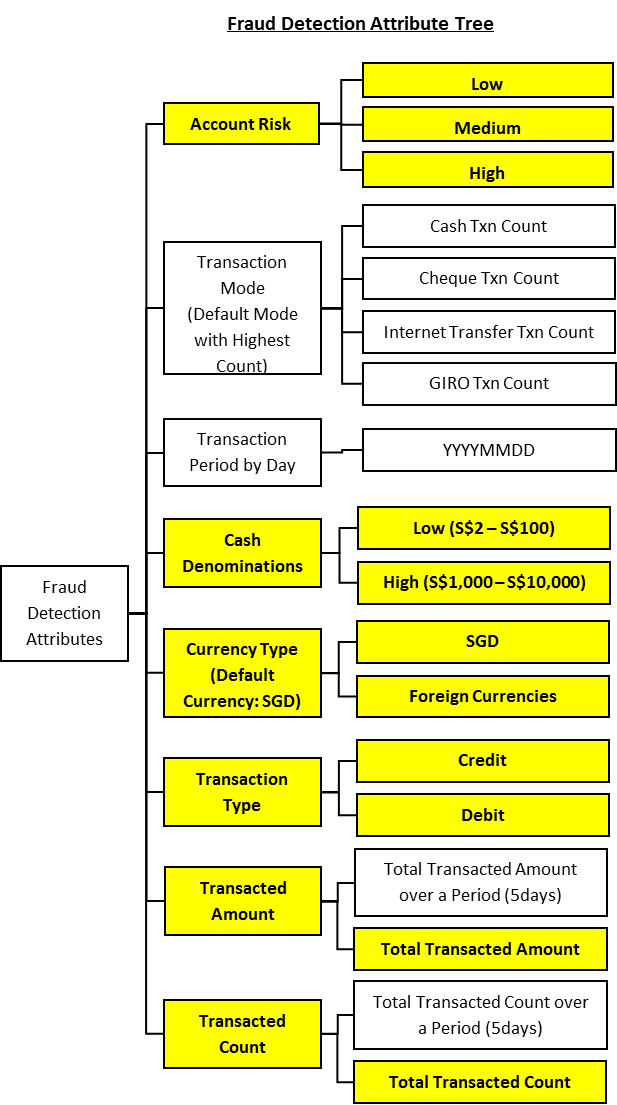
**Tree 1**



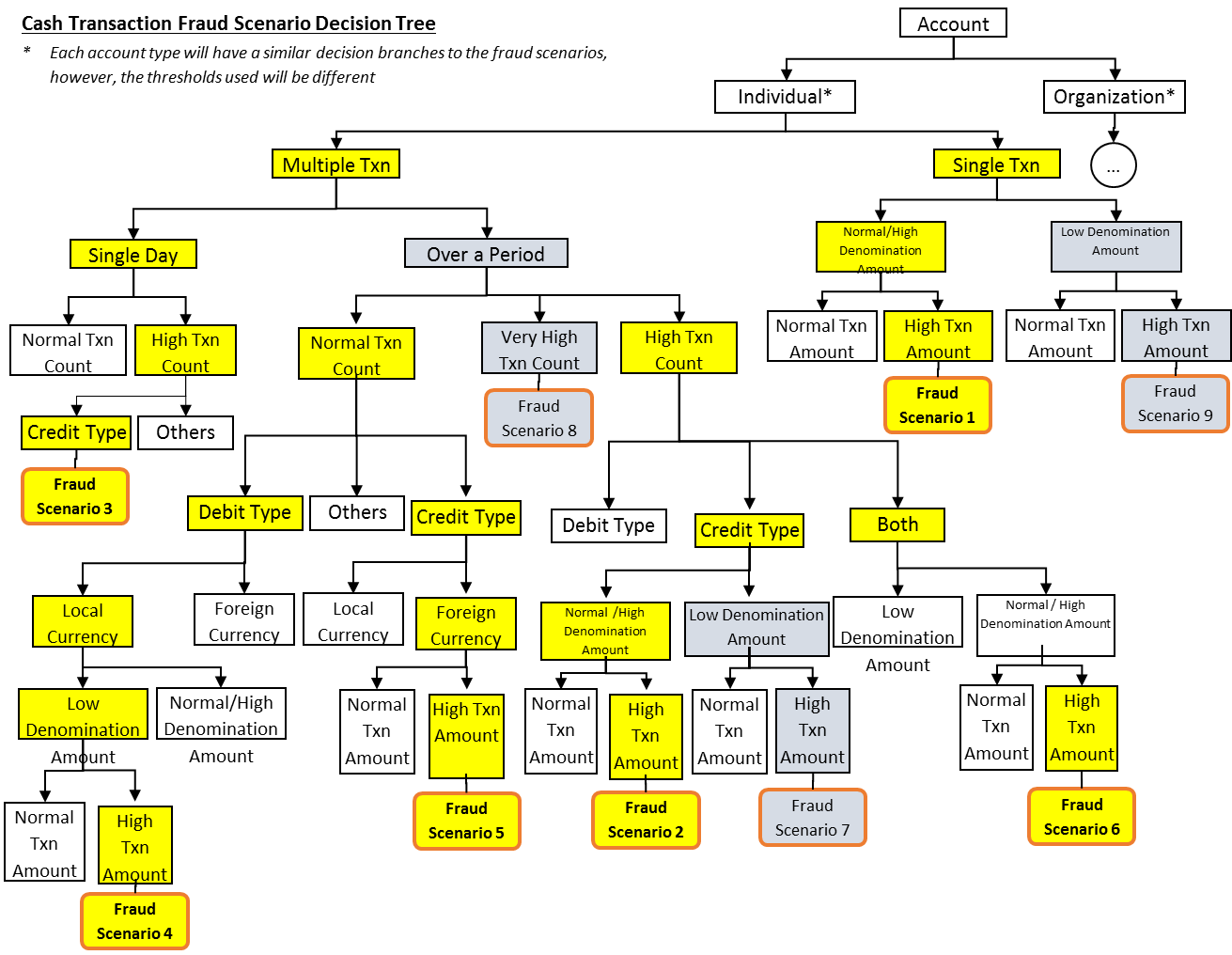
**Tree 2**

****

**Tree 3**

****

**Tree 4**

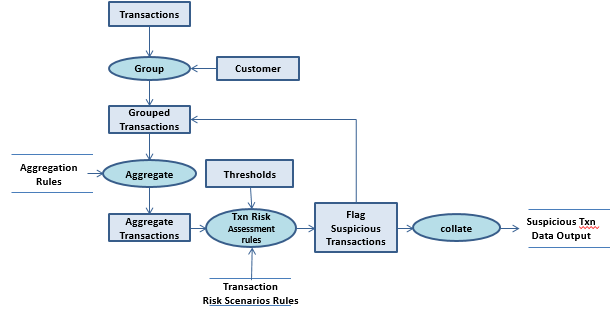
****

# 4.0 solution outline

## 4.1 System Scope

Six fraud scenarios focusing on Cash Transactions (e.g. debit or credit via credit card, cash, ATMs, tellers; does not include SWIFT transfers). Users can adjust parameters to affect system sensitivity.

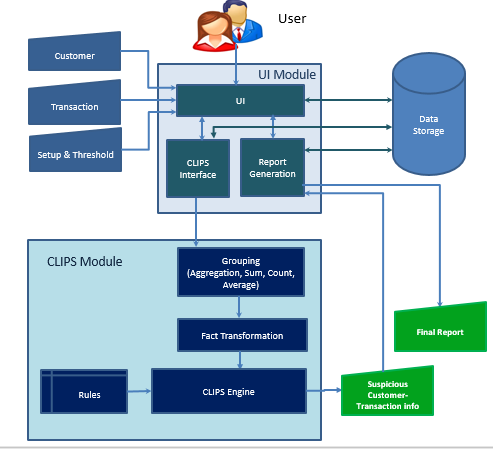
## INFERENCE DIAGRAM





## High Level System Architecture

The System Architecture contains two main components - GUI and CLIPS modules. User access system via GUI to upload Customer, Transaction and Threshold files. GUI will stores the data in database and grouping transactions such as aggregation, Sum, Count and average to facts before passing to CLIPS.



CLIPS will process facts based transaction processing rules and threshold to identify suspicious transaction. If transaction is suspicious then it populates suspicious flag with Customer and fact id to generate the output. We use .Net as GUI to interface with Clips engine as well as MySQL database.

## System Features

### **Structures & FEATURES**

Data structures include Facts such as Customer & Transactions (*see facts\_threshold.clp*). All scenarios are coded as Rules (*see rule\_report\_gene.clp*). We also use a set of aggregation rules *(see sum\_amt\_cnt.clp)* to prepare Transaction and Customer.

**GUI features and functions**

1. Filter out the necessary Accounts & Customers from threshold.
2. Check Customers’ risk for appropriate fraud
3. Run Account’s transactions through scenarios
4. Return Account + Customer + the Frauds that have happened

### **Scenarios**

The workhorse of our fraud detection system.

1. **Multiple large amount cash transaction, when user’s default mode is cheque transaction**

SPILT transactions into debit & credit

FOR each debit & credit categories

IF transactions source != cheque & value is above threshold, THEN COUNT transactions

If Transaction Count > Transaction\_Threshold Count, then fraud of Scenario 1 has happened

1. **Multiple deposits made to an account using a number of different ways (e.g. cash deposit, money transfer). These add up to a large amount over a short period of time**

IF transactions are within the period of time (defined as 1 day) & is DEPOSIT, THEN COUNT the transactions & SUM the Amount

IF SUM\_Amount > value threshold, THEN fraud of scenario 2 has happened

1. **Multiple credits to the account on the same day**

IF transactions are multiple CREDIT & on same day, THEN COUNT Transactions

IF transactions\_count > count\_threshold & total\_credit\_value > max\_credit\_threshold THEN fraud of scenario 3 has happened

1. **Small denominations debited to an account. Resulting in a large total amount**

IF transactions denomination is LOW & transaction value>threshold, THEN fraud of scenario 4 has happened

1. **Large amounts of foreign currency money transferred to an account**

IF transactions are CREDIT & currency != SGD & value > threshold, THEN fraud of scenario 5 has happened

1. **Multiple credits and debits to an account over a short period of time**

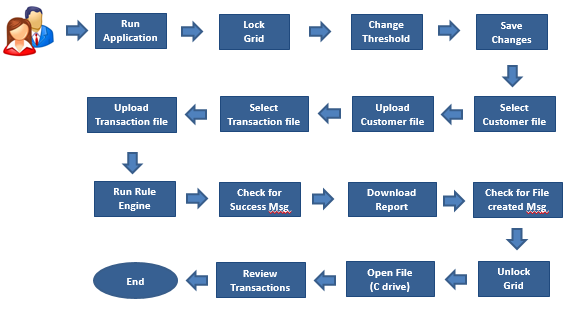
IF transactions are within the period of time (defined as 1 day) & is DEPOSIT, THEN COUNT the Debit\_Transactions & SUM the Amount

IF transactions are within the period of time (defined 1 day) & is CREDIT, THEN COUNT the Credit\_Transactions & SUM the Amount

IF Debit\_Transactions & Credit\_Transactions are > threshold individually, THEN fraud of scenario 6 has happened

* + 1. **User Flow**

The following diagram illustrates the user flow of our system.



## AssumptionS

1. Every customer has 1 bank account which consists of 1…N transactions
2. Users of this system only worry about customers and cash transactions. They are not worried about external entities (e.g. other banks) or transaction types (e.g. credit card, wire)
3. System will test transactions against the default currency (SGD)
4. Personal particulars and company profile data are not relevant in these scenarios…for now

## System Limitation

The system is designed for six fraud scenarios with fairly fixed thresholds. After running data through the engine, we then store the results in an excel file for the compliance officers to use.

# 5.0 Conclusion & refrences

**Learnings**

Knowledge encapsulating money laundering/fraud transactions is vast. We found it quite hard to extract, model, and account for different relationships between scenarios. As such, we focused on a few common scenarios.

Also turning customer and transactions into facts isn’t as straightforward as we thought. We had to use multi-templates to represent customers and transactions *(See Users Manual).*

**Improvements**

While fraud scenarios can be discrete and standalone, it is the combination of these scenarios that make it representative of real-life scenarios. But what scenarios go together? It might be useful to run machine learning algorithms to find common patterns and combinations across fraud scenarios.

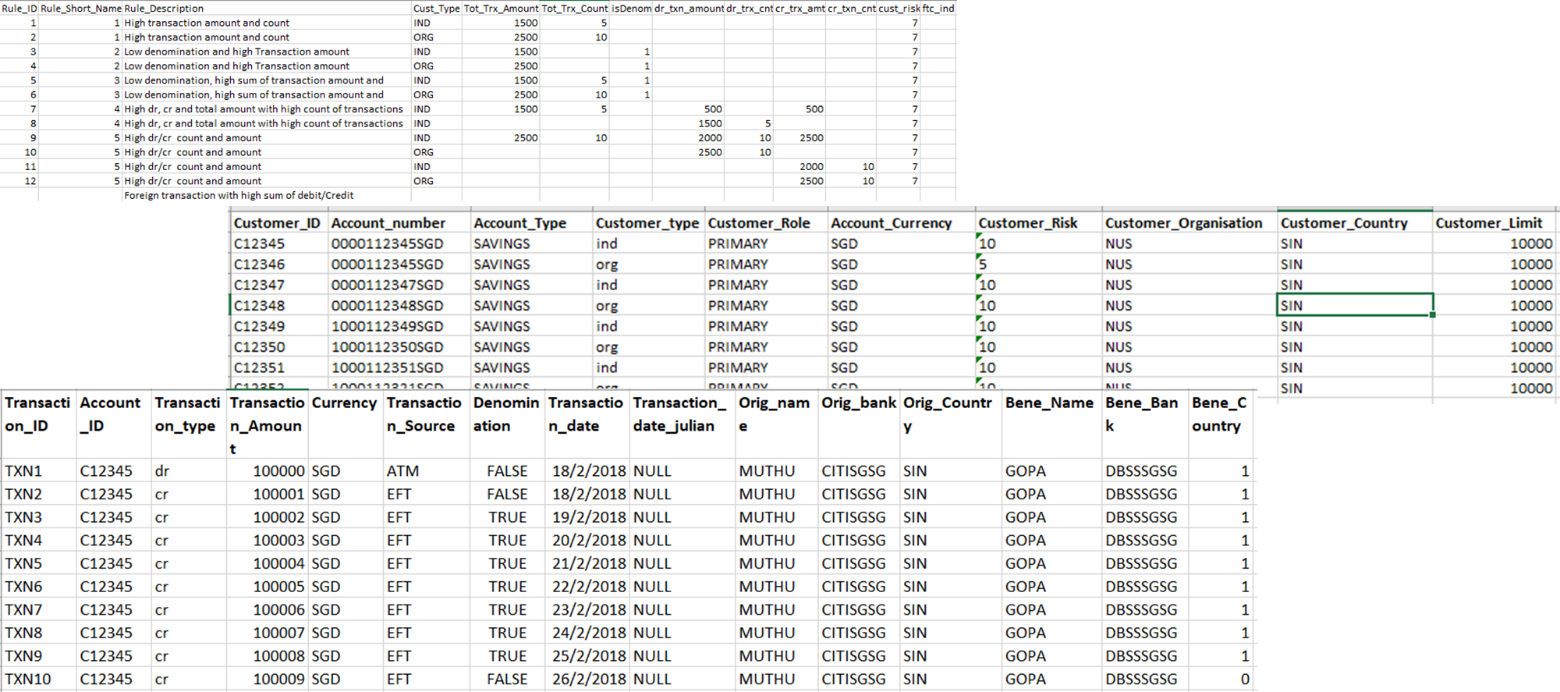
Alternatively, we could use machine learning algos (e.g. decision trees) on the transaction datasets to identify fraud. This should make it far more scalable than current rules-based systems.

**References**

1. Oracle Financial Services Anti Money Laundering And Fraud Analytics (Oracle)
2. [Fraud and Anomaly Detection using Oracle Advanced Analytics Part 1 Concepts](https://www.youtube.com/watch?v=O78IYjPLIPY&feature=youtu.be)
3. [Rule Based System (Payer rules integrated and simplified)](https://www.youtube.com/watch?v=WXA7k9iRG14)
4. [Avoiding suspicious activity reports in cash transactions](file:///C:\Users\User\Dropbox%20(Personal)\KE%20Project\Project%20Docs\1.%09http:\onewealthyteam.com\avoiding-suspicious-activity-reports-in-cash-transactions)
5. [Types of Suspicious Activities or Transactions](http://fiubelize.org/types-of-suspicious-activities-or-transactions/)
6. Best Practices for Anti Money Laundering (AML): System Selection and Implementation (Oracle)
7. [Financial crime in funds transfer systems](https://www.pwc.com/sg/en/publications/assets/financial-crime-funds-transfer-systems.pdf)
8. [Payment Fraud in Digital Currency | DataEngConf SF '17](https://www.youtube.com/watch?v=vNMELg85XxA)

# APPENDIX A: sample input & system output

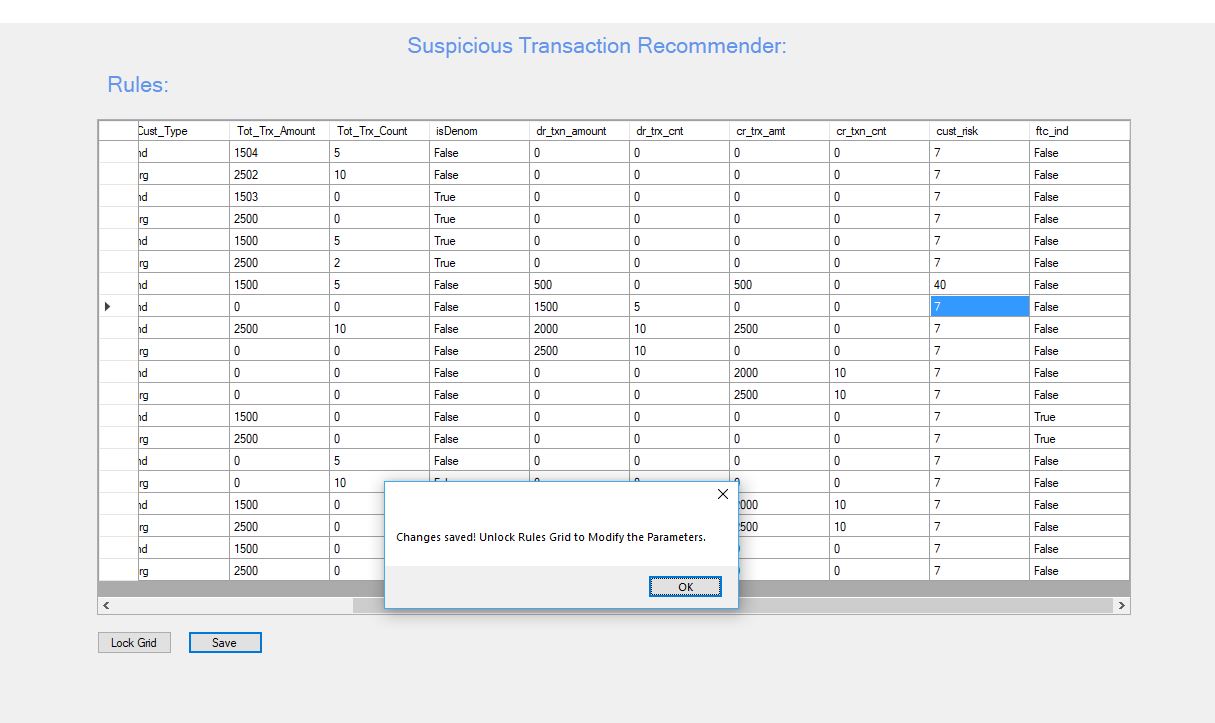
**Use Case 1:** Customer with Normal Transactions which does not classify under any of the 6 fraud detection scenarios

****

**Use Case 2:** Customer with Fraudulent Transactions who is identified under one of the 6 fraud detection scenarios



**Use Case 3:** How change of threshold value impacts classification of transaction as fraudulent to non-fraudulent



# APPENDIX B: user’s manual

**Installation**

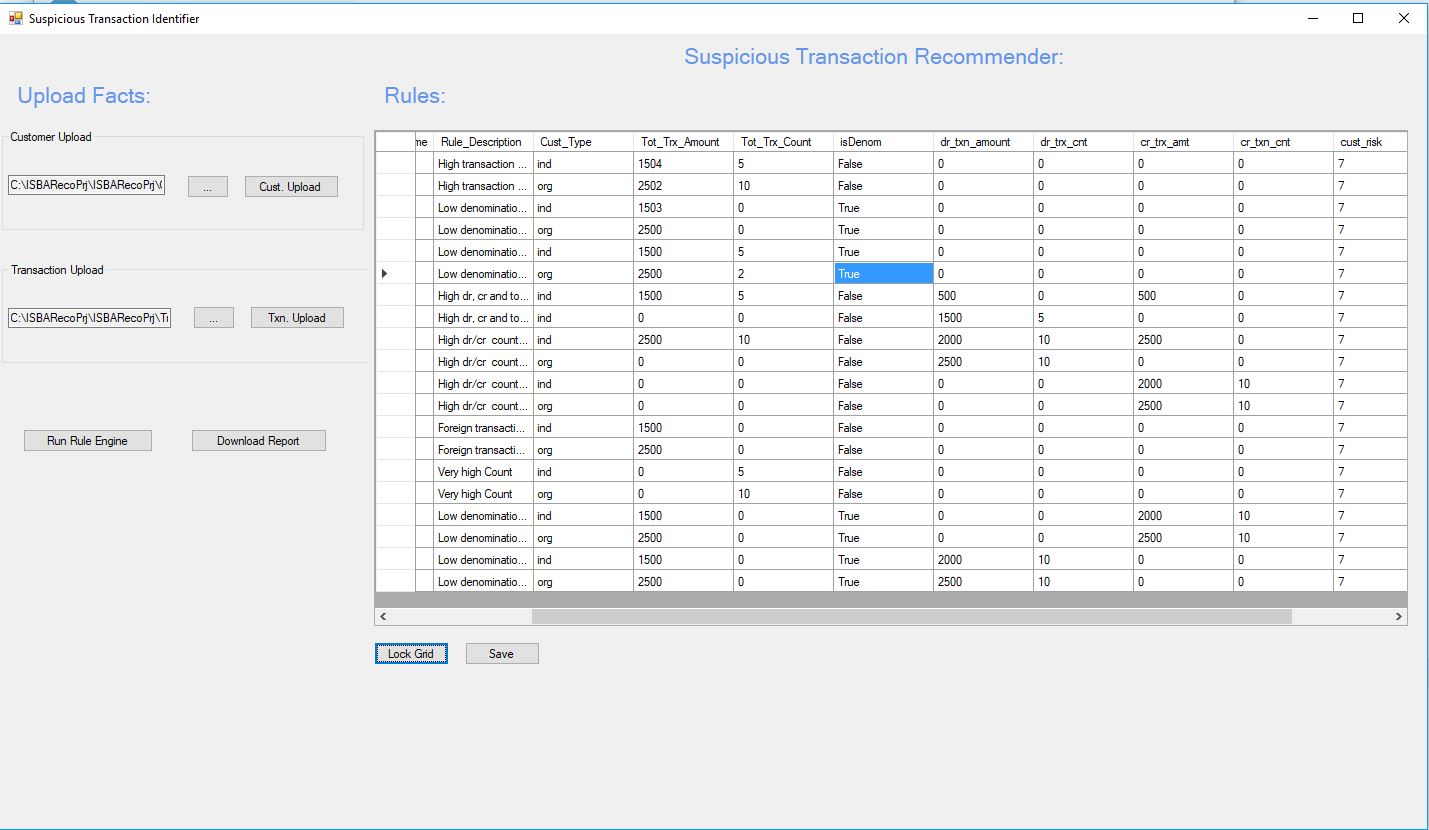
1. Install Sql Server Express latest version
2. Change the server name in the data source = "XXX" in app.config file in the release directory.
3. Data source will be the server name set up in the local directory

##connectionString="Data Source=VINAYAK\SQLEXPRESS;Initial Catalog=ISBA;Integrated Security=True;Pooling=False"

1. Install .net latest version
2. Install SQL server management studio and visual studio community edition if required to run the code
3. Install the MS Excel connectivity package - shared in the same directory

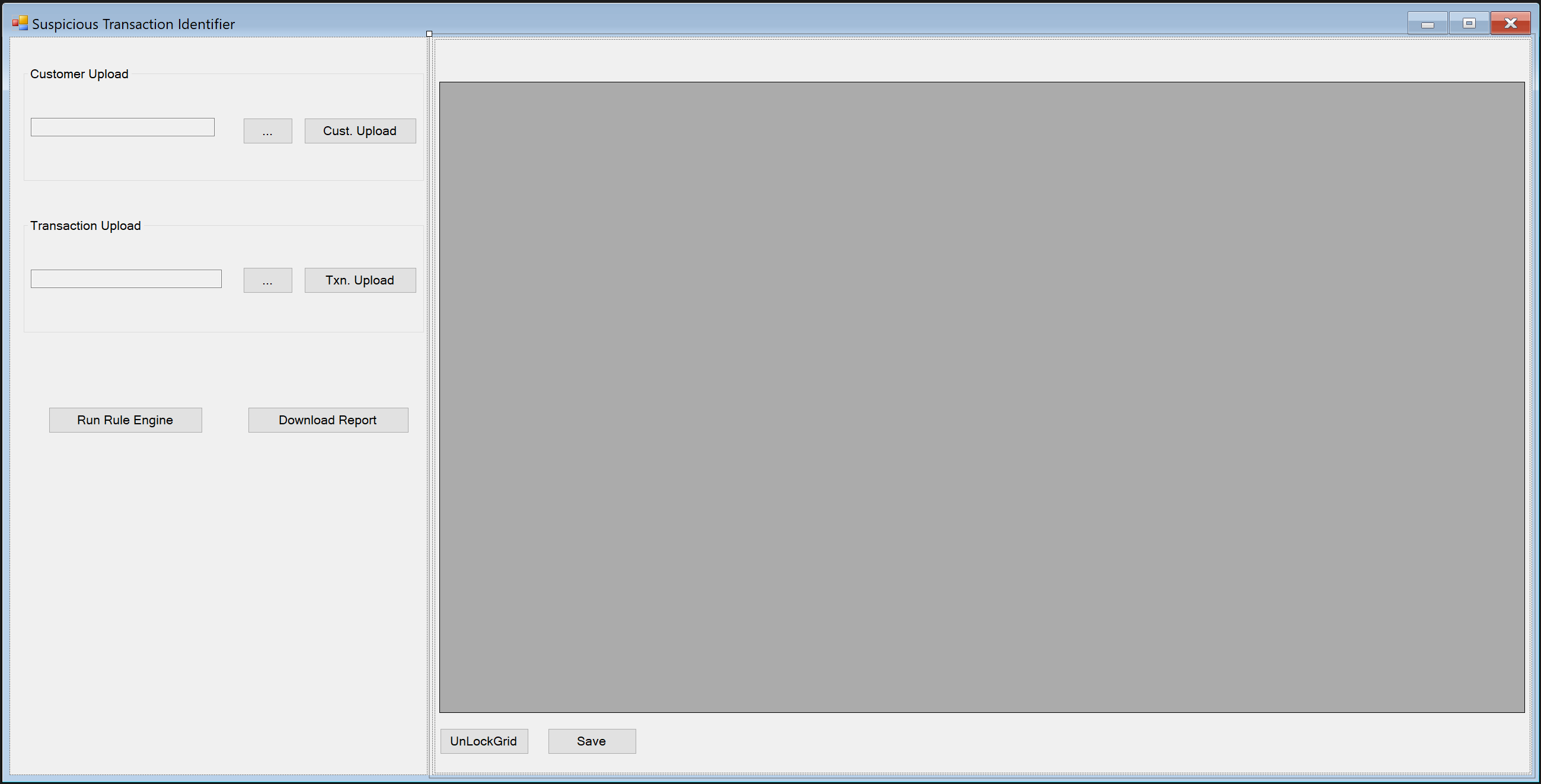
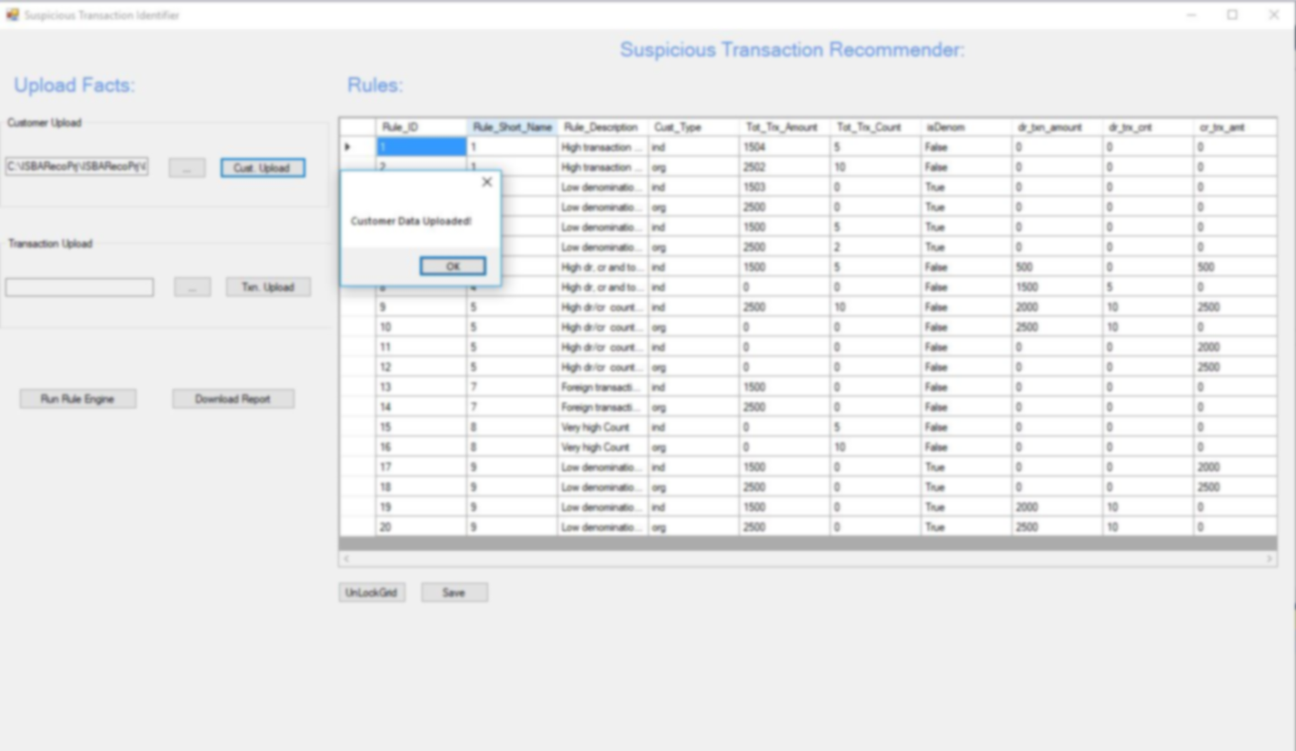
**Running “Catch me… if you can” User Manual**

## GUI Overview



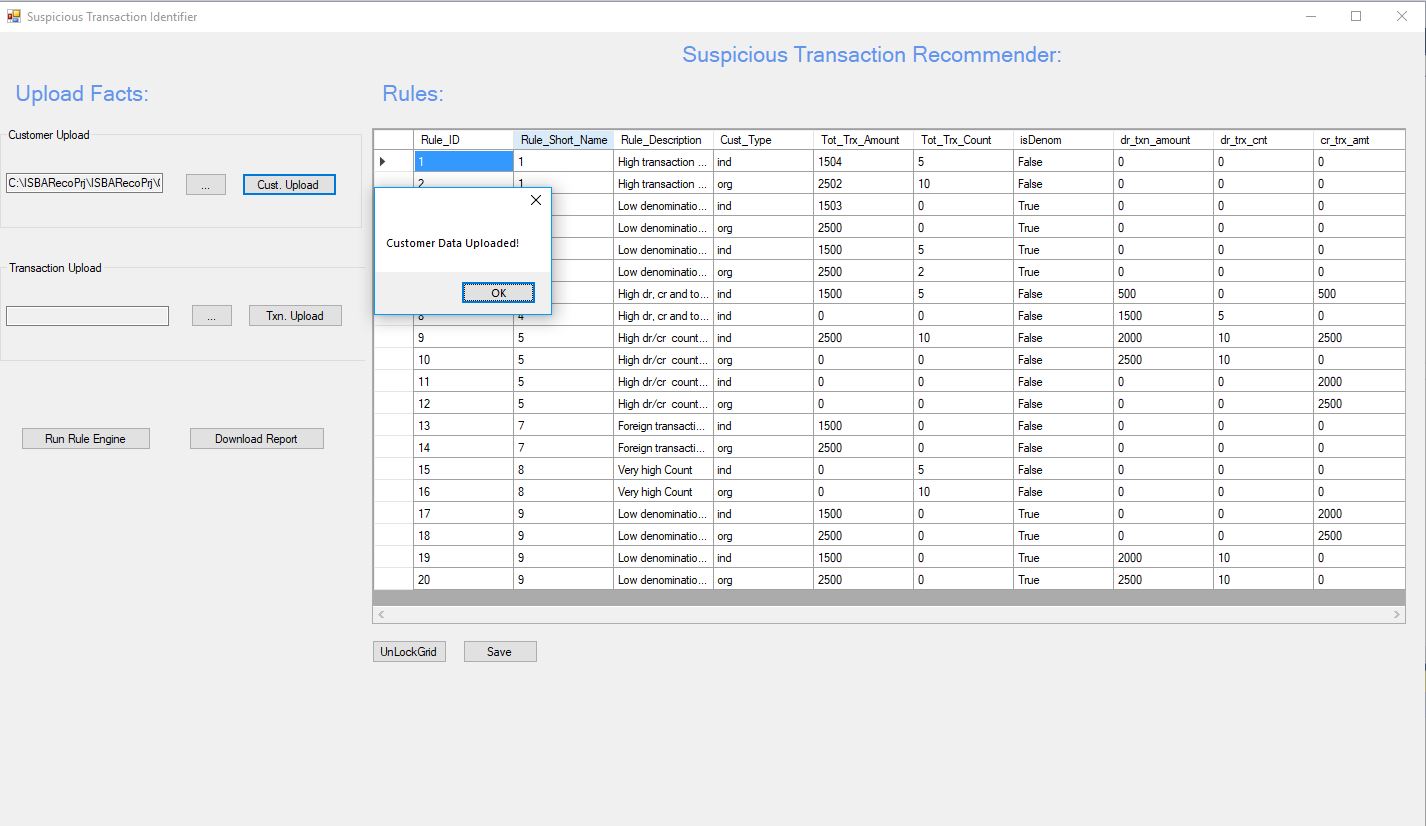
## Step 1: Upload your Customer table

* Click on button A and select the relevant customer table to be loaded
* Once done, click on the button B to upload into the database
* A popup window will indicate a successful upload



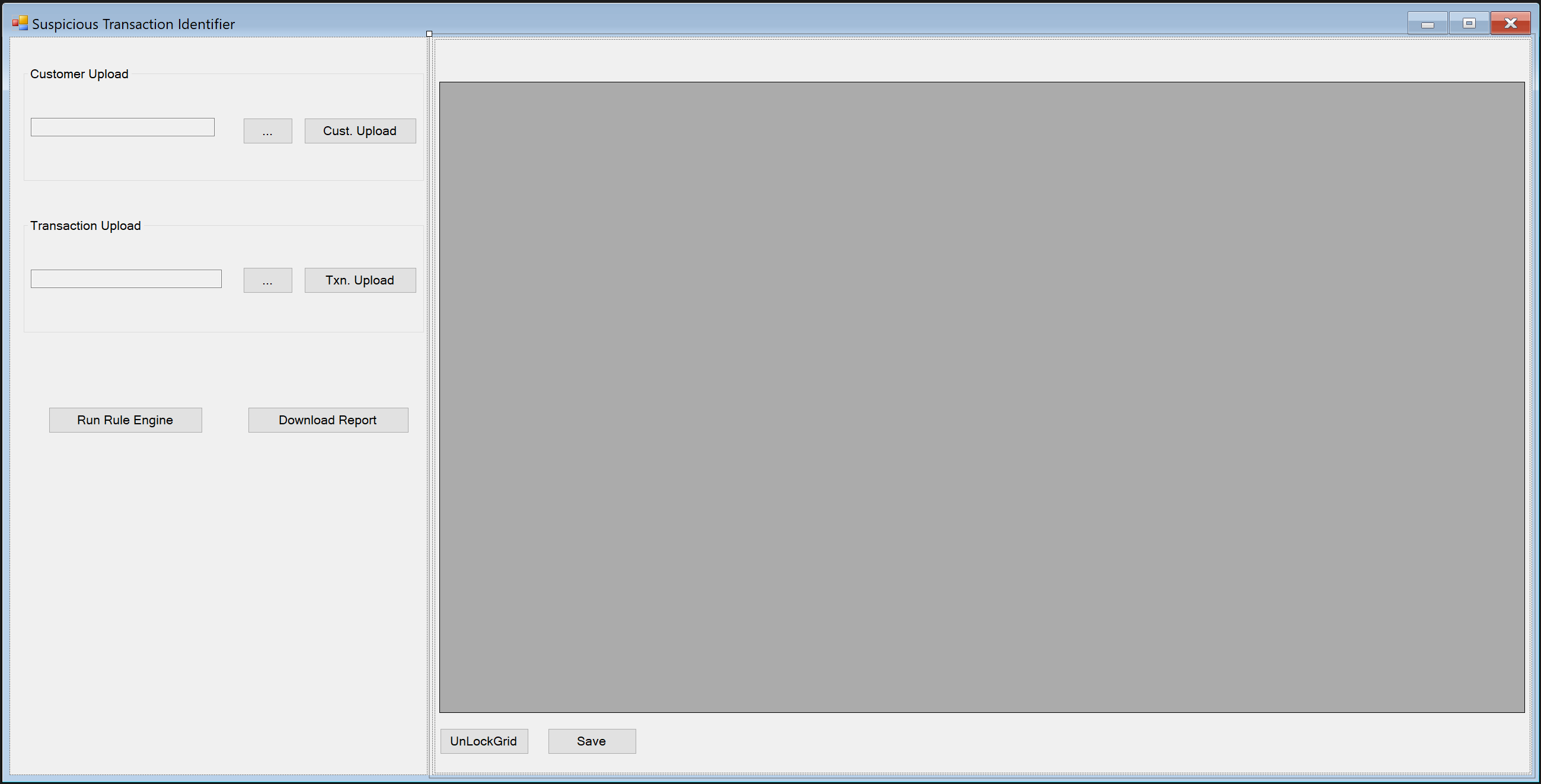
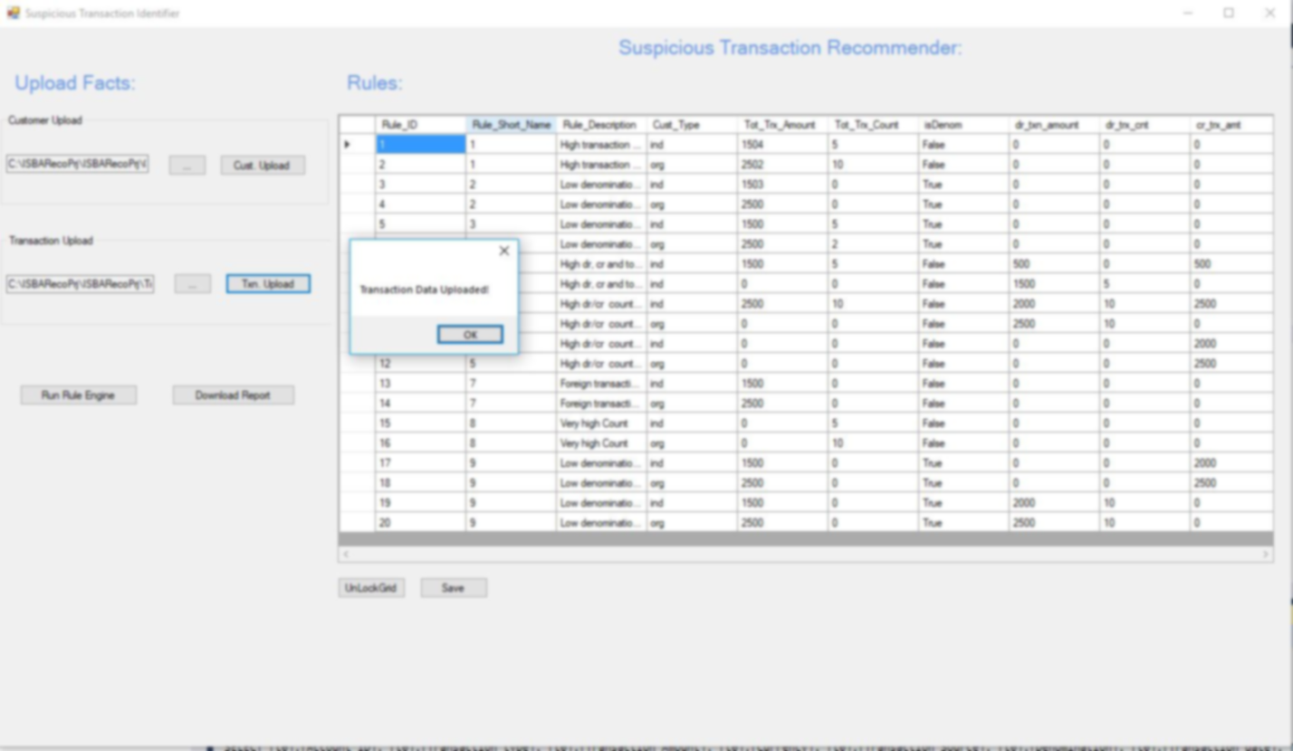
A

B



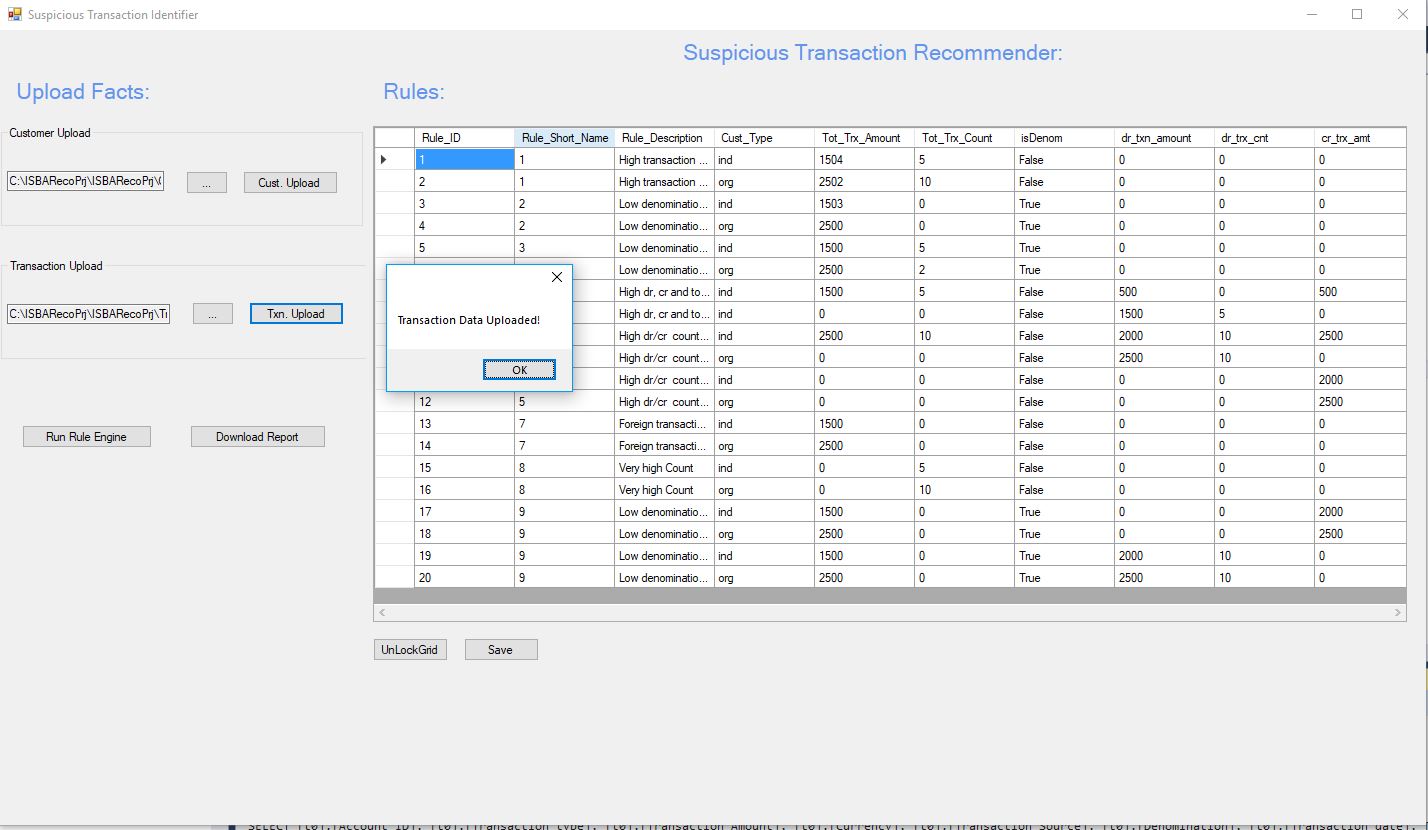
## Step 2: Upload your Transaction Table

* Click on button A and select the relevant transaction table to be loaded
* Once done, click on the button B to upload into the database
* A popup window will indicate a successful upload



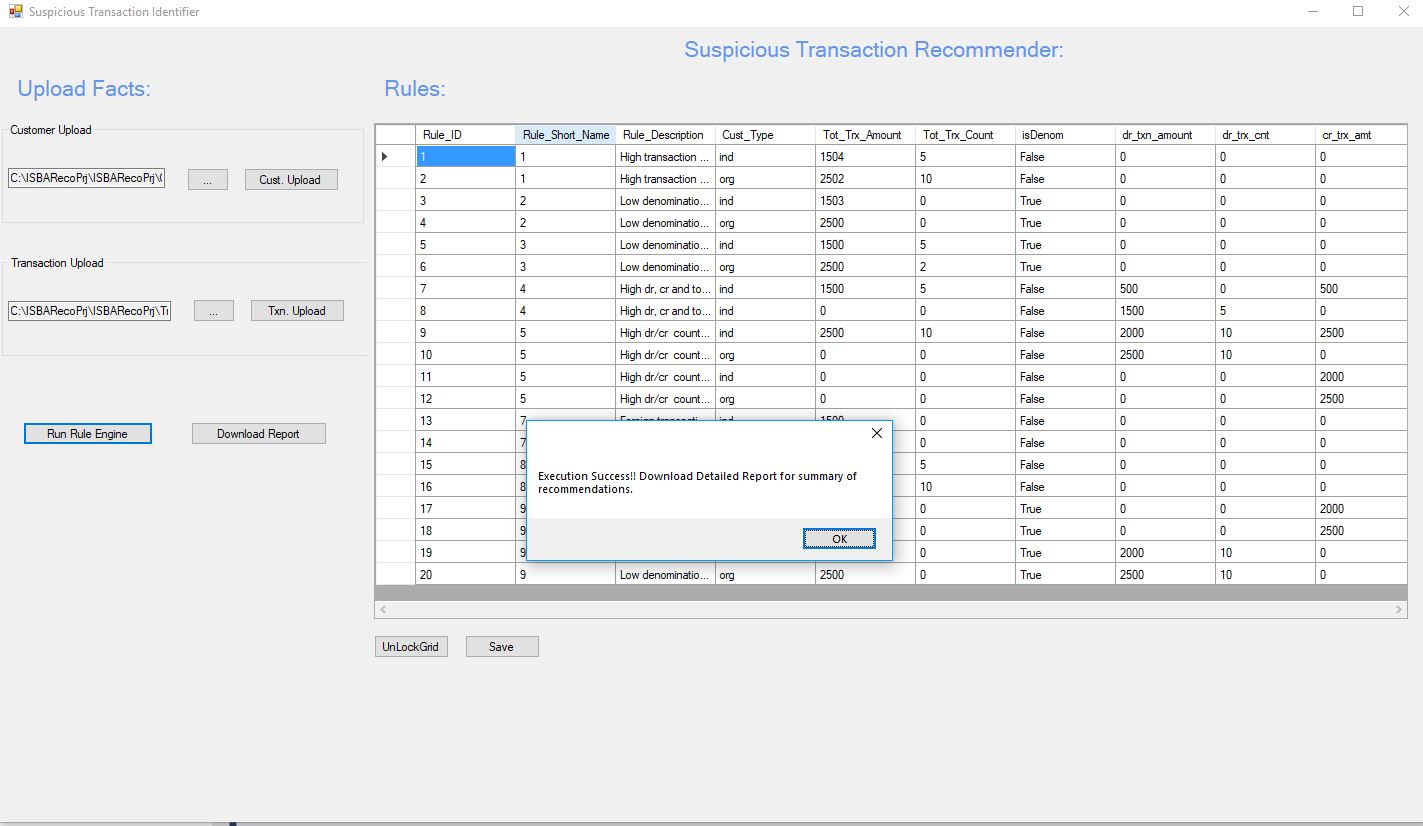
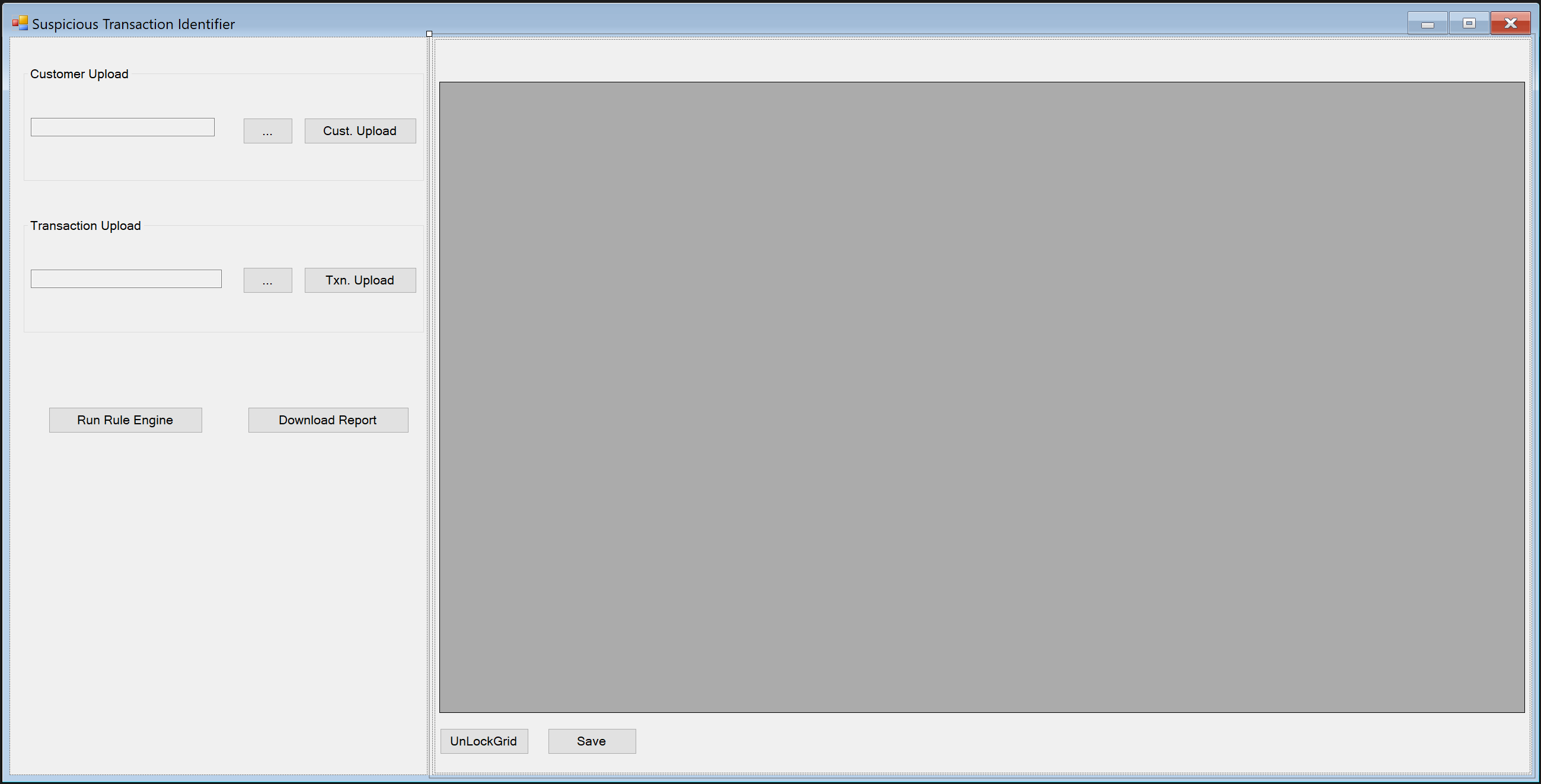
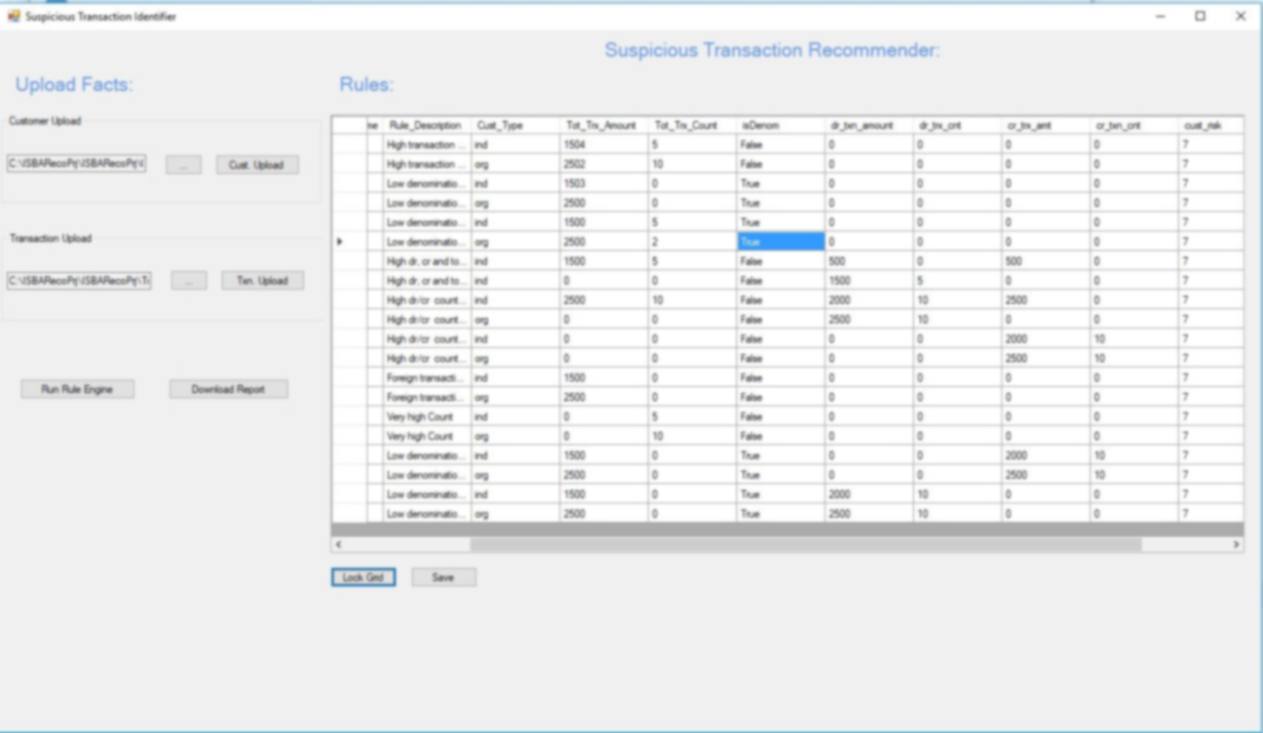
A

B



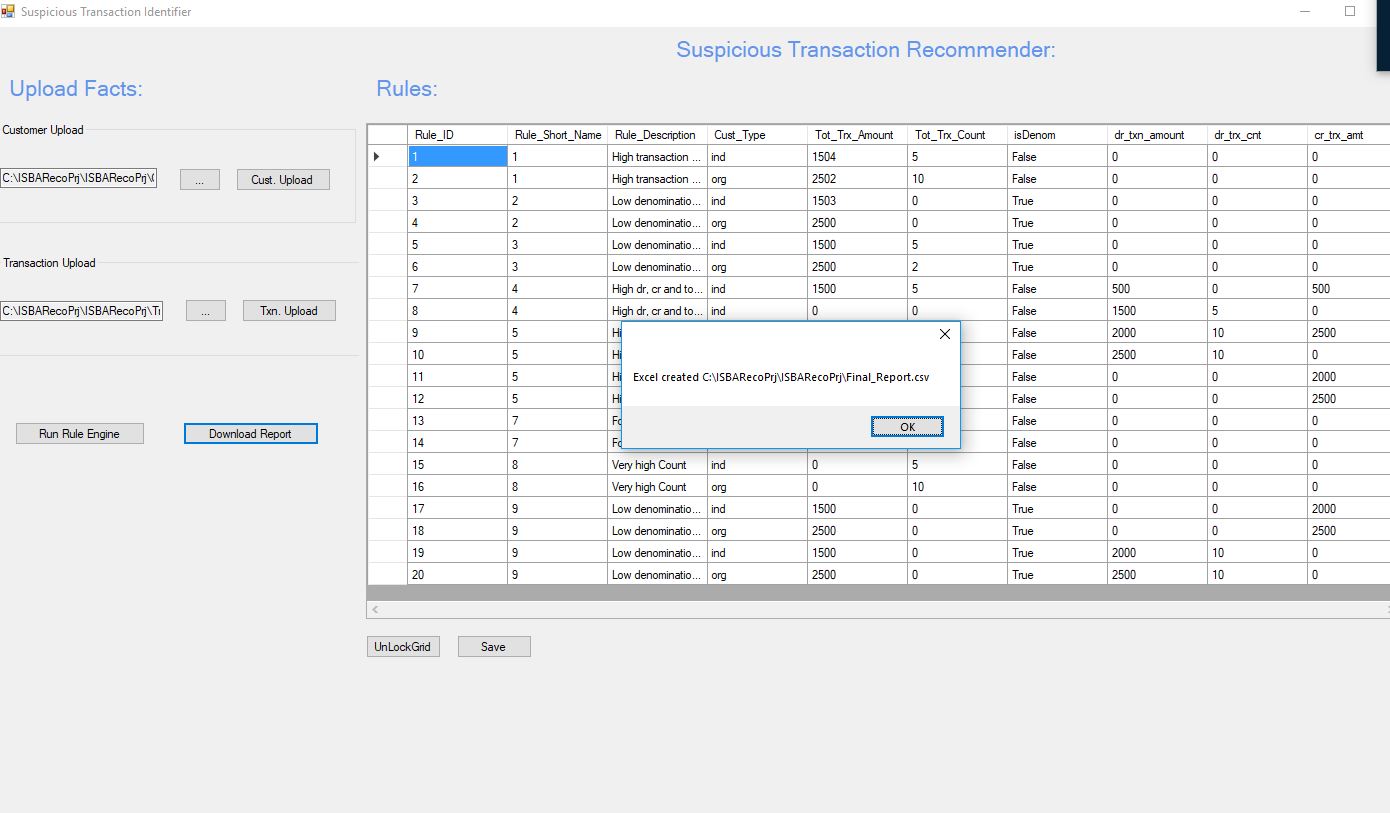
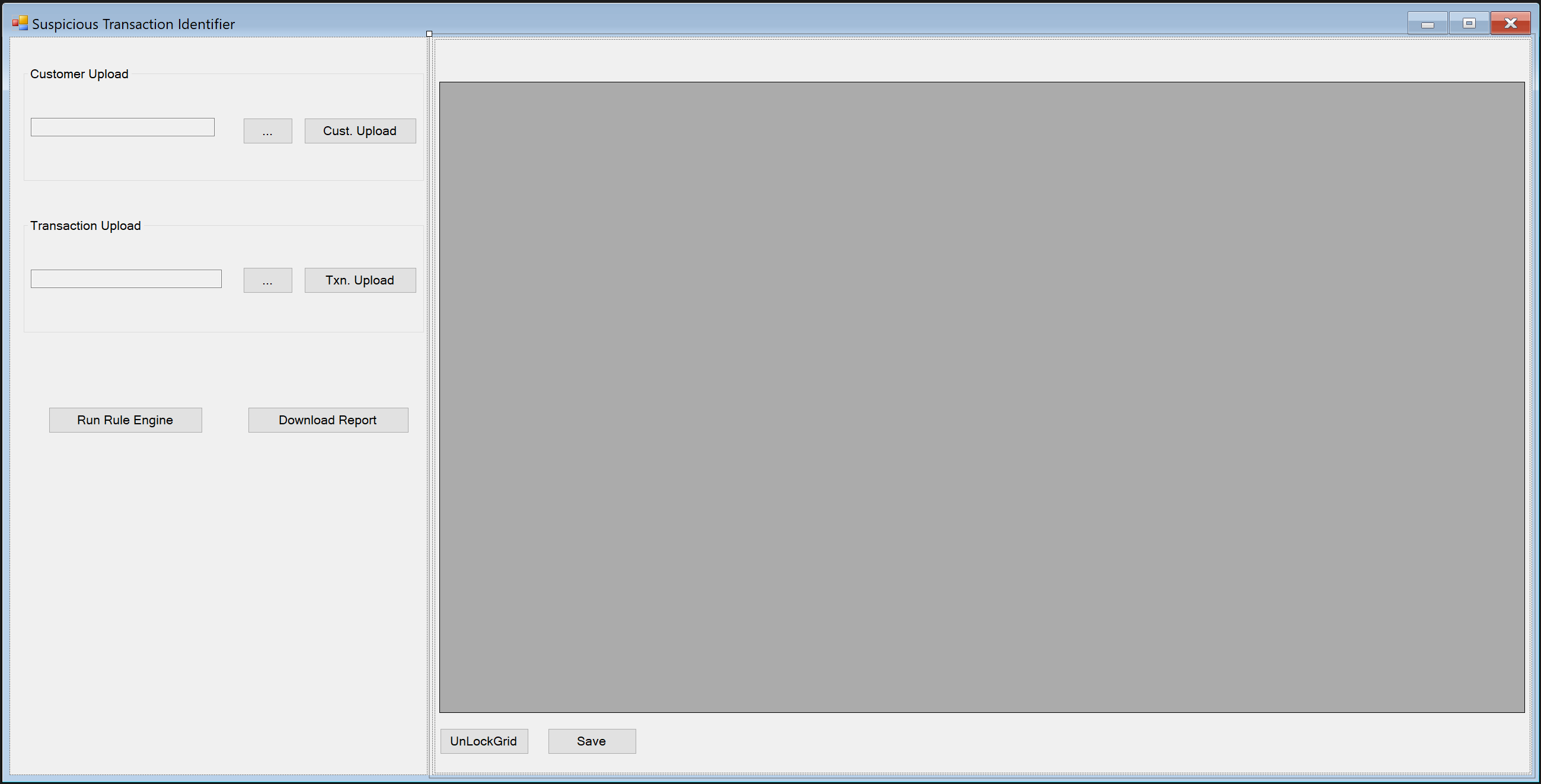
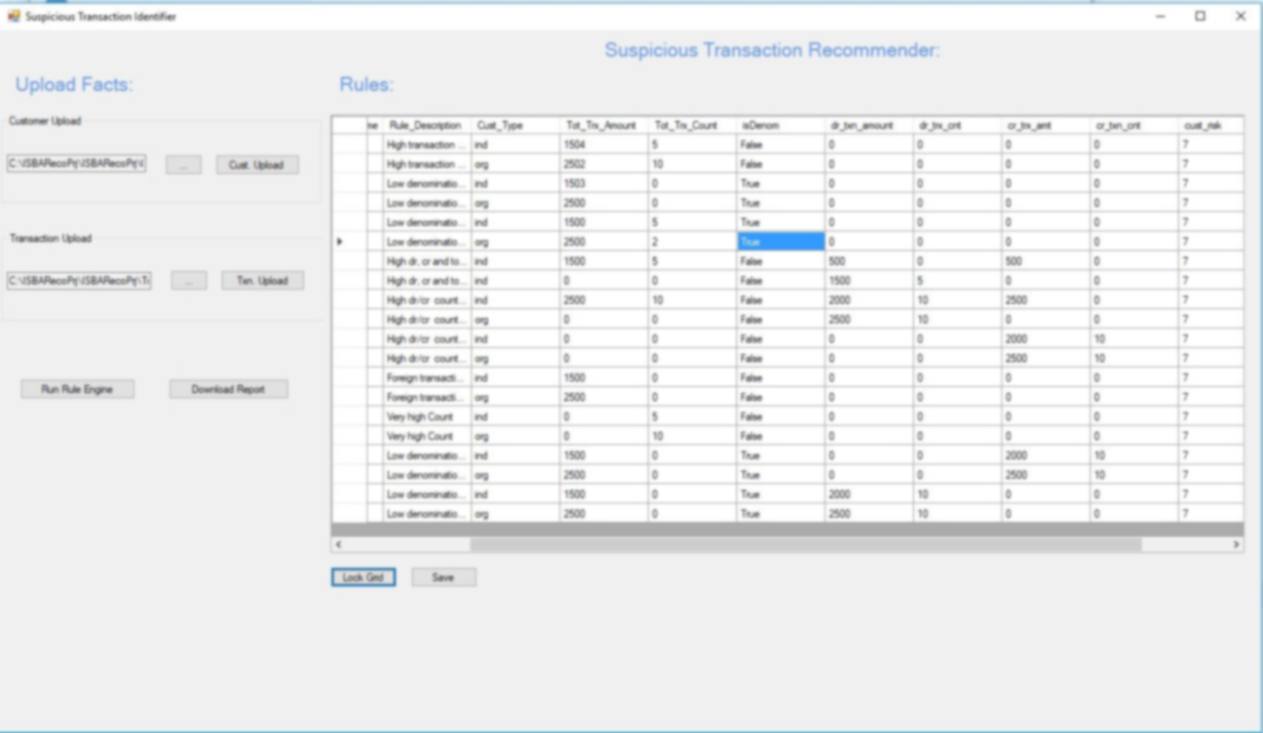
## Step 3: Run the Rules Engine

* Click on button “Run Rule Engine” to compile and process the data
* Once done, the relevant threshold will be shown in the window labelled “A”



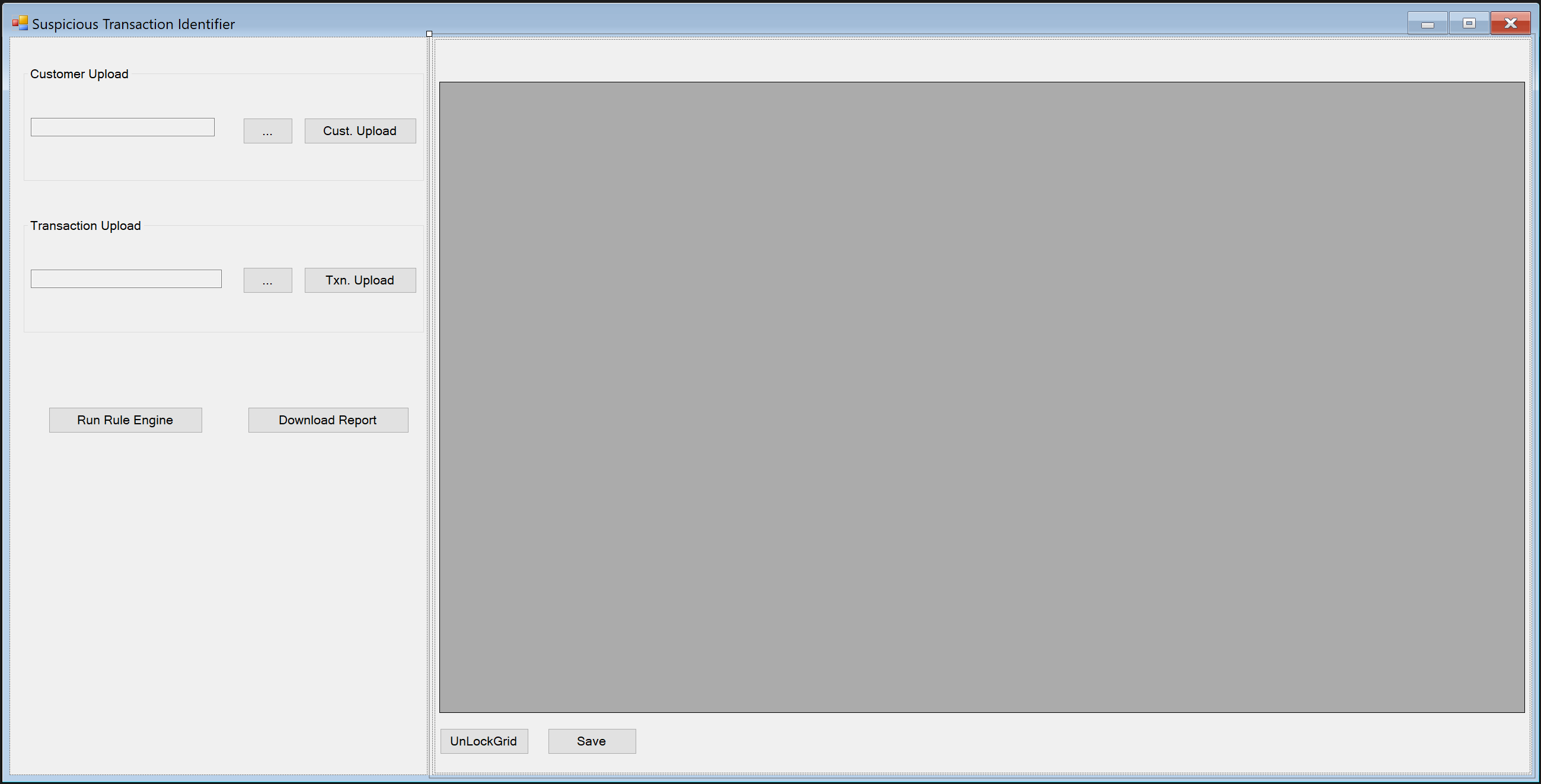
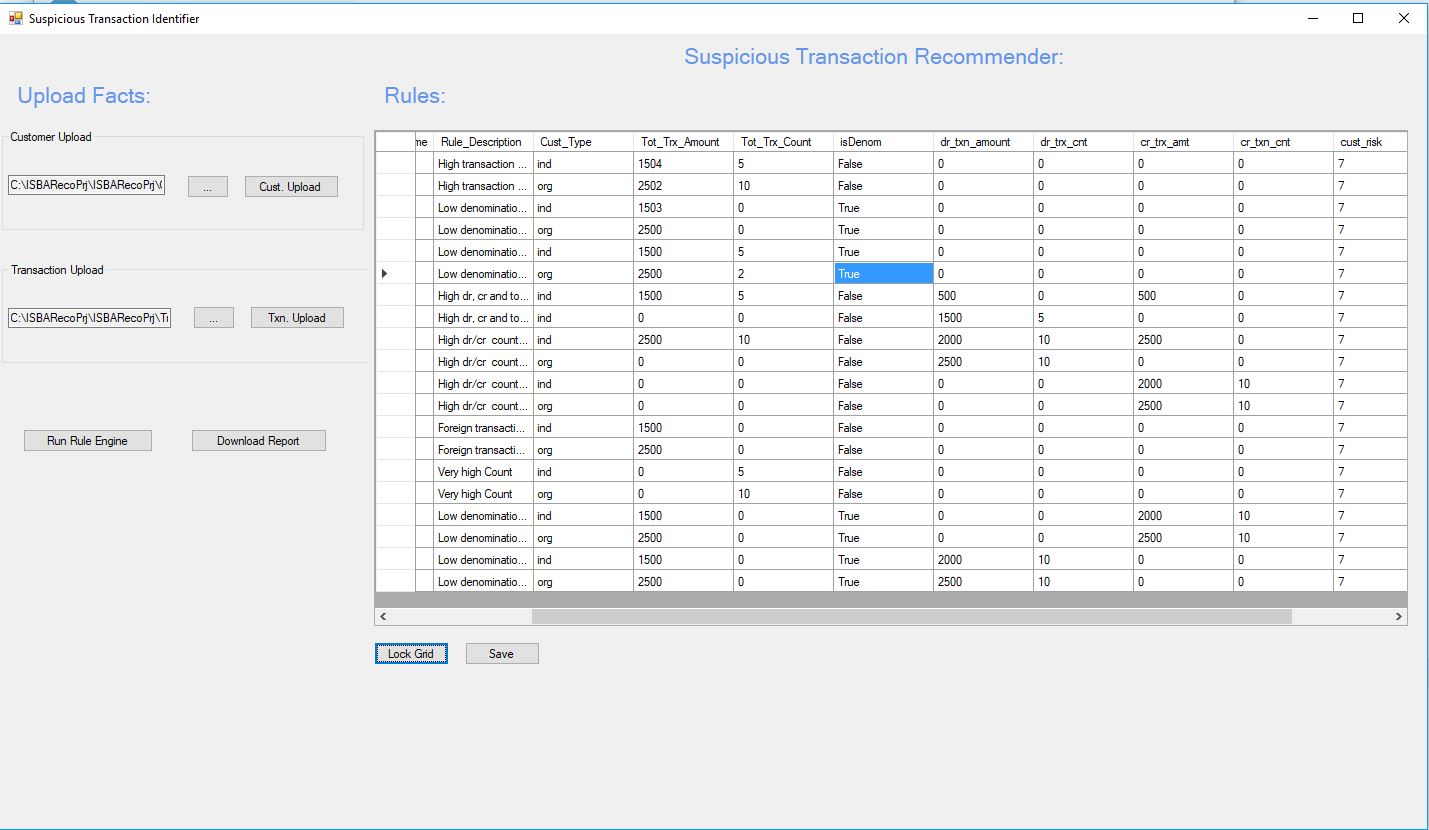
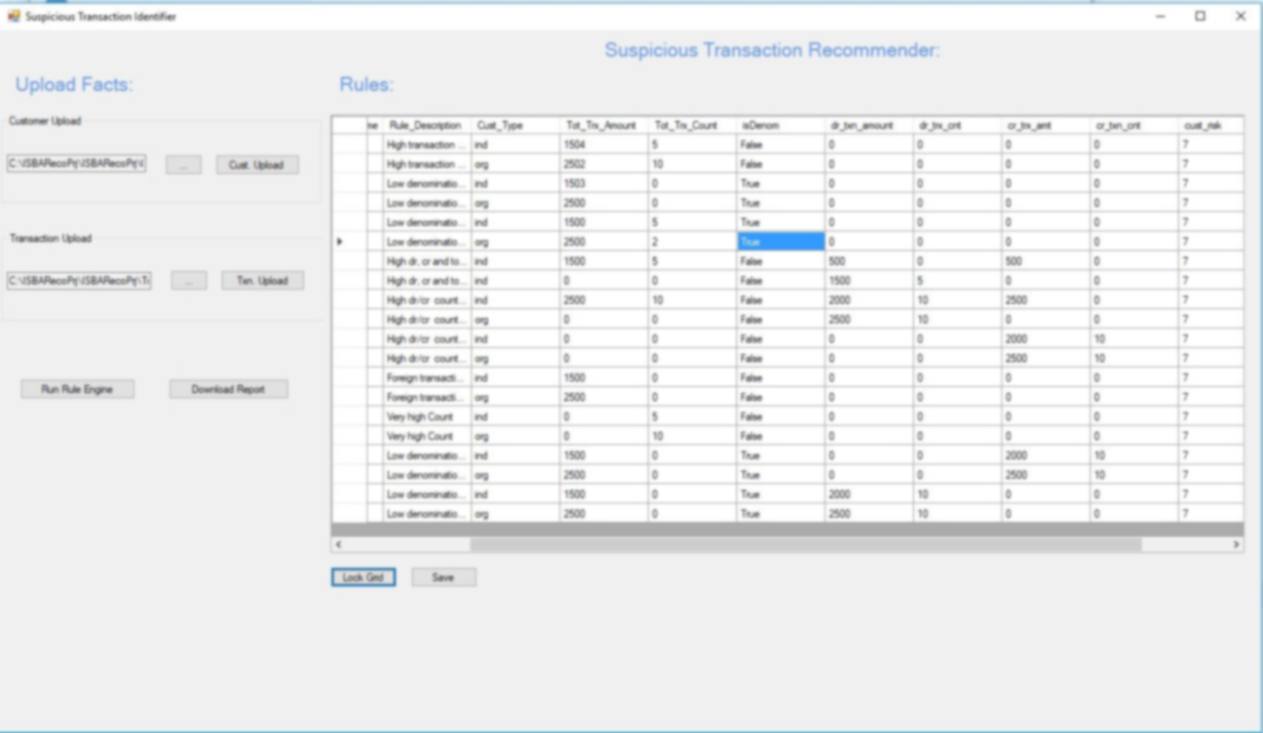
## Step 4: Download Report

* Click on button “Download Report” to download and view the list of fraudulent transactions
* A popup window will indicate that the report is successfully generated, proceed to the indicated folder to access the file
* Upon review, you may consider the following
  + The relevant fraudulent transactions are genuine, no adjustment needed
  + The listed fraudulent transactions need to be fine-tuned, proceed to step 5



## Step 5: Adjust Threshold

* Click on button “UnlockGrid” to release the threshold fields in the window labelled “A”
* Adjust the threshold limits and click on the button “Save”
* Re-run Step 3 and Step 4 to review the transactions



**Sample Report**



**Customer & Transaction Facts**

|  |  |
| --- | --- |
| **Customer Fact** | **(deftemplate customer\_trxn**  (slot customer\_id) (slot cust\_type) (slot trxn\_id) (slot total\_trxn\_amt) (slot benef\_id) (slot ftc\_ind) (slot denom) (slot cust\_risk\_ind) (slot dr\_cr\_ind)  ) |
| **Customer Transactions Detail** | **(deffacts cust\_trxn\_dtl**  (customer\_trxn (customer\_id 1) (cust\_type ind) (dr\_cr\_ind dr) (trxn\_id 10004) (total\_trxn\_amt 3000) (benef\_id 20) (ftc\_ind 0) (denom 0) (cust\_risk\_ind 6))  (customer\_trxn (customer\_id 1) (cust\_type ind) (dr\_cr\_ind dr) (trxn\_id 10000) (total\_trxn\_amt 3000) (benef\_id 19) (ftc\_ind 0) (denom 0) (cust\_risk\_ind 2))  (customer\_trxn (customer\_id 1) (cust\_type ind) (dr\_cr\_ind cr) (trxn\_id 10001) (total\_trxn\_amt 5000) (benef\_id 23) (ftc\_ind 1) (denom 1) (cust\_risk\_ind 10))  (customer\_trxn (customer\_id 1) (cust\_type ind) (dr\_cr\_ind dr) (trxn\_id 10002) (total\_trxn\_amt 15000) (benef\_id 23) (ftc\_ind 0) (denom 1) (cust\_risk\_ind 2)  ) |
| **Aggregation Rule** | **(defrule total\_amt\_cnt** =>  (bind ?count 0)  (bind ?sum 0)  (do-for-all-facts ((?f customer\_trxn)) TRUE  (bind ?count (+ ?count 1))  (bind ?sum (+ ?sum ?f:total\_trxn\_amt)))  (assert (trxn\_tot\_amt ?sum))  (assert (trxn\_tot\_cnt ?count))  ;(printout t ?sum " sum " ?count " count " crlf)  **)** |
| **Scenario Rule** | **(defrule scn1\_trxn\_cnt\_check\_ind**  (scn1\_ind\_trxn\_amt ?trxn\_amt\_th)  (scn1\_ind\_trxn\_cnt ?trxn\_cnt\_th)  (trxn\_tot\_amt ?trxn\_tot\_amt)  (trxn\_tot\_cnt ?trxn\_tot\_cnt)  (customer\_trxn (customer\_id ?id) (cust\_type ?cust\_type) (trxn\_id ?trxn\_id))  (test (> ?trxn\_tot\_cnt ?trxn\_cnt\_th))  (test (eq ?cust\_type ind))  (test (> ?trxn\_tot\_amt ?trxn\_amt\_th))  =>  ;(assert (scn1\_ind\_cnt\_chck no))  (open "data\_output.dat" data1 "a")  (printout data1 (str-cat ?id","?trxn\_id",SCENARIO1") crlf)  (close data1)  (printout t ?id " is individual category alerted in scnearion high count and high value transaction " crlf)  **)** |