Software Requirements Specification

For

Market Basket Analysis

Date: 12/11/2021

Prepared by

Specialization	SAP ID	Student Name
AI & ML	500075162	ARRA SAI NITISH REDDY
AI & ML	500075774	ARPIT
AI & ML	500075930	ATULIT SHUKLA

UPES

UNIVERSITY WITH A PURPOSE

Department of Informatics
School Of Computer Science
UNIVERSITY OF PETROLEUM & ENERGY STUDIES,
DEHRADUN- 248007. Uttarakhand

Table of Contents

Revision History

Date	Change	Reason for Changes	Mentor Signature

1. INTRODUCTION

Apriori algorithm is a data mining technique that is a classic algorithm of association rule mining, Apriori algorithm is the originality algorithm of **Boolean association rules** of mining frequent itemsets, which was proposed by **R.Agrawal and R. Srikan in 1994**

1.1. Purpose of the Project:

This algorithm is used to find out all the **frequent itemsets** based on **prior knowledge** of frequent itemset properties. Currently, association rules mining problems are highly valued by researchers in databases, artificial intelligence, statistics, information retrieval, visibility, information science and many other fields. **Association rules** are created by analyzing data for frequent if/then patterns and using the criteria **support and confidence** to identify the most important relationships.

Motivation:

Although E-Commerce is opening a gateway of business opportunities, it also creates worries for the offline retailers, backed by their huge investors which would result in effective frequent item mining, other data mining techniques of consumer data to increase their sales and to provide more effective marketing strategies which would make offline retailers unable-able to compete equally. An individual offline retailer cannot afford, an expert consumer data analysis to provide a competing strategy.

1.2. Project Scope:

This project helps small retailers to determine **effective marketing strategies** by performing **market basket analysis** using the **apriori algorithm** for **frequent itemset mining** on their past data to discover **associations and correlations** among items, which would be integrated with their Point of Sales System(POS)

1.3. Target Beneficiary:

Some of the target beneficiaries:

- Small scale business owners
- Retail supermarkets
- Product sales Manager

1.4. References:

- Data Mining: Concepts and Techniques by Jiawei Han, Micheline Kamber and Jian Pei
- HackerEarth blog: https://www.hackerearth.com/blog/developers/beginners-tutorial-apriori-algorithm-data-mining-r-implementation/
- Wikipedia: https://en.wikipedia.org/wiki/Apriori_algorithm
- https://personal.utdallas.edu/~chung/Fujitsu/UML_2.0/Rumbaugh--UML_2.0_Reference_CD.pdf
- https://www.google.co.in/books/edition/Implementation_and_Analysis_of_Apriori_A/xWzInQEACAAJ?hl=en&kptab=getbook
- https://www.geeksforgeeks.org/apriori-algorithm/
- https://www.javatpoint.com/apriori-algorithm-in-machine-learning
- https://www.educative.io/edpresso/what-is-the-apriori-algorithm
- https://www.tutorialspoint.com/uml/uml_standard_diagrams.htm
- https://www.lucidchart.com/blog/types-of-UML-diagrams

2. PROJECT DESCRIPTION

2.1. Reference Algorithm:

Market basket analysis uses *Apriori Algorithm* for mining frequent items and generating association rules from those items to generate knowledge. Technical details and data structure-wise implementation of the Apriori algorithm for our purpose is described below:

Explanation of general terms used in the Apriori Algorithm:

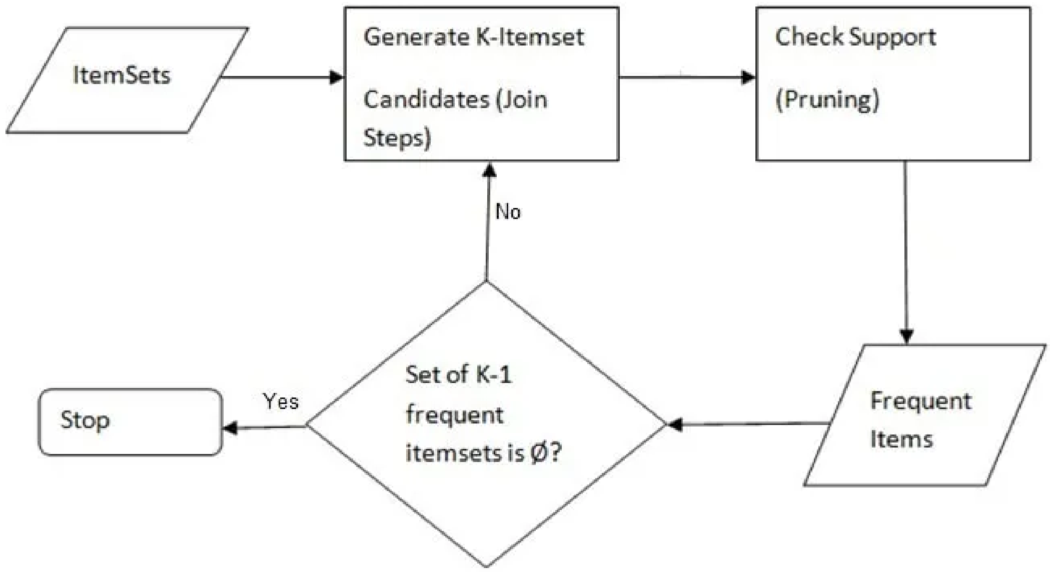
- *Itemset:* Collection or a set of items grouped together is called an itemset. Itemset consisting of k items is called k-itemset where $k >= 2$
- *Frequent Itemset*: An itemset is called frequent if it satisfies a minimum threshold value for support and confidence
- *Support*: It signifies items' frequency of occurrence
- *Confidence*: It signifies the conditional probability of occurrence i.e., one item purchased after other

Apriori Algorithm: It is based on the principle that subsets of frequent itemsets are also members of frequent itemsets. It aims to find frequent itemsets that run on a set of data. This algorithm employs mainly two steps "join" and "prune" to reduce the search space. Apriori Algorithm uses an iterative approach to get all the frequent itemsets.

- *Join Step*: This step is responsible for generating $(K+1)$ from K itemsets using the $F(k-1) * F(k - 1)$ algorithm for candidate generation
- *Prune Step:* To reduce the size of candidate itemsets, this step is performed. This step checks if the candidate generated satisfies the minimum support, if not it is removed from the set since it is regarded as infrequent

The general flow of Apriori Algorithm for frequent itemset mining:

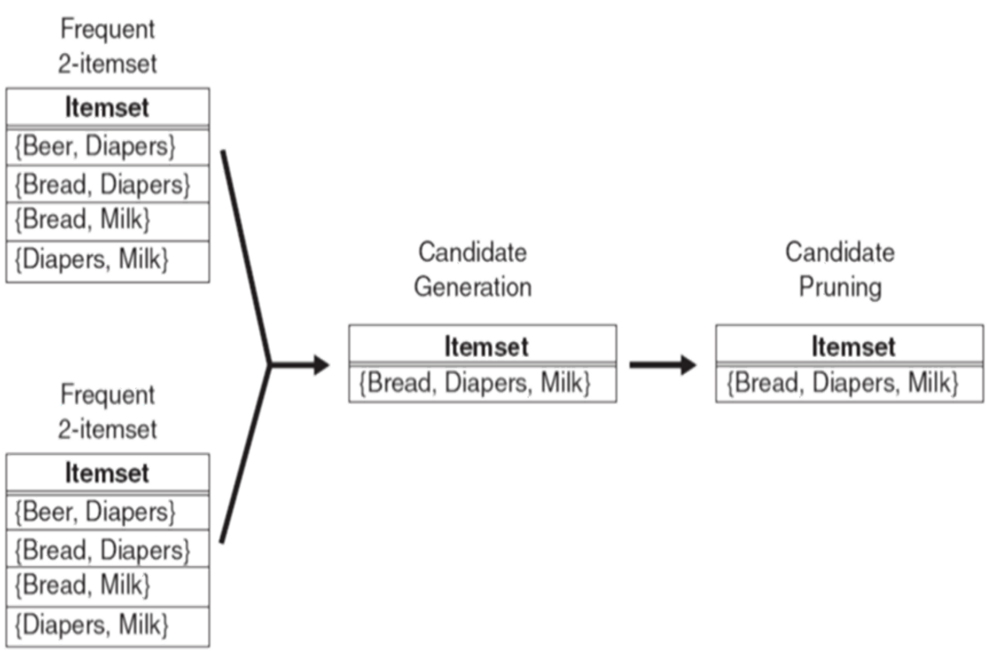
1. Initially, scan the database to generate 1-frequent itemsets
2. Candidate generation using $F(k-1)*F(k-1)$ algorithm (Join Step); Generate length $(K+1)$ candidate itemsets from K-itemsets
3. Prune step: Candidates generated in the above step is checked against the minimum support threshold value
4. The algorithm is terminated when frequent itemsets or candidates cannot be formed further.



(Source: https://www.hackerearth.com/blog/developers/beginners-tutorial-apriori-algorithm-data-mining-r-implementation/)

Candidate generation using *f(k-1) x f(k-1)* algorithm :

- In this procedure, it merges a pair of frequent (k - 1)-itemsets only if their (k - 2) items are identical
- An additional pruning step is required in this algorithm to ensure that remaining (k - 2) subsets of candidates are frequent

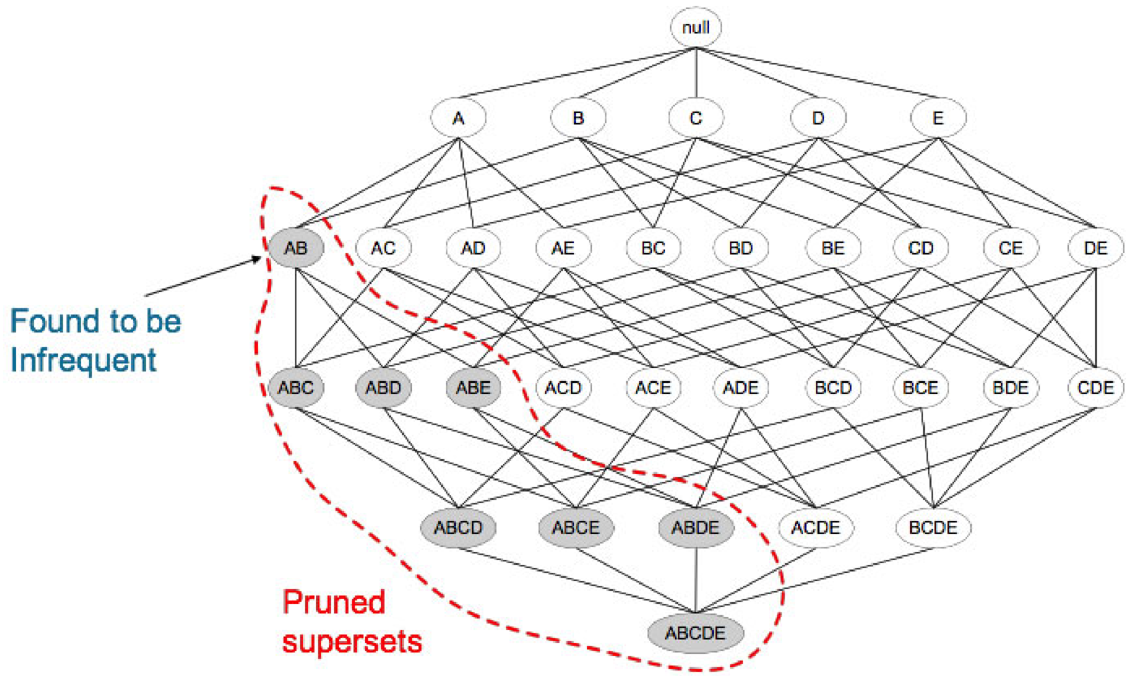


(Source: https://chih-ling-hsu.github.io/2017/03/25/apriori)

Rule Generation in Apriori Algorithm:

- A level-wise approach is used for generating association rules. Every level in this approach corresponds to the number of items that belong to the rule
- The theorem that is used to compare rules generated from the same frequent itemset Y:

  If a rule X → Y-X does not satisfy the confidence threshold, then any rule X' → Y-X' where X' is a subset of X, must not satisfy the confidence threshold as well.
- The first extraction of those rules that have high confidence and only one-item in the rules consequent are extracted. New candidate rules can be generated from these rules.
- The below diagram shows association rules in form of a lattice structure for the frequent itemset {a, b, c, d}



(Source: http://jcsites.juniata.edu/faculty/rhodes/ml/assocRules.html)

- If a low-confidence node is found, then the entire subgraph that is spanned by the node is to be pruned as shown in the above figure.

Data structures:

A combination of data structures is used to contain information for this project

- A structure consisting of the dynamic array (vector) is used for Frequent itemsets that consist of sets of k-frequent itemsets using vector<vector<string>> which represents the nested name of items in a particular itemset and vector<int> support to store their support count which is of integer type

```
struct FrequentItemsets
{
    vector<vector<string>> itemsets;
    vector<int> supports;
};
```

- A structure consisting of a dynamic array (vector) of string is used to contain the left-hand side and right-hand side of a particular association rule.

```
struct AssociationRule
{
    vector<string> lhs;
    vector<string> rhs;
};
```

- A structure consisting of dynamic arrays (vector) and AssociationRule structure described above is used to store Association rules, vector<int> is used to contain support and vector<double> is used to contain confidence values

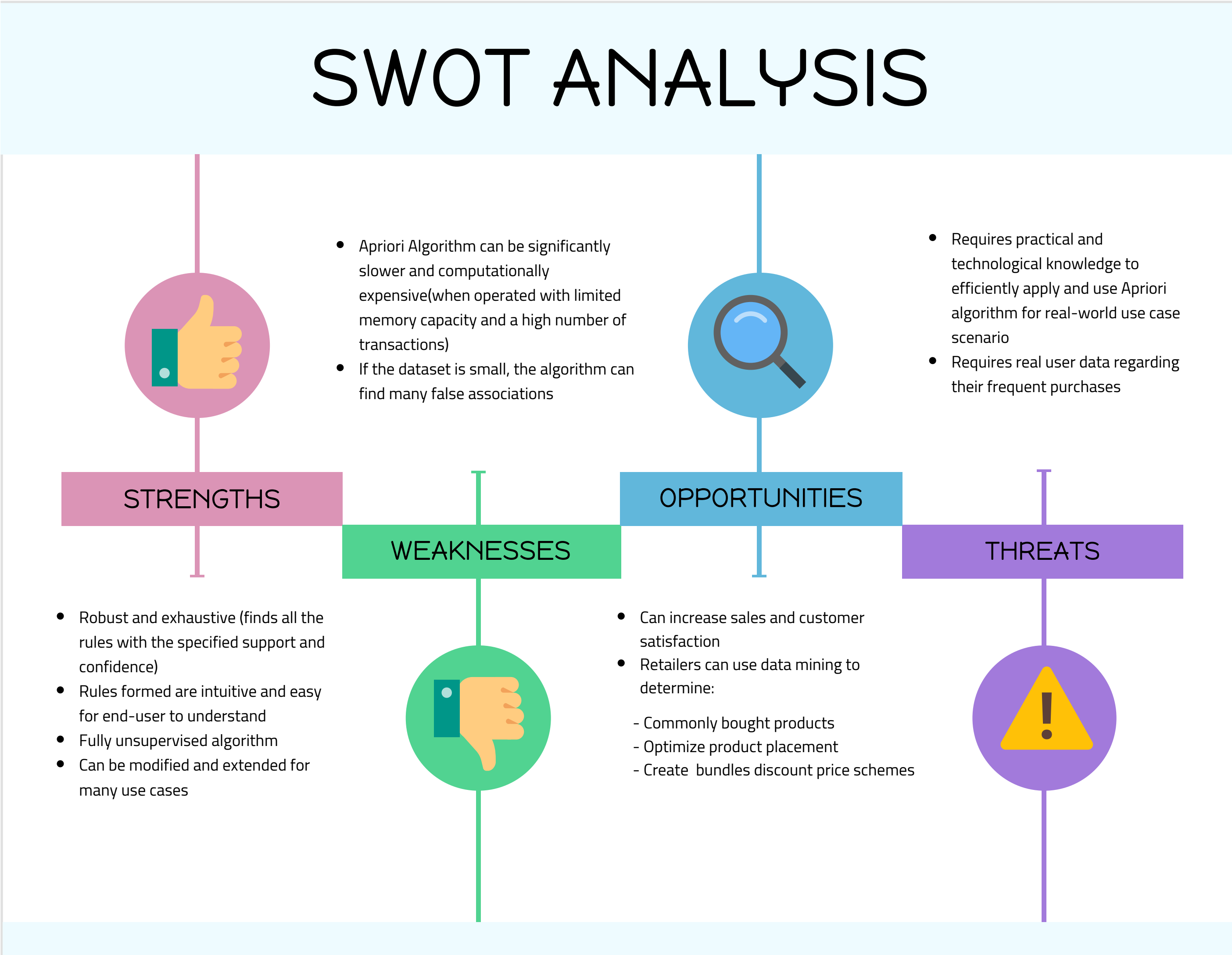
```
struct Association
{
    vector<AssociationRule> rules;
    vector<int> supports;
    vector<double> confidences;
};
```

- Although not primarily, but Hashmap data structure is also used for comparing support count in one-frequent-itemset generation function

2.2. Characteristic of Data:

- Source of data: https://www.kaggle.com/irfanasrullah/groceries
- Data Description: The dataset consists of transactional data of items in the shopping cart of a grocery store
- Characteristic of the dataset :
    - There are a total of 6 columns representing a maximum of 6 items per transaction
    - The data set is preprocessed(external) to exclude transactions with the singular items as the information gain is low and limited to a maximum of six transactions as its computationally expensive

2.3. SWOT Analysis



2.4. Project Features

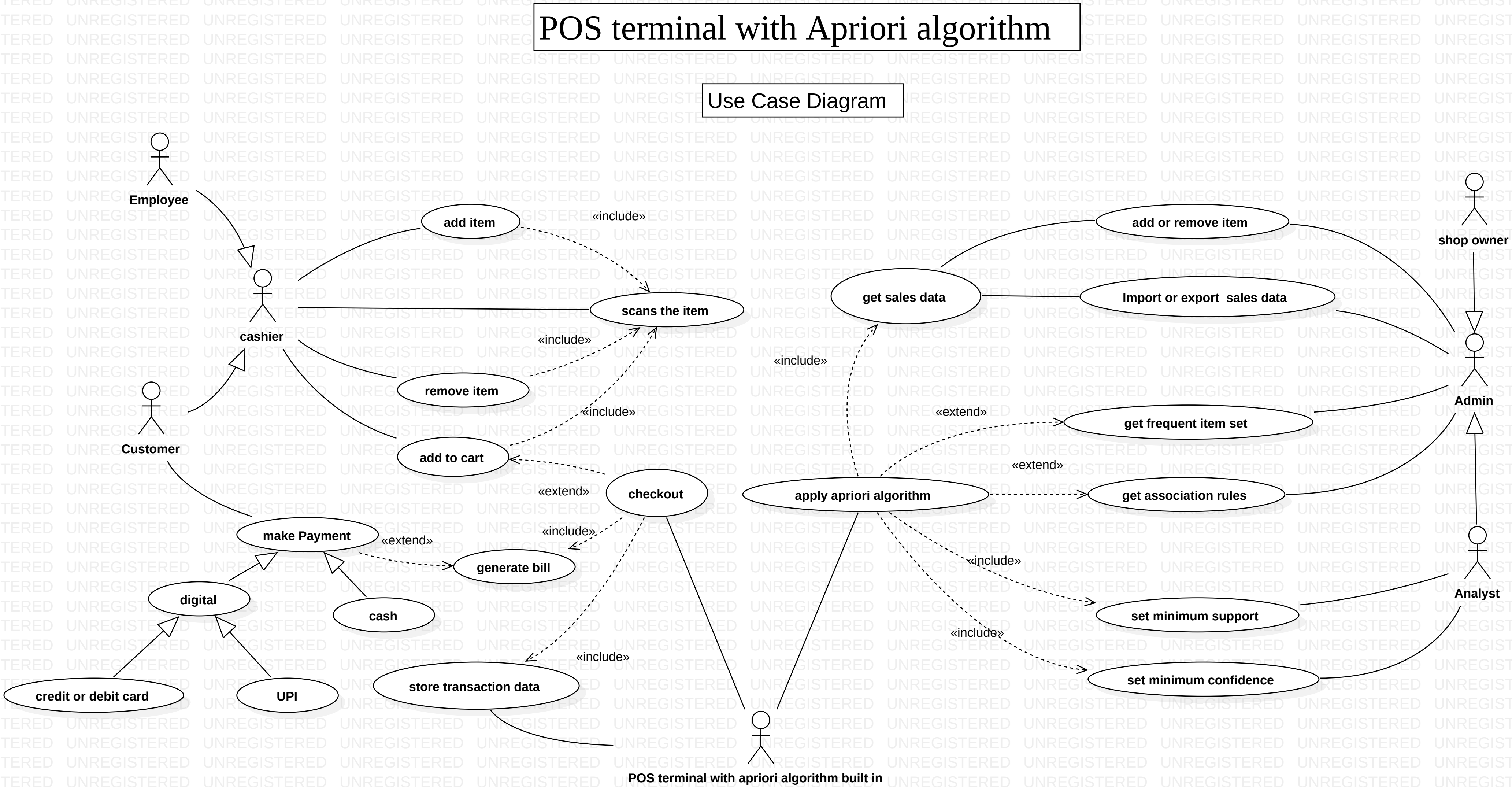
- Here the user is a shop-keeper or an analyst
- The user can perform actions like:
    - Import or export the transaction-data
    - Set minimum support and minimum confidence
    - Request for frequent itemsets
    - Request for association rules
- Heare the cashier is considered as an external agent who can perform the tasks of a POS(point of sales) system
- Each transaction is stored which then be used to perform the apriority algorithm

2.5. User Classes and Characteristics

- There are three user classes majorly :
    - **Admin** which is generalized into:
        - *owner/shop-keeper* who is capable to execute the algorithm
        - An *analyst* who is responsible for determining the minimum support and minimum confidence
    - **Cashier** which is generalized into:
        - *Employees* who perform normal checkout
        - *Customers* who can perform self-checkout
    - **POS terminal with apriori algorithm built-in**: it is a machine that can generate association rules and frequently bought itemsets upon admin request
    -

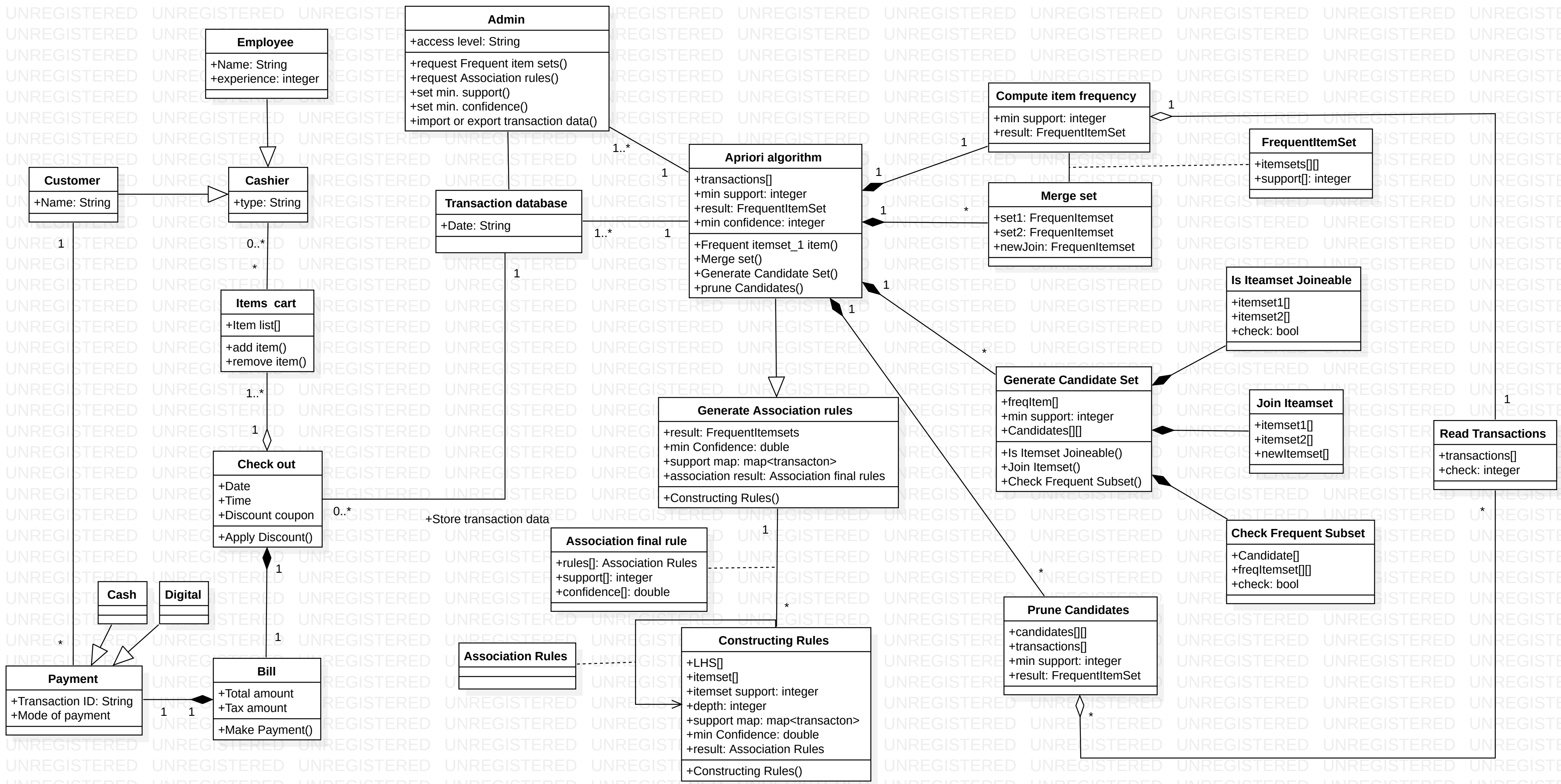
2.6. Design and Implementation Constraints:

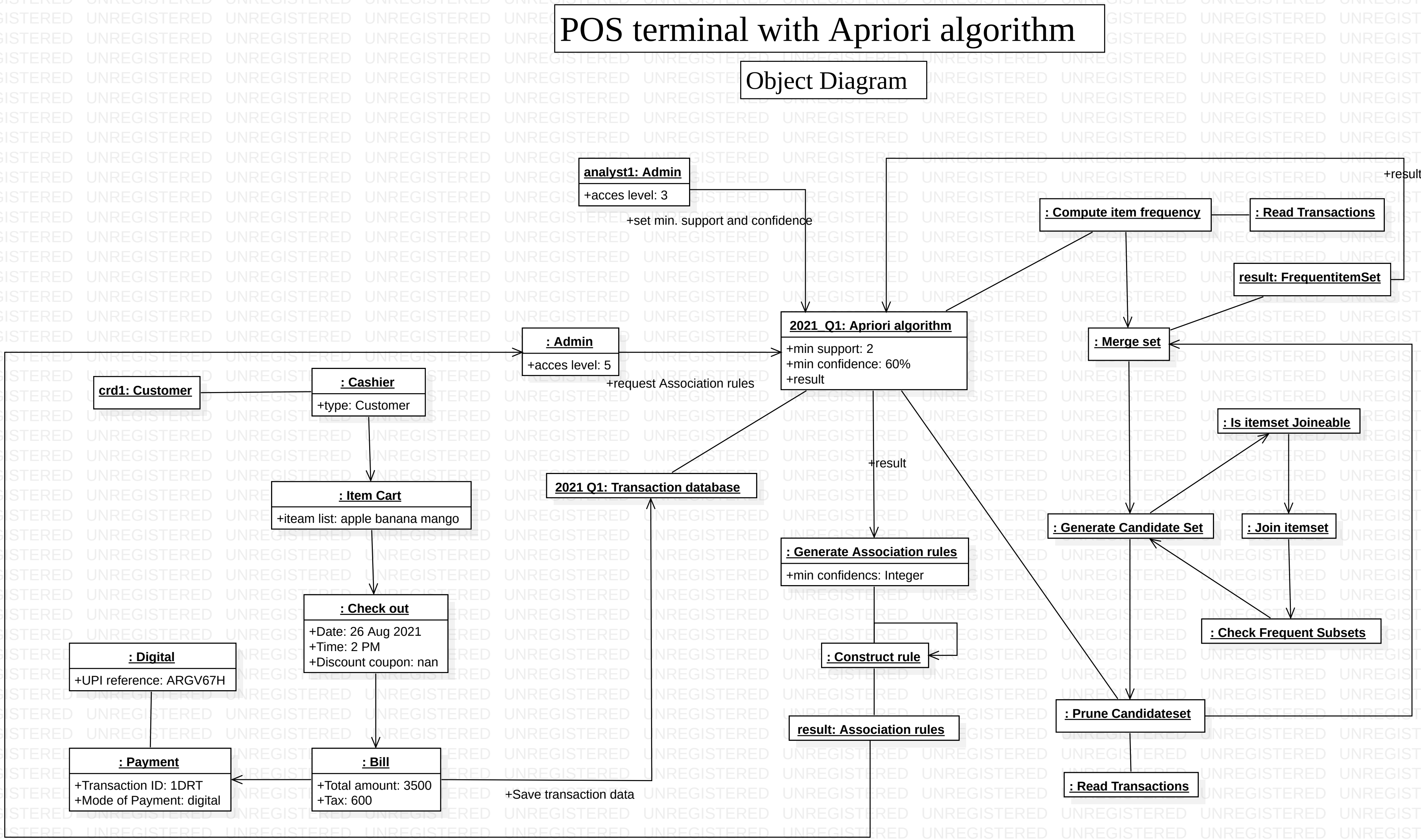
- The input transactional data must be Flat-File
- Minimum support and minimum confidence must be provided prior to the algorithm execution
- The admin must pose prior knowledge of the program in order to understand the output
- The integrable platform must be capable of executing the C++ program

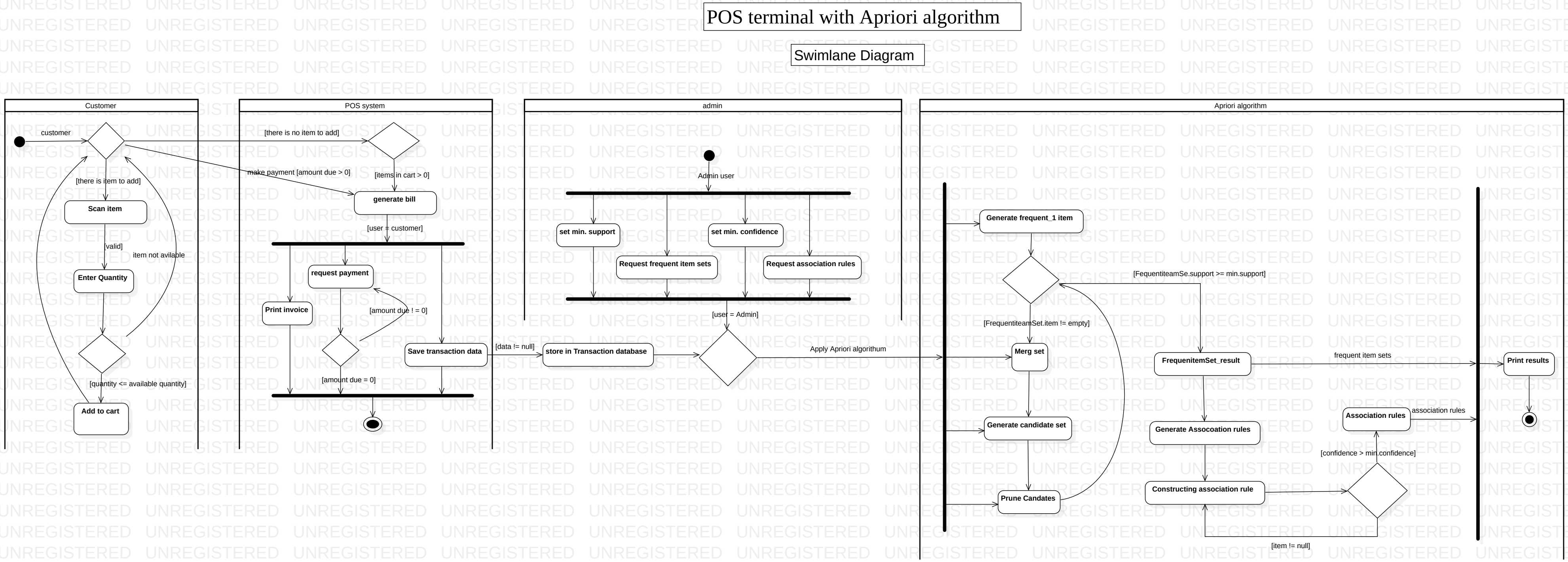

POS terminal with Apriori algorithm
Use Case Diagram
Employee
cashier
Customer
add item
scans the item
remove item
add to cart
checkout
make Payment
generate bill
digital
cash
credit or debit card
UPI
store transaction data
«include»
«include»
«include»
«extend»
«extend»
«include»
«include»
POS terminal with apriori algorithm built in
get sales data
add or remove item
Import or export sales data
apply apriori algorithm
get frequent item set
get association rules
set minimum support
set minimum confidence
«include»
«extend»
«extend»
«include»
«include»
shop owner
Admin
Analyst

POS terminal with Apriori algorithm

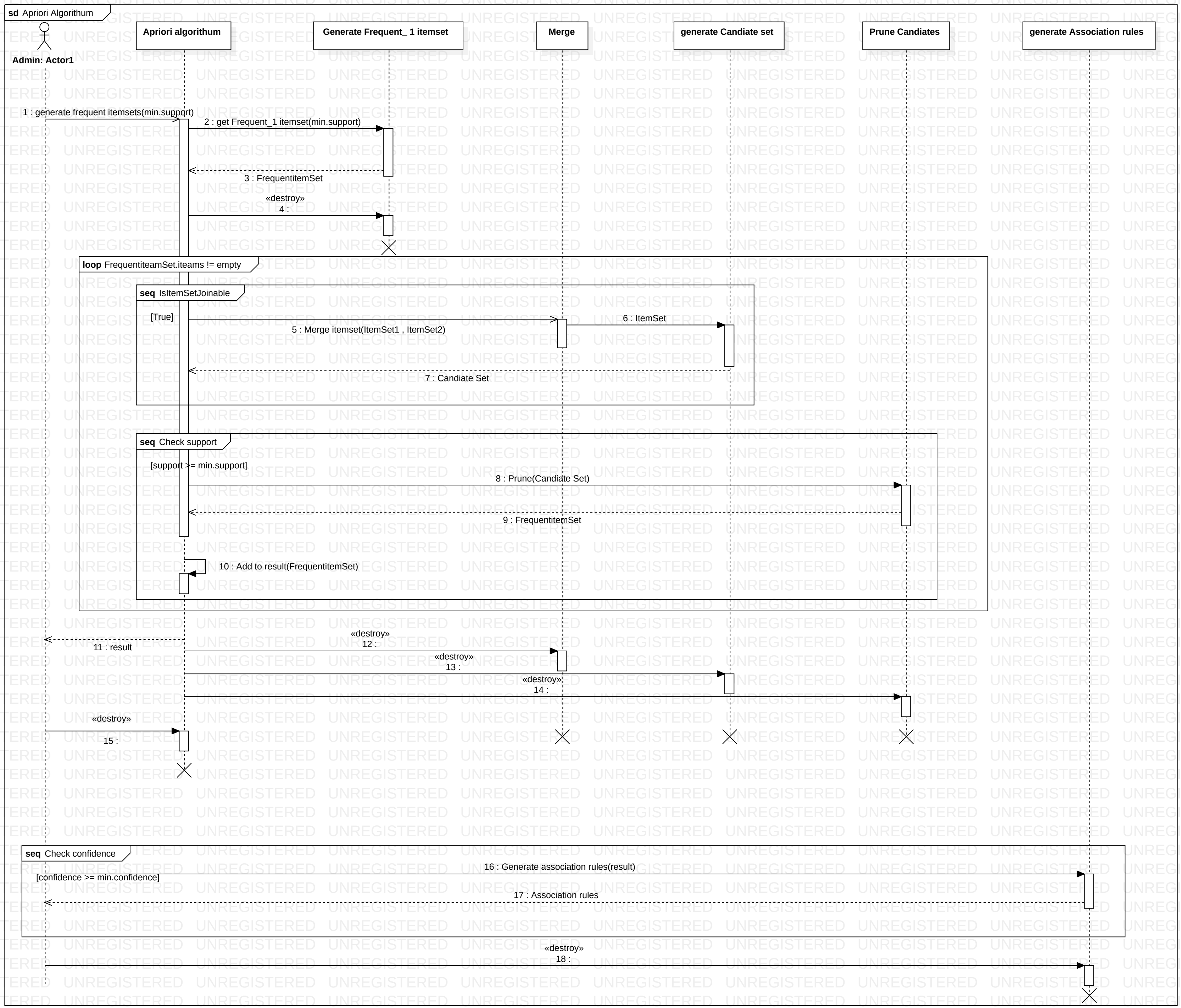
Class Diagram




POS terminal with Apriori algorithm
Object Diagram
analyst1: Admin
+acces level: 3
+set min. support and confidence
: Compute item frequency
: Read Transactions
+result
result: FrequentitemSet
2021_Q1: Apriori algorithm
+min support: 2
+min confidence: 60%
+result
: Merge set
: Admin
+acces level: 5
+request Association rules
crd1: Customer
: Cashier
+type: Customer
: Is itemset Joineable
+result
: Item Cart
+iteam list: apple banana mango
2021 Q1: Transaction database
: Generate Candidate Set
: Join itemset
: Generate Association rules
+min confidencs: Integer
: Check out
+Date: 26 Aug 2021
+Time: 2 PM
+Discount coupon: nan
: Check Frequent Subsets
: Digital
+UPI reference: ARGV67H
: Construct rule
: Prune Candidateset
result: Association rules
: Payment
+Transaction ID: 1DRT
+Mode of Payment: digital
: Bill
+Total amount: 3500
+Tax: 600
+Save transaction data
: Read Transactions


POS terminal with Apriori algorithm
Swimlane Diagram
Customer
POS system
admin
Apriori algorithm
customer
[there is no item to add]
make payment [amount due > 0]
[there is item to add]
Scan item
[valid]
item not avilable
Enter Quantity
[quantity <= available quantity]
Add to cart
[items in cart > 0]
generate bill
[user = customer]
request payment
Print invoice
[amount due ! = 0]
Save transaction data
[amount due = 0]
[data != null]
store in Transaction database
Admin user
set min. support
set min. confidence
Request frequent item sets
Request association rules
[user = Admin]
Apply Apriori algorithm
Generate frequent_1 item
[FequentiteamSe.support >= min.support]
[FrequentiteamSet.item != empty]
Merg set
Generate candidate set
Prune Candates
FrequenitemSet_result
frequent item sets
Print results
Generate Assocoation rules
Constructing association rule
Association rules
association rules
[confidence > min.confidence]
[item != null]

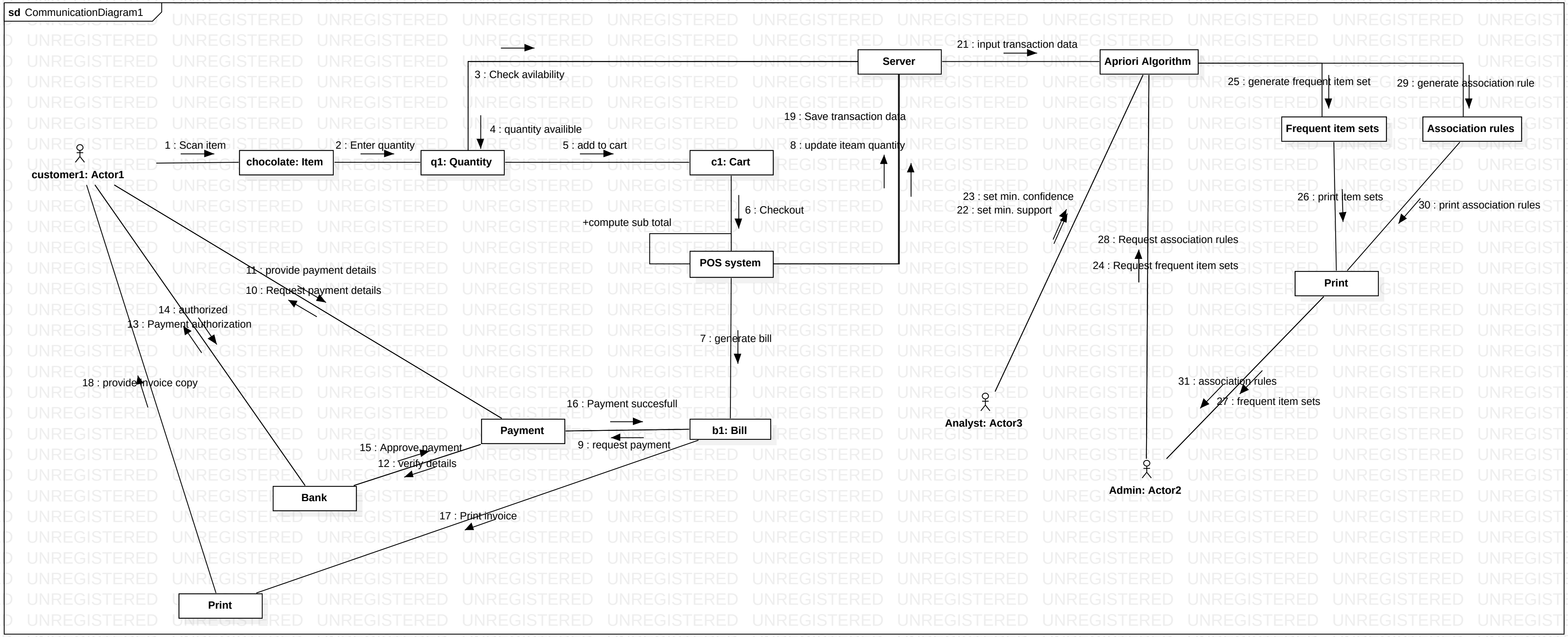
Model::Collaboration1::Interaction1::Apriori Algorithm

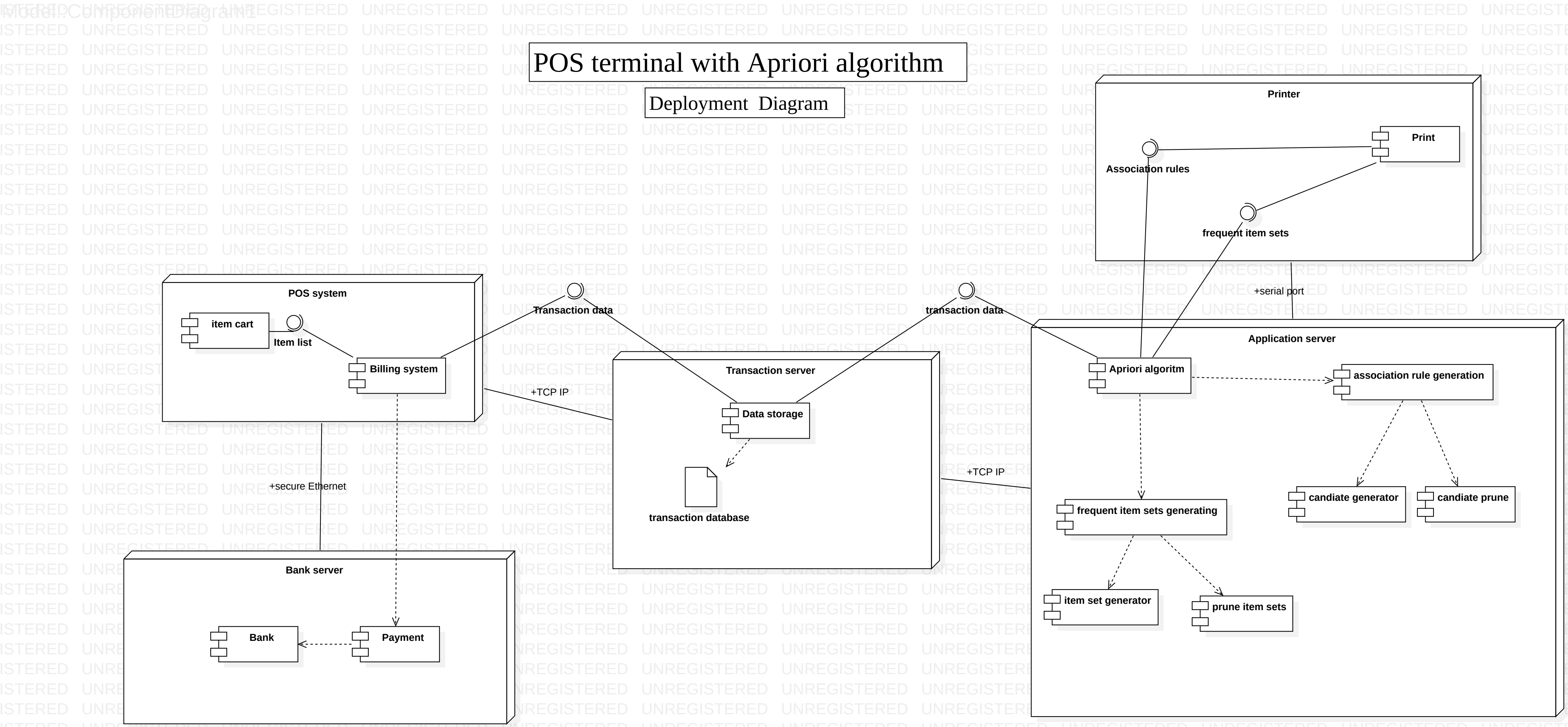


Model::Collaboration1::Interaction1::CommunicationDiagram1

POS terminal with Apriori algorithm

Communication Diagram




Model::ComponentDiagram1
POS terminal with Apriori algorithm
Deployment Diagram
Printer
Print
Association rules
frequent item sets
+serial port
POS system
item cart
Item list
Billing system
Transaction data
transaction data
+TCP IP
Transaction server
Data storage
transaction database
+TCP IP
Application server
Apriori algoritm
association rule generation
frequent item sets generating
candiate generator
candiate prune
item set generator
prune item sets
+secure Ethernet
Bank server
Bank
Payment

2.8. Assumption and Dependencies

- We have assumed that the executing POS terminal is integrated with the apriori algorithm in a use-case scenario
- The apriori algorithm is dependent on the analyst for minimum support and minimum confidence
- The minimum support and minimum confidence is acquired from an analyst who is aware of the sales model used
- The price movement of the products is not reflected in the algorithm performance

3. SYSTEM REQUIREMENTS

3.1. User Interface:

User input is provided by the user on the initial screen when prompted by the program for dataset_filename, support value and confidence value

- The data generated is outputted on the terminal screen and thus only a command-line terminal is needed for interacting with this program

3.2. Software Interface

The software is divided majorly into two modules:

a. **Frequent itemset generation module**: This module is responsible for generating frequent itemsets using the Apriori algorithm. It uses *8 helper functions* for generating the output of candidates. All the functions use and definitions are described below:

- ***AprioriAlgo()*** - this is a primary function that is responsible for applying the apriori algorithm and returning the result as a FrequentItemset structure
- ***Frequent_one_itemsets()*** - As evident by the name, this function generates the initial one frequent itemsets and returns result in FrequentItemset structure.
- ***Merge()*** - This function returns the merged result of frequent k-itemset
- ***genCandidates()*** - This function is responsible for generating all the possible candidates using the F(k - 1) * F(k - 1) algorithm which is described in the algorithm section of the SRS document.
- ***IsJoinItemsets()*** - This is a helper function for genCandidates() and checks the condition i.e, returns true for merging a pair of frequent (k −1)-itemsets only if their first k −2 items, arranged in lexicographic order, are identical
- ***JoinItemsets()*** - helper function of *genCandidates()* to join itemsets obtained from *IsJoinItemset* function
- ***hasInfrequentSubset()*** - helper function of *genCandidates* is responsible for candidate pruning before candidate-generation since an additional candidate pruning step is required so that it is ensured that remaining (k - 2) subsets of the candidate are frequent
- ***pruneCandidates()*** - After genCandidates has generated all the possible candidates, this function is responsible for filtering out only those candidates that have a *support_count* >= *minimum_support_count*

The *AprioriAlgo()* function executes until the FrequentItemsets vector becomes empty, after this happens all the frequent itemsets are stored in our vector with their respective support counts and can be outputted on the screen for analysis.

b. Association rule generation module: This module is responsible for generating association rules using the Apriori algorithm. It uses *2 helper functions* for generating the output of candidates. All the functions use and definitions are described below:

- ***genAssociationRules()*** - Function to generate association rules from frequent items set based on confidence provided by the user and returns result in Association structure.
- ***ConstructRule()*** - Consists of the main logic used to generate association rules from frequent itemsets by using the left-hand side and right-hand side items

After the *genAssociationRules()* function is completed its execution, all the association rules are stored in respective LHS and RHS vectors and can be outputted on the screen for analysis.

3.3. Database Interface

This program requires a *Flat-File Database* or simply put a file as our database system, in our particular case we are using a .txt file as input for the initial dataset for processing by other modules

3.4. Protocols

There are no external protocols limited to this project. Some other requirements of the program are specified below:

- Presence of a well-formatted dataset that contains items separated by commas enclosed in square brackets
- Appropriate file permission for the dataset file such that it is not denied access for use by the C++ Program and is readily readable by the file stream of the C++ program

4. NON-FUNCTIONAL REQUIREMENTS

4.1. Performance requirements

- A really robust and high-performance algorithm is not needed for generating frequent itemsets and association rules
- Since this algorithm is mainly to be used by store retailers, e-commerce owners to improve their marketing, selling strategy and performance, it doesn't have to be as fast as a real-time system or close to that
- Performance requirement is on the medium scale since we don't need real-time communication (very high performance) and also at the same time we don't need very slow performance.

4.2. Security requirements

- Security requirements are dependent upon business requirements and particular use case
- If the frequent itemsets and association rules that are generated by the program possess any trade secrets or confidential information about the products, password protection or encryption could be used

4.3. Software Quality Attributes

4.4. Quality Attributes	**Performance**
adaptability	Medium
availability	Medium
correctness	Medium
flexibility	Heigh
interoperability	Low
maintainability	Heigh
portability	Heigh
reliability	Medium
reusability	Heigh
robustness	Heigh
testability	Medium
usability	Heigh

- **Adaptability**:
  The apriori algorithm is not adaptable by itself as the transaction data will be getting updated in real-time, but when integrated with the POS system which can be scheduled to generate association rules at each end of a financial day it can be adaptable
- **Availability**:
  Its availability is *average* as the apriori algorithm involves various tasks like frequent itemset generation, candidate pruning and association rule generation
- **Correctness:** it depends on the association rules generated from the real-time transactional data
- **Flexibility**: it is *highly* flexible, as the association rules are generated as a result of user-inputted minimum support and confidence
- **Robustness**: it is highly robust, as it finds all the rules with the specified support and confidence

5. Other Requirements

Appendix A: Glossary

- Market Basket Analysis: data-mining techniques used by retailers to find customer purchasing patterns and optimize their sales strategy
- Itemset:
- Apriori Algorithm: Popular data-mining algorithm for mining frequent itemsets and generating association rules
- Itemset: Collection or a set of items grouped together is called an itemset. Itemset consisting of k items is called k-itemset where k >= 2
- Frequent Itemset: An itemset is called frequent if it satisfies a minimum threshold value for support and confidence
- Support: It signifies items' frequency of occurrence
- Confidence: It signifies the conditional probability of occurrence i.e., one item purchased after other