Ain Shams University

Faculty of Computer & Information Sciences

Information Systems Department

جامعة عين شمس

كلية الحاسبات و المعلومات

قسم نظم المعلومات

Live Data Tracking
&
Analysis Dashboard

By:

1. Wardshan Essam
2. Youssef Gamal
3. Youssef Morris
4. Wassim Joesph
5. Mina Ossama

Year: 2021

Under Supervision of:

Wedad Hussein,

Assistant Professor,

Information Systems Department,

Faculty of Computer and Information Sciences,

Ain Shams University.

Under Supervision of:

Menna Maged,

Teaching Assistant,

Information Systems Department,

Faculty of Computer and Information Sciences,

Ain Shams University.

Acknowledgment

Firstly we would thank God for being able to complete this project with success. Then we would like to express our special thanks for our supervisor Dr. Wedad Hussein who didn't keep any effort in encouraging us to do a great job, providing our team with valuable information and advices.

Last but not least we want to thank every team member for his great effort and hard work that helped us to achieve our goals.

Abstract

Today's data is generated by an infinite amount of sources - IoT sensors, servers, security logs, applications, or internal/external systems. It's almost impossible to regulate structure, data integrity, or control the volume or velocity of the data generated. Therefore, a tool for handling such kinds of data was needed.

In this project we are building a tool for the tracking and analysis of live data that offers a variety of features. The tool can provide information about frequent sold items on daily, monthly, yearly, quarter yearly and seasonal basis. It can also provide store branches profit details on daily, monthly, yearly and quarter yearly basis. Finally it can provides notification alert for every change as well as recommendations for decision making.

Market sales data is so hard to being collected and seen by managers day by day. We presented a web application that supply managers with details about daily, monthly, yearly, and quarter yearly profits and reports by automatically collecting the data from the stores every single period of time and filter it for easy way to track his stores data. Also a recommendation system that shows the recommended decisions in a visualized way to help the manager to increase his revenue.

Table of Contents

List of Figures

List of Abbreviations

- **IoT** : Internet of things
- **MSI** : Market step inspector (Project Name)
- **VFKM** : Very Fast K-Means
- **ANNCAD** : Adaptive Nearest Neighbor Classification Algorithm for Data Streams
- **FI** : Frequent Items
- **SQL** : Structured Query Language
- **SSMS** : SQL Server Management Studio
- **Mdf** : Main database file
- **API** : Application Programming Interface
- **EXE** : Executable

Chapter 1
Introduction

1.1. Motivation

At the past, several problems have been facing the world about collecting large amounts of data and processing it. Like searching for a student name in thousands of names on written papers which will take huge time and effort to find it.

The rapid growth in Big Data information science and technology in general and the complexity and volume of data in particular have introduced new challenges for the research community. Many sources produce data continuously like IOT.

IOT (internet of things) enables for increased awareness, security, and power-efficiency and so on. But, large IOT systems are very complex which implies among other things a traditional data analysis techniques alone are not adequate!

Data streaming appeared for solving this problem. At first the batching processing was used, which involved storing all the data in a data warehouse to facilitate search.

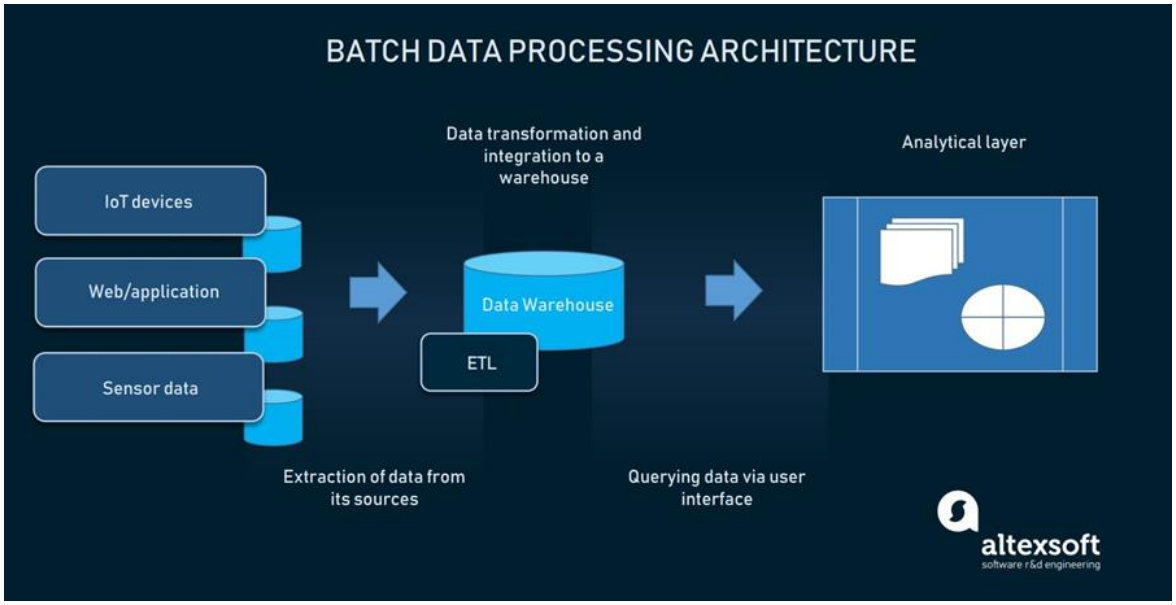


Fig 1.1 Batch Data Processing

Then another type of processing appeared which is the streaming processing that helped us process all the data we need live from the sources and put it in a data warehouse then anyone can achieve it at once.

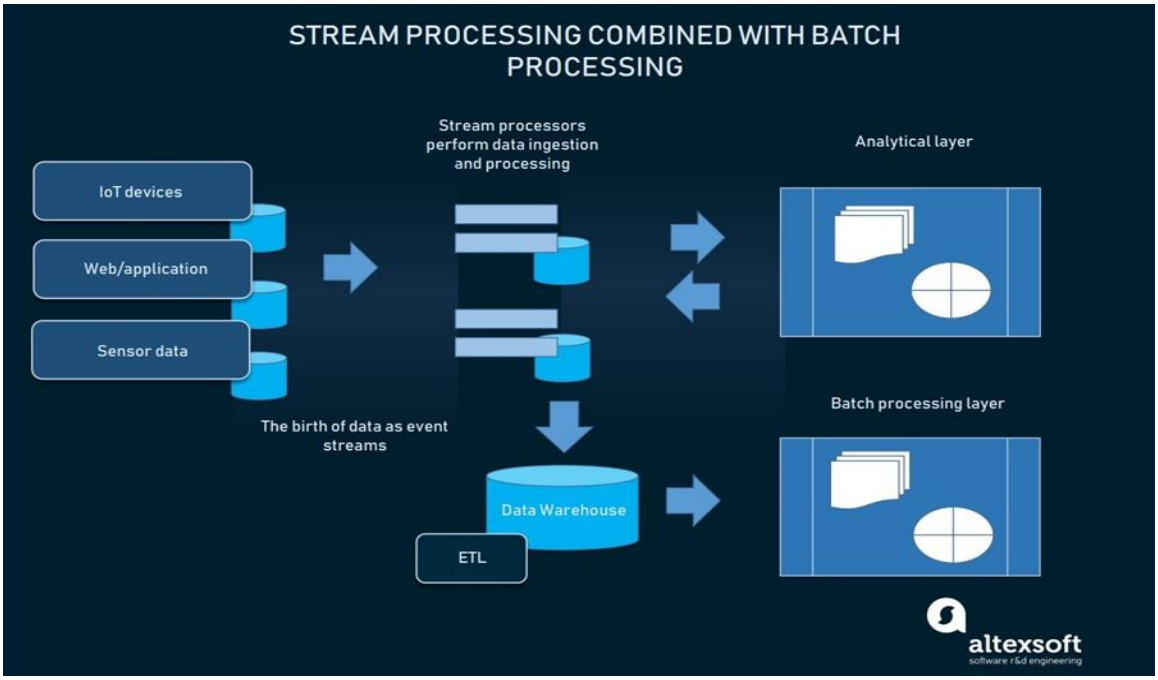


Fig 1.2 Stream Data Processing

So data streaming is a very important process that is used to alleviate challenges that face us in the real world like:

- Distributed deployment.
- Parallel deployment.
- Ordering and determinism.
- Shared-nothing vs. shared-memory parallelism.
- Load balancing.
- Elasticity.
- Fault tolerance.
- Data sharing.

1.2. Problem Definition

The explosive growth of data has brought many changes in the way of managing and processing data, big amount of data posing a great challenge in managing those data. Data are getting bigger because we continuously receive new streams of data, so the volume of data increase so fast and the management of this data is getting harder with time. It's hard to collect data from different places so we can process them and manage them. Data amounts are getting larger but there is starvation for information. So, a specialist is needed to convert the collected data to information that is useful and can be used, which cost more money. It takes a lot of time and costs a lot of money to track the collected data.

1.3. Objective

The Objective of our System is to make the managers benefit from the large market sales data streams by live tracking & processing data streams through mining techniques then display the extracted knowledge on an interactive dashboard managers by showing profits and reports daily, monthly, yearly and quarter yearly along with some recommendations for seasons to help in making decisions.

1.4. Document Organization

The following chapters of this document are organized as follows:

Chapter 1: Introduction

This chapter gives an overview about the problem that faces the world and how is has been solved along with our objective of our project.

Chapter 2: Background

This chapter gives an overview of the scientific background behind the project. We discuss all the information about data streams and how it helps us.

Chapter 3: Analysis and Design

This chapter presents the outcomes of the analysis and design phases of the project. It discusses the functional and nonfunctional requirements of the system and its architecture. This chapter also offers a detailed account of the functionalities and processes offered by the project expresses as UML diagrams.

Chapter 4: Implementation

This chapter explains the implementation of the system by showing what each function means and doing.

Chapter 5: User Manual

This chapter offers a guideline for how to open the system and how to use it properly.

Chapter 6: Conclusions and Future Work

This chapter shows the conclusions of our system work done along with our perspective for the possible directions of the future work.

Chapter2
Background

2.1 Definition live data tracking and analytics

"The goal is to turn data into information, and information into insight."

-Carly Fiorina-

Nowadays live data tracking & analytics has become a central concern for making timely right decisions.

As Live data tracking & analytics lets users track, analyze and understand data as it arrives in a system. Logic and mathematics are applied to the data so it can give users insights for making real-time decisions.

Live data tracking helps us to trace & inspect the data starting from its early beginning once it has been created, till we extract the needed knowledge from it.

Live data analytics allows businesses to get insights and act on data immediately or soon after the data enters their system.

Live data systems for analytics answer queries within seconds. They handle large amounts of data with high velocity and low response times.

Analytics can be on-demand or continuous. On-demand extracted knowledge results when the user requests it. Continuous updates users as events happen and can be programmed to respond automatically to certain events. For example, live data web analytics is updated every time the user load the web page with the preset parameters.

The Major benefit in live data tracking & analytics is speed. The less time a business must wait to access data between the time it arrives and is processed, the faster a business can use data insights to make changes and act on critical decisions. For instance, analyzing monitoring data from a manufacturing line would help early intervention before machinery malfunctions.

Live data tracking & analytics offers the following advantages over traditional analytics:

1. Timely right decisions.
2. Support decisions makers.
3. Live update for the need knowledge.
4. Being aware with the change of the behavior in the business.
5. Help the business to easily approach the users need.

2.2 Difficulties in live data collection and tracking

Live Data tracking & analytics are extremely important for risk managers. They improve decision-making, increase accountability, benefit financial health, and help employees predict losses and monitor performance.

However, achieving these benefits is easier said than done. There are several challenges that can impede live Data tracking & analytics ability to collect, track & analysis data.

There, are some of the challenges that faces the live Data collecting, tracking & analytics:

1. **The amount of data being collected**

   "We are drowning in data, but starving for knowledge" - John Naisbitt

   Continuously, a lot of data streams are being delivered with a huge amount of raw data that can't be used without being analyzed.

2. **Collecting meaningful and live data**

   Not all received data can be helpful in analysis and design making as outdated data can have significant negative impacts on decision-making.

   Due to the noise impact on the live data it mess up the meaningful information that comes out from live data collection.

3. **Visual representation of data**

   To be understood and impactful, extracted knowledge from live data often needs to be visually presented in graphs or charts. While these tools are incredibly useful, it's difficult to build them manually. Taking the time to pull information from multiple areas and put it into a reporting tool is frustrating and time-consuming.

4. **Budget**

   Another challenge is that to make live data collecting, tracking and analysis it need high skilled developers and expensive tools which is a big obstacle to be done.

5. **Shortage of skills**

   Depending on Employees to make decisions, knowledge extraction or build up reports is too risky due to high rate of human error and they may not have the knowledge or capability to run in-depth data analysis.

2.3 Types & fields of live data

Many types of live data that is being generated continuously from a variety of sources is considered streaming data, as:

1. Log files
2. E-commerce purchases
3. Market sales purchases
4. Weather events
5. Utility service usage
6. Geo-location of people and things
7. Server activity

And, these types of data are being generated & used in a lot of different fields, like:

- Location data.
- Fraud detection.
- Real-time stock trades.
- Marketing, sales, and business analytics.
- Customer/user activity.
- Monitoring and reporting on internal IT systems.
- Log Monitoring: Troubleshooting systems, servers, devices, and more.
- SIEM (Security Information and Event Management): analyzing logs and real-time event data for monitoring, metrics, and threat detection.
- Retail/warehouse inventory: inventory management across all channels and locations, and providing a seamless user experience across all devices.
- Ride share matching: Combining location, user, and pricing data for predictive analytics - matching riders with the best drivers in term of proximity, destination, pricing, and wait times.

2.4 Live data stream algorithms

Live data streams tracking & analytics need to go through an algorithms and techniques to extract from it the needed knowledge to help in decisions making process, the algorithm and the techniques used for the knowledge extraction depend on the purpose use of the extracted knowledge.

As if the purpose is Frequency Counting and Time Series Analysis so the following algorithms will fit:

- Lossy counting [8]:
    - The lossy count algorithm is an algorithm to identify elements in a data stream whose frequency count exceed a user-given threshold. The algorithm works by dividing the Data Stream into 'Buckets' as for frequent items, but fill as many buckets as possible in main memory one time.
- Approximate Frequent Counts [17]:
    - The approximate frequent counts algorithms are used for computing frequency counts exceeding a user-specified threshold over data streams. The algorithms are simple and have provably small memory footprints. Although the output is approximate, the error is guaranteed not to exceed a user-specified parameter. The algorithms can easily be deployed for streams of singleton items like those found in IP network monitoring, can also handle streams of variable sized sets of items exemplified by a sequence of market basket transactions at a retail store. For such streams, we describe an optimized implementation to compute frequent item sets in a single pass.

- FP Stream [18];
    - Frequent pattern mining is an important unsupervised learning task in data mining, with multiple applications. It can be used purely as an unsupervised, exploratory approach to data, and as the basis for finding association rules. It can also be used to find discriminative features for classification or clustering.

As if the purpose is clustering for live data into groups so the following algorithms will fit:

- STREAM and LOCAL SEARCH [19]:
    - It assumes that our data actually arrives in chunks X1 ... Xn, where each Xi is a set of points that fits in main memory. We can turn any stream of individual points into a chunked stream by simply waiting for enough points to arrive.
- VFKM [20][21][22]:
    - Very Fast K-Means (VFKM) is a clustering algorithm for data streams developed as a faster version of K-Means, to mine large quantities of continuous rapid streaming information. Its objective is to extend VFKM to a ubiquitous environment for execution on resource constrained devices.
- CluStream [23];
    - The CluStream method is a method of clustering data streams, based on the concept of micro clusters. Micro clusters are data structures which summarize a set of instances from the stream, and is composed of a set of statistics which are easily updated and allow fast analysis.

As if the purpose is Classification of live data into segments so the following algorithms will fit:

- SCALLOP [24]:
    - It provides a set of decision rules on demand which improves its simplicity and helpfulness for the user. SCALLOP updates the knowledge model every time a new example is read, adding interesting rules and removing out–of–date rules. As the model is dynamic, it maintains the tendency of data. Experimental results with synthetic data streams show a good performance with respect to running time, accuracy and simplicity of the model.
- ANNCAD [25]:
    - The algorithm achieves excellent performance by using small classifier ensembles where approximation error bounds are guaranteed for each ensemble size. The very low update cost of our incremental classifier makes it highly suitable for data stream applications. Tests performed on both synthetic and real-life data indicate that our new classifier outperforms existing algorithms for data streams in terms of accuracy and computational costs.

2.5 Relationship between live data tracking and the market

*"You can have data without information, but you cannot have information without data."***– *Daniel Keys Moran***

The relationship between the market and live data tracking are the same as "Daniel Keys" say's; the market is our data but without tracking and analytics there won't be information, so tracking and analytics systems are like the lost child without the market data and also the market data without the live tracking and analytics systems is meaningless.

The live data tracking and analytics turns the market from a sleepy & sorrow place with no passion to a race track that is full of motivation and ambition, as the market user is updated with the real-time data changes & challenges to shows where additional attention is needed and where requirements are being met, For example:

- Viewing orders as they happen for better tracking and to identify trends.
- Continually updated customer activity like page views and shopping cart use to understand user behavior.
- Targeting customers with promotions as they shop for items in a store, influencing real-time decisions.
- Employees can see how they are performing and what is expected of them, they can make on-the-spot corrections to better reach those performance standards.

2.6 Analysis dashboarding

A live dashboard is a performance tool used to analyze, track, and report on the organization's data in real time with the help of interactive data visualizations. These real time dashboards are automatically updated and provide the user with instant access to critical data, Through the use of data visualizations, dashboards simplify complex data and extracted knowledge to provide users with at a glance awareness of current performance.

Live dashboards take their name from automobile dashboards; in fact, when you think about it, live dashboards are used in much the same way as automobile dashboards. Under the hood of your vehicle, there may be hundreds of processes that impact the performance of your vehicle. Your dashboard summarizes these events using visualizations so you have the peace of mind to concentrate on safely operating your vehicle.

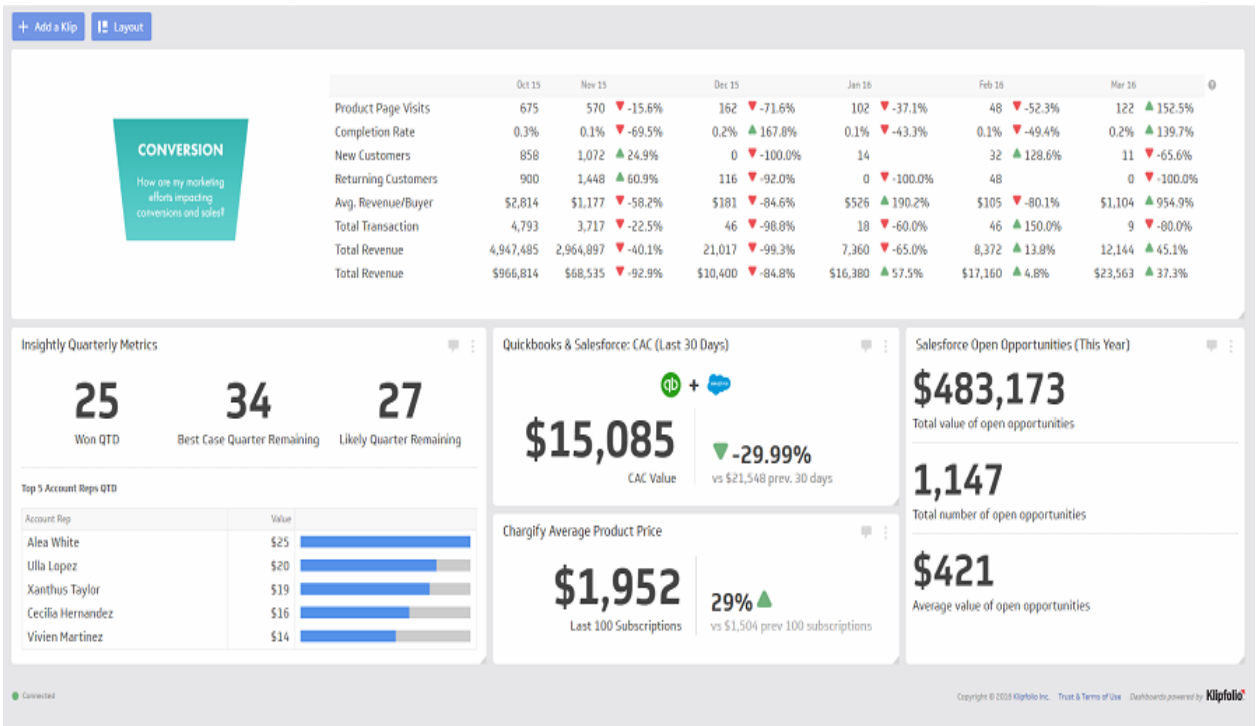


Fig 2.1 Dashboard example

2.7 Related work

Points of comparisons	Live GPS Vehicle-Tracking Devices	Klipfolio	Foodics
Definition	Real-time tracking uses a GPS tracking device to send updates in ten second to two minute intervals to a fleet manager or dispatcher and lets them know the location of a vehicle.	Sales data tracking dashboard	The #1 POS System in Middle East for Restaurants and cafe. Cashier, Digital Menu, Inventory, Integration with 3rd party apps, and much more. Big Data Analysis. Third Party Apps. Regular System Updates. 24/7 Support. POS in the Cloud. Multi-Language.

Features	Manage your fleet efficiently	Manage the performance of each of your sales reps or sales teams	Manage inventory and call center
	Track what your fleet are doing 24/7	Track the sales of each salesman 24/7	----
	------	------	Generate reports upon request
	Location data	Sales data	Restaurants and cafes data only
	2 parts (GPS tracking device and web app)	2 parts (Web app and database)	2 parts (Desktop app and database)

Chapter3

Analysis & Design

3.1 System Overview

3.1.1 System Architecture

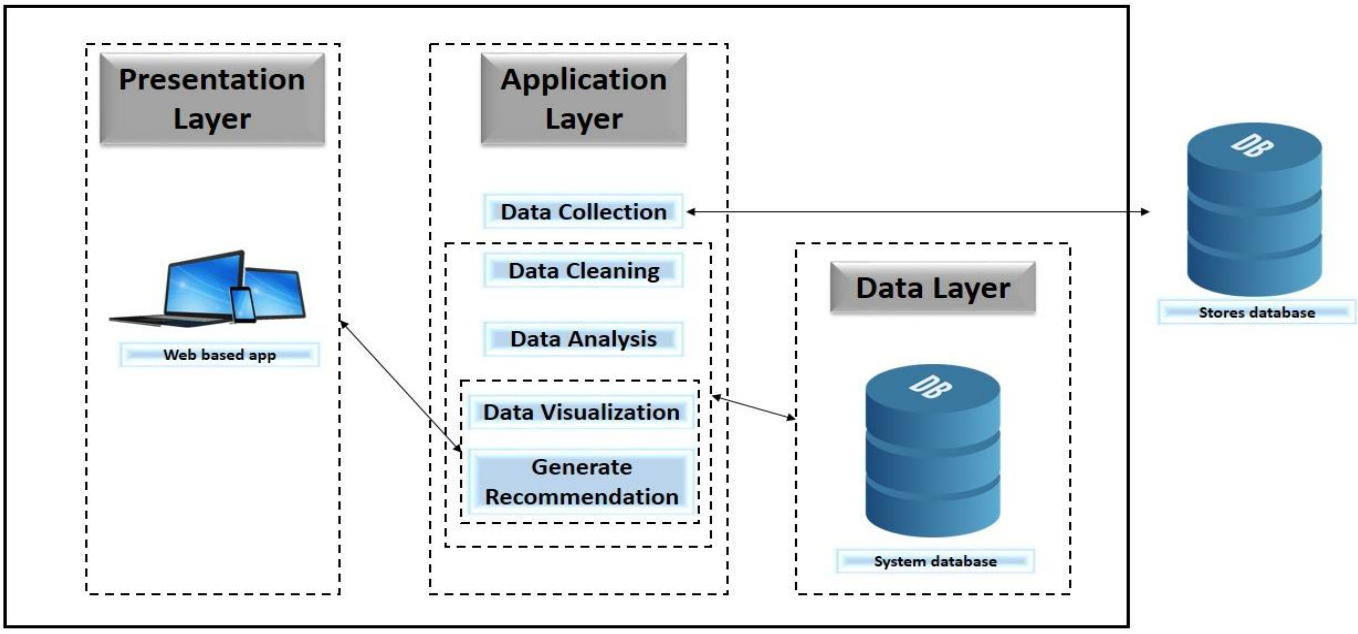


Fig 3.1 System Architecture

Scenario:

1. MSI web application admins will install the services & database on organization's server.
2. The admins will fill up organization branches details in database.
3. The system service will automatically internally work on tracking data through:
    a. Collecting data streams from organization branches.
    b. Filtering collected data stream.
    c. Applying Lossy counting algorithm on the filtered stream.
    d. Applying a recommendation technique on the extracted Knowledge.
4. Reports and statistics will be generated and available to the users to view and download through the web application in interactive manner.

3.1.2 Functional Requirements

- Access the system through Sign up/in authentications.
- Access the branch dashboard by the branch owner.
- Access & Collect the data streams from each branch.
- Filter the collected live data stream.
- Process & analyze the filtered stream to extract the hidden knowledge.
- Displaying statistics and reports in interactive way on dashboard.
- Recommend decisions based on extracted knowledge.
- Send notifications on each update in system.

3.1.3 Non-Functional Requirements

- **Usability:**
  System is easy to install as the developers installing it would not face any problems.
- **Reusability:**
  System is generic live data tracking system used for market data tracking and especially for market analysis.
- **Learnability:**
  System is easy to know and understand the system as the interface model of the system is so close to the user model.
- **Manageability:**
  System is manageable as the branch owner is able to monitor the performance his branch.
- **Security:**
  System is secured as no system use without sign up/in authentications and manger couldn't change his personal data without system admin approval.
- **Maintainability:**
  System could be easily maintained and updated without interrupting daily process.

3.1.4 System Users:

- **Intended Users:**

  System Users should be branch manager.

- **User's Characteristics:**

  User should have basic knowledge of computer use and especially web applications.

3.2 System Analysis and Design

3.2.1 User Case Diagram

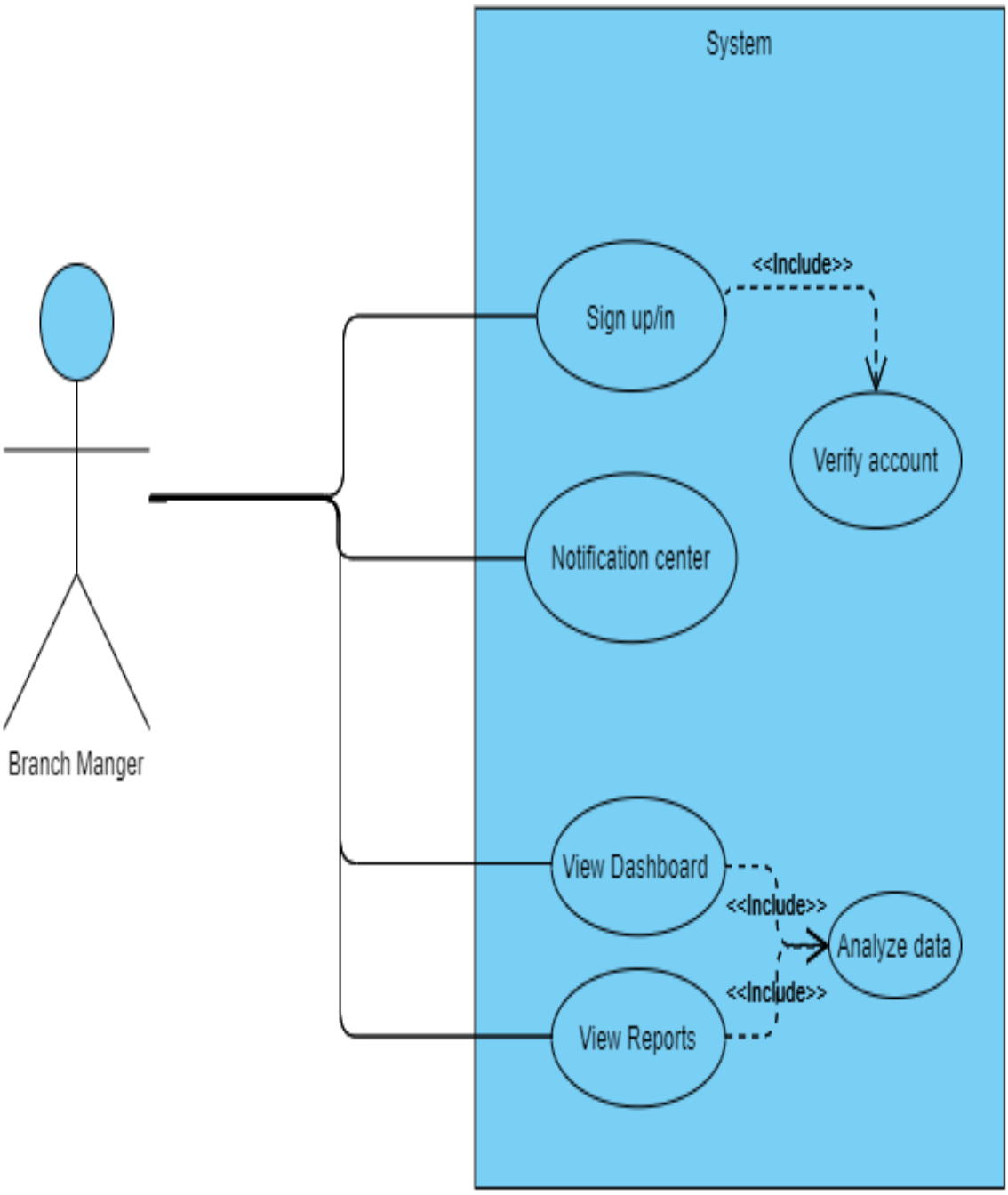


Fig 3.2 System Use case diagram

a. Sign up/in:

- **Description**

  The system shall allow the user to enter the system using his username and password.

  In case of not having an account branch manager could sing up using his information.

- **Input**

  User Name & Password.

  Or, branch manager information.

- **Output**

  Successful access for system or fail in case of wrong authentication information.

b. Notification Center:

- **Description**

  On the update of knowledge in dashboard or reports it rise the flag of notification.

- **Input**

  None.

- **Output**

  Detailed massage of the updated part.

c. Verify Account:

- **Description**

  Check that the entered authentication data is correct or not in case of sign in.

  This is included from sign in.

- **Input**

  Enter Username and password.

- **Output**

  Accept or deny of the entered username and password.

d. View Reports:

- **Description**

  Accessing the reports to give a detailed information about the extracted hidden knowledge.

  The reports are categorized to daily, monthly, yearly and quarter yearly reports for both sales and frequent items.

- **Input**

  None.

- **Output**

  View the reports of the given branch.

e. View Branch Dashboard:

- **Description**

  Accessing the dashboard that visualize the extracted knowledge.

  This dashboard gives the ability to see a Specific Branch progress according to the signed in manger.

- **Input**

  None.

- **Output**

  View the dashboard of the branch.

f. Analyze Data:

- **Description**

  This is included from both view branch dashboard and view reports.

  It collects the DataStream automatically from all branches then it make the needed filtration & analysis processes on the row data in the life stream to extract the desired knowledge for each branch.

  It also work to find the needed knowledge for each season to make the branch ready for this season before it starts.

- **Input**

  The live data stream

- **Output**

  Knowledge and information for each branch.

3.2.2 Flow Chart

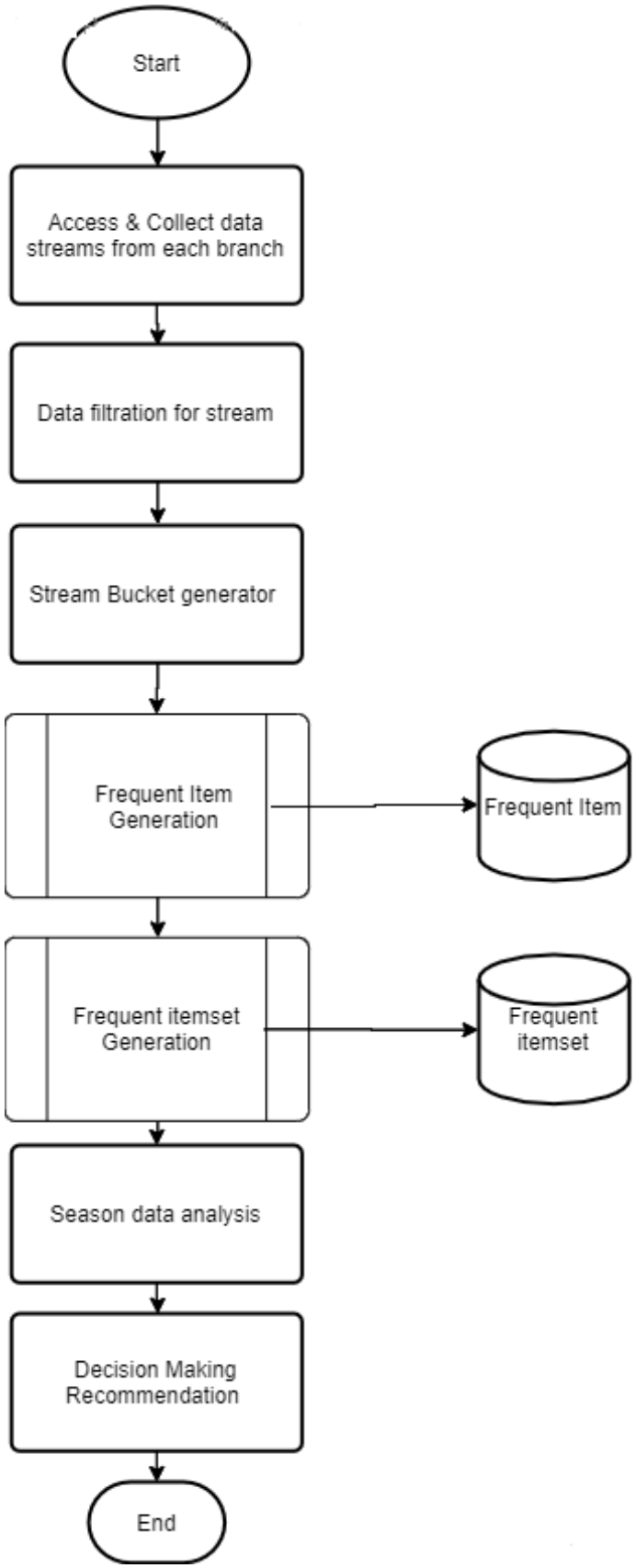


Fig 3.3 Flow Chart Part 1

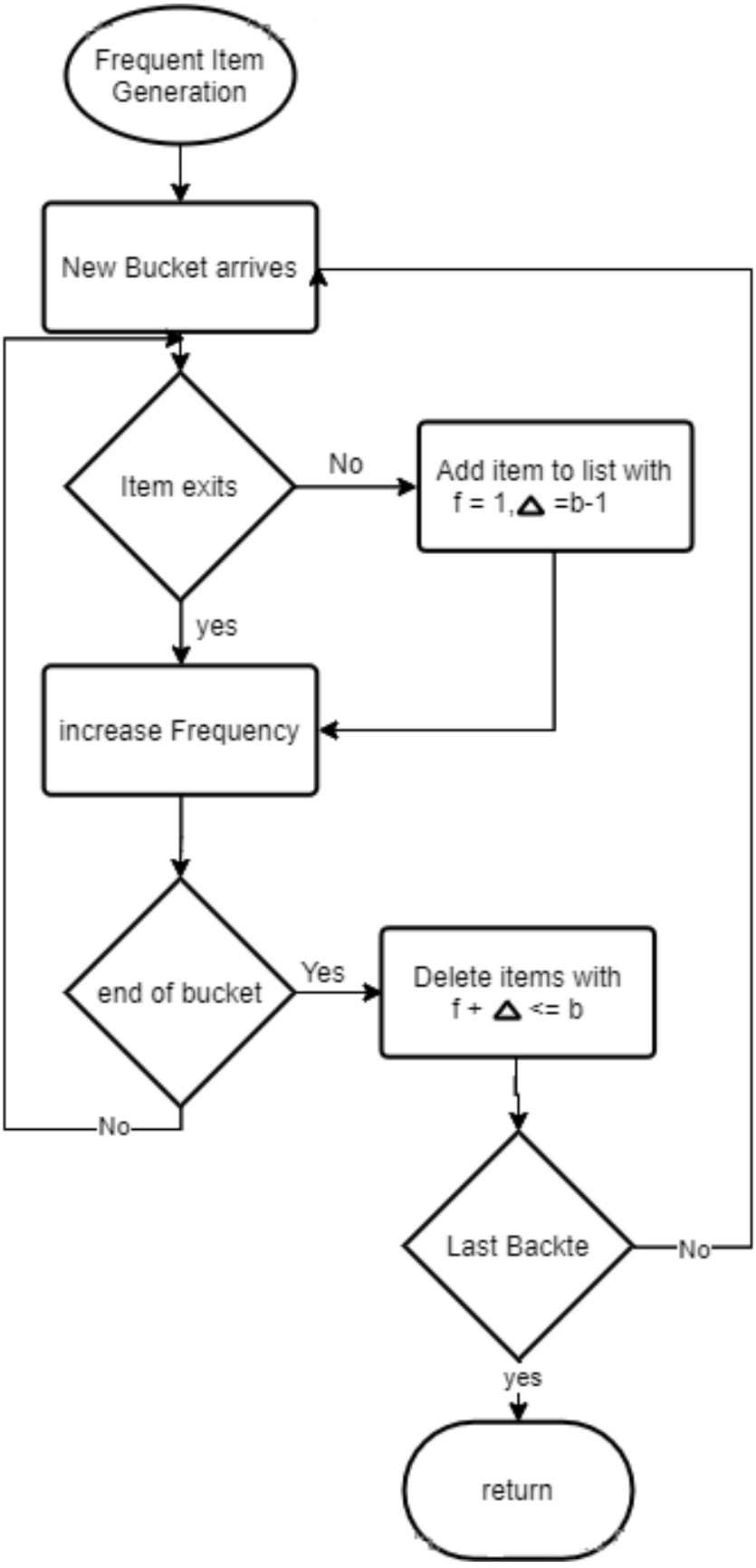


Fig 3.4 Flow Chart Part 2

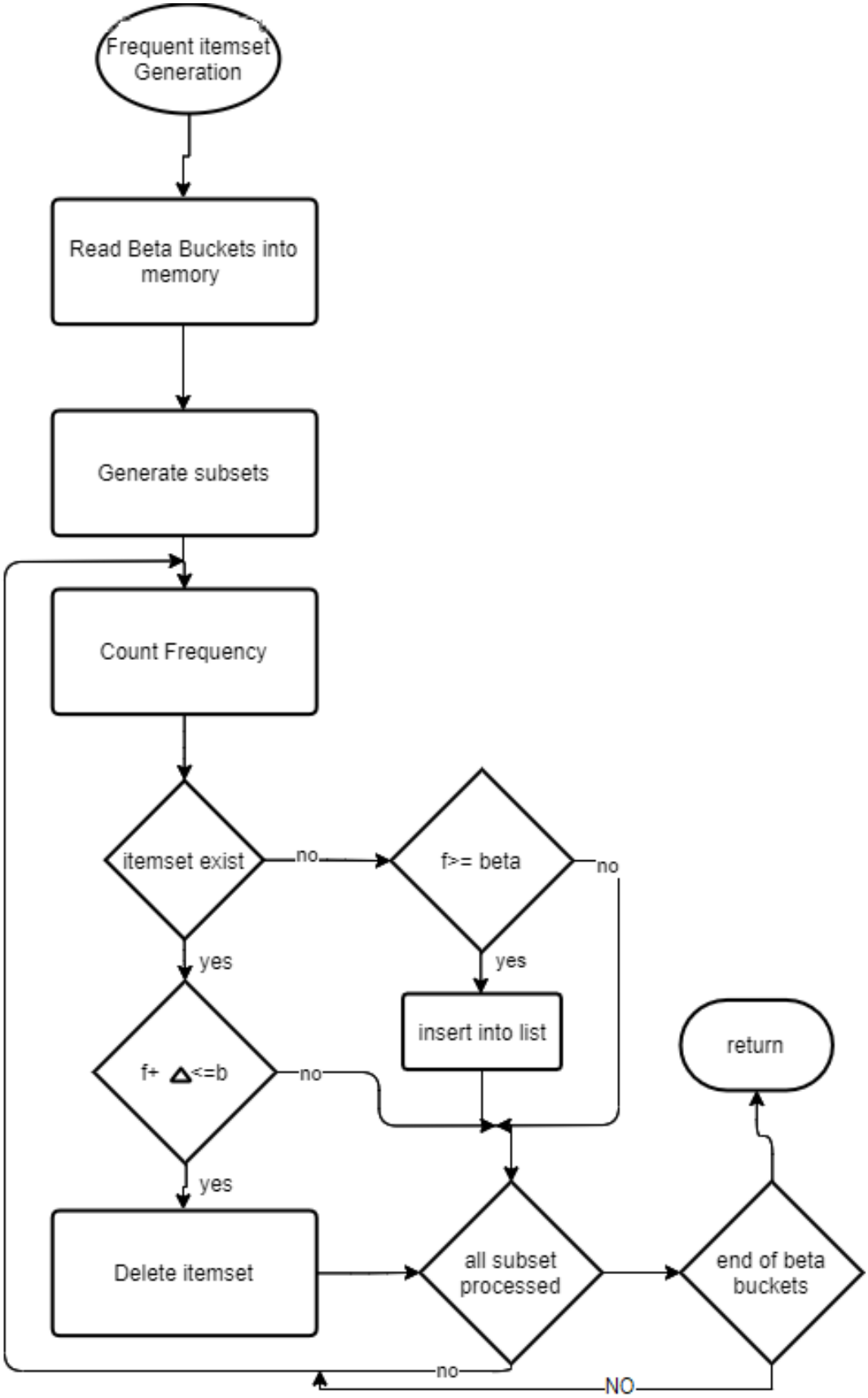


Fig 3.5 Flow Chart Part 3

3.2.3 Class Diagram

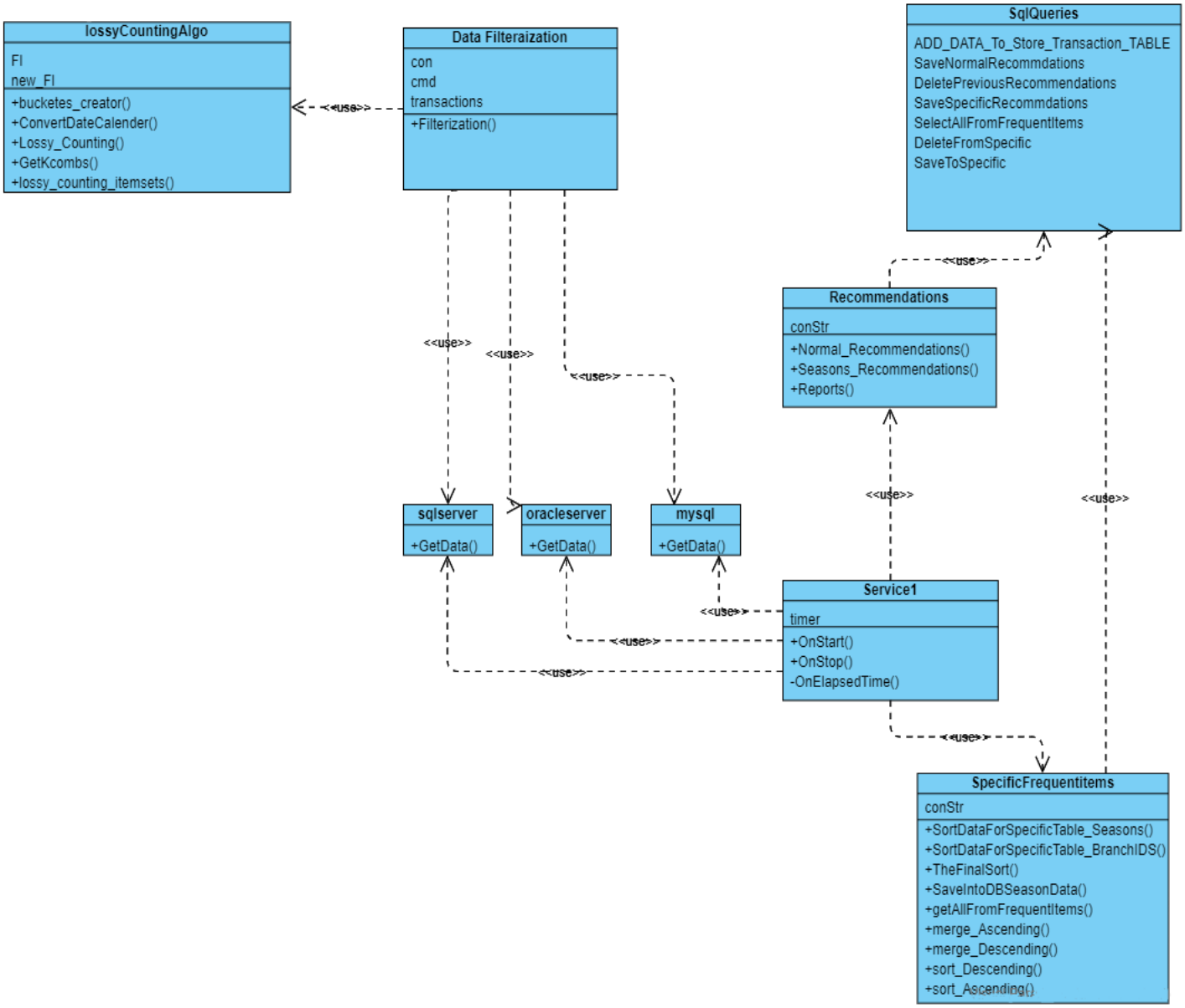


Fig 3.6 Class diagram

3.2.4 Sequence diagram

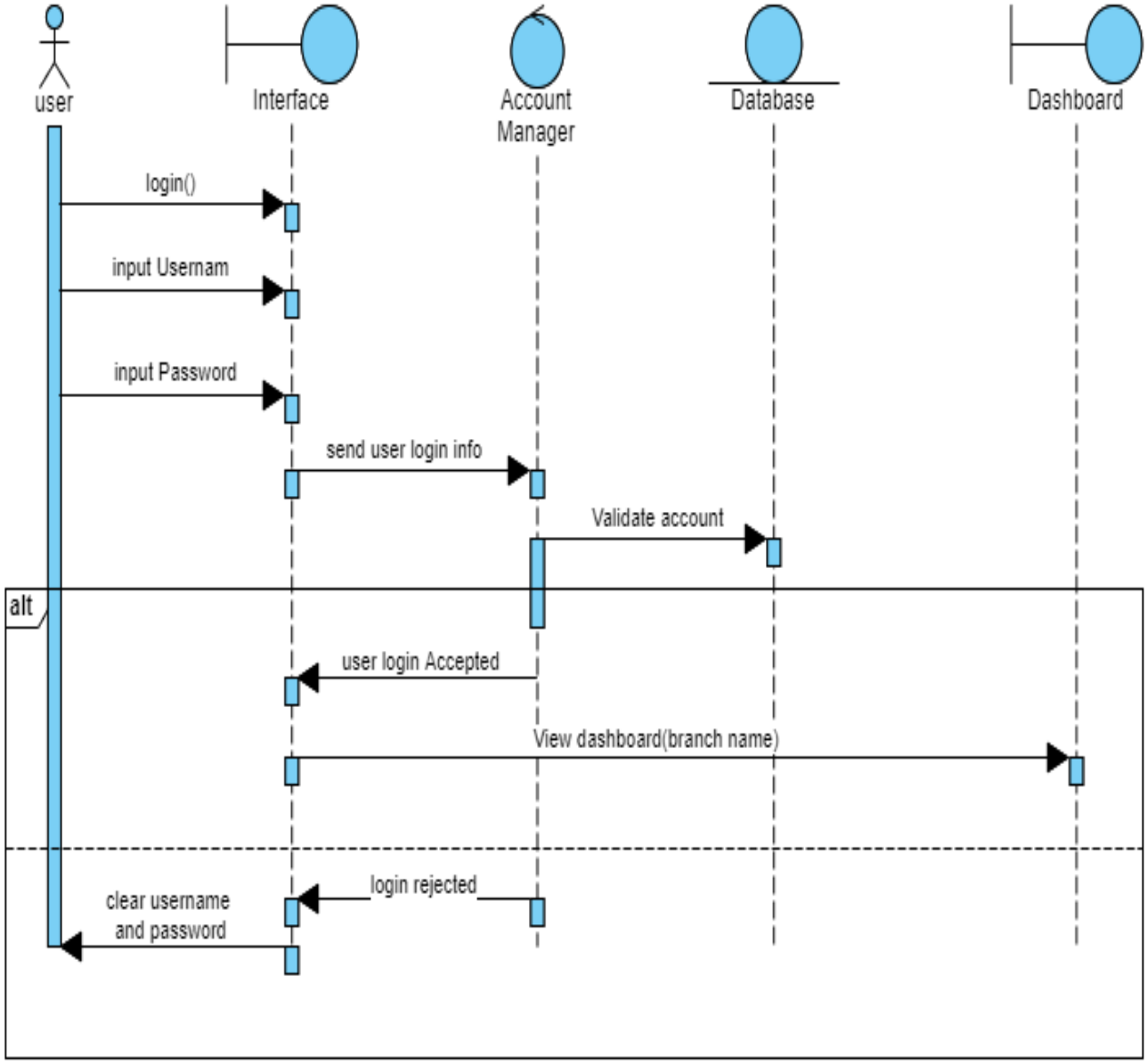


Fig 3.7 Sequence diagram (1) system login and dashboard view

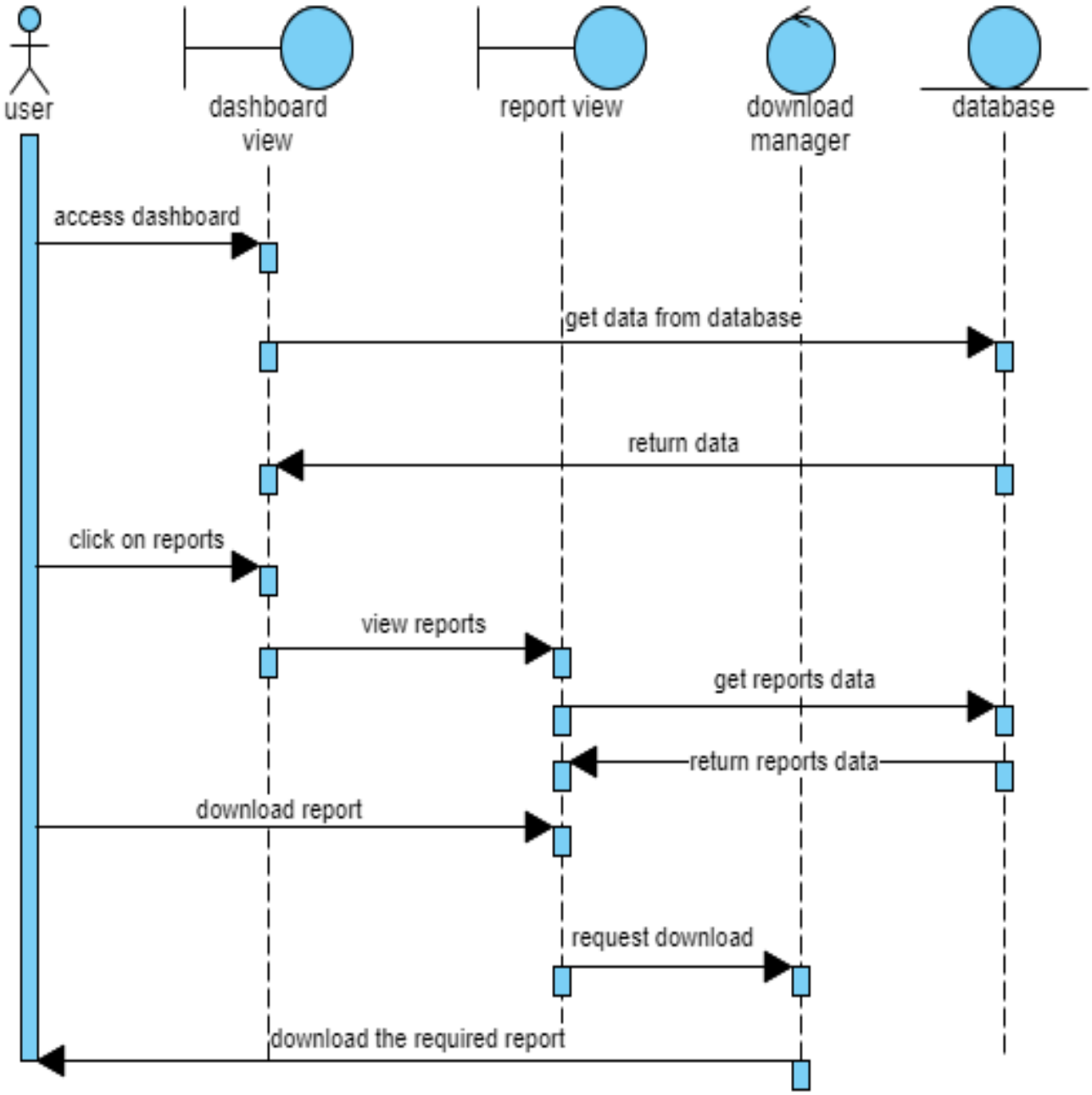


Fig 3.8 Sequence diagram (2) system generated reports view and download

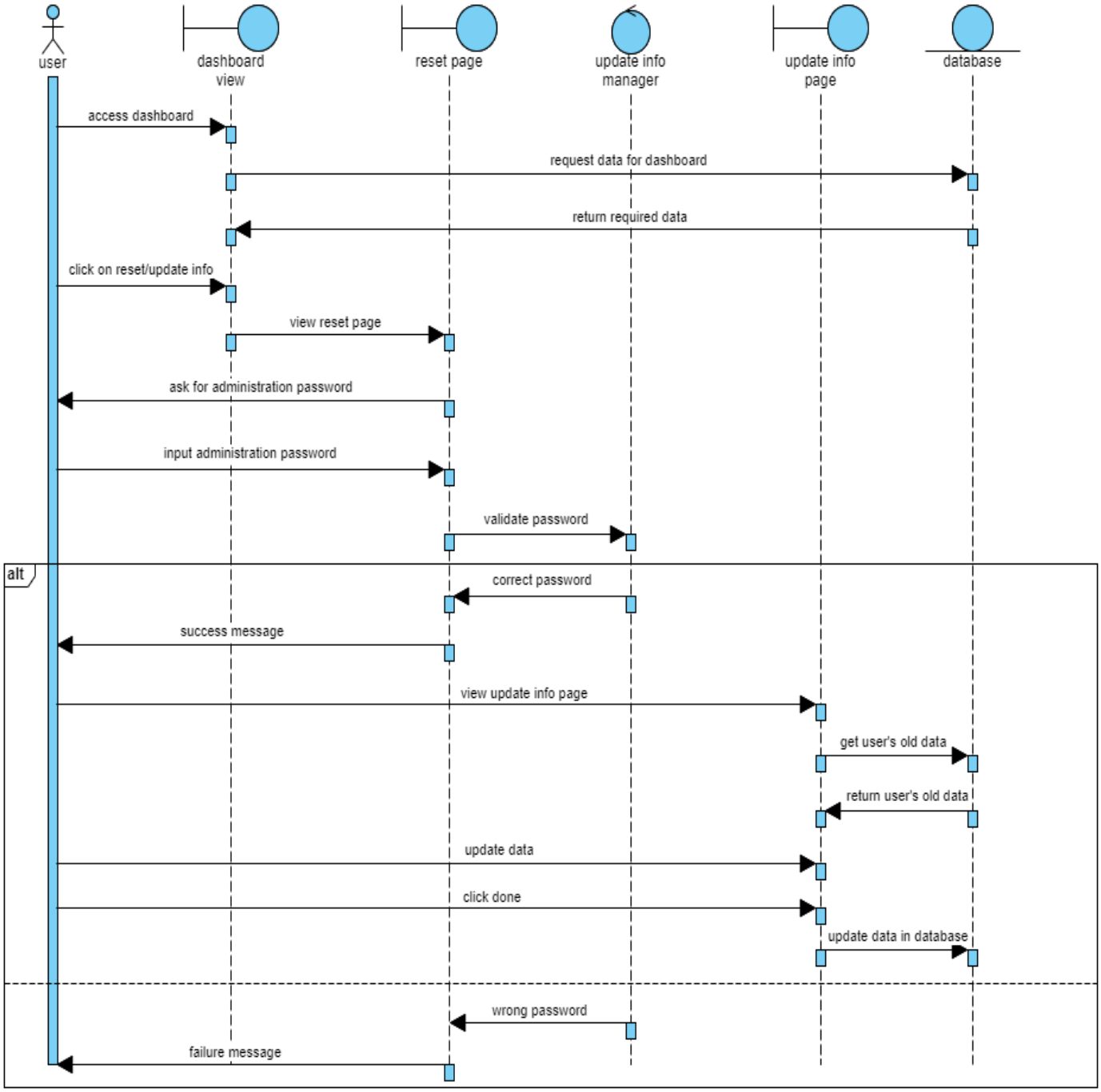


Fig 3.9 Sequence diagram (3) User access his personal information

3.2.5 Database diagram

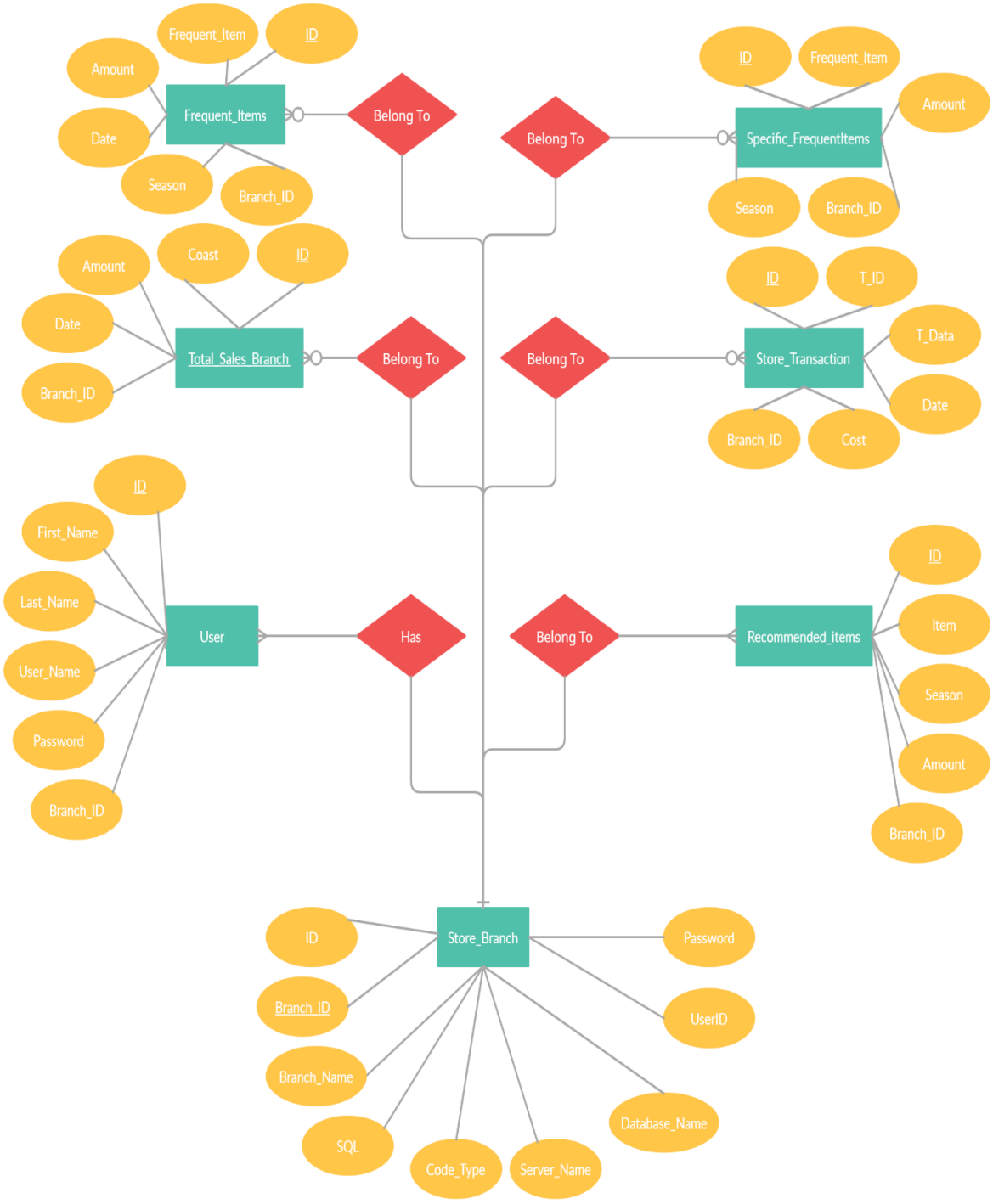


Fig 3.10 Entity Relationship Diagram (ERD)

Chapter 4

Implementation

4.1 Gathering Data

Transactional data is collected from all the branches of specific brand i.e. (Carrefour) every 12 hours which is done on many steps.

4.1.1 Gathering Data Steps

1. **Get All Branches from Database:** Getting from table **Store Branches** all the branches of the brand so we can get from it all the transactional data.
2. **Get Important Fields For Each Branch:** Getting For each branch from table **Store Branches** those fields:
   - **SQL:** The Sql command line that is performed on the database of the branch.
   - **Code Type:** Stores may be working by Sql server or oracle server or MySQL server so here it's saved which code of server that the store is working with.
   - **Server Name:** The name of the server of the branch which we are trying to connect with.
   - **Database Name:** The name of the database of the branch which we are trying to connect with.
   - **User ID:** The User ID of database of the branch which we are trying to connect with.
   - **Password:** The password of database of the branch which we are trying to connect with.

3. **Establishing a connection with each branch:** By the data we get from database we can now request a connection and wait for a response.
4. **Receiving stream of data for each branch:** If the request of accessing data is accepted then a stream of data of the previous 12 hours will be the response.

4.2 Data Filtration

After establishing a connection between the branch and our system, a stream of data is sent to **Data Filtration** class to filter the data.

4.2.1 Data Filtration Steps

1. **Get a stream of data:** Getting a stream of the transactional data of the branch.
2. **Check for each record:**
   - ✓ **Cost:** cost cannot be not existing or negative number.
   - ✓ **Date**: date should be the date of today and cannot be null value.
   - ✓ **Transaction Data**: transaction data cannot be an empty string.
3. **Send the stream of data to the lossy count algorithm:** after filtering the data, send it to the lossy counting algorithm so we can get the frequent items one set and two set.
4. **Save the filtered transactional data into Store Transaction table:** after filtering the data, we need to save it into store transaction table.

4.3 Generating the frequent items

After filtering the data, we pass it to the **lossy counting algorithm** so we can get the frequent one item sets and the frequent two items sets.

4.3.1 Lossy Counting Algorithm General Description:

4.3.1.1. Lossy Counting algorithm:

- The lossy count algorithm is an algorithm to identify elements in a data stream whose frequency count exceed a user-given threshold. The algorithm works by dividing the Data Stream into 'Buckets' as for frequent items, but fill as many buckets as possible in main memory one time. The frequency computed by this algorithm is not always accurate, but has an error threshold that can be specified by the user. The run time space required by the algorithm is inversely proportional to the specified error threshold, hence larger the error, the smaller the footprint.
- It was created by eminent computer scientists Rajeev Motwani and Gurmeet Singh Manku. This algorithm finds huge application in computations where data takes the form of a continuous data stream instead of a finite data set, for e.g. network traffic measurements, web server logs, clickstreams.

4.3.1.2. The general algorithm followed is outlined as follows:

- Step 1: Divide the incoming data stream into buckets of width $= 1/\epsilon$, where ϵ is mentioned by us as the error bound (along with minimum support threshold = σ).
- Step 2: increment the frequency count of each item according to the new bucket values. After each bucket, decrement all counters by 1.
- Step 3: Repeat – Update counters and after each bucket, decrement all counters by 1

4.3.1.3. Lossy Counting Algorithm Flowchart of generating 1-item frequent items:

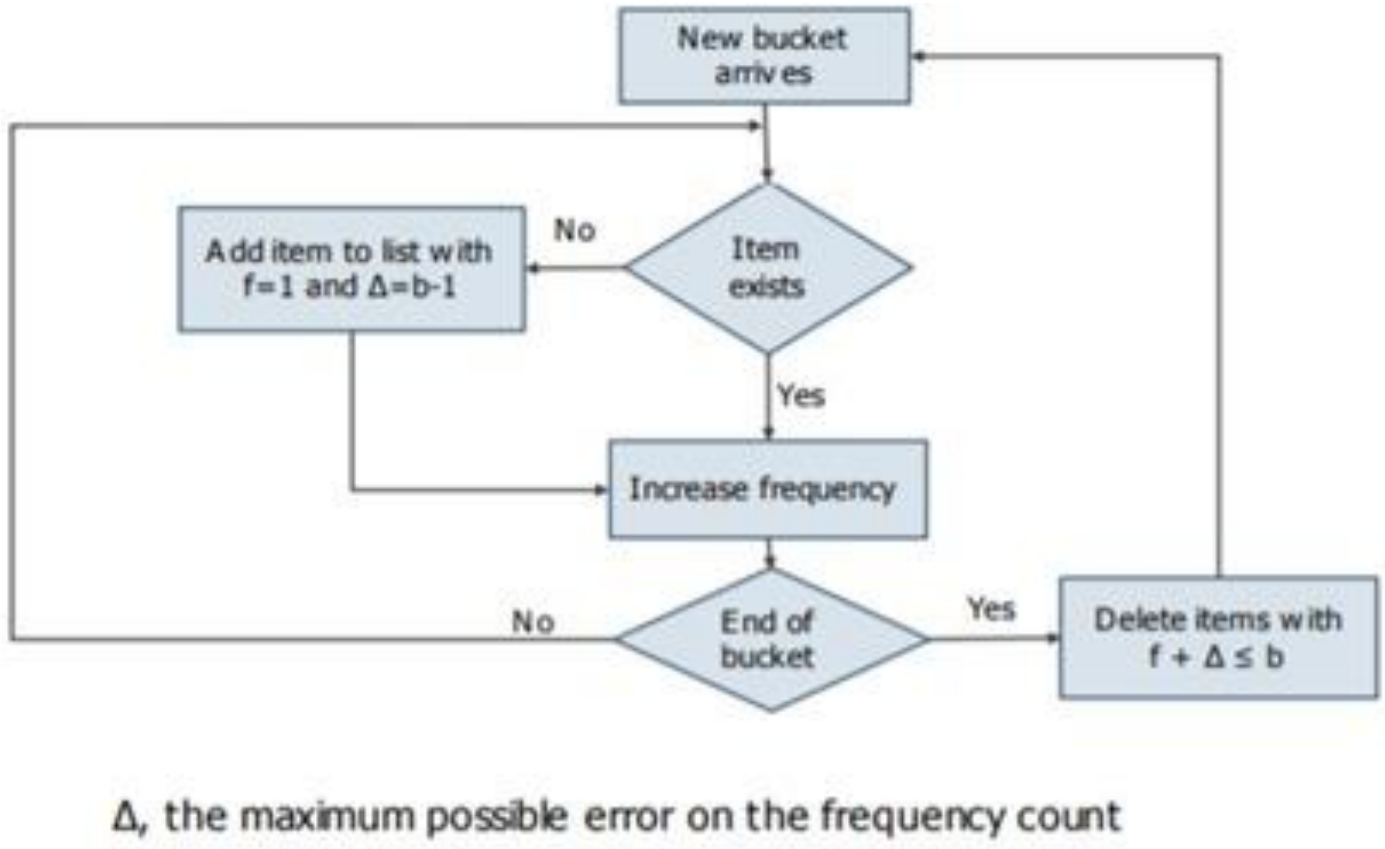


Fig 4.1 Lossy Counting Algorithm 1-item generation flowchart

4.3.1.4. Lossy Counting Algorithm Flowchart of generating 2-item frequent items:

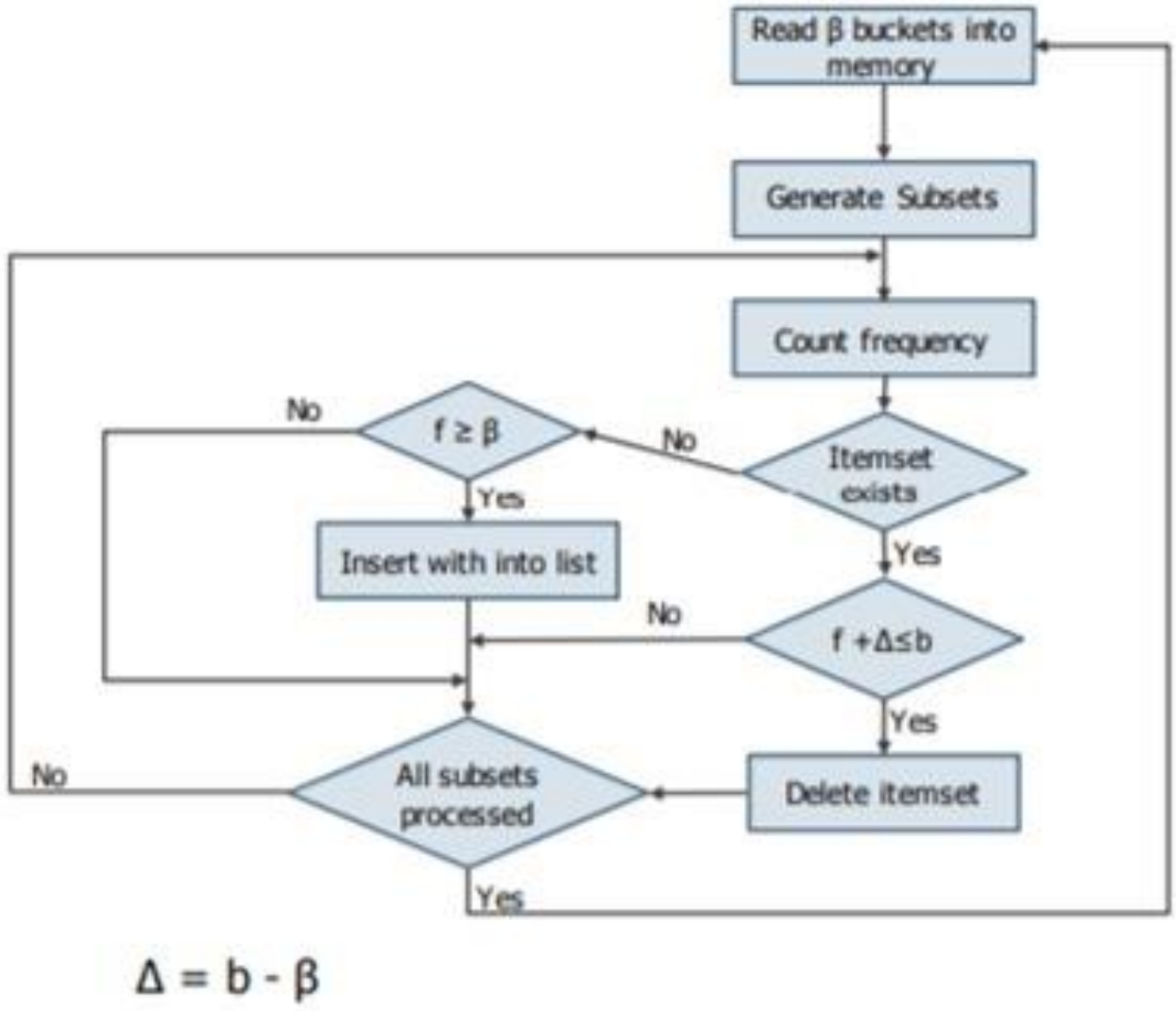


Fig 4.2 Lossy Counting Algorithm 2-item generation flowchart

4.3.2 Generating the (1-Item Frequent Sets) Steps

1. **Get a stream of data:** Getting a stream of the transactional data after the filtration step.
2. **Send the stream of the transactional data into (Buckets Creator Function):** This function is responsible for dividing the stream of data into buckets of width $\omega = 1/\epsilon$, where ϵ is set by us as the error bound (along with minimum support threshold =σ).
3. **Send each bucket to (Lossy Counting Function):** This function takes as **input** the bucket, min frequency, error bound and bucket number.

Lossy Counting Function Steps:

1) Loop on the bucket row by row (Row is composed of each transactional data row of the stream), The row format is as follows:

ID	T_ID	T_Data	Date	Cost	Branch_ID

Each T_Data of each row can be as example [‘milk’,’coffee’,’tea’]

2) Check if each item i.e. (‘milk’) of the T_Data is in the frequent list which we initialized to null.
   - **If the item is in the list, add 1 to its frequency in the list.**
   - **If the item is not in the list, add the item to the frequency list with initialized frequency=1 and its delta**

***Each Item in the frequent item list has a delta value which is calculated by the equation:**

$$\Delta = \boldsymbol{b} - \mathbf{1}$$

Delta is Δ , Bucket number is **b**

3) Repeat step 2 till the end of the bucket.
4) After finishing the frequent items list generation, we loop on this list to check an important condition for the lossy counting algorithm.

$$f + \Delta \leq b$$

*__f__ is the frequency of each item, Δ is delta, __b__ is the bucket number:

- If this condition is true then that item in the frequent items list will be deleted, because then that item will be false negative i.e. that item is not frequent and won't be frequent.
- If this condition is false then that item stays in the frequent items list.

4. Save the 1-item frequent sets into the Frequent Items table:

After finishing the lossy counting algorithm on the stream of the transactional data, Save the 1-item frequent sets into the database.

4.3.3 Generating the (2-Item Frequent Sets) Steps:

1. **Get a stream of data:** Getting a stream of the transactional data after the filtration step.
2. **Send the stream of the transactional data into (Buckets Creator Function):** The 2nd part of this function is responsible for dividing the stream of data into **beta** buckets of width ($\omega = 1/\epsilon$ * beta bucket size), where ϵ is set by us as the error bound (along with minimum support threshold =), beta bucket size is 5 which memory can hold.
3. **Send each bucket to (Lossy Counting Item-sets Function):** This function takes as **input** the beta bucket, min frequency, error bound, beta bucket number and beta bucket size.

Lossy counting (2-item Frequent sets) Steps:

1) Loop on the bucket row by row (Row is composed of each transactional data row of the stream), The row format is as follows:

ID	T_ID	T_Data	Date	Cost	Branch_ID

Each T_Data of each row can be as example [‘milk’,’coffee’,’tea’]

2) Generate subsets for each row (This step is done by function **GetKCombs** () takes a list of items and returns all possible combinations of this list).

i.e. (if T_Data= [‘milk’,’coffee’,’tea’]) Then subsets will be:

- [‘milk’, ‘coffee’]
- [‘milk’, ‘tea’]
- [‘coffee’, ‘tea’]

3) Save all possible subsets with its branch id and season.
4) Loop on the subsets for counting its frequency in each branch.
5) Check if there is a repeated subset i.e. [‘milk’, ‘coffee’] and [‘coffee’, milk’], if there, delete all the repeated ones and leave only one subset with the addition of frequency of all the repeated subset.

6) Check if subsets is in frequent item list:

- If the subset exist: then check if $\boldsymbol{f + \Delta \leq b}$
  - ✓ **If condition is true then remove the subset**
  - ✓ **If the condition is false then add the subset to the frequent item list with an initialized frequency equals to 1 and delta equal to** $\Delta = b - \beta$

- If the subset does not exist: add it to the frequent item list with an initialized frequency equals to 1 and delta equal to

$$\Delta = b - \beta$$

4. **Save the 2-item frequent sets into the Frequent Items table:** After finishing the lossy counting algorithm on the stream of the transactional data, Save the 2-item frequent sets into the database.

4.4 Improving Response Time in our Web App

4.4.1 Introduction to Response Time

Response time is the total amount of time it takes to respond to a request for service. That service can be anything from a memory fetch, to a disk IO, to a complex database query, or loading a full web page. Ignoring transmission time for a moment, the response time is the sum of the service time and wait time.

4.4.2 Importance of Response Time

The Importance of Response Time in the Service Industry: The old saying "time is money" is prominent in the service industry. A quick response is valued, as long it is a quality one. Managers know the key to profits in their service businesses depends on customer satisfaction and productivity. Develop a smooth running system with highly efficient employees to produce an impressive time response that gives your company a competitive advantage.

4.4.3 How we reduced response time in our app

We thought of a way to prepare our database so we could retrieve from it specific rows and not going along all the database, this will save time and money.

4.4.4 How we reduced response time in our service code

1) Every Year and For each season (Summer, Winter, Ramadan, Christmas) in every branch we save top 10 most sold items so we could make recommendations based on this later.

4.4.4.1. Formatting Specific Frequent Items Table for Seasons Steps

In table **frequent items** we save1-item sets and 2-item sets for each data stream we're having, and this is a very big amount of data, so in table **Specific frequent items** we save only in descending order the top 10 most sold items in each branch in each season by the next steps:

- Get all the data from table **Frequent Items** in a dataset.
- Sort the dataset ordered by **Season Name.**
- Sort in each season the **branches IDS** in ascending order "By **Merge Sort Algorithm**".
- Sort in each branch the most sold Frequent Items in descending order "By **Merge Sort Algorithm**".
- Save only the top 10 most sold frequent items for every branch in every season, in table **Specific Frequent Items.**

2) Every Day in every branch we save top 10 most sold items so we could make Normal/Daily recommendations based on this later.

4.4.4.2. Formatting Specific Frequent Items Table for Daily FI Steps

In table **frequent items** we save1-item sets and 2-item sets for each data stream we're having, and this is a very big amount of data, so in table **Specific frequent items** we save only in descending order the top 10 most sold items in each branch every day by the next steps:

- Get all the data from table **Frequent Items** in a dataset
- Sort the dataset ordered by **Season Name** aka "Normal"
- Sort in each season the **branches IDS** in ascending order "By **Merge Sort Algorithm**"
- Sort in each branch the most sold Frequent Items in descending order "By **Merge Sort Algorithm**"
- Delete From table **Specific Frequent Items** the items that have season= normal, so we could save only the frequent items of today in it.
- Save only the top 10 most sold frequent items for every branch of today, in table **Specific Frequent Items.**

4.4.5 Merge Sort Algorithm

Merge Sort is a Divide and Conquer algorithm. It divides the input array into two halves, calls itself for the two halves, and then merges the two sorted halves. The merge () function is used for merging two halves. The merge (arr, l, m, r) is a key process that assumes that arr [l...m] And arr [m+1...r] are sorted and merges the two sorted sub-arrays into one.

Best &Average &Worst Complexity: n* log (n)

*Merge Sort Flow Chart

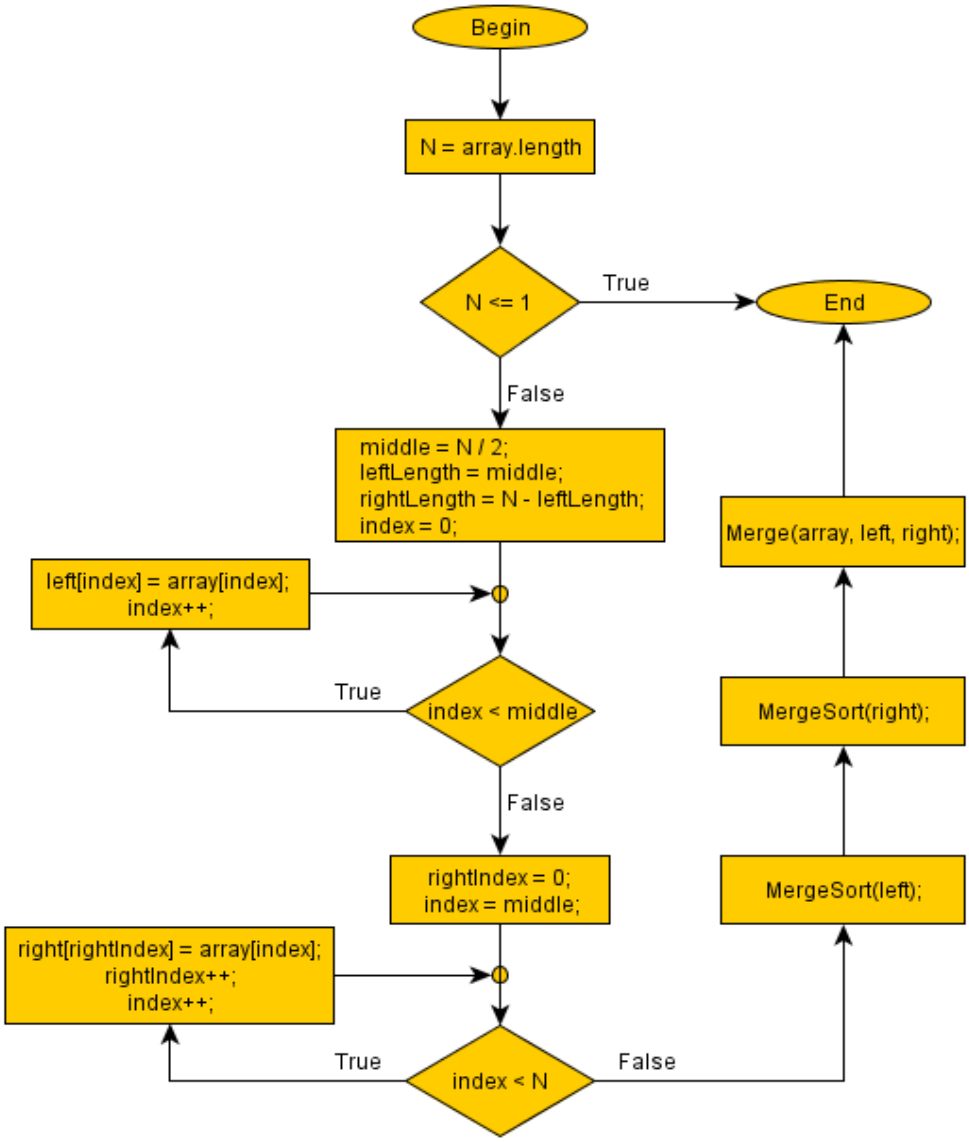


Fig 4.3Merge Sort flow chart

4.4.6 Recommendations System

After we saved the top 10 most sold frequent items for each branch in every season (normal/daily, summer, winter, Ramadan, Christmas) **Sorted,** we save it now into the **Recommendations** table.

- **For Season Normal/Daily:** we save the 10 top most sold frequent items for each branch.
- **For the rest of the seasons:**
    A. Search for the repeated frequent items over the previous years.
    B. Get the average of the amount that the stores needed to be provided with.
    C. Put the calculated average amounts for each item in each season for each branch in table **Recommendations.**

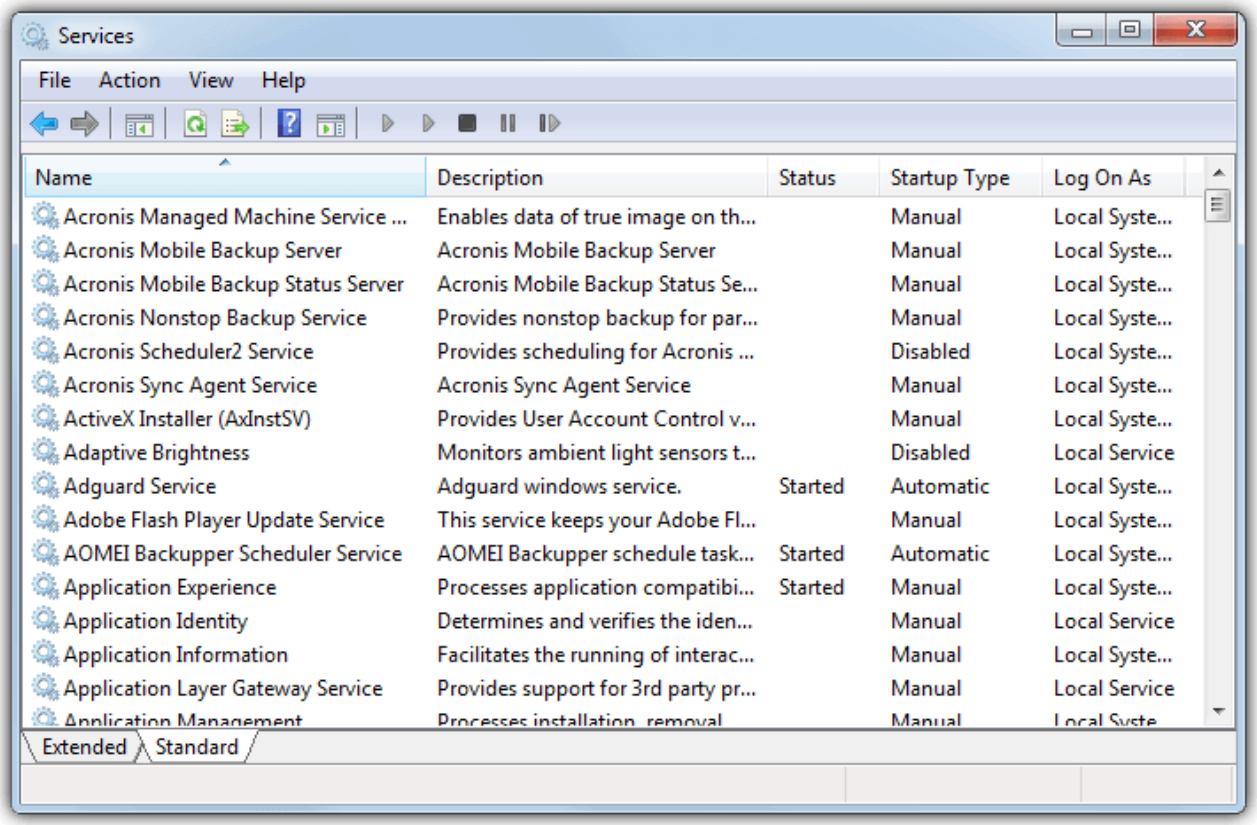
Chapter 5

User Manual

5.1 Windows services

5.1.1 Windows services idea:

- Windows Services are a core component of the Microsoft Windows operating system and enable the creation and management of long-running processes.
- Unlike regular software that is launched by the end user and only runs when the user is logged on, Windows Services can start without user intervention and may continue to run long after the user has logged off. The services run in the background and will usually kick in when the machine is booted. Developers can create Services by creating applications that are installed as a Service, an option ideal for use on servers when long-running functionality is needed without interference with other users on the same system.

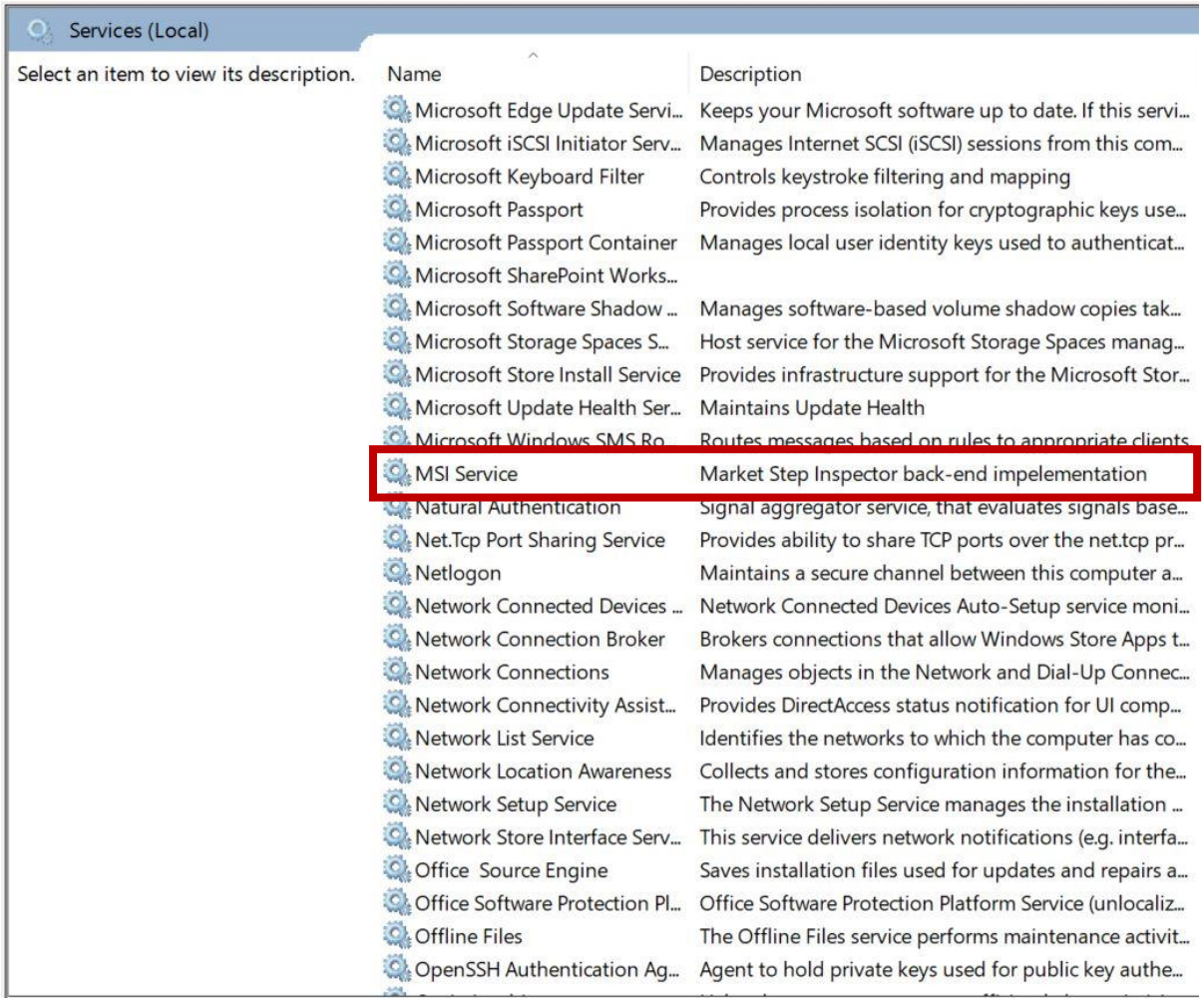


- **Fig 5.1 windows service example**

Points of comparisons	Regular Applications	Windows Services
Launch Mechanism	A regular application is manually launched by the end user from the desktop or Start Menu. Examples include web browsers, document editing software and PDF readers.	Windows Services start when the machine is switched on.
User interface	Regular applications have a user interface	Windows Services do not have a user interface; they run in the background and the user does not directly interact with them.
Multiple instance	Regular applications can allow multiple copies if several users are logged into the same machine.	Only one instance of a Windows Service runs on a device.
Turn off statues	A regular application will stop when you log out.	A windows service won't exit when you or anyone else logs off the PC.

5.1.2 Our windows service installation:

1. Start up the command prompt (CMD) with administrator rights.
2. Type [your **windows service Full path** to exe]
3. Press return and that's that!



- **Fig 5.2 MSI Service**

5.2 Database

5.2.1 Database Idea:

- A database is an organized collection of structured information, or data, typically stored electronically in a computer system. The data can then be easily accessed, managed, modified, updated, controlled, and organized. Most databases use structured query language (SQL) for writing and querying data.

5.2.2 Our database attachment:

1. In SQL Server Management Studio Object Explorer, connect to an instance of the SQL Server Database Engine, and then click to expand that instance view in SSMS.
2. Right-click **Databases** and click **Attach**.

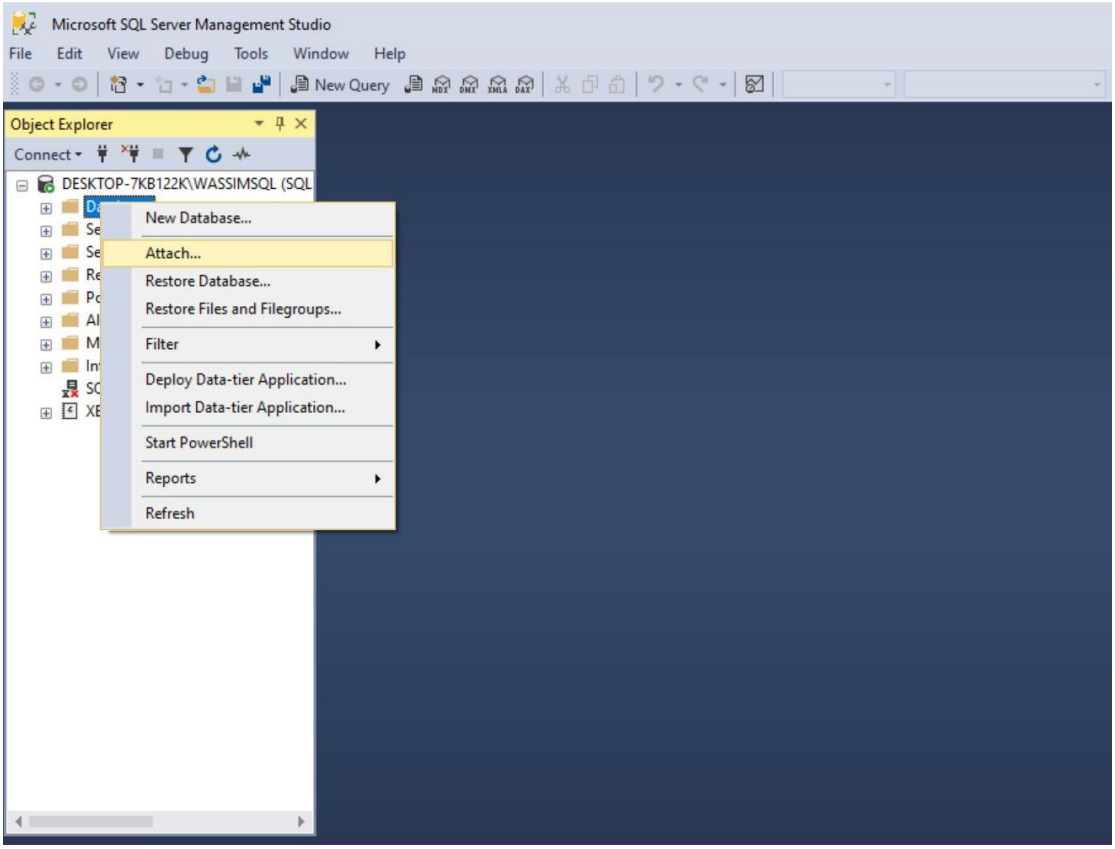


Fig 5.3 Database Attachment

3. In the **Attach Databases** dialog box, to specify the database to be attached, click **Add**; and in the **Locate Database Files** dialog box, select the disk drive where the database resides and expand the directory tree to find and select the (.mdf) file of the database.

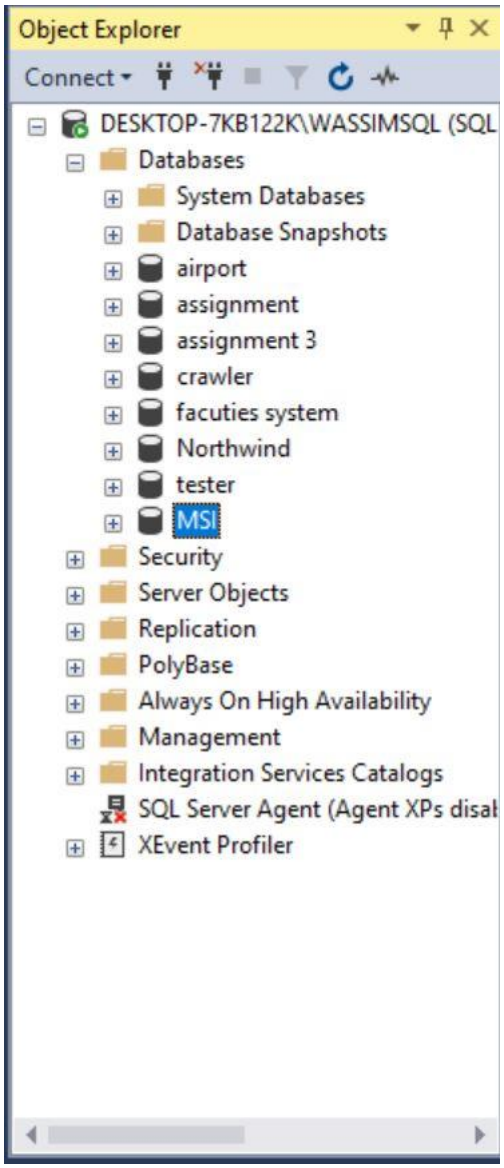


Fig 5.4 MSI database

5.3 Web Application

5.3.1 Web application idea:

- A **web application** (or **web app**) is application software that runs on a web server, unlike computer-based software programs that run locally on operating systems.

5.3.2 How to use our web application:

1. Run the web API (.exe) then the web app using any web engine.
2. Sign in / up to your store.

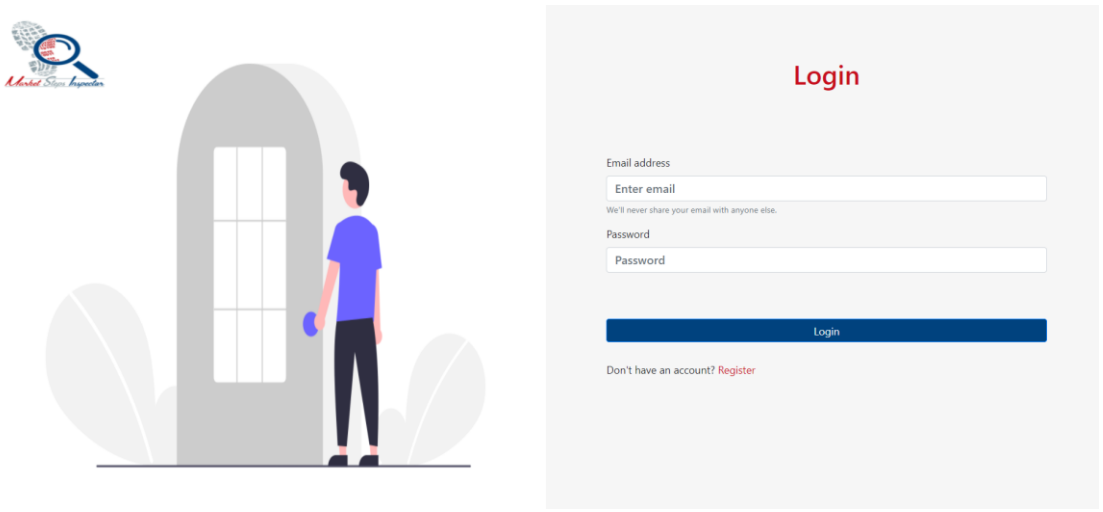


Fig 5.5 MSI Web App Login

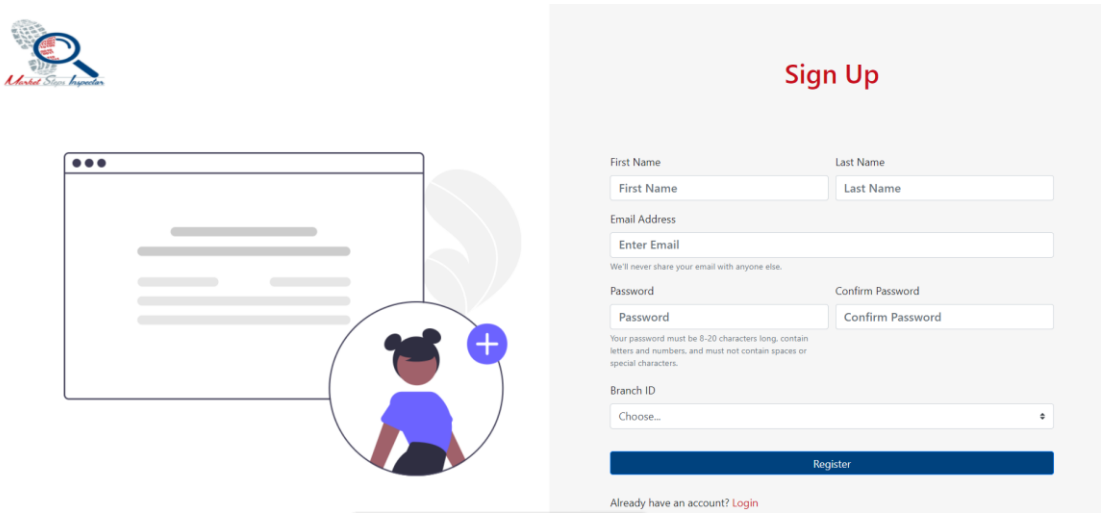


Fig 5.6 MSI Web App Sign Up

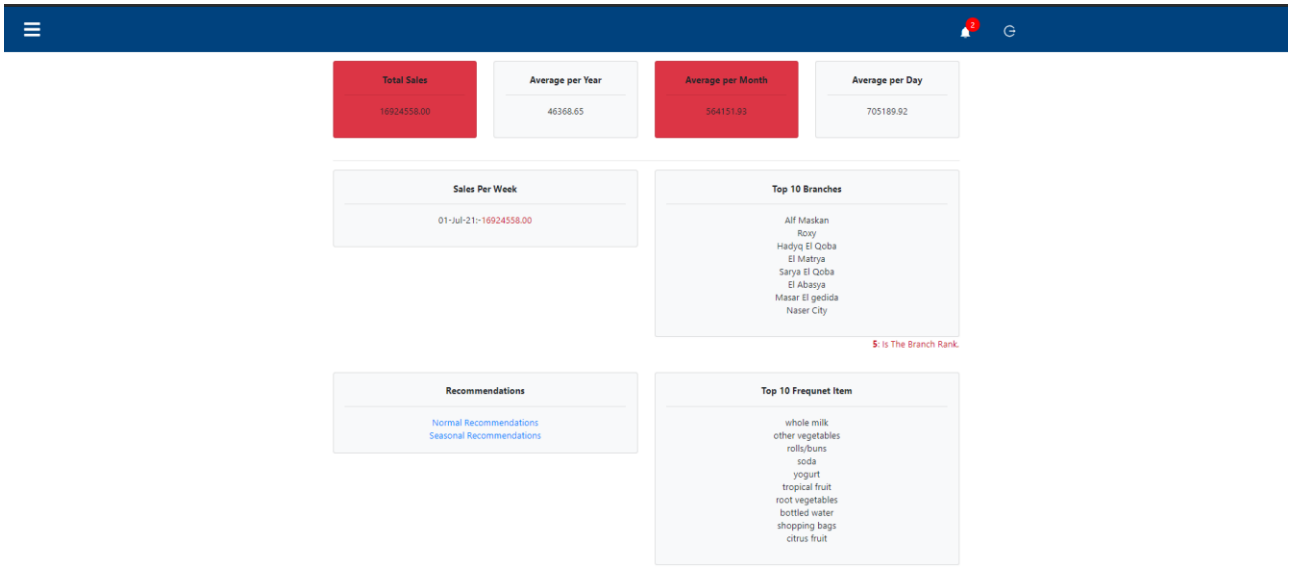


Fig 5.7 MSI Web App Dashboard

3. Accesses the sales report from navigation bar, then click sales report by identifying (daily, monthly, yearly, quarter yearly).

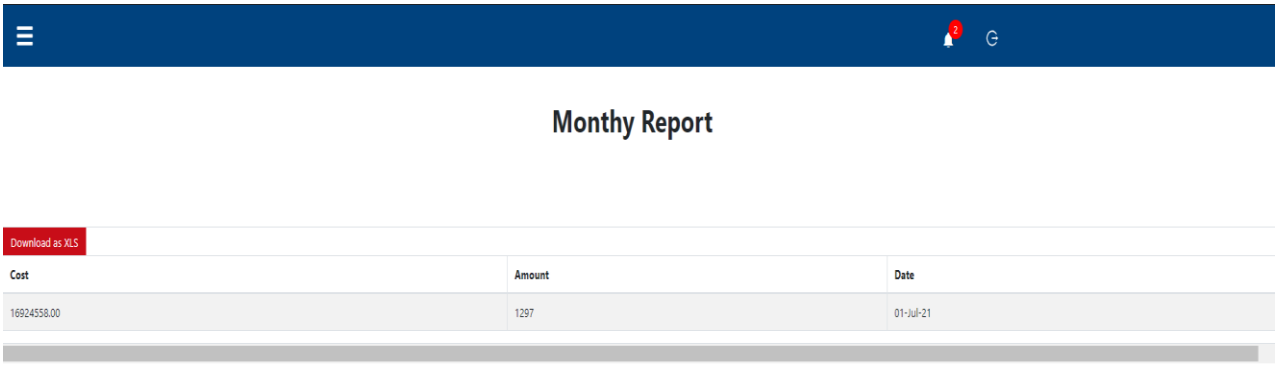
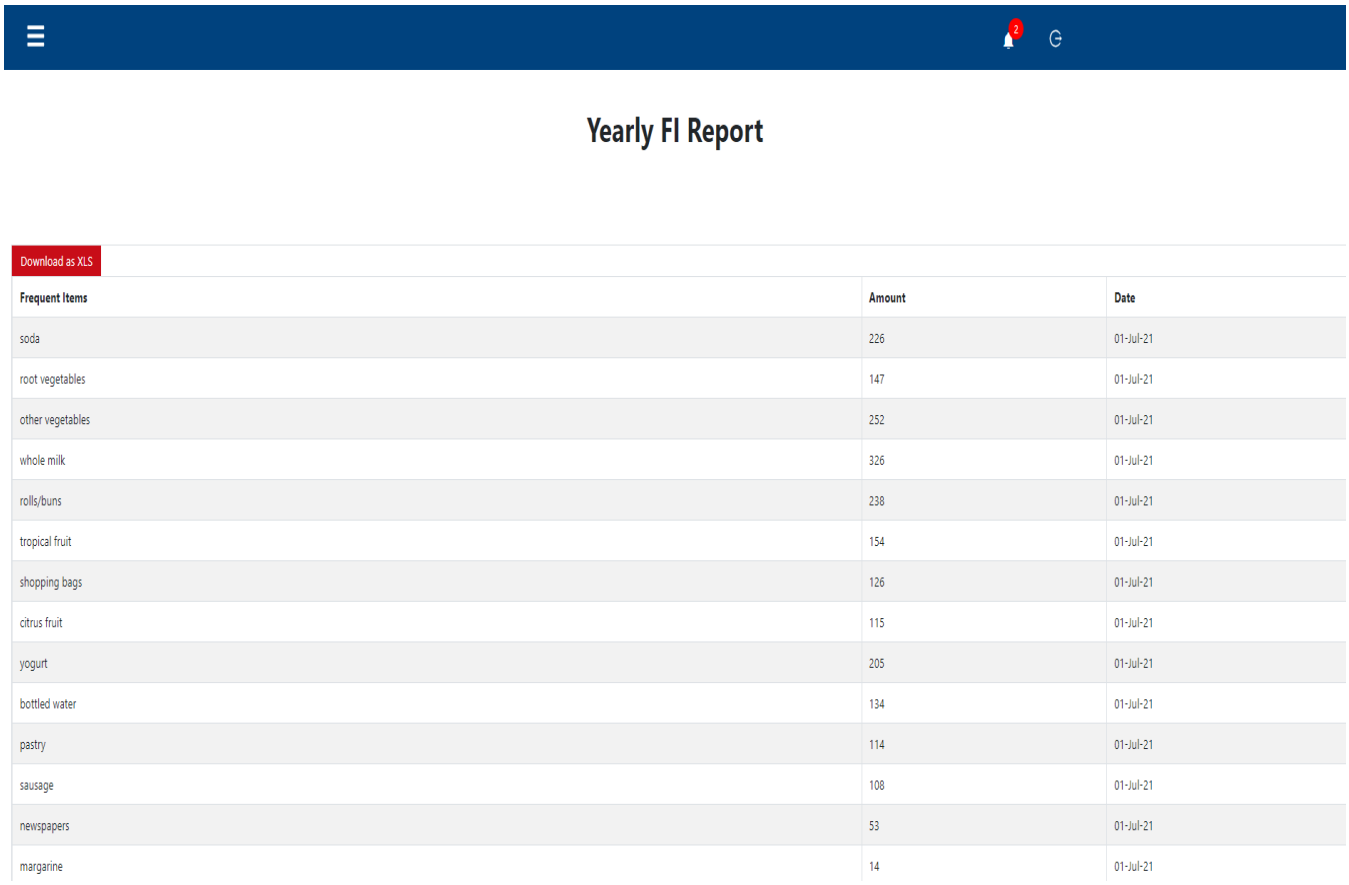


Fig 5.8 MSI Web App Sales Report

4. Accesses the Frequent item report from navigation bar, then click FI report by identifying (daily, monthly, yearly, quarter yearly).



Yearly FI Report

Download as XLS

Frequent Items	Amount	Date
soda	226	01-Jul-21
root vegetables	147	01-Jul-21
other vegetables	252	01-Jul-21
whole milk	326	01-Jul-21
rolls/buns	238	01-Jul-21
tropical fruit	154	01-Jul-21
shopping bags	126	01-Jul-21
citrus fruit	115	01-Jul-21
yogurt	205	01-Jul-21
bottled water	134	01-Jul-21
pastry	114	01-Jul-21
sausage	108	01-Jul-21
newspapers	53	01-Jul-21
margarine	14	01-Jul-21

Fig 5.9 MSI Web App Frequent Item(s) Report

5. To see recommendations from dashboard, click on normal or specific recommendations, (if we are in winter, summer, Ramadan, Christmas use specific recommendation, else use normal recommendations).

Fig 5.10 MSI Web App Recommendation

6. To log out, click on the log out icon on the right top of the screen. //pic

Fig 5.11 MSI Web App Log out

Chapter 6

6.1 Conclusion

The rapid growth of data over time have made a great challenge in managing and processing those data in markets the managers of very big store branches have getting hard, limited times and effort to follow up his branches profit, frequent items and top sales products that affects his decisions for more revenue and popularity.

Our project is a web application that is used by managers for easy following up his branches profits, frequent items and reports daily, monthly, yearly and quarter yearly along with some recommendations for seasons to help in making decisions. Depending on 3 specific modules:

1. Live data tracking windows services (c#):

Collecting data from the data centers, data services, databases, etc....., then applying the needed analysis and data techniques to filter it and save it in our database.

2. Live data tracking database (Sql server):

This is our system database that has been retrieved by the live data tracking service filtered to extract from it the needed statistics and reports.

3. Live data tracking web based application (ASP.NET web app):
   - It is the web application that contains the dashboard including the most Statistics that is done by the service and stored in the db.
   - Tabs and Side menus to see the reports and statistics of the deeper analysis.
   - Controls and credentials on (the service running and stopping, on the used database, and off course all of this under login/register credentials).

Then integrating the result of all the previous modules to show the better way to show the managers his branches data aiming for less effort, time and making better decisions and higher goals.

6.2 Future Work

- **Our future work mainly on:**
    1. Transfer our database to cloud for more reliable access.
    2. Making a cross platform project not only a web application.
    3. Increase employee's performance tracking.
    4. Make an aggregated dashboard that include all branches in it.
    5. Adding a way of communication between the store owners and another store owner like a chat messenger.
    6. Supporting all types of databases.

References

1. https://www.cs.waikato.ac.nz/~abifet/SAC2018/ws_1.html
2. https://www.confluent.io/learn/data-streaming
3. https://www.omnisci.com/technical-glossary/real-time-analytics#:~:text=Real%20time%20analytics%20lets%20users,for%20making%20real%2Dtime%20decisions
4. https://www.tibco.com/blog/2013/06/28/13-cool-data-quotes/#:~:text=Why%20We%20Love%20Data,Geoffrey%20Moore%2C%20author%20and%20consultant
5. https://www.clearrisk.com/risk-management-blog/challenges-of-data-analytics-0
6. https://www.confluent.io/learn/data-streaming/
7. https://www.linkedin.com/pulse/fascinating-examples-show-why-streaming-data-real-time-bernard-marr#:~:text=Log%20files%2C%20e%2Dcommerce%20purchases,time%20streaming%20data%20is%20created
8. https://micvog.com/2015/07/18/frequency-counting-algorithms-over-data-streams/
9. https://en.wikipedia.org/wiki/Lossy_Count_Algorithm
10. http://www.mathcs.emory.edu/~cheung/Courses/584/Syllabus/07-Heavy/Manku.html
11. https://en.wikipedia.org/wiki/Response_time_(technology)#:~:text=Response%20time%20is%20the%20total,service%20time%20and%20wait%20time
12. https://www.geeksforgeeks.org/merge-sort/
13. https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada
14. https://www.fourthsource.com/data/importance-real-time-data-five-reasons-need-22014#:~:text=Real%2Dtime%20data%20helps%20keep,where%20requirements%20are%20being%20met
15. http://iieta.org/sites/default/files/Journals/RCES/04.1_06.pdf

16. https://stackify.com/what-are-windows-services/
17. Manku G.S., Motwani R. (2002). Approximatefrequency counts over data streams, Proceedings ofthe 28th International Conference on VLDS, HongKong, China
18. Giannella C., Han J., Pei J., Yan X., Yu P.S. (2003)Mining frequent patterns in data streams at multipletime granularities, Data Mining: next generationchallenges and future directions, MIT/AAAI Press,pp. 191-212.
19. O'Callaghan L., Mishra N., Meyerson A., Guha S.,Motwani R. (2002). Streaming-data algorithms forhigh-quality clustering, Proceedings of IEEEInternational Conference on Data Engineering.
20. Domingos P., Hutten G. (2002). Mining high-speeddata streams, Proceedings of the Association forComputing Machinery 6th International Conferenceon Knowledge Discovery and Data Mining.
21. Domingos P., Hulten G. (2001). A general methodfor scaling up machine learning algorithms and itsapplication to clustering, Proceedings of theEighteenth International Conference on MachineLearning, Morgan Kaufmann. pp. 106-113.
22. Hulten G., Spencer L., Domingos P. (2001). Miningtime-changing data streams, Proceedings of theseventh ACM SIGKDD international conference onKnowledge discovery and data mining, San Francisco,California. pp. 97-106.
23. Aggarwal C., Han J., Wang J., Yu P.S. (2003). Aframework for clustering evolving data streams,Proceedings of the 29th VLDB Conference, Berlin,Germany.
24. Ferrer-Troyano F.J., Aguilar-Ruiz J.S., Riquelme J.C.(2004). Discovering decision rules from numericaldata streams, Proceedings of the 2004 ACMsymposium on Applied computing, Nicosia, Cyprus.pp. 649-653.
25. Law Y., Zaniolo C. (2005). An adaptive nearestneighbor classification algorithm for data streams,Proceedings of the 9th European Conference on thePrincipals and Practice of Knowledge Discovery inDatabases, Verlag, Springer.