

Software Asset Management

submitted by

Rafikul Alam

Class Roll No: 002210503050

Exam Roll: MCA244049

Master of Computer Application

under the supervision of

Prof. (Dr.) Sanjoy Kumar Saha

in the

Department of Computer Science and Engineering

Faculty of Engineering and Technology

Jadavpur University

2024

Faculty of Engineering and Technology
Jadavpur University

CERTIFICATE OF RECOMMENDATION
To Whom It May Concern

I hereby recommend that the project report entitled “Software Asset Management” has been carried out by Rafikul Alam(Reg. No.: 163642 of 2022-23, Roll No.: 002210503050, Exam Roll: MCA244049), under my guidance and supervision and be accepted in partial fulfillment of the requirement for the degree of Master of COMPUTER APPLICATION in DEPARTMENT of COMPUTER SCIENCE and ENGINEERING, JADAVPUR UNIVERSITY during the academic year 2023-2024.

.....
Prof. (Dr.) Sanjoy Kumar Saha

Project Supervisor
Dept. of Computer Science and Engineering
Jadavpur University, Kolkata 700032

.....
Prof. (Dr.) Nandini Mukhopadhyay

Head of the Department
Dept. of Computer Science and Engineering
Jadavpur University, Kolkata 700032

.....
Prof. Dipak Laha

Dean, Faculty Council
of Engineering & Technology
Jadavpur University, Kolkata 700032

Faculty of Engineering and Technology
Jadavpur University

CERTIFICATE OF APPROVAL

This is to certify that the project report entitled “**Software Asset Management**” is a bonafide record of work carried out by Rafikul Alam, in fulfillment of the requirements for the award of the degree of Master of Computer Application in the Department of Computer Science and Engineering, Jadavpur University. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the project report only for the purpose for which it has been submitted.

Signature of Examiner 1

Date :

Signature of Examiner 2

Date :

Faculty of Engineering and Technology

Jadavpur University

**DECLARATION OF COMPLIANCE WITH
ACADEMIC ETHICS**

I hereby declare that this project report entitled "Software Asset Management" is an outcome of my genuine and independent work, as part of my Master of Computer Application program.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, all materials and results that are not original to this work have been fully cited and referenced.

Rafikul Alam

Registration No.: 163642 of 2022-23

Examination Roll No.: MCA244049

Thesis Title :- Software Asset Management

Signature with Date

ACKNOWLEDGEMENT

I wish to record my heartfelt gratitude towards all the people who have helped me in the completion of this project.

I offer sincere thanks to my internal guide Prof. Sanjoy kumar saha and my external guide, Mr. Rishi Bandyopadhyay of OpenLM, for their kind support, inspiration, and thoughtful insights.

I sincerely thank Professor Nandini Mukhopadhyay, Head of the Department of Computer Science and Engineering at Jadavpur University, for her continuous support and guidance throughout the project.

I am extremely thankful to OpenLM and the Department of Computer Science and Engineering at Jadavpur University for providing all the facilities and necessary help towards the development of this project.

I am grateful to my parents, family members and friends for their continuous support, inspiration, encouragement and blessings, without which this project would not have succeeded.

.....
RAFIKUL ALAM

Date:

Registration No.: 163642 of 2022-23

Class Roll No.: 002210503050

Examination Roll No: MCA244049

Department of Computer Science & Engineering

Jadavpur University Kolkata 700032

ABSTRACT

Software Asset Management (SAM) is a crucial business practice that focuses on the effective management, control, and protection of software assets throughout their lifecycle. It involves the optimization of the purchase, deployment, maintenance, utilization, and disposal of software applications within an organization. SAM is fundamentally intended to be part of an organization's information technology (IT) business strategy.

The primary goals of SAM are to reduce IT costs and limit business and legal risks related to the ownership and use of software, while maximizing IT responsiveness and end-user productivity. It is particularly important for large corporations in terms of redistributing licenses and managing legal risks associated with software ownership and expiration.

SAM systems play a vital role in tracking license expiration, thus enabling the company to operate ethically and within software compliance regulations. This is crucial for eliminating legal costs associated with license agreement violations and forms an integral part of a company's reputation management strategy.

Both these aspects are important forms of risk management and are critical for the long-term business strategies of large corporations. Therefore, the effective implementation of SAM is not just a necessity, but a strategic requirement for organizations to ensure efficient and compliant use of software assets.

This project aims to develop a comprehensive Software Asset Management (SAM) system. The SAM system is designed to be deployed to customers, enabling them to efficiently manage their software licenses. The SAM system is built using a robust technology stack including .NET, GraphQL, and MongoDB, ensuring scalability and performance. It provides an intuitive interface for tracking, managing, and optimizing software licenses. It also helps in providing cost mapping with license usage. The system is capable of handling various types of software licenses, offering flexibility to the users.

Any company seeking to streamline their software license management can deploy this system. By providing a centralized platform for license management, the SAM system aids in compliance, cost control, and resource optimization.

TABLE OF CONTENT

INTRODUCTION.....	9
Purpose of the Project.....	9
Importance of Software asset management in contemporary business.....	10
Objective of SAM System.....	11
PROBLEM STATEMENT.....	13
❖ Difficulty in Tracking Software Licenses and Ensuring Compliance with Licensing Agreements.....	13
❖ Lack of Visibility into Software Usage Patterns and Associated Costs.....	13
❖ Inefficient Procurement Processes and Vendor Management.....	13
❖ Security Risks Due to Outdated Software Versions and Vulnerabilities.....	13
❖ Complex Reporting Requirements for Software Usage and Compliance.....	13
METHODOLOGY.....	14
Development Approach - Agile Development.....	14
Technologies Used.....	15
❖ .NET.....	15
❖ C#.....	15
❖ Apache Kafka.....	15
❖ MongoDB.....	16
❖ GraphQL API.....	16
Architecture.....	17
❖ System Architecture.....	17
❖ API Structure.....	17
❖ Kafka Structure.....	18
Data Models.....	19
❖ Vendor Information.....	19
❖ Software Information.....	19
❖ Features Information.....	20
❖ License Pricing Information.....	20
❖ License Type Information.....	20
❖ Purchase cost Information.....	21
❖ Procurement Information.....	22
License Models for Iteration - I.....	23
❖ Perpetual Licensing.....	23
❖ Subscription Licensing.....	24
❖ Token-based Licensing.....	24

API's.....	24
❖ GetAllVendors.....	24
❖ GetVendorById.....	24
❖ GetVendorNamesByFilterValue.....	25
❖ AddVendor.....	25
❖ UpdateVendor.....	25
❖ DeleteVendor.....	25
❖ GetSoftwareById.....	25
❖ GetAllSoftwares.....	25
❖ AddSoftware.....	26
❖ UpdateSoftware.....	26
❖ DeleteSoftware.....	26
❖ GetLicensePricingById.....	26
❖ GetProcurementById.....	26
❖ AddProcurement.....	26
❖ UpdateProcurement.....	27
❖ ImportCsv.....	27
Automated processes With Kafka.....	27
❖ License and it's usage details update from OpenLM server.....	27
RESULTS.....	29
FUTURE WORK.....	33
❖ Expansion of License Types.....	33
❖ Enhanced License Upgradation.....	33
REFERENCES.....	34

INTRODUCTION

In today's tech-driven world, managing software effectively is crucial for businesses. Software Asset Management (SAM) helps companies handle software from purchase to disposal, aiming to reduce costs, avoid legal issues, and improve efficiency.

This project introduces a SAM system designed to streamline software license management. Built with modern technologies like .NET, GraphQL, and MongoDB, the system is scalable and efficient. It offers an easy-to-use interface for tracking, managing, and optimizing software licenses, and helps map costs to license usage. The system supports various software license types, providing flexibility for different business needs.

Any company can benefit from this application, which aids in compliance, cost control, and resource optimization, making it an essential tool for efficient software management.

Purpose of the Project

The purpose of the project is to develop a comprehensive Software Asset Management(SAM) system for OpenLM ,with primary focus on efficiently managing an organization's software resources. The envisioned SAM system will encompass essential functionalities, including inventory management and procurement information tracking for software,vendors , contacts, contracts ,renewal. In addition to maintaining the procurement information, it will also help in providing cost mapping with license usage.

Example: A company purchase licenses from Dassault System for applications CATIA with feature HD2

SAM team (at customer) will need to maintain the following basic information -

Vendor	Dassault Systems (example)
Country of Purchase	USA
Product	CATIA (example)
Features	CATIA - HD2 (example)
License Type	CATIA - floating
Quantity of licenses	HD2 - 4 (example)
Date	Purchase Date - Annual Maintenance Contract renewal - License Expiry Date -
Pricing Model	Perpetual

Cost	For Perpetual – Purchase Cost & AMC cost Subscription cost Total and Per license cost for the above
------	---

Depending on the pricing model and cost information of a license for a software feature, it will be possible to derive the cost associated with the license for particular unit of time(i.e. Cost per hour/cost per minute/cost per second). Additionally the usage information of that software feature can be retrieved from the OpenLM server via Kafka messages. Once the license usage information and license cost per unit of time is available, the usage cost of the software license can be derived. The usage cost information thus derived can be utilized by reporting services to generate cost related reports.

Importance of Software asset management in contemporary business

- **Cost Control and Optimization** - SAM helps businesses optimize their software spending by providing insights into software usage, license agreements, and costs. By identifying unused or underutilized licenses and avoiding unnecessary purchases, organizations can reduce software expenditure and maximize their return on investment.
- **Compliance and Risk Mitigation** - Compliance with software licensing agreements and regulations is essential to avoid legal and financial penalties. SAM ensures that businesses use software licenses in accordance with contractual terms and industry regulations, reducing the risk of non-compliance and associated liabilities.
- **Security Enhancement** - Proper management of software assets enhances cybersecurity by ensuring that all software applications are up-to-date with the latest security patches and updates. SAM helps organizations identify and mitigate security vulnerabilities, reducing the risk of data breaches and cyberattacks.
- **Operational Efficiency** - SAM streamlines IT operations by providing centralized control and visibility over software assets. It simplifies tasks such as software deployment, updates, and license renewals, reducing administrative overhead and enhancing operational efficiency.
- **Strategic Decision-Making** - SAM provides valuable data and insights that support strategic decision-making. By analyzing software usage patterns, costs, and trends, businesses can make informed decisions about software investments, vendor negotiations, and technology adoption strategies.
- **Vendor Management and Negotiations** - SAM enables businesses to manage vendor relationships more effectively by providing detailed information about

software licenses and usage. This empowers organizations to negotiate favorable terms with software vendors and optimize licensing agreements to align with their business needs.

- **Resource Optimization** – SAM helps organizations optimize resource allocation by identifying opportunities to consolidate software licenses, standardize software platforms, and eliminate redundant or obsolete applications. This allows businesses to allocate resources more efficiently and reduce IT complexity.

Objective of SAM System

- **Efficient License Management** – The SAM system aims to efficiently manage software licenses across the organization, including tracking licenses, ensuring compliance with licensing agreements, and optimizing license usage to minimize costs.
- **Comprehensive Inventory Management** – The SAM system is designed to maintain a comprehensive inventory of all software assets within the organization, including details such as software versions, installations, and usage data.
- **Procurement Tracking and Vendor Management** – Another objective is to track software procurement processes and manage vendor relationships effectively, including recording vendor information, contract details, renewal dates, and negotiating favorable terms with software vendors.
- **Cost Analysis and Optimization** – The SAM system seeks to provide cost analysis and optimization capabilities, enabling organizations to understand their software spending, identify cost-saving opportunities, and optimize software usage to maximize return on investment.
- **Compliance Assurance** – Ensuring compliance with software licensing agreements and regulatory requirements is a critical objective of the SAM system, involving monitoring license usage, identifying potential compliance issues, and implementing remediation measures as needed.
- **Security and Risk Management** – The SAM system aims to enhance cybersecurity and mitigate risks associated with software vulnerabilities by monitoring software versions, applying patches and updates in a timely manner, and reducing exposure to security threats.
- **Reporting and Decision Support** – Providing comprehensive reporting and decision support capabilities is essential for the SAM system, including generating insightful reports on software usage, costs, compliance status, and optimization opportunities, empowering stakeholders to make informed decisions.

- **Integration and Scalability** – The SAM system is designed for seamless integration with existing IT infrastructure and scalability to accommodate future growth and changes in software usage patterns.

PROBLEM STATEMENT

❖ Difficulty in Tracking Software Licenses and Ensuring Compliance with Licensing Agreements

- Many organizations encounter difficulties in meticulously tracking their software licenses across diverse departments and locations, thus exposing themselves to compliance risks.
- The complexity of ensuring adherence to intricate licensing agreements and regulatory frameworks poses a significant challenge, particularly within large enterprises characterized by a myriad of software assets.

❖ Lack of Visibility into Software Usage Patterns and Associated Costs

- Organizations often grapple with a lack of comprehensive visibility into the utilization patterns of their software assets across their infrastructure.
- Without access to detailed usage data, accurately assessing software costs and allocating resources effectively becomes a daunting task, potentially leading to inadvertent overspending or underutilization of software licenses.

❖ Inefficient Procurement Processes and Vendor Management

- The procurement of software licenses and the management of vendor relationships can be time-consuming and inefficient, especially in scenarios involving multiple vendors and complex contractual arrangements.
- Inadequate vendor management practices may culminate in suboptimal pricing, missed renewal deadlines, or challenges in negotiating favorable terms, thereby impeding efficient resource allocation.

❖ Security Risks Due to Outdated Software Versions and Vulnerabilities

- Outdated software versions present substantial security risks, given the potential existence of unpatched vulnerabilities susceptible to exploitation by malicious entities.
- Maintaining a secure and up-to-date software environment necessitates proactive measures such as continuous monitoring, timely patching, and systematic updating, which can pose challenges when implemented at scale.

❖ Complex Reporting Requirements for Software Usage and Compliance

- Organizations frequently confront intricate reporting requirements pertaining to software usage and compliance, particularly prevalent within regulated industries.
- The generation of comprehensive reports that accurately reflect software usage, license entitlements, and compliance status necessitates robust reporting capabilities and real-time access to pertinent data sources.

METHODOLOGY

Development Approach - Agile Development

Our project utilizes Agile methodology, a dynamic and iterative approach to software development. Agile focuses on flexibility, collaboration, and responsiveness to change, allowing us to deliver high-quality software quickly and efficiently.

In Agile, our development process is segmented into short, time-boxed periods known as sprints. Each sprint involves our cross-functional team working closely together to complete small, incremental pieces of functionality, referred to as user stories. This structure enables us to collect feedback frequently, ensuring our outputs align closely with user requirements and expectations.

Continuous feedback and adaptation are key components of Agile. We regularly evaluate our progress and solicit input from users to refine and adjust our approach for subsequent sprints. This iterative cycle helps us to quickly adapt to changing needs and market dynamics, enhancing the relevance and utility of the software we develop.

To facilitate our Agile practices, we employ a range of tools and techniques, including daily stand-up meetings, sprint planning sessions, and regular retrospectives. These activities enhance communication and collaboration within our team, boosting our efficiency and effectiveness in reaching our project objectives.

Overall, Agile allows us to consistently deliver software that effectively meets user needs, promoting a culture of ongoing improvement and quick adaptation. Through diligent application of Agile principles and practices, we aim to achieve successful outcomes in our software development projects.

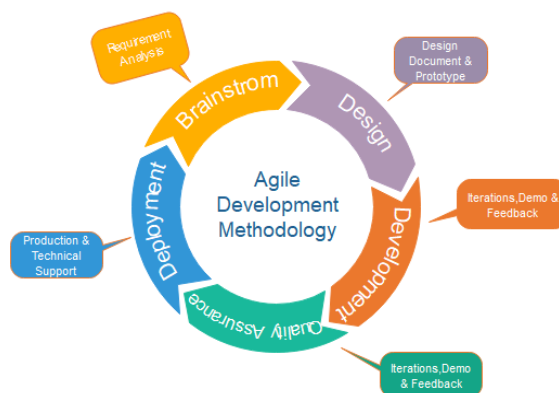


Fig. Agile Model

Technologies Used

❖ .NET

- .NET was chosen for its comprehensive ecosystem and robust framework, which is highly favored for enterprise-level applications. .NET provides a solid foundation for developing a secure and scalable SAM system. Its extensive library and support for network protocols facilitate secure communications and data protection crucial for managing sensitive information about software licenses and user data. .NET's capability to handle large-scale distributed systems ensures that the SAM can scale as the number of managed assets grows across the organization. This scalability is vital for businesses that manage thousands of software licenses and need to ensure system performance and reliability.

❖ C#

- C# is the primary language supported by the .NET framework, making it an obvious choice for our project. Its object-oriented capabilities encourage clean and modular code, making it easier to manage as the project scales. C# is used to implement business logic in the SAM system, such as license validation, compliance checks, and usage monitoring. The language's advanced features like LINQ (Language Integrated Query) simplify complex queries on large datasets, which are common in SAM systems when analyzing software usage patterns. C# supports asynchronous programming models that are vital for enhancing the performance of I/O-bound applications, such as those in software asset management, where database access and network communication are frequent.

❖ Apache Kafka

- Apache Kafka is used for managing real-time data feeds. Its robust publish-subscribe model helps us efficiently handle the stream of data regarding software license usage and updates. Kafka's ability to handle high throughput and fault tolerance makes it suitable for our needs to process and distribute large volumes of event data reliably and in real-time.

❖ MongoDB

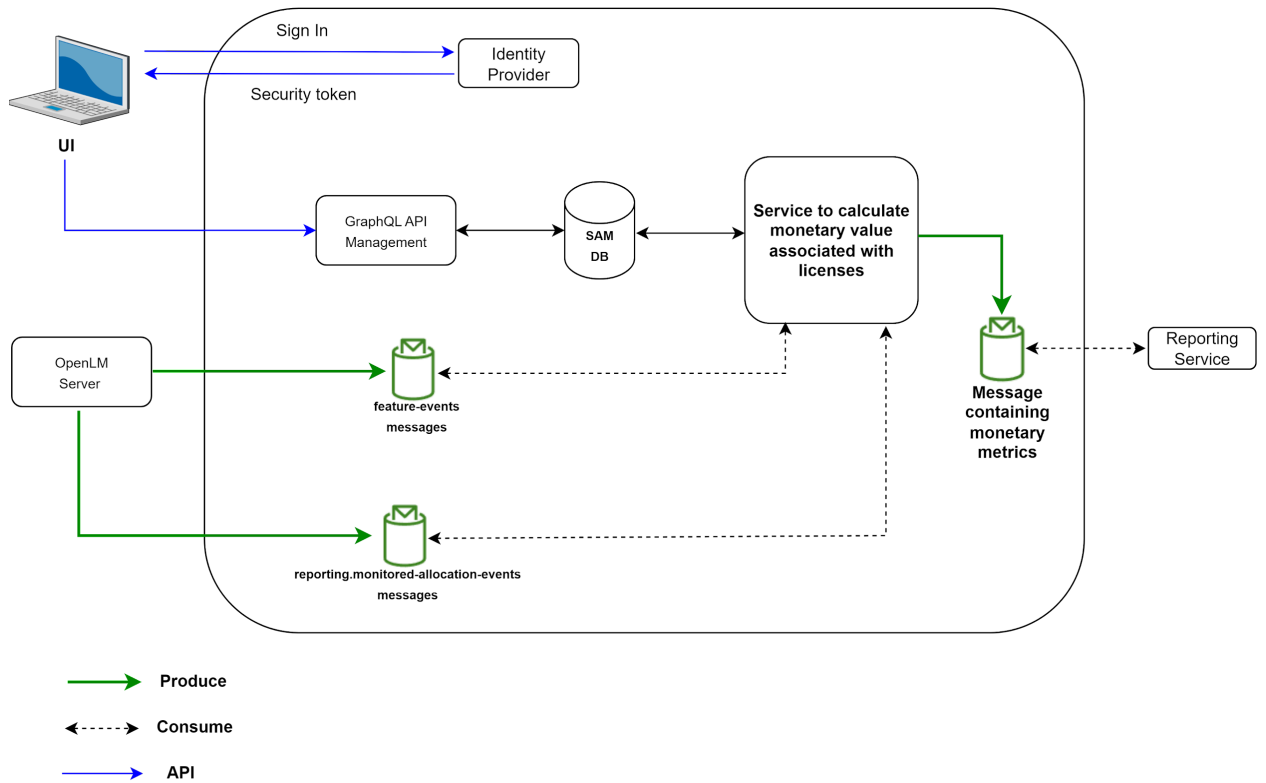
- MongoDB is a document-oriented NoSQL database, chosen for its agility in handling large volumes of unstructured data. Unlike relational databases, MongoDB's schema-less nature allows it to store varied and evolving data types, which is advantageous for managing the diverse and dynamic data sets typical in software asset management. Its powerful indexing and querying capabilities enable quick searches and real-time updates, essential for operational efficiency in asset tracking. In the SAM system, MongoDB stores diverse data such as software license information, vendor details, software details, and transaction records. Its flexible document model is particularly suitable for accommodating the variety of data involved in software asset management. MongoDB's aggregation framework supports complex queries and operations necessary for generating insights about software utilization and optimization, which can drive cost-saving decisions in an organization.

❖ GraphQL API

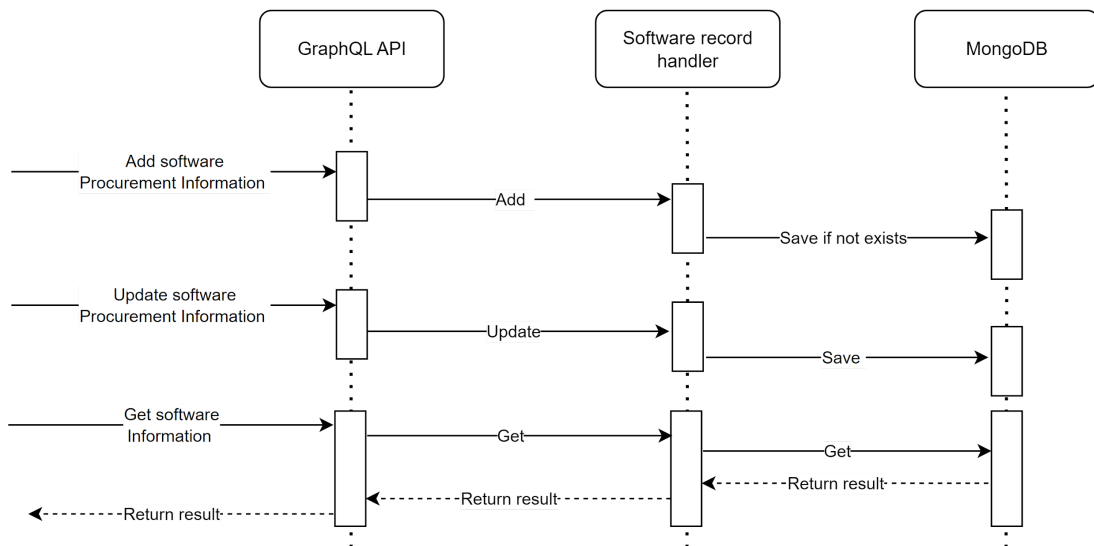
- GraphQL was selected for its ability to optimize data retrieval processes. It allows clients to specify exactly what data they need, which minimizes both the number of requests and the amount of data transferred, thereby improving performance and reducing load times. This is particularly beneficial in environments where network performance impacts user experience, such as in web applications for software asset management.

Architecture

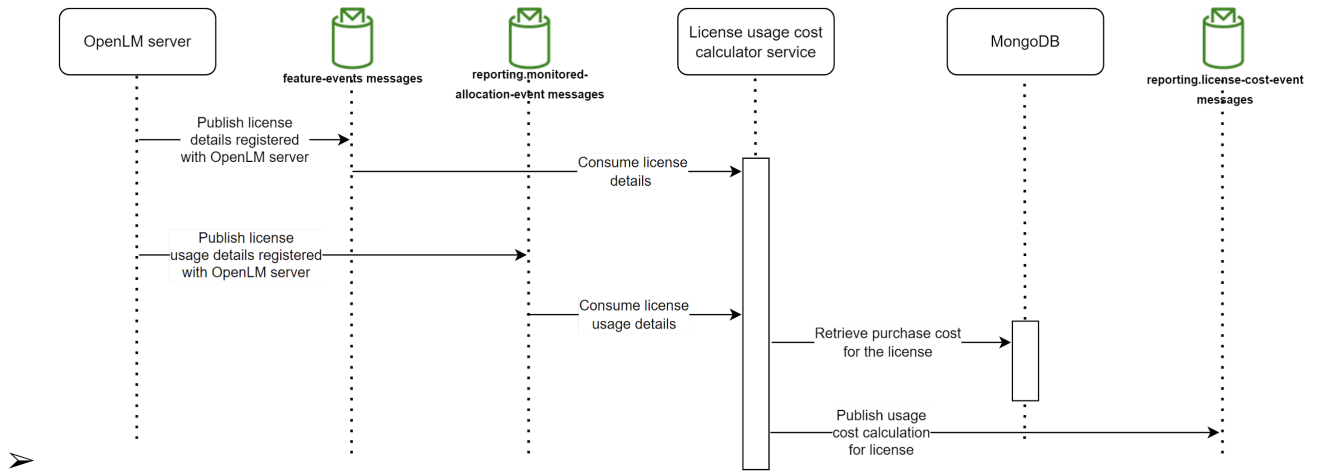
❖ System Architecture



❖ API Structure



❖ Kafka Structure



Data Models

❖ Vendor Information

- Vendor information consists of details about the companies or individuals who provide the software to an organization. It includes the names and contact information of the software suppliers or vendors.



Property Name	Data type	Comment
VendorID	string	Unique Id of the vendor
VendorName	string	Name of the vendor
VendorDescription	string	Description about the vendor
ContactName	string	Name of the contact person from vendor side
ContactEmail	string	Email of the contact
ContactPhone	string	Phone number of the contact

❖ Software Information

- Software information consists of details like name and version of the software. Additionally, it also contains a short description about the software, the category to which it belongs(e.g engineering, application etc.) and the software type (on premise/ cloud services)



Property Name	Data type	Comment
SoftwareID	string	Unique Id of the software
SoftwareName	string	Name of the software
Version	string	Version of the software
VendorId	string	Id of the vendor
Features	Feature[]	Array of features procured for the software

❖ Features Information

- Feature information consists of specific functionalities or characteristics of a software program. It helps users understand what the software can do and assists in managing and optimizing its usage based on its features.



Property Name	Data type	Comment
FeatureName	string	Name of the feature
FeatureVersion	string	Version of the feature

❖ License Pricing Information

- License pricing information consists of details about the cost and terms associated with using a particular software. It helps organizations understand how much they need to pay for the software licenses and any relevant pricing structures or agreements.



Property Name	Data type	Comment
PricingId	string	Unique Id of the pricing
PricingType	string	Type of pricing
HasPurchaseCost	boolean	Indicates if license has purchase cost
HasMaintenanceCost	boolean	Indicates if license has maintenance cost
HasSubscriptionCost	boolean	Indicates if license has subscription cost
HasTokenCost	boolean	Indicates if license has token cost
MaintenanceRenewalType	string	Type of maintenance renewal

❖ License Type Information

- License type information consists of details about the type of license used e.g Floating, NamedUser, NodeLocked, SingleUse, NamedUserCasual, HostBased, Unmanaged, UserNameLocked, Token, Disposable

Property Name	Data type	Comment
LicenseId	string	Unique Id of the License
LicenseType	string	Type of License

❖ Purchase cost Information

- Purchase cost information consists of details about the actual purchase made like the purchase date, purchase quantity, purchase cost, start & end date of the license etc. Every time a new software package is purchased/subscribed or renewed, it will be treated as a fresh purchase. So the PurchaseOrderNumber will be unique for each entry.



Property Name	Data type	Comment
PurchaseId	string	Unique Id of the Purchase
PurchaseOrder	string	Purchase Order Number
PurchaseQuantity	int	Quantity of Purchase
LicenseStartDate	datetime	Start date of license activation
LicenseEndDate	datetime	End date of license activation
PurchaseCost	decimal	Price of license file
MaintenanceCost	decimal	Price of maintaining the software
SubscriptionCost	decimal	Price of the License Subscription
TokenCost	datetime	Price of the License token cost
AMCStartDate	datetime	Annual maintenance cost start date
AMCEndDate	datetime	Annual maintenance cost end date
DepreciationPeriodInYears	int	Depreciation of currency over year
PurchaseCurrency	string	Currency in which license is purchased
PurchaseCountry	string	country from where license is

		purchased
CountryCurrency	string	Currency of the country from where license is purchased
LocalToPurchaseCurrencyRatio	decimal	Conversion rate of local to purchase currency

❖ Procurement Information

- Procurement information consists of details like vendor name, software name, software version, features of the software that are being procured, pricing type, license type, and purchase cost information
-

Property Name	Data type	Comment
ProcurementId	string	Unique Id of the procurement
VendorId	string	Id of the vendor
SoftwareId	string	Id of the software
Features	Feature[]	List of feature of the software procured
BusinessOwner	string	Department which owns the license
ServerHostName	string	Name of server in which network license are hosted
NodeHostNames	string[]	Name of hosts on which standalone licenses are installed
PricingId	string	Id of the Pricing type
LicenseType	string	LicenseType information

PurchaseInfo	JSON (PurchaseInformation)	PurchaseOrder -string PurchaseDate - date time PurchaseQuantity - int LicenseStartDate - datetime LicenseEndDate - dateTime (nullable) PurchaseCost - decimal128 (nullable) MaintenanceCost - decimal128 (nullable) SubscriptionCost - decimal128(nullable) TokenCost - decimal128(nullable) AMCStartDate - datetime(nullable) AMCEndDate - datetime(nullable) DepreciationPeriodInYears - int(nullable) PurchaseCurrency - string PurchaseCountry - string CountryCurrency - string LocalToPurchaseCurrencyRatio - decimal128
--------------	-------------------------------	---

License Models for Iteration - I

In the initial phase of development, the Software Asset Management (SAM) system will support three primary licensing models: Perpetual, Subscription, and Token-based. Each model entails distinct cost structures and renewal mechanisms, aimed at accommodating varying software procurement needs within organizations.

❖ Perpetual Licensing

- Perpetual licenses involve an upfront procurement cost, coupled with subsequent annual maintenance expenses. Upon initial purchase, the procurement cost is recorded in the system's purchase cost information table under the "purchasingCost" field. Subsequent renewals incur maintenance costs, which are documented in the same table under the "maintenanceCost" field. To ascertain the usage cost of a perpetual license over a specific duration, both the procurement and maintenance costs are taken into account.

❖ Subscription Licensing

- Subscription-based licenses entail recurring costs for a predetermined duration of validity. The initial subscription purchase cost and subsequent renewal expenses are captured under the "subscriptionCost" field within the purchase cost information table. To calculate the usage cost of a subscription license during a given timeframe, the subscription costs incurred within that period are considered.

❖ Token-based Licensing

- Token-based licensing operates on the premise of utilizing a specified number of tokens to access software features for a predefined duration. The cost associated with token usage is logged in the system's purchase cost information table under the "tokenUsageCost" field. To determine the usage cost of a token-based license over a specific timeframe, the per-token cost and the number of tokens utilized within that duration are taken into consideration.

Each licensing model adheres to distinct procurement and cost management methodologies, tailored to address the diverse requirements of software asset management. By providing comprehensive tracking and calculation mechanisms for procurement and usage costs, the SAM system aims to facilitate informed decision-making and optimization of software resources within organizations.

API's

❖ GetAllVendors

- Retrieves a list of all vendors registered in the system, providing a comprehensive overview of the available software suppliers.

❖ GetVendorById

- Fetches detailed information about a specific vendor based on their unique identifier, facilitating targeted vendor management and communication.

❖ GetVendorNamesByFilterValue

- Returns a filtered list of vendor names based on the provided search criteria, offering quick access to relevant vendors.

❖ AddVendor

- Allows users to add a new vendor to the system, providing essential details such as contact information and vendor description.

❖ UpdateVendor

- Enables users to modify existing vendor information, ensuring accurate and up-to-date vendor records within the system.

❖ DeleteVendor

- Removes a vendor from the system, along with associated software and procurement records, streamlining vendor management processes.

❖ GetSoftwareById

- Retrieves detailed information about a specific software product based on its unique identifier, aiding in software asset management and tracking.

❖ GetAllSoftwares

- Returns a comprehensive list of all software products registered in the system, facilitating holistic software inventory management.

❖ GetAllSoftwaresByVendorName

- Filters software products based on vendor name, allowing users to view software offerings from specific vendors.

❖ AddSoftware

- Enables users to add a new software product to the system, specifying details such as software name, version, and associated vendor.

❖ UpdateSoftware

- Allows users to update existing software information, ensuring accurate representation of software assets within the system.

❖ DeleteSoftware

- Removes a software product from the system, along with associated license and procurement records, streamlining software asset management processes.

❖ GetLicensePricingById

- Fetches detailed information about a specific license pricing model based on its unique identifier, aiding in understanding software licensing costs and terms.

❖ GetProcurementById

- Retrieves detailed information about a specific procurement activity based on its unique identifier, providing insights into software procurement history and transactions.

❖ AddProcurement

- Facilitates the addition of a new procurement record to the system, capturing essential details such as vendor, software, pricing, and purchase information.

❖ UpdateProcurement

- Allows users to update existing procurement records, ensuring accuracy and completeness of procurement data within the system.

❖ ImportCsv

- This API operation facilitates the bulk addition of procurement information from a CSV (Comma-Separated Values) file into the SAM system. Users can upload a CSV file containing procurement data, such as vendor details, software information, pricing, and purchase records. The system parses the CSV file, validates the data, and adds it to the SAM database. This functionality streamlines the process of importing large volumes of procurement data, reducing manual data entry efforts and ensuring data accuracy and consistency within the SAM system.

Automated processes With Kafka

❖ License and it's usage details update from OpenLM server

- Kafka, a distributed event streaming platform, plays a pivotal role in enabling real-time data processing and communication within Software Asset Management (SAM) systems. Designed to handle high volumes of data and provide fault-tolerant message delivery, Kafka serves as a reliable backbone for managing license and usage information efficiently.
- In the SAM ecosystem, Kafka acts as a bridge between the OpenLM server, which generates messages about license procurement, updates, and allocations, and the license usage cost calculator service. These messages, published to Kafka topics, serve as a real-time feed of critical events related to software licenses.
- One of Kafka's key strengths lies in its ability to decouple producers (such as the OpenLM server) from consumers (such as the license usage cost calculator

service), allowing them to operate independently and asynchronously. This decoupling ensures that updates to license information do not disrupt ongoing operations, and consumers can process messages at their own pace, ensuring system scalability and resilience.

- Furthermore, Kafka's distributed architecture ensures high availability and fault tolerance, crucial for handling mission-critical data in SAM systems. By leveraging Kafka's partitioning and replication mechanisms, organizations can distribute message processing across multiple nodes, mitigating the risk of data loss and ensuring continuous operation even in the face of hardware failures or network disruptions.
- Overall, Kafka empowers SAM systems to seamlessly integrate real-time updates about license procurement, updates, and allocations, enabling accurate and timely evaluation of license usage costs. Its robust architecture, coupled with its scalability and fault tolerance features, makes it an indispensable component in modern SAM architectures, driving efficiency, reliability, and agility in software license management.

RESULTS

Browser: <https://google.com>

SAM Management

Vendors

VENDORS

+ Add vendor Delete

☐	Vendor	Vendor Description	Contact Name	Contact Email	Contact Phone	
☐	Microsoft	Matlab	3920003	3920003	+112233	edit
☐	Microsoft	Matlab	3920003	3920003	+112233	
☐	Microsoft	Matlab	3920003	3920003	+112233	
☐	WPS	Matlab	3920003	3920003	+112233	
☐	WPS	Matlab	3920003	3920003	+112233	

< 1 2 ... 10 > Show up to 25

Browser: <https://google.com>

SAM Management

Vendors

ADD/EDIT VENDOR

Vendor Name *

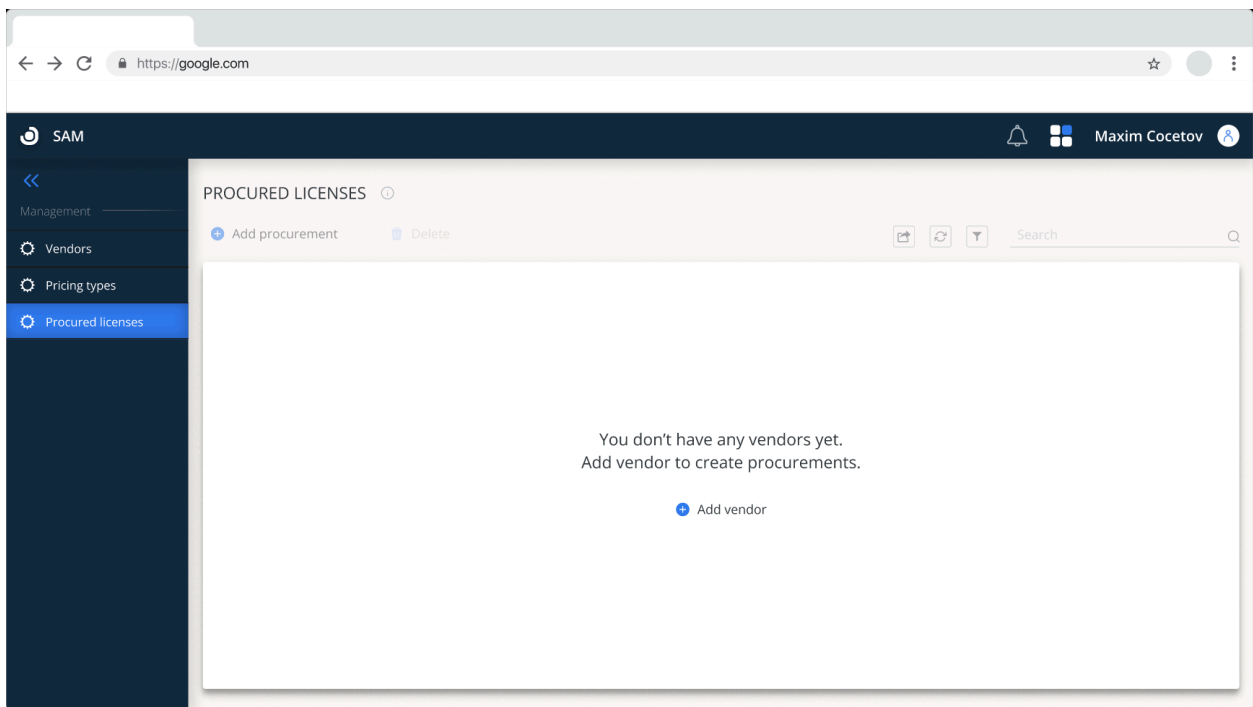
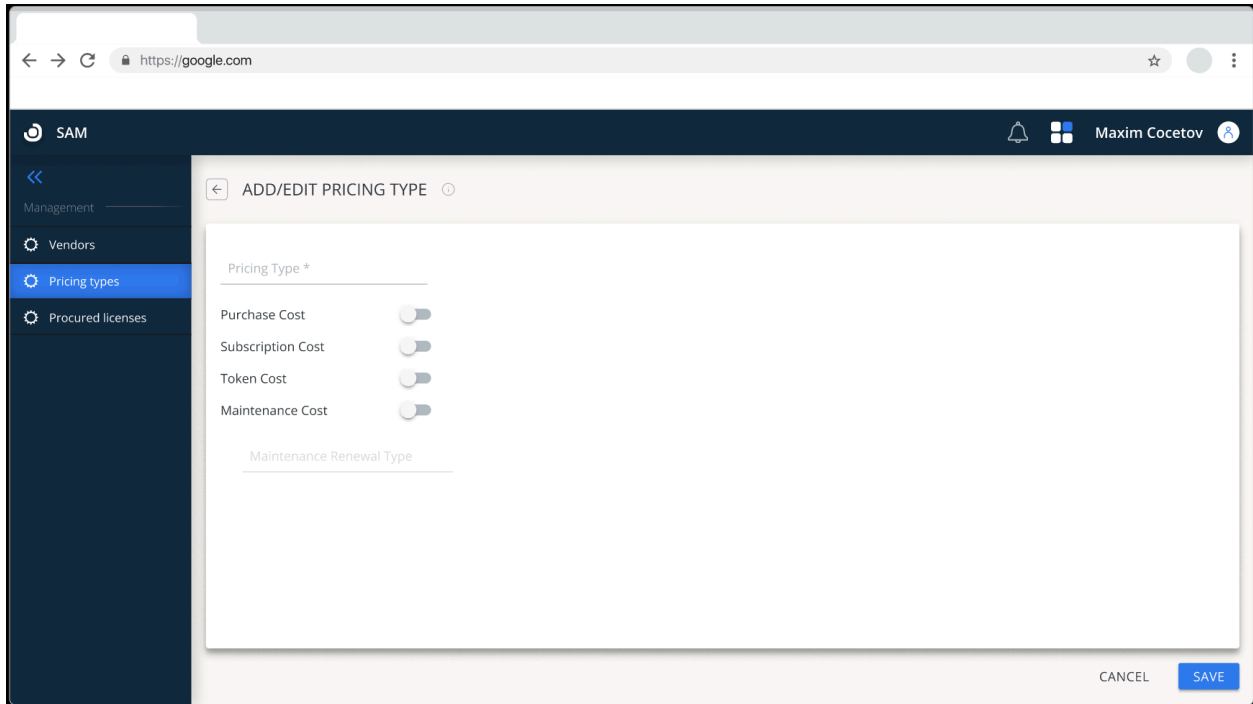
Vendor Description

Contact Name

Contact Email

Contact Phone

CANCEL SAVE



← → ↻ https://google.com ☆ ● ⋮

SAM

Management

Vendors

Pricing types

Procured licenses

PROCURED LICENSES ⓘ

+ Add procurement

🗑 Delete

📄 🔄 ⚙

Search

Q

☐	Vendor ⓘ	Software name	Software version	Feature	Description	Pricing Type	Purchase date	
☐	Microsoft	Matlab	3920003	3920003	Matlab Vendor	Named	12/01/2023 16:09:17	📄 ✎
☐	Microsoft	Matlab	3920003	3920003	Matlab Vendor	Named	12/01/2023 16:09:17	
☐	Microsoft	Matlab	3920003	3920003	Matlab Vendor	Named	12/01/2023 16:09:17	
☐	WPS	Matlab	3920003	3920003	Matlab Vendor	Named	12/01/2023 16:09:17	
☐	WPS	Matlab	3920003	3920003	Matlab Vendor	Named	12/01/2023 16:09:17	

< 1 2 ... 10 > Show up to 25 ▾

← → ↻ https://google.com ☆ ● ⋮

SAM

Management

Vendors

Pricing types

Procured licenses

ADD/EDIT PROCUREMENT ⓘ

General

Features

Purchase info

Vendor and Software

Vendor * ▾ Software Name * ▾ Software Version * Business Owner *

License ⓘ

License Type ▾ Server Hostname

+ Add Node Hostname

+ Remove

☐ Node Hostname

☐ 000f_1

☐ 000f_2

CANCEL

SAVE

SAM

Management

Vendors

Pricing types

Procured licenses

ADD/EDIT PROCUREMENT

General

Features

Purchase info

+

Add Feature

+

Remove

Feature Name

000f_1

000f_2

CANCEL

SAVE

SAM

Management

Vendors

Pricing types

Procured licenses

ADD/EDIT PROCUREMENT

General

Features

Purchase info

Purchase and Costs

Pricing Type *

Purchase Date *

Purchase Order *

Purchasing Quantity *

Purchase Cost

Purchase Currency

Purchase Country

Country Currency

Currency Conversion Ratio

Subscription Cost

Token Cost

Deprecation Period

License

License Type

License Start Date

License End Date

Maintenance

CANCEL

SAVE

FUTURE WORK

As the Software Asset Management (SAM) system progresses, upcoming iterations will concentrate on broadening the spectrum of supported license types and refining functionalities for license upgradation. This strategic advancement aims to cater to the diverse licensing needs of organizations and accommodate evolving software procurement practices.

❖ Expansion of License Types

- Future iterations will incorporate additional license types beyond the initial three models (Perpetual, Subscription, and Token-based). This expansion may include the introduction of specialized licensing models tailored to specific software categories or industry requirements. Examples of potential license types could encompass Concurrent User Licensing, Metered Licensing or Named Device Licensing,. Each new license type will be accompanied by distinct cost structures, renewal mechanisms, and usage tracking functionalities, enhancing the system's versatility and applicability across various organizational contexts.

❖ Enhanced License Upgradation

- In addition to supporting diverse license types, future iterations will prioritize the implementation of robust mechanisms for license upgradation. License upgradation refers to the process of transitioning from a lower-tier license to a higher-tier license, typically to unlock additional features or accommodate increased usage demands. The SAM system will incorporate intuitive workflows and automation capabilities to streamline the upgradation process, enabling organizations to efficiently manage license transitions and optimize software utilization. Moreover, advanced analytics and decision-support features will empower users to make informed upgradation decisions based on usage patterns, cost-benefit analyses, and strategic objectives.

By expanding the range of license types and refining license upgradation functionalities, the SAM system will establish itself as a comprehensive solution for software license management, capable of addressing the evolving needs and complexities of modern organizations. These advancements will foster greater flexibility, transparency, and cost-effectiveness in software procurement and utilization, ultimately driving operational efficiency and maximizing the value derived from software investments.

REFERENCES

1. Introduction - Hot Chocolate - ChilliCream GraphQL Platform - <https://chillicream.com/docs/hotchocolate/v13>
2. Programming Concepts - C# | Microsoft Learn- <https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/>
3. C# Tutorial (C Sharp) - <https://www.w3schools.com/cs/index.php>
4. Introduction to GraphQL - <https://graphql.org/learn/>
5. Software asset management - Wikipedia - https://en.wikipedia.org/wiki/Software_asset_management
6. A .NET library for reading and writing CSV files. Extremely fast, flexible, and easy to use. | CsvHelper - <https://joshclose.github.io/CsvHelper/>
7. GitHub - ChilliCream/graphql-platform: Welcome to the home of the Hot Chocolate GraphQL server for .NET, the Strawberry Shake GraphQL client for .NET and Banana Cake Pop the awesome Monaco based GraphQL IDE. - <https://github.com/ChilliCream/graphql-platform>
8. SOLID Principles In C# With Examples - <https://www.c-sharpcorner.com/UploadFile/damubetha/solid-principles-in-C-Sharp/>
9. Hot Chocolate - GraphQL Tutorials - <https://www.apollographql.com/tutorials/intro-hotchocolate/03-hot-chocolate>
10. ASP.NET Core | Open-source web framework for .NET - <https://dotnet.microsoft.com/en-us/apps/aspnet>
11. Quick Start - C#/.NET v2.25 - <https://www.mongodb.com/docs/drivers/csharp/current/quick-start/>