Development of Contract Management System using Spring Boot

Rajnandini Dubey[1], Dr. Prakash Biswagar[2]

[1]*Department of Electronics and Communication Engineering & R V College of Engineering, Bengaluru, India*
[2] *Department of Electronics and Communication Engineering & R V College of Engineering, Bengaluru, India*

---***---

Abstract - Businesses revolve around contracts. Service agreements are established after a lot of negotiation with a wide range of vendors to offer products and services that assist in running an organization. These lay the foundation for every agreement by defining the terms and establishing the parameters through which customers obtain and pay for these products and services. The traditional approach to managing contracts entails numerous risks, including incomplete contract disclosure, insufficient delegation of authority and responsibility, fraud, corruption, communication breakdowns, and delayed financial decisions that impede project progress, mostly as a result of manual intervention in managing contracts. This necessitates the use of a systematic and efficient approach of contract administration. This paper speaks about an automated contract management system which is a transparent and efficient method of managing contracts, built using Spring Boot.

Key Words: Contracts, Automated contract management system, Spring Boot

1. INTRODUCTION

Managing contracts and information assets is complex and expensive, managing performance of those contracts is even harder. Although there are numerous elements to contract management, we can distill the procedure into five distinct stages as follows: creation, cooperation, signing, tracking, and renewal. Each stage consists of various steps. To sum up, the entire procedure can be broken down into nine steps, each of which contributes to one of the five primary stages.

Creation

1. Initial requests: The contract management process starts with identifying contracts and necessary papers in order to support the goal of the contract.

2. Authoring contracts: Writing a contract by hand takes time but employing automated contract management solutions can make the process much more efficient.

Cooperation

3. Negotiating the contract: Employees should be able to review different drafts of the contract after it is written and resolve any differences to cut down on negotiation time.

Signing

4. Contract approval: The entire process slows down during the phase of requesting management approval. Users can avoid this by designing customized approval procedures that include parallel and serial approvals in order to maintain the pace of decision making.

5. Contract execution: By using electronic signatures and fax support, customers can expedite the signature process when executing the contract.

Tracking

6. Obligation management: It takes a lot of project management to make sure that partners fulfill their obligations, and that the contract's value doesn't fall in the early going.

7. Revisions and amendments: It can be challenging to gather all the paperwork needed for the contract's initial drafting. The system must modify the original contract when issues that were neglected are discovered.

8. Auditing and reporting: Contract audits are crucial for identifying both parties' adherence to the provisions of the agreement and any potential problems that may arise in the future.

Renewal

9. Renewing contracts: Manual contract management techniques frequently result in missed chances for renewal and lost revenue. Automating the process allows an organization to identify renewal opportunities and create new contracts [1].

Fig.1 depicts how all the above steps can be done using an automated contract management system. The benefits of an automated contract management system over manual contract management will be discussed later in the paper.

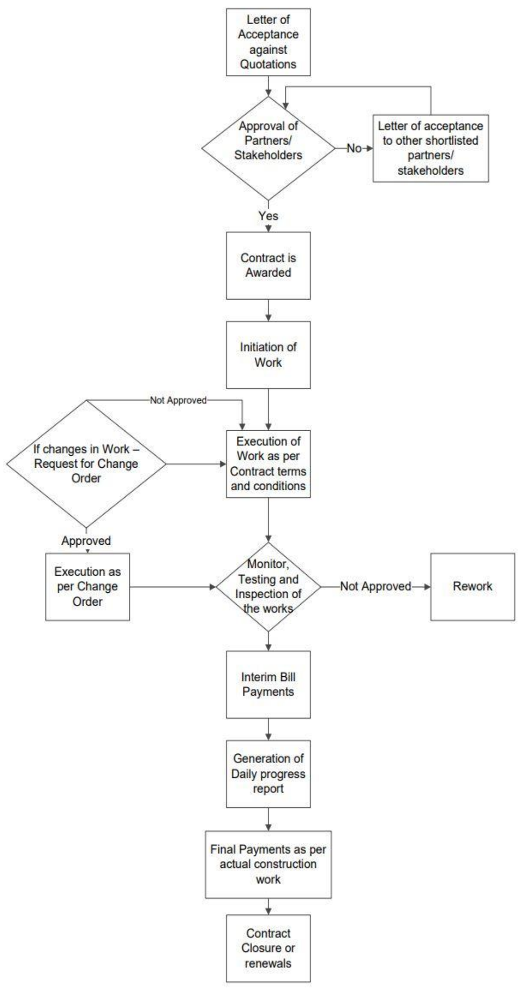
Letter of Acceptance against Quotations
Approval of Partners/ Stakeholders
No
Letter of acceptance to other shortlisted partners/ stakeholders
Yes
Contract is Awarded
Initiation of Work
Not Approved
If changes in Work – Request for Change Order
Execution of Work as per Contract terms and conditions
Approved
Execution as per Change Order
Monitor, Testing and Inspection of the works
Not Approved
Rework
Interim Bill Payments
Generation of Daily progress report
Final Payments as per actual construction work
Contract Closure or renewals

Fig-1: Automated Contract Management Process workflow

Initially, this project was done using the Play framework which was based on monolithic architecture. But in recent years, traditional monolithic architectures no longer match the demands of scalability and quick development cycles, which has led to the tremendous popularity of microservices architecture. Hence, migration from Play framework to Spring Boot, a microservice architecture-based framework, was done to separate the frontend from the backend. This makes it easier to code for a single specific functionality versus coding for an overall product [3].

2.1 PLAY FRAMEWORK

Web apps are developed using the Play framework. It was written in Scala and may be used with other programming languages, such as Java, that are compiled to JVM bytecode [4]. It is an open-source framework. Play framework and Spring Boot, both have an in-built server. This framework has a built-in Java compiler as one of its distinguishing features. We have the option of running an application and keeping it running in the background due to this characteristic of Play framework. Applications that have their source code changed will immediately show the changes, negating the need for another round of application deployment to the server. If a compiler error occurs, its message will be shown right away on the web browser.

The project migration from Play framework to Spring Boot was done due to many reasons:

i. It is an old technology, which is why most of the developers didn’t know how to code using this framework.

ii. It has not been updated since a long time, which is why it misses out on many new and important features.

iii. It is less secure which may lead to a security breach, causing a threat to the organization.

iv. It needs to be configured manually, which is time taking and inefficient.

2.2 SPRING BOOT

Spring Boot extends the Spring framework. It is used to create applications that are ready to run locally, and do not require any other software also known as stand-alone applications. Fig.2 depicts the Spring Boot flow architecture.

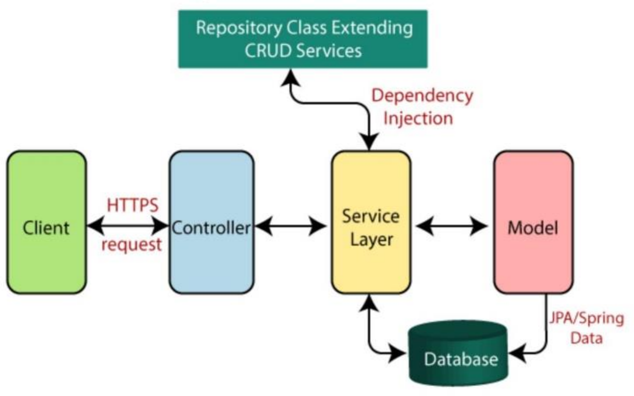

Fig-2: Spring Boot flow architecture [5]

Fig-2 can be summarized as follows:

i. A HTTP request like, GET, PUT, POST or DELETE, is made by the client.

ii. The controller then manages the HTTP request. It associates the URL with the request. It decides which method in the service layer needs to be called accordingly.

iii. The business logic resides within the service layer. Through Java Persistence Library (JPA), the logic is applied to the database that is mapped to the model class.

iv. The controller returns a Java Server Page (JSP) as response.

3. OBSERVATIONS

The entire project can be divided into four categories. Each of the categories have various modules within them, some of which are mentioned below.

<u>Operations</u>

Customer

As the name suggests, this module holds information about the customer. This includes basic information about the customer like name, contact number, email, address etc. Along with this, additional information is present in four tabs:

i. Financial: Information like financial year end date, credit check company name and total spend is present under this tab.

ii. Review: Information like review start date, review frequency, reviewer email and notification message is present under this tab. This ensures continuous evaluation of contracts and prevents contracts from being forgotten.

iii. Compliance: Customers must have a Certificate of Compliance issued by the Compliance Certification Board (CCB). This is essential for trade. This tab contains information about the expiry date of such certificates and sends a reminder to the reviewer email specified. This frequency can be set by the reminder frequency option present. Customer risk rating is also present under this tab.

iv. Billing: Each contract utilizes various commodities, and the usage of these commodities is tracked regularly by a cost tracker. The bill is sent to the email address specified here automatically according to the billing period information present under this tab.

Supplier

This module is like the customer module, except, it holds information about the supplier. This includes basic information about the supplier like name, contact number, email, address etc. It also has three tabs- financial, review and compliance which hold information related to the supplier.

<u>Tracking</u>

Cost Tracker

This module is used for tracking costs according to the commodities used. The pricing can be either variable or fixed. In either case, a connector is required to fetch information from the database about the commodities used. Information like minimum cost and cost per unit for each commodity is specified here.

If the pricing is variable, a calendar is present to specify the cost for each day and the scheduler runs accordingly to calculate the total cost.

Commodity

This module contains information about the commodities used in a contract i.e., name, description and region.

Service Level Agreement (SLA)

A SLA outlines the metrics used to measure service as well as any penalties or remedies that may be imposed if agreed-upon service standards are not met, thus defining the degree of service that is anticipated from a vendor.

This module contains information about the KPI (Key Performance Indicator) along with the criteria, business impact cost and cost to rectify. The review period frequency, reviewer email and notification message can also be set in this module.

Key Performance Indicator (KPI)

Establishing key performance indicators enable the contract managers to ascertain the performance of contracts over a period [6].

This module gives information about the name, metric, value and description of the KPI.

Connector

Connectors can be of two types:

i. Email Connector: These are used to enable mail flow between Office 365 and email servers that are present in the on-premises environment and apply security restrictions to them.

ii. Data Connector: A data connector is defined as a process that runs on a schedule, extracts data from a source location and writes into a destination location [7]. In this project, the “source location” are databases which can be of three types: MySQL, SQL Server and Oracle.

This module gives information about the connector type, name, server, URL, description and authentication details of the connector. If the connector is a data connector, details like the port number, database type, schema name and table name is also present in this module.

Configuration

Reports

This module acts like a filter for all the modules in the project. It contains a list of all the fields in all the modules which can be selected, after which any selection criteria (>, <, =, contains, does not contain etc.) is chosen for each field and then a sorting criterion is selected. The output is displayed in the form of a PDF which can be downloaded.

Notification

Since there are so many contracts of an organization, there is a high chance that a contract is filed and forgotten, causing missed reviews, renewals, and other crucial stages in the contract life cycle. The alerts each contract will send, who will get them, and with what messages have all been defined using the notification module. This ensures that the contract specifies what must occur at the appropriate moment.

Setup

Currency

This module is used for defining the currency used in a contract. It contains information about the currency name, symbol, descriptor and region name.

Outgoing Mail

This module contains information about the email servers. It includes the server’s name, port number, username and password.

All the modules discussed above along with some modules which are beyond the scope of this paper are used in the contract module to define all the contracts, each with a unique ID.

4. RESULTS

REST APIs were made for all the above-mentioned modules. This includes APIs for adding, editing, deleting, listing, fetching a module by its ID and importing a module.

These APIs were then tested manually using Swagger and the appropriate JSON objects were received as the output.

5. CONCLUSION

Remote access to the system enables saves a lot of time and effort required to track and administer contracts. It helps in performance monitoring of all contracts and agreements and gives all the stakeholders a single view of the contract position. It consolidates the contract database which makes the process of contract management very efficient. This plays an essential role in improving the security of an organization’s sensitive information since automated contract management systems store confidential documents in the cloud. Limits can be set on who has access, editing and sharing privileges, hence ensuring security of critical information. This was not possible when the process was done manually, because organizations used to consult and collaborate with employees and staff working remotely from other offices including external contractors which made data leaks and breaches highly probable. Using an automated contract management system, ensures that an organization only pays for the resources utilized, because when this process was done manually, due to lack of oversight, organizations incurred losses. It also helps in building trust between organizations, by increasing the transparency of the contract management process. Since each contract is unique, with a different set of policies and regulatory requirements, it used to be difficult to manage all obligations at once. Automated contract management keeps track of the status and requirements of each contract. This reduces the amount of risk an organization faces, which leads to better business.

REFERENCES

[1] https://www.businessnewsdaily.com/4813-contract-management.html [online]

[2] https://www.ics.com/ [online]

[3] Pillai, M. and Adavi, P., 2013. Intelligent Contract Management. *International Journal of Scientific and Research Publications*, *3*(1)

[4] https://en.wikipedia.org/wiki/Play_Framework [online]

[5] https://www.javatpoint.com/spring-boot-architecture [online]

[6] https://docucollab.com/key-performance-indicators-for-the-effective-management-of-contracts/ [online]

[7] https://www.datacoral.com/blog/data-connectors-101/ [online]