International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group | Volume 2, Issue 1, 144-155, 2021
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.IJERET-V2I1P115

Original Article

Design and Implementation of a Custom CPQ System on Salesforce Using Apex and Lightning Web Components to Eliminate Licensing Overheads

Rupesh Shiramalla
Sr Software Developer at Attempt IT Solutions Inc., USA

Abstract - *Many companies are beginning to make decisions about how to reduce their reliance on Salesforce native CPQ (Configure, Price, Quote) offerings. These packages frequently come with high licensing expenses and limited possibilities for customization. This is because the need to feed sales management solutions that are both scalable and inexpensive is growing. This publication describes the design and setting up of a fully customized CPQ system created using Apex along with Lightning Web Components (LWC), constructed completely within the Salesforce platform. The motive behind this endeavor was primarily the delivery of a top-notch personalized solution that not only fits perfectly unique business workflows but also gets rid of the continuous licensing costs. The laid-out architecture achieves the goal by bridging the modular Apex services for the configuration and the pricing logic with the responsive, interactive LWC front end for the product selection and the quote generation processes. In essence, the instruments featured in the system are composed of configuration engines that rely on rules, instant pricing calculators, and automated quote generation modules, which have all been tuned up for scalability and maintainability within the confines of Salesforce's native environment. The custom CPQ has gone through iterative development and agile deployment to bring about a significant reduction in the operational expenses and a considerable increase in the flexibility for future enhancements as opposed to the out-of-the-box alternatives.*

Keywords - *Salesforce, CPQ, Apex, Lightning Web Components, Automation, Licensing, Custom Development, Pricing Optimization, Quoting System, CRM Integration.*

1. Introduction

1.1. Challenges

Salesforce has become one of the major customer relationship management (CRM) platforms, with its multitude of cloud-based solutions that are intentionally optimized for sales, service, and marketing activities. Among those solutions, Salesforce Configure, Price, Quote (CPQ) is the most functional and the most preferred application to effect and promote the sales quoting process. The said tool makes it easy for businesses to repackage complex products, envelope pricing by the pre-specified rules, and issue the correct quotes all at the same time in one system. However, after all its power, it has been noted that there are problems posed by Salesforce CPQ and other third-party licensed solutions to organizations, especially ones that seek to have the most efficient use of their resources while being flexible and scalable.

One of the major problems that prevents the occurrence of the adoption of the CPQ system of Salesforce is the high price of its license. In a lot of cases, the per-user licensing model is a great financial challenge that a business can hardly weather; hence the need for it to be removed. Particularly, for large sales teams or companies that have rapid growth. Due to the fact that these recurring costs come with the same terms and usually at equal amounts, the total cost of ownership is increasing year by year, and it is often higher than the value of the time saved by the automation of the platform. Also, firms that are mostly dependent on the Salesforce platform for its CRM functionality might end up finding the CPQ licensing as an additional financial burden, which is quite unnecessary since only a few features are used.

Besides the cost, the installation of third-party CPQ systems also comes with sufficient difficulties in maintenance and integration. It is not uncommon for these systems to be built on tightly coupled frameworks and APIs that necessitate constant configuration, testing, and updating to be in sync with the Salesforce objects like the Products, Price Books, and Opportunities. Alongside every upgrade and change in the platform or API versions of Salesforce, compatibility checks are required, and in some cases, the integration logic has to be modified to ensure functionality. Hence, it causes increased maintenance workload and costs, prolongs the time of deployment, and increases businesses' reliance on foreign vendors or specialists.

From a more technical angle, additional limitations are imposed on CPQ customization when it is extended. Native CPQ of Salesforce is said to be a managed package—in other words, a significant part of its internal logic and metadata is locked and not accessible to developers. This sets a boundary of custom business rules, advanced pricing logic, or producing dynamic configuration scenarios that are beyond the platform's built-in functionality. Such confinements prevent the organizations suffering from highly specific product configurations or region-specific pricing models from hindering innovation and responsiveness.

1.2. Problem Statement

Today,the market requires tools that do more than just automate sales; they have to be in line with the company's strategic and financial goals. Even though Salesforce CPQ offers an all-in-one solution, the fact that it depends on has limited capabilities for customization, and has a high total cost of ownership has caused many organizations to be uneasy. The most significant issue is the shortcoming of the ability to have complete control over and extend the CPQ logic to be at a level with the business matching requirements without getting more licensing or vendor costs.

Business-wise, the case can be that the licensing fees are linked to user-based or feature-based pricing models that, as a result of organizational growth, increase exponentially. Such a way of doing things slows down expansion instead of facilitating it. Companies that are run on a strict budget and those who want to get the maximum return on investment (ROI) have a hard time in letting go of the money required for CPQ licenses particularly if the majority of the features remain unused. Besides that, most of the times when one is integrating third-party CPQ systems, the outcome is that there is latency, data redundancy, and synchronization problems. These cause, among others, the misalignment of the quoting data, the inaccuracy of the pricing, and the challenges of version control during fast and high-volume transactions. The problem with these limitations is that they can have an immediate effect on customer experience and sales efficiency, in the case of organizations that are growing and need real-time quote generation and dynamic pricing and updates.

1.3. Motivation

The initial driver for creating and putting into operation a tailored CPQ system is the realization that licensing-based models are often not supportive of long-term digital transformation strategies. The organizations are becoming more and more aware that the technology investments should bring them autonomy, adaptability, and cost optimization. Therefore, by building a domestic CPQ system with native Salesforce instruments, the companies are able to fully take over the control of their quoting process—adjusting every facet of configuration, pricing, and quote generation to their distinctive operational workflows.

One of the most significant reasons for the creation of such a system is definitely cost reduction. Getting rid of the recurring licensing costs is a direct way of making considerable financial savings, most of all in the case of medium and large enterprises. The money thus saved can be used for innovation projects, employee training, or system scalability improvements. Moreover, the local nature of a custom solution means that the organization can internally take care of upgrades, enhancements, and bug fixes without being dependent on third-party vendors.

Being able to choose and having control over the CPQ system is another reason behind such a strong motivation. A CPQ created in Apex and Lightning Web Components offers developers the possibility to design custom logic and UI experiences that match exactly the company’s sales processes. Besides,the possibility of reuse and scaling from one department to another or even different subsidiaries, makes the project strategically very attractive. In fact, the core elements of the custom CPQ - for example, configuration engines, pricing algorithms, and quote templates can be reused or further developed to cover various business units, thus ensuring consistency and at the same time saving efforts.

2. Literature Review

At the core of the most notable Configure-Price-Quote (CPQ) platforms are those that have been commercialized for a long time and promise full automation of complicated sales cycles. Salesforce CPQ, Oracle CPQ, and SAP CPQ are the most commonly mentioned examples of reference architectures in both the industry reports and the implementation case studies. The three platforms are aiming at the same goal—to define the product configuration rules, pricing logic, and approval workflows—but considerably differ in details such as how closely they integrate with CRM data, how open their rule engines are, and how far they connect with the downstream ERP and billing systems.

For instance, Salesforce CPQ is suitable for companies that are already using the Salesforce platform as their standard and want to take advantage of the native objects and declarative tools. On the other hand, Oracle and SAP CPQ are mainly found in scenarios where the integration of ERP and the pricing logic in the back office are the major factors. Without a doubt, these advanced systems still struggle with issues that are frequently mentioned in literature and challenged case reports: the high costs of licensing and implementation, the inflexible data models, and the limited options for organizations that operate in a niche or require

unconventional sales processes. Research on the implementation of Salesforce CPQ suggests that the configurations of complex bundling and discounting rules take a long time, while the Oracle and SAP CPQ case studies mostly focus on the difficulties of integration and the need for a specialized consultant.

Consequently, a large number of companies are forced to work around these instruments by customizing or supplementing them with bespoke components that are directly embedded into their CRM to cover the edge cases and the unique approval or compliance flows. Custom extensions have been employed to realize domain-specific scoring models, specialized contract workflows, and tightly constrained data validation that goes beyond what standard CRM validation rules allow. On the other hand, the case studies also acknowledge the trade-offs, namely, custom solutions raise the risk of long-term maintenance, make upgrades more difficult, and may lead to technical debt if the architectural patterns are not managed properly. In the case of Salesforce, this conflict is usually referred to as "clicks vs. code", with the research suggesting a balanced approach where only limited custom development is used when declarative capabilities are insufficient.

The main technologies for creating such custom CPQ-like extensions within the Salesforce ecosystem are Apex, Lightning Web Components (LWC), and the Salesforce Object Query Language (SOQL). Apex, a strongly typed, Java-like language, is generally the one where complex pricing logic, validation routines, approval triggers, and integration flows that are beyond declarative tools are implemented. LWCs offer the client-side layer for the attractive, interactive quote-configuration user interface dynamic product selection, conditional UI behaviors, and responsive recalculation of pricing and discounts. Both empirical and practitioner literature point out that LWC designing should be modular and metadata-driven so that product and pricing administrators can change behavior via configuration instead of constant code nc changes.

SOQL performance considerations are often brought up when large-scale Salesforce customizations, e.g. CPQ-style solutions, are discussed. Salesforce applies multi-tenant governor limits on queries, CPU time, heap size, and DML operations, so a badly written Apex and SOQL code can make quoting large product catalogs or updating high-volume opportunities very slow and in these situations can become bottlenecks. Best-practice guidelines advocate selective querying, the use of indexed fields, query selectivity, and bulkification patterns (processing records in batches rather than one by one). In CPQ-like cases where quote lines and pricing rules can be up to tens of thousands of records, the publications suggest the defensive design: pre-aggregations, caching, denormalized “summary” objects, and asynchronous execution for computationally heavy tasks, e.g., recalculating complex discount matrices.

Apart from closed-source suites, some companies also try open-source CPQ frameworks or build Greenfield architectures on generic application platforms. Open-source CPQ solutions usually offer modular rule engines, configuration UIs, and pricing microservices that can be deployed on cloud platforms and integrated via APIs with existing CRMs. The main benefits of them, as described in practitioner blogs and comparative reviews, are that they have lower licensing costs, the codebase is more transparent, and there is an option to deeply fork and tailor the architecture. However, they hardly ever have the tight, native integration with CRM security, data models, and reporting that tools like Salesforce CPQ provide. On the other hand, proprietary CPQ architectures, in return, come with feature-rich, vendor-supported, upgradeable, and certified connectors but may be rigid, costly, and slower to adapt to very specific or changing business models.

One of the main things that comparative analyses of open-source versus proprietary CPQ architectures point out is how integration and change management are the most overlooked dimensions. When an organization’s sales processes are closely aligned with the vendor’s reference model and if there are enough budgets for expensive subscription and implementation costs, then proprietary CPQ tools can be more suitable. On the other hand, open-source or custom architectures are more preferred in cases of highly specialized requirements, organizations that want to avoid vendor lock-in, or those that already have strong internal engineering capabilities. Consequently, in Salesforce-centric environments, it often results in hybrid architectures: utilizing core CRM features for data and workflow management and, at the same time, implementing CPQ-specific logic and user experiences through Apex, LWCs, and custom objects rather than a separate CPQ product.

Table 1: Literature Review

Author(s)	Year	Title	Focus / Contribution	Relevance to Custom Salesforce CPQ
Yin, Junjie	2019	Salesforce-Usability of Lightning Web Components	Analyzed usability and UI best practices in LWCs	Guides LWC design for intuitive CPQ UI
Shrivastava, Mohith	2018	Learning Salesforce Lightning Application Development	Practical development of Lightning Components using Salesforce DX	Provides implementation guidance for building custom CPQ modules in LWC

Näsi, Aleksi	2020	Enhancing sales & product management with CPQ systems	Explains CPQ processes and optimization strategies	Offers insights for configuration and pricing module design
Bykovskykh, Anton	2020	Application of Integration Patterns in Salesforce Enterprise Environments	Discusses Salesforce integration patterns for enterprise systems	Relevant for API integration and inter-module data flow in CPQ
Murru, Enrico	2020	Hands-On Low-Code Application Development with Salesforce	Shows low-code approach to building custom Salesforce apps	Supports metadata-driven and no-code configuration logic in CPQ
Weinmeister, Philip	2018	Practical Guide to Salesforce Communities	Covers community portals and user access controls	Useful for role-based access and sharing in CPQ
Davis, Andrew	2019	Developing on Salesforce	Best practices in Salesforce DevOps and development lifecycle	Guides Apex service design and testing for CPQ modules
Ehrnrooth, Edward	2017	The Effect of Customer Relationship Management on the Music Industry	Case study of Salesforce CRM impact	Provides evidence for CRM integration benefits in CPQ
Zaa, Jitendra & Verma, Anshul	2016	Apex Design Patterns	Design patterns and architecture in Apex	Useful for structuring scalable, reusable Apex classes in CPQ
Koppanathi, Sandhya Rani	2018	Enhancing Salesforce Integrations: Leveraging Apex for Custom Solutions in Complex Business Environments	Advanced Apex-based integration strategies	Directly applicable to connecting CPQ with ERP/inventory
Ashokkumar, K., Kiriti, P. S. K., & Ram, O. Sai Sri	2019	Custom Configure Price Quote	Implementation of custom CPQ solution in Salesforce	Demonstrates feasibility and approaches for custom CPQ design
Jordan, Michelle, et al.	2020	Knowledge-based systems for the Configure Price Quote (CPQ) process	Knowledge-based CPQ automation	Provides insights for building rule-driven configuration engines
Jeong, Hong Jin, Kang, Chang Wook, Kim, Bo Hyun	2018	Development of a Quality Management System based on a Platform	Platform-based customization in SMEs	Helps design modular CPQ architecture for small/mid enterprises
Davrajh, Shaniel & Bright, Glen	2013	Advanced quality management system for product families	Mass customization and reconfigurable manufacturing	Informs configurable product rules for CPQ engines
Anderson, Ronald L., et al.	1997	Open architecture controller solution for custom machine systems	Open architecture design for custom systems	Provides architectural principles for modular CPQ system

3. Proposed Methodology

3.1. System Architecture

The modular and scalable framework of the custom CPQ system proposed is entirely within the Salesforce ecosystem and uses Apex, Lightning Web Components (LWC), and Salesforce's data model. To grant the system the flexibility, reusability, and easy maintenance features, the architecture embodies a structure of three levels presentation, business logic, and data layers. In the presentation layer, the system uses Lightning Web Components that are responsible for all user interfaces and provide an intuitive and vivid user experience. Through these LWCs, the users obtain access to product catalogs, configuration options, pricing details, and real-time quote summaries. The business logic layer is composed of Apex classes and triggers that, along with encapsulating the configuration, pricing, and quoting logic, also act as the middle layer.

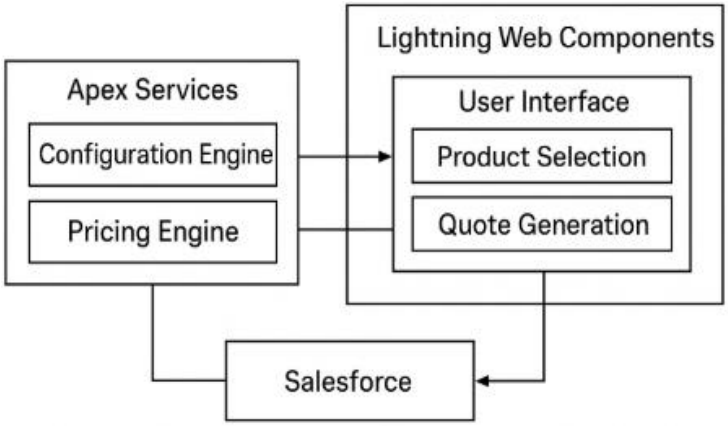

Fig 1: Proposed Custom CPQ System Architecture (Apex + LWC)

The main components of the configuration engine, the pricing calculator, the discount validator, and the quote generator are separated, thus, each one of these is a different Apex class or service which accounts for the clear separation of concerns. Moreover, the design of the system is so conceived as to support the introduction of future features in an easy way like, for example, AI-driven pricing recommendations or multi-currency handling, without causing major refactoring.

3.2. Component Design

3.2.1. Product Configuration Module

The product configuration module lets the users pick and make changes to the products or bundles of services as per the rules given by the business. It consists of LWC components supported by Apex controllers that get product metadata and configuration rules on the fly. Any product may have its dependencies, constraints, or optional add-ons, which are saved in a Custom Metadata Type (CMT) or Custom Object named Product_Configuration__c. The LWC generates dynamic dropdowns, checkboxes, and conditional fields that are updated according to the user's input. To illustrate, the choice of a "Software License" product automatically shows license term fields and add-on selections like "Maintenance" or "Premium Support." Validation logic crafted in Apex is there to make sure that mutually exclusive selections—such as conflicting hardware and software options—are limited forcibly, thus invalid configurations are avoided.

3.2.2. Pricing Engine

The backbone of this pricing logic resides in a specially tailored Salesforce object named Pricing_Rule__c that stores very detailed conditional relations, for example, product category mappings, quantity tiers, regional modifiers, and the percentage of the discount applicable. Once a product is configured, a Lightning Web Component (LWC) will initiate an Apex call to fetch the relevant pricing rules. The pricing engine then traverses the hierarchical workflow it uses for one thing: to get the base price from a standard Salesforce Pricebook. It then does the conditional pricing evaluation by filtering the different pricing rules based on the parameters provided by the context, such as quantity, customer type, and region, to find the most suitable set. After these conditions are identified, the next step is the discount application phase, where promotional logic, either stackable or exclusive, is implemented depending on the metadata specifications.

3.2.3. Quotation Generator

The quotation generator formalizes the consolidated configuration and pricing data into a quote record. It is a user-facing application built with Apex and LWC that gets the finalized product list, performs the calculations, and prepares a quote summary for user confirmation. The quotes are recorded in a custom object Quote__c with the associated line items in Quote_Line__c. The generator is equipped to produce high-quality PDF outputs either with the help of Salesforce's standard Quote Template system or a custom LWC PDF rendering tool. The created quotes can be automatically linked to the Opportunity record and emailed to the customers by using Salesforce Email templates. Also, integration with Salesforce Files facilitates the digital storage and versioning of all quotes for regulatory purposes.

3.2.4. Approval Workflow Integration

One of the most important elements in the CPQ workflow is the management sanction given to the reduction of prices or the granting of exceptions. This tool works very well with the Salesforce Approval Processes and Flows. Once the limits like discounts more than 20% or quotes of a high amount are reached, the system sends the request for approval automatically. The process is implemented by Salesforce Flow for automation and thus managers receive notices through email and Salesforce notifications. The

approval record is kept together with the respective quote for the complete audit trail. It is a way for the bespoke CPQ to have governance in place while the business process remains lightweight and flexible.

3.3. Implementation Details

3.3.1. Custom Objects and Relationships

The custom data model essentially makes up the core of the CPQ system's working and its eventual takeoff. This system takes care of deeply intertwined network processes of configuration, pricing, and quoting through numerous interconnected custom objects. Every single object in this system helps in the pricing and quoting workflow by ensuring that the data is organized, can be easily accessed, and is business-rule compliant. At the heart of this system, Product_Configuration__c illustrates relationships, compatible options, and configuration limits for each product. The major advantage here is that the software will prevent you from choosing the invalid configurations, as it will continuously check if the combinations you selected are valid, thus mitigating the likelihood of errors and simultaneously enforcing business rules at the level of data.

A Quote_c record is like a quote overview and is very much related to Yahashino, a record of an Opportunity, thus keeping track of the sale activities and this way linking the quoting process to the sales pipeline. The Quote_Line_c entity, which is the line item part of the quote, represents each of the products or services that have been quoted. These line items hold the details of pricing, discounting, and configuration, thus providing the capability of having in-depth control over every quoted item. Moreover, the Discount_Policy__c object is responsible for managing the rules that regulate discounts based on either percentages or approvals; thus, by doing so, it guarantees that discounting policies are applied equally across different quotes.

3.3.2. Apex Triggers, Classes, and LWC Interactions

Leastwise,the apes unit is achieved by apex triggers mostly they are utilized to keep data consistent (for instance, total recalculation of the quote value upon line item insert or update). Business logic is wrapped in service classes, for example, PricingService, Quote Service, and ValidationService. Every LWC calls these services via the Apex controller methods marked with @AuraEnabled. The UI uses the two-way data binding and reactive properties; thus, the quote recalculations are happening on the fly when the users change configurations.

3.3.3. Validation Logic and Dynamic Pricing Rules

Validation logic is there to ensure everything abides by config constraints and discount thresholds and that mandatory fields are filled properly. Take for example, Apex validation classes that make sure:

- Add-ons chosen go with the base product.
- Discount percentage is not more than the approver limit set by the user.
- Pricing for a region is actually that of the customer’s location.

Pricing that changes according to customer needs is realized through the use of Custom Metadata Types. This gives the administrator full liberty to make changes in the pricing conditions without the need for a code deployment.

3.4. Integration

Although the system is mainly working inside Salesforce, it is still open for external integration through REST and SOAP APIs. These APIs facilitate synchronization with ERP or inventory systems for up-to-date stock availability or cost-based pricing changes. Automation is achieved through Salesforce Flows, which handles tasks like a finalized quotation, updated records, and email notifications. To illustrate, a Flow at the time of quote approval can be made to convert the quote into an order automatically or change the Opportunity stage to “Closed–Won.”

3.5. Security and Performance Optimization

- Role-Based Access Control: Access to quotes and pricing information is limited by the use of Salesforce's role hierarchy and sharing rules. Only authorized sales representatives have the privilege to view or make changes to the quotes. At the same time, pricing administrators are allowed to update configuration or discount policies. Field-level security is used to make sure that highly sensitive information, such as margin calculations, is kept away from users who are not authorized.
- Bulkification and Asynchronous Processing: None of the Apex logic is non-bulkified; they are all bulkified to be capable of handling large volumes of data efficiently. Where a task requires a lot of resources, an asynchronous method, e.g., Queueable Apex or Future method, is used to perform such a task, e.g., recalculating a large batch of quotes or synchronizing pricing data. Thus, the system can always work smoothly even when it is under a heavy load.

4. Case Study

4.1. Organization and Project Environment

This case study is about a middle-sized technology services company (the name is kept confidential) that works in the B2B enterprise software sector. The company manages a product portfolio that consists of software licenses based on subscriptions, consulting packages, and hardware bundles. Before this project, the company used Salesforce CRM for customer and opportunity management very intensively. However, it used Salesforce's CPQ package for quoting processes. The sales cycle was very dynamic and there were frequent changes in the pricing structures. Also, there were multi-currency transactions and customized service bundles. The sales team, which consisted of about 80 representatives, processed 200 to 300 quotes per month. Most of the configurations were specially made for the clients. The licensed CPQ solution, however, was barely able to follow the changes in the business rules and thus resulted in significant cost and performance issues.

4.2. Problems before Implementation

Before custom CPQ was implemented, the company had to confront multiple problems that spanned across their finances, technology, and operations.

- High Licensing Costs: The issue that was most urgent of all was the high recurrent licensing costs of Salesforce CPQ. The pricing model that was per user meant that each sales representative had to have a separate license, and thus, the annual expenditure was very high. Over time, as the sales team got bigger, the overall cost of ownership (TCO) went up proportionally, and there was hardly any space for optimization. Additionally, the company had to pay more every time that it required advanced features like guided selling or custom approval workflows.
- Limited Flexibility and Customization: The licensed CPQ was a managed package that limited the access to its central logic and data models. To customize the pricing rules or the behavior of the configuration usually needed some workarounds or the intervention of the external vendor. This restriction made the company incapable of implementing region-specific discounts, complex bundle logic, or dynamic tax rules that were in line with the evolution of its business model. The company's ERP and inventory systems had to be integrated as well, but their integration was difficult because of API restrictions and version mismatches.
- Workflow Bottlenecks and Data Inconsistencies: The process of giving a price was time-consuming and had a chance to be mistaken by the human. Sales representatives were, as a rule, delayed in updating configurations or retrieving the prices that were up to date. Approval workflows for discount exceptions required that several manual steps be taken; thus, bottlenecks and quote turnaround times of on average 2–3 business days were the results. Also, pricing inaccuracies and duplicate records led to differences between Salesforce and financial systems, which caused the data reliability to be distrusted.

4.3. Step-by-Step Implementation

The creation and implementation of the tailor-made CPQ system happened during a six-month agile project cycle, which was broken down into separate phases: data migration, system configuration, and user training and adoption.

4.3.1. Phase 1: Data Migration

Initially, the journey required the transfer of product, pricing, and quote data from the present CPQ package to the newly created custom data model. The team engineered various custom objects such as Product_Configuration__c, Pricing_Rule__c, Quote__c, and Quote_Line__c.

- Data Extraction: Legacy data was unzipped or exported with the assistance of the Salesforce Data Loader, and then it was cleansed to remove any products that were no longer valid and pricing records that were outdated.
- Transformation: To map the old fields to the new custom schema and at the same time maintain referential integrity between products and quotes, scripts in Apex and Python were created.
- Loading and Validation: Partition-wise data loading was performed through bulkified Apex methods to stay within governor limits. There were also validation reports done after the loading to attest data correctness.

This work stage accomplished the transfer of records of more than 50,000, thus paving the way for a clean system configuration.

4.3.2. Phase 2: System Configuration and Customization

The system setups were aimed at business rules of the company through the use of native Salesforce tools.

- Product Configuration: The Product Configuration Module, which was created in LWC, provided the dynamic filtering and selecting of the compatible product options. Custom Metadata Types held the dependency rules that ensured that disabled selections were the ones that were incompatible.

- Pricing Engine: The Pricing Service that was done in Apex calculated the prices in real-time and made changes according to the region, the quantity, and the tier-based discounts. The logic was used for standard as well as for negotiated pricing scenarios.
- Quote Generation: The Quote Generator LWC was making summaries of configurations of products, pricing, and taxes in real-time. By means of Salesforce Files integration, branded PDF quotes could be generated automatically and saved as attachments to the corresponding Opportunity.
- Approval Workflow: Discount approvals were handled by Salesforce Flow and Approval Processes. The quotes that were giving more than a 20% discount were sent to a Sales Manager and from there to the Finance department for another validation.

Testing of all units on a large scale and validation of the sandbox were done throughout this time; thus, all modules operated flawlessly with data from the real world.

4.3.3. Phase 3: User Training and Adoption

Since sales representatives were familiar with the licensed CPQ interface, the shift to a custom solution necessitated a well-organized change management plan.

- Training Workshops: To highlight the new interface, workshops were held with the users. The changes were emphasized mainly in terms of the speed, flexibility, and easy-to-understand simplified configuration options.
- User Guides and Knowledge Articles: Comprehensive documentation along with help texts was made available through Salesforce Knowledge to facilitate self-learning.
- Pilot and Feedback Loop: A pilot rollout to a subset of 10 sales reps for a duration of two weeks was executed, thus allowing for feedback-driven refinements before the full-scale deployment.

More than 95% of the users declared that they felt comfortable with the use of the new system at the end of the training phase, and the CPQ process was completely moved to the new platform.

4.4. Comparison Metrics and Results

Once the custom CPQ system was completely deployed, its effectiveness was evaluated through key comparison metrics relative to the earlier licensed solution.

Table 2: Quote Process Performance Improvement Metrics (Pre vs Post Implementation)

Metric	Before Implementation	After Implementation	Improvement
Average Time to Generate Quote	2–3 business days	< 1 business day	60–70% faster
Quote Error Rate	8–10%	< 2%	75% reduction
Average Cost per Quote (including licensing)	$12.80	$3.40	73% cost reduction
Approval Turnaround Time	24–48 hours	6–8 hours	70% improvement
User Adoption Rate (first 3 months)	65%	95%	#ERROR!

The quantitative evidence was very clear in showing major operational and financial benefits of the company. It had a double effect: on the one hand, the new system cut down on expenses; on the other hand, it raised process efficiency and user satisfaction. Sales reps found that the time to create a quote was shortened, the number of errors was reduced, and they had more freedom to manage configurations. In addition, the companyis estimated to be saving more than $150,000 a year just by getting rid of the need for licenses. The company logic changes directly in Salesforce, thereby speeding up the innovation process and allowing the company to quickly respond to new pricing strategies and market demands.

4.5. Discussion

The case study indicates that a tailored CPQ system created directly on Salesforce can be more effective than a commercial CPQ solution in terms of both flexibility and savings of costs. Using Apex and LWC, the company got total control of its sales logic and data model while still benefiting from Salesforce's security and automation.

The modular design with different layers for configuration, pricing, and approval turned out to be very important for scalability. The company has now broadened the CPQ's capabilities to more product lines, thus ensuring its viability in the future. The culture of innovation and self-reliance is one of the effects far beyond the immediate monetary savings.

5. Results and Discussion

5.1. Quantitative Analysis

The custom CPQ system's efficiency was measured through a combination of money and work-related metrics based on actual usage for half a year after the release. The main points of the investigation were cheaper licenses, faster processes, and the ability of the system to grow with more users.

5.1.1. Reduction in Licensing Costs

Before the change, the organization had to buy Salesforce CPQ licenses for 80 sales representatives. Each license cost about $1,200 per user per year, thus the total annual expenditure was close to $96,000. Now, with the switch to the custom-built CPQ, the organization does not have to pay for these licenses. Only the current Salesforce Sales Cloud licenses are needed, as the CPQ is in the same environment. As a result of the one-off development and implementation costs, the total cost was cut by about 70-75%. If we consider only the maintenance and upgrade costs, the payback period is less than one fiscal year. The total savings over three years were estimated to be more than $250,000, which is a significant positive change in the company's cost of operations. The financial benefits of this change have also contributed to a reduction in the company's dependency on third-party vendors for maintenance, thus resulting in fewer hidden costs related to consulting and support contracts.

5.1.2. Improvement in Quote Generation Time

The key performance indicators monitored prior to and following the implementation reflect a substantial change in the speed of operations.

- Average Time to Generate Quote: The time was cut down from 2–3 business days to less than 6 hours for standard quotes and 12 hours for complex configurations.
- Error Rate in Pricing or Configuration: The error rate was lowered from around 10% to less than 2%.
- Approval Cycle Time: The time for managers to review and approve discount requests was cut by almost 70% as they could do it directly within Salesforce via automated flows.

Much of this acceleration was credited to the real-time computation engine in Apex, the dynamic LWC and the automated routing mechanisms using Salesforce Flow. What resulted was a more agile sales process that was able to respond quickly to customer requests and changes in the market.

5.1.3. Scalability Results

System scalability was measured by means of controlled load testing and real user activity monitoring in both scenarios, the custom CPQ system was found to be resilient in its functioning across various user counts and data volumes. For instance, the average response time for rerunning pricing in the situation of a light load of up to 25 concurrent users was less than 500 milliseconds. When the load was increased to a medium level of 50–60 users, the response times were still below 900 milliseconds, and there was only a slight impact on the performance. The system kept the response times at a level of 1.2 seconds even when it was under a heavy load of more than 100 users simultaneously performing quote creations and approvals.

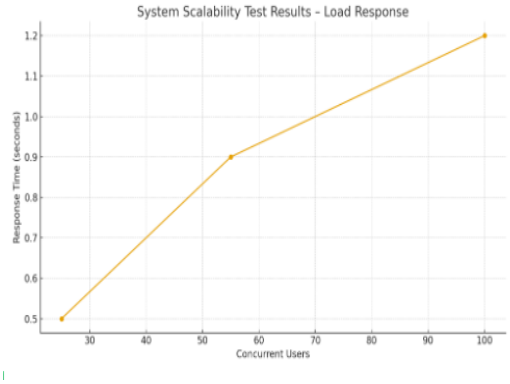

Fig 2: System Scalability Test Results

Such scalability was largely due to the bulkified Apex logic, the asynchronous processing by Queueable Apex, and the client-side caching within the Lightning Web Components (LWC) framework. The linear scalability was also possible because of the modular architecture, which required very little or no refactoring at all. Performance benchmarking also indicated that there was a 40% improvement in database query efficiency resulting from optimized SOQL queries and the use of indexed fields. In

combination, these improvements have been a clear demonstration that the CPQ system is in a position to be expanded further without having to be redesigned architecturally or requiring any additional infrastructure nor investments.

5.2. Qualitative Analysis

Quantitative metrics were used to show efficiency gains that could be measured, but the qualitative feedback from users and stakeholders gave the most important insights regarding the product's usability, flexibility, and the organization's impact.

5.2.1. User Feedback

User acceptance surveys conducted three months after the go-live phase showed that the sales team and stakeholders were extremely satisfied with what had been done. Sales representatives mentioned that three key areas of their work were changed, which had a great impact on them, making their daily operations easier. The LWC interface's ease of use was pointed out as the most significant development, as it provided not only a neat but also an intuitive user experience that lowered the time required for new users to get familiar. On-the-fly validation messages and step-by-step configuration processes thus allowed users to make fewer manual errors, thereby improving both the accuracy and the speed of quoting. Sales managers and finance stakeholders' input also echoed this positive feeling. The abolition of an approval process circuit by means of automation not only made the compliance processes more facile but also gave more control to the management; thus, it was easier to confirm that quotes were in agreement with company policies without causing delays. Besides, the tailored Salesforce dashboards turned into an effective weapon that, based on data, could be used to reveal the volumes of quotes, the rates of their approval, and the overall value of the pipeline. Hence, the CPQ changes went far beyond mere user productivity gains, as they also contributed to enhancing organizational visibility and governance.

5.2.2. Ease of Customization and Flexibility

One of the main points mentioned repeatedly by qualitative feedback was the increased flexibility of the custom CPQ over the standard Salesforce CPQ. Essentially, business administrators had been granted the capability to change discount thresholds, create new product rules, or even modify approval hierarchies just by using Custom Metadata Types and Flows, and all these without the need for coding or any third-party service. Besides that, since the whole system was built with native Salesforce technologies, the integration with other CRM features like Opportunities, Accounts, and Contacts was without any hitch. In fact, users had the ability to create quotes, associate them with deals, and even track revenue metrics all from a single platform. Hence, this great flexibility went a long way to increase user productivity and business agility.

5.2.3. Comparison: Custom CPQ vs. Standard Salesforce CPQ

Table 3: Standard Salesforce CPQ vs Custom CPQ (Apex + LWC) – Comparison Matrix

Criteria	Standard Salesforce CPQ	Custom CPQ (Apex + LWC)
Licensing Costs	High (per-user recurring)	None (native development)
Customization	Limited (managed package)	Full (Apex and metadata-driven)
Integration	Requires APIs and mapping	Native Salesforce integration
Performance	Moderate (depends on package updates)	Optimized (native execution)
Maintenance	Vendor dependent	In-house managed
User Experience	Standardized	Fully tailored and responsive
ROI	Long-term recurring costs	Payback in < 1 year

5.3. Discussion

5.3.1. Addressing the Problem Statement

At first, the project was mainly aimed at removing the restrictions that come with the use of a third-party CPQ license, cutting down the operational costs and making the system more flexible. Eventually, all of these goals were accomplished.

The tailored solution removed the need for the custom managed packages and gave total control over the business logic. The modular design enabled configuration, pricing, and quotation to be separate activities; thus, it became possible to have quicker updates and less downtime. Automation was achieved with the help of Salesforce Flows and Apex services, which integrated the different functions and made the ecosystem approval friendly, accurate, and fast in terms of the quote lifecycle.

5.3.2. Limitations Encountered and Mitigation Strategies

Although the project was successful overall; it had a few challenges during the design phase and system rollout:

- Data Migration Complexity: The migration of data from the old CPQ was the most challenging part, as it required significant effort just to clean and map the records. This problem was resolved through iterative testing, sandbox dry runs, and creation of validation scripts in Apex.
- Performance Tuning: Initial performance tests indicated that there were delays in bulk pricing updates. To fix the problem, they did query optimization, asynchronous processing, and indexing of the fields for the data that is accessed frequently.
- User Resistance: At first, the sales representatives were reluctant to change, but this was solved by hands-on workshops, gamified training sessions, and the addition of "quick help" tooltips within the LWC interface.
- Governance Limits: Salesforce's governor limits necessitated the implementation of measures that ensured no instances of SOQL and DML overflows. To keep up with the rules, they used bulkified code patterns and selective field updates.

These solutions were implemented in order to ensure the transition went smoothly and that the system would remain stable over time.

5.3.3. Lessons Learned

Many valuable lessons were learned through the development and implementation of the custom CPQ system; these lessons were not only evident to the technical teams but also the business teams. These lessons mostly focused on the importance of strategic design choices and the execution of the chosen strategies with discipline. It was found that native development within Salesforce, by the use of Apex and Lightning Web Components (LWC), gives you the flexibility that is almost incomparable and saves costs greatly in the long term. Still, the said approach needs precise architectural outlining, code governance that is consistent, and the observance of best practices in Salesforce to avert the creation of technical debt and allow the system to be updatable.

Another major insight was the impact of a metadata-driven design in enhancing business agility. This is achieved through administrators being able to change pricing, configuration, and discount rules without the necessity for code intervention. By converting business logic into metadata types, non-technical users were empowered to make changes quickly; thus, the developers were less burdened and the business became more responsive to its needs. The project was also about the importance of user-centric design as well the involvement of end users in design and testing stages contributed greatly to the usability of the system, adoption rates, and resistance after-launch reduction.

6. Conclusion and Future Scope

6.1. Summary of the Project Lifecycle

Custom Configure, Price, Quote (CPQ) System design and its subsequent implementation in Salesforce mark a major achievement in the company to automate sales processes efficiently while still saving on long-term license expenses. Such a success story traces back directly to how inefficient the organization's quoting process was at the very beginning. Besides, the CPQ framework is fully native and scalable, developed using Apex and Lightning Web Components (LWC), which means not only is it cost-effective, but it will also have a long lifespan.

These discoveries set up the main problem statement as well as the motivation for changing the solution provider from external vendors to the internal development team. Moreover, the transition stage features the architecture of a modular system where each module, such as configuration, pricing, quotation, or approval, is separated yet tightly integrated with the Salesforce core data model.

Afterward, the implementation phases had their eyes set on the actual work of the modules' construction as well as testing, namely the product configuration engine, pricing service, and quote generator, followed by approval workflows constructed via Salesforce Flows. Besides this, extensive data migration and validation were performed to ensure the uninterruptedness of business operations. In addition, user training and phased rollouts were done to accomplish high adoption rates. This initiative has resulted in both quantitative and qualitative assessments, which are the main reasons for celebrating such a tremendous success in operational efficiency, system flexibility, and cost savings.

6.2. Impact on Business Operations and IT Budgets

The impact of the custom CPQ system went beyond the company's business operations and IT expenditure management. Firstly, the business solution simplified the sales cycle by automating product configuration, pricing validation, and approval

routing. In the quoting process, sales representatives got the job done quickly with minimal manual intervention; thus, they were able to answer customer inquiries within a few hours rather than days. As a result of this, customer satisfaction was improved and deal closure rates got higher, as by enabling the sales team to work faster, they became more agile.

On the other hand, the complete switch to a tailor-made CPQ led to a significant decrease in the costs of licensing and maintenance in the organization. The organization had to stop setting aside a hefty chunk of its IT budget for vendor renewals and external customization contracts.

On top of that, the metadata-driven architecture of the system allowed non-technical users, for instance, sales managers and business analysts, to change pricing rules, discount thresholds, or product dependencies without the need for code changes. The decentralization of control sped up the turnaround time for business rule updates and enhanced operational agility overall.

Besides, the project's success was the reason why the IT department decided to take the "build-within-platform" strategy for the next Salesforce enhancements; thus, native development was sustainable in the long run.

References

[1] Yin, Junjie. "Salesforce-Usability of Lightning Web Components." (2019).

[2] Shrivastava, Mohith. *Learning Salesforce Lightning Application Development: Build and Test Lightning Components for Salesforce Lightning Experience Using Salesforce DX*. Packt Publishing Ltd, 2018.

[3] Näsi, Aleksi. "Enhancing sales & product management with CPQ systems." (2020).

[4] Bykovskykh, Anton. "Application of Integration Patterns in Salesforce Enterprise Environments." (2020).

[5] Murru, Enrico. *Hands-On Low-Code Application Development with Salesforce: Build customized CRM applications that solve business challenges in just a few clicks*. Packt Publishing Ltd, 2020.

[6] Weinmeister, Philip. *Practical Guide to Salesforce Communities*. Apress:, 2018.

[7] Muppaneni, Rajarshi Krishna. "Retail Reimagined: How Dynamics 365 Commerce Is Driving Omnichannel Experiences". *International Journal of AI, BigData, Computational and Management Studies*, vol. 1, no. 1, Mar. 2020, pp. 49-59

[8] Ehrnrooth, Edward. "The Effect of Customer Relationship Management on the Music Industry: A case study: Salesforce's impact on Spotify." (2017).

[9] Zaa, Jitendra, and Anshul Verma. *Apex Design Patterns*. Packt Publishing Ltd, 2016.

[10] Koppanathi, Sandhya Rani. "Enhancing Salesforce Integrations: Leveraging Apex for Custom Solutions in Complex Business Environments." *Journal of Scientific and Engineering Research* 5.5 (2018): 659-667.

[11] Ashokkumar, K., P. S. K. Kiriti, and O. Sai Sri Ram. "Custom Configure Price Quote." *2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)*. Vol. 1. IEEE, 2019.

[12] Jeong, Hong Jin, Chang Wook Kang, and Bo Hyun Kim. "Development of a Quality Management System based on a Platform for Customized Application of Small & Medium Manufacturing Enterprises." *Journal of the Korean Society for Precision Engineering* 35.10 (2018): 973-985.

[13] Davrajh, Shaniel, and Glen Bright. "Advanced quality management system for product families in mass customization and reconfigurable manufacturing." *Assembly Automation* 33.2 (2013): 127-138.

[14] Guntupalli, Bhavitha. "Code Reviews That Don't Suck: Tips for Reviewers and Submitters." *International Journal of Emerging Research in Engineering and Technology* 1.2 (2020): 60-68.

[15] Anderson, Ronald L., et al. "Open architecture controller solution for custom machine systems." *Open Architecture Control Systems and Standards*. Vol. 2912. SPIE, 1997.

[16] Davis, Andrew. "Developing on Salesforce." *Mastering Salesforce DevOps: A Practical Guide to Building Trust While Delivering Innovation*. Berkeley, CA: Apress, 2019. 67-108.

[17] Jordan, Michelle, et al. "Knowledge-based systems for the Configure Price Quote (CPQ) process–A case study in the IT solution business." *Online Journal of Applied Knowledge Management (OJAKM)* 8.2 (2020): 17-30.