International Journal of Computer Science and Engineering Innovations
ISSN: 3107-9458 | Volume 2 Issue 4, 28-40, October-December 2026
DOI: https://doi.org/10.64137/31079458/IJCSEI-V2I4P105
Received: 16/09/2026 Revised: 28/09/2026 Accepted: 04/10/2026 Published: 07/10/2026

Scholastic Research Publication

Original Article

How to Pass Parameters for a Lightning Component Called via a Quick Action

BAPU RAO SRIGADDE
Salesforce Developer at Thermo Fisher Scientific, USA.

ABSTRACT: *Passing parameters to a Lightning component that is called through a Salesforce Quick Action is not straightforward. The main reason is that the Quick Action framework does not support URL parameters or direct attribute injection at the moment of invocation. This investigation into easily implementable solutions for developers managed to circumvent that limitation by using standard Lightning methods and the contextual record data. Salesforce developers will be better off when they know how to create a component that obtains its necessary data, such as record IDs, field values from the configuration, or user-driven context, without having to take the risk of unsupported workarounds. The problem is that Quick Actions generally execute components in an isolated context where no dynamic inputs are accessible, so developers are frequently faced with the issue of passing values securely and in a way that can be maintained. The answer is to use the force:recordData interface, component implements declarations, Lightning Data Service, and the action's parent record context to retrieve or determine parameters at runtime. Besides these, there are also measures such as the use of wrapper objects, custom metadata triggers, event-driven communication, and conditional logic in the component's initialization that have been analyzed for cases that require complex data flows. The findings indicate that despite limitations on direct parameter passing, developers can still get hold of the data they want by making a component that finds its own context, uses the recordId which is given by the Quick Action by default, and, if necessary, obtains additional information through Apex or LDS on the fly. Thus, it is feasible to have tidy, more modular components that are behaviorally consistent regardless of the location of the action. To sum up, the recommended patterns reduce the necessity for URL hacks, improve security, and ensure Lightning architecture is in line with the latest standards.*

KEYWORDS: *Lightning Component, Salesforce Quick Action, Parameter Passing, Aura Component, Lwc, Component Communication, Salesforce Development.*

1. INTRODUCTION

1.1. CHALLENGES

It is not as simple as many developers think to pass parameters to Lightning components that are launched through Salesforce Quick Actions. Quick Actions seem to be at hand at first glance in order to easily combine the user's interaction with the logic of a custom component. Nevertheless,.the framework behind them puts very strict rules as to the way data can be sent if a component is to be opened. In contrast to other Salesforce entry points such as Lightning pages, navigation events, or custom buttons, Quick Actions do not support the usage of URL-based parameters or query strings. What this implies is that developers are not allowed to just add variables to the path of the invocation, nor can they use the usual methods that are utilized in the rest of the Salesforce platform.

Moreover, there are only a few Quick Actions that developers can use in order to gain access to contextual information. They can find the recordId that is being automatically inserted, but local record details, related objects, or custom state values are not accessible unless they get them by means of Lightning Data Service or Apex. This issue arises in situations where a component has to rely on several inputs for it to work correctly, e.g., the case of the form being filled with default data, a user being already selected, or a parent object being used as a reference. In order to reconstruct the context, developers have no other option but to do it themselves; thus, there is more complexity and inconsistency in the patterns that they use.

Besides that, regular Quick Action settings do not imply the possibility of specifying custom parameters at the sample data level. Parameter mapping can be done in flows or screen components, but the Quick Action one has only simple fields such as the label, icon, and component selection. Hence, it limits the degree of flexibility that exists and, at the same time, makes developers dependent on indirect ways if they want to find out what kind of parameters the action intends to pass, e.g., by querying for the context, employing custom metadata, or embedding logic in component initialization. Also, the constraints are different for Aura and Lightning Web Components; thus, there is more confusion. All these problems slow down the passing of parameters, which actually should be an easy task, and it becomes a far more complicated one that has to be solved by making careful architectural decisions to be able to provide it with reliability and maintainability.

This work is an open access and licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License
(https://creativecommons.org/licenses/by-nc-nd/4.0/)

1.2. PROBLEM STATEMENT

The major topic of this work is how to deliver custom parameters to a Lightning component that was initiated by a Salesforce Quick Action. Salesforce does not provide a direct, formal way to send extra values to Lightning components when they are started by a Quick Action, even though they are widely used in apps. Adding the recordId automatically is a fantastic feature; however, it is not enough for components that need to change a lot of information. Developers usually need certain values, such as user roles, workflow states, preselected options, or links to other records. Quick Actions doesn't provide you with any of them by default.

Both Aura and Lightning Web Component (LWC) implementations encounter this limitation, although to some extent differently. Aura components are able to access attributes in a more flexible way but still cannot get inputs directly from the Quick Action definition. On the other hand, LWC components provide advanced APIs but they depend wholly on the platform to provide the context. In the absence of support for custom attribute mapping in Quick Actions, developers have to look for different ways like server-side queries, wrapper objects, or conditional logic activated during initialization.

The difficulty has been increased due to the lack of complete and clear Salesforce documentation and the inconsistent experience of developers across different Salesforce clouds and setups. Most developers find the limitations only after they try the standard patterns that work in other parts of the platform. Therefore, confusion, workarounds, and trial-and-error methods frequently occur. The issue is not the absence of the necessary technical skills but the lack of a neat, framework-supported method for parameter passing. This paper is about identifying the problem more clearly and providing the insights that are organized and lead to the effective solutions for this challenge.

1.3. MOTIVATION

It is quite necessary to address this parameter-passing problem first as developers and organizations are progressively employing Lightning components to carry out business processes that are of main importance.The requirement for user experiences that are dynamic and aware of the context has escalated due to the fact that Salesforce is becoming more and more the center of enterprise workflows. There are plenty of on-the-ground examples that demonstrate the significance of custom parameters. A sales process could be an example in which a sales representative clicks a Quick Action to generate a follow-up task where certain fields are prefilled based on product type, opportunity stage, or assigned team. In cases where developers are not able to inject parameters, they have to take additional steps that slow the workflow down and elevate the cognitive load of users.

Moreover,custom forms' illustrations could be those that are used as Quick Actions. In most cases, they require default selections, visibility rules, or preloaded data that are indispensable for a particular business unit. If components are not permitted to obtain these values directly, developers will have to employ intricate logic to identify the missing parameters. Consequently, this prolongs the development time, lowers the maintenance level, and can cause erratic behavior across different orgs, especially if the configuration differs.

When Lightning components in Salesforce can change their working method depending on the environment without the users having to manually go through different screens, companies definitely get a profit out of it. By employing parameter-based behaviors, approval processes can become more effective, users can be taken through certain journeys, and background logic can be done without the user's intervention. As a result, the number of clicks is decreased, the possibility of errors is lowered, and the consistency of data entry is preserved. In business terms, the amplification of efficiency through sales, service, operations, and custom applications is resulting in real cost savings and an increase in user satisfaction.

When developers figure out how to deal with this problem, they are able to come up with more user-friendly solutions that really solve the operational needs. Automatically passing parameters with a well-defined plan allows deeper automation, quicker workflows, and business units that can smartly adapt to the surroundings. Finally, solving this problem is essentially returning to Salesforce the function of a multi-purpose, user-friendly platform for the development of customer experiences that are impactful and personalized.

2. LITERATURE REVIEW

A study of knowledge base articles, platform documentation, and community input reveals that the technology environment in which one can use Lightning components within Salesforce Quick Actions to pass custom parameters is quite complex. Developers are very well supported by Salesforce with very clear and detailed instructions on how to build components in both Aura and Lightning Web Components (LWC). Nevertheless, the documentation on Quick Actions mostly talks about creating them, displaying them in the UI, and the user interface that the user gets, rather than how programmatic parameter passing works. Therefore, most of the practical knowledge is leaked through developers who independently test and try the solution, community-driven explanations, and recognized community patterns that have been published in blogs and discussion forums.

The Salesforce docs show very clearly the differences between standard Quick Actions, global actions, and object-specific actions. Standard Quick Actions aim to make it easy and fast to do the most common tasks such as creating related records or

updating existing ones, and they are directly merged into page layouts. Object-specific actions are the ones that work within a particular record's context, and hence they automatically provide the recordId, which is underlying for use in Lightning components. Global actions, on the other hand, are like robots that work on their own without depending on any record context and thus can be a good choice for organization-wide utilities or workflows that are not going to use specific object data. The main difference here is that only object-specific actions give you contextual data; global actions, albeit being quite handy, do not have any built-in way for parameter transfer unless components get their own context or users give input.

Their main focus in the documentation for Aura components is the framework's ability to accept attributes and internalize values passed through markup or controllers. Normally, Aura components revolve around attribute-driven flexibility; thus, they can be easily configured when deployed in apps or other components. However, if an Aura component is used as a Quick Action, the platform does not provide a location for setting attribute values. So, in cases where Aura enables parameter passing at the component level, Quick Actions are not the ones to display those inputs.

Salesforce advises developers to manage context through the Lightning Data Service using force:recordData, which is a way of automatically getting the fields from the record that are relevant to the Quick Action. This method is perfect for getting object-level data but is insufficient for the scenario where there is a need to pass some random custom parameters that are not coming from the record.

Lightning Web Components offer a development model that is more contemporary, with better performance and easier templating. Salesforce documentation positions LWC as being dependent on the platform for providing the standard context, like recordId and object API name, when the components are used in the entry points that support them, such as Quick Actions or pages in the Lightning App Builder. However, LWC does not have a feature similar to attribute definitions in Aura, which can be supplemented by Quick Actions at the time of invocation.

The transition to LWC presents a challenge, as noted in several sources, which mention that due to the reactive nature and strict encapsulation of LWC, developers have to resort to services like @wire with getRecord, Apex calls, or parent container components to get more values. Salesforce acknowledges that LWCs deployed in Quick Actions should only refer to the attributes that are predefined and injected by the platform, thus indicating that it is a parameter-passing limitation that has been reinforced.

Comparisons of various invocation patterns in Lightning development indicate that PageReference, force:createRecord, and force:navigateToComponent are methods that can offer more flexibility in terms of data transfer. Navigation through PageReference provides developers with the ability to put the parameters of the state into the state object of the reference; thus, they can exactly determine what the component gets. It is a very typical usage scenario in console apps or custom navigation flows.In the same way, force:createRecord gives developers the opportunity to set the default field values of new records in advance; hence, indirectly, it also allows the passing of certain kinds of parameters. However, none of these methods actually direct to Quick Actions, since Quick Actions are functioning in a constrained modal environment and do not accept configuration-based parameters.

force:recordData is the main way through which Salesforce supports the retrieval of record data from components that are launched from Quick Actions. It is a very efficient tool for client-side field retrieval without the need for Apex calls, and thus it is a recommended approach in Salesforce examples. However, its application is limited to cases where parameters can be guessed or taken from the record itself. In a situation where developers have to provide some values which are not related to the parent record (e.g. configuration flags, custom workflow steps, or references to other objects), force:recordData cannot be of any help.

Informal debates among users of developer forums and question-and-answer sites, chiefly initiated by the community, have brought into focus typical workarounds that the users internally know and employ. One of the most frequently mentioned patterns is stuffing the code in the component's startup to figure out behavior from the current context. Some cases involve loading custom metadata to decide component mode, getting related records to determine workflow stage or using the names of Quick Action configurations to indicate component behavior. Although these methods can work, they are roundabout and do not provide the accuracy of exact parameter passing.

Most of the time, the second option that comes to mind when people think of event-based communication is Aura components. One of the advantages of the event-driven structure in Aura is that it can be used to spread the data to components or the parent containers that need it. Developers might embed the Lightning component into some mediator wrapper component through which they can pass parameters by attributes to the wrapper and then fire application or component events to send the values to the inner component. This stepwise technique enables parameterization at the expense of architectural complexity and the fact that it cannot be similarly done for LWCs since the latter can neither fire nor handle Aura events.

Blog writers and Salesforce architects are of the opinion that LWCs should use hybrid methods, like wrapping an Aura around an LWC. When the parameter passing is not done via Quick Actions, the Aura wrapper can take the configuration from the scene and then send it to the LWC through its open properties. Though this method may be beneficial in some big business applications, it does not solve the problem at the heart of the Quick Action invocation limitation. It just offers more control when components are called in different contexts, like Lightning pages or apps.

3. PROPOSED METHODOLOGY

Passing arguments to the components of Lightning that are called through Quick Actions needs a way that functions under the limitations of Salesforce but still allows the components to be dynamic. The outlined method centers around three interrelated layers: utilizing the context injected by the platform, enhancing component logic to get or deduce more inputs, and employing metadata-driven or wrapper-based solutions when it is really necessary to pass parameters. Besides, each subsection of the article discusses different approaches to be used in both Aura and Lightning Web Components (LWC), and gives reasons for using Lightning Data Service, lightning:quickActionAPI, and custom metadata for stable implementation patterns.

3.1. PARAMETER PASSING IN AURA COMPONENTS

Aura components have a more adaptable attribute system, and thus they were the natural first choice for effectively handling parameters. In case Quick Actions are not able to pass values directly to component attributes, developers can use Aura to create structured fields for receiving and then populate those fields by using context or logic that is derived. This method relies on a mixture of attributes, record-driven loading, and metadata-based lookup to get back the parameters that are not passed.

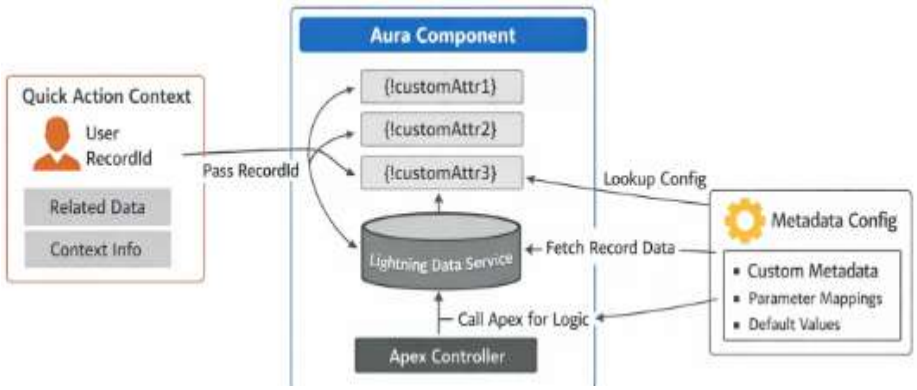


FIGURE 1 Aura Component Attribute-Based Parameter Reconstruction

3.1.1. AURA ATTRIBUTE FRAMEWORK

Aura components allow developers to define attributes such as:

- <aura:attribute name="recordId" type="String"/>
- <aura:attribute name="mode" type="String" default="default"/>
- <aura:attribute name="configValues" type="Object"/>

Imagine these attributes as temporary containers for parameters that might be coming from a record query, a custom metadata reference, or wrapper logic. In case the Quick Actions are not directly putting these attributes, they still look like normal places for the parameters to be after they have been gotten.

3.1.2. LEVERAGING LIGHTNING DATA SERVICE (LDS)

LDS provides Aura components with the means of accessing record context if they are fired through object-specific Quick Actions. A conceptual snippet shows how the component gets the context:

```
<force:recordData
aura:id="recLoader"
recordId="{!v.recordId}"
layoutType="FULL"
targetFields="{!v.record}"/>
```

Such a system yields situational data that may be understood as the input parameters of a function, e.g., an object type, its status, or user ownership. Subsequently, Aura controllers use this fetched data to logically decide whether to change or add more attributes.

3.1.3. USING LIGHTNING:QUICKACTIONAPI

Aura is likewise compatible with lightning:quickActionAPI, through which the element can see other actions, start actions on the fly, or get some metadata facts about the available actions. In principle:

```
var actionAPI = component.find("quickActionAPI");
actionAPI.getAvailableActions().then(...)
```

By means of this interface, it is not possible to send custom arguments. However, it can serve as a means for initiating different actions or figuring out which action resulted in the component being called, thus giving components the option of changing their modus operandi accordingly. Programmers are able to link the names of Quick Actions with the custom metadata that stands for the parameters anticipated.

3.1.4. CUSTOM METADATA-DRIVEN PARAMETER LOOKUP

Aura components can determine parameters dynamically by referencing the custom metadata records that correspond to the Quick Action name. For example, a metadata record can specify default values, UI modes, or behavior flags for a Quick Action called "Special_Create".

Conceptual flow:

- For getting the name of the Quick Action, the component context is the source that is referred to.
- Custom metadata is queried to get the record where ActionName is equal to the current quick action.
- Fields of metadata in the map are turned into Aura attributes (mode, configValues)
- Component logic is derived from these parameters implicitly.

Using this approach, admins are able to manage the parameters in a totally declarative manner without having to modify the code.

3.1.5. WRAPPER COMPONENT TECHNIQUE

It happens that developers surround the target Aura component with another Aura wrapper, which can predefine attributes. For situations that are not inside Quick Actions, this wrapper basically acts as a way to pass parameters. In the case of using Quick Actions, the wrapper will be just another place to compute or load parameters.

Sample wrapper layout:

```
<c:TargetComponent mode="{!v.derivedMode}" configValues="{!v.presetValues}"/>
```

The wrapper gets the inputs from LDS, metadata, or Apex and then it passes them to the child. However, Quick Actions cannot also inject parameters into the wrapper; this layer is still useful for dividing the logic and making the main component easier.

3.1.6. CONCEPTUAL FLOW FOR AURA PARAMETER RECONSTRUCTION

Flow Description:

- Quick Action is what lets the user use the Aura component.
- Just the recordId is passed to the Aura.
- LDS gets the full record context.
- The component gets the necessary information from metadata or server data to understand the parameters.
- The component changes attributes and gets the behavior ready.
- The user works with the component's rebuilt parameters.

This method with several layers keeps Aura adaptable, and at the same time it is compatible with the Quick Action limitations.

3.2. COMBINED RECOMMENDATIONS AND IMPLEMENTATION BLUEPRINT

An advanced parameter-passing technique for a single comprehensive method could be further enhanced by bringing in the properties of Aura, LWC, metadata, and LDS. The following blueprint provides an orderly way that can be utilized by most of the Quick Action components' architectures.

3.2.1. METHOD OVERVIEW

- Start with context Use recordId as the anchor for all logic since it is the only guaranteed parameter.
- Load record details via LDS Extract key fields to determine context-driven parameters.
- Map Quick Action to metadata Create custom metadata entries representing intended parameters for each action.
- Compute parameters at runtime Based on metadata and record data, compute values such as mode, visibility flags, or behavior overrides.
- Pass parameters to components
  - Simply in Aura
  - By means of Apex + reactive properties in LWC
  - Or use an Aura wrapper if LWCs need a complicated set of parameters

- Initialize component behavior Allow components to adapt layout, prefill forms, or alter logic based on the derived parameters.

3.2.2. CONCEPTUAL PROCESS FLOW (TEXT-BASED)

Start → Quick Action Launch → Inject recordId → Load record with LDS → Retrieve metadata → Compute parameters → Set component state → Display dynamic UI → End

TABLE 1 Comparison Table

Method / Tool	Aura Support	Lwc Support	Ability To Carry Custom Parameters	Notes
Lightning Data Service	Yes	Yes	Limited (derived only)	Uses record data, no custom values
quickActionAPI	Yes	No	Indirect	Helps detect invocation source
Custom Metadata	Yes	Yes	Strong	Declarative, flexible
Wrapper Component	Yes	Requires Aura wrapper	Strong	Bridges missing LWC features

4. CASE STUDY

Pre-Populating Form Values in a Lightning Component Launched via a Quick Action

4.1. BACKGROUND OF THE CASE

The sales department of a fictional company named SummitTech Services was looking for a way to make the capture of qualification details for new opportunities more efficient for their sales team. Sales representatives regularly accessed an internal form -implemented as a Lightning component- to write down details of the follow-up meeting, customer priority indicators, and initial scoring values. The existing method extensively required manual clicking and referencing, which led to inconsistent data and delays that could have been avoided.

Management ordered that the form should fill some fields automatically with business logic, e.g., opportunity stage, customer tier, account health score, and owner role. It was also required that the reps access the form by just one click through a Quick Action on the Opportunity record page. But since Quick Actions do not allow passing parameters directly, the component had to have a way to guess or rebuild the input values.

Therefore, the problem is about creating a Lightning component that can be invoked from a Quick Action, and it can intelligently and dynamically figure out which fields to prepopulate in the absence of parameters.

4.2. TECHNICAL ARCHITECTURE

SummitTech has decided on a hybrid style which intertwined Lightning Web Components for the user interface with an Aura wrapper for startup and parameter management. The core design was:

- Quick Action → Aura Wrapper → LWC Form Component
- Using Lightning Data Service (LDS) to get record context
- Custom Metadata to link Quick Actions with behavior configurations
- Apex helper for metadata lookup and conditional logic
- Reactive LWC properties for UI defaults

The team chose this structure to get around the restriction of LWC, which doesn't allow Quick Actions to assign values to custom @api properties. Thanks to the Aura wrapper, they managed to work out the parameters and then place them in the LWC in a tidy manner.

4.3. PARAMETER EXTRACTION STRATEGY

As a result, the team had to come up with a layered extraction strategy that would be able to reconstruct all the inputs needed before the Lightning Web Component could render since native parameters passing through Quick Actions were missing. First, the process of getting basic context from the record itself was started. While using Lightning Data Service in the Aura wrapper, the component fetched Opportunity fields such as StageName, AccountId, OwnerId, and Customer_Tier__c that were not only the pieces of information but also the basis of the next logic.

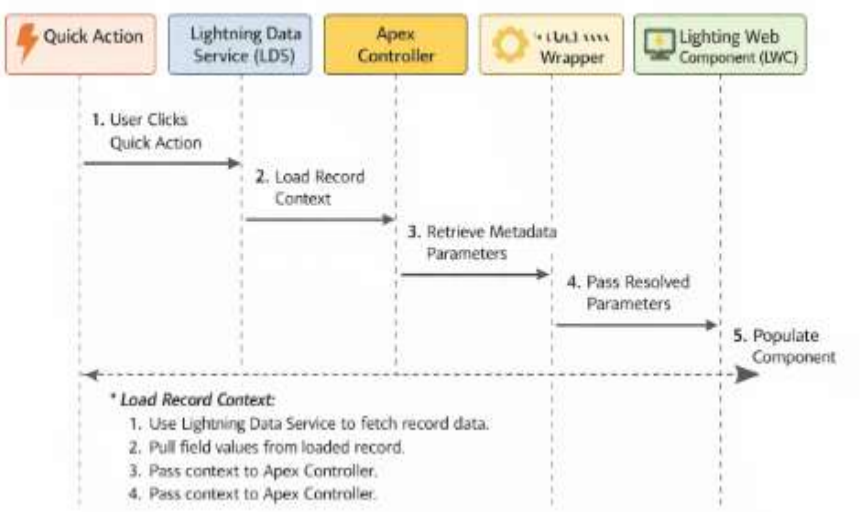


FIGURE 2 Runtime Parameter Extraction Sequence

Then, metadata-driven configuration was used by the system to figure out how the component should work for the particular Quick Action that was executed. Every Quick Action had a link to a corresponding entry in a custom metadata type that allowed the attributes like default priority, required sections, and recommended next steps to be defined by the administrators. When the Quick Action was performed, the wrapper recognized the action name, asked Apex for the corresponding metadata record, and got the configuration values.

After that, business rules in Apex figured out the dynamic defaults by looking at the record and metadata inputs, e.g., a shorter follow-up time for early-stage opportunities, higher priority for the top-tier accounts, or more qualification questions if the opportunity owner was from a certain group of roles. The wrapper, via public @api properties in the LWC, such as defaultPriority, showAdvancedQuestions, and followUpDate, put together with the help of these reconstructed parameters, devamlılığında LWC'ye injected them. In this way, the LWC was able to render as if it had directly received custom parameters, while Salesforce's Quick Action framework does not allow for parameter passing explicitly.

4.4. QUICK ACTION SETUP

Basically, the Quick Action in this study was the tool used to open the dynamic form. Therefore, the way it was set up was the most important part of the entire SummitTech solution. SummitTech developed a local Quick Action specifically for the Opportunity object and changed its type to Lightning Component, directing it to the Aura wrapper that was responsible for parameter reconstruction and context initialization. The action was named "Qualify Lead" and added to the Opportunity page layout under Salesforce Mobile & Lightning Experience Actions so that sales reps could access it consistently across desktop and mobile interfaces.

After the setup was done, the Quick Action turned up as a button, visually separating itself from other buttons on the record page. When users clicked the button, Salesforce opened a modal containing the wrapped Lightning Web Component, which displayed a fully pre-populated qualification form based on the derived parameters.

It is worth noting that the Quick Action configuration alone didn't offer a way to pass custom values to the component, but its neat integration with the page layout made it possible for users to have a smooth, one-click operation while the wrapper logic that was underneath dealt with all the extraction of the contextual data and preparation of the dynamic form.

TABLE 2 Functional Responsibilities and Outcomes Of The Multi-Layer System Architecture

Component Layer	Responsibility	Outcome
Quick Action	Launches wrapper with only recordId	Entry point
Aura Wrapper	Retrieves record + metadata, computes parameters	Produces final parameter set
LDS	Supplies context from Opportunity record	Enables dynamic logic
Apex	Fetches metadata + applies business rules	Returns structured settings
LWC	Renders UI using injected parameters	Pre-populated, dynamic form

5. RESULTS AND DISCUSSION

The parameter-reconstruction technique, along with the other changes shown in the figures above, has the effect of making it look like the Quick Action intrinsically passes the parameters; the goal of pre-populating forms or customizing user experiences depending on real-time business conditions is thus fulfilled.

The main idea is that Lightning Data Service, metadata-driven logic, and wrapper-based techniques are used in such a way that a system is capable of understanding the context even if explicit parameters have not been provided through the Quick Action.

Results from the application of the suggested parameter-reconstruction approach indicate that there is a possibility of such flexible and context-aware behavior of Lightning components accessed from Quick Actions without the need for native parameter passing.

This is illustrated by the parameter reconstruction technique, along with other changes, that Quick Action intrinsically passes the parameters. Thus, the goal of pre-populating forms or customizing user experiences based on real-time business conditions is accomplished.

5.1. EXPECTED BEHAVIOR OF PARAMETERS

The method showed the behavior that was specified by re-enacting: Lightning components were quite efficient in getting and using default values that mirrored the state of the record, business rules given in the metadata, and conditions figured out in Apex. Thus, for instance, priorities, recommended follow-up times, or the revealing of additional form fields for next steps could simply have been made default automatically without the need for any further action. Even if the LWC had not received any direct parameters from the Quick Action, it was demonstrated with the correct initial values. That was the point: parameters were created and delivered in such a manner that the local calling code could not recognize them as actual parameters passing.

5.2. IMPROVEMENTS OVER PREVIOUS APPROACHES

Developers were using very inflexible Apex logic, URL hacks, or manual entry most of the time to get dynamic forms to behave before this approach. These methods were not maintainable, and, for example, URL parameters are now blocked in Lightning Experience. So, the method that is being used now removes the unsupported patterns that were used before and, at the same time, offers more flexibility. In particular, metadata-driven configuration empowers administrators to change the rules of behavior without writing any code. Also, the introduction of a wrapper component helps with better separation of different parts, so the LWC can stay UI-focused while the calculation is done from the outside. On the whole, this greatly improves the level of maintenance and correctness as well.

5.3. PERFORMANCE CONSIDERATIONS

Regarding efficiency, the solution causes the system to slow down slightly because of LDS loading and metadata retrieval. Although these are asynchronous operations, they usually finish in a very short time, especially if the small amount of data is taken into account. So, the users get the very first visual feedback due to the appearance of loading indicators. Apex calls are fairly light as they only fetch a few metadata records, and LDS caching is even more efficient for repeated interactions. The system speed is still up to the level of enterprise UI standards and, in many cases, it is quicker than going to another page or manually typing data.

5.4. DEVELOPER PRODUCTIVITY GAINS

Through this architecture, the developer workflow gets much better. Developers do not need to hardcode component logic for each business unit or workflow. Instead, they use metadata and component composition. The wrapper absorbs the complicated logic; thus, the LWC can be simpler. This separation reduces debugging time and increases reuse is more used. Developers are able to make different Quick Actions by utilizing the same components without the necessity of writing new logic; they just specify new metadata values. Besides that, the method is in line with Salesforce's recommendation for configuration rather than customization, thus different teams can collaborate with fewer changes in the code.

5.5. POTENTIAL ENHANCEMENTS

Later changes could be to store more dynamic rules in the metadata, thereby lessening the dependence on Apex logic. For instance, a declarative rule engine would enable business analysts to set up the behavior more accurately. There is also an upgrade that involves caching the metadata results for consecutive Quick Action use, which will improve performance. If Salesforce decides to extend Quick Action parameter functionalities for LWCs in the future, this framework would still be compatible, as the wrapper can be gradually removed.

6. CONCLUSION AND FUTURE SCOPE

While figuring out how to pass parameters for a Lightning component that is called via Quick Actions, the limitations of Salesforce's current framework were not only quite obvious, but also developers' inventive ways to overcome them were revealed.

It turned out that Quick Actions do not allow parameters to be passed directly; however, one can quite sufficiently reconstruct the parameters by using a mixture of Lightning Data Service, metadata-driven configuration, Apex logic, and wrapper-based orchestration. The example helped to elucidate the theoretical notions by showing how a complex, dynamic form could be made to 'understand' the context and, therefore, be filled with the corresponding values just by employing recordId as the starting point. Apart from making the interaction more user-friendly, this, in fact, led to the improvement of consistency, reduction of errors made manually, and the confirmation of the theoretical component behavior that is not static but flexible and context-aware.

Efficient parameter handling should be at the top of the list of priorities for companies that rely on Quick Actions to make their workflows more straightforward, automate their decision-making, and lower their process friction. If sales, service, or operations teams were to depend on components that automatically adjust to the current business context, parameter passing would no longer be just a technical issue but rather a main factor that influences productivity, data accuracy, and user satisfaction. The hybrid solution here reveals that the problem of architecting in a way that most effectively leverages the benefits of both Aura and Lightning Web Components while also considering platform metadata as a kind of central configuration source is very significant. Developers, in this way, are adhering to Salesforce best practices by building solutions that are not only scalable and maintainable but also remain valid even when there is no native platform support for custom Quick Action parameters.

In a similar way, Salesforce could make the issue significantly better in the future. As an example, the addition of feature-rich Quick Action APIs, full LWC support for parameter bindings, and more event-driven communication could make component interactions on-the-fly much simpler than they are now. Furthermore, the increase in the use of dynamic forms, platform metadata, and declarative automation makes it possible to recalculate parameters in a more advanced way without the need for custom code. As the platform is transitioning to a more modular, metadata-driven architecture, there are going to be new possibilities that will allow developers to send the parameters directly to LWCs, thus reducing the need for wrapper components and computational workarounds. Until then, the methods described here are reliable, future-ready, and thus enterprises can take them as a basis to provide intelligent, efficient, and user-focused Lightning experiences.

REFERENCES

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

[2] V. B. Pogadadanda, "Optimizing AXI-Based Memory Subsystems in Modern SoC Designs," *International Journal of Emerging Trends in Multidisciplinary Research*, pp. 64–83, Mar. 2025, doi: 10.67228/30715636/WCMEAI-2025P105.

[3] J. Yin, "Salesforce- Usability of Lightning Web Components," *Theseus.fi*, 2019. https://www.theseus.fi/items/34d231b1-16f6-4520-b928-cc396655eacf.

[4] D. Takkalapally and S. Domala, "LogSpect-AI: Predictive Log Intelligence for Autonomous Performance Assurance," *American International Journal of Computer Science and Technology*, vol. 5, pp. 11–22, 2023, doi: 10.63282/3117-5481/aijcst-v5i2p102.

[5] J. Yu, *Salesforce Lightning Reporting and Dashboards*. Packt Publishing Ltd, 2017. Accessed: Nov. 13, 2026. [Online]. Available: https://books.google.com/books?id=jJZGDwAAQBAJ

[6] L. Sriram Datla and S. Sai Malay, "From Drift to Discipline: Controlling AWS Sprawl Through Automated Resource Lifecycle Management," *American Journal of Cognitive Computing and AI Systems*, vol. 8, pp. 20–43, June 2024, Accessed: Nov. 13, 2026. [Online]. Available: https://ajccai.org/index.php/publication/article/view/12

[7] J. Keel, *Salesforce.Com Lightning Process Builder and Visual Workflow*. Berkeley, CA: Apress, 2016. doi: 10.1007/978-1-4842-1691-0.

[8] S. S. Allenki, "Reducing Security Vulnerabilities With Encryption, IAM, and Regular Audits," *International Journal of Emerging Trends in Computer Science and Information Technology*, vol. 4, pp. 265–275, 2023, doi: 10.63282/3050-9246.ijetcsit-v4i1p127.

[9] Y. S. K. Reddy Sudini, "A Data Migration Platform That Never Stops the Business," *International Journal of Modern Innovations and Emerging Trends*, vol. 7, no. 2, pp. 01–04, Nov. 2024, doi: 10.67228/30715628/IJMIET-2024PII5R7K.

[10] Sharif Shaalan, *Salesforce for Beginners: A Step-by-Step Guide to Creating, Managing, and Automating Sales and Marketing Processes*. Packt Publishing Ltd, 2020. [Online]. Available: https://books.google.com/books?id=-4LoDwAAQBAJ

[11] Vanapalli, Satyendra Kumar. "Business Process Reengineering Using CRM Workflow Automation." *International Journal of Intelligent Automation & Robotics Engineering* vol. 7, no. 2, pp. 01-17, 2024.

[12] M. Kommuru, "From Challenges to Solutions: Key Pain Points and Turnarounds in AI Systems," *International Journal of Artificial Intelligence & Digital Transformation*, vol. 8, pp. 01–20, 2025, doi: 10.67228/30713315/ijaidt-2025pii2r5s.

[13] A. Fawcett, *Salesforce Lightning Platform Enterprise Architecture: Architect and Deliver Packaged Applications That Cater to Enterprise Business Needs*. Packt Publishing Ltd, 2019. [Online]. Available: https://books.google.com/books?id=wlu8DwAAQBAJ

[14] R. S. Rao Nimmala, "Building World-Class Compute Platforms: Validation, Quality, and Performance Engineering," *International Journal of Intelligent Automation & Robotics Engineering*, vol. 8, no. 1, pp. 01–18, May 2025, doi: 10.67228/30715725/IJIARE-2025PI5N7R.

[15] P. Goodey, *Salesforce CRM Admin Cookbook.: Solutions to Help You Implement, Configure, and Customize Your Business Applications With Salesforce CRM and Lightning Experience*. Packt Publishing Ltd, 2017. [Online]. Available: https://books.google.com/books?id=TvNFDwAAQBAJ.

[16] V. B. Pogadadanda, "RTL Sign-Off Methodologies for Large-Scale ASIC Designs," *International Journal of Modern Innovations and Emerging Trends*, vol. 6, no. 2, pp. 01–17, Dec. 2023, doi: 10.67228/30715628/IJMIET-2023PII9V5Q.

[17] S. P. Teja Perumalla, "From Wallets to Financial Operating Systems: Redefining Consumer Finance Interfaces," *International Journal of Computer Science and Engineering Innovations*, pp. 117–126, Nov. 2025, doi: 10.64137/3107-9458/ICACSIS-111.

[18] J. Keel, "Lightning Process Builder Basics," *Salesforce.com Lightning Process Builder and Visual Workflow*, pp. 199–213, 2016, doi: 10.1007/978-1-4842-1691-0_7.

[19] Y. S. K. R. Sudini, "Designing Hybrid Identity: Workforce and Customer Access at Scale," *American International Journal of Computer Science and Technology*, vol. 7, pp. 128–132, 2025, doi: 10.63282/3117-5481/aijcst-v7i5p111.

[20] sL. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, 3rd ed. Upper Saddle River (N.J.): Addison-Wesley, 2013.

[21] D. Takkalapally, "6GSyn: AI-Driven Synthetic Data Generation for Next-Generation Wireless Performance Evolution," *International Journal of Emerging Trends in Computer Science and Information Technology*, vol. 6, pp. 168–177, 2025, doi: 10.63282/3050-9246.ijetcsit-v6i1p120.

[22] R. S. Rao Nimmala, "Engineering Reliability Lessons From Enterprise Servers, GPUs, and Data Centers," *International Journal of Applied Data Science & Modern Computing*, vol. 6, no. 1, pp. 01–18, May 2023, doi: 10.67228/30713498/IJADSMC-2023PI2P6J.

[23] Venkatesh Satla, "API-First Enterprise Architecture for Government Digital Transformation," *International Journal of Artificial Intelligence & Digital Transformation*, vol. 7, no. 1, pp. 01–20, Jan. 2024, doi: 10.67228/30713315/IJAIDT-2024PIM3R7P.

[24] H. P. Breivold, I. Crnkovic, and M. Larsson, "A Systematic Review of Software Architecture Evolution Research," *Information and Software Technology*, vol. 54, no. 1, pp. 16–40, Jan. 2012, doi: 10.1016/j.infsof.2011.06.002.

[25] L. Sriram Datla and S. Sai Malay, "Zero-Touch Decommissioning in Healthcare Clouds: An Automation Playbook With AWS Nuke and GuardRails," *Los Angeles Journal of Intelligent Systems and Pattern Recognition*, vol. 5, pp. 1–24, Mar. 2025, Accessed: Nov. 13, 2026. [Online]. Available: https://www.lajispr.org/index.php/publication/article/view/65.

[26] S. Kumar Vanapalli, "Best Practices in Salesforce and Microsoft Dynamics CRM Customization," *International Journal of Computer Science and Engineering Innovations*, pp. 82–91, Nov. 2025, doi: 10.64137/3107-9458/ICACSIS-108.

[27] I. Gorton, *Essential Software Architecture*. Springer Berlin Heidelberg, 2006.

[28] S. S. Allenki, "Troubleshooting Backup, Restore, and Data Export Failures in Relational Databases," *International Journal of AI, BigData, Computational and Management Studies*, vol. 6, pp. 127–137, 2025, doi: 10.63282/3050-9416.ijaibdcms-v6i2p115.

[29] Venkatesh Satla, "Enterprise Application Resilience Using Kubernetes, Kafka, and Reactive," *International Journal of Emerging Trends in Multidisciplinary Research*, pp. 167–185, Mar. 2025, doi: 10.67228/30715636/WCMEAI-2025P111.

[30] B. Eden, "Metadata and Its Application," *Library Technology Reports*, vol. 38, no. 5, pp. 1–78, 2002, Accessed: Nov. 13, 2026. [Online]. Available: https://link.gale.com/apps/doc/A93085596/AONE

[31] B. Bater, A. Hibbert, and K. Banga, "Business Metadata Capturing Enterprise Knowledge * Microsoft SharePoint 2007 Technologies Planning Design and Implementation * SOA in Practice the Art of Distributed System Design," *ITNOW*, vol. 50, no. 4, pp. 27–27, July 2008, doi: 10.1093/itnow/bwn082.

[32] M. Kommuru, "Assessing the Limitations of LangChain in Production Environments," *American International Journal of Computer Science and Technology*, vol. 6, pp. 71–83, 2024, doi: 10.63282/3117-5481/aijcst-v6i4p107.

[33] S. P. Teja Perumalla, "Smart Passive Tokenization: A New Model for On-Demand Credentialization," *International Journal of Emerging Trends in Multidisciplinary Research*, pp. 186–201, Mar. 2025, doi: 10.67228/30715636/WCMEAI-2025P112.

[34] J. Kiswani, M. Muhanna, and A. Qusef, "Using Metadata in Optimizing the Design and Development of Enterprise Information Systems," in *2017 8th International Conference on Information and Communication Systems (ICICS)*, IEEE, Apr. 2017, pp. 188–193. doi: 10.1109/iacs.2017.7921969.

[35] "Metadata Principles and Practicalities," *www.dlib.org*. https://www.dlib.org/dlib/april02/weibel/04weibel.html

[36] D. Takkalapally and M. R. Takkellapally, "AdaptCacheAI: Adaptive Hybrid Caching With Machine-Learned Eviction for Dynamic Cloud Workloads," *International Journal of Emerging Research in Engineering and Technology*, vol. 4, p. 165, Jan. 2023, doi: 10.63282/3050-922x.ijeret-v4i1p118.

[37] Y. S. K. Reddy Sudini, "Building a Modern Data Lake for Enterprise Analytics," *International Journal of Data Engineering and Intelligent Computing*, vol. 7, no. 2, pp. 01–04, Sept. 2024, doi: 10.67228/30715717/IJDEIC-2024PII7P2X.

[38] V. K. B. Pogadadanda, "Emerging trends in chiplet interconnect architectures for next-generation SoCs," International Journal of Humanities Science Innovations and Management Studies, vol. 2, no. 3, pp. 44–59, 2025.

[39] Madhurima Kommuru, "Multi-Cloud Vs Hybrid Cloud: Aligning Cloud Strategy With Business Goals," *International Journal of Emerging Trends in Multidisciplinary Research*, pp. 219–239, 2025, doi: 10.67228/30715636/WCMEAI-2025P114.

[40] D. Takkalapally, "HoloSearchAI: AI-Driven Latency Optimization Framework for Distributed Search Systems," *International Journal of Emerging Trends in Computer Science and Information Technology*, vol. 4, p. 217, Jan. 2023, doi: 10.63282/3050-9246.ijetcsit-v4i3p122.

[41] Y. S. K. Reddy Sudini, "Engineering Reliability, Resiliency, and High Availability," *International Journal of Modern Research in Science & Engineering*, vol. 7, no. 1, pp. 01–05, Feb. 2024, doi: 10.67228/30716357/IJMRSE-2024PI3R6T.

[42] S. K. Vanapalli, "Building High-Impact Training Programs for CRM Adoption in Enterprises," *American International Journal of Computer Science and Technology*, vol. 7, pp. 148–157, 2025, doi: 10.63282/3117-5481/aijcst-v7i3p112.

[43] V. Satla, "Building Secure JWT-Based Authentication Frameworks for Enterprise Java Applications," *American International Journal of Computer Science and Technology*, vol. 3, no. 2, pp. 39–52, Mar. 2021, doi: 10.63282/3117-5481/aijcst-v3i2p105.

[44] R. S. Rao Nimmala, "Enterprise System Validation: A Complete Guide to Servers, Storage, and Infrastructure Testing," *International Journal of Emerging Trends in Multidisciplinary Research*, vol. 8, no. 1, pp. 01–18, June 2025, doi: 10.67228/30715636/IJETMR-2025PI6H4W.

[45] V. B. Pogadadanda, "High-Speed PCIe and CXL Integration Challenges in Modern ASICs," *International Journal of Emerging Trends in Multidisciplinary Research*, vol. 8, no. 1, pp. 01–18, May 2025, doi: 10.67228/30715636/IJETMR-2025PI8Q5R.

[46] Y. S. K. Reddy Sudini, "Architecting Enterprise-Scale Cloud Networking," American International Journal of Computer Science and Technology, vol. 6, no. 1, pp. 116-119, 2024, doi: 10.63282/3117-5481/AIJCST-V6I1P111

[47] Takkalapally, DevenderRao, and Mahender Rao Takkellapally. "GC-TuneHFT: AI-Based Garbage Collection Optimization in High-Frequency Trading Environments". *American International Journal of Computer Science and Technology*, vol. 5, no. 6, Nov. 2023, pp. 25-37

[48] M. Kommuru, "Smarter AI Agents: Optimizing Tokens the Right Way," *International Journal of Machine Learning and Predictive Analytics*, vol. 8, pp. 01–19, 2025, doi: 10.67228/3142788x/ijmlpa-2025pi3w9k.

[49] S. Kumar Vanapalli, "Enhancing Workforce Productivity Through CRM Training and Enablement," *International Journal of Emerging Trends in Multidisciplinary Research*, pp. 103–118, Mar. 2025, doi: 10.67228/30715636/WCMEAI-2025P107.

[50] M. Kommuru, "Selecting the Right Data Pipeline Architecture for Reliability and Scale," *International Journal of Data Engineering and Intelligent Computing*, vol. 8, pp. 01–16, 2025, doi: 10.67228/30715717/ijdeic-2025pii5l2z.

[51] V. B. Pogadadanda, "CDC and RDC Analysis Methodologies for Timing-Critical FPGA Systems," *International Journal of Computer Science and Engineering Innovations*, pp. 138–150, Nov. 2025, doi: 10.64137/3107-9458/ICACSIS-113.

[52] D. Takkalapally and M. R. Takkellapally, "AI-SynPerf: Synthetic Data Intelligence Framework for 5G Mobile Performance Simulation," *International Journal of Emerging Trends in Computer Science and Information Technology*, vol. 5, pp. 182–194, 2024, doi: 10.63282/3050-9246.ijetcsit-v5i1p118.

[53] Venkatesh Satla and Santhosh Boggula, "Modernizing Legacy Banking Applications With Spring Boot Micro Services," *International Journal of Commerce, Finance and Digital Economy*, vol. 4, no. 1, pp. 01–18, Jan. 2021, doi: 10.67228/3071642X/IJCFDE-2021PI5E1P.

[54] V. B. Pogadadanda, "High-Bandwidth NoC Design for AI Accelerator Platforms," *American International Journal of Computer Science and Technology*, vol. 7, pp. 134–146, 2025, doi: 10.63282/3117-5481/aijcst-v7i2p111.

[55] R. S. Rao Nimmala, "The Art of System Debugging: Root Cause Analysis for Hardware and Software Engineers," *International Journal of Modern Innovations and Emerging Trends*, vol. 7, no. 1, pp. 01–18, 2024, doi: 10.67228/30715628/IJMIET-2024PI9X2M.

[56] D. Takkalapally, "ShiftLeft-AI: Machine Learning Framework for Proactive Performance Assurance in CI/CD Pipelines," *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, vol. 5, pp. 285–296, 2024, doi: 10.63282/3050-9262.ijaidsml-v5i4p126.

[57] M. Kommuru, "Why Usage-Based Pricing Is Breaking Developer Workflows—and How to Build Local AI Agents," *International Conference on Advanced Computing, Security, and Intelligent Systems*, pp. 11–24, 2025, doi: 10.64137/3107-9458/icacsis-102.

[58] V. B. Pogadadanda, "Scalable NoC Integration Strategies for AI and Data Center Accelerators," *International Journal of Applied Data Science & Modern Computing*, vol. 7, no. 2, pp. 01–18, Nov. 2024, doi: 10.67228/30713498/IJADSMC-2024PII4X8N.

[59] S. K. Vanapalli, "Identifying Operational Inefficiencies Through CRM Data Analysis," *International Journal of Data Engineering and Intelligent Computing*, vol. 7, pp. 01–13, 2024, doi: 10.67228/30715717/ijdeic-2024pii2a7m.

[60] Venkatesh Satla and Santhosh Boggula, "Docker and Kubernetes Adoption Strategies for Large-Scale Java Applications," *International Journal of Applied Data Science & Modern Computing*, vol. 4, no. 1, pp. 01–19, May 2021, doi: 10.67228/30713498/IJADSMC-2021PI6U1K.

[61] L. Sriram Datla, "Smarter Provisioning in Healthcare IT: Integrating SCIM, GitOps, and AI for Rapid Account Onboarding," *Journal of Artificial Intelligence & Machine Learning Studies*, vol. 8, pp. 75–96, Dec. 2024, Accessed: Nov. 13, 2026. [Online]. Available: https://jaimls.org/index.php/publication/article/view/12.

[62] R. S. Rao Nimmala, "Hardware Validation at Scale: Best Practices for High-Performance Computing Platforms," *International Journal of Data Engineering and Intelligent Computing*, vol. 7, no. 2, pp. 01–20, Nov. 2024, doi: 10.67228/30715717/IJDEIC-2024PII7L5V.

[63] M. Kommuru, "IDE Agent Mode Explained: When to Use It and Which Model to Pick," *International Journal of Modern Innovations and Emerging Trends*, vol. 8, pp. 01–20, 2025, doi: 10.67228/30715628/ijmiet-2025pii4t1p.

[64] D. Takkalapally and S. Domala, "AutoScriptAI: Model-Driven Framework for Autonomous Script Generation and Lifecycle Maintenance," *American International Journal of Computer Science and Technology*, vol. 6, pp. 58–72, 2024, doi: 10.63282/3117-5481/aijcst-v6i3p106.

[65] V. B. Pogadadanda, "Design and Integration of HBM/LPDDR Subsystems in Chiplet SoCs," *International Journal of Data Engineering and Intelligent Computing*, vol. 7, no. 2, pp. 01–17, Sept. 2024, doi: 10.67228/30715717/IJDEIC-2024PII3K9M.

[66] S. K. Vanapalli, "The Evolution of Business Analysis in the Era of Intelligent CRM Systems," *International Journal of Emerging Trends in Multidisciplinary Research*, vol. 7, pp. 01–16, 2024, doi: 10.67228/30715636/ijetmr-2024pi5e3b.

[67] Venkatesh Satla, "Event-Driven Enterprise Systems Using Kafka and Spring Boot," *International Journal of Data Engineering and Intelligent Computing*, vol. 4, no. 2, pp. 01–18, July 2021, doi: 10.67228/30715717/IJDEIC-2021PII8H4A.

[68] M. Kommuru and A. N. K. Doodala, "Performance Bottlenecks Necks in Data Heavy Python Applications," *International Journal of Applied Data Science & Modern Computing*, vol. 6, pp. 01–19, 2023, doi: 10.67228/30713498/ijadsmc-2023pii2q7p.

[69] R. S. R. Nimmala, "Mastering Platform Validation: Real-World Strategies for Modern Computing Systems," *American International Journal of Computer Science and Technology*, vol. 5, pp. 68–79, 2023, doi: 10.63282/3117-5481/aijcst-v5i1p106.

[70] R. Shiramalla, "Design and Development of an Intuitive Salesforce Interface Using Lightning Web Components for Cross-Application Opportunity Data Access," *American International Journal of Computer Science and Technology*, vol. 5, pp. 36–46, 2023, doi: 10.63282/3117-5481/aijcst-v5i2p104.

[71] S. K. Vanapalli, "Standardizing Enterprise Workflows Through CRM-Driven Process Modeling," *American International Journal of Computer Science and Technology*, vol. 6, pp. 118–127, 2024, doi: 10.63282/3117-5481/aijcst-v6i2p112.

[72] V. B. Pogadadanda, "Network-on-Chip Architecture Generation Using Arteris FlexNoC," *American International Journal of Computer Science and Technology*, vol. 6, pp. 120–132, 2024, doi: 10.63282/3117-5481/aijcst-v6i1p112.

[73] M. Kommuru, S. Thatraju, and A. N. K. Doodala, "Challenges of Deep Learning in Natural Language Processing: A Healthcare-Oriented Perspective," *International Journal of Machine Learning and Predictive Analytics*, vol. 3, pp. 01–19, 2020, doi: 10.67228/3142788x/ijmlpa-2020pii4wv3.

[74] Venkatesh Satla, "Enterprise UI Modernization With Angular and React in Hybrid Cloud Environments," *International Journal of Innovative Research in Humanities & Technology*, vol. 5, no. 2, pp. 01–19, Sept. 2022, doi: 10.67228/3071561X/IJIRHT-2022PII7U1T.

[75] R. S. R. Nimmala, "The Validation Engineer's Handbook: Testing, Debugging, and Delivering Reliable Hardware Platforms," *American International Journal of Computer Science and Technology*, vol. 4, pp. 35–49, 2022, doi: 10.63282/3117-5481/aijcst-v4i3p104.

[76] S. K. Vanapalli, “Harnessing Artificial Intelligence Within CRM to Improve Customer Insights,” *International Journal of Artificial Intelligence & Digital Transformation*, vol. 1, pp. 01–15, 2023, doi: 10.67228/30713315/ijaidt-2023pi7y3l.

[77] M. Kommuru, "Concurrency Is Hard: Java Vs Python Pitfalls You Can't Ignore," *International Journal of Modern Research in Science & Engineering*, vol. 5, pp. 01–18, 2022, doi: 10.67228/30716357/ijmrse-2022pii8e1y.

[78] V. B. Pogadadanda, "Design Optimization Techniques for Xilinx FPGA Memory IPs," *International Journal of Intelligent Automation & Robotics Engineering*, vol. 6, no. 1, pp. 01–17, Mar. 2023, doi: 10.67228/30715725/IJIARE-2023PI2K7T.

[79] R. Shiramalla and S. Katangoori, "Zero Trust Architecture for Salesforce LWC Using Adaptive Authentication Models," *American International Journal of Computer Science and Technology*, vol. 7, pp. 123–135, 2025, doi: 10.63282/3117-5481/aijcst-v7i1p110.

[80] Venkatesh Satla, "Implementing CI/CD Pipelines for Microservices Using Jenkins, Github Actions, and AWS Codepipeline," *International Journal of Emerging Trends in Multidisciplinary Research*, vol. 5, no. 2, pp. 01–17, Nov. 2022, doi: 10.67228/30715636/IJETMR-2022PII7K2S.

[81] R. S. Rao Nimmala, "From Silicon to System: A Practical Guide to CPU, GPU, and Platform Validation," *International Journal of Modern Research in Science & Engineering*, vol. 5, no. 2, pp. 01–19, July 2022, doi: 10.67228/30716357/IJMRSE-2022PII1Z8K.

[82] M. Kommuru, "Instruction Files Are the Key to Useful AI in Software Development," *International Journal of Artificial Intelligence & Digital Transformation*, vol. 8, pp. 01–15, 2025, doi: 10.67228/30713315/ijaidt-2025pi6h4v.

[83] Venkatesh Satla, "Infrastructure as Code With Terraform: Accelerating Enterprise Cloud Adoption," *International Journal of Modern Research in Science & Engineering*, vol. 5, no. 1, pp. 01–16, Feb. 2022, doi: 10.67228/30716357/IJMRSE-2022PI4Q7S.

[84] S. K. Vanapalli, "Predictive Analytics in CRM: Transforming Data Into Strategic Assets," *American International Journal of Computer Science and Technology*, vol. 5, pp. 47–58, 2023, doi: 10.63282/3117-5481/aijcst-v5i2p105.

[85] Rupesh Shiramalla, "Optimizing Workflow Automation in Salesforce Using Event-Driven Pub/Sub Models," *International Journal of Emerging Trends in Multidisciplinary Research*, pp. 84–102, Mar. 2025, doi: 10.67228/30715636/WCMEAI-2025P106.

[86] Madhurima Kommuru, *Architecting at Scale: From Monoliths to Intelligent Microservices*. Libertatem Media Private Limited, 2024. Accessed: Nov. 13, 2026. [Online]. Available: https://books.google.com/books?id=QLwEEgAAQBAJ

[87] Venkatesh Satla and Santhosh Boggula, "Designing High-Performance Angular Applications With NgRx and Reactive Programming," *International Journal of Modern Innovations and Emerging Trends*, vol. 5, no. 1, pp. 01–15, Apr. 2022, doi: 10.67228/30715628/IJMIET-2022PI9Q4L.

[88] S. S. Allenki, "Query Optimization in PostgreSQL With Autovacuum, Indexing, and Pg_Stat_Statements," *International Journal of Computer Science and Engineering Innovations*, pp. 104–116, Nov. 2025, doi: 10.64137/3107-9458/ICACSIS-110.

[89] L. S. Datla, "Postmortem Culture in Practice: What Production Incidents Taught Us About Reliability in Insurance Tech," *International Journal of Emerging Research in Engineering and Technology*, vol. 3, pp. 40–49, 2022, doi: 10.63282/3050-922x.ijeret-v3i3p105.

[90] S. K. Vanapalli, "Data-Driven Decision Making: Integrating AI Tools With CRM Platforms," *International Journal of Applied Data Science & Modern Computing*, vol. 6, pp. 01–21, 2023, doi: 10.67228/30713498/ijadsmc-2023pi7y3l.

[91] V. Satla, "OAuth2 and JWT Security Patterns for Cloud-Native Microservices," *American International Journal of Computer Science and Technology*, vol. 5, pp. 70–80, 2023, doi: 10.63282/3117-5481/aijcst-v5i2p107.

[92] M. Kommuru, "Tokenization Explained: What It Is, Why It Matters, and How to Work Around Its Limits," *International Journal of Machine Learning and Predictive Analytics*, vol. 8, pp. 01–18, 2025, doi: 10.67228/3142788x/ijmlpa-2025pii8c2n.

[93] S. K. Vanapalli, "Reducing Financial Fraud Using Machine Learning and CRM Data Models," *International Journal of Machine Learning and Predictive Analytics*, vol. 6, pp. 01–20, 2023, doi: 10.67228/3142788x/ijmlpa-2023piq8m4r.

[94] Venkatesh Satla, "Cloud-Native Java Applications on AWS ECS and Kubernetes," *International Journal of Applied Data Science & Modern Computing*, vol. 6, no. 2, pp. 01–16, Aug. 2023, doi: 10.67228/30713498/IJADSMC-2023PII6T2W.

[95] Y. S. K. Reddy Sudini, "Leading an Enterprise On-Premises to Cloud Migration," *International Journal of Applied Data Science & Modern Computing*, vol. 5, no. 1, pp. 01–05, June 2022, doi: 10.67228/30713498/IJADSMC-2022PI4S3P.

[96] Rupesh Shiramalla, "Code Quality Optimization in Salesforce Using Predictive Static Analysis Models," *International Journal of Computer Science and Engineering Innovations*, pp. 25–36, Nov. 2025, doi: 10.64137/3107-9458/ICACSIS-103.

[97] S. K. Vanapalli, "Designing Scalable CRM Architectures for Global Enterprises," *American International Journal of Computer Science and Technology*, vol. 4, no. 4, pp. 36–46, 2022, doi: 10.63282/3117-5481/aijcst-v4i4p104.

[98] Devenderrao Takkalapally, "AccelLoadAI: Hardware-Assisted Packet Processing for AI-Enhanced Load Balancing and VIP Management," *International Journal of Computer Science and Engineering Innovations*, pp. 60–70, Nov. 2025, doi: 10.64137/3107-9458/ICACSIS-106.

[99] M. Kommuru, "How to Spot and Fix Python Performance Issues Remotely- Strategies and Techniques," *American International Journal of Computer Science and Technology*, vol. 7, pp. 115–127, 2025, doi: 10.63282/3117-5481/aijcst-v7i5p110.

[100] V. B. Pogadadanda, "Architecture and Validation of UltraRAM Readback Mechanisms in FPGA Devices," *International Journal of Machine Learning and Predictive Analytics*, vol. 6, no. 1, pp. 01–21, May 2023, doi: 10.67228/3142788X/IJMLPA-2023PI8X5N.

[101] Venkatesh Satla, "Observability in Distributed Systems Using Prometheus, Grafana, and ELK Stack," *International Journal of Data Engineering and Intelligent Computing*, vol. 6, no. 2, pp. 01–18, Oct. 2023, doi: 10.67228/30715717/IJDEIC-2023PII7L5H.

[102] Y. S. K. Reddy Sudini, "An Intelligent Agent for Root-Cause Detection and Remediation," *International Journal of Intelligent Automation & Robotics Engineering*, vol. 5, no. 1, pp. 01–04, May 2022, doi: 10.67228/30715725/IJIARE-2022PI9M4V.

[103] S. K. Vanapalli, "Optimizing Customer Lifecycle Management Using Salesforce and Dynamics 365," *International Journal of Commerce, Finance and Digital Economy*, vol. 4, pp. 01–14, 2021, doi: 10.67228/3071642x/ijcfde-2021pi8g4r.

[104] S. S. Allenki, "Applying Cloud Security Best Practices in Regulated Environments," *American International Journal of Computer Science and Technology*, vol. 5, pp. 48–60, 2023, doi: 10.63282/3117-5481/aijcst-v5i3p105.

[105] S. K. Vanapalli, "Bridging Business and Technology: The Role of CRM in Digital Transformation," *International Journal of Artificial Intelligence & Digital Transformation*, vol. 2, pp. 01–17, 2022, doi: 10.67228/30713315/ijaidt-2022pii9j1t.

[106] V. S. Satla and Ramesh Jindam, "Building Fault-Tolerant Microservices With Spring Cloud and Resilience Patterns," *International Journal of Intelligent Automation & Robotics Engineering*, vol. 6, no. 2, pp. 01–19, Dec. 2023, doi: 10.67228/30715725/IJIARE-2023PII9B2K.
[107] S. K. Vanapalli, "AI-Driven Fraud Detection in Financial Systems: A CRM-Centric Approach," *International Journal of Machine Learning and Predictive Analytics*, vol. 5, pp. 01–14, 2022, doi: 10.67228/3142788x/ijmlpa-2022pii4a5x.
[108] Y. S. K. Reddy Sudini, "Building a Company From the Ground up: The Cloud Carve-Out," *International Journal of Commerce, Finance and Digital Economy*, vol. 6, no. 2, pp. 01–05, Oct. 2023, doi: 10.67228/3071642X/IJCFDE-2023PII4N6C.
[109] Mallikarjun Vppalapati, "Designing Storage for Energy Limits Rather Than Performance Targets," *International Journal of Emerging Trends in Multidisciplinary Research*, pp. 48–63, Mar. 2025, doi: 10.67228/30715636/WCMEAI-2025P104.
[110] Venkatesh Satla and Ramesh Jindam, "Scalable Data Processing Architectures With Mongodb, Postgresql, and Kafka," *International Journal of Data Engineering and Intelligent Computing*, vol. 7, no. 1, pp. 01–16, May 2024, doi: 10.67228/30715717/IJDEIC-2024PI3M8G.
[111] DevenderRao Takkalapally, "Kernelsight-AI: Intelligent Kernel Profiling and Bottleneck Detection in Cloud Infrastructure," *International Journal of Emerging Trends in Multidisciplinary Research*, pp. 14–30, Mar. 2025, doi: 10.67228/30715636/WCMEAI-2025P102.
[112] V. B. Pogadadanda, "FPGA Validation Strategies for Multi-Protocol Communication Systems," *International Journal of Innovative Research in Humanities & Technology*, vol. 5, no. 1, pp. 01–22, Apr. 2022, doi: 10.67228/3071561X/IJIRHT-2022PI6R3V.
[113] Venkatesh Satla, "Leveraging OpenShift on AWS (ROSA) for Enterprise Container Platforms," *International Journal of Modern Research in Science & Engineering*, vol. 7, no. 1, pp. 01–18, Mar. 2024, doi: 10.67228/30716357/IJMRSE-2024PI7Z4V.
[114] S. K. Vanapalli, "Transforming Enterprise Operations Through Advanced CRM Customization Strategies," *International Journal of Commerce, Finance and Digital Economy*, vol. 4, pp. 01–14, 2021, doi: 10.67228/3071642x/ijcfde-2021pi5d8f.
[115] Y. S. K. Reddy Sudini, "An Intelligent Agent for Root-Cause Detection and Remediation," *International Journal of Intelligent Automation & Robotics Engineering*, vol. 5, no. 1, pp. 01–04, May 2022, doi: 10.67228/30715725/IJIARE-2022PI9M4V.
[116] V. B. Pogadadanda, "FPGA-Based Aurora 64B/66B Protocol Validation Using GTM Transceivers," *International Journal of Artificial Intelligence & Digital Transformation*, vol. 5, no. 1, pp. 01–17, June 2022, doi: 10.67228/30713315/IJAIDT-2022PI8G3A.
[117] Balkishan Arugula and Suni Karimilla, "Modernizing Core Banking Systems: Leveraging AI and Microservices for Legacy Transformation," *Artificial Intelligence, Machine Learning, and Autonomous Systems*, vol. 9, pp. 36–67, 2025, Accessed: Nov. 13, 2026. [Online]. Available: https://amlas.net/index.php/publication/article/view/14.
[118] Venkatesh Satla, "Designing Secure Healthcare Microservices for State and Federal Systems," *International Journal of Computer Science and Engineering Innovations*, pp. 151–163, Nov. 2025, doi: 10.64137/3107-9458/ICACSIS-114.
[119] M. Vppalapati, "Stateful Infrastructure in a Stateless World: The Control Surface Paradox," *American International Journal of Computer Science and Technology*, vol. 5, pp. 59–69, 2023, doi: 10.63282/3117-5481/aijcst-v5i2p106.
[120] M. Vppalapati, "Automation-Induced Fragility in Highly Reliable Storage Platforms," *American International Journal of Computer Science and Technology*, vol. 5, pp. 60–71, 2023, doi: 10.63282/3117-5481/aijcst-v5i6p106.
[121] M. Vppalapati, "Power-Bound Storage Design: Architecting Systems for Electrical Scarcity," *International Journal of AI, BigData, Computational and Management Studies*, vol. 5, pp. 208–217, 2024, doi: 10.63282/3050-9416.ijaibdcms-v5i1p121.
[122] M. Vppalapati, "Cooling Domains as First-Class Failure Boundaries in Storage Architecture," *American International Journal of Computer Science and Technology*, vol. 6, pp. 96–106, 2024, doi: 10.63282/3117-5481/aijcst-v6i2p110.
[123] Mallikarjun Vppalapati and P. Kumar Talasila, "The Collapse of Predictability in Large-Pool Redundant Storage," *International Journal of Applied Data Science & Modern Computing*, vol. 7, no. 1, pp. 01–15, June 2024, doi: 10.67228/30713498/IJADSMC-2024PI7C3N.
[124] S. K. Vanapalli, "From Data to Decisions: Leveraging CRM Platforms for Business Intelligence," *International Journal of Applied Data Science & Modern Computing*, vol. 4, pp. 01–16, 2021, doi: 10.67228/30713498/ijadsmc-2021pi1z6n.
[125] V. B. Pogadadanda, "Design Approaches for SECDED ECC Architectures in Automotive Systems," *International Journal of Commerce, Finance and Digital Economy*, vol. 5, no. 2, pp. 01–16, Sept. 2022, doi: 10.67228/3071642X/IJCFDE-2022PII2S3A.
[126] Y. S. K. Reddy Sudini, "Leading an Enterprise On-Premises to Cloud Migration," *International Journal of Applied Data Science & Modern Computing*, vol. 5, no. 1, pp. 01–05, June 2022, doi: 10.67228/30713498/IJADSMC-2022PI4S3P.
[127] Balkishan Arugula and Vasu Nalmala, "Migrating Legacy Ecommerce Systems to the Cloud: A Step-by-Step Guide," *Los Angeles Journal of Intelligent Systems and Pattern Recognition*, vol. 3, pp. 342–367, Dec. 2023, Accessed: Nov. 13, 2026. [Online]. Available: https://lajispr.org/index.php/publication/article/view/60