Journal of Computer Science and Technology Studies
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Composable Personalization Architecture: A Scalable Framework for Modular AI-Powered Experience Engineering

Bhargav Trivedi
New Jersey Institute of Technology, USA
Corresponding author: Bhargav Trivedi. **Email:** hellobtrivedi@gmail.com

| ABSTRACT
Personalized experiences have become a more important strategic objective in many industries, but organizations still deal with architectural and other barriers to scaling, experimenting, and maintaining cross-channel loyalty. This article proposes a Composable Personalization Architecture (CPA), built to enable the value of personalization in a modular, layered approach. CPA decouples central personalization functions such as signal collection, decision-making logic, and content delivery into interoperable services that are later connected at shared telemetry and orchestration layers. This separation of concerns allows organizations to integrate new models more easily, ultimately allowing for controlled experimentation and adaptation to changing business needs. The architecture has been tested in enterprise environments, and organizations found improved agility, system robustness, and engagement. By delivering personalization as a composable capacity instead of a monolithic feature, CPA more effectively addresses the sustainable and adaptable design of a customer's experience at scale.

| KEYWORDS
Composable architecture, personalization, event-driven systems, telemetry, customer data platforms, explainable AI

| ARTICLE INFORMATION
ACCEPTED: 12 July 2025 **PUBLISHED:** 04 August 2025 **DOI:** 10.32996/jcsts.2025.7.8.53

1. Introduction

In today's rapidly evolving landscape of digital commerce and digital services, a meaningful and individualized customer experience has become a focus of organizations that are looking to differentiate themselves [1]. As users are exposed to and using a variety of channels and devices, including web, mobile, in-store kiosks, etc., organizations are under pressure to respond with relevant, contextual experiences in a timely manner, cognizant of each user's unique preferences, behaviors, and intent.

That said, it is difficult to construct personalized experiences at scale. Many legacy systems are monolithic and tightly coupled, which creates friction in incorporating new data sources or experimenting with personalized content and an incapacity to tap into new or emerging touchpoints unless new, custom builds are done or significant re-engineering occurs [2]. Even in many of the modern platforms, personalization initiatives or efforts over time are tightly coupled to specific products or teams, creating siloed implementations, poor or disparate user experiences, and limitations in measuring or governing personalization initiatives.

In order to respond to these gaps, an increasing number of architectural models have come to be utilized that emphasize modularity, adaptability, and composability [2]. The design principles of these models favor loosely coupled services, shared data pipelines, and reusable decision logic, which allows personalization systems to be dynamically reconfigured over time without the burden of tight system dependencies. By decoupling the important functions associated with signals that offer users a choice, including the collection of user signals, the logic of the decision making process, the delivery of interoperable content, and the feedback loops operated by the service designers, organizations are able to enjoy more flexibility, scalability, and transparency in the end-to-end personalization of customer interactions [3].

This article presents the Composable Personalization Architecture (CPA), a pragmatic framework that helps teams create, execute, and scale personalization in a modular and interoperable way. The CPA organizes personalization into a set of clear, independent layers, each unique in scope and responsibility related to personalization functions like telemetry collection, decision point (rules or model) evaluation, orchestration, and front-end integration. This structure will lead to iterative experimentation, platform neutrality, and organizational clarity, which will empower teams to innovate, govern, and scale personalization in complicated environments.

2. Principles of Design

The Composable Personalization Architecture is grounded in a set of guiding design principles that unlock flexibility, modularity, and the long-term evolution of personalized experiences [5]. These principles correspond with technical best practices and organizational needs for agility, transparency, and reuse.

2.1 Modularity

Every component of the CPA framework has been designed as a discrete and self-contained service [2]. The discrete nature of the CPA framework enables each of these services - telemetry system, decision engine, and front-end or API integrations - to be designed, deployed, and maintained independently from one another, resulting in fewer cross-team dependencies, shorter development timelines, and localized updates without interference to the entire personalization stack.

2.2 Data Unification

Effective personalization relies on a continuous collection and analysis of user behaviors [4]. CPA advocates for establishing a universal schema for telemetry and customer signals across all channels and services. Organizations can avoid fragmentation across teams and systems by standardizing the capture of data and defining an agreed-upon description of events, such as views of products, interactions with a shopping cart, and content impressions.

2.3 Interoperability

Personalization strategies typically shift as they incorporate new machine learning models, third-party tools, or as the organization's business goals change [5]. CPA's purpose is to support interoperability among these elements with interfaces and protocols that define how they can interact. This way, people will be able to plug in or remove models, rules engines, and front-end experiences with little integration burden, hence minimizing vendor lock-in and fostering continuous innovation.

2.4 Transparency and Explainability

In contexts where individual decisions have consequences for user engagement, revenue, and/or compliance, it is critical to know how and why decisions are made [3, 6]. CPA promotes transparency with recommended traceable decision flows and logging for audits. Regardless of whether personalization is based on heuristics or statistical models, every decision path has a documented and traceable record, which can be explored and explained, helping business parties and keeping auditors satisfied.

2.5 Reusability and Governance

CPA also includes the ability to reuse personalization components - like decision templates, ranking rules, and telemetry definitions - across use cases and teams [5]. Reuse brings more consistency in the user experience while shortening governance and quality assurance activities. Core teams can define and curate a set of approved components, and content teams can create an experience within their own objectives. Overall, these principles provide an architectural framework for a personalization strategy that will be scalable over time as technologies and business context change.

Design Principle	Key Characteristics	Primary Benefits	Implementation Approach
Modularity	Independent, self-contained services	Reduced dependencies, faster development cycles	Microservices architecture
Data Unification	Shared telemetry schema across channels	Consistent insights, reduced fragmentation	Event-driven data pipelines
Interoperability	Well-defined interfaces and protocols	Reduced vendor lock-in, continuous innovation	API-first design
Transparency	Traceable decision flows, audit logging	Regulatory compliance, stakeholder visibility	Explainable AI frameworks

Reusability	Shared components and templates	Consistent UX, simplified governance	Component libraries

Table 1: CPA Design Principles and Their Benefits [2, 3, 4, 5, 6]

3. Reference Architecture

The Composable Personalization Architecture (CPA) is structured in layers [5]. Each layer reflects a particular area of responsibility in the personalization life cycle. Each abstraction/distributed responsibility allows organizations to design or change specific elements to the architecture without negatively impacting other areas, supporting long-term collaborative maintenance, evolution and flexibility.

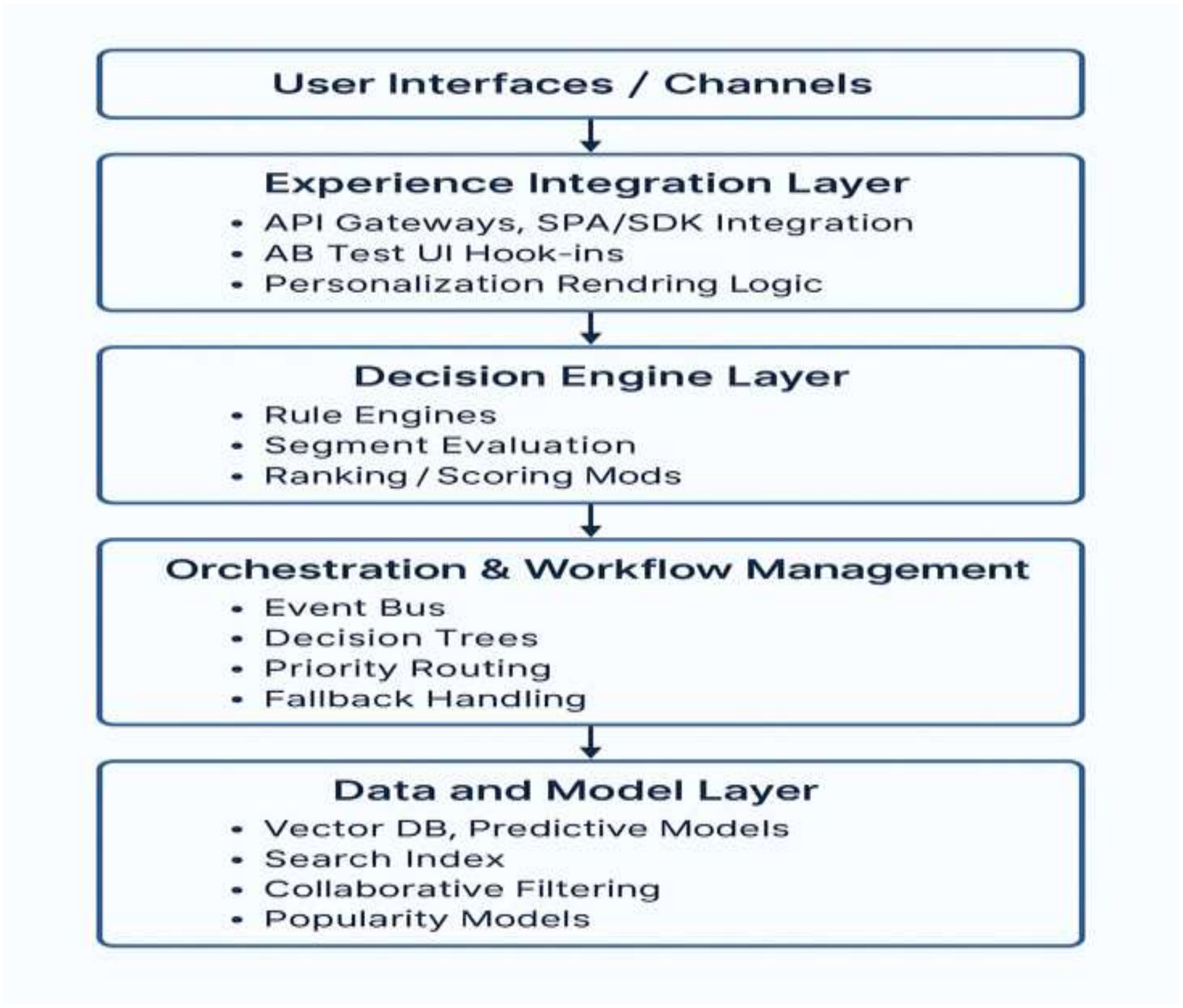

Figure 1: High-level architecture of the Composable Personalization Framework. Each layer is modular and interacts with adjacent layers via defined interfaces to support scalability, reuse, and explainability.

3.1 Experience Integration Layer

The outer, or presentation layer, communicates directly with customer-facing applications--including websites, mobile applications, and in-store displays [2]. It acts as the intermediary between back-end personalization decisions and true front-end user interactions. It pulls personalized content, product recommendations, or ranking results by means of APIs and renders them using the application's presentation logic.

Each channel often has its own rendering constraints and user behavior. Because of this variability, the presentation layer is meant to be adaptable. The presentation layer is also capable of supporting client-side instrumentation to assess how users react to personalized content, thus allowing for downstream analysis of usage engagement and effectiveness [4].

3.2 Decision Engine Layer

The Decision Engine Layer of CPA is the heart of CPA and it evaluates personalization strategies in real time [1]. It is responsible for triggering logic based on user context, business rules, and behavioral data. It can employ rule-based engines for static configurations and algorithmic or model components for dynamic personalization.

The layer has been structured to be composable in nature. For example, a team could combine standard segmentation rules (i.e., geography, loyalty tier) with predictive scoring models (i.e., affinity, odds of churn) to create an experience for the user. Multiple decision strategies can be tested and evaluated at the same time and chosen based on some experimentation findings or contextual relevance [3].

3.3 Telemetry and Signal Collection Layer

Solid personalization relies on quality behavioral data [4]. The Telemetry Layer captures granular user interactions each click or search action, scrolling depth, cart additions, and conversions and sends those details through a single pipeline to the measures. Each event is standardized against a common schema, providing service interoperability and facilitating easier analysis.

The telemetry system is mostly implemented with asynchronous event processing tools, allowing ingestions to scale while minimizing performance impacts on end-user systems [4]. In addition to capturing the signals, they are also saved for training and validating decision models, or executing controlled experimentation.

3.4 Orchestration Layer

The Orchestration Layer manages data and decision flow along the personalization pipeline [5]. It manages which services are activated, under what conditions, how responses are prioritized, and how to implement fallbacks if modules do not activate or return uncertain results.

This event-driven orchestration provides a declarative configuration model that allows teams to define rules for personalization flow without having to hard-code the logic for it [4]. It also enables decision routing at runtime – for example activating one type of recommendation strategy for new users while activating another for returning customers – or dynamically swapping models during an A/B test.

3.5 Data and Model Layer

At the lowest level of the architecture is the Data and Model Layer, which contains all of the underlying intelligence used for personalization, including recommendation algorithms, user preference models, search-rank logic, and even heuristics based on content [1]. Each model is treated as a modular service with a defined interface.

Importantly, the framework enables a model–agnostic architecture so that teams can experiment with multiple models (e.g., collaborative filtering; nearest-neighbor; hybrid) and plug those in based on business objectives or maturity of data [1]. Output from this layer is routed upward to the decision engine, where it is aggregated into a cohesive experience for the user.

These layers create a cohesive architecture for organizations to scale their personalization capabilities in a systematic way [5]. The modular nature of the architecture promotes positive ongoing innovation, as well as a framework for governance, observability, and collaboration across teams throughout the personalization lifecycle.

CPA Layer	Primary Functions	Key Technologies	Integration Points	Failure Handling
Experience Integration	UI rendering, client instrumentation	React, mobile SDKs, CDN	APIs, event tracking	Graceful degradation
Decision Engine	Rule evaluation, model orchestration	Rules engines, ML pipelines	Real-time APIs	Fallback strategies
Telemetry & Signals	Event capture, data standardization	Kafka, event schemas	All touchpoints	Buffering, retry logic
Orchestration	Flow coordination, experiment routing	Workflow engines, A/B platforms	Cross-layer communication	Circuit breakers

Data & Models	ML models, recommendation algorithms	TensorFlow, collaborative filtering	Model serving APIs	Model versioning

Table 2: Layer-by-Layer Functional Mapping [1, 2, 4, 5]

4. Use Case Scenarios

The true power of the Composable Personalization Architecture is its flexibility to fit multiple business contexts without having to completely rebuild your existing systems [2]. By separating the personalization workflow into separate but connected layers, CPA allows for fast experimentation, across-channel consistency, and scalable decisions. Below, the article describes three example scenarios across e-commerce, media streaming, and retail loyalty programs.

4.1 Scenario 1: E-Commerce Product Personalization

Providing relevant product recommendations can be valuable when it comes to improving not only conversion rates but also average order value (AOV) on the e-commerce platform [1]. With CPA, the telemetry layer can capture the user's experience of browsing, which includes product views, search terms, and cart events. That will be normalized and sent to the data/model layer.

After data normalization, collaborative filtering and vector-based retrieval models are then used to generate a candidate list of product suggestions [1]. The decision engine applies ranking rules (based on promos, inventory state, user segments such as first-time shoppers) before the orchestration layer handles run-time experimentation (e.g., A/B tests) and fallbacks if model failure occurs. The experience integration layer serves the suggestions in a carousel on the dynamic homepage or within personalized category areas.

The approach of recomponentizing the components of the application enables merchandising and data science teams to focus on their own areas of expertise, experimenting with either models or business logic without disrupting the core storefront application [2].

4.2 Scenario 2: Media Streaming and Content Discovery

A video-on-demand system might use CPA to personalize content recommendations on the home page of both its web and mobile apps [1]. In that context, the telemetry layer collects implicit feedback - watch time, skips, pause events - so that information can be fed back into the data/model layer to make the user interest models better.

The decision engine uses genre affinity scores, recency bias, and subscription tier rules to make decisions about which titles should be recommended above others [6]. For instance, if there are some high-affinity dramas, binge-watchers may get those prominently positioned in their recommendations, or new releases may be prioritized for more casual, but returning, viewers. The orchestration layer routes new users to a cold-start routing strategy, while returning users receive routing aimed at personalizing pipelines.

The modularity of the architecture also allows integration of editorial content curation (such as festive specials or featured series) with algorithmic results, so that content teams have some control and oversight without having to drop automation [3].

4.3 Scenario 3: Retail Loyalty and Offer Personalization

In a brick-and-click context, CPA can be used to tailor promotional incentives provided to a customer over SMS, email, and an in-store kiosk [6]. Identifying signals from purchase history, loyalty tier, and seasonal trends across online and offline systems is critical to pull promotions together.

In the CPA modular format, a rules-based engine may generate weekly coupons for loyal patrons, while predictive churn analytics may identify at-risk customers for targeted win-back promotions. The orchestration layer resolves active promotional conflicts from the customer perspective, applying eligibility criteria in real time [5].

In the experience layer, the same personalized logic can be surfaced across all the digital touchpoints, meaning the customer gets the same message whether they log into a mobile app or scan a loyalty card in the store [2].

These examples point to the fact that CPA enables data-informed, sustainable, and channel-agnostic personalization [1]. By keeping business rules close to technical architecture, CPA enables teams to deliver an individual customer experience while keeping (business model) operational agility and (technology system/solution) architectural integrity.

Use Case	Telemetry Signals	Decision Logic	Models Used	Integration Channels	Business Metrics
E-Commerce Product Recommendations	Product views, cart actions, search queries	Ranking rules, inventory status, and user segments	Collaborative filtering, vector retrieval	Homepage, category pages, mobile app	Conversion rate, AOV
Media Streaming Discovery	Watch duration, skips, pause events	Genre affinity, recency bias, subscription tiers	Content-based filtering, implicit feedback	Homepage, mobile app, smart TV	Engagement time, retention
Retail Loyalty Offers	Purchase history, loyalty tier, seasonal trends	Promotion eligibility, churn prediction	Rules-based, predictive models	SMS, email, and in-store kiosks	Redemption rate, CLV

Table 3: Use Case Implementation Matrix [1, 6]

5. Evaluation

In order to evaluate the potential application effectiveness of the Composable Personalization Architecture, CPA has been adopted and tested in a number of large-scale digital commerce and service environments [5]. Although each organization implements and uses CPA in varied ways, consistency in improvements has been documented across the board within flexibility of the system, deployment speed, and business outcomes.

5.1 Deployment Velocity and Experimentation Efficiency

One of the earliest benefits seen in CPA adoption is the improved flexibility to deploy and change personalization strategies [2]. Due to the modularity of template logic, underlying data models, and telemetry systems, teams can test new recommendation algorithms, segmentation rules, or ranking methods without needing to make a complete stack change.

In typical enterprise environments, CPA has enabled a 3× improvement in A/B test rollout frequency, reducing iteration cycles from weeks to days [1]. Additionally, integration of new recommendation models can now be completed in under 48 hours, compared to prior timelines of 1–2 weeks. These improvements are attributed to the separation of concerns between layers, which reduces coordination overhead and simplifies governance reviews [2].

5.2 System Resilience and Operational Consistency

The CPA architecture also provides better fault isolation and fallback options, especially via the orchestration layer [5]. For example, it is possible for a model endpoint to fail or send back incomplete results. When that happens, the system can run those models with routing rules that allow it to fall back to the default logic, such as rule-based promotions or ranking by popularity, without degrading the experience from a user perspective.

Under production situations, this has resulted in virtually zero customization downtime during peak load times and a 20 to 30% drop in faults connected to personalization modules visible to users. Such resilience is especially helpful during heavy-traffic events like product debuts or seasonal sales [2].

5.3 Cross-Channel Consistency and User Experience Metrics

Another major opportunity for advancement lies with the consistency of personalized experiences across channels [1]. Because the decision engine and data model services are reused by all of the front-end integrations, users get a consistent recommendation and messaging experience on web, mobile, or email.

Between 8% and 15%, several pilot deployments of CPA have noted a noticeable improvement in user engagement measures, including click-through rate (CTR) and dwell time [5]. Furthermore, conversion rates on custom landing sites have grown 12–18% more than those on static material. These results point to composable personalization's ability to produce significant value for end customers in addition to its improvement of engineering results.

5.4 Maintainability and Team Collaboration

Ultimately, qualitative feedback from engineering and product teams suggested a significant improvement in maintainability and cross-functional collaboration [2]. Clearly defined interfaces between the layers allow data scientists, backend engineers, and front-end developers to work more independently, resulting in fewer bottlenecks and attempts at things that are duplicated.

Teams have reported faster onboarding of new engineers due to well-defined service boundaries, as well as easier debugging and testing of personalization flows due to layer-specific logging and metrics [5]. These organizational benefits are often as impactful as the technical ones, particularly in large or distributed teams.

In conclusion, the CPA framework has been shown to provide considerable evidence of improvement in the delivery of personalization, both technically and in business terms. The metrics by which each implementation reported the improvement were dependent on the implementation, but the evidence suggesting improved agility, reliability, and impact of the personalization indicates that CPA is a viable pattern for scalable and maintainable personalization in complicated systems [1, 2, 5].

Metric Category	Before CPA	After CPA	Improvement	Contributing CPA Features
Deployment Velocity	1-2 weeks per model	48 hours per model	3-7× faster	Modular architecture
A/B Test Frequency	Monthly cycles	3× more tests/quarter	300% increase	Decoupled experimentation
System Uptime	Occasional failures	Near-zero downtime	99.9%+ reliability	Fault isolation
Cross-Channel CTR	Baseline	8-15% increase	Measurable lift	Consistent decision logic
Conversion Rate	Baseline	12-18% increase	Significant improvement	Unified personalization
Developer Onboarding	2-3 weeks	3-5 days	70% reduction	Clear service boundaries

Table 4: Performance Metrics and Improvements [1, 2, 5]

6. Applied Case Study: Retail Personalization at Scale

To demonstrate the practical application of Composable Personalization Architecture (CPA) in a live production environment, the article provides an example of a national retail organization that has physical and digital storefronts [1]. The organization experienced typical struggles: outdated legacy-fitting systems, inconsistent user journeys across diverse channels, and inability to experiment with other personalization tactics.

6.1 Context and Problem Statement

The e-commerce platform used by the organization was originally developed using a tightly coupled monolithic architecture that bound personalization logic to backend systems [2]. This scenario created a very challenging environment to change recommendations, run controlled experiments, and coordinate a cross-channel strategy. Any personalization change would require deploying the entire stack and weak telemetry standardization, limited evaluation of overall performance or user behavior across channels [4].

6.2 CPA-Based Re-Architecture

A modernization initiative was launched to align the personalization system with CPA principles [5]. The transformation focused on five key areas, beginning with layered modularity, where personalization functions such as recommendation logic, A/B testing infrastructure, and UI rendering were separated into independent services with well-defined interfaces [2].

The initiative also introduced signal unification through a common telemetry schema, enabling consistent tracking across web, mobile, and in-store systems [4]. Business rules and model outputs were abstracted into a configurable decision engine, supporting real-time personalization and fallback strategies through composable decision logic [1].

Low-latency content variants and edge-based traffic routing were made possible by edge-based experiments carried out using a serverless A/B testing framework installed through a content distribution network (CDN). At long last, the platform was moved to

Oracle Cloud Infrastructure, hence enhancing scalability and allowing on-demand resource allocation for peak shopping times via cloud-native infrastructure [5].

6.3 Business Impact and Measured Outcomes

The redesigned personalization system provided clear benefits for operating agility as well as user experience [1]. Thanks to more pertinent, dynamically served recommendations, the company saw a 20% rise in cart conversion rates. The number of quarterly experiments deployed rose by 3x as well, therefore speeding up innovation cycles.

Enabled by service-level decoupling, the system showed a 30% reduction in time-to-market for personalisation changes [2]. Better resilience and uptime during holiday peak traffic events, supported by fault-tolerant orchestral and scalable infrastructure, were also noted [5].

6.4 Lessons Learned

The implementation showed how crucial clearly defined service boundaries and schema governance are while adopting a composable architecture [2, 4]. Moreover, it emphasized the need for observation and testing in extracting company knowledge. Above all, it confirmed that personalization is most successful when approached as a composable, whole-system ability rather than a product characteristic [1].

This case confirms CPA as a workable and scalable blueprint for companies in digital commerce trying to update their personalization system while producing quantifiable business results [5].

7. Discussion and Future Work

The Composable Personalization Architecture (CPA) marks a change in the creation of personalizing systems, not only as separate suggestion engines or target rules but also as coordinated ecosystems of interoperable services [2]. Although CPA establishes a distinct architectural pattern that has proven effective across fields, it also brings up serious issues and chances for ongoing improvement.

7.1 Benefits Realized Through Composability

The main understanding of CPA is that when every functional layer is separately created and run, personalization initiatives become much more scalable and bearable [5]. Adoption of CPA has been shown by groups to have advantages in addition to speed of experimentation and system performance, as well as in better developer independence, streamlined governance, and less cross-team interaction [2].

This modular technique fits with new industry developments including headless commerce, API-first platforms, and cloudnative microservices [2]. CPA lets companies quickly iterate, include unique models, and localize logic without sacrificing worldwide consistency by applying these ideas to the area of customization.

7.2 Challenges and Open Considerations

Despite its advantages, implementing CPA introduces new design considerations. Inter-layer latency presents challenges as modular systems require careful coordination to avoid added latency, especially in real-time personalization [5]. Data governance becomes complex as shared telemetry and signal pipelines demand robust data validation, lineage tracking, and schema evolution mechanisms [4].

Versioning and model drift management becomes critical as multiple models and rules operate concurrently, requiring careful management of version compatibility and ensuring predictable behavior over time [1]. Organizational change poses another difficulty as teams might need to embrace new cultural habits and processes supporting distributed ownership of personalization layers [2].

These difficulties highlight the necessity of technology and standards that support the creation of controlled customization, observability, and fault tolerance systems [5].

7.3 New Developments and Prospects

In the future, a number of advancements could improve the CPA framework even further. Increased data privacy regulations like the CCPA and GDPR are addressed with privacy-aware personalization [3]. Future iterations of the CPA may include consent-aware signal collecting, differential privacy methods, and dynamic opt-in procedures. These features would aid in striking a balance between the efficacy of personalization and the moral use of data.

Zero-shot and few-shot personalization represents another frontier where recent advances in foundation models and transfer learning suggest a future where personalization engines require less user history to make relevant predictions [1]. CPA's plug-and-play model layer is well-suited to incorporating such capabilities as they mature.

Federated and edge personalization addresses the trend where devices become more powerful and privacy concerns grow [5]. Edge-based personalization, where models are deployed directly on user devices, may reduce reliance on centralized inference services. CPA can evolve to support hybrid cloud-edge deployments with local fallback logic.

Standardized personalization APIs represent an opportunity to further enhance interoperability, where the community may benefit from shared standards for personalization APIs, decision formats, and experimentation protocols [2]. CPA could serve as a reference implementation to guide such standardization efforts.

The CPA model offers a solid basis for modular, scalable personalization in essence [5]. Still, careful development in reaction to changing user expectations, legal environments, and technical breakthroughs will determine its ultimate success. CPA may continue to be a flexible blueprint for next-generation personalization systems if it remains faithful to its principles of composability, observability, and adaptability.

Conclusion

The composable personalizing design is the contemporary blueprint for delivering uniform, scalable customization throughout digital interactions. By arranging personalization into well-defined layers of data collection, orchestration, decision-making, and experience integration CPA enables firms to grow rapidly without sacrificing clarity or control.

Through its modular design, which fosters innovation by means of easier experimentation and integration, allows crossfunctional teams and lowers technical debt. CPA gives the architectural knowledge needed to elegantly and deliberately handle these problems as user expectations rise and technologies get more complex.

CPA is a versatile base rather than a fixed remedy; it changes to fit changing channels, technology, and business objectives, hence giving businesses pursuing long-term personalization maturity investments a strategic tool.

Funding: This research received no external funding.
Conflicts of Interest: The authors declare no conflict of interest.

References

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," IEEE Trans. on Knowledge and Data Engineering, vol. 17, no. 6, pp. 734–749, 2005. https://ieeexplore.ieee.org/document/1423975

[2] Nicola Dragoni et al., "Microservices: Yesterday, today, and tomorrow," in Present and Ulterior Software Engineering, Springer, pp. 195–216, 2017. https://link.springer.com/chapter/10.1007/978-3-319-67425-4_12

[3] Finale Doshi-Velez, Been Kim, "Towards a rigorous science of interpretable machine learning," arXiv preprint arXiv:1702.08608, 2017. https://arxiv.org/abs/1702.08608

[4] Jay Kreps, "The log: What every software engineer should know about real-time data's unifying abstraction," LinkedIn Engineering Blog, 2014. https://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying

[5] Len Bass, Software Architecture in Practice, Addison-Wesley, 2003.https://www.researchgate.net/publication/224001127_Software_Architecture_In_Practice

[6] Yongfeng Zhang et al., "Explainable Recommendation: A Survey and New Perspectives," ACM Computing Surveys, 2020. https://dl.acm.org/doi/abs/10.1561/1500000066