Logixal's Composable Frontend Framework

Logixal's Commerce Storefront (LCS)

LOGIXAL INC
20 Commerce Drive,
Suite 135, Cranford,
NJ 07016,
USA.

Key Contact: Deepak Gala
Email Id: contact_us@logixal.com
Authors: Anand Ved, Gaurav Deshpande

Introduction

Composable architecture, a modern approach to software design, prioritizes modularity, flexibility, and the ability to assemble distinct components into a cohesive, functioning ecosystem. Composable architecture breaks the system into smaller, interchangeable pieces that can be developed, deployed, and maintained separately allowing organizations to create more scalable, adaptable, and resilient systems that can evolve with changing business needs. In the context of e-commerce, **Composable Commerce** is a term that reflects this architecture.

> By 2025, 60% of new SaaS designs will support both the UI-first and API-first access, making preparedness for composability a common cloud application trait. - **Gartner**

Backend composability enables modularity of backend services and systems to meet specific business needs by decoupling core business logic, data services, and functionality to create a scalable, flexible, and efficient backend architecture.

Similarly, Frontend Composability is an essential component in Digital Commerce. By allowing businesses to build modular, reusable, and adaptable frontends, composability enables faster time to market, better user experiences, and greater innovation—all critical factors for staying competitive in the rapidly evolving digital landscape.

Frontend Composability focuses on creating dynamic, customizable, and reusable UI components enhancing the user experience, while Backend Composability centres on decoupling backend services. When used together, these approaches allow businesses to respond faster to market demands, deliver better customer experiences, and ensure the system can grow and evolve with business needs.

For businesses already using a commerce platform, to transit towards composability involves a strategy of assessing their existing platform, initiate gap analysis, identify **B**est **O**f the **B**reed (BOB) services and define the journey roadmap to either go for greenfield transition or re-platform.

The focus of this whitepaper is to enable businesses to quickly move towards composability of their platforms.

With this vision, we have built composability in frontend and utilized it effectively for backend thereby achieving the benefits of composability across all tiers of the solution.

Challenges

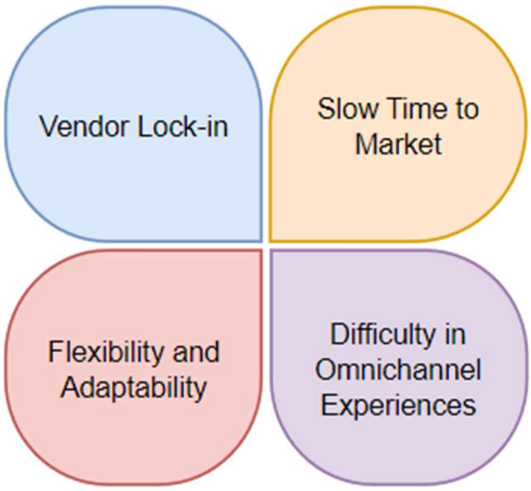

Figure 1: Business Challenges

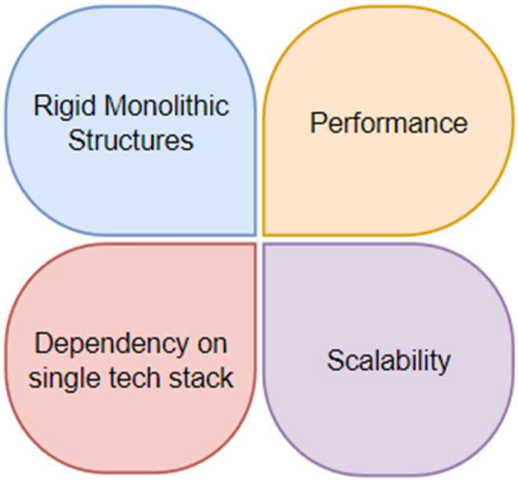

Figure 2: Technical Challenges

Business Challenges	Technical Challenges
Vendor Lock-in as entire stack is tightly integrated in monolith platforms	Rigid Monolithic Structures have tight coupling between frontend and backend
Slow Time to Market leading to loss of competitive advantage and erosion of customer trust	Performance issues lead to poor user experience, increased support costs
Difficulty in supporting consistent Omnichannel Experiences	Entire system is required to be scaled leading to large CAPEX and OPEX
Flexibility & Adaptability is a challenge while integrating new third-party services	Tech stack dependency limiting ability to adopt newer technologies or frameworks

Solution

With evolving business and customer needs, fast changing digital landscape, newer technologies, need of rapid feature launches, faster TTM and easy integrations along with vendor independence, Composable Commerce is emerging as the go-to architecture for businesses prioritizing agility and innovation. A digital commerce platform should be composed of modular components, such as product management, checkout, payments, and customer experience, which can be integrated and orchestrated through APIs enabling businesses to build a custom stack by selecting "best-of-breed" services for each component, providing greater control over their technology and customer experience.

The fastest approach to transforming a legacy system is to decouple the frontend (FE) from the backend, allowing the modernization process to happen incrementally and efficiently.

Transforming Legacy to Modernized platform:

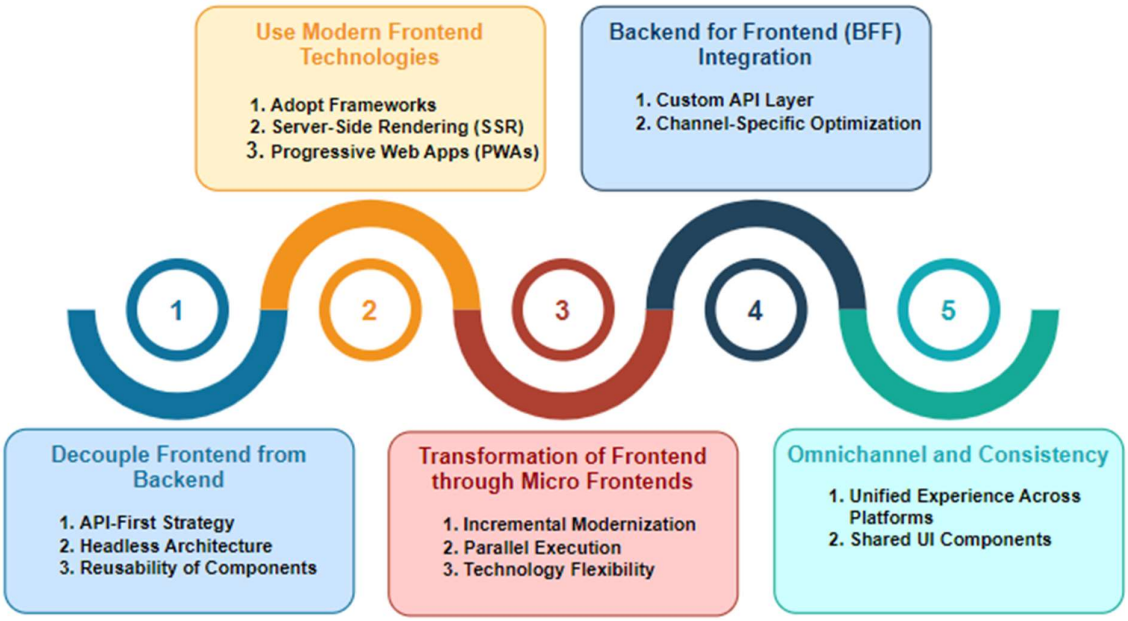

Figure 3 Legacy to Modernized platform transformation

The transformation process involves:

1. Decouple frontend from backend:
    a. API-First Strategy: Adopting API-first approach is core to decoupling, where the backend exposes Domain driven contracts via RESTful or GraphQL APIs creating a clear separation between frontend and backend, enabling independent updates to the UI without needing to overhaul the backend logic.

   b. Headless Architecture: Headless architecture separates content management, business logic, and presentation layers allowing the frontend to consume data from the backend via domain driven contracts, enabling faster updates and iterations to the UI.
   c. Reusability of Services: Frontend modular components can be reused across various platforms, making it easier to extend functionality without redeveloping from scratch.
2. Use of Modern Frontend Technologies
   a. Adopt Modern Frameworks: Modern JavaScript frameworks for the frontend ensures faster and responsive UI which is easier to maintain.
   b. Server-Side Rendering (SSR): Use of frameworks like NextJs, Nodejs and integration of SSR enhances performance by rendering the frontend on the server improving initial load times critical for better user experience and SEO.
   c. Progressive Web Apps (PWAs): PWAs combine the best of web and mobile applications by allowing apps to work offline, send push notifications, and offer app-like performance in the browser, leading to enhanced user experiences and better performance, even in poor network conditions.
3. Transformation of frontend through Micro Frontends:
   a. Incremental Modernization: Smaller, independent and self-contained Micro Frontends allows to incrementally enhance, build, deploy and maintain them separately.
   b. Parallel Execution: Micro Frontends allow different teams to work on separate parts of the UI (e.g., cart, checkout, product page) simultaneously thereby reducing bottlenecks and accelerates the transformation process without impacting the entire system.
   c. Technology Flexibility: While minimising disruption and allowing fast modernization, Micro Frontends allow gradual introduction to modern technologies (like React or Vue.js) while keeping parts of the legacy frontend operational.
4. Backend for Frontend (BFF) Integration:
   a. Custom Aggregation API Layer: A tailored aggregator BFF layer ensures that the frontend receives specific data needed from multiple backend microservices, speeding up frontend performance and avoiding over-fetching or under-fetching data which is a widespread problem in monolithic systems.
   b. Channel-Specific Optimization: BFF allows each frontend (e.g., web, mobile, tablet) to have its own performance optimized API layer, accelerating onboarding for different channels with minimal impact to other channels.
5. Omnichannel and Consistency:

a. Unified Experience across Platforms: Consistent and omnichannel experiences, by decoupling frontend, are essential for modern commerce, where customers expect seamless interactions across devices.
b. Shared UI Components: Componentizing the frontend ensures UI elements reusability across different platforms, reducing development time while ensuring a consistent user experience.

Logixal's Composable Frontend Logixal implements a MACH applied flexible, scalable, and modular architecture for our composable enterprise commerce solutions with an ecosystem of best-of-the-breed solutions, providing our customers flexibility, pluggability and swap-ability to choose as per their needs.

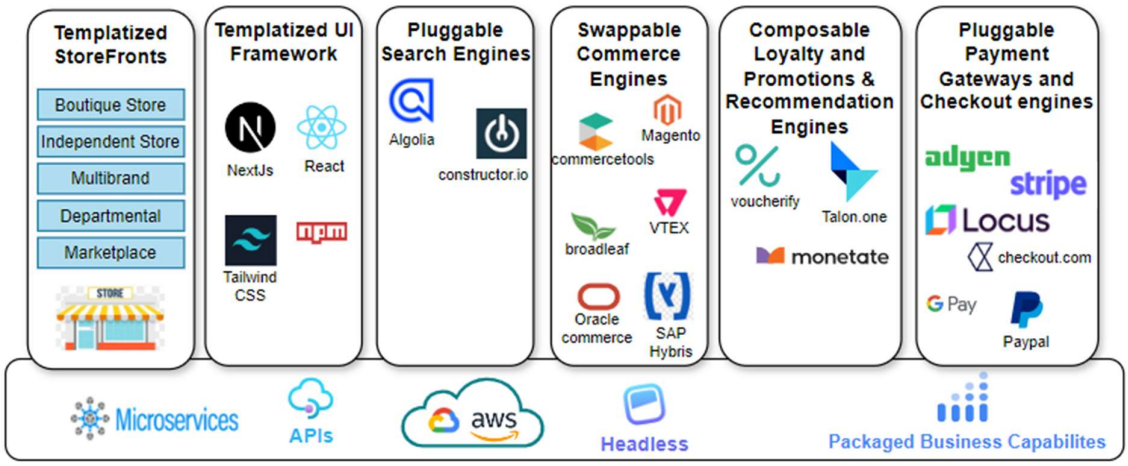

Figure 4: Composable Commerce

Logixal's Composable Frontend is built on two major genres:

- Customizable and templatized storefronts – **L**ogixal **C**ommerce **S**torefront **(LCS)**
- Highly flexible backend – **B**ackend **F**or **F**rontend (**BFF**).

Logixal Commerce Storefront (LCS)

Composable Frontend architecture, latest evolution in the front-end development landscape builds upon the principles of Micro Frontends by introducing a higher level of abstraction and flexibility. In a composable architecture, front-end components can be composed dynamically at runtime, allowing for greater customization and personalization. With Frontend Composability, we achieve:

- Dynamic Composition by on-the-fly arrangement of components using templatized layouts for our Storefronts.
- Decoupled Components by promoting modularity and flexibility.
- API-driven by enabling loose coupling via Domain driven contracts.
- Headless CMS integration to provide flexible and scalable content delivery.

Logixal's Templated Storefront with various templates for storefronts, implementing best practices of Frontend Composability to build dynamic, reusable, flexible and scalable user interfaces.

LCS Frontend is managed by Composable CMS system to provide ease of customization for Storefront content (text, images, videos) and provides an admin portal for templatize layout management.

Logixal Commerce Storefront provides readily available accelerator templates for Boutique, Independent, Multi brand, Marketplace and Departmental stores for the customer to quickly utilize out of the box frontends for their store with minimal brand customization efforts.

Frontend components enable developers to build highly customizable user experiences while remaining independent of the backend system's structure.

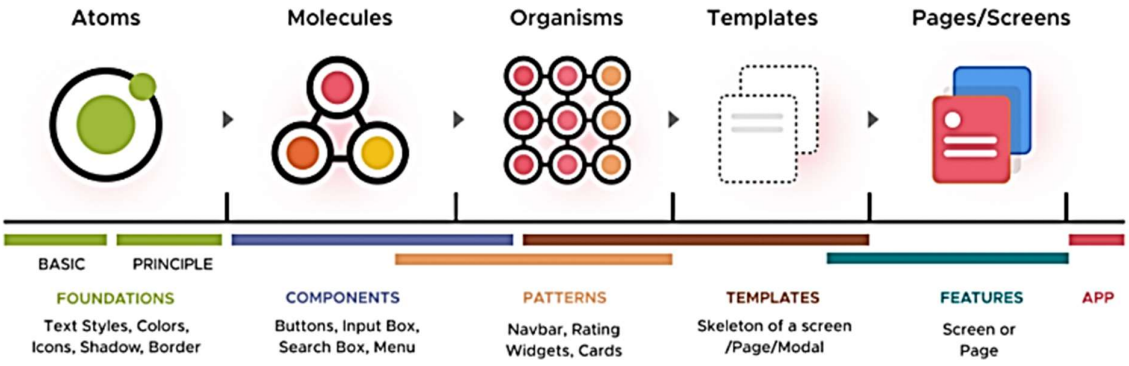

Figure 5 Building blocks for UI

Major components of LCS include:

Templatized UI Framework built using:

- Micro Frontends – smaller, independent composable building blocks consisting of atoms, molecules, organisms.
- Reusable UI components – employed across multiple interfaces.
- Consists of 80+ reusable component libraries, Atomic Design pattern, Storybook and is CDN, SSG, ISG, CSR, SSR, SEO enabled & ADA, PWA and OWASP compliant.

Storefront UI Admin consisting of:

- Tenant Management: Ability to manage Tenant and associated attributes.
- Store Management: Multiple Storefront templates and Tenant Storefront association.
- Users Integration: Ability to control Admin Rights and Tenant Based User Access Rights.
- Layout Management: Page Builder and ready to use components OOTB.
- Configuration controls: Feature controls and integration attribute controls.

Backend For Frontend (BFF)

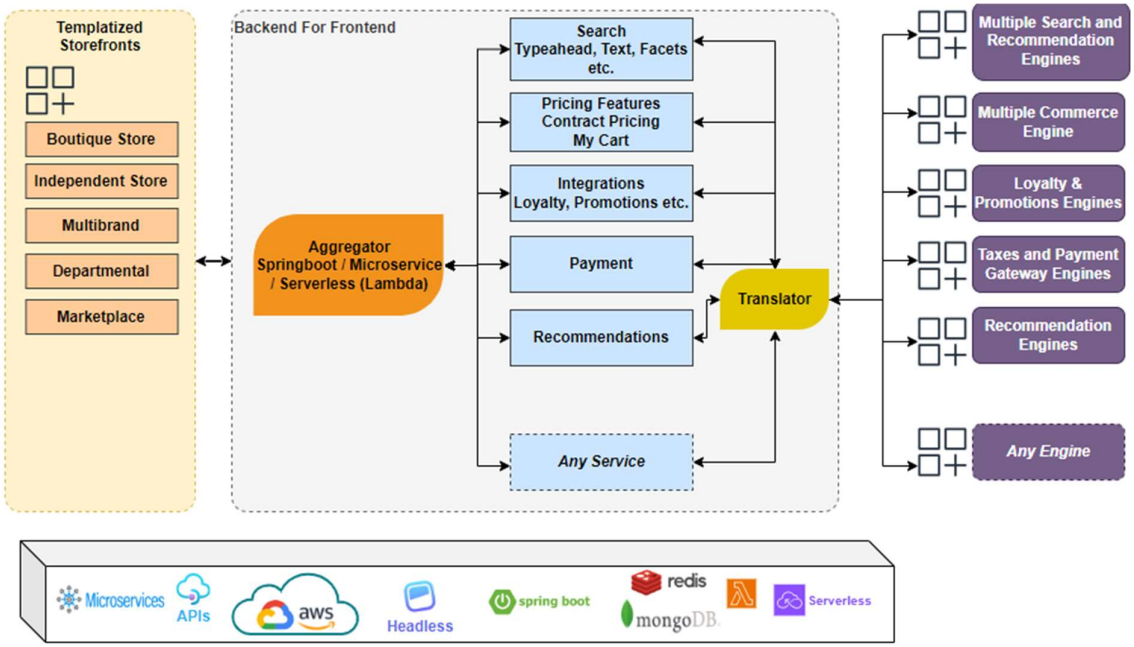

Figure 6: Backend For Frontend architecture

Backend **F**or **F**rontend (**BFF**) solution is based on MACH architecture and caters to the needs of translating communication between domain services to achieve composability. BFF is deployable both on-premises as well as on cloud.

Major components of BFF include:

- Commerce Store Interoperability
- Aggregator: Aggregator is configurable at store-level component responsible for initiating & forwarding multiple requests and aggregating response from these services and build a single standard JSON response for the Storefront channels
- Individually scalable generic domain-driven solution agnostic Microservices.
- Cloud Native supported by Spring boot & serverless using Lambda & Google Cloud Functions
- Pluggable Search Engines: Integrates with various MACH allied solutions like Algolia, Broadleaf search, Contructor.io
- Swappable Commerce Engines: Integrates with Commerce-tools, VTex, Broadleaf, Magento, Oracle Commerce
- Composable Loyalty and Promotions Engines: Integrates with Talon.one
- Choice of Payment Gateways: Integrates with Ayden, Stripe, Checkout
- Pluggable Recommendation Engines: Integrates with home-built DES, AWS Personalize
- BOB solutions and integrations

Benefits Delivered

Business Benefits	Technical Benefits
Decoupling of frontend and backend, composable architectures reduce dependency on any single vendor allowing integration with best of the breed multiple backends, services, and tools for the business needs.	With Frontend Composability each component is independent and allows for the separation of concerns. This makes it easy to update, replace, or add new features with minimal impact, enabling faster and more agile development.
With reusable components, businesses can quickly compose new interfaces or roll out features across different platforms. This accelerates time to market by allowing the use of pre-built or easily customizable frontend components.	Smaller composable micro frontends often utilize APIs (REST or GraphQL) to fetch only the data required for each component, leading to more efficient data retrieval and better performance.
Reusable components in composable architectures can be deployed across multiple channels, ensuring consistency while still allowing customization for each platform. This makes it easier to offer a seamless omnichannel experience.	Frontend Composability enables independent scaling of individual components optimizing resource usage and performance.
Composable Frontends allow easy integration of third-party services via APIs enabling easy adaptation. For example, businesses can quickly add or replace a payment provider or customer service tool without needing to refactor the entire frontend.	Frontend Composability allows for a mix-and-match approach, where businesses can choose the best tools, adopt modern technologies and frameworks for specific frontend components.

Summary

In summary, Fronted and Backend for Frontend Composability overcomes the challenges of rigid, slow, and Monolithic Frontends by providing modularity, flexibility, and agility. It allows businesses to rapidly adapt their user interfaces, improve scalability, enhance performance, and deliver consistent, personalized experiences across multiple channels, all while maintaining a future-proof architecture.

With LCS Platform, you can seamlessly integrate your existing systems or choose from a wide array of pre-built integrations to create a harmonious, omnichannel customer experience. This forward-looking methodology centres on the selection of the most suitable platforms for each aspect of your business, empowering you to seamlessly integrate these platforms into a harmonious ecosystem, thus ensuring a consistent, cutting-edge CX across all customer touchpoints. Our modular approach allows you to optimize your resources and budget, generating higher return on investment.

Logixal's Frontend integration utilizing Templatized UI Storefront has reduced the onboarding time of new customers by 40% on the platform.

About Logixal

Since 2005, Logixal is focused on composable solutions & digital banking enterprises. Logixal has partnered with industry leading platforms to provide custom implementations or in-house enterprise solutions. For over 18 years we have been providing services to Retail, Luxury Goods, Manufacturers, Distributors, Media, Telecom, and Subscription model-based businesses serving customers across Americas, Europe, Asia, Africa & Middle East.

Visit http://www.logixal.com for more information.

Footnotes/Sources

- MACH Alliance https://machalliance.org/
- Gartner: https://www.gartner.com/en/insights