genesis

Code Less, Do More™

The Low-code Advantage

A technology briefing on the Genesis platform

2021

Introduction

I am delighted to welcome you to our second edition of the Genesis 2021 Expo broadcasts, one that we have devoted entirely to the technological development of our Low-Code Application Platform.

James Harrison
Genesis Co-Founder and CTO

Low-code is definitely a strong trend in IT at the moment and for good reason: it delivers business applications faster, cheaper, and with lower risk. The Genesis LCAP was built specifically for financial markets use cases and this is one of the things that really sets us apart from other low-code platforms. Few cross-industry LCAPs have the low latency and high throughput imperative for financial markets and most lack the tools to build applications that understand financial instruments.

What is also unique about Genesis is how we work with our clients who can buy "ready-made" applications, partner with us to build applications, or directly purchase the platform to create the applications on their own. Our library of industry applications is growing very rapidly (as it should because we build these applications using low-code), and we have highlighted several of them in previous Expo broadcasts, but you will find a complete and up-to-date list here.

Today, we want to update you on the evolution of the Genesis platform itself.

Last year, we completely revamped the web presentation layer in the platform. We did this in response to what we felt the market really needed. We saw that when it came to web development, our clients and prospects tended to fall into one of three camps of competing JavaScript frameworks, React, Angular and Vue, with clients clearly having their favorites.

This posed a problem to us: which framework do we pick? If we opt for one, we risk alienating a subset of clients yet picking them all would put an enormous burden

What is also **unique about Genesis** is how we work with our clients who can **buy "ready-made" applications**, partner with us to **build applications**, or directly purchase the platform to **create the applications on their own.**

We made the bold decision to become entirely **framework-agnostic.** [...] We would then appeal to all the clients **regardless of framework preferences.**

on our platform development team. What we also found is that particular frameworks lend themselves particularly well to certain use cases, while others did not. After careful consideration, we made the bold decision to become entirely framework-agnostic. And in doing so we adopted Web Components across the entire platform, sitting a layer below each of the frameworks, but still providing specific framework hooks. We would then appeal to all the clients regardless of framework preferences.

We introduced the Genesis Web Component strategy in our August Expo last year, but in our February Expo, Aaron Mendez, our Program Manager for web strategy, went into this in more detail.

Our second technical update focuses on GPAL, the Genesis Platform Abstraction Layer. This platform was born out of the real-life experience of building applications, analyzing where developers were spending their time, and how we could eliminate unnecessary steps, as well as ramp up the reusability of components with far fewer bugs. The GPAL initiative is very comprehensive and was introduced to financial markets in our September and February Expos.

In our February Expo, Jay Taylerson, VP of Application Development discussed parts of the GPAL that focus on real-time data distribution to user interfaces.

As usual, our Expo audience asked some searching and interesting questions so we include those and the full reply of our Genesis panelists. Martin Sreba, Global Head of Strategic Accounts, who led the broadcast, will conclude this technical paper. Martin, Aaron and Jay are part of an amazing team that tripled global revenues in 2020, and is continuing to enhance the platform, and grow our ecosystem of partners and industry applications. You can find their complete presentations here, and reach out to us directly if you have any questions or comments, or would like a fuller demo.

Over to you guys!

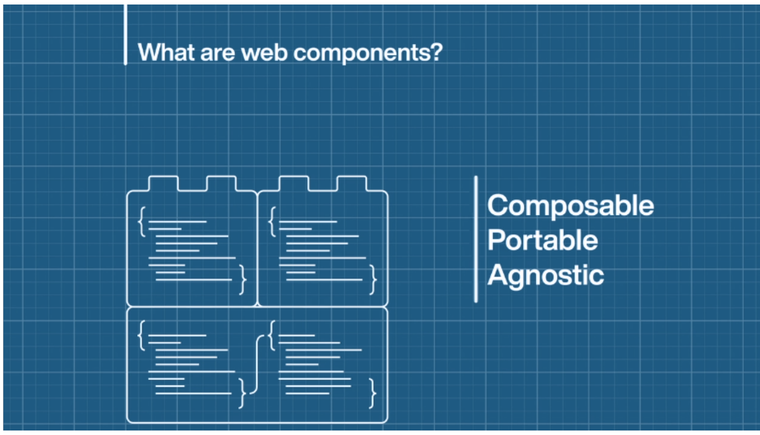
One of the goals of our design process is to support **out-of-the-box visual signatures**

Aaron Mendez
Genesis Program Manager for Web Strategy

Web components are a set of web platform APIs that allow you to create new custom, reusable, encapsulated HTML tags to use in web pages and web apps. They have been around for some time but are now coming to the fore with major adoptions by Salesforce, Google, IBM, and a slew of financial markets firms adopting them for in-house development. Web components run natively in modern browsers, and as such tend to be lighter and faster than, say, Angular or React components. When used with modern JavaScript and TypeScript they also free developers from the walled gardens and endless churn of heavy JavaScript frameworks.

Today, we are taking a look at some specific web components in the Genesis platform, how they are constructed and maintained, and how this sets up application developers to quickly and easily compose powerful user interfaces.

The Genesis LCAP has the concept of Lego bricks: reusable components snapped together quickly to express complex workflows, and build new financial markets applications. It is a concept we apply across both the server and UI layers of the platform. In the creation of web UIs, the web components are the Lego bricks.



In a previous Expo, we showed how micro user interfaces built with Genesis can interoperate with other web tools through FDC3. Today, we're going to touch on the UX journey and developer experience of Genesis UI where there are three aspects to consider.

First, the design system and design language, second, the component library itself. And third, how components are composed into applications. We'll save the third aspect for a dedicated future Expo event.

So, to the design language.

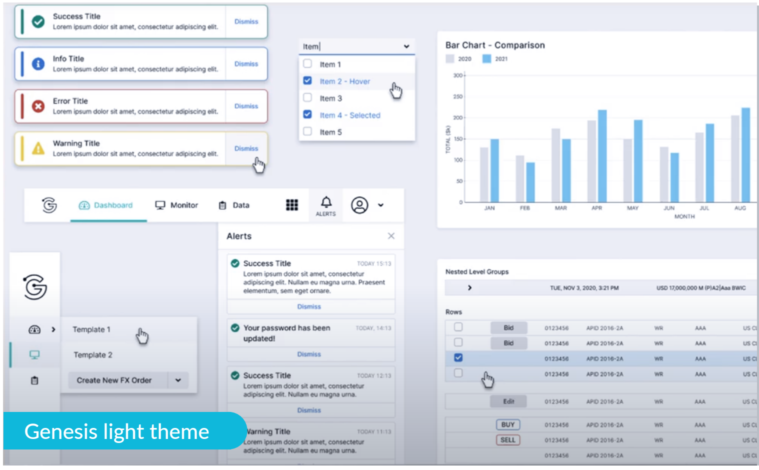
The Genesis design language can be thought of as a scalable framework of decisions and team behaviors across a product portfolio to converge on a cohesive experience.

One of the goals of our design process is to support out-of-the-box visual signatures with light and dark Genesis elements. Here we show the default light theme, and the corresponding dark theme, showing several more of the common UI components that get composed into end applications.

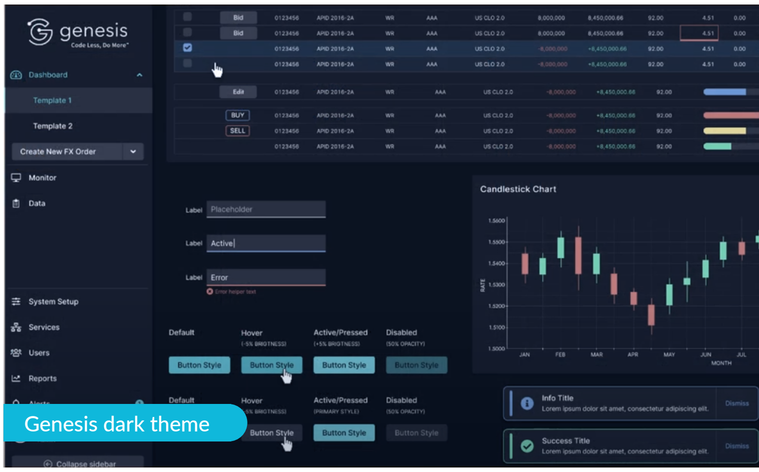
A design system is not just a classification of components, but an evolving process to develop superior and consistent user interfaces. It has to be scalable and dynamic as UIs need to evolve. It creates an inventory of components that can be used as building blocks for both designers and developers. Furthermore, it eliminates inconsistencies by being the single source of truth. And finally, it streamlines the design and development process of the component library itself.

The Genesis design system has to be flexible because the platform is used to build applications for clients that bring their own strong branded visual signatures. This means the system must both support white labeling and assure our solid usability underpinnings and continue to benefit the end product. And what is more, we need to quickly and efficiently effect this across the whole library.

What does such a UI design system consist of? It includes building blocks such as color palettes, typographic scales, icon libraries, page layout, grid, utilities, and component primitives. From these fine-grained building blocks, UI patterns are assembled into more coarse-grained component modules, templates, and workflows. The design system comes with a set of rules including design principles and implementation guidelines to ease decision-making both now and in the future.



Genesis light theme



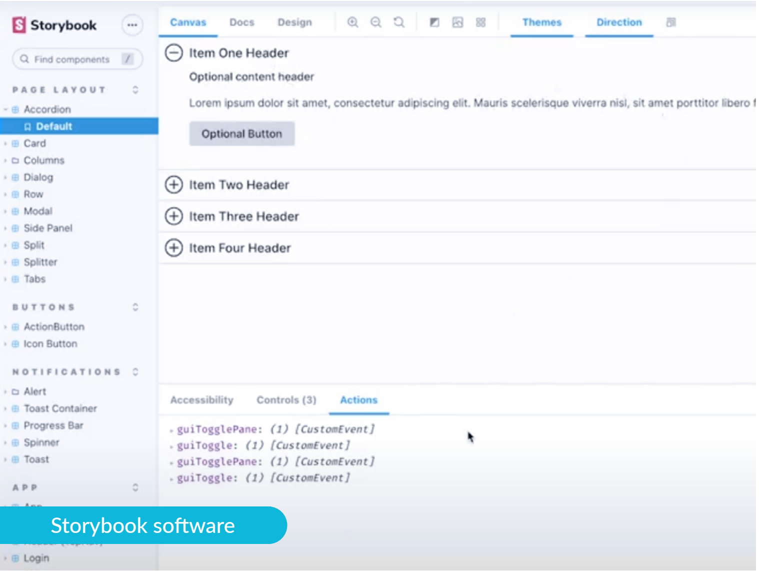
Genesis dark theme

Now let's turn to the component library.

The Genesis engineering team has put together a comprehensive set of reusable components for composing UIs. The components are targeted to cover the needs of a wide range of financial market applications. They are adaptable to be extended with additional styling and navigation. The idea is to radically speed up the build process of each specific application.

The library was designed from a look-and-feel perspective using the industry-standard tool Figma. The specific behaviors and functions of the components were developed in TypeScript by the Genesis web engineering team. The resulting libraries are fully documented, and browsable in another industry-standard tool called Storybook. The components themselves are distributed using NPM packages into application development environments.

Storybook is a very viable open-source tool. Our first Storybook screenshot takes us inside the storybook environment for application developers and designers; it is like a testing sandbox, API docs style guide, and visual design reference all rolled into one.



Item One Header

Optional content header

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris scelerisque viverra nisi, sit amet porttitor libero f

Optional Button

Item Two Header

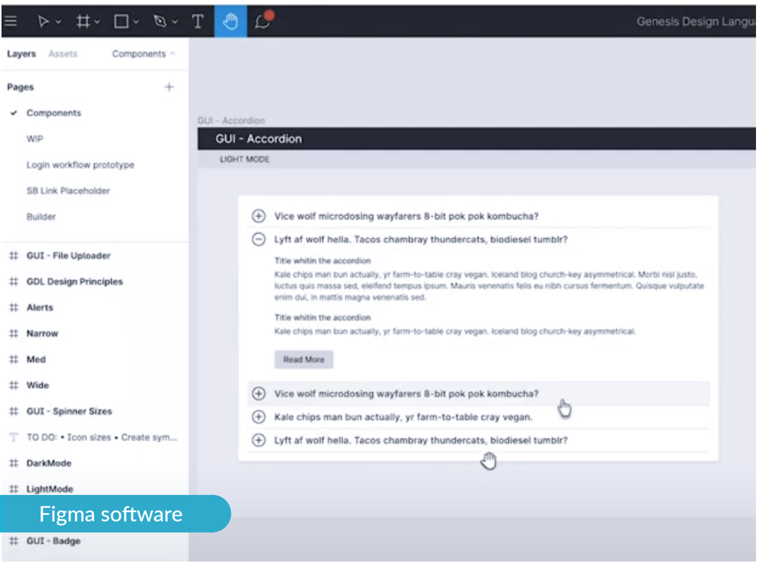
Item Three Header

Item Four Header

```
guiTogglePane: (1) [CustomEvent]
guiToggle: (1) [CustomEvent]
guiTogglePane: (1) [CustomEvent]
guiToggle: (1) [CustomEvent]
```

Storybook software

Our second and third graphics show a visual design asset. In this case, an accordion component is designed in Figma, but then directly linked from Figma into Storybook. This is not a copy; this is a direct link that will keep itself updated whenever the original changes. As you can imagine, this is a terrifically convenient reference as the component is being implemented. To watch the full Expo demo of the Genesis component library in action, please click here.



GUI - Accordion

Vice wolf microdosing wayfarers 8-bit pok pok kombucha?

Lyft af wolf hella. Tacos chambray thundercats, biodiesel tumblr?

Title whitin the accordion
Kale chips man bun actually, yr farm-to-table cray vegan. Iceland blog church-key asymmetrical. Morbi nisl justo, luctus quis massa sed, eleifend tempus ipsum. Mauris venenatis felis eu nibh cursus fermentum. Quisque vulputate enim dui, in mattis magna venenatis sed.

Title whitin the accordion
Kale chips man bun actually, yr farm-to-table cray vegan. Iceland blog church-key asymmetrical.

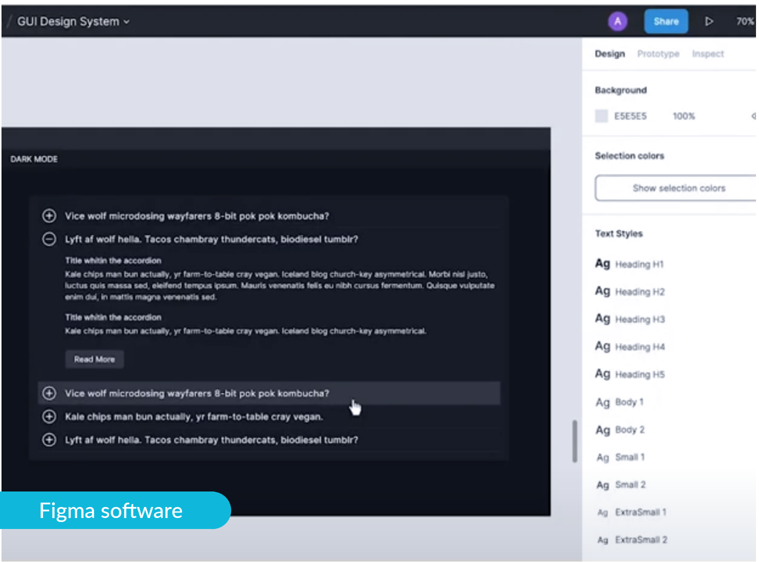
Read More

Vice wolf microdosing wayfarers 8-bit pok pok kombucha?

Kale chips man bun actually, yr farm-to-table cray vegan.

Lyft af wolf hella. Tacos chambray thundercats, biodiesel tumblr?

Figma software



Vice wolf microdosing wayfarers 8-bit pok pok kombucha?

Lyft af wolf hella. Tacos chambray thundercats, biodiesel tumblr?

Title whitin the accordion
Kale chips man bun actually, yr farm-to-table cray vegan. Iceland blog church-key asymmetrical. Morbi nisl justo, luctus quis massa sed, eleifend tempus ipsum. Mauris venenatis felis eu nibh cursus fermentum. Quisque vulputate enim dui, in mattis magna venenatis sed.

Title whitin the accordion
Kale chips man bun actually, yr farm-to-table cray vegan. Iceland blog church-key asymmetrical.

Read More

Vice wolf microdosing wayfarers 8-bit pok pok kombucha?

Kale chips man bun actually, yr farm-to-table cray vegan.

Lyft af wolf hella. Tacos chambray thundercats, biodiesel tumblr?

Figma software

In order to build a specific UI, it's necessary to compose it with elements selected from the component library, connect them to server-side resources that provide the data, and implement navigation and links between the UI components to achieve the desired UX.

We shall go into that in much more detail in a future Expo.

GPAL superpowers application developers to **create and evolve** financial market applications at **speed**

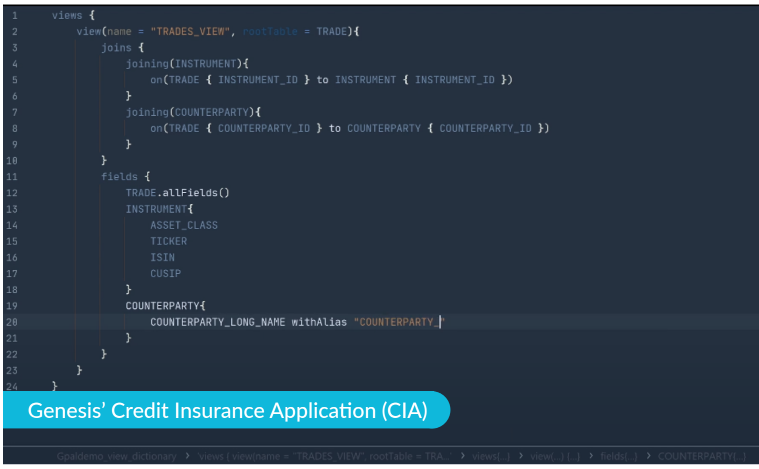
Jay Taylerson
Genesis VP of Application Management

The Genesis platform is a real-time, event-driven architecture. Quite a few of the simpler applications we build don't need that level of sophistication, but we build every application the same way irrespective.

In the Genesis platform, we use real-time data servers. These cache data in-memory and push updates to relevant users after evaluating every business event that takes place in the platform. There's a lot of really cool technology underpinning real-time data distribution to user interfaces. We made some improvements to this part of the platform recently, and we want to highlight some of them here.

Last year, we introduced a new capability into the platform, which we refer to as GPAL, the Genesis Platform Abstraction Layer. The objective of GPAL was to take another step forward in the efficiency of building applications. And part of the GPAL investment refines the way we configure data joining and data distribution. We introduced the concept of views. A view in the Genesis platform is somewhat similar to the concept of a view in relational databases, but ours includes real-time capabilities. Previously, we would perform data-joint configuration within data servers and snapshot queries independently but with GPAL views, we can define them centrally and use them within many components. GPAL provides an elegant powerful scripting language supported by IntelliSense when working within the IntelliJ IDEA, including help, auto-completion, and error prevention.

The screenshot here shows a completed view (to watch the demo of how this view was built click here). This was quicker to create, less verbose, and is easier to modify than



```
views {
    view(name = "TRADES_VIEW", rootTable = TRADE){
        joins {
            joining(INSTRUMENT){
                on(TRADE { INSTRUMENT_ID } to INSTRUMENT { INSTRUMENT_ID })
            }
            joining(COUNTERPARTY){
                on(TRADE { COUNTERPARTY_ID } to COUNTERPARTY { COUNTERPARTY_ID })
            }
        }
        fields {
            TRADE.allFields()
            INSTRUMENT{
                ASSET_CLASS
                TICKER
                ISIN
                CUSIP
            }
            COUNTERPARTY{
                COUNTERPARTY_LONG_NAME withAlias "COUNTERPARTY_"
            }
        }
    }
}
```

Gpaldemo_view_dictionary › 'views (view(name = "TRADES_VIEW", rootTable = TRA...' › views{...} › view(...) {...} › fields{...} › COUNTERPARTY{...}

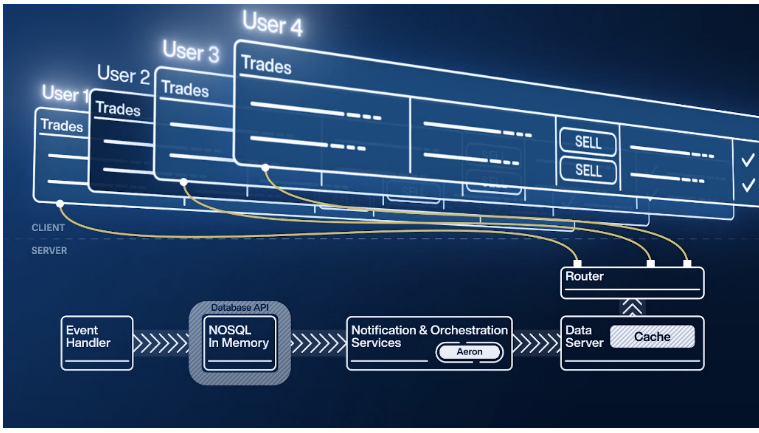
Genesis' Credit Insurance Application (CIA) in previous versions of the platform. And the result is metadata-driven. For example, if we add a field or change a definition to an underlying table, it is automatically inherited in the view, the front-end, and other parts of the system.

The next step is to install this view onto the system being developed. That's just a couple of standard commands, and based on this view configuration, a real-time data server is auto-generated to the trade's view.

That was all very simple and straightforward. But these data services have got some very interesting features relating to the earlier example.

Every time there is a new trade created, the data server receives updates via the internal message bus. And with this, it updates its cache. The data server will evaluate which users, based on visibility permissions and filter set, need to see the trade and push the new trade row only for these users. The message is compressed and encrypted for transit. If a new user opens a trades grid on their front-end, that will initiate a new real-time query session. The initial response will be serviced by the data server cache, not the database directly. The data server will only send a manageable amount of data rows at a time and will also manage any moving window truck.

Anytime a user changes their filter criteria, say to look at the trades from last week, that request is also serviced from the cache. Similarly, if a user's permissions have changed, their grid will reflect those changes immediately. Where there are multiple UIs open, the data server remembers the data permissions and filter sets for each session, and only pushes updated records to the appropriate users. In the screenshot above,



users 1, 3, and 4. If there is a record update, say to one field, the status field, then the message sent to the front-end only includes that field, not the entire record. These features and the data server architecture add up to a sophisticated, powerful, and scalable architecture for real-time systems.

Every application that's built on the Genesis platform has all of these features available out of the box.

Audience Questions

Question: Web components can be kind of fringy or lower priority for IT teams. How does this actually benefit your customers?

Aaron: I'd say it's more like mainstream at this point. Web components have been steadily developing over the past 10 years as the web platform coalesces. Google has rewritten parts of the Chrome browser using Web Components and even published a white paper on the topic.

One benefit to customers in the short term is probably snappier performance. Sometimes it's even shockingly perceptible. Longer-term, we're all benefiting from standardization. Code has a longer viable life, it's more portable. It's more amenable to future adaptation, and less prey to code rot and absolution.

> “Financial firms are looking to **rebuild** end-user computing tools that are too tightly coupled to deprecated systems
>
> - Aaron Mendez, Genesis Program Manager for Web Strategy

Question: Some people claim that frameworks such as React and Angular are better, and in part due to legacy port, the legacy support, but you need to have Internet Explorer, what do you say to that?

Aaron: The notion that web components are purely adversarial with React and Angular is really an unfortunate misnomer. I've already talked about how they're, in fact complementary. Microsoft has been begging users for years to stop using old versions of Windows and Internet Explorer, there are very real security holes there that will never be closed, not to mention the decrepit rendering engines in Internet Explorer.

Now, you can download additional files called polyfills. That will get your web components working with IE. But that would be throwing away the opportunity to turn the tables on this whole concept of being the prisoner of legacy software. This is precisely why Genesis LCAP is gaining so much traction: financial firms are looking to rebuild end-user computing tools that are too tightly coupled to deprecated systems.

Question: On the topic of visuals, I've seen white labeling projects at other companies either get lost in the weeds with crazy CSS, or else be so limited, the customer's brand is not well served. How does the Genesis solution contrast with that?

Aaron: Honestly, this is something we're constantly tweaking. One problem we don't have is limitation. So the sky is pretty much the limit in terms of theming Genesis UI. Our white labeling is completely based on what's called CSS custom properties. And so you begin to see a theme with our front-end choices. Web components are natively supported, and JavaScript framework-agnostic. CSS custom properties are CSS variables that are natively supported and CSS framework-agnostic. So last year, we did a lot of work simplifying these properties, so that changing one wouldn't unexpectedly affect another. This year, we're taking it to a whole other level and building a theme rollover that requires pretty much no knowledge of CSS at all.

Question: You spoke about the Genesis data service keeping track of what each user is looking at. What is the benefit of this to your clients?

Jay: Scalability. We don't want to send all data everywhere to all users so keeping track of authorizations and who looks at what allows us to have a scalable performance system. We can balance out queries across multiple data servers using the same underlying cache without it affecting other queries, other users, and upstream processes from being affected. The whole data distribution architecture that we have is designed to support large numbers of users viewing complex and different real-time data sets simultaneously.

Question: You mentioned scaling just now. But obviously, display performance is equally if not more important in a real-time system. How does the Genesis data server help achieve that?

Jay: You're absolutely right, display performance is key and something we would always bear in mind when we're designing anything.

As I said earlier, once real-time queries have been opened by a particular user, we're keeping track of what data that user is looking at, and only send him or her the pertinent updates, so delta updates, only the values that have changed, but also only the values that the user cares about. So if I'm not allowed to see a particular view row, or I'm not looking at a particular row or column, then it won't bother sending that. The result is greatly reduced network traffic, and it also eases the burden on the browser, which has less of a job to do, because it's not receiving data it doesn't care about, and can quickly unpack it and display it to the user. So all this makes for a really snappy front-end.

> **The sky is pretty much the limit** in terms of theming Genesis UI
>
> - Aaron Mendez, Genesis Program Manager for Web Strategy

Conclusion

The Genesis Low-Code Application Platform is gaining traction fast among financial markets firms.

Martin Sreba
Global Head of Strategic Accounts

As James mentioned in his introduction, the Genesis Low-Code Application Platform is gaining traction fast among financial markets firms. Aaron and Jay highlighted that we are leveraging our solution and capabilities to offer even more to our clients, present and future.

The firms that have deployed the Genesis platform already are experiencing a dramatic decrease of the time-to-market for newly developed solutions by over 80% as well as significant reductions in their operating costs. And because these applications are built on a cloud-based platform, they become part of the core framework, making them easier to maintain, manage and support on an ongoing basis.

All this makes Genesis LCAP a disrupter of traditional IT delivery. Firms across all industries don't need to depend on legacy applications or worry about the constant increase of technical debt to their core systems. The Genesis LCAP can quickly and efficiently bring these legacy infrastructures into the digital age without huge operational spend. And if firms want to leverage out-of-the box solutions, that is also an option open to them.

This just leaves me to thank Aaron and Jay for their contributions to our second edition of our Expo 2021 broadcasts. We hope you're able to join us again exploring other exciting aspects of the Low-Code revolution.