

# SOFTWARE DEVELOPMENT PROJECTS

*Working in an agile environment (TDD, BDD, DDD Pair Programming, etc.)*

Here, I showcase some of my most **impactful contributions** across multiple companies—projects where I've **driven architecture forward**, elevated **developer experience**, and delivered measurable results.

The visuals included are taken from publicly accessible product interfaces and are provided solely for illustration.

## Core Pillars / Philosophy

---

- ◇ Intuitive & Dynamic Design
- ◇ Hide Complexities away from Developer
- ◇ Promote Developer Happiness
- ◇ Accessible Architectural Design
- ◇ Cross Team Collaboration

# Helcim

---

Lead SENIOR SOFTWARE DEVELOPER/ARCHITECT

## AI-ASSISTED CODE GENERATION PIPELINE

### Problem:

- **Legacy Success:** Custom **PHP** web app powered growth for years
- **Scalability Limits:** Became a **bottleneck** for scalability & feature delivery
- **Modernization Path:** Migration to **Laravel + Vue.js**
- **AI Philosophy:** **Ethical AI** → automation with **developer control**
- **Business Goal:** Faster **velocity** and **sustainable growth**

### Solution:

- **Pipeline Design:** **Schema/contract-first** pipeline
- **AI Guidance:** AI assists devs to author **configs** from **Figma flows**, **DB schemas**, **business rules**
- **Codegen Engine:** Configs drive the **codegen CLI** to generate **production-ready** features
- **Determinism:** **Versioned templates** ensure **repeatable output** and **clean diffs**
- **Quality & Safety:** **PR → CI → QA → Deploy** with **contract checks** and **test coverage**
- **Developer Control:** Humans approve configs, PRs, and merges

## Outcomes:

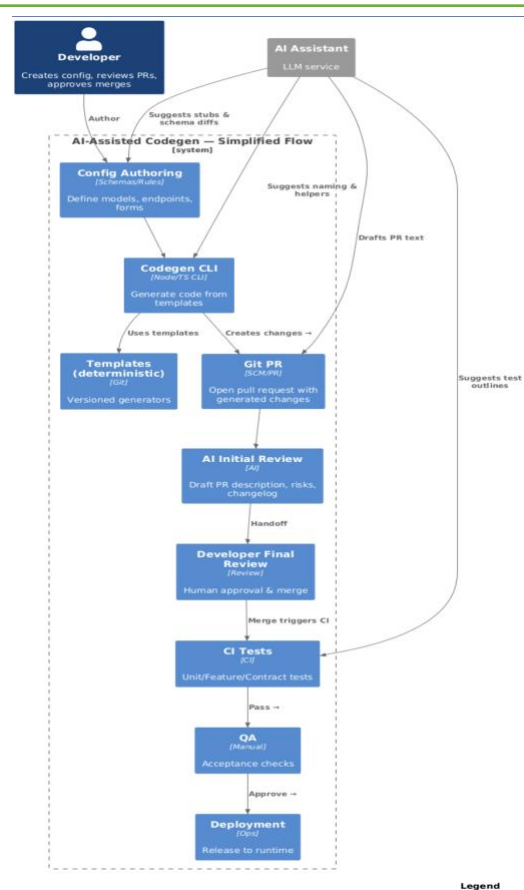
- **Lead time ↓** — features in **hours, not weeks**
- **Consistency ↑** — **schema/contract-first** across services
- **Focus ↑** — devs spend time on **domain logic**, not boilerplate
- **Onboarding ↓** — editable **configs** accelerate new dev ramp
- **Safe changes** — **versioned templates + PR gates**
- **Quality enforced** — **contracts + tests** in CI; **QA sign-off** before deploy
- **Scale advantage** — repeatable automation across teams

## Systems Design:

*Goal: contract-first pipeline turning config into **production-ready features**.*

### C4 Diagram Walkthrough

- **Owner: Dev owns config/PR/merge.**
- **Config: Schemas; AI stubs; human edits.**
- **CLI: Generate via templates; AI hints.**
- **Templates: Versioned, deterministic.**
- **PR/Review: PR → AI pre-review → dev approves.**
- **Release: CI → QA → Deploy (+ migrations).**
- **Principles: Dev-led, contract-first, deterministic, traceable.**



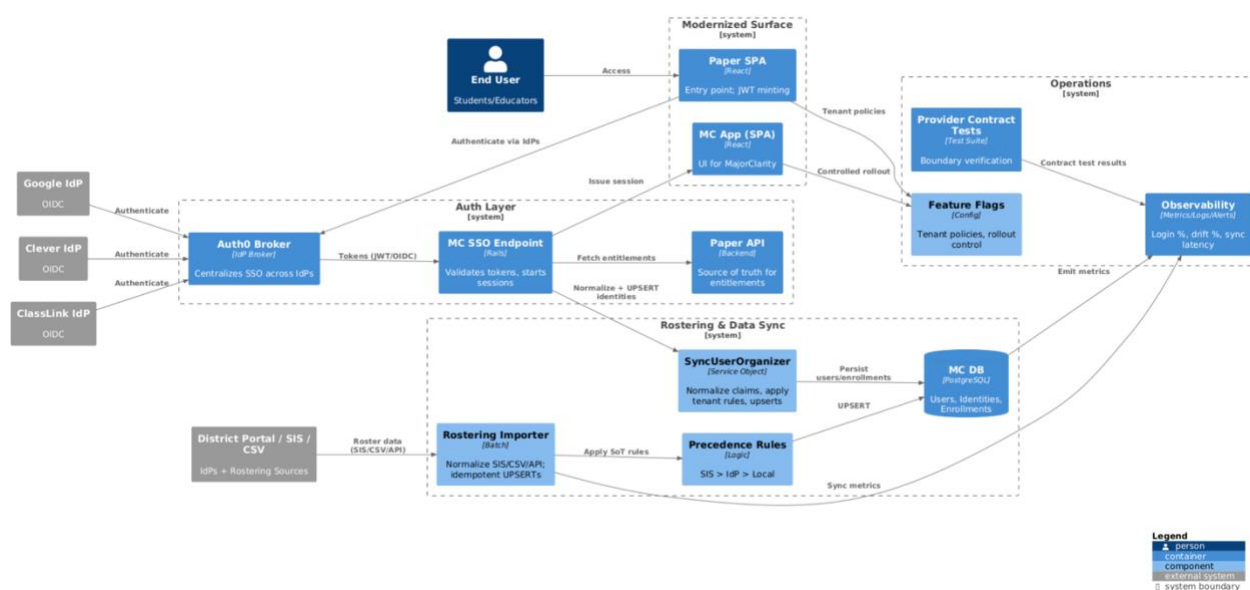
# MAJOR CLARITY BY PAPER

Lead SENIOR SOFTWARE DEVELOPER/ARCHITECT

## EFFICIENT LISTING IMPORTER AUTOMATION

Led the design and implementation of a Unified Authentication & rostering platform following a major acquisition, consolidating two platforms into a single SSO experience.

Built an idempotent rostering pipeline with source-of-truth reconciliation, achieving near-zero data drift, eliminating duplicate identities, and enabling faster district onboarding at scale.



### Situation

- **Post-acquisition** with two **platforms** → fractured UX
- **Identity sprawl** across multiple **IdPs**
- **Roster drift** in **Rails monolith** hot path





## Task

- **Unify identity** and **login** across products
- **Normalize rostering** with reliable source of truth
- **Modernize safely** without breaking legacy monolith
- **Reduce support load** and improve onboarding speed





## Action

- Implemented **brokered SSO** via **Auth0** with **claims normalization**
- Built an **idempotent rostering importer** with **source-of-truth precedence**
- Introduced a **unified data model** for identities/enrollments/sync\_runs
- Added **boundary contracts** (LightService + typed validators) to **fail fast**
- Enabled **feature flags**, **bulk imports**, and **observability metrics**

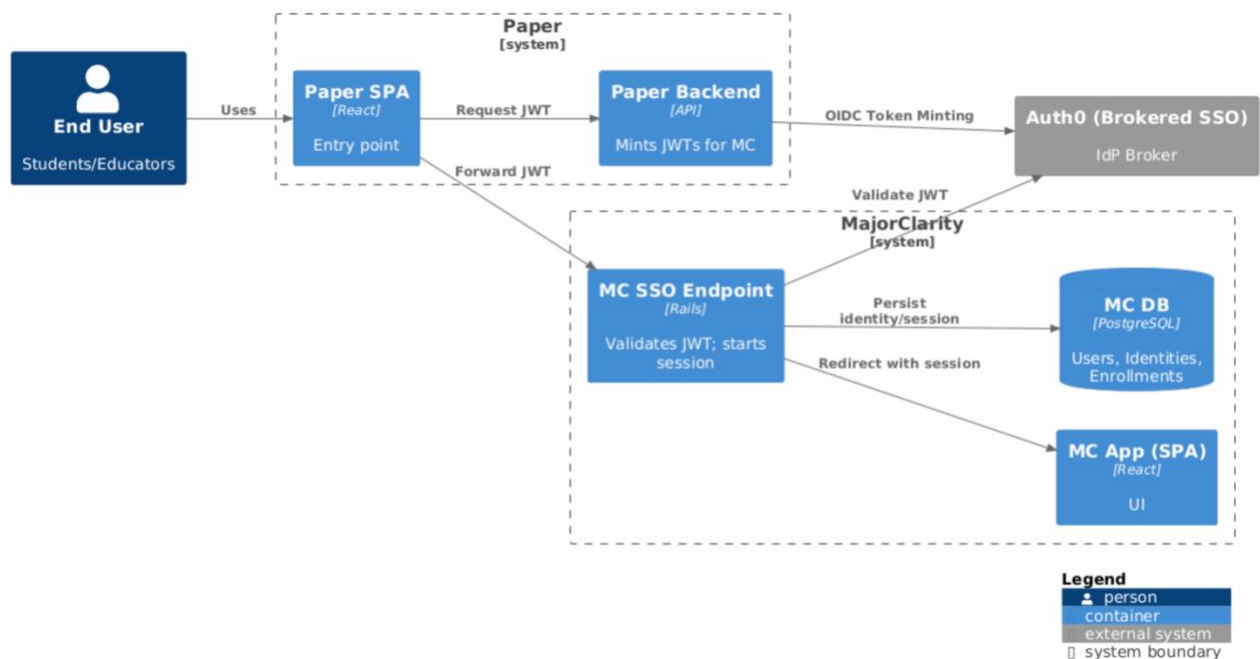
## Result

-  **Seamless login** across Paper + MC; **duplicate accounts eliminated; support load reduced.**
-  **Consistent data:** automated reconciliation → **near-zero drift**; entitlements stable across sessions.
-  **Operational excellence:** clear metrics, alerts, and contract tests → **fewer incidents, faster MTTR.**
-  **Developer focus:** contracts + idempotency remove boilerplate → engineers focus on **domain logic.**

## Highlights

-  **Security:** JWT/IdP validation, **PKCE**, **nonce/state**, **JWKS** verification.
-  **Consistency:** **idempotent UPSERTs**, **source-of-truth precedence**.
-  **Observability:** login success/failure %, sync drift %, upsert counts.
-  **Fallbacks:** break-glass **local accounts**; Paper API preferred for entitlements with **local mirror** fallback.

### 1) Unified SSO Across Paper + MajorClarity



## Situation

- Two **platforms**, two **login paths**, fractured **UX**.
- Business goal: present **one product** with seamless authentication.

## Task

- **Unify identity** and **consolidate login** across both products.

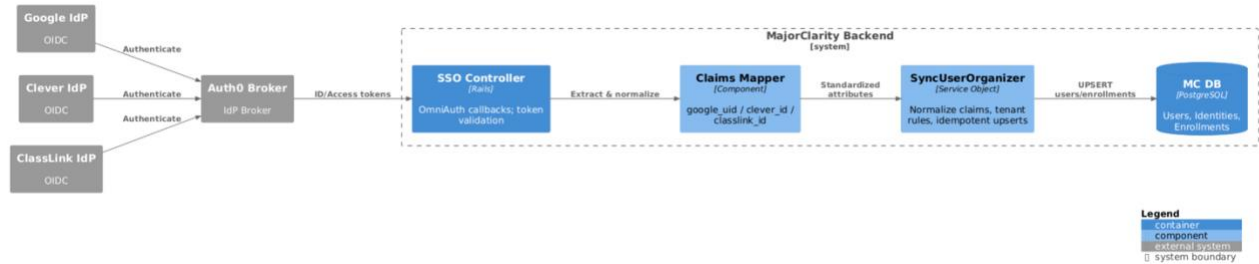
## Action

- Introduced **brokered SSO** via **Auth0** (centralized provider handling).
- Implemented **token validation, claims normalization** (tenant/roles), **silent re-auth, refresh rotation, logout propagation**.
- Established **short-lived, audience-scoped JWTs** for inter-app federation.

## Result

- **One SSO experience** across Paper + MajorClarity.
  - **Duplicate accounts eliminated; support load reduced.**
-

## 2) Centralized Login & Claims Normalization



### Situation

- Multiple **IdPs**: Google, Clever, ClassLink + local credentials; no **unified policy**.

### Task

- **Consolidate identity** and standardize **claims/entitlements**.

### Action

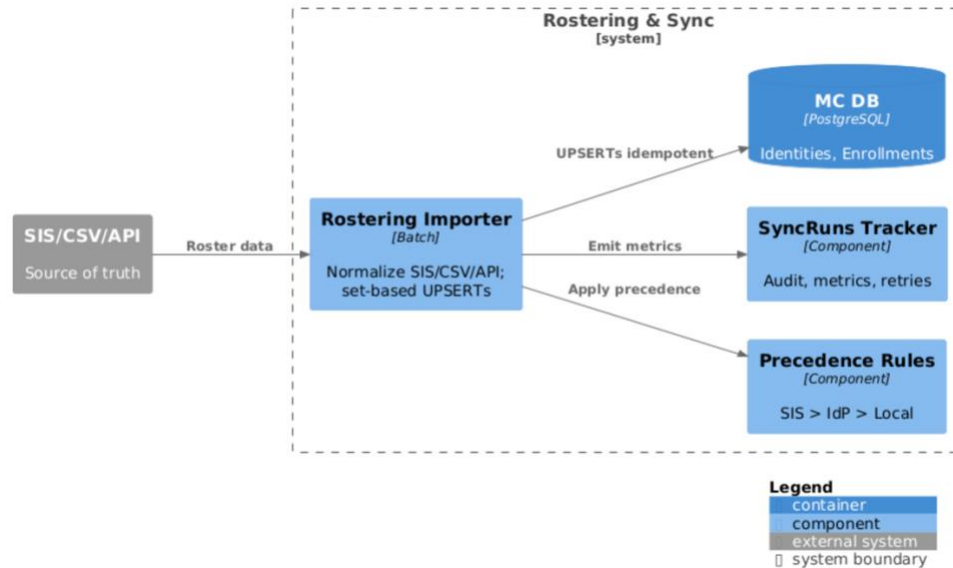
- Centralized login via **Auth0 broker** with **OmniAuth callbacks**.
- **SyncUserOrganizer** normalizes claims (google\_uid / clever\_id / classlink\_id) and applies **tenant rules**.
- Validated tokens (issuer, audience, expiry, **nonce/state**, **JWKS** signature).

### Result

- **Unified identity model** across providers; **cleaner entitlements**.
- **Reduced identity mismatch risk**.



### 3) Idempotent Importer & Source-of-Truth Reconciliation



#### Situation

- **SIS data drift vs app; duplicate users/enrollments.**

#### Task

- Eliminate **data drift** and maintain **roster consistency** across Paper + MC.

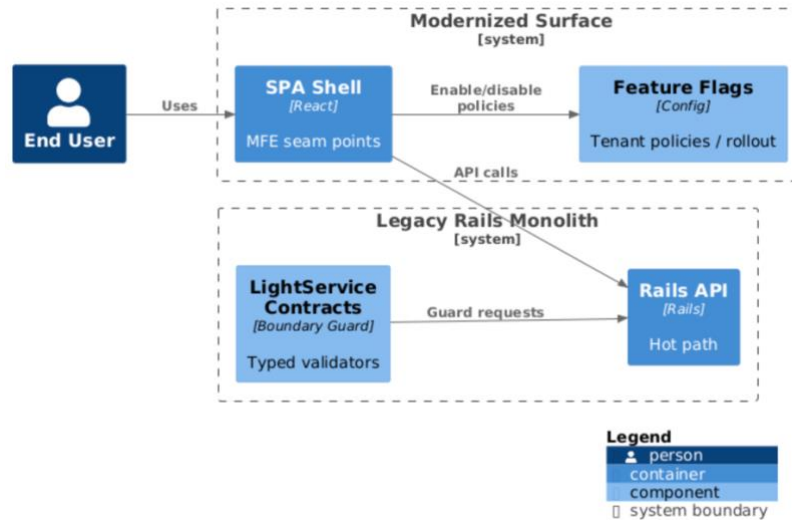
#### Action

- Built **idempotent rostering importer** for **SIS/CSV/API** inputs.
- Applied **source-of-truth precedence** (**SIS > IdP > Local**).
- Unified data model: **identities, enrollments, sync\_runs**.

#### Result

- **Automated reconciliation** → **near-zero drift; stable entitlements**.
- **Fewer roster-related support tickets**.

#### 4) Legacy Constraints (Safe Modernization via SPA/MFE Seam Points)



#### Situation

- **Rails monolith** on the hot path; **uneven tenancy rules**; risky to change.

#### Task

- **Modernize safely** without destabilizing the monolith.

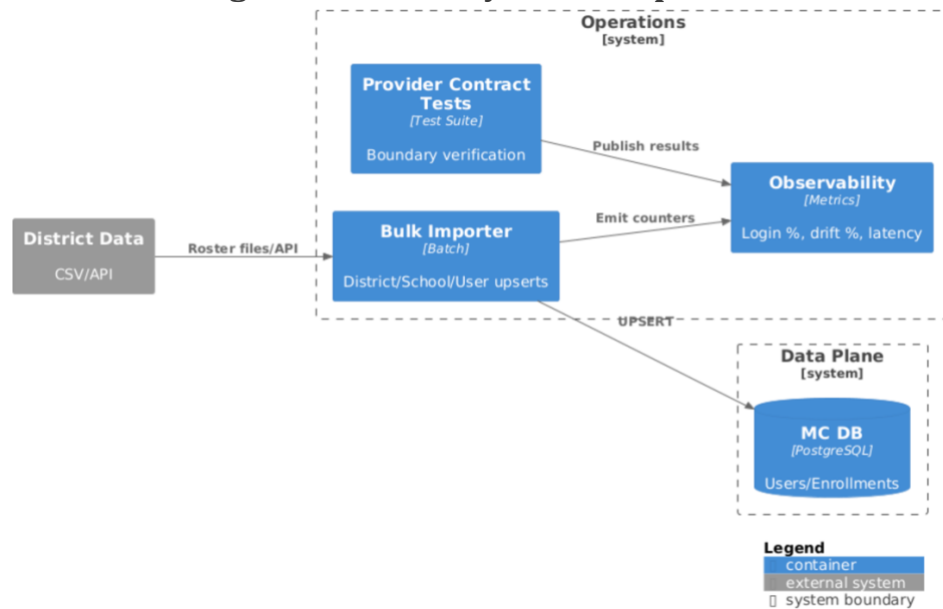
#### Action

- Added **SPA/MFE seam points** to de-risk UI integration.
- Introduced **LightService boundary contracts + typed validators**.
- Enabled **feature flags** and **versioned configs**.

#### Result

- **Modernized surface** without big-bang rewrite; **safe rollback** paths.
- **Resiliency** preserved through controlled fallbacks.

## 5) Scalable Onboarding & Observability-Driven Ops



### Situation

- High **support tickets** (login/entitlements); **slow district onboarding**.

### Task

- Reduce **ops burden** and accelerate **onboarding**.

### Action

- **Bulk imports** at scale (district/school/user mappers; **set-based UPSERTs**).
- **Observability**: drift rate, login success %, sync latency, batch insert/update counts.
- **Provider contract tests** at boundaries; actionable error reporting.

### Result

- **Onboarding time reduced; fewer incidents; faster recovery.**

# SPACELIST

---

SENIOR SOFTWARE DEVELOPER/ARCHITECT

## EFFICIENT LISTING IMPORTER AUTOMATION

### Situation

- **Spacelist** struggled with **inefficiencies** in managing **large volumes** of property listings.
- **Onboarding new clients** took **weeks**, creating bottlenecks.
- **High processing costs** led to:
  - Frequent **file preview errors**
  - Long **processing delays**
  - Increased **manual interventions** driving up **operational costs**

### Task

- **Streamline** the listing import and onboarding process.
- **Reduce costs** by eliminating redundant operations.
- **Improve reliability** of asset handling and file previews.

### Action

- Built an **automated listing importer** with **declarative per-client configuration**.
- Implemented **field mapping** to simplify onboarding workflows.

- Added **duplicate detection** using **E-Tags** and **filenames** to skip unnecessary asset re-uploads.
- Resolved **file preview errors** by **bulk updating S3 assets** with correct metadata.

## Result

- **Onboarding time reduced: weeks → days**
- **Sitemap generation** cut from **6 hours → 50 minutes**
- **File preview errors resolved** through metadata fixes
- **Costs lowered** by eliminating redundant image uploads

# GEOFORCE

---

SENIOR SOFTWARE DEVELOPER/ARCHITECT

## OKTA INTEGRATION AUTOMATION

**Implemented an enterprise-wide Okta SSO integration** to replace a **fragmented user management system**, reducing **security risks**, eliminating **multiple logins**, and improving the **user experience**.

Delivered a **custom-branded login service** and **automated provisioning** with **Terraform-based Infrastructure-as-Code**, enabling **seamless user migration**, consistent **policy enforcement**, and scalable **identity and access management** across all applications.

### Situation

- **Geoforce** operated with a **fragmented user management system** requiring **multiple logins** across applications.
- This caused **security vulnerabilities**, a **poor user experience**, and **high operational costs** from password resets and manual access management.






### Task

- **Unify authentication** into a single **SSO solution**.
- Ensure **seamless migration** of existing users without disruption.
- Improve **security**, **scalability**, and **operational efficiency** through automation.

## Action

- Introduced **Okta** as the **central SSO solution** across all applications.
- Built a **custom-branded Okta login service** to broker authentication consistently.
- Migrated **pre-existing users** into Okta with minimal friction.
- Adopted **Infrastructure-as-Code** with **Okta Terraform plugins** to automate provisioning of **apps, user profiles, groups, and security policies**.

## Result

-  **Unified authentication** across applications → eliminated multiple logins and strengthened **security**.
-  Delivered a **custom-branded login experience** → improved **user experience**.
-  **Seamless migration** of existing users → minimal disruption.
-  **Automated provisioning** with Terraform → reduced **manual effort** and increased **operational efficiency**.
-  Achieved greater **scalability** and consistent **policy enforcement** → improved security posture.

# RELAY PLATFORM

---

SENIOR SOFTWARE LEAD DEVELOPER/ARCHITECT

## CYBER INSURANCE QUOTING PLATFORM

I led the design and implementation of a **standalone multi-carrier Quoting Platform** for **Relay**, created to simplify quote generation across diverse **broker carriers**. The platform addressed **fragmented workflows**, supported highly **dynamic insurance products**, and became the **most successful product in company history**.

### Situation

- **Relay Platform** needed to expand into **Cyber Insurance** with a **multi-carrier quoting product**.
- Each carrier exposed **unique API workflows** covering quotes, binds, and renewals.
- Existing systems struggled with **fragmentation, manual processes**, and **slow onboarding** of carriers.
- The challenge was to design a solution that could **scale across carriers**, provide a **consistent broker experience**, and **support complex underwriting logic**.









## Task

- Build a **unified quoting platform** that could:
  - Handle **multiple external carrier APIs** with different workflows.
  - Provide **reusable workflows** across both **manual and automated quotes**.
  - Deliver a **responsive broker experience** with **real-time updates**.
  - Support **scalable dynamic forms** for complex insurance scenarios.

## Action

- Designed **workflow-oriented async services** to integrate carrier APIs, with quotes normalized into a **standard format**.
- Implemented **event-driven orchestration**: quote state changes (e.g., *pending* → *generated*) triggered **notifications, webhooks, and async jobs**.
- Built **stateless, JSON-Schema-driven components** (RJSF/AntD) to handle **dynamic forms** for underwriting, endorsements, and conditional logic.
- Introduced an **API Edge (GraphQL/REST)** behind an **API Gateway**, exposing consistent APIs to the SPA.
- Developed **per-carrier adapters** to handle custom workflows, while maintaining **idempotent persistence** in **Postgres**.
- Delivered a **real-time user experience** via **WebSockets/Push** for state changes and auto-refresh.
- Automated **email notifications** and **outbound integrations** to ensure seamless broker/carrier communications.

## Result

-  Launched the **most successful product** in Relay's history.
  -  **Unified quoting workflows** across carriers and manual inputs.
  -  Delivered a **scalable, dynamic UI** supporting complex cyber insurance scenarios.
  -  Improved **user experience** with **real-time state updates** and reduced broker friction.
  -  Accelerated **carrier onboarding** with a flexible, adapter-based integration approach.
  -  Enabled **future growth** with a system designed to evolve alongside the business.
- 

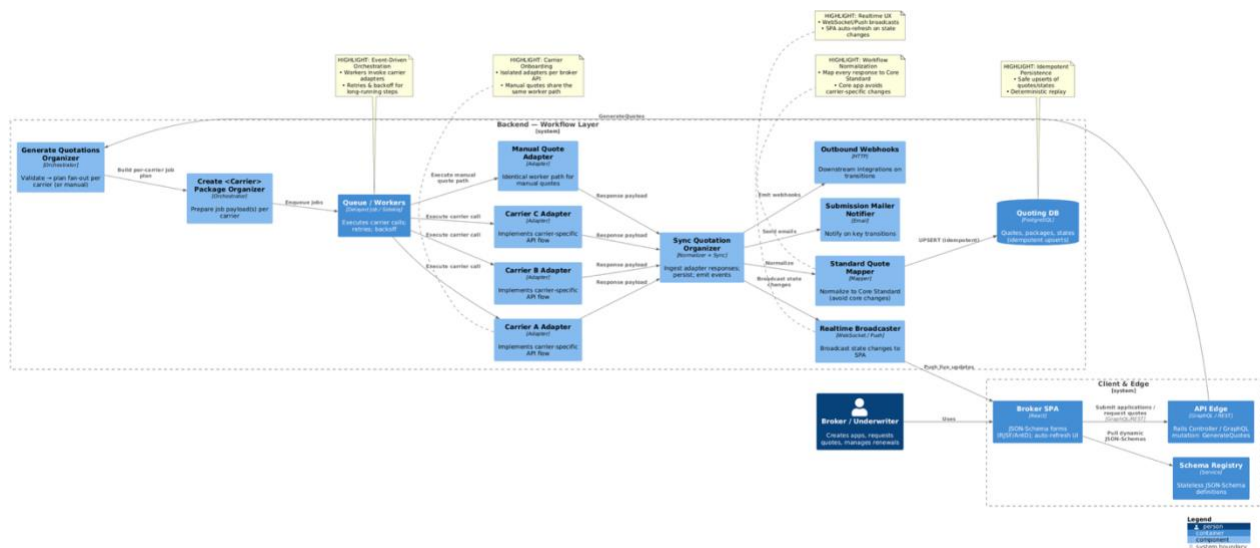
**Problem:** Introduce a standalone multiple-carrier Quoting Platform designed to simplify the generation of quotes by tailoring them to customer requirements, terms, and endorsements.

This project stands as one of the most ambitious endeavours I've had the privilege to be a part of. I take great pride in it, primarily because of the outstanding teamwork, collaboration, and mentorship that have been integral to every facet of the project's success.

A particularly distinctive feature of this project, which played a crucial role in shaping its architectural design, was the requirement to onboard multiple broker carriers. These carriers each presented unique API workflows, encompassing the entire process from generating initial Quotes to completing renewals.

## Workflow-Oriented Async Services

- 🔗 **Carrier Onboarding:** Each broker's **API workflow** is integrated via **isolated adapters**, while **manual quotes** follow the same workflow patterns.
- 🔄 **Workflow Normalization:** All responses are transformed into a **core standard format**, ensuring the **core app never changes** regardless of carrier differences.
- ⚡ **Event-Driven Orchestration:** Workflow transitions (e.g., quote generated, bound, or renewed) trigger **email notifications**, **webhook broadcasts**, and **real-time updates**.
- 🔗 **Scalable Architecture:** The workflow layer isolates carrier complexity, enabling **parallel integrations**, faster onboarding, and a **consistent broker experience**.
- 👁️ **Realtime UX:** Users see **live state changes** through **event broadcasts** (WebSockets/Push), keeping the SPA automatically refreshed.



## Dynamic Components

- ✍ **Schema-Driven Forms:** All broker and carrier inputs (proposal, underwriting, endorsements, conditionals) are rendered from **JSON Schema definitions**, ensuring consistency and flexibility.
- 🔄 **Stateless Configurations:** UI behavior is driven by **external schema configs**, not hard-coded logic—making updates safe and scalable.
- 🧩 **Composable Components:** Leveraged **RJSF + AntD widgets** as reusable building blocks, adaptable to any insurance product variation.
- ⚡ **Scalable Growth:** New coverages and product permutations are introduced by updating schemas, not rewriting code—allowing the platform to **scale with business needs**.
- 🚀 **Proven Success:** This approach powered the **most successful product in Relay's history**, delivering both speed to market and maintainability.

[illegible][illegible]

# TRUFLA TECHNOLOGY

---

SENIOR SOFTWARE DEVELOPER/ARCHITECT

## Progressive React Adoption inside Ember.js

The screenshot shows a web application interface. On the left, there's a sidebar with a user profile for 'Des' (Desmond O'Leary) and various fields like Name, Company, Email, Phone, Website, Source, Address, and Tags. The main content area is titled 'React.js' and contains a form with fields for Insured Name, Website, Country, State, City, and Zip Code. The form is styled with a light blue background and has a 'Back to quotes' link at the top.

### Situation

- The team was tasked with delivering a large-scale project after initial planning.
- The frontend relied on an **outdated Ember.js (2.13.1)** application with **no test coverage**.
- Upgrading Ember posed significant **risks** due to breaking changes, limited documentation, and poor ecosystem support.






### Task

- Modernize the frontend without disrupting **existing functionality**.
- Enable delivery of new features with **test coverage** and a **scalable framework**.
- De-risk the project to ensure successful delivery for a major client.

## Action

- Conducted analysis of **upgrade paths** (Ember 3.3.x vs 3.4.x) and evaluated alternatives.
- Introduced **React** incrementally inside the existing application, leveraging its **flexibility to embed within HTML**.
- Built **new functionality in React** while preserving legacy Ember components.
- Implemented a **testing framework** to bring coverage from zero to hundreds of automated tests.
- Overcame **build tool constraints** (Ember's Broccoli) and **compilation issues** by isolating dependencies.

## Result

-  Delivered a modernized UI with an **updated look and feel**, aligned with client expectations.
-  Successfully released **new features** without breaking existing functionality.
-  Improved quality with **hundreds of automated tests** introduced.
-  Helped secure a **major client contract** previously at risk.
-  Enhanced the team's ability to **attract talent** with a modernized tech stack.

# JSONSchema Driven DYNAMIC Components

## Situation

- The product required **rapidly evolving forms** to keep pace with future business needs.
- Manually coding forms was **time-consuming**, hard to scale, and increased **maintenance costs**.





## Task

- Implement a **scalable, maintainable solution** for building forms dynamically.
- Ensure that form creation was **schema-driven**, allowing teams to focus on **end-to-end testing** rather than repetitive UI work.

## Action

- Adopted **RJSF (React JSON Schema Form)** layered on top of custom **Ant Design components**.
- Defined forms by simply declaring a **JSON Schema**, removing the need for custom form logic.
- Replaced many RJSF components with **Ant Design counterparts** to reduce long-term maintenance.
- Leveraged **RJSF theming** to streamline Ant Design integration.
- Addressed challenges around **re-renders and conditional fields** through collaborative debugging and optimization.

## Result

-  Enabled creation of forms **entirely from JSON Schemas**, cutting development overhead.
-  Reduced repetitive UI testing, allowing focus on **end-to-end Cypress test coverage**.
-  Improved scalability and flexibility for future product requirements.
-  Delivered a consistent and modern UX through **Ant Design** integration.



# HRAND/MEDI-DIRECT INC.

---

SENIOR SOFTWARE DEVELOPER/ARCHITECT

## B.D.D. UI & API Test Automation Framework

Architected and delivered a scalable framework that combines **developer experience** with **ease of use**. Built on **intuitive, well-documented, and fully tested interfaces** that hide complexity while ensuring consistency. This foundation enables **rapid refactoring, robust API/UI test coverage**, and smooth **onboarding of new developers**—allowing the team to scale quickly by defining functionality through simple [feature files](#).

### Situation

- The project was committed to building with a **Behavior-Driven Development (BDD)** approach.
- Initial **API and UI tests** were highly **boilerplate-heavy**, fragmented, and lacked ongoing **maintenance**.
- This created a **risk of inefficiency** and duplication, slowing delivery and limiting test value for stakeholders.






### Task

- Create a **unified automation framework** that would:
  - Reduce **boilerplate** across API and UI tests.
  - Promote **transparency and reusability** of test scenarios for developers and business stakeholders.
  - Scale efficiently as new **resources** and **UI pages** were added.

## Action

- Designed and implemented a **generic step library** to maximize **step reusability** across scenarios.
- Converted **background steps into fixture data** for easy referencing across tests.
- Provided **detailed syntax guidance** so developers and stakeholders could write **clean, understandable scenarios**.
- Built a **request builder** that automatically generated and verified API requests, ensuring data was correctly **persisted and audited**.
- Extended the same request model to **control the UI**, synchronizing input updates with **automated validations**.
- Introduced **helper methods** to detect references (e.g., updating an existing field) and adjust behavior dynamically.

## Result

-  Delivered a **clear, concise interface** that hid complexity, letting developers focus on business logic.
-  Created an **extremely pluggable framework**, adaptive to new resources and UI pages with minimal effort.
-  Unified API and UI testing, eliminating duplication and streamlining maintenance.
-  Scaled test coverage significantly while enabling faster **end-to-end Cypress testing**.
-  Widely **adopted and well received**, transforming BDD from a process overhead into a **core productivity driver**.

# HUBSTAFF INC.



---

SENIOR SOFTWARE ARCHITECT

## Situation

- Engaged as part of an **intensive trial period**, working directly with the **CTO** and **Lead Architect**.
- Responsible for translating **high-level requirements** into **functional and technical specifications** for multiple development teams.
- Projects included **Time Off management, Timesheets, and SSO security enhancements**.

## Overall Result

-  Successfully delivered all three projects (**Time Off, Timesheets, SSO**) **within the quarter**.
-  Earned approval at the end of the trial period and contributed to ongoing team success.

# TIME OFF MANAGEMENT

## Task

- Design a system to manage **public holidays** (paid/unpaid) and **time-off policies** (unpaid, hourly, and yearly accruals).

## Action

- Conducted in-depth **research** and **competitive analysis** of similar products.
- Produced **mockups** and **workflow diagrams** through multiple iterations with stakeholders.
- Defined clear **functional and technical requirements** for implementation.

## Result

- ☒ Delivered a **comprehensive design** enabling flexible **holiday and time-off policy management**.
- ☒ Improved clarity and alignment between business stakeholders and developers.

# TIMESHEETS



## Task

- Implement a **timesheet system** based on activity recorded from **Hubstaff client applications**, integrated with invoicing, billing, and scheduling.

## Action

- Designed detailed **workflows and specifications** to capture activity, idle time, and pay-schedule submissions.
- Collaborated across teams to ensure compatibility with **invoicing and scheduling systems**.

## Result

-  Delivered a **scalable timesheet feature** that provided accurate visibility into **employee activity and idle time**.
-  Successfully integrated with **billing and scheduling systems**, ensuring smooth payroll operations.

# SSO Security Enhancements



## Task

- Strengthen the existing **SSO system** to address **session invalidation gaps**.

## Action

- Implemented **front-channel logout** per the **OpenID specification**, ensuring all sessions in the same browser were invalidated.
- Developed **back-channel logout**, enabling the identity provider to **invalidate all sessions** across browsers and relying parties.

## Result

-  Closed critical **security gaps** in the SSO system.
-  Ensured **consistent session invalidation** across browsers and applications.

# CISCO SYSTEMS



---

SENIOR SOFTWARE / PRODUCT DEVELOPER

## Situation

- Responsible for delivering **high-priority features** and **modernization projects** across a complex Rails-based platform.
- Features needed to be **feature-flagged** for safe **continuous integration** and **quick rollback** due to the sensitive nature of the system.

## Overall Result

-  Successfully delivered **critical features** and **platform upgrades** while maintaining **zero production incidents**.
-  Built a foundation for **scalable growth**, improved **security**, and **enhanced user experience** across the platform.

# MODULARIZED AUTHORIZATION



## Task

- Extract a legacy **login process** into a **reusable service** and strengthen platform-wide identity management.

## Action

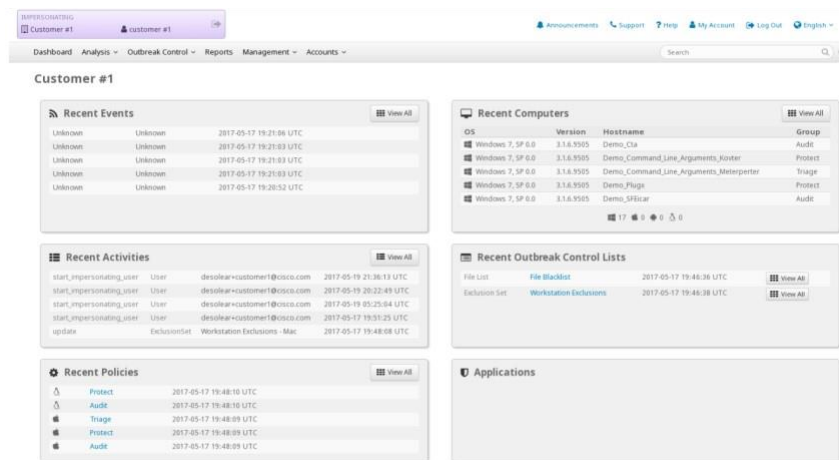
- Collaborated with senior developers to refactor login into a **dedicated engine**, now serving as a **microservice**.
- Introduced **SSO support** and implemented a proprietary **Identity Provider**.

## Result

-  Delivered a **modular, reusable auth service** across multiple applications.
-  Enhanced **security** and simplified **user management**.



# PARTNER ACCOUNTS





## Task

- Design and deliver a **new feature** allowing partner companies to manage their clients and access reporting.

## Action

- Introduced **Single Table Inheritance (STI)** for clean separation.
- Designed **daily reporting** across **MySQL, Cassandra, and MongoDB**.
- Built **aggregate partner reports** and a **Partner Dashboard**.
- Added **impersonation features** to enable account management on behalf of customers.

## Result

-  Created a **high-priority revenue-generating feature**, one of the year's most impactful projects.
-  Delivered **accurate, multi-source reporting** and **improved partner visibility**.

# GLOBAL ANNOUNCEMENTS

## AMP for Endpoints upgraded to version 1.1.1502678831 ✕

For the list of new features and/or fixes, please see the [release notes](#) 



### Task

- Implement a system for **platform-wide announcements** (general, maintenance, upgrades).

### Action

- Built backend on **Starburst Gem Foundation**, with tracking of **read/unread states**.
- Developed frontend with **announcement types**, preview-as-you-type, and rich UI.

### Result

-  Introduced a **global communication channel** improving visibility of critical updates.
-  Delivered a **user-friendly experience** for announcements across the app.

# BUSINESS CREATION REVAMP

The screenshot shows a web form for creating a business. It is organized into several sections:

- Business Name:** A text input field with a red border and a placeholder "can't be blank".
- USER:** Fields for First Name, Last Name, and Login Email, all with red borders and "can't be blank" messages. A Notification Email field is below, with a hint "Leave blank if same as Login Email".
- LICENSES:** Fields for Connectors (value 50) and Term (months) (value 2).
- DEFAULT USER PREFERENCES:** A dropdown for Locale with the text "Select Language".
- Business Relationship:** A dropdown menu currently showing "Customer".
- Payment State:** A dropdown menu currently showing "Evaluation".
- Incident Response:** Two radio buttons, "Incident Response" and "On Prem", both currently unselected.
- Device ID:** A text input field with a hint "e.g. AMPPCA001".
- Expiration date:** A date input field.
- CONFIRMATION EMAILS:** A "Send email to" section with radio buttons for "Support user" (selected) and "Customer". A "Sales emails" text input field contains "sales@cisico.com...".

A green "Create" button is located at the bottom right of the form.



## Task

- Revamp how **user accounts and businesses** were provisioned, addressing DB locks and scalability.

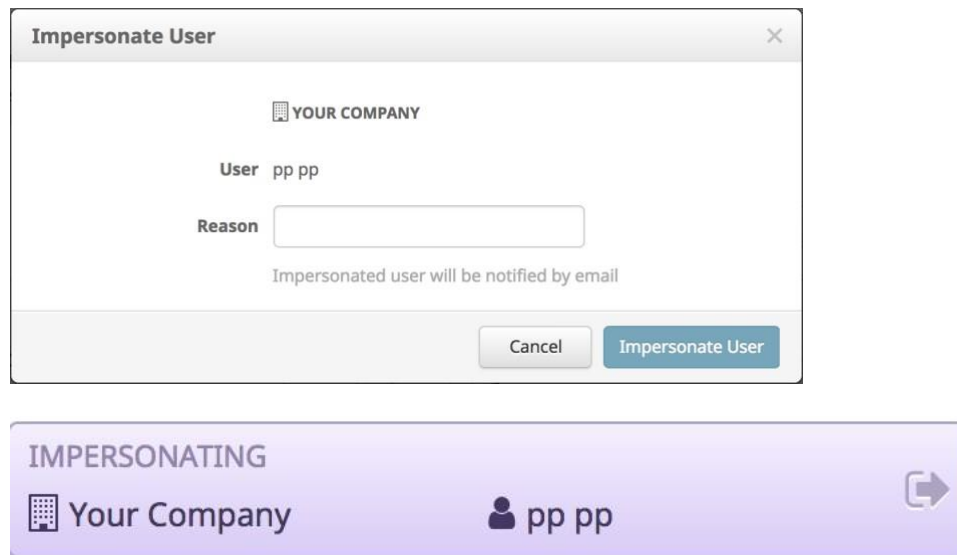
## Action

- Diagnosed **deadlocks** in provisioning code; migrated from **attribute tables** to consolidated columns.
- Removed redundant defaults, moved provisioning to **Delayed Job**, and incrementally refactored.
- Improved frontend with **unified error handling, consistent validation**, and Gmail-inspired UX.

## Result

-  Eliminated **DB locking issues** and improved scalability.
-  Delivered a **modernized user experience** and simplified account setup.

# IMPERSONATION



The image shows a user interface for impersonation. At the top is a dialog box titled "Impersonate User" with a close button (X). Inside the dialog, there is a company icon and the text "YOUR COMPANY". Below this, the "User" is listed as "pp pp". There is a "Reason" label followed by an empty text input field. A note states "Impersonated user will be notified by email". At the bottom of the dialog are two buttons: "Cancel" and "Impersonate User". Below the dialog is a purple horizontal bar. The bar contains the word "IMPERSONATING" in all caps. To the left is a company icon and the text "Your Company". To the right is a user icon and the text "pp pp". On the far right of the bar is a right-pointing arrow icon.



## Task

- Enable support staff to **impersonate customer accounts** safely and transparently.

## Action

- Implemented impersonation with the **Pretender gem**, adding audit logs and customer-visible activity tracking.
- Required **explicit customer opt-in agreements**.
- Built a **support-only widget** with reason capture and visual indicators.

## Result

-  Strengthened **support workflows** while ensuring **security and transparency**.
-  Increased customer trust with **full visibility of impersonation actions**.

# RAILS 4 & RUBY UPGRADE



## Task

- Upgrade **Rails** and **Ruby** versions without customer disruption.

## Action

- Used **feature flags** to test dual Ruby versions before global rollout.
- Incrementally upgraded **Rails 4**, replacing deprecated features, updating gems, and fixing specs.
- Increased **test coverage** to ensure parity pre- and post-upgrade.

## Result

-  Seamlessly upgraded core stack with **no customer impact**.
-  Improved maintainability and long-term platform stability.

# BOOTSTRAP 3 & SITE WIDE LOCALIZATION

## Task

- Upgrade UI framework to **Bootstrap 3** and deliver **multi-language support** across the platform.

## Action

- Led **Bootstrap migration** from 2.x → 3.x, coordinating across multiple teams.
- Implemented **site-wide localization**, including DB-stored constants.
- Created **FAQ wikis** for Bootstrap migration and localization tips to onboard contributors and translators quickly.
- Feature-flagged rollout to reduce risk.

## Result

- 🌐 Delivered **multi-language support**, opening new markets.
- 😊 Introduced a **modernized UI** with no production issues.
- 🤝 Enabled smooth **team collaboration and onboarding** during a major platform overhaul.