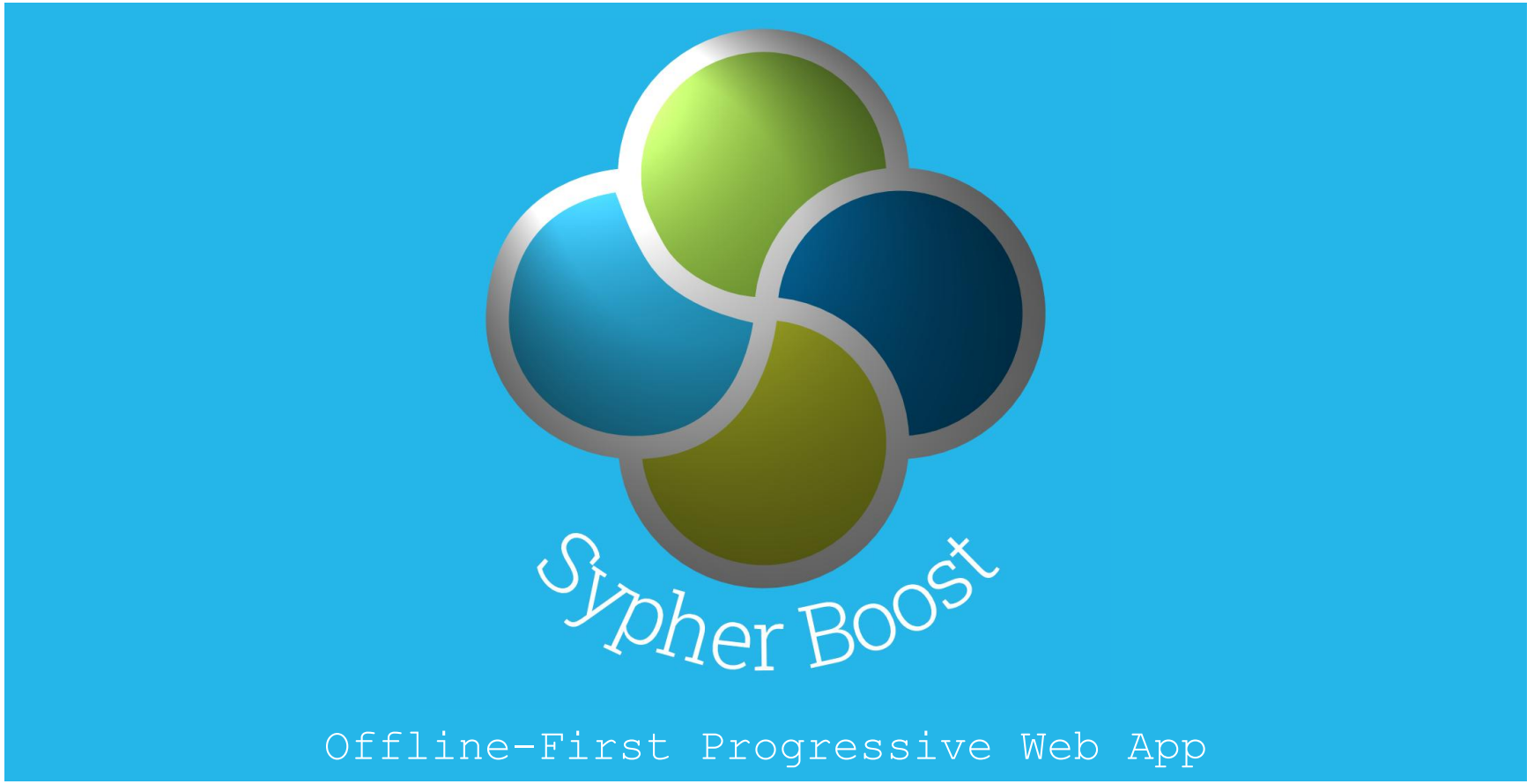
Proof of Concept

SYPHER BOOST | OFFLINE PROGRESSIVE WEB APP

FEASIBILITY AND TECHNICAL DESIGN



Introduction

This proof of concept examines methods for presenting data visualizations in situations where network connectivity and latency would degrade the user experience. The approach explored here is the packaging of Sypher into a Progressive Web App (PWA). Issues outside of the realm of PWA are beyond the scope of this POC.

Progressive Web App Implementation

In order for Sypher to become an *installable* PWA it *must* meet these *minimal* requirements:

1. Service Worker - JavaScript that manages network / cache communication
2. Manifest - JSON that describes the scope and assets of the app

Service Worker

A service worker *must* be served from the root of its scope. This is a challenge in a Salesforce app because JavaScript typically exists in the static resources sub-folder.

The simple solution was to create a Visualforce page that acted as the Server Worker. This page queries the body of the actual JavaScript file stored in the static resources folder.

Service Worker / Visualforce Page

```
public with sharing class NXT_sypher_pwa_sw {
   public String getContent() {
     StaticResource worker = [
       SELECT Body FROM StaticResource WHERE Name = 'NXT_sypher_pwa_sw' LIMIT 1
     ];
     return worker.Body.toString();
   }
}
```

Manifest

The manifest *must* contain scoped references to the static assets required to install the PWA on the home screen. Again, this is challenging in a Salesforce app because the location of these assets are dynamic and subject to change within each environment / deployment. Meaning that the absolute href would contain a transient static resource timestamp. The solution was to read dynamically generated links in the DOM then build the manifest as a data URI reference.

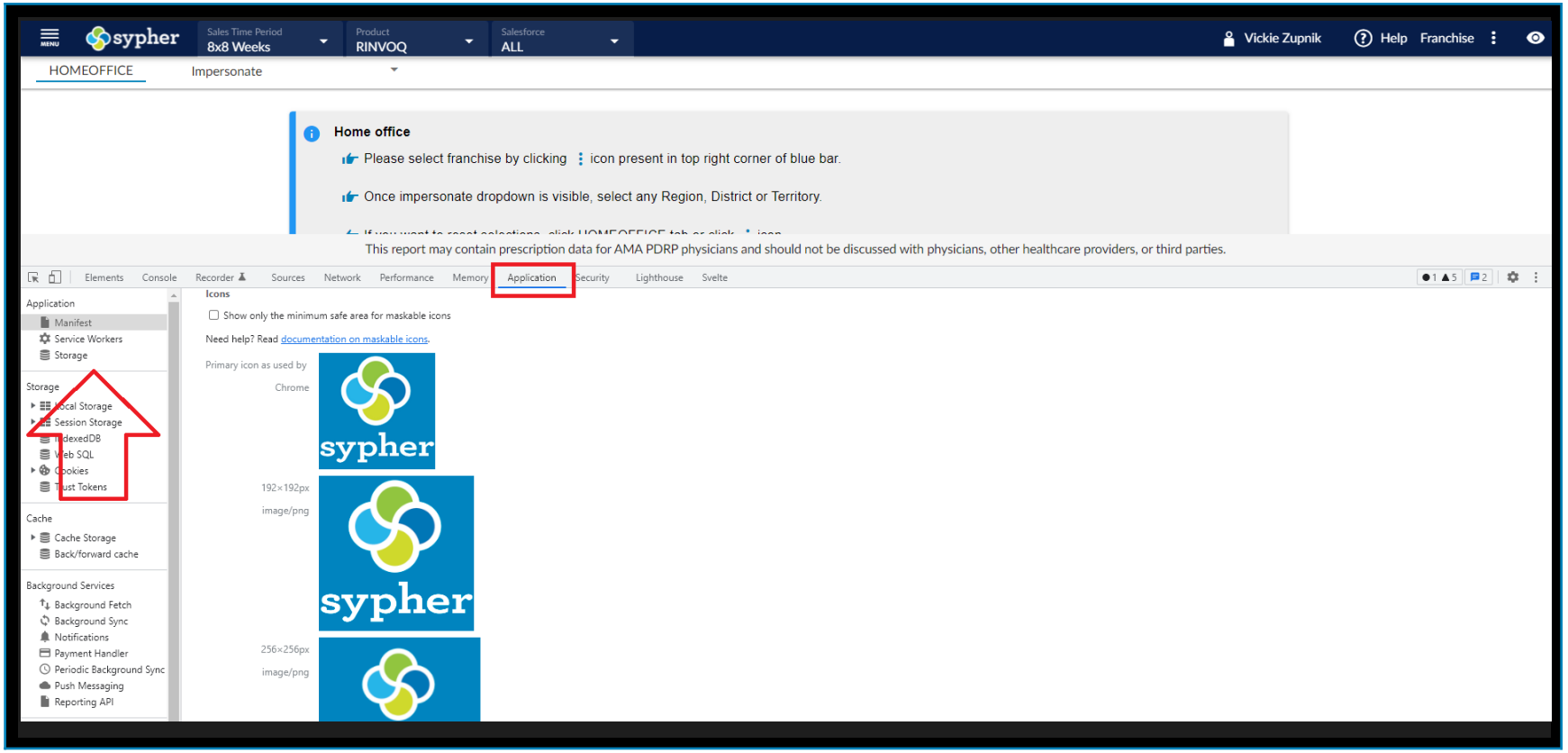
Manifest JSON - SyPWA static class | Apple Touch Icon logic

```
    static injectManifest(){  //  Inject dyn link to manifest
        let aPath = this.doc.querySelector("link[href*='/resource/'][href*='/SYPHER_']")?.href.split("/")
        if( aPath ){
            let nTS = aPath.indexOf("resource")  //  Resource timestamp
            nTS = aPath[ ++nTS ]
            if( nTS && !Number.isNaN( nTS )){
                let eH = this.doc.querySelector("head")
                let elLnk = this.doc.createElement("link")
                let elLnkATI = this.doc.createElement("link")
                let elLnkATIP = this.doc.createElement("link")
                let sManMU = `
                {
                    "short_name": "Sypher", "name": "Sypher Web Application",
                    "description": "Sypher Next Generation",
                    "icons": [
                      {"src": "__/resource/##/@@/assets/abbv_sypher_192.png",
                        "type": "image/png", "sizes": "192x192"},
                      {"src": "__/resource/##/@@/assets/abbv_sypher_256.png",
                        "type": "image/png", "sizes": "256x256", "purpose": "maskable"},
                      {"src": "__/resource/##/@@/assets/abbv_sypher_512.png",
                        "type": "image/png", "sizes": "512x512"}
                    ],
                    "scope": "` + syPWAOpt.syPWA_Manifest_scope + `", "start_url": "` + syPWAOpt.syPWA_Manifest_start_url + `",
                    "theme_color": "#0082BA", "background_color": "#ffffff", "display": "standalone"
                }`
                sManMU = sManMU.replaceAll("##", nTS).replaceAll("__", this.doc.location.origin).replaceAll("@@",
syPWAOpt.syPWA_Manifest_static )
```

```
                elLnk.setAttribute("rel", "manifest")
                elLnk.setAttribute("href", "data:application/manifest+json," + encodeURIComponent( sManMU ))
                eH.appendChild( elLnk )

                elLnkATI.setAttribute("rel", "apple-touch-icon")
                elLnkATI.setAttribute("href", "__/resource/##/@@/assets/abbv_sypher_192.png".replaceAll("##", nTS).replaceAll("__",
this.doc.location.origin).replaceAll("@@", syPWAOpt.syPWA_Manifest_static ))
                eH.appendChild( elLnkATI )

                elLnkATIP.setAttribute("rel", "apple-touch-icon-precomposed")
                elLnkATIP.setAttribute("sizes", "152x152")
                elLnkATIP.setAttribute("href", "__/resource/##/@@/assets/abbv_sypher_192.png".replaceAll("##", nTS).replaceAll("__",
this.doc.location.origin).replaceAll("@@", syPWAOpt.syPWA_Manifest_static ))
                eH.appendChild( elLnkATIP )
            }
        }
    }
```


sypher
Sales Time Period
8x8 Weeks
Product
RINVOQ
Salesforce
ALL
Vickie Zupnik
Help
Franchise
HOMEOFFICE
Impersonate
Home office
Please select franchise by clicking icon present in top right corner of blue bar.
Once impersonate dropdown is visible, select any Region, District or Territory.
This report may contain prescription data for AMA PDRP physicians and should not be discussed with physicians, other healthcare providers, or third parties.
Elements
Console
Recorder
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Svelte
Application
Manifest
Service Workers
Storage
Storage
Local Storage
Session Storage
IndexedDB
Web SQL
Cookies
Trust Tokens
Cache
Cache Storage
Back/forward cache
Background Services
Background Fetch
Background Sync
Notifications
Payment Handler
Periodic Background Sync
Push Messaging
Reporting API
Icons
Show only the minimum safe area for maskable icons
Need help? Read documentation on maskable icons.
Primary icon as used by Chrome
192×192px
image/png
256×256px
image/png

The Service Worker Explained

Once the service worker is registered and the client is claimed then it can begin serving and caching network requests. The service worker can be registered / installed via a custom UI prompt, or in this case a click on the icon in the Chrome / Safari omnibox.

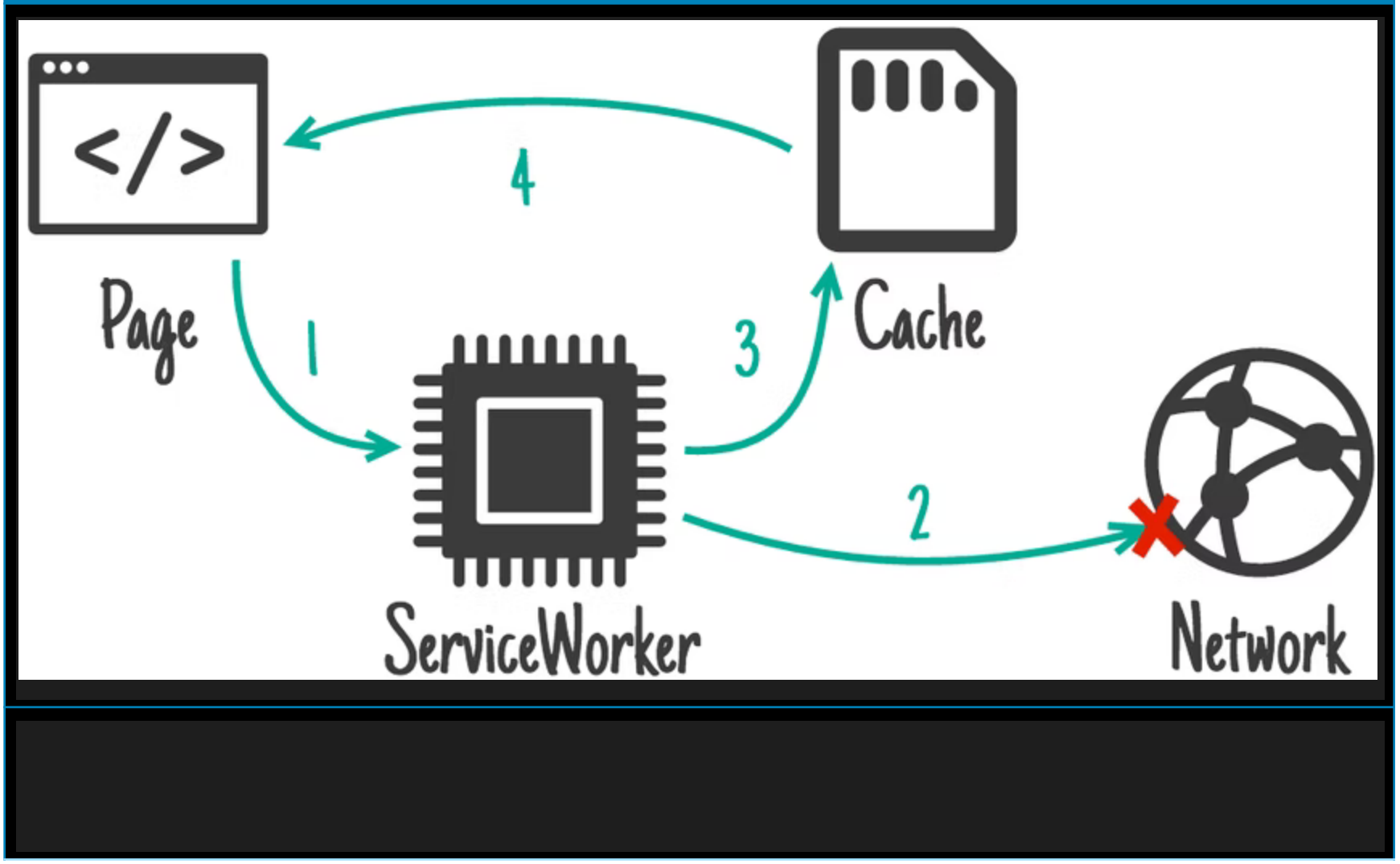
Google Workbox v6

Google Workbox is a framework for implementing various service worker caching and routing strategies.

The Network First Strategy

1. You go to the network first for a request, and place the response in the cache.
2. If you're offline at a later point, you fall back to the latest version of that response in the cache.

Network First, falling back to Cache



```
importScripts(
  'https://storage.googleapis.com/workbox-cdn/releases/6.4.1/workbox-sw.js'
);

const {registerRoute} = workbox.routing;
const {NetworkFirst} = workbox.strategies;

workbox.LOG_LEVEL = "debug";
clients.claim();

const cacheName = 'caL1';
const matchCallback = ({request}) => {
  return true;
};
const matchCallbackPOST = ({request}) => {
  return true;
};
const networkTimeoutSeconds = 3;

registerRoute(
  matchCallback,
  new NetworkFirst({
    networkTimeoutSeconds,
    cacheName,
  })
);
```

Custom Application Caching (SyPWA API)

Addressing the POST Limitation

While it is true that the service worker *can* intercept POST requests, unfortunately it cannot be put into the cache. Only GET can be put into the cache. Enter the *SyPWA API*; The solution was to, in plain JavaScript, overload the Visualforce remoting `invokeAction` method. This logic will put into IndexedDB the LZW compressed response to each POST, then when offline query and return that decompressed response. The SyPWA API binds to Visualforce remoting only. It has no dependencies on Sypher application logic, third party libraries, or Vue. The next challenge was to produce a unique *deterministic* key for each POST instance / request that would take into account all of the filters. The solution was to hash the event object resulting in a unique and reproducible key.

Local Storage Persistence (IndexedDB)

The SyPWA API stores all L1, L2, and KPI responses in a compressed indexedDB database. Utilizing IndexedDB as a storage medium has proven successful. Specifically, controlling IndexedDB with an async key-val pattern was the right decision from a complexity to power ratio perspective. The SyPWA API can then serve this content online or offline depending on the caching strategy. This API can revalidate any cached item in the background (async). The API can precache units, based on speculative user journeys, or any predefined use case. Stress testing has revealed no practical storage limitations on Windows, Mac, or iOS.

The SyPWA API is built with web technologies and as such, is subject to web security norms. Specifically the volume of data within indexedDB is subject to reclamation by the operating system. This differs for each platform. Should any or all the cached content disappear the API will fail gracefully and Sypher will essentially function as it does today. Safari: We believe that we can store at least 500MB and that storage should not be at risk of eviction as long as the app is added to the desktop.

Asynchronous key-value access to indexedDB

The key-val library wraps the native indexedDB API into a promise based subsystem in which entries can be *easily* stored and retrieved. This open-source library is entirely encapsulated within the SyPWA API.

Overload Visualforce Remoting

```
static overloadInvoke (){
    this.vfr.invokeAction = function() {  //  Overload remoting invokeAction
        if( arguments[1] ){
            let f0 = (arguments[0] + SyPWA.getHash( JSON.stringify(arguments[1])) ), f2 = arguments[2]
            //  Deterministic key
            if( SyPWA.getIsOnLine() ){
                let dStart = Date.now()
                SyPWA.lastRequest = dStart
                let fAction = function( result, event ){
                    let oEv = {"start": dStart, "event": event, "args": arguments}
                    if( result ) SyPWA.ikvSet( f0 + "_rs", SyPWA.pack( result ) )
```

```
                    if( event )  SyPWA.ikvSet( f0 + "_ev", SyPWA.pack( oEv ) )
                    if( typeof f2 == "function" ) f2( result, event )
                    oEv.end = Date.now()  //  Request is completed
                    if( event )  SyPWA.ikvSet( f0 + "_ev", SyPWA.pack( oEv ) )
                }
                if( typeof f2 == "function" ) arguments[2] = fAction
                SyPWA.fProxyInvoke.apply( this, [].slice.call(arguments) )
            }else{
                if( typeof f2 == "function" ) SyPWA.ikvGetMany( f0 + "_rs", f0 + "_ev", f2 )
            }
        }
```

Performance Intelligence

Each entry into the cache (indexedDB) is time stamped with the start and end of the request, so as to determine the duration of the original online network request. Potentially advanced caching logic might decide whether or not to make a network request based on its knowledge of how long that request is likely to take.

Network State

Note, the `navigator.onLine` properties are used. This in itself can cause known issues when a device has access to a LAN but no access to the Internet. The SyPWA API exposes two properties that can be used to determine and even set network state for testing in simulation environments that do not support a soft “airplane mode”.

Last successful network request (within API)	Network state override
`SyPWA.lastRequest`	`SyPWA.syPWA_IsOnLine_override (t/f/null)`

The SyPWA API listens to browser events that signal a change in network state. These events are then broadcast as a web notification (if the device supports).

Sypher specific caching strategies

Day of Week Revalidate

Revalidate the entire cache on the first session of a given day of the week.

On a pre-configured day of the week (default Tuesday) when the user first visits the page the entire cache will be revalidated. This logic will only run once a week if the user visits the site at least once on the configured day of the week.

Online Cache First Revalidate

*Does **not** require that a PWA be installed. Sometimes referred to as the "Stale while revalidate" pattern.*

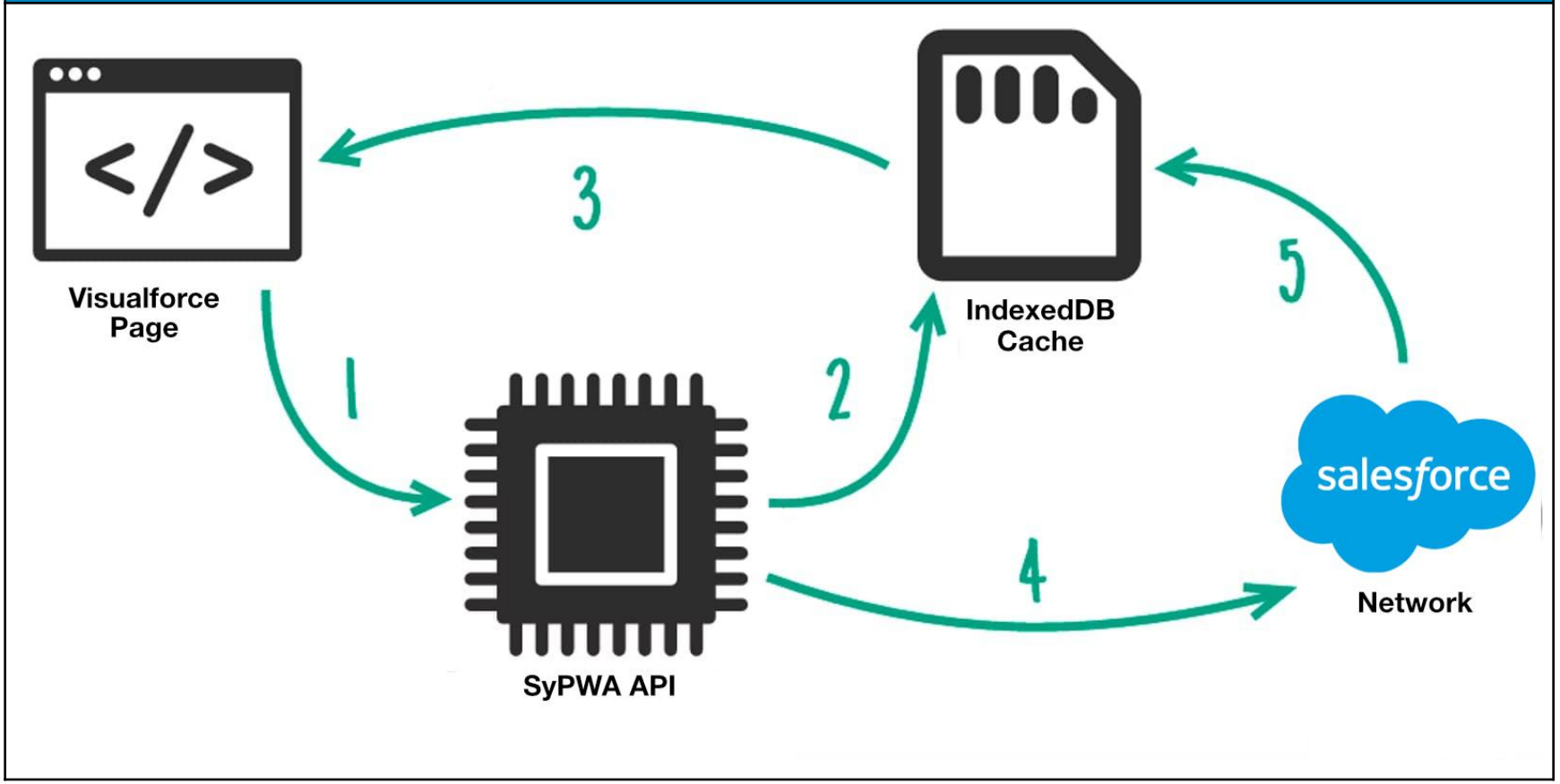
This strategy will intercept the remoting request and instantly serve from the indexedDB cache if the following conditions are satisfied:

• The network state is online `SyPWA.getIsOnLine()` • The SyPWA API strategy is `onlineCacheFirstRevalidate`	• The request exists in the cache • The entry in the cache is less than X days old

In an asynchronous, non-blocking, or lazy fashion the cache will then be refreshed from the network after about 16 seconds.

This strategy has no offline behavior but defaults to the SyPWA API behavior of serving from the cache if available.

Strategy | Online Cache First Revalidate



Online Background Sync (precaching) Level One, Two, or KPI card.

Does require that a PWA be installed. Specifically, offline functionality requires a PWA.

The strategy will present a UI (Sypher Boost) in which the user may choose to “Save for Offline”, “Clear Sypher Cache”, and potentially “Install PWA”.

Sync for Offline

Cards will be requested and cached including all possible selectable options. The user will see a progressive status while this is in progress.

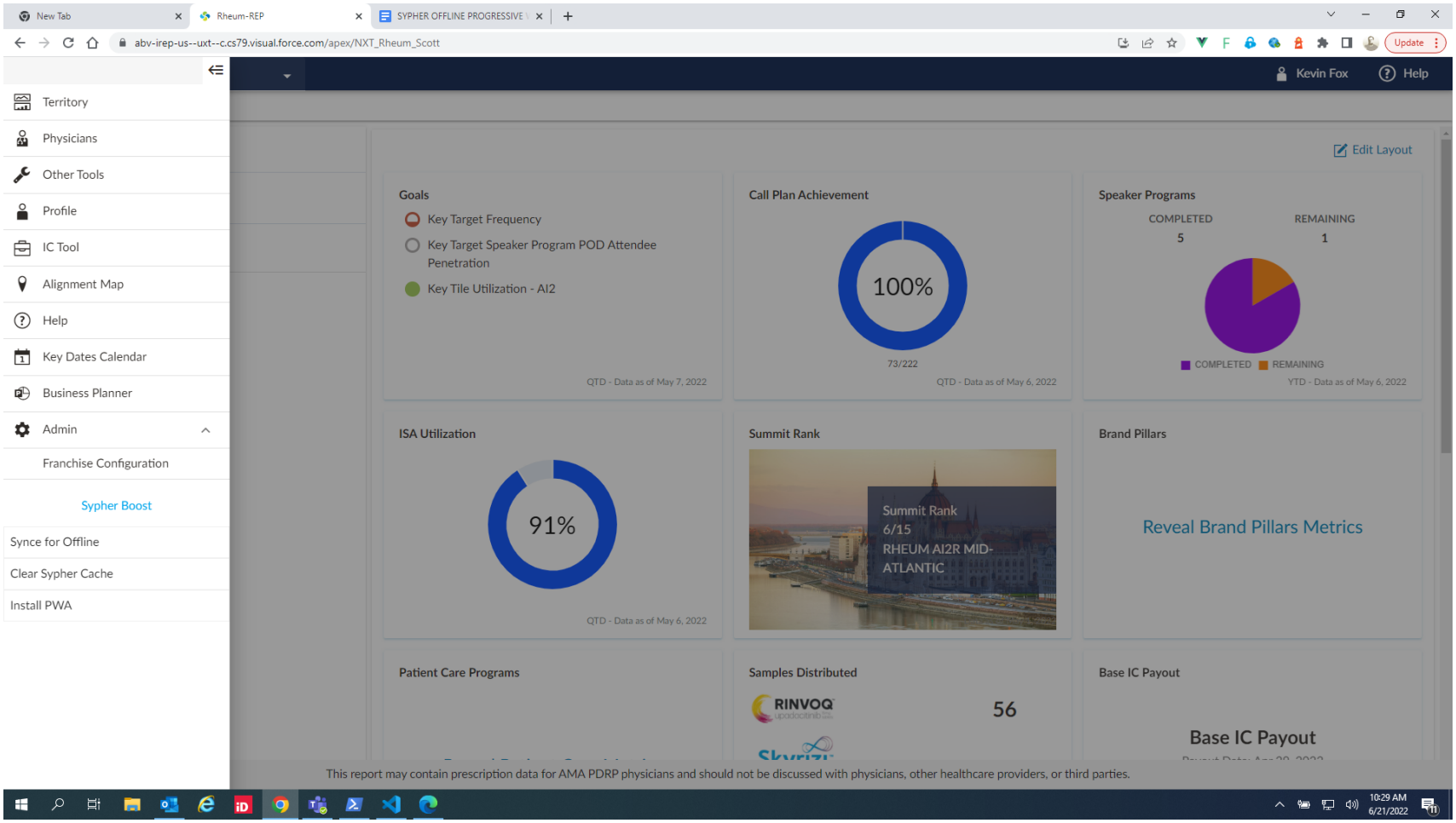
Clear Sypher Cache

Clears the indexedDB database. Useful for troubleshooting and upgrading.

Install PWA

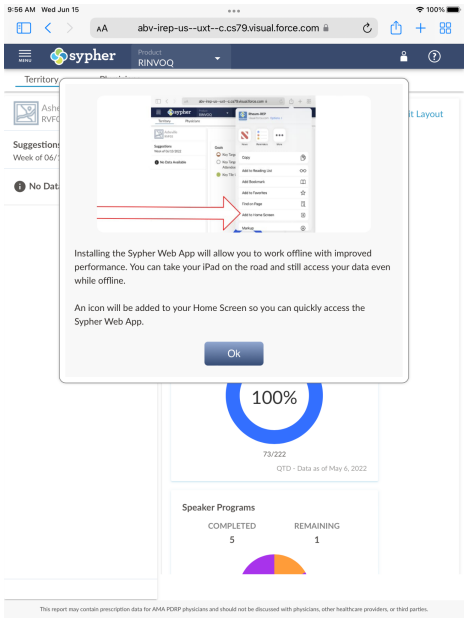
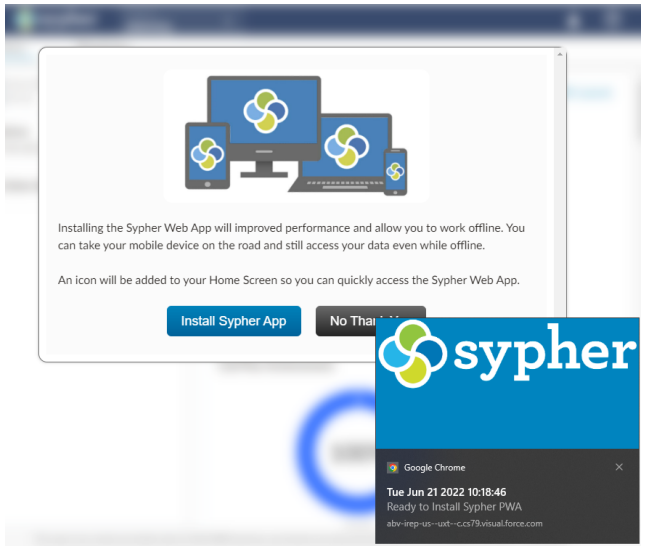
This option will only appear if the browser supports user actuated web app installations.

UX | Offline Sync



Target Environment

To install the PWA on an iPad

PWA Install UX (iPad)	PWA Install UX (desktop)
	
1. Login as a Kevin Fox 2. Manually change the url document to NXT_Rheum_Scott 3. Click the Safari arrow-box the select “Add to Home Screen”	

The target environment is an iPad / Safari / iOS. We've established VM testing infrastructure for the following models:

- iPad touch – 7th gen
- iPad Pro – 9.7 inch
- **iPad – 9th gen**
- iPad Air – 4th gen
- iPad Pro 11 inch – 3rd gen
- iPad mini – 6th gen

Conclusive Feasibility Statement (TLDR)

A Sypher Progressive Web App is a feasible solution. It has been proven that the data visualizations can be presented while offline. It has been proven that the PWA can be installed in the target environment.

Observations

1. It will likely require additional training to help end users understand how to install the PWA and what to expect while offline. However we can also expect the users *may* find it easier to use a PWA by the nature of it being a first-class and branded app on their device.
2. An effort has been made to make the PWA apparatus test automation friendly. For example there are internal API calls to dump, search, sort, and clear the contents of the cache. The SyPWA API has a debugging switch that allows for verbose status messages to be written to the dev tools console.

The Way Forward

The case has been presented for PWA to solve a very narrowly defined set of problems, mostly revolving around offline redundancies. However there are many other opportunities that present themselves as soon as we make the PWA leap.

Web / Push Notifications

The data visualization dashboard can be overwhelming for knowledge workers. They are expected to consume complex graphs and then synthesize into actionable decisions. It would be powerful if we could *proactively* alert a user in real-time when a chart is trending in the wrong direction. With web notifications we can command attention even when the browser is iconified or the Sypher tab is hidden among a million other tabs.

Unlimited Offline Storage

There is a new class of PWA that offers unlimited local storage with some additional and powerful advantages. These apps allow the user to access a local SQLite container. Think of this as the last step of your data's ETL voyage. You can do anything that you can do with any other relational database in SQL. Consider also what an impact this would have on the developer's experience; When an issue is reported by QA they simply provide the database. The biggest challenge facing devs is always a lack of accurate data, this could be the solution.

Speculative pre-caching (predictive)

Consider application logic that determines a user's most likely requests based on the individual's usage history. It can then pre-cache the top five cards most likely to be requested by a particular user / role. This has the potential to make the most commonly used data visualizations appear near instantaneous.

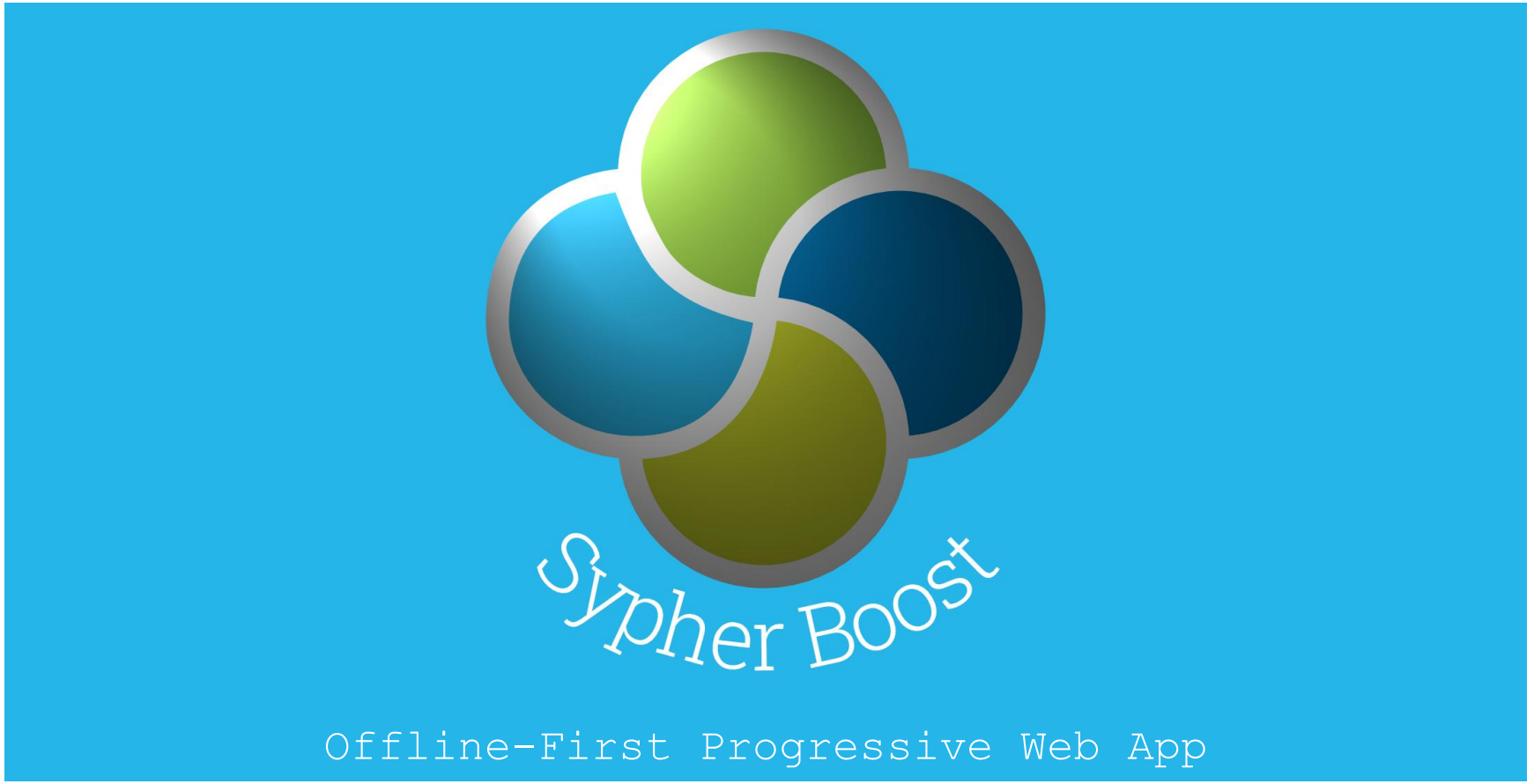
References

Google Workbox
LZWCompress.js
Curated PWA Resources
SQL Viz
idb-keyval
Offline Storage Quotas
https://demo.agektmr.com/storage/

GOOGLE ANALYTICS

SYPHER USAGE ANALYTICS

DATA VISUALIZATION CARD HEATMAP



Introduction

The Sypher Heatmap component captures the amount of time that each card is ***visible*** to an end user on an iPad or desktop.

The summation of card activity is then packaged into the analytics data layer to be consumed by Adobe Analytics or Google Analytics. Activity that occurs while offline will be updated upon reconnection to the network if the app has not been closed.

Google Analytics / Tag Manager

Snippet

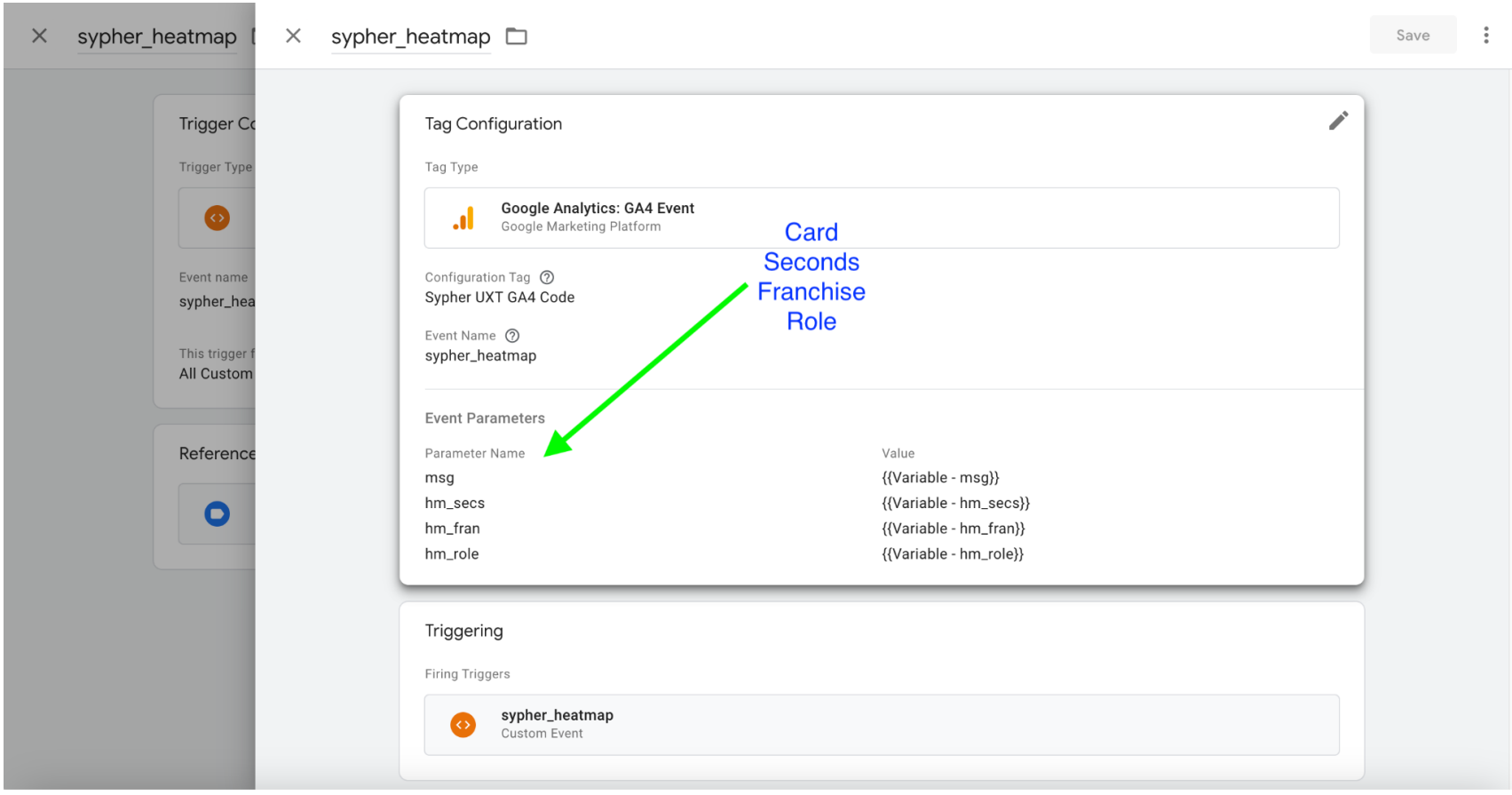
Google Tag Manager is used to integrate the custom data layer events. The snippet below must exist inside any VF page that is to produce analytics.

Tag Manager Snippet | document head

```
<!-- Google Tag Manager -->
<script>(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=
'https://www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j,f);
})(window,document,'script','dataLayer','GTM-XXXXXXXX');</script>
<!-- End Google Tag Manager -->
```

Triggers

The `sypher_heatmap` trigger captures card name, seconds, franchise, and role. This trigger utilizes 4 custom data layer variables.

Sypher_heatmap trigger


The Sypher Heatmap Component Explained

The Sypher Heatmap component utilized the ***Intersection Observer*** pattern to track when and for how long block elements (L2 cards only) are visible in the user agent viewport.

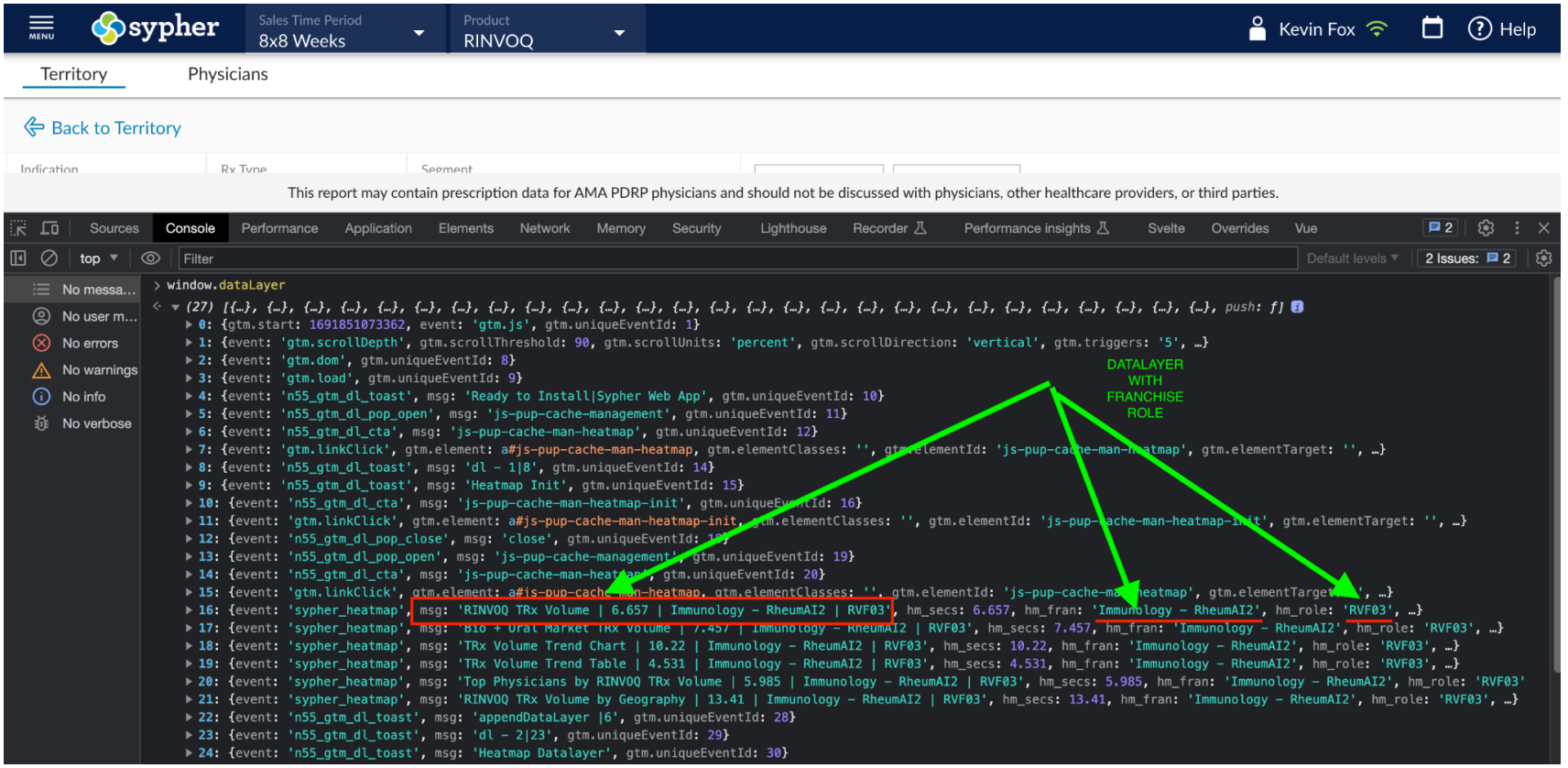
The component is configured to only report if a card is visible for greater than 3 seconds. This threshold is configurable. There are some edge cases wherein data may lose fidelity. For example if the user opens another tab or abruptly closes the browser while a L2 card is within the viewport.

The component captures the existence of L2 cards within the browser's viewport, however the application (Sypher) partially obstructs the viewport with an overlaying menu. Meaning that the Heatmap may report that a particular L2 card is in view a few ***microseconds*** before it is actually visible. This discrepancy is so small that it is not statistically significant.

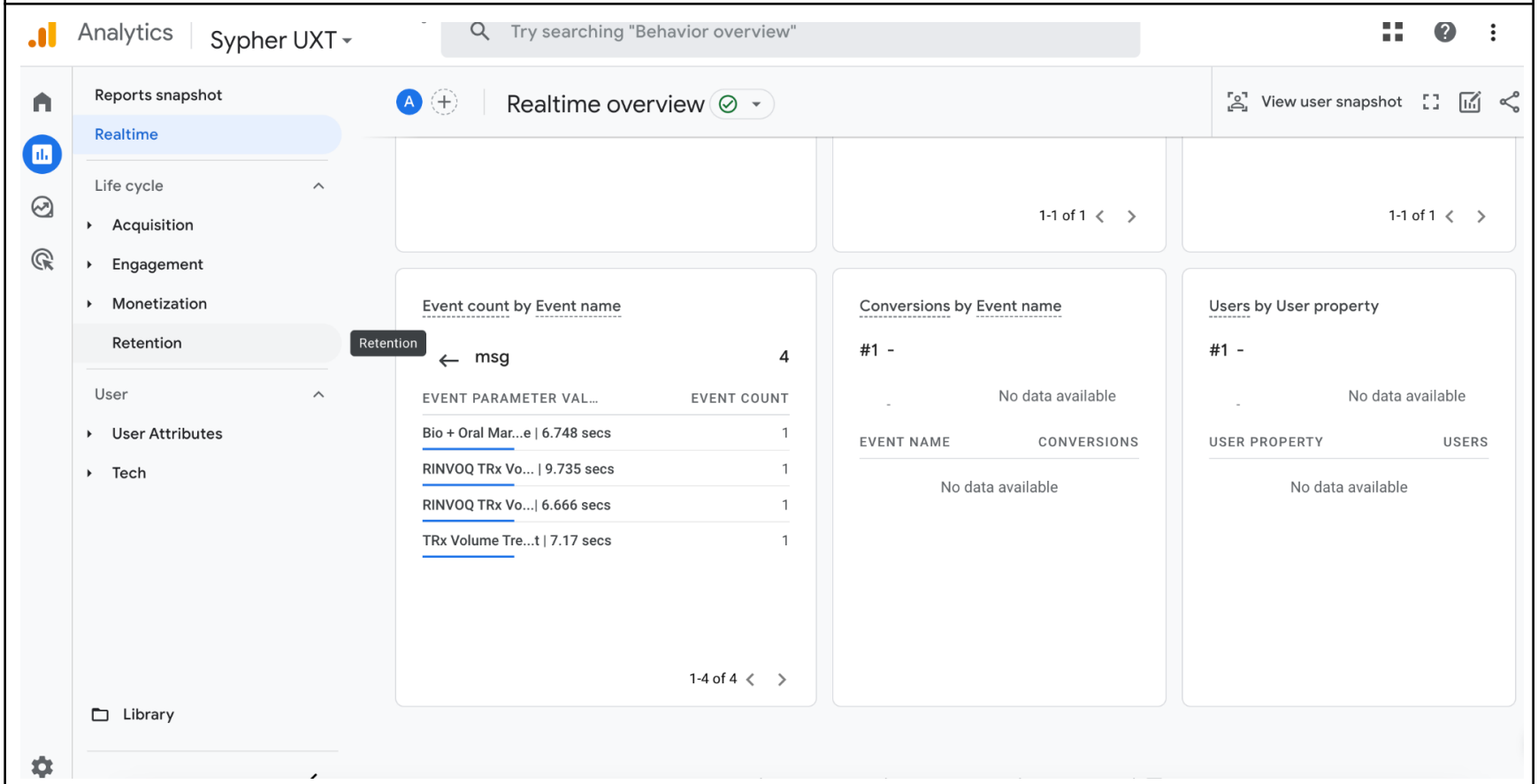
The data layer entries that the heatmaps create are additive, meaning that there may be more than one for a single component. This is because the end user viewed a particular L2 component, exited the card, then reentered the card.

Testing

Functionality within the browser can be verified by inspecting the data layer in the console.

Data layer within console


View the events in Google Analytics



Source Code

Sypher Heatmap JavaScript Class

```
class SyHeatmap { // Sypher Heatmap Begin
    static oObserved = {}; static aObservedEl = []; static aQryContext = []
    static oIntObserver = null; static NTHRESH_SECS = 3; static bIsInit = false;
    static reInit ( _q, _c = document ){ // DOM bind to context element
        if( _q && _c ){
            this.aQryContext = [ _q, _c ]
            this.oObserved = {};
            this.aObservedEl = [ ... _c.querySelectorAll( _q[ 0 ] ) ];
            this.aObservedEl.forEach( ( elO )=>{
                let elOsib = elO.nextElementSibling
                const sCap = elOsib.heatmapCaption = elO.innerHTML
                this.oObserved[ sCap ] = elOsib
                this.oObserved[ sCap ].heatmapTime = []
            } )
            this.oIntObserver = new IntersectionObserver( ( entries )=>{
                entries.forEach( ( oEnt )=>{
                    if( oEnt.target?.heatmapCaption ){
                        const sCap = oEnt.target.heatmapCaption
                        if( this.oObserved[ sCap ].heatmapTime.length ){
                            this.oObserved[ sCap ].heatmapTime.push( {"state": oEnt.isIntersecting, "ts": new Date().getTime() })
                        }else{ // No first time false (vis when page loads)
                            if( oEnt.isIntersecting ){
                                this.oObserved[ sCap ].heatmapTime.push( {"state": oEnt.isIntersecting, "ts": new Date().getTime() })
                            }
                        }
                        if( oEnt.isIntersecting ){
                            oEnt.heatmapTotal = SyHeatmap.totalHeatmapTime( this.oObserved[ sCap ].heatmapTime ); // Sum and dif array vals
                            if( syPWAOpt && syPWAOpt?.syPWA_DEBUG_LOG ){
                                console.log( syPWAOpt.syPWA_DEBUG_LOG, " total " + oEnt.target.heatmapCaption + " | " + oEnt.heatmapTotal )
                             }
                        }
```

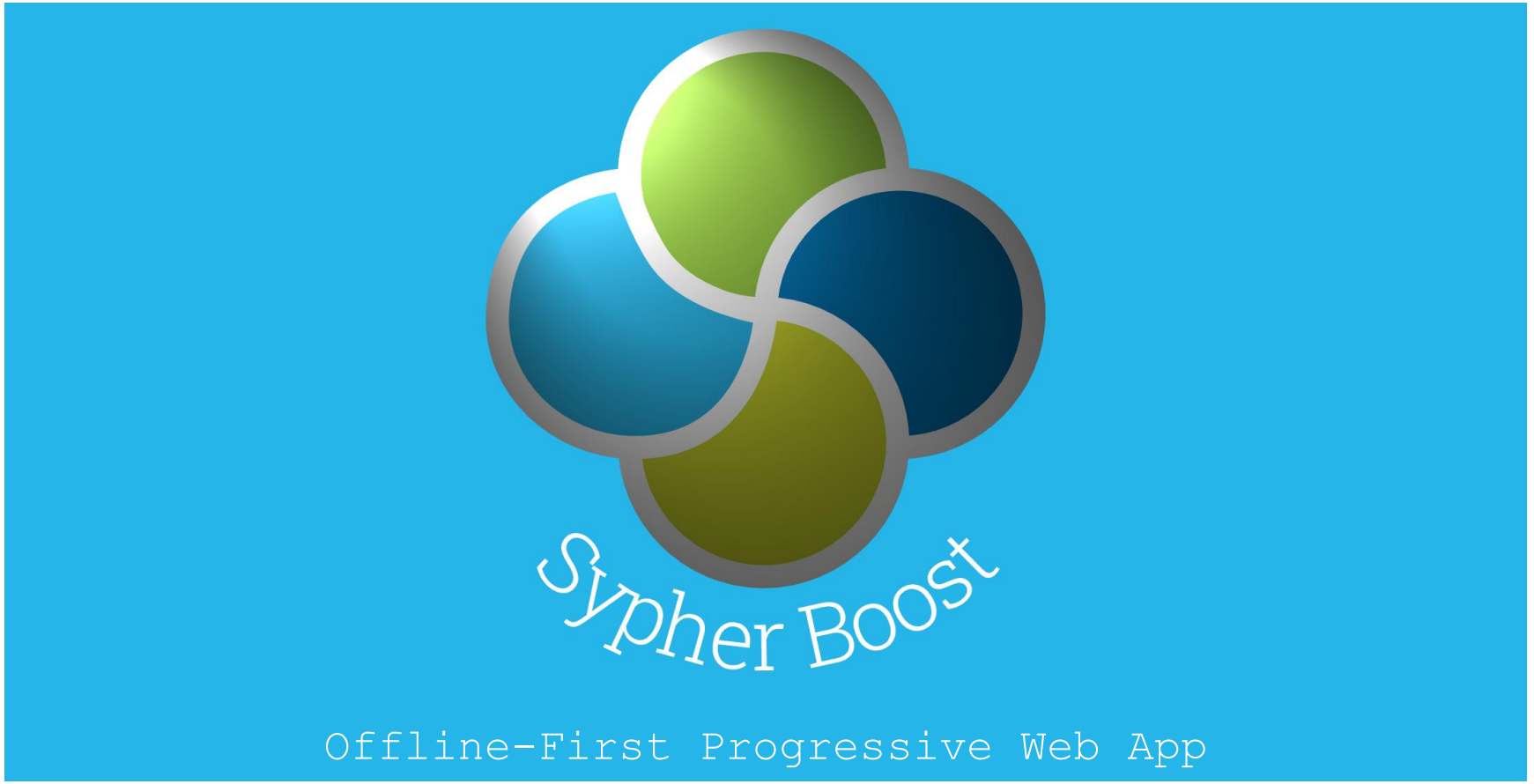
```
                }
            } )
        } )
        this.aObservedEl.forEach( ( elObs )=>{
            let sCap = this.oObserved[ elObs?.innerHTML ]
            if( sCap ) this.oIntObserver.observe( sCap )
        } )
        if( !this.bIsInit ){
            this.bIsInit = true;
            setInterval( ()=>{ SyHeatmap.threshold() }, this.NTHRESH_SEC )
        }
        return this;
    }
}
static totalHeatmapTime ( aHeatmapTime ){ // Return total time on component in secs
    let nTotStart = 0; let nTotEnd = 0; // Note: IntrSec Observ will fire FALSE once upon page load for each entry not visible
    if( aHeatmapTime.length ){ // Append a FALSE as NOW if the last item is not FALSE (currently in viewport)
        let aDTO = [ ... aHeatmapTime ]
        if( aDTO[ aDTO.length - 1 ].state == true ) aDTO.push( {"state": false, "ts": new Date().getTime() } )
        aDTO.forEach( ( oHMTimes )=>{
            if( oHMTimes.state ) nTotStart = nTotStart + oHMTimes.ts
            if( !oHMTimes.state ) nTotEnd = nTotEnd + oHMTimes.ts
        } )
    }
    return ( nTotEnd - nTotStart ) / 1000; // in seconds
}
static genHeatmap ( nThresh = this.NTHRESH_SECS ){ // Return a simple arry of current hm usage filt threshold
    let aCurHM = []
    if( this.aObservedEl.length ){
        for ( const sCap in this.oObserved ) {
            let nTotal = SyHeatmap.totalHeatmapTime( this.oObserved[ sCap ].heatmapTime )
            if( nTotal && ( nTotal >= nThresh ) ) aCurHM.push( {"caption": sCap, "secs": nTotal } )
        }
    }
    return aCurHM;
```

```
    }
    static resetHeatMap(){
        this.aObservedEl.forEach( ( elObs )=>{
            let sCap = this.oObserved[ elObs?.innerHTML ]
            if( sCap ) this.oIntObserver.unobserve( sCap )
        } )
    }
    static appendDataLayer (){ // Iterate filtered heatmap and add to DL - return count
        let iCnt = 0
        if( window.dataLayer ) {
            SyHeatmap.genHeatmap().forEach( ( oHMSum )=>{
                let sMsg = oHMSum.caption + " | " + oHMSum.secs + " | " + sypher.salesforceGlobal.franchiseconfig.Name + " | " +
sypher.salesforceGlobal.loginuser.UserRole.Name
                window.dataLayer.push( { "event": "sypher_heatmap", "msg": sMsg, "hm_secs": oHMSum.secs, "hm_fran":
sypher.salesforceGlobal.franchiseconfig.Name, "hm_role": sypher.salesforceGlobal.loginuser.UserRole.Name } )
                iCnt++;
            })
        }
        SyHeatmap.resetHeatMap() // Reset and Rebind
        SyHeatmap.reInit( this.aQryContext[ 0 ], this.aQryContext[ 1 ])
        return iCnt;
    }
    static threshold (){
        if( this.aQryContext.length ){ // Must have already been fired
            let elSame = this.aQryContext[ 1 ].querySelector( this.aQryContext[ 0 ] );
            if( elSame ){
                if( elSame.innerHTML != this.aObservedEl[0]?.innerHTML ){ SyHeatmap.appendDataLayer() }
            }else{ SyHeatmap.appendDataLayer() }
        }
    }
} // Sypher Heatmap End
```

Optimized Architectural Tier

SYPHER BOOST | PROXY AGENT

TECHNICAL DESIGN Phase I



Introduction

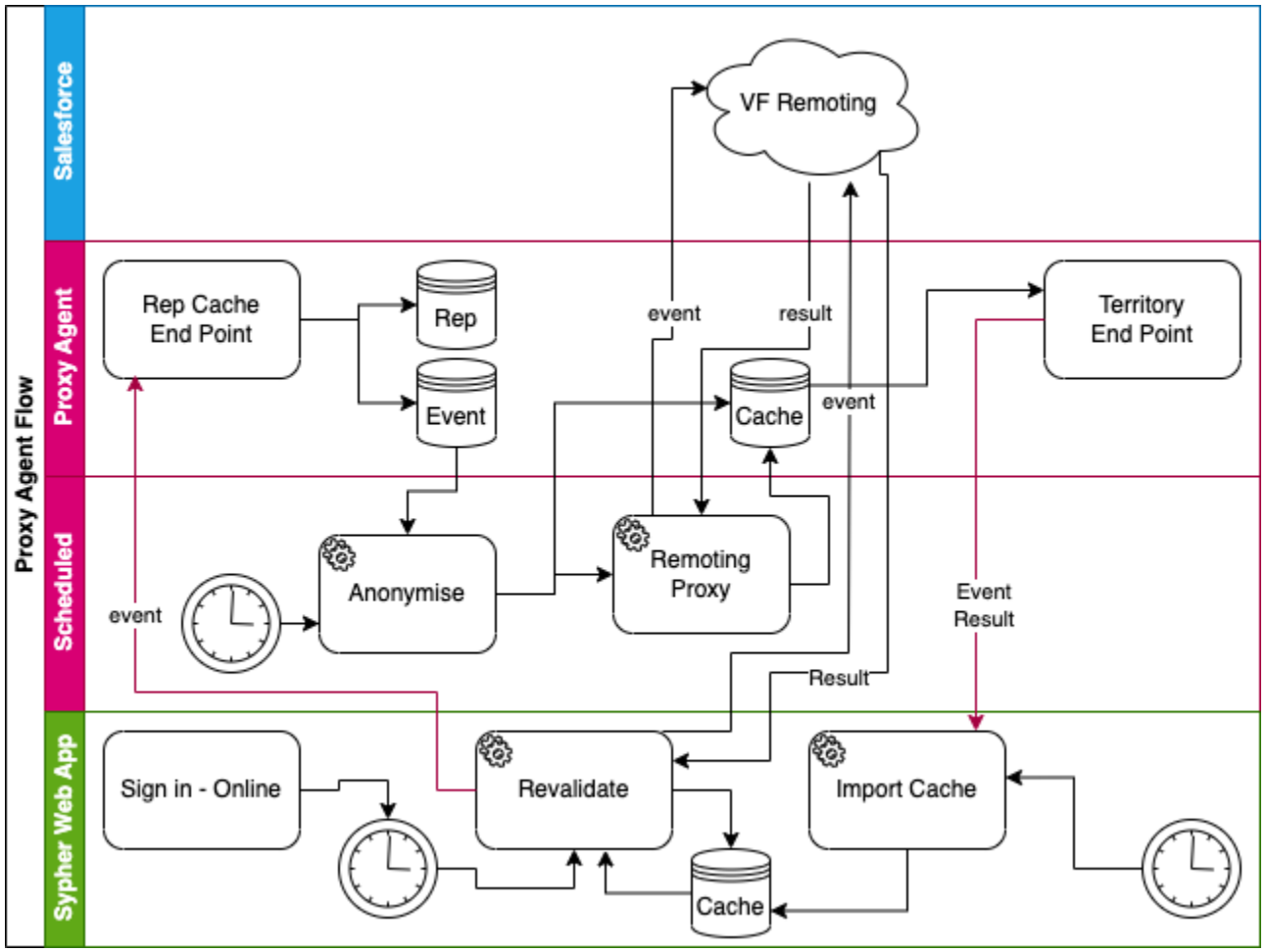
This document is the Sypher Proxy Agent detailed technical design.

Optimized Architectural Tier

The Sypher Proxy Agent is a service that securely optimizes, aggregates, pre-processes, and stages Sypher cached data visualizations in a way that allows reps to access data at near instantaneous speed even while offline.

Future-proof: The Sypher Proxy Agent is strategically positioned to directly process additional, disparate data sources in the future.

Process Flow Diagram



Data Structures

Entity Relations Diagram

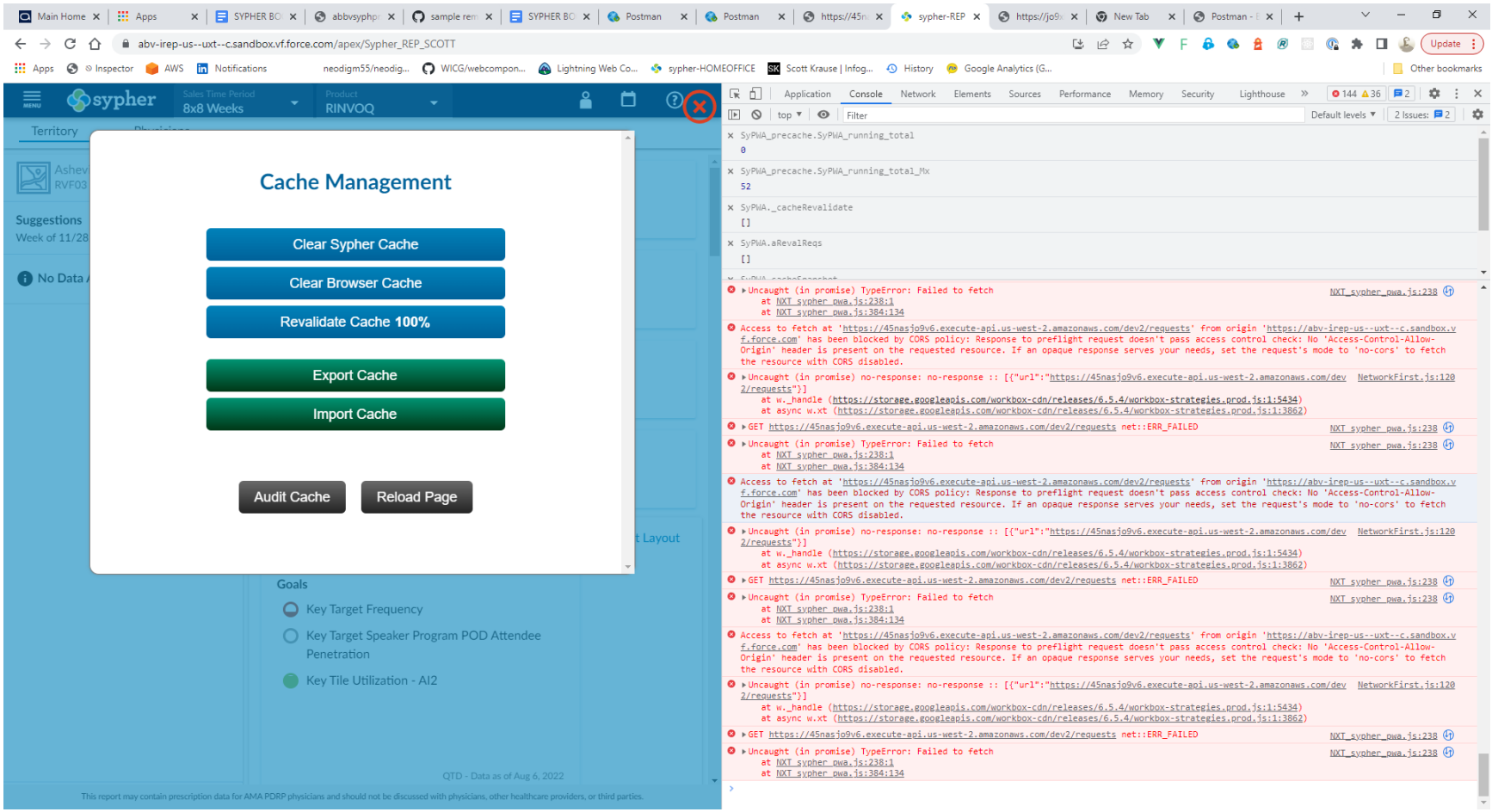
Cache JSON transport

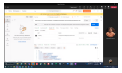
REST API

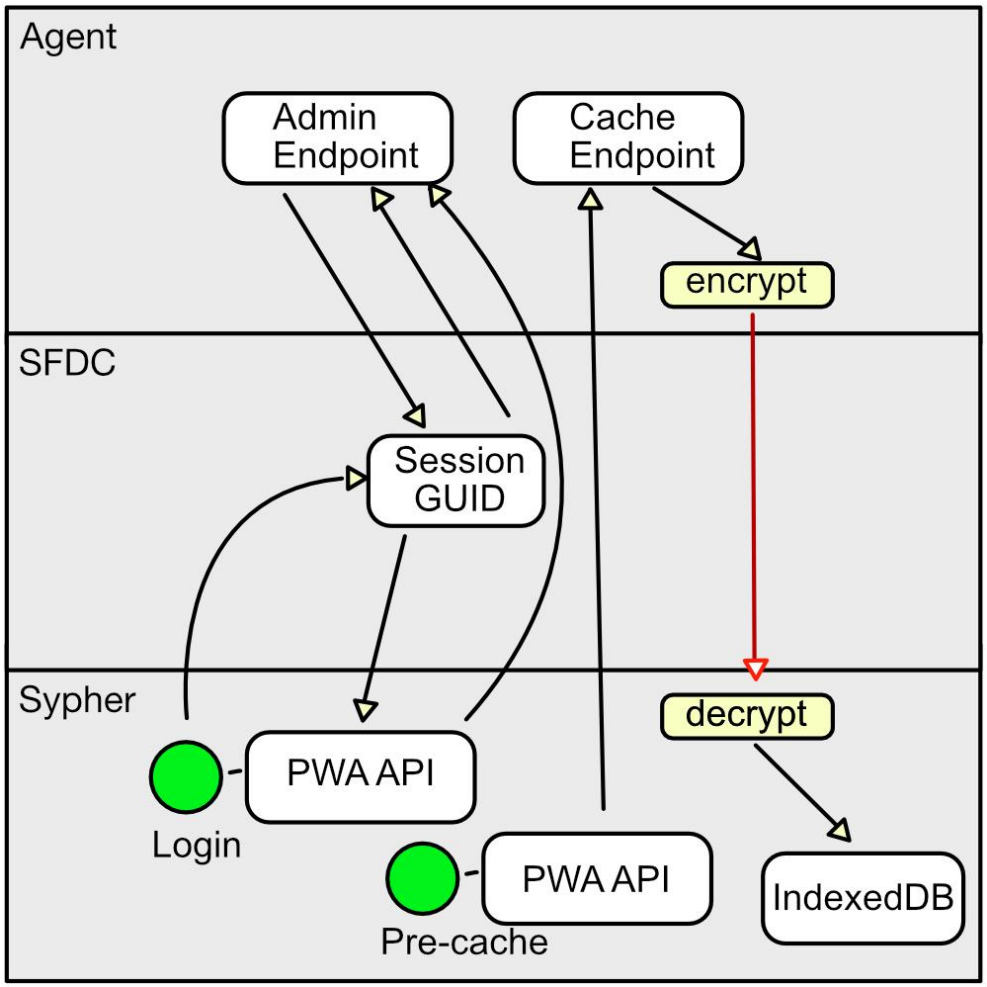
End Point	Desc
Requests List	List current contents of the Requests repository

Phased Deliverables

#	Phase	Deliverable	Level of Effort
1	Design	Establish objectives for the POC.This document.	**S**
2	API	Define end points, data contracts, and transition logic.	**M**
3	Deploy Service	AWS - should be accessible via Postman / Insomnia. CORS - Whitelist EB	**L**
4	Integrate with Sypher	Client side consumption - VF Page - Client Automation API	**S**
5	Dev Scheduled Tasks	Batch Cache asset processing	**M**
6	Integrate with SF	Remoting automation	**L**
7	Executive Summary	Publish findings: Stress test, ...	**S**


Cache Management
Clear Sypher Cache
Clear Browser Cache
Revalidate Cache 100%
Export Cache
Import Cache
Audit Cache
Reload Page




Agent
Admin Endpoint
Cache Endpoint
encrypt
SFDC
Session GUID
Sypher
decrypt
PWA API
Login
PWA API
Pre-cache
IndexedDB