

SAG Foundation Performance Audit Report

This is an in-depth performance audit of the SAG-AFTRA Foundation website that covers both the application code and WP Engine hosting environment. It identifies potential performance bottlenecks and provides recommendations for optimizing the website to perform as fast as possible and efficiently utilize server resources.

1- The “Program Calendar” Filter

The data displayed on the "program calendar" page loads significantly slowly on both the public page and the member portal. The average TTFB for this AJAX is around 4s and is expected to drop to <2s after optimizations.

a- The “N+1 Query” Problem

The way the filter is currently working is that the `while($query->have_posts())` loop in the **get_filtered_programs** function makes multiple new requests to the database per event to find:

- The user's RSVP status
- The location name
- The program type
- The eligibility rules

If we have 12 events on the page, the code goes to the database, gets the first event's location, comes back. Goes to the database, gets the first event's RSVP status, comes back. It repeats this dozens of times per event. It's just like going to the grocery store 50 times to buy 50 items, instead of buying all of them all in one cart.

Solution: We should "bulk fetch" this data with single database queries and match them up in PHP.

b- The Missing Database Index

As highlighted in the Claude report, the **wp_events** table doesn't have an index, so MySQL has to scan every single row in the table to find the right event.

Solution: We should add a composite index on the **post_id** and **deleted_at** columns in the events table. This alone should speed up the events page.

c- The admin-ajax.php Heavy Lifting

The events filter uses WordPress's default **admin-ajax.php** system. When we use this method, WordPress has to load the entire WordPress code, plus every single active plugin, including heavy ones like Wordfence and Gravityforms, just to fetch events.

Solution: We should move the **get_filtered_programs** to a custom REST API Endpoint (e.g, /wp-json/saf/v1/programs). REST API endpoints are significantly lighter and faster because they don't force the server to load all the unnecessary admin-heavy scripts.

Same recommendation for the "Staff List" filter and every other admin screen using AJAX calls. Replace them with rest api endpoints. They are more clean, fast and efficient.

Basically, we need a complete refactoring of the filters logic.

d- Eligibility Rules:

Right now, for every event, we check a user's eligibility status like membership level, location, or past history on the fly, and this puts a heavy processor drain.

Solution: Cache the user's eligibility status. If a user is a "Member Local", they are going to be a Member Local for the whole session. Store that result temporarily using WordPress Transients, so code only calculates their status once per hour, rather than every time they click a filter.

2- Optimize the Database:

The total database size determined by wp-engine is 401.64MB out of which 162.34MB is redundant space. It leads to inefficient storage and slower queries. The database needs a thorough cleanup.

Trends	Database details	Top 50 files
Total storage: 23.53 GB		
SQL		401.64 MB
Autoloaded		83.71 KB
Bloat		162.34 MB
Bloat %		40%
Tables		85

3- Unwanted Plugins Loading on Every Page

This problem is both on the frontend and backend (admin portal and member portal) of the website. Many unwanted plugins load their assets, such as CSS and JS files, on each page load, increasing HTTP requests and the total page size. We should only load relevant plugins on each page. For instance, the Wordfence and Gravity Forms plugins are loading on every page.

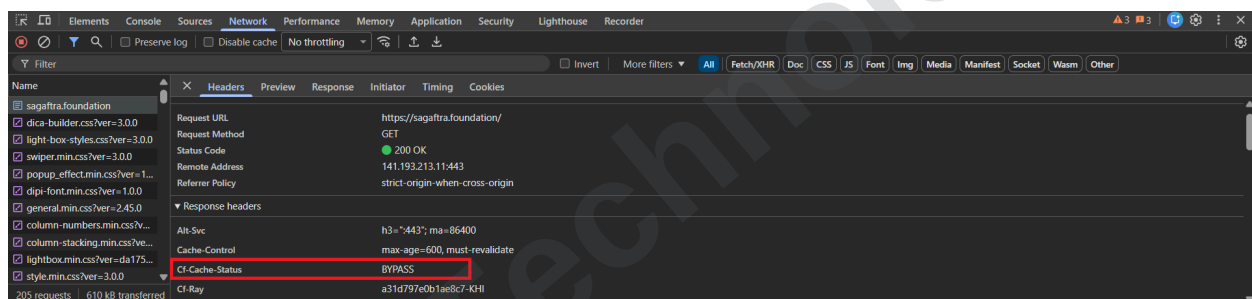
The screenshot shows the Chrome DevTools Network tab with the 'Initiator' column selected. A red box highlights the 'Request initiator chain' for the dashboard page. The chain lists the following resources:

- http://safra.local/dashboard
- http://safra.local/wp-content/plugins/dg-carousel/assets/css/dica-builder.css?ver=3.0.0
- http://safra.local/wp-content/plugins/dg-carousel/styles/light-box-styles.css?ver=3.0.0
- http://safra.local/wp-content/plugins/dg-carousel/styles/swiper.min.css?ver=3.0.0
- http://safra.local/wp-content/plugins/di-pixel/dist/public/css/popup_effect.min.css?ver=1.0.0
- http://safra.local/wp-content/plugins/di-pixel/dist/admin/css/di-pa-font.min.css?ver=1.0.0
- http://safra.local/wp-content/plugins/di-pixel/dist/public/css/general.min.css?ver=2.45.0
- http://safra.local/wp-content/plugins/di-responsive-helper/assets/public/css/column-numbers.min.css?ver=2.4.3
- http://safra.local/wp-content/plugins/di-responsive-helper/assets/public/css/column-stacking.min.css?ver=2.4.3
- http://safra.local/wp-content/plugins/di-to-image-hover/styles/lightbox.min.css?ver=04766a243ead61cb29315784b35a7e
- http://safra.local/wp-content/plugins/di-to-image-hover/images/next.png
- http://safra.local/wp-content/plugins/di-to-image-hover/images/loading.gif
- http://safra.local/wp-content/plugins/di-to-image-hover/images/close.png
- https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css?ver=04766a243ead61cb29315784b35a7e
- https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/fonts/fontawesome-webfont.woff2?v=4.7.0
- http://safra.local/wp-content/plugins/saf-plugin/assets/css/bootstrap.min.css?ver=04766a243ead61cb29315784b35a7e
- data:image/svg+xml,%3csvg xmlns='http://www.w3.org/2000/svg' viewBox='0 0 16 16' fill='%230502c5' %3e%3cpath fill-rule='evenodd' d='M1.646 4.646 5.5 0 11.414 14.142 15.293 0 11.414 4.646 5.5 0 1.646 4.646z' /%3e%3c/svg %3e
- data:image/svg+xml,%3csvg xmlns='http://www.w3.org/2000/svg' viewBox='0 0 16 16' fill='%230502c5' %3e%3cpath d='M2.93 2.93 11.41 11.41 14.14 0 0 14.14 2.93 11.41z' /%3e%3c/svg %3e
- https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/fonts/fontawesome-webfont.woff2?v=4.7.0
- https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/fonts/fontawesome-webfont.woff2?v=4.7.0
- https://safra.local/wp-content/plugins/saf-plugin/assets/css/public-style.css?ver=04766a243ead61cb29315784b35a7e
- https://safra.local/wp-content/plugins/saf-plugin/assets/css/public-style.css?ver=04766a243ead61cb29315784b35a7e
- https://safra.local/wp-content/plugins/saf-plugin/assets/css/public-style.css?ver=04766a243ead61cb29315784b35a7e

Solution: For the frontend, we can use a plugin called “Perfmatters” to control which plugin scripts we want to load on each page. For Backend pages, Perfmatters might work as well, but if it doesn’t, then we would have to use WordPress functions called **wp_dequeue_script** and **wp_dequeue_style** to automatically unload unwanted scripts on each page and target pages by url structure (wildcards) so we don’t have to repeat the process of de-queuing scripts for every single page.

4- The Divi Theme Split Testing Cookie

Et_pb_ab_view_page_4401101. This is a Divi theme split testing cookie. This cookie is loading on every single page, preventing cloudflare to cache the pages at the edge location, resulting in pages getting served from the WP Engine server cache and sometimes directly from the server, putting strain on hosting resources. It’s currently “ON” in the global header template of the divi theme builder option. We would have to turn it off from there.



5- Deactivate Unused Plugins

I noticed there are some plugins installed on the website that I don’t see being used anywhere, like the “Divi Booster” plugin. So remove such plugins from the website.

6- Smart Slider Pro Plugin

This plugin is loading 5 sliders on the homepage, increasing the total page size. Load only the ‘hero section’ slider and load the rest of the sliders after the home page is loaded completely, and lazy load the slider images.

For the rest of the pages where this plugin is loading sliders, do the same thing.

7- Recurring JavaScript Errors (Mentioned in Consolidated Report)

I believe that's a minor issue and not affecting the loading time much.

8- Front-End Asset Weight (Mentioned in Consolidated Report)

That is significantly affecting the loading time and is already discussed above in the report.

9-. Legacy Rails (Mentioned in Consolidated Report)

It is a good idea to do that, and I would put it in medium priority. It has a low impact on current load times.

10- Admin Portal (Staff-Facing) Performance (Mentioned in Consolidated Report)

I agree, but the main culprit is not the mentioned JavaScript files but the AJAX call /admin/events screen makes, as discussed in point 1c above.

Conclusion / My Recommendation:

The biggest bottleneck is with the application code, especially the custom plugin 'saf-plugin'. We should optimize it first, as recommended.

I believe that the most time taking task would be to redo the "**Program Calendar**" filter logic as explained above. So we should prioritize that.

The hosting resources also do not seem to be appropriate for the current setup.

During normal traffic, I observed that sometimes, the WordPress regular Admin dashboard pages were slow to navigate and Divi Builder was lagging as well, indicating a potential need for a server upgrade.

However, application-level optimizations must be the primary focus, as they are the root cause of the slowness, before considering an upgrade to hosting capacity.