Members of the Libertarian Party,

I am excited to present the evaluation of our CiviCRM platform from CiviCRM experts, Skvare, to the membership. The report outlines specific technical recommendations as well as analysis on CiviCRM's role in the Party ecosystem. I wanted to provide a breakdown of the technical recommendations for laymen and outline next steps we are taking based on these recommendations to increase fundraising, empower our state affiliates, and address technical debt.

Technical Debt and Recommendations

Technical debt "describes the consequences of software development actions that intentionally or unintentionally prioritize client value and/or project constraints such as delivery deadlines, over more technical implementation, and design considerations." In English, what this means is that businesses frequently find themselves in positions where daily activities are hampered because the maintenance of tools that worked for years have not been prioritized. In the worst cases, businesses are forced to scale back revenue-generating operations to address these systemic technology issues to avoid being completely unable to function.

Skvare points out in their report that while their assessment of our CiviCRM system is "generally positive," they warn that there are no "silver bullet" solutions. This is because the Libertarian Party finds itself in a classic case of owing technical debt. We operate on a 2-year party leadership cycle within a 4-year election cycle, which incentivizes the Party to operate more like a publicly traded company that must meet quarterly goals to please shareholders than a nonprofit that is building an organization to accomplish a goal. For many years, the party has chosen short term efforts over long term technological health. This is rational behavior given the incentive structure and limited resources of the party, but we must be able to understand how we got to this position if we hope to navigate out of it. Skvare says "in our work with similar organizations, we've consistently seen that too little focus on ongoing development leads to increases in other areas: more support and training issues, more administration and maintenance work, and just generally more staff time spent fielding complaints from dissatisfied users." We must adopt a focus on continuous development so that our staff and volunteers can get back to the productive activities that make the party function. Skvare has laid out issues and solutions on which to focus this development effort to get us back to a fully operational level as quickly as possible:

1. Permissions
    a. There is no unified permission model in our CiviCRM system. The Party needs to enable national, state, and local users to securely access the information they need at their level of engagement. The report finds that our permission model is overly complex. Skvare states that "they have never seen so many Civirules in one site" to manage permissions. The permission model inhibits quick and secure user access and is far too restrictive for state affiliates. Skvare has provided a set of action items to simplify our permission model which will empower states as well as have an impact on website speed due to the current permission model being very resource intensive.
2. CiviCRM component configuration
    a. CiviCRM relies on several components to function, such as a web server platform and a database. These components are separate from CiviCRM and have their own configurations that must be managed. Skvare has reported that many of these component settings are not optimal and has provided a list of recommended configurations. Fixing these settings can lead to substantial gains in the speed of the website.
3. Documentation:
    a. In an organization like the Libertarian Party, extremely strong documentation is needed to ensure disparate and decentralized users can effectively use the system. CRMs are inherently complicated and often unintuitive pieces of software; without good documentation we must rely on user-to-user training. Evan McMahon in Indiana has been Herculean in his efforts to train as many people as possible on CiviCRM and I cannot thank him enough, but Skvare has highlighted that our current documentation is

not sustainable. They have provided top priority processes to document and recommended ways to store the documentation so that it is secure, but accessible to as many users as possible.

4. Use Case
    a. Our CiviCRM system was initially implemented by a group of technical party members who frequently had to push big features without planning or direction to convince state boards to join the network and prevent prior LNC boards from canceling the project. There was not time in the initial development to carefully consider how to best serve the ecosystem. Skvare points out that while CiviCRM is “is an effective solution” for meeting the party’s ongoing needs, it is a trap to try to “be all things to all people,” and careful planning must be used. For instance, it is not possible to accurately query our membership numbers natively in CiviCRM because of the way reporting works. CiviCRM also natively lacks many critical fundraising tools that our staff and volunteers experience using to successfully produce growth in other roles. Custom work could be done to make these features available in CiviCRM, but Skvare recommends that we carefully decide what it is important to have available in CiviCRM itself and what makes sense to integrate into another tool. They have provided detailed processes to help us make these determinations so that we can address our most immediate CiviCRM pain points, such as fundraising.
5. Hardware
    a. One area that we have little to no technical debt in is our actual hardware infrastructure. This is in large part due to the efforts of Ken Moellman who has worked tirelessly over the years to keep our hardware configuration in alignment with industry standards.

Next Steps

1) The clearest throughline of the report is that it is going to take a substantial amount of time and resources to get CiviCRM to a place where it is effectively powering our organization. Given the Party’s current fundraising woes, time is a luxury we do not have. As such, we have identified new fundraising tools, to address short-term fundraising needs: WooCommerce and Anedot have modern web features and checkout UX capabilities that will allow our staff to circumvent many Civi Contribution Page constraints, iWave will allow us to leverage screening small batches of major donors, our new database visualizer will allow us to accurately analyze and segment our data, and new MailChimp/Drip/SMS tools will allow us actually email those segments, overcoming native Civi constraints to email and text useful donor segments. By implementing these, we will be much better positioned to engage lapsed members, run effective campaigns to generate new leads, and convert those new leads into active donors. This is the exact type of decision that generates further technical debt, but you will see in the steps below that we have a plan to continuously support and improve our software suite to address all our technical debt in an effective manner.

2) As time is a precious commodity, we must implement the outlined technical recommendations as quickly as possible. We will achieve this by engaging Skvare to help implement these recommendations. Given their expertise, Skvare can much more rapidly implement the fixes and tweaks needed to remedy the areas that are slowing the rest of the organization down.

3) Skvare stresses the importance of implementing a requirements gathering process that collates the needs of the entire organization and makes determinations about what tool should perform which function. We have spoken about and begun this requirements gathering process. Let me detail those steps here:

1. Identify the Stakeholders

Identify the key individuals from the national, state, and county party fundraisers, activists, and leaders who will be using the CRM platform most frequently. They will be the primary users, and their needs and concerns should be taken into consideration first.

2. Stakeholder Interviews

Organize individual or group interviews with the identified stakeholders to understand their needs, pain points, and what they expect from a CRM platform.

Users: Focus on understanding their day-to-day operations, what kind of data they deal with, what kind of data they need, the tasks they perform that could be simplified or automated, and the reports they usually generate, understand their fundraising strategies, their interactions with donors, their reporting needs, and any challenges they face in regards to fundraising that could be addressed by the CRM

Non-user leaders: Their concerns may be more strategic. We will try to understand their long-term goals for the party, what kind of metrics they are interested in, and how the CRM can assist in making informed decisions.

3. Stakeholder Surveys

Supplement interviews with surveys to gather additional data and reach stakeholders who might not be available for interviews. These surveys should aim to quantify the needs identified in the interviews, establish the priority of various features, and identify any additional needs not covered in the interviews.

4. User Stories and Use Case Development

Based on the information gathered from interviews and surveys, we will create user stories and use cases. These will help in detailing the specific needs of users, how they interact with the system, and what the system should provide in return, as well as help lay out a roadmap of when and what order things can be done.

5. Consolidation and Prioritization

Consolidate all the gathered information and prioritize the requirements based on the needs and strategic goals of the party. It is likely that we will have more requirements than we can address at once, so it is crucial to determine which requirements are most important.

6. Validation Workshops

Conduct validation workshops where we present the prioritized list of requirements to the stakeholders. The goal is to validate that we correctly understood their needs and that they agree with the priorities we set.

7. Documentation and Communication

Finally, document all the requirements in a formal requirements specification document and share it with all the stakeholders. Ensure that everyone is on the same page regarding the requirements for the CRM platform.

It is important to remember that requirement gathering is an iterative process. As we go through this process, new needs may emerge, and priorities may change. We will stay flexible and keep communication lines open with all stakeholders.

4) As I am sure many of you who have interacted with us can tell, our small tech team spends almost all its limited time fighting fires to simply keep CiviCRM online. To address this, we are going to expand our technical team to enable continuous development. We are hiring additional technical staff with expertise in administering and improving large systems will allow us to focus on continuous development, so that new features enabling better fundraising and more effective political activism for both the national party and our state affiliates are delivered on a consistent and predictable basis as well as continuing to expand and harness volunteer talent. This steady flow of new features being defined by and evaluated against the needs of our users will allow us to continue to level up and scale our efforts instead of having to keep trying to squeeze more toothpaste out of the same tubes the party has been using for years.

Long Term

Implementing these steps will make huge strides in our staff's ability to leverage their talents and return the party to a healthy position. To fully capture the gains of these strides and avoid reverting to the mean, we must have a vision of what an excellent CRM would look like for our ecosystem. As I mentioned, the specific features and user experience will be determined as a whole movement via a broad requirements gathering process. I would like to lay out some core principles we should all keep in mind that, if successfully implemented, would take us from an organization that has historically struggled with technology to a leader in political tech:

- Simple user experience and easy to train users
- Designed specifically around leaders, organizers, and end users working to grow the party and make a political impact
- Private, secure, cancel-proof
- Open source
- Easily deployable and maintainable by states on their own infrastructure with maintained set-up documentation
- Accurate Reports available in the tool
- Documentation available in the tool
- Native two-way data flow between national and states that can be managed by either side
- API available to states to get contact/contribution/event data in/out of the system so they aren't forced to use our solution
- Highly performant (aka ***fast***)
- Clear geographic and functional Role Based Access that is maintained by states but used to determine national data access
    - This means we have a small number of roles that could assigned to someone without compromising the segregation of permissions.
- Easy to access development environment
- Robust Continuous Integration/Continuous Delivery process
- Automated patching
- FEC compliant
- Can take payment natively in cryptocurrency
- Active data sanitation

In closing, I would like to particularly thank John Fetsko, Jay Norton, and the members of the IS Committee. They have worked at all hours of the day to fix outages and navigate our technological landscape. I also would like to thank Lainie Huston, Drew Hreha, Matt Hudson, and David Aitken on staff. They have poured endless dedication and creativity into running as effective of fundraisers as is currently possible within our constraints. I cannot wait to see what they accomplish with these new tools and improvements. I hope this breakdown has given you a thorough understanding of Skvare's technical recommendations and our plan to address technical needs. I am excited to work with you all as we make our CRM platform the best it can possibly be to help us achieve liberty outcomes.

Andy Buchkovich

Chief Technical Officer

CiviCRM & Infrastructure Assessment Report

Prepared for the Libertarian Party

May 17, 2023

Submitted by Skvare, LLC

Helping Others Help Others

Table of Contents

Goals of this Assessment

Skvare is providing the national Libertarian Party with this external third-party evaluation to meet two primary objectives:

1. Evaluation of the current system's multi-site infrastructure, including hosting configuration and code, in order to recommend and prioritize areas for improvement.
2. Evaluate the effectiveness of CiviCRM as a product for meeting the party's current and ongoing needs.

To meet these objectives, Skvare has provided the following services:

1. System performance: users continue to find that WordPress and CiviCRM are slow, with basic searches in CiviCRM requiring 5 to 10 seconds to complete. Skvare's DevOps team has reviewed server and infrastructure configuration and is sharing recommendations for improving performance.
2. Multi-site configuration: the WordPress+CiviCRM system uses a complex multi-site configuration to support separate affiliate websites and control access to the contact/donor database. Skvare has reviewed existing documentation and evaluated the current configuration. We have provided a summary of our findings below.
3. User access review: Skvare has reviewed roles and permissions in the WordPress/CiviCRM system. We are providing a summary of findings and a recommended outline for identifying, prioritizing, defining and implementing potential improvements
4. Documentation: The party reports that there is a lack of internal documentation. We have assessed the party's documentation needs and are providing recommendations for improvements.

Assessment Overview

Skvare's technical assessment of the platform is generally positive.

Our review of the server and infrastructure found no significant issues. We have provided a set of recommendations which may improve performance (see **Technical Assessment and Recommendations: Performance** section below). It is both good and bad news that we found no "silver bullet" solutions that we expect will dramatically improve speed and performance.

In this evaluation of the effectiveness of CiviCRM as a product for meeting the party's current and ongoing needs, our preliminary finding is that CiviCRM is indeed an effective solution. A more thorough internal evaluation process is outlined in the **Strategy** section of this report.

Our review of system configuration found a few idiosyncrasies and areas for improvement, detailed later in this report. We're recommending a process of cleanup and simplification, particularly around users and permissions. There are some notable improvements that can be made without changing existing functionality – that is, the system can be made to do what it does now, but quicker. Most potential improvements will need further analysis. Some would definitely change system behavior, which we anticipate would be weighed against the benefits of that change.

Planning and Strategy

During this engagement, we identified several potentially significant improvements in the organization's **planning and strategy** around its technical offerings. This section explains those findings.

Future Planning

As a software provider, the Libertarian Party should expect that development will be continuous and ongoing. *Our understanding is that the organization sees the "development phase" of its WordPress+CiviCRM system as nearing an end. We strongly recommend the organization affirm and normalize the expectation that, as long as it provides software to its affiliates, the development phase will* never *truly end.*

The Libertarian Party has asked Skvare to review some key areas for immediate improvements. (See **Goals of this Assessment** at the top of this report.) When these improvements are completed, others will immediately rise to the forefront. New bugs and bottlenecks will be discovered. Users will demand new features as their affiliates' business needs change.

In our work with similar organizations, we've consistently seen that *too little* focus on ongoing development leads to increases in other areas: more support and training issues, more administration and maintenance work, and just generally more staff time spent fielding complaints from dissatisfied users.

Skvare's recommendation: plan for and prioritize ongoing development throughout the life of the system.

We realize this may seem like a predictable thing for a web development company to say! This recommendation is entirely independent of any self-interest. We recommend increasing the party's technology staffing level by adding internal or external developers, employees or volunteers. For CiviCRM improvements in the short term, a development firm with CiviCRM experience is highly preferable.

Strategic Prioritization

Or, *Avoiding the "Be All Things to All People" Trap*

When a single organization adopts a piece of software, they decide what they need to use it for, and then adjust the software to meet those needs. The Libertarian Party's users are more diverse: each individual user has their own needs, expectations, and skills; and then each affiliate has its own organizational needs and priorities. **It's naturally challenging to make it easy for everyone to do everything.**

For the purposes of this assessment, the Libertarian Party has identified several priority improvement areas:

1. System speed and performance (includes optimization of user permissions)
2. User-friendliness for both internal (national party) and external (affiliate) users
3. User access, specifically the need to grant more permissions to affiliate users

Items 2 and 3 are somewhat at odds with each other. Let's consider:

Balancing Features & Friendliness

With technical systems, there's a spectrum between simple and powerful:

Simple

- Very user-friendly
- Requires minimal training to get started
- Few features and limited options outside a standard setup

Powerful

- Delivers complex functionality
- Highly configurable, with many options and combinations of options
- Requires training to manage effectively

Of course, it is possible to build systems that are both user-friendly and powerful. We've all seen websites that do a great job balancing the two. This is achieved with a high level of work across many technical areas: strategy and planning, user engagement, development (typically across multiple specialty areas), graphic design, UX (user experience) design, testing, and more.

To be frank: it is expensive.

Skvare's recommendation: rather than aiming to build an ultra-high-end technology platform, make a strategic decision to pursue *either simple or powerful*. Either approach will require commitment and involve tough decisions. Here's how each path might look:

Simple

- Remove all but the most essential features and the most widely used options.
- Remove as many options as possible from WordPress and CiviCRM menus.
- Replace complex WordPress / CiviCRM features, such as Advanced Search/Search

Powerful

- Grant more permissions to affiliate users. (For recommendations on doing this in an organized fashion, see **User Permissions** later in this document.)
- Prioritize training and detailed documentation for affiliate users. Require

Kit, with simplified custom-built versions. Engage outside expertise in UX/UI to design the new versions.
- De-prioritize documentation for affiliate users, focusing instead on intuitive usability for people who don't read the instructions.
- Conduct informal focus groups and monitored UAT: watch real users navigate the system to identify usability problems, then resolve them.

affiliate users to participate in comprehensive regular "refresher" trainings so they can continue to use the system correctly and efficiently.
- More users doing more things = higher error rates and less data accuracy. Frequently examine data to identify problems and trends in problems. Build administrator reports and back-end processes to detect and clean up bad data.
- To the greatest extent possible, adopt a hands-off approach to affiliate-generated errors and inaccurate data.

Summary of Technology Platform Priorities

Skvare's work on this assessment included eliciting the following set of technology priorities, in order from most to least important:

1. Streamline the administration side of our current solutions
2. Deliver improved functionality for national party users and national fundraising purposes
3. Deliver easier and more user-friendly solutions for affiliate end users
4. Improve the power and performance of tech solutions
5. Add new features and/or expand solutions for affiliates
6. Grow our user base (not a priority)
7. Pare down: offer fewer / reduced solutions (not a priority)

The Libertarian Party reports that the most critical element of the system is **data sharing** between national and affiliates.

Strategic Selection of Features

The Libertarian Party already has a technology platform, which already has a full set of features. Why are we talking about "*selection* of features"?

The resources required for ongoing maintenance, development, and support are much higher for a more complex system with a wider array of features. Every day that a feature is available on this platform is a day that the organization has decided, even passively, to continue to offer that feature.

Skvare's recommendation: the Libertarian Party's board of directors, or a subcommittee/task force it designates, should: perform a strategic review of the platform's features and the needs of its users, conduct a SWOT analysis for each feature, and affirmatively decide which features will continue to be part of the system.

Will this process take some time? Yes! But so does providing ongoing administration, development, training, and support. By going through this strategic planning process *now, the organization will ensure that its staff and resources are 100% focused on the most essential features with the highest ROI for both national and affiliates.*

For each feature, consider and answer questions such as:

- "What are our users' needs in this area?"
- "How well does this feature meet everyone's needs *now*?"
- "What other solutions exist which the national party and/or affiliates could use instead?"
- "How does our feature stack up against the alternatives?"
- "If we were to remove this feature from our system and adopt/recommend an alternative, what would be involved in the transition process?"
- "What are the short-term and long-term consequences of maintaining this feature vs discontinuing it?"

Special Considerations

It is not *necessarily* necessary for every feature to exist within a single unified platform. When considering features separately, keep an open mind regarding which pieces could potentially be split off from the whole.

It may be important to share data between, for example, a donor database and an event registration system. While keeping an open mind about separate features, also consider where data sharing is most important. Sharing data within a single system is easier than sharing it between two separate systems.

Strategic Feature Analysis: Part 1

Parts of this matrix are pre-filled with information provided to Skvare by the Libertarian Party. This analysis should be completed by the board or a board committee/task force. To complete: fill in all the blank boxes. Modify pre-filled content as needed.

Feature	**National Party IT Solution**	**Advantages of National Solution**	**Disadvantages**
CRM: contact and donor database, search, reporting	CiviCRM	Centralized data: donor and donation data shared between national and affiliates, and vice versa.	Affiliates have limited control over their own events, donation forms and other features. Requires significant maintenance, management, training and support from national staff.
CMS: chapter websites	Wordpress with multisite: supports separate websites for each chapter, managed through a single system		
Marketing: email, SMS	CiviCRM		
Phone banking, walk lists	CiviCRM		
Online donations	CiviCRM + payment processor		
Survey tool	CiviCRM		
Event management	CiviCRM + Wordpress with custom development		
Identity management	Wordpress	Out-of-the-box	Wordpress is not designed to perform this function. In current state, no unified identity management across platforms—only managing

			permissions and not cohesively in a one-stop-shop mechanism

Strategic Feature Analysis: Part 2

Parts of this matrix are pre-filled with information provided to Skvare by the Libertarian Party. This analysis should be completed by the board or a board committee/task force. To complete: this matrix, Part 2 of the analysis, will almost certainly require more space than this grid provides. You're encouraged to break out each feature into its own SWOT analysis document. The grid on this page is offered as a starting point.

Feature	**Major Alternatives**	**Libertarian Party Solution vs Alternatives: Pros & Cons**	**Conclusion** ***Continue or Discontinue as a National Party Offering?***
CRM: contact and donor database, search, reporting	• NationBuilder • Anedot		
CMS: chapter websites	• Independent Wordpress sites • Squarespace • Other major CMS solutions		
Marketing: email, SMS	• Anedot • Mailchimp • Constant Contact		
Phone banking, walk lists	• eCanvasser		
Online donations	• Stripe • Paypal • Donorbox • Anedot		
Survey tool	• Google Forms • JotForm • Mailchimp		
Event management	• EventBrite • Cvent		
Identity management			

Strategy and Planning: Summary

In this evaluation of the effectiveness of CiviCRM as a product for meeting the party's current and ongoing needs, our preliminary finding is that CiviCRM is indeed an effective solution.

Critical to this question, however, is the question "what are the party's current and ongoing needs?" The preceding Strategy sections of this report are provided as recommended processes for defining those needs and evaluating WordPress+CiviCRM as a long-term solution. The proposed approach focuses primarily on the affiliate side of the system, which is by far the more unique set of requirements compared with the national CRM and donor database.

Technical Recommendations: Server and Infrastructure

Moving on from the strategic side of this assessment, Skvare has conducted a review of the WordPress+CiviCRM system's server and infrastructure.

Apache Web Server

The following performance-related settings currently use default values. Our recommendation is to consider increasing these settings, which can yield performance improvements. Note that setting any of these too high may lead to resource exhaustion.

- MaxClients: controls maximum simultaneous connections that Apache will allow.
- Enable KeepAlive (done); adjust KeepAlive timeout and MaxKeepAliveRequests. Allows multiple requests to be sent over a single TCP connection; sets the maximum number of requests that can be sent over a single TCP connection when KeepAlive is enabled.
- Timeout: sets the amount of time that Apache will wait for a response from a client before closing the connection.
- ServerLimit: sets the maximum number of worker processes that Apache will create.
- ThreadLimit: sets the maximum number of worker threads per process.
- StartServers: sets the number of worker processes that Apache will start initially.
- MinSpareThreads and MaxSpareThreads: minimum and maximum number of idle worker threads that Apache will keep running.
- ListenBacklog: sets the maximum length of the queue of pending connections.

- Enable EnableMMAP and EnableSendfile: These variables control whether Apache will use memory-mapped files and the sendfile() system call for sending files, respectively. Enabling them can improve performance.

PHP

- Set opcache.enable to 1 to enable opcode caching for PHP files.
- Increase opcache.memory_consumption to 512MB to allow for more scripts to be cached in memory.
- Set opcache.max_accelerated_files to a higher value, such as 10000, to cache more PHP files.
- Increase memory_limit to a higher value, such as 8192M, to allow PHP scripts to use more memory.
- Set max_execution_time to a higher value, such as 600, to allow PHP scripts to run longer before timing out.
- Increase post_max_size to a higher value, such as 128M, to allow for larger file uploads.
- Increase upload_max_filesize to a higher value, such as 128M, to allow for larger file uploads.
- Set realpath_cache_size to a higher value, such as 4096K, to cache more file paths in memory for faster file access.
- Set realpath_cache_ttl to a higher value, such as 86400, to cache file paths in memory for longer periods of time.

MySQL

- innodb_buffer_pool_size: Consider increasing this value from its current setting of 96GB to 120GB or higher to allow MySQL to use more memory for caching data.
- innodb_log_file_size: This setting is already fairly large at 4GB, but increasing it could improve write performance. However, it may also increase recovery time in the event of a crash. Try 8GB or 16GB and monitor the impact on performance.
- innodb_flush_log_at_trx_commit: This setting determines how often MySQL writes transaction logs to disk. The default value of 1 provides maximum durability, but it can also impact write performance. Try setting it to 2 or 0 to see if it improves performance without sacrificing data integrity.
- tmp_table_size and max_heap_table_size: These settings control the maximum size of temporary tables used by MySQL. Increasing them could improve performance, but be careful not to set them too high or you could risk running out of memory.
- query_cache_size: Enabling the query cache might improve performance for read-heavy workloads. Currently set to 0, consider setting it to a higher value.

- join_buffer_size: Currently set to 4GB, this value can be reduced to 4M or 8M which is enough to handle most joins.
- Consider upgrading the version of MariaDB on the database server. The current version is 10.6.12, which has long-term support, but it will expire in three years. Upgrading to a newer version can improve performance.

Structure

- Database Server: Use a separate database server to host the MariaDB. This will help in separating the database load from the web servers and provide better performance.
- Database Replication: Set up database replication to create a secondary database server for redundancy and failover.
- Content Delivery Network (CDN): Use a CDN to cache and serve static assets like images, videos, and other files. This will reduce the load on the web servers and improve the application's performance.
- Cache: Use a caching mechanism like Memcached or Redis to cache frequently accessed data. This will help in reducing the load on the database server and improve the application's performance.
- Monitoring: Use a monitoring solution like Prometheus or Zabbix to monitor the infrastructure and application health. This will help in identifying and resolving issues before they impact the users.
- Load Balancer: Use a load balancer to distribute traffic across multiple web servers. This will help in distributing the load and provide high availability.

Technical Recommendations: General

Skvare is providing the following set of tactical recommendations, based on our review of the WordPress+CiviCRM system and discussions with the Libertarian Party team. These recommendations can be implemented in the short term, some of them immediately.

Internal CiviCRM search and reporting

Currently, national party and affiliate users are using CiviCRM’s Search Kit and Search Builder features for standard search and reporting. Skvare’s recommendation is that these search features be hidden and/or discouraged for most users. Instead, direct them to Basic Search and Advanced Search by modifying the CiviCRM navigation menu. *This menu has already been modified to place Basic and Advanced searches under the “Legacy Search” heading. We recommend reverting that change.*

From CiviCRM's Search Kit documentation:
SearchKit is a powerful search query builder with extensive options for displaying results. **It is appropriate for site builders, power users and developers but non-technical staff are likely to need training or to spend some time learning the interface.** Alternatively a more experienced user or developer may create SearchKit searches and make them available to other users via the menu system or as a dashlet on the main CiviCRM screen.

When to implement: right now, unless it's preferable to announce this change to affiliate users in advance. In that case we recommend you schedule this change for a date of your choice and make the announcement now.

End User Support and Documentation

Although this assessment did not include a review of the external help and support site, we did observe that it is a separate site which requires its own separate login. Without knowing the reasons it was set up this way, we recommend considering a migration of that content to the WordPress site. There are a variety of options for configuring a subset of WordPress site content so it's visible only to authorized users.

Simplify and Streamline Technical Tools

Based on our system review, we think it is very likely that speed and performance issues are caused by the complexity of WordPress+CiviCRM configuration. In this section we outline a recommended approach for simplifying and streamlining.

Create a Reference Instance

We recommend **creating a "vanilla" WordPress + CiviCRM instance** for the technical team to use for diagnostic and development purposes. This would be in addition to dev instances which more closely match the production system.

A "vanilla" instance will support the creation of developer documentation (next step). It will also facilitate minor changes like the search menu changes recommended above (**Internal CiviCRM Search and Reporting**) by enabling you to easily see what CiviCRM's unmodified search menu looks like.

At Skvare we maintain several such instances for internal use. When investigating an issue, one common question that arises is "is this [system behavior] normal?" It's valuable to check against a "clean" install and immediately know whether you're troubleshooting something specific to your site vs a standard behavior.

Create Developer Documentation

This recommendation will come as no surprise to the Libertarian Party's technical team.

We recommend documenting the site's WordPress plugins, CiviCRM extensions, and custom code. A simple list of plugins and extensions will not be very useful; it's far more important to document what each one does and where in the system this behavior can be observed.

Fortunately, the site's custom code is not very extensive at all, and quite limited in its functionality.

Remove Unnecessary Add-Ons

Some WordPress plugins and CiviCRM extensions can have significant impacts on performance. The Libertarian Party's system has a notably high number of both.

Recommendation: uninstall plugins and extensions which may be negatively affecting performance.

Clearly this process will require caution! Once the purpose and behavior of each plugin and extension is documented (**Developer Documentation**, above) you can begin to analyze whether that behavior is needed. Before removing a plugin or extension, verify that it is not required by another plugin or extension, or by custom code.

Some extensions which are likely contributing to performance issues:

- Activity Type ACL
- Access Control by Financial Type for Reports: causes significant performance degradation on some large sites.
- Custom CiviCRM Permissions (see **User Permissions** section for recommendations on this extension)

Some extensions are improving performance and should be left installed and enabled, such as:

- INNODB triggers
- CSS/JS aggregator

Convert CiviRules to Custom Code

We have never seen so many CiviRules in one site. Many of them control adding contacts to CiviCRM Groups: state-specific groups, email lists, etc. It looks like hundreds of rules may be firing in response to simple actions such as adding or editing a contact.

The reason contacts need to be added to these groups is clear. CiviCRM's multisite and ACL-based permissions – the permissions which control which affiliate users can see which contacts – rely on CiviCRM groups. They do not work with Smart Groups; the groups must be Basic. Therefore, some process is necessary to add contacts to, say, the Alaska state party group if that contact is in Alaska.

An experienced CiviCRM developer should move this functionality to a custom CiviCRM extension.

- Instead of hundreds of separate rules firing at all times, contact add/edit will trigger one process which handles all the logic of multiple states and any other group membership required.
- For enhanced accuracy, an additional cleanup process may be developed as a Scheduled Job which would run overnight. It would check for group changes that should have happened but failed for any reason.

Additionally, it looks like the current collection of CiviRules do not fully handle group changes or removals: they add a contact to a group when an address is added, but appear not to include a mechanism to remove a contact from a group when its address is deleted or changed. (Due to the sheer quantity of rules, it is quite possible this behavior does exist and we just didn't spot it.)

Optimize Smart Groups and ACLs

See the full set of recommendations here:
https://docs.civicrm.org/sysadmin/en/latest/setup/optimizations/

Basically all of these recommendations are relevant for the Libertarian Party. They have been implemented at some point in the past, as evidenced by these (good, and appropriate for this system) settings in civicrm.settings.php:

```
$civicrm_setting['CiviCRM Preferences']['smart_group_cache_refresh_mode'] =
'deterministic';
$civicrm_setting['CiviCRM Preferences']['acl_cache_refresh_mode'] =
'deterministic';
```

Most of the settings described in the above-linked documentation have different effects depending on the site and its activity level, whether and how various CiviCRM features are used, and the values of other related settings. We recommend reviewing them again now.

If possible, decrease the frequency of the Flush Group Cache scheduled job in CiviCRM. It is currently set to (Always).

Cache

The current settings for the CiviCRM “Clean-up Temporary Data and Files” job are good: database cache is retained until manually cleared.

We recommend enabling Memcache or Redis to improve performance.

User Permissions

Granting additional user permissions, which the party has reported is one of its short-term goals, is fundamentally a business decision. The risk is that increased permissions will require additional training and support, and introduce a greater risk of incorrect setup or inaccurate data.

From the technical side, the current challenge in changing permissions is that they are managed in a lot of different places. This section summarizes where and how permissions are controlled, and provides recommendations for simplifying.

Conceptual Overview: User Permissions, Multisite and ACLs

- Multi-site / ACLs - based on CiviCRM Groups - controls who you can see*
- User roles & permissions - based on WordPress user roles - controls what you can do*

**There are exceptions. This is a generalization.*

Additional access rules and permissions are used to define contact summary layout, menu items, and other UI elements.

WordPress and CiviCRM Multisite

Multisite is configured correctly. WordPress and CiviCRM domains are based on domain groups. The relevant CiviCRM Groups are flagged as Reserved (good) and named "[STATE ABBREV] State."
A system of CiviRule actions adds contacts to the appropriate group. See **Convert CiviRules to Custom Code**, above, for an important recommendation on simplifying this process.

Because multisite CRM behavior requires constant checking of which users can see which contacts, this setup itself can be fairly resource intensive. See **Optimize Smart Groups and ACLs** above for recommendations on optimization. Multisite relies on ACL functionality, so the ACL recommendations apply.

Additional Permissions Features and Settings

WordPress User Roles via User Role Editor

In WordPress, a small set of non-standard user roles is created via the User Role Editor plugin. These roles are: Administrator, Anonymous, Contributor, Editor, State Admin, Subscriber, Volunteer. Given our understanding of the system, this seems entirely appropriate. Both WordPress and CiviCRM permissions are controlled via this plugin.

CiviCRM: WordPress Access Control

WordPress Access Control permissions are configured within CiviCRM > Users and Permissions. There is significant overlap between these settings and the ones controlled by the WordPress User Role Editor plugin (above), with the WordPress plugin evidently superseding these settings.

Recommendation: there is certainly no need for this comprehensive list of permissions to be managed in two places, but this one is native CiviCRM, and the WordPress version comes with the ability to have user roles such as "State Admin." Conduct a round of testing on a dev instance to confirm that the WordPress plugin's settings do in fact take precedence, and then make a note in the developer documentation to ignore this CiviCRM settings page going forward.

If testing on a dev instance reveals more complexity – such as the need for a permission to be turned on in both places in order to be active in practice – then decide which will be the authoritative source of permissions. Set all permissions in the other to all-on or all-off, as needed such that the authoritative version works as expected.

CiviCRM ACLs

CiviCRM ACLs are in use in addition to parallel multisite permissions. These appear to layer on top of multisite domains and the contacts accessible there, with ACL regions each covering multiple states.

Recommendation: Upon review, it appears there are very few users in the ACL user groups. Are these critical to maintain? Can this handful of regional users be safely granted wider system access instead? Removing this additional layer of visibility-type permissions may improve performance.

CiviCRM Groups

In the system CiviCRM Groups are used for (among other things): Domains, ACLs, and WordPress Group Sync.

Recommendation: review all CiviCRM groups flagged as "domain access"; delete unused ones.

WordPress Groups and Related Plugins

There are 28 WordPress Groups including Candidate Advanced, Candidate Basic, Super State Admin, Super Staff Admin. WordPress groups are synced with CiviCRM groups via the CiviCRM Groups Sync extension. (Adding a contact to a CiviCRM group grants a set of permissions. Effects appear in WordPress menus, CiviCRM contact summary screen, etc.)

The same extensive set of permissions which can be controlled via the WordPress User Role extension and CiviCRM's WordPress Access Control settings are also controlled here.

Recommendation: Prioritize this feature for removal. Perform a detailed review of the functionality it currently provides. Determine which functionality is necessary to maintain and which can be discontinued. Transition necessary functionality to other WordPress and CiviCRM features.

CiviCRM Extension: Contact Summary Editor

The Contact Summary Editor extension allows different CiviCRM users to see different versions of the contact summary screen (the main screen in CiviCRM when you're looking at a contact record). It relies on CiviCRM Groups.

This feature represents a deviation from the rule that *groups control who you can see; roles control what you can do*. Contact layouts have been modified based on user role (State Admin, Development, etc.) which is a logical use for this feature, but these are definitely user roles which in this case are defined as CiviCRM Groups.

Recommendation: review and simplify this configuration. There are a number of competing/inactive layouts saved in the system. For example, since the Development layout follows the Administrator layout (in the list which corresponds to their rank/priority), and Development is being shown to no users that *aren't* included in Administrator, the Development layout will never be shown.

By themselves these conflicts are not causing any problems; they're not doing anything at all. Removing unused and logically-deprecated layouts will help identify whether and how much this feature is actually being used. Ideally it can be simplified to a single layout for all users, removing the need for Groups involvement and any coordination with user roles.

CiviCRM Extension: Custom CiviCRM Permissions

The Custom CiviCRM Permissions extension has been used to create what are effectively *user roles* but have been added here as *permissions*. This is somewhat out of place. CiviCRM permissions are things like "view all contacts," "create contribution," or "delete activity," not things like "state admin."

Tentatively, it appears that these roles were defined, via this extension, as CiviCRM permissions. There are a lot of occurrences of
`if ( ! current_user_can( '[PERMISSION]' ) )`
in this site's custom code. It appears that these *roles* were defined as *permissions* specifically so this one method could be used to check the current user's role for particular operations.

Recommendation: **remove this**. Modify custom code to check permissions in a standard way, preferably by checking for a relevant CiviCRM permission rather than an effectively hardcoded user role. When the code updates have been completed, uninstall this extension.

CiviCRM Custom Extensions

There are a number of customs which specifically affect permissions within the system, such as most of the code in /civicrm_custom/CRM/

Suggestion: **clean up these custom extensions by reviewing them individually**. For each one: document its functionality, research and document the purpose, confirm the continued need (or not).

As a preliminary or short-term step, have an experienced CiviCRM developer review custom extensions for potential resource and performance issues.