Application Planning and Cost Models for Miocar

August 2024

National Center for Sustainable Transportation
Undergraduate Research Fellowship Report

Harjot Gill, Computer Science
Home University: UC Davis

National Center for Sustainable Transportation
A USDOT University Transportation Center

ITS UCDAVIS
Institute of Transportation Studies

About the National Center for Sustainable Transportation

The National Center for Sustainable Transportation is a consortium of leading universities committed to advancing an environmentally sustainable transportation system through cutting-edge research, direct policy engagement, and education of our future leaders. Consortium members include: the University of California, Davis; California State University, Long Beach; Georgia Institute of Technology; Texas Southern University; the University of California, Riverside; the University of Southern California; and the University of Vermont. More information can be found at: ncst.ucdavis.edu.

Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated in the interest of information exchange. The report is funded by a grant from the U.S. Department of Transportation's University Transportation Centers Program. However, the U.S. Government assumes no liability for the contents or use thereof.

The U.S. Department of Transportation requires that all University Transportation Center reports be published publicly.

Acknowledgments

This fellowship was funded by a grant from the National Center for Sustainable Transportation (NCST), supported by the U.S. Department of Transportation (USDOT) through the University Transportation Centers program. The author would like to extend their sincere gratitude to the National Center for Sustainable Transportation (NCST) and the U.S. Department of Transportation (USDOT) for their invaluable support of university-based research in the field of transportation. We deeply appreciate the Institute of Transportation Studies at UC Davis (ITS-Davis) academic and research faculty for allowing students to engage in funded research in areas aligned with their passions. A special note of thanks is extended to Anna Espitallier for facilitating this remarkable opportunity. I am particularly grateful to Dr. Christopher Nitta for his unwavering guidance and mentorship throughout this fellowship. Additionally, I would like to thank Dahlia Garas for her insightful guidance and support on this project. Lastly, I would like to thank the staff at Miocar for their promptness and professionalism, which made our collaboration both seamless and productive.

NCST

TABLE OF CONTENTS

Introduction

Miocar is a non-profit organization in California, that offers an electric vehicle (EV) car-sharing service with a mission to provide affordable and eco-friendly transportation. Miocar operates in regions including Richmond, Stockton, Tracy, Escalon and Tulare/ County, offering an accessible transportation solution to residents. Their service is open to anyone over 21 with a valid U.S. driver's license, including AB 60 license holders, ensuring inclusivity.

As the user base of Miocar expands, the organization has encountered challenges related to manual processes and operational inefficiencies. Recognizing the need for a more scalable and efficient solution, Miocar approached us to explore potential alternatives to their current system. Rather than focusing solely on developing a new application, this project will research and analyze various alternatives to streamline their operations. By evaluating the costs and benefits of different technological solutions, we aim to provide Miocar with a comprehensive understanding of how they can optimize their backend processes, reduce manual labor, and cost-effectively support their growth.

Overview

The primary objective of this project is to develop a solution for the troubles of their current application. A new app solution would have to allow users to create accounts, make reservations, and interact with their reserved vehicles through an intuitive and user-friendly interface. In addition to improving the customer-facing side, the project will address inefficiencies in the backend system used by Miocar staff, reducing manual labor and improving overall operational efficiency.

This report outlines the use cases, requirements, and cost considerations involved in the development and implementation of the app, as well as potential solutions that will optimize both user experience and backend operations. By modernizing Miocar's technological infrastructure, the project will contribute to the organization's mission of providing sustainable, accessible transportation to the residents of California

Existing Literature

In this project, the existing literature is limited due to the unique nature of Miocar's operations and the specific requirements of the organization. Unlike traditional research projects, which often rely on extensive published studies, our work is centered around the practical and dynamic needs of a growing organization.

While formal literature may be scarce, we can create our own body of knowledge by reverse-engineering the existing Miocar application. By analyzing the current system's architecture, workflows, and pain points, we can identify opportunities for optimization. This approach allows us to tailor our research and solutions to the specific context of Miocar, ensuring that any proposed changes align closely with their operational needs.

An important piece of existing literature that proved valuable in the later stages of our research was the Tampermonkey documentation. Tampermonkey is a popular browser extension that allows users to create and run custom scripts on web pages, effectively modifying their functionality and appearance. This aligns closely with our project goals, as we were exploring ways to enhance and customize the current application used by Miocar.

The Tampermonkey documentation provided insights into how we could potentially implement a web extension solution for Miocar's needs. This approach offers several advantages, including customization, cost-effectiveness, rapid deployment, and user familiarity. It allows for targeted modifications to the existing web application without requiring a complete system overhaul, which can be more economical than building an entirely new application from scratch. Changes can be implemented and tested quickly, allowing for iterative improvements, while staff can continue using a familiar interface while benefiting from new features and optimizations.

This literature helped inform our exploration of creating a web extension for the current backend app setup, as mentioned earlier in the report. By leveraging Tampermonkey's capabilities, we could potentially add new functionalities, modify existing features, and address Miocar's specific needs without disrupting their entire operational workflow. Additionally, we conducted extensive research into AWS services to consider a new application as a solution. We specifically looked into AWS Lambda for serverless computing, Amazon S3 for scalable storage, and Amazon RDS for managed database services. These cloud services could potentially exceed the extension solution by providing a robust, scalable infrastructure to support the needs of Miocar on their own terms.

Our research into AWS services aligns well with our project objectives of improving both the customer-facing side and replacing the backend system used by Miocar staff. We could aim to create a solution that not only reduces manual labor and improves operational efficiency but also provides a solid foundation for future growth and expansion of Miocar's digital infrastructure.

Research Methods

To carry out our methods, we needed to understand The overall needs and requirements of Miocar. To conduct a comprehensive analysis, a multi-faceted research approach was employed, focusing on both backend and frontend improvements, as well as the development of a cost model. The research began with a thorough evaluation of Miocar's current system, specifically their backend API service used for vehicle hardware. This involved reviewing existing documentation to understand the current system's interactions with vehicle components and identifying potential areas for enhancement. Reverse engineering of the current application was also carried out to map out its functionalities, and limitations, which included a detailed examination of API endpoints and their integration with Miocar's operational processes.

Following this, attention was directed towards identifying improvement areas for both backend and frontend systems. For the backend, the focus was on analyzing Miocar's existing app to find opportunities for reducing manual labor through automation. This included examining workflows related to booking management, member status maintenance, and vehicle service scheduling. While the primary focus was on backend improvements, preliminary considerations for enhancing the front end were also made, aiming to ensure that the user experience for vehicle reservations, interactions, and support features would be intuitive and effectively aligned with backend enhancements.

Cost Model and Services

A crucial component of the research was the creation of a cost model to compare the financial implications of developing a new application versus maintaining the current system. This analysis included calculating the costs associated with software development, implementation, and potential savings from improved efficiencies. Various alternative solutions were explored, such as developing an extension versus a complete system overhaul. The cost model assessed the potential benefits and drawbacks of each solution, considering factors such as labor costs, service fees, and impacts on existing operations.

Additionally, the research included an investigation into hosting services, with a primary focus on Amazon Web Services (AWS) for hosting the application. This analysis assessed AWS's capabilities in terms of scalability, reliability, and cost-effectiveness, ensuring that the chosen hosting solution would meet the application's performance requirements while staying within budget.

Data Collection

Data collection for this analysis was primarily sourced from Miocar. The data provided included crucial statistics such as the costs associated with their current software provider, monthly operational costs, and the number of stations, active users, and vehicles in their fleet. These figures served as the foundation for evaluating the financial implications of various solutions. By analyzing these statistics, we aimed to develop a cost model that compares the existing application with potential new solutions and alternative approaches.

Link to the extensive data model

Labor actions	Hours/vehicle per month	Hours/member per month	Hours/bookling per month
Member lookup		0.03	
Member status maintenance (Banned?, Suspended?)		0.07	
Weekly vehicle service bookings (cleaning, maintenance)	0.07		
Manual Booking data entry			0.04
License data entry and MVR check		0.70	
Weekly vehicle cleaning	1.70		

Figure 1: Time consumption of the manual Processes by Miocar staff

App Users/Vehicle count information	**As of March 2024**
Number of active members	126.00
Members per vehicle in the fleet	2.50
Number of vehicles needed (Est)	50.00
Booking data	
Total miles driven	23,837.00
Total reservations made	259.00
Miles per reservation	92.03
Monthly Vehicle mileage increment (miles)	476.74
Members per reservation ratio	2.06
Reservation count	259.00

Figure 2: Booking Data and user information for March 2024 and bonus analysis features

Vehicle Maintenance unit costs	As of March 2024
Average Cost of Tires	550.00
Average Tire life (miles)	40,000.00
Average Cost of Brake Maintenance (miles)	450.00
Average Brake life for EVs (miles)	100,000.00
Average Cost of Cleaning Supplies (bulk)	20.00
Average Cost of Electricity per mile	0.05
New Application Costs (Assuming same backend and database)	
Compute Cost (AWS)	7.59
Storage (AWS)	0.00
Database (AWS or existing)	0.00
Elastic Load Balancer (AWS, If needed)	0.00
Security (Free)	0.00
Total	7.59

Figure 3: Unit cost of consumables and a typical AWS hosting service

Hardware/Vehicle Costs	As of March 2024
Invers HW provider	250.00
Tires	327.76
Brakes	107.27
Cleaning supplies	40.00
Energy	1,191.85
Zev Coop service monthly cost	35.00

Figure 4: Hardware and consumable calculated cost

Labor Costs	As of March 2024
Average Labor Wage	28.00
Member lookup cost	105.84
Member status maintenance (Banned?, Suspended?)	235.20
Weekly vehicle service bookings (cleaning, maintenance)	483.47
Manual Booking data entry	290.08
License data entry and MVR check (Assuming constant growth of 5%)	370.44
Weekly Vehicle Cleaning	2,380.00

Figure 5: Cost of labor computed

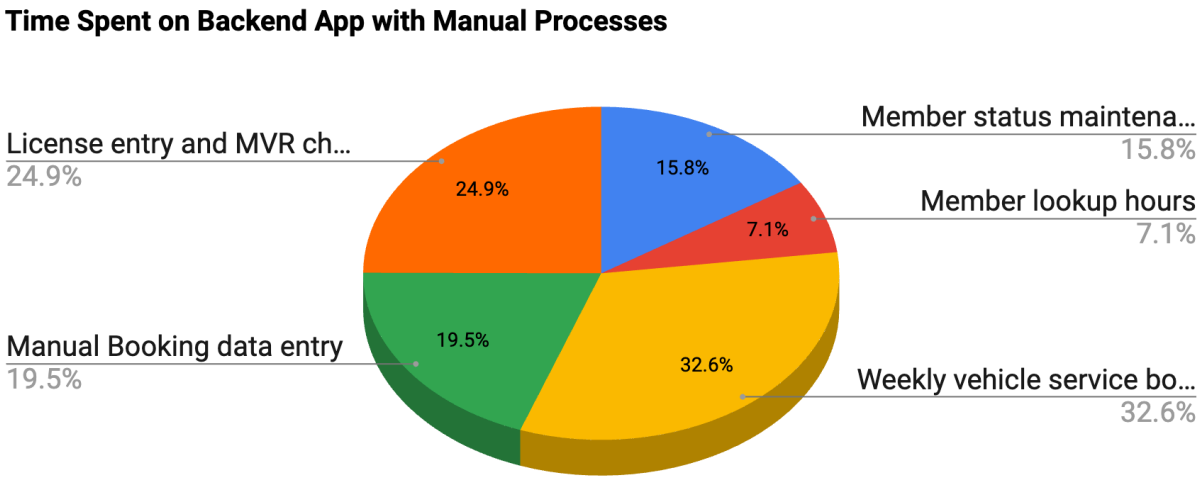


Figure 6: Breakdown of TIme cost for manual processes

Application/Extension implementation projected impact	As of March 2024
Automated work hours savings	43.99
Estimated Savings	1,231.60

Figure 7: Total Savings of an Application/Extension Solution

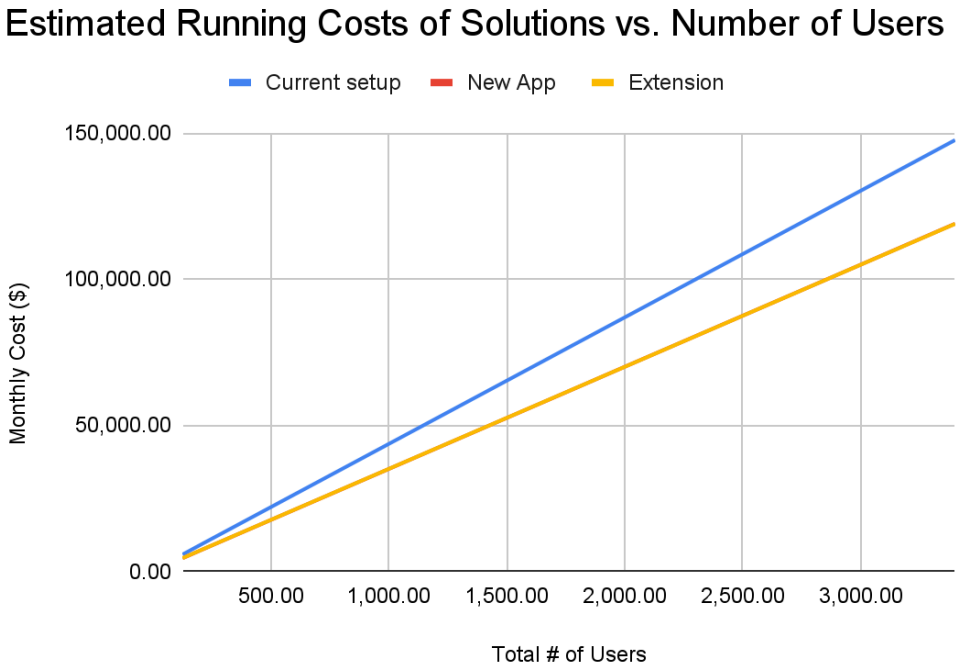


Figure 8: Cost projection of solutions as Users scale

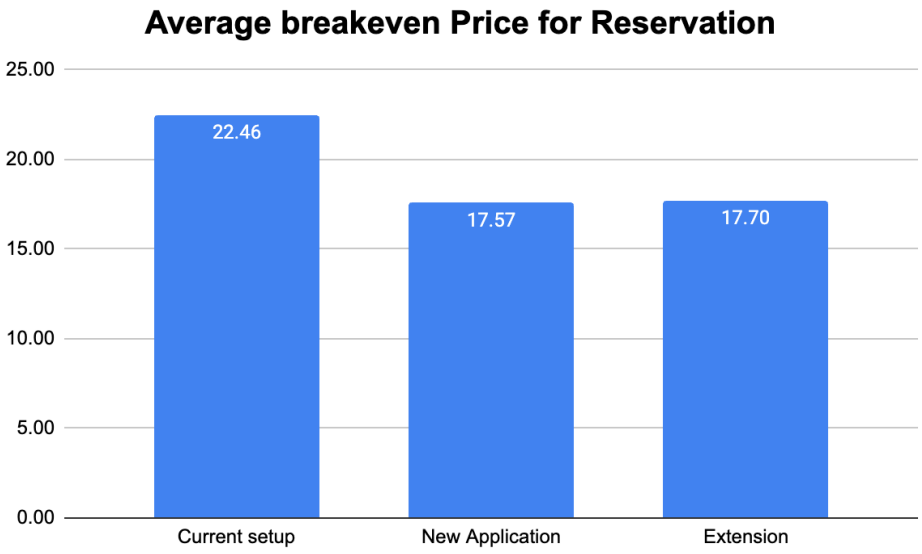


Figure 9: Average price needed per booking for a breakeven

Results

Summary

The analysis of Miocar's operational costs and potential improvements revealed several significant findings. Currently, the total monthly cost for Miocar's existing setup is approximately $5,816.90, which includes labor, hardware, and software maintenance expenses. By transitioning to a new application, these costs could be reduced to around $4,550.30 per month, offering savings through the automation of manual processes, a reduction in labor hours, and overall operational efficiency gains. Alternatively, developing an extension to the current system could lower the monthly costs to $4,585.30. While this option presents slightly higher costs compared to a complete new application, it provides a more gradual improvement with less upfront investment. Additionally, the breakeven price per reservation would decrease from $22.46 in the current setup to $17.57 with the new application and $17.70 with the extension. Similarly, the cost per user would drop from $46.17 in the current setup to $36.11 with the new application and $36.39 with the extension, indicating significant potential savings and increased sustainability as Miocar's user base continues to grow.

Discussion

The findings from this analysis underscore the potential benefits of investing in a new application or extension to improve Miocar's operational efficiency. The current setup, while functional, incurs significant costs associated with manual labor and outdated processes. By transitioning to a more automated system, either through a new application or a system extension, Miocar could significantly reduce its operational costs.

The new application offers the greatest potential savings, particularly by reducing labor hours associated with repetitive tasks like member lookup, vehicle service bookings, and manual data entry. Additionally, the breakeven price per reservation and cost per user would be significantly lower, making the service more sustainable as the user base grows.

The extension, while not as cost-effective as the new application, still presents a viable option for improving efficiency with a lower initial investment. It offers a middle ground between maintaining the current setup and a full-scale system overhaul.

Overall, the analysis suggests that Miocar should consider implementing a new application to achieve the greatest long-term savings and efficiency gains. However, if budget and time constraints are a concern, an extension could provide meaningful improvements at a lower cost at a much quicker rate. In either case, the move toward automation and modernization is likely to yield significant benefits for Miocar's operations and its members.

Conclusion

Overview

Our comprehensive analysis has shown that Miocar's current system, while functional, faces challenges as the organization continues to expand its user base and fleet. Through our research, we explored several options, including creating a completely new application, making incremental improvements to the existing system, and implementing a browser extension.

Our findings suggest that a browser extension offers a balanced solution, effectively addressing the current pain points without the extensive costs and time commitment associated with developing a new system from the ground up. The analysis shows that while both the new application and the extension offer significant savings compared to the existing setup, the extension requires far less upfront investment and development effort. By modifying and adding a few key features, Miocar can achieve substantial improvements in the few areas that are needed, increasing efficiency and cost savings without the need to build an entirely new system from scratch. This approach allows for a smoother transition, minimizing disruption while still addressing the organization's growing needs and operational challenges. As Miocar continues to expand, the extension offers a scalable solution that balances both financial prudence and technological advancement.

The next steps in our research and development process are crucial in determining the most effective solution for Miocar. As it stands, implementing a browser extension is emerging as the most viable option. This approach addresses the immediate needs of the organization by enhancing backend operations with minimal disruption, offering a cost-effective solution that avoids the complexities of developing a new system from scratch.

Summary of Options

Firstly, the new application development option would involve designing an API to streamline interactions between the frontend and backend systems. This API would abstract complex transactions and simplify common requests. If this path is chosen, careful planning for data migration and transition would be necessary to ensure seamless integration with existing systems.

On the other hand, the extension development option focuses on leveraging the Tampermonkey framework, which is a popular extension that allows for web page manipulation and solving the issues at hand. By mastering this tool, we can implement the required changes incrementally, allowing for gradual improvements to the website’s functionality and user experience.

Suggested route

Given the current analysis, the extension approach is prioritized due to its cost-effectiveness and minimal impact on existing operations. The next steps involve finalizing the extension development within the Tampermonkey framework to ensure it meets all functional requirements. Additionally, while the backend optimizations are critical, we will also direct our efforts towards enhancing the frontend user experience, making the system more intuitive and accessible for users. Lastly, although the extension is the immediate focus, we will continue to research API design and application development to prepare for future scaling needs. By following this plan, we aim to provide a solution that not only meets Miocar's current needs but also positions them for future growth and technological advancement.

Post-Research Solution Overview

A New Application as a Solution

This comprehensive analysis explores the infrastructure requirements and considerations for hosting a new application, with a focus on utilizing Amazon Web Services (AWS) as the cloud platform (Amazon Web Services). The breakdown covers essential components needed for a robust and scalable application deployment.

Compute resources form the backbone of the application, running the core business logic and processing user inputs. AWS offers various options, including Amazon EC2 for scalable virtual servers, AWS Elastic Beanstalk for simplified deployment and management, and AWS Lambda for serverless computing. These services can be tailored to handle varying workloads and can be scaled as the application grows.

Storage is another crucial aspect, with options like Amazon S3 for general storage of web applications and mobile apps, and Amazon EBS for block storage volumes (Amazon Web Services). The analysis suggests a starting point of 20 GB for general storage, with the potential to expand to 100 GB or more as needed.

For database management, there remains the possibility of reusing the existing database provided by ZEV Coop. However, AWS offers robust database solutions such as Amazon RDS for relational databases, DynamoDB for NoSQL databases, and Amazon Aurora for high-performance relational databases. These options provide flexibility in data management and scalability for reasonable costs.

Networking and content delivery are addressed through services like Amazon CloudFront for content delivery, Route 53 for DNS management, and Elastic Load Balancing for distributing incoming traffic. These services ensure efficient data transfer and improved application performance.

Security is a concern, and AWS Identity and Access Management (IAM) is highlighted as a free service for controlling access to AWS resources. Additional security measures like AWS Shield for DDoS protection and AWS WAF for web application firewall protection are also mentioned as potential inclusions in a more advanced setup.

Storage costs are estimated at $11.50 per month for 100 GB. The total cost for a basic to complex application infrastructure on AWS is projected to be around $30-$50 per month. In terms of user capacity, an API-driven mobile or web application could potentially handle 500-3000 users per month, with active concurrent users likely limited to 50-200, depending on API usage patterns. For monitoring and management, services like Amazon CloudWatch, AWS CloudTrail, and AWS X-Ray are recommended to ensure optimal performance, security, and compliance.

While this overview primarily focuses on AWS services, it's worth noting that other cloud providers like Google Cloud Platform and Microsoft Azure offer similar services and capabilities. The choice of cloud provider should be based on specific application requirements, budget constraints, and long-term scalability needs, in which AWS seems to excel (Kinsta). The flexibility of AWS services allows for starting with a basic package and scaling up to more advanced configurations as the application grows and user demands increase (Barr).

The TamperMonkey Framework

TamperMonkey is a powerful browser extension that enables users to run custom JavaScript code on web pages, offering extensive customization and automation capabilities. At its core, TamperMonkey works by injecting user-defined scripts into web pages as they load (Tampermonkey for Beginners, 2024). These scripts can be set to execute at different times during the page load process, allowing for precise control over when modifications take place.

From a technical standpoint, TamperMonkey provides a wide array of features and functionalities. It allows for comprehensive DOM manipulation, enabling scripts to add, remove, or modify HTML elements on a page (Tampermonkey, 2024). The extension also grants access to its own API, which facilitates cross-origin requests, data storage, and browser interaction. Scripts can attach event listeners to elements, intercept and modify AJAX requests, inject custom CSS styles, and even load external libraries or resources to expand their capabilities.

Advanced capabilities of TamperMonkey include the ability to bypass same-origin policy restrictions for cross-origin requests, persistent data storage across sessions, display of notifications, the addition of custom menu items to the TamperMonkey context menu, and script synchronization across devices using cloud storage services. These features make TamperMonkey a versatile tool for a variety of applications, including web scraping, UI enhancements, **task automation**, custom ad-blocking solutions, API integration, and accessibility improvements (Tampermonkey/tampermonkey, 2024).

When using TamperMonkey, it's crucial to develop scripts responsibly to avoid security risks. Powerful scripts can impact page load times and browser performance, and website updates may break them, requiring maintenance. Some browsers may also limit TamperMonkey functionalities for security reasons.

Solution Integration

Since we adopted Tampermonkey as our solution, we've made significant strides in addressing several previously discussed issues. Our solutions aim to display the webpage of the system's functionality and efficiency, particularly in member management and booking processes. Solutions and Working scripts can be accessed here (Gill, 2024).

Signup Start and End Filtering

One of the primary improvements we implemented is the member lookup feature based on sign-up date as seen in Figure 10 below. This solution involved creating a sophisticated date range picker that seamlessly integrates with the existing member list. The Tampermonkey script injects this functionality into the member management page, allowing administrators to quickly filter and find members based on when they joined. Note that this is currently a cosmetic solution. To make this feature fully functional, we'd need to delve deeper into the website's technology and backend. This would involve observing network activity and API calls to the ZEV Coop servers to send queries and receive results, which we'd then display. This feature would be particularly useful for tracking new member acquisitions, analyzing sign-up trends, and targeting specific cohorts for communications or promotions. It primarily solves a problem that Miocar has faced in searching for members who signed up on a particular date.

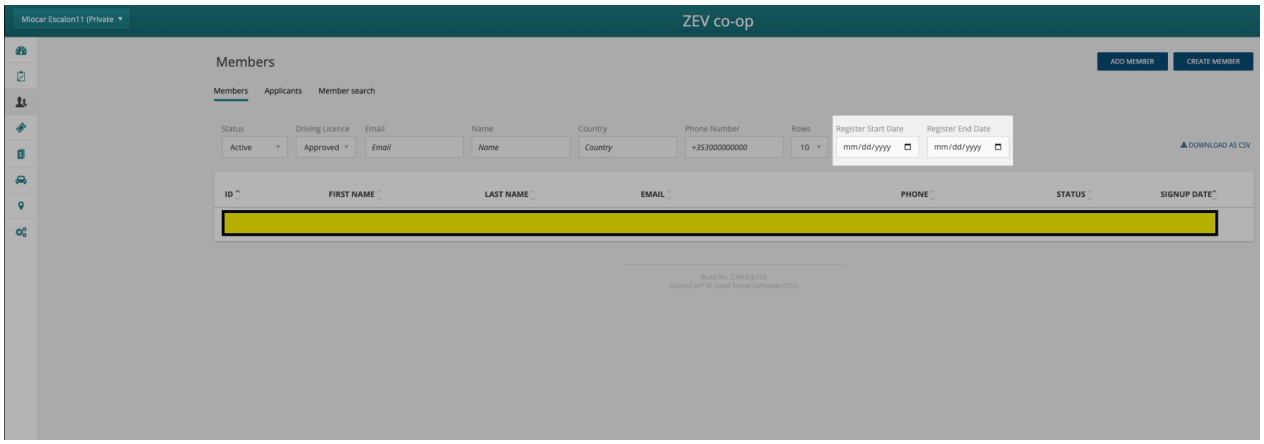
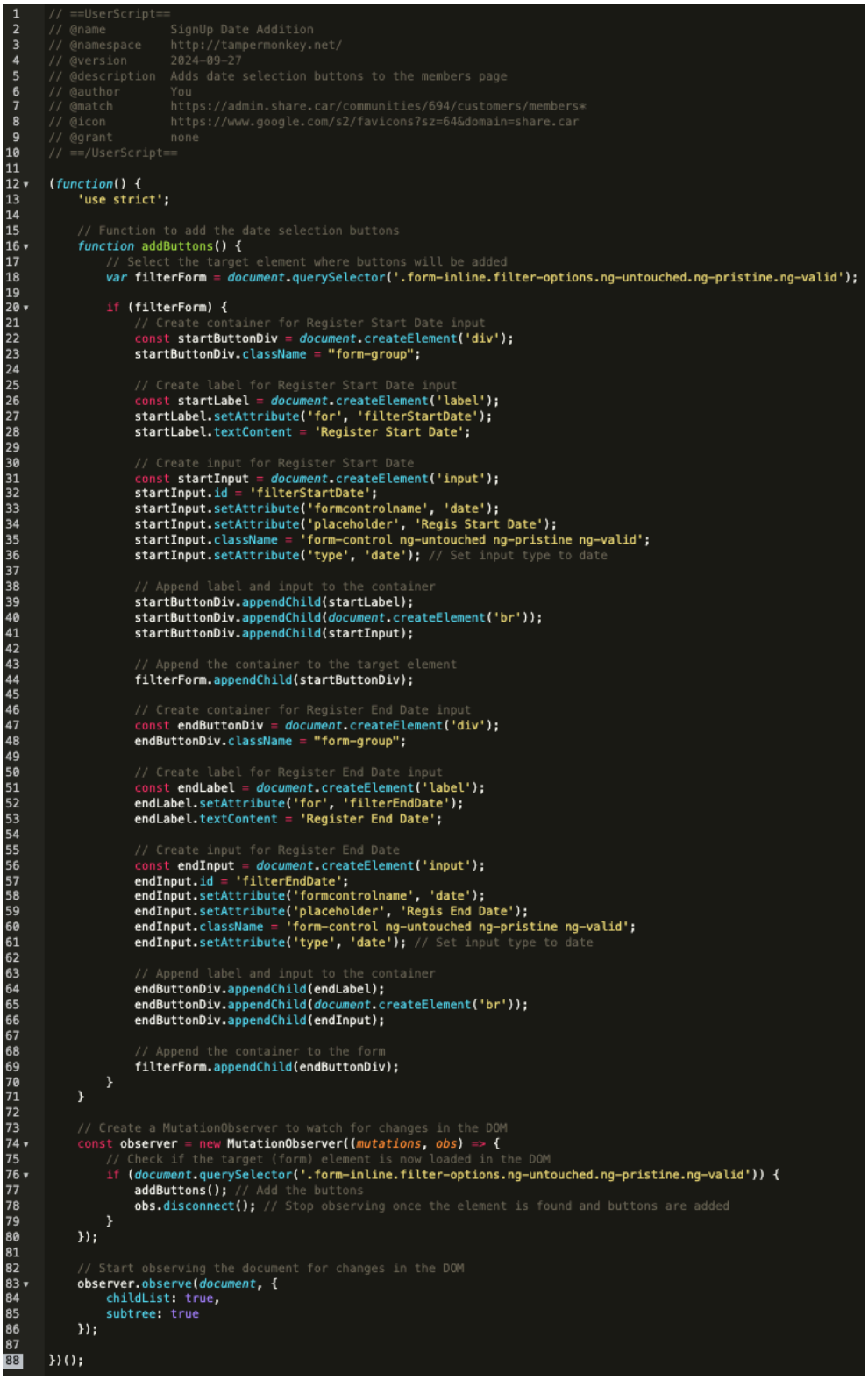


Figure 10: Changes made visually to the Backend Website.



```
// ==UserScript==
// @name         SignUp Date Addition
// @namespace    http://tampermonkey.net/
// @version      2024-09-27
// @description  Adds date selection buttons to the members page
// @author       You
// @match        https://admin.share.car/communities/694/customers/members*
// @icon         https://www.google.com/s2/favicons?sz=64&domain=share.car
// @grant        none
// ==/UserScript==

(function() {
    'use strict';

    // Function to add the date selection buttons
    function addButtons() {
        // Select the target element where buttons will be added
        var filterForm = document.querySelector('.form-inline.filter-options.ng-untouched.ng-pristine.ng-valid');

        if (filterForm) {
            // Create container for Register Start Date input
            const startButtonDiv = document.createElement('div');
            startButtonDiv.className = "form-group";

            // Create label for Register Start Date input
            const startLabel = document.createElement('label');
            startLabel.setAttribute('for', 'filterStartDate');
            startLabel.textContent = 'Register Start Date';

            // Create input for Register Start Date
            const startInput = document.createElement('input');
            startInput.id = 'filterStartDate';
            startInput.setAttribute('formcontrolname', 'date');
            startInput.setAttribute('placeholder', 'Regis Start Date');
            startInput.className = 'form-control ng-untouched ng-pristine ng-valid';
            startInput.setAttribute('type', 'date'); // Set input type to date

            // Append label and input to the container
            startButtonDiv.appendChild(startLabel);
            startButtonDiv.appendChild(document.createElement('br'));
            startButtonDiv.appendChild(startInput);

            // Append the container to the target element
            filterForm.appendChild(startButtonDiv);

            // Create container for Register End Date input
            const endButtonDiv = document.createElement('div');
            endButtonDiv.className = "form-group";

            // Create label for Register End Date input
            const endLabel = document.createElement('label');
            endLabel.setAttribute('for', 'filterEndDate');
            endLabel.textContent = 'Register End Date';

            // Create input for Register End Date
            const endInput = document.createElement('input');
            endInput.id = 'filterEndDate';
            endInput.setAttribute('formcontrolname', 'date');
            endInput.setAttribute('placeholder', 'Regis End Date');
            endInput.className = 'form-control ng-untouched ng-pristine ng-valid';
            endInput.setAttribute('type', 'date'); // Set input type to date

            // Append label and input to the container
            endButtonDiv.appendChild(endLabel);
            endButtonDiv.appendChild(document.createElement('br'));
            endButtonDiv.appendChild(endInput);

            // Append the container to the form
            filterForm.appendChild(endButtonDiv);
        }
    }

    // Create a MutationObserver to watch for changes in the DOM
    const observer = new MutationObserver((mutations, obs) => {
        // Check if the target (form) element is now loaded in the DOM
        if (document.querySelector('.form-inline.filter-options.ng-untouched.ng-pristine.ng-valid')) {
            addButtons(); // Add the buttons
            obs.disconnect(); // Stop observing once the element is found and buttons are added
        }
    });

    // Start observing the document for changes in the DOM
    observer.observe(document, {
        childList: true,
        subtree: true
    });

})();
```

Figure 11: Sample Code for SignUp Start and End Date query addition. Also available here

Member Signup Date Display

To further enhance member information visibility and organization, we added a new column to the member table that displays the sign-up date for each member. This addition provides immediate access to crucial temporal data about each member. Complementing this, we implemented a placeholder for a sorting functionality that enables administrators to order the list by sign-up date in both ascending and descending order. This sorting capability significantly improves the ability to track and manage new members, identify long-standing members, and analyze membership patterns over time. The combination of the new column and sorting feature offers a more intuitive and efficient way to navigate and understand the member database. It's important to note that this solution is also cosmetic only. It addresses Miocar's challenge of sorting members by sign-up date, eliminating the need to navigate to the end of the list to find the latest signups which was a tedious problem they faced. While this visual enhancement integrates with the theme of the website, implementing full functionality would require deeper integration with the website's backend and API.

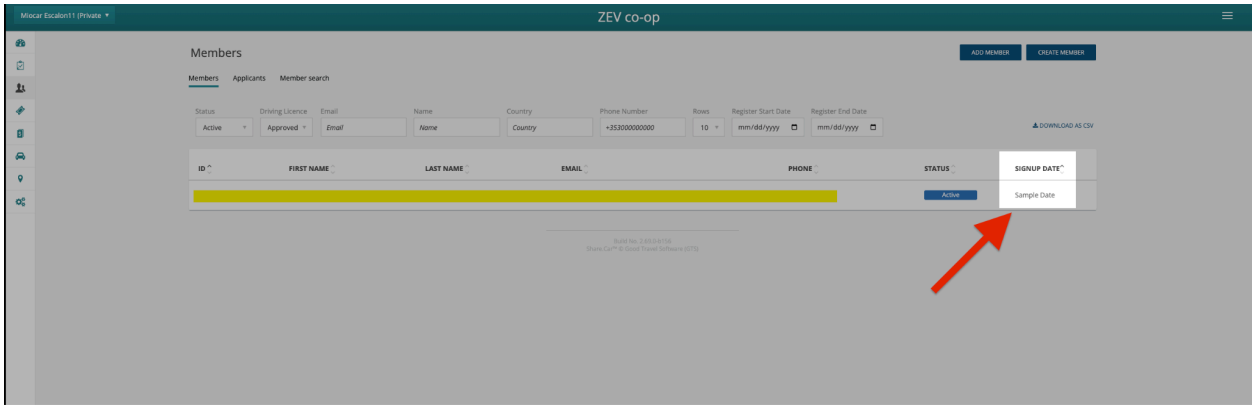


Figure 12: Changes made visually to the Backend Website to include Signup Data.

Booking Data Export Functionality

Addressing the challenge of manual data entry, we are exploring options for a comprehensive booking data export feature. This solution aims to automate the previously time-consuming process of transferring booking information to external spreadsheets. However, the implementation details are still under discussion.

Currently, Miocar stores all their data on a Google Sheet. Simply exporting this table to CSV format would not be ideal, as it would require Miocar to regenerate charts and graphs with each new CSV file, essentially starting from scratch each time. A more efficient solution might involve directly adding new rows to the existing Google Sheet that is used to store trip information, preserving the continuity of data, and maintaining any linked visualizations.

There's also a possibility that Miocar could store its data in a database, which would open up different avenues for data export and integration. The exact method of implementation will require further discussion and analysis to ensure it meets Miocar's specific needs and integrates smoothly with its current data management practices.

Once implemented, this automation would not only save considerable time but also significantly reduce the potential for human error in data transfer. Administrators would be able to quickly generate reports, analyze booking trends, and share data with stakeholders more efficiently, greatly enhancing the overall effectiveness of the booking management process. We will continue to explore the best approach for this feature in future discussions with Miocar.

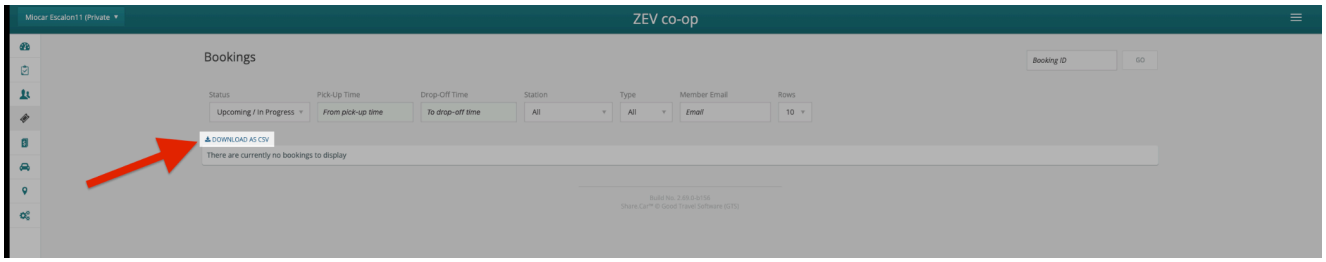


Figure 13: Changes made visually to the Backend Website to display the export button

Summary

These solutions were implemented using individually crafted Tampermonkey scripts that interact with the existing web application. This approach allows for rapid prototyping and deployment of these enhancements, demonstrating the power and flexibility of client-side scripting in improving web applications. The implementation of these features will result in a significant improvement in the efficiency of both member management and booking processes once implemented. Administrators will soon have more powerful and user-friendly tools at their disposal, allowing them to better serve members, save time on tedious tasks, and manage the car-sharing service more effectively and efficiently. As we continue to gather feedback and usage data on these new features, we anticipate further refinements and potentially the development of additional enhancements to further streamline operations and improve the user experience for the Miocar Staff.

Due to the limited access provided to Miocar's backend system, it was not feasible to develop and test prototype solutions for several identified operational challenges. These challenges include the manual scheduling of vehicle service appointments, the inefficient process of data entry for new member applications, and other operational inefficiencies within the current system. This restricted access significantly hampered our ability to design, implement, and evaluate potential improvements to streamline these processes. More comprehensive access to the backend infrastructure would be necessary to effectively address these issues and propose viable solutions, which will be something we will acquire in the future.

References

1. "Tampermonkey." Tampermonkey, www.tampermonkey.net. Accessed 30 Sep. 2024.

2. "Tampermonkey/tampermonkey." GitHub, github.com/Tampermonkey/tampermonkey. Accessed 30 Sep. 2024.

3. "Tampermonkey for Beginners." OpenUserJS, openuserjs.org/about/Tampermonkey-for-Beginners. Accessed 30 Sep. 2024.

4. Gill, Harjot. "Miocar Cost Monthly Model Spreadsheet",

https://docs.google.com/spreadsheets/d/1uWcs4_FBT8cXaWJkrXh0TCxY9g6RTn4ChZPwbOI5d6M/edit?gid=0#gid=0. 14 Aug. 2024.

5. Gill, Harjot. "Miocar-WebExtension-Scripts",

https://github.com/Hummaton/Miocar-WebExtension-Scripts. 30 Sep. 2024.

6. Amazon Web Services. "AWS Products." Amazon Web Services, Inc., 2024, aws.amazon.com/products/. Accessed 30 Sep. 2024.

7. Kinsta. "Cloud Computing Comparison: AWS vs. Azure vs. Google Cloud in 2024." Kinsta, 2024, kinsta.com/blog/cloud-market-share/. Accessed 30 Sep. 2024.