“Ss. Cyril and Methodius” University in Skopje
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Seminar work for the subject

Advanced Web Design

Topic:

Calendex – Unified Calendar

Mentor:

Dr. Boban Joksimoski

Prepared by:

Daniel Ilievski - 223021

Petar Hristovski - 223084

Skopje, February 2025

Abstract

Managing multiple calendars from different sources can be challenging, especially when users need to consolidate their events into a single view. The *Calendex* application addresses this issue by allowing users to import multiple ICS based calendars and displaying all events in one unified interface.

The application is built using Vue.js to provide a reactive and component-based architecture that guarantees a seamless user experience. Tailwind CSS and ShadCN Vue keep the design modern and responsive. ICS file parsing is handled by ical.js. Pinia handles state management efficiently, while event data is stored locally with the local storage API, so there is no need for any kind of backend.

Key features include a dynamic calendar dashboard, an event overview section for listing events and a guides section to assist users. The project follows Git-based version control, which allows for smooth development and tracking of changes.

This report describes the development process, technologies used, problems encountered, solutions found and potential improvements and ideas for future versions. The application provides a lightweight, easy-to-use solution for users who need to merge multiple calendars in one unified calendar.

Summary

Introduction

Today, everyone living in the digital world has to manage several calendars such as work schedules, personal appointments, or event subscriptions. However, there is no easy way to actually merge several ICS based calendars into one single, easily accessible calendar without relying on third-party cloud services. Due to this problem, inefficiencies are created, requiring users to check multiple sources separately.

For this, *Calendex* was created as a light and pure frontend application, capable of importing ICS files and displaying all events in a single view. Unlike cloud-based solutions, this project provides a solution whereby all data will be processed and stored locally so that your privacy and independence from any other service are kept.

Technologies used

The *Calendex* application uses modern web technologies to provide a fast, efficient, and user-friendly experience. The selection of these technologies was based on performance, ease of use, and maintainability. Below is a breakdown of the key technologies used:

Vue.js – Frontend Framework

Vue.js is a progressive JavaScript framework used for building the user interface of this application. It was chosen for its:

- Reactive and component-based architecture, making UI updates seamless.
- Lightweight nature, ensuring fast loading times.
- Ease of integration with other libraries.

Tailwind CSS – Styling Framework

Tailwind CSS was used to style the application due to its:

- Utility-first approach, allowing for rapid development.
- Customization capabilities, making it easy to match the desired UI theme.
- Minimal CSS file size, improving performance.

ShadCN Vue – UI Components

ShadCN Vue was used to provide pre-built, customizable UI components, enhancing the design and usability of the application. It integrates well with Vue and Tailwind CSS, ensuring a modern look.

Git – Version Control

Git was used for tracking code changes and managing the development workflow. The repository is hosted on GitHub, allowing easy collaboration and version management.

Application features and functionality

Importing calendars

The importing calendars feature of the application allows users to import calendars from two main methods: by providing an URL or by uploading an ICS file. This is a fundamental aspect of the application, as it enables seamless integration of external calendars into the application.

The user interface is designed to be user-friendly and offers a form for users to input the required details. Users can provide a name for the calendar, choose a color, and select the type of calendar, such as Google, Outlook, or Apple.

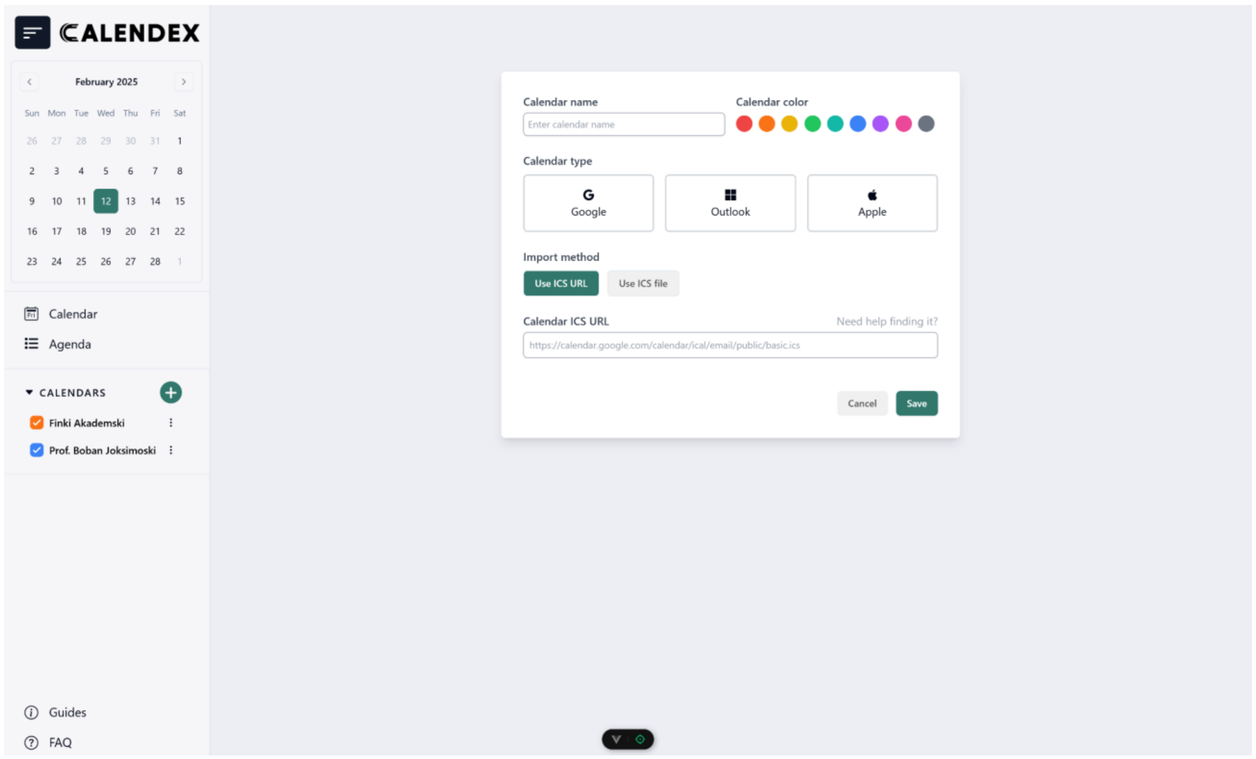
If the user opts to import a calendar via URL, the application provides a text input field where users can enter the URL of the ICS file. Additionally, there is a helpful link to guide users in locating the URL of their ICS calendar. Once the URL is entered, the application fetches the calendar data using a CORS proxy server and processes the events from the ICS content.

Alternatively, if the user selects the ICS file option, they can upload a calendar file directly. The application uses a file input field to select the ICS file from the user's local storage. The file is read, and the events are extracted from it.

The functionality behind both methods is built on the ICAL.js library, which parses the ICS content into events. These events are then mapped to the app's internal event model, including details like start and end dates, times, frequency, summary, description, and location. This ensures that all imported events are displayed accurately in the calendar.

Form validation is performed as well, to ensure that all required fields, such as the calendar name, color, type, and the selected import method, are filled out. If any fields are missing or invalid, error messages are displayed, preventing users from proceeding until all required inputs are provided.

In conclusion, this feature enables users to effortlessly import external calendars either by URL or by uploading ICS files, making it highly adaptable to various calendar services.

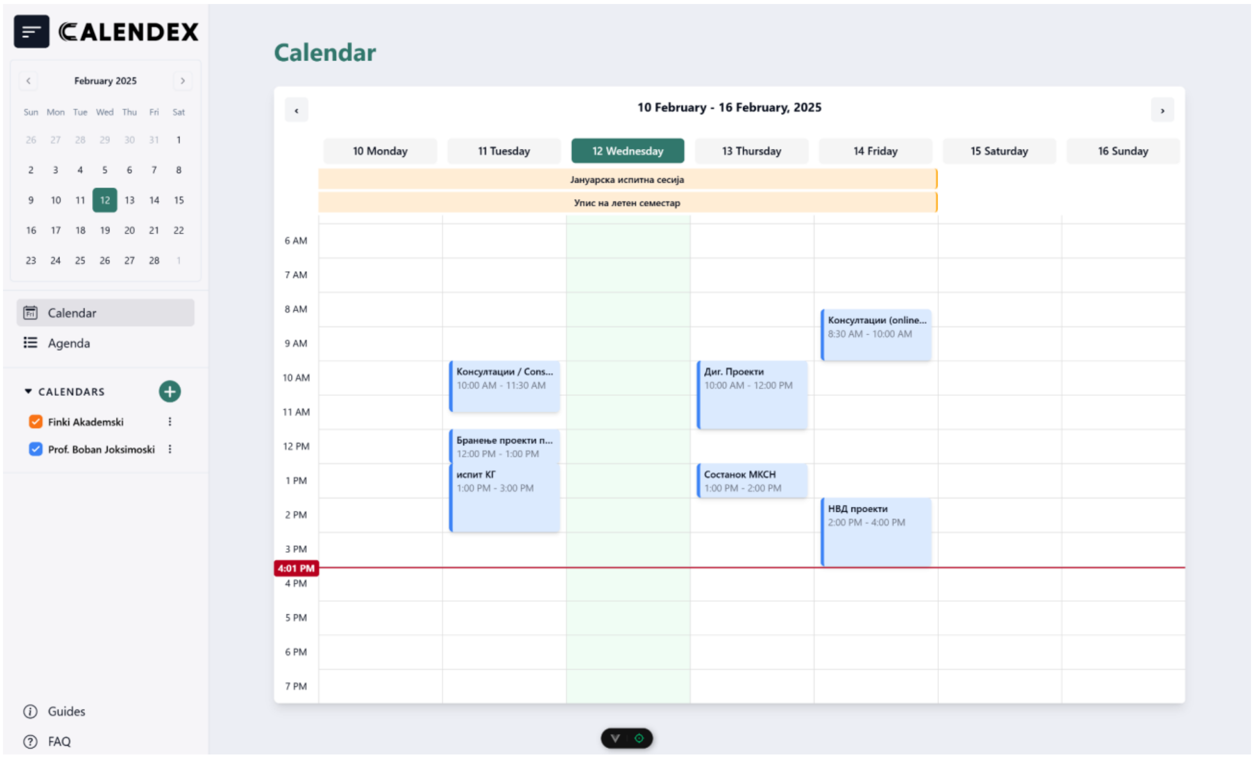


Calendar dashboard

The calendar dashboard is the main section of the application and presents a structured and clean representation of events from various calendars. It is a read-only screen where one can see their calendars but cannot perform any action on the events. The dashboard updates dynamically to show events depending on which calendars are checked in the sidebar, so users can view any set of events they wish at any given time. Moreover, the imported calendars are refreshed every 10 minutes, ensuring that the latest events are always available and up to date, providing users with the most accurate and current schedule information. Every event is colored based on which calendar it is part of, making it easy to tell sources apart. Events that overlap are resized automatically by narrowing their widths, preserving clarity and legibility, while multi-day events are positioned atop the main grid to indicate them as lasting longer than a single day.

The calendar grid itself is organized just like the Google Calendar we're used to, with days organized in a grid for the week. An active time line runs across the grid to provide a visual representation of the current time, and the current day is also highlighted in green as a convenient point of reference for users. Events are rendered and cleared dynamically according to active calendar selections ensuring that users always see only the relevant events.

The dashboard fully supports recurring events, handling a wide range of repetition patterns. Events can repeat daily, weekly, monthly, or yearly, with additional flexibility in defining how often they occur. The logic supports events that happen a specified number of times, continue until a specified date, or repeat indefinitely. It also allows for more complicated recurrence rules, like events that happen on certain weekdays or with a step interval, meaning they might repeat every few days, weeks, or months.



With this feature, users can view their entire calendar from multiple sources in one place, allowing them to stay organized and keep track of their schedule effortlessly.

Event overview

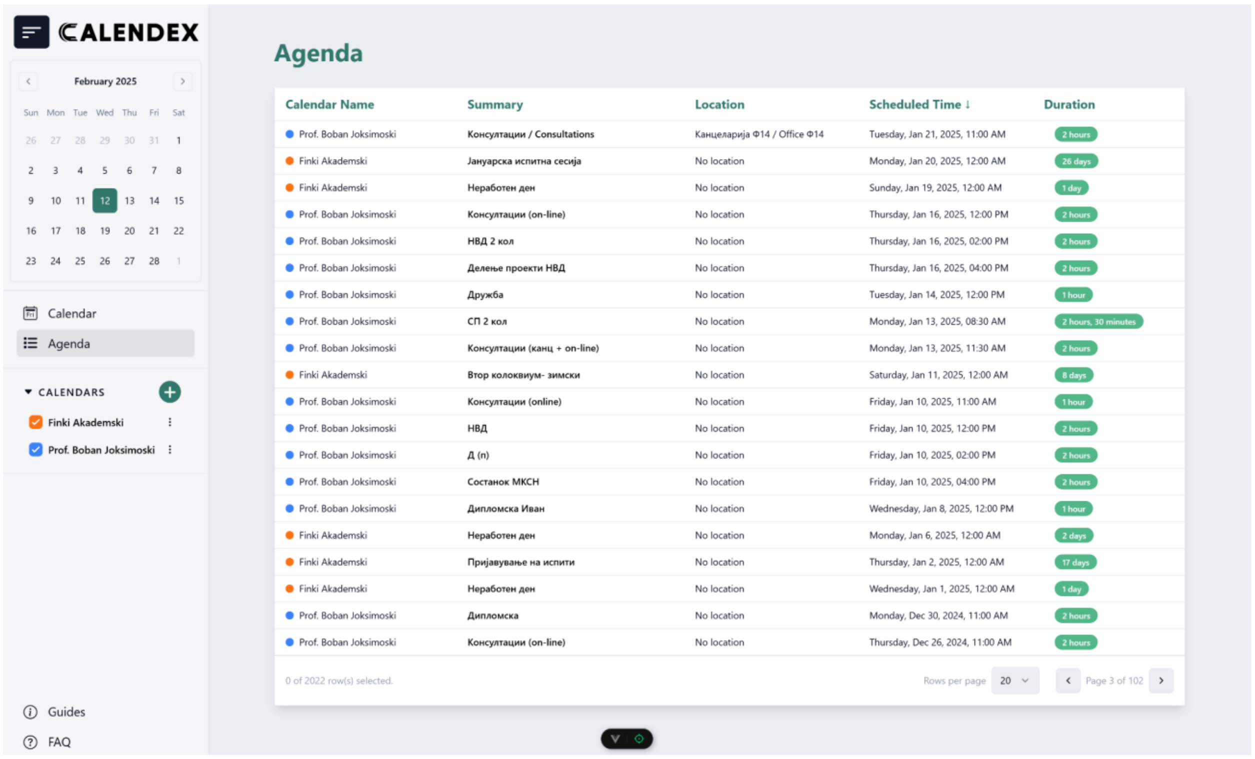
The event overview is another key feature of this application, providing users with a clear and organized view of their events. This feature displays crucial event details in a table format, making it easy for users to navigate and manage their schedule.

The table is designed with several columns that allow users to quickly interpret the event data. The calendar name column identifies which calendar the event belongs to. The summary column presents a brief description or title of the event. The location column indicates where the event will take place. The scheduled time column displays when the event starts, and the duration column indicates the event's length, showing. This structured table layout ensures that users can easily find and understand their event details without unnecessary complexity.

One of the important features is sorting. The user can sort the events by any of the columns in the table. By clicking on a column header, the user can choose to sort the events in ascending or descending order. This allows the user to quickly find events based on their preferences.

The pagination feature is also included. This means that the user can choose how many events they want to see on each page. They can pick from 5, 10, 20, or 50 events per page. At the bottom of the table, there are buttons to navigate between pages. The user can see which page they are on and how many pages are available. This makes it easier to manage large sets of events.

The design of the table is clean and modern, thanks to Tailwind CSS and ShadCN Vue. Each event is also color-coded based on the calendar it belongs to, which helps the user quickly identify events from different sources.



Agenda

Calendar Name	Summary	Location	Scheduled Time ↓	Duration
● Prof. Boban Joksimoski	Консултации / Consultations	Канцеларија Ф14 / Office Ф14	Tuesday, Jan 21, 2025, 11:00 AM	2 hours
● Finki Akademski	Јануарска испитна сесија	No location	Monday, Jan 20, 2025, 12:00 AM	26 days
● Finki Akademski	Неработен ден	No location	Sunday, Jan 19, 2025, 12:00 AM	1 day
● Prof. Boban Joksimoski	Консултации (on-line)	No location	Thursday, Jan 16, 2025, 12:00 PM	2 hours
● Prof. Boban Joksimoski	НВД 2 кол	No location	Thursday, Jan 16, 2025, 02:00 PM	2 hours
● Prof. Boban Joksimoski	Делење проекти НВД	No location	Thursday, Jan 16, 2025, 04:00 PM	2 hours
● Prof. Boban Joksimoski	Дружба	No location	Tuesday, Jan 14, 2025, 12:00 PM	1 hour
● Prof. Boban Joksimoski	СП 2 кол	No location	Monday, Jan 13, 2025, 08:30 AM	2 hours, 30 minutes
● Prof. Boban Joksimoski	Консултации (канц + on-line)	No location	Monday, Jan 13, 2025, 11:30 AM	2 hours
● Finki Akademski	Втор колоквиум- зимски	No location	Saturday, Jan 11, 2025, 12:00 AM	8 days
● Prof. Boban Joksimoski	Консултации (online)	No location	Friday, Jan 10, 2025, 11:00 AM	1 hour
● Prof. Boban Joksimoski	НВД	No location	Friday, Jan 10, 2025, 12:00 PM	2 hours
● Prof. Boban Joksimoski	Д (п)	No location	Friday, Jan 10, 2025, 02:00 PM	2 hours
● Prof. Boban Joksimoski	Состанок МКСН	No location	Friday, Jan 10, 2025, 04:00 PM	2 hours
● Prof. Boban Joksimoski	Дипломска Иван	No location	Wednesday, Jan 8, 2025, 12:00 PM	1 hour
● Finki Akademski	Неработен ден	No location	Monday, Jan 6, 2025, 12:00 AM	2 days
● Finki Akademski	Пријавување на испити	No location	Thursday, Jan 2, 2025, 12:00 AM	17 days
● Finki Akademski	Неработен ден	No location	Wednesday, Jan 1, 2025, 12:00 AM	1 day
● Prof. Boban Joksimoski	Дипломска	No location	Monday, Dec 30, 2024, 11:00 AM	2 hours
● Prof. Boban Joksimoski	Консултации (on-line)	No location	Thursday, Dec 26, 2024, 11:00 AM	2 hours

0 of 2022 row(s) selected.

Rows per page 20

Page 3 of 102

In conclusion, the event overview is an important feature of the application. It helps users view and sort their events efficiently, making it a useful tool for anyone who needs to organize their schedule.

State management with Pinia and localStorage

In this project, state management is handled using Pinia, a lightweight state management library designed for Vue.js applications. Pinia is used to manage the state of the calendars, including their details and events. The calendar data is stored in a central store, ensuring a consistent state across components and easy manipulation of the data when required.

The store is initialized with a state that holds the array of calendars retrieved from localStorage (if available). If no calendar data is found in localStorage, the state defaults to an empty array. This ensures that the calendar data persists across page reloads, providing a more stable user experience.

The store contains several actions to interact with the calendar data, such as adding a new calendar, editing existing ones, and deleting calendars. When a new calendar is added or an existing one is updated, it is stored in the localStorage. This keeps the user's calendar data synchronized between sessions.

A key function in this store is updateFilteredEvents(), which is responsible for dynamically updating the displayed events based on the user's selection of calendars. The user can select multiple calendars via checkboxes in the sidebar, where each selection triggers the updateFilteredEvents(). This function filters through the calendars, finding the ones that are selected, and aggregates the events associated with those calendars. The filtered events are then stored in the filteredEvents state. If no calendars are selected, the filteredEvents array is cleared.

This functionality ensures that the user can easily toggle the visibility of events from different calendars, and it updates the view in real-time as the selections change. The data persistence feature using localStorage ensures that the calendar states, including selected calendars and their events, are maintained across different sessions, providing a seamless user experience even after page reloads or closing and reopening the application.

By using Pinia for state management and localStorage for persistent storage, this approach allows for effective handling of dynamic user interactions, while ensuring the calendar data remains intact and synchronized across different components and sessions.

Faced challenges and solutions

During the development of this application, we faced a few challenges. Below is an explanation of each major one.

CORS restrictions

Importing calendars via URLs was a significant challenge during development. Browsers have strict Cross-Origin Resource Sharing (CORS) policies. They restrict making a direct request to external servers, unless they give permission explicitly. This restriction prevented the application from fetching ICS files directly from third-party calendar services like Google Calendar, Outlook, and iCloud.

To get around this limitation, we used a CORS proxy server hosted on Heroku.The proxy server acts as an intermediary, fetching the ICS file on behalf of the application and returning the response back. This approach allows the application to fetch calendar information, while still not violating browser security policies.

Parsing ICS files and displaying events

Displaying the many different types of events, such as multi-day events and recurring events with complicated rules, was not easy. Some repeated daily, weekly, or monthly; others followed more specific patterns like the nth weekday or until a set date.

For handling this, we used the ical.js library to parse the ICS files. The recurring events were expanded into individual occurrences. We also had to adapt the calendar to span multi-day events and handle overlapping events. With those adjustments, the calendar would display all of the imported events correctly.

Time zone differences

Another challenge arose with the events being saved in different time zones, causing incorrect positioning in the calendar. Some ICS files included time zone information, while others used UTC or local time, leading to inconsistencies.

To fix this, we converted all event scheduled times to the user’s current time zone. This ensured that events appeared at the correct times regardless of their original time zone, which provided a consistent and accurate display across all imported calendars.

Version control and development workflow

During the development of Calendex, we followed light version control and collaboration to make sure that it went through with smooth and efficient progress. Given that the project was developed by two developers, we decided to go with the single-branch workflow, utilizing only the main branch in our GitHub repository. This is so that the development process remains quite simple and there are no conflicts, while also having all changes integrated and available immediately.

This, therefore, shows that Git being the version control system, while hosting on GitHub, facilitates the tracking of changes in our code efficiently and maintenance of history. Working within one branch also makes sure that at any one time, any developer can have a current version of a modification, which would eliminate several problems with a merge of a few branches later on.

Because we kept in close contact, it was easier to communicate and not let major conflicts arise. We were discussing on messaging platforms and in meetings to keep ourselves updated about what each was doing to avoid duplications of work. This system allowed us to commit often, test the updates immediately, and ensure that changes were integrated well before the update was pushed to the repository. Code reviews were very informal. We discussed the changes as they occurred and decided on committing. The features and bug fixes were implemented in small, manageable updates to keep the codebase stable. Therefore, frequent commits were made and always pushed to keep us in sync, instead of long-lived feature branches. Clear commit messages were provided to explain changes done, thus enabling tracking of changes done within the history of the project.

The single-branch workflow is not precisely the right choice for big teams, always; here, it worked wonderfully because of how closely we collaborated and set up the steps. Anyway, if the project ever grows or involves other developers in the future, we will probably apply a feature-branch workflow in order to handle parallel development inside the stability of the main branch.

Overall, it gave us the opportunity to have an efficient development process, avoiding unneeded complexity in the code and keeping the codebase clean, updated constantly.

Future improvements & ideas

As the application evolves in future, there are numerous possibilities for expanding its features and enhancing its functionality.

Improved mobile responsiveness

Enhanced mobile experiences with layout optimizations for touch and navigation would mean better scaling, touch-friendly elements, and accessibility on small screens. Besides that, swipe gestures would enable users to switch dates and views with ease.

Superior search

Improving the search functionality could bring a lot of benefits. Adding a more advanced date-parsing system would allow users to quickly and accurately find events based on dates, keywords, or tags. This improvement would be especially beneficial for users managing multiple calendars, ensuring a more efficient and precise search experience.

User authentication

Implementing user authentication would enable synchronization across multiple devices, ensuring that imported calendars, settings, and preferences are easily accessible from any platform. By storing user data in the cloud, changes made on one device would automatically reflect on others. Possible authentication methods include OAuth-based login using Google, Microsoft, and Apple accounts, providing a secure and convenient way for users to access their unified calendar.

Calendar view modes

The calendar could be enhanced with multiple view modes, allowing users to switch between daily, monthly, weekly and yearly perspectives. This would provide greater flexibility in displaying both short- and long-term schedules, making it easier to navigate and plan events efficiently.

Different appearance on mobile devices

Adapting the display logic to automatically switch between calendar and agenda views based on screen size would enhance usability. On mobile devices, a list-based agenda might be more practical compared to a full calendar grid. This would ensure that events can remain easily readable. It would also improve the navigation and provide a more intuitive user experience.

Theme modes

Another great future improvement would be adding an option in settings to switch between themes/light and dark mode.

Event reminders

Providing a notification system for users to remind them of any upcoming events could be a great idea. The user would be allowed to define their own custom reminder settings: a choice to receive notifications a set amount of time before the beginning of the event-such as 10 minutes, 1 hour, or 1 day in advance and also an option to opt for daily summary emails.

Event creation and management

Event creation and management via integration with third-party calendar APIs such as Google, Outlook, and Apple Calendar could be a complex but yet very powerful feature. The user could create or edit events within the application, and the changes would automatically be synced with his or her preferred external service.

Guides for applications

Currently, the application includes guides only for the web-based Google, Outlook and iCloud calendar services. In the future, the support could be expanded to include step-by-step instructions for mobile and desktop applications as well. This will ensure that users across all platforms can easily integrate and manage their calendars, improving accessibility and ease of use.

Conclusion

The challenge of keeping track of many calendars is tackled effectively with Calendex. It's a lightweight and efficient solution for bringing together ICS-based calendars into one single, unified view. Built with Vue.js, Tailwind CSS, and ShadCN Vue, this application is modern, responsive, and smooth. Using ical.js for parsing ICS and Pinia for state management, this enables smooth handling of data while local storage ensures privacy without external servers.

During the development of this project, various kinds of technical challenges were faced and solved, which included efficient state management, responsive design implementation, and smooth calendar integration. So far, the application has been able to offer users a pragmatic alternative to cloud-based calendar services with full control over their data.

In general, Calendex offers a useful, user-friendly, and privacy-first solution for those who want to bring multiple calendar sources together into one streamlined interface.

References

[1] Vue.js, Vue.js Team, https://vuejs.org/

[2] Tailwind CSS, Tailwind Labs, https://tailwindcss.com/

[3] ShadCN Vue, https://www.shadcn-vue.com/

[4] Git, Git SCM Team, https://git-scm.com/

[5] Pinia, Vue Core Team, https://pinia.vuejs.org/

[6] Vue Router, Vue.js Team, https://router.vuejs.org/

[7] Vite, Evan You & Vite Team, https://vite.dev/

[8] date-fns, https://date-fns.org/

[9] DOMPurify, https://www.npmjs.com/package/dompurify

[10] ical.js, https://www.npmjs.com/package/ical

[11] Bootstrap Icons, https://icons.getbootstrap.com/

[12] Font Awesome Icons, https://fontawesome.com/icons