Software Engineering: Custom Date-Picker Component

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Isabella Felaco

Spring, 2024

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Advisor

Briana Morrison, Department of Computer Science

Software Engineering: Custom Date-Picker Component

CS4991 Capstone Report, 2024

Isabella Felaco
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
isfelaco@gmail.com

ABSTRACT
OpenGov, a cloud-based software company with the goal of empowering local governments, needed a custom date-picker component to utilize in their products. During my internship at the company, I was tasked with creating this component from a base design. I coded the custom component entirely from scratch in React.JS using Typescript (a language similar to JavaScript) and Cascading Style Sheets (CSS). As a result, the component was integrated company-wide into their suite of products and for use with other teams. I even utilized it in future projects that I contributed to. This component could be improved to integrate open-source code that generates date-picking. Additionally, I plan to work at a software company that serves as a government contractor, to continue my interest in frontend software development and creating government-used software.

1. INTRODUCTION

OpenGov's motto is "powering more effective and accountable government." All of my work, including my work on the custom date-picker, was designed to further the goal of increasing trust and transparency in governments,

During my internship, I worked in the Permitting and Licensing division. The software being developed was a portal for constituents and government employees to submit and review documents, respectively. Applicants and reviewers have different views with different privileges. The applicants are able to fill out forms custom-made by employees, and employees are able to view submissions, review them, and send feedback directly to the applicant.

The primary purpose of the date-picker was to allow applicants to select a date in a custom date form field. Later, it was added to other workflows to allow date selection wherever needed, such as in selecting when a form is due. This software is important for several reasons. First, it increases effectiveness because the process of filing and approving documents is streamlined, and applicants receive immediate, custom feedback. Additionally, it increases citizen participation because of the ease in submitting applications; constituents no longer have to go to offices in person or mail printed documents. Last, it increases accountability in government because files that are publicly accessible are even more accessible when stored on the cloud. The date-picker component contributes to meeting these goals.

2. RELATED WORKS

My work on the date-picker component utilized several technologies and coding strategies. The technology I leveraged most

was React.JS (React), a JavaScript(JS) frontend framework. For this project, I wrote in TypeScript (TSX), a high-level programming language that adds type-checking to JS. A main feature of React is that it is declarative and component-based, which is helpful in designing UIs.

Secondly, I used date-fns, a toolset for manipulating JS's built-in Date object (date-fns, 2024). To preface, the JS Date object defines a timestamp down to the millisecond (Mozilla, 2024). The Date object has many functions for manipulating dates, but date-fns has extended functions that were required for this component. I used many of the functions throughout my component for advanced functionality.

Another resource I utilized was Figma. Figma is an interface design application that allows UI/UX designers to share their designs with developers. I used this software to get precise details on my component and make sure it met the exact specifications, as UI designers have very specific visions that directly affect user experience. It gave me a starting point a reference for asking for additional details or clarification.

Finally, I utilized styled-components, a library used to write CSS in JS while building custom components in React. Every style is tied to a specific component, allowing for easy maintenance and clarity in the code (Styled Components, 2024). There is also the ability to implement class hierarchy and extend classes. It was particularly useful for this task, as I had a lot of custom styles and needed to style individual components.

3. PROJECT DESIGN

I was tasked with creating a custom date-picker component. This component was an inline input with an icon that opened an interactive calendar. Users were to be able to select a date in two ways: by typing in the input, or by selecting a date in the calendar. Our company already had inputs that formatted dates, so my task was to extend that component to include the interactive calendar.

3.1 The Design Choice

For this component, I decided to create from scratch rather than use pre-existing code, such as MUI's React Date Picker. This was ultimately due to the fact we needed advanced functionality that these components did not provide, such as selecting multiple dates, ranges of dates, and blacking out certain dates. It was also more coherent to keep using date-fns, as it was regularly used throughout our code space.

3.2 Creating the Design

I was given a design in Figma that I had to recreate in React. The Figma design included views for different states of the component, such as the open calendar, selecting a date, and navigating through the date-picker. These designs gave me a starting point before I developed the actual functionality.

In order to create this component, I utilized styled-components. Styled-components allowed me to create custom styles for different components within the date-picker. For example, each square in the calendar had a unique design that changed according to user input (e.g., it was colored blue when selected, white when not selected, or gray when disabled). The calendar icon that opened the calendar had similar behaviors in that it changed colors depending on whether the calendar was open or not.

3.3 Implementing Functionality

The date-fns functions I used most were for formatting and comparing dates. For example, I used the "isEqual" method to determine if the date selected was equal to the current date (as given by the generic JS Date

object), in order to give it a custom style. I also used the "parse" function to manipulate data and use dates in different formats, such as changing the format that the data came in into a usable form, and then returning it to the original format.

3.4 Challenging Features

The greatest difficulty I encountered while creating this component was configuring for date ranges as input to the component and as user input. For context, the component was to be used as a form component where users could select dates and submit it with a form. The forms were customizable, so admins could configure the settings of each of the components. The date picker was to have settings allowing or disallowing the selection of certain dates and of multiple dates at a time. This meant that when rendered, the component would receive arguments that told it whether or not to allow these features. I had to determine the best format that this data should come in and figure out a way to manipulate it to implement the functionality.

The data for disabling certain dates came in the form of a regular expression, where specific dates or patterns of dates (for instance, every other Sunday) could be disabled. I had to parse this regular expression and apply the pattern to the calendar's formatting. Additionally, I had to do form validation so that if the user manually typed an invalid date, error feedback would be displayed.

For allowing users to select multiple dates, I used React's "useContext" hook. A React hook is a built-in function that allows the user to tap into the React state and lifecycle features from functional components (React, 2024). Specifically, "useContext" is employed alongside the "useState" hook to share state between deeply nested components (W3Schools, 2024). I utilized the context function to facilitate sharing the state of the calendar (i.e., what dates were currently selected) to the parent component, the input itself. This also allowed me to populate the input with the selected date once a calendar square was clicked by the user.

3.5 Ensuring Accessibility

An important aspect of the design process was ensuring that the component was accessible and Section 508-compliant. This included making the entire component compatible with keyboard control. To do this, I utilized React's `tabIndex` attribute, which creates an order for elements to be focused when navigating with the tab button. Additionally, I made sure the calendar pop-up was accessible by enabling users to press the enter button to open and close it. Users could also navigate the calendar through the keyboard using the left and right arrows and tab to enter the calendar month.

4 RESULTS

Upon completion, the custom date-picker component was directly integrated into the larger project that I was working on. This was a workflow titled "Requesting Changes," where admins could create custom forms, users could fill them out and submit them, then admins could provide direct and immediate feedback via the online platform. This component was added to the list of form fields available when creating forms. It was used to provide more accessibility and flexibility in date selection.

After its success in the Requesting Changes project, the component was later integrated company-wide. It was translated by other engineers to be part of our reusable component library for other platforms.

5 CONCLUSION

The date-picker component is an integral part of OpenGov's form submission workflow. It is now used company-wide on all forms and has been expanded to include additional configurations that can be set by the admin of the form. By being keyboard-accessible, t increases accessibility to more users and it provides more specificity in what dates are allowed to be selected than with the original date input.

This component was important to not only the company, but to my development as a computer scientist. I began my internship with very little knowledge of frontend computer programming, but with mentorship and outside practice, I eventually became a more effective software engineer. Because of this experience, I now have the ability to create my own independent projects and will be continuing as a full-time full-stack software engineer.

6 FUTURE WORK

Since my work on the date-picker component, it has been expanded by other developers to include more options for disabling dates. Other expansions could include more robust testing as only simple unit tests were originally written.

My work as a software engineer will continue with OpenGov full-time. I will be working in both frontend and backend code bases, thus expanding by professional coding experience.

REFERENCES

Mozilla. (n.d.). Date - JavaScript. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date

date-fns. (n.d.). date-fns | Modern JavaScript date utility library. https://date-fns.org/v3.3.1/docs

Styled Components. (n.d.). Documentation. https://styled-components.com/docs/basics
https://mui.com/x/react-date-pickers/date-picker/

ReactJS. (n.d.). Hooks overview. https://legacy.reactjs.org/docs/hooks-overview.html#:~:text=Hooks%20are%20functions%20that%20let,if%20you'd%20like.)

W3Schools. (n.d.). React useContext hook. https://www.w3schools.com/react/react_usecontext.asp