Kerim Şahin Optimizes Data-Heavy Blazor Apps with Syncfusion

Reduced development time by eliminating custom UI component creation

Improved UI performance in data-heavy Blazor applications

Lower development costs and improved profitability

Introduction

Şahin is an independent freelance software developer specializing in modern application development using technologies such as .NET Core, Blazor, .NET MAUI, TypeScript, and Angular. Working across multiple client projects with tight deadlines, he focuses on delivering high-quality, data-driven applications efficiently. To remain competitive and profitable as a solo developer, he relies on tools that minimize development time without compromising performance or user experience.

Problem Summary

Before adopting Syncfusion, Şahin spent a significant amount of time building custom UI components from scratch for each project, which led to performance bottlenecks and inefficiencies—especially in Blazor WebAssembly applications handling large datasets. Although backend database queries executed quickly, frontend rendering struggled, resulting in slow page loads and poor browser performance. Developing, debugging, and maintaining these custom components increased costs, consumed valuable project time, and made it difficult to meet tight client deadlines.

> *"It's an affordable price and it is free to use, up to a certain point."*
>
> **– Şahin**

Industry

IT services and consulting

Company

Kerim Hilmi Personal

Region

Turkey

Customer Detail

Kerim Şahin, Senior Software Engineer

Use Case

Şahin needed a reliable UI component library that could seamlessly integrate into his existing .NET and Blazor projects, efficiently handle large volumes of data, and reduce the time spent on repetitive UI development tasks. The solution had to support common features such as data listing, filtering, paging, and dialogs while maintaining high performance and offering flexibility across different project types.

BEFORE SYNCFUSION

Challenges

- Building custom UI components from scratch consumed excessive development time.
- Poor performance when handling large datasets in Blazor WebAssembly.
- Slow browser rendering despite fast server-side database queries.
- Building custom UI components from scratch consumed excessive development time.
- Increased project costs due to extended development cycles.
- Tight client deadlines requiring faster turnaround times.

AFTER SYNCFUSION

Solutions

- Adopted Syncfusion's prebuilt UI components instead of custom-built alternatives.
- Implemented the Syncfusion DataGrid to manage listing, filtering, and server-side paging.
- Used Syncfusion Dropdown components for specialized selection scenarios.
- Leveraged the Dialog component extensively for consistent user interactions.
- Took advantage of continuously updated components released by Syncfusion

Implementation

Şahin integrated Syncfusion components directly into his existing application architecture with minimal disruption. The DataGrid component was configured to perform server-side operations such as filtering, sorting, and paging, which significantly reduced the processing load on the client browser. Built-in features eliminated the need for custom logic while ensuring optimal performance even with large datasets.

The Dialog and Dropdown components were incorporated as reusable UI elements across multiple projects, allowing Kerim to standardize user interactions and reduce repetitive development tasks. As Syncfusion released new components and updates, Şahin was able to adopt them quickly due to their consistent API design and comprehensive documentation, further accelerating his development workflow.

Result

By using Syncfusion components, Kerim transformed time-consuming development tasks into a streamlined process. Applications performed faster, handled large datasets more efficiently, and required significantly less manual customization. This allowed him to complete projects quicker, move seamlessly to new engagements, and focus more on delivering business value to his clients.

Future Plan

Şahin plans to continue expanding his use of Syncfusion components across future projects, particularly as he explores more advanced Blazor and .NET MAUI applications. He also intends to adopt newly released components to further standardize development and improve application usability.

Conclusion

Syncfusion's prebuilt UI components enabled Kerim Şahin to overcome performance challenges, reduce development time, and deliver high-quality applications faster. By replacing custom-built components with reliable, feature-rich alternatives, he improved both productivity and application performance—making Syncfusion a key part of his long-term development toolkit.