We would like to create a page builder that allows the users to create new pages by adding/removing components from a page.

The idea is to create a way for the user to be able to add/removed components from a webpage, creating the possibility for the user to create new websites without the need of developing knowledged, beeing able to focus only on the content and pages structure.

Lumina

Project Overview:
Lumina is a revolutionary web application that enables users to effortlessly design and publish stunning websites without the need for any coding knowledge. Leveraging the power and flexibility of NextJS, our platform offers an intuitive drag-and-drop interface combined with a rich library of customizable components, empowering users to bring their creative visions to life with ease.

Key Features:

1. **Intuitive Drag-and-Drop Interface**: Our user-friendly interface allows users to easily drag and drop components onto a canvas, making website design an intuitive and enjoyable experience.
2. **Extensive Component Library**: NextJS WebBuilder comes equipped with a vast library of pre-designed components, including text blocks, images, videos, buttons, forms, and more. Each component is fully customizable, giving users the flexibility to tailor their websites to their exact specifications.
3. **Flexible Customization Options**: Users can personalize every aspect of their website's appearance and behavior, from adjusting text styles and colors to adding animations and interactions, all without writing a single line of code.
4. **Dynamic Page Structure**: With NextJS WebBuilder, users have full control over the structure of their website. They can easily add, remove, and rearrange pages, create hierarchical navigation menus, and organize their content to create a seamless user experience.
5. **Responsive Design**: Our platform automatically generates websites that are fully responsive, ensuring a consistent and optimal viewing experience across desktops, tablets, and smartphones.
6. **Preview and Publish**: Users can preview their websites in real-time before publishing them to the web. Once satisfied with the design, they can easily publish their websites to a custom domain or a subdomain provided by our platform.

Technology Stack:

Frontend: NextJS
Backend: Any system capable of saving data, preferably Funfunz
Deployment: Vercel, any static file server (no server side render), Docker, Kubernetes, ...
Target Audience:
NextJS WebBuilder is designed for entrepreneurs, small/large businesses, bloggers, and creative individuals who want to establish a professional online presence without the complexity of traditional web development.

Why Invest in NextJS WebBuilder?

Market Demand: The demand for user-friendly website builders is rapidly growing as more individuals and businesses seek to establish their online presence. NextJS WebBuilder taps into this lucrative market by offering a best-in-class solution that combines simplicity with powerful features.

Competitive Advantage: Our use of NextJS sets us apart from traditional website builders by offering superior performance, scalability, and flexibility. With NextJS WebBuilder, users can create dynamic and interactive websites that rival those built by professional developers.

Revenue Potential: NextJS WebBuilder offers multiple revenue streams, including subscription plans, premium templates, custom domain sales, and third-party integrations. With a scalable business model and a growing user base, the potential for revenue growth is substantial.

Vision for the Future: We envision NextJS WebBuilder evolving into a comprehensive platform for digital content creation, encompassing not only websites but also blogs, e-commerce stores, and more. By investing in our platform, you're investing in the future of online content creation.

Tech side

Architecture

Actor → Lumina: request
Lumina → Renderer: renders a renderer
Renderer → FE Component: renders a FE component
FE Component → Data Fetcher: uses the data fetcher to grab the required data
Data Fetcher → Backend: makes the connection with the backend
Backend → Data Fetcher: responds back
Data Fetcher → FE Context: saves the data in the FE context
FE Context → Renderer: triggers a context changed on the renderers
FE Context → FE Component: triggers a context changed on the FE Component
FE Component → Renderer 2: renders another renderer inside for its children
Renderer 2 → FE Component: finalized rendering
FE Component → Renderer: finalized rendering
Renderer → Lumina: finalized rendering
Lumina → Actor: response