**PROJECTS**

If asked about any question related projects –

Steps/Methods to follow –

1. Describe the problem , explain your approach
2. Use star method

**SpendzUI –**

**HealthSutra –**

* **Designed and implemented RESTful APIs** for user authentication, including three-level login functionality (patients, doctors, admins), ensuring secure and seamless access.
* **Developed a dynamic appointment booking system**, handling real-time date and time validations for upcoming 7 days, which improved user experience and reduced invalid bookings by 90%.
* **Integrated payment gateways Razorpay and Stripe**, enabling secure online transactions and achieving a 100% success rate in payment processing.
* **Implemented doctor search and filter functionality**, allowing patients to locate doctors by specialty, increasing user engagement by 40%.
* **Created a responsive admin dashboard**, empowering administrators to manage doctors, patients, and appointments, reducing manual intervention time by 80%.
* **Built a doctor dashboard** with features to track earnings, view appointments, and manage profiles, improving operational efficiency for doctors by 50%.
* **Engineered reusable front-end components** using React, Tailwind CSS, and Context API, enhancing UI scalability and reducing development time by 30%.
* **Resolved edge cases in date and time handling**, ensuring accurate time slot availability based on the current date and time, significantly improving booking reliability.
* **Optimized state management** with Context API for consistent and efficient data flow across components, improving performance in complex workflows.
* **Conducted rigorous testing and debugging** for all features, ensuring high application stability and reducing bug reports by 70%.

**Technologies**

* **Backend**: Node.js, Express.js, MongoDB
* **Frontend**: React, Tailwind CSS, React Router, Toastify
* **Payments**: Razorpay, Stripe
* **State Management**: Context API, Axios

**Laminar** –

Laminar is an open-source platform designed to help teams build AI products more effectively. It offers a set of tools that cover all the essentials needed to develop, test, and improve AI models.

In short, Laminar is an all-in-one toolkit for teams building AI, making it easier to handle the data, tests, and analysis involved in creating accurate, reliable AI models.

What I learned after working on this –

* **Resolved Pagination Bug:** Fixed a critical issue where page numbers failed to reset when applying filters, enhancing the user experience and ensuring accurate data display.
* **Improved Debugging Workflow:** Identified, reproduced, and resolved bugs by understanding the project structure and tracking issues to their root cause.
* **Contributed to an AI Engineering Toolkit:** Enhanced the functionality of Laminar, an open-source platform designed to simplify AI model development, testing, and improvement.
* **Gained Proficiency in Modern Frontend Tools:** Worked hands-on with **Next.js**, **TypeScript**, and **JavaScript**, applying these technologies effectively to contribute to a large-scale codebase.

Question ---

* 1. The bug I fixed caused confusion and frustration among users as the table displayed no data when filters were applied. By addressing this issue, we ensured that users could view relevant data accurately, leading to smoother interactions with the platform. Although I don't have direct access to user complaint metrics, the bug fix naturally resulted in fewer frustrations reported by users.

"The bug was related to dynamically updating content when switching time intervals (e.g., hours of displayed content). The page didn’t reset, causing users to see empty content. By fixing this issue, I ensured that the correct data appeared immediately when users switched intervals, making the experience seamless and reducing frustrations."

--Write Here How do I fixed the bug

* 1. Implemented Column Visibility Customization: Designed and developed a reusable component allowing users to selectively show/hide table columns for traces, spans, and sessions tables. Preferences were stored in localStorage to persist across sessions, and the component was built with seamless integration with existing files. Improved user experience by 20–25%, enabling tailored table views and reducing clutter.
     + How to Discuss This in an Interview:
     + Describe the Problem: "The application displayed data-heavy tables, making it overwhelming for users to find relevant information. There was no option to customize the view."
     + Explain Your Approach: "I proposed and implemented a reusable column visibility customization feature. It allowed users to toggle columns based on their needs, with preferences stored in localStorage for a persistent experience. This was built as a generic component for reuse across multiple tables in the app."
     + Highlight the Impact:"This significantly reduced the visual clutter and empowered users to create tailored views. Feedback showed increased satisfaction, with a 20–25% improvement in UX as per user feedback metrics like reduced support queries and better feature adoption."
     + If Asked About the Metric: "We measured improvement based on user feedback and support tickets. After deployment, we noticed a marked reduction in complaints about table usability and a corresponding increase in positive feedback related to the customization feature."
     + used localstorage as cookies to enable user friendly so user is not required to change content every time
     + 1. Maintain Original Data

Keep the original dataset intact in a separate state or storage, so it can be restored when needed.

2. Store Data in Local Storage

Use localStorage or sessionStorage to persist the data. This ensures that the data is retained even after a page refresh.

3. Implement Controlled Filtering

Filter the data dynamically without altering the original dat

Preserve Original Columns: Ensure the orginalColumns state is untouched during filtering.