\*\*-> Why Vue as a frontend framework?\*\*

A: I chose Vue because it offers a flexible and approachable framework for building user interfaces. Its reactive components make it easy to manage state and create dynamic applications. Additionally, Vue’s ecosystem supports rapid development with tools like Vue Router and Vuex, which enhance functionality and maintainability. It’s particularly preferred for projects that require a gentle learning curve while still delivering high performance.

\*\*-> Why TypeScript?\*\*

A: TypeScript adds static typing to JavaScript, which helps catch errors during development rather than at runtime. This leads to improved code quality and maintainability, especially in larger applications. It also provides better tooling and autocompletion features in IDEs, enhancing developer productivity. TypeScript is preferred in projects where long-term maintenance and collaboration among multiple developers are essential.

\*\*-> Why REST API?\*\*

A: I opted for a REST API because it’s a well-established architectural style for designing networked applications. REST allows for stateless interactions, which improves scalability and performance. Additionally, it uses standard HTTP methods, making it easy for frontend applications to communicate with the backend. REST APIs are preferred when you need a simple and reliable way to connect various services or clients.

\*\*-> Why PostgreSQL as a database?\*\*

A: PostgreSQL is a powerful open-source relational database known for its robustness and compliance with SQL standards. It supports advanced features like JSONB, which allows for flexible data handling and querying. Additionally, its strong emphasis on data integrity and support for complex queries make it ideal for applications that require reliable data management. PostgreSQL is preferred when dealing with complex data relationships and large datasets.

\*\*-> Biggest challenge during implementation?\*\*

A: One of the biggest challenges was ensuring smooth communication between the frontend and backend, especially with asynchronous operations. Managing state effectively in Vue while handling API calls required careful planning and testing. I also had to optimize performance to ensure a responsive user experience, particularly when dealing with large datasets. This taught me the importance of thorough debugging and performance monitoring.

\*\*-> Impact?\*\*

A: The implementation resulted in a highly responsive web platform that significantly improved user engagement and satisfaction. By leveraging Vue and TypeScript, the development process became more efficient, reducing the overall time to market. Additionally, the clean architecture facilitated easier updates and feature additions, leading to a scalable solution that can grow with user demands.

### 2. Geneweaver - \*\*-> Why REST API?\*\*

A: REST API was chosen for Geneweaver because it provides a clear and consistent way to expose the application’s functionality over the web. This made it easy to integrate various components of the application, such as the frontend interface and backend services. REST’s stateless nature and use of standard HTTP methods allowed for efficient data exchange and simplified the architecture. It was preferred for ensuring that the application could easily interact with other services or platforms.

\*\*-> Impact?\*\*

A: The impact of implementing REST API in Geneweaver was significant. It allowed for seamless data retrieval and manipulation, enhancing the overall user experience. The modular architecture made it easier to develop and maintain the application, allowing for quick iterations based on user feedback. Ultimately, it contributed to the project’s success by ensuring scalability and flexibility for future enhancements.

### 3. eCommerce Website Using the MERN Tech Stack

\*\*-> Why MERN tech stack?\*\*

A: I chose the MERN stack (MongoDB, Express, React, Node.js) for the eCommerce website due to its full-stack JavaScript capabilities, which streamline the development process. Using JavaScript on both the client and server sides allows for easier data handling and faster development cycles. MongoDB’s NoSQL database is excellent for handling dynamic product data, while React provides a highly responsive user interface. This stack is preferred for applications requiring real-time updates and a seamless user experience.

### 4. Spring Boot Website

\*\*-> Why Spring?\*\*

A: Spring is a comprehensive framework that simplifies Java development by providing powerful features for building enterprise-level applications. It promotes good design practices and offers a wide range of functionalities, such as dependency injection and aspect-oriented programming. Spring is preferred for applications that require robustness and scalability, making it a popular choice in the enterprise environment.

\*\*-> Why Spring Boot?\*\*

A: Spring Boot simplifies the process of setting up and configuring Spring applications by providing convention over configuration. It allows developers to get started quickly with minimal setup, making it ideal for microservices and rapid development. The embedded server feature also enables easy deployment, which enhances development efficiency. Spring Boot is preferred when building standalone applications that need to be deployed quickly and efficiently.

\*\*-> Why Spring MVC?\*\*

A: Spring MVC is an essential component of the Spring framework that provides a powerful model-view-controller architecture. It helps separate concerns in web applications, making the codebase easier to manage and maintain. Spring MVC is particularly advantageous for building scalable web applications, as it allows for flexible routing and view rendering. It is preferred when you want to develop complex web applications with a clear structure and organization.

Feel free to adjust any of these responses to better reflect your personal experiences or preferences!