Comprehension Questions

1. What does each letter in “MERN” represent, and how is TypeScript introduced into each component?
2. Which file extensions are typically used for React and Express code in a TypeScript MERN project?
3. In the building-a-house analogy, what do the frontend, backend, and database tiers correspond to?
4. Describe the four-course meal analogy: what role does each MERN component play?
5. According to the factory assembly-line analogy, what are the three main stages of a MERN request flow?
6. How does using TypeScript end-to-end eliminate data format conversions across the stack?
7. List two scenarios or project requirements that make the MERN stack an ideal choice.
8. Compare MERN to a custom full-stack approach in terms of onboarding and code reusability.
9. What advantages do shared TypeScript interfaces and DTOs provide between client and server?
10. Explain how Express.ts (running on Node.ts) and React.tsx collaborate to form a complete TypeScript/JSON application.