🔧 Backend Developer Live Coding Challenges

1. Design a RESTful CRUD API

- **Task**: Build a RESTful API to manage a collection of resources (e.g., books, users, products).
- **Requirements**:
    - Implement endpoints for Create, Read, Update, and Delete operations.
    - Use appropriate HTTP methods and status codes.
    - Integrate with a database (e.g., PostgreSQL, MongoDB).
    - Handle error cases gracefully.
- **Evaluation Criteria**:
    - Adherence to REST principles.
    - Code organization and modularity.
    - Error handling and input validation.
    - Use of ORM or database abstraction layers.

2. Implement Authentication and Authorization

- **Task**: Add user authentication to an existing API.
- **Requirements**:
    - Implement user registration and login functionality.
    - Use JWT or session-based authentication.
    - Protect certain endpoints to be accessible only by authenticated users.
    - Implement role-based access control for specific resources.
- **Evaluation Criteria**:
    - Security best practices (e.g., password hashing, token expiration).
    - Proper middleware usage for protecting routes.
    - Scalability considerations

🎨 **Frontend Developer Live Coding Challenges**

1. Build a Dynamic To-Do List

- **Task**: Create a to-do list application with interactive features.
- **Requirements**:
  - Add, edit, and delete tasks.
  - Mark tasks as completed.
  - Filter tasks based on status (all, active, completed).
  - Persist tasks using local storage.
- **Evaluation Criteria**:
  - State management and component structure.
  - User experience and responsiveness.
  - Code readability and maintainability.

2. Implement a Searchable Data Table

- **Task**: Display tabular data with search and sort functionalities.
- **Requirements**:
  - Fetch data from a public API or mock data source.
  - Implement client-side search filtering.
  - Allow sorting by different columns.
  - Paginate results for better usability.
- **Evaluation Criteria**:
  - Efficient rendering of large datasets.
  - Accessibility considerations.
  - Responsive design implementation.

3. Create a Responsive Dashboard Layout

- **Task**: Design a dashboard interface with multiple widgets.
- **Requirements**:
  - Implement a responsive grid layout.
  - Include various components (e.g., charts, tables, notifications).
  - Ensure compatibility across different screen sizes.
  - Use a CSS framework or custom styling.

- **Evaluation Criteria**:
  - Visual design and layout consistency.
  - Responsiveness and adaptability.
  - Code modularity and reuse of components.

Preparation Tips:

- Familiarize yourself with common frontend frameworks (e.g., React, Vue.js) and backend technologies (e.g., Node.js, Express).
- Practice building full-stack applications to understand end-to-end workflows.
- Review best practices for API design, state management, and responsive design.
- Utilize online coding platforms to simulate live coding environments.