🏆 TECHTONIC Evaluation Report

Problem: Design and implement a web-based Student Management System that allows users to perform basic CRUD (Create, Read, Update, Delete) operations on student records. The frontend should be developed using React and styled with HTML, CSS, and Bootstrap to ensure a clean and responsive user interface. The backend should be built using Express.js, exposing RESTful API endpoints to handle requests and interact with a MySQL database for persistent storage of student information such as name, email, and course. The system should include input validation, modular code organization, and follow best practices in API design and component-based architecture.

Date: 6/5/2025, 4:18:16 pm

Total Score: 63 / 100

Student Solution

I built the frontend using React to show a form and a list of students. I used state hooks to manage input data and the list. I connected it to an Express server which handles requests for adding and getting students using MySQL. I used Bootstrap for basic layout and created separate sections for the form and student list. I also ensured that the form has input validation so users cannot submit empty values. I structured the code to follow a simple component-based approach.

Feedback Details

React UI Components: The submission demonstrates the use of React with separate sections for the form and student list, though the implementation remains basic. **8/10**

RESTful API Design: The Express server handles adding and retrieving students, but lacks full CRUD operations like update and delete. **6/10**

MySQL CRUD Operations: Integration with MySQL is mentioned; however, the operations seem limited to adding and fetching records. **6/10**

Form Validation: The implementation includes basic form input validation to avoid empty submissions, which meets the requirement at a fundamental level. **8/10**

State Management: Use of React state hooks for managing form inputs and listing students shows a good grasp of state management. **8/10**

Code Readability: The code is structured in a component-based manner, though additional clarity through comments and further modularity could improve readability. **7/10**

Responsiveness (CSS): Bootstrap is used for basic layout and styling, indicating awareness of responsive design principles, but details on advanced responsiveness are minimal. **7/10**

Modular Structure: The code is divided into sections for the form and list, demonstrating a modular approach, yet there's room for a more refined structure. **7/10**

Error Handling: There is little evidence of robust error handling in both the frontend or backend, reducing the reliability of the system. **3/10**

Deployment Readiness: The submission does not address deployment configurations or processes, leaving the application less prepared for production environments. **3/10**