TECHTONIC Evaluation Report

Problem: Design and implement a web-based Student Management System

Date: 2025-05-07 07:38

Total Score: 97.0 / 100

Problem Statement

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.

Student Solution

To build the student management system, I divided the UI into reusable React components such as `StudentForm` for input and `StudentList` for displaying data. I used `useState` and `useEffect` for managing state and fetching the initial list of students. For API communication, I used Axios to connect to a RESTful Express.js backend, which performs SQL operations on a MySQL database using parameterized queries. All CRUD routes are implemented cleanly using Express Router. The form includes real-time validation for name and email fields. I styled the application using Bootstrap to ensure responsiveness and a consistent look. I also added error messages for failed operations and organized the project structure to be deployment-ready.

Feedback Details

Category	Feedback	Score
React UI Components	The student clearly demonstrates reusable React components through components like StudentForm and StudentList.	10
RESTful API Design	The use of Axios and Express Router for clean CRUD endpoints meets RESTful design principles effectively.	10
MySQL CRUD Operations	Utilizing parameterized queries for MySQL operations shows proper handling of database interactions.	10
Form Validation	Real-time validation for the name and email fields is adequately covered, showing an awareness of input validation best practices.	10
State Management	The use of useState and useEffect indicates a proper understanding of React state management and lifecycle methods.	10
Code Readability	The description suggests a well-organized codebase, though more evidence or examples of commenting and naming conventions could enhance readability further.	8

Category	Feedback	Score
Responsiveness (CSS)	Styling with Bootstrap ensures a responsive and clean UI that adapts well to various devices.	10
Modular Structure	The project is structured in a modular way with separate components and Express routers, reflecting good architectural practices.	10
Error Handling (API)	Including error messages for failed operations demonstrates basic API error handling, though additional details could further strengthen this aspect.	9
Deployment Readiness	The project has been organized with deployment in mind, suggesting a clear structure and readiness for production use.	10