Student Management System

Course: T101 - Tele Demo Course

Created: June 25, 2025

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.

Knowledge Topics

- MySQL CRUD Operations
- Form Validation
- React State Management
- RESTful API Principles

Scoring Rubric

Assessment Aspect	Marks
Database Schema Design	25 marks
API Endpoint Implementation	30 marks
Frontend Component Architecture	25 marks
Input Validation & Error Handling	20 marks

Knowledge Pills

■ 1. MySQL CRUD Operations

Overview

MySQL CRUD Operations refer to the fundamental database manipulations: Create, Read, Update, and Delete. These operations form the backbone of data management in many applications. In the context of our alternate Library Management System, CRUD operations are essential for handling the lifecycle of book records stored in a MySQL database.

Key Principles and Components

• **Create:** Inserting new book entries into the database.
• **Read:** Retrieving one or multiple book details.
• **Update:** Modifying existing book records.
• **Delete:** Removing unwanted or outdated book entries.

Why MySQL CRUD Operations Matter

These operations ensure that every book record is accurately maintained. They are critical to ensuring data integrity and consistency, which is key when records are constantly added, modified, or removed. The detailed operations support both administrative decisions and user interactions.

Common Applications or Variations

• Basic CRUD in student or book management systems.
• Advanced transactions using stored procedures.
• Integration with ORMs to simplify database tasks.

How This Helps in the Alternate Example

In our Library Management System, CRUD operations allow librarians to add new books, update details, retrieve records for checkout, and delete records of outdated books. This modular approach ensures easy management and scalability of the system.

■ 2. Form Validation

Overview

Form Validation is the process of checking user input to ensure data is correct before it is processed or stored. In web applications like our alternate Library Management System, validating input reduces errors and prevents faulty data from entering the system.

Key Principles and Components

• **Client-Side Validation:** Provides immediate feedback using JavaScript or Angular forms.
• **Server-Side Validation:** Checks data against business rules ensuring security.
• **Regular Expressions:** Validates formats like ISBN, email addresses, and phone numbers.
• **Error Messaging:** Clearly informs users about inputs that need correction.

Why Form Validation Matters

It ensures that only clean and expected data is processed by the system, thereby enhancing the reliability and security of the application. In our Library Management System, it prevents issues such as duplicate book entries and data corruption.

Common Applications or Variations

• Real-time input validation on forms.
• Using validation libraries to standardize rules.
• Customization for specific fields such as ISBN numbers.

How This Helps in the Alternate Example

Effective form validation in the Library Management System guarantees that all book data entered by librarians is accurate, formatted correctly, and complete. This minimizes errors during the CRUD operations and enhances overall user experience.

■ 3. React State Management

Overview

React State Management involves maintaining and updating the data that determines how components render and behave. While our alternate example uses Angular, the principle of state management is pivotal in modern frontend frameworks, and similar practices apply in Angular with services or state containers.

Key Principles and Components

• **Local State:** Manages data within a single component.
• **Global State:** Shares data across multiple components using stores.
• **Immutable Data Structures:** Ensures predictable state changes.
• **Event Handling:** Updates state based on user interactions.

Why State Management Matters

Proper state management ensures that the user interface reflects the current data accurately. In library systems, this is crucial for displaying up-to-date book records, form validation errors, and other dynamic content.

Common Applications or Variations

• Using Redux in React for global state management.
• Leveraging Context API for smaller applications.
• Utilizing services or NgRx in Angular for centralized state management.

How This Helps in the Alternate Example

In the Library Management System, efficient state management enables dynamic updates of book lists, instant validation feedback, and real-time UI changes when CRUD operations are performed. This ensures a responsive and user-friendly experience for librarians.

■ 4. RESTful API Principles

Overview

RESTful API Principles are based on the architectural style of Representational State Transfer, designed to allow communication between client and server over HTTP. In our alternate Library Management System, RESTful APIs are used by the Spring Boot backend to process requests related to book records.

Key Principles and Components

• **Statelessness:** Each API call contains all necessary information.
• **Uniform Interface:** Uses standard HTTP methods like GET, POST, PUT, and DELETE.
• **Client-Server Separation:** The front-end (Angular) communicates with the back-end independently.
• **Resource-Based:** Focuses on resources such as book records.

Why RESTful API Principles Matter

These principles ensure efficient, modular, and scalable communication between the frontend and backend. They provide a framework that is widely understood and implemented, ensuring interoperability across diverse systems.

Common Applications or Variations

• CRUD operations on resources.
• Microservices architecture implementation.
• Integration with third-party services via standardized endpoints.

How This Helps in the Alternate Example

For the Library Management System, RESTful APIs facilitate the interaction between Angular components and Spring Boot services. They allow librarians to perform operations such as adding, updating, and deleting book records seamlessly, improving system performance and usability.

Additional Resources

■ Supporting Document: TECHTONIC_Full_Features_Report.pdf
■ Video Tutorial: https://www.youtube.com/watch?v=HISRUrJsD08

Generated on June 26, 2025 at 03:05 PM