

CRUD OPERATION USING NODE.JS

A Micro Project Report

Submitted by

**M.RAM SUBRAMANIAN
Reg.no: 99220040706**

**B.Tech - COMPUTER SCIENCE AND ENGINEERING,
Specialization- Artificial intelligence and Machine
learning**



**Kalasalingam Academy of Research and Education
(Deemed to be University)**

Anand Nagar, Krishnankoil - 626 126

FEBRUARY 2024



KALASALINGAM
ACADEMY OF RESEARCH AND EDUCATION
(DEEMED TO BE UNIVERSITY)
Under sec. 3 of UGC Act 1956. Accredited by NAAC with "A" Grade



SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Bonafide record of the work done by M.Ram Subramanian – 99220040706 in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Specialization of the Computer Science and Engineering, during the Academic Year – Even Semester (2023-24)

Dr.P.PandiSelvam

Project Guide

Assistant Professor

COMPUTER SCIENCE AND ENGINEERING

**Kalasalingam Academy of Research and
Education
Krishanan Kovil- 626126**

Mr.Gnana Kumar

Faculty In-Charge

Assistant Professor

**COMPUTER SCIENCE AND
ENGINEERING**

**Kalasalingam Academy
of Research and
Education
Krishanan Kovil- 626126**

Mrs.K.Deepa Lakshmi

Evaluator

Assistant Professor

COMPUTER SCIENCE AND ENGINEERING

**Kalasalingam Academy of
Research and Education
Krishnan kovil - 626126**

Abstract

An innovative web-based tool called the Student Data Management System (SDMS) was created to improve and expedite how student records are handled in educational institutions. Created with state-of-the-art technologies, including the MySQL database, the Node.js backend, the Express.js server framework, and body-parser for request parsing, this system offers a complete solution for managing student data via Create, Read, Update, and Delete (CRUD) operations. Enhancing the security, accuracy, and effectiveness of student data management procedures is the main objective of the SDMS. This lowers administrative burden and improves decision-making in educational institutions. Providing an intuitive interface for entering, retrieving, editing, and removing student records, the system guarantees that academic administrators and teachers can access and handle vital student data.knowledge effortlessly. Sensitive student data is protected by strong security mechanisms and data integrity checks, which make the SDMS a significant leap in educational technology. Its scalable architecture makes it an indispensable tool for promoting a more orderly, accessible, and safe learning environment. It is made to satisfy the changing data management needs of educational institutions of all sizes.

Contents

1 Chapter 1 Introduction	1
1.1 Why we need curd operation?.....	1
1.2 About the project.....	3
2 Chapter 2 Methodology	4
2.1 Research Methodology/Tool used.....	4
2.1.1 About Node.js.....	4
2.1.2 About Custom Vision.....	5
2.2 Step-By-Step Procedure.....	6
3 Chapter 3 Implementation	15
3.1 Real world applications.....	15
4 Conclusion and Future Work	17
4.1 Conclusion.....	17
4.2 Future work.....	18
5 References	20
6 Certification	21

Chapter 1

Introduction

1.1 Why we curd operation?

The acronym CRUD, which stands for Create, Read, Update, and Delete, is essential to any software application's data management process and serves as the basis for communicating with database systems. These functions give users and systems the ability to carry out necessary actions such as creating new records, getting data for analysis or presentation, updating current data, and getting rid of undesired or outdated data. Because CRUD activities can guarantee that data is correct, dynamic, and represents the current state of the system or business process they support, they are essential. CRUD activities, for instance, enable the insertion of new products, the viewing of product listings, the editing of product data, and the removal of discontinued items from an e-commerce platform. Without the capacity to execute Unless these activities were carried out effectively, data would become static, which would result in errors, inefficiencies, and a lack of system responsiveness. Therefore, across a variety of application types and business domains, CRUD processes play a critical role in preserving the relevance and integrity of data, which in turn promotes decision-making, operational efficiency, and user delight.

1.2 About the project

The CRUD Operation project uses a combination of Node.js, Express.js, Body-parser, and MySQL to maintain student records in a realistic, hands-on manner. It provides a contemporary response to the long-standing difficulties educational institutions have in effectively and safely handling enormous volumes of student data. The project creates a dynamic and interactive system that makes it easy to add, get, modify, and delete student information by implementing CRUD (Create, Read, Update, Delete) actions. This feature is essential for maintaining data integrity, keeping it current, and limiting access to authorized workers. This improves administrative effectiveness and decision-making in educational settings. The project has a stable and expandable server-side configuration thanks to the integration of Node.js and Express.js, and MySQL offers a dependable student record database, and Body-parser makes it simple to parse request bodies. This project is a useful tool for educational administrators, helping to improve the overall administration of student information and streamline processes. It also provides as an example of how basic web development and database management principles can be applied.

Chapter 2

METHODOLOGY

2.1 Research methodology/Tool used

In order to design, construct, and assess a system that can handle student records effectively, a comprehensive approach is used in the study methodology for CRUD operations using Node.js in the context of student data management. The study begins with a thorough analysis of the body of literature, with an emphasis on modern data management techniques and technologies, Node.js applications, and the principles of CRUD operations. This theoretical underpinning helps to highlight areas in which student data management might be innovatively improved. The next step involves designing a prototype system that integrates a database like MySQL or MongoDB to hold student data and uses Node.js and Express.js for server-side activities. The practical implementation of CRUD operations—creating new student records, reading and retrieving student information, updating current records, and deleting records—is emphasized in this phase.

To guarantee precision, effectiveness, and security in the processing of data, the development process is iterative and includes continuous testing. Both qualitative and quantitative techniques are used to examine data gathered during testing phases, such as operational efficiency, user input, and performance indicators. This examination looks at problems and potential areas for improvement in addition to evaluating the system's efficacy. Scalability, security, and user experience are given top priority in the research process, which aims to provide insightful information about how student data management systems should be optimized. The research's conclusion offers a thorough assessment of the system that was created, recording discoveries, lessons discovered, and suggestions for further research or implementations. By using this methodology, the study seeks to improve the administration and accessibility of educational settings by utilizing Node.js and CRUD processes.

2.1.1 About Node.js

With the help of the robust and adaptable open-source server environment Node.js, programmers can create a variety of JavaScript web apps. Node.js was created with the intention of offering a real-time, high-performance, and scalable network application framework. It was first released in 2009 by Ryan Dahl. With its ability to translate JavaScript code into quicker machine code, the V8 JavaScript runtime engine powers Node.js, which makes it incredibly effective for data-intensive real-time applications that span multiple distributed devices.

Node.js's non-blocking, event-driven architecture, which dispenses with the expense of thread management to handle several connections at once, is one of its key characteristics. Because of this, Node.js is especially well-suited for creating online apps like chat that need a continuous connection from the browser to the server.

2.1.2 About Custom vision

By offering a reliable, scalable, and smooth experience, the bespoke vision for CRUD operations using Node.js in this project seeks to completely transform the management of student data. The project uses Node.js, MySQL for the database, and Express.js for the web framework to provide educational institutions with a quick, easy-to-use platform for managing student records. A responsive web application that guarantees real-time data processing and enables quick upgrades and student information retrieval is part of the concept. A special focus is on the user experience, making sure that administrators and educators can operate the system with ease, access data through a clear, safe interface, and explore it with ease. Tools for creating customized reports, data filtering choices, and advanced search capabilities was incorporated to improve the usability of the system. Using best practices for data encryption, user authentication, and authorization to protect sensitive data is a high priority when it comes to security. Additionally, a great degree of customization is planned for the system, allowing institutions to modify its features to meet their unique needs. This innovative method uses Node.js to develop a dynamic, effective, and safe platform that not only simplifies but also improves student data administration.

2.2.2 Step-By-Step Procedure

Creating a detailed algorithm for CRUD (Create, Read, Update, Delete) operations using Node.js, especially in the context of managing student records in a database like MySQL, involves several steps. This algorithm assumes you have Node.js and MySQL installed, and you have basic knowledge of JavaScript and SQL. Let's outline the steps involved in implementing each CRUD operation.

1) Setting Up Your Node.js Environment

Install Node.js and npm:

Visit the official Node.js website and download the installer for your operating system. The installer includes both Node.js and npm (Node Package Manager), which is used to manage dependencies for Node.js applications.

Initialize a New Node.js Project:

Open your terminal or command prompt.

Navigate to the directory where you want to create your crud operation.

Run `npm init`. This command will prompt you to enter several pieces of information about your project (like the name, version, and description). You can either fill these out or press enter to accept the defaults. This process creates a `package.json` file in your project directory, which tracks your project's dependencies and other configuration.

1. Setup Project:

Initialize a new Node.js project and install necessary packages like `express` for creating the server, `mysql` for database connection, and `body-parser` to parse request bodies.

2. Database Setup:

Create a MySQL database and a table to store student records. Each record might include fields like `id`, `name`, `age`, and `class`.

3. Establish Connection:

Write code to connect your Node.js application to your MySQL database using the `mysql` package.

Create (C):

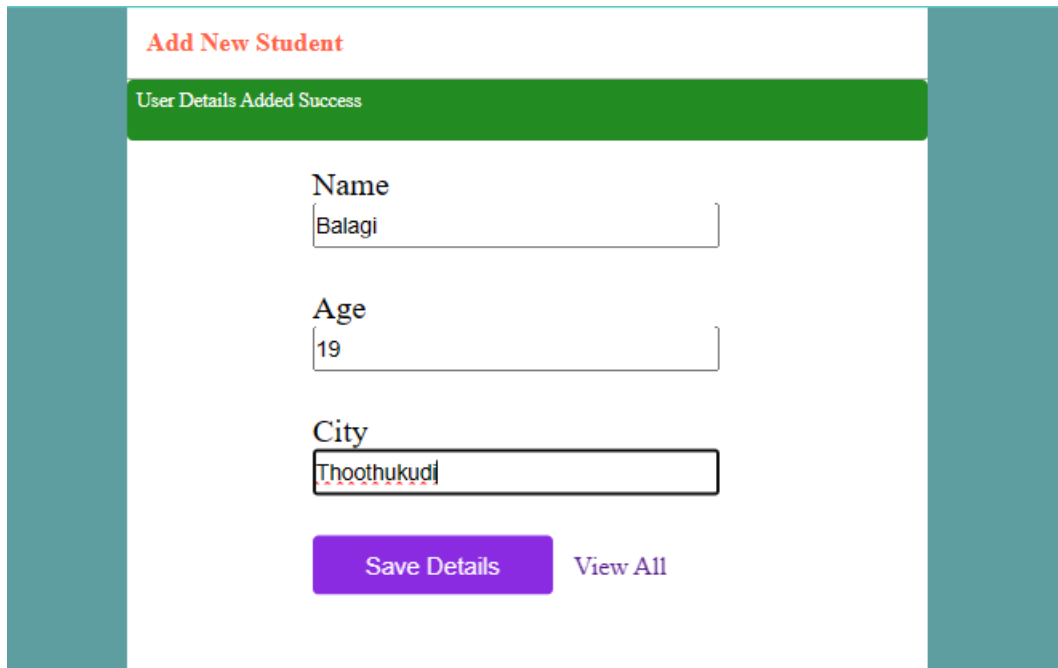
1. Endpoint Creation:

Define an endpoint (e.g., /students) with a POST method where the request body contains the new student data.

2. **Data Extraction:** Use body-parser to extract student data from the request body.

3. **Insert Query:** Construct an SQL query to insert the extracted data into the student table.

4. **Execute Query:** Use the MySQL connection to execute the query, handling any errors and sending an appropriate response back to the client.



The screenshot shows a web interface for adding a new student. At the top, there is a red header bar with the text "Add New Student". Below this is a green success message bar that says "User Details Added Success". The form contains three input fields: "Name" with the value "Balagi", "Age" with the value "19", and "City" with the value "Thoothukudi". At the bottom of the form, there are two buttons: a purple "Save Details" button and a blue "View All" button.

Read (R):

1. **Endpoint for Retrieval:** Define an endpoint (e.g., /students) with a GET method to retrieve student records. Optionally, accept query parameters for filtering.

2. **Construct Query:** Based on the request, construct an SQL query to select records from the student table. If filters are provided, include WHERE clauses accordingly.

3. **Execute and Respond:** Execute the query and send the retrieved records back to the client in the

response.

Student Management System					
					Add User
#	Name	Age	City	Edit	Delete
7	M Ram Subramanian	19	Tirunelveli	Edit	Delete
8	afsar jameel	19	Tenkasi	Edit	Delete
9	Harri	20	Sankankovil	Edit	Delete
10	SURYAK	20	Tenkasi	Edit	Delete
11	nithish	20	tenka	Edit	Delete
12	Balagi	19	Thoothukudi	Edit	Delete

Update (U):

1. **Endpoint for Update:** Define an endpoint (e.g., /students/:id) with a PUT method, where :id is the identifier of the student to update.
2. **Extract Data:** Use body-parser to extract the updated student data from the request body.
3. **Update Query:** Construct an SQL query to update the specific record in the student table based on the provided ID and data.
4. **Execute Query:** Execute the query and handle the response, indicating success or failure.

Edit Student Details

Name

M Ram Subramanian

Age

19

City

Tirunelveli

Save Changes

View All

Delete (D):

- Endpoint for Deletion:** Define an endpoint (e.g., /students/:id) with a DELETE method, where :id is the identifier of the student to delete.
- Delete Query:** Construct an SQL query to delete the specific record from the student table based on the provided ID.

OUPUT:

Student Management System					
					Add User
#	Name	Age	City	Edit	Delete
7	M Ram Subramanian	19	Tirunelveli	Edit	Delete
8	afsar jameel	19	Tenkasi	Edit	Delete
9	Harri	20	Sankankovil	Edit	Delete
10	SURYAK	20	Tenkasi	Edit	Delete
11	nithish	20	tenka	Edit	Delete
12	Balagi	19	Thoothukudi	Edit	Delete

Chapter 3

Implementation

1.1 Realtime applications

A real-time application for CRUD operations entails the development of a dynamic system that enables users to interact with data in near-instantaneous fashion, with updates reflected across all connected clients in real-time. Leveraging technologies such as Node.js, Express.js, WebSocket, and a NoSQL database like MongoDB, this application would offer a seamless and responsive user experience.

In this application, as users create, read, update, or delete data, such as student records, changes are immediately propagated to all connected clients without the need for manual refreshes. For instance, when a new student record is added by one user, all other users viewing the same page would instantly see the new entry appear without needing to reload the page. Similarly, if a student record is updated or deleted, those changes would be reflected across all connected clients in real-time.

This real-time functionality is achieved through the use of WebSocket technology, which establishes a persistent connection between the server and clients, allowing for bi-directional communication. When a CRUD operation is performed, the server pushes the updates to all connected clients, ensuring that everyone has the most up-to-date information at all times.

Furthermore, the application can incorporate features like notifications or alerts to inform users of changes made by others in real-time, enhancing collaboration and communication among users. Additionally, conflict resolution mechanisms may be implemented to handle scenarios where multiple users attempt to update the same data simultaneously.

Overall, a real-time CRUD application offers a highly interactive and collaborative environment, ideal for scenarios where data changes frequently and users need immediate access to the most current information. Whether it's in educational settings, project management, or collaborative document editing, real-time CRUD applications revolutionize the way users interact with data, making workflows more efficient and responsive.

Chapter 4

Conclusion and Future work

4.1 Conclusion

In conclusion, leveraging CRUD operations within a Node.js environment provides a robust and efficient framework for managing data, such as student records, in web applications. Through the use of Node.js alongside Express.js for routing, a database like MySQL or MongoDB for storage, and potentially real-time communication technologies like WebSocket, developers can build dynamic, responsive, and scalable applications. CRUD operations form the backbone of these applications, allowing for the creation, retrieval, updating, and deletion of data in a structured and systematic manner. This not only enhances the user experience by ensuring data accuracy and immediacy but also significantly improves backend efficiency. Moreover, the event-driven, non-blocking nature of Node.js makes it particularly well-suited for handling the demands of modern, data-intensive applications that require real-time functionality and high levels of concurrency. Implementing CRUD operations in such an environment encourages best practices in software design, promotes data integrity, and enables developers to tackle complex challenges with more straightforward, maintainable solutions. Thus, the integration of CRUD operations with Node.js stands as a testament to the power and versatility of modern web development techniques, driving forward innovations in data management and application development.

4.2 Future work

The future of CRUD operations using Node.js is poised for significant advancements, integrating emerging technologies and methodologies to further enhance functionality, scalability, and user experience. One promising direction is the incorporation of AI and machine learning algorithms to intelligently handle data, predict user behavior, and automate routine tasks, such as data categorization and anomaly detection. This integration could vastly improve data management systems, making them more intuitive and efficient. Additionally, the adoption of GraphQL over traditional REST APIs presents an opportunity to streamline data retrieval processes in CRUD operations, allowing for more flexible and efficient data queries that can reduce bandwidth usage and improve performance.

Moreover, the evolution of real-time data processing and IoT (Internet of Things) integration opens new avenues for CRUD operations in Node.js applications. By harnessing real-time data streams, applications can offer more dynamic and responsive user experiences, particularly in sectors like e-commerce, gaming, and social media. In the realm of IoT, Node.js could play a crucial role in managing and interacting with the vast amounts of data generated by connected devices, offering new possibilities for automation and smart device management.

In conclusion, the future work for CRUD operations using Node.js is set to embrace technological advancements and emerging trends, focusing on making data management systems more intelligent, efficient, and secure. By staying at the forefront of these developments, Node.js will continue to be a vital tool for developers, enabling them to build innovative applications that meet the evolving needs of users and businesses alike.

Chapter 5

References

1. Youtube

<https://www.youtube.com/watch?v=G0jO8kUrg->

[I&pp=ygUlY2ltcGxllHRvZG8gbGlzdCBwcm9qZWNoIHVzaW5nIG5vZGVqcw%3D%3D](https://www.youtube.com/watch?v=G0jO8kUrg-I&pp=ygUlY2ltcGxllHRvZG8gbGlzdCBwcm9qZWNoIHVzaW5nIG5vZGVqcw%3D%3D)

2. Various websites

<https://www.geeksforgeeks.org/how-to-make-to-do-list-using-nodejs/>

<https://github.com/topics/todo-nodejs>

<https://support.glitch.com/t/how-to-create-a-todo-list-in-node-js/65098>

<https://www.tatvasoft.com/blog/node-js-best-practices/>

<https://www.freecodecamp.org/news/how-to-build-a-todo-app-with-react-typescript-nodejs-and-mongodb/>

Chapter 6

Certification

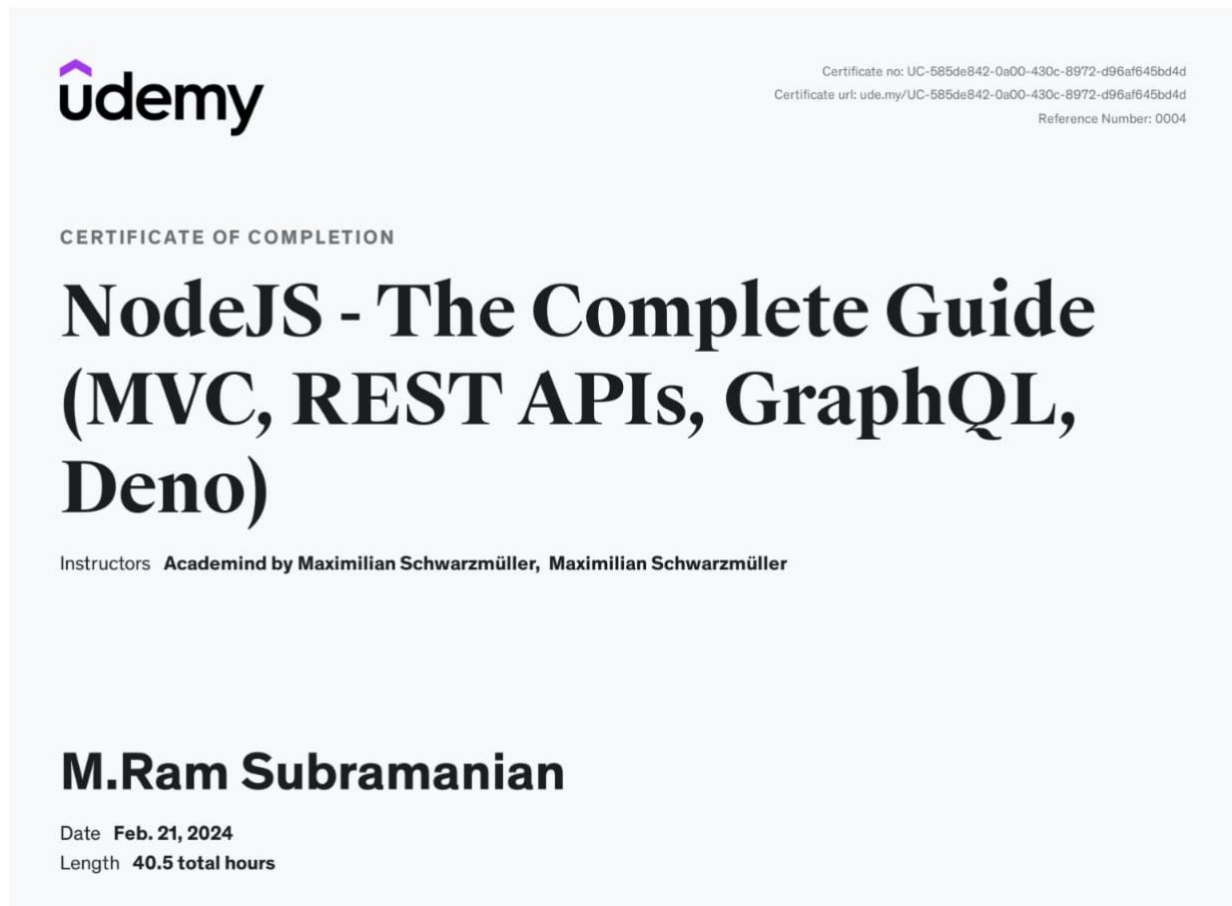


Figure 6.1: Certification details