CRUD OPERATION USING NODE.JS

INTRODUCTION

CRUD operations are the foundation of most web applications, enabling the management of data by allowing users to create, read, update, and delete information. In the context of Node.js, a popular JavaScript runtime environment, these operations are key to developing dynamic, responsive web applications. Node.js's non-blocking, event-driven architecture makes it especially suited for data-driven tasks, providing a scalable and efficient backend solution

ABOUT SOFTWARE TOOL

- Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser.
- Express.js is a web application framework for Node.js, designed for building web applications and APIs. It is known for its performance and minimalistic structure.

OBJECTIVE

The objective of utilizing CRUD operations with Node.js is to enable efficient data management in web applications, leveraging Node.js's event-driven architecture for responsive and scalable app performance. This approach aims to streamline the creation, retrieval, updating, and deletion of data, enhancing user interaction and simplifying development through JavaScript's full-stack potential, ensuring applications are robust, maintainable, and user- friendly.

TIMELINE OF WORK PROPOSAL

18.01.2024 TO 22.03.24	**Udmey course completion**
08.03.24 TO 22.03.24	**Project-Crud Operation using Node.js**
20.03.24 TO 24.03.24	**PPT and Report**

ALGORITHM USED

The core steps involved in processing each of the CRUD operations: Create, Read, Update, and Delete. Here's how you can structure it:

1. Create Action:

User submits a form to add new data.
Flow: User → Submit Form → Server (Node.js) processes POST request → Database (INSERT).

Result: New record created in the database.

2. Read Action:

User requests to view data.
Flow: User → Request Data → Server processes GET request → Database (SELECT) → Data returned to User.

Result: Requested data is displayed to the user.

3. Update Action:

User submits a form to update existing data.Flow: User → Submit Update Form → Server processes PUT/PATCH request → Database (UPDATE).Result: Existing record is updated in the database.

4. Delete Action:

User requests to delete data.Flow: User → Request Delete → Server processes DELETE request → Database (DELETE).Result: Specified record is deleted from the database.

STEPS TO BUILD APPLICATION

1.Setting Up Your Node.js Environment:

- Install Node.js and npm (Node package manager) from the official Node.js website.
- Initialize a new Node.js project and create a package.json file by running npm init.

Initialize a Node.js Project: Run npm init in your project directory to create a package.json file.Install Express.js: Use npm install express for your web server framework.

2. Install Additional DependenciesDatabase Client:

Depending on your database (e.g., MongoDB, MySQL), install the appropriate client (e.g., npm install mongoose for MongoDB).Body Parser Middleware: For parsing incoming request bodies, install with npm install body-parser.

3. Configure Your Server:

Create a server file (e.g., server.js) and set up Express to listen on a port. Use middleware like body-parser for request processing.

STEPS TO BUILD APPLICATION

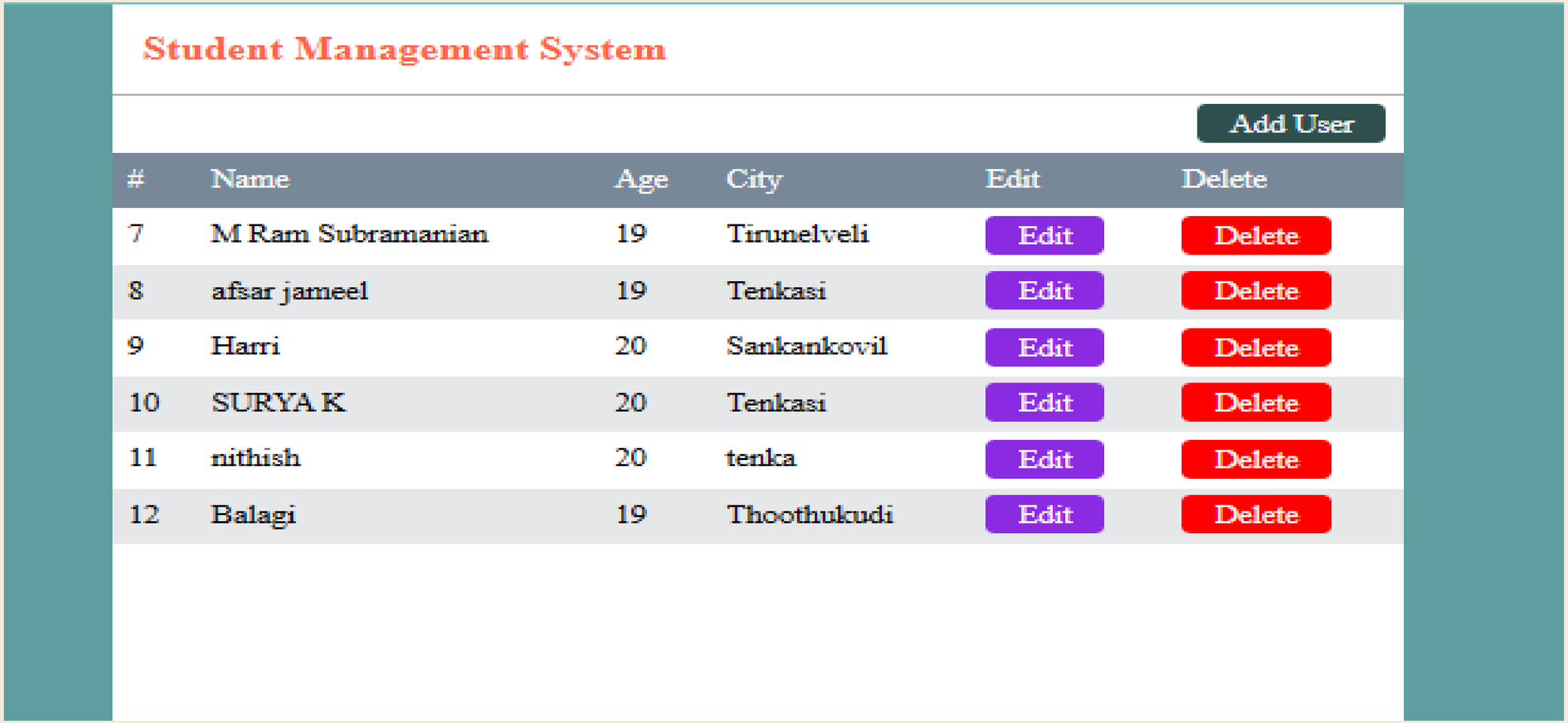
4. **Connect to the Database:**
In your server file or a separate module, establish a connection to your database using the client library you installed.

5. Define Models:
Define data models or schemas that represent the structure of the data in your database.

6.Implement CRUD Routes;
Create Route: Set up a POST route to add new items to the database.Read Routes: Implement GET routes to retrieve one or all items from the database.Update Route: Create a PUT or PATCH route to modify existing items in the database.Delete Route: Establish a DELETE route to remove items from the database.

7. Test Your API;
Use tools like Postman or curl to test each of your CRUD routes and ensure they work as expected.

RESULT AND DISCUSSION



CONCLUSION

The CRUD operation project utilizing Node.js and Express.js has successfully demonstrated the foundational elements of building a dynamic web application capable of handling data persistence and manipulation. This project underscores the versatility and efficiency of Node.js as a server-side platform, combined with the Express.js framework, to manage database interactions for creating, reading, updating, and deleting records. By achieving a functional data management system, the project lays the groundwork for further development and scalability. This endeavor not only enhances our understanding of web development concepts but also opens avenues for incorporating advanced features and technologies in future projects.

REPORTED LITERATURE

Flanagan, D. (2020). JavaScript: The Definitive Guide. O'Reilly Media.

Cantelon, M., Harter, M., Holowaychuk, T., & Rajlich, N. (2017). Node.js in Action. Manning Publications.

.Holmes, B. (2018). Beginning Node.js, Express & MongoDB Development. Independently published.

Niazi, A. (2019). "Integrating NoSQL Databases in Node.js Applications." Journal of Web Engineering, 18(7), 525-543. This article discusses the benefits and methodologies of incorporating NoSQL databases into Node.js applications, with a focus on CRUD operations.