**Project Title**

**CALCULATOR**

**Project Overview**

**Introduction:**

The CALCULATOR project is designed to handle simple addition operations. The primary purpose of this project is to provide back-end service that allows client to store the history of their calculation within a database.

* The main purpose of this project is to build a robust and scalable API that performs addition operation on two numbers within an application.
* Many modern applications require a way to store the history of their calculation into a database on the application. Also, this application can be used without the internet as the data is stored in a local database file.

**Technical Details**

**Technical Stack:**

Backend: Node.js

Framework: Express.js

Database: SQLite

Other Technologies: View Engine, EJS

**Architecture:**

There are basically 4 layers of the CALCULATOR Architecture:

1. **Client Layer:** In thisthe client sends the HTTP requests (GET, POST, PUT, DELETE) to interact with the API.
2. **Router Layer:** This layer is responsible to handle the client requests to the appropriate controller functions.
3. **Controller Layer:** Handles the core logic for addition operations, interacting with the database through the models and sending responses back to the client.
4. **Database Layer:** Stores and manages data.

**Installation**

* **Node.js Installation:** Ensure that Node.js is installed. (Check using **node -v** command).
* **Project Setup:** Create a directory using command **mkdir calculator** and go change the directory using **cd calculator.**
* **Initializing Project:** Initialize node.js project using command **npm init -y** which creates a file package.json.
* **Packages Installation:** Install necessary packages (ejs, express, sqlite3, nodemon) using command **npm i ejs express sqlite3 nodemon.**
* **ejs:** To perform the server side rendering of frontend using ejs view engine.
* **Express:** A web framework for handling HTTP requests.
* **Sqlite3:** File-based RDBMS.
* **Nodemon:** For restart the server automatically when the code has changes to save.
* **Configuration:** Create **app.js** to configure the Express server.
* **Running the Application:** Start the application using the command **npm start**.

**Conclusion**

The CALCULATOR project using NodeJS successfully implemented for addition operations. By utilizing Express for routing client’s HTTP requests and SQLite for data management, a scalable and maintainable architecture was achieved.

**Key Objectives:**

* Build a working API that handles data operations.
* Organized the code by separating different tasks.
* Used SQLite to save the data efficiently.

**Lessons Learned:**

* Breaking the code into small parts makes it easier to manage.
* Handling errors properly is important for smoother operations.
* Improved skills in working with databases like SQLite.

**Future Enhancements:**

* Adding all the other operations of a calculator like subtraction, multiplication, etc.

**Contact Information**

**Project Lead:** Nikunj Bansal

**Email:**nikunj.banssal@gmail.com

**GitHub:** bnikunj31