**CRM Application**

Backend: Node.js with Express

Database: PostgreSQL

Frontend: Index.pug

1. **Project Initialization and Dependency Installation:**
   * Initialize **package.json** using **npm init**.
   * Install required dependencies (**express**, **PostgreSQL**, **dotenv**, **cors**, **nodemon**) using **npm install**.
2. **Module Imports and Environment Setup:**
   * Import necessary modules (**http**, **dotenv**, **path**, **cors**, **express**) into **server.js**.
   * Create **.env** file for environment variables (e.g., server port, database credentials).
3. **Database Setup:**
   * Create a PostgreSQL database named **postgres**.
   * Configure database settings such as host, port, username, and password.
   * Create a table named **MyTable** with columns (**id**, **name**, **email**, **phonenumber**) using SQL queries.
4. **SQL Queries and Database Operations:**
   * Use SQL queries (**INSERT**, **DELETE**, **UPDATE**, **SELECT**) to manipulate data in the database.
   * Implement functions in **server.js** for performing these operations.
5. **Connection Initialization:**
   * Initialize a connection to the PostgreSQL database using the **pg** module.
   * Ensure the connection is established when the server starts (**npm start**).
6. **Backend Functionality:**
   * Implement functions (**select**, **insert**, **update**, **delete**) in **server.js** to handle database operations.
   * Test each function to ensure it successfully executes the corresponding SQL query.
7. **Frontend Integration:**
   * Create an **index.pug** file for the frontend.
   * Implement buttons for displaying customers, deleting customers, and updating customers.
8. **API Endpoints:**
   * Use GET API for homepage (frontend) and POST for insert, delete, and update operations (backend).
9. **Data Display on Frontend:**
   * When the frontend is loaded, run a select query to fetch all data from the database and display it in a dynamic table.
10. **Form Submission and Data Insertion:**
    * Use a form with input boxes to add details.
    * When the "Add User" button is pressed, execute the backend query to insert the given data into the database.
11. **Success Message Handling:**
    * If data entry is successful, send a string "User added" to indicate successful addition.
    * Redirect back to the localhost homepage to see the new entry in the table.
12. **Future Development:**
    * Implement delete and update functionality in the frontend.
    * Backend queries and functions are already written to accommodate them.

**Accessing the Application**

* Access the frontend using **localhost:8080**.