Step 1: Set Up Your Environment

1. Initialize a new Node.js project:

“bash

Mkdir crud-api

Cd crud-api

Npm init -y

“

2. Install necessary packages:

“ bash

Npm install express sqlite3 body-parser

“

Step 2: Create the API Server

1. Create a file named `server.js`:

“javascript

Const express = require(‘express’);

Const bodyParser = require(‘body-parser’);

Const sqlite3 = require(‘sqlite3’).verbose();

Const app = express();

Const port = 3000;

```
App.use(bodyParser.json());

// Initialize the database
Const db = new sqlite3.Database(‘:memory:’);

Db.serialize(() => {
  Db.run(“CREATE TABLE users (id INTEGER PRIMARY KEY, name TEXT, email TEXT)”);
});

// Create a new user
App.post(‘/users’, (req, res) => {
  Const { name, email } = req.body;
  Db.run(“INSERT INTO users (name, email) VALUES (?, ?)”, [name, email],
function(err) {
    If (err) {
      Return res.status(400).json({ error: err.message });
    }
    Res.json({ id: this.lastID });
  });
});

// Read all users
App.get(‘/users’, (req, res) => {
  Db.all(“SELECT * FROM users”, [], (err, rows) => {
    If (err) {
      Return res.status(400).json({ error: err.message });
    }
```

```
    Res.json({ users: rows });
  });
});

// Read a single user
App.get('/users/:id', (req, res) => {
  Const { id } = req.params;
  Db.get("SELECT * FROM users WHERE id = ?", [id], (err, row) => {
    If (err) {
      Return res.status(400).json({ error: err.message });
    }
    Res.json({ user: row });
  });
});

// Update a user
App.put('/users/:id', (req, res) => {
  Const { id } = req.params;
  Const { name, email } = req.body;
  Db.run("UPDATE users SET name = ?, email = ? WHERE id = ?", [name, email, id],
function(err) {
    If (err) {
      Return res.status(400).json({ error: err.message });
    }
    Res.json({ updated: this.changes });
  });
});
```

```
// Delete a user
App.delete('/users/:id', (req, res) => {
  Const { id } = req.params;
  Db.run("DELETE FROM users WHERE id = ?", [id], function(err) {
    If (err) {
      Return res.status(400).json({ error: err.message });
    }
    Res.json({ deleted: this.changes });
  });
});

App.listen(port, () => {
  Console.log(`Server running at http://localhost:${port}/`);
});
"
```

Step 3: Run the Server

1. Start the server:

```
"bash
Node server.js
"
```

Explanation

1. Dependencies:
   - `express`: A web framework for Node.js.
   - `body-parser`: Middleware to parse JSON request bodies.

- `sqlite3`: SQLite database.

2. Database Initialization:

- An in-memory SQLite database is used for simplicity.
- A `users` table is created with `id`, `name`, and `email` fields.

3. CRUD Routes:

- `POST /users`: Creates a new user.
- `GET /users`: Retrieves all users.
- `GET /users/:id`: Retrieves a single user by `id`.
- `PUT /users/:id`: Updates a user's `name` and `email` by `id`.
- `DELETE /users/:id`: Deletes a user by `id`.

This example provides a basic framework for a CRUD API. For a production system, consider using a persistent database and adding more robust error handling, validation, and security measures.