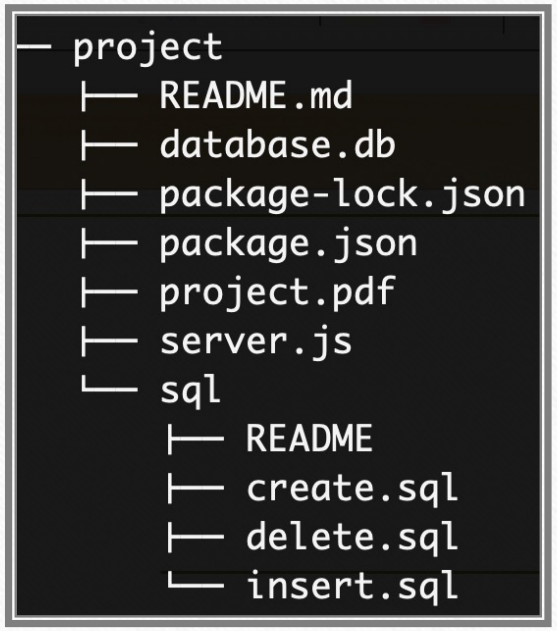
NodeJS Backend Project

Project structure

- README.md: Contains information about the project and instructions.
- database.db: The SQLite database file.
- package-lock.json & package.json: Manage project dependencies.
- project.pdf: The project presentation.
- server.js: The main server file.
- sql/: Contains SQL scripts for creating, deleting, and inserting data into the database.



Main functionalities

- Create, read, update, and delete data in a SQLite database via a Node.js server.
- The server processes HTTP GET requests to retrieve data, POST requests to add data, UPDATE requests to modify data, and DELETE requests to remove data.
- It also handles errors and sends responses in JSON format.

Technologies Used

- Node.js
- SQLite
- Express.js

Challenges

1. Setting up and managing the SQLite database.
2. Implementing the HTTP methods with proper error handling.
3. Ensuring the server responses are in the correct JSON format.

Solutions

1. Express.js middleware for error handling
   Use 'app.use()' to define a middleware function that checks for errors and responds with the appropriate HTTP status code and message.

2. Express.js ‘res.json()’ method
   It automatically sends responses as JSON.

Timeline

1. Create necessary scripts to setup SQLite database
2. Connect to the SQLite database from the Node.js application
3. Implement GET HTTP request to retrieve data from the database
4. Implement POST HTTP request to add data to the database
5. Implement PUT HTTP request to update data in the database
6. Implement DELETE HTTP request to remove data from the database
7. Implement error handling for all HTTP methods
8. Test all endpoints and ensure they work as expected
9. Write documentation for the project, including setup and usage instructions
10. Step 1-9 done on 16th Nov, but Modification on 5th, 12th, 19th Dec.

Code

Create a basic Express server connected to a SQLite database

```
1  const express = require("express"); // Import Express
2  const app = express(); // Initialize Express app
3  app.use(express.json()); // Enable to parse JSON body in POST requests
4
5  const sqlite3 = require("sqlite3").verbose(); // Import SQLite
6  let db = new sqlite3.Database("./database.db"); // Connect to SQLite database
7  const table = "Clothes"; // Define table name
8
```

```
194  app.listen(8080, () => {
195    // Start server
196    console.log(`Server is running on http://localhost:8080 ...`);
197  });
```

Define variables in the beginning

```
9   //// Define HTTP status codes
10  // The requested resource resides temporarily under a different URI
11  const HTTP_STATUS_FOUND = 302;
12  // OK status, request succeeded
13  const HTTP_STATUS_OK = 200;
14  // Request succeeded and a new resource was created as a result
15  const HTTP_STATUS_CREATED = 201;
16  // Client sent an invalid request
17  const HTTP_STATUS_BAD_REQUEST = 400;
18  // Server encountered an unexpected condition which prevented it from fulfilling the request
19  const HTTP_STATUS_INTERNAL_SERVER_ERROR = 500;
20
```

Define variables in the beginning

```
//// Define SQL queries
// SQL query to select all records from table
const SELECT_ALL = `SELECT * FROM ${table}`;
// SQL query to select record from table by id
const SELECT_ID = `SELECT * FROM ${table} WHERE id = ?`;
// SQL query to find records by name
const SEARCH_BY_NAME = `SELECT * FROM ${table} WHERE name LIKE ?`;
// SQL query to find records by size and color
const SEARCH_BY_SIZE_AND_COLOR = `SELECT * FROM ${table} WHERE LOWER(size) = LOWER(?) AND LOWER(color) = LOWER(?)`;
// SQL query to insert new record into table
const INSERT_INTO = `INSERT INTO ${table} (name, type, size, color, price) VALUES (?, ?, ?, ?, ?)`;
// SQL query to update a record in the table
const UPDATE = `UPDATE ${table} SET name = ?, type = ?, size = ?, color = ?, price = ? WHERE id = ?`;
// SQL query to delete a record from the table by id
const DELETE = `DELETE FROM ${table} WHERE id = ?`;

```

Retrieve existing records (HTTP GET method)

```
37  app.get("/", (req, res) => {
38    // Root endpoint
39    res.status(HTTP_STATUS_FOUND).redirect("/clothes"); // Redirect to '/clothes'
40  });
41
42  app.get("/clothes", (req, res) => {
43    // GET endpoint for all records
44    db.all(SELECT_ALL, [], (err, rows) => {
45      // Execute SQL query
46      if (err) {
47        // If error occurs
48        res
49          .status(HTTP_STATUS_INTERNAL_SERVER_ERROR)
50          .json({ error: err.message, code: err.code }); // Send error response
51        return;
52      }
53      res.json(rows); // Send response with all rows
54    });
55  });
56
```

Retrieve a specific record (HTTP GET method)

```
57  app.get(`/clothes/:id`, (req, res) => {
58    // GET endpoint for a specific record
59    const values = [req.params.id]; // ID parameter from URL
60
61    db.get(SELECT_ID, values, (err, row) => {
62      // Execute SQL query
63      if (err) {
64        // If error occurs
65        res
66          .status(HTTP_STATUS_INTERNAL_SERVER_ERROR)
67          .json({ error: err.message, code: err.code }); // Send error response
68        return;
69      }
70      res.json(row); // Send response with the row
71    });
72  });
73
```

Search item(s) by name (HTTP GET method)

```
74  // Endpoint to handle GET requests to '/search'
75  app.get(`/search`, (req, res) => {
76    // Check if name query parameter is given
77    if (req.query.name) {
78      // Store query parameter in an array
79      const values = [`%${req.query.name}%`];
80
81      // Run SQL query to search by name
82      db.all(SEARCH_BY_NAME, values, (err, rows) => {
83        // If there is an error, send error response
84        if (err) {
85          res
86            .status(HTTP_STATUS_INTERNAL_SERVER_ERROR)
87            .json({ error: err.message, code: err.code }); // Send error response
88          return;
89        }
90        // If no error, send rows returned by the query as response
91        res.json(rows);
92      });
93    }
```

Search item(s) by size and color (HTTP GET method)

```
94   // Check if both size and color query parameters are given
95   else if (req.query.size && req.query.color) {
96     // Store query parameters in an array
97     const values = [req.query.size, req.query.color];
98
99     // Run SQL query to search by size and color
100    db.all(SEARCH_BY_SIZE_AND_COLOR, values, (err, rows) => {
101      // If there is an error, send error response
102      if (err) {
103        res
104          .status(HTTP_STATUS_INTERNAL_SERVER_ERROR)
105          .json({ error: err.message, code: err.code }); // Send error response
106        return;
107      }
108      // If no error, send rows returned by the query as response
109      res.json(rows);
110    });
111  }
112
113  // If neither size and color nor name query parameters are given, send error message
114  else {
115    res
116      .status(HTTP_STATUS_BAD_REQUEST)
117      .json({ error: "Invalid query parameters", code: "400" }); // Send custom error response
118  }
119 });
120
```

Create a new record (HTTP POST method)

```
121  app.post(`/clothes`, (req, res) => {
122    // POST endpoint
123    const values = [
124      req.body.name, // Values from request body
125      req.body.type,
126      req.body.size,
127      req.body.color,
128      req.body.price,
129    ];
130
131    db.run(INSERT_INTO, values, function (err) {
132      // Execute SQL query
133      if (err) {
134        // If error occurs, return 500 status (Internal Server Error) and error details
135        res
136          .status(HTTP_STATUS_INTERNAL_SERVER_ERROR)
137          .json({ error: err.message, code: err.code }); // Send error response
138        return;
139      }
140      // If no error, return 201 status (Created) and the ID of the newly created record
141      res
142        .status(HTTP_STATUS_CREATED)
143        .json({ message: "Row added", ID: this.lastID });
144    });
145  });
146
```

Update an existing record (HTTP PUT method)

```
147  app.put(`/clothes/:id`, (req, res) => {
148    // PUT endpoint for updating a record
149    const values = [
150      req.body.name, // Values from request body
151      req.body.type,
152      req.body.size,
153      req.body.color,
154      req.body.price,
155      req.params.id, // ID from URL
156    ];
157
158    db.run(UPDATE, values, function (err) {
159      // Execute SQL query
160      if (err) {
161        // If error occurs, return 500 status (Internal Server Error) and error details
162        res
163          .status(HTTP_STATUS_INTERNAL_SERVER_ERROR)
164          .json({ error: err.message, code: err.code }); // Send error response
165        return;
166      }
167      // If no error, return 200 status (OK) and a message indicating the update was successful
168      res
169        .status(HTTP_STATUS_OK)
170        .json({ message: "Row updated", changes: this.changes });
171    });
172  });
173
```

Delete an existing record (HTTP DELETE method)

```
174  app.delete(`/clothes/:id`, (req, res) => {
175    // DELETE endpoint for deleting a record
176    const values = [req.params.id]; // ID from URL
177
178    db.run(DELETE, values, function (err) {
179      // Execute SQL query
180      if (err) {
181        // If error occurs, return 500 status (Internal Server Error) and error details
182        res
183          .status(HTTP_STATUS_INTERNAL_SERVER_ERROR)
184          .json({ error: err.message, code: err.code }); // Send error response
185        return;
186      }
187      // If no error, return 200 status (OK) and a message indicating the deletion was successful
188      res
189        .status(HTTP_STATUS_OK)
190        .json({ message: "Row deleted", changes: this.changes });
191    });
192  });
193
```

Test calls to backend

GET all clothing items

```
(base) jingjingyang@jingjings-MacBook-Pro ~ % curl -i -s http://localhost:8080

HTTP/1.1 302 Found
X-Powered-By: Express
Location: /clothes
Vary: Accept
Content-Type: text/plain; charset=utf-8
Content-Length: 30
Date: Tue, 19 Dec 2023 23:21:44 GMT
Connection: keep-alive
Keep-Alive: timeout=5

Found. Redirecting to /clothes%
```

GET all clothing items

```
(base) jingjingyang@jingjings-MacBook-Pro ~ % curl -i -s http://localhost:8080/c
lothes

HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 357
ETag: W/"165-20EAdHwAIDZ0E8V9O2NHHPCAnwE"
Date: Tue, 19 Dec 2023 23:22:12 GMT
Connection: keep-alive
Keep-Alive: timeout=5

[{"id":1,"name":"Denim Jacket","type":"Jacket","size":"M","color":"Blue","price"
:59.99},{"id":2,"name":"Leather Boots","type":"Shoes","size":"8","color":"Black"
,"price":89.99},{"id":3,"name":"Floral Dress","type":"Dress","size":"S","color":
"Multicolor","price":45.99},{"id":4,"name":"Cotton T-Shirt","type":"Shirt","size
":"L","color":"White","price":15.99}]%
```

GET a specific item

```
(base) jingjingyang@jingjings-MacBook-Pro ~ % curl -i -s http://localhost:8080/c
lothes/1

HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 86
ETag: W/"56-aGu82n/Q/4/aqfNMoVuUmqfb98w"
Date: Tue, 19 Dec 2023 23:22:40 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"id":1,"name":"Denim Jacket","type":"Jacket","size":"M","color":"Blue","price":
59.99}%
```

GET a non-existing resource

(Content-Length: 0)

```
(base) jingjingyang@jingjings-MacBook-Pro ~ % curl -i -s http://localhost:8080/c
lothes/111

HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Date: Tue, 19 Dec 2023 23:23:15 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 0
```

Search item(s) by name

```
(base) jingjingyang@jingjings-MacBook-Pro ~ % curl -i -s 'http://localhost:8080/search?name=dress'
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 93
ETag: W/"5d-CNih251ungQb6/x0LXQk1Sh/R6M"
Date: Tue, 19 Dec 2023 23:23:55 GMT
Connection: keep-alive
Keep-Alive: timeout=5

[{"id":3,"name":"Floral Dress","type":"Dress","size":"S","color":"Multicolor","price":45.99}]%
```

Search item(s) by size and color

```
(base) jingjingyang@jingjings-MacBook-Pro ~ % curl -i -s 'http://localhost:8080/
search?size=8&color=black'
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 89
ETag: W/"59-58RkYFIrHHhfDx9i2SR5O7yYXs8"
Date: Tue, 19 Dec 2023 23:24:35 GMT
Connection: keep-alive
Keep-Alive: timeout=5

[{"id":2,"name":"Leather Boots","type":"Shoes","size":"8","color":"Black","price
":89.99}]%
```

Search item(s) by invalid params
(HTTP_STATUS_BAD_REQUEST = 400)

```
(base) jingjingyang@jingjings-MacBook-Pro ~ % curl -i -s 'http://localhost:8080/
search?type=shoes'
HTTP/1.1 400 Bad Request
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 49
ETag: W/"31-POaEPDAd+UuhsWVz6t2X28RVWzE"
Date: Tue, 19 Dec 2023 23:28:21 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"error":"Invalid query parameters","code":"400"}%
```

POST a new item

```
(base) jingjingyang@jingjings-MacBook-Pro ~ % curl -i -s -X POST -H "Content-Typ
e: application/json" -d '{"name":"Wool Sweater","type":"Sweater","size":"L","col
or":"Red","price":60.99}' http://localhost:8080/clothes
HTTP/1.1 201 Created
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 30
ETag: W/"1e-xenpjjVRWCDVYZnFJEsKzbedMY4"
Date: Tue, 19 Dec 2023 23:32:15 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"message":"Row added","ID":5}%
```

POST with invalid data

(HTTP_STATUS_INTERNAL_SERVER_ERROR = 500)

```
(base) jingjingyang@jingjings-MacBook-Pro ~ % curl -i -s -X POST -H "Content-Typ
e: application/json" -d '{"NAME":"Wool Sweater","type":"Sweater","size":"L","col
or":"Red","price":60.99}' http://localhost:8080/clothes
HTTP/1.1 500 Internal Server Error
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 98
ETag: W/"62-c7lWgXgnuxptGJFK6mMgCYGwZZo"
Date: Tue, 19 Dec 2023 23:33:27 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"error":"SQLITE_CONSTRAINT: NOT NULL constraint failed: Clothes.name","code":"S
QLITE_CONSTRAINT"}%
```

PUT to update(replace) an item

```
(base) jingjingyang@jingjings-MacBook-Pro ~ % curl -i -s -X PUT -H "Content-Type
: application/json" -d '{"name":"Denim Jacket","type":"Jacket","size":"XL","colo
r":"Brown","price":50.99}' http://localhost:8080/clothes/1
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 37
ETag: W/"25-JB1Tin60AHjLd75ON4t0/Q1kVD8"
Date: Tue, 19 Dec 2023 23:34:02 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"message":"Row updated","changes":1}%
```

PUT with invalid data

("changes":0)

```
(base) jingjingyang@jingjings-MacBook-Pro ~ % curl -i -s -X PUT -H "Content-Type
: application/json" -d '{"name":"Denim Jacket","type":"Jacket","size":"XL","colo
r":"Brown","price":50.99}' http://localhost:8080/clothes/111
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 37
ETag: W/"25-H9XUMJUwB1GDqSwx5WsIscmVTlk"
Date: Tue, 19 Dec 2023 23:35:03 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"message":"Row updated","changes":0}%
```

DELETE an item

```
(base) jingjingyang@jingjings-MacBook-Pro ~ % curl -i -s -X DELETE http://localh
ost:8080/clothes/2
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 37
ETag: W/"25-F8iueZV1O1Skkf6zApRwls6i7iU"
Date: Tue, 19 Dec 2023 23:46:09 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"message":"Row deleted","changes":1}%
```

Summary

- In conclusion, I have successfully implemented a backend server using Node.js and SQLite in this project. I created a basic HTTP server, handled different types of HTTP requests, and connected to a SQLite database. I ensured to incorporate error handling and provide responses in JSON format, which are crucial for effective communication with clients. This project has been a foundational step in my journey towards mastering backend development.