

Fun 05 : MySQL

App.jsx

- Mostly remain the same as fun04 but I edited the function “save” to save on web (react) as local and save on database(sql) as always.

```
18 const save = async () => {
19   const now = new Date().toLocaleString();
20   const newEntry = { male_count: mancount, female_count: womancount, timestamp: now };
21
22   // Save locally
23   setLog([...log, `Man: ${mancount}, Woman: ${womancount} - Saved on: ${now}`]);
24   setTotalSum(totalSum + mancount + womancount);
25
26   // Save to database
27   try {
28     const response = await fetch("http://localhost:3001/save", {
29       method: "POST",
30       headers: { "Content-Type": "application/json" },
31       body: JSON.stringify(newEntry),
32     });
33
34     if (response.ok) {
35       fetchRecords();
36     } else {
37       console.error("Failed to save data.");
38     }
39   } catch (error) {
40     console.error("Error saving data:", error);
41   }
42 };
43
44 const fetchRecords = async () => {
45   try {
46     const response = await fetch("http://localhost:3001/records");
47     const data = await response.json();
48
49     setDbLogs(
50       data.map(
51         (entry) =>
52           `Man: ${entry.male_count}, Woman: ${entry.female_count} - Saved on: ${entry.timestamp}`
53       )
54     );
55   } catch (error) {
56     console.error("Error fetching records:", error);
57   }
58 };
59
```

DataBase.sql

- Created the table that I want to save.

```
App.jsx # index.css JS server.cjs DataBase.sql X
fun05 DataBase > DataBase.sql
1 CREATE DATABASE IF NOT EXISTS fun05;
2 USE fun05;
3
4 CREATE TABLE IF NOT EXISTS records (
5     id INT AUTO_INCREMENT PRIMARY KEY,
6     male_count INT NOT NULL,
7     female_count INT NOT NULL,
8     timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
9 );
```

Server.cjs

- This part for connect the web to mysql and created the table for records the data.

```
App.jsx # index.css # server.cjs X DataBase.sql
fun05 DataBase > fun05 > # server.cjs > app.get("/records") callback > con.query() callback
1 var mysql = require("mysql2");
2 var express = require("express");
3 var cors = require("cors");
4 var bodyParser = require("body-parser");
5
6 var app = express();
7 app.use(cors());
8 app.use(bodyParser.json());
9
10 // Connect to MySQL using the "fun05" database
11 var con = mysql.createConnection({
12     host: "localhost",
13     user: "root",
14     password: "root",
15     database: "fun05"
16 });
17
18 con.connect(function (err) {
19     if (err) {
20         console.error("Error connecting to MySQL:", err);
21         throw err;
22     }
23     console.log("Connected to MySQL!");
24
25     // Create the records table if it does not exist
26     const createTable = `
27     CREATE TABLE IF NOT EXISTS records (
28         id INT AUTO_INCREMENT PRIMARY KEY,
29         male_count INT NOT NULL,
30         female_count INT NOT NULL,
31         timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
32     )
33     `;
34     con.query(createTable, function (err, result) {
35         if (err) {
36             console.error("Error creating table:", err);
37             throw err;
38         }
39         console.log("Table ready!");
40     });
41 });
42
43 // API: Save data - Insert new record with male and female counts
44 app.post("/save", function (req, res) {
45     const { male_count, female_count } = req.body;
46
47     if (male_count === undefined || female_count === undefined) {
48         return res.status(400).send({ message: "Invalid data" });
49     }
50
51     const sql = "INSERT INTO records (male_count, female_count) VALUES (?, ?)";
52     con.query(sql, [male_count, female_count], function (err, result) {
53         if (err) {
54             console.error("Error saving record:", err);
55             return res.status(500).send({ message: "Error saving data" });
56         }
57         res.send({ message: "Data saved successfully!" });
58     });
59 });
60
61 // API: Fetch records with optional date filtering (expects startDate and endDate as YYYY-MM-DD)
62 app.get("/records", function (req, res) {
63     const { startDate, endDate } = req.query;
64     let sql = "SELECT * FROM records";
65     let params = [];
66
67     if (startDate && endDate) {
68         sql += " WHERE timestamp BETWEEN ? AND ?";
69         params = [startDate, endDate];
70     }
71
72     sql += " ORDER BY timestamp DESC";
73
74     con.query(sql, params, function (err, results) {
75         if (err) {
76             console.error("Error fetching records:", err);
77             return res.status(500).send({ message: "Error fetching records" });
78         }
79         res.json(results);
80     });
81 });
82
83 // Start the server on Port 3001
84 app.listen(3001, function () {
85     console.log("Server running on port 3001");
86 });
87
```

Result

The screenshot shows a web application on the left and MySQL Workbench on the right. The web application has a 'Counter' section with 'Man' at 12 and 'Woman' at 7. Below it is a 'Visitor Log' with two entries: 'Man: 15, Woman: 10 - Saved on: 3/4/2025, 12:57:00 AM' and 'Man: 12, Woman: 7 - Saved on: 3/4/2025, 12:58:14 AM'. The MySQL Workbench shows a database with a table 'records' containing 10 rows. The last two rows are highlighted with a red box: (61, 15, 10, 2025-03-04 00:57:00) and (62, 12, 7, 2025-03-04 00:58:14). The 'Action Output' pane shows messages for using the database, creating the table, and selecting records.

Counter

Man
12
UP DOWN

Woman
7
UP DOWN

Save Reset Sum All

Visitor Log

Man: 15, Woman: 10 - Saved on: 3/4/2025, 12:57:00 AM

Man: 12, Woman: 7 - Saved on: 3/4/2025, 12:58:14 AM

MySQL Workbench

Local instance MySQL80

Navigation: customers records records

Limit to 1000 rows

1 • SELECT * FROM fun05.records;

Result Grid

id	male_count	female_count	timestamp
53	5	6	2025-03-04 00:54:40
54	5	3	2025-03-04 00:54:44
55	5	4	2025-03-04 00:54:51
56	5	4	2025-03-04 00:56:00
57	5	4	2025-03-04 00:56:01
58	5	4	2025-03-04 00:56:01
59	0	0	2025-03-04 00:56:03
60	2	5	2025-03-04 00:56:07
61	15	10	2025-03-04 00:57:00
62	12	7	2025-03-04 00:58:14

Table: records

Columns: id, male_count, female_count, timestamp

records 1 x

Action Output

Time Action Message

233 00:57:11 USE fun05 0 row(s) affected

234 00:57:11 CREATE TABLE IF NOT EXISTS records (id INT AUTO_INCREMENT... 0 row(s) affected, 1 warning(s): 1050 Table 'records' already exists

235 00:57:47 SELECT * FROM fun05.records LIMIT 0, 1000 61 row(s) returned

Reset and Save new data : On web also clear the data but all of them will be save at Database (SQL)

The screenshot shows the web application and MySQL Workbench after a reset. The web application's 'Counter' section now shows 'Man' at 15 and 'Woman' at 10. The 'Visitor Log' has two entries: 'Man: 22, Woman: 19 - Saved on: 3/4/2025, 1:04:10 AM' and 'Man: 15, Woman: 10 - Saved on: 3/4/2025, 1:04:15 AM'. The MySQL Workbench shows the 'records' table with 11 rows. The last two rows are highlighted with a red box: (63, 22, 19, 2025-03-04 01:04:10) and (64, 15, 10, 2025-03-04 01:04:15). The 'Action Output' pane shows messages for creating the table and selecting records.

Counter

Man
15
UP DOWN

Woman
10
UP DOWN

Save Reset Sum All

Visitor Log

Man: 22, Woman: 19 - Saved on: 3/4/2025, 1:04:10 AM

Man: 15, Woman: 10 - Saved on: 3/4/2025, 1:04:15 AM

MySQL Workbench

Local instance MySQL80

Navigation: customers records records

Limit to 1000 rows

1 • SELECT * FROM fun05.records;

Result Grid

id	male_count	female_count	timestamp
55	5	4	2025-03-04 00:54:51
56	5	4	2025-03-04 00:56:00
57	5	4	2025-03-04 00:56:01
58	5	4	2025-03-04 00:56:01
59	0	0	2025-03-04 00:56:03
60	2	5	2025-03-04 00:56:07
61	15	10	2025-03-04 00:57:00
62	12	7	2025-03-04 00:58:14
63	22	19	2025-03-04 01:04:10
64	15	10	2025-03-04 01:04:15

Table: records

Columns: id, male_count, female_count, timestamp

records 2 x

Action Output

Time Action Message

234 00:57:11 CREATE TABLE IF NOT EXISTS records (id INT AUTO_INCREMENT... 0 row(s) affected, 1 warning(s): 1050 Table 'records' already exists

235 00:57:47 SELECT * FROM fun05.records LIMIT 0, 1000 61 row(s) returned

236 00:57:51 SELECT * FROM fun05.records LIMIT 0, 1000 61 row(s) returned