Exercise: “Simple School System” — Node.js + Express + MySQL (Docker)

A **very small CRUD project**, focused on:

- Express routing
- MySQL connection via `mysql2`
- Basic SQL (`SELECT / INSERT / UPDATE / DELETE`)
- Clean request/response handling

No joins. No relations. One table only.

Goal

Build an Express server that manages **students** in a school using a **single MySQL table**.

Part 0 — Environment Setup (student responsibility)

Tech stack

- Node.js
- Express
- MySQL running in Docker
- `mysql2` library

Expected outcome

- MySQL runs on `localhost:3306`
- Database name: `school_db`

- Server connects successfully using `mysql2`

Part 1 — Database Design (VERY SIMPLE)

Table: `students`

column	type	rules
id	INT	PK, AUTO_INCREMENT
name	VARCHAR(100)	NOT NULL
age	INT	NOT NULL
class_name	VARCHAR(20)	NOT NULL

Notes

- No foreign keys
- No timestamps
- No constraints beyond NOT NULL

Part 2 — Project Structure (minimal)

Required files:

- `server.js` – Express setup + listen
- `db.js` – MySQL connection (mysql2)
- `students.router.js` – all student endpoints

Part 3 — API Endpoints (CRUD)

General Rules

- JSON only
- Parameterized SQL (`?`)
- Clear status codes
- Simple error messages

---

1) Create Student

POST `/students`

Body

```
{
  "name": "Noam Cohen",
  "age": 15,
  "className": "9A"
}
```

Success (201)

```
{
  "id": 1,
  "name": "Noam Cohen",
  "age": 15,
  "className": "9A"
}
```

Errors (400)

- Missing fields
- Age not a number

---

2) Get All Students

GET `/students`

Success (200)

```
{
  "count": 2,
  "students": [
    { "id": 1, "name": "Noam Cohen", "age": 15, "className": "9A" },
    { "id": 2, "name": "Yael Levi", "age": 14, "className": "8B" }
  ]
}
```

---

3) Get Student by ID

GET `/students/:id`

Success (200)

```
{
  "id": 1,
  "name": "Noam Cohen",
  "age": 15,
  "className": "9A"
}
```

Not found (404)

```
{ "error": "STUDENT_NOT_FOUND" }
```

---

4) Update Student

PUT `/students/:id`

Body

```
{
  "name": "Noam Cohen",
  "age": 16,
  "className": "10A"
}
```

Success (200)

```
{
  "id": 1,
  "name": "Noam Cohen",
  "age": 16,
  "className": "10A"
}
```

Errors

- 400 – invalid body
- 404 – student not found

---

5) Delete Student

DELETE `/students/:id`

Success (200)

```
{
  "message": "Student deleted"
}
```

Not found (404)

```
{ "error": "STUDENT_NOT_FOUND" }
```

Part 4 — Functional Requirements

Students must demonstrate:

- Express routing
- MySQL connection via `mysql2`
- Correct SQL per endpoint

- Proper HTTP status codes
- Clean JSON responses

Bonus (Optional, not required)

- `GET /students?className=9A`
- Validation helper function
- Use `mysql2/promise`