# Node.js API

This small Node + Express API is a learning playground. It contains several simple endpoints you can call from the browser, curl, or Postman. The project uses an in-memory array for users so you can focus on HTTP, routing, and JSON without needing a database.

## What you'll learn

- How to run a Node.js/Express server
- How to call GET/POST/PATCH/DELETE endpoints
- How to read and send JSON payloads
- Input validation and error handling
- Middleware for logging requests
- Partial updates with PATCH

## Prerequisites

- Node.js (recommend v20+)
- npm (comes with Node)

## Quick start

1. Install dependencies

```
npm install
```

2. Start the server

```
npm start
# Or you can start directly using node:
node server.js
```

3. Open the API in your browser or use curl: `http://localhost:3000`

## Endpoints (for students)

- GET /health — simple health check
- GET /users — list all users; add `?minAge=18` to filter by age
- GET /users/:id — get a user by id
- POST /users — create a user; JSON body: `{ "name": "Alice", "age": 20 }`
- PATCH /users/:id — update a user; JSON body: `{ "name": "Alice Updated" }` or `{ "age": 21 }`
- DELETE /users/:id — delete a user by id

### Example responses

GET /users

```
[
  { "id": 1, "name": "Rick", "age": 23 },
  { "id": 2, "name": "Aadrija", "age": 22 }
]
```

POST /users (example)

Request body:

```
{ "name": "Someone", "age": 21 }
```

Response (201 Created):

```
{ "id": 3, "name": "Someone", "age": 21 }
```

PATCH /users/1 (example)

Request body:

```
{ "name": "Rick Updated" }
```

Response (200 OK):

```
{ "id": 1, "name": "Rick Updated", "age": 23 }
```

## Features

- **RESTful API**: Full CRUD operations on users (Create, Read, Update, Delete)
- **Input Validation**: Names must be at least 2 characters, ages between 13-120
- **Request Logging**: All requests are logged to the console with timestamps
- **Error Handling**: Proper HTTP status codes and error messages
- **CORS Support**: Allows cross-origin requests for frontend integration
- **Query Filtering**: Filter users by minimum age

## Try it with curl (Windows PowerShell examples)

```
# Get all users
curl http://localhost:3000/users | ConvertFrom-Json

# Create a new user
curl -Method POST http://localhost:3000/users -Headers @{"Content-Type"="application/json"} -Body
(ConvertTo-Json @{name='Someone'; age=21}) | ConvertFrom-Json

# Update a user (PATCH)
curl -Method PATCH http://localhost:3000/users/1 -Headers @{"Content-Type"="application/json"} -Body
(ConvertTo-Json @{name='Updated Name'}) | ConvertFrom-Json

# Get a user by id
curl http://localhost:3000/users/1 | ConvertFrom-Json

# Delete a user
curl -Method DELETE http://localhost:3000/users/1 | ConvertFrom-Json
```

## What you can try next

1. Add authentication with JWT tokens for user endpoints.
2. Persist users to a JSON file instead of memory.
3. Add rate limiting middleware to prevent abuse.
4. Implement user search by name (case-insensitive).
5. Add more complex validation (email format, etc.).

## Notes

- This project is intentionally simple to keep focus on HTTP and JSON.
- The in-memory store will reset when the server restarts.