LAB 7 – Fetch API + Tailwind (Play CDN)

Objective:

By the end of this lab, students can:

- Use Tailwind utility classes to build a simple UI.
- Use fetch() for GET and POST.
- Handle UI states: Loading ⟶ Success ⟶ Error.
- Parse JSON with res.json() and render results on the page

Lab instruction

- The LAB 7 instruction and lab resources are posted on CMU Mango: LAB7 – Fetch API + Tailwind (Play CDN)
- There are 2 assignments according to the LAB 7 sheet posted on the channel.
- The LAB 7 is worth 20 points in total.
- Score criteria: full point (for output correct); -1 (for output does not correct); -1 (for not follow problem constraint)
- **Assignment Submission:**
  - Upload your solutions to CMU Mango assignments. The submission later than the due date will get 50% off your score. At the close date, you cannot submit your assignment to the system.
  - Be prompt for TA calling to verify your work on your computer.

Requirements (Important)

- Keep your project in **CommonJS** (tsconfig: "module": "commonjs")
- Do not commit or submit **node_modules/**
- Your project must run in mode:
  - npm run dev

Project Setup

Download and unzip a starter code for this lab. Then you will file the directory has the following file structure.

File structure:

```
Lab07-fetch-tailwind/
  get/
    index.html
    app.js
  post/
    index.html
    app.js
```

Part A — GET: Load JSON and show it in the UI

A1) Create UI (Tailwind) (3 pts)

Open get/index.html:

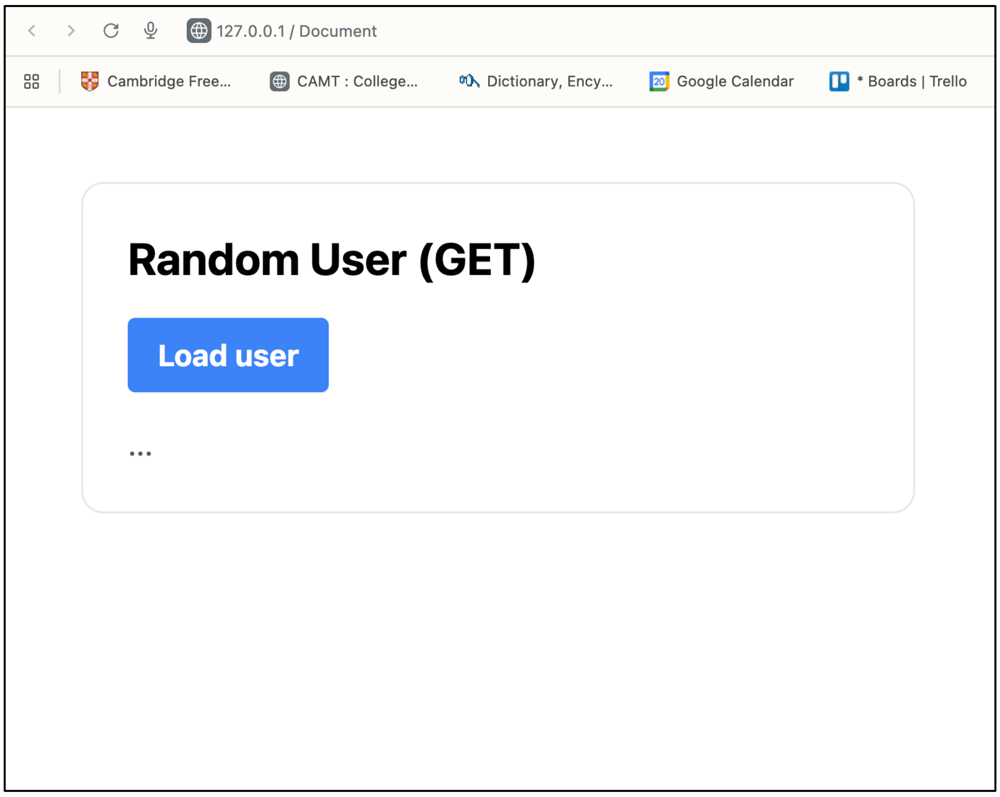
- Use Tailwind Play CDN in <head>. (You can find the Tailwind Play CDN at Tailwind CSS website)
- Create a card layout with:
  - Title: Random User (GET)
  - Button: Load user
  - Status text area
  - Result area (hidden at first)

Minimum UI elements (IDs required):

- btnLoad
- status
- result

Tailwind requirement: Use at least 6 utility classes (layout + spacing + typography).

Example of the card is below.



Random User (GET)

...

A2) Implement GET with fetch (7 pts)

Open get/app.js

When user clicks Load user:

1. Show status: Loading...
2. Hide previous result (if any)
3. Fetch from: https://randomuser.me/api/
4. If !res.ok, show error message
5. Parse JSON and render: name + email + avatar
6. Show status: Loaded successfully.

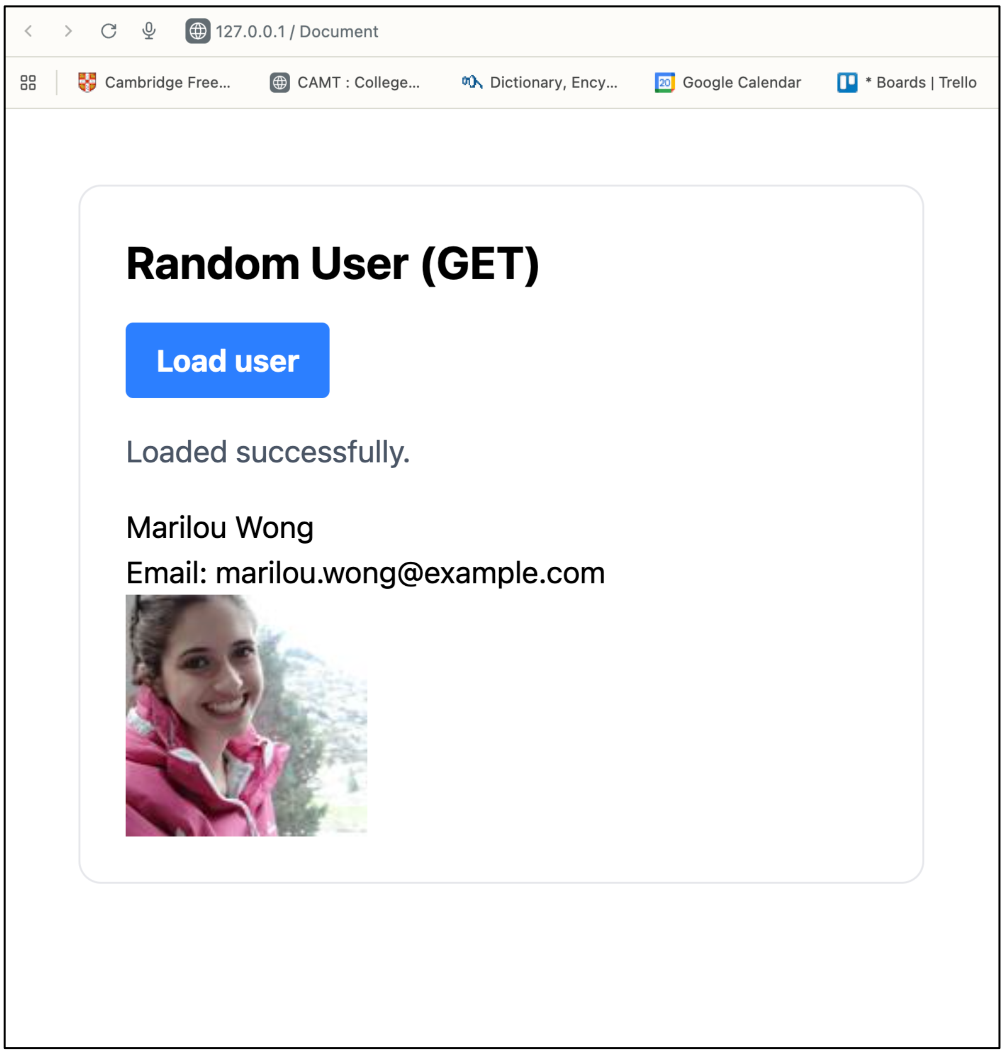
Must-have code pattern:

- async/await
- try/catch
- res.ok check
- await res.json()

Output requirements:

- Name
- Email
- Avatar image

The expected result is below.


127.0.0.1 / Document

Random User (GET)

Loaded successfully.

Marilou Wong
Email: marilou.wong@example.com

Part B — POST: Send JSON data to an API

B1) Create UI (Tailwind) (3 pts)

Open post/index.html with:

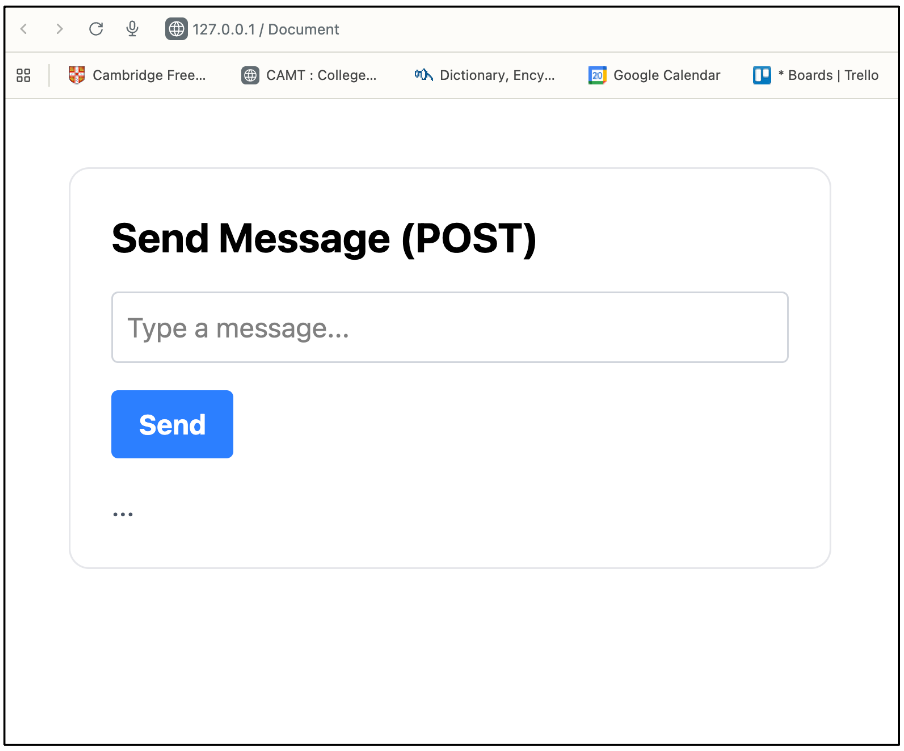
- Title: Send Message (POST)
- Text input (placeholder: "Type a message...")
- Button: Send
- Status text area

- Output area to show JSON response

Minimum IDs:

- msg
- btnSend
- status
- output

Example of the card is below.



127.0.0.1 / Document

Cambridge Free... CAMT : College... Dictionary, Ency... Google Calendar • Boards | Trello

Send Message (POST)

Type a message...

Send

...

B2) Implement POST with fetch (5 pts)

Open post/app.js

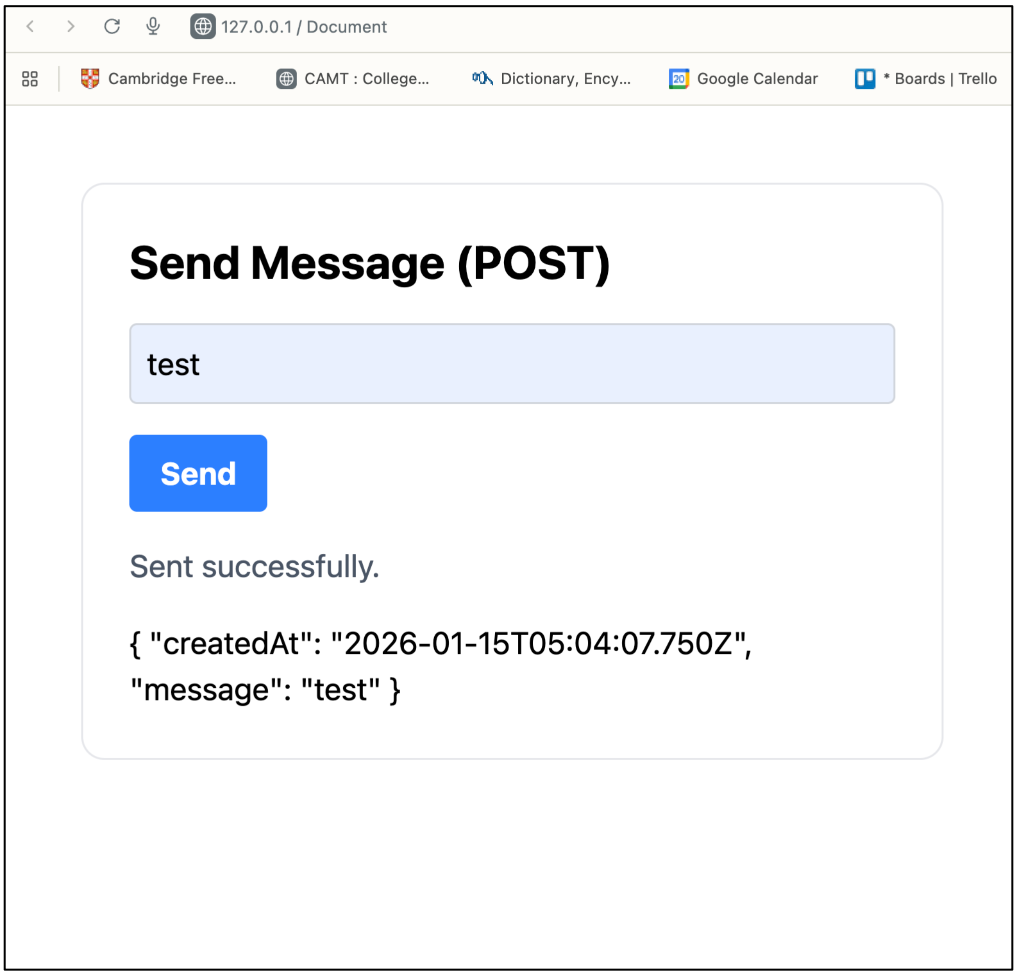
When user clicks Send:

1. Validate input (if empty ⟶ show "Please type a message first.")
2. Show status: Sending...
3. Send POST request to: https://httpbin.org/post
4. Include: method: "POST"; header "Content-Type": "application/json"; body: JSON.stringify({ message, createdAt })
5. Parse response JSON and display what you sent (the echoed JSON)

Output requirements:

- Show “Sent successfully.” on success
- Show error message on failure
- Display JSON nicely formatted (use JSON.stringify(obj, null, 2))

The expected result is below.



Send Message (POST)

Sent successfully.

```
{ "createdAt": "2026-01-15T05:04:07.750Z",
"message": "test" }
```

Part C — UI states (Required) (2 pts)

In both pages, implement these UI behaviors:

- Disable the button while loading/sending
- Re-enable it after request finishes (success or error)
- Status must clearly show one of: Loading/Sending, Success, Error

Deliverables (Submission)

Submit one GitHub repository link containing your LAB 7 project.

Include:

- get/ folder (HTML + JS)
- post/ folder (HTML + JS)
- 2 screenshots: (1) GET page after loading a user, (2) POST page after sending a message and showing response

Checklist (Self-check before submitting)

- GET works and displays name/email/avatar
- POST works and displays echoed JSON
- Uses try/catch and checks res.ok
- Buttons disable during request
- Tailwind UI is readable and consistent
- Screenshots included

Bonus (Optional, +1)

Add a “Clear” button on each page to reset UI (status + result/output).
