Take-Home Assignment: TinyLink

Build a small web app similar to **bit.ly**, where users can shorten URLs, view click statistics, and manage links.

1. You may use **Node.js + Express** or **Next.js** — your choice.
2. Use any lightweight CSS (plain CSS or Tailwind).
3. Host it using free services such as **Vercel**, **Render**, or **Railway**, with a free database service such as **Neon (Postgres)**.
4. You can use AI as an assistant; but be prepared to understand and explain what it is doing.

Estimated time to complete: about **2 days**.

Rules

- You can use ChatGPT, if you wish to. Still, you need to understand the design and implementation.
- Do not have others do the project for you. You will fail later anyways in the interview. Why put ourselves through that trouble?
- We are going to use automated testing. So, it is imperative that you follow the URL conventions.
- Even if you cannot complete it, do as much as you can. You may get partial credit.

What you submit

1. We expect a **public URL** for testing.
2. We expect a **github URL** for reviewing the code.
3. We expect link to a video explaining your solution and walk us through the code
4. Link to ChatGPT transcript or any other LLM, if you took help from it

Core Features

The application will have a few pages and support a few APIs. Overall, here is the functionality it should support:

Create short links

- Take a long URL and optionally a **custom short code** (e.g., `docs → https://example.com/docs`) and create a redirection URL as <yourwebsite>/<shortcode>.
- Validate the URL before saving.
- **Custom codes are globally unique** across all users; if a code already exists, show an error.

Redirect

- Visiting `/{code}` performs an **HTTP 302 redirect** to the original URL.

- Each redirect increments the total-click count and updates the “last clicked” time.

Delete a link

- Users can delete existing links.
- After deletion, `/{code}` must return **404** and no longer redirect.

Main pages

Dashboard:

- Table of all links:
  - Short code
  - Target URL
  - Total clicks
  - Last clicked time
- Actions: **Add** and **Delete** [When adding sure to take custom code as an option]
- Optional search/filter by code or URL.

Stats page:

- Stats page `/code/:code` for viewing details of a single link.

Healthcheck:

- System details and uptime details

Interface & UX Expectations

Deliver a **clean, thoughtful interface.**

- **Layout & Hierarchy:** clear structure, readable typography, sensible spacing.
- **States:** show empty, loading, success, and error states.
- **Form UX:** inline validation, friendly error messages, disabled submit during loading, visible confirmation on success.
- **Tables:** sort/filter, truncate long URLs with ellipsis, functional copy buttons.
- **Consistency:** shared header/footer, uniform button styles, consistent formatting.
- **Responsiveness:** layout adapts gracefully to narrow screens.
- **Polish:** minimal but complete; avoid raw unfinished HTML.

Pages & Routes

Purpose	Path	Auth
Dashboard (list, add, delete)	`/`	Public

Purpose	Path	Auth
Stats for a single code	`/code/:code`	Public
Redirect	`/:code`	Public
Health check	`/healthz`	Public

Hosting (Free Examples)

- **Next.js on Vercel + Neon Postgres**
- **Node + Express on Render + Neon Postgres**
- **Node + Express on Railway + Postgres**

Provide a `.env.example` listing required environment variables (e.g., database URL, base URL).

Autograding & Testability

To enable automated testing, please follow these conventions:

Stable URLs

- `/` — Dashboard
- `/code/:code` — Stats page
- `/:code` — Redirect (302 or 404)

Health Endpoint

- `GET /healthz` → returns status `200`
  Example response (feel free to add more):

  { "ok": true, "version": "1.0" }

API end points

Method	Path
`POST /api/links`	Create link (`409` if code exists)
`GET /api/links`	List all links
`GET /api/links/:code`	Stats for one code
`DELETE /api/links/:code`	Delete link

Rules:

- Codes follow `[A-Za-z0-9]{6,8}`.

What We'll Check (Automated + Manual)

1. `/healthz` returns 200.
2. Creating a link works; duplicate codes return 409.
3. Redirect works and increments click count.
4. Deletion stops redirect (404).
5. UI meets expectations (layout, states, form validation, responsiveness).

Additional Notes

- Field names, endpoint paths, and response structures must match this spec to pass automated testing.
- Clear commits, modular code, and working deployment will be given extra credit consideration.