Take-Home Assignment: Pastebin-Lite

Build a small “Pastebin”-like application. A user can create a text paste and share a link to view it.

This assignment is used as a **preliminary filter**.
Your submission will be evaluated **primarily by automated tests** against your deployed application.

You may use **Node.js / Next.js** or any equivalent stack. Deployment on **Vercel** is recommended.

What to Submit

- A **deployed URL** (for example: `https://your-app.vercel.app`)
- A **public git repository**
- A short **README** explaining:
  - how to run the app locally
  - what persistence layer you used
  - any important design decisions

Functional Requirements

User capabilities

1. Create a paste containing arbitrary text.
2. Receive a shareable URL for that paste.
3. Visit the URL to view the paste.
4. Pastes may become unavailable based on optional constraints.

Constraints on a Paste

A paste may optionally include one or both of the following constraints:

- **Time-based expiry** (TTL)
- **View-count limit**

Once a constraint is triggered, the paste becomes unavailable.

If both constraints are present, the paste becomes unavailable as soon as **either** constraint triggers.

Required Routes

Health check

`GET /api/healthz`

- Must return HTTP 200
- Must return JSON
- Must respond quickly
- Should reflect whether the application can access its persistence layer

Example response:

{ "ok": true }

Create a paste

`POST /api/pastes`

Request body (JSON):

{

"content": "string",

"ttl_seconds": 60,

"max_views": 5

}

Rules:

- `content` is required and must be a non-empty string
- `ttl_seconds` is optional; if present, it must be an integer ≥ 1
- `max_views` is optional; if present, it must be an integer ≥ 1

Response (JSON, 2xx):

{

"id": "string",

"url": "https://your-app.vercel.app/p/<id>"

}

Error cases:

- Invalid input must return a 4xx status with a JSON error body

Fetch a paste (API)

`GET /api/pastes/:id`

Successful response (JSON, 200):

{

"content": "string",

"remaining_views": 4,

"expires_at": "2026-01-01T00:00:00.000Z"

}

Notes:

- `remaining_views` may be `null` if unlimited
- `expires_at` may be `null` if no TTL
- Each successful API fetch **counts as a view**

Unavailable cases:

- Missing paste
- Expired paste
- View limit exceeded

All unavailable cases must return:

- HTTP 404
- JSON response

View a paste (HTML)

`GET /p/:id`

- Returns HTML (200) containing the paste content
- If the paste is unavailable, return HTTP 404

Paste content must be rendered safely (no script execution).

Deterministic Time for Testing

Your application must support deterministic expiry testing.

If the environment variable `TEST_MODE=1` is set:

- The request header
  `x-test-now-ms: <milliseconds since epoch>`
  must be treated as the current time **for expiry logic only**

If the header is absent, use real system time.

Persistence Requirement

Automated tests run against your **deployed application**.

If you deploy on a serverless platform (such as Vercel), in-memory storage alone is usually insufficient. Use a persistence layer that survives across requests (KV/Redis/Postgres/etc), and document your choice.

Automated Tests Your System Must Pass

Our grader will run the following checks against your deployed URL:

Service checks

- `/api/healthz` returns HTTP 200 and valid JSON
- All API responses return valid JSON with correct `Content-Type`
- Requests complete within a reasonable timeout

Paste creation

- Creating a paste returns a valid `id` and `url`
- Returned URL points to `/p/:id` on your domain

Paste retrieval

- Fetching an existing paste returns the original content
- Visiting `/p/:id` returns HTML containing the content

View limits

- Paste with `max_views = 1`:
  - first API fetch → 200
  - second API fetch → 404
- Paste with `max_views = 2`:
  - two successful fetches
  - third fetch → 404

Time-to-live (TTL)

- Paste with `ttl_seconds` is available before expiry
- After expiry (using `x-test-now-ms`), paste returns 404

Combined constraints

- Paste with both TTL and max views becomes unavailable when the **first** constraint triggers

Error handling

- Invalid inputs return appropriate 4xx responses with JSON errors
- Unavailable pastes consistently return HTTP 404

Robustness

- No negative remaining view counts
- No serving a paste beyond its constraints under small concurrent load

UI Expectations (High-Level)

- Users can create a paste via the UI
- Users can view a paste via the shared link
- Errors (invalid input, expired paste) are shown clearly

UI design and styling are not graded heavily, but the flows must work.

Time Expectation

This assignment is intended to take **2–4 hours**.

Submission

Send us:

- Deployed URL
- Git repository URL
- Short notes (persistence choice, notable decisions)

Good luck.

Coding & Repository Guidelines (Automatically Checked)

Your submission must satisfy the following repository-level requirements.
These are **checked automatically** before functional tests run.

Repository structure

- A file named **README.md** must exist at the repository root.
- README.md must contain:
    - a short project description
    - instructions to run the project locally
    - a note describing the persistence layer used
- The repository must not be empty and must contain source code (not only build artifacts).

Code quality signals (lightweight)

These are not stylistic nitpicks; they are simple, machine-checkable signals:

- No hardcoded absolute URLs pointing to `localhost` in committed code.
- No secrets, tokens, or credentials committed to the repository.
- Server-side code must not rely on global mutable state that breaks across requests in serverless environments.

Build & runtime

- The project must install and start using standard commands documented in README.md.
- The deployed app must start successfully without manual database migrations or shell access.

Failure to meet these guidelines may result in the submission being rejected **before** functional testing.