Developer Challenge

Context and Problem Statement

SecureLog is an non profit organization that provides encryption services. They want to revamp their encryption service with modern technologies. Build a full stack application that encrypts and decrypts payloads while providing searchable logs to past requests.

Requirements

UI

Create a React application with the ability to do the following:

- Enter an encryption key and a payload, call this service's API, log the request, and display the encrypted data
- Enter a decryption key and a payload, call this service's API, log the request, and display the decrypted data
- Paginated view of past requests

Our evaluation will not only be based on the design and layout of the application. However, we are looking for good UX and error handling.

API

Create a FastAPI or SpringBoot API that meets the following details:

1. POST /api/v1/encrypt

- Accept a JSON object containing { key: str, data: str }
- Respond with { data: str } where data is the resulting encrypted data from encrypting the data with the public key
- POST /api/v1/decrypt

- Accept a JSON object containing { key: str, data: str } where key is a private key and data is some encrypted data from private key's keypair
- Respond with { data: str } where data is the resulting data from decrypting the payload with the private key
- GET /api/v1/logs

- Paginate logs of requests made to this service, include two URL parameters: size (log count) and offset (number of logs from the beginning)

- Logs should be of the following type { id: str, timestamp: datetime, ip: str, data: str }
  - ID should be a UUID
  - Note that datetime should be a UNIX timestamp
- The response of this endpoint should be a collection of logs

PostgreSQL should be used to store logs. Logs should match the format defined in the API section.

The application as a whole should be containerized. Use docker-compose to spin up an instance of PostgresSQL and create a Dockerfile to build and image of your API. Include a linter workflow for both the web and server in your repository.

Using github actions include a linter workflow for the front-end and backend. The action should be triggered on every push.

Submission Guidelines

- Github repository link. The repo should contain both the frontend and backend (Remember to make it public so we can access)
- Deployed URL (optional, bonus): Deploy your app in AWS or any other provider of your choice.
- Add a README.md that includes installation and development instructions. A brief overview of how the application works. Bonus points if you include an architecture diagram.
- The repository should contain a docker-compose.yaml to run the full stack application (web + api + db), and a Dockerfile for the frontend and backend.
- .github/workflows/web-lint.yml and .github/workflows/server-lint.yml
- Repository structure

```
.github/workflows/
├web-lint.yml
    server-lint.yml
README.md
docker-compose.yml
web/
├package.json
    package-lock.json
    src/
    Dockerfile
server/
├src/
    Dockerfile
```

- ***You must submit your repo and deployment link via this Google Form***

SUBMISSION DEADLINE: October 10th @ 11:59 PM

Tips

- Get the ecryption/decryption working first, then add logging, pagination, and finally polish the UI/UX
- Be clear in the README.md. Include commands for steup, a description of each service, adn sample API requests
- Show clear messages in the UI for invalid keys, empty payloads, or server errors instead of generic errors
- Focus on UX details, add loading states, disable buttons while requests are being served
- Make sure your gh workflows actually run and pass

Revision #9
Created 24 September 2025 16:40:38 by Miguel Merlin
Updated 29 October 2025 20:15:46 by Miguel Merlin