Bilgi Lovelace

bilgirei@protonmail.com | github.com/bilgi42

WORK EXPERIENCE

Rorshok (Remote) Portland/US
Backend Software Engineer and Digital Coordination Manager May 2024 – Present

- Designed and coded a piece of software (Chat Task) as a solo developer. It uses Hugging Face API and utilizes multiple LLM's to make a to-do list automatically while having a meeting. Used FastAPI for backend, Qt6 for frontend and Docker in Ubuntu Server to deploy it.

- Built Rorshok Reader that turns multiple RSS feeds and websites into a personalized feed page. Used Python, BeautifulSoup, Gemini API, Reportlab and Newspaper3k.

- Rewrote Rorshok Reader in Rust, only using Axum, Tokio, Reqwest, and Serde. Achieved up to 20-100 times performance increases, a much cleaner codebase, and better results in web-crawling by using concurrency features of Rust/Tokio and content cleaning beforehand with pattern matching.

- For frontend for Rorshok Reader, I rewrote the whole application with Svelte/Tailwind CSS instead of the previous HTML/Tailwind CSS approach. It resulted in snappier UX and code maintainability for the future.

- Made several websites for clients using Sveltekit, Cloudflare Tunnels, Cloudflare Workers, and Cloudflare R2 storage solutions.

- Deployed servers in QEMU powered environments

- Used NixOS and disko for reproducible builds in our servers.

- Worked with LLMs, most with Gemini family, for data labeling
- Managed development teams for a Wordpress project
- Managed a website and a newsletter, using Ghost

Freelancing (Remote) London/UK
DevOps Intern July 2021 – October 2021

- Used LAMP stack to deploy websites
- Managed Linux servers
- Coded a RESTful API in Java 11 and Spring
- Coded Bash and Python scripts for automation

PROJECTS

Rorshok Reader (rorshokreader.com)
- Rewrote every backend components in Rust, frontend components in Svelte
- Wrote a Gemini wrapper for functions that's ~22% faster than Google's own Gemini library in Python. It also allows more concurrency thanks to Tokio.
- Other read/write operations and the web server itself have seen massive decreases in memory usage and latency. It's also the fastest web scraper out there for the data we need.
- My content cleaning algorithm cleans 70-90% of the unnecessary content beforehand without any loss of any actually useful information, tested with 100+ websites

Represence - Rich presence for applications with websockets (github.com/bilgi42/represence)
- It streams application data (in most of the cases for me, Visual Studio Code) with an extension, and the listener sends that information in REST endpoints and to a websocket for reduced latency. It's near real time.

- It allows users to stream their rich presence data in their websites and/or their applications with a really minimal footprint. The application itself consumes ~15 MB RAM and 0.1% of CPU resources.

Several open source contributions
- I have merged pull requests in some open source projects like:

- Zero Email, YC X25 (0.email)
- Cheating Daddy. An open source alternative to Cluely, ~3000 stars on Github (cheatingdaddy.com)
- wut_rust (crates.io/crates/huh)

bilgi.works - My personal website
- Coded a little portfolio website is Svelte 5
- It uses Represence for fetching real-time data from my computer
- Deployed it with Cloudflare Workers, it gets updated with my private Github repo.

bilgi.pink - My blog
- Used Hugo for building it, I chose the Archie theme and made personal changes with forking the theme
- Deployed it with Cloudflare pages, used Giscus for comments while maintaining a static website structure

lovelace.works - My photo gallery
- Coded a website for one of my personal hobbies, photography. Used HTML and Tailwind CSS
- Kept the same theme from bilgi.works
- Deployed it with Cloudflare Pages

Chat Task
- Used Python, Hugging Face API, OpenAI Whisper Large V3 and Meta Llama 3.1 1B to make an AI pipeline for an internal piece of software to solve our problems for a task manager. It listens to meetings and takes notes & assigns tasks automatically. Server side, it's using FastAPI. For frontend, it uses PyQt6.

Polybar Crypto Tracker
- Developed a multi wallet and multi coin capable cryptocurrency to fiat tracker for Polybar (A popular Linux taskbar program). Used Python, Blockcypher API and Coingecko API. Planning to make a version for all popular desktop native platforms (Windows, MacOS, GNOME and KDE) for taskbars.

GNOME Crypto-Track
- Ported Polybar Crypto Tracker to GNOME as an extension, using GSJ. Published on GNOME Extensions. Also have a copy on Github (MIT License)

ADDITIONAL

Programming Languages: Rust, Python, Java, Svelte, Nix, Haskell
Frameworks and other tools : Tokio, Axum, Flask, Fast API, Sveltekit, Tailwind CSS, Cloudflare Tunnels, Cloudflare Workers, Cloudflare R2, LLMs for Agentic Pipelines, DevOps, Linux, NixOS, Git, Github
Languages: English (C1), Turkish (Native)