Shashwat Agrawal

+91 9420153231 | shashwatagrawal473@gmail.com | Linkedin | Github

Experience

Open Source Contribution

Pandas

- Optimized performance by using regular expression in **format_is_iso** function. [PR]
- Resolved a bug related to **validate_percentile** function. [PR]
- Updated documentation in alignment with the deprecation of **infer_datetime_format** following PDEP4 standards. [PR]

Collaborative Project

Text Content Summarizer *Python, Flask*

- Led the development of a flexible Content Summarization API, utilizing Flask, the sumy library, and mediawikiapi to enable URL, text, and keyword-based summarization with the Latent Semantic Analysis(LSA).
- Leveraged mediawikiapi to retrieve Wikipedia content based on user-provided keywords.
- Reduced content summarization time by 40% through the utilization of the LSA and efficient API design.

Projects

tec.h | *C*

- Developed **TEC (Test Engine for C)**, a lightweight, header-only unit testing library designed for seamless integration with C projects.
- Provides an intuitive interface for writing and executing tests with minimal setup, enhancing developer productivity.
- Ongoing improvements to expand functionality and optimize performance.

Byte Machine | *Rust*

- Designed and implemented an 8-bit virtual machine with a custom instruction set, complete with CPU emulation and memory management system.
- Engineered a custom assembler that translates assembly code to bytecode, featuring support for labels, variables, and basic control flow.
- Implemented key components such as a stack, registers, and memory management, mimicking real-world processor functionality.
- Optimized bytecode execution for performance and extensibility, allowing future enhancements and additional instructions.

DNS Server | *Python*

- Engineered a minimalist **DNS server** using only the Python standard library, ensuring full compliance with **RFC 1035** specifications.
- Deeply analyzed networking protocols and DNS mechanics, meticulously implementing request parsing and response handling.
- Achieved an average query response time of **100ms** under typical load, optimizing efficiency through caching and streamlined request processing.

HTTP Server | *Rust, Tokio, Multi-threading*

- Developed a high-performance **multi-threaded and asynchronous HTTP server** leveraging **Tokio** for concurrency.
- Achieved a **100% success rate** handling **100,000 requests** at 50 concurrent requests/sec.
- Optimized response times ranging from **400µs to 32.8ms**, with an average of **5.9ms** and peak throughput of **8,423 requests/sec**.

Technical Skills

Languages: C, Rust, Python, JavaScript, SQL (Postgres)
Frameworks: Flask, Django, Node.js
Developer Tools: Git, Docker, Podman, NeoVim, Arch Linux
Libraries: Pandas, Matplotlib, Numpy, Tokio, Axum