Shashwat Agrawal

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

Education

SiddhiVinayak Technical Campus — Khamgaon, Maharashtra
Diploma in Computer Science and Engineering — *July 2024*

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

DNS Server | *Python*

- Crafted a minimalist DNS server exclusively leveraging the Python standard library, ensuring adherence to RFC 1035 specifications.
- Navigated RFC 1035 specifications meticulously, ensuring robustness while exploring protocols and networking intricacies.
- Achieved an average DNS query response time of 100ms under typical load conditions.

Containers | *Rust, Linux SysCalls*

- Engineered containerization solutions from scratch, delving into Linux system fundamentals and mastering containerization concepts.
- Ensured 99.9% process isolation using Linux namespaces and cgroups in containerized environments.
- Created a rootless version of container too getting valuable insights about them.

Load Balancer | *Rust*

- Built a robust multi-threaded load balancer, employing the round-robin algorithm to efficiently distribute requests across servers.
- Deployed a flexible server configuration system allowing easy customization of server pool without code modifications.
- Reducing server load by 75%(with 4 servers) as each server now handles only a quarter of the total requests.

Interpreter | *Rust*

- Designing and developing an interpreter for a custom programming language.
- Implemented core components including tokenizer, lexer, REPL, and an initial parser.

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

- 100% success rate processing 100,000 requests at 50 concurrent requests per second. Response times: 0.0004 - 0.0328 seconds(400µs - 32.8ms), with an average of 5.9ms and peak throughput of 8,423.3129 requests/sec.
- Constructed both a multi-threaded and asynchronous version of the HTTP server, leveraging the Tokio library for the asynchronous part.

Technical Skills

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