Daksh Upadhyay

work@dakshu.xyz | +1(XXX)-XXX-XXX | Website | GitHub

Education

The XYZ University — September 2021 – Apr 2025 (Anticipated)

Bachelor Of Computer Science

Skills

Languages: Rust, C#, JavaScript, HTML/CSS.
Technologies: Yugabyte, PostgreSQL, Git, GraphQL, gRPC, Figma, ASP.NET Core.

Professional Experience

KeepWorks

Bengaluru, Karnataka
Software Engineering Intern — Dec 2020 – March 2021

⇒ Create an open-source crate to query ISO country data to implement multiple currencies, time zones and languages in our product. **This allowed customers from various countries to use our product.**
⇒ Design a procedural macro which made server-side sanitizing easy by allowing you to write attributes on the struct declaration itself. The **first open-source crate from KeepWorks**, this **decreased time spent on sanitizing approximately 500 data structures across the codebase**.

KeepWorks

Bengaluru, Karnataka
Software Engineering Contractor – Backend Rust Team — May 2021 – March 2022

⇒ Lay foundations for backend applications, connect them to Yugabyte database and setup migrations.
⇒ Provide upgrades for our internal CLI tool to allow testing for all our projects and logs in one console instance, **this made testing twice as fast for our all our developers.**
⇒ Integrate gRPC and GraphQL microservices in a single rust project to facilitate communication between them which allowed for easier integration between microservices in our products.
⇒ Write CRUD operations with diesel and database schemas for our products.

Open-Source Projects

Octane

⇒ Octane features a http implementation and its own multithreaded web server built from scratch with express like routing.
⇒ Provide SSL/TLS support with both OpenSSL and RustTLS.
⇒ Use Tokio to provide multithreading at IO Level.
⇒ Write integration tests and unit tests to test server.

RToml

⇒ Rtoml features a hand written lexer and a hand written parser.
⇒ Comply with the official TOML spec.
⇒ Provide support for arrays, numbers, strings, multiline strings with escaping, inline tables and table aliasing in keys.
⇒ Write Unit tests for every module, perform integration testing and use existing toml-testing suite

Co-Chat

⇒ Use C# and ASP.NET Core with razor pages to create a web chat interface.
⇒ Use SignalR to implement websockets on frontend and backend.
⇒ Write the frontend in pure CSS (without any framework), vanilla JS to handle UI components.

Croc-Look

⇒ A cargo-expand alternative which allows you to debug your procedural macro
⇒ cross-term as the backend with clap to parse CLI arguments.
⇒ Features a watch option which opens a TUI which shows **live reloaded** generated code from your macros.
⇒ Parse rust structs, functions and trait impls to traverse through it for collecting exactly what the user asked for without any clutter.
⇒ Write a 5 min long blog post tutorial on croc-look.

Iso-rs

⇒ Parse iso data json file using serde, phf-codgen to generate lookup tables at compile time.
⇒ Build.rs generates lookup table and helper methods to query data.
⇒ Place country capitals. Alpha_3, alpha_2 and region data generation behind feature flags to save memory if not needed.

Sanitizer

⇒ Provides a procedural macro to automatically generate sanitizing logic for fields with certain types.
⇒ Used syn to parse rust tokens, quote to generate code.
⇒ Write unit tests to check sanitizers credibility