GSoC 2025 Proposal: Developing an Interactive Terminal User Interface (TUI) for Sugar using Textual

Applicant: Yuvraj Biswal
Email: yuvrajbiswalofficial@gmail.com
GitHub: https://github.com/Yuvraj-cyborg
LinkedIn: https://www.linkedin.com/in/yuvraj-biswal-a802b0259/
University: Silicon University, Odisha, India
Timezone: Indian Standard Time
Project: Create TUI for Sugar Using Textual
Organization: OSL Incubator
Mentors: Ivan Ogasawara, Luis Casas, Sandro Loch, Felipe Paes

1. Synopsis

This project aims to develop a Terminal User Interface (TUI) for Sugar, a tool designed to simplify container management based on Docker Compose V2. Inspired by interfaces like k9s for Kubernetes, this TUI will provide users with a visual, interactive, and efficient way to manage Sugar service groups, view logs and statistics, and inspect service details directly within the terminal. The TUI will be built using the Python library Textual and accessible via the `sugar tui` command, significantly enhancing Sugar's usability and user experience.

2. Benefits to the Community

Integrating a TUI into Sugar offers substantial advantages for its users and the OSL Incubator community:

- Improved User Experience: Provides a graphical interface within the terminal, making Sugar's features more intuitive and accessible, especially for those less comfortable with complex command-line arguments or new to container management.
- Enhanced Productivity: Allows users to perform common tasks (start/stop/restart services, check logs/stats) faster through interactive

navigation and commands within a single interface, streamlining development and operational workflows.
- Increased Discoverability: The visual nature of a TUI helps users easily discover and utilize the full range of Sugar's capabilities without needing to memorize specific commands or flags.
- Modern Tooling: Leverages Textual, a modern Python TUI framework, keeping Sugar aligned with contemporary developer tools and potentially attracting new users interested in rich terminal applications.
- Lower Barrier to Entry: Simplifies the learning curve for Sugar, making it easier for new users to adopt and benefit from its container management simplification features.
- Value Addition to OSL Incubator: Delivers a significant, user-facing feature enhancement to an OSL Incubator project, increasing its maturity and appeal within the open-source ecosystem.

3. Project Goals & Deliverables

The primary outcome is a functional and well-integrated TUI for Sugar.

- Core Goals:
    - Develop a robust and user-friendly TUI for Sugar using the Textual library.
    - Integrate the TUI seamlessly into the existing Sugar CLI (`sugar tui`).
    - Provide access to key Sugar functionalities through the TUI.
- Key Features/Deliverables:
    - Functional TUI Application: Accessible via `sugar tui`.
    - Group Selection: Interface to list and select configured Sugar service groups.
    - Service Management: Functionality within the TUI to start, restart, and stop services in the selected group.
    - Logs Viewing: Capability to display logs for individual services within the TUI.
    - Stats Viewing: Interface to show basic statistics (e.g., status, potentially resource usage if easily accessible via Docker Compose) for services.
    - Service Details Display: Ability to view detailed information like mapped ports, volumes, and relevant configuration aspects for a selected service.
    - Updated Documentation: Integration of TUI usage instructions and

examples into Sugar's official documentation.
    - Blog Post: An article explaining the new TUI feature, its development, and benefits.
    - CI Tests: Implementation of tests (unit/integration where feasible for TUI components) within the CI pipeline to ensure TUI stability.

4. Technical Approach

- Core Framework: The TUI will be developed using Textual, a Python library chosen for its modern features, widget toolkit, layout system, and asynchronous capabilities, enabling the creation of a rich, responsive terminal interface.
- Integration with Sugar: The TUI code will reside within the Sugar project structure. It will interact with Sugar's existing Python codebase to:
    - Discover and parse Sugar configuration files (`sugar.yml`).
    - List available service groups and services.
    - Invoke Sugar's functions or underlying Docker Compose V2 commands to manage service lifecycles (start, stop, restart).
    - Fetch data required for the TUI, such as logs, service status, and configuration details, by interfacing with Docker Compose V2 or the Docker engine API directly if necessary.
- Data Handling: Asynchronous operations (`async`/`await` supported by Textual) will be used for potentially blocking tasks like fetching logs or stats to keep the UI responsive.
- Error Handling: Graceful error handling will be implemented within the TUI to manage issues like invalid configurations, Docker daemon connection problems, or failed commands, providing clear feedback to the user.

5. Personal Background

I am a second-year Computer Science major at Silicon University. I possess a diverse technical skillset highly relevant to developing the Sugar TUI:

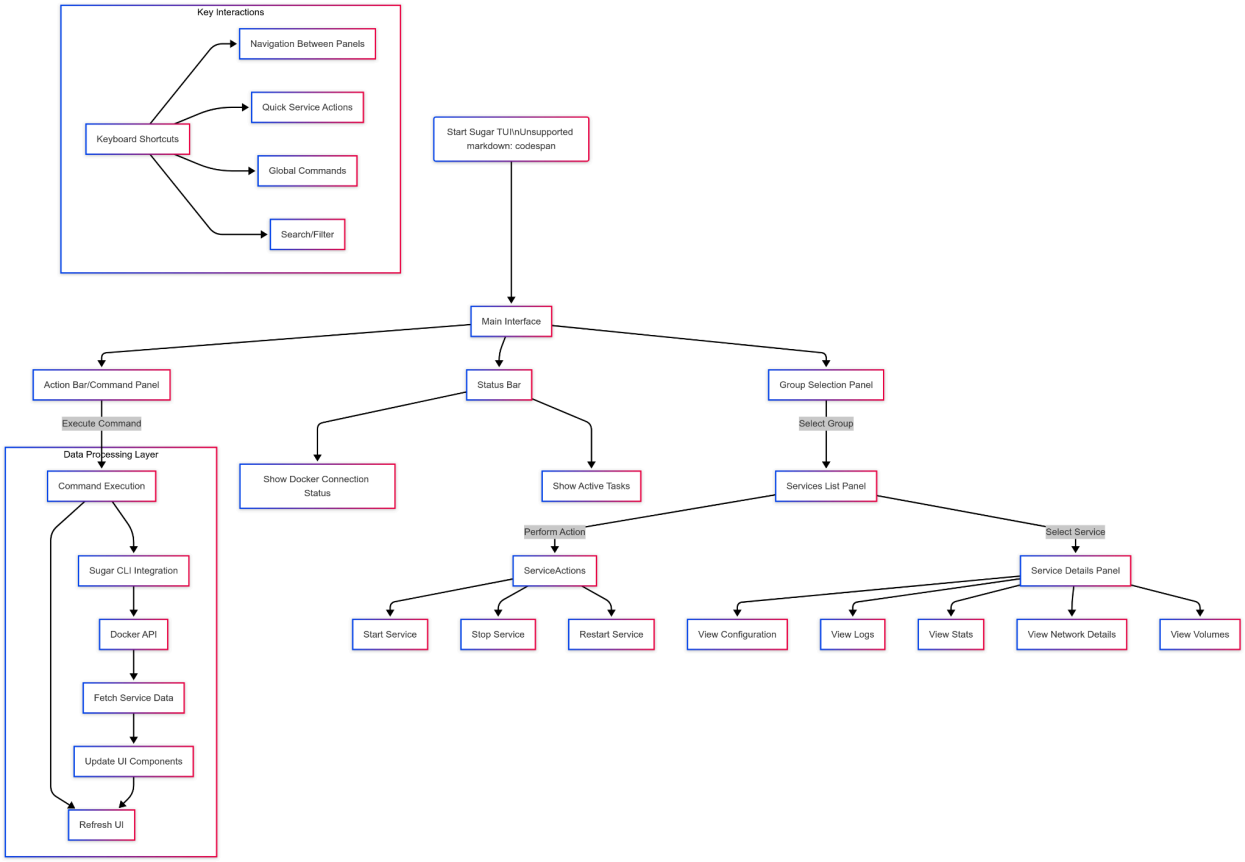
- Programming Languages: I have intermediate to advanced Python skills, essential for working with Sugar and the Textual library. My experience also includes JavaScript (including proficiency with modern frameworks like [Mention specific JS frameworks, e.g., React, Vue, Node.js]) and Rust (demonstrating systems programming capability) showcasing a broad understanding of different programming paradigms.

- CLI Development: My previous experience building a CLI tool using Rust provides me with direct, practical knowledge of command-line application design, argument parsing, and user interaction in a terminal environment – skills directly transferable to building an effective TUI.
- Containerization (Docker): I have practical experience using Docker and Docker Compose, primarily for managing development databases (PostgreSQL). While my experience is basic, I understand the core concepts of services, containers, and compose files, and I am confident in my ability to quickly learn the specifics needed to interact with them via Sugar's abstractions or directly for the TUI.
- Database & Systems Knowledge: Experience with Oracle Database and low-level languages (C, Assembly) further demonstrates my technical breadth and problem-solving capabilities.
- Open Source & Collaboration: I understand the importance of clear code, documentation, and collaboration using tools like Git and GitHub. I am enthusiastic about contributing to open source through GSoC and the OSL Incubator.

My combined experience in Python, CLI tool development, UI/UX principles from web development, and foundational Docker knowledge makes me well-equipped to tackle this project. I am eager to learn Textual and contribute a high-quality TUI to the Sugar project.

6. Engagement with the Organization

- Understanding the Project: I have reviewed the Sugar TUI project idea, the linked GitHub issue (#42), the Sugar repository (understanding its purpose and reliance on Docker Compose V2), and the Textual documentation to grasp the project's scope and technical requirements.
- Here’s a scrap flow of project i made with my understanding of the project and mermaid AI for a better outlook :



- Communication Plan: If selected, I will maintain active and clear communication with the mentors (Ivan Ogasawara, Luis Casas, Sandro Loch, Felipe Paes) via preferred channels. I plan regular updates, prompt questions, and active participation in discussions.
- Collaboration: Although I haven't contributed to Sugar previously, I am keen to integrate with the development workflow, follow project guidelines, and actively seek feedback throughout the GSoC period to ensure the TUI meets the community's needs.

7. Proposed Timeline (350 Hours)

This is a potential breakdown over a typical GSoC duration (approx. 12 weeks):

Community Bonding Period:

    - Deep dive into Sugar's codebase, architecture, and interaction with Docker Compose.
    - Master Textual fundamentals and advanced features (widgets, layouts, async).
    - Set up the development environment.
    - Discuss and refine TUI design mockups/wireframes with mentors.

Week 1-2: Initial Setup & Basic Structure

    - Set up the basic Textual application structure within the Sugar project.
    - Implement integration to read Sugar config and list service groups.
    - Create the main layout structure (e.g., group panel, service panel).

Week 3-4: Service Listing & Details

    - Display services within the selected group, showing basic status.
    - Implement the panel/view to show detailed service information (ports, volumes, etc.).

Week 5-7: Service Management Implementation

    - Implement UI elements (buttons, keybindings) for start, stop, restart actions.
    - Integrate these actions with Sugar's core functions or Docker Compose commands.
    - Add visual feedback for actions (e.g., status changes, notifications).

Week 8-9: Log Viewing

- Implement the log viewing panel/screen.
- Fetch and display logs (potentially streaming) for selected services asynchronously.

Week 10: Stats Viewing & Refinement

- Implement the stats viewing feature (displaying available status/info).
- Focus on UI polishing, usability improvements, and refining error handling.

Week 11-12: Testing, Documentation & Finalization

- Write unit and integration tests for TUI components where feasible.

- Write comprehensive documentation for using the `sugar tui`.
- Write the required blog post.
- Final code cleanup, testing, and preparation for submission.

8. Conclusion

The proposed Sugar TUI project will significantly enhance the tool's usability by providing an interactive, terminal-based interface for managing containerized services. My background in Python, CLI development (Rust), and foundational Docker knowledge, combined with my enthusiasm for learning Textual and contributing to open source, makes me a suitable candidate. I am eager to work with the OSL Incubator mentors to deliver a valuable addition to the Sugar ecosystem.

Thank you for considering my application.