GSoC 2025 Proposal

LLM4S

Implement an agentic toolkit for Large Language Models

SCALA CENTER

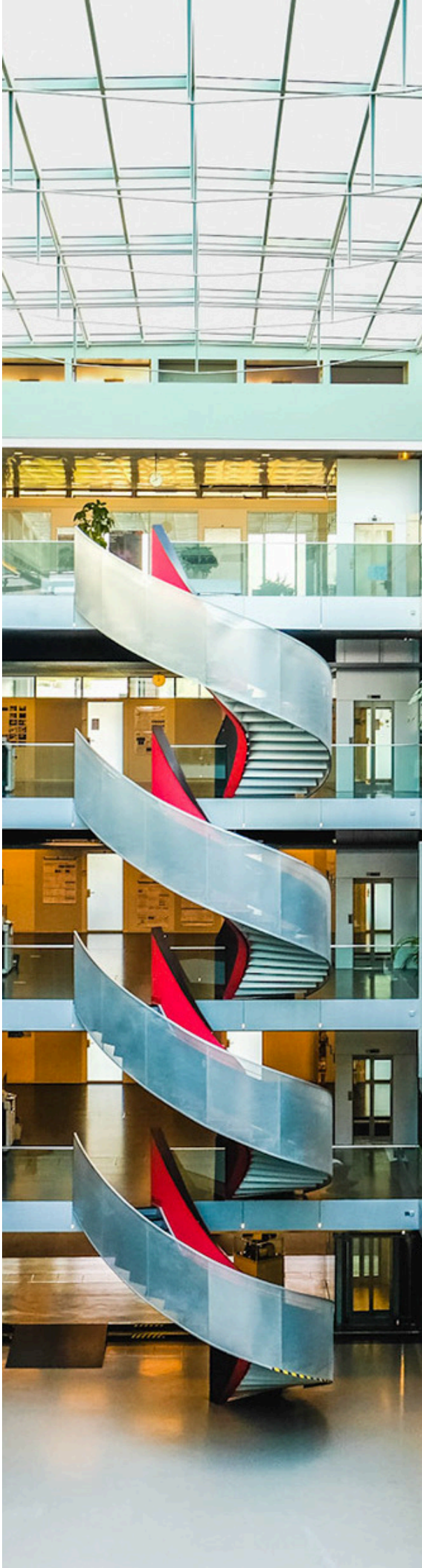
30.03.2024

Large Project

MENTORS

Kannupriya Kalra
kannupriyakalra@gmail.com

Rory Graves
rory.graves@fieldmark.co.uk



Elvan Konukseven

Ecole Polytechnique fédérale de Lausanne (EPFL), Switzerland

Name and Contact Information

Name: Elvan Konukseven
Email: elvankonukseven0@gmail.com
GitHub: elvankonukseven
Website: elvankonukseven.com
Location: Lausanne, Switzerland
Timezone: UTC+2

Synopsis

LLMs are revolutionizing software, but their inherent uncertainty poses significant challenges. Inconsistent outputs and potential for errors demand a more robust development approach. Why settle for the dynamically-typed wild west when you can harness the power and safety of Scala's type system? LLM4S is building a Scala-native AI toolkit that leverages functional programming principles and strict compile-time checks to ensure the reliability and maintainability of LLM-powered applications.

This project aims to extend the initial agentic loop and implement essential tools, empowering Scala developers to build robust, reliable, and explainable AI solutions. It's time to bring type-safe AI to Scala – because accuracy and trust are non-negotiable.

What it means to accomplish

● **Agentic Loop Implementation**: Enhance the current agentic loop architecture that allows LLMs to reason, plan, and execute actions through provided tools.

● **Tool Interface Design**: Create a flexible, type-safe interface for defining and registering tools that LLMs can invoke during execution.

● **MCP toolset:** Enable dynamic interaction of LLMs with external tools.

● **Core Tools Development**: Implement fundamental tools such as file system access, web browsing capabilities, and API interaction tools.

● **Context Management**: Develop mechanisms for maintaining conversation context and tool execution history to enable coherent, multi-turn interactions.

● **Testing Framework**: Create a testing framework specifically designed for validating agentic LLM behaviors and tool interactions.

● **Documentation and Examples**: Provide extensive documentation and example applications demonstrating the agentic toolkit's capabilities and best practices.

● **Human-in-the-Loop Capability (optional)**: Design and implement a feature enabling human oversight and intervention in the agentic loop, allowing users to review, approve, or modify actions before execution for critical operations or enhanced safety.

How will it benefit LLM4S?

● **Enhanced Capabilities**: The agentic toolkit will significantly expand LLM4S's capabilities, allowing it to support a wider range of applications beyond simple prompt-response interactions.

● **Developer Experience**: Scala developers will benefit from a type-safe, functional approach to building LLM-powered applications with reduced boilerplate and increased reliability.

● **Ecosystem Maturity**: With core components like tools, testing, and documentation in place, LLM4S can serve as a foundational layer for broader AI development in Scala.

● **Interoperability & Standards**: MCP support opens the door to future integration with external agents, tools, or hosted services, keeping LLM4S compatible with emerging AI infrastructure.

● **Competitive Advantage**: This implementation will position LLM4S as a leading Scala toolkit for LLM development, offering features comparable to or exceeding those available in other languages.

● **Practical Applications**: Organizations using Scala will be able to rapidly prototype and deploy LLM-based solutions for data analysis, content generation, customer service, and more.

● **Community Growth**: A solid, documented, and testable agentic toolkit will attract contributors, projects, and organizations to the Scala + LLM space.

Deliverables

● **Agentic Loop Core:** Complete the implementation of the agentic reasoning and execution loop. This should support multi-step workflows, error recovery, and tool re-invocation when necessary.

● **Tool Registry and Management System:** A flexible system for registering, discovering, and invoking tools. Support structured input/output schemas, validation, and dynamic registration.

● **MCP Toolset Integration:** Implement an MCP-compatible tool layer. Include an MCP client to dynamically discover and invoke external tools exposed via MCP servers. Ensure full compliance with the MCP specification to support future LLM ecosystems.

● **Core Tools Development:** Implement fundamental tools such as file system access, web browsing capabilities (e.g. search + scrape), and HTTP API interaction. These tools should follow the same interface standards and be compatible with both native and MCP tool runtimes.

● **Comprehensive Tests:** Unit and integration tests ensuring the reliability of the agentic toolkit.

● **Documentation:** Technical documentation covering architecture, usage patterns, and extension points.

● **Example Applications:** Sample applications demonstrating real-world usage scenarios.

Additional Deliverables Based on Progress

If development progresses ahead of schedule, the following additional deliverables may be implemented:

- **Human-in-the-Loop Capability:** Framework for users to review, approve, or modify proposed actions and system for incorporating user feedback into agent's reasoning processé.
- **Long-term Memory**: Persistent storage for agent knowledge and context management.
- **Dynamic Tool Discovery**: Runtime tool discovery based on context.

Milestones

Phase I

- **Core Agentic Architecture Implementation (milestone 1).**
- **Tool Interface Definition and Registry System (milestone 2).**

Phase II

- **File System and Basic Utility Tools Implementation (milestone 3).**
- **Web Access and API Interaction Tools (milestone 4).**

Phase III

- **Testing Framework and Comprehensive Tests (milestone 5).**
- **Documentation and Example Applications (milestone 6).**

Schedule

Community Bonding Period - Pre-development Phase (May 8 - June 1)

- **Strengthen Scala Proficiency**: Focus on advanced Scala concepts like higher-kinded types, type classes, and effect systems that will be used in the project.
- **LLM Research**: Gain deeper understanding of LLM capabilities, limitations, and agentic patterns by studying existing implementations and academic papers.
- **Codebase Familiarization**: Set up a development environment and thoroughly explore the existing LLM4S codebase to understand its architecture and design principles.
- **Establish Communication**: Set up regular check-ins with mentors and discuss project details, clarify technical requirements, and align on expectations.

Phase I: Foundation (June 1 - July 14)

1. Core Agentic Architecture Implementation (week 1-3)

- **Agent Loop**: Type-safe implementation of the ReAct loop with reasoning steps
- **Reasoning System**: Components for planning, decision-making, and task decomposition
- **State Management**: Type-safe agent state tracking with immutable data structures
- **Testing**: Unit tests for core components
- **Outcome**: A functional agentic loop capable of multi-step reasoning

2. Tool Interface Definition and Registry System (week 4-6)

- **MCP toolset :** Implement an MCP-compatible tool layer with an MCP client.
- **Tool Definition DSL**: Type-safe interface for defining tools with input/output types
- **Tool Registry**: System for discovering and managing available tools
- **Serialization Framework**: JSON conversion for tool inputs/outputs
- **Tool Execution**: Pipeline for executing tools with proper error handling
- **Documentation**: Developer documentation for tool creation
- **Outcome**: Complete tool interface system that integrates with the agentic loop

Phase II: Tools Implementation (July 15 - August 11)

3. File System and Basic Utility Tools Implementation (week 7-8)

- **File Operations**: Reading, writing, and manipulating files with proper error handling
- **Directory Operations**: Listing, navigating, and working with directories
- **File Metadata**: Tools for extracting and using file metadata
- **Text Processing**: Utilities for working with text content
- **Containerization**: Integration with LLM4S container system for secure execution
- **Outcome**: Comprehensive file system tools package with security controls

4. Web Access and API Interaction Tools (week 9-10)

- **Web Fetching**: Tools for retrieving and parsing web content
- **HTTP Client**: Type-safe API for making HTTP requests with various methods
- **Content Processing**: HTML, JSON, and XML parsing utilities
- **Rate Limiting**: Built-in rate limiting to prevent API abuse
- **Outcome**: Complete web interaction tools package with proper error handling

Phase III: Quality and Documentation (August 12 - August 25)

5. Testing Framework and Comprehensive Tests (week 11)

- **Agentic Testing Framework**: Specialized framework for testing LLM agent behaviors
- **Mock LLM Responses**: System for creating deterministic LLM responses
- **Integration Tests**: End-to-end tests covering realistic agent scenarios
- **Benchmarking**: Performance testing and optimization (SWE bench)
- **Multi-provider Testing**: Tests with different LLM providers (OpenAI, Anthropic)
- **Outcome**: Robust testing system ensuring reliable agent behavior

6. Documentation and Example Applications (week 12)

- **Technical Documentation**: Comprehensive documentation of all components
- **Tutorials**: Step-by-step guides for common use cases
- **Example Applications**:
    - **Code Assistant**: Agent that helps with coding tasks
    - **Research Agent**: Tool for gathering and synthesizing information
    - **Data Analyst**: Agent for working with data files and analysis
- **API Reference**: Complete API reference documentation
- **Outcome**: Fully documented framework with practical examples

Availability

Throughout the GSoC program period (May 8th to September 1st), I will dedicate my time according to a two-phase schedule. During the initial phase (May 8th to June 27th), I will commit 12-15 hours weekly to the project while balancing other obligations (exams). Starting June 27th through the project conclusion on September 1st, I will increase my commitment substantially to 40-45 hours per week, allowing for focused, intensive development. Additionally, I have complete availability on weekends throughout the entire program duration, providing flexibility for mentor meetings, collaborative sessions, or catching up on tasks when necessary.

Publishing the code

Pull Request Strategy:

- I plan to submit at least one pull request (PR) for each milestone achieved throughout the project.
- These PRs will be directed to a designated branch specifically created for this project.
- Once all milestones are completed, this project branch will undergo merging into the main branch.

Adherence to Best practices:

- Throughout my contributions, I aim to adhere to best practices such as maintaining clear and concise code and having adequate commit messages.
- I'll ensure thorough documentation where necessary to enhance the project's usability and maintainability.
- Actively participating in code reviews and discussions will be a priority to ensure the quality and coherence of the project codebase.

Responsiveness to Feedback:

- I commit to remaining responsive to feedback from project maintainers and community members.
- I'll iterate on my contributions as needed to align with project objectives and standards, ensuring continuous improvement and alignment with community expectations.

Past contributions

I have demonstrated my commitment to the LLM4S project through **helpful contributions.** My most significant PR implemented a text summarization example that showcases practical usage of the LLM4S API, organizing examples into a dedicated actions package and including a complete workflow with metrics. I also created a reusable summarizeText helper method and improved the library's error messaging. During the process, I took feedback into consideration and implemented the suggestions, making my PR better.

These contributions demonstrate my technical understanding of the project and ability to implement features that enhance the library's usability.

Work Tracking

I will **track my progress using GitHub's tracking platform,** allowing mentors to monitor my work. This will provide a clear record of my progress, help facilitate feedback, and showcase my professional approach to project management.

Creation of a Blog

To document my GSoC journey, I will **maintain a blog**, showcasing my progress, challenges, and key learnings. This will serve as both a personal record and a way to share insights with the broader open-source community.

Final Presentation

At the end of the project, I will **present my work** to the **Scala community.** This will be an opportunity to network, receive feedback, and improve my public speaking skills while showcasing my contributions to experienced Scala developers. This step is crucial in engaging with the broader community and ensuring that my work is recognized and potentially built upon in the future.

About Me

Education

University: Ecole Polytechnique fédérale de Lausanne (EPFL), Switzerland.
Major: Computer Science (BSc)
Graduation year: 2025

Why this project is important to me

The first time I got practically **exposed to AI agents** was during an **AWS hackathon** that my team and I won. Before that, I had an idea what agents were but it was only when I actually tried it in a specific application that I was really amazed. We used browser use, a tool that gives browser access to the agent. It was really surprising on the first try, it only took one prompt to make the agent send a text for me, and even have a follow up conversation with that person.
Here is a link to the linkedin post for more info about the project and the GitHub link.

I'm **passionate** about the **world of agents**, I think the different possibilities to **build meaningful applications** has never been so high. Nothing would be more satisfying for me than actively contributing to making **LLMs accessible in Scala.**
Moreover, as an EPFL student, I've had the chance to be taught Scala from Martin Odersky himself, the designer of Scala.

Thus, this project perfectly aligns with my current skills and learning goals, offering a hands-on opportunity to dive deeper into the Scala ecosystem and contribute to setting the base for a meaningful, **robust and reliable AI SDK**.

Beyond personal development, contributing to LLM4S allows me to **give back to the open-source community**, fostering innovation and collaboration on a global scale. Knowing that my contributions can benefit countless developers worldwide is incredibly rewarding.

Working on this project also connects me with **experienced developers and mentors**, providing valuable **guidance, feedback,** and **networking opportunities** within the **Scala community.**

In essence, this project represents a **significant milestone** in my journey as a Scala developer and **open-source contributor**, offering the chance to learn, grow, and make a meaningful impact while connecting with a vibrant community of **technology enthusiasts.**

Past Experience

I've always been the kind of person who enjoys diving into **new challenges**, and studying **Computer Science at EPFL** has given me plenty of opportunities to do just that. Over the years, I've worked on all sorts of projects—some academic, some competitive, and some

just for fun—all of which have **shaped how I approach software development**. My recent introduction to open-source contribution has been particularly exciting, and I'm eager to keep pushing in that direction.

Throughout my degree, I've built a solid foundation in computer science, tackling projects that range from **functional programming** assignments to **full-stack development**. I've worked with a variety of languages, including Java, C, Python, Kotlin, and Scala, adapting to different paradigms depending on the problem at hand. Functional programming has been a big focus for me, and I find its principles super useful when structuring complex applications.

One of the highlights of my academic journey has been learning Scala directly from **Martin Odersky**, the **creator of the language**. His course at EPFL provided deep insights into functional programming and the **design philosophy behind Scala**. It wasn't just about learning syntax, it was about understanding the reasoning behind the language's features and how to write expressive, scalable code. This experience has given me a **strong foundation in Scala** and made me even more excited to contribute to projects in the ecosystem.

One of the most exciting experiences I've had was **winning an AWS** hackathon. My team and I built a working prototype under tight deadlines, and it was a crash course in rapid development, cloud technologies, and teamwork. **Hackathons** in general have been a great way for me to improve my ability to think on my feet and **develop solutions under pressure.**

I also worked on a mobile app in Kotlin with a team of seven people, which really drove home the **importance of collaboration in software development**. From setting up version control workflows to following agile development practices, I got to experience the full development cycle firsthand. Beyond just coding, I learned how to **communicate effectively** with teammates, handle merge conflicts, and build something that actually works for users.

More recently, I started contributing to open-source through the LLM4S repository, where I've already had **two pull requests merged**. This has been a great way to get hands-on experience with a real-world codebase and to better understand how large-scale projects are structured. It's also been a valuable learning experience in terms of working with a community, following contribution guidelines, and writing code that fits into an existing architecture.

Beyond software development, I've also had experiences that have shaped the way I think and work. I spent a summer as a Founder's Associate at Voltfang, a startup building smart battery storage solutions. My role involved working on everything from business development to **process automation**, giving me a broader perspective on **how technology fits into real-world industries.** I also work as a student partner at a venture capital fund, where I help evaluate startups, interview founders, and think critically about what makes a product or business model work.

All of these experiences, whether in **software, startups,** or venture capital have reinforced my love for **solving problems** and **building things that matter.** Contributing to **open source** feels like the **next natural step** for me, and I'm excited about the opportunity to apply what I've learned while continuing to grow as a developer.