Team workflow and tasks

Multi-Agent Workflow and SHL Specification

json


```
{
  "teams": [
    {
      "name": "Team Alpha",
      "project_manager": {"agent_id": 1, "name": "Alice", "role": "Project
Manager", "persona": "Experienced coordinator with systems background"},
      "agents": [
        {"agent_id": 2, "name": "Bob",   "role": "Research Lead",
"persona": "Analytical researcher focusing on technical depth"},
        {"agent_id": 3, "name": "Carol", "role": "Developer",
"persona": "Skilled programmer implementing core features"},
        {"agent_id": 4, "name": "Dave",  "role": "Quality Analyst",
"persona": "Detail-oriented tester ensuring correctness"}
      ],
      "tasks": [
        "Restructure the original prompt into an efficient, structured format
(e.g. JSON) to guide the process",
        "Gather background research on SHL and related technologies",
        "Investigate optimization techniques (e.g. DeepSeek’s MoE and caching)
for integration",
        "Coordinate with Team Beta to align on requirements and prevent
context shifts"
      ]
    },
    {
      "name": "Team Beta",
      "project_manager": {"agent_id": 5, "name": "Eve",   "role": "Project
Manager", "persona": "Organized leader with documentation expertise"},
      "agents": [
        {"agent_id": 6, "name": "Frank", "role": "Technical Writer",
"persona": "Experienced writer creating detailed specifications"},
        {"agent_id": 7, "name": "Grace", "role": "UX/Designer",
"persona": "User-focused designer crafting clear diagrams and examples"},
```

```
        {"agent_id": 8, "name": "Heidi", "role": "Integration Lead",
"persona": "Engineer ensuring consistency and code integration"}
      ],
      "tasks": [
        "Develop the comprehensive SHL specification document based on Team
Alpha's research",
        "Review and refine content (proofread, grammar, clarity) for the spec
and prompt structure",
        "Draft optimization strategies (e.g. caching, quantization) based on
DeepSeek and other models",
        "Perform peer-review and grading: provide detailed feedback, suggest
improvements, and update the spec as needed"
      ]
    }
  ],
  "process": "Each team has a Project Manager who delegates tasks and can
reassign roles based on agent strengths. Agents collaborate on complex sub-
tasks but maintain clear roles: helper agents offer feedback without
overriding, and the lead agent asks for assistance (e.g. proofreading or
grading).",
  "guidelines": [
    "Use a hierarchical multi-agent design: each team's manager coordinates
specialists in research, writing, testing, etc., mirroring a typical project
org structure:contentReference[oaicite:0]{index=0}.",
    "Apply multi-agent benefits: dividing work among specialized agents can
improve success rates by up to ~70% compared to a single
agent:contentReference[oaicite:1]{index=1}.",
    "Integrate DeepSeek optimizations: plan to use techniques like context
caching and low-precision models to reduce inference
cost:contentReference[oaicite:2]{index=2}:contentReference[oaicite:3]
{index=3}.",
    "Maintain accuracy: each contribution is peer-reviewed. Helper agents
provide detailed grades and justification, encouraging refinements as needed."
  ]
}
```


SHL Technical Specification

Figure: Example of a technical team preparing a detailed specification. SHL (the **Super Hybrid Language**) is a proposed domain-specific programming framework designed to coordinate multi-agent workflows and optimize large-language-model (LLM) collaboration. In this specification, we describe **what SHL is, why it was created, and how it works**. A technical specification serves as a blueprint bridging project goals and implementation monday.com. It outlines SHL's intended behavior, core features, performance requirements, and development plan. After reading this document, all stakeholders (developers, managers, and involved LLM agents) will know how SHL is designed, what it can and cannot do, and how to implement it monday.com. We present both the **desktop-focused version** and a planned alternative platform as a fork (e.g. a cloud or web variant). Throughout, we ensure clarity on features, rules, and definitions, integrating research-driven optimizations (such as DeepSeek-inspired caching and Mixture-of-Experts) to minimize token cost and maximize accuracy.

Key Sections

- **Product Overview** – Introduce SHL: its purpose, high-level design, and motivation. Explain *what SHL does and why*, including target users and use cases. This section lays out the vision and will align the team on the problem SHL solves.
- **Features & Requirements** – List and describe SHL's capabilities. For example, this might include the supported programming constructs, concurrency model, data types, and user-facing functionality. Specify system requirements (e.g. target OS: Windows/Linux for the desktop version) and dependencies (e.g. required runtimes or libraries). This corresponds to the "Features and requirements list" of a spec sheet nulab.com.
- **Technical Specifications** – Detail the internal design and implementation plans. This may cover the language syntax, compiler/interpreter architecture, memory management strategy, data structures, and integration with LLM APIs (Gemini, Claude, etc.). Provide diagrams or tables if helpful. Include parameters like expected performance (e.g. target inference latency) and constraints (e.g. memory usage, platform limitations). These precise details are akin to the "Technical specifications" in a product data sheet openstrategypar....

- **Optimizations and Token Efficiency** – Outline how SHL minimizes resource usage. For example, incorporate *context caching* so repeated prompts reuse previous results (inspired by DeepSeek's disk caching) intuitionlabs.ai api-docs.deepse... . Explain using mixed-precision quantization or a Mixture-of-Experts (MoE) model to reduce compute per query intuitionlabs.ai . If relevant, note that the design is open-source so large users can self-host with low incremental cost (as DeepSeek enables) intuitionlabs.ai . Summarize DeepSeek's pricing model: input tokens cost much less when cached (e.g. ~$0.14 vs $0.55 per million) datastudios.org , so designing for repeated prompt structure yields up to ~90% savings intuitionlabs.ai datastudios.org .
- **Development Timeline** – Provide a schedule of milestones: e.g. design specification complete, prototype implementation, testing phases, release candidate, and final release. Assign these to sprints or dates. This ensures that work is planned and trackable.
- **Risks and Challenges** – Identify potential issues. For example, SHL's reliance on LLMs means we must handle latency variability and version compatibility. There may be debugging complexity in a multi-agent system. We must also mitigate semantic drift when passing context between modules. Listing these upfront helps allocate contingency efforts.
- **User Stories / Personas** – (Optional) Describe how end-users will interact with SHL. For instance: *"As a data scientist, I want SHL to automatically chunk a multi-document analysis task across agents, so that I get a comprehensive summary efficiently."* Using a persona framework ensures the features align with real needs.
- **Glossary and Definitions** – Define any specialized terms or acronyms (e.g. "Mixture-of-Experts (MoE)", "context caching", the "R1 model") used in the spec for clarity.

Design and Architecture

SHL's architecture follows the multi-agent paradigm: distinct components handle tasks in parallel, coordinated by a central controller. We adopt a **hierarchical multi-agent pattern** nagainfo.com : each team has a lead agent (Project Manager) who breaks down the project into subtasks and oversees specialized agents (e.g. research, development, testing). This mirrors a managerial structure where a top-level manager delegates to sub-team leads nagainfo.com . Such specialization enhances scalability and clarity. In practice, one agent might focus on information retrieval (feeding data), another on code generation, and another on verification; all feeding into SHL's unified output. Collaboration between agents is sequential when needed (pipeline flow) or parallel for independent subtasks. As research shows, coordinating multiple agents significantly boosts success on complex tasks (by ~70%) compared to a lone agent nagainfo.com .

The **data flow** in SHL works as follows: input tasks (user queries, code snippets, data streams) enter the SHL front-end, which uses context caching to map repeated patterns to existing computations api-docs.deepse... . The core engine then splits the task into logical units (according to the user's instructions), dispatching them to the appropriate LLM-based modules or code functions. Results are aggregated and post-processed. We will provide sequence diagrams and flowcharts in the full specification doc.

Optimizations and DeepSeek Techniques

To minimize token usage and cost, SHL leverages techniques inspired by DeepSeek's breakthroughs:

- **Context Caching:** As shown in DeepSeek's API, repeated message prefixes yield cache hits api-docs.deepse... . We will design SHL prompts so that stable context (e.g. system instructions or common data) is reused across queries. This can cut costs by ~75–90% for repeated prompts intuitionlabs.ai , since cached tokens cost only a few cents per million intuitionlabs.ai datastudios.org .
- **Mixture-of-Experts & Sparse Activation:** DeepSeek-V3 demonstrates that a large model (671B parameters) can activate only a subset (~37B) per query via MoE intuitionlabs.ai . SHL's model selection will consider similar sparse-activation architectures to reduce compute. For instance, some tasks might use a smaller, specialized model ("expert") instead of always calling the largest model.

- **Quantization:** We adopt low-precision arithmetic (e.g. 4-bit or 8-bit weights) to speed up inference without significant accuracy loss intuitionlabs.ai. This mirrors DeepSeek's use of 4-bit quantization for faster, more efficient runs. In practice, this allows SHL to run on moderate hardware and reduce latency/power.
- **Open-Source Deployment:** SHL will be released as open-source. Organizations can self-host SHL components on their servers, paying only for electricity and GPU time (similarly to DeepSeek's model) intuitionlabs.ai. This eliminates licensing fees and further reduces marginal cost.

These optimizations are critical because, like DeepSeek's R1 model, SHL aims for **large context capacity** (handling long documents or codebases). For example, DeepSeek R1 supports a 64K-token window datastudios.org, enabling multi-document analysis. SHL's design targets a comparable context window to manage extensive user inputs in one session.

Additional Considerations

- **Accuracy and Validation:** We will implement iterative testing and validation. For code generation tasks, SHL will include a verification step (e.g. running test cases) to ensure correctness. Each agent's output is cross-checked by others when possible, as recommended for multi-agent systems to avoid errors.
- **Scaling Up:** Although this specification focuses on the desktop-based version, we anticipate a future fork (e.g. web/mobile). We design SHL modularly so that core logic can be reused in different environments without loss of fidelity. Any environment-specific details will be added in an extension of this spec.
- **Documentation and Tooling:** SHL's spec will be accompanied by thorough documentation, examples, and developer guides. We plan to use version control (e.g. Git) with continuous integration tests. The spec itself will live in a central repository as the "single source of truth," per best practices for product data sheets openstrategypar... .

Summary

In summary, this SHL specification provides a comprehensive blueprint covering **what we're building**, **why we're building it**, and **how it will work** monday.com monday.com. By following these guidelines and incorporating proven strategies from multi-agent AI and cost-optimized LLM research, SHL will be robust, efficient, and transparent. This document, once finalized, will guide developers and AI agents alike in implementing every feature of SHL without ambiguity.