Agentic SDLC with Ollama — Workflow Documentation

# 1. Overview

This project implements an agentic Software Development Lifecycle (SDLC) workflow powered by Ollama-hosted LLMs. It orchestrates multiple agents (Product, Design, Code, Test, Deploy, etc.) that sequentially process a given specification to generate artifacts such as product docs, design diagrams, code, test cases, and deployment steps.

# 2. Architecture

Components:  
1. Agents (internal/agents) - Each agent wraps an LLM and executes a specific SDLC phase.  
2. Orchestrator (internal/orchestrator) - Runs the workflow end-to-end.  
3. Contracts (internal/contracts) - Defines Agent, Orchestrator, and Report interfaces.  
4. CLI (cmd/agentd) - Entry point.

# 3. Workflow

Step 1: CLI Execution  
go run ./cmd/agentd -spec examples/sample-spec.md -model deepseek-r1 -out .workspace  
  
Step 2: Orchestration  
simpleOrchestrator.Execute(ctx, spec)  
  
Step 3: Agent Execution  
Agents build system + human prompts, call the LLM, and return output.  
  
Step 4: Context Management  
Each agent sees only the original spec + last agent’s output. Prevents context overflow.

# 4. Data Flow

Spec.md -> Orchestrator -> Product -> Design -> Code -> Test -> Deploy -> Outputs + Combined.md

# 5. Outputs

Artifacts generated under `.workspace/outputs`:  
- 01-Product.md  
- 02-Design.md  
- 03-Code.md  
- 04-Test.md  
- 05-Deploy.md  
- combined.md (all phases together)

# 6. Context Window Control

Controlled in baseLLMAgent.Run via llms.WithMaxTokens(2048).  
Each agent gets Spec + Last output only.

# 7. Example Run

== Agentic SDLC with Ollama (deepseek-r1) ==  
  
Artifacts:  
 - 01-Product → .workspace/outputs/01-Product.md  
 - 02-Design → .workspace/outputs/02-Design.md  
 - 03-Code → .workspace/outputs/03-Code.md  
 - 04-Test → .workspace/outputs/04-Test.md  
 - 05-Deploy → .workspace/outputs/05-Deploy.md  
  
Done. Combined output in .workspace/outputs/combined.md

# 8. Extensibility

Add new agents by defining new roles.  
Swap models (deepseek-r1, llama3, etc.) via CLI flag.  
Adjust context with WithMaxTokens.