Document Builder Project Plan

Automated Modular Document Generation System

Author: Mohan Raj

Date: 2025-10-03

# Table of Contents

Auto-generated from headings

# 1. Project Overview

The Document Builder is a modular system that generates well-structured documents based on user-provided topics. It uses SwarmGo agents for orchestration and LLMs (Ollama, LongCat) for content generation. Documents can include diagrams, flowcharts, formatted text, and output in DOCX or PDF.

# 2. Goals & Objectives

* - Generate documents automatically from topics.
* - Include diagrams and flowcharts where relevant.
* - Support multiple output formats (DOCX/PDF).
* - Modular architecture using SwarmGo.
* - Multi-LLM integration with caching and logging.

# 3. System Architecture

## 3.1 Components

|  |  |
| --- | --- |
| **Component** | **Role** |
| **Orchestrator** | Coordinates agents, handles workflow. |
| **Content Agent** | Generates text content using LLMs. |
| **Diagram Agent** | Creates diagrams, flowcharts, UML, architecture visuals. |
| **Style Agent** | Formats content, applies DOCX/PDF templates. |
| **Validation Agent** | Reviews content for completeness, grammar, and correctness. |
| **Caching Agent** | Stores prompt-response pairs to avoid redundant LLM calls. |

## 3.2 Workflow Diagram

Flowchart Placeholder:

User Input (topic, format, options)  
 |  
 v  
 Orchestrator (SwarmGo)  
 |  
 ---------------------  
 | | |  
Content Diagram Style  
 Agent Agent Agent  
 | | |  
 ---------------------  
 |  
 Validation Agent  
 |  
 Output Generator  
 (DOCX / PDF)

# 4. User Input Specification

* - Topic: string describing the document topic.
* - Format: docx or pdf.
* - Include diagrams: boolean.
* - Target audience: developer, manager, student, etc.
* - Optional: Level of detail.

# 5. LLM Integration

Content Agent uses Ollama or LongCat for generating text content. Diagram Agent can use LLMs to generate Mermaid/PlantUML code. Caching: JSON/SQLite-based cache to prevent repeated API calls. Logging: store prompts and LLM responses in daily log files.

# 6. Document Output Plan

## 6.1 DOCX

* - Headings, tables, bullets.
* - Embedded images/diagrams.
* - Table of contents.
* - Styles applied consistently.

## 6.2 PDF

* - Same as DOCX but converted using PDF engine (WeasyPrint / pdfkit / gofpdf).
* - Diagrams: generate SVG/PNG → embed in document.

# 7. Development Milestones

|  |  |  |
| --- | --- | --- |
| **Milestone** | **Description** | **Duration** |
| **M1** | Orchestrator + SwarmGo skeleton | 1–2 weeks |
| **M2** | LLM agent integration (Ollama & LongCat) | 1 week |
| **M3** | Content & outline generation | 1–2 weeks |
| **M4** | Diagram generation agent | 1–2 weeks |
| **M5** | Style & formatting agent | 1–2 weeks |
| **M6** | Validation agent | 1 week |
| **M7** | Output generation (DOCX/PDF) | 1 week |
| **M8** | Optional features (multi-language, interactive diagrams) | 2 weeks |

# 8. Optional Enhancements

* - Interactive PDFs with hyperlinks.
* - Multi-language support.
* - User-editable sections before final output.
* - Versioning for document updates.

# 9. Suggested Libraries & Tools

* - Golang: SwarmGo, unidoc, gofpdf
* - Python (optional): python-docx, pdfkit, WeasyPrint
* - Diagram Tools: Mermaid, Graphviz, PlantUML
* - Logging & Caching: JSON/SQLite, daily log files
* - LLM API: Ollama (local), LongCat (cloud)