Content Generation Module

Compliance-First AI Content Generation (POC)

---

1. Purpose of This Module

This document focuses **only on content generation**, not on rule updates or governance workflows.

It explains: - How user prompts are converted into compliant content - How AI models are orchestrated safely - How compliance is enforced **before, during, and after generation** - How quality, cost, and predictability are maintained

This module should be read after the **Master Overview Document** and before deeper technical modules.

---

2. Problem with Traditional AI Content Generation

Most AI content tools follow this pattern:

```
Generate content → Check compliance → Fix violations
```

This creates problems in regulated domains: - Non-compliant content is generated first - Rewrites increase token usage and cost - Manual review loops slow delivery - Compliance becomes reactive instead of proactive

---

3. Core Design Principle

> **Compliance must shape content generation, not correct it afterward.**

In this system: - Rules are loaded **before generation** - Prompts are constrained **before reaching the LLM** - AI is treated as a language engine, not a decision maker

---

4. Supported Content Generation Inputs

4.1 Prompt-Based Generation

Users (agents) provide natural language prompts such as:

> “Create a marketing paragraph for a Term Life Insurance product.”

The system assumes: - Users are not prompt experts - Prompts may be vague or incomplete - Compliance intent may be missing

---

4.2 Document-Assisted Generation

Users may upload: - PDF - DOCX - Markdown

Use cases: - Rewrite existing brochures - Summarize policy text - Generate compliant variants

Uploaded documents **do not bypass rules**.

---

5. Content Generation Workflow (Step-by-Step)

```
User Prompt / Document
        ↓
Prompt Enhancer
        ↓
Load Active Compliance Rules (SQL DB)
        ↓
Retrieve Regulatory Context (Vector DB – RAG)
        ↓
(Optional) Web Search Wrapper (Language Only)
        ↓
LLM Content Generation (AWS Bedrock)
        ↓
Strict AI Compliance Review (AWS Bedrock)
        ↓
Deterministic Rule Validation
        ↓
Final Compliant Content
```

---

6. Prompt Enhancer (Pre-Generation Control)

6.1 Why Prompt Enhancement Is Required

User prompts often: - Lack constraints - Miss disclaimers - Trigger multiple retries

This increases: - Token usage - Cost - Risk of violation

6.2 What the Prompt Enhancer Does

The prompt enhancer: - Clarifies intent - Injects compliance constraints - Normalizes tone and scope - Produces a **single high-quality prompt**

Example:

User Prompt

```
Write an ad for term insurance
```

Enhanced Prompt (Internal)

```
Generate a professional marketing paragraph for a Term Life Insurance product
in India.
Constraints:
- Do not claim guaranteed returns
- State benefits are subject to policy terms
- Include a mandatory disclaimer
Tone: Clear, compliant, consumer-friendly
```

---

7. Regulatory Grounding (RAG)

7.1 Purpose of RAG

RAG ensures the LLM is **grounded in actual regulatory language**, not assumptions.

7.2 How It Works

- Regulatory documents are chunked and embedded
- Relevant clauses are retrieved per request
- Context is injected into the generation prompt

7.3 What RAG Is NOT

- It does not decide compliance
- It does not replace rules
- It does not update databases

---

8. Optional Web Search Wrapper (Language Enhancement)

8.1 Purpose

A Perplexity-style web search wrapper may be used to: - Improve phrasing - Improve readability - Improve marketing tone

---

8.2 Strict Constraints

- Web results are **temporary**
- Web content is **sanitized**
- No facts, numbers, or claims are copied
- Compliance rules always override

Web search improves *how content sounds*, not *what it claims*.

---

9. LLM Content Generation (AWS Bedrock)

9.1 Role of the Generator Model

The generator model: - Produces human-readable marketing content - Operates within strict constraints - Does not make compliance decisions

9.2 Model Characteristics

- Deployed via AWS Bedrock
- No fine-tuning in POC
- Prompt-driven behavior

---

10. Strict AI Compliance Reviewer

10.1 Purpose

A second LLM reviews generated content with a **zero-tolerance mindset**.

10.2 Reviewer Responsibilities

- Identify risky language
- Flag rule violations
- Output structured signals (JSON)

The reviewer **cannot approve content**.

---

11. Deterministic Rule Validation

After AI signals are produced:

- SQL-stored rules are applied deterministically
- Hard rules override AI output
- Soft rules trigger auto-fix or annotations

Final decisions are **code-driven, not AI-driven**.

---

12. Final Output to the User

The user receives: - Final compliant content - Compliance status - Rule references - Explanatory notes

Example:

```
{
  "status": "approved",
  "notes": "Disclaimer automatically added",
  "rules_checked": ["IRDAI-3.4", "IRDAI-5.1"]
}
```

---

13. Cost & Quality Control

Content generation costs are controlled by: - Prompt enhancement (fewer retries) - Rule short-circuiting - Structured reviewer outputs - Controlled token limits

Quality is ensured by: - RAG grounding - Dual-model validation - Deterministic enforcement

---

14. What This Module Does NOT Cover

This document intentionally excludes: - Rule creation or updates - Admin workflows - Fine-tuning strategies - Deployment automation

These are covered in separate modules.

---

15. Key Takeaway

> This content generation module demonstrates how AI can safely generate compliant financial content by enforcing regulatory constraints **before generation**, validating outputs **after generation**, and keeping final authority in deterministic systems.

This module serves as the definitive reference for content generation behavior in the Compliance AI POC.