# Content Management System (CMS) – Requirement Document

The goal is to build a **Contextual Content Management System (CMS)** that allows users to read relevant help documentation while configuring any assets in the application.
The system must support dynamic content, version history and an AI-enabled auto-documentation.

This CMS will provide both:

- **Admin capabilities** (content creation, updates, publishing)
- **User-facing content retrieval** (contextual help to assist the users for configuring the assets)

## Objectives

- Provide contextual documentation for every asset type. Contextual documentation is a help/documentation section that is shown to the user depending on the context of the asset the user is configuring. It helps providing real time, asset specific help and guide users to understand and configure components without leaving the UI.
- Allow documentation teams or admins to create, manage, version, and publish help content.
- Ensure traceability of changes with full audit logs.
- Make documentation scalable using AI-enabled content extraction.
- Expose API interfaces to be consumed by UI and automation workflows.

## Functional Requirements

### Content Creation & Storage

#### Templates

- System must support predefined content templates for different asset types which can be used for different assets. Template defines the structure or layout of documentation for an asset
- Example, A Template can define the following sections:
    - Title
    - Description
    - Overview
    - Configuration fields explanation
    - Examples
    - Best practices
    - Limitations
    - Links to external docs

#### Backend Storage

- Content must be stored in structured format (e.g., JSON schema or Markdown).
- Each content entry must have:
  - Template ID
  - Content ID
  - Version number
  - Author
  - Timestamp
  - Status (Draft / In Review / Published)
  - Associated asset type
  - Actual documentation content
- Support hierarchical storage (template → sections → subsections).

### Versioning

- Every content update must create a new version.
- Older versions must remain accessible.
- Changes should not overwrite published versions.
- Support rollback to previous versions.

## Content Publishing

### Publish API

- API to publish new or updated content.
- Only allowed if:
  - All validations pass
  - Required metadata exists

### Update API

- Update existing content with:
  - Revised text
  - New sections
  - Fixes
- Update flow must not overwrite the published version until publish action is explicitly taken.

## Auditing

### Audit Logs

Each action must be logged:

- Created by (User ID)
- Edited by
- Content id
- Version
- Action type (Draft/In review/Publish/Unpublish)

### Workflow Support

States:

- Draft
- In Review
- Published
- Deprecated

Transitions:

- Draft → Review
- Review → Published OR Draft (If requested changes)
- Published → Deprecated

## Content Retrieval APIs

### Fetch Latest Content

API Requirements:

- Given a content id, return the latest published version.
- Must return in the format requested (Markdown/JSON/HTML).
- Support localization.

### Fetch Previous Versions

- Query parameters:
  - Content Id
  - version number
- Should return metadata (author, timestamp) + content.

## AI-Powered Content Engine

### User Input Options

- Upload PDF, Word, Markdown, or Text file.
- Provide a URL to fetch documentation from the internet.

### AI Processing

AI must extract relevant information:

  - Overview
  - Configuration fields
  - Data type constraints
  - Examples
  - Error handling guidelines
- Generate a structured output aligned with CMS templates.

### Admin Review

- Auto-generated content goes into Draft state.
- Admin can review, modify, and publish.

## Web-Based UI Template Editor

### Load Existing Template

- Fetch template via API to create a content.
- Display last saved version of content for updates.

### Rich Text Editing Features

- WYSIWYG editor with:
  - Bold/Italic/Underline
  - Headings
  - Bullet lists
  - Code blocks
  - Image upload (stored in CMS or external bucket)
  - Hyperlinks
  - Inline formatting
- Ability to reorganize sections through drag-and-drop.

### Version History Display

- Show all previous versions of content with timestamps.
- Compare content between two versions.
- Allow select version → restore as draft.

## Research Requirements

The system should evaluate whether existing open-source CMS tools can be reused based on below criteria

- Extensibility

- Versioning
- AI integration capability
- Licensing constraints

## Out of scope

### Access Control (RBAC)

- Roles needed:
  - Content Viewer – Can fetch help docs
  - Content Editor – Can draft/edit content
  - Content Approver – Can publish content

    - Admin – Full access, including configuration and cleanup

**Search API**

- Full-text search on:
    - Titles
    - Sections
    - Keywords