CMS-MASTER DOCUMENT

Author	**Akshay Kumar**
Version	1
Approved by	
Reviewed by	**1. Rajesh Rishi** **2. Bhavnesh Sharma** **3. Aadil Raza**
Date	16 Dec 2024

Table of Contents - CMS Documentation

Power of Task Initiator	Dashboard/Home, Pages, Requests, Profile, History	Link
Power of PMO	Dashboard/Home, Pages, Requests, Approval, Profile, History	Link
Power of Designer	Dashboard/Home, Pages, Requests, Approval, Profile, History	Link
Feature List	Multi Language Support, Login/Logout, Dashboard/Home, Roles & Permissions, Pages, Content Requests/Approvals, Users, Profile Section, Audit Logs	Link
CMS-Rules	Role-Based Rules, Page Assignment and Editing Rules, Request Handling Rules, Flow Completion & Data Integrity Rules	Link
High-Level Design (HLD)	System Architecture, Core Components, Key Features	Link
Low-Level Design (LLD)	Database Schema, Tables Structure	Link
Workflow Example	Page Creation Workflow, Drafting, Approval Request, Review, Publishing	Link
ERD and Relationship Details	Entities, Associative Entities, Relationships	Link
Additional Considerations	Scalability, Security, Deployment	Link
Roadmap	Phase 1: Planning, Phase 2: Development, Phase 3: Testing & Deployment	Link

WHAT TO DO

▼ Product Requirements Document (PRD)

Overview

The CMS is a comprehensive content management system designed to streamline content creation, management, and publishing processes. It features role-based access control, version control, and workflow management. The system supports multiple user roles, including Super Admin, Task Initiator, PMO, and Designer, each with specific permissions such as viewing, creating, editing, deactivating, discarding, publishing, and approving content.

Features:

- **Role-Based Access Control:** Roles like Admin, Editor, Verifier, and Publisher with specific permissions.
- **Version Control and Logs:** Tracking changes for auditing and rollback.
- **Workflow Management:** Requests and approvals for publishing content.
- **Multi-Language Support:** Content management in multiple languages.
- **Dashboard/Home:** Centralized dashboard for all roles with insights and analytics.
- **Roles & Permissions:** Detailed role and permission management.
- **Content Requests/Approvals:** Streamlined process for content requests and approvals.
- **Audit Logs:** Comprehensive tracking of user actions and system events.

▼ Roles

- Super Admin
- Task Initiator
- PMO
- Designer
- Etc(can be add more).

▼ Permissions

- **View**: Allows the user to view content, pages, or other resources.
- **Create**: Grants the user permission to create new content, pages, or entries within the CMS.
- **Edit/Update**: Permits the user to modify or update existing content or pages.

- **Deactivate**: Allows the user to **Deactivate** content or pages.
- Discard : Allow user to discard any content or request and can not be restored.
- **Publish**: Allow user to publish the content/Switch to or restore the published content from the any version of content.
- **Approve**: Permission to **approve or reject** submitted content.
- **Settings**: Grants access to the CMS configuration and settings.
- **Manage Users**: Provides the ability to add, edit, or remove users, and assign them roles and permissions.
- **Access Analytics/Reports**: Permission to view reports, analytics, or activity logs in the system.

▼ Power of Super Admin

▼ **Dashboard/Home:**

- Insights : Total pages, total user, total Publish request, total rejected requests, rejection ratio, acceptance ratio,
- View recent page updates(update by who, stage, status, updated content, full details of page)
- Other analytics

▼ Pages:

- Access of all pages
- Live edit any page and direct publish
- Draft the content
- Edit and send the content for review to the designer and designer can just update that and publish, designer cant send it back to super admin
- View page status and content for any ongoing changes
- Access of all versions(page history)
- Can revert back to any version of page

- Assign page to user/role(task initiator, PMO, Designer(verifier)

▼ Users:

- View user list and their profile and assigned permissions and role and pages
- Create/Edit/Activate/Deactivate users profile with password
- Assign/Remove roles
- Upload the Excel to create bulk user

▼ Roles & Permission :

- Create new roles and assign permission
- Activate/Deactivate roles

▼ Content Updating Request:

- View list of all requests
- Review/Reject/Discard/Edit/Approve and publish direct the specific request of any stage
- Reject and revert the request back to the sender with comments
- Edit and publish the content directly
- Publish the request behalf of Designer)
- Schedule/Reschedule the publication

▼ Power of Task Initiator

▼ **Dashboard/Home:**

- Insights
- Rejection/Acceptance ration
- Total Request count
- Recent Request
- ETC.

▼ **Pages**

- Can only access pages assigned by the Super Admin.
- Initiate content editing of assigned pages, then submit for review to PMO.
- See the preview of the edited content
- Draft the content
- Delete (hard delete) the draft
- View associated update history(versions) of the specific page
- Can add Projects, Services, News and blogs, Career details in the page

▼ Requests

- View list of recent submitted request
- Send reminders in the system to the PMO if the approval is pending for a long time.
- Edit only rejected (reverted) request and send again for review with comments

▼ Profile

- See Profile(assigned roles, permissions & pages)
- Update Profile Photo

▼ History

- View history and preview for all the associated request/Changes

---

▼ Power of PMO

▼ **Dashboard/Home:**

- Insights
- Rejection/Acceptance ration
- Total Request count
- Recent Request
- ETC.

▼ **Pages**

- Preview of pages assigned by the Super Admin.
- No direct edit access

▼ Requests

- View list of submitted request by the task initiator for assigned pages
- Compare Live and new content.
- Edit the request and approve or reject or Discard(cant be restored) the request with comments.
- Approved and send request to the designer.

▼ Approval

- View list of the approved request and their status of further stage
- Send the rejected request *(by designer)* back to task initiator with comments
- Edit the rejected (reverted) request (*by designer)* and submit for review to designer with comments
- Send reminders to Designers if content is pending publication.

▼ Profile

- See Profile(assigned roles, permissions & pages)
- Update Profile Photo

▼ History

- View history(versions) and preview for all the associated request/Changes of assigned pages

---

▼ Power of Designer

▼ **Dashboard/Home:**

- Insights
- Rejection/Acceptance ration
- Total Request count

- Recent Request
- ETC.

▼ **Pages**

- Preview of pages assigned by the Super Admin.
- No direct edit access

▼ Requests

- View list of submitted request by the PMOs for assigned pages
- Compare Live and new content.
- Edit the request and Direct Publish or reject the request with comments.
- Edit the request and Direct Publish or reject(send back) or Discard(cant be restored) the request with comments.

▼ Approval

- View list and preview of the approved(published) request.

▼ Profile

- See Profile(assigned roles, permissions & pages)
- Update Profile Photo

▼ History

- View history(versions) and preview for all the associated request*(Published/Rejected)* of assigned pages

---

▼ Feature List

Multi Language Support for content

▼ Login / Logout (All Roles)

Login:

- **Action**: Submit login credentials (username, password) via an API call to authenticate the user.
- **API Call**: `POST /login` – Authenticates the user and returns a session token or JWT.

- **Permissions**: Validates user credentials and assigns roles/permissions based on the account type.

Logout:

- **Action**: User clicks "Logout" to terminate the session.
- **API Call**: `POST /logout` – Logs out the user and terminates the session/token.
- **Permissions**: Invalidates the session token and clears stored credentials.

Password Reset (Non super admin roles)

- Action : User can reset their password.
- **API Call**: `PUT/resetPassword` – Logs out the user and terminates the session/token.
- **Permissions**: Validates the reset token, allows non-super admins to reset only their own password, and requires re-login after a successful reset.

▼ Dashboard/Home (All Roles)

- **View Dashboard Insights**:
  - **Action**: Load dashboard with aggregated data (user count, page count, total requests, etc.)
  - **API Call**: `GET /dashboard/overview` – Returns statistics like total pages, total users, publish requests, rejections, etc.
  - **Permissions**: Ensures role-specific data access (e.g., Super Admin sees all, Task Initiator sees their own data).

- **View Recent Updates**:
  - **Action**: Display list of recent updates to pages, including who made the changes, stage, and status etc.

- **API Call**: `GET /pages/recent-updates` – Fetches the list of recent page updates with metadata.
- **Permissions**: Filters data based on the user's role and their access to pages.

- **Filter Dashboard Insights**:
  - **Action**: Apply filters for the dashboard, like date range, page types, or status.
  - **API Call**: `GET /dashboard/filters` – Allows users to filter insights based on custom criteria.
  - **Permissions**: Filters should be role-based (Super Admin has access to all data; others only to assigned content).
  - **Filter/Search**: **By date range**, **by user**, **by page**, **by status** (pending, approved, rejected).

- **View Notifications/Alerts*(optional)***:
  - **Action**: Display notification or alert (e.g., rejection or approval notifications).
  - **API Call**: `GET /notifications` – Fetches the list of notifications relevant to the user.
  - **Permissions**: Notifications depend on user actions (PMO/Designer would see notifications about content they've reviewed).
  - **Filter/Search**: **By type**, **by date**, **by user**.

▼ Roles & Permissions

- **Create New Role**
  - **Action**: Create a new user role with specific permissions.
  - **API Call**: `POST /roles/create`
  - **Permissions**: Super Admin only.
- **Edit Role**

- **Action**: Edit an existing role to modify its permissions or name.
- **API Call**: `PUT /roles/{roleId}/edit`
- **Permissions**: Super Admin only.

- **Deactivate Role**
  - **Action**: **Deactivate** an existing role that is no longer needed.
  - **API Call**: `PUT /roles/{roleId}`
  - **Permissions**: Super Admin only.

▼ Pages (Super Admin, Task Initiator, PMO, Designer)

- **View Pages**:
  - **Action**: Display the list of all pages or specific assigned pages.
  - **API Call**: `GET /pages` – Retrieves all pages, filtered by permissions for each role.
  - **Permissions**: Task Initiator, PMO, and Designer only see pages assigned to them, Super Admin sees all pages.
  - **Filter/Search**: **By page name**, **by status** (published, draft), **by date** (created/updated).

- **Edit Page Content**:
  - **Action**: Edit the content of a page.
  - **API Call**: `PUT /pages/{pageId}/edit` – Submits changes to the page.
  - **Permissions**: Role-based access: Task Initiators only edit assigned pages; Super Admin can edit any page.

- **Draft Page**:
  - **Action**: Save page content as a draft without publishing.
  - **API Call**: `POST /pages/{pageId}/draft` – Saves the page as a draft.
  - **Permissions**: Task Initiator and Super Admin can draft pages. Designers and PMOs can't draft pages, only approve/reject.

- **Submit for Review**:
  - **Action**: Submit page content for review by PMO or Designer.
  - **API Call**: `POST /pages/{pageId}/submit` – Sends page content to the next review stage.
  - **Permissions**: Task Initiator submits content, PMO/Designer can approve or reject it.

- **Publish Page**:
  - **Action**: Publish the content, making it live.
  - **API Call**: `POST /pages/{pageId}/publish` – Changes page status to "published."
  - **Permissions**: Designer or Super Admin can publish content.

- **Revert to Previous Version**:
  - **Action**: Revert page content to a previous version (undo changes).
  - **API Call**: `POST /pages/{pageId}/revert` – Reverts to the selected historical version.
  - **Permissions**: Super Admin has the ability to revert any page; Task Initiator/PMO/Designer can't revert pages.

- **View Page History/Versions**:
  - **Action**: View page version history.
  - **API Call**: `GET /pages/{pageId}/history` – Fetches a list of past versions of the page.
  - **Permissions**: All roles can view history, but only Super Admin can access all page histories.
  - **Filter/Search**: **By version number**, **by date**.

- **Pagination** :The list should support pagination to handle a large number of users efficiently.

▼ Content Requests/Approvals (Super Admin, Task Initiator, PMO, Designer)

- **Submit Content Request**:
  - **Action**: Task Initiator submits content for review.
  - **API Call**: `POST /requests/submit` – Creates a new request for content submission.
  - **Permissions**: Only Task Initiators can submit requests.

- **View Requests List**:
  - **Action**: View all content requests(list), including status (pending, approved, rejected).
  - **API Call**: `GET /requests` – Retrieves the list of all content requests.
  - **Permissions**: Task Initiator sees their own requests; PMO/Designer sees the ones they are responsible for.
  - **Filter/Search**: **By request type**, **by status**, **by user**, **by date**.

- **Approve/Reject Content Request**:
  - **Action**: PMO or Designer approves or rejects content.
  - **API Call**: `POST /requests/{requestId}/approve` or `POST /requests/{requestId}/reject` – Approve or reject the request.
  - **Permissions**: PMO/Designer can approve or reject. Task Initiators can't approve or reject.

- **Send Request Back for Edits**:
  - **Action**: Send the request back to Task Initiator for revisions (with or without comments).

    - **API Call**: `POST /requests/{requestId}/send-back` – Sends request back for changes.
    - **Permissions**: PMO/Designer can send requests back to Task Initiator for revisions.

- **Send Reminder for Pending Request**:
    - **Action**: Send a reminder to the next stage reviewer (e.g., PMO to Task Initiator, Designer to PMO).
    - **API Call**: `POST /requests/{requestId}/reminder` – Sends reminder for approval or feedback.
    - **Permissions**: Task Initiator can remind PMO, PMO can remind Designer, Super Admin can remind anyone.

- **Schedule/Reschedule Publication**:
    - **Action**: Super Admin or Designer schedules/reschedules content for publication.
    - **API Call**: `POST /requests/{requestId}/schedule` or `POST /requests/{requestId}/reschedule` – Schedules content for future publication.
    - **Permissions**: Super Admin and Designer can schedule/reschedule.

▼ Users (Super Admin Only)

Create New User

- **Action**: Super Admin creates a new user profile with password in the system.
- **API Call**: `POST /users/create` – Creates a new user profile and assigns roles to the user.
- **Permissions**: Only **Super Admin** can create new users.

Upload and Create New User

- **Action**: Super Admin uploads the excel sheet to creates new users profile in bulk with password in the system.
- **API Call**: `POST /users/create` – Creates a new user profile and assigns roles to the user.
- **Permissions**: Only **Super Admin** can create new users.

View Profile of user

- **Action**: Super Admin can see the full profile of any user with history, role, permission and other details.
- **API Call**: `GET /users/{userId}/profile` –Retrieves the user's profile details.
- **Permissions**: Only **Super Admin** can see the profiles of any user.

Edit User Profile

- **Action**: Super Admin edits an existing user profile, such as updating roles, permissions, password and other details.
- **API Call**: `PUT /users/{userId}/edit` – Updates the user’s profile details, including role assignments and permissions.
- **Permissions**: Only **Super Admin** can edit any user's profile.

Assign/Remove Roles for User

- **Action**: Super Admin can assign or remove roles for users based on their responsibilities and permissions.
- **API Call**: `POST /users/{userId}/roles` – Assigns or removes specific roles (e.g., Task Initiator, PMO, Designer).
- **Permissions**: Only **Super Admin** can assign or remove roles.

Deactivate/Activate User

- **Action**: Super Admin can deactivate or activate a user account to manage access control.

- **API Call**: `PUT /users/{userId}/status` – Changes the user's account status to active or deactivated.
- **Permissions**: Only **Super Admin** can deactivate or activate a user account.

Filter/Search User List

- **Action**: Super Admin can search and filter through the user list based on various parameters (role, status).
- **API Call**: `GET /users?filter={criteria}` – Fetches filtered user data by role, status, or other criteria.
- **Filters/Search Criteria**:
  - **By Role**: Filter users based on their assigned roles (e.g., Task Initiator, PMO, Designer).
  - **By Status**: Filter users by their account status (active or deactivated).
  - **By Username**: Search for a specific user by their username.
- **Permissions**: Only **Super Admin** can access the full list of users and apply these filters.

▼ Profile Section (Non-Super Admin Roles)

- **View Own Profile**
  - **Action**: View personal details (roles, permissions, tasks, history, Assigned Pages).
  - **API Call**: `GET /users/{userId}/profile`
  - **Permissions**: View own profile.
- **Edit Own Profile**
  - **Action**: Edit name, contact info, and photo. Cannot change roles/permissions.
  - **API Call**: `PUT /users/{userId}/profile`
  - **Permissions**: Edit own profile only.

▼ Audit Logs

- **Action**: View every activity logs for the entire system (user actions, content edits, etc.).
- **API Call**: `GET /audit-logs`
- **Permissions**: Super Admin only

▼ CMS-Rules

▼ Role-Based Rules

▼ Super Admin

- Can **assign** and **remove** pages from users (Task Initiator, PMO, Designer) through the **Page Section**.
- Can **edit** any page's content from the **Page Section** (only if there are no ongoing processes) and **publish** it directly.
- Can **view** all pages, requests, and approval lists.
- Has the power to **approve**, **reject**, or **discard** any request, whether initiated from the **Page Section** or **Request Section**.
- Can **edit content** directly from the **Request Section**, even when other roles are involved.
- Can **Activate/Deactivate users**, **assign roles**, and **manage permissions**.
- User can not be released from a page if there is any ongoing process is running associated to user and page.

▼ Task Initiator

- Can only **edit pages** if:
  - The page content is in **publish mode** or has **no ongoing processes** (i.e., no requests are pending or in progress for that page).
- Can **submit** content requests for review to PMO.
- Can **draft** page content for submission.
- Cannot **edit** pages directly from the **Page Section** if a request is in progress for that page or if the page is in any other process

state.

- Can only **view** assigned pages and their statuses.

▼ PMO

- Can **view** pages but cannot **edit** them directly from the **Page Section**.
- Can **approve**, **reject**, or **discard** content requests from the **Request Section**.
- Can **submit** requests back to Task Initiator for revision if necessary.
- **Cannot** perform any actions in the **Page Section** related to editing or publishing content.
- Can **approve** or **reject** requests and send content for further review or publishing.

▼ Designer

- Can **view** pages but cannot **edit** them directly from the **Page Section**.
- Can **approve** or **reject** content requests from the **Request Section**, and **publish** the content.
- Can **directly publish** content from the **Request Section** after reviewing the changes.
- Can **discard** content requests (i.e., permanently remove them without further actions).

▼ Page Assignment and Editing Rules

- **Pages** can only be **assigned** or **removed** from users (Task Initiator, PMO, Designer) through the **Page Section** by the **Super Admin**.
- **Page Editing**:
  - **Task Initiator** & **Super Admin** can **edit** a page only if:
    - The page is in **publish mode**.
    - There are **no ongoing processes** (no active requests or review stages).

    - **PMO** and **Designer** cannot **edit** content in the **Page Section** but can **edit** requests through the **Request Section** (for approval).
- **Publishing**:
    - **Super Admin** can **publish** pages directly from the **Page Section**.
    - **Designer** can **publish** content through the **Request Section** after approval.
    - **Task Initiator** cannot publish content but can submit it for review.

▼ Request Handling Rules

Request Section:

- **Task Initiator** can only **submit** content requests for review to PMO.
- **PMO** and **Designer** can:
    - **Approve**: Send requests further for review or publish the content.
    - **Reject**: Send the request back for changes to the Task Initiator.
    - **Discard**: Permanently discard the request (no further review or edit).
- **Super Admin** can:
    - **Approve**, **Reject**, or **Discard** requests from either **Request Section**.
    - **Edit** content directly in **Request Section** if necessary, but not in the Page Section unless there is **no ongoing process**.

▼ **Handling Ongoing Processes & Restrictions**

- **Page Editing and Removal Restrictions**:
    - **Task Initiator & Super Admin** can **edit** only if there are **no active requests or ongoing processes**.
    - **Requests cannot be removed** once initiated, except by the **Super Admin** under specific circumstances (discard or finalize action).
- **Approval Process**:

- If a **Task Initiator** submits a page for review, it will go through stages of approval/rejection:
  - **PMO** can approve or reject.
  - **Designer** can approve/reject/discard or publish after approval.
- **Super Admin** can intervene at any stage for any content request and has full power to approve, reject, or discard requests from either section.

▼ **Flow Completion & Data Integrity Rules**

- **Data integrity** must be maintained through the process. No data should be lost or corrupted due to roles and permissions.
- **If any process is on hold** (e.g., review stage, awaiting approval), **nothing can be removed or discard**.
- Once content reaches the **"discard"** state, it **cannot be recovered**, and no further actions are possible on that request.
- Only the **Super Admin** has the power to override certain states (such as rejecting, discarding, or editing content across the board).

▼ Access Control and Conflict Avoidance

- **Roles should not conflict** with each other. Task Initiators will always be able to make changes to pages only when they are allowed (i.e., no ongoing requests or approval processes).
- There will be **no deadlocks** in the system, as all permissions are carefully partitioned.
- **Only one person** (Task Initiator, PMO, or Designer) will have control of editing a page at any time to avoid conflicts.

▼ Other Key Notes

- **Only Super Admin** can **assign roles** and **remove users** from the system.
- **Task Initiator**, **PMO**, and **Designer** roles cannot **remove users** or change user roles/permissions.
- **Page status and approval flow** must always be tracked to ensure no contradictory actions are performed on a page.

▼ API List for CMS

▼ Dashboard

- **GET** `/dashboard/overview`
  - **Payload**: None
  - **Data Keys**: `{totalUsers, activeUsers, pendingRequests, recentActivities}`
  - **Description**: Get CMS stats overview.
- **GET** `/dashboard/analytics`
  - **Payload**: `{dateRange, metrics}`
  - **Data Keys**: `{pageViews, userEngagement, trafficSources, conversionRate}`
  - **Description**: Get filtered analytics data.

▼ Role & Permissions

- **GET** `/roles`
  - **Payload**: None
  - **Data Keys**: `{id, name, permissions}`
  - **Description**: Get roles list.
- **POST** `/roles`
  - **Payload**: `{name, permissions}`
  - **Data Keys**: `{id, name, permissions}`
  - **Description**: Create role.
- **PUT** `/roles/{id}`
  - **Payload**: `{name, permissions}`
  - **Data Keys**: `{id, name, permissions}`
  - **Description**: Update role.

▼ Users

- **GET** `/users`
  - **Payload**: `{filter, pagination}`

    - **Data Keys:** `{id, name, email, role, status, createdAt}`
    - **Description:** Get users list.
- **POST** `/users`
    - **Payload:** `{name, email, role, password}`
    - **Data Keys:** `{id, name, email, role, status, createdAt}`
    - **Description:** Create user.
- **PUT** `/users/{id}`
    - **Payload:** `{name, email, role}`
    - **Data Keys:** `{id, name, email, role, status}`
    - **Description:** Update user.

▼ **Pages**

- **GET** `/pages`
    - **Payload:** `{filter, pagination}`
    - **Data Keys:** `{id, title, status, author, createdAt, updatedAt}`
    - **Description:** Get pages list.
- **POST** `/pages`
    - **Payload:** `{title, content, status, author}`
    - **Data Keys:** `{id, title, content, status, author, createdAt}`
    - **Description:** Create page.
- **PUT** `/pages/{id}`
    - **Payload:** `{title, content, status}`
    - **Data Keys:** `{id, title, content, status, updatedAt}`
    - **Description:** Update page.

▼ **Requests**

- **GET** `/requests`
    - **Payload:** `{filter, pagination}`

- **Data Keys:** `{id, pageId, type, status, createdBy, createdAt}`
- **Description:** Get requests list.

- **POST** `/requests`
  - **Payload:** `{pageId, type, details}`
  - **Data Keys:** `{id, pageId, type, status, createdBy, createdAt}`
  - **Description:** Submit request.
- **PUT** `/requests/{id}`
  - **Payload:** `{status, comments}`
  - **Data Keys:** `{id, pageId, type, status, updatedBy, updatedAt}`
  - **Description:** Update request status.

▼ **Approvals**

- **GET** `/approvals`
  - **Payload:** `{filter, pagination}`
  - **Data Keys:** `{id, requestId, status, approver, createdAt}`
  - **Description:** Get approvals list.
- **POST** `/approvals/{id}/approve`
  - **Payload:** `{comments}`
  - **Data Keys:** `{id, requestId, status, approver, updatedAt}`
  - **Description:** Approve request.
- **POST** `/approvals/{id}/reject`
  - **Payload:** `{comments}`
  - **Data Keys:** `{id, requestId, status, approver, updatedAt}`
  - **Description:** Reject request.

▼ **Logs**

- **GET** `/logs`
  - **Payload:** `{filter, pagination}`
  - **Data Keys:** `{id, action, user, timestamp, details}`
  - **Description:** Get logs list.

▼ **Profile**

- **GET** `/profile`
  - **Payload**: None
  - **Data Keys**: `{id, name, email, role, preferences}`
  - **Description**: Get profile.
- **PUT** `/profile`
  - **Payload**: `{name, email, password}`
  - **Data Keys**: `{id, name, email, role, updatedAt}`
  - **Description**: Update profile.

▼ **Settings**

- **GET** `/settings`
  - **Payload**: None
  - **Data Keys**: `{key, value, description}`
  - **Description**: Get settings.
- **PUT** `/settings`
  - **Payload**: `{key, value}`
  - **Data Keys**: `{key, value, updatedAt}`
  - **Description**: Update settings.

HOW TO DO

▼ High-Level Design (HLD)

System Architecture

Core Components:

Frontend (Admin UI):

- React-based interface for managing pages, roles, permissions, and workflows.
- Includes a dashboard with analytics and logs.

Backend:

- Node.js with Express.js for API services.
- Prisma ORM for database management with PostgreSQL.

Client UI:

- Next.js application dynamically rendering content from the CMS.

Database:

- PostgreSQL schema managing users, roles, pages, sections, requests, and logs.

Communication:

- REST APIs for CRUD operations.

Key Features

Authentication and Authorization:

- JWT-based authentication.
- Role-based permissions defining access levels.

Content Management:

- CRUD operations for pages and nested sections.
- Dynamic data modeling for reusable page components.

Workflow Management:

- Requests for draft approval, rejection, or publishing.
- Status tracking (e.g., Draft, Pending, Published).

Version Control:

- Maintain page history for rollbacks and audits.

Audit Logs:

- Track user actions and system events.

▼ Low-Level Design (LLD)

Database Schema

Tables

1. **Users**:
    - `id` (UUID, Primary Key)
    - `name` (VARCHAR)
    - `email` (VARCHAR, Unique)
    - `password` (VARCHAR)
    - `roleId` (UUID, Foreign Key → Roles.id)
    - `createdAt` (TIMESTAMP)
    - `updatedAt` (TIMESTAMP)
2. **Roles**:
    - `id` (UUID, Primary Key)
    - `name` (VARCHAR, e.g., Admin, Editor)
    - `permissions` (JSONB)
3. **Pages**:
    - `id` (UUID, Primary Key)
    - `title` (VARCHAR)
    - `slug` (VARCHAR, Unique)
    - `status` (ENUM: Draft, Pending, Published)
    - `content` (JSONB)
    - `authorId` (UUID, Foreign Key → Users.id)
    - `createdAt` (TIMESTAMP)
    - `updatedAt` (TIMESTAMP)
4. **PageHistory**:
    - `id` (UUID, Primary Key)
    - `pageId` (UUID, Foreign Key → Pages.id)
    - `version` (INT)

- `content` (JSONB)
- `createdAt` (TIMESTAMP)

5. **Requests**:
   - `id` (UUID, Primary Key)
   - `pageId` (UUID, Foreign Key → Pages.id)
   - `type` (ENUM: Create, Update, Discard)
   - `status` (ENUM: Pending, Approved, Rejected)
   - `details` (JSONB)
   - `createdBy` (UUID, Foreign Key → Users.id)
   - `createdAt` (TIMESTAMP)
6. **Logs**:
   - `id` (UUID, Primary Key)
   - `userId` (UUID, Foreign Key → Users.id)
   - `action` (VARCHAR, e.g., Create Page, Edit Section)
   - `details` (JSONB)
   - `createdAt` (TIMESTAMP)

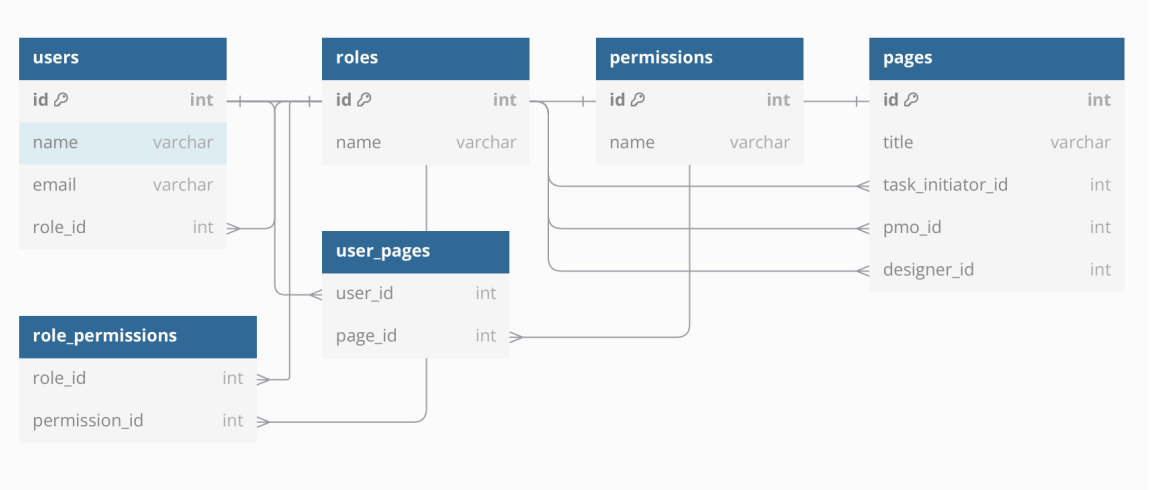
▼ Workflow Example

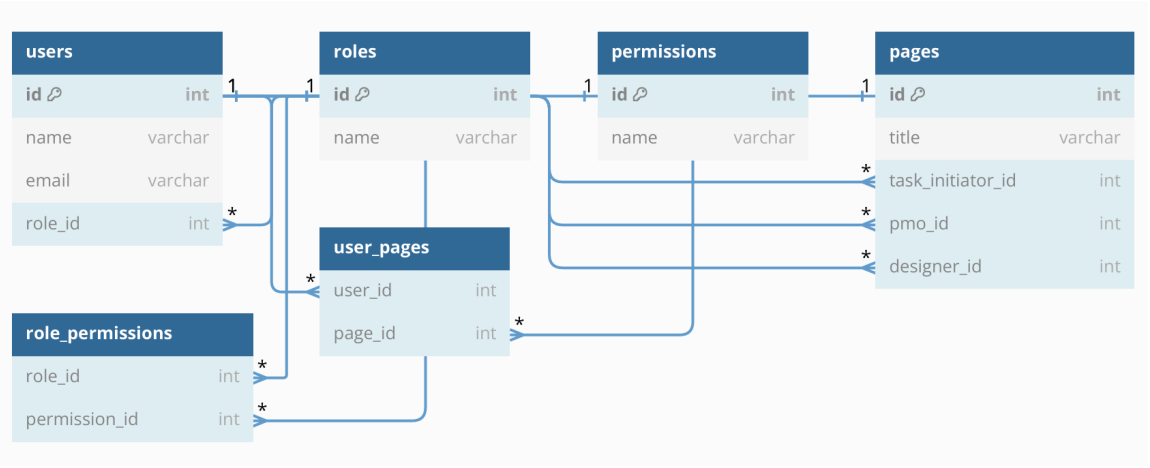
Page Creation Workflow

1. **Drafting**:
   - A user creates a page in Draft status, stored in the `Pages` table.
2. **Approval Request**:
   - A request entry is added to the `Requests` table with status "Pending".
3. **Review**:
   - Verifiers review the request and update its status.
4. **Publishing**:

- Approved content moves to Published status, with a version saved in `PageHistory`.

▼ ERD and Relationship Details





Textual ERD Representation:

1. **Entities**:
   - **User**:
     - Fields: `id`, `name`, `email`, `role_id`
     - Relationships:
       - 1 User → 1 Role (Each user can only have one role)
       - 1 User → Many Permissions (Through Role)

        - 1 User → Many Pages (Through UserPage mapping)
    - **Role**:
        - Fields: `id`, `name`
        - Relationships:
            - 1 Role → Many Users
            - 1 Role → Many Permissions (Through RolePermission)
    - **Permission**:
        - Fields: `id`, `name`
        - Relationships:
            - Many Permissions → Many Roles (Through RolePermission)
    - **Page**:
        - Fields: `id`, `title`, `task_initiator_id`, `pmo_id`, `designer_id`
        - Relationships:
            - 1 Page → 1 Task Initiator (User)
            - 1 Page → 1 PMO (User)
            - 1 Page → 1 Designer (User)
            - 1 Page → Many Users (Through UserPage mapping)
2. **Associative Entities**:
    - **RolePermission**:
        - Fields: `role_id`, `permission_id`
        - Maps: Many Roles → Many Permissions
    - **UserPage**:
        - Fields: `user_id`, `page_id`
        - Maps: Many Users → Many Pages

▼ Additional Considerations

Scalability

- Use database indexing for faster query performance.
- Employ caching (e.g., Redis) for frequently accessed data.
- Implement horizontal scaling for API servers and load balancing.

Security

- Encrypt sensitive data like passwords using bcrypt.
- Secure API endpoints with middleware for role-based access control.

Deployment

- **Frontend**:_____.
- **Backend**: _____.
- **Database**: Managed PostgreSQL on _____.

▼ Roadmap

Phase 1: Planning (Week 1-2)

1. Define requirements and finalize architecture.
2. Set up repositories and workflows.

Phase 2: Development (Weeks 3–15)

1. Implement backend APIs and database schema.
2. Develop Admin UI for content and user management.
3. Build Client UI for content rendering.

Phase 3: Testing & Deployment (Weeks 15–20)

1. Perform end-to-end testing.
2. Optimize for performance.
3. Deploy applications and document usage.
