

1: Build a Chat Agent Platform for Developers

Your challenge is to build a plug-and-play platform that helps developers create custom chat agents in just a few steps. These agents should be powered by LLMs (like [Grog](#)) and work with content stored in Contentstack.

What You Need to Build

1. LLM Model API - An API that:

- Sends and receives chat messages (supports streaming responses)
- Lets developers switch between different LLM providers (like OpenAI, Anthropic, etc.)
- Connects to **Contentstack's Delivery APIs** (via **MCP**) to fetch content based on the user's question

2. Chat SDK - A lightweight SDK that:

- Makes it easy to embed chat agents in any website or app
- Works well with React (ideally using React Hooks)
- Connects to your LLM Model API to power conversations

Developers should be able to use your platform to build **domain-specific chat agents** without needing to understand all the backend logic.

For example, on a travel website using Contentstack, the chatbot should respond to: "What tours are available for Italy?"

Your platform should take care of:

- Understanding the question and querying the right content
- Sending back a useful response — all with minimal setup

Contentstack Products	Purpose
CMS	To model and manage structured content
Launch	To build or embed your chat UI using its APIs
Contentstack MCP	To connect to the content delivery layer and pull live content
Developer Hub (OAuth)	To register the app and access stack-level APIs

2: Build a Smart, Embeddings-Based Search App for Contentstack

Your task is to build a **Contentstack Marketplace App** that adds **semantic search** to any stack. Unlike basic keyword search, this app should understand the meaning behind what users are looking for and return the most relevant content — even if the exact words don't match.

What You Need to Build

1. Semantic Search Engine

- Accept free-form, natural language search queries (e.g., “red high top sneakers with stripes”)
- Return the most relevant entries using **semantic similarity**, not just keyword matching
- Rank results based on how closely they match the intent of the query

2. Embeddings Integration

- Use LLM-based embeddings (like **OpenAI**, **Cohere**, or open-source models)
- Represent content entries and queries as vectors in a shared space
- Use vector similarity to compare and rank entries

3. CMS Sync & Indexing

- Keep your search index up to date with content changes
- Use **Contentstack Webhooks** to automatically update the index when entries are created, updated, or deleted
- Ensure low-latency updates so new content is searchable quickly

4. Marketplace App UI

Build a UI inside Contentstack (hosted via **Launch**) where users can:

- Enter search queries and see ranked results with similarity scores
- Optionally filter by content type or locale

5. Scalability

- Your search system should handle **thousands of entries** across multiple content types
- Use tools like **FAISS**, **Weaviate**, or **Pinecone** for fast and efficient vector search

Contentstack Products	Purpose
CMS	To store and manage all your content entries
Launch	To host your Marketplace App and UI inside Contentstack
Automate	To keep the search index synced with content changes

3: Smart Find and Replace for Editors

Build a **Contentstack Marketplace App** that helps editors perform smart, context-aware **find and replace** across their content — going beyond basic text replacement.

What You Need to Build

1. Contextual Replacement

- Don't just replace words — understand the meaning.
- Example: Replace “*Gemini 2.5 Pro*” with “*Claude Sonnet*”, not “*Claude 2.5 Pro*”.

2. Smart Link Updates

- When replacing text inside hyperlinks, update **both** the display text and the URL.
- Example: Change “Google” linking to <https://google.com> → “Bing” linking to <https://bing.com>.

3. Named Entity Replacement

- Find and replace named entities like company names, emails, etc.
- Example: Replace “Alpha Company” and <mailto:contact@alphacompany.com> with “Omega Corporation” and new email.

4. Deep Content Coverage

Your app must scan and edit Rich Text fields, Tables and nested components, Links, metadata, and custom fields

5. Brandkit Integration

Use **Brandkit** to enforce brand-approved terms. Avoid banned or outdated phrases. Maintain consistent tone and style

Contentstack Products	Purpose
CMS	Access and edit content entries
Brandkit	Suggest smart, brand-aware replacements
Launch	Host your app UI within the Contentstack dashboard
Automate	Trigger follow-up actions

4. Build a Locale Variants App for Translations

Your challenge is to build a Marketplace app that helps teams manage multilingual content using variant groups and fallback logic — all within Contentstack.

What You Need to Build

1. Variant Manager UI - An interface where users can:
 - a. Create variant groups (e.g., Languages)
 - b. Add locales (e.g., Hindi, Marathi) with fallback rules (e.g., Marathi → Hindi → English)
2. Entry Sidebar Extension - A sidebar that:
 - a. Shows all variants for a given entry
 - b. Highlights missing translations
 - c. Lets users generate or override translations (manually or via AI)
 - d. Supports bulk variant creation for content types
3. Personalize API Integration - Use the Variants API (via Contentstack Personalize) to:
 - a. Store and retrieve locale-specific versions
 - b. Attach variant params like `marathi_hindi_en` to each locale
 - c. Implement fallback logic when a field is missing

Developers and content teams should be able to:

- Define a fallback hierarchy for locales
- Deliver the closest available translation without manual checks
- Use the variant param in frontend code to fetch the correct version

Contentstack Products	Purpose
CMS	To manage localized entries
Launch	To visualize and preview variant-driven pages
Personalize API	To store and resolve variant fallback logic
Developer Hub (OAuth)	To register the app and access stack-level APIs