🧱 **Component Breakdown**

**1. Document Upload Panel**

Platform: Web/Mobile

Tech: React / React Native / Flutter

Features: Upload PDFs, images; drag-and-drop; mobile scan; metadata tags

**2. Cloud Storage (S3)**

Role: Stores raw documents securely

Trigger: AWS Lambda/EventBridge → kicks off the AI agent

**3. AI Classification Agent**

Subcomponents:

OCR: AWS Textract / Tesseract / Claude-Vision

Content Embeddings: OpenAI/GPT-4 / Amazon Comprehend / BERT

Classifier: Prompt-based LLM or fine-tuned model

Output:

Raw metadata

Predicted labels

Embeddings for clustering

**4. Category Manager**

Modes:

User-Defined Schema: Predefined JSON/YAML of categories

AI-Inferred: Clustering + keyword extraction + summarization

Features:

Category suggestions

Top keywords/samples per cluster

Schema validation

**5. Human-in-the-Loop Review Panel**

Platform: Web UI (Next.js)

Core Features:

View category clusters

See sample documents

Approve/reject labels

Give natural language feedback

Write rules (e.g., “Docs with ‘1040’ → Tax Form”)

Learning:

Few-shot examples stored

Rules converted into labeling functions

Prompt updated incrementally

**6. Config Generator**

Output:

Final document labels

UI configuration file (JSON/YAML):

json

Copy

Edit

{

"category": "Federal Tax Form",

"fields": ["year", "SSN", "amount"],

"highlight": { "expired": true },

"groupBy": "tax\_year"

}

**7. UI Code Generator**

Goal: Generate frontend components + backend handlers

Inputs:

Category-specific metadata

User-config rules

Output:

React/Tailwind component

Configurable component props

Optional API calls (CRUD via REST or GraphQL)

Tech:

GPT-4 Function Calling / LangChain + ReAct agent

Vercel-style component library (shadcn/ui or custom)

**8. Microservice Integration**

Config Options:

Frontend: Component registry or plug-in UI loader

Backend: API config, document type routes, webhook triggers

CI/CD: Export config files or integrate with backend deployment tools