Compliance Copilot Phase 1

1. Overview

Compliance Copilot (Revali) is an AI-powered emergency management application that helps emergency managers evaluate, update, and maintain compliance for their emergency plans. The application provides dashboards for plan status monitoring, reference document management, and AI to assist with plan improvements and general questions and answers.

2. Important Definitions

Term	Definition
Document	Used when referring to a Reference Document and a Plan, together.
Reference Document	Knowledge base document that the Gap Analysis AI and Copilot AI uses to help produce gap analysis and metrics.
Reference Type	The type of reference document. Includes standards (NFPA 1600, FEMA CPG 101), plan templates, general knowledge criteria, guidelines, regulations, and best practices.
Plans	Document that is being analyzed for compliance by AI.
Plan Type	Type of plan such as Emergency Operations Plan (EOP), Hazard Mitigation Plan (HMP), Continuity of Operations Plan (COOP), Incident Action Plan (IAP), After Action Report (AAR), or other specialized emergency plans.
Plan Status	Lifecycle status: In review, archived, or active.
Gap Analysis AI	Component that performs analysis of Plans and produces metrics output.

Copilot AI	Interactive chatbot in the bottom right that answers questions based on the knowledge base of reference documents and plans.

3. Implementation

3.1 Database Design Update

- Complete database schema design (details in Appendix A)
- Implementation using Supabase for vector database capabilities
- Schema.sql file structure for table creation

3.2 Reference Documents Page

Functionality:

- Houses industry standards, guidelines, and regulations
- Stores exemplar 100% compliant plan templates for each plan type
- Serves as knowledge base for compliance requirements and recommendations
- Accessible to both the Copilot and Gap Analysis AI components

User Process Flow:

1. Upload Reference document
    - Select Reference Document Type from dropdown
    - Add Description (used as metadata in vector database)
    - Optionally add tags for categorization
2. System processes document
    - Performs intelligent document chunking based on size and content type
    - Supabase vectorizes chunks and stores with appropriate metadata

UI Components:

- Document library with filtering and search capabilities
- Document preview with metadata display
- Upload form with validation
- List view with sortable columns (name, type, date added, size)
- Grid view with document thumbnails for visual browsing

3.3 Plans Page

Functionality:

- Repository for user's actual emergency plans requiring analysis
- Tools for viewing, editing, and analyzing plans
- Displays compliance scores and improvement recommendations
- Tracks version history of plan updates

User Process Flow:

1. User uploads document or creates new plan
2. Intake form collects:
    - Plan Type (dropdown selection)
    - Description (used as metadata in vector database)
    - Target compliance level (standard/reference to measure against)
    - Department/Organization section ownership
3. System shows Gap Analysis loading indicator
4. User is directed to Gap Analysis Results page upon completion

UI Components:

- Plan library with filtering by type, status, and compliance score
- Quick-view compliance indicators (color-coded)
- Action buttons for common tasks
- Compact view showing critical metrics at a glance
- Bulk operations for multiple plans (analyze, archive, export)

3.4 Gap Analysis Results Page

Key Components:

- Annotated Document View with inline suggestions
- Compliance Score Display with breakdown by section
- Color-Coded suggestion highlighting by severity/importance

Compliance Score Elements:

- Overall score (0-100%) prominently displayed
- Section-by-section breakdown with recommendations

Visual Design:

- Color-coded indicators:
    - Red (0-40%): Significant improvements needed
    - Yellow (41-70%): Moderate compliance
    - Green (71-100%): Strong compliance
- Expandable/collapsible sections for detailed analysis

UI Layout:

1. Compliance Score (Top section)
2. Overall Analysis Summary with key findings
3. Annotated Document with:
    - Original document text
    - Color-coded highlights for issues
    - Inline suggestion comments
4. Section-by-Section Breakdown with expandable details
5. Action Buttons:
    - Accept All Changes
    - Review Changes Individually
    - Generate Alternative Suggestions

3.5 Gap Analysis AI

Functionality:

- Specialized analytical engine for compliance evaluation
- Compares user plans against appropriate reference standards
- Identifies missing elements based on plan type requirements
- Generates compliance scores and improvement recommendations

Scoring Algorithm:

- Simple binary assessment: each required element is either present (1) or missing (0)
- Calculate compliance score: (Number of items present / Total required items) × 100
- Color-coded results:
    - Red (0-40%): Significant improvements needed
    - Yellow (41-70%): Moderate compliance
    - Green (71-100%): Strong compliance

Technical Implementation:

- Built on google gemini with fine-tuning (RAG) for emergency management domain
- Retrieval-augmented generation using vector embeddings from reference documents
- Custom scoring algorithm based on weighted compliance factors
- NLP-based content analysis for semantic understanding of plan components

3.6 Copilot AI Integration

Functionality:

- Interactive chatbot interface in bottom-right corner
- Access to knowledge base of reference documents and plans
- Answers user questions about compliance requirements

Technical Implementation:

- Built on google gemini with context-aware retrieval
- Integration with vector database for relevant document retrieval
- Conversation history maintenance for context-aware responses
- UI with expandable/collapsible panel and persistent presence

User Interaction Capabilities:

- Natural language queries about compliance standards
- Document-specific questions and guidance
- Suggested improvement recommendations

4. Appendix A: Database Design

4.1 Vector Embeddings Table

- Unified storage architecture for plans and reference documents referred to as "Documents" used by AI as knowledge base and for RAG
- Chunking Strategy
    - Intelligent chunking based on document size and content:
        - Small documents (<1000 words): Embed entire document
        - Medium documents (1000-5000 words): Moderate chunking with 10-15% overlap
        - Large documents (>5000 words): Aggressive chunking with semantic boundaries
    - Preservation of document structure (headers, sections)
    - Special handling for images, tables, lists, and other formatted content

4.2 Original Document Storage (Blob)

- Separate blob storage for original documents
- Segregated buckets for plans and reference documents
- Support for multiple formats (PDF, Word, TXT, Markdown)
- Version control for document updates
- The system uses a two-part approach to document storage:
    - Blob Storage System:
        - Original document files (PDFs, Word docs, etc.) are stored in an external blob storage system (e.g., AWS S3, Azure Blob Storage, or Supabase Storage)
        - Files are stored in their native format for preservation and retrieval
        - Storage buckets are organized by document type (plans vs. reference documents)
    - Document Files Database Table:

        - Tracks metadata about files stored in the blob storage system
        - Maps files to their corresponding documents in the documents table
        - Stores file paths or URLs needed to retrieve files from blob storage
        - Maintains file integrity information and upload history
    - Workflow:
        - When a document is uploaded, the file is sent to blob storage
        - The storage system returns a file path or URL
        - A document_files record is created with this path linked to the appropriate document ID
        - When retrieval is needed, the application uses the stored path to fetch the file from blob storage

The document_files table does NOT store the actual file content, only the information needed to locate and validate the files in the blob storage system. This separation allows for efficient database operations while maintaining the relationship between structured document data and the original files.

4.3 Metrics Table (Schema)

- Structured table for compliance analytics, detailed below

4.4 Data Structure

Vector Database Structure:

JavaScript

```
-- Documents Table - For both reference documents and plans
CREATE TABLE documents (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    document_type VARCHAR(20) NOT NULL, -- reference or plan
    doc_subtype VARCHAR(50) NOT NULL, -- standard, guideline, regulation, EOP, HMP, COOP, etc.
    title VARCHAR(255) NOT NULL,
    description TEXT,
    status VARCHAR(20), -- in_review, active, archived (for plans)
    user_id UUID REFERENCES users(id),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    tags TEXT[],
    version INTEGER DEFAULT 1,
    department VARCHAR(100),
    source_org VARCHAR(255),
    pub_date DATE,
    authority_level VARCHAR(50), -- For reference docs: guideline, requirement, regulation
    last_review_date DATE,
    next_review_date DATE,
    owner VARCHAR(100),
    compliance_score NUMERIC(5,2), -- For plans only
    metadata JSONB -- For any additional document-specific metadata
);
```

```
-- Vector Embeddings Table - Stores chunked content with vector embeddings
CREATE TABLE vector_embeddings (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    document_id UUID REFERENCES documents(id),
    content TEXT NOT NULL,
    embedding vector(1536), -- For OpenAI embeddings
    metadata JSONB, -- Contains additional context about the chunk:
                    -- {
                    --   "section_title": "Response Procedures",
                    --   "page_number": 12,
                    --   "document_title": "County EOP 2023",
                    --   "chunk_type": "section",
                    --   "importance_level": "high",
                    --   "related_sections": ["Evacuation", "Communications"]
                    -- }
    chunk_index INTEGER,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

Explanation: Single `documents` table that stores metadata for both reference documents and plans, distinguished by the `document_type` field. This approach better aligns with the unified storage concept mentioned in the original document.

The `vector_embeddings` table then stores the actual content chunks and their vector embeddings, with a direct foreign key reference to the parent document.

Blob Storage Structure:

JavaScript

```
-- Document Blob Files Table
CREATE TABLE document_files (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    document_id UUID NOT NULL REFERENCES documents(id),
    original_filename VARCHAR(255) NOT NULL,
    mime_type VARCHAR(100) NOT NULL,
    file_path TEXT NOT NULL,
    file_size BIGINT NOT NULL,
    uploaded_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    content_hash VARCHAR(64), -- For integrity verification
    is_encrypted BOOLEAN DEFAULT FALSE
);
```

Metrics Data Structure:

JavaScript

```
-- Gap Analysis Results Table
CREATE TABLE document_files (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    document_id UUID NOT NULL REFERENCES documents(id),
    original_filename VARCHAR(255) NOT NULL,
    mime_type VARCHAR(100) NOT NULL,
    file_path TEXT NOT NULL,
    file_size BIGINT NOT NULL,
    uploaded_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    content_hash VARCHAR(64), -- For integrity verification
    is_encrypted BOOLEAN DEFAULT FALSE
);
```

5. Data Flow

Document Upload Process:

User → Upload Document → API → Document Processor → Store Original File in Blob Storage → Extract Text → Chunk Text → Generate Embeddings → Store in Vector Database → Store Metadata in Supabase Database

Analysis Process:

User → Request Analysis → API → Gap Analysis AI → Retrieve Reference Standards from Vector Database → Retrieve Plan Content from Vector Database → Compare Content Against Requirements → Calculate Compliance Score → Store Results in Supabase → Return Results to User Interface

Copilot Interaction:

User → Ask Question → Copilot AI → Search Knowledge Base in Vector Database → Generate Response → Return Answer to User

Data Relationships:

- **Supabase Database: Stores document metadata and analysis results**
- **Vector Database: Stores text chunks with embeddings for semantic search**
- **Blob Storage: Stores original document files**
- **API Layer: Coordinates all operations between components**

6. Current Project Structure

Unset

```
.
├── client
│   ├── public
│   │   ├── favicon.ico
│   │   ├── index.html
│   │   ├── logo192.png
│   │   ├── logo512.png
│   │   ├── manifest.json
│   │   └── robots.txt
│   ├── src
│   │   ├── components
│   │   │   ├── copilot
│   │   │   │   └── Copilot.js
│   │   │   ├── dashboard
│   │   │   │   ├── ComplianceOverview.js
│   │   │   │   ├── MetricsPanel.js
│   │   │   │   └── StatusSummary.js
│   │   │   ├── documents
│   │   │   │   ├── DocumentCard.js
│   │   │   │   ├── DocumentList.js
│   │   │   │   ├── UploadForm.js
│   │   │   │   └── Viewer.js
│   │   │   ├── gapanalysis
```

```
|   |   |   |   ├── AnnotatedDocument.js
|   |   |   |   ├── ComplianceScore.js
|   |   |   |   ├── GapSummary.js
|   |   |   |   └── RecommendationPanel.js
|   |   |   └── layout
|   |   |       ├── Header.js
|   |   |       ├── PrivateRoute.js
|   |   |       └── Sidebar.js
|   |   ├── context
|   |   |   ├── AuthContext.js
|   |   |   └── DocumentContext.js
|   |   ├── hooks
|   |   |   ├── useDocuments.js
|   |   |   └── useGapAnalysis.js
|   |   ├── pages
|   |   |   ├── Dashboard.js
|   |   |   ├── Documents.js
|   |   |   ├── GapAnalysisResults.js
|   |   |   ├── Login.js
|   |   |   ├── Plans.js
|   |   |   ├── ReferenceDocuments.js
|   |   |   └── Register.js
|   |   ├── services
|   |   |   ├── api.js
```

```
|   |   |   ├── auth.js
|   |   |   └── storage.js
|   |   ├── utils
|   |   |   ├── formatting.js
|   |   |   └── validation.js
|   |   ├── App.css
|   |   ├── App.js
|   |   ├── App.test.js
|   |   ├── index.css
|   |   ├── index.js
|   |   ├── logo.svg
|   |   ├── reportWebVitals.js
|   |   └── setupTests.js
|   ├── README.md
|   ├── package-lock.json
|   ├── package.json
|   ├── postcss.config.js
|   └── tailwind.config.js
├── server
|   ├── config
|   |   ├── SUPABASE_SETUP.md
|   |   ├── db.js
|   |   ├── schema.sql
|   |   └── supabase.js
```

```
|   ├── controllers
|   |   ├── aiController.js
|   |   ├── authController.js
|   |   ├── plansController.js
|   |   └── referenceController.js
|   ├── middleware
|   |   ├── auth.js
|   |   └── errorHandler.js
|   ├── models
|   |   ├── GapAnalysis.js
|   |   ├── Plan.js
|   |   ├── ReferenceDocument.js
|   |   └── User.js
|   ├── routes
|   |   ├── aiRoutes.js
|   |   ├── authRoutes.js
|   |   ├── plansRoutes.js
|   |   └── referenceRoutes.js
|   ├── services
|   |   ├── embedding.js
|   |   ├── gapAnalysis.js
|   |   └── storage.js
|   ├── utils
|   |   ├── chunking.js
```

```
|   |   └── scoring.js
|   ├── package-lock.json
|   ├── package.json
|   └── server.js
├── package-lock.json
└── package.json
```