Adding new documents to the Knowledge Bank

Introduction

This document will contain a guide on how to add new documents to the Knowledge Bank of VINO AI for students. This process may be intricate, so this document is made to instruct the developers through the process. First, we will walk you through the general guidelines for creating files and then show how to upload them to the Knowledge Bank (ChromaDB for vector storage, Supabase for SQL persistence).

Note: User documents (acquired through the application) are uploaded to both databases (ChromaDB and Supabase) automatically. If the documents are in the PDF format, they follow a simplified chunking process.

Requirements

- **Preferred file extention**: `.md` (why?)
- **No need for Table of Contents**: ToC gets created automatically when a document is processed by upload_supa_chroma.py.
- **Section Limit**: Sections under 300 tokens will be processed as a whole, while larger paragraphs will be broken down further.
- **Existing Folders**: Make sure `kb/` and `kb_new/` exist.
- **No Duplicates**: No duplicate names are allowed. The program will throw an error.

Document Structure

The document should follow this basic structure:

Required Elements:

- **Title**: Use a clear, descriptive heading separated by `===`.
- **Summary Section**: Brief overview of the topic (recommended first section).
- **Main Content**: Organized with clear H1, H2 and H3 headings.
- **Practical Examples**: Include "Application Examples" or similar sections when relevant.
- **Step-by-Step Guides**: For procedural content, provide numbered steps with actionable items.

- **Considerations/Limitations**: Include potential issues or important notes.
- **Resource Links**: External references and further reading.

Content Guidelines

- **Use descriptive section headings**: Avoid generic titles like "Overview" - be specific about what each section covers.
- **Include practical examples**: Show real-world applications relevant to students.
- **Add actionable items**: When describing processes, include "Actionable for students:" subsections.
- **Structure for scannability**: Use bullet points, numbered lists, and clear hierarchies.
- **Define key terms**: Bold important concepts on first use.
- **Cross-reference related documents**: Link to other KB documents when relevant.

Quality Checklist

Before processing any document, verify:

- ☐ Clear, descriptive title with proper H1 formatting
- ☐ TOC markers abscent
- ☐ Summary section included
- ☐ All technical terms defined on first use
- ☐ "Actionable for students:" sections included
- ☐ Real-world examples relevant to student projects
- ☐ Considerations/limitations section present
- ☐ Resource links functional and relevant
- ☐ Consistent formatting throughout

Why Markdown?

- **Lightweight and readable**: Easy to write and read in plain text format.
- **Universal compatibility**: Supported across platforms and can be converted to HTML, PDF, and other formats.
- **Version control friendly**: Text-based format works well with Git for tracking changes and collaboration.
- **Structured data extraction**: Headers, lists, and code blocks are easily parsed programmatically.
- **AI-friendly format**: Large language models process Markdown naturally and effectively.
- **Future-proof**: Simple text format ensures long-term accessibility regardless of software changes.
- **Rich formatting support**: Supports tables, links, images, code blocks, and mathematical expressions.
- **Fast processing**: Minimal overhead compared to complex document formats like DOCX or PDF.
- **Search optimization**: Clean structure improves vector embedding quality for semantic search.

Document Processing Steps

Step-by-Step Guide for Adding Documents

1. **Create the Document**:
    - Save file with `.md` extension in the `kb_new/` directory
    - Use descriptive filename (e.g., `ProjectManagement.md`, not `PM.md`)

2. **Structure Your Content**:
    - Start with clear H1 title using `===` underline format
    - Include all required sections from the template
    - Actionable for developers: Use the SCRUM.md file as a structural reference
3. **Quality Review**:
    - Verify all sections have descriptive headings
    - Check that "Actionable for students:" items are included where relevant
    - Ensure examples are student-project focused
    - Actionable for developers: Have another team member review before processing
4. **Processing & Upload**:
    - Run `uv run upload_supa_chroma.py` to process the document
    - Verify successful upload to knowledge bank in the terminal
    - Check if files were uploaded correctly to both Chroma and Supa (Check Upload)
    - Actionable for developers: Document any processing errors for troubleshooting

File Organization

- **Location**: All KB documents go from `kb_new/` to the `kb/` directory after processing
- **Naming**: Use PascalCase for filenames (e.g., `SoftwareTesting.md`)

Check Upload

To verify that documents were correctly uploaded to both ChromaDB and Supabase, you can follow these steps:

- **ChromaDB**:

1. Open database.py
2. Uncomment `list_documents_in_collection("frameworks")`
3. Run this command: `uv run database.py`
4. Check terminal output, the added (chunked) documents should appear at the end.

- **Supabase**:

1. Go to https://supabase.com/dashboard/org/vxhqckyitkquxdtoulcw (you must be granted access from the developers)
2. Navigate to `Table Editor` section
3. File metadata will be uploaded to `filemetadata` table
4. Full text will be uploaded to `largeobject` table
5. Navigate to `Storage` section
6. Documents uploaded manually can be found at `knowledge-base`