

REQUEST SUMMARY

This project was undertaken at the request of an individual with an anti-government mindset and a healthy dose of justifiable paranoia. Their perspective does not reflect my own beliefs, anxieties, or current threat assessment level.

The core request was for a lightweight, local database to store and search survival-related documents. Modern, web-based solutions were explicitly off the table.

The intended use cases included, but were not limited to:

- Cataloging homesteading knowledge (gardening, farming, animal husbandry, carpentry, etc.)
- Storing basic survival skills (hunting, trapping, water purification, primitive shelter, etc.)
- Preparing for societal collapse, martial law, or other worst-case scenarios
- Navigating the end of civilization as we know it
- Exploring one's inner Kaczynski—should society truly go off the rails

While the initial spec was simply to store full plain text documents in a searchable heap, I expanded the scope immediately to make the tool usable in a real-world context (even one involving improvised snares and canned squirrel).

Key functionality added:

- Classification by author, category, and subcategory
- Keyword associations for improved search relevance
- File storage system for non-text documents, images, audio, video, or anything not suited to storage as plain text
- Automated database backups, filestore backups, and database optimization

IMPLEMENTATION STRATEGY – AKA: HOW I BUILT THE APOCALYPSE-PROOF FILING CABINET

To keep the solution lightweight, portable, and fully local (because clouds are just someone else's computer), I developed the application in Python.

Given that flashy visuals were not a requirement—and may even be considered a liability in off-grid scenarios—I used Tkinter, Python's built-in GUI library. It delivers just enough interface to get the job done, without drawing too much attention or memory.

For the data layer, I selected SQLite for its simplicity, portability, and zero-config usage. It offered the ideal balance between structure and low overhead.

WHY THIS PROJECT

This application demonstrates my ability to transform loosely defined and unconventional requirements into a structured, functional, and user-friendly tool. It reflects my strengths in lightweight app development, practical UI design, and backend organization—plus a sense of humor when dealing with end-of-world scenarios.

TECH STACK

- Language: Python 3.12
- GUI: Tkinter
- Database: SQLite (FTS5 enabled)
- Deployment: Standalone, Windows-compatible
- Dependencies: None (uses standard library only)

BACKEND DESIGN

A schema to store:

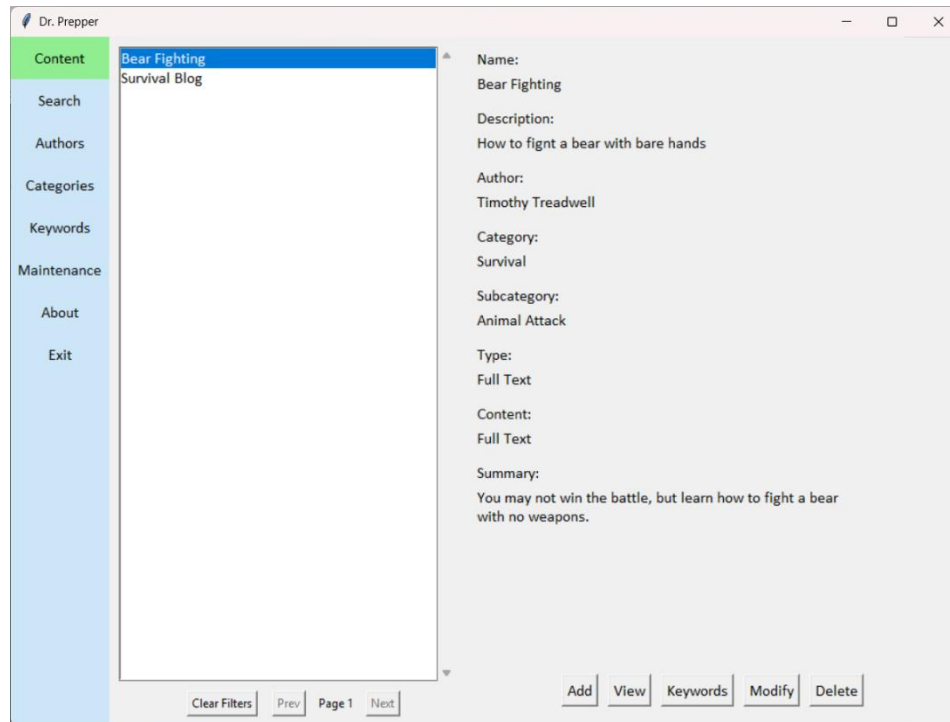
- Content
- Authors
- Categories and subcategories
- Keyword associations
- A full-text index on the primary content body to support the original request
- An additional full-text index on the 500-character summary field to improve discoverability for non-plain-text entries

APPLICATION FEATURES

- Content Manager – Create, edit, and delete content entries, with support for:
 - Title, description, author, category, and subcategory
 - Summary (500-character max) for reference and indexed search
 - Content Types:
 - Full text: Original request—plain, searchable text
 - File-based: Store and retrieve arbitrary file types (PDFs, media, etc.) – Opens with the default associated application
 - Web address: Store a URL and summary – Opens in the default browser (admittedly not very usable post-apocalypse)
- Author Manager – Manage authors (in case it wasn't self-explanatory)
- Category Manager – Manage categories and their subcategories
- Keyword Manager – Manage keyword tags for enhanced search granularity
- Search Engine – Flexible querying capabilities, including:
 - Keyword
 - Author
 - Category / Subcategory
 - Summary (full-text indexed)
 - Full content (for entries of type "Full text" only – no deep search in external files)
- Maintenance Manager – Backup database, backup filestore, optimize database (vacuum and rebuild indexes)

INTERFACE

CONTENT MANAGER

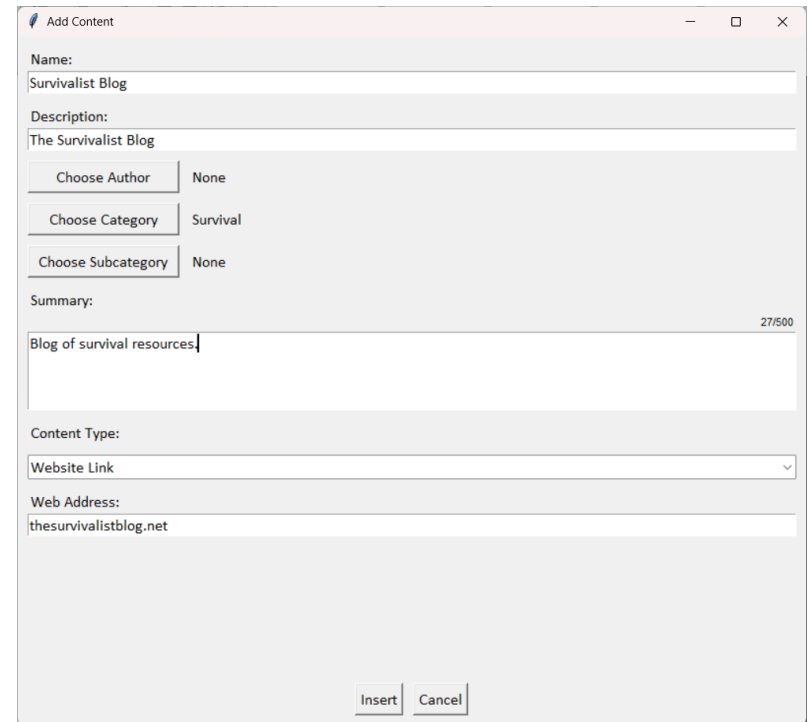


The Content Manager application window, titled "Dr. Prepper", features a sidebar with navigation links: Content (highlighted), Search, Authors, Categories, Keywords, Maintenance, About, and Exit. The main area displays a list of content items: "Bear Fighting" (selected) and "Survival Blog". To the right, a detailed view for the selected item shows the following information:

- Name:** Bear Fighting
- Description:** How to fight a bear with bare hands
- Author:** Timothy Treadwell
- Category:** Survival
- Subcategory:** Animal Attack
- Type:** Full Text
- Content:** Full Text
- Summary:** You may not win the battle, but learn how to fight a bear with no weapons.

At the bottom of the window, there are buttons for "Add", "View", "Keywords", "Modify", and "Delete". A pagination bar at the very bottom includes "Clear Filters", "Prev", "Page 1", and "Next".

ADD NEW CONTENT



The Add New Content application window, titled "Add Content", contains the following form fields and controls:

- Name:** Survivalist Blog
- Description:** The Survivalist Blog
- Choose Author:** None
- Choose Category:** Survival
- Choose Subcategory:** None
- Summary:** Blog of survival resources. (Character count: 27/500)
- Content Type:** Website Link (dropdown menu)
- Web Address:** thesurvivalistblog.net

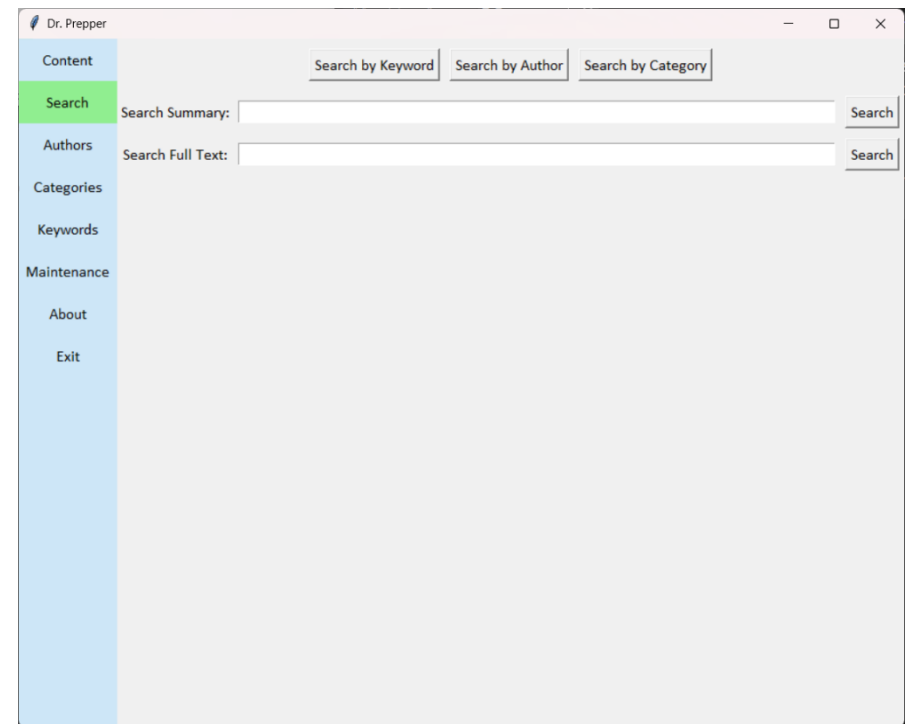
At the bottom right, there are "Insert" and "Cancel" buttons.

INTERFACE – CONTINUED

VIEW FULL TEXT CONTENT

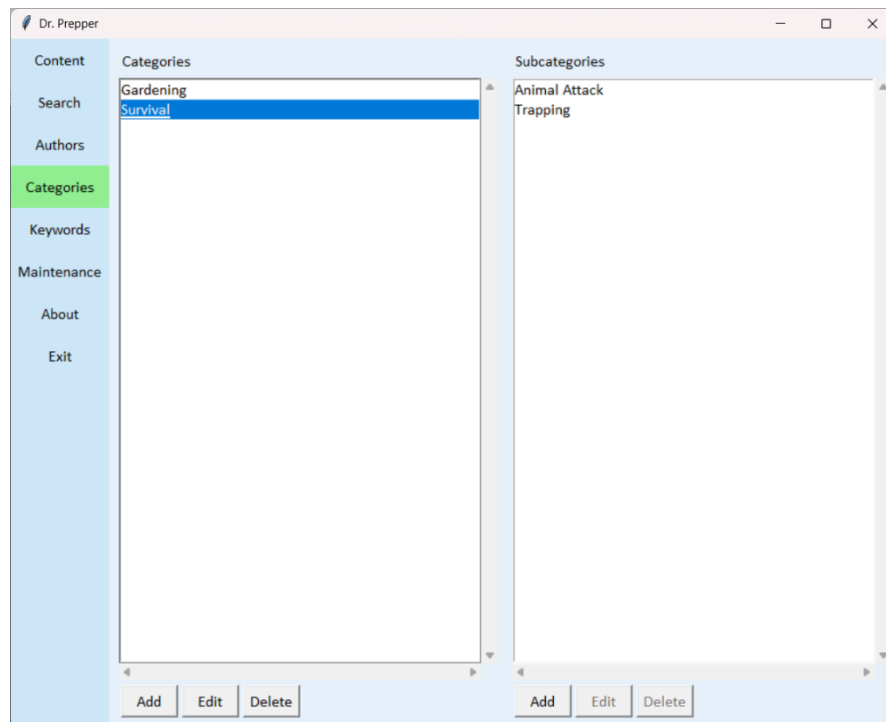


SEARCH

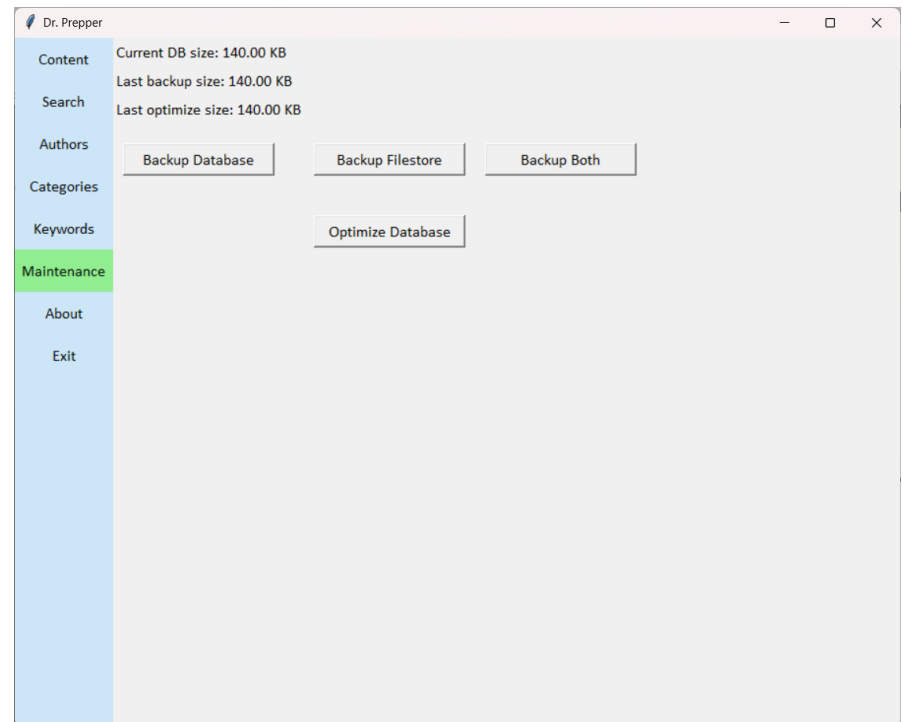


INTERFACE – CONTINUED

CATEGORY MANAGER



MAINTENANCE MANAGER



INTERFACE – CONTINUED

ABOUT

