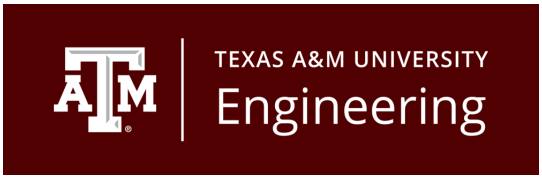
Project Report

CortexFS

by

Shreyas Shivakumar Kasetty (534002320)



Contents

1

Introduction

Introducing CortexFS: Your Smart File Organizer

Is your computer full of messy, cluttered folders? The organization of files can sometimes be frustrating and time-consuming. That's where CortexFS comes in! CortexFS is an intelligent file organization tool powered by Large Language Models to help you get your files organized without breaking a sweat. Be it a batch of old, disorganized files or newly downloaded ones, CortexFS ensures your system stays neat and accessible.

What is CortexFS?

CortexFS is a smart file organization system designed to automate and simplify file management. It analyzes your files and provides personalized suggestions for organizing them into appropriate folders. CortexFS learns from your habits and preferences, ensuring its recommendations improve over time. With CortexFS, you'll save hours of manual sorting and have more time to focus on what matters.

Key Features

Batch File Organizer

CortexFS scans your existing directories and organizes files into a logical folder structure. It suggests an ideal placement for each file, making it easy to declutter your system in minutes.

File Placement Suggestions

CortexFS watches for new downloads and analyzes them in real time, recommending the best folder to store each file. You can accept or modify suggestions, and the system learns from your feedback.

Intuitive Desktop App

CortexFS features an intuitive interface with three key screens to streamline your file organization process:

- **Settings Page:** Easily toggle the watch mode on or off to suit your preferences.
- **Suggestions Page:** Review personalized recommendations for newly downloaded files and decide where they should be stored
- **Batch Organize Page:** Quickly organize large groups of files with just a few clicks, making decluttering effortless.

Why Use CortexFS?

- **Save Time:** Automates tedious file organization tasks.
- **Simplified Workflow:** Keeps your files neat without interrupting your day.

2

Methodology

2.1. General Overview

CortexFS simplifies file organization using a intuitive desktop application designed for real-world needs. Here is how users interact with it:

2.1.1. Batch Organize Workflow

If a users have disorganized directories with unrecognizable files, they can navigate to the "Batch Organize" option in the app's sidebar and follow the below steps:

- The user provides the path to the directory they want to organize using an input field
- The "Organize" button initiates a REST API call to the FastAPI backend; the system uses local LLMs to summarize each file for privacy, without exposing personal data
- Summaries are then sent to another llm agent using grok for fast inference, which collaborates with an organization agent to intelligently sort the files into appropriate categories or folders.

2.1.2. Role of LLM agents

LLM agents handle both summarization and organization of files with no need for complicated knowledge graphs. While scaling may pose some issues, this approach is good enough for practical use.

2.2. File Analysis and Suggestions

CortexFS is designed to support multimodal file organization with a focus on widely used formats to ensure versatility. The system handles coding files such as Python and Java, various document types including PDFs, Microsoft Office files, CSVs, JSON, XML, text files, README, and log files, as well as common image formats. A special feature of CortexFS is the "Watchdog" mode, which will watch a given directory in real time for new files and immediately organize them. In other words, when it detects a new file, the watchdog will trigger a create event of the file, summarize the content, and analyze the folder structure of the target directory to suggest the best fit. If no folder already exists to accommodate it, it will create a new path in the target directory and sends a suggestion notification to the application. It can be reviewed and accepted or rejected by the user anytime.

2.3. Architecture

CortexFS is designed to have a modular and efficient architecture in the backend, seamlessly integrating intelligent file organization capabilities. It is powered by FastAPI, a modern web framework that enables serving RESTful APIs for various functionalities. RabbitMQ acts as the backbone of client-server communication, providing reliable message-based interactions between frontend application and the backend. It uses Langchain library to effectively build LLM-based agents, such as the summarizer and organizer, which form the core of the system's intelligence for advanced file analysis and organization. The frontend Application is developed using **ElectronJS** which is helpful in building cross-platform native applications.

The system has several key modules described below:

- **Summarizer:** This module utilizes open source LLMs like llama3.2 to summarize the file contents and generate quick summaries. These summaries capture the essence of the file for the understanding of the system and help in determining its ideal placement in the file system.
- **Organizer:** This module is responsible for classification and organization into folders, based on the output of the summarizer. This module makes decisions about where to file documents based on its understanding of existing directory structures and user preferences. This module or agent utilizes the grok api to make fast decisions using llms.
- **WatchDog:** This module would continuously monitor any changes happening in the directories set by the user and initiate the summarizer and organizer modules for processing every time a new file is created in a directory being monitored.
- **Notification:** This module is responsible for sending suggestions to the desktop application as notification using RabbitMQ message queue.

2.3.1. Architecture Diagram:

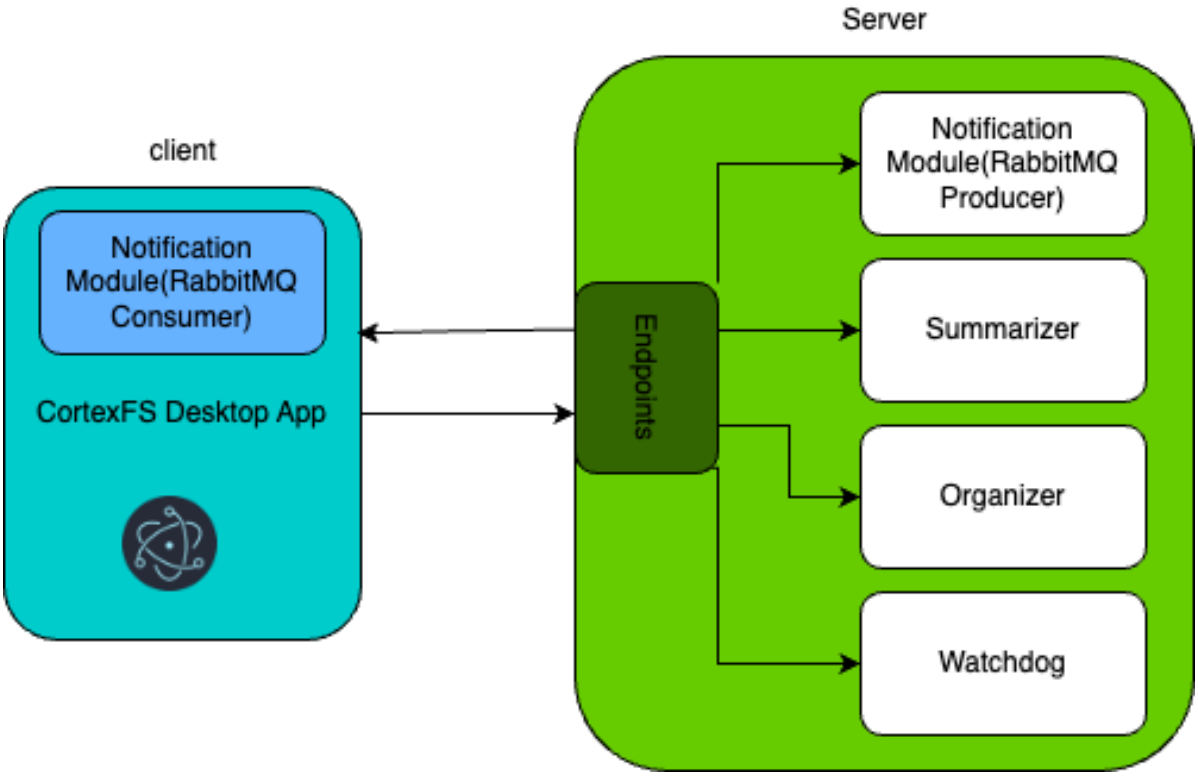


Figure 2.1: CortexFS Architecture Diagram

3

Testing & Evalution

3.1. Overview of Tests

CortexFS was tested and evaluated on all various aspects of the system to ensure the desired functionality of smooth and efficient file organization. My tests covered various facets of the system, including unit testing, end-to-end testing of correctness, reliability, performance at individual modules, and system levels. I have automated the unit testing of modules. However, other aspects of the system testing and evaluation were conducted manually.

3.1.1. Type of tests:

Following are the type of tests conducted on CortexFS to ensure its correctness and robustness:

1. **Unit Tests:**
   - Focused on individual components like the `FileEventProducer`, `WatchdogHandler`, and FastAPI endpoints.
   - Verified the correctness of logic, including directory monitoring, file event handling, and file organization tasks.
2. **Integration Tests:**
   - Validated the interaction between different modules, such as the summarizer and organizer agents communicating with the FastAPI backend through RabbitMQ.
   - Ensured cohesive functioning of modules to produce the desired results.
3. **End-to-End Tests:**
   - Simulated real-world scenarios involving user interactions with the application.
   - Covered workflows such as starting directory monitoring, generating file organization suggestions, and committing file placement changes.
4. **Error Handling and Robustness Tests:**
   - Tested the system for various edge cases, such as invalid directory paths or missing source files.
   - Ensured the system handled errors gracefully and provided meaningful feedback to the user.

Table 3.1: Unit Tests

Short Test Description	Result
Verify that the file monitoring starts correctly for a valid directory	Passed
Assert that starting monitoring for an invalid directory raises FileNotFoundError	Passed
Ensure file monitoring stops correctly with proper logs	Passed
Test the generation of file organization suggestions using LLM summarization	Passed
Handle creation events in the watchdog handler and delegate to producer	Passed
Check health of the FastAPI service via health-check endpoint	Passed
Validate batch organization endpoint with valid paths	Passed
Ensure batch organization endpoint handles invalid paths correctly	Passed
Verify the commit endpoint moves a file successfully to the target directory	Passed
Handle commit-suggestion endpoint errors when the source path does not exist	Passed
Verify start-producer endpoint starts file monitoring for valid directories	Passed
Ensure start-producer endpoint returns an error for invalid directories	Passed
Stop producer successfully through the stop-producer endpoint	Passed

Table 3.2: End-to-End Tests Covering All Features

Test Description	Result
Test the **Batch Organize** feature by selecting a directory with mixed files and verifying that the system organizes files into appropriate folders based on content.	Passed
Verify that the **File Placement Suggestions** feature correctly monitors the watch directory and suggests appropriate target locations for newly downloaded files.	Passed
Check that toggling the **Watch Mode** in the **Settings** page correctly starts and stops the monitoring of the watch directory.	Passed
Ensure that users can **accept**, **reject**, or **modify** file placement suggestions and that the system moves or leaves files accordingly.	Passed
Test that the system handles files of **supported types** (e.g., documents, images, code files) and organizes them appropriately.	Passed
Verify that the system gracefully handles **unsupported file types** by notifying the user or skipping them without errors.	Passed
Test the performance of the **Batch Organize** feature on a directory with a large number of files to assess scalability.	Passed
Ensure that the system maintains **privacy** by processing files locally without sending personal data to external services.	Passed
Check that the **Notifications** module correctly sends a single notification instead of redundant suggestion notifications	Passed
Test the end-to-end workflow of changing **Settings**, organizing files, and verifying that changes persist across sessions.	Passed

3.2. Evaluation

The evaluation of CortexFS focuses on performance, usability, and reliability. Let's explore the evaluation results and methods one by one.

3.2.1. Performance

The performance was measured in terms of processing time for two key features: Watch Mode and Batch Organization. Tests were conducted on these features by varying the workload to see how well they scale. It was measured by timing how long it takes for the system to process files under different scenarios.

- **Watch Mode:** The time taken to detect, summarize, and suggest file placement for newly created files was measured.

- **Batch Organization:** The system's ability to process and organize a directory with a specific number of files was evaluated, focusing on summarization and organization time.

3.2.1.1. Watch Mode Performance

Table 3.3: Performance Metrics for Watch Mode

File Size Range	Summarization Time (s)	Organization Time (s)	Total Time (s)
1–3 MB	4.12	1.08	6.20
1–5 MB	7.47	0.82	9.29
5–10 MB	18.48	1.09	20.58

3.2.1.2. Batch Organization Performance

Table 3.4: Performance Metrics for Batch Organization

Number of Files	Average File Size	Summarization Time (s)	Organization Time (s)	Total Time (s)
5	3.6 MB	27.2	1.5	32.22
10	1.9 MB	42.9	2	47.71
15	1.33 MB	82.87	3.05	90.99
20	1.4 MB	103.79	4.23	113.36

3.2.1.3. Observations

- **Watch Mode**: As the file size increases the processing time increases linearly as the amount data it has to organize increases as well.
- **Batch Organization**: Scales linearly with the number of files, but the total time increases significantly as the file count grows. This is expected due to the need to process all files in one go.
- Summarization is the primary bottleneck, given the use of local LLM.

These tables provide a baseline for evaluating the current capabilities of CortexFS and identifying opportunities for future optimizations.

4

Related Work

Very little work has been done in the area of using large language models for file organization. This is a rather new field, and CortexFS is an attempt to bring a practical solution to this problem. One existing project that explores a similar idea is LlamaFS. While LlamaFS introduced the concept of using LLMs to organize files, its solution is incomplete and lacks several important features needed for real-world use.

A significant drawback of LlamaFS is that it does not possess the Watch Mode functionality in CortexFS. It does have a watch functionality but not for monitoring and moving downloaded files. The latter is one of the essential features that make CortexFS so distinctive. It allows the system to track a given directory continuously and notify users when a new file has been added. The system recommends an appropriate folder for the file, in which the user can directly accept or reject the suggestion. This makes file organization dynamic and interactive; this is not supported by LlamaFS.

Another difference is how CortexFS handles privacy. It generates file summaries locally using open-source LLMs in CortexFS, which means no personal data is sent to external services. LlamaFS doesn't make it clear how it handles privacy, which can make users wary of trusting it. CortexFS is also designed to be more practical and modular, with the use of technologies such as FastAPI, RabbitMQ, and ElectronJS. These make it scalable, reliable, and easy to use across different platforms. LlamaFS does not have a well-defined architecture like this, which limits its usability.

While LlamaFS laid a foundation for using LLMs in file organization, CortexFS picks the idea and builds substantial improvement upon that. Focused on solving real-world problems of users with features such as Watch Mode, Privacy First design, and many usability improvements, it makes them practical for the users.

5

Future Work

While CortexFS does a great job of managing files with the features so far implemented, there's still plenty of room for improvement. Some of the features never made it into this version from the original plan but are worth exploring in the future. Here's what I think could be pursued next:

One such feature could be **semantic search**. Currently, file searching is based on names or folders, but it would be nice if users could search based on the content of the files. For example, you might type something like "AI project report," and the system would find the relevant files even if the name isn't a perfect match. This would make it much easier when you are in a hurry or forget where you saved any particular files.

Other functionality could include the **management of temporary files**. Many of us download files that are required for a short time only, such as receipts or drafts of various assignments. CortexFS may have an option to tag such files as temporary and then automatically delete them after some set period. Of course, there would be options to extend the expiry or to retain the files if necessary. In this way, everything will remain clean without too much effort.

Another area for improvement is in personalization. CortexFS could learn from how the user interacts with it, where they tend to save a certain type of file, for instance, and over time start to make smarter suggestions. For instance, if you always put your PDFs in a "Work" folder, the system would suggest that folder first without asking.

There is huge scope of improvement in the scale and performance of CortexFS. Right now, the system makes use of local LLMs for the summarizing of files-a privacy-friendly aspect that comes with the tradeoff of not being that fast, a problem especially in regard to the batch organization feature. This may irritate users when they have to process a lot of files, which may lead to dissatisfaction. Making the summarization and organization faster would go a long way in improving the overall usability of the tool.

Another thing I noticed is that the batch organization tool only focuses on cleaning up messy directories. It takes files in a disorganized folder and creates a proper structure within the same location. While this is helpful, users might also want an option to move these organized files to another location in the file system. For instance, a user who has just sorted the files in the "Downloads" folder may want to move them to some permanent location, such as "Work Documents" or "Photos Archive." Adding this functionality in the future would make the tool more flexible and practical.

Adding the above features can enhance the overall usability and functionality of CortexFS and can be a very useful productivity tool that everyone can use.

6

Conclusion

Working on CortexFS has been such a rewarding and challenging experience. From the very ideation of making file organization a tad bit simpler and smarter, its fulfilling to finally see an idea take shape into something more tangible. It brings in quite a few practical solutions, like Watch Mode or batch organization, to some problems people have with cluttered file systems every day. These can be minor features, yet they can save time and avert frustration, thereby making file management not as much of a chore. That said, there's still a lot to improve. From speeding up local LLM-based summarization to adding flexibility in file placement, there's plenty of room to make CortexFS even better. This project has taught me the importance of balancing usability with performance and designing tools that adapt to real-world needs. It's been a great learning experience, and I'm excited to see how CortexFS evolves in the future to help even more users.

7

Project Resources

The following table provides a list of resources related to the CortexFS project, including the source code, demonstration, and documentation.

Table 7.1: Project Resources

Resource	**Link**
GitHub Project Repo	https://github.com/shreyasskasetty/CortexFS
GitHub Repository (Backend)	https://github.com/shreyasskasetty/CortexFS/tree/main/backend
GitHub Repository (Frontend)	https://github.com/shreyasskasetty/CortexFS/tree/main/desktop-app
Demo Video	Link to the Video