



Family Document Manager System

Project spec Ver. 2

CHI 584 - Python Application Development

Fall 2024

Abduljaleel Hosawi

Abstract

This file proposes the main ideas of the Family Document Manager System and defines its initial features and services in addition to a user story that gives an impression of the interaction in the program.

Hosawi, Abduljaleel M

Spec Ver 2.0

Project Name: Family Document Manager (FDMS Ver 2)

Table (1) summary of changes

Current Modifications in Ver 2	Previous version
<ul style="list-style-type: none">- Version 2 of the FDMS ver 2 is 90% developed, and if all goes well, the full release could be finished by the end of the last milestone.- Task Vignettes has been updated to run with version 2 of the program.- All initial windows have been improved to match the new version.- Some features will be automated, such as creating a document ID and creating a PDF without a related folder.- Document status has been merged with the description.- An appendix has been added to better identify the contents without deleting important parts at the end of this document.	1.2

Project Description:

The **Family Document Manager** is a comprehensive application designed to store and organize family documents efficiently. This application addresses the common problem of locating and managing various family documents, which are often scattered across multiple locations. By centralizing document storage, the Family Document Manager ensures that important documents are easily accessible when needed, such as during university applications, medical emergencies, or financial planning.

Document Categories:

1. **Education:** Public school documents, university education, and higher education.
2. **Health:** Hospital and insurance documents.
3. **Financial:** Financial documents.
4. **Government and Private Services:** Service request documents.
5. **Self-Development:** Self-development documents.
6. **Family Photos:** Family photos.

Data to be Stored:

- Document No (Auto)
- Document title
- Document issue date (now is auto)
- Document pdf (now is auto)
- Document description (this is merged with classification, status, and notes).

Program Features:

- New document data will be entered manually directly from the command UGI window.
- Search and retrieve files based on keywords.
- Update documents.
- Delete documents.

Milestone

I am proud of myself, Dr. Chris. Completing my first-ever full application (FDMS version 1) was a great achievement for me. It has motivated me to start doing more in the next version. Table (2) contains a reminder of my millstone for the next version:

Table (2) Project Milestones for FDMS Version 2

<p>First, GUI design and prototype: I will review and update the creation of wireframes and prototypes for the new GUI. I have already provided prototypes, but as the project progressed, many changes occurred that required incorporation into the final version of the project. This includes defining the design, user flow, and key functionality. Conducting user testing to gather feedback and improve the design.</p>
<p>Second, GUI integration with the backend: After developing the GUI, it will be linked to existing backend functionality. I will work on ensuring smooth interaction between the GUI and the DocumentRepository class, including adding, updating, deleting, and searching documents. Conducting comprehensive testing to ensure stability and usability.</p>

First, the GUI design and prototype:

So far, I have reviewed and redesigned the application windows and I'm working to convert them into tabs to make it easier to navigate between their sections. The update includes incorporating the changes described above, which are changes that resulted from feedback and comments during the correction of my previous reports or for the purpose of improving the code or to make using the program easier, as required by HCI applications. I also expect that when finished, the new GUI models will be clearer than the previous ones. I expect the current models to continue to be modified and improved accordingly before reaching the final models as the project progresses. The updates included many changes that required incorporation or deletion in the previous version of the project. This includes features such as document browsing preview and proofreading during data entry. The focus was mainly on the main functions.

For the integration of the GUI with the backend, after I finished developing the new GUI as in Task Vignettes for FDMS ver. 2, I am still working on programming. After finishing the DocumentRepository class, all the main tasks, including creating and adding a new document, updating an existing document, deleting an existing document, and searching for a document, are expected to be ready. I also plan to conduct comprehensive testing to ensure stability and ease of use.

Updated Task Vignettes for FDMS ver. 2

1. Manual Data Entry

Vignette: My wife Sarah, opens the program and navigates to the data entry section. She manually inputs the details of a new document, including its title, description, and classification. Once she completes the entry, she clicks 'Create' to store the information in the database.

Technical Details:

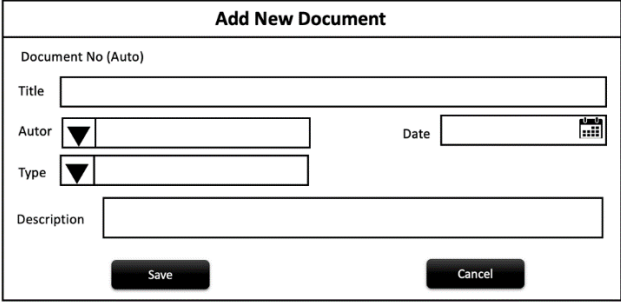
Input Fields: An auto document No, Title, Document Copy, Description.

Save Mechanism: Store data in the excel file.



Add New Document	
Title	<input type="text"/>
Description	<input type="text"/>
Classification	<input type="text"/>
<div><input type="button" value="Create"/> <input type="button" value="Cancel"/></div>	

Fig. (1) The new Mockup for create new document.



The mockup shows a window titled "Add New Document". Inside, there is a label "Document No (Auto)" above a text field. Below this are three rows: "Title" with a text field, "Autor" with a dropdown arrow and a text field, and "Type" with a dropdown arrow and a text field. To the right of the "Autor" field is a "Date" label and a date picker field showing "11/11/2023". Below these is a "Description" label and a large text area. At the bottom are "Save" and "Cancel" buttons.

Fig. (2) the old mockup

The following field has been removed from the old form based on comment: category, document type, date is automatically taken from the operating system, and the check feather has been removed.


2. Search and Retrieve File Data

Vignette: My daughter Emily needs to find a document based on its keywords. She uses the search bar to enter keywords like the document title. The program quickly retrieves and displays a list of matching documents. Emily clicks on a document to view its details.

Technical Details:

Search Fields: Keywords.

Result Display: Show a list of matching documents with brief details.



The mockup shows a window titled "Search for Document". It contains a label "Enter Keyword" next to a text field. Below the text field is a blue "Search" button. At the bottom, there is a "Result" label next to a large empty rectangular box for displaying search results.

Fig. (3) The new Mockup for searching for documents.

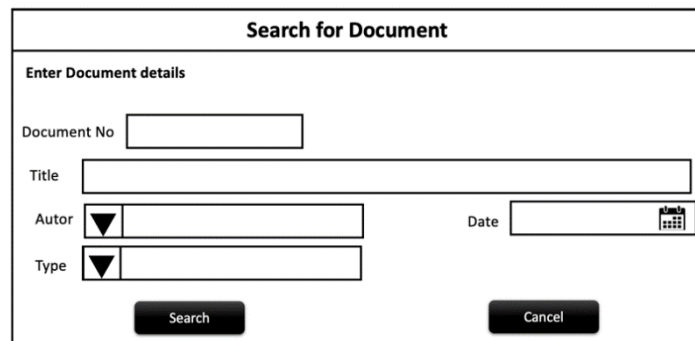


Fig. (4) The old Mockup for create new document.

The following field has been removed from the old form based on comment: author, document type, and date is automatically taken from the operating system.

3. Update Document Data

Vignette: My other son, Mark, needs to update the details of an existing document. He selects from the existing document and clicks to start editing’. After marking the necessary updates and clicks ‘Save’ to apply the changes.

Technical Details:

Editable Fields: Title, Description.

Save Mechanism: Update the existing record in the excel file.

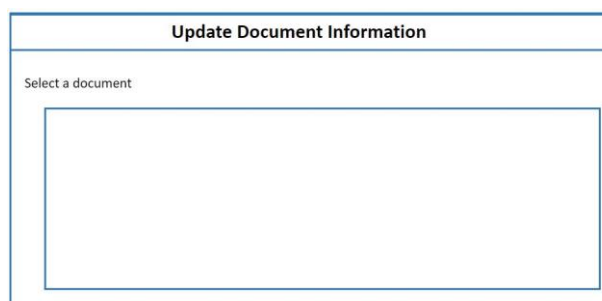
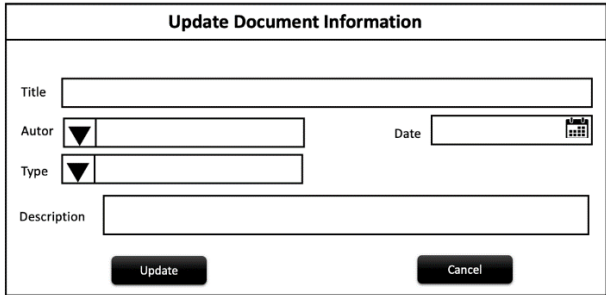


Fig. (5) The new Mockup for update documents.



The mockup shows a form titled "Update Document Information". It contains the following fields and controls:

- Title:** A single-line text input field.
- Author:** A dropdown menu with a downward arrow icon.
- Date:** A date picker control with a calendar icon.
- Type:** A dropdown menu with a downward arrow icon.
- Description:** A multi-line text input field.
- Buttons:** Two buttons at the bottom, "Update" and "Cancel", both with a dark background and white text.

Fig. (6) The old Mockup for update documents.

Author, date, and document type fields have been removed to improve the quality of the application. Version control has also been removed based on feedback.

5. Delete Documents

Vignette: My second daughter, Lisa, needs to delete outdated documents. She selects them from the list, and clicks 'Delete'. The program prompts for confirmation before permanently removing the documents from the excel file.

Technical Details:

Confirmation Prompt: Ensure user confirms deletion.

Audit Trail: Log deletion actions for accountability.



Fig. (7) The new Mockup for delet documents.

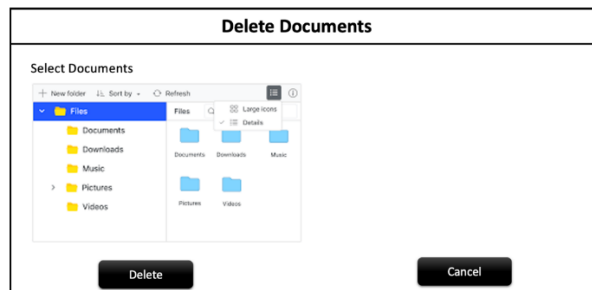


Fig. (8) The old Mockup for delete documents.

The feature of browsing documents or using the preview method has been repealed, and a more convenient and efficient method has been used.

Data Types in the Flow

- **Lists:** For storing multiple documents or search results.
- **Dictionaries:** For storing document attributes (key-value pairs) .
- **String:** For storing file path and search query.

Self-Assessment

After working on the spec, the most significant or unexpected change I had to make from the initial sketch was the decision to remove several unnecessary classes and files, which I think made the project much easier and more organized. I also decided to store the data locally. Given the sensitive and important nature of the documents, and since the application would only have one user, I felt there was no compelling reason to store the data online.

As for my confidence in implementing the spec as it is currently written, I have largely hit the mark and have met all of the specifications required in the project proposal. Although this is the first full application I have developed on my own, while I believe I can meet most of the project requirements in the next phase, I am not sure that I will achieve 100% of the scope.

I will have to redesign some of the interfaces I have previously proposed due to changes and updates to the app, and when linking the interfaces to the code, I will have to revisit the designs of the interfaces and other system components. I suspect there are many areas I don't know much about that may need your help in fully integrating all the parts into one easy-to-use GUI application.

Appendix

General description of the project

The "Family Document Manager" will be a super handy app designed to help families store, organize, and retrieve all their family documents in one place. It will solve the common hassle of trying to find important papers that are usually scattered all over the house and devices. With this app, families will be able to keep everything in one spot, making it easy to grab what they need for things like college applications, medical emergencies, or financial planning. Since the program is family-specific and will be designed for a single user, and due to the sensitivity of family data, the program will not use any external mechanisms such as packages, APIs, or email. Instead, the program will use a desktop GUI and will not require a Command Line Interface (CLI).

This project will be important because it will make managing family documents much easier. In the world of Human-Computer Interaction (HCI), this app will be a great example of user-friendly design. It will have a simple and intuitive interface that anyone can use, no matter their age or tech skills. By following HCI principles, the Family Document Manager will not only make life easier but also help you stay organized and ready for anything.

Plus, the way it will centralize and streamline document management will show how HCI aims to create tech that boosts our abilities and improves our lives. With this app, important documents will be just a few clicks away, giving you the confidence to handle any situation. This project will really show how well-designed tech can solve real problems and make life more efficient and stress-free.

User Story Table for FDMS Ver 1

User Story ID	As a	I want to	So that
1	User	Enter data manually from the CLI	I can add new documents directly into the system.
2	User	Search for documents by keywords	I can easily find specific documents when needed.
3	User	Update document data	I can keep the information up-to-date.
4	User	Delete documents	I can remove outdated or unnecessary documents.
5	User	Generate reports on documents	I can have an overview of all stored documents when searching for files.

Table (1) user story outlines the key functionalities of the Family Document Manager.

Final UML Class Diagram for the Family Document Manager

Class Name	Method/Variables	Purpose
Document	id, title, description, classification, file_path, upload_date	Attributes of the Document class.
	get_fields()	Returns a list of field names in the Document class.
	to_dict()	Converts the Document object to a dictionary.
DocumentRepository	__init__()	Initializes the repository with a CSV file path.
	load_documents()	Loads documents from the CSV file into memory.
	_add_document_from_row()	Creates a Document object from a CSV row and adds it to the list.
	save_documents()	Saves all documents to the CSV file.
	add_document()	Adds a new document to the repository and saves it.
	create_document_txt_file()	Creates a text file containing the document's details.
	_generate_txt_file_path()	Generates a file path for the document's text file.
	list_documents()	Prints a list of all documents in the repository.
	remove_document()	Removes a document from the repository based on user input.
	_confirm_deletion()	Asks for user confirmation before deleting a document.

	_delete_document_files()	Deletes the text and PDF files associated with a document.
	_remove_document_from_list()	Removes a document from the list based on its ID.
	search_documents()	Searches for documents containing a given keyword.
	_basic_search()	Performs a basic search on all document fields.
	get_document_by_id()	Retrieves a document by its ID.
	add_document_interactive()	Interactively adds a new document to the repository.
	generate_unique_id()	Generates a unique ID for a new document.
	generate_file_path()	Generates a file path for a new document's PDF file.
	update_document_interactive()	Interactively updates an existing document in the repository.
main.py	DocumentManagementSystem	Manages the document repository and user interactions.
	display_menu()	Displays the main menu options.
	run()	Runs the main loop to handle user choices.
	update_document()	Updates an existing document.
	delete_document()	Deletes a document.
	search_documents()	Searches for documents based on user input.

Technical "flow"

Data Flow Analysis

Inputs:

- Upload user documents from local storage as a PDF Manually provide metadata by the user (title, description, classification, etc.)
- Search query results by the user via searches for specific documents.

Outputs:

- Organized document repository.
 - Search results based on metadata.
 - Reports generation in various formats (PDF).
 - Document previews by displaying images of documents.
-

Processes:

- **Data Entry and Retrieval:** adding and fetching data based on user queries.
- **Data Update:** Modifying existing records.
- **Data Deletion:** Removing records from the database and files from the folder.