Family Document Manager System

project spec

CHI 584 - Python Application Development

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 1.1

Table (1) summary of changes made after receiving the feedback

- No need for data validation now as that can be done better once you've decided on a GUI. - The following Fields members are removed: ID from User class, notes merged with description, and removed from Document class based on the feedback I received. - - Program features have changed: - There will be no importing data from files (e.g., Excel files). - There will be no display or preview image of the document. - There will be no generation of reports about the documents. - The only file types supported will be pdf	

Project Name: Family Document Manager

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 title
- Document classification
- Document issue date
- Document image
- Document description
- Document status (available/lost/damaged/invalid)

Program Features:

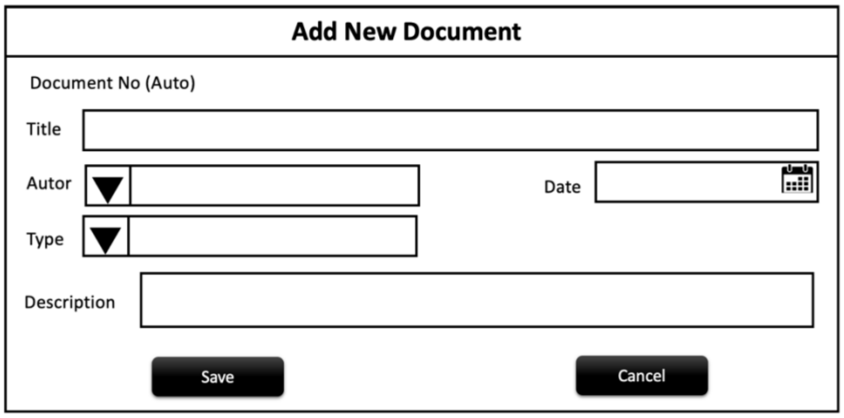
- Manual data entry directly from the program window.
- Search and retrieve file data based on document properties.
- Update document data.
- Delete documents.
- Issue reports on documents.
- Export documents.

Task Vignettes (User activity "flow")

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, category, description, and date of creation. Once she completes the entry, she clicks ‘Save’ to store the information in the database.

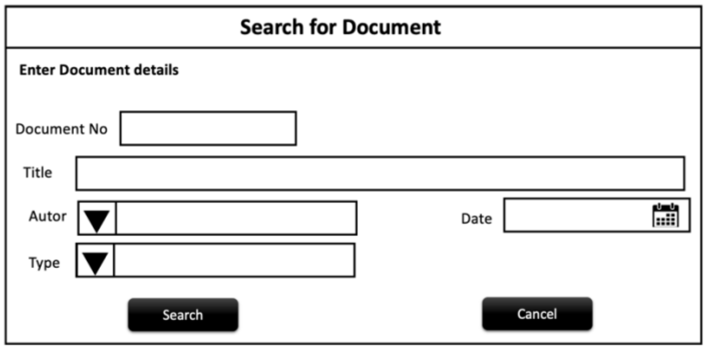
Technical Details:
Input Fields: Document No, Title, Author, Date, Document Type, Document Copy, Description.
Validation: Ensure all required fields are filled.
Save Mechanism: Store data in a structured database.
Mockup:



2. Search and Retrieve File Data

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

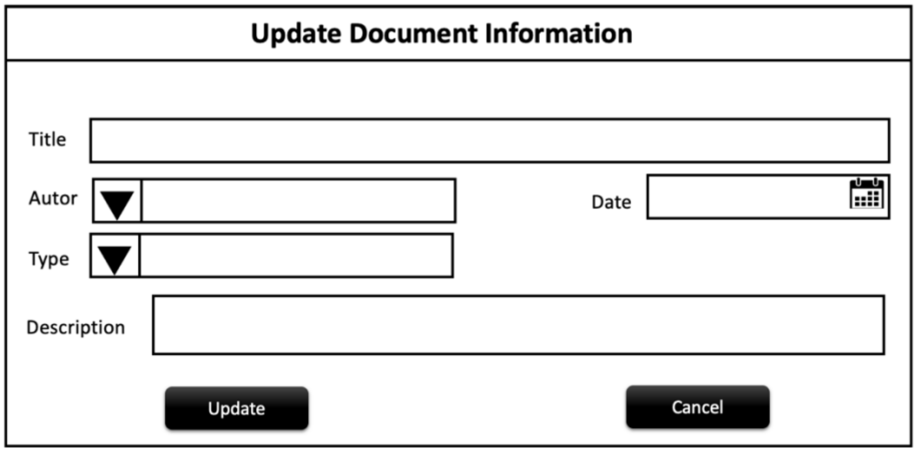
Technical Details:
Search Fields: Document No, Title, Author, Date, Keywords.
Search Algorithm: Implement a fast and efficient search algorithm.
Result Display: Show a list of matching documents with brief details.
Mockup:



3. Update Document Data

Vignette: My other son, Mark, needs to update the details of an existing document. He searches for the document, selects it from the search results, and clicks ‘Edit’. Mark updates the necessary fields and clicks ‘Save’ to apply the changes.

Technical Details:
Editable Fields: Title, Author, Date, Document Type, Description.
Version Control: Maintain a history of changes.
Save Mechanism: Update the existing record in the database.
Mockup:

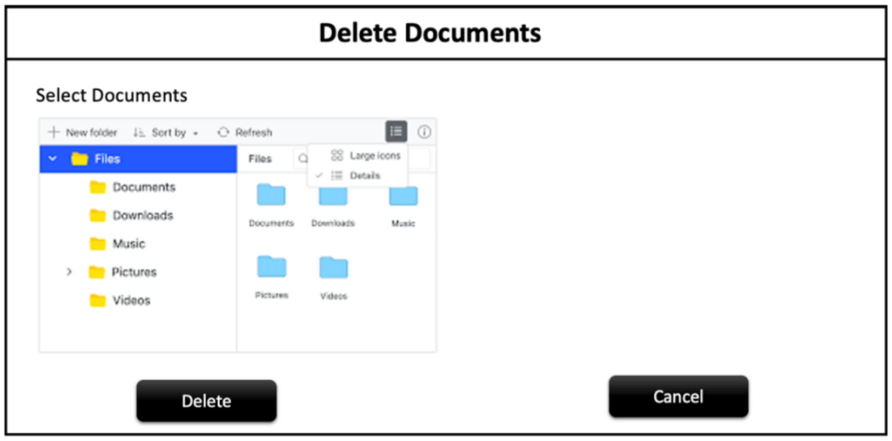


5. Delete Documents

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

Technical Details:
Confirmation Prompt: Ensure user confirms deletion.
Data Integrity: Remove all associated data.
Audit Trail: Log deletion actions for accountability.

Mockup:



Some other technical considerations I'm thinking about:

- Database: use a local DB (sqlite3 — DB-API 2.0 interface for SQLite databases.

Technical "flow"

Data Flow Analysis

Inputs:

- Upload user documents from local storage (various file formats: PDF, JPG, DOC, etc.)
- 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, CSV).
- Document previews by displaying images of documents.
- Exported files by downloadable documents.
-

Processes:

- **Data Validation**: Ensuring the correctness of user inputs and file formats.
- **Data Storage**: Storing validated data in the database.
- **Data Retrieval**: Fetching data based on user queries.
- **Data Update**: Modifying existing records.
- **Data Deletion**: Removing records from the database.
- **Report Generation**: Creating summaries or detailed reports of stored data.
- **Image Preview:** Displaying document images.
- **Data Export:** Exporting documents to local storage.

Possible UML Class Diagram for the Family Document Manager

```
classDiagram


   class User {
      username
      email
      password
   }

   class Document {
      id
      title
      description
      classification
      file_path       # I think there is no need for this variable as all docs will be stored in
DB.
      upload_date
      status
   }

   class  DocumentRepository {
      documents
      add_document()
      remove_document()
      search_documents()
      get_document_by_id()
   }

   class MetadataManager {
      validate_metadata()
      save_metadata()
   }

   class FileUploader {
      upload_document()
      save_document_to_storage()
   }

Main.py
```

DocumentRepository "1" -- "*" Document User "1" -- "*" DocumentRepository
إذا كان الكود التالي مخزن في الملف س

Possible Data Flow Diagram

Sequence Diagram

participant User

participant FileUploader

participant DocumentRepository

participant MetadataManager

User->>FileUploader: Upload document

FileUploader->>DocumentRepository: Create new document

DocumentRepository->>MetadataManager: Get metadata from user

MetadataManager-->DocumentRepository: Return metadata

DocumentRepository->>FileUploader: Save document to storage

FileUploader-->DocumentRepository: Return document ID

DocumentRepository-->User: Document uploaded successfully

Possible Data Types in the Flow

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

- **Arrays**: For handling bulk data operations.
- **DataFrames**: For generating reports.
- **Images**: For document previews.
- **String**: For storing file path and search query.
- **Document:** Document object (custom class).

Self-Assessment

- **After working through the spec, what was the biggest or most unexpected change you had to make from your sketch?**
    - I decided to store the data locally because the documents are sensitive and important, and the application has only one user. So I don't see a good reason to put the data online.
- **How confident do you feel that you can implement the spec as it's written right now?**
    - In fact, this is the first full application I will develop by my hand. So I think that I will succeed in fulfilling most of the project requirements, but I am not confident that I will be able to get the scope 100% done.
- **What is the biggest potential problem that you NEED to solve (or you'll fail)?**
    - Unforeseen risks or major obstacles, especially if they are new and challenging to me. Also, I think time will be an enemy that I will have to watch.
- **What parts are you least familiar with and might need my help?**
    - Integrate the parts completely into a single application. Connect the application to the database and graphical interfaces.

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

User Story ID	As a	I want to	So that
1	User	Enter data manually	I can add new documents directly into the system.
2	User	Search for documents by properties	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.
6	User	Generate reports on documents	I can have an overview of all stored documents.
8	User	Export documents	I can share or backup documents easily.

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