Version 1 Review Assignment

Abduljaleel Hosawi

Project title: Family Documents Manager System (FDMS) Version 1

Introduction

This report outlines the achievements made over the past few weeks in developing the core functionalities of the Family Documents Manager System (FDMS). Additionally, it includes a self-assessment of the project's progress and identifies any necessary course corrections.

Project Overview

The FDMS Version 1 is a command-line interface (CLI) application designed to efficiently store and organize family documents. The application captures essential metadata for each document, such as ID, file name, and category, and stores this information in a `document.csv` file. It also supports creating PDF copies of the documents. The document categories include Education, Health, Financial, Government and Private Services, Self-Development, and Family Photos, which can be entered by the user.

Users can manually create new files and input their information. They can also search for documents using any associated keywords, update document metadata, or delete files from the system. This application addresses the common issue of locating and managing various family documents, which are often scattered across multiple locations and devices. By centralizing document storage, the FDMS ensures that important documents are easily accessible when needed.

Some Technical Details

- **Error Handling**: If there are issues loading documents from the CSV file, errors are printed, and problematic rows are skipped.
- **File Management**: The application ensures that directories for storing documents and text files are created if they do not exist.
- **Interactive Prompts**: The application uses interactive prompts to guide the user through creating, updating, and deleting documents.
- **Automate IDs for documents**: The application automatically creates an ID for each new document.
- **Automate new document creation:** The application creates a new text document and stores the document data including the number, title, description, classification, and path.
- **Automatically assign creation date:** The application ensures that the document creation date is automatically stored by the operating system.
- **Command-Line Interface (CLI)**: The application operates through a CLI, making it accessible and straightforward to use.
- **Object-Oriented Programming (OOP)**: The project is built using OOP principles and was initially planned to include multiple code files and several classes such as User, Document, DocumentRepository, MetadataManager, and FileUploader. However, some of these classes are cancelled.
- **Document storage:** data is stored and managed in the document document.csv.
- **Code Quality and Complexity**: Throughout the project's development, continuous assessment of code complexity and quality was conducted. Feedback from various assignments led to improvements and the elimination of some classes to streamline the codebase.

Table (1) summary of Classes used in the FDMS project

Class Name	Method/Variable	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.

Table (1) provides a concise overview of the methods and variables in each class file and their purposes.

Table (2) the data structures used in the FDMS

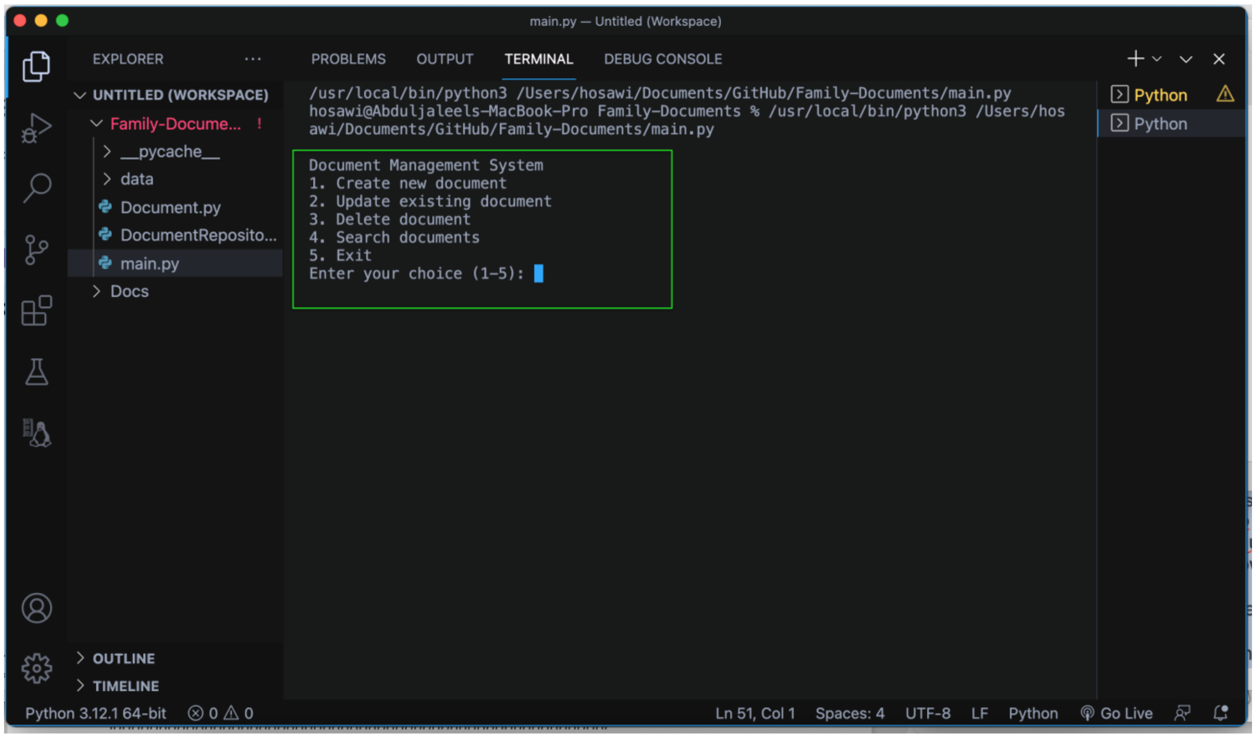
Data Structure	**Type**	**Usage**
Document	Class	Represents a document with attributes like id, title, description, classification, file_path, and upload_date.
documents	List	Stores instances of Document objects in DocumentRepository.
csv_file	Path	Represents the path to the CSV file used to store document data.
reader	DictReader	Reads rows from the CSV file into dictionaries for processing.

Table (2) highlights the key data structures and their roles.

Demonstration of the FDMS Ver 1

Step-by-Step Description and Demonstration:

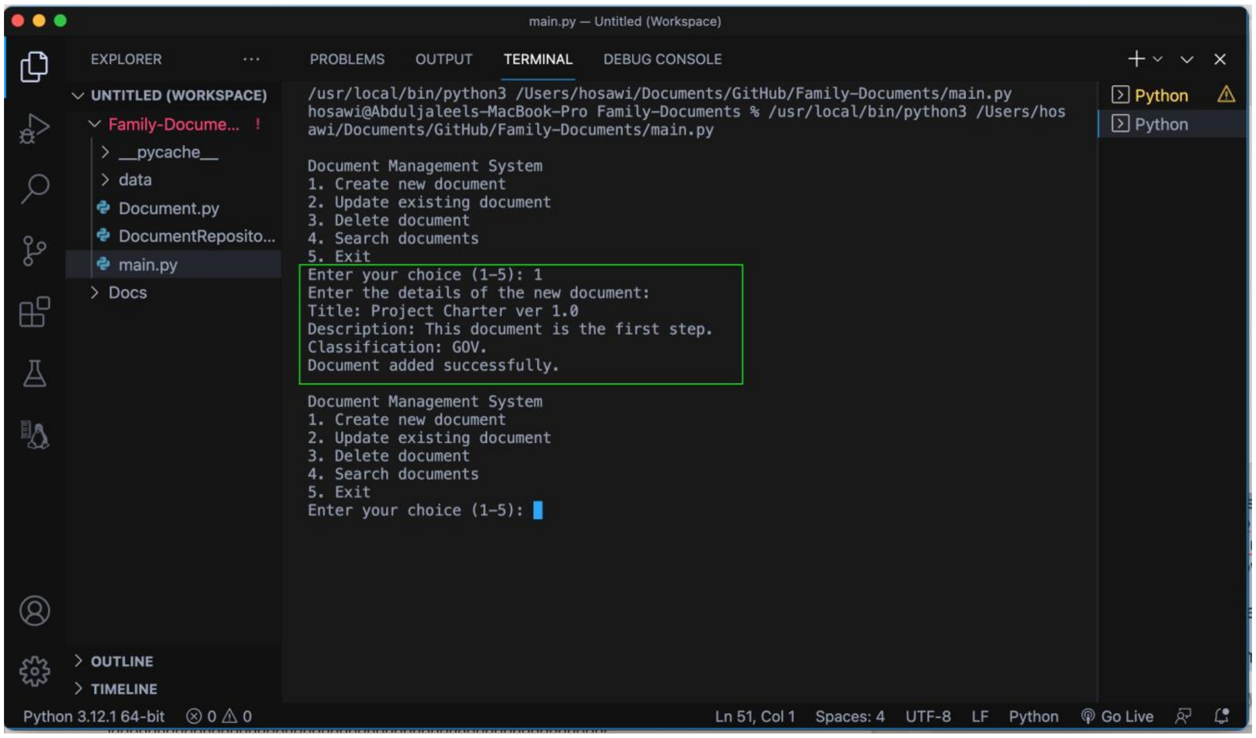
1. **Initialization**:
    a. The DocumentManagementSystem class initializes a DocumentRepository instance, which loads documents from a CSV file (data/document.csv).
2. **Main Menu**:
    a. The user is presented with a menu with options to create, update, delete, search documents, or exit the program.



```
/usr/local/bin/python3 /Users/hosawi/Documents/GitHub/Family-Documents/main.py
hosawi@Abduljaleels-MacBook-Pro Family-Documents % /usr/local/bin/python3 /Users/hos
awi/Documents/GitHub/Family-Documents/main.py

Document Management System
1. Create new document
2. Update existing document
3. Delete document
4. Search documents
5. Exit
Enter your choice (1-5):
```

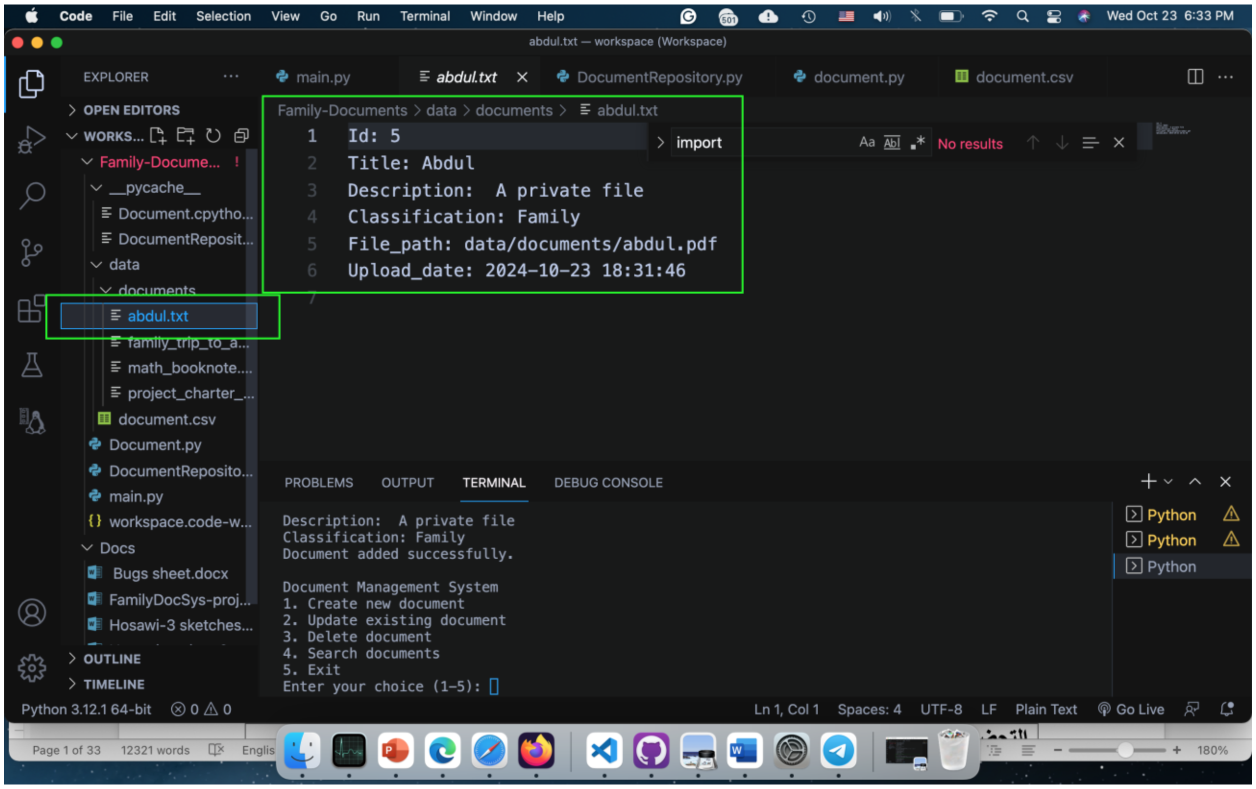
Figure (1) main menu dispaly.

3. **Creating a New Document**:
   a. **User Input**: The user selects the option to create a new document and provides details like title, description, and classification.
   b. **Document Creation**: A unique ID and file path are generated, and the document is added to the repository.
   c. **File Creation**: A text file with the document's details is created.



```
/usr/local/bin/python3 /Users/hosawi/Documents/GitHub/Family-Documents/main.py
hosawi@Abduljaleels-MacBook-Pro Family-Documents % /usr/local/bin/python3 /Users/hos
awi/Documents/GitHub/Family-Documents/main.py

Document Management System
1. Create new document
2. Update existing document
3. Delete document
4. Search documents
5. Exit
Enter your choice (1-5): 1
Enter the details of the new document:
Title: Project Charter ver 1.0
Description: This document is the first step.
Classification: GOV.
Document added successfully.

Document Management System
1. Create new document
2. Update existing document
3. Delete document
4. Search documents
5. Exit
Enter your choice (1-5):
```

Figure (2) adding a new Document process.

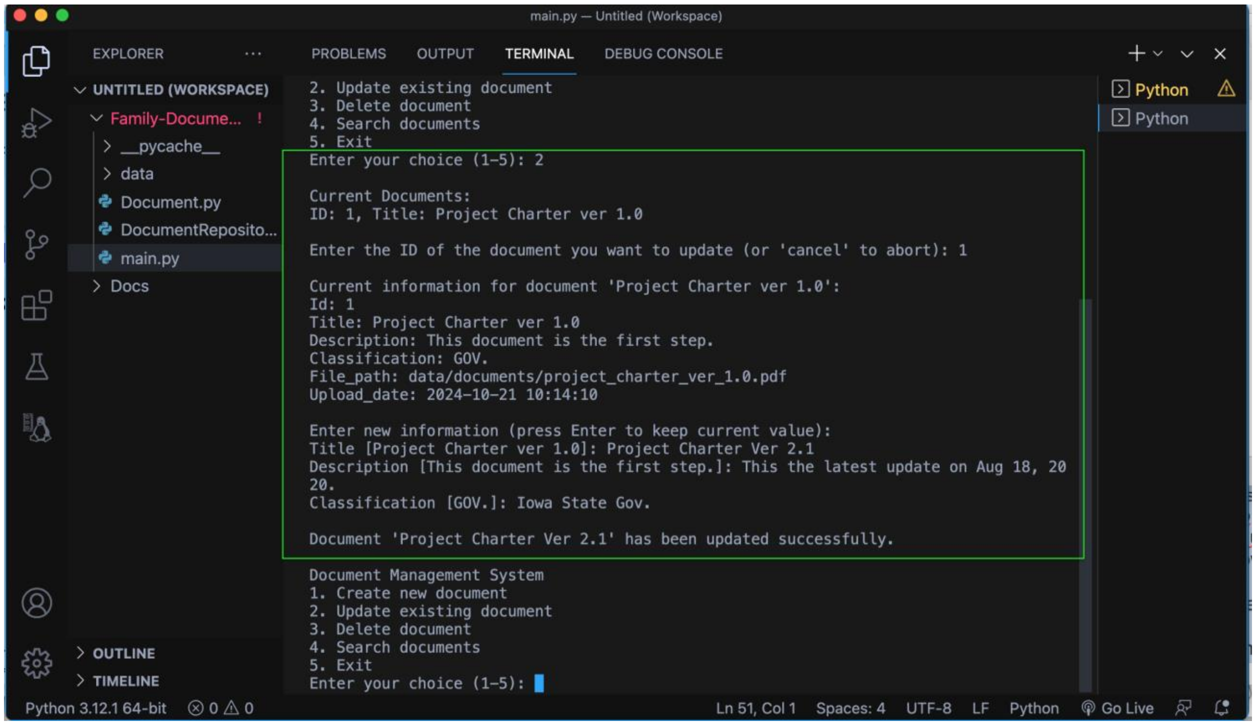


```
Id: 5
Title: Abdul
Description:  A private file
Classification: Family
File_path: data/documents/abdul.pdf
Upload_date: 2024-10-23 18:31:46
```

```
Description:  A private file
Classification: Family
Document added successfully.

Document Management System
1. Create new document
2. Update existing document
3. Delete document
4. Search documents
5. Exit
Enter your choice (1-5):
```

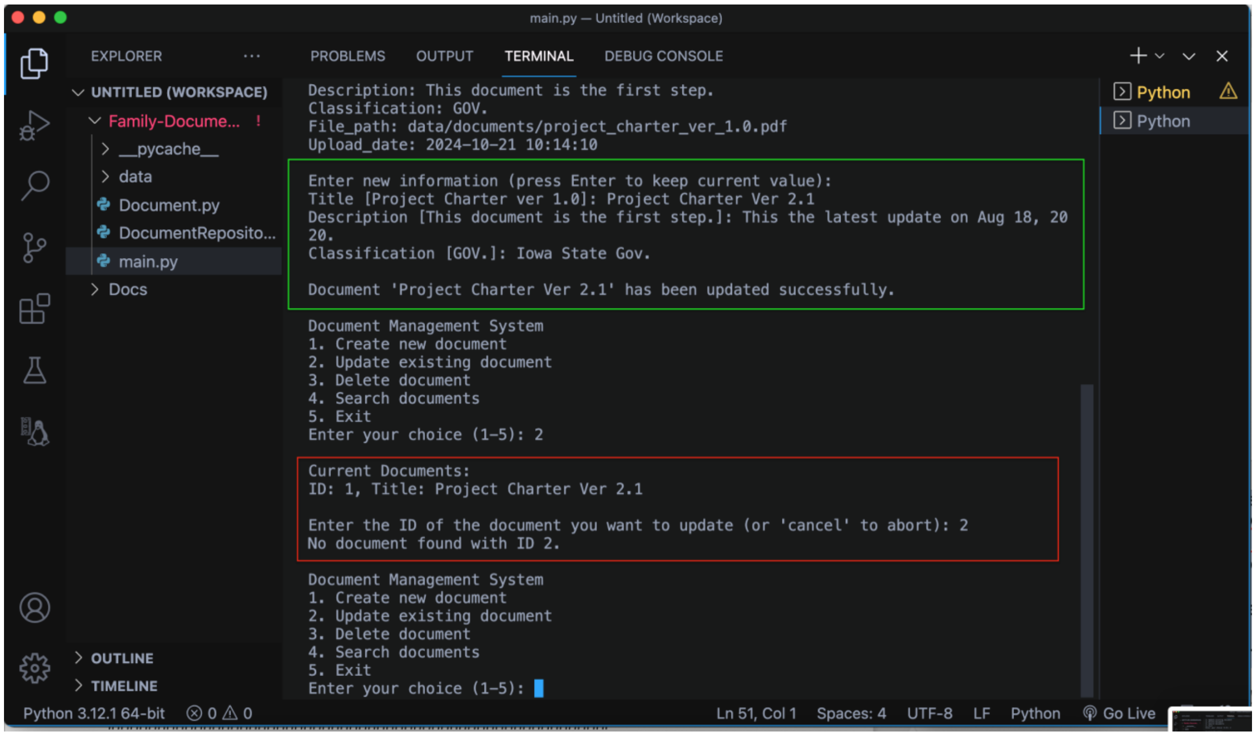
Figure (3) stores the new document details inside the same document.

4. **Updating an Existing Document**:
    a. **List Documents**: The user is shown a list of current documents.
    b. **User Input**: The user selects a document to update by its ID.
    c. **Update Details**: The user can update the title, description, and classification. The upload date is automatically updated.
    d. **File Update**: If the title changes, the old text file is deleted, and a new one is created.



```
2. Update existing document
3. Delete document
4. Search documents
5. Exit
Enter your choice (1-5): 2

Current Documents:
ID: 1, Title: Project Charter ver 1.0

Enter the ID of the document you want to update (or 'cancel' to abort): 1

Current information for document 'Project Charter ver 1.0':
Id: 1
Title: Project Charter ver 1.0
Description: This document is the first step.
Classification: GOV.
File_path: data/documents/project_charter_ver_1.0.pdf
Upload_date: 2024-10-21 10:14:10

Enter new information (press Enter to keep current value):
Title [Project Charter ver 1.0]: Project Charter Ver 2.1
Description [This document is the first step.]: This the latest update on Aug 18, 20
20.
Classification [GOV.]: Iowa State Gov.

Document 'Project Charter Ver 2.1' has been updated successfully.

Document Management System
1. Create new document
2. Update existing document
3. Delete document
4. Search documents
5. Exit
Enter your choice (1-5):
```

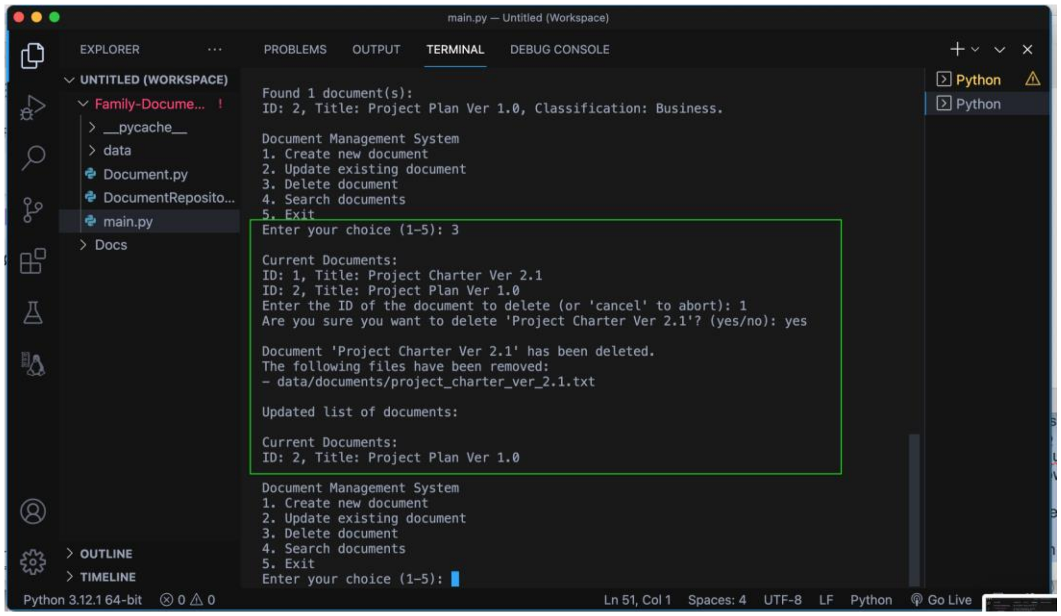
Figure (4) document update process.



```
Description: This document is the first step.
Classification: GOV.
File_path: data/documents/project_charter_ver_1.0.pdf
Upload_date: 2024-10-21 10:14:10

Enter new information (press Enter to keep current value):
Title [Project Charter ver 1.0]: Project Charter Ver 2.1
Description [This document is the first step.]: This the latest update on Aug 18, 20
20.
Classification [GOV.]: Iowa State Gov.

Document 'Project Charter Ver 2.1' has been updated successfully.

Document Management System
1. Create new document
2. Update existing document
3. Delete document
4. Search documents
5. Exit
Enter your choice (1-5): 2

Current Documents:
ID: 1, Title: Project Charter Ver 2.1

Enter the ID of the document you want to update (or 'cancel' to abort): 2
No document found with ID 2.

Document Management System
1. Create new document
2. Update existing document
3. Delete document
4. Search documents
5. Exit
Enter your choice (1-5):
```

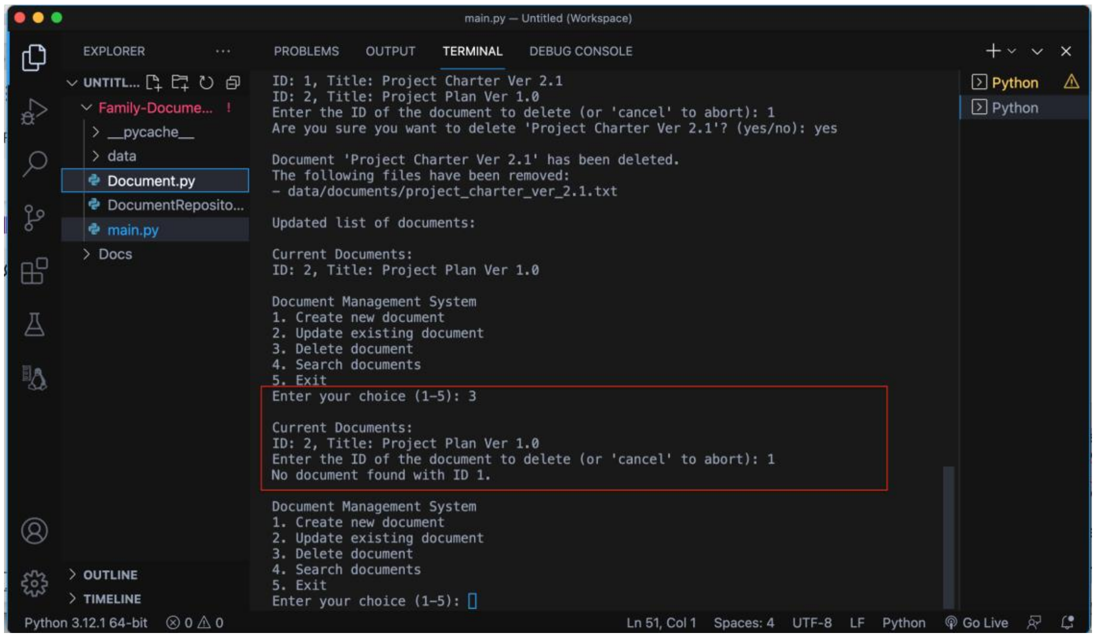
Figure (5) Document update confirmation or error note for wrong entered.

5. **Deleting a Document**:
    a. **List Documents**: The user is shown a list of current documents.
    b. **User Input**: The user selects a document to delete by its ID.
    c. **Confirmation**: The user confirms the deletion.
    d. **File Deletion**: The associated text and PDF files are deleted, and the document is removed from the repository.



```
Found 1 document(s):
ID: 2, Title: Project Plan Ver 1.0, Classification: Business.

Document Management System
1. Create new document
2. Update existing document
3. Delete document
4. Search documents
5. Exit
Enter your choice (1-5): 3

Current Documents:
ID: 1, Title: Project Charter Ver 2.1
ID: 2, Title: Project Plan Ver 1.0
Enter the ID of the document to delete (or 'cancel' to abort): 1
Are you sure you want to delete 'Project Charter Ver 2.1'? (yes/no): yes

Document 'Project Charter Ver 2.1' has been deleted.
The following files have been removed:
- data/documents/project_charter_ver_2.1.txt

Updated list of documents:

Current Documents:
ID: 2, Title: Project Plan Ver 1.0

Document Management System
1. Create new document
2. Update existing document
3. Delete document
4. Search documents
5. Exit
Enter your choice (1-5):
```

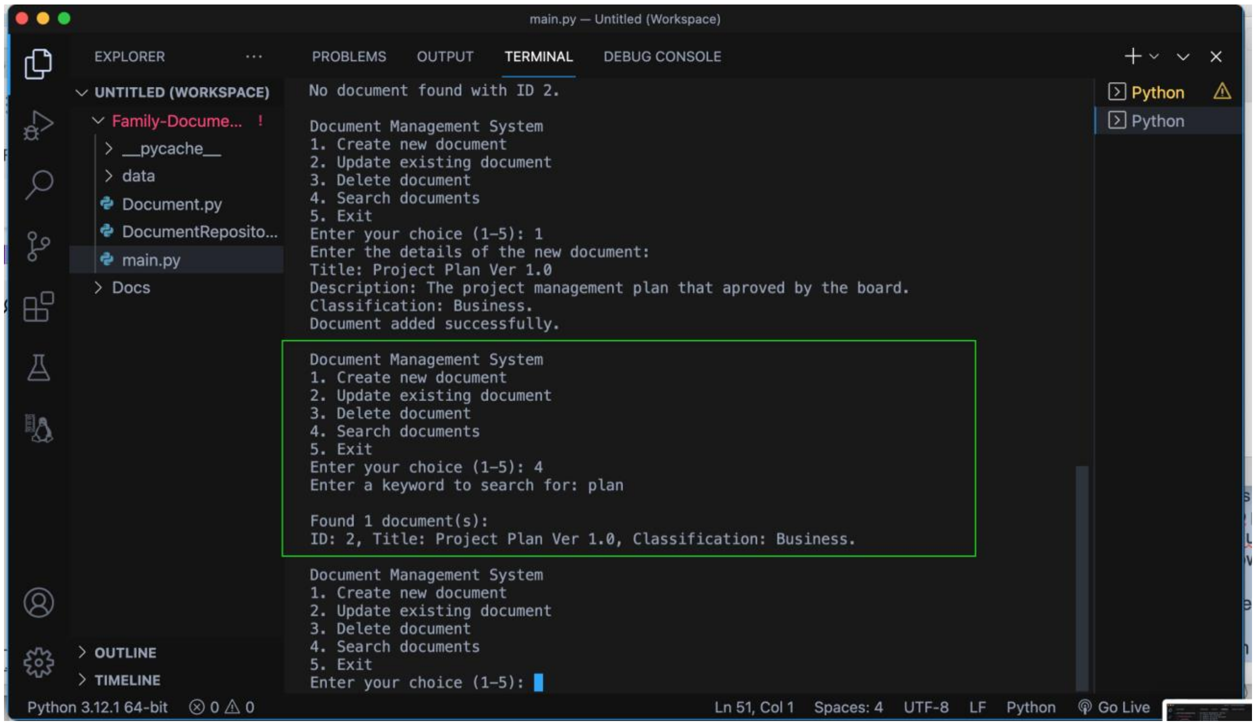
Figure (6) current documents list and deletion process.



```
ID: 1, Title: Project Charter Ver 2.1
ID: 2, Title: Project Plan Ver 1.0
Enter the ID of the document to delete (or 'cancel' to abort): 1
Are you sure you want to delete 'Project Charter Ver 2.1'? (yes/no): yes

Document 'Project Charter Ver 2.1' has been deleted.
The following files have been removed:
- data/documents/project_charter_ver_2.1.txt

Updated list of documents:

Current Documents:
ID: 2, Title: Project Plan Ver 1.0

Document Management System
1. Create new document
2. Update existing document
3. Delete document
4. Search documents
5. Exit
Enter your choice (1-5): 3

Current Documents:
ID: 2, Title: Project Plan Ver 1.0
Enter the ID of the document to delete (or 'cancel' to abort): 1
No document found with ID 1.

Document Management System
1. Create new document
2. Update existing document
3. Delete document
4. Search documents
5. Exit
Enter your choice (1-5):
```

Figure (7) deletion error message.

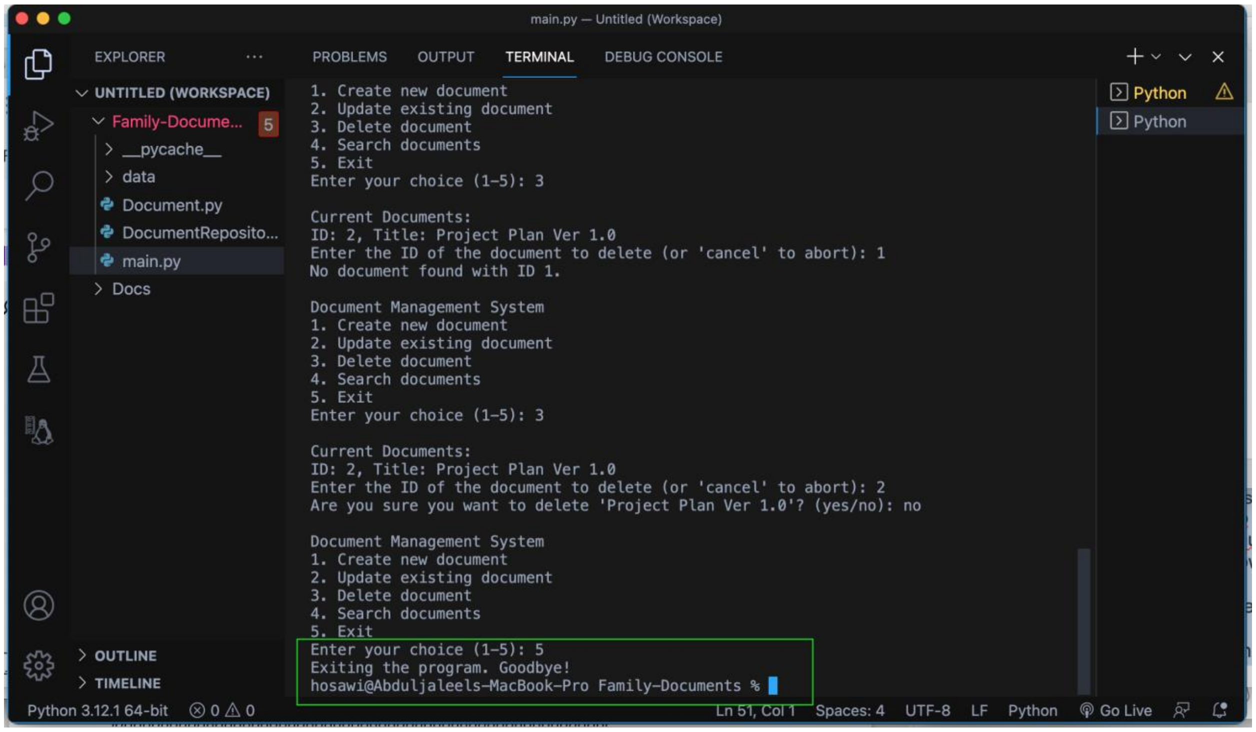
6. **Searching for Documents**:
   a. **User Input**: The user enters a keyword to search for.
   b. **Search Results**: The application searches all document fields for the keyword and displays matching documents.



```
No document found with ID 2.

Document Management System
1. Create new document
2. Update existing document
3. Delete document
4. Search documents
5. Exit
Enter your choice (1-5): 1
Enter the details of the new document:
Title: Project Plan Ver 1.0
Description: The project management plan that aproved by the board.
Classification: Business.
Document added successfully.

Document Management System
1. Create new document
2. Update existing document
3. Delete document
4. Search documents
5. Exit
Enter your choice (1-5): 4
Enter a keyword to search for: plan

Found 1 document(s):
ID: 2, Title: Project Plan Ver 1.0, Classification: Business.

Document Management System
1. Create new document
2. Update existing document
3. Delete document
4. Search documents
5. Exit
Enter your choice (1-5): 
```

Figure (8) Search Results

7. **Exiting the Program**:
    a. The user selects the option to exit, and the program terminates.



```
1. Create new document
2. Update existing document
3. Delete document
4. Search documents
5. Exit
Enter your choice (1-5): 3

Current Documents:
ID: 2, Title: Project Plan Ver 1.0
Enter the ID of the document to delete (or 'cancel' to abort): 1
No document found with ID 1.

Document Management System
1. Create new document
2. Update existing document
3. Delete document
4. Search documents
5. Exit
Enter your choice (1-5): 3

Current Documents:
ID: 2, Title: Project Plan Ver 1.0
Enter the ID of the document to delete (or 'cancel' to abort): 2
Are you sure you want to delete 'Project Plan Ver 1.0'? (yes/no): no

Document Management System
1. Create new document
2. Update existing document
3. Delete document
4. Search documents
5. Exit
Enter your choice (1-5): 5
Exiting the program. Goodbye!
hosawi@Abduljaleels-MacBook-Pro Family-Documents %
```

Figure (9) exit the FDMS

Issues I encountered in the FDSM project

As a beginner programmer embarking on my first self-made project, I encountered numerous challenges. Initially, my lack of confidence and an overwhelming sense of dread were significant obstacles, especially since my previous attempts to learn programming had only been partially successful. I had studied various languages including C, Visual Basic, C++, and Java, but each time I only developed isolated functions without integrating them into a cohesive project. This left me doubting my ability to complete the project, particularly in the beginning stages.

One of the major hurdles was dealing with files, as I had never used CSV files to store document metadata before. However, the assignment that preceded the project helped me overcome this barrier. The feedback I received suggesting that I use a CSV file instead of a database was invaluable, and I am grateful to you, Dr. Chris, for this recommendation.

Among the issues that consumed a significant amount of time were handling programming errors. These included problems such as file not found errors, incorrect data formats, semantic errors, and the challenges of creating TXTs. Ensuring that TXT files were created and stored correctly, along with dealing with file paths, was particularly demanding.

Despite these obstacles, I persevered and gained a deeper understanding of the complexities involved in programming projects. This experience, though challenging, has been immensely educational and rewarding.

Project Milestones for FDMS Version 2

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.

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.