Team members: Aishwarya Kharche and Shikha Kanojia

Library Management System Documentation

Overview

The Library Management System is a Python application that allows users to manage a collection of books. Users can add, delete, update, and display books, as well as scrape book titles from an online source and send this information via email.

Features

- Add new books to the collection.
- Delete books by ID.
- Display all books in the collection.
- Read book details by ID.
- Update book details by ID.
- Scrape book titles from an online source.
- Send scraped book titles via email.

Dependencies

This project requires the following Python libraries:

- requests: For making HTTP requests.
- BeautifulSoup: For parsing HTML content.
- smtplib: For sending emails.
- datetime: For handling date and time.

Installation

To install the required libraries, use the following command:

bash

Copy code

pip install requests beautifulsoup4

Code Structure

Classes

Book

Defines a book object with its attributes.

```
class Book:
    def __init__(self, id, title, price, copies):
        self.id = id
        self.title = title
        self.price = price
        self.copies = copies

    def __str__(self):
        return f'[id={self.id}, title={self.title}, price={self.price}, copies={self.copies}

    def __repr__(self):
        return self.__str__()
```

Functions

Book Management Functions

- book_add(id, title, price, copies): Adds a new book to the collection.

python

Copy code

```
def book_add(id, title, price, copies):
    global books
    book = Book(id, title, price, copies)
    books.append(book)
```

- book_readbyid(id): Displays details of a book by ID.

python

Copy code

- book_updatebyid(id): Updates details of a book by ID.

python

Copy code

```python
def book_updatebyid(id):
    global books
    for book in books:
        if book.id == id:
            title = input("Enter new title for the book: ")
            price = float(input("Enter new price for the book: "))
            copies = int(input("Enter new number of copies: "))
            book.title = title
            book.price = price
            book.copies = copies
            print(f"Book with id {id} has been updated.")
            return
    print(f"Book with id {id} not found.")
```

- book_display(): Displays all books in the collection.

python

Copy code

```python
def book_display():
    global books
    for book in books:
        print(book)
```

Web Scraping Function

- scrape_books(): Scrapes book titles from the website "https://books.toscrape.com/" and saves them to a text file.

python

Copy code

```python
def scrape_books():
    url = 'https://books.toscrape.com/'
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')
    headings = soup.find_all('h3')

    books_info = []
    for heading in headings:
        book_title = heading.find('a')['title']
        books_info.append(book_title)

    with open('books_info.txt', 'w', encoding='UTF-8') as book_file:
        for book in books_info:
            book_file.write(book + '\n')

    print('Books information gathered and saved to books_info.txt')
    return books_info
```

Email Function

- send_email(books_info): Sends an email containing the scraped book titles.

python

Copy code

```python
def send_email(books_info):
    serial = subprocess.check_output('wmic bios get serialnumber').decode("utf-8").replace

    connection = smtp.SMTP_SSL('smtp.gmail.com', 465)
    email_addr = 'your_email@gmail.com'
    email_passwd = 'your_password'  # replace with your actual password
    connection.login(email_addr, email_passwd)

    receiver = 'receiver_email@gmail.com'
    dt_time = datetime.now().strftime("%m/%d/%Y, %H:%M:%S")
    subject = f'Books Info {dt_time} @{serial}'
    body = "Here are the scraped book titles:\n\n" + "\n".join(books_info)

    msg = MIMEMultipart()
    msg['From'] = email_addr
    msg['To'] = receiver
    msg['Subject'] = subject
    msg.attach(MIMEText(body, 'plain'))

    connection.sendmail(email_addr, receiver, msg.as_string())
    connection.close()
    print('Email sent successfully.')
```

Menu Functions

- menu(): Displays the main menu and processes user input.

python

Copy code

```
def menu():
    choice = int(input('''1 - Add book
2 - Delete book by id
3 - Display all books
4 - Read book by id
5 - Update book by id
6 - Scrape books and send email
7 - Send email with book titles
8 - End
Your choice: '''))

    # Process the choice (omitted for brevity)
    return choice
```

menus(): Loops through the menu until the user chooses to exit.

```
def menus():
    choice = menu()
    while choice != 8:
        choice = menu()
    print('Thank you for using the library management system.')
```

Main Program

The driver program starts by calling the menus() function, which handles user interactions.

python

Copy code

```
if __name__ == "__main__":
    menus()
```

Usage

1. Run the program.

2. Choose an option from the menu:
    - Add a book
    - Delete a book by ID
    - Display all books
    - Read book details by ID
    - Update book details by ID
    - Scrape book titles and send an email
    - Exit the program
3. Follow the prompts based on your selection.

Security Considerations

- Be cautious with email credentials. Consider using environment variables or secure vaults to store sensitive information.
- Ensure the send_email function uses secure methods for authentication.

Future Enhancements

- Implement a database for persistent storage of books.
- Add user authentication and authorization.
- Enhance error handling and validation for user inputs.
- Allow bulk operations (e.g., importing multiple books).

Conclusion

This Library Management System is a simple yet effective way to manage book collections, leveraging web scraping and email functionalities for a seamless experience. It serves as a foundation that can be expanded with additional features and improvements.