Library Management System

Sta. Rosa, Andrei

Garcia, Daniel

Uanan, Christian

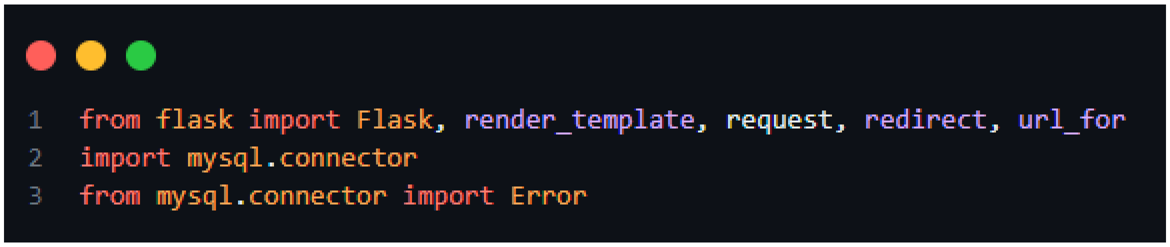
HARDWARE & SOFTWARE USED

- Python 3.12
- Flask 3.1
- MySQL 8.0.35
- MySQL Workbench
- HTML & CSS

PROGRAMMING LANGUAGE USED

- Python
- MySQL

APPLICATION STRUCTURE

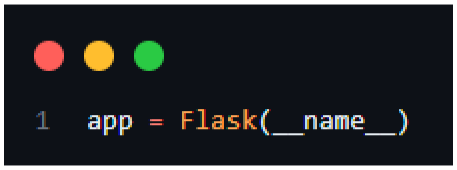


```python
from flask import Flask, render_template, request, redirect, url_for
import mysql.connector
from mysql.connector import Error
```

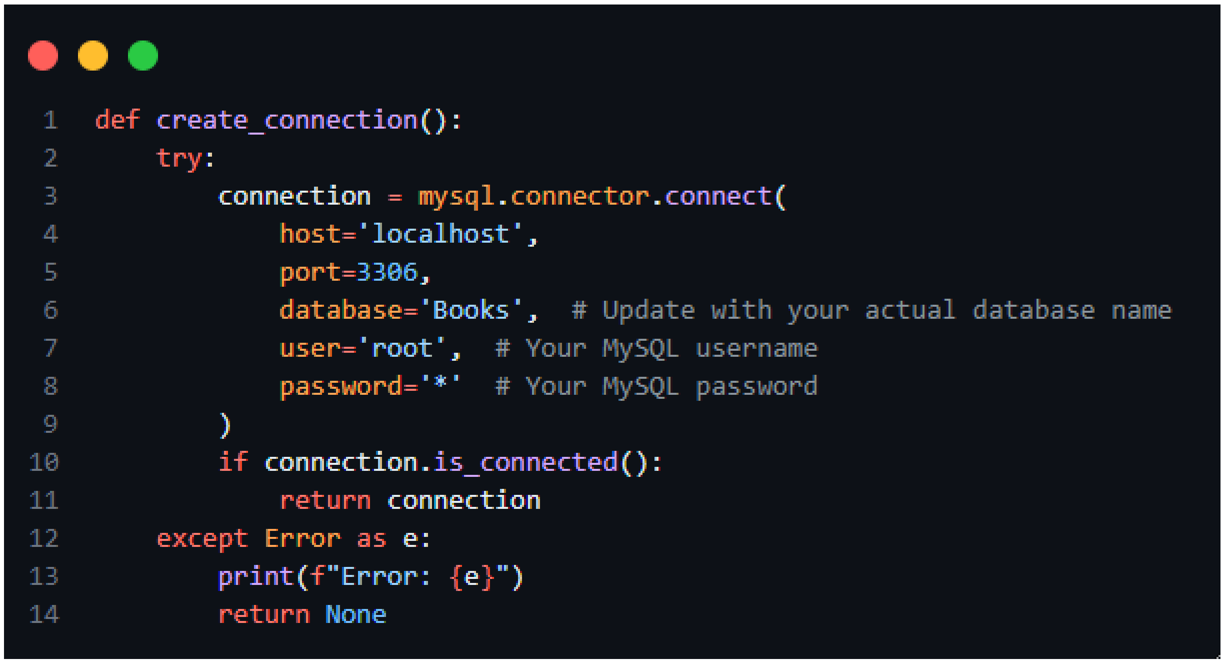
FLASK: Used for creating web application

mysql.connector: Used for connecting and interacting with the MySQL database.

Initializes the Flask Application



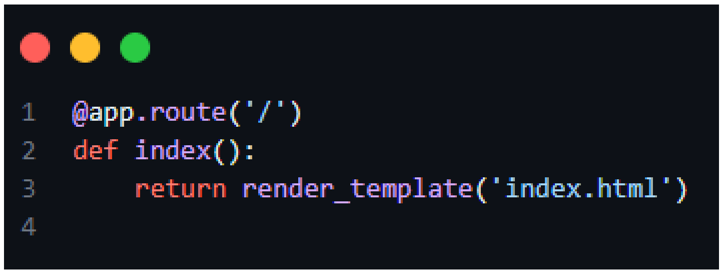
```python
app = Flask(__name__)
```



```python
def create_connection():
    try:
        connection = mysql.connector.connect(
            host='localhost',
            port=3306,
            database='Books',  # Update with your actual database name
            user='root',  # Your MySQL username
            password='*'  # Your MySQL password
        )
        if connection.is_connected():
            return connection
    except Error as e:
        print(f"Error: {e}")
        return None
```

Establishes a connection to the MySQL database and returns the connection object. Handles errors by printing them to the console.

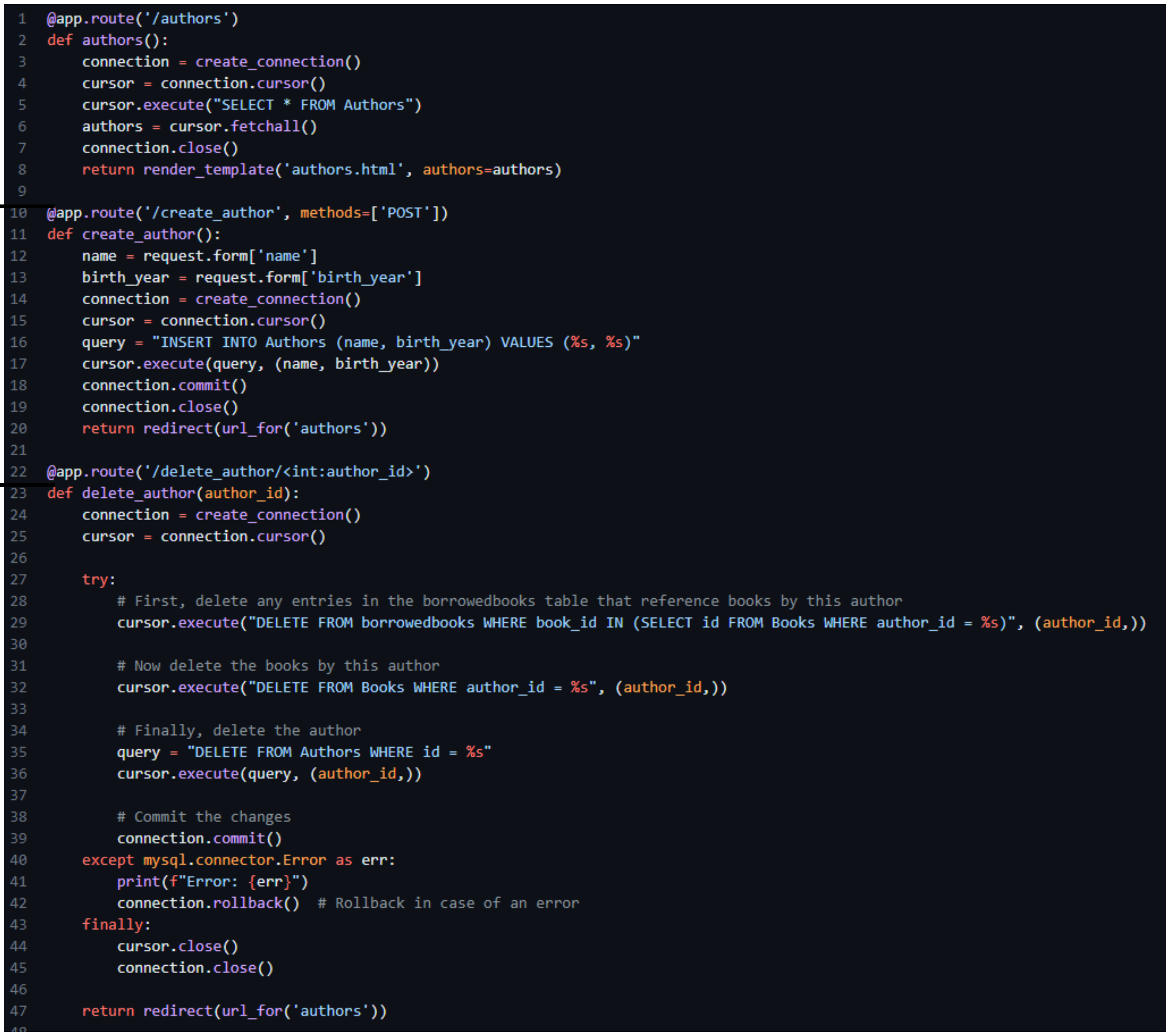
ROUTES AND VIEWS



```python
@app.route('/')
def index():
    return render_template('index.html')
```

Renders the main page of the application.

AUTHOR ROUTES

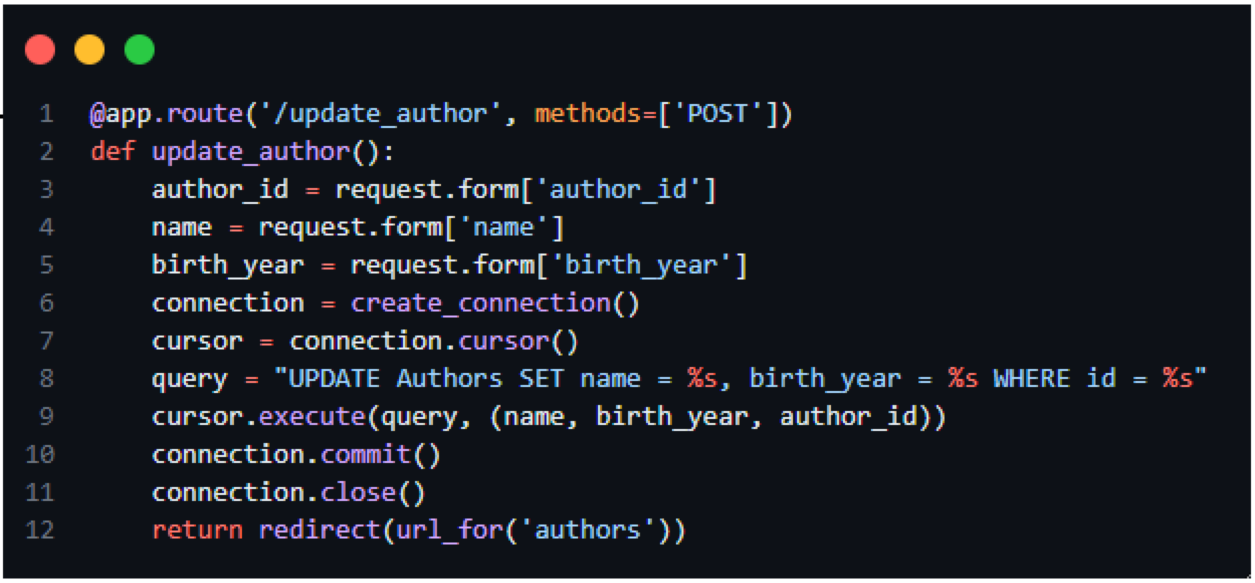


```python
@app.route('/authors')
def authors():
    connection = create_connection()
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM Authors")
    authors = cursor.fetchall()
    connection.close()
    return render_template('authors.html', authors=authors)

@app.route('/create_author', methods=['POST'])
def create_author():
    name = request.form['name']
    birth_year = request.form['birth_year']
    connection = create_connection()
    cursor = connection.cursor()
    query = "INSERT INTO Authors (name, birth_year) VALUES (%s, %s)"
    cursor.execute(query, (name, birth_year))
    connection.commit()
    connection.close()
    return redirect(url_for('authors'))

@app.route('/delete_author/<int:author_id>')
def delete_author(author_id):
    connection = create_connection()
    cursor = connection.cursor()

    try:
        # First, delete any entries in the borrowedbooks table that reference books by this author
        cursor.execute("DELETE FROM borrowedbooks WHERE book_id IN (SELECT id FROM Books WHERE author_id = %s)", (author_id,))

        # Now delete the books by this author
        cursor.execute("DELETE FROM Books WHERE author_id = %s", (author_id,))

        # Finally, delete the author
        query = "DELETE FROM Authors WHERE id = %s"
        cursor.execute(query, (author_id,))

        # Commit the changes
        connection.commit()
    except mysql.connector.Error as err:
        print(f"Error: {err}")
        connection.rollback()  # Rollback in case of an error
    finally:
        cursor.close()
        connection.close()

    return redirect(url_for('authors'))
```

Fetches and displays all authors from the database.

Handles the creation of a new author.

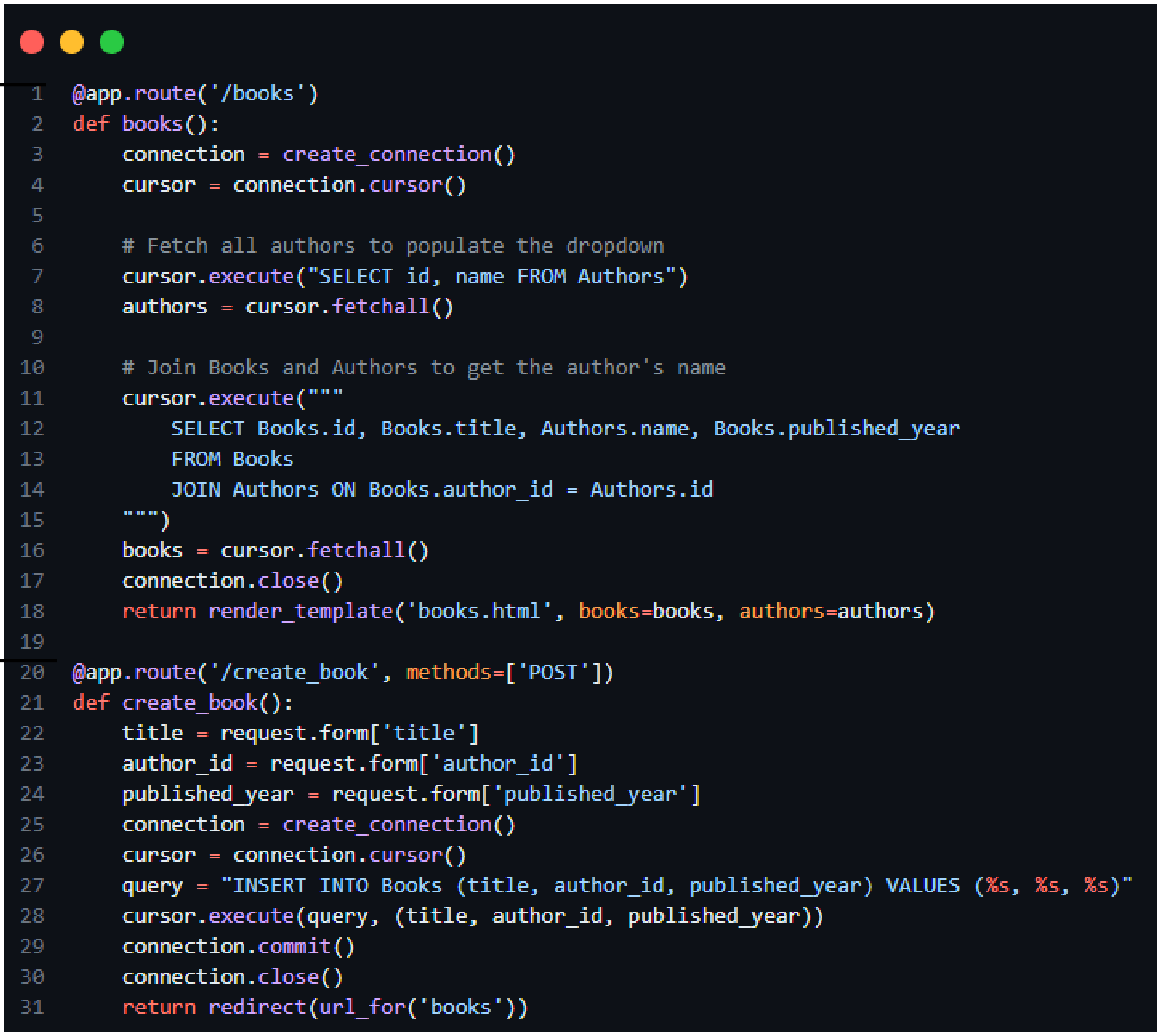
Deletes an author by their ID.



```python
@app.route('/update_author', methods=['POST'])
def update_author():
    author_id = request.form['author_id']
    name = request.form['name']
    birth_year = request.form['birth_year']
    connection = create_connection()
    cursor = connection.cursor()
    query = "UPDATE Authors SET name = %s, birth_year = %s WHERE id = %s"
    cursor.execute(query, (name, birth_year, author_id))
    connection.commit()
    connection.close()
    return redirect(url_for('authors'))
```

Updates an existing author's details.

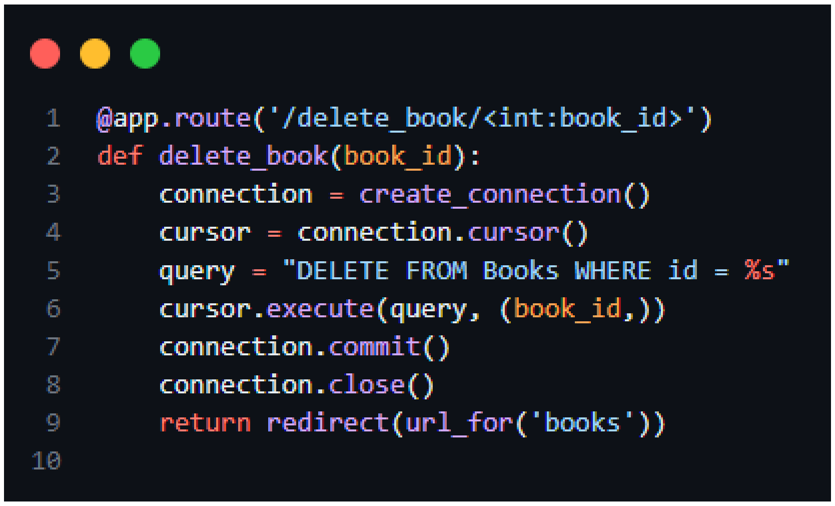
BOOK ROUTES



```python
@app.route('/books')
def books():
    connection = create_connection()
    cursor = connection.cursor()

    # Fetch all authors to populate the dropdown
    cursor.execute("SELECT id, name FROM Authors")
    authors = cursor.fetchall()

    # Join Books and Authors to get the author's name
    cursor.execute("""
        SELECT Books.id, Books.title, Authors.name, Books.published_year
        FROM Books
        JOIN Authors ON Books.author_id = Authors.id
    """)
    books = cursor.fetchall()
    connection.close()
    return render_template('books.html', books=books, authors=authors)

@app.route('/create_book', methods=['POST'])
def create_book():
    title = request.form['title']
    author_id = request.form['author_id']
    published_year = request.form['published_year']
    connection = create_connection()
    cursor = connection.cursor()
    query = "INSERT INTO Books (title, author_id, published_year) VALUES (%s, %s, %s)"
    cursor.execute(query, (title, author_id, published_year))
    connection.commit()
    connection.close()
    return redirect(url_for('books'))
```

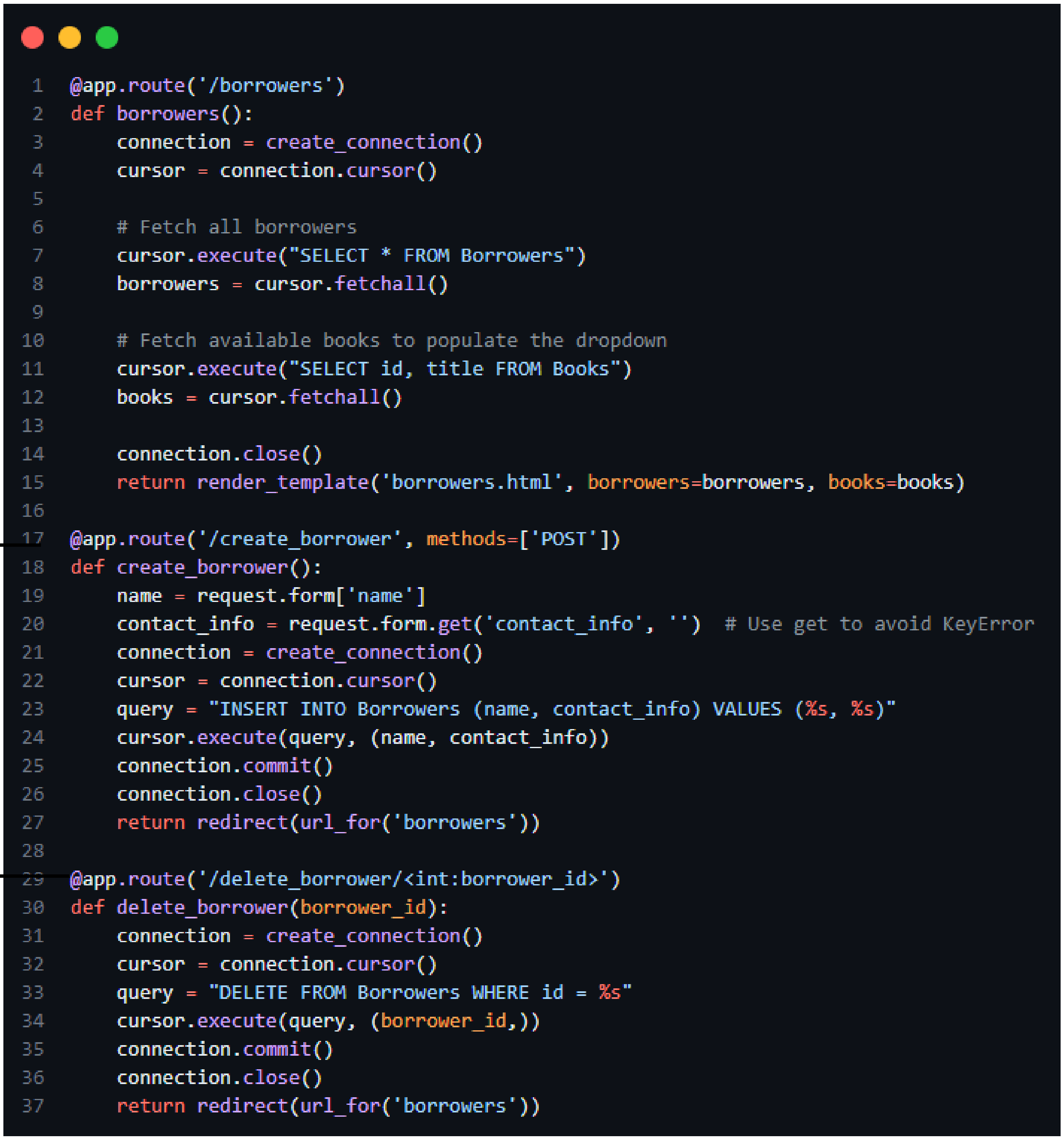
Fetches and displays all books along with their authors.

Handles the creation of a new book.



```python
@app.route('/delete_book/<int:book_id>')
def delete_book(book_id):
    connection = create_connection()
    cursor = connection.cursor()
    query = "DELETE FROM Books WHERE id = %s"
    cursor.execute(query, (book_id,))
    connection.commit()
    connection.close()
    return redirect(url_for('books'))
```

Deletes a book by its ID

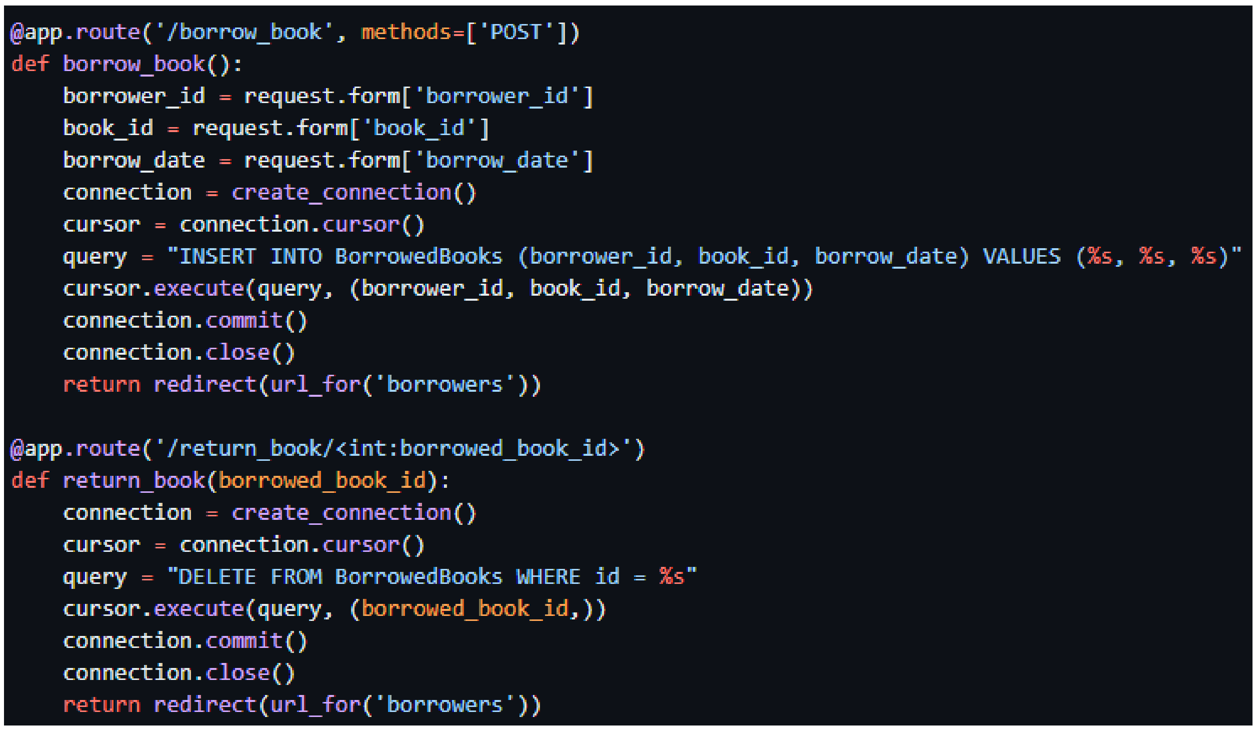


```python
@app.route('/borrowers')
def borrowers():
    connection = create_connection()
    cursor = connection.cursor()

    # Fetch all borrowers
    cursor.execute("SELECT * FROM Borrowers")
    borrowers = cursor.fetchall()

    # Fetch available books to populate the dropdown
    cursor.execute("SELECT id, title FROM Books")
    books = cursor.fetchall()

    connection.close()
    return render_template('borrowers.html', borrowers=borrowers, books=books)

@app.route('/create_borrower', methods=['POST'])
def create_borrower():
    name = request.form['name']
    contact_info = request.form.get('contact_info', '')  # Use get to avoid KeyError
    connection = create_connection()
    cursor = connection.cursor()
    query = "INSERT INTO Borrowers (name, contact_info) VALUES (%s, %s)"
    cursor.execute(query, (name, contact_info))
    connection.commit()
    connection.close()
    return redirect(url_for('borrowers'))

@app.route('/delete_borrower/<int:borrower_id>')
def delete_borrower(borrower_id):
    connection = create_connection()
    cursor = connection.cursor()
    query = "DELETE FROM Borrowers WHERE id = %s"
    cursor.execute(query, (borrower_id,))
    connection.commit()
    connection.close()
    return redirect(url_for('borrowers'))
```

Fetches and displays

Handles the creation

Deletes a borrower by

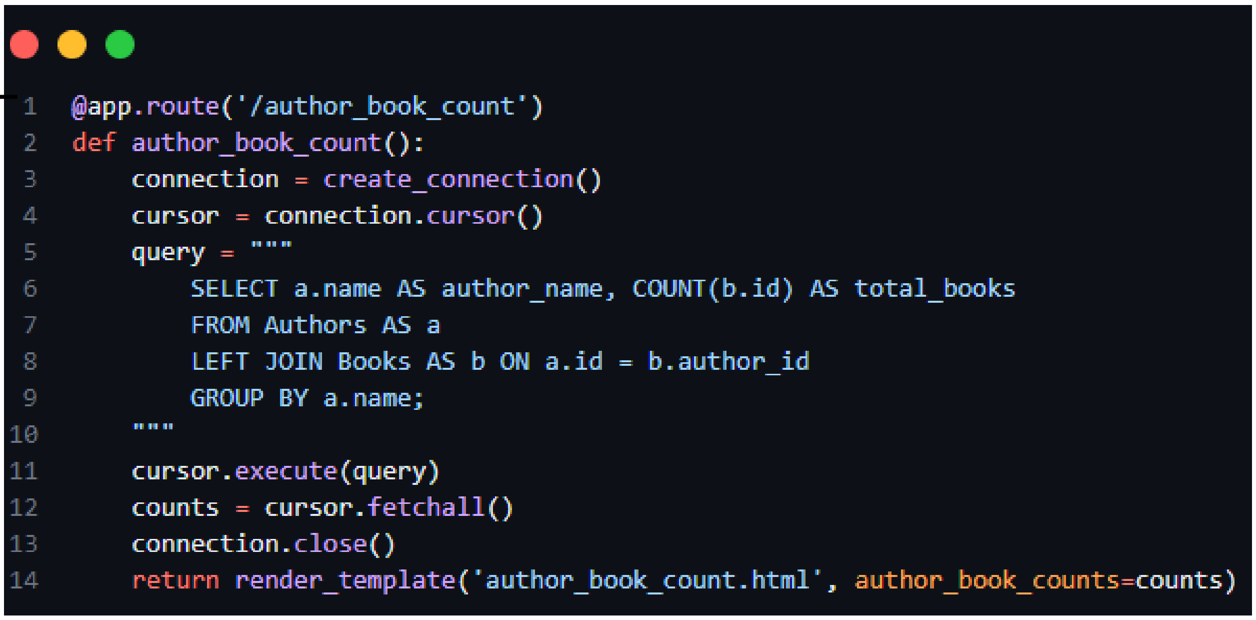


```python
@app.route('/borrow_book', methods=['POST'])
def borrow_book():
    borrower_id = request.form['borrower_id']
    book_id = request.form['book_id']
    borrow_date = request.form['borrow_date']
    connection = create_connection()
    cursor = connection.cursor()
    query = "INSERT INTO BorrowedBooks (borrower_id, book_id, borrow_date) VALUES (%s, %s, %s)"
    cursor.execute(query, (borrower_id, book_id, borrow_date))
    connection.commit()
    connection.close()
    return redirect(url_for('borrowers'))

@app.route('/return_book/<int:borrowed_book_id>')
def return_book(borrowed_book_id):
    connection = create_connection()
    cursor = connection.cursor()
    query = "DELETE FROM BorrowedBooks WHERE id = %s"
    cursor.execute(query, (borrowed_book_id,))
    connection.commit()
    connection.close()
    return redirect(url_for('borrowers'))
```

Records the borrowing of a book by a borrower.

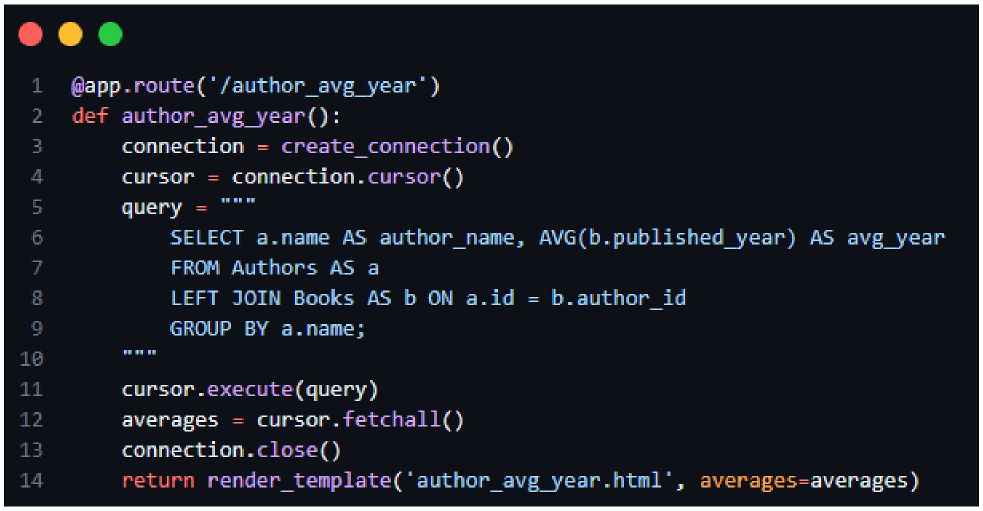
Handles the return of a borrowed book.

ADDITIONAL QUERIES



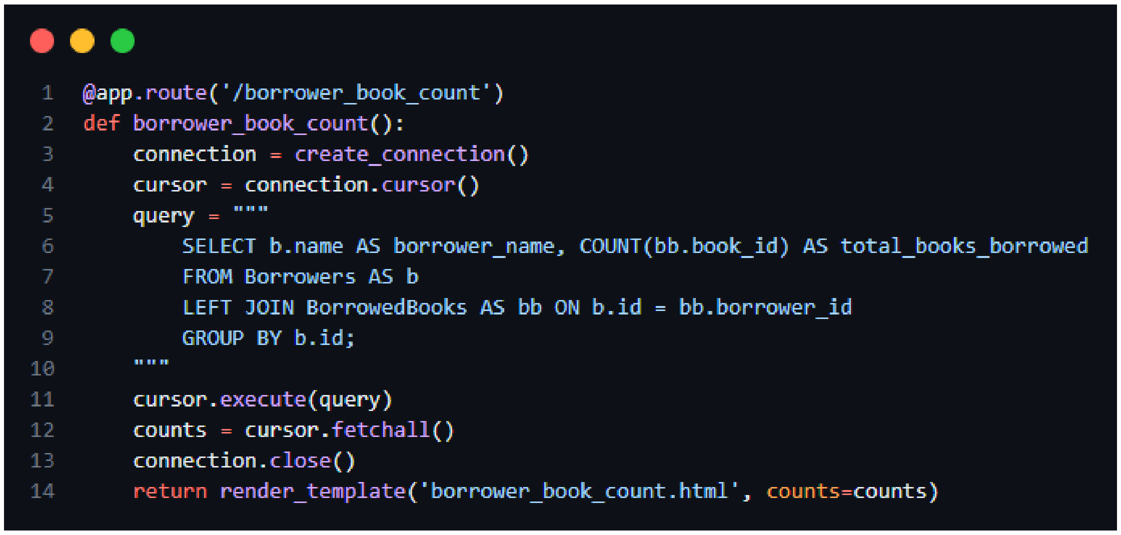
```python
@app.route('/author_book_count')
def author_book_count():
    connection = create_connection()
    cursor = connection.cursor()
    query = """
        SELECT a.name AS author_name, COUNT(b.id) AS total_books
        FROM Authors AS a
        LEFT JOIN Books AS b ON a.id = b.author_id
        GROUP BY a.name;
    """
    cursor.execute(query)
    counts = cursor.fetchall()
    connection.close()
    return render_template('author_book_count.html', author_book_counts=counts)
```

Displays the count of books for each author.



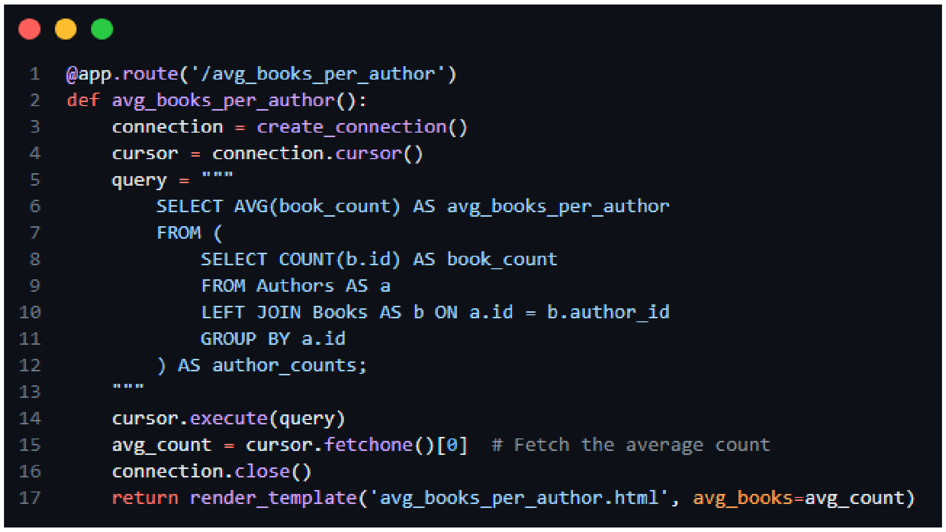
```python
@app.route('/author_avg_year')
def author_avg_year():
    connection = create_connection()
    cursor = connection.cursor()
    query = """
        SELECT a.name AS author_name, AVG(b.published_year) AS avg_year
        FROM Authors AS a
        LEFT JOIN Books AS b ON a.id = b.author_id
        GROUP BY a.name;
    """
    cursor.execute(query)
    averages = cursor.fetchall()
    connection.close()
    return render_template('author_avg_year.html', averages=averages)
```

Calculates the average published year of books for each author.



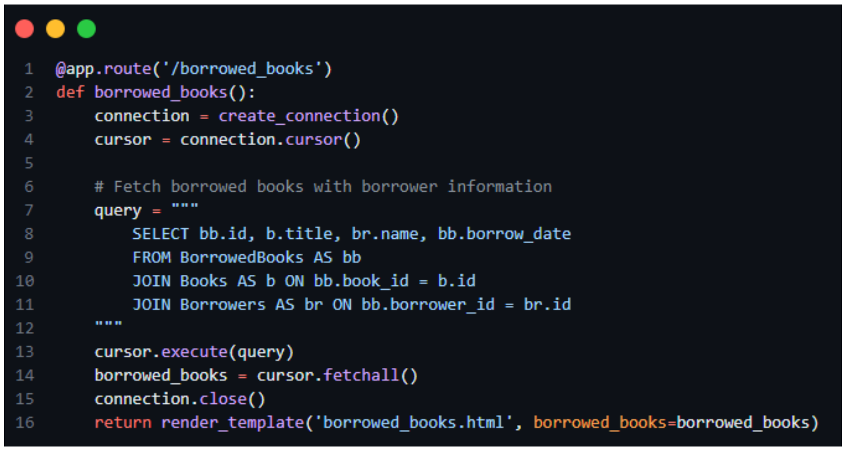
```python
@app.route('/borrower_book_count')
def borrower_book_count():
    connection = create_connection()
    cursor = connection.cursor()
    query = """
        SELECT b.name AS borrower_name, COUNT(bb.book_id) AS total_books_borrowed
        FROM Borrowers AS b
        LEFT JOIN BorrowedBooks AS bb ON b.id = bb.borrower_id
        GROUP BY b.id;
    """
    cursor.execute(query)
    counts = cursor.fetchall()
    connection.close()
    return render_template('borrower_book_count.html', counts=counts)
```

Displays the total number of books borrowed by each borrower.



```python
@app.route('/avg_books_per_author')
def avg_books_per_author():
    connection = create_connection()
    cursor = connection.cursor()
    query = """
        SELECT AVG(book_count) AS avg_books_per_author
        FROM (
            SELECT COUNT(b.id) AS book_count
            FROM Authors AS a
            LEFT JOIN Books AS b ON a.id = b.author_id
            GROUP BY a.id
        ) AS author_counts;
    """
    cursor.execute(query)
    avg_count = cursor.fetchone()[0]  # Fetch the average count
    connection.close()
    return render_template('avg_books_per_author.html', avg_books=avg_count)
```

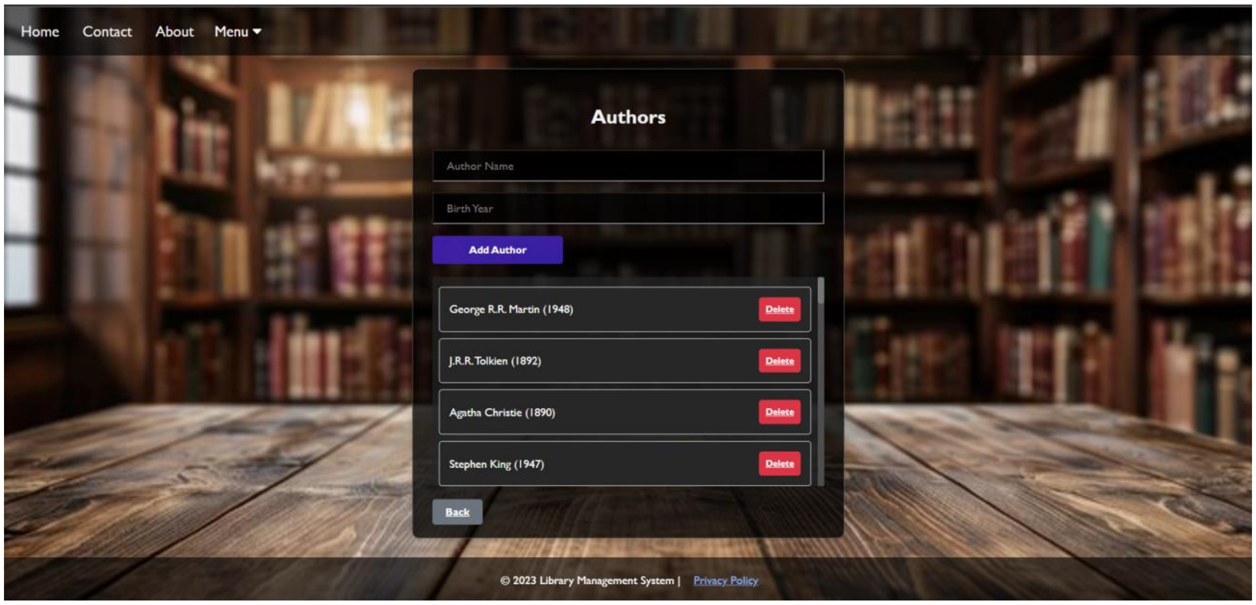
Calculates the average number of books per author.



```python
@app.route('/borrowed_books')
def borrowed_books():
    connection = create_connection()
    cursor = connection.cursor()

    # Fetch borrowed books with borrower information
    query = """
        SELECT bb.id, b.title, br.name, bb.borrow_date
        FROM BorrowedBooks AS bb
        JOIN Books AS b ON bb.book_id = b.id
        JOIN Borrowers AS br ON bb.borrower_id = br.id
    """
    cursor.execute(query)
    borrowed_books = cursor.fetchall()
    connection.close()
    return render_template('borrowed_books.html', borrowed_books=borrowed_books)
```

Displays a list of borrowed books along with borrower information.

INTERFACES



Authors

Author Name

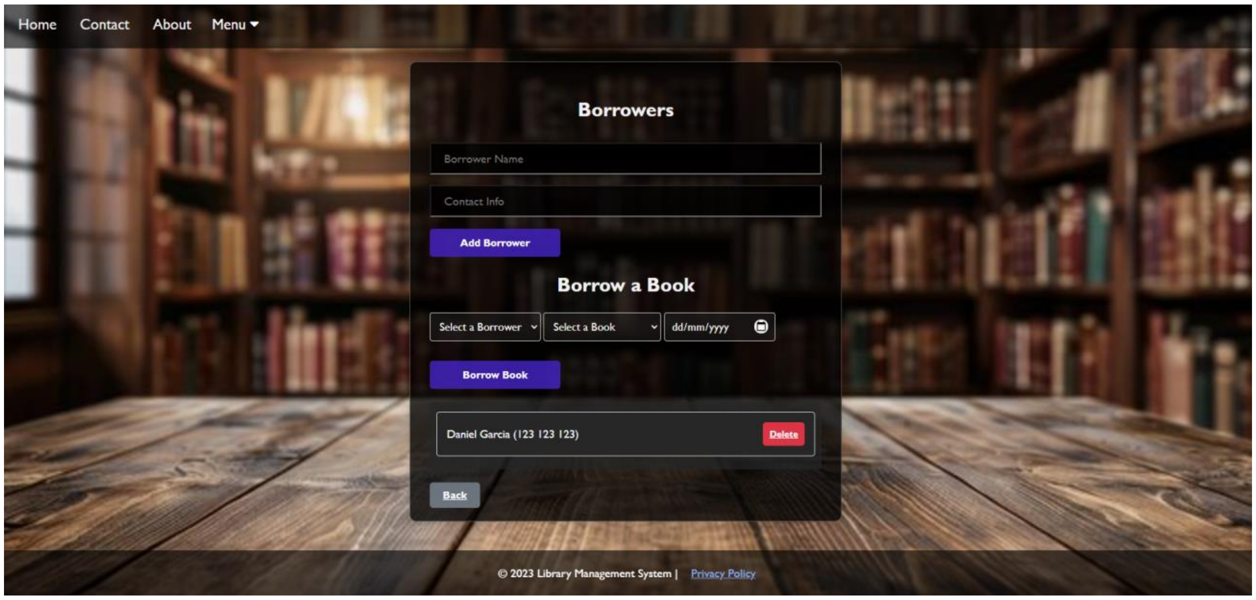
Birth Year

Add Author

- George R.R. Martin (1948) Delete
- J.R.R. Tolkien (1892) Delete
- Agatha Christie (1890) Delete
- Stephen King (1947) Delete

Back

© 2023 Library Management System | Privacy Policy



Borrowers

Borrower Name

Contact Info

Add Borrower

Borrow a Book

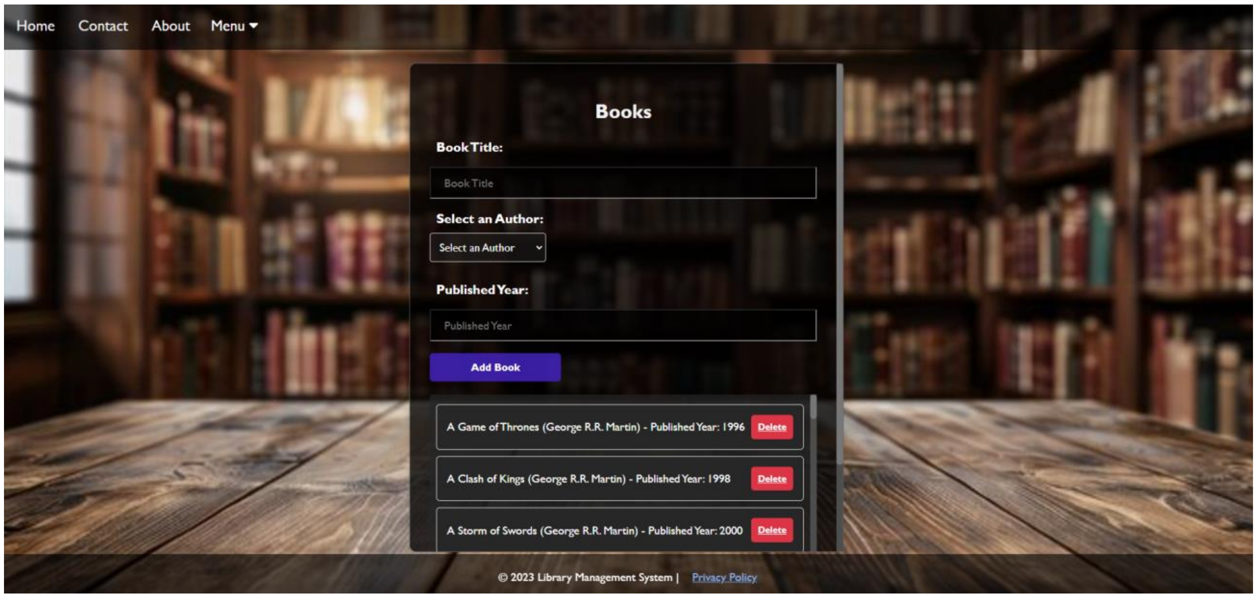
Select a Borrower | Select a Book | dd/mm/yyyy

Borrow Book

- Daniel Garcia (123 123 123) Delete

Back

© 2023 Library Management System | Privacy Policy



Books

Book Title: Book Title

Select an Author: Select an Author

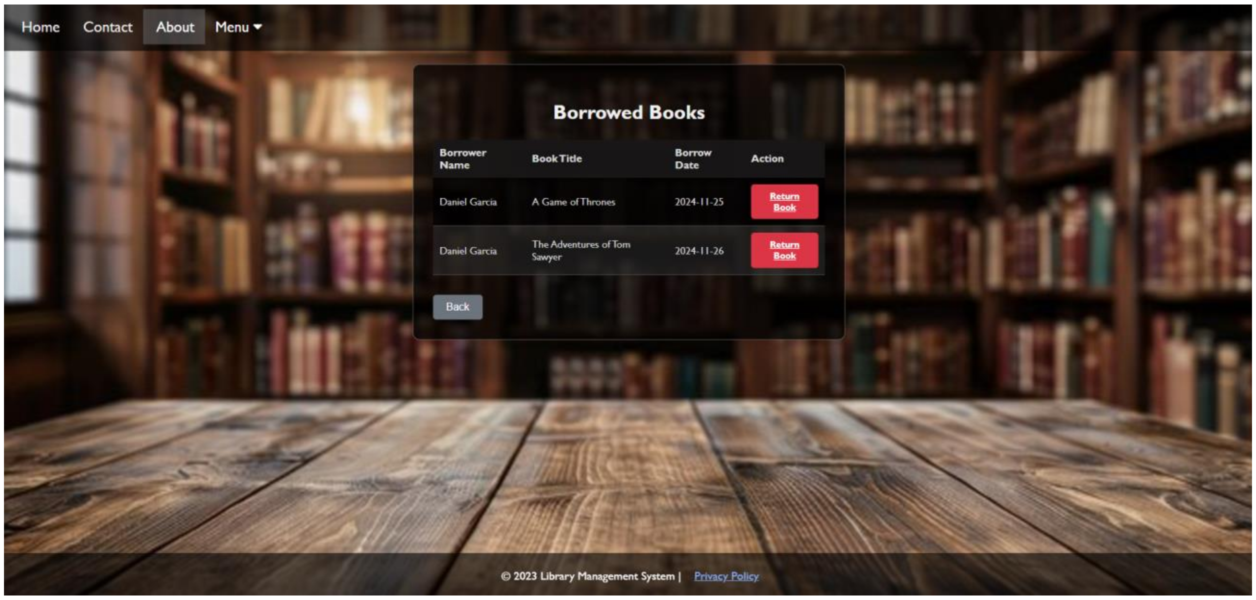
Published Year: Published Year

A Game of Thrones (George R.R. Martin) - Published Year: 1996

A Clash of Kings (George R.R. Martin) - Published Year: 1998

A Storm of Swords (George R.R. Martin) - Published Year: 2000

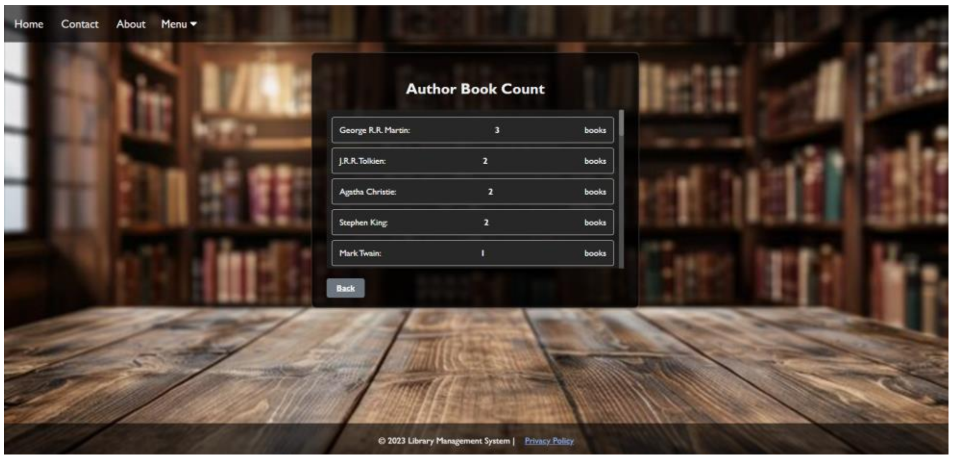
© 2023 Library Management System | Privacy Policy



Borrowed Books

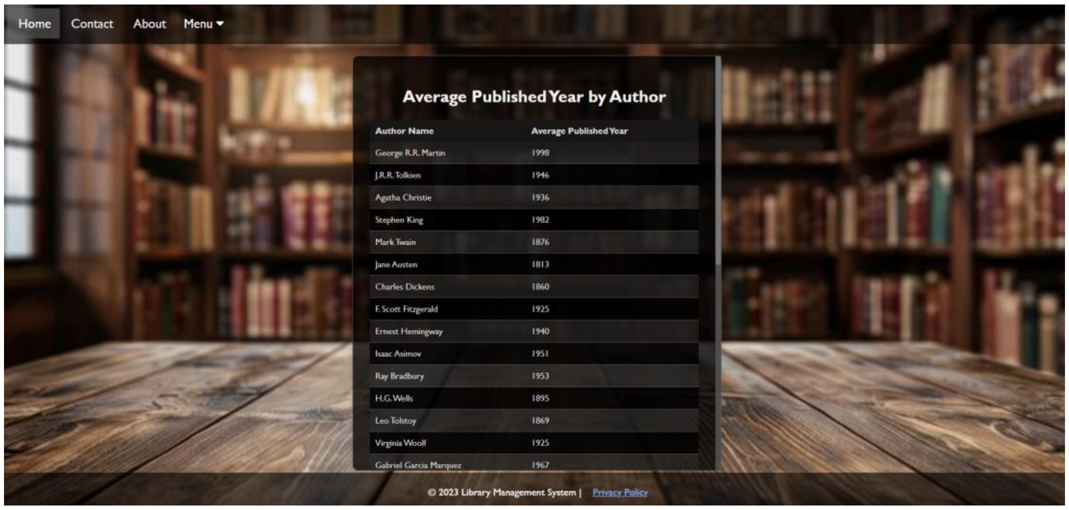
Borrower Name	Book Title	Borrow Date	Action
Daniel Garcia	A Game of Thrones	2024-11-25	Return Book
Daniel Garcia	The Adventures of Tom Sawyer	2024-11-26	Return Book

© 2023 Library Management System | Privacy Policy



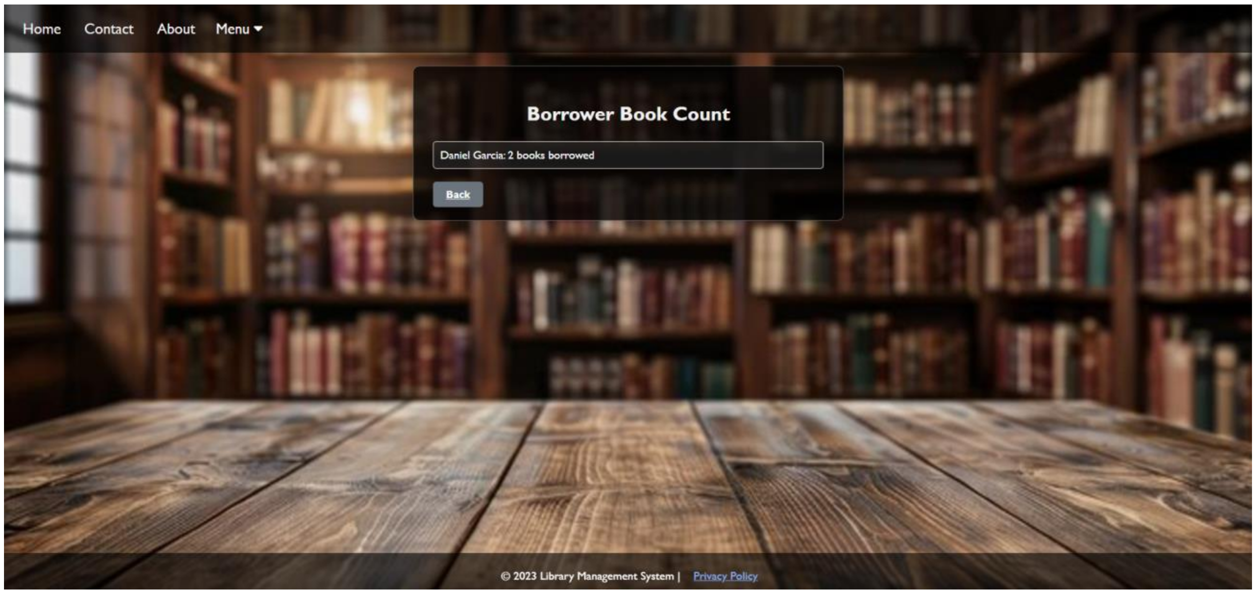
Author Book Count

George R.R. Martin:	3	books
J.R.R. Tolkien:	2	books
Agatha Christie:	2	books
Stephen King:	2	books
Mark Twain:	1	books



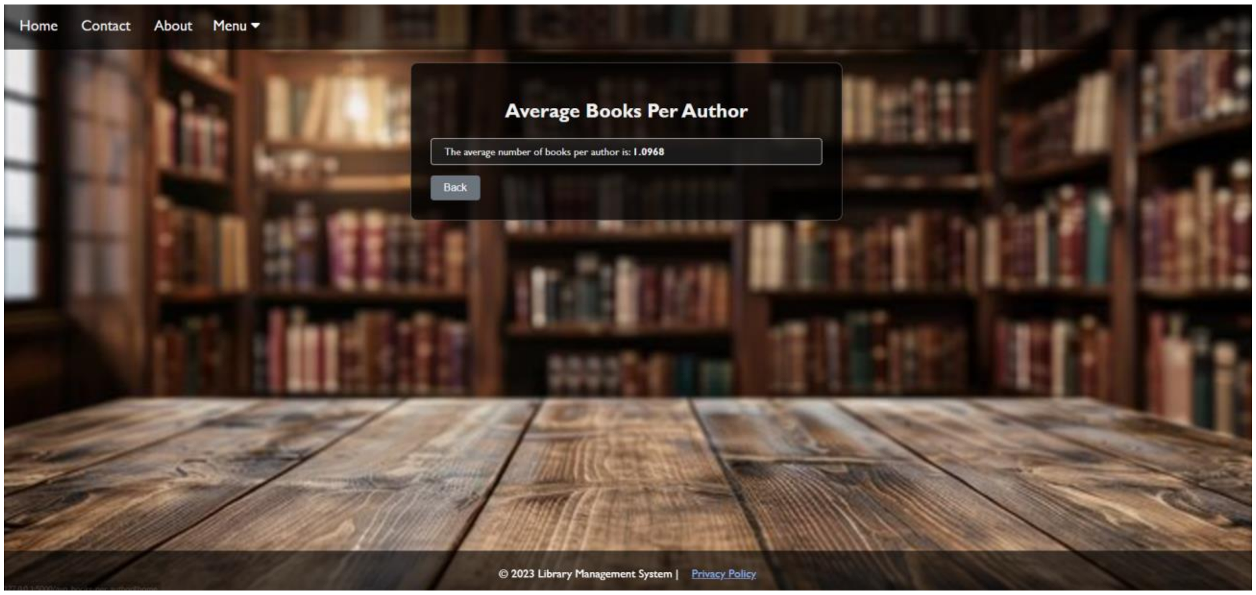
Average Published Year by Author

Author Name	Average Published Year
George R.R. Martin	1998
J.R.R. Tolkien	1946
Agatha Christie	1936
Stephen King	1982
Mark Twain	1876
Jane Austen	1813
Charles Dickens	1860
F. Scott Fitzgerald	1925
Ernest Hemingway	1940
Isaac Asimov	1951
Ray Bradbury	1953
H.G. Wells	1895
Leo Tolstoy	1869
Virginia Woolf	1925
Gabriel Garcia Marquez	1967



Borrower Book Count

Daniel Garcia: 2 books borrowed



Average Books Per Author

The average number of books per author is: **1.0968**