Index.js  
  
function deleteNote(noteId) {

    fetch("/delete-note", {

      method: "POST",

      body: JSON.stringify({ noteId: noteId }),

    }).then((\_res) => {

      window.location.href = "/";

    });

  }

Base.html  
  
<!DOCTYPE html>

<html>

  <head>

    <meta charset="utf-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <link

      rel="stylesheet"

      href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"

      integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"

      crossorigin="anonymous"

    />

    <link

      rel="stylesheet"

      href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css"

      crossorigin="anonymous"

    />

    <title>{% block title %}Home{% endblock %}</title>

  </head>

  <body>

    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">

      <button

        class="navbar-toggler"

        type="button"

        data-toggle="collapse"

        data-target="#navbar"

      >

        <span class="navbar-toggler-icon"></span>

      </button>

      <div class="collapse navbar-collapse" id="navbar">

        <div class="navbar-nav">

          <img src="/static/bookshelf.png" height="40" style="margin-top: 5px" />

          {% if user.is\_authenticated %}

          <a

            class="nav-item nav-link"

            id="home"

            href="/"

            style="font-size: 20px"

            >Home</a

          >

          <a class="nav-item nav-link" id="logout" href="/logout"  style="font-size: 20px">Logout</a>

          {% else %}

          <a class="nav-item nav-link" id="login" href="/login"  style="font-size: 20px">Login</a>

          <a class="nav-item nav-link" id="signUp" href="/sign-up"  style="font-size: 20px">Sign Up</a>

          {% endif %}

        </div>

      </div>

    </nav>

    {% with messages = get\_flashed\_messages(with\_categories=true) %} {% if

    messages %} {% for category, message in messages %} {% if category ==

    'error' %}

    <div class="alert alert-danger alter-dismissable fade show" role="alert">

      {{ message }}

      <button type="button" class="close" data-dismiss="alert">

        <span aria-hidden="true">&times;</span>

      </button>

    </div>

    {% else %}

    <div class="alert alert-success alter-dismissable fade show" role="alert">

      {{ message }}

      <button type="button" class="close" data-dismiss="alert">

        <span aria-hidden="true">&times;</span>

      </button>

    </div>

    {% endif %} {% endfor %} {% endif %} {% endwith %}

    <div class="container">{% block content %} {% endblock %}</div>

    <script

      src="https://code.jquery.com/jquery-3.2.1.slim.min.js"

      integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"

      crossorigin="anonymous"

    ></script>

    <script

      src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"

      integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"

      crossorigin="anonymous"

    ></script>

    <script

      src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"

      integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"

      crossorigin="anonymous"

    ></script>

    {% block javascript %}

    <script type="text/javascript">

      function deleteNote(noteId) {

        fetch("/delete-note", {

          method: "POST",

          body: JSON.stringify({ noteId: noteId }),

        }).then((\_res) => {

          window.location.href = "/";

        });

      }

    </script>

    {% endblock %}

  </body>

</html>

Home.html

{% extends "base.html" %}

{% block title %}Home{% endblock %}

{% block content %}

<h1 align="center">Book Library</h1>

<img src="/static/bookshelves.png" style="margin: 8px;" />

<!-- Search Form -->

<form method="GET" action="{{ url\_for('views.home') }}" class="mb-4">

    <div class="form-row">

        <div class="col">

            <input type="text" class="form-control" name="query" placeholder="Search by title, author, or ISBN" value="{{ request.args.get('query', '') }}">

        </div>

        <div class="col-auto">

            <button type="submit" class="btn btn-primary">Search</button>

        </div>

    </div>

</form>

<!-- Display Books -->

<div class="table-responsive">

    <table class="table table-striped">

        <thead>

            <tr>

                <th>Title</th>

                <th>Author</th>

                <th>ISBN</th>

                <th>Status</th>

                <th>Action</th>

            </tr>

        </thead>

        <tbody>

            {% for book in books %}

            <tr>

                <td>{{ book.title }}</td>

                <td>{{ book.author }}</td>

                <td>{{ book.isbn }}</td>

                <td>

                    {% if book.available %}

                        <span class="text-success">Available</span>

                    {% else %}

                        <span class="text-danger">Checked Out</span>

                    {% endif %}

                </td>

                <td>

                    {% if book.available %}

                        <form method="POST" action="{{ url\_for('views.checkout\_book', bookId=book.id) }}" style="display:inline;">

                            <button type="submit" class="btn btn-sm btn-success">Check Out</button>

                        </form>

                    {% else %}

                    <form method="POST" action="{{ url\_for('views.checkin\_book', bookId=book.id) }}" style="display:inline;">

                        <button type="submit" class="btn btn-sm btn-danger">Return</button>

                    </form>

                    {% endif %}

                </td>

            </tr>

            {% endfor %}

        </tbody>

    </table>

</div>

{% endblock %}

Login.html

{% extends "base.html" %} {% block title %}Login{% endblock %} {% block content

    %}

    <form method="POST">

      <h3 align="center">Login</h3>

      <div class="form-group">

        <label for="email">Email Address</label>

        <input

          type="email"

          class="form-control"

          id="email"

          name="email"

          placeholder="Enter email"

        />

      </div>

      <div class="form-group">

        <label for="password">Password</label>

        <input

          type="password"

          class="form-control"

          id="password"

          name="password"

          placeholder="Enter password"

        />

      </div>

      <br />

      <button type="submit" class="btn btn-primary">Login</button>

    </form>

    {% endblock %}

sign\_up.html {% extends "base.html" %} {% block title %}Sign Up{% endblock %} {% block

    content %}

    <form method="POST">

      <h3 align="center">Sign Up</h3>

      <div class="form-group">

        <label for="email">Email Address</label>

        <input

          type="email"

          class="form-control"

          id="email"

          name="email"

          placeholder="Enter email"

        />

      </div>

      <div class="form-group">

        <label for="firstName">First Name</label>

        <input

          type="text"

          class="form-control"

          id="firstName"

          name="firstName"

          placeholder="Enter first name"

        />

      </div>

      <div class="form-group">

        <label for="password1">Password</label>

        <input

          type="password"

          class="form-control"

          id="password1"

          name="password1"

          placeholder="Enter password"

        />

      </div>

      <div class="form-group">

        <label for="password2">Password (Confirm)</label>

        <input

          type="password"

          class="form-control"

          id="password2"

          name="password2"

          placeholder="Confirm password"

        />

      </div>

      <br />

      <button type="submit" class="btn btn-primary">Submit</button>

    </form>

    {% endblock %}

\_\_init\_\_.py

from flask import Flask

from flask\_sqlalchemy import SQLAlchemy

from flask\_login import LoginManager

db = SQLAlchemy()

def create\_app():

    app = Flask(\_\_name\_\_)

    app.config['SECRET\_KEY'] = 'your\_secret\_key'

    app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///library.db'

    db.init\_app(app)

    from .views import views

    from .auth import auth

    app.register\_blueprint(views, url\_prefix='/')

    app.register\_blueprint(auth, url\_prefix='/')

    from .models import User, Book  # Import the Book model

    with app.app\_context():

        db.create\_all()

    login\_manager = LoginManager()

    login\_manager.login\_view = 'auth.login'

    login\_manager.init\_app(app)

    @login\_manager.user\_loader

    def load\_user(id):

        return User.query.get(int(id))

    return app

Auth.py from flask import Blueprint, render\_template, request, flash, redirect, url\_for

from .models import User

from werkzeug.security import generate\_password\_hash, check\_password\_hash

from . import db

from flask\_login import login\_user, login\_required, logout\_user, current\_user

auth = Blueprint('auth', \_\_name\_\_)

@auth.route('/login', methods=['GET', 'POST'])

def login():

    if request.method == 'POST':

        email = request.form.get('email')

        password = request.form.get('password')

        user = User.query.filter\_by(email=email).first()

        if user:

            if check\_password\_hash(user.password, password):

                flash('Logged in successfully!', category='success')

                login\_user(user, remember=True)

                return redirect(url\_for('views.home'))

            else:

                flash('Incorrect password, try again.', category='error')

        else:

            flash('Email does not exist.', category='error')

    return render\_template("login.html", user=current\_user)

@auth.route('/logout')

@login\_required

def logout():

    logout\_user()

    return redirect(url\_for('auth.login'))

@auth.route('/sign-up', methods=['GET', 'POST'])

def sign\_up():

    if request.method == 'POST':

        email = request.form.get('email')

        first\_name = request.form.get('firstName')

        password1 = request.form.get('password1')

        password2 = request.form.get('password2')

        user = User.query.filter\_by(email=email).first()

        if user:

            flash('Email already exists.', category='error')

        elif len(email) < 4:

            flash('Email must be greater than 3 characters.', category='error')

        elif len(first\_name) < 2:

            flash('First name must be greater than 1 character.', category='error')

        elif password1 != password2:

            flash('Passwords don\'t match.', category='error')

        elif len(password1) < 7:

            flash('Password must be at least 7 characters.', category='error')

        else:

            new\_user = User(email=email, first\_name=first\_name, password=generate\_password\_hash(

                password1))

            db.session.add(new\_user)

            db.session.commit()

            login\_user(new\_user, remember=True)

            flash('Account created!', category='success')

            return redirect(url\_for('views.home'))

    return render\_template("sign\_up.html", user=current\_user)

Book\_managing.py   
  
import sqlite3

from search import search\_book

#add a book to the database

def add\_book(title, author, isbn):

    conn = sqlite3.connect("library.db")

    cursor = conn.cursor()

    #inserting the book into the db

    try:

        cursor.execute("INSERT INTO books (title, author, isbn) VALUES (?, ?, ?)",

                       (title, author, isbn))

        conn.commit()

        print("Book added successfully!")

    except sqlite3.IntegrityError:

        print("Error: ISBN already exists.")

    conn.close()

#updated delete to use search book and confirmations

def delete\_book(query, field="title"):

    book\_id = search\_book(query, field)

    if not book\_id:

        print("No book found to delete.")

        return

    confirmation = input(f"Are you sure you want to delete book ID {book\_id}? (yes/no): ").strip().lower()

    if confirmation != "yes":

        print("Deletion canceled.")

        return

    conn = sqlite3.connect("library.db")

    cursor = conn.cursor()

    cursor.execute("DELETE FROM books WHERE id = ?", (book\_id,))

    affected\_rows = cursor.rowcount

    if affected\_rows > 0:

        print("Book successfully deleted.")

    else:

        print("No book found.")

    conn.commit()

    conn.close()

#updating book details

def update\_book(query, field="title"):

    book\_id = search\_book(query, field)

    if not book\_id:

        print("No book found to update.")

        return

    confirmation = input(f"Are you sure you want to update book ID {book\_id}? (yes/no): ").strip().lower()

    if confirmation != "yes":

        print("Update canceled.")

        return

    title = input("Enter new title (leave blank to keep unchanged): ").strip()

    author = input("Enter new author (leave blank to keep unchanged): ").strip()

    isbn = input("Enter new ISBN (leave blank to keep unchanged): ").strip()

    conn = sqlite3.connect("library.db")

    cursor = conn.cursor()

    updates = []

    values = []

    #no idea but it works

    if title:

        updates.append("title = ?")

        values.append(title)

    if author:

        updates.append("author = ?")

        values.append(author)

    if isbn:

        updates.append("isbn = ?")

        values.append(isbn)

    if updates:

        values.append(book\_id)

        query = f"UPDATE books SET {', '.join(updates)} WHERE id = ?"

        cursor.execute(query, values)

        conn.commit()

        print("Book updated successfully.")

    else:

        print("No changes provided.")

    conn.close()

Book.py

# Defines what a Book is and its attributes

class Book:

    def \_\_init\_\_(self,title,author,isbn,availability):

        self.title = title

        self.author = author

        self.isbn = isbn

        self.availability = availability

Checkout.py

# Check in / Check out functions

import sqlite3

def check\_out\_book(isbn):

    """

    Marks a book as checked out (sets available to 0) if it is currently available.

    """

    conn = sqlite3.connect("library.db")

    cursor = conn.cursor()

    # Check if the book exists and get its current availability

    cursor.execute("SELECT available FROM books WHERE isbn = ?", (isbn))

    result = cursor.fetchone()

    if result is None:

        print("Error: Book not found.")

    elif result[0] == 0:

        print("This book is already checked out.")

    else:

        cursor.execute("UPDATE books SET available = 0 WHERE isbn = ?", (isbn))

        conn.commit()

        print("Book checked out successfully!")

    conn.close()

def check\_in\_book(isbn):

    """

    Marks a book as checked in (sets available to 1) if it is currently checked out.

    """

    conn = sqlite3.connect("library.db")

    cursor = conn.cursor()

    # Check if the book exists and get its current availaility

    cursor.execute("SELECT available FROM books WHERE isbn = ?", (isbn))

    result = cursor.fetchone()

    if result is None:

        print("Error: Book not found.")

    elif result[0] == 1:

        print("This book is already checked in.")

    else:

        cursor.execute("UPDATE books SET available = 1 WHERE isbn = ?", (isbn))

        conn.commit()

        print("Book checked in successfully!")

    conn.close()

db.py

#initialize db funciton creates db if one doesnt exist

import sqlite3

def initialize\_database():

    conn = sqlite3.connect("library.db") #other functions connect to library.db

    cursor = conn.cursor()

    cursor.execute("""

    CREATE TABLE IF NOT EXISTS books (

        id INTEGER PRIMARY KEY AUTOINCREMENT,

        title TEXT NOT NULL,

        author TEXT NOT NULL,

        isbn TEXT UNIQUE NOT NULL,

        available INTEGER DEFAULT 1

    )

    """)

    conn.commit()

    conn.close()

#inserting random books to populate the db

def insert\_books():

    books = [

        ("1984", "George Orwell", "9780451524935", 1),

        ("To Kill a Mockingbird", "Harper Lee", "9780061120084", 1),

        ("The Great Gatsby", "F. Scott Fitzgerald", "9780743273565", 1),

        ("Moby-Dick", "Herman Melville", "9781503280786", 1),

        ("Pride and Prejudice", "Jane Austen", "9781503290563", 1),

        ("War and Peace", "Leo Tolstoy", "9781400079988", 1),

        ("Ulysses", "James Joyce", "9781840226355", 1),

        ("The Catcher in the Rye", "J.D. Salinger", "9780316769488", 1),

        ("Brave New World", "Aldous Huxley", "9780060850524", 1),

        ("The Hobbit", "J.R.R. Tolkien", "9780547928227", 1)

    ]

    conn = sqlite3.connect("library.db")

    cursor = conn.cursor()

    for book in books:

        try:

            cursor.execute("""

            INSERT INTO books (title, author, isbn, available)

            VALUES (?, ?, ?, ?)

            """, book)

        except sqlite3.IntegrityError:

            pass

    conn.commit()

    conn.close()

models.py

from . import db

from flask\_login import UserMixin

from sqlalchemy.sql import func

class User(db.Model, UserMixin):

    id = db.Column(db.Integer, primary\_key=True)

    email = db.Column(db.String(150), unique=True)

    password = db.Column(db.String(150))

    first\_name = db.Column(db.String(150))

class Book(db.Model):

    id = db.Column(db.Integer, primary\_key=True)

    title = db.Column(db.String(150), nullable=False)

    author = db.Column(db.String(150), nullable=False)

    isbn = db.Column(db.String(20), unique=True, nullable=False)

    available = db.Column(db.Integer, default=1)  # 1 = Available, 0 = Checked Out

Search.py

import sqlite3

import sqlite3

def search\_book(query, field="title"):

    """

    Search for books in the database by a given field.

    Parameters:

    - query: The search term.

    - field: The database column to search (e.g., "title", "author", "isbn").

      Defaults to "title".

    Returns:

    - The book ID of the first matching result, or None if no match is found.

    """

    allowed\_fields = ["title", "author", "isbn"]

    if field not in allowed\_fields:

        print(f"Invalid search field. Choose one of: {allowed\_fields}")

        return None

    conn = sqlite3.connect("library.db")

    cursor = conn.cursor()

    sql\_query = f"SELECT id, title, author, isbn, available FROM books WHERE {field} LIKE ?"

    cursor.execute(sql\_query, ('%' + query + '%',))

    result = cursor.fetchone()

    if result:

        book\_id, title, author, isbn, available = result

        status = "Available" if available == 1 else "Checked Out"

        print(f"ID: {book\_id} | Title: {title} | Author: {author} | ISBN: {isbn} | Status: {status}")

        conn.close()

        return book\_id

    else:

        print("No books found matching your query.")

        conn.close()

        return None

Views.py

from flask import Blueprint, render\_template, request

from flask\_login import login\_required, current\_user

from .models import Book  # Import the Book model

from . import db

views = Blueprint('views', \_\_name\_\_)

@views.route('/', methods=['GET', 'POST'])

@login\_required

def home():

    print ("here")

    query = request.args.get('query', '').strip()

    print (query)

    # Fetch books based on search query

    if query:

        books = Book.query.filter(

            Book.title.contains(query) |

            Book.author.contains(query) |

            Book.isbn.contains(query)

        ).all()

    else:

        books = Book.query.all()

    print(books)

    return render\_template("home.html", user=current\_user, books=books)

@views.route('/checkout/<bookId>', methods=['GET', 'POST'])

@login\_required

def checkout\_book(bookId):

    # temporarily just show book selected for checkout -- remove once available property is updated correctly

    #books = Book.query.filter(Book.id == bookId)

    # get book

    books = Book.query.get(bookId)

    # update 'available' property

    books.available = 0

    db.session.commit()

    # return updated list of books

    books = Book.query.all()

    return render\_template("home.html", user=current\_user, books=books)

@views.route('/checkin/<bookId>', methods=['GET', 'POST'])

@login\_required

def checkin\_book(bookId):

    # temporarily just show book selected for checkout -- remove once available property is updated correctly

    #books = Book.query.filter(Book.id == bookId)

    # get book

    books = Book.query.get(bookId)

    # update 'available' property

    books.available = 1

    db.session.commit()

    # return updated list of books

    books = Book.query.all()

    return render\_template("home.html", user=current\_user, books=books)

Main.py

from website import create\_app

app = create\_app()

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)