Endpoints:

/users (GET, POST, PUT, DELETE)

GET: Retrieves a list of all users in the system
POST: Creates a new user in the system
PUT: Updates an existing user's information
DELETE: Deletes an existing user from the system

/books (GET, POST, PUT, DELETE)

GET: Retrieves a list of all books in the system
POST: Creates a new book in the system
PUT: Updates an existing book's information
DELETE: Deletes an existing book from the system

/books/{book_id}/images (POST)

POST: Uploads images of a book to Azure blob

/transactions (GET, POST)

GET: Retrieves a list of all transactions in the system
POST: Creates a new transaction in the system

/transactions/{transaction_id} (PUT)

PUT: Updates an existing transaction's information, such as marking a transaction as completed or canceled

/courses (GET)

GET: Retrieves a list of all courses associated with books in the system. This endpoint could be useful for filtering books by course

Properties:

USERS

id: unique identifier for each user
username: username chosen by the user
first_name: first name of the user
last_name: last name of the user
email_address: email address of the user
profile_picture: user's profile picture

BOOKS

id: unique identifier for each book
author: author of the book
title: title of the book
edition: edition of the book
description: brief description of the book
isbn: International Standard Book Number of the book
course_id: identifier of the course associated with the book
seller_id: identifier of the user who is selling the book
condition: condition of the book (new, used, etc.)

TRANSACTIONS

id: unique identifier for each transaction
book_id: identifier of the book involved in the transaction
buyer_id: identifier of the user who is buying the book
interested_patrons: comma-separated list of identifiers of users who have expressed interest in buying the book
winning_patron_id: identifier of the user who won the auction or was the first to purchase the book
transaction_date: date and time when the transaction occurred
transaction_amount: amount paid for the book in the transaction

COURSES

id: unique identifier for each course
name: code and name of the course

<u>Inputs:</u>

User

username (string)
first_name (string)
last_name (string)
email (string)
profile_picture (string)

Book

author (string)
title (string)
edition (string)
description (string)
isbn (string)
condition (string)
seller_id (int)
course_id (int)
price (double)

Image

book_id (int)
image (string)

Transaction

book_id (int)
buyer_id (int)
price (double)

Course

course_name (string)

<u>Output:</u>

User

user_id (int)
username (string)
first_name (string)
last_name (string)
email (string)
profile_picture (string) **??**

Book

book_id (int)
author (string)
title (string)
edition (string)
description (string)
isbn (string)
condition (string)
seller_id (int)
course_id (int)
price (float)
image (string)

Transaction

transaction_id (int)
book_id (int)
buyer_id (int)
seller_id (int)
price (float)
status (string)
timestamp (datetime)

Course

course_id (int)
course_name (string)

<u>Response/Status Codes:</u>

200 OK: The server has successfully fulfilled the request and the client can expect a response

400 Bad Request: The server cannot or will not process the request due to a client error, such as malformed syntax or invalid data

401 Unauthorized: The client must authenticate itself to get the requested response

403 Forbidden: The client does not have access rights to the content, usually because of a lack of credentials or permissions

404 Not Found: The server cannot find the requested resource

500 Internal Server Error: A generic error message indicating that the server encountered an unexpected condition that prevented it from fulfilling the request

Mock API Server: