Book Store API Documentation

This API enables users to perform **CRUD operations** (Create, Read, Update, Delete) on a collection of books stored in a PostgreSQL database using a Flask-based backend.

Base URL (for local development)

arduino
CopyEdit
`http://localhost:5000`

Endpoints

---

1. Get All Books

Endpoint: `/books`
Method: `GET`
Description: Retrieve all books from the database.

Sample Response (200 OK):

json
CopyEdit

```
[
  {
    "id": 1,
    "title": "The Alchemist",
    "author": "Paulo Coelho",
    "price": 399.99
  },
  {
    "id": 2,
    "title": "Sapiens",
    "author": "Yuval Noah Harari",
```

```
    "price": 499.00
  }
]
```

---

2. Get Book by ID

Endpoint: `/books/<id>`
Method: `GET`
Description: Retrieve a single book by its ID.

Sample Request:
`GET /books/1`

Sample Response (200 OK):

json
CopyEdit
```
{
  "id": 1,
  "title": "The Alchemist",
  "author": "Paulo Coelho",
  "price": 399.99
}
```

Error Response (404 Not Found):

json
CopyEdit
```
{
  "error": "Book not found"
}
```

---

3. Add a New Book

Endpoint: `/books`
Method: `POST`
Description: Add a new book to the database.

Request Body (JSON):

json
CopyEdit

```
{
  "title": "Clean Code",
  "author": "Robert C. Martin",
  "price": 599.00
}
```

Note: The `id` is auto-generated unless handled manually.

Response (201 Created):

json
CopyEdit

```
{
  "message": "Book added"
}
```

Error Response (400 Bad Request):

json
CopyEdit

```
{
  "error": "Missing field: title"
}
```

---

4. Update an Existing Book (with Optional ID Change)

Endpoint: `/books/<id>`
Method: `PUT`
Description: Update an existing book's details. Optionally, the book's `id` can also be changed if the new ID does not already exist.

Request Body (JSON):

json
CopyEdit

```
{
  "id": 10,
  "title": "Updated Title",
  "author": "Updated Author",
  "price": 699.99
}
```

Success Response (200 OK):

json
CopyEdit

```
{
  "message": "Book updated successfully",
  "book": {
    "id": 10,
    "title": "Updated Title",
    "author": "Updated Author",
    "price": 699.99
  }
}
```

Error Responses:

- `404 Not Found`: Book with the given ID does not exist.
- `400 Bad Request`: Required fields are missing.
- `409 Conflict`: The new ID already exists in the database.

---

5. Delete a Book

Endpoint: `/books/<id>`
Method: `DELETE`
Description: Delete a book by its ID.

Sample Request:

```
DELETE /books/1
```

Response (200 OK):

json
CopyEdit

```
{
  "message": "Book deleted"
}
```

Error Response (404 Not Found):

json
CopyEdit

```
{
  "error": "Book not found"
}
```

API Endpoint URLs

Action	Method	URL	Request Body
Get all books	`GET`	`http://127.0.0.1:5000/books`	—
Get book by ID	`GET`	`http://127.0.0.1:5000/books/<id>`	—
Add new book	`POST`	`http://127.0.0.1:5000/books`	JSON (see examples below)
Update book by ID	`PUT`	`http://127.0.0.1:5000/books/<id>`	JSON (updated fields)
Delete book by ID	`DELETE`	`http://127.0.0.1:5000/books/<id>`	—

Replace `<id>` with the actual book ID, e.g., `/books/1`.

Technologies Used

- Python (Flask)
- PostgreSQL
- SQLAlchemy (ORM)
- HTML/CSS/JavaScript (Frontend)