Book Catalog API Documentation

Samar Noor Riaz

Introduction

The Book Catalog API is a lightweight, RESTful API designed to manage a catalog of books efficiently. Built with modern Python technologies, including FastAPI, SQLAlchemy, and Pydantic, this API provides a robust solution for creating, retrieving, updating, and deleting book records.

Features

The Book Catalog API offers the following features to support book management:

- **CRUD Operations**: Create, read, update, and delete book records, enabling full management of book data in a database.
- **Pydantic Validation**: Ensures data integrity by validating fields like `published_year` to be within a realistic range (e.g., 0 to the current year), preventing invalid data entries.
- **Asynchronous Support**: Leverages FastAPIs async capabilities to handle multiple requests concurrently, improving performance for high-traffic applications.
- **Integration and Unit Tests**: Includes comprehensive tests in the `tests/` folder to verify both business logic (`test_crud.py`) and API endpoints (`test_routes.py`), ensuring reliability.
- **Auto-generated Swagger UI**: Provides an interactive interface at `http://127.0.0.1:8000/docs` for testing and exploring API endpoints.
- **SQLite Database**: Uses a lightweight, serverless database for quick setup, with support for switching to other databases.

Tech Stack

The API is built using the following technologies:

- **FastAPI**: A high-performance web framework for Python, known for its speed and automatic OpenAPI documentation.
- **SQLAlchemy**: An ORM tool that simplifies database interactions and supports multiple database backends.
- **SQLite**: A lightweight, file-based database ideal for development and testing.
- **Pydantic**: A data validation and serialization library for robust input validation.

- **Pytest**: A testing framework for running unit and integration tests in the `tests/` folder.

Prerequisites

To set up and run the Book Catalog API, ensure you have:

- **Python 3.8 or higher**: Required for compatibility with FastAPI and dependencies.
- **Git**: For cloning the repository from GitHub.
- **(Optional) PostgreSQL or MySQL**: For production-grade database setups.
- A terminal or command-line interface for executing commands.

Project Structure

The project is organized for modularity and maintainability:

```
book-catalog/
|
 app/
|    main.py          # Entry point for the FastAPI application
|    models/         # SQLAlchemy models for database schema
|    schemas/        # Pydantic schemas for data validation
|    crud/           # Business logic for CRUD operations
|    database/       # Database connection and setup
|
 tests/
|    test_crud.py    # Unit tests for business logic
|    test_routes.py  # Integration tests for API endpoints
|
 requirements.txt    # Project dependencies
 README.md           # Project documentation
```

Setup Instructions

Follow these steps to set up and run the Book Catalog API locally:

1. **Clone the repository**: Download the project source code from GitHub.

   ```
   git clone https://github.com/Samar-Riaz/book-catalog.git
   cd book-catalog
   ```

   This clones the repository and navigates into the project directory.

2. **Create a virtual environment**: Isolate dependencies to avoid conflicts.

   ```
   python -m venv venv
   source venv/bin/activate  # On Windows: venv\Scripts\activate
   ```

The terminal prompt should indicate the virtual environment is active.

3. **Install dependencies**: Install packages listed in `requirements.txt`, which includes specific versions for reproducibility.

```
pip install -r requirements.txt
```

This installs FastAPI, SQLAlchemy, Pydantic, Pytest, and other dependencies.

4. **Run the application**: Start the FastAPI server with Uvicorn.

```
uvicorn app.main:app --reload
```

The `-reload` flag enables auto-reload for development. Visit http://127.0.0.1:8000/docs to access the Swagger UI.

Running Tests

The `tests/` folder contains a test suite with unit and integration tests. Ensure `pytest` is installed via `pip install -r requirements.txt` before running tests:

- **Unit Tests**: Test the business logic in the `crud/` module.

```
pytest tests/test_crud.py
```

- **Integration Tests**: Verify the API endpoints.

```
pytest tests/test_routes.py
```

- **Run all tests**: Execute all tests in the `tests/` folder.

```
pytest
```

API Endpoints

The API provides the following endpoints:

Method	Endpoint	Description
GET	`/books/`	List all books (async)
GET	`/books/{id}`	Get a book by ID
POST	`/books/`	Create a new book
PUT	`/books/{id}`	Update a book by ID
DELETE	`/books/{id}`	Delete a book by ID

Example Requests and Responses

- **Create a Book (POST /books/)**:

```
curl -X POST "http://127.0.0.1:8000/books/" -H "Content-Type:
    application/json" -d '{"title": "Sample Book", "author":
   "John Doe", "published_year": 2023}'
```

Response:

```
{
    "id": 1,
    "title": "Sample Book",
    "author": "John Doe",
    "published_year": 2023,
    "summary": null
}
```

- **Get a Book by ID (GET /books/{id})**:

```
curl -X GET "http://127.0.0.1:8000/books/1"
```

Response:

```
{
    "id": 1,
    "title": "Sample Book",
    "author": "John Doe",
    "published_year": 2023,
    "summary": null
}
```

- **List All Books (GET /books/)**:

```
curl -X GET "http://127.0.0.1:8000/books/"
```

Response:

```
[
    {
        "id": 1,
        "title": "Sample Book",
        "author": "John Doe",
        "published_year": 2023,
        "summary": null
    },
    {
        "id": 2,
        "title": "Another Book",
        "author": "Jane Smith",
        "published_year": 2020,
        "summary": "A fascinating read"
    }
]
```

Validations

The API enforces the following validations:

- `published_year`: Integer between 0 and the current year (e.g., 2025).

- `title` and `author`: Required fields; cannot be empty.
- `summary`: Optional field; can be null or a string.

Database Configuration

The application uses SQLite by default for simplicity. To switch to PostgreSQL or MySQL:

1. Update the database URL in `app/database/__init__.py`. For example:

```
SQLALCHEMY_DATABASE_URL = "postgresql://user:
   password@localhost:5432/book_catalog"
```

2. Install the required database driver (e.g., `pip install psycopg2`).
3. Run migrations if applicable (not included in this project).

Troubleshooting

Common issues and solutions:

- **ModuleNotFoundError**: Ensure the virtual environment is activated and dependencies are installed via `pip install -r requirements.txt`.
- **Port Conflict**: If port 8000 is in use, use `uvicorn app.main:app -port 8001 -reload`.
- **Database Connection Errors**: Check the database URL in `app/database/__init__.py` and ensure the database server is running.
- **Test Failures**: Ensure the database is initialized and check `tests/test_crud.py` or `tests/test_routes.py` for error messages.
- **Slow Performance**: Disable `-reload` for resource-constrained machines.
- **Dependency Issues**: Run `pip install -upgrade pip` and reinstall dependencies.

Report issues at https://github.com/Samar-Riaz/book-catalog/issues.

Development Tips

To extend or modify the API:

- **Adding Authentication**: Use `fastapi-users` to add user authentication and authorization.
- **Advanced Search**: Add query parameters to `/books/` for filtering by author or year.
- **Database Migrations**: Use Alembic for schema changes when switching databases.
- **Logging**: Implement logging with the `logging` module to monitor API usage.

- **Deployment**: Deploy using Docker or a cloud platform like Heroku.

Test changes thoroughly using `pytest` to maintain reliability.

Notes

- Swagger UI (`/docs`) and ReDoc (`/redoc`) provide interactive documentation and testing interfaces.
- The API is designed for extensibility, supporting features like authentication or advanced search.

Author

Samar Noor Riaz

Contributing

Contributions are welcome! To contribute:

1. Fork the repository from https://github.com/Samar-Riaz/book-catalog.
2. Create a feature branch (`git checkout -b feature-name`).
3. Commit your changes with clear messages (`git commit -m "Add feature X"`).
4. Submit a pull request with a detailed description.

Report bugs or suggest features at https://github.com/Samar-Riaz/book-catalog/issues.