Python flask : RESTful guide using flask_restful API and Resource Module with SQLAlchemy

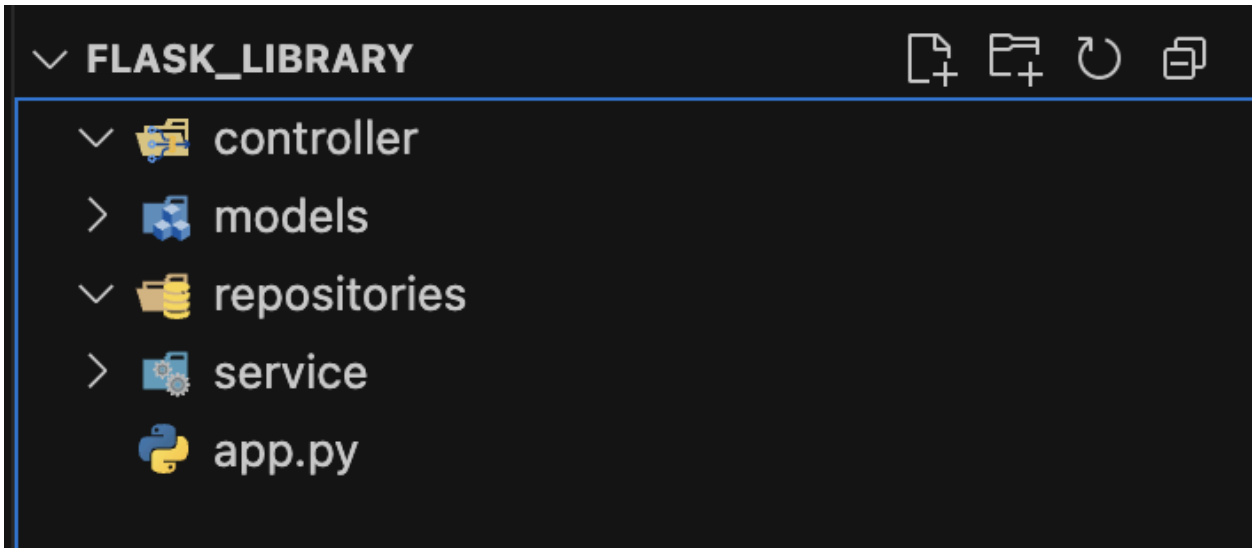
Step 1: Prerequisites

Install the following:

- Python through https://www.python.org/downloads/
- Virtual Environment (venv)

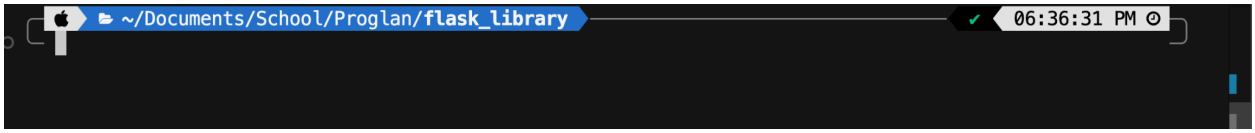
Step 2: Initialization

- Follow this folder architecture:



- In your terminal, go to your root directory ("flask_library")

NOTE: make sure you are in the root directory

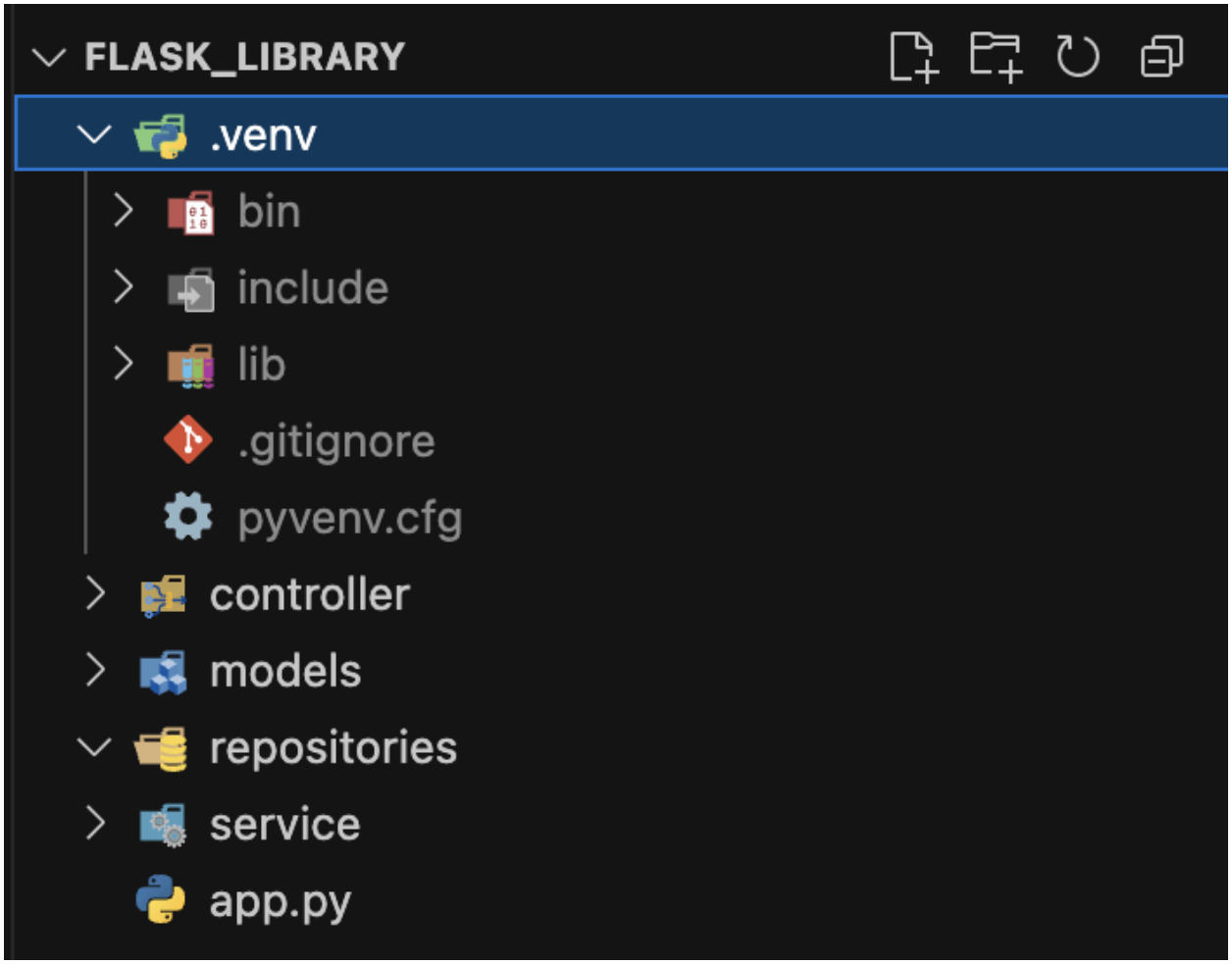


To generate the Virtual environment, input this command:

python3 -m venv .venv - *for mac*

python -m venv .venv - *for windows*

It will create a .venv folder which should look like this:

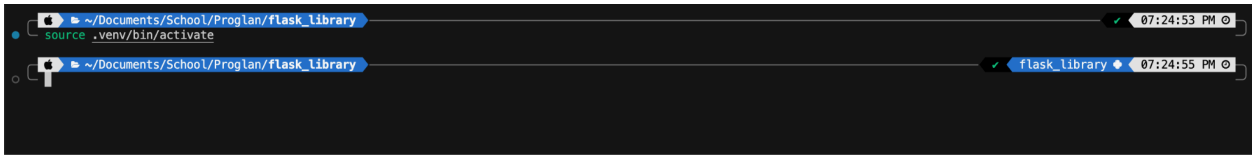


*We use **venv (virtual environment)** to create a private space for each Python project, so the packages you install don't interfere with other projects or your system Python. This way, one project can use Flask 3.0 while another uses Flask 2.2 without conflict, and it makes your project easier to share and run anywhere by keeping dependencies isolated.*

Once .venv is made, **activate** it:

For *mac/linux*: **source .venv/bin/activate**

For *windows (powershell)*: **.venv\Scripts\activate**



See that in the left portion of the CLI, it shows that the .venv is now activated,

*Note: to deactivate, just prompt "**deactivate**"*

Now, we'll install the dependencies:

- Flask
- Flask_restful
- Flask_sqlalchemy

*We install **Flask**, **Flask-RESTful**, and **Flask-SQLAlchemy** because each has a specific role in building our API. **Flask** is the core web framework that lets us create routes and run a web server. **Flask-RESTful** extends Flask with tools (like `Resource` and `Api`) that make it easier to build REST APIs with clean `GET`, `POST`, `PUT`, `DELETE` endpoints. **Flask-SQLAlchemy** is an Object Relational Mapper (ORM) that lets us interact with the database using Python classes and objects instead of raw SQL, making database operations simpler, safer, and more organized.*

Command:
pip install flask flask_sqlalchemy flask_restful

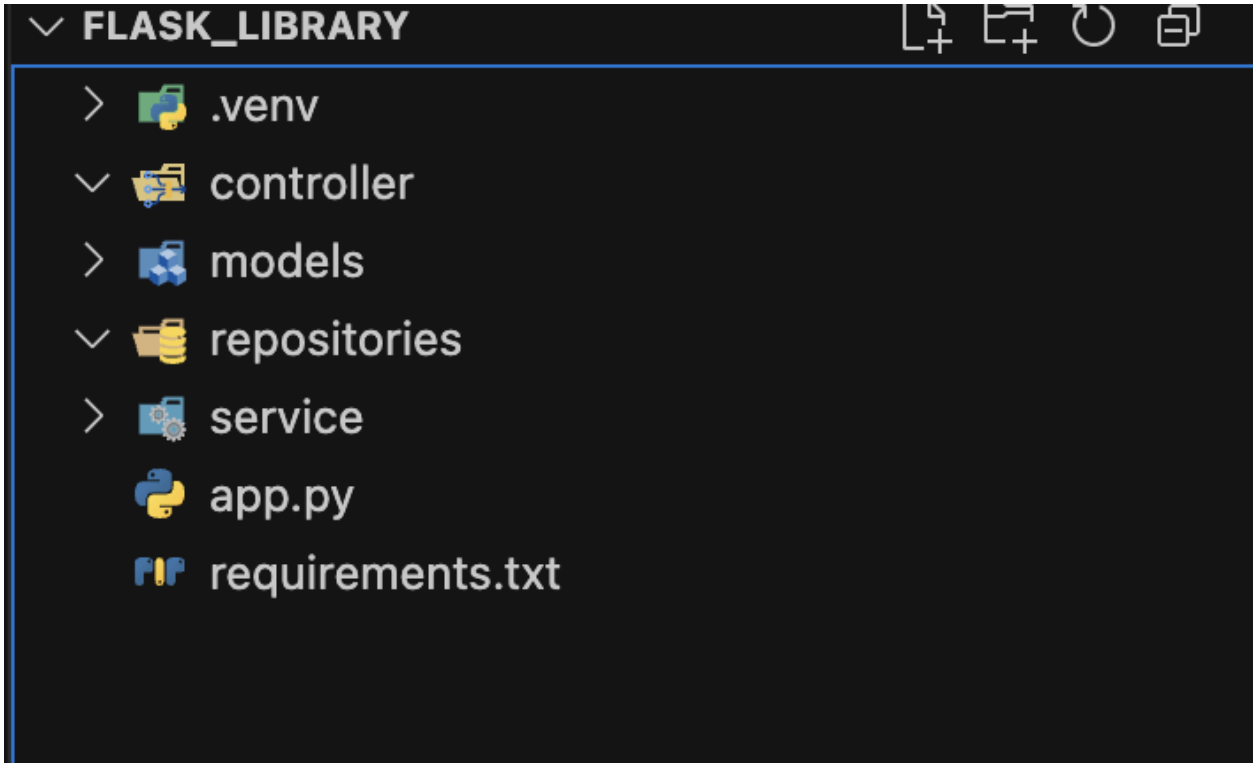
```
~/Documents/School/Proglan/flask_library                                  flask_library  07:24:55 PM
pip install flask flask_sqlalchemy flask_restful
```

Now, let's make the requirements.txt. Follow this command:

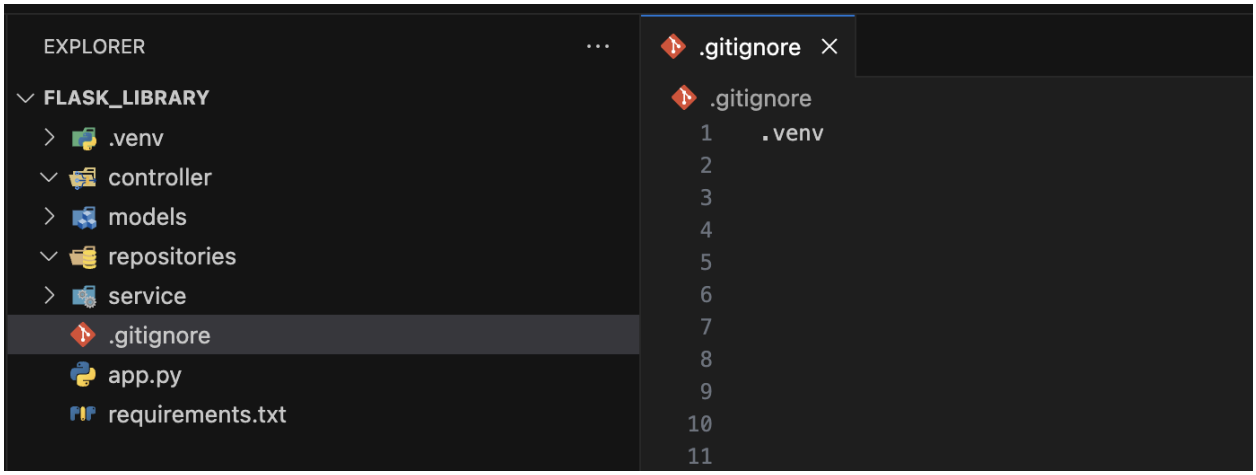
pip freeze > requirements.txt

```
~/Documents/School/Proglan/flask_library                                  5s  flask_library  07:29:10 PM
pip freeze > requirements.txt
```

Notice that:



(Optional) add .gitignore inside the root directory:



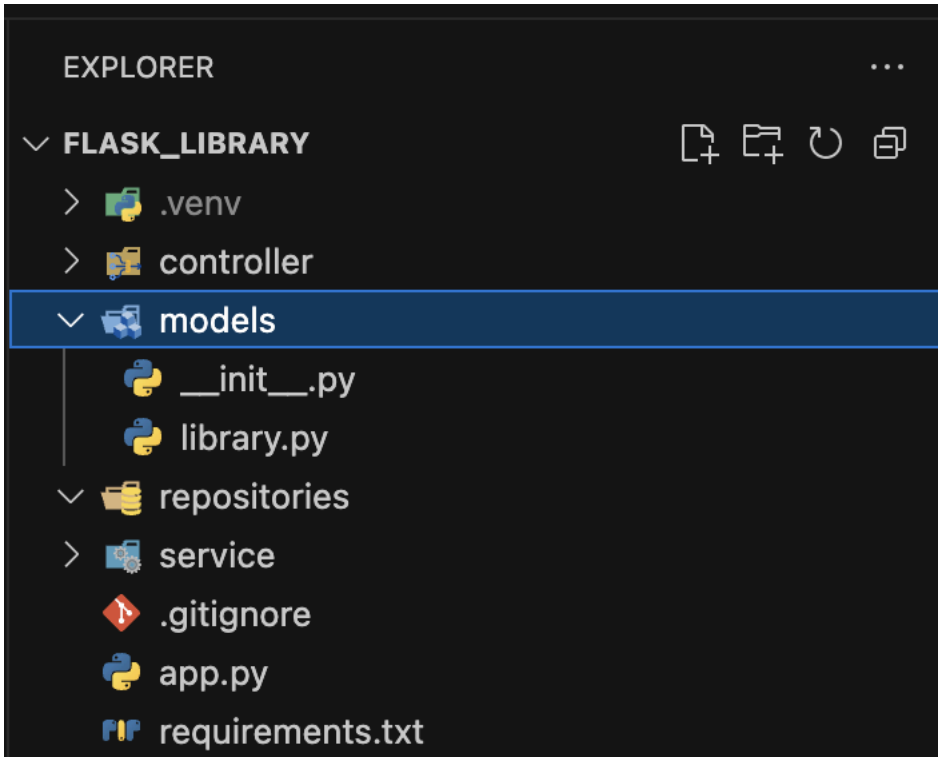
You add `.venv` to `.gitignore` because it's just your local Python environment—big, machine-specific, and easily recreated from `requirements.txt`—so there's no need to track it in Git.

(Optional) In any case that you cloned an existing code, you can just download the dependencies via the *requirements.txt*, use the command*:*

pip install -r requirements.txt

Step 3: Models

Add the following files in the models folder



Inside __init__.py:

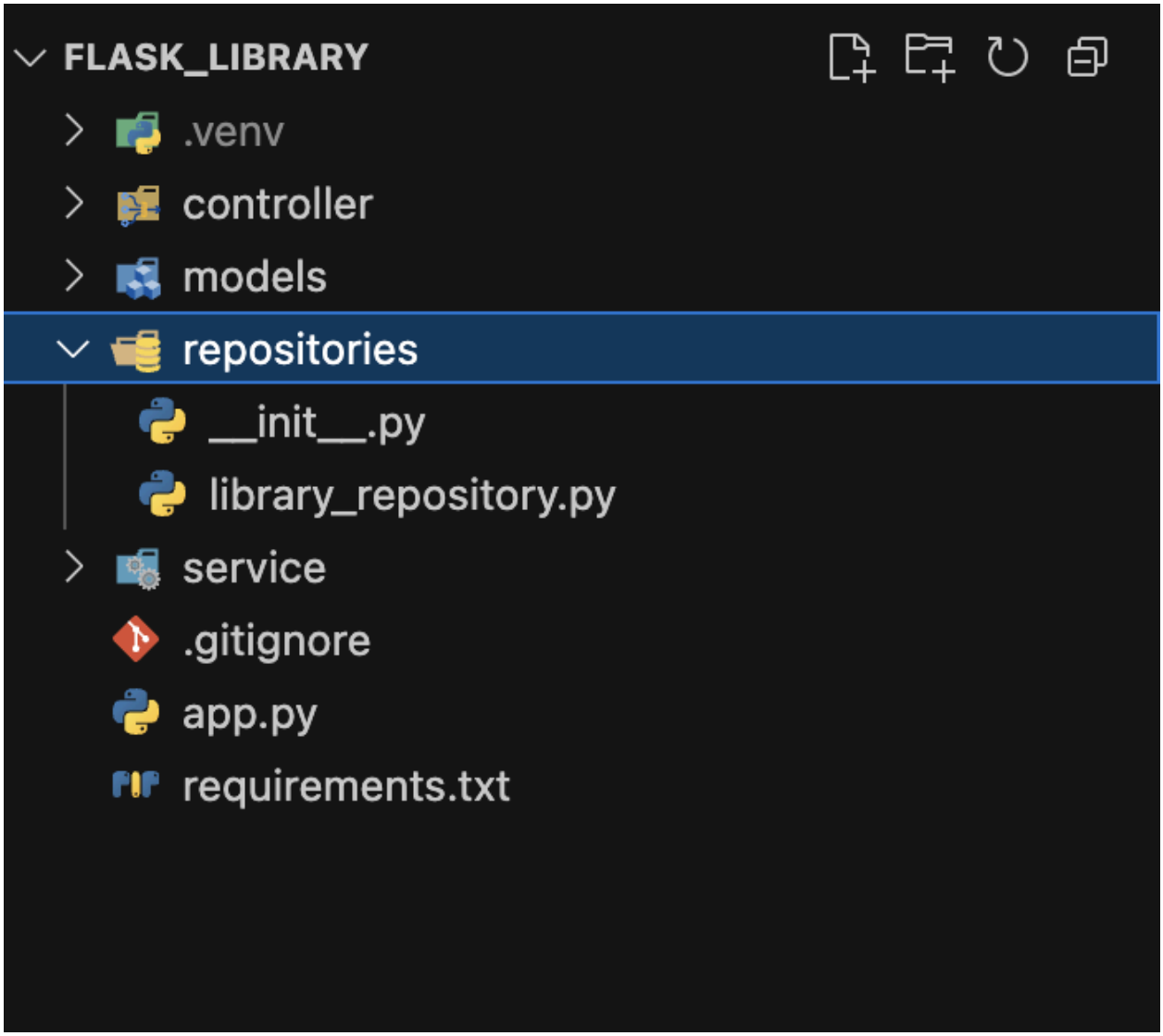
```python
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

from .library import Library
```

Inside library.py:

```python
from . import db

class Library(db.Model):
    __tablename__ = 'library'

    isbn = db.Column(db.String(13), primary_key=True)
    title = db.Column(db.String(80), nullable=False)
    author = db.Column(db.String(80), nullable=False,)

    def to_dict(self):
        return {
            "isbn": self.isbn,
            "title": self.title,
            "author": self.author
        }

```

Step 4: Repository

Add the following files in the repositories folder



Inside __init__.py

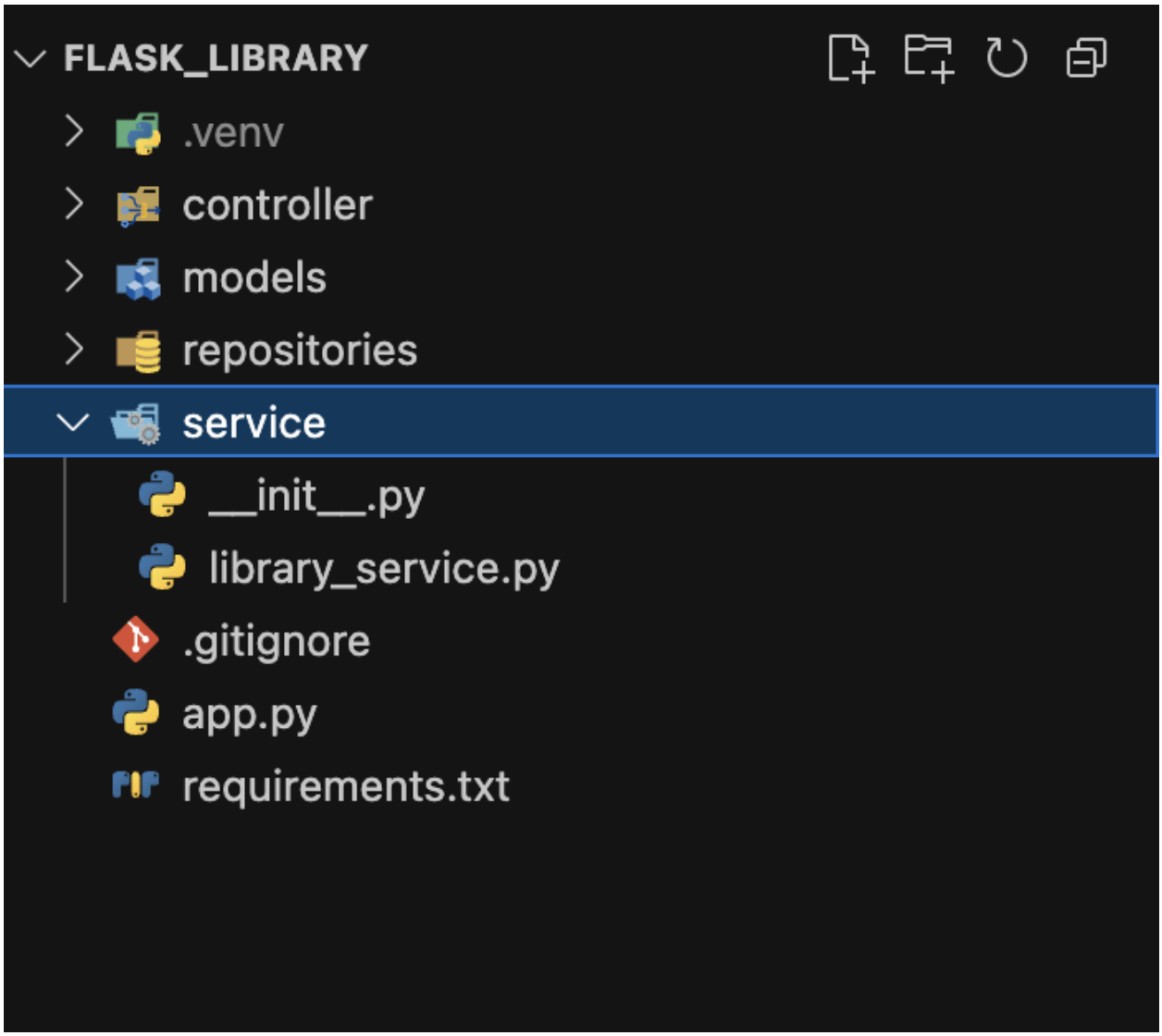
```
from .library_repository import LibraryRepository
```

Inside library_repository.py

```python
from models import Library, db


class LibraryRepository:
    @staticmethod
    def find_all():
        return Library.query.all()

    @staticmethod
    def find_by_id(isbn):
        return Library.query.get(isbn)

    @staticmethod
    def save(book):
        db.session.add(book)
        db.session.commit()
        return book

    @staticmethod
    def update(isbn, data: dict):
        book = Library.query.get(isbn)
        if not book:
            return None

        for key, value in data.items():
            if hasattr(book, key):
                setattr(book, key, value)

        db.session.commit()
        return book

    @staticmethod
    def delete(isbn):
        book = Library.query.get(isbn)
        if not book:
            return None

        db.session.delete(book)
        db.session.commit()
        return book

```

Step 5: Service

Add the following files in the service folder



Inside __init__.py

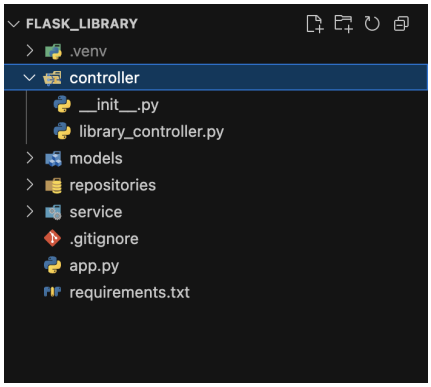
```
from .library_service import LibraryService
```

Inside library_service.py:

```python
from repositories import LibraryRepository
from models import Library


class LibraryService:
    @staticmethod
    def get_all():
        return LibraryRepository.find_all()

    @staticmethod
    def get_by_id(isbn):
        return LibraryRepository.find_by_id(isbn)

    @staticmethod
    def create(data):
        book = Library(
            isbn=data['isbn'],
            title=data['title'],
            author=data['author']
        )
        return LibraryRepository.save(book)

    @staticmethod
    def update(isbn, data):
        return LibraryRepository.update(isbn, data)

    @staticmethod
    def delete(isbn):
        return LibraryRepository.delete(isbn)

```

Step 6: Controller

Add the following files in the controller folder



Inside __init__.py:

```python
from .library_controller import LibraryListResource, LibraryResource
```

Inside library_controller.py:

```python
from flask_restful import Resource
from flask import request
from service import LibraryService
from models import db


class LibraryListResource(Resource):
    def get(self):
        books = LibraryService.get_all()
        return [book.to_dict() for book in books], 200

    def post(self):
        try:
            data = request.get_json()
            # Validate required fields
            if 'isbn' not in data or 'title' not in data or 'author' not in data:
                return {"message": "ISBN, title, and author are required"}, 400

            book = LibraryService.create(data)
            return book.to_dict(), 201
        except Exception as e:
            db.session.rollback()
            return {"error": "Unexpected error occurred", "details": str(e)}, 500


class LibraryResource(Resource):
    def get(self, isbn):
        book = LibraryService.get_by_id(isbn)
        if not book:
            return {"message": "Book not found"}, 404
        return book.to_dict(), 200

    def put(self, isbn):
        try:
            data = request.get_json()
            book = LibraryService.update(isbn, data)
            if not book:
                return {"message": "Book not found"}, 404
            return book.to_dict(), 200
        except Exception as e:
            db.session.rollback()
            return {"error": "Unexpected error occurred", "details": str(e)}, 500

    def delete(self, isbn):
        try:
            book = LibraryService.delete(isbn)
            if not book:
                return {"message": "Book not found"}, 404
            return {"message": f"Book with ISBN {isbn} deleted"}, 200
        except Exception as e:
            db.session.rollback()
            return {"error": "Unexpected error occurred", "details": str(e)}, 500

```

Step 7: app.py (entry point)

For simplicity, we used SQLite.

```python
from flask import Flask
from flask_restful import Api
from models import db
from controller import LibraryResource, LibraryListResource

app = Flask(__name__)
api = Api(app)

# DB Configuration
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

db.init_app(app)

@app.before_request
def create_tables():
    db.create_all()

# Library routes
api.add_resource(LibraryListResource, '/api/library')
api.add_resource(LibraryResource, '/api/library/<string:isbn>')

if __name__ == '__main__':
    app.run(debug=True)
```

Step 8: Run the application

To run the application, use the command:

python3 app.py *(for mac)*

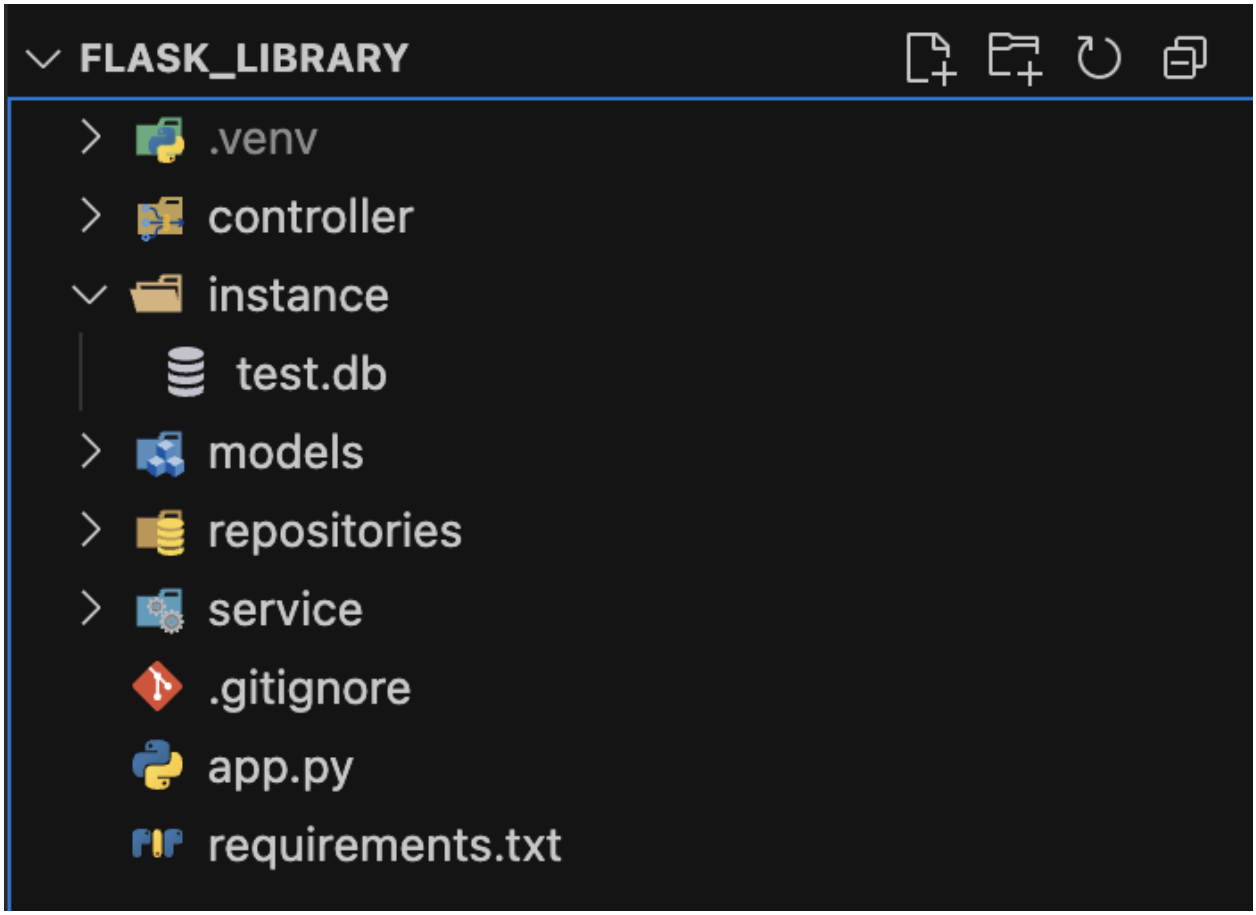
python app.py *(for windows)*

NOTE: make sure you are in the root directory and the .venv is activated.

You should see this if it is successful:

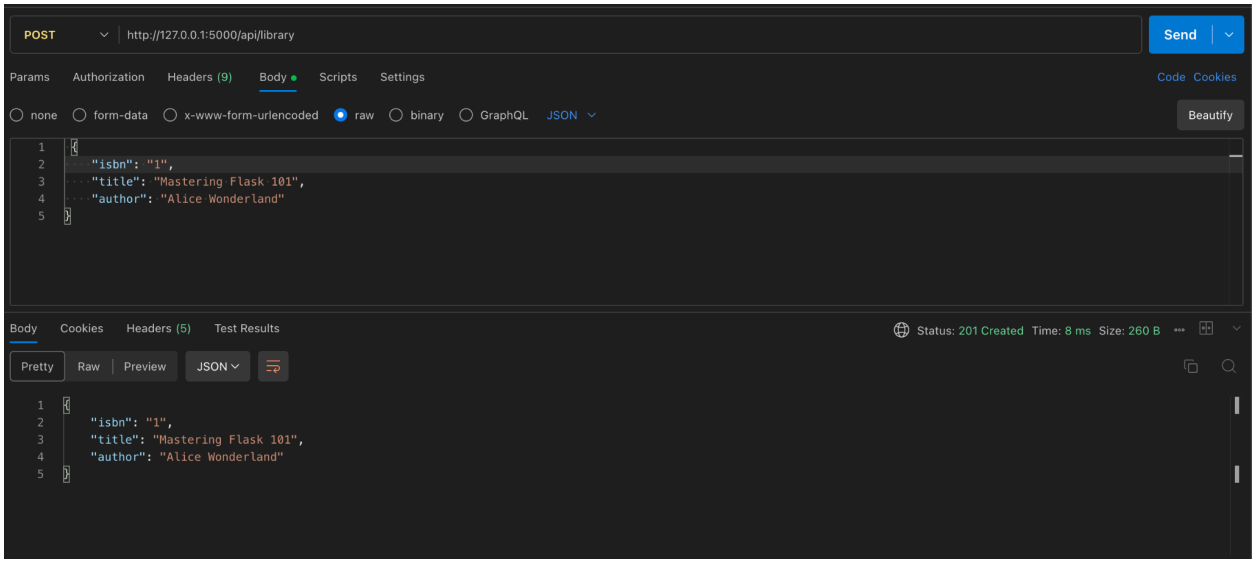
```
~/Documents/School/Proglan/flask_library                                  flask_library  02:17:23 PM
 python3 app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 118-863-738
```

Once it is successful, there will be an instance folder is automatically generated in the directory:

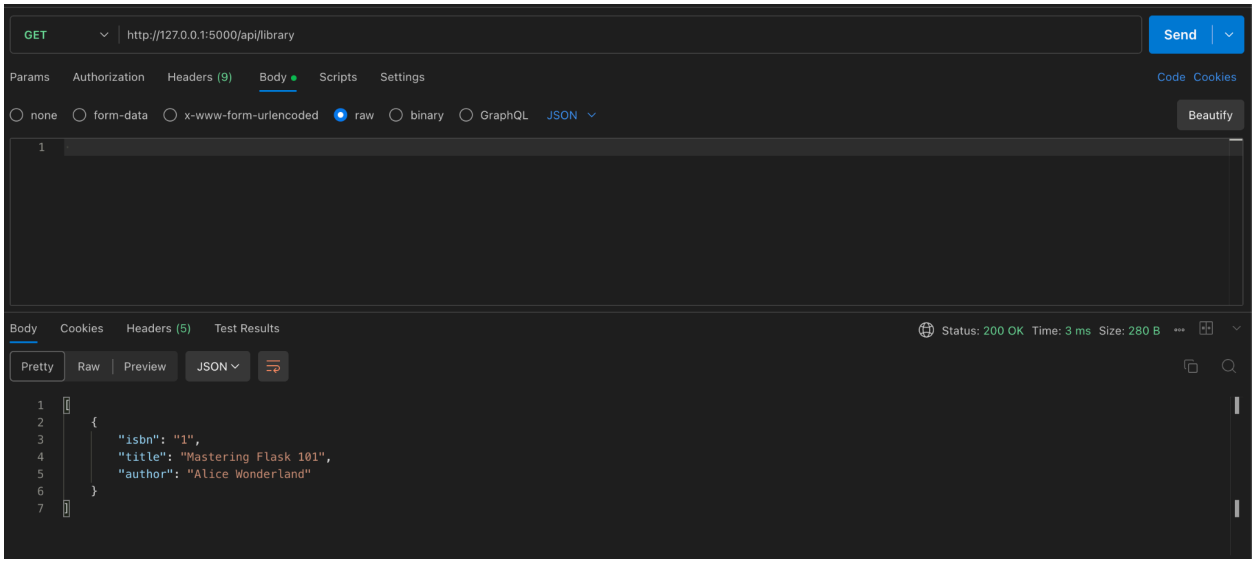


Step 9: Test through postman:

POST: /api/library

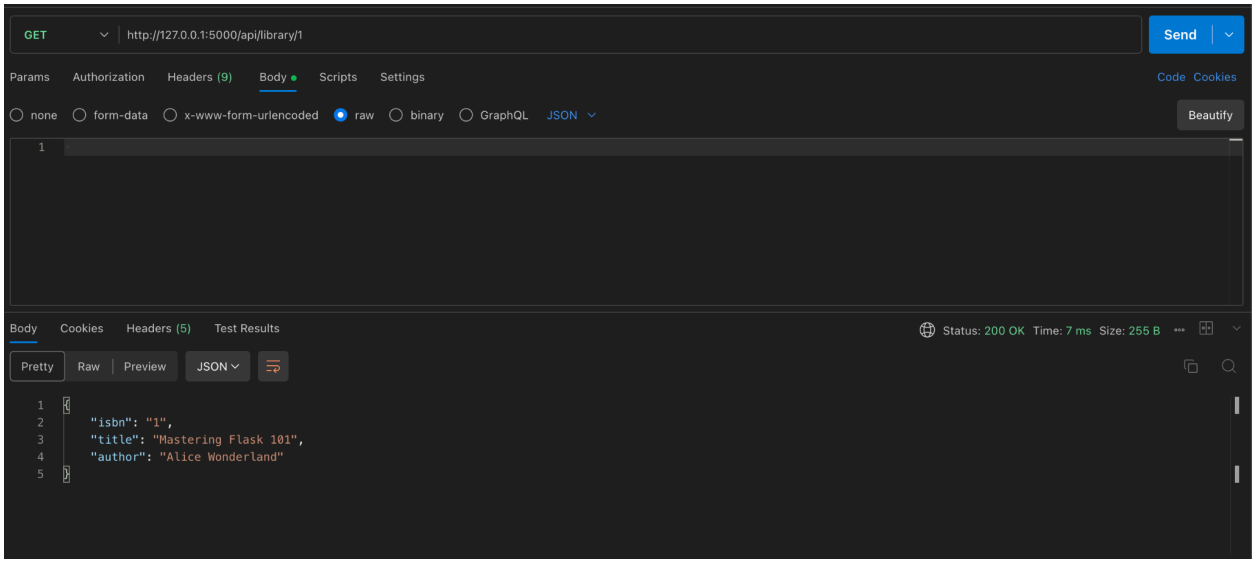


GET: /api/library

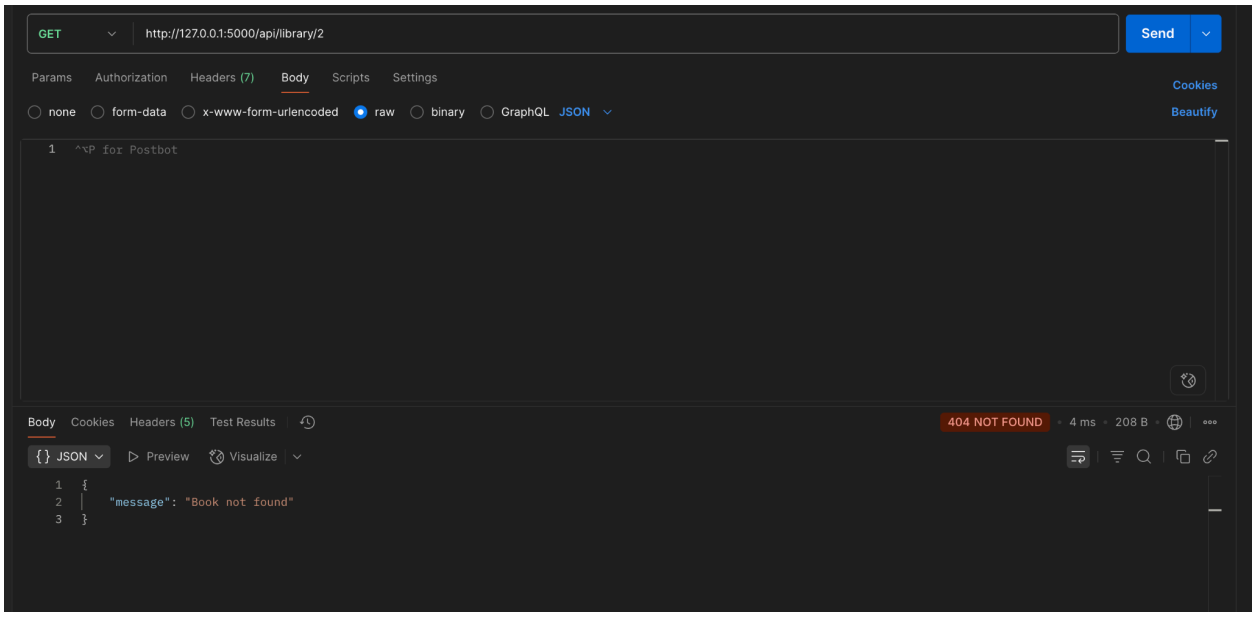


GET: /api/library/<string:isbn>

If book exists:

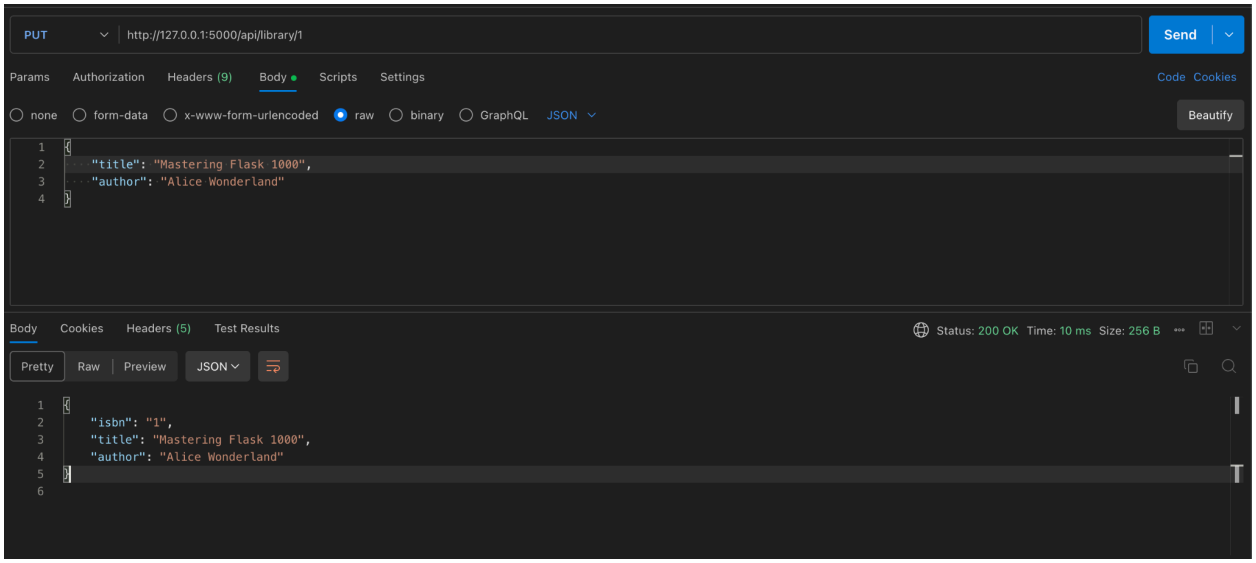


If book does not exist:

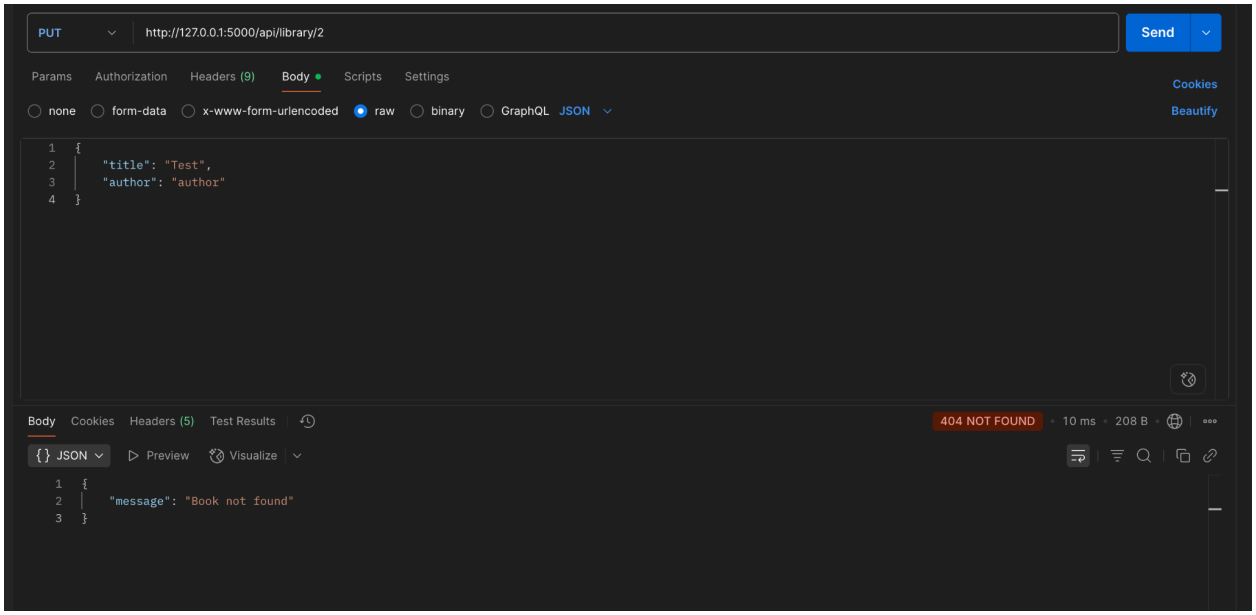


UPDATE: /api/library/<string:isbn>

If book exists:

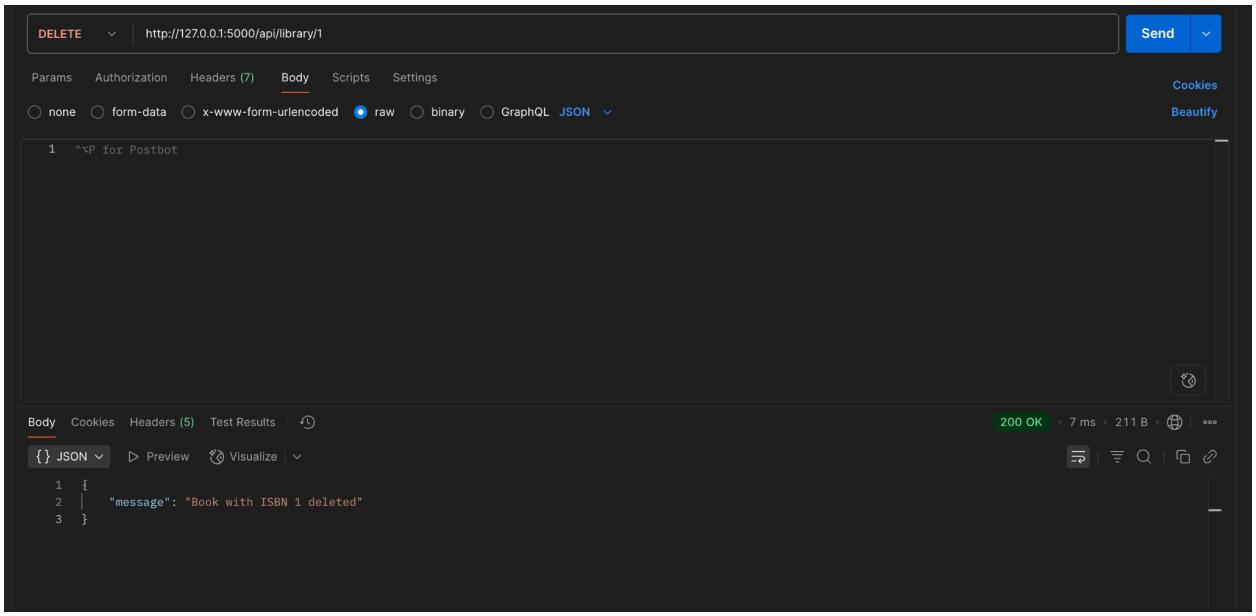


If book does not exist:



DELETE: /api/library/<string:isbn>

If book exists:



If book does not exist: