1) Certainly! To create a simple API in Python 3, yol can use a web framework like Flask. Here's a basic example to get me started:

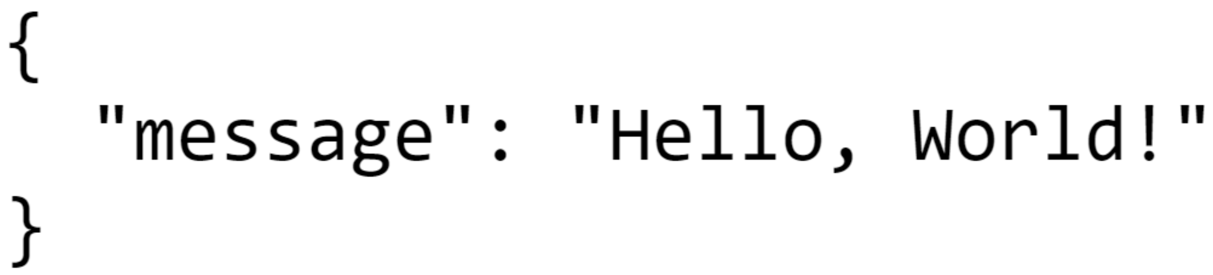
Install Flask:

```
pip install Flask
```

Create a file named `app.py` with the following code:

```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/api', methods=['GET'])
def hello_world():
    return jsonify(message='Hello, World!')

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

This code sets up a basic Flask application with a single route `/api` that responds to GET requests with a JSON message.



```
{
  "message": "Hello, World!"
}
```

Visit `http://127.0.0.1:5000/api` in your browser or use a tool like `curl` or Postman to make a GET request, and you should see the JSON response.

Feel free to expand on this example to add more routes, handle different HTTP methods, and connect to databases or external services as needed for your API.

MAIN PROJECT

Q1) how to create a library management system with crud where there will be an integrated table like Books: and the columns will be name as string price as float author as string
and use alembic for automatic tables created in python 3?

Solution:
To create a simple library management system with CRUD operations and an integrated table using SQLAlchemy and Alembic in Python 3, you can follow these steps:

1) Install SQLAlchemy and Alembic by running:

```
pip install SQLAlchemy Alembic
```

2) Open PyCharm or your preferred IDE and create a new Python project:
3) It's good practice to use a virtual environment. Create and activate a virtual environment:

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

4) Install Flask-SQLAlchemy for database integration:

```
pip install Flask-SQLAlchemy
```

5) Create a file named `app.py` with the following code:

```
from flask import Flask, render_template, request, redirect, url_for
from flask_sqlalchemy import SQLAlchemy
from flask_migrate import Migrate

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///library.db'
db = SQLAlchemy(app)
migrate = Migrate(app, db)

class Book(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(255), nullable=False)
    price = db.Column(db.Float, nullable=False)
    author = db.Column(db.String(255), nullable=False)

@app.route('/')
def index():
    books = Book.query.all()
    return render_template('index.html', books=books)

@app.route('/add', methods=['GET', 'POST'])
def add_book():
    if request.method == 'POST':
        name = request.form['name']
        price = float(request.form['price'])
        author = request.form['author']

        new_book = Book(name=name, price=price, author=author)
        db.session.add(new_book)
        db.session.commit()

        return redirect(url_for('index'))

    return render_template('add_book.html')

@app.route('/edit/<int:book_id>', methods=['GET', 'POST'])
def edit_book(book_id):
    book = Book.query.get(book_id)

    if request.method == 'POST':
        book.name = request.form['name']
        book.price = float(request.form['price'])
        book.author = request.form['author']

        db.session.commit()
        return redirect(url_for('index'))

    return render_template('edit_book.html', book=book)

@app.route('/delete/<int:book_id>')
def delete_book(book_id):
    book = Book.query.get(book_id)
    db.session.delete(book)
    db.session.commit()
    return redirect(url_for('index'))

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

6) Create the following HTML templates in a folder named `templates`:
EX : `templates/index.html`:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Library Management System</title>
</head>
<body>
    <h1>Library Management System</h1>
    <table border="1">
        <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Price</th>
            <th>Author</th>
            <th>Actions</th>
        </tr>
        {% for book in books %}
        <tr>
            <td>{{ book.id }}</td>
            <td>{{ book.name }}</td>
            <td>{{ book.price }}</td>
            <td>{{ book.author }}</td>
            <td>
                <a href="{{ url_for('edit_book', book_id=book.id) }}">Edit</a>
                <a href="{{ url_for('delete_book', book_id=book.id) }}" onclick="return confirm('Are you sure you want to delete this book?')">Delete</a>
            </td>
        </tr>
        {% endfor %}
    </table>
    <p><a href="{{ url_for('add_book') }}">Add Book</a></p>
</body>
</html>
```

Create `templates/add_book.html`

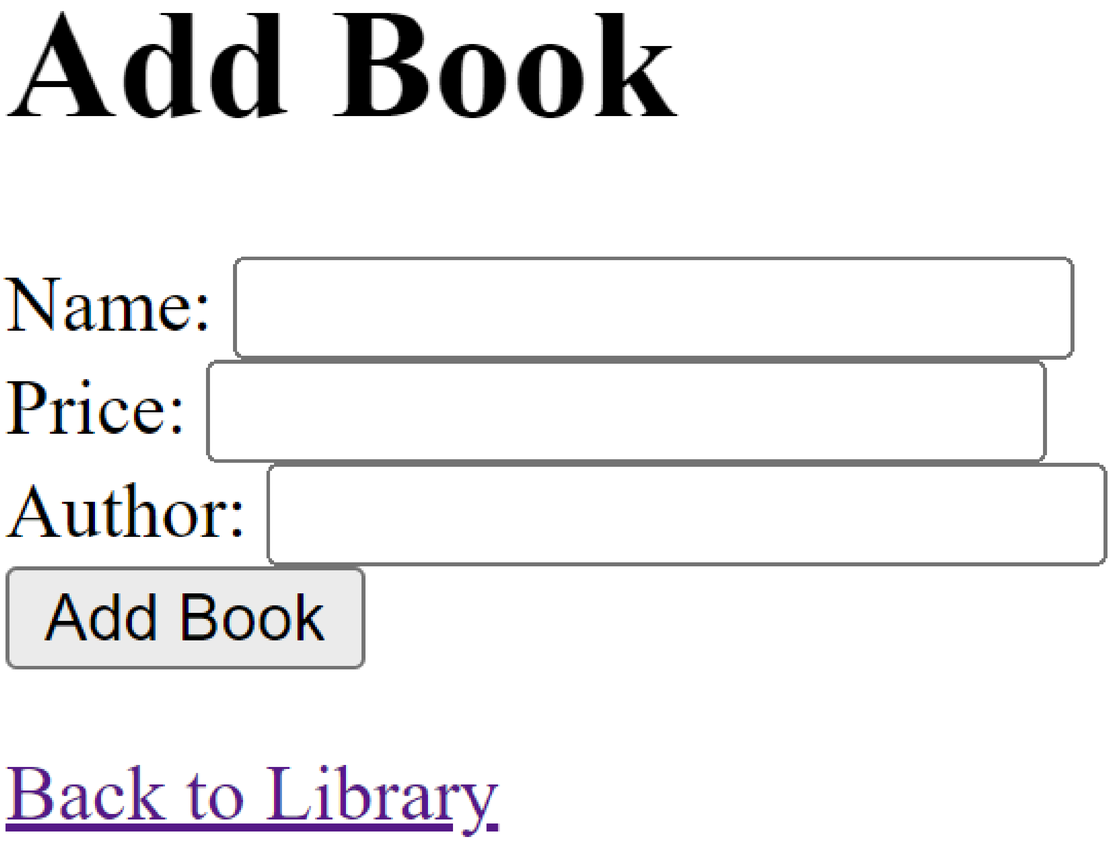
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible"
content="IE=edge">
    <meta name="viewport"
content="width=device-width, initial-scale=1.0">
    <title>Add Book</title>
</head>
<body>
    <h1>Add Book</h1>
    <form method="post" action="{{
url_for('add_book') }}">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name"
required><br>

        <label for="price">Price:</label>
        <input type="number" id="price"
name="price" step="0.01" required><br>

        <label for="author">Author:</label>
        <input type="text" id="author"
name="author" required><br>

        <input type="submit" value="Add Book">
    </form>
    <p><a href="{{ url_for('index') }}">Back to
Library</a></p>
</body>
</html>
```

CREATE templates/edit_book.html :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Edit Book</title>
</head>
<body>
    <h1>Edit Book</h1>
    <form method="post" action="{{ url_for('edit_book', book_id=book.id) }}">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" value="{{ book.name }}" required><br>

        <label for="price">Price:</label>
        <input type="number" id="price" name="price" step="0.01" value="{{ book.price }}" required><br>

        <label for="author">Author:</label>
        <input type="text" id="author" name="author" value="{{ book.author }}" required><br>

        <input type="submit" value="Save Changes">
    </form>
    <p><a href="{{ url_for('index') }}">Back to Library</a></p>
</body>
```

```
</html>
```

For http://127.0.0.1:5000/add :



Add Book

Name:

Price:

Author:

Back to Library

FOR DATABASE :

The error you're encountering, OperationalError: no such table: book, indicates that the table named book does not exist in your SQLite database.

To resolve this issue, you need to create the database tables. Flask-Migrate can help with this. Follow these steps:

1) In your terminal, ensure your virtual environment is activated.
2) Run the following commands to initialize Flask-Migrate:

```
flask db init
```

This command initializes a migrations directory in your project.

3) Now, create an initial migration:

```
flask db migrate -m "Initial migration"
```

This command generates a migration script based on your models.

4) Apply the migration to create the tables:

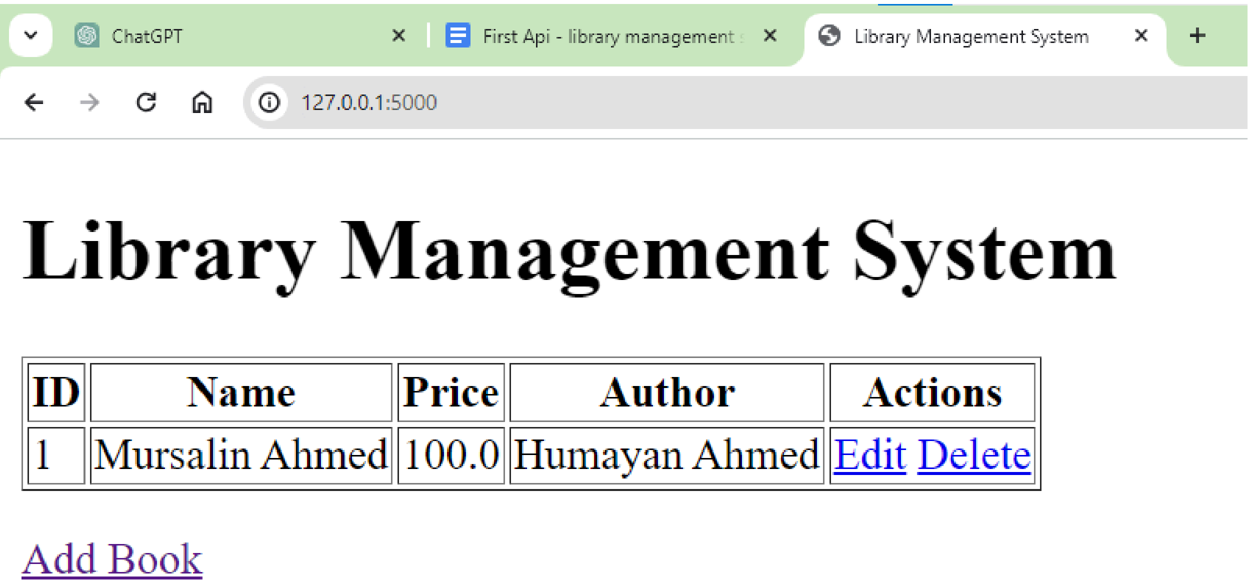
```
flask db upgrade
```

This command applies the migration and creates the necessary tables in your database.

After performing these steps, your tables should be created, and you should be able to run your Flask application without encountering the "no such table" error.

Make sure to perform these steps in the same environment where you installed Flask-Migrate and where your Flask application is located. If you encounter any issues, please double-check the commands and ensure that you are using the correct virtual environment.

Now output :



Library Management System

ID	Name	Price	Author	Actions
1	Mursalin Ahmed	100.0	Humayan Ahmed	Edit Delete

Add Book