Step 1: install all below listed libraries / create requirements.txt file and add this code

aniso8601==9.0.1  
blinker==1.7.0  
click==8.1.7  
Flask==3.0.0  
Flask-Cors==4.0.0  
Flask-RESTful==0.3.10  
flask-swagger-ui==4.11.1  
gunicorn==21.2.0  
itsdangerous==2.1.2  
Jinja2==3.1.2  
MarkupSafe==2.1.3  
packaging==23.2  
python-dotenv==1.0.0  
pytz==2023.3.post1  
PyYAML==6.0.1  
six==1.16.0  
Werkzeug==3.0.1

go to terminal and install one by one using below command.

# Update the python package manager

pip3 install --upgrade pip

# Install the required libraries

pip3 install Flask

pip3 install Flask-Cors

pip3 install Flask-RESTful

pip3 install flask-swagger-ui

pip3 install gunicorn

pip3 install python-dotenv

Step 2: create a folder called resource in pycharm project and add two files “swaggerConfig.py” and “bookResource.py”.

Code for swaggerConfig.py is below

from flask\_restful import Resource  
from flask import jsonify  
import json  
  
  
class SwaggerConfig(Resource):  
 def get(self):  
 with open('static/swagger/config.json', 'r') as config\_file:  
 config\_data = json.load(config\_file)  
 return jsonify(config\_data)

Step 3: code for bookResource.py

from flask\_restful import Resource  
from flask import request  
import json  
  
books = [{"id": 1, "title": "Java book"},  
 {"id": 2, "title": "Python book"}]  
  
  
class BooksGETResource(Resource):  
 def get(self):  
 return books  
  
class BookGETResource(Resource):  
 def get(self, id):  
 for book in books:  
 if book["id"] == id:  
 return book  
 return None  
  
  
class BookPOSTResource(Resource):  
 def post(self):  
 book = json.loads(request.data)  
 new\_id = max(book["id"] for book in books) + 1  
 book["id"] = new\_id  
 books.append(book)  
 return book  
  
  
class BookPUTResource(Resource):  
 def put(self, id):  
 book = json.loads(request.data)  
 for \_book in books:  
 if \_book["id"] == id:  
 \_book.update(book)  
 return \_book  
  
  
class BookDELETEResource(Resource):  
 def delete(self, id):  
 global books  
 books = [book for book in books if book["id"] != id]  
 return "", 204

Step 4: create a folder called “static” in that create one more folder “config” add config.json file in that.

Code for config.json file:

{  
 "openapi": "3.0.3",  
 "info": {  
 "title": "Flask API",  
 "description": "",  
 "version": "1.0.0"  
 },  
 "servers": [  
 {  
 "url": "http://localhost:5000"  
 },  
 {  
 "url": "http://127.0.0.1:5000"  
 }  
 ],  
 "tags": [  
 {  
 "name": "book",  
 "description": "CRUD Operations"  
 }  
 ],  
 "paths": {  
 "/books": {  
 "get": {  
 "tags": [  
 "book"  
 ],  
 "summary": "Retrieve all books",  
 "responses": {  
 "200": {  
 "description": "Successful",  
 "content": {  
 "application/json": {  
 "schema": {  
 "$ref": "#/components/schemas/Book"  
 }  
 }  
 }  
 }  
 }  
 },  
 "post": {  
 "tags": [  
 "book"  
 ],  
 "summary": "Create a book",  
 "requestBody": {  
 "content": {  
 "application/json": {  
 "schema": {  
 "type": "object",  
 "properties": {  
 "title": {  
 "type": "string",  
 "example": "Nuxt"  
 }  
 }  
 }  
 }  
 }  
 },  
 "responses": {  
 "200": {  
 "description": "Created successfully",  
 "content": {  
 "application/json": {  
 "schema": {  
 "type": "object",  
 "properties": {  
 "token": {  
 "type": "string"  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
 },  
 "/books/{id}": {  
 "get": {  
 "tags": [  
 "book"  
 ],  
 "summary": "Retrieve specific book",  
 "parameters": [  
 {  
 "name": "id",  
 "in": "path",  
 "required": true,  
 "schema": {  
 "type": "string"  
 },  
 "description": ""  
 }  
 ],  
 "responses": {  
 "200": {  
 "description": "Successful",  
 "content": {  
 "application/json": {  
 "schema": {  
 "$ref": "#/components/schemas/Book"  
 }  
 }  
 }  
 },  
 "404": {  
 "description": "Not found"  
 }  
 },  
 "security": [  
 {  
 "bearerAuth": []  
 }  
 ]  
 },  
 "put": {  
 "tags": [  
 "book"  
 ],  
 "summary": "Update a book",  
 "parameters": [  
 {  
 "name": "id",  
 "in": "path",  
 "required": true,  
 "schema": {  
 "type": "integer",  
 "format": "int32"  
 },  
 "description": "ID of the book to be updated"  
 }  
 ],  
 "requestBody": {  
 "content": {  
 "application/json": {  
 "schema": {  
 "type": "object",  
 "properties": {  
 "title": {  
 "type": "string",  
 "example": "SQL"  
 }  
 },  
 "required": [  
 "title"  
 ]  
 }  
 }  
 }  
 },  
 "responses": {  
 "200": {  
 "description": "Updated successfully",  
 "content": {  
 "application/json": {  
 "schema": {  
 "type": "object",  
 "properties": {  
 "token": {  
 "type": "string"  
 }  
 }  
 }  
 }  
 }  
 },  
 "404": {  
 "description": "Book not found"  
 }  
 }  
 },  
 "delete": {  
 "tags": [  
 "book"  
 ],  
 "summary": "Delete a book",  
 "parameters": [  
 {  
 "name": "id",  
 "in": "path",  
 "required": true,  
 "schema": {  
 "type": "integer",  
 "format": "int32"  
 },  
 "description": "ID of the book to be deleted"  
 }  
 ],  
 "responses": {  
 "204": {  
 "description": "Deleted successfully"  
 },  
 "404": {  
 "description": "Book not found"  
 }  
 }  
 }  
 }  
 },  
 "components": {  
 "schemas": {  
 "Book": {  
 "type": "object",  
 "properties": {  
 "id": {  
 "type": "integer",  
 "format": "int64",  
 "example": 1  
 },  
 "title": {  
 "type": "string",  
 "example": "Java"  
 }  
 }  
 }  
 }  
 }  
}

Step 5: Create folder called “util” in that create file called”common.py” and add the code

import dotenv  
import os  
import json  
  
  
class ENVIRONMENT:  
 def \_\_init\_\_(self):  
 project\_dir = os.path.join(os.path.dirname(\_\_file\_\_), os.pardir)  
 dotenv\_path = os.path.join(project\_dir, '.env')  
 dotenv.load\_dotenv(dotenv\_path)  
 self.domain = os.getenv("DOMAIN")  
 self.port = os.getenv("PORT")  
 self.prefix = os.getenv("PREFIX")  
  
 def get\_instance(self):  
 if not hasattr(self, "\_instance"):  
 self.\_instance = ENVIRONMENT()  
 return self.\_instance  
  
 def getDomain(self):  
 return self.domain  
  
 def getPort(self):  
 return self.port  
  
 def getPrefix(self):  
 return self.prefix  
  
  
domain = ENVIRONMENT().get\_instance().getDomain()  
port = ENVIRONMENT().get\_instance().getPort()  
prefix = ENVIRONMENT().get\_instance().getPrefix()  
  
  
def build\_swagger\_config\_json():  
 config\_file\_path = 'static/swagger/config.json'  
  
 with open(config\_file\_path, 'r') as file:  
 config\_data = json.load(file)  
  
 config\_data['servers'] = [  
 {"url": f"http://localhost:{port}{prefix}"},  
 {"url": f"http://{domain}:{port}{prefix}"}  
 ]  
  
 new\_config\_file\_path = 'static/swagger/config.json'  
  
 with open(new\_config\_file\_path, 'w') as new\_file:  
 json.dump(config\_data, new\_file, indent=2)

Step 5: add a file .env and add below code :

DOMAIN=127.0.0.1  
PORT=5000  
PREFIX=

Step 6: in application create a file “application.py” add below code

from flask import Flask, jsonify, redirect  
from flask\_restful import Api, MethodNotAllowed, NotFound  
from flask\_cors import CORS  
from util.common import domain, port, prefix, build\_swagger\_config\_json  
from resources.swaggerConfig import SwaggerConfig  
from resources.bookResource import BooksGETResource, BookGETResource, BookPOSTResource, BookPUTResource, BookDELETEResource  
from flask\_swagger\_ui import get\_swaggerui\_blueprint  
  
# ============================================  
# Main  
# ============================================  
application = Flask(\_\_name\_\_)  
app = application  
app.config['PROPAGATE\_EXCEPTIONS'] = True  
CORS(app)  
api = Api(app, prefix=prefix, catch\_all\_404s=True)  
  
# ============================================  
# Swagger  
# ============================================  
build\_swagger\_config\_json()  
swaggerui\_blueprint = get\_swaggerui\_blueprint(  
 prefix,  
 f'http://{domain}:{port}{prefix}/swagger-config',  
 config={  
 'app\_name': "Flask API",  
 "layout": "BaseLayout",  
 "docExpansion": "none"  
 },  
)  
app.register\_blueprint(swaggerui\_blueprint)  
  
# ============================================  
# Error Handler  
# ============================================  
  
  
@app.errorhandler(NotFound)  
def handle\_method\_not\_found(e):  
 response = jsonify({"message": str(e)})  
 response.status\_code = 404  
 return response  
  
  
@app.errorhandler(MethodNotAllowed)  
def handle\_method\_not\_allowed\_error(e):  
 response = jsonify({"message": str(e)})  
 response.status\_code = 405  
 return response  
  
  
@app.route('/')  
def redirect\_to\_prefix():  
 if prefix != '':  
 return redirect(prefix)  
  
  
# ============================================  
# Add Resource  
# ============================================  
# GET swagger config  
api.add\_resource(SwaggerConfig, '/swagger-config')  
# GET books  
api.add\_resource(BooksGETResource, '/books')  
api.add\_resource(BookGETResource, '/books/<int:id>')  
# POST book  
api.add\_resource(BookPOSTResource, '/books')  
# PUT book  
api.add\_resource(BookPUTResource, '/books/<int:id>')  
# DELETE book  
api.add\_resource(BookDELETEResource, '/books/<int:id>')  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)