Cloud Computing Practical 8 Report

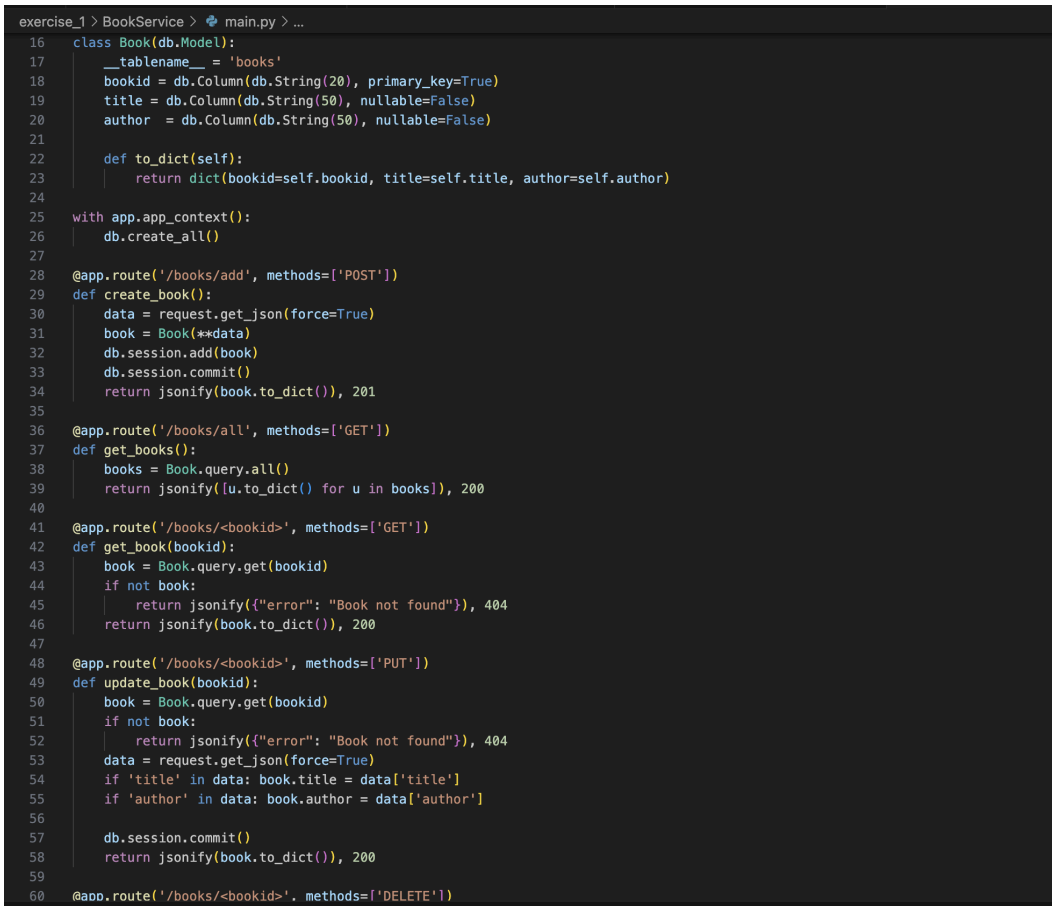
The report illustrates how every exercise has been executed and the output of the results along with the screenshots

Exercise 1 Service setup:

We have 2 services set up for exercise 1- **UserService and BookService**.
The main.py for **UserService** has already been provided to us and using the same structure we have set up **BookService**'s main.py file.
We create the **Dockerfile** for each of them- one Dockerfile for UserService and one Dockerfile for BookService.
The Dockerfile uses **python:3.9-slim as the image** and **creates the work directory as /app** where we **copy all the app source code** and **run all the commands**. We first update the apt package index, install gcc and libpq-dev (for using psycopg2 which is Postgres driver for Python). We then **install the requirements and run** using requirements.txt into /app and copy the UserService code (similarly we do for BookService in its respective Dockerfile). We then **run the app on port 5006** and when the container starts, we run the command "**python -u main.py" in Dockerfile**. All these commands are written inside the Dockerfile as ->

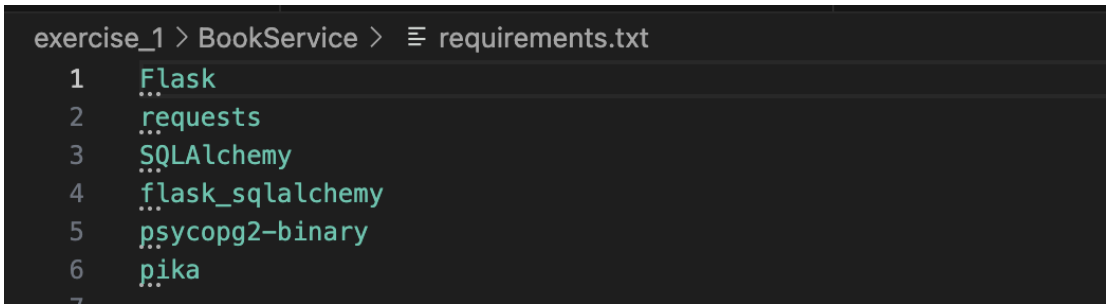
Similar to UserService, we create **BookService's main.py, requirements.txt and Dockerfile.**

Bookservice's main.py (Similar to the UserService's main.py except the Postgres DB' table name, columns etc have been changed)



```python
class Book(db.Model):
    __tablename__ = 'books'
    bookid = db.Column(db.String(20), primary_key=True)
    title = db.Column(db.String(50), nullable=False)
    author  = db.Column(db.String(50), nullable=False)

    def to_dict(self):
        return dict(bookid=self.bookid, title=self.title, author=self.author)

with app.app_context():
    db.create_all()

@app.route('/books/add', methods=['POST'])
def create_book():
    data = request.get_json(force=True)
    book = Book(**data)
    db.session.add(book)
    db.session.commit()
    return jsonify(book.to_dict()), 201

@app.route('/books/all', methods=['GET'])
def get_books():
    books = Book.query.all()
    return jsonify([u.to_dict() for u in books]), 200

@app.route('/books/<bookid>', methods=['GET'])
def get_book(bookid):
    book = Book.query.get(bookid)
    if not book:
        return jsonify({"error": "Book not found"}), 404
    return jsonify(book.to_dict()), 200

@app.route('/books/<bookid>', methods=['PUT'])
def update_book(bookid):
    book = Book.query.get(bookid)
    if not book:
        return jsonify({"error": "Book not found"}), 404
    data = request.get_json(force=True)
    if 'title' in data: book.title = data['title']
    if 'author' in data: book.author = data['author']

    db.session.commit()
    return jsonify(book.to_dict()), 200

@app.route('/books/<bookid>', methods=['DELETE'])
```

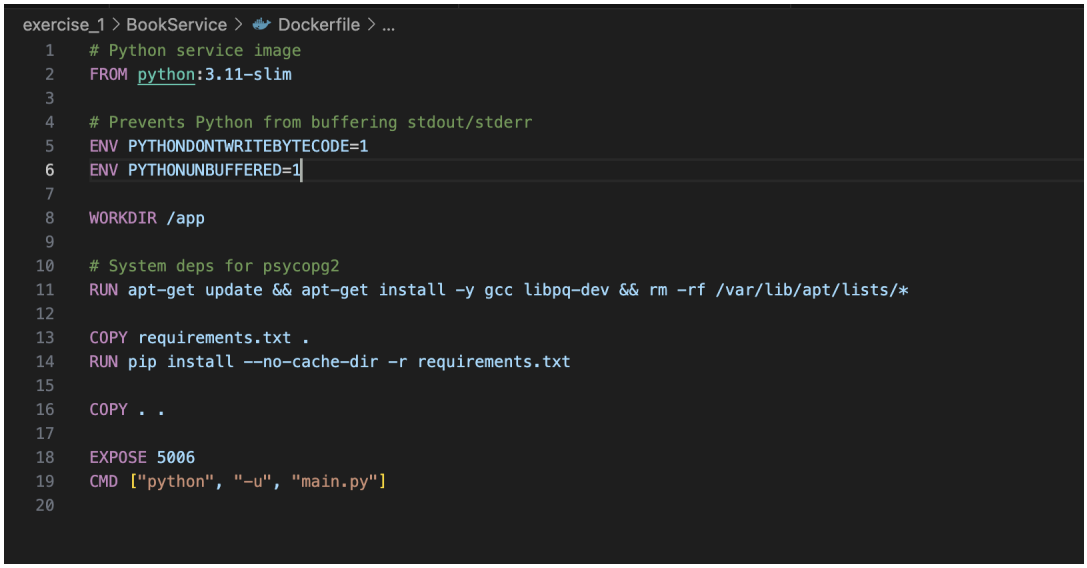
BookService's requirements.txt



```
Flask
requests
SQLAlchemy
flask_sqlalchemy
psycopg2-binary
pika
```

BookService's Dockerfile

Similar to UserService's Dockerfile except we **run this from port 5006**



docker-compose.yml

Now to integrate both the services, we create a docker-compose.yml file which has the following services- **UserService, BookService, a database called library_management_db** (based on a Postgres database) and set up the necessary network and volume for the database service.

- **library_management_db:**
  Runs Postgres 15 and creates a database called **library_management_db** with username and password. All the data is kept in the named volume **library_management_data**.
- **user_service:**
  Builds **an image from ./UserService** and **exposes port 5002** on machine to 5002 on container. It connects to **Postgres** using env variables and declares its dependencies on the DB container so that it starts first.
- **book_service:**
  Same as user_service but **built from ./BookService** and exposed on **port 5006**. It uses the same **Postgres DB** via the same connection variables.

The docker-compose.yml for exercise 1 can be found in the exercise_1 folder.

Steps to test the services and results:

We run the following command from inside exercise_1 folder -
docker compose up –build

The output of the command is as follows-

```
lakshitasejra@Lakshitas-MacBook-Air exercise_1 % docker compose up --build
WARN[0000] /Users/lakshitasejra/Cloud Computing Practicals/practical_8/exercise_1/docker-compose.yml: the attribute `version` is obsolete,
 it will be ignored, please remove it to avoid potential confusion
[+] Building 1.0s (21/21) FINISHED
 => [internal] load local bake definitions                                                                                    0.0s
 => => reading from stdin 1.21kB                                                                                              0.0s
 => [user_service internal] load build definition from Dockerfile                                                             0.0s
 => => transferring dockerfile: 444B                                                                                          0.0s
 => [book_service internal] load build definition from Dockerfile                                                             0.0s
 => => transferring dockerfile: 444B                                                                                          0.0s
 => [book_service internal] load metadata for docker.io/library/python:3.11-slim                                              0.7s
 => [book_service internal] load .dockerignore                                                                                0.0s
 => => transferring context: 2B                                                                                               0.0s
 => [user_service internal] load .dockerignore                                                                                0.0s
 => => transferring context: 2B                                                                                               0.0s
 => [book_service 1/6] FROM docker.io/library/python:3.11-slim@sha256:193fdd0bbcb3d2ae612bd6cc3548d2f7c78d65b549fcaa8af75624c474744  0.0s
 => => resolve docker.io/library/python:3.11-slim@sha256:193fdd0bbcb3d2ae612bd6cc3548d2f7c78d65b549fcaa8af75624c47474444d    0.0s
 => [user_service internal] load build context                                                                                0.0s
 => => transferring context: 94B                                                                                              0.0s
 => [book_service internal] load build context                                                                                0.0s
 => => transferring context: 94B                                                                                              0.0s
 => CACHED [user_service 2/6] WORKDIR /app                                                                                    0.0s
 => CACHED [user_service 3/6] RUN apt-get update && apt-get install -y gcc libpq-dev && rm -rf /var/lib/apt/lists/*           0.0s
 => CACHED [book_service 4/6] COPY requirements.txt .                                                                         0.0s
 => CACHED [book_service 5/6] RUN pip install --no-cache-dir -r requirements.txt                                              0.0s
 => CACHED [book_service 6/6] COPY . .                                                                                        0.0s
 => CACHED [user_service 4/6] COPY requirements.txt .                                                                         0.0s
 => CACHED [user_service 5/6] RUN pip install --no-cache-dir -r requirements.txt                                              0.0s
 => CACHED [user_service 6/6] COPY . .                                                                                        0.0s
 => [user_service] exporting to image                                                                                         0.1s
 => => exporting layers                                                                                                       0.0s
 => => exporting manifest sha256:4cae71f96033fc1c2f86ffa7ed1a2cf4dbe97d78e41485c5ca95eecdeff86115                             0.0s
 => => exporting config sha256:9a86d755e050381af1c6ca4758f953c57f05a492ce567e450132d282cff6baea                               0.0s
 => => exporting attestation manifest sha256:144634c8bb7c434cf37998af844779f7d224a2124f086036ea2c2f0198bac2f6                 0.0s
 => => exporting manifest list sha256:338c179c8d8fc1a186f53edafe6d38e3a8e5a3b476c11d75076495484ec961b2                        0.0s
 => => naming to docker.io/library/exercise_1-user_service:latest                                                             0.0s
 => => unpacking to docker.io/library/exercise_1-user_service:latest                                                          0.0s
 => [book_service] exporting to image                                                                                         0.1s
 => => exporting layers                                                                                                       0.0s
 => => exporting manifest sha256:93971011ddc173791202b37cafc7ac3aab94c0ce7c86b0b18f62efc781eca19d                             0.0s
 => => exporting config sha256:e7d68085ad946b3c5281853949e82e6bfe156186be708832264baa0966f2acd2                               0.0s
 => => exporting attestation manifest sha256:e41a48f42cbb973aa7b42127d665660719feff6378a8b5dd829a389e68693316                 0.0s
 => => exporting manifest list sha256:0da24a373a7501ca36bc6a3099448785fe9e874a774ac2ef4275f92095aa8aa2                         0.0s
 => => naming to docker.io/library/exercise_1-book_service:latest                                                             0.0s
 => => unpacking to docker.io/library/exercise_1-book_service:latest                                                          0.0s
 => [book_service] resolving provenance for metadata file                                                                     0.0s
 => [user_service] resolving provenance for metadata file                                                                     0.0s
WARN[0001] Found orphan containers ([library_db]) for this project. If you removed or renamed this service in your compose file, you can r
un this command with the --remove-orphans flag to clean it up.
[+] Running 5/5
 ✔ exercise_1-user_service          Built                                                                                     0.0s
 ✔ exercise_1-book_service          Built                                                                                     0.0s
 ✔ Container library_management_db  Created                                                                                   0.0s
 ✔ Container user_service           Created                                                                                   0.1s
 ✔ Container book_service           Created                                                                                   0.0s
Attaching to book_service, library_management_db, user_service
library_management_db  |
library_management_db  | PostgreSQL Database directory appears to contain a database; Skipping initialization
library_management_db  |
```

```
[+] Running 5/5
 ✔ exercise_1-user_service          Built                                                                                     0.0s
 ✔ exercise_1-book_service          Built                                                                                     0.0s
 ✔ Container library_management_db  Created                                                                                   0.0s
 ✔ Container user_service           Created                                                                                   0.1s
 ✔ Container book_service           Created                                                                                   0.0s
Attaching to book_service, library_management_db, user_service
library_management_db  |
library_management_db  | PostgreSQL Database directory appears to contain a database; Skipping initialization
library_management_db  |
library_management_db  | 2025-11-28 12:28:20.057 UTC [1] LOG:  starting PostgreSQL 15.15 (Debian 15.15-1.pgdg13+1) on aarch64-unknown-linu
x-gnu, compiled by gcc (Debian 14.2.0-19) 14.2.0, 64-bit
library_management_db  | 2025-11-28 12:28:20.058 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
library_management_db  | 2025-11-28 12:28:20.058 UTC [1] LOG:  listening on IPv6 address "::", port 5432
library_management_db  | 2025-11-28 12:28:20.062 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
library_management_db  | 2025-11-28 12:28:20.066 UTC [29] LOG:  database system was shut down at 2025-11-21 01:38:54 UTC
library_management_db  | 2025-11-28 12:28:20.070 UTC [1] LOG:  database system is ready to accept connections
book_service           |  * Serving Flask app 'main'
book_service           |  * Debug mode: off
user_service           |  * Serving Flask app 'main'
user_service           |  * Debug mode: off


book_service           | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server ins
tead.
user_service           | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server ins
tead.


user_service           |  * Running on all addresses (0.0.0.0)

book_service           |  * Running on all addresses (0.0.0.0)


user_service           |  * Running on http://127.0.0.1:5002

book_service           |  * Running on http://127.0.0.1:5006



user_service           |  * Running on http://172.26.0.3:5002
book_service           |  * Running on http://172.26.0.4:5006

user_service           | Press CTRL+C to quit


book_service           | Press CTRL+C to quit
```

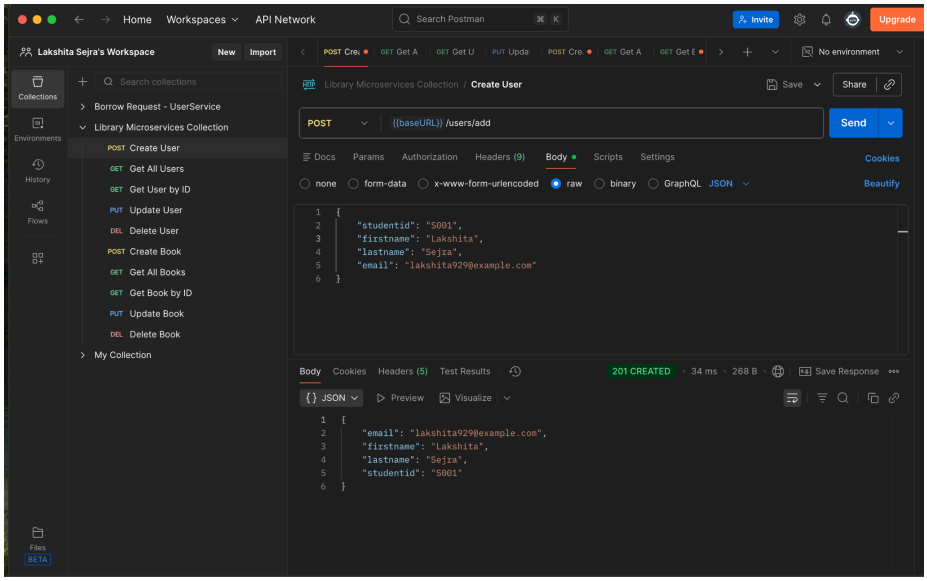
To test the endpoints of the UserService and BookService, we **install Postman and use that to test the api endpoints**.

For **UserService we test 5 api endpoints on port 5002**-

1) **Create User (POST):**
   Method- POST
   URL- http://localhost:5002/users/add
   Body (JSON)-

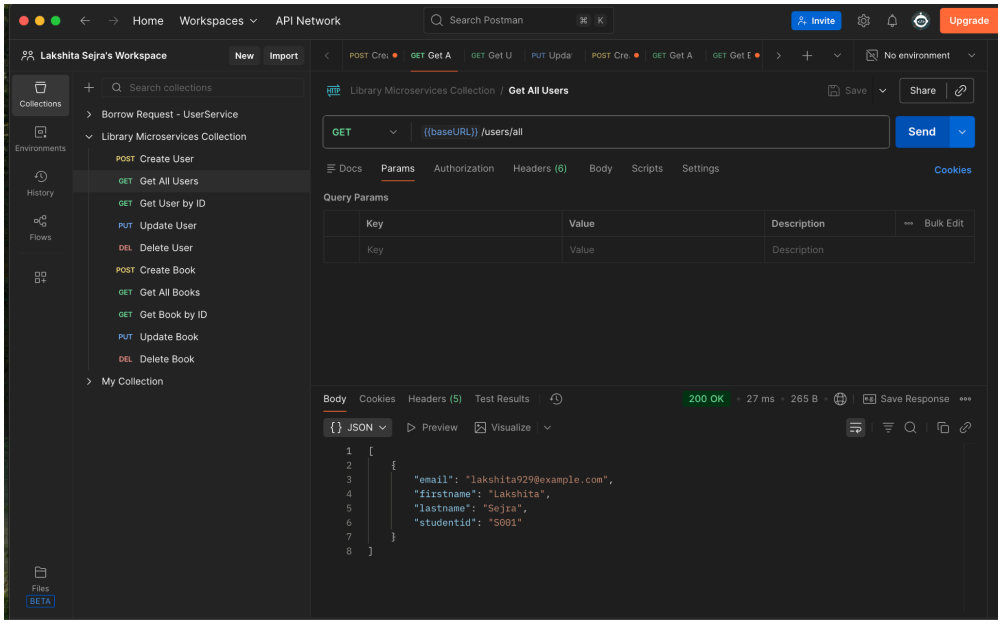
```
{
   "studentid": "S001",
   "firstname": "Lakshita",
   "lastname": "Sejra",
   "email": "lakshita929@example.com"
}
```

   The output on sending the POST api call is (HTTP 201 created)-



2) **Get All Users (GET):**
   Method- GET
   URL- http://localhost:5002/users/all

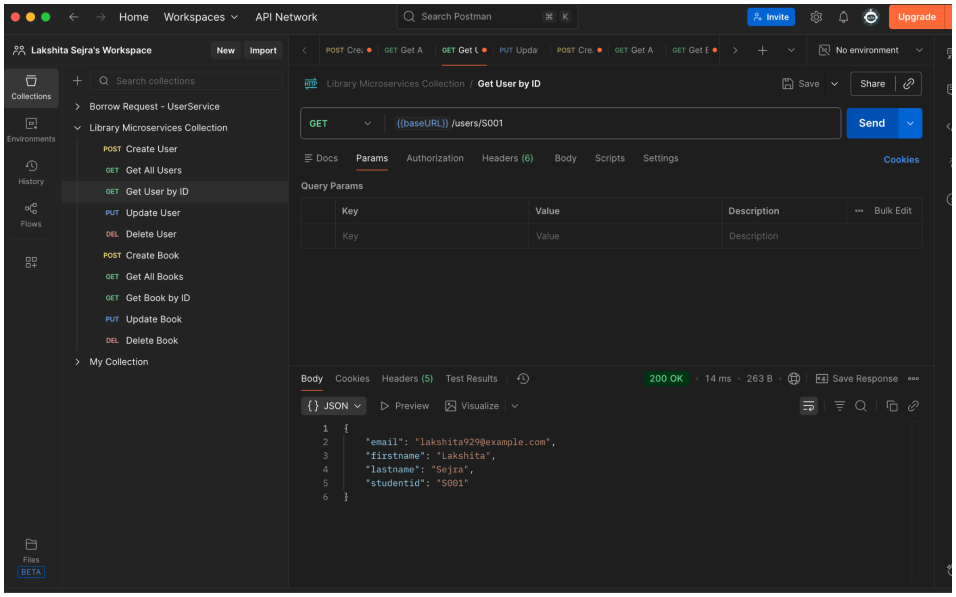
   The output of sending the GET api call is (HTTP 202 OK)-



We get a JSON array containing the previously created users (in the screenshot)

3) **GET User by ID (GET):**
   Method- GET
   URL- http://localhost:5002/users/S001

   The output of sending the GET api call for user ID S001 is (HTTP 202 OK)-



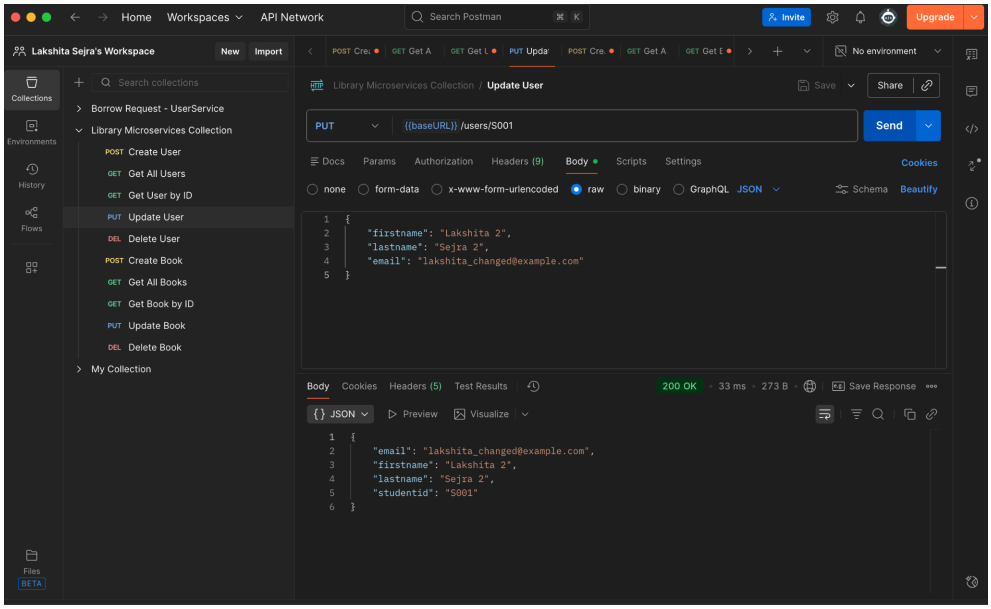
We get a JSON array containing the details for user ID S001 (in the screenshot)

4) **Update User (PUT):**
   Method- PUT

URL- http://localhost:5002/users/S001
Body (JSON)-

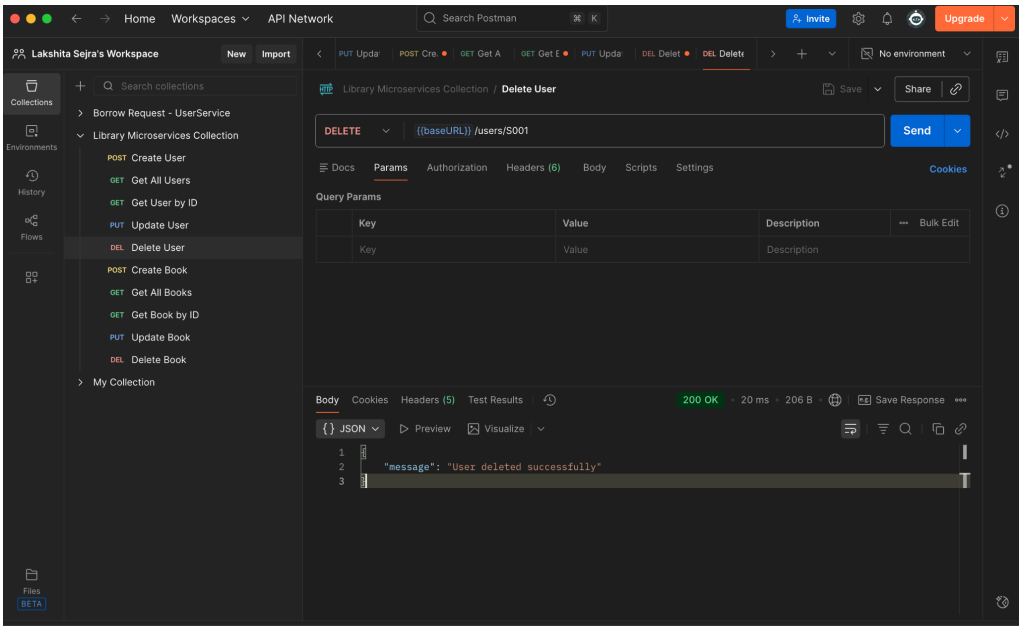
```
{
   "firstname": "Lakshita 2",
   "lastname": "Sejra 2",
   "email": "lakshita_changed@example.com"
}
```

The output of sending the PUT api call for user ID S001 is (HTTP 200 OK)-



5) **Delete User (DELETE):**
   Method- DELETE
   URL- http://localhost:5002/users/S001

The output of sending the DELETE api call for user ID S001 is (HTTP 200 OK)-

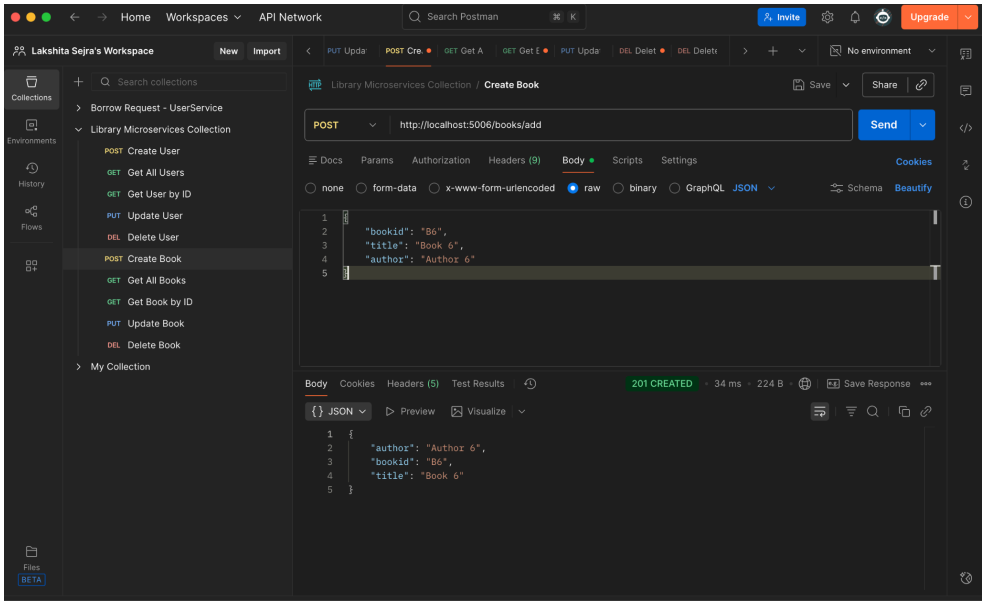


For **BookService we test 5 api endpoints on port 5006**-

6) **Create Book (POST):**
   Method- POST
   URL- http://localhost:5006/books/add
   Body (JSON)-

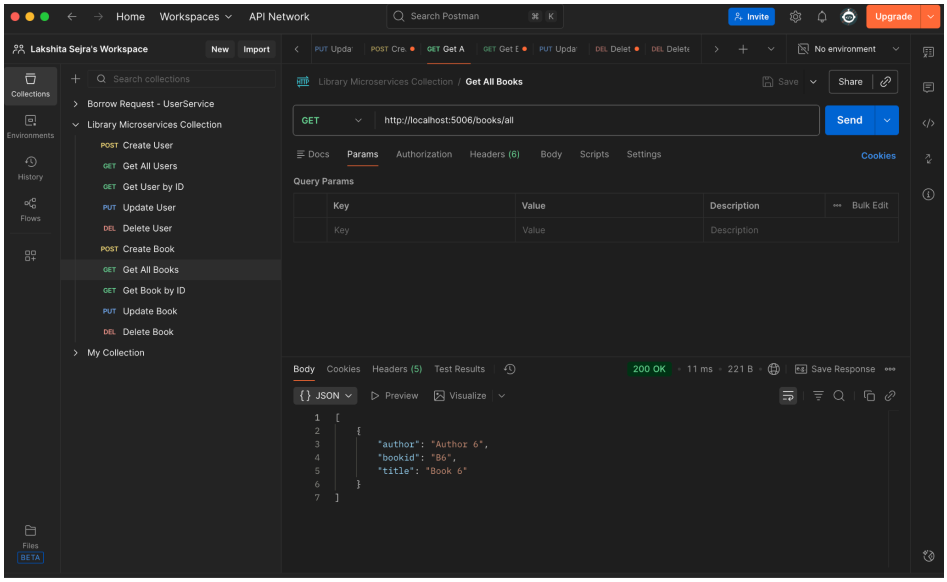
```
{
    "bookid": "B6",
    "title": "Book 6",
    "author": "Author 6"
}
```

   The output on sending the POST api call is (HTTP 201 created)-



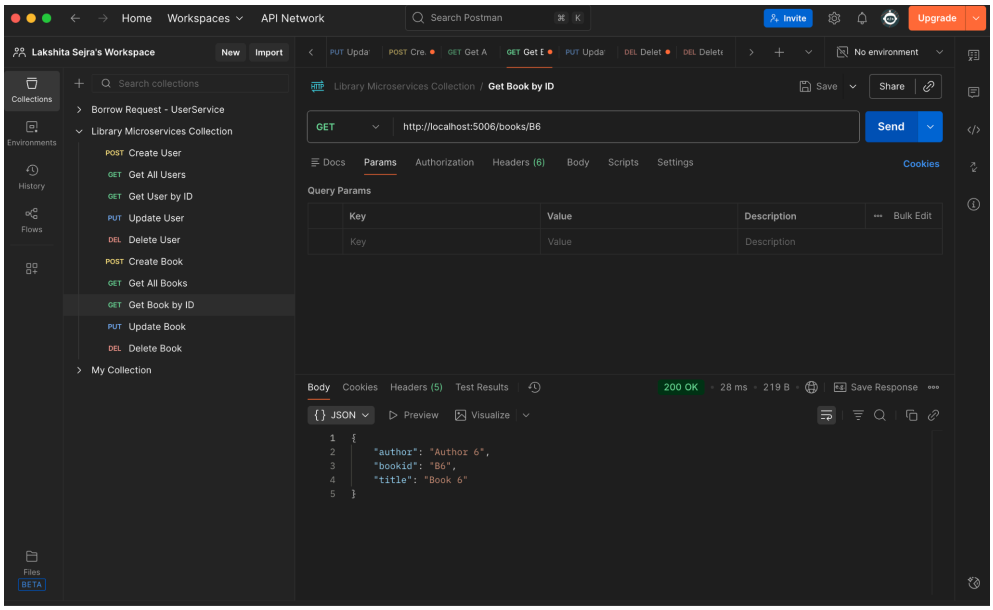
7) **Get All Books (GET):**
   Method- GET
   URL- http://localhost:5006/books/all

   The output of sending the GET api call is (HTTP 202 OK)-



8) **GET Book by ID (GET):**
Method- GET
URL- http://localhost:5006/books/B6

The output of sending the GET api call for user ID B6 is (HTTP 202 OK)-

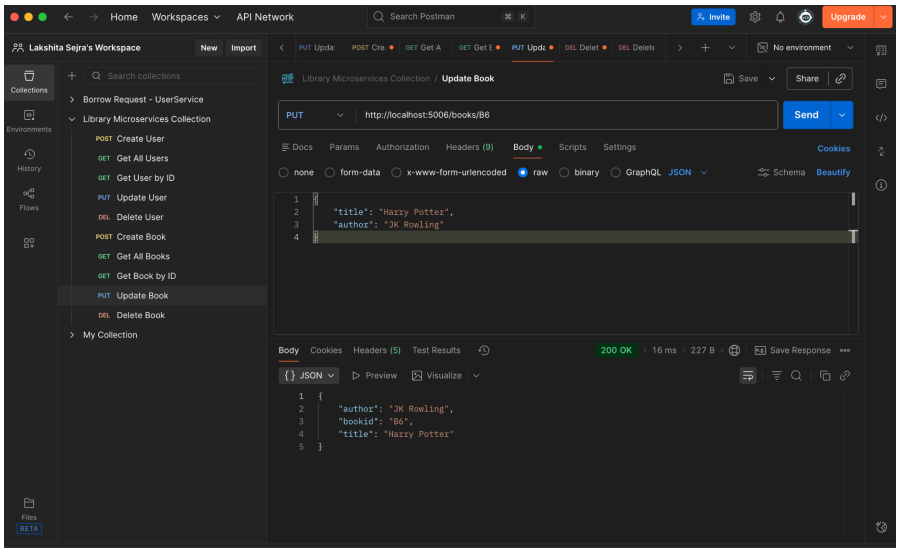


9) **Update Book (PUT):**
Method- PUT
URL- http://localhost:5006/books/B6
Body (JSON)-

```
{
   "title": "Harry Potter",
```

```
    "author": "JK Rowling"
}
```

The output of sending the PUT api call for Book ID B6 is (HTTP 200 OK)-

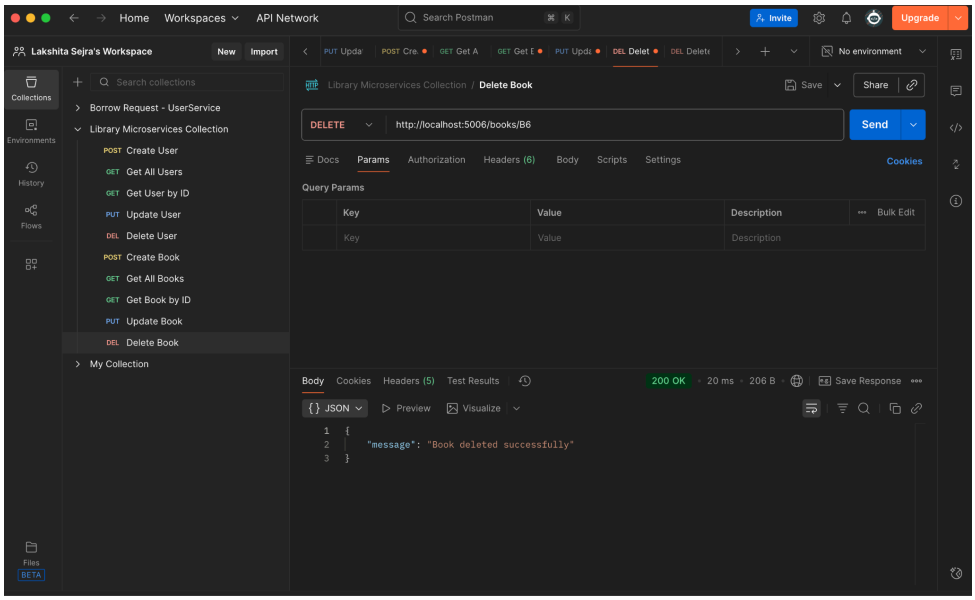


10) Delete Book (DELETE):

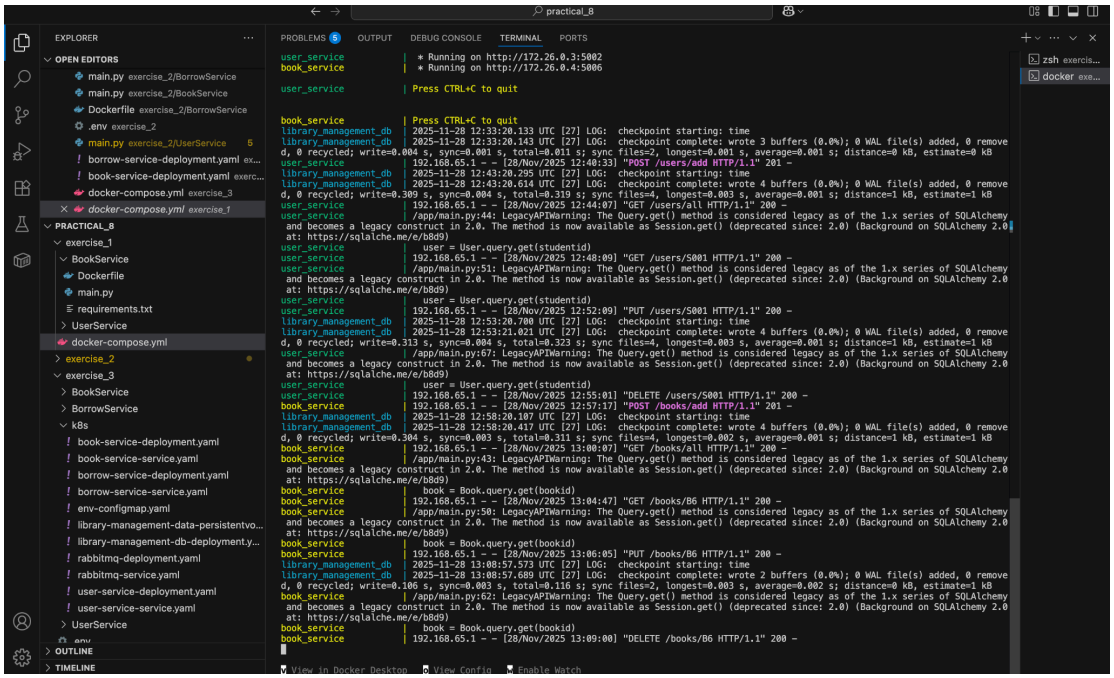
Method- DELETE

URL- http://localhost:5006/books/B6

The output of sending the DELETE api call for book ID B6 is (HTTP 200 OK)-



The output of the terminal after testing all these api endpoints is as follows:



Exercise 2 (Asynchronous communication with RabbitMQ):

We copy the exercise 1 files into a new folder called exercise 2 and set up a new service called **BorrowService.**

The **BorrowService** folder consists of the following files (The code for all these files can be found in the code folder under exercise_2/BorrowService) -

1) **main.py** -
    - It reads the **DB connection settings** from the env variables and configures SQLAlchemy.
    - We define **Borrow table** with id, student_id and book_id and create it
    - It then reads **RabbitMQ config settings**- RabbitMQ user, password, host and port from **env variables** (RABBITMQ_*)
    - We define 2 helper functions- **student_exists(student_id) and book_exists(book_id)** for validating the ids by calling the UserService and BookService api calls respectively.
    - **process_borrow_request**: When a message arrives at the borrow_book queue, we parse the JSON, get student_id and book_id and run inside app.app_context(). It **validates the student and book id using the above helpers, and checks if the student doesn't have more than 5 borrowed books.** If all the conditions are met, it **inserts a new Borrow row in DB** and commits.
    - **RabbitMQ Listener**- We connect to RabbitMQ using host/port and username/password (taken from env variables). It then opens a channel and makes sure the borrow_book queue exists otherwise it creates it. When a message arrives at borrow_book, the loop reads the messages from RabbitMQ and passes them to process_borrow_request function (which we described in the above pointer)
2) **Dockerfile**-
    - The Dockerfile uses **python:3.9-slim as the image** and **creates the work directory as /app** where we **copy all the app source code** and **run all the commands**.
    - We first update the apt package index, install gcc and libpq-dev (for using psycopg2 which is Postgres driver for Python).
    - We then **install the requirements and run** using requirements.txt into /app and copy the BorrowService code.
    - We then **run the app on port 5004** and when the container starts, we run the command "**python -u main.py" in Dockerfile**.

docker-compose.yml (for exercise 2):

This docker-compose.yml file consists of **Postgres Database, 3 microservices (UserService, BookService and BorrowService) and RabbitMQ** all in **one network (library_management_network)**

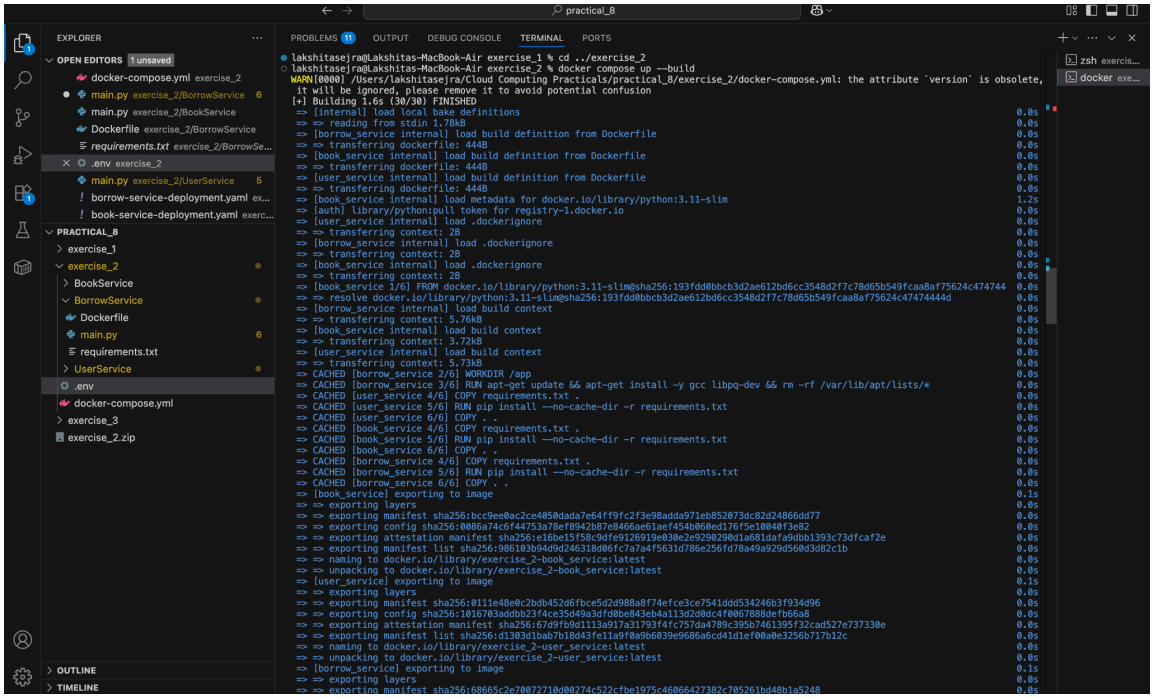
- **Postgres DB (library_management_db)**: Runs **Postgres 15** and creates a database called **library_management_db** with username and password. All the data is kept in the named volume **library_management_data**.
- **RabbitMQ:** Uses **rabbitmq:3.13-rc-management** image and uses **port 5672** and **web UI on 15672**. The credentials are taken from the **.env file**.
- **user_service:**
  Builds **an image from ./UserService** and **exposes port 5002** on machine to 5002 on container. It connects to **Postgres** using **env variables** and also connects to **RabbitMQ host**. It declares its dependencies on the DB container and RabbitMQ so that they start first.
- **book_service:**
  Same as user_service but **built from ./BookService** and exposed on **port 5006**. It uses the same **Postgres DB** via the same connection variables and depends on Postgres DB.
- **borrow_service:**
  Builds **an image from ./BorrowService** and **exposes port 5004**. It connects to **Postgres** using **env variables** and also connects to **RabbitMQ host**. It declares its **dependencies** on the DB container and RabbitMQ so that they start first.

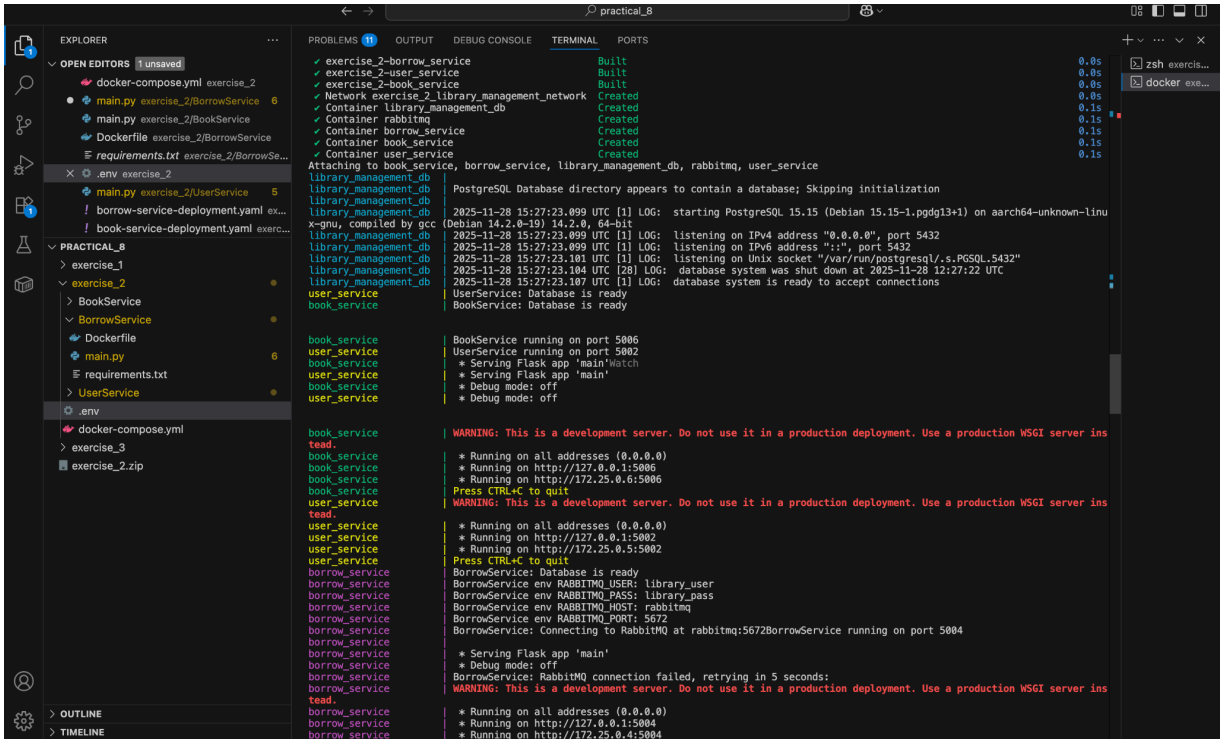
The **.env file** used is as follows-

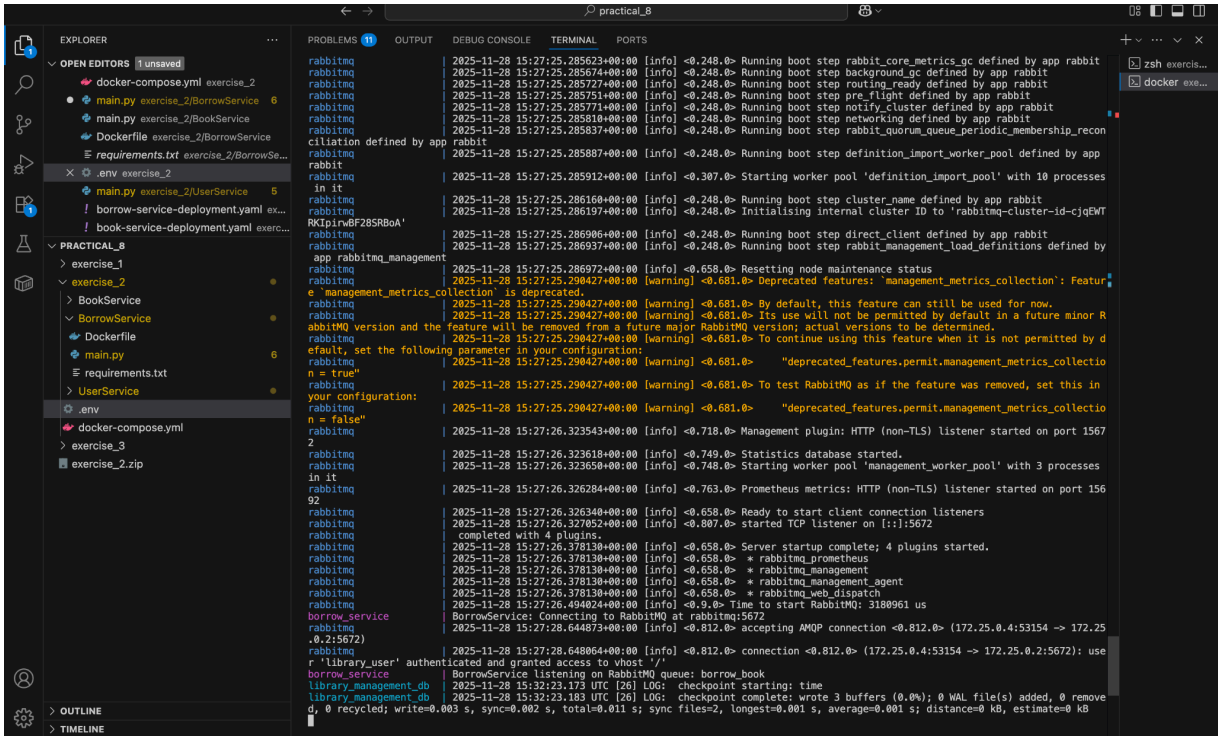
```
# RabbitMQ credentials (Exercise Two requirement)
RABBITMQ_DEFAULT_USER=library_user
RABBITMQ_DEFAULT_PASS=library_pass

# Optional: keep these here for clarity
RABBITMQ_HOST=rabbitmq
RABBITMQ_PORT=5672
```

To run and test the services:

1) Run **docker compose up –build** (from inside exercise_2 folder)
   The output of the command is as follows-


practical_8
EXPLORER
OPEN EDITORS 1 unsaved
docker-compose.yml exercise_2
main.py exercise_2/BorrowService 6
main.py exercise_2/BookService
Dockerfile exercise_2/BorrowService
requirements.txt exercise_2/BorrowSe...
.env exercise_2
main.py exercise_2/UserService 5
borrow-service-deployment.yaml ex...
book-service-deployment.yaml exerc...
PRACTICAL_8
exercise_1
exercise_2
BookService
BorrowService
Dockerfile
main.py 6
requirements.txt
UserService
.env
docker-compose.yml
exercise_3
exercise_2.zip
OUTLINE
TIMELINE
PROBLEMS 11
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
lakshitasejra@Lakshitas-MacBook-Air exercise_1 % cd ../exercise_2
lakshitasejra@Lakshitas-MacBook-Air exercise_2 % docker compose up --build
WARN[0000] /Users/lakshitasejra/Cloud Computing Practicals/practical_8/exercise_2/docker-compose.yml: the attribute `version` is obsolete,
it will be ignored, please remove it to avoid potential confusion
[+] Building 1.6s (30/30) FINISHED
zsh exercis...
docker exe...


practical_8
EXPLORER
OPEN EDITORS 1 unsaved
PRACTICAL_8
OUTLINE
TIMELINE
PROBLEMS 11
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
zsh exercis...
docker exe...
Attaching to book_service, borrow_service, library_management_db, rabbitmq, user_service
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
Press CTRL+C to quit
BorrowService: Database is ready
BorrowService: RabbitMQ connection failed, retrying in 5 seconds:



2) **BorrowService API endpoint testing using Postman:**
   - **Send Borrow Request (RabbitMQ):**

Method- POST

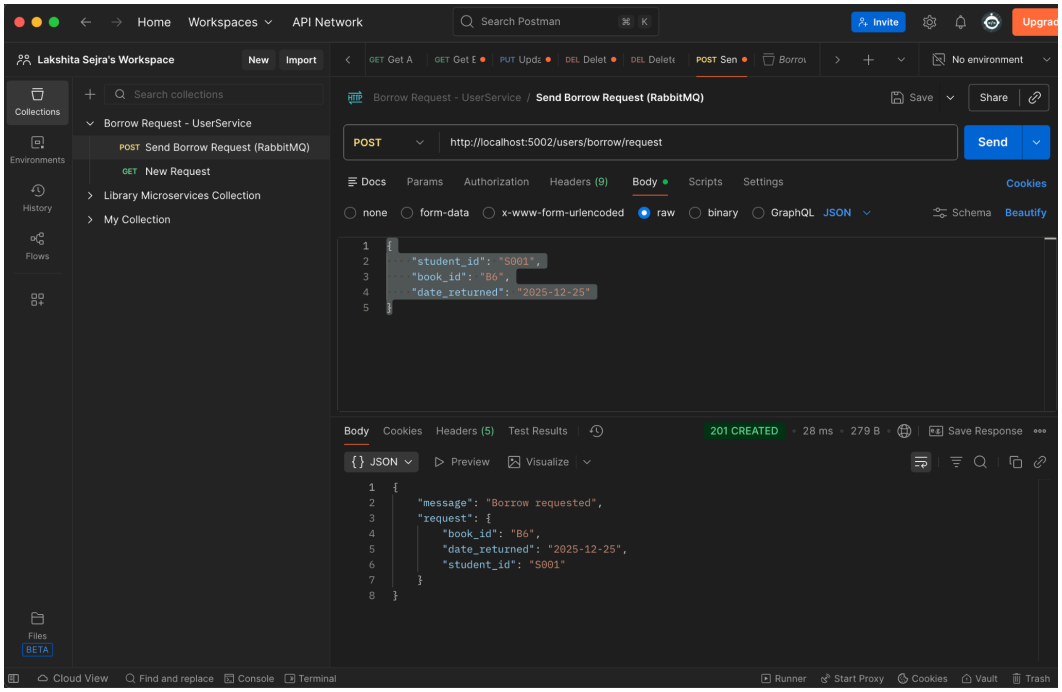
URL- http://localhost:5002/users/borrow/request

Body-

```
{
    "student_id": "S001",
    "book_id": "B6",
    "date_returned": "2025-12-25"
}
```

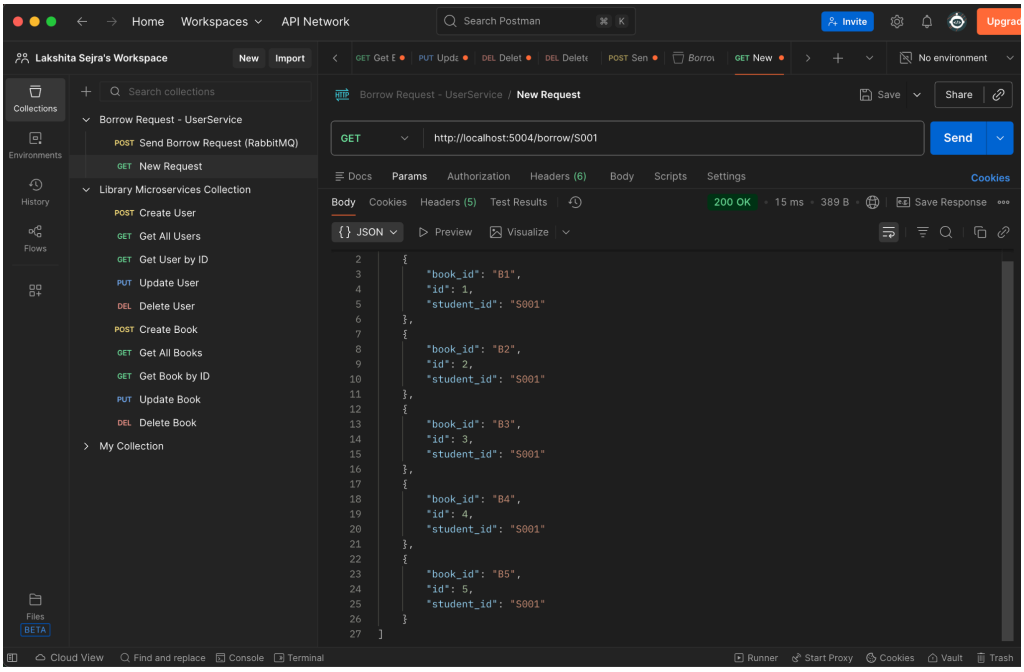
We do this similar Send Borrow Request for 6 Books B1 to B6 for one student id S001. We do so in order to test if the student gets to borrow more than 5 books.

Output-

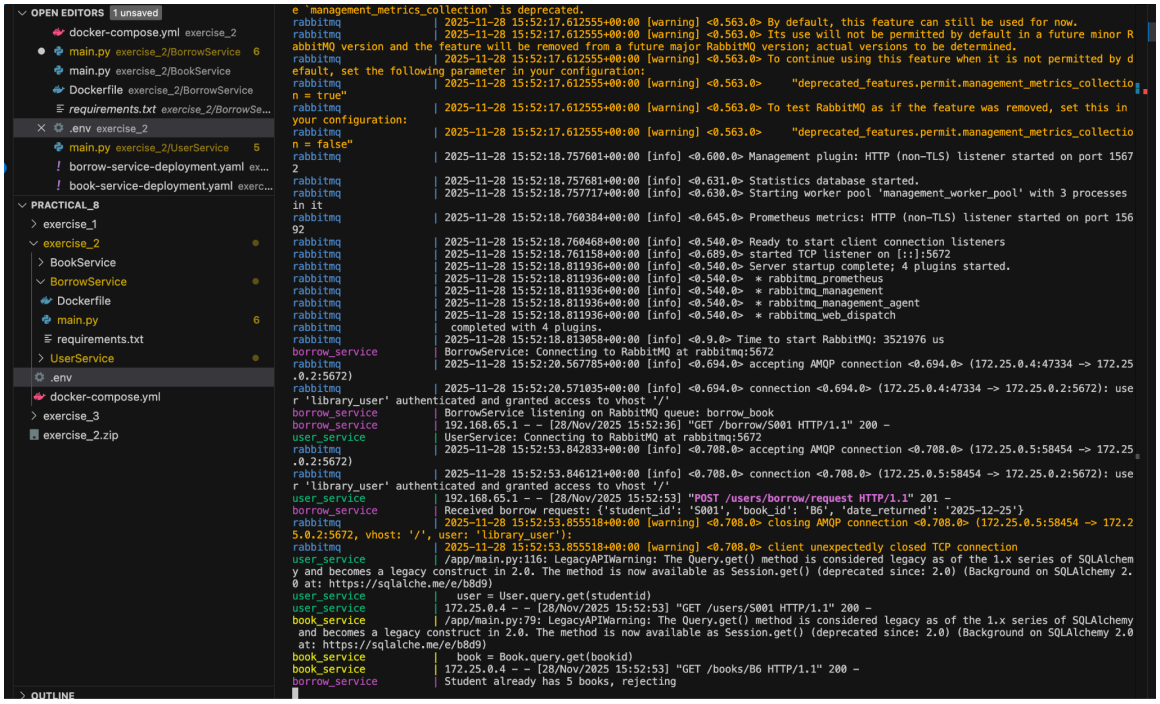


- **Borrow Request (GET)**
  Method- GET
  URL- http://localhost:5004/borrow/S001

  Output-
  <u>More than 5 books condition check-</u> We see that although we made **6 borrow requests for the student S001**, when we try to get the output of the borrowed request, **we see only 5 books** because we have the **condition that more than 5 books are not allowed to be borrowed for one user id.**



The terminal output for the same GET request is as follows-



We can see that the GET request is **B6** (6th book) for student id S001 is **rejected** with the message- **“Student already has 5 books, rejecting”**

Exercise 3: Deployment with Kubernetes

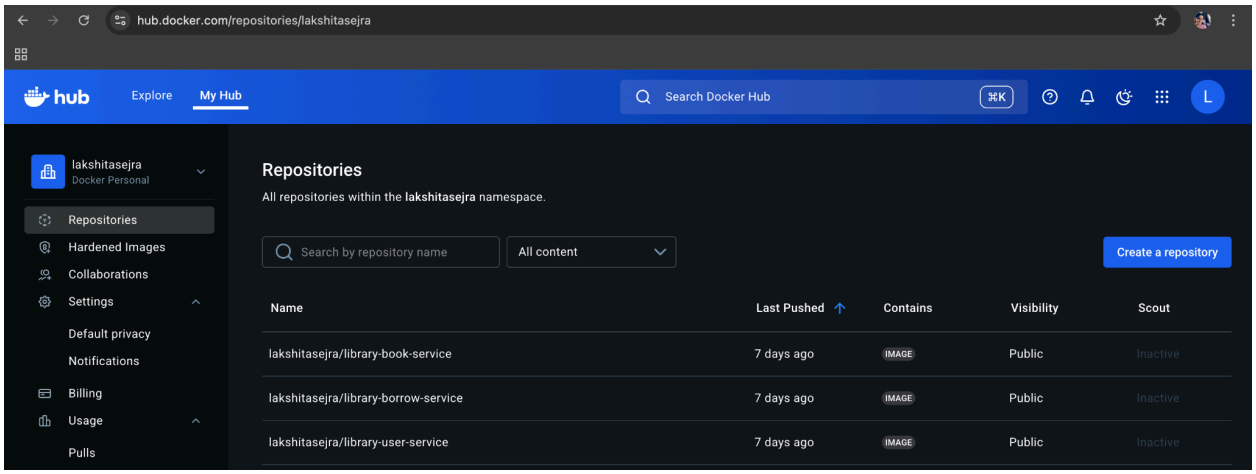
We copy the files and folders from exercise 2 to exercise 3 folder.

1) Publishing docker images to Docker Hub:

We build and push the docker images for **3 microservices (UserService, BookService and BorrowService) to Docker Hub** using the following commands-

```
# Build images
docker build -t lakshitasejra/library-user-service:exercise3 ./UserService
docker build -t lakshitasejra/library-book-service:exercise3 ./BookService
docker build -t lakshitasejra/library-borrow-service:exercise3 ./BorrowService

# Login and push
docker login
docker push lakshitasejra/library-user-service:exercise3
docker push lakshitasejra/library-book-service:exercise3
docker push lakshitasejra/library-borrow-service:exercise3
```

The images are successfully published to docker hub-



2) Modifying the docker-compose.yml:

In the docker-compose.yml file, we make the modification for each microservice to read the image from docker hub-

For UserService -> image: lakshitasejra/library-user-service:exercise3
For BookService -> image: lakshitasejra/library-book-service:exercise3
For BorrowService -> image: lakshitasejra/library-borrow-service:exercise3

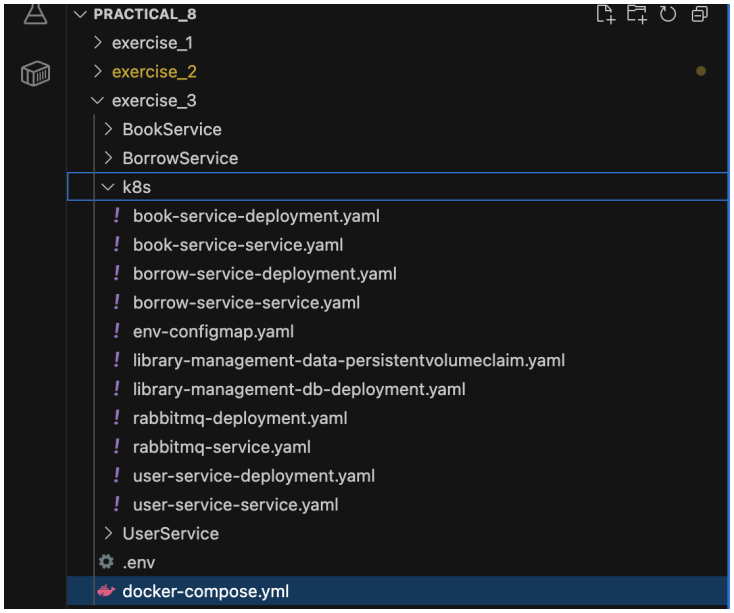
The rest of the docker-compose.yml file remains the same.

3) **Generating Kubernetes manifests from docker-compose.yml using Kompose:**

We first create a **folder named k8s** where all the Kubernetes manifests are stored.

Then we run the following kompose command to generate the manifests inside k8s-
kompose convert -f docker-compose.yml -o k8s/

This creates Deployments, Services, PVCs, ConfigMap files inside k8s.



4) **Apply everything to Kubernetes:**

We apply everything to Kubernetes using the following command-
kubectl apply -f k8s/

We then run the following commands to get pods and svc:

kubectl get pods
Output of the above command is:

```
lakshitasejra@Lakshitas-MacBook-Air exercise_3 % kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
book-service-6ddb94fbcf-pcms9         1/1     Running   0          6d14h
borrow-service-6746f69795-f9n9r       1/1     Running   0          6d14h
library-management-db-59d67fdc97-p8j8p   1/1     Running   0          6d14h
rabbitmq-5b7787d786-jb788             1/1     Running   0          6d15h
user-service-69cdb99f47-vm584         1/1     Running   0          6d14h
lakshitasejra@Lakshitas-MacBook-Air exercise_3 %
```

kubectl get svc

Output of the above command is:

```
lakshitasejra@Lakshitas-MacBook-Air exercise_3 % kubectl get svc
NAME                    TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)              AGE
book-service            ClusterIP   10.107.192.221   <none>        5006/TCP             6d15h
borrow-service          ClusterIP   10.106.166.245   <none>        5004/TCP             6d15h
kubernetes              ClusterIP   10.96.0.1        <none>        443/TCP              6d15h
library-management-db   ClusterIP   10.103.164.227   <none>        5432/TCP             6d14h
rabbitmq                ClusterIP   10.108.112.92    <none>        5672/TCP,15672/TCP   6d15h
user-service            ClusterIP   10.111.77.167    <none>        5002/TCP             6d15h
lakshitasejra@Lakshitas-MacBook-Air exercise_3 %
```

5) **Port Forward from Services to Host:**

We run 3 terminals in parallel for 3 microservices. For each service, we port forward to host using the following commands-

For UserService

kubectl port-forward svc/user-service 5002:5002

```
lakshitasejra@Lakshitas-MacBook-Air exercise_3 % kubectl port-forward svc/user-service 5002:5002
Forwarding from 127.0.0.1:5002 -> 5002
Forwarding from [::1]:5002 -> 5002
```

For BookService

kubectl port-forward svc/book-service 5006:5006

```
lakshitasejra@Lakshitas-MacBook-Air exercise_3 % kubectl port-forward svc/book-service 5006:5006
Forwarding from 127.0.0.1:5006 -> 5006
Forwarding from [::1]:5006 -> 5006
```

For BorrowService

kubectl port-forward svc/borrow-service 5004:5004

```
lakshitasejra@Lakshitas-MacBook-Air exercise_3 % kubectl port-forward svc/borrow-service 5004:5004
Forwarding from 127.0.0.1:5004 -> 5004
Forwarding from [::1]:5004 -> 5004
```

6) **Testing each of the port forwarded services using Postman:**

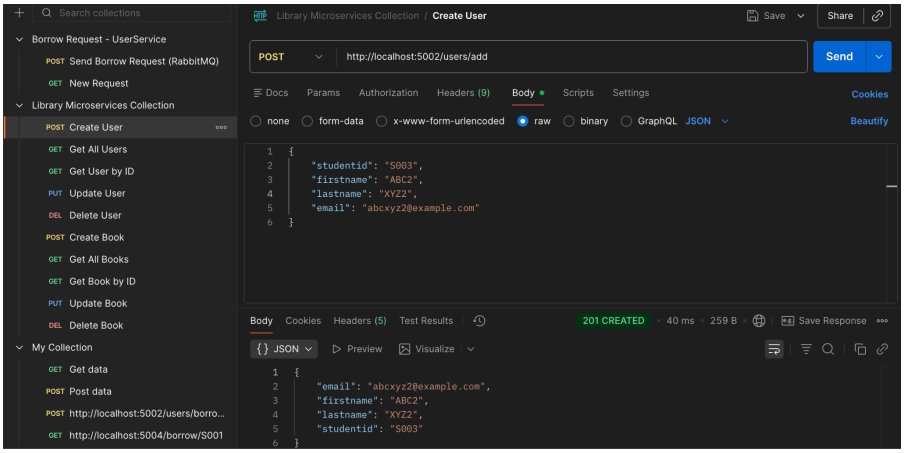
- **Create User (UserService)**

  POST http://localhost:5002/users/add

  Body (JSON):

```
{
   "studentid": "S003",
   "firstname": "ABC2",
   "lastname": "XYZ2",
   "email": "abcxyz2@example.com"
}
```

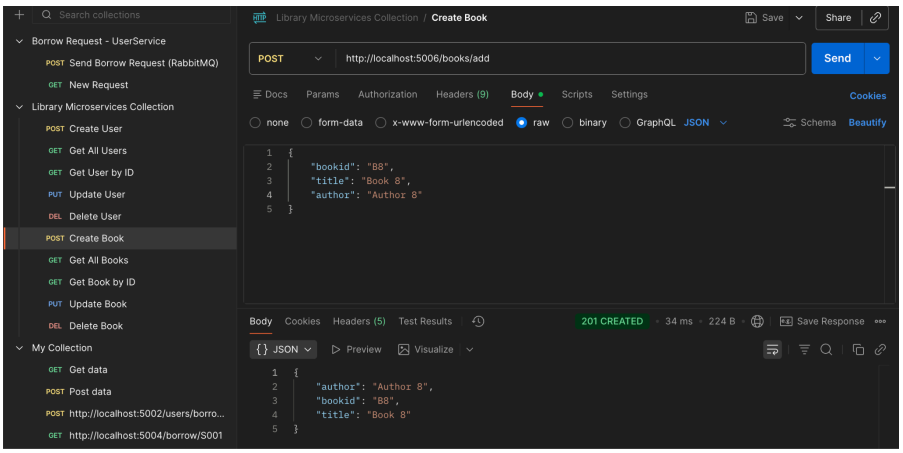
Postman Output (201 CREATED):



- **Create Book (BookService)**
  POST http://localhost:5006/books/add
  Body (JSON):

```
{
    "bookid": "B8",
    "title": "Book 8",
    "author": "Author 8"
}
```

Postman Output (201 CREATED):

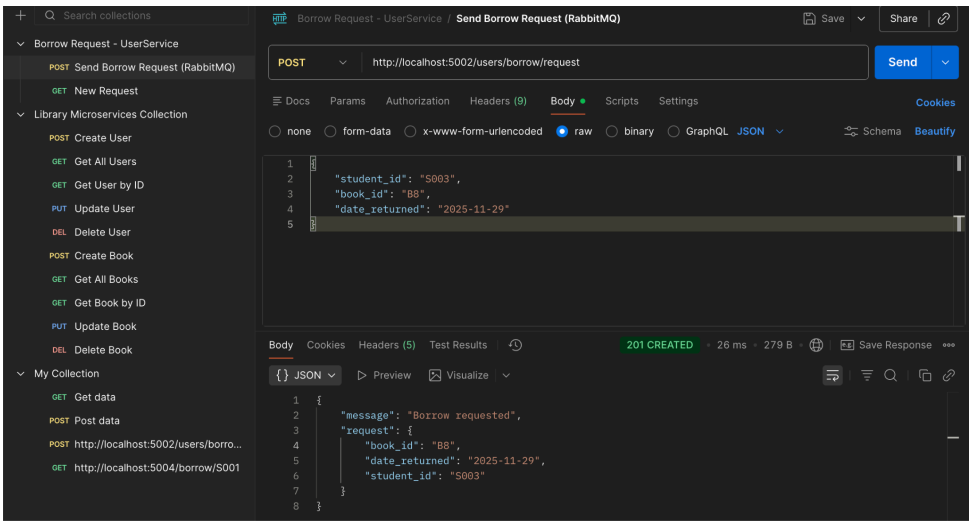


- **Send Borrow Request (UserService -> RabbitMQ -> BorrowService)**
  POST http://localhost:5002/users/borrow/request
  Body (JSON):

```
{
```

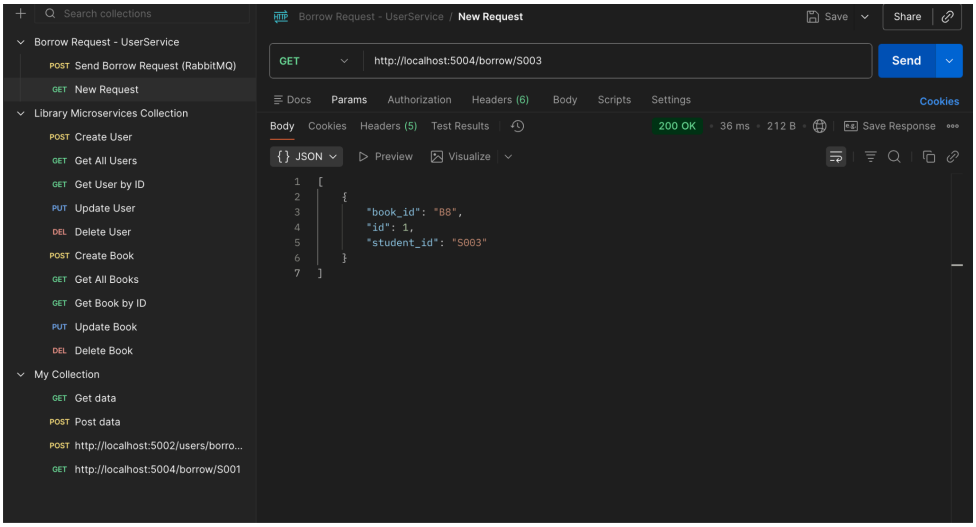
```
  "student_id": "S003",
  "book_id": "B8",
  "date_returned": "2025-11-29"
}
```

Postman Output (201 CREATED):

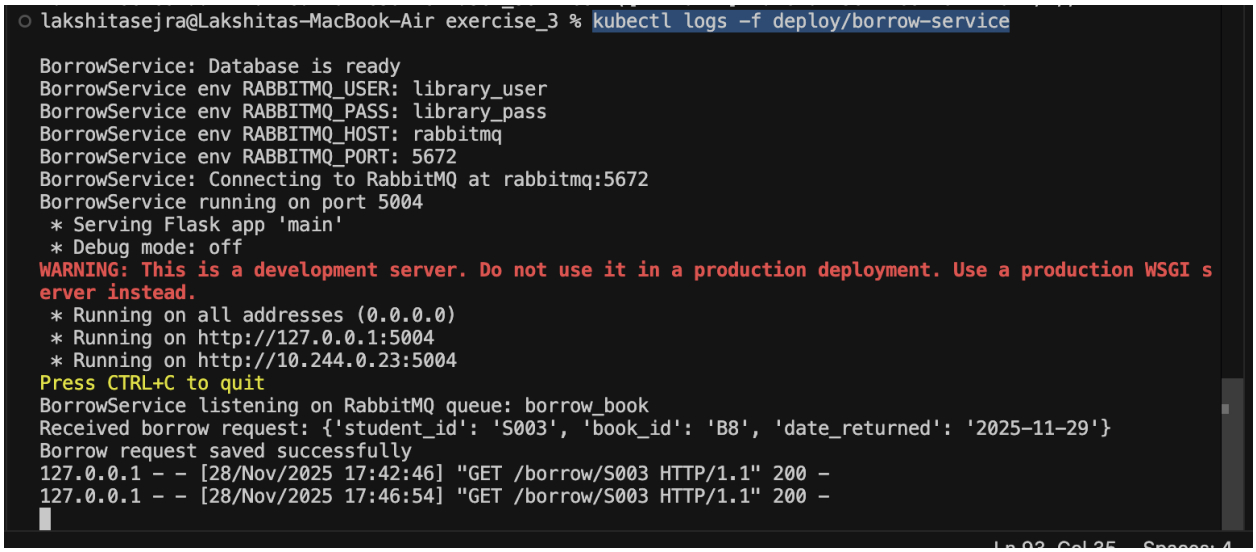


- **Check borrowed books (BorrowService):**
  GET http://localhost:5004/borrow/S003

Postman Output (200 OK):



Kubectl logs check for borrow-service:
Running kubectl logs -f deploy/borrow-service



All 3 of our microservices - user-service, book-service, borrow-service are running completely fine on Kubernetes cluster and we have successfully tested them as well using Postman.