

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 ->

```
exercise_1 > UserService > Dockerfile > ...
1  # Python service image
2  FROM python:3.11-slim
3
4  # Prevents Python from buffering stdout/stderr
5  ENV PYTHONUNBUFFERED=1
6  ENV PYTHONUNBUFFERED=1
7
8  WORKDIR /app
9
10 # System deps for psycopg2
11 RUN apt-get update && apt-get install -y gcc libpq-dev && rm -rf /var/lib/apt/lists/*
12
13 COPY requirements.txt .
14 RUN pip install --no-cache-dir -r requirements.txt
15
16 COPY . .
17 |
18 EXPOSE 5002
19 CMD ["python", "-u", "main.py"]
20
```

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)

```
exercise_1 > BookService > main.py > ...
16   class Book(db.Model):
17     __tablename__ = 'books'
18     bookid = db.Column(db.String(20), primary_key=True)
19     title = db.Column(db.String(50), nullable=False)
20     author = db.Column(db.String(50), nullable=False)
21
22     def to_dict(self):
23       return dict(bookid=self.bookid, title=self.title, author=self.author)
24
25 with app.app_context():
26   db.create_all()
27
28 @app.route('/books/add', methods=['POST'])
29 def create_book():
30   data = request.get_json(force=True)
31   book = Book(**data)
32   db.session.add(book)
33   db.session.commit()
34   return jsonify(book.to_dict()), 201
35
36 @app.route('/books/all', methods=['GET'])
37 def get_books():
38   books = Book.query.all()
39   return jsonify([u.to_dict() for u in books]), 200
40
41 @app.route('/books/<bookid>', methods=['GET'])
42 def get_book(bookid):
43   book = Book.query.get(bookid)
44   if not book:
45     return jsonify({"error": "Book not found"}), 404
46   return jsonify(book.to_dict()), 200
47
48 @app.route('/books/<bookid>', methods=['PUT'])
49 def update_book(bookid):
50   book = Book.query.get(bookid)
51   if not book:
52     return jsonify({"error": "Book not found"}), 404
53   data = request.get_json(force=True)
54   if 'title' in data: book.title = data['title']
55   if 'author' in data: book.author = data['author']
56
57   db.session.commit()
58   return jsonify(book.to_dict()), 200
59
60 @app.route('/books/<bookid>', methods=['DELETE'])
```

BookService's requirements.txt

```
exercise_1 > BookService > requirements.txt
1  Flask
2  requests
3  SQLAlchemy
4  flask_sqlalchemy
5  psycopg2-binary
6  pika
7
```

BookService's Dockerfile

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

```
exercise_1 > BookService > 📄 Dockerfile > ...
1 # Python service image
2 FROM python:3.11-slim
3
4 # Prevents Python from buffering stdout/stderr
5 ENV PYTHONDONTWRITEBYTECODE=1
6 ENV PYTHONUNBUFFERED=1
7
8 WORKDIR /app
9
10 # System deps for psycopg2
11 RUN apt-get update && apt-get install -y gcc libpq-dev && rm -rf /var/lib/apt/lists/*
12
13 COPY requirements.txt .
14 RUN pip install --no-cache-dir -r requirements.txt
15
16 COPY . .
17
18 EXPOSE 5006
19 CMD ["python", "-u", "main.py"]
20
```

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@Lakshita-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 .dockignore 0.0s
--> => transferring context: 2B 0.0s
--> [user_service internal] load .dockignore 0.0s
--> => transferring context: 2B 0.0s
--> [book_service internal] load build definition from Dockerfile 0.0s
--> => resolving docker.io/library/python:3.11-slim@sha256:193fd0bbc3d2ae612bd6cc3548d2f7c78d65b549fcfaa8af75624c4747444d 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 sha256:4cae71f96033fc1c2f86ffa7ed1a2cf4dbe97d78e41485c5ca95eeceff86115 0.0s
--> => exporting config sha256:a986d75e050381af1cc6ca4758f953c57f05a492ce567e450132d282cff6baea 0.0s
--> => exporting attestation manifest sha256:144634cbb7c434cf37998a7844779f7d22424124f086036ea2c2f0198bac2f6 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 sha256:93971011ddc173791202b37caf73acaa94c0ce7c86b0b18f62efc781eca19d 0.0s
--> => exporting config sha256:7d68085ad946b3c5281853949e82e6bfef156186be708832264baa0966f2acd2 0.0s
--> => exporting attestation manifest sha256:e41a48f42ccb973aa7b42127d665660719eff6378ab5dd0829a389e68693316 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 run 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 | PostgreSQL Database directory appears to contain a database; Skipping initialization
library_management_db | library_management_db | PostgreSQL Database directory appears to contain a database; Skipping initialization
library_management_db | library_management_db | library_management_db | PostgreSQL Database directory appears to contain a database; Skipping initialization

```

```

[+] 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 | PostgreSQL Database directory appears to contain a database; Skipping initialization
library_management_db | library_management_db | PostgreSQL Database directory appears to contain a database; Skipping initialization
library_management_db | library_management_db | library_management_db | PostgreSQL Database directory appears to contain a database; Skipping initialization
library_management_db | 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-linux-gnu, compiled by gcc (Debian 14.2.0-19) 14.2.0, 64-bit
library_management_db | library_management_db | 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 | library_management_db | library_management_db | 2025-11-28 12:28:20.058 UTC [1] LOG:  listening on IPv6 address "::", port 5432
library_management_db | library_management_db | 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 | library_management_db | 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 | library_management_db | 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 instead.
user_service | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.

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)-

The screenshot shows the Postman interface with a dark theme. On the left, the sidebar displays 'Lakshita Sejra's Workspace' with collections like 'Borrow Request - UserService' and 'Library Microservices Collection'. Under 'Library Microservices Collection', there is a 'POST Create User' endpoint. The 'Body' tab is selected, showing the JSON payload from the previous step. The 'Headers' tab shows 'Content-Type: application/json'. Below the body, the response section shows a green '201 CREATED' status with a timestamp of '34 ms' and a size of '268 B'. The response body is identical to the request body.

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)-

The screenshot shows the Postman interface with the following details:

- Collection:** Library Microservices Collection
- Request Type:** GET
- URL:** {{baseUrl}}/users/all
- Body:** JSON response (Preview tab) showing a JSON array of users:


```

1 [
2   {
3     "email": "lakshita929@example.com",
4     "firstname": "Lakshita",
5     "lastname": "Sejra",
6     "studentid": "S001"
7   }
8 ]
      
```
- Status:** 200 OK
- Time:** 27 ms
- Size:** 265 B

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)-

The screenshot shows the Postman interface with the following details:

- Collection:** Library Microservices Collection
- Request Type:** GET
- URL:** {{baseUrl}}/users/S001
- Body:** JSON response (Preview tab) showing a single user:


```

1 {
2   "email": "lakshita929@example.com",
3   "firstname": "Lakshita",
4   "lastname": "Sejra",
5   "studentid": "S001"
6 }
      
```
- Status:** 200 OK
- Time:** 14 ms
- Size:** 263 B

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)-

The screenshot shows the Postman interface with a collection named 'Library Microservices Collection'. A PUT request is selected with the URL `(baseUrl) /users/S001`. The request body contains the JSON provided above. The response tab shows a 200 OK status with the message: `1 {
2 "email": "lakshita_changed@example.com",
3 "firstname": "Lakshita 2",
4 "lastname": "Sejra 2",
5 "studentid": "S001"
6 }`.

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)-

The screenshot shows the Postman interface with a collection named 'Library Microservices Collection'. A DELETE request is selected with the URL `(baseUrl) /users/S001`. The request body is empty. The response tab shows a 200 OK status with the message: `1 {
2 "Message": "User deleted successfully"
3 }`.

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)-

The screenshot shows the Postman interface. On the left, there's a sidebar with 'Lakshita Sejra's Workspace' containing collections like 'Borrow Request - UserService' and 'Library Microservices Collection'. Under 'Library Microservices Collection', several endpoints are listed: POST Create User, GET Get All Users, GET Get User by ID, PUT Update User, DEL Delete User, POST Create Book, GET Get All Books, GET Get Book by ID, PUT Update Book, and DEL Delete Book. The 'POST Create Book' endpoint is selected. The main panel shows a POST request to 'http://localhost:5006/books/add'. The 'Body' tab is selected, showing the JSON payload from above. Below the request, the response is shown with a status of '201 CREATED'. The response body is also a JSON object identical to the one sent:

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

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)-

The screenshot shows the Postman interface with a collection named 'Library Microservices Collection'. A GET request is made to 'http://localhost:5006/books/all'. The response is a 200 OK status with a response time of 11 ms and a body size of 221 B. The response JSON is:

```

1 [
2   {
3     "author": "Author 6",
4     "bookid": "B6",
5     "title": "Book 6"
6   }
7 ]

```

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)-

The screenshot shows the Postman interface with a collection named 'Library Microservices Collection'. A GET request is made to 'http://localhost:5006/books/B6'. The response is a 200 OK status with a response time of 28 ms and a body size of 219 B. The response JSON is:

```

1 {
2   "author": "Author 6",
3   "bookid": "B6",
4   "title": "Book 6"
5 }

```

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)-

The screenshot shows the Postman interface with the following details:

- Collection:** Library Microservices Collection
- Request Type:** PUT
- URL:** <http://localhost:5006/books/B6>
- Body (raw JSON):**

```

1 {
2   "title": "Harry Potter",
3   "author": "JK Rowling"
4 }

```
- Response Status:** 200 OK
- Response Body (JSON):**

```

1 {
2   "author": "JK Rowling",
3   "bookid": "B6",
4   "title": "Harry Potter"
5 }

```

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 screenshot shows the Postman interface with the following details:

- Collection:** Library Microservices Collection
- Request Type:** DELETE
- URL:** <http://localhost:5006/books/B6>
- Query Params:**

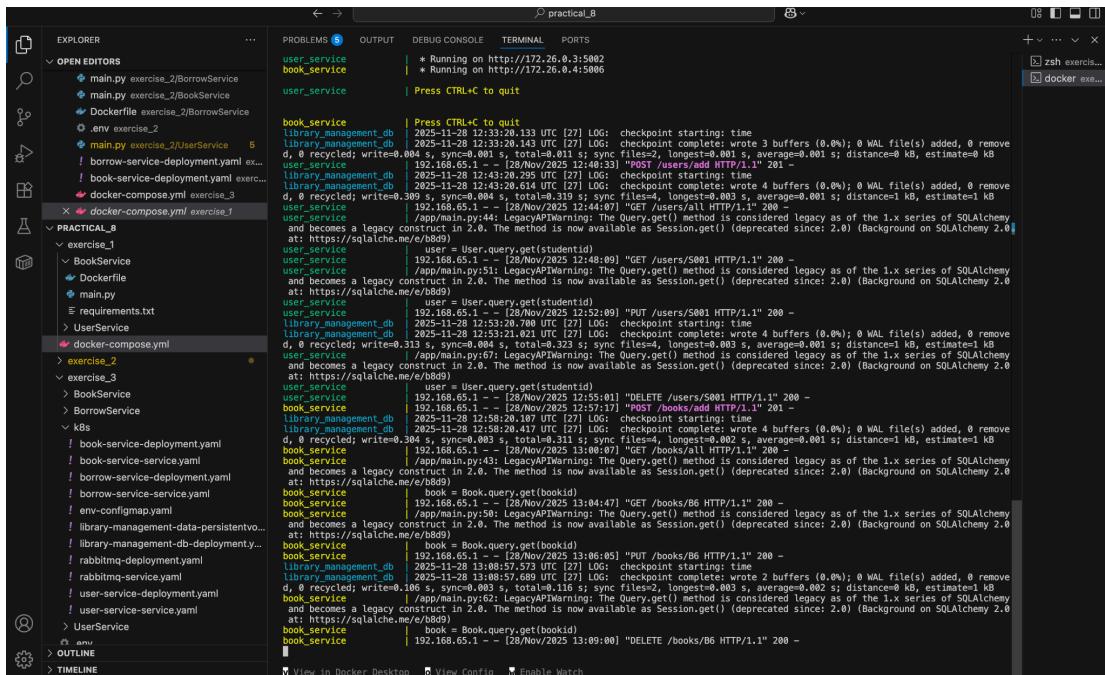
Key	Value	Description
Key	Value	Description
- Response Status:** 200 OK
- Response Body (JSON):**

```

1 {
2   "message": "Book deleted successfully"
3 }

```

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



The screenshot shows a VS Code interface with a terminal tab open. The terminal displays the output of several API requests, primarily from the `book_service` and `user_service` components. The logs include timestamps, URLs, and HTTP methods like GET, POST, and DELETE. Some entries show SQLAlchemy warnings about legacy session usage.

```
practical_B
[2025-11-28 12:33:26,333] INFO book_service | * Running on http://172.26.0.3:5002
[2025-11-28 12:33:26,333] INFO book_service | * Running on http://172.26.0.4:5006
[2025-11-28 12:33:26,333] INFO user_service | Press CTRL+C to quit
[2025-11-28 12:33:26,333] INFO book_service | 2025-11-28 12:33:26,333 UTC [27] LOG: checkpoint starting; time=0:00:00.000; wrote 3 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.004 s, sync=0.001 s, total=0.011 s; sync files=2, longest=0.001 s, average=0.001 s; distance=0 kB, estimate=0 kB
[2025-11-28 12:33:26,333] INFO user_service | 192.168.65.1 - - [28/Nov/2025 12:46:33] "POST /users/add HTTP/1.1" 201 -
[2025-11-28 12:33:26,333] INFO library_management_db | 2025-11-28 12:33:26,333 UTC [27] LOG: checkpoint starting; time=0:00:00.000; wrote 4 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.003 s, sync=0.003 s, total=0.006 s; sync files=4, longest=0.003 s, average=0.0005 s; distance=1 kB, estimate=1 kB
[2025-11-28 12:33:26,333] INFO user_service | 192.168.65.1 - - [28/Nov/2025 12:44:07] "GET /users/all HTTP/1.1" 200 -
[2025-11-28 12:33:26,333] INFO user_service | user = User.query.get(studentid)
[2025-11-28 12:33:26,333] INFO user_service | 192.168.65.1 - - [28/Nov/2025 12:48:09] "GET /users/S001 HTTP/1.1" 200 -
[2025-11-28 12:33:26,333] INFO user_service | /app/main.py:51: LegacyAPIWarning: The Query.get() method is considered legacy as of the 1.x series of SQLAlchemy and becomes a legacy construct in 2.0. The method is now available as Session.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
[2025-11-28 12:33:26,333] INFO user_service | user = User.query.get(studentid)
[2025-11-28 12:33:26,333] INFO user_service | 192.168.65.1 - - [28/Nov/2025 12:57:09] "PUT /users/S001 HTTP/1.1" 200 -
[2025-11-28 12:33:26,333] INFO user_service | /app/main.py:51: LegacyAPIWarning: The Query.get() method is considered legacy as of the 1.x series of SQLAlchemy and becomes a legacy construct in 2.0. The method is now available as Session.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
[2025-11-28 12:33:26,333] INFO book_service | 192.168.65.1 - - [28/Nov/2025 12:57:17] "POST /books/add HTTP/1.1" 201 -
[2025-11-28 12:33:26,333] INFO library_management_db | 2025-11-28 12:33:26,333 UTC [27] LOG: checkpoint complete; wrote 4 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.309 s, sync=0.004 s, total=0.319 s; sync files=4, longest=0.003 s, average=0.0005 s; distance=1 kB, estimate=1 kB
[2025-11-28 12:33:26,333] INFO user_service | 192.168.65.1 - - [28/Nov/2025 12:44:07] "GET /users/all HTTP/1.1" 200 -
[2025-11-28 12:33:26,333] INFO user_service | user = User.query.get(studentid)
[2025-11-28 12:33:26,333] INFO user_service | 192.168.65.1 - - [28/Nov/2025 12:57:17] "PUT /users/S001 HTTP/1.1" 200 -
[2025-11-28 12:33:26,333] INFO user_service | /app/main.py:51: LegacyAPIWarning: The Query.get() method is considered legacy as of the 1.x series of SQLAlchemy and becomes a legacy construct in 2.0. The method is now available as Session.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
[2025-11-28 12:33:26,333] INFO user_service | user = User.query.get(studentid)
[2025-11-28 12:33:26,333] INFO user_service | 192.168.65.1 - - [28/Nov/2025 12:57:17] "DELETE /users/S001 HTTP/1.1" 200 -
[2025-11-28 12:33:26,333] INFO book_service | 192.168.65.1 - - [28/Nov/2025 12:57:17] "POST /books/add HTTP/1.1" 201 -
[2025-11-28 12:33:26,333] INFO library_management_db | 2025-11-28 12:33:26,333 UTC [27] LOG: checkpoint starting; time=0:00:00.000; wrote 2 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.308 s, sync=0.003 s, total=0.318 s; sync files=2, longest=0.003 s, average=0.0005 s; distance=1 kB, estimate=1 kB
[2025-11-28 12:33:26,333] INFO book_service | 192.168.65.1 - - [28/Nov/2025 13:00:47] "GET /books/B6 HTTP/1.1" 200 -
[2025-11-28 12:33:26,333] INFO book_service | /app/main.py:58: LegacyAPIWarning: The Query.get() method is considered legacy as of the 1.x series of SQLAlchemy and becomes a legacy construct in 2.0. The method is now available as Session.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
[2025-11-28 12:33:26,333] INFO book_service | book = Book.query.get(bookid)
[2025-11-28 12:33:26,333] INFO book_service | 192.168.65.1 - - [28/Nov/2025 13:00:47] "GET /books/B6 HTTP/1.1" 200 -
[2025-11-28 12:33:26,333] INFO book_service | /app/main.py:58: LegacyAPIWarning: The Query.get() method is considered legacy as of the 1.x series of SQLAlchemy and becomes a legacy construct in 2.0. The method is now available as Session.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
[2025-11-28 12:33:26,333] INFO book_service | book = Book.query.get(bookid)
[2025-11-28 12:33:26,333] INFO book_service | 192.168.65.1 - - [28/Nov/2025 13:09:00] "DELETE /books/B6 HTTP/1.1" 200 -
```

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-

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

lakshitsa@lakshitsa-MacBook-Air exercise_2 % cd ../exercise_2
lakshitsa@lakshitsa-MacBook-Air exercise_2 % docker compose up --build
WARN [0000] /Users/lakshitsa/Cloud Computing Practicals/practical_8/exercise_2/docker-compose.yml: the attribute 'version' is obsolete, it will be removed in the future. Use 'image' instead to avoid potential confusion
[+] Building 1.6s (30/38) FINISHED
=> reading from stdin 1.78kB
=> lbook_service internal load build definition from Dockerfile
=> lbook_service internal load Dockerfile:444B
=> [book service internal] load build definition from Dockerfile
=> transferring dockerfile: 444B
=> [user service internal] load build definition from Dockerfile
=> lbook_service internal load Dockerfile:444B
=> lbook service internal load metadata for docker.io/library/python:3.11-slim
=> [auth] python:pull token for registry-1.docker.io
=> [user service internal] load .dockerignore
=> [user service internal] load .dockerignore
=> [book service internal] load .dockerignore
=> [user service internal] load context: 2B
=> lbook service internal load build definition from Dockerfile
=> [book service internal] load build context
=> transferring context: 1.78kB
=> [book service internal] load build context
=> [user service internal] load build context
=> transferring context: 3.72kB
=> [user service internal] load build context
=> transferring context: 5.76kB
=> lbook service internal load build context
=> CACHED [user service internal] RUN apt-get update && apt-get install -y gcc libpq-dev && rm -rf /var/lib/apt/lists/*
=> CACHED [user service 5/6] COPY requirements.txt
=> CACHED [user service 5/6] RUN pip install --no-cache-dir -r requirements.txt
=> CACHED [user service 5/6] COPY requirements.txt
=> CACHED [book service 5/6] RUN pip install --no-cache-dir -r requirements.txt
=> CACHED [book service 5/6] COPY ..
=> CACHED [book service 4/6] COPY requirements.txt
=> CACHED [book service 4/6] RUN pip install --no-cache-dir -r requirements.txt
=> CACHED [book service 3/6] RUN pip install --no-cache-dir -r requirements.txt
=> CACHED [book service 2/6] RUN pip install --no-cache-dir -r requirements.txt
=> lbook service exporting to image
=> exporting layers
=> exporting manifest sha256:bcc9e0c2c8e893dd02d72d1ff8f2/3d89add071ab652073d482d24865dd77
=> exporting manifest sha256:0086e74d1475378ef8942b678466e51aae454b0e60d176fca1094f3a82
=> exporting attestation manifest sha256:e16e15f58c9fe9126919e39e2e29290d1a581da7a9bb1393c73dfcaf2e
=> exporting manifest list sha256:986e103b94d924631806fc7a7a4f5631d78e256f7d8a4992956d03d82c1b
=> naming to docker.io/library/exercise_2-book_service:latest
=> naming to docker.io/library/exercise_2-user_service:latest
=> [user service] exporting to image
=> unpacking to docker.io/library/exercise_2-user_service:latest
=> lborrow_service exporting to image
=> exporting layers
=> exporting manifest sha256:0111e8e0c2b0d452d6fbce5d2d988a074efcc3c7514dd534246b3f934d96
=> exporting manifest sha256:033939093dbbd234ec359d49393fd938e3e4a1132d08c410067886fe0b6e08
=> exporting attestation manifest sha256:019a17313394e1744f7570d47883c35d97461195f3ca527e27330e
=> exporting manifest list sha256:d13031bab31b8d43f1e1a9f0a90b039e50806cd1d1e000e32500717b12c
=> naming to docker.io/library/exercise_2-user_service:latest
=> lbook_service exporting to image
=> exporting layers
=> exporting manifest sha256:68665c2e70077710d80774c522cfbe1975c45866427382c705261b48b1a5748

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

lakshitsa@lakshitsa-MacBook-Air exercise_2 % cd ../exercise_2
lakshitsa@lakshitsa-MacBook-Air exercise_2 % docker compose up --build
[+] Building 0.0s (0/38) FINISHED
=> reading from stdin 0.05kB
=> lbook_service internal load build definition from Dockerfile
=> lbook_service internal load Dockerfile:444B
=> [book service internal] load build definition from Dockerfile
=> lbook_service internal load build context
=> [book service internal] load Dockerfile:444B
=> [user service internal] load build definition from Dockerfile
=> [user service internal] load build context
=> [user service internal] load Dockerfile:444B
=> lbook service internal load build context
=> [user service internal] load build context
=> [book service internal] load build context
=> [user service internal] load build context
=> lbook service internal load build context
=> CACHED [user service internal] RUN apt-get update && apt-get install -y postgresql && rm -rf /var/lib/apt/lists/*
=> CACHED [user service 5/6] COPY requirements.txt
=> CACHED [user service 5/6] RUN pip install --no-cache-dir -r requirements.txt
=> CACHED [user service 5/6] COPY requirements.txt
=> CACHED [book service 5/6] RUN pip install --no-cache-dir -r requirements.txt
=> CACHED [book service 5/6] COPY ..
=> CACHED [book service 4/6] COPY requirements.txt
=> CACHED [book service 4/6] RUN pip install --no-cache-dir -r requirements.txt
=> CACHED [book service 3/6] RUN pip install --no-cache-dir -r requirements.txt
=> CACHED [book service 2/6] RUN pip install --no-cache-dir -r requirements.txt
=> lbook service exporting to image
=> exporting layers
=> exporting manifest sha256:a38d0544f100f1167f83a902e7a28833933d7d11a5686d1d1450a93611f258f938f359d
=> exporting manifest sha256:a303a652a96a2e54483777230990f3101310f197a5d1282254a4433a97281413427a55466c9d
=> exporting attestation manifest sha256:a303a652a96a2e54483777230990f3101310f197a5d1282254a4433a97281413427a55466c9d
=> exporting manifest list sha256:13031bab31b8d43f1e1a9f0a90b039e50806cd1d1e000e32500717b12c
=> naming to docker.io/library/exercise_2-book_service:latest
=> naming to docker.io/library/exercise_2-user_service:latest
=> lbook_service exporting to image
=> exporting layers
=> exporting manifest sha256:756168665c2e70077710d80774c522cfbe1975c45866427382c705261b48b1a5748

LIBRARY MANAGEMENT

library_management_db PostgreSQL Database directory appears to contain a database; Skipping initialization
library_management_db PostgresSQL database directory appears to contain a database; Skipping initialization
library_management_db PostgresSQL database directory appears to contain a database; Skipping initialization
library_management_db x86_64, compiled by gcc 4.8.5 20150623 (Debian 4.8.5-19) 64-bit
library_management_db 2025-11-28 15:27:23,099 UTC [1] LOG: starting PostgreSQL 15.10 (Debian 15.10-1.pgdg13+1) on aarch64-unknown-linux-gnu, compiled by gcc 4.8.5 20150623 (Debian 4.8.5-19) 64-bit
library_management_db 2025-11-28 15:27:23,099 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
library_management_db 2025-11-28 15:27:23,099 UTC [1] LOG: listening on IPv6 address "::", port 5432
library_management_db 2025-11-28 15:27:23,101 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
library_management_db 2025-11-28 15:27:23,104 UTC [28] LOG: database system was shut down at 2025-11-28 12:27:22 UTC
library_management_db 2025-11-28 15:27:23,107 UTC [1] LOG: database system is ready to accept connections
library_management_db UserService: database is ready
book_service BookService: database is ready

BORROW SERVICE

book_service BookService running on port 5006
user_service UserService running on port 5002
book_service * Serving Flask app 'main' with
book_service * Serving Flask app 'main'
book_service * Debug mode: on
book_service * Debug mode: off

USER SERVICE

user_service WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
user_service * Running on all addresses (0.0.0.0)
user_service * Running on http://127.0.0.1:5000
user_service * Running on http://127.25.0.5:5006
user_service Press CTRL+C to quit
user_service WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
user_service * Running on all addresses (0.0.0.0)
user_service * Running on http://127.0.0.1:5002
user_service * Running on http://127.25.0.5:5002
user_service * Debug mode: off
borrow_service BorrowService database is ready
borrow_service BorrowService env RABBITMQ_USER: library_user
borrow_service BorrowService env RABBITMQ_PASS: library_pass
borrow_service BorrowService env RABBITMQ_HOST: rabbitmq
borrow_service BorrowService env RABBITMQ_PORT: 5672
borrow_service BorrowService connecting to RabbitMQ at rabbitmq:5672BorrowService running on port 5004
borrow_service BorrowService * Serving Flask app 'main'
borrow_service BorrowService * Debug mode: off
borrow_service BorrowService: RabbitMQ connection failed, retrying in 5 seconds:
borrow_service WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
borrow_service * Running on all addresses (0.0.0.0)
borrow_service * Running on http://127.0.0.1:5004
borrow_service * Running on https://127.0.0.1:5004
borrow_service * Running on http://127.25.0.4:5004

```

practical_8
[info] <0.248.0> Running boot step rabbit_core_metrics_gc defined by app rabbit
[info] <0.248.0> Running boot step background_gc defined by app rabbit
[info] <0.248.0> Running boot step routing_ready defined by app rabbit
[info] <0.248.0> Running boot step pre_flight defined by app rabbit
[info] <0.248.0> Running boot step notify_cluster defined by app rabbit
[info] <0.248.0> Running boot step networking defined by app rabbit
[info] <0.248.0> Running boot step rabbit_quorum_queue_periodic_membership_recon
[info] <0.248.0> Starting worker pool 'definition_import_worker_pool' with 10 processes
[info] <0.307.0> Starting worker pool 'definition_import_pool' with 10 processes
[info] <0.248.0> Running boot step definition_import_pool defined by app rabbit
[info] <0.248.0> Running boot step definition_import_worker_pool defined by app rabbit
[info] <0.248.0> Running boot step cluster_name defined by app rabbit
[info] <0.248.0> Initialising internal cluster ID to 'rabbitmq-cluster-id-cjqEWfRKiPirvBF28SRB0A'
[info] <0.248.0> Running boot step direct_client defined by app rabbit
[info] <0.248.0> Running boot step rabbit_management_load_definitions defined by app rabbit
[info] <0.248.0> Resetting node maintenance status
[warning] <0.681.0> Deprecated features: 'management_metrics_collection': Feature 'management_metrics_collection' is deprecated.
[warning] <0.681.0> By default, this feature can still be used for now.
[warning] <0.681.0> Its use will not be permitted by default in a future minor RabbitMQ version and the feature will be removed from a future major RabbitMQ version: actual versions to be determined.
[warning] <0.681.0> To continue using this feature when it is not permitted by default, set the following parameter in your configuration:
[warning] <0.681.0>     management.metrics.collection = true
[warning] <0.681.0> To test RabbitMQ as if the feature was removed, set this in your configuration:
[warning] <0.681.0>     management.metrics.collection = false
[warning] <0.718.0> Statistics database started.
[warning] <0.748.0> Starting worker pool 'management_worker_pool' with 3 processes
[warning] <0.763.0> Prometheus metrics: HTTP (non-TLS) listener started on port 1567
[warning] <0.763.0> Prometheus metrics: HTTP (non-TLS) listener started on port 1567
[info] <0.658.0> Ready to start client connection listeners
[info] <0.887.0> started TCP listener on [:]:5672
[info] <0.887.0> completed with 4 plugins.
[info] <0.658.0> Server startup complete; 4 plugins started.
[info] <0.658.0> * rabbitmq_prometheus
[info] <0.658.0> * rabbitmq_management
[info] <0.658.0> * rabbitmq_management_agent
[info] <0.658.0> * rabbitmq_web_dispatch
[info] <0.658.0> Time to start RabbitMQ: 3180961 us
[info] <0.812.0> Connecting to RabbitMQ at rabbitmq:5672
[info] <0.812.0> Connection <0.812.0> (172.25.0.4:53154 -> 172.25.0.2:5672): user 'library_user' authenticated and granted access to vhost '/'
[info] <0.812.0> connection <0.812.0> (172.25.0.4:53154 -> 172.25.0.2:5672): using library_management_db
[info] <0.812.0> BorrowService listening on RabbitMQ queue: borrow_book
[info] <0.812.0> library_management_db
[info] <0.812.0> library_management_db
[info] <0.812.0> LOG: checkpoint completed write 3 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled, write=0.003 s, sync=0.002 s, total=0.011 s, sync files=2, longest=0.001 s, average=0.001 s, distance=0 kB, estimate=0 kB

```

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-

Lakshita Sejra's Workspace

Borrow Request - UserService

POST Send Borrow Request (RabbitMQ)

GET New Request

Body (raw) JSON

```

1 {
2   "student_id": "S001",
3   "book_id": "B6",
4   "date_returned": "2025-12-25"
5 }

```

201 CREATED

Body Cookies Headers (5) Test Results

{} JSON Preview Visualize

```

1 {
2   "message": "Borrow requested",
3   "request": {
4     "book_id": "B6",
5     "date_returned": "2025-12-25",
6     "student_id": "S001"
7   }
8 }

```

- **Borrow Request (GET)**

Method- GET

URL- <http://localhost:5004/borrow/S001>

Output-

More than 5 books condition check- 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.**

Borrow Request - UserService / New Request

GET http://localhost:5004/borrow/S001

Body (JSON) Preview Visualize

```

1 [
2   {
3     "book_id": "B1",
4     "id": 1,
5     "student_id": "S001"
6   },
7   {
8     "book_id": "B2",
9     "id": 2,
10    "student_id": "S001"
11  },
12  {
13    "book_id": "B3",
14    "id": 3,
15    "student_id": "S001"
16  },
17  {
18    "book_id": "B4",
19    "id": 4,
20    "student_id": "S001"
21  },
22  {
23    "book_id": "B5",
24    "id": 5,
25    "student_id": "S001"
26  }
27 ]

```

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

```
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/BorrowService
  .env exercise_2
  main.py exercise_2/UserService 5
  borrow-service-deployment.yaml ex...
  book-services-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

e 'management_metrics_collection' is deprecated.
rabbitmq | 2025-11-28 15:52:17.612555+00:00 [warning] <0.563.0> By default, this feature can still be used for now.
rabbitmq | 2025-11-28 15:52:17.612555+00:00 [warning] <0.563.0> Its use will not be permitted by default in a future minor R
abbitMQ version and the feature will be removed from a future major RabbitMQ version; actual versions to be determined.
efault, set the following parameter in your configuration:
rabbitmq | 2025-11-28 15:52:17.612555+00:00 [warning] <0.563.0> To continue using this feature when it is not permitted by d
efault, set the following parameter in your configuration:
n = true'
rabbitmq | 2025-11-28 15:52:17.612555+00:00 [warning] <0.563.0> "deprecated_features.permit.management_metrics_collectio
n = false"
rabbitmq | 2025-11-28 15:52:18.757601+00:00 [info] <0.600.0> Management plugin: HTTP (non-TLS) listener started on port 1567
2
rabbitmq | 2025-11-28 15:52:18.757681+00:00 [info] <0.631.0> Statistics database started.
rabbitmq | 2025-11-28 15:52:18.757717+00:00 [info] <0.630.0> Starting worker pool 'management_worker_pool' with 3 processes
in it
rabbitmq | 2025-11-28 15:52:18.760384+00:00 [info] <0.645.0> Prometheus metrics: HTTP (non-TLS) listener started on port 156
92
rabbitmq | 2025-11-28 15:52:18.760468+00:00 [info] <0.540.0> Ready to start client connection listeners
rabbitmq | 2025-11-28 15:52:18.761158+00:00 [info] <0.689.0> started TCP listener on [:15672]
rabbitmq | 2025-11-28 15:52:18.761158+00:00 [info] <0.540.0> Server startup complete; 4 plugins started.
rabbitmq | 2025-11-28 15:52:18.831939+00:00 [info] <0.540.0> * rabbitmq_management
rabbitmq | 2025-11-28 15:52:18.831939+00:00 [info] <0.540.0> * rabbitmq_management_agent
rabbitmq | 2025-11-28 15:52:18.831939+00:00 [info] <0.540.0> * rabbitmq_web_dispatch
rabbitmq | 2025-11-28 15:52:18.831939+00:00 [info] <0.540.0> completed with 4 plugins.
rabbitmq | 2025-11-28 15:52:18.831939+00:00 [info] <0.540.0> Time to start RabbitMQ: 3521976 us
borrow_service | BorrowService: Connecting to RabbitMQ at rabbitmq:5672
rabbitmq | 2025-11-28 15:52:20.567854+00:00 [info] <0.694.0> accepting AMQP connection <0.694.0> (172.25.0.4:47334 -> 172.25
.0.2:5672)
rabbitmq | 2025-11-28 15:52:20.571035+00:00 [info] <0.694.0> connection <0.694.0> (172.25.0.4:47334 -> 172.25.0.2:5672): use
r 'library_user' authenticated and granted access to vhost '/'
borrow_service | BorrowService listening on RabbitMQ queue: borrow_book
borrow_service | 192.168.65.1 -- [28/Nov/2025 15:52:36] "GET /borrow/S001 HTTP/1.1" 200 -
user_service | UserServices: Connecting to RabbitMQ at rabbitmq:5672
rabbitmq | 2025-11-28 15:52:20.571035+00:00 [info] <0.708.0> accepting AMQP connection <0.708.0> (172.25.0.5:58454 -> 172.25
.0.2:5672)
rabbitmq | 2025-11-28 15:52:20.571035+00:00 [info] <0.708.0> connection <0.708.0> (172.25.0.5:58454 -> 172.25.0.2:5672): use
r 'library_user' authenticated and granted access to vhost '/'
user_service | 192.168.65.1 -- [28/Nov/2025 15:52:53] "POST /users/borrow/request HTTP/1.1" 201 -
borrow_service | Received borrow request: {'student_id': 'S001', 'book_id': 'B6', 'date_returned': '2025-12-25'}
rabbitmq | 2025-11-28 15:52:20.571035+00:00 [warning] <0.708.0> closing AMQP connection <0.708.0> (172.25.0.5:58454 -> 172.2
5.0.2:5672, vhost: '/', user: 'library_user')
rabbitmq | 2025-11-28 15:52:20.571035+00:00 [warning] <0.708.0> client unexpectedly closed TCP connection
user_service | [app/main.py:16: LegacyAPIWarning: The Query.get() method is considered legacy as of the 1.x series of SQLAlchemy
and becomes a legacy construct in 2.0. The method is now available as Session.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0
at: https://sqlalchemy.me/e/b8d9)]
user_service | user = User.query.get(studentid)
user_service | 172.25.0.4 -- [28/Nov/2025 15:52:53] "GET /users/S001 HTTP/1.1" 200 -
book_service | /app/main.py:79: LegacyAPIWarning: The Query.get() method is considered legacy as of the 1.x series of SQLAlchemy
and becomes a legacy construct in 2.0. The method is now available as Session.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0
at: https://sqlalchemy.me/e/b8d9)
book_service | book = Book.query.get(bookid)
book_service | 172.25.0.4 -- [28/Nov/2025 15:52:53] "GET /books/B6 HTTP/1.1" 200 -
borrow_service | Student already has 5 books, rejecting
```

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-

The screenshot shows the Docker Hub interface with the URL `hub.docker.com/repositories/lakshitasejra`. On the left, there's a sidebar with options like 'Repositories', 'Hardened Images', 'Collaborations', 'Settings', 'Default privacy', 'Notifications', 'Billing', 'Usage', and 'Pulls'. The main area is titled 'Repositories' and shows three entries:

Name	Last Pushed	Contains	Visibility	Scout
lakshitasejra/library-book-service	7 days ago	IMAGE	Public	Inactive
lakshitasejra/library-borrow-service	7 days ago	IMAGE	Public	Inactive
lakshitasejra/library-user-service	7 days ago	IMAGE	Public	Inactive

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

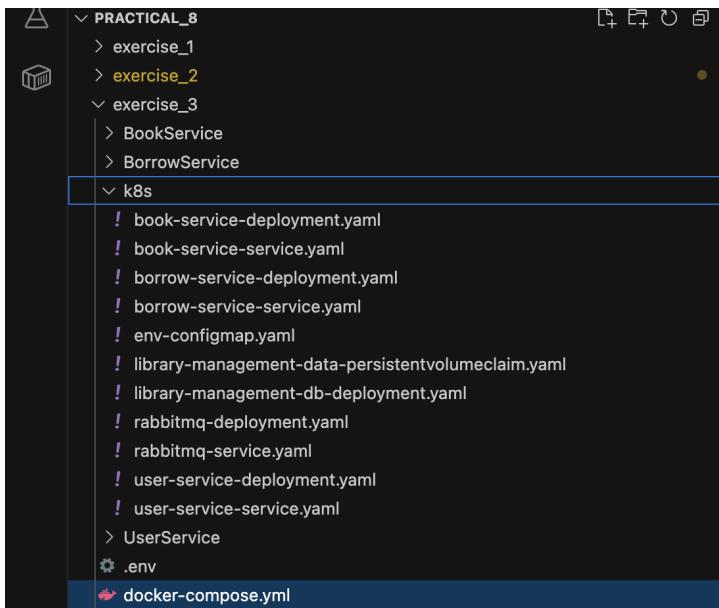
The rest of the docker-compose.yml file remains the same.

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:

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

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):

The screenshot shows the Postman interface with a collection named "Library Microservices Collection". A POST request is made to `http://localhost:5002/users/add`. The request body is a JSON object:

```
{  
  "studentid": "S003",  
  "firstname": "ABC2",  
  "lastname": "XYZ2",  
  "email": "abcyxyz2@example.com"  
}
```

The response status is **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):

The screenshot shows the Postman interface with a collection named "Library Microservices Collection". A POST request is made to `http://localhost:5006/books/add`. The request body is a JSON object:

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

The response status is **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):

The screenshot shows the Postman interface with a successful POST request. The request URL is `http://localhost:5002/users/borrow/request`. The response status is `201 CREATED`, and the response body is:

```

{
  "message": "Borrow requested",
  "request": {
    "book_id": "B8",
    "date_returned": "2025-11-29",
    "student_id": "S003"
  }
}

```

- **Check borrowed books (BorrowService):**

GET <http://localhost:5004/borrow/S003>

Postman Output (200 OK):

The screenshot shows the Postman interface with a successful GET request. The request URL is `http://localhost:5004/borrow/S003`. The response status is `200 OK`, and the response body is:

```

[
  {
    "book_id": "B8",
    "id": 1,
    "student_id": "S003"
  }
]

```

Kubectl logs check for borrow-service:

Running kubectl logs -f deploy/borrow-service

```
lakshitasejra@Lakshitas-MacBook-Air exercise_3 % kubectl logs -f deploy/borrow-service
BorrowService: Database is ready
BorrowService env RABBITMQ_USER: library_user
BorrowService env RABBITMQ_PASS: library_pass
BorrowService env RABBITMQ_HOST: rabbitmq
BorrowService env RABBITMQ_PORT: 5672
BorrowService: Connecting to RabbitMQ at rabbitmq:5672
BorrowService running on port 5004
 * Serving Flask app 'main'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5004
 * Running on http://10.244.0.23:5004
Press CTRL+C to quit
BorrowService listening on RabbitMQ queue: borrow_book
Received borrow request: {'student_id': 'S003', 'book_id': 'B8', 'date_returned': '2025-11-29'}
Borrow request saved successfully
127.0.0.1 -- [28/Nov/2025 17:42:46] "GET /borrow/S003 HTTP/1.1" 200 -
127.0.0.1 -- [28/Nov/2025 17:46:54] "GET /borrow/S003 HTTP/1.1" 200 -
[  ]
```

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.