

# Cloud PDF Report: PROJECT2 – Tom Gibson – 23222095

# reference, for accessing the endpoints

<http://localhost:15672/#/queues>

<http://localhost:5002/users/all>

<http://localhost:5006/books/all>

## Exercise one: Service setup

- 1.1) Create a folder with the name PRACTICAL2\_FirstName\_LastName\_StudentID and a subfolder named exercise\_one
- 1.2) Download the source code UserService.zip from Brightspace and extract it to the folder exercise\_one
- 1.3) Using a base image of your choice, create a Dockerfile that will be used to run the UserService. Make sure the service runs on port 5002 in the container
- 1.4) In the exercise\_one folder, create a docker\_compose.yml file. Add two services: UserService and a database service. The database service should be based on a Postgres database. Set up the necessary network and volume for the database service
- 1.5) Run docker-compose up and, using curl, test all the endpoints and ensure they work as expected (save data, retrieve, update and delete based on student\_id).

docker-compose up --build

Test the /users/add Endpoint:

```
curl -X POST -H "Content-Type: application/json" \
-d '{"studentid": "S001", "firstname": "John", "lastname": "Doe", "email": "john.doe@example.com"}' \ http://localhost:5002/users/add
```

```

PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_one
> curl -Method POST -Uri "http://localhost:5002/users/add" \
>> -Headers @{ "Content-Type" = "application/json" } ` 
>> -Body '{ "studentid": "S001", "firstname": "John", "lastname": "Doe", "email": "john.d
oe@example.com" }'

StatusCode      : 201
StatusDescription : CREATED
Content          : {"email":"john.doe@example.com","firstname":"John","lastname":"Doe",
"studentid":"S001"}

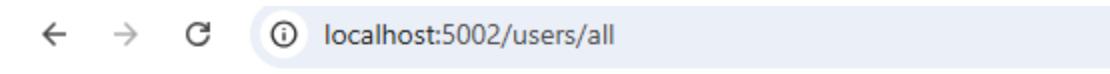
RawContent      : HTTP/1.1 201 CREATED
                  Connection: close
                  Content-Length: 88
                  Content-Type: application/json
                  Date: Sun, 24 Nov 2024 13:39:22 GMT
                  Server: Werkzeug/3.1.3 Python/3.10.15

                  {"email":"john.doe@example.co...
Forms           : {}
Headers         : {[["Connection", "close"], ["Content-Length", 88], ["Content-Type",
application/json], ["Date", "Sun, 24 Nov 2024 13:39:22 GMT"]...]}
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : mshtml.HTMLDocumentClass
RawContentLength : 88

```

Retrieve All Users (GET /users/all):

/users/all



Pretty-print

```
[{"email": "john.doe@example.com", "firstname": "John", "lastname": "Doe", "studentid": "S001"}, {"email": "Ther.Dev@example.com", "firstname": "Theresa", "lastname": "Devine", "studentid": "S002"}]
```

...or:

curl -Method GET http://localhost:5002/users/all

```
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_one
> curl -Method GET http://localhost:5002/users/all

StatusCode      : 200
StatusDescription : OK
Content          : [{"email":"john.doe@example.com","firstname":"John","lastname":"Doe"
                  ,"studentid":"S001"}, {"email":"Ther.Dev@example.com","firstname":"Th
eresia","lastname":"Devine","studentid":"S002"}]

RawContent      : HTTP/1.1 200 OK
                  Connection: close
                  Content-Length: 184
                  Content-Type: application/json
                  Date: Sun, 24 Nov 2024 14:03:16 GMT
                  Server: Werkzeug/3.1.3 Python/3.10.15

Forms           : {}
Headers         : [[Connection, close], [Content-Length, 184], [Content-Type,
                  application/json], [Date, Sun, 24 Nov 2024 14:03:16 GMT]...]
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : mshtml.HTMLDocumentClass
RawContentLength : 184
```

Retrieve a Specific User (GET /users/S002):



...or:

```
curl -Method GET http://localhost:5002/users/S002
```

```

PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_one
> curl -Method GET http://localhost:5002/users/S002

StatusCode      : 200
StatusDescription : OK
Content          : {"email":"Ther.Dev@example.com", "firstname":"Theresa", "lastname":"De
                     vine", "studentid":"S002"}

RawContent      : HTTP/1.1 200 OK
                  Connection: close
                  Content-Length: 94
                  Content-Type: application/json
                  Date: Sun, 24 Nov 2024 14:05:18 GMT
                  Server: Werkzeug/3.1.3 Python/3.10.15

                  {"email":"Ther.Dev@example.com", "f...
Forms           : {}
Headers         : {[Connection, close], [Content-Length, 94], [Content-Type,
                     application/json], [Date, Sun, 24 Nov 2024 14:05:18 GMT]...}
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : mshtml.HTMLDocumentClass
RawContentLength : 94

```

Update a User (PUT /users/<studentid>) by studentid:

```

> curl -Method PUT -Uri "http://localhost:5002/users/S001" `

>> -Headers @`"Content-Type" = "application/json"` `

>> -Body '{"firstname": "Jane", "lastname": "Smith", "email": "jane.smith@example.com"}'

```

```

Administrator: Command Prompt X Administrator: Windows PowerShell X + x
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_one
> curl -Method PUT -Uri "http://localhost:5002/users/S001" `

>> -Headers @`"Content-Type" = "application/json"` `

>> -Body '{"firstname": "Jane", "lastname": "Smith", "email": "jane.smith@example.com"}'

StatusCode      : 200
StatusDescription : OK
Content          : {"email":"jane.smith@example.com", "firstname":"Jane", "lastname":"Sm
                     ith", "studentid":"S001"}

RawContent      : HTTP/1.1 200 OK
                  Connection: close
                  Content-Length: 92
                  Content-Type: application/json
                  Date: Sun, 24 Nov 2024 14:45:52 GMT
                  Server: Werkzeug/3.1.3 Python/3.10.15

                  {"email":"jane.smith@example.com", ...
Forms           : {}
Headers         : {[Connection, close], [Content-Length, 92], [Content-Type,
                     application/json], [Date, Sun, 24 Nov 2024 14:45:52 GMT]...}
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : mshtml.HTMLDocumentClass
RawContentLength : 92

```

Delete a User (DELETE /users/<studentid>), by studentid:

```
curl -Method DELETE http://localhost:5002/users/S001
```

The terminal window shows the command `curl -Method DELETE http://localhost:5002/users/S001` being run. The response includes status code 200, OK, and a message "User deleted successfully". The browser window shows the URL `localhost:5002/users/all` with a JSON response containing one user entry.

```
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_one
> curl -Method DELETE http://localhost:5002/users/S001

StatusCode : 200
StatusDescription : OK
Content : {"message":"User deleted successfully"}

RawContent : HTTP/1.1 200 OK
Connection: close
Content-Length: 40
Content-Type: application/json
Date: Sun, 24 Nov 2024 14:48:21 GMT
Server: Werkzeug/3.1.3 Python/3.10.15

        {"message":"User deleted successfu...
Forms : {}
Headers : {[Connection, close], [Content-Length, 40], [Content-Type, application/json], [Date, Sun, 24 Nov 2024 14:48:21 GMT]...}
Images : {}
InputFields : {}
Links : {}
ParsedHtml : mshtml.HTMLDocumentClass
RawContentLength : 40
```

```
Pretty-print □
[{"email":"Theresa.Dev@example.com", "firstname":"Theresa", "lastname":"Devine", "studentid":"S002"}]
```

1.6) Duplicate UserService and create BookService

1.7) Make the necessary modifications in the code to satisfy the properties of the BookService described above. BookService should run on port 5006

Verify database exists, adding 1 book:

The terminal window shows the command `curl -Method POST -Uri "http://localhost:5006/books/add"` being run with headers and body. The response includes status code 201, CREATED, and a message "Book added successfully". The browser window shows the URL `localhost:5006/books/all` with a JSON response containing one book entry.

```
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_one> curl -Method POST -Uri "http://localhost:5006/books/add"
>> -Headers @{
  "Content-Type" = "application/json"
} `

>> -Body '{
  "isbn": "000-000-000",
  "title": "Test Book",
  "author": "John Author",
  "year": 2024
}' `

StatusCode : 201
StatusDescription : CREATED
Content : {"author":"John Author", "isbn":"000-000-000", "title":"Test Book", "year":2024}

RawContent : HTTP/1.1 201 CREATED
Connection: close
Content-Length: 78
Content-Type: application/json
Date: Tue, 26 Nov 2024 10:09:30 GMT
Server: Werkzeug/3.1.3 Python/3.10.15

        {"author":"John Author", "isbn...
Forms : {}
Headers : {[Connection, close], [Content-Length, 78], [Content-Type, application/json], [Date, Tue, 26 Nov 2024 10:09:30 GMT]...}
Images : {}
InputFields : {}
Links : {}
ParsedHtml : mshtml.HTMLDocumentClass
RawContentLength : 78
```

```
Pretty-print □
[{"author": "John Author", "isbn": "000-000-000", "title": "Test Book", "year": 2024}]
```

Retrieve all books:

```
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_one> curl -Method GET http://localhost:5006/books/all

StatusCode      : 200
StatusDescription : OK
Content         : [{"author": "John Author", "isbn": "000-000-000", "title": "Test Book", "year": 2024}]

RawContent      : HTTP/1.1 200 OK
                  Connection: close
                  Content-Length: 80
                  Content-Type: application/json
                  Date: Tue, 26 Nov 2024 10:18:28 GMT
                  Server: Werkzeug/3.1.3 Python/3.10.15

                  [{"author": "John Author", "isbn": "0..."}]
Forms           : {}
Headers         : {[["Connection", "close"], ["Content-Length", 80], ["Content-Type", "application/json"], ["Date", "Tue, 26 Nov 2024 10:18:28 GMT"]...]}
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : mshtml.HTMLDocumentClass
RawContentLength : 80
```

### Get book by ID/ ISBN:

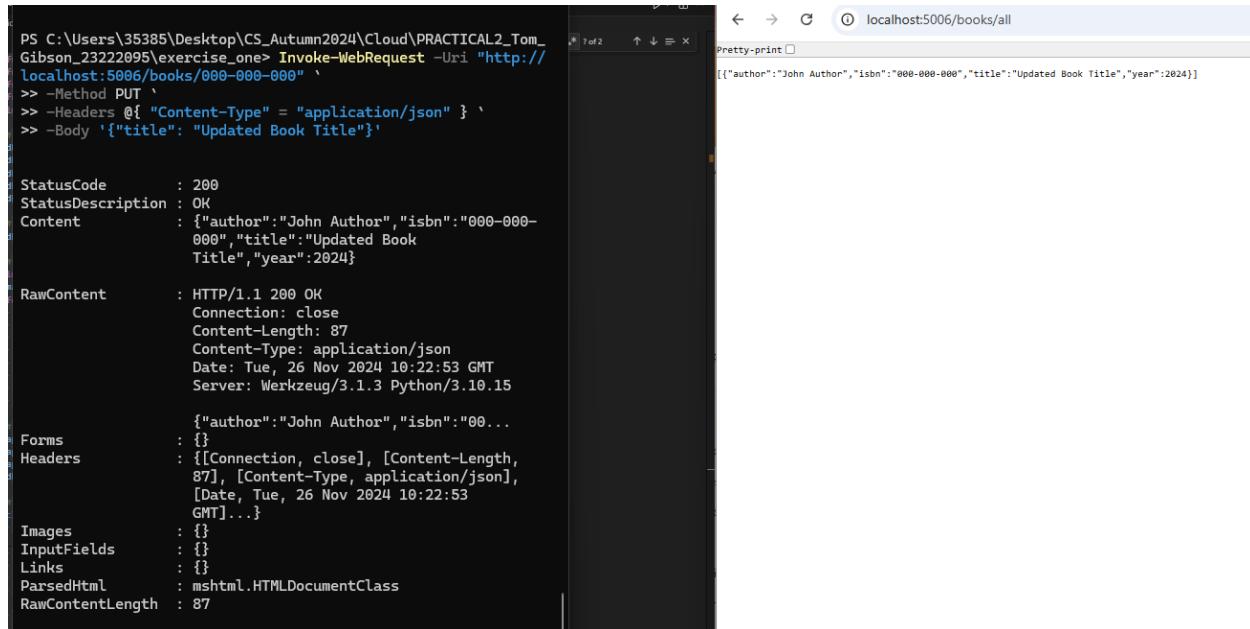
```
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_one> curl -Method GET http://localhost:5006/books/000-000-000

StatusCode      : 200
StatusDescription : OK
Content         : {"author": "John Author", "isbn": "000-000-000", "title": "Test Book", "year": 2024}

RawContent      : HTTP/1.1 200 OK
                  Connection: close
                  Content-Length: 78
                  Content-Type: application/json
                  Date: Tue, 26 Nov 2024 10:19:10 GMT
                  Server: Werkzeug/3.1.3 Python/3.10.15

                  {"author": "John Author", "isbn": "00..."}
Forms           : {}
Headers         : {[["Connection", "close"], ["Content-Length", 78], ["Content-Type", "application/json"], ["Date", "Tue, 26 Nov 2024 10:19:10 GMT"]...]}
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : mshtml.HTMLDocumentClass
RawContentLength : 78
```

## Update a book:



The screenshot shows a terminal window on the left and a browser window on the right.

**Terminal Output (PowerShell):**

```
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_one> Invoke-WebRequest -Uri "http://localhost:5006/books/000-000-000" `>> -Method PUT `>> -Headers @{'Content-Type' = "application/json"} `>> -Body '{"title": "Updated Book Title"}'
```

**Browser Output (localhost:5006/books/all):**

```
Pretty-print □
[{"author": "John Author", "isbn": "000-000-000", "title": "Updated Book Title", "year": 2024}]
```

## Exercise Two: Asynchronous communication with RabbitMQ

2.1) Duplicate exercise\_one to exercise\_two

2.2) Set up RabbitMQ

2.3) Make some modifications to theUserService

- connect to the RabbitMQ service

The screenshot shows the RabbitMQ Management Interface at `localhost:15672/#/queues`. The interface includes a navigation bar with tabs: Overview, Connections, Channels, Exchanges, **Queues and Streams**, and Admin. The main content area displays a table of queues. One queue is listed: `/borrow_queue` (classic type, running state, 0 ready, 0 unacked, 0 total messages). Below the table is a pagination section and a link to add a new queue.

- Remember to change the username and password on the second line to read from environment variables

```
# Establish connection to RabbitMQ
credentials = pika.PlainCredentials(
    os.getenv("RABBITMQ_DEFAULT_USER", "guest"),
    os.getenv("RABBITMQ_DEFAULT_PASS", "guest")
)
```

- Run docker-compose up and make sure the UserService works as intended. You can use curl or postman to make a POST request to borrow a book

```
users_db=# \dt
      List of relations
 Schema |   Name    | Type  | Owner
-----+-----+-----+-----+
 public | books    | table | postgres
 public | borrows  | table | postgres
 public | users    | table | postgres
(3 rows)
```

localhost:5002/users/all

Pretty-print □

```
[{"email": "alice.smith@example.com", "firstname": "Alice", "lastname": "Smith", "studentid": "S10001"}, {"email": "bob.johnson@example.com", "firstname": "Bob", "lastname": "Johnson", "studentid": "S10002"}, {"email": "charlie.brown@example.com", "firstname": "Charlie", "lastname": "Brown", "studentid": "S10003"}, {"email": "diana.williams@example.com", "firstname": "Diana", "lastname": "Williams", "studentid": "S10004"}, {"email": "eve.davis@example.com", "firstname": "Eve", "lastname": "Davis", "studentid": "S10005"}, {"email": "frank.miller@example.com", "firstname": "Frank", "lastname": "Miller", "studentid": "S10006"}]
```

localhost:5006/books/all

Pretty-print □

```
[{"author": "Mark Lutz", "bookid": "B10001", "title": "Learning Python", "year": 2021}, {"author": "Luciano Ramalho", "bookid": "B10002", "title": "Fluent Python", "year": 2019}, {"author": "Robert C. Martin", "bookid": "B10003", "title": "Clean Code", "year": 2008}, {"author": "Erich Gamma", "bookid": "B10004", "title": "Design Patterns", "year": 1994}, {"author": "Andrew Hunt", "bookid": "B10005", "title": "The Pragmatic Programmer", "year": 1999}, {"author": "Thomas H. Cormen", "bookid": "B10006", "title": "Introduction to Algorithms", "year": 2009}]
```

Code for adding a user:

```
Invoke-WebRequest -Uri "http://localhost:5002/users/add" -Method POST -Headers @{"Content-Type" = "application/json"} -Body '{ "studentid": "S10001", "firstname": "Alice", "lastname": "Smith", "email": "alice.smith@example.com"}'
```

Code for adding a book:

```
Invoke-WebRequest -Uri "http://localhost:5006/books/add" -Method POST -Headers @{"Content-Type" = "application/json"} -Body '{ "bookid": "B10001", "title": "Learning Python", "author": "Mark Lutz", "year": 2021 }'
```

```
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_two\BorrowService> Invoke-WebRequest -Uri "http://localhost:5008/borrow" -Method POST -Headers @{
>>   "Content-Type" = "application/json"
>> } -Body '{
>>   "studentid": "S10001",
>>   "bookid": "B10001"
>> }'

StatusCode : 201
StatusDescription : CREATED
Content : {"message": "Book B10001 successfully borrowed by student S10001."}

RawContent : HTTP/1.1 201 CREATED
Connection: close
Content-Length: 67
Content-Type: application/json
Date: Fri, 29 Nov 2024 10:31:34 GMT
Server: Werkzeug/3.1.3 Python/3.10.15

        {"message": "Book B10001 succe...
Forms : {}
Headers : {[Connection, close], [Content-Length, 67], [Content-Type, application/json], [Date, Fri, 29 Nov 2024 10:31:34 GMT]...}
Images : {}
InputFields : {}
Links : {}
ParsedHtml : mshtml.HTMLDocumentClass
RawContentLength : 67
```

## Invalid Students and Books Check Functions:

```
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_two\BorrowService> Invoke-WebRequest -Uri "http://localhost:5008/bor
row" -Method POST -Headers @{
>>   "Content-Type" = "application/json"
>> } -Body '{
>>   "studentid": "InvalidStudent",
>>   "bookid": "B10002"
>> }'
Invoke-WebRequest : {"error":"Student InvalidStudent does not exist."}
At line:1 char:1
+ Invoke-WebRequest -Uri "http://localhost:5008/borrow" -Method POST -H ...
+ ~~~~~
+ CategoryInfo          : InvalidOperationException: (System.Net.HttpWebRequest:HttpWebRequest) [Invoke-WebRequest], WebException
+ FullyQualifiedErrorId : WebCmdletWebResponseException,Microsoft.PowerShell.Commands.InvokeWebRequestCommand
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_two\BorrowService> Invoke-WebRequest -Uri "http://localhost:5008/bor
row" -Method POST -Headers @{
>>   "Content-Type" = "application/json"
>> } -Body '{
>>   "studentid": "S10001",
>>   "bookid": "InvalidBook"
>> }'
Invoke-WebRequest : {"error":"Book InvalidBook does not exist."}
At line:1 char:1
+ Invoke-WebRequest -Uri "http://localhost:5008/borrow" -Method POST -H ...
+ ~~~~~
+ CategoryInfo          : InvalidOperationException: (System.Net.HttpWebRequest:HttpWebRequest) [Invoke-WebRequest], WebException
+ FullyQualifiedErrorId : WebCmdletWebResponseException,Microsoft.PowerShell.Commands.InvokeWebRequestCommand
```

## Student can only borrow 5 books:

```
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_two\BorrowService> Invoke-WebRequest -Uri "http://localhost:5008/bor
row" -Method POST -Headers @{
>>   "Content-Type" = "application/json"
>> } -Body '{
>>   "studentid": "S10001",
>>   "bookid": "B10006"
>> }'
Invoke-WebRequest : {"error":"Student S10001 has already borrowed 5 books."}
At line:1 char:1
+ Invoke-WebRequest -Uri "http://localhost:5008/borrow" -Method POST -H ...
+ ~~~~~
+ CategoryInfo          : InvalidOperationException: (System.Net.HttpWebRequest:HttpWebRequest) [Invoke-WebRequest], WebException
+ FullyQualifiedErrorId : WebCmdletWebResponseException,Microsoft.PowerShell.Commands.InvokeWebRequestCommand
```

## c. Be able to return a list of all books borrowed by a given user at a given time

```
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_two\BorrowService> Invoke-WebRequest -Uri "http://localhost:5008/bor
rows/S10001" -Method GET

StatusCode      : 200
StatusDescription : OK
Content         : [{"bookid": "B10001", "borrow_date": "Fri, 29 Nov 2024 10:31:34 GMT", "studentid": "S10001"}, {"bookid": "B10002", "borrow_date": "Fri, 29 Nov 2024 10:36:46 GMT", "studentid": "S10001"}, {"bookid": "B10003", "borro...
RawContent      : HTTP/1.1 200 OK
                  Connection: close
                  Content-Length: 437
                  Content-Type: application/json
                  Date: Fri, 29 Nov 2024 10:37:48 GMT
                  Server: Werkzeug/3.1.3 Python/3.10.15
Forms           : {}
Headers         : {[Connection, close], [Content-Length, 437], [Content-Type, application/json], [Date, Fri, 29 Nov 2024 10:37:48 GMT]...}
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : mshtml.HTMLDocumentClass
RawContentLength : 437
```

Verification using adminer:

<http://localhost:8080>

Language: English ▾ PostgreSQL » database » users\_db » public » Select: borrows

Adminer 4.8.1

DB: users\_db ▾ Schema: public ▾

SQL command Import  
Export Create table

select books  
select borrows  
select users

Select data Show structure Alter table New item

Select Search Sort Limit Text length Action

Limit: 50 Text length: 100 Action: Select

SELECT \* FROM "borrows" LIMIT 50 (0.001 s) Edit

<input type="checkbox"/> Modify	studentid	bookid	borrow_date
<input type="checkbox"/> edit	S10001	B10001	2024-11-29 10:31:34.02106
<input type="checkbox"/> edit	S10001	B10002	2024-11-29 10:36:46.243959
<input type="checkbox"/> edit	S10001	B10003	2024-11-29 10:36:50.043895
<input type="checkbox"/> edit	S10001	B10004	2024-11-29 10:36:53.532729
<input type="checkbox"/> edit	S10001	B10005	2024-11-29 10:36:56.20415

Whole result Modify Selected (0) Export (5)  
 5 rows

Import

### Student returning a borrowed book:

```
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_two\BorrowService> Invoke-WebRequest -Uri "http://localhost:5008/return" -Method DELETE -Headers @{
>>     "Content-Type" = "application/json"
>> } -Body @{
>>     "studentid": "S10001",
>>     "bookid": "B10001"
>> }

StatusCode : 200
StatusDescription : OK
Content : {"message":"Book B10001 successfully returned by student S10001."}

RawContent : HTTP/1.1 200 OK
Connection: close
Content-Length: 67
Content-Type: application/json
Date: Fri, 29 Nov 2024 10:53:41 GMT
Server: Werkzeug/3.1.3 Python/3.10.15

        {"message":"Book B10001 successful...
Forms : {}
Headers : {[Connection, close], [Content-Length, 67], [Content-Type, application/json], [Date, Fri, 29 Nov 2024 10:53:41 GMT]...}
Images : {}
InputFields : {}
Links : {}
ParsedHtml : mshtml.HTMLDocumentClass
RawContentLength : 67
```

Exercise Three: Deployment with Kubernetes

1. Duplicate exercise\_two to exercise\_three. All new changes should now be made in this folder.
2. For our deployment to work, we need convert our docker-compose.yml file into Kubernetes deployment. Follow the instruction on the kompose webpage.
3. To test and make sure each service is working, setup port-forward from the pod to the host

```
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_three> kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
adminer-5f7f659fc7-cmtvj   1/1     Running   0          171m
bookservice-97ccd7999-pqxhp 1/1     Running   0          6m56s
borrowservice-cf6bc8575-msmdh 1/1     Running   11 (31m ago) 58m
database-7c8b546d69-npssd   1/1     Running   0          41m
debug-pod        1/1     Running   0          32m
dns-test         1/1     Running   0          33m
rabbitmq-84f79c7685-ngk8x   1/1     Running   0          76m
rabbitmq-test    0/1     Completed  0          65m
rabbitmq-tester  0/1     Completed  0          84m
userservice-dcf4f88d7-29kc6  1/1     Running   11 (31m ago) 58m
```

Port forwarding works for each pod:

```
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_three> kubectl port-forward pod/rabbitmq-84f79c7685-ngk8x 5672:5672
Forwarding from 127.0.0.1:5672 -> 5672
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_three> kubectl port-forward pod/rabbitmq-84f79c7685-ngk8x 15672:15672
Forwarding from 127.0.0.1:15672 -> 15672
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_three> kubectl port-forward pod/userservice-dcf4f88d7-29kc6 5002:5002
Forwarding from 127.0.0.1:5002 -> 5002
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_three> kubectl port-forward pod/borrowservice-cf6bc8575-msmdh 5008:5008
Forwarding from 127.0.0.1:5008 -> 5008
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_three> kubectl port-forward pod/bookservice-97ccd7999-pqxhp 5006:5006
Forwarding from 127.0.0.1:5006 -> 5006
```

And testing using curl again:

```
PS C:\Users\35385\Desktop\CS_Autumn2024\Cloud\PRACTICAL2_Tom_Gibson_23222095\exercise_three> curl http://localhost:5006/books/all

StatusCode      : 200
StatusDescription : OK
Content          : [{"author": "Mark Lutz", "bookid": "B10001", "title": "Learning Python", "year": 2021}, {"author": "Luciano Ramalho", "bookid": "B10002", "title": "Fluent Python", "year": 2019}, {"author": "Robert C. Martin", "bookid": ...}
RawContent       : HTTP/1.1 200 OK
                  Connection: close
                  Content-Length: 513
                  Content-Type: application/json
                  Date: Fri, 29 Nov 2024 15:49:35 GMT
                  Server: Werkzeug/3.1.3 Python/3.10.15

Forms            : {}
Headers          : {[Connection, close], [Content-Length, 513], [Content-Type, application/json], [Date, Fri, 29 Nov 2024 15:49:35 GMT]...}
Images           : {}
InputFields      : {}
Links            : {}
ParsedHtml       : mshtml.HTMLDocumentClass
RawContentLength : 513
```