



IT4020

Modern Topics in IT

4th Year, 2nd Semester

Microservices

Assignment 04

Submitted to
Sri Lanka Institute of Information Technology

In partial fulfillment of the requirements for the
Bachelor of Science Special Honors Degree in Information Technology

14-June-2022

Declaration

I certify that this report does not incorporate without acknowledgement, any material previously submitted for a degree or diploma in any university, and to the best of my knowledge and belief it does not contain any material previously published or written by another person, except where due reference is made in text.

Group member details:

Registration Number	Name
IT18210002	Kasthuriarachchi K.M.W
IT18540536	Lanerolle T.Y
IT18218572	Prashadika W.M.J
IT18062120	Wickramasinghe W.A.P.C

Drive link: -

https://drive.google.com/file/d/1g1dQzqmHxNxc_tZ-CvRvXfOPxVgM2zzi/view?usp=sharing

Table of Contents

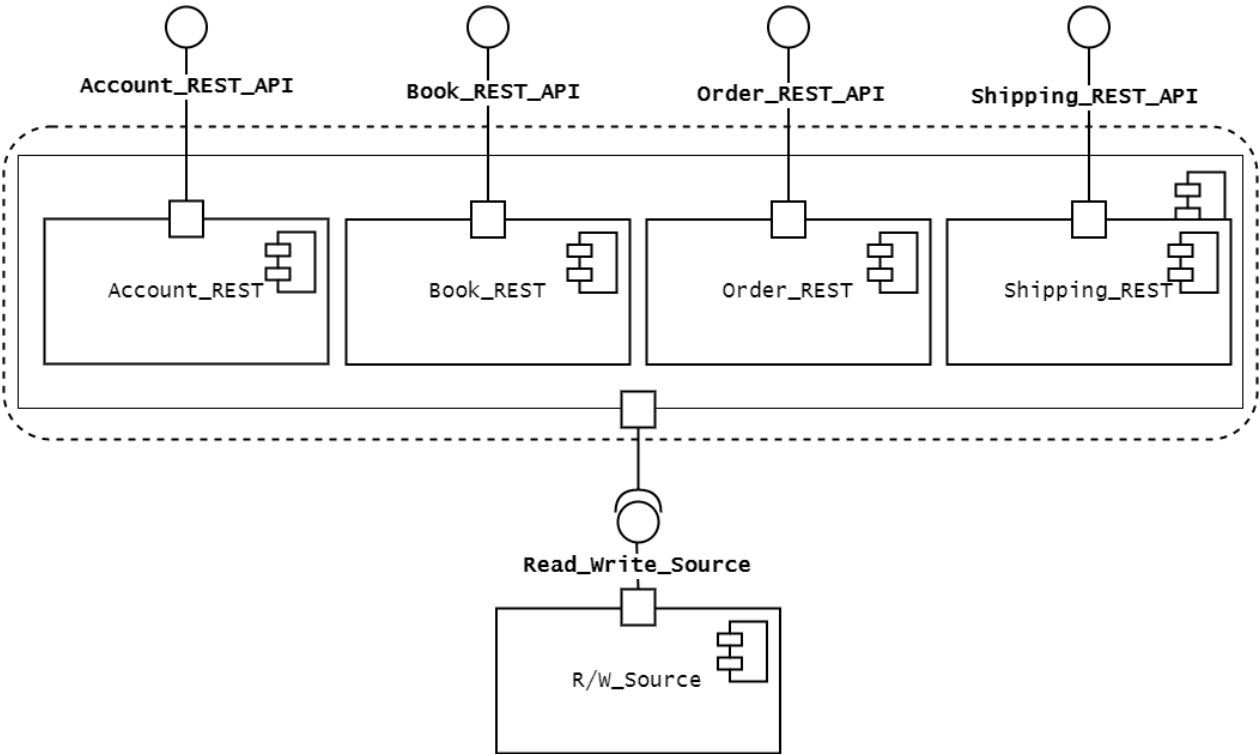
01. Explain the solution using a component diagram

02. Member's contribution

2.1 Screenshots of the outputs

2.2 Steps to deploy and test

Component Diagram



Solution

Here, we are discussing about an online book shop with microservices architecture. This application includes 4 modules. Account service, Book inventory service, Order service and Shipping service. Account service is responsible for supply functions relates to account management. Book inventory service is responsible for supply functions related to book inventory management. Order inventory service is responsible for supply functions related to order management. Shipping service is responsible for supply functions related shipping management. Modules are distributed among 4 members. All modules can perform crud operations.

IT18210002 – online bookshop Order service

```
@RestController
@RequestMapping("/orders")

public class OrderController {
    @Autowired
    private OrderServiceImpl orderService;

    @PostMapping(consumes = "application/json", produces = "application/json")
    public @ResponseBody
    OrderResponse makeOrder(@RequestBody OrderRequest request) {
        System.out.println("Order Details : " + request);
        return orderService.makeOrder(request);
    }
}

@EnableSwagger2
@SpringBootApplication
public class OrderServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(OrderServiceApplication.class, args);
    }
}
```

Book Inventory Service (IT18540536 – Lanerolle T.Y)

Screenshots of the output

The screenshot displays the Eclipse IDE interface. The main editor shows the `BookController.java` file with the following code:

```
21 @PostMapping(consumes = "application/json", produces = "application/json")
22 public @ResponseBody BookResponse addBook(@RequestBody BookRequest bookRequest) {
23     var bookResponse = new BookResponse();
24     String newbookid = UUID.randomUUID().toString();
25     var bookReq = bookRequest;
26     bookReq.setBookId(newbookid);
27     bookResponse.setBookId(newbookid);
28     bookResponse.setMessage("New Book Added successfully");
29     allBooks.add(bookReq);
30     return bookResponse;
31 }
32
33 @PutMapping(consumes = "application/json", produces = "application/json")
34 public @ResponseBody BookResponse updateBook(@RequestParam String bookid,
35 @RequestBody BookRequest bookRequest) {
```

The right-hand side of the IDE shows the Outline view with the following structure:

- com.mtiti.microservices.bookservice.controller
- BookController
 - allBooks: List<BookRequest>
 - index(): String
 - addBook(@RequestBody BookRequest): BookResponse
 - updateBook(@RequestParam String, @RequestBody BookRequest): BookResponse
 - allBooks(): List<BookRequest>
 - searchBook(@RequestParam String): BookRequest
 - removeBook(@RequestParam String): BookResponse

The bottom console window shows the output of the Spring Boot application:



```
2022-06-14 17:09:15.620 INFO 1724 --- [main] c.m.m.b.BookServiceApplication : Starting BookServiceApplication on DESKTOP-L0J00NI with PID 1724 (I
2022-06-14 17:09:15.625 INFO 1724 --- [main] c.m.m.b.BookServiceApplication : No active profile set, falling back to default profiles: default
2022-06-14 17:09:17.573 INFO 1724 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-06-14 17:09:17.591 INFO 1724 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-06-14 17:09:17.592 INFO 1724 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.33]
2022-06-14 17:09:17.756 INFO 1724 --- [main] o.s.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-06-14 17:09:17.756 INFO 1724 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 2051 ms
2022-06-14 17:09:18.046 INFO 1724 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2022-06-14 17:09:18.395 INFO 1724 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2022-06-14 17:09:18.402 INFO 1724 --- [main] c.m.m.b.BookServiceApplication : Started BookServiceApplication in 3.69 seconds (JVM running for 4.5
```

Book Service Collection

✓	Microservices - Book
	6 requests
GET	default
POST	add new book
PUT	update book
PUT	remove book
GET	get all books
GET	search book

Default request

▶ default

Examples 0 ▾ | BUILD  

GET ▾

http://localhost:8080/book/

Send ▾

Save ▾

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Code

Query Params


	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (5)

Test Results

 Status: 200 OK Time: 178 ms Size: 195 B Save Response ▾


Pretty



Raw

Preview

Visualize

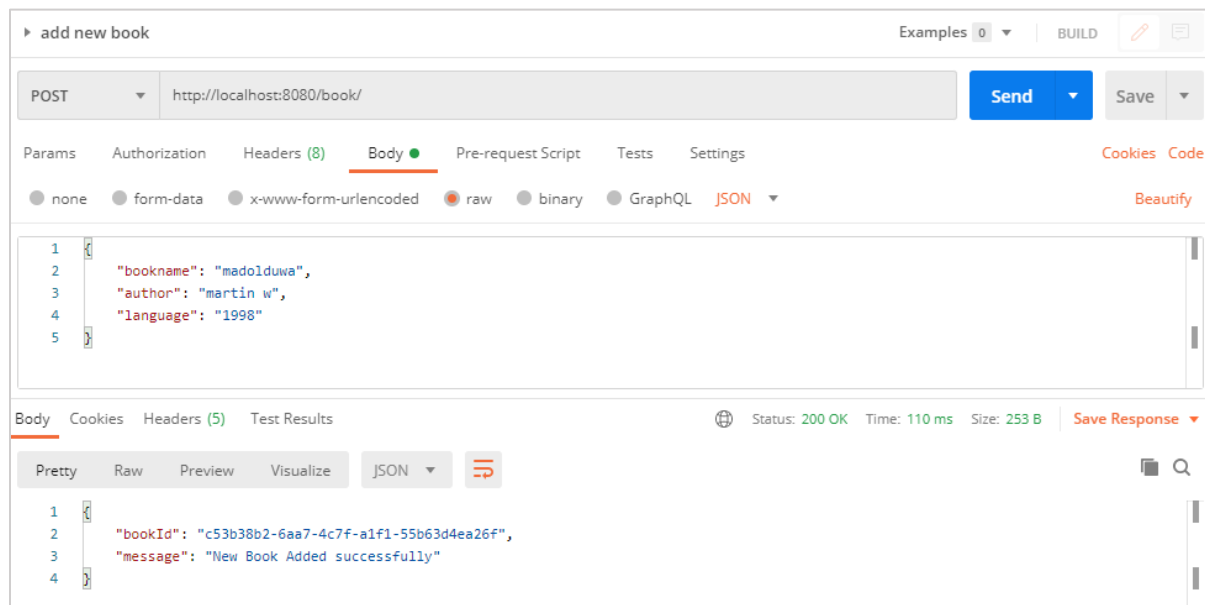
Text ▾



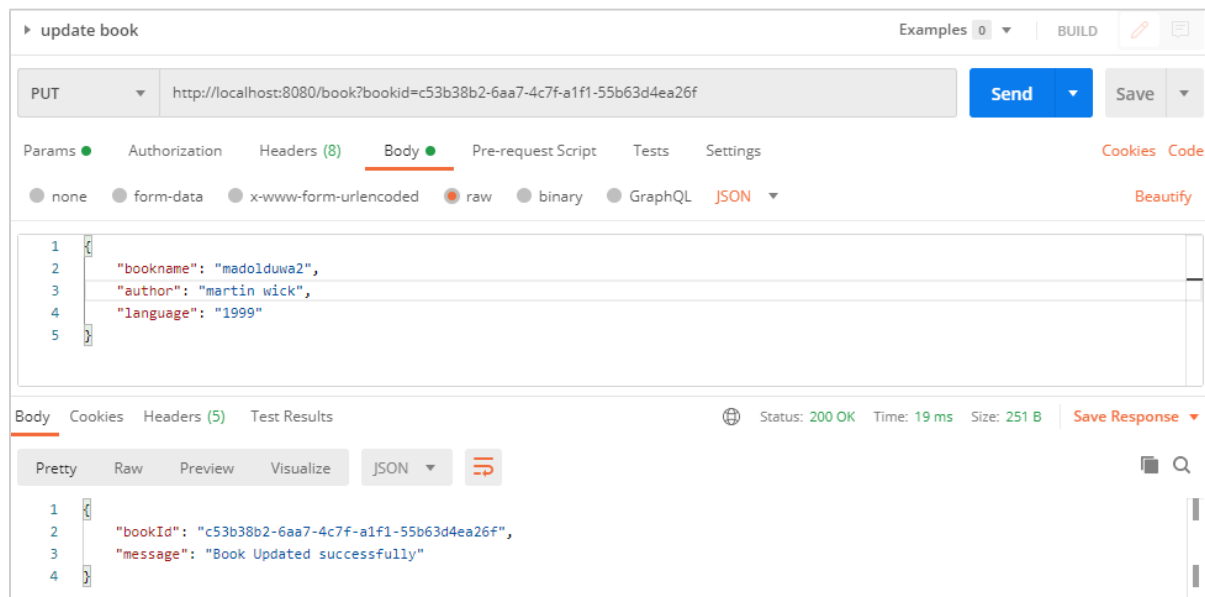
 

1 Hello Welcome to Book Service !

Add new book request



Update book request



Remove book request

remove book

Examples 0 BUILD

DELETE http://localhost:8080/book?bookid=c53b38b2-6aa7-4c7f-a1f1-55b63d4ea26f Send Save

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies Code

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	bookid	c53b38b2-6aa7-4c7f-a1f1-55b63d4ea26f			
	Key	Value	Description		

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 14 ms Size: 251 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "bookId": "c53b38b2-6aa7-4c7f-a1f1-55b63d4ea26f",
3   "message": "Book Removed successfully"
4 }
```

Get all books request

get all books

Examples 0BUILD

GET

http://localhost:8080/book/all

Send

Save

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Code

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body

Cookies

Headers (5)

Test Results

Status: 200 OKTime: 23 msSize: 280 BSave Response

Pretty

Raw

Preview

Visualize

JSON

```
1 [
2   {
3     "bookid": "c53b38b2-6aa7-4c7f-a1f1-55b63d4ea26f",
4     "bookname": "madolduwa2",
5     "author": "martin wick",
6     "language": "1999"
7   }
8 ]
```

Search book request

search book Examples 0 BUILD

GET http://localhost:8080/book/search?bookid=c53b38b2-6aa7-4c7f-a1f1-55b63d4ea26f Send Save

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> bookid	c53b38b2-6aa7-4c7f-a1f1-55b63d4ea26f			
Key	Value	Description		

Body Cookies Headers (5) Test Results Status: 200 OK Time: 13 ms Size: 278 B Save Response

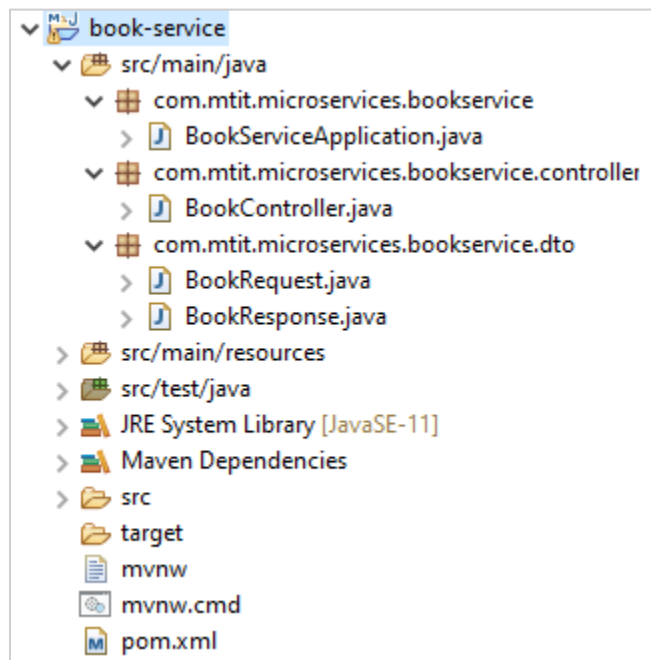
Pretty Raw Preview Visualize JSON ≡

```

1 {
2   "bookid": "c53b38b2-6aa7-4c7f-a1f1-55b63d4ea26f",
3   "bookname": "madolduwa2",
4   "author": "martin wick",
5   "language": "1999"
6 }
```

Steps to deploy and test

Structure



Book Service Application

```
BookServiceApplication.java
1 package com.mtit.microservices.bookservice;
2
3 import org.springframework.boot.SpringApplication;
4
5
6 @SpringBootApplication
7 public class BookServiceApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(BookServiceApplication.class, args);
11     }
12 }
13
14
```

Book Request

```
BookRequest.java
1 package com.mtit.microservices.bookservice.dto;
2
3 public class BookRequest {
4
5     private String bookid;
6     private String bookname;
7     private String author;
8     private String language;
9
10    public String getBookid() {
11        return bookid;
12    }
13
14    public void setBookid(String bookid) {
15        this.bookid = bookid;
16    }
17
18    public String getBookname() {
19        return bookname;
20    }
21
22    public void setBookname(String bookname) {
23        this.bookname = bookname;
24    }
25
26    public String getAuthor() {
27        return author;
28    }
29
30    public void setAuthor(String author) {
31        this.author = author;
32    }
33
34    public String getLanguage() {
35        return language;
36    }
37 }
```

Book Response


```
BookResponse.java
1 package com.mtit.microservices.bookservice.dto;
2
3 public class BookResponse {
4
5     private String bookId;
6     private String message;
7
8     public String getBookId() {
9         return bookId;
10    }
11
12    public void setBookId(String bookId) {
13        this.bookId = bookId;
14    }
15
16    public String getMessage() {
17        return message;
18    }
19
20    public void setMessage(String message) {
21        this.message = message;
22    }
23 }
24
```

Book Controller

```
BookController.java
1 package com.mtit.microservices.bookservice.controller;
2
3 import com.mtit.microservices.bookservice.dto.BookRequest;
4
5
6
7
8
9
10 @RestController
11 @RequestMapping("/book")
12 public class BookController {
13
14     List<BookRequest> allBooks = new ArrayList<BookRequest>();
15
16     @GetMapping("/")
17     public String index(){
18         return "Hello Welcome to Book Service !";
19     }
20
21     @PostMapping(consumes = "application/json", produces = "application/json")
22     public @ResponseBody BookResponse addBook(@RequestBody BookRequest bookRequest) {
23         var bookResponse = new BookResponse();
24         String newbookid = UUID.randomUUID().toString();
25         var bookReq = bookRequest;
26         bookReq.setBookid(newbookid);
27         bookResponse.setBookId(newbookid);
28         bookResponse.setMessage("New Book Added successfully");
29         allBooks.add(bookReq);
30         return bookResponse;
31     }
32
33     @PutMapping(consumes = "application/json", produces = "application/json")
34     public @ResponseBody BookResponse updateBook(@RequestParam String bookid,
35         @RequestBody BookRequest bookRequest) {
36         boolean isExists = false;
37         for (BookRequest bookReq : allBooks) {
38             if(bookReq.getBookid().equals(bookid)) {
39                 isExists = true;
40                 allBooks.remove(bookReq);
41             }
42         }
43     }
44 }
```

Book Controller – index method

```
@GetMapping("/")
public String index(){
    return "Hello Welcome to Book Service !";
}
```

Book Controller – add book method

```
@PostMapping(consumes = "application/json", produces = "application/json")
public @ResponseBody BookResponse addBook(@RequestBody BookRequest bookRequest) {
    var bookResponse = new BookResponse();
    String newbookid = UUID.randomUUID().toString();
    var bookReq = bookRequest;
    bookReq.setBookid(newbookid);
    bookResponse.setBookId(newbookid);
    bookResponse.setMessage("New Book Added successfully");
    allBooks.add(bookReq);
    return bookResponse;
}
```

Book Controller – update book method

```
@PutMapping(consumes = "application/json", produces = "application/json")
public @ResponseBody BookResponse updateBook(@RequestParam String bookid,
    @RequestBody BookRequest bookRequest) {
    boolean isExists = false;
    for (BookRequest bookReq : allBooks) {
        if(bookReq.getBookid().equals(bookid)) {
            isExists = true;
            allBooks.remove(bookReq);
            break;
        }
    }
    var bookResponse = new BookResponse();
    bookResponse.setBookId(bookid);
    if(isExists) {
        bookRequest.setBookid(bookid);
        allBooks.add(bookRequest);
        bookResponse.setMessage("Book Updated successfully");
    }else {
        bookResponse.setMessage("Book Updated failed");
    }
    return bookResponse;
}
```

Book Controller – get all books method

```

@GetMapping("/all")
public @ResponseBody List<BookRequest> allBooks(){
    return allBooks;
}

```

Book Controller – search book method

```

@GetMapping("/search")
public @ResponseBody BookRequest searchBook(@RequestParam String bookid){
    for (BookRequest bookReq : allBooks) {
        if(bookReq.getBookid().equals(bookid)) {
            return bookReq;
        }
    }
    return null;
}

```

Book Controller – remove book method

```

@DeleteMapping(produces = "application/json")
public @ResponseBody BookResponse removeBook(@RequestParam String bookid) {
    boolean isExists = false;
    for (BookRequest bookReq : allBooks) {
        if(bookReq.getBookid().equals(bookid)) {
            isExists = true;
            allBooks.remove(bookReq);
            break;
        }
    }
    if(isExists) {
        var bookResponse = new BookResponse();
        bookResponse.setBookId(bookid);
        bookResponse.setMessage("Book Removed successfully");
        return bookResponse;
    }else {
        return null;
    }
}

```

Swagger API Document

admins Secured Admin-only calls ^	
PUT	/book update a book inventory item v ↵
DELETE	/book delete a book inventory item v ↵
developers Operations available to regular developers ^	
GET	/book/all get all books v ↵
GET	/book/search search book v ↵
POST	/book adds a book inventory item v ↵
Models ^	
BookRequest > ↵	
BookResponse > ↵	

Activate W
Go to Settings

Create a Book

POST

/book adds a book inventory item

Adds a book to the system

Parameters

Try it out

Name	Description
bookRequest	Add book
object (body)	<div>Example Value Model</div> <div><pre>{ "bookid": "ssad374384-sdsd-sfvdv-vdev", "bookname": "Madolduwa", "author": "Martin M", "language": "Sinhala" }</pre></div> <div>Parameter content type</div> <div>application/json</div>

Responses

Response content type application/json

Code	Description
200	New Book Added successfully
400	invalid input, object invalid

Get all Books

GET

/book/all get all books

^ ↩

By passing in the appropriate options, you can search for available inventory in the system

Try it out

Parameters

No parameters

Responses

Response content type application/json

Code	Description
200	search results matching criteria Example Value Model <pre>[{ "bookId": "ssad374384-sdsd-sfvdv-vdsv", "bookName": "Madoldawa", "author": "Martin W", "language": "Sinhala" }]</pre>
400	bad input parameter

Activate W
Go to Settings

Get searched Book

GET

/book/search search book

By passing in the appropriate options, you can search for available book in the system

Parameters

Try it out

Name	Description
bookid	search book id
string	
(query)	<input type="text" value="bookid"/>

Responses

Response content type application/json

Code	Description
200	<div>search results matching criteria</div> <div>Example Value Model</div> <div><pre>{ { "bookid": "ssad374384-sdsd-sfvdv-vdsv", "bookname": "Madoldana", "author": "Martin W", "language": "Sinhala" } }</pre></div>

| 400 | bad input parameter |

Update Book Details

PUT

/book update a book inventory item

update a book in the system

Parameters

Try it out

Name	Description				
bookid	update book id				
string (query)	<input type="text" value="bookid"/>				
bookRequest	Add book				
object (body)	<table><thead><tr><th>Example Value</th><th>Model</th></tr></thead><tbody><tr><td><pre>{ "bookId": "asand374384-sdsd-sfvdy-vdsv", "bookName": "Madoldawa", "author": "Martin W", "language": "Sinhala" }</pre></td><td></td></tr></tbody></table>	Example Value	Model	<pre>{ "bookId": "asand374384-sdsd-sfvdy-vdsv", "bookName": "Madoldawa", "author": "Martin W", "language": "Sinhala" }</pre>	
Example Value	Model				
<pre>{ "bookId": "asand374384-sdsd-sfvdy-vdsv", "bookName": "Madoldawa", "author": "Martin W", "language": "Sinhala" }</pre>					
Parameter content type <input type="text" value="application/json"/>					

Responses

Response content type application/json

Activate WGo to Settings

Code	Description
------	-------------

Remove a Book

DELETE

/book delete a book inventory item

^ ↩

delete a book in the system

Parameters

Try it out

Name	Description
bookid	remove book id
string	
(query)	<input type="text" value="bookid"/>

Responses

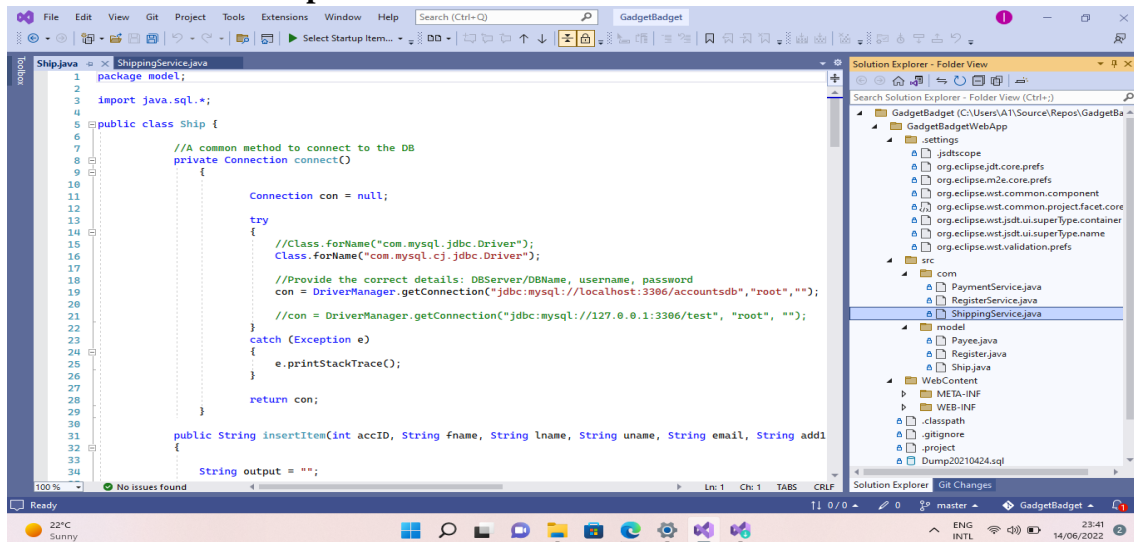
Response content type application/json

Code	Description
200	Book Removed successfully
400	No data

Activate WGo to Settings

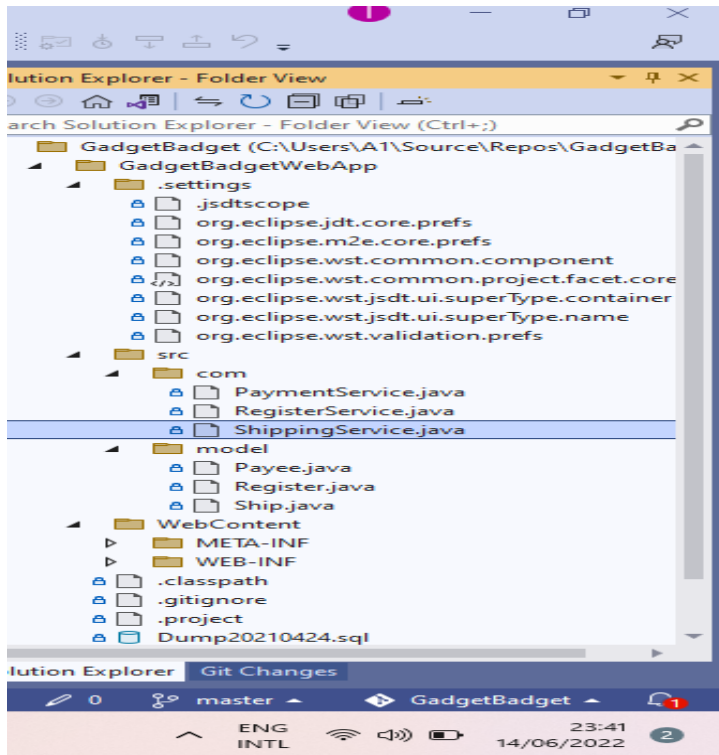
Shipping Service (IT18218572 – Prashadika W.M.J)

Screenshots of the output

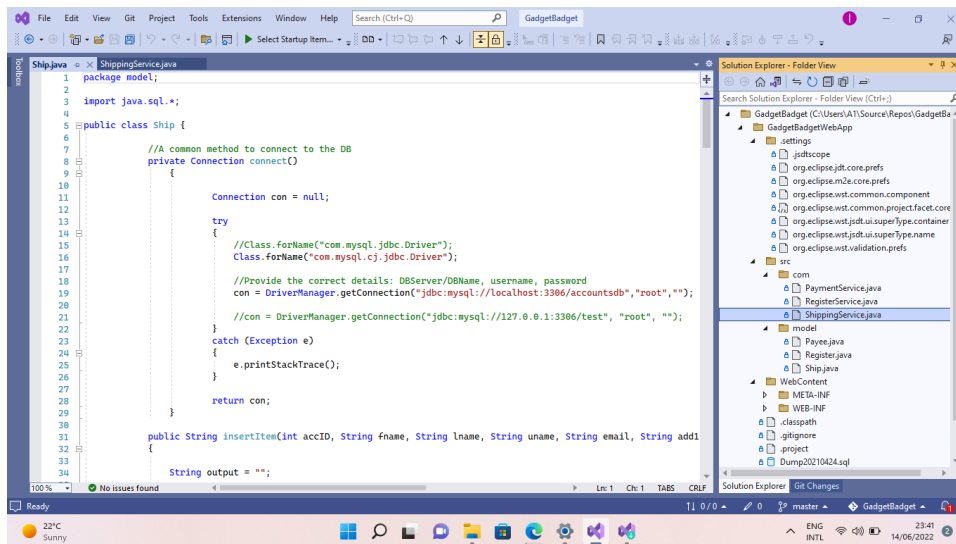


Steps to deploy and test

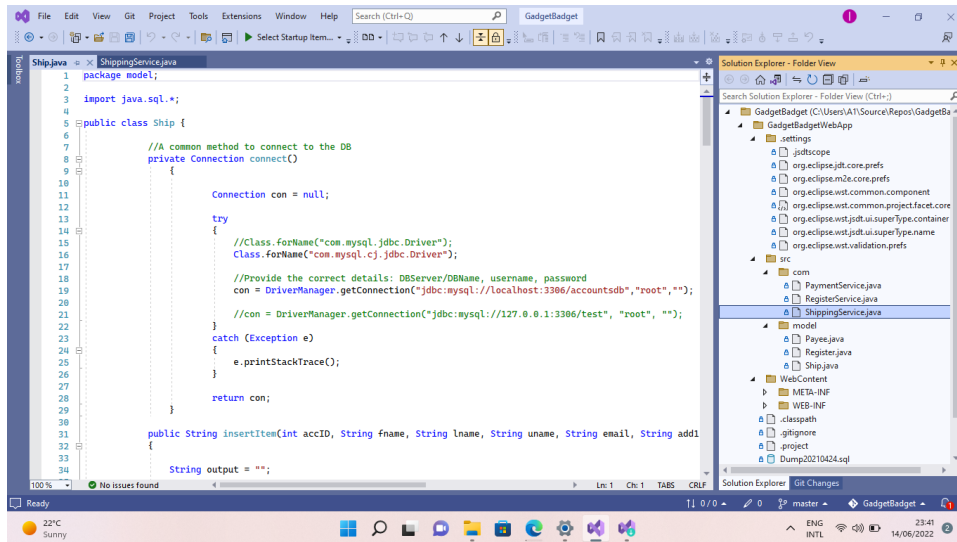
Structure



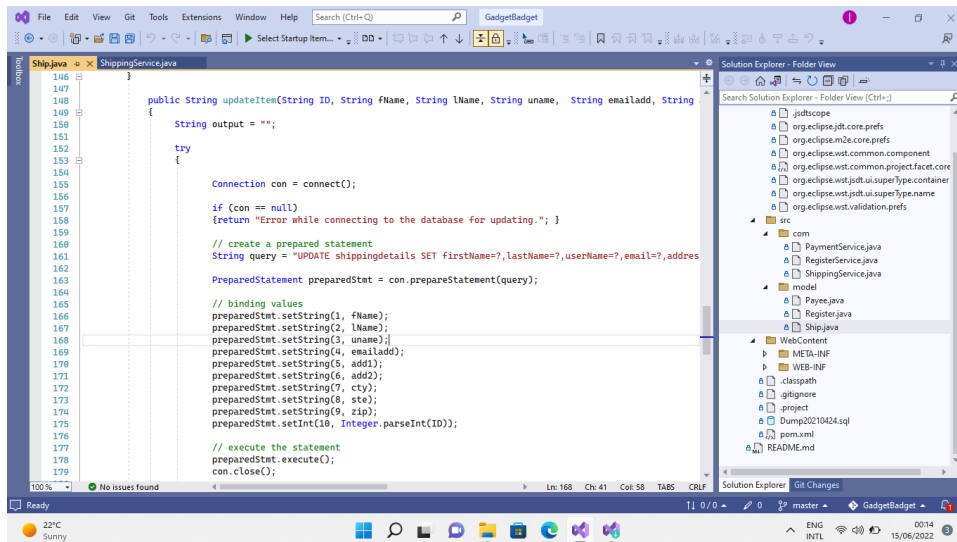
Shipping service



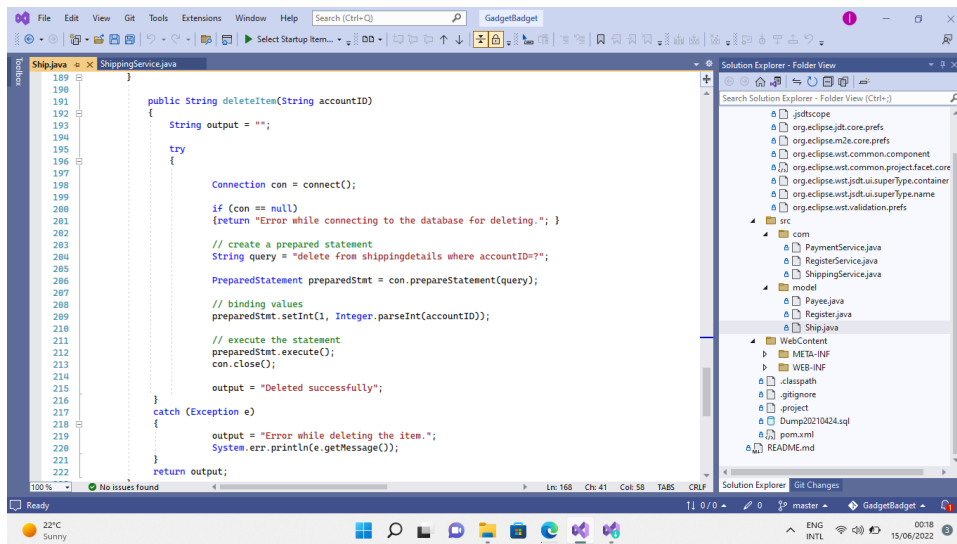
Shipping add method



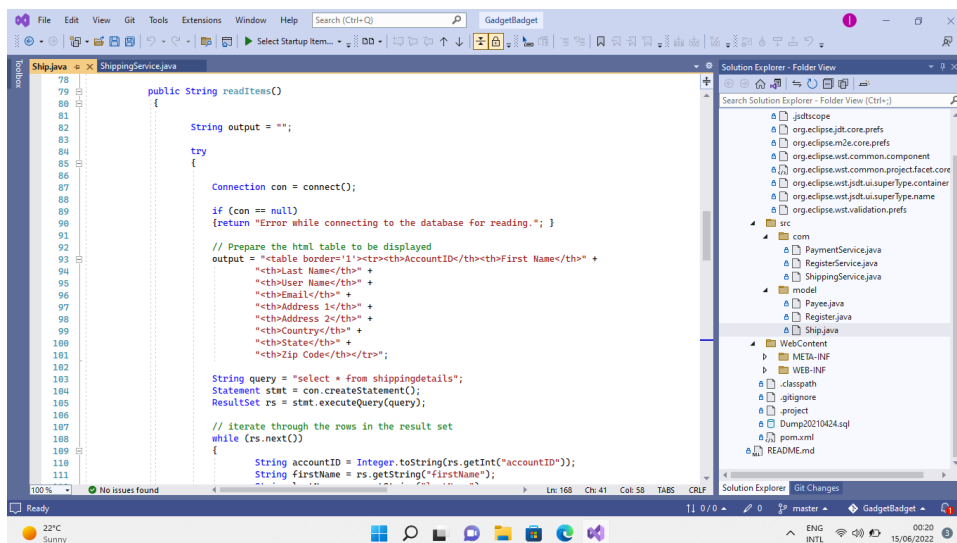
Shipping update method



Shipping delete method



Shipping edit method



Payment services component (IT18062120)

In this book inventory system has payment facility. I used microservice architecture to develop this part. The user who orders the book they can pay for that using this payment service facility. User can insert payment details and pay for the order. If there were any missing details has entered from the user the user can edit those details and submit that details or the can delete the details, which they entered using the system. I attached the screenshots of the outputs below.

Source code link:

<https://drive.google.com/file/d/1yUfLNSn51iXoMpN425CNXSFT2AKDuN4Z/view?usp=sharing>

Flowchart

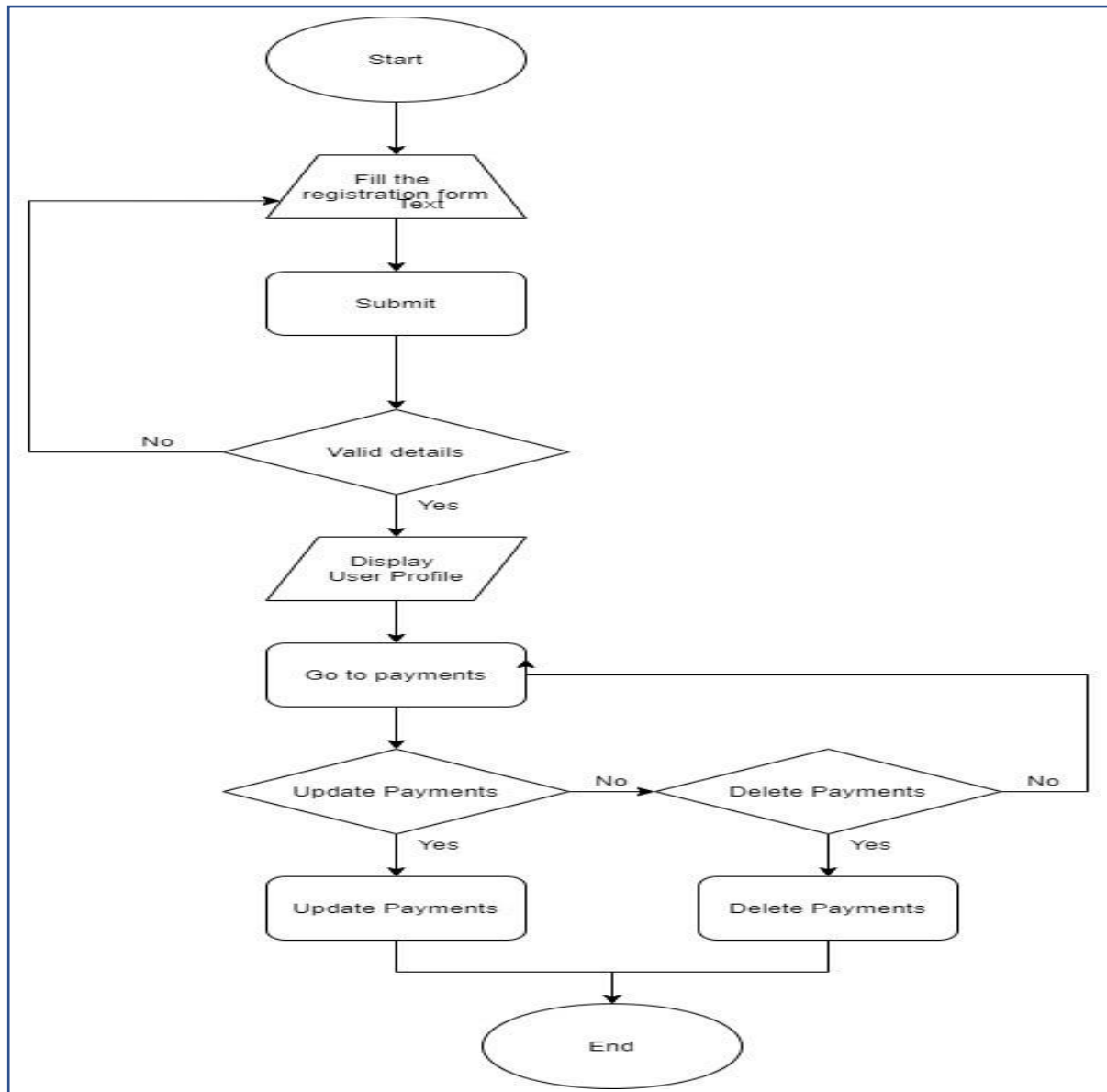


Figure 1

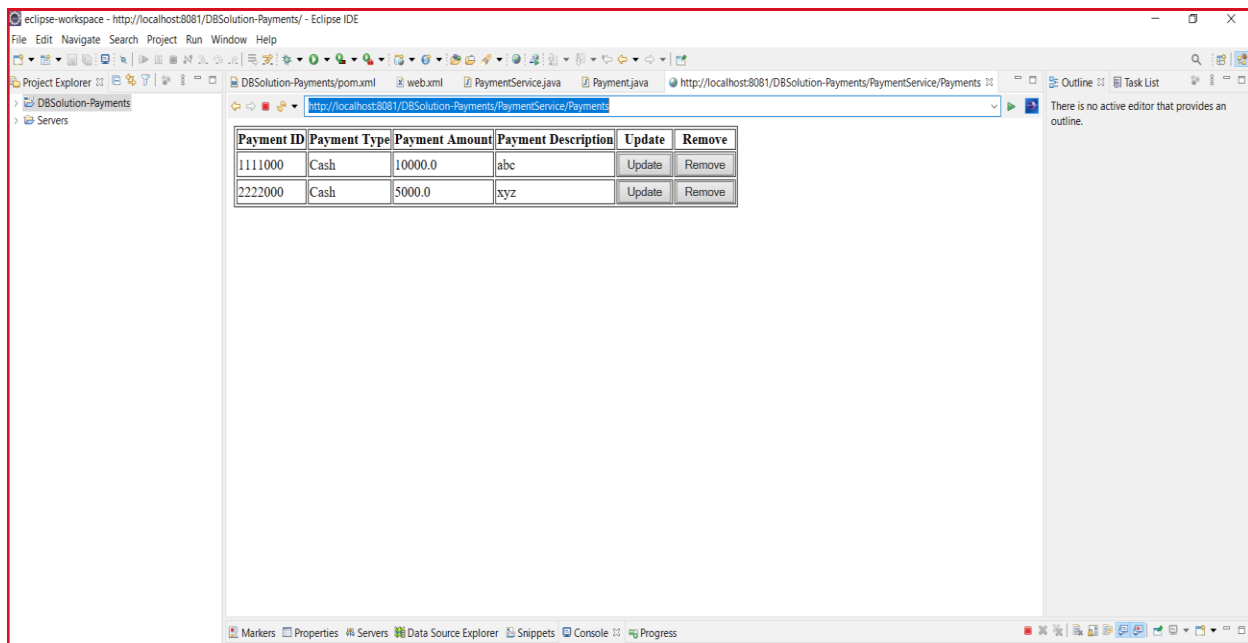


Figure 2

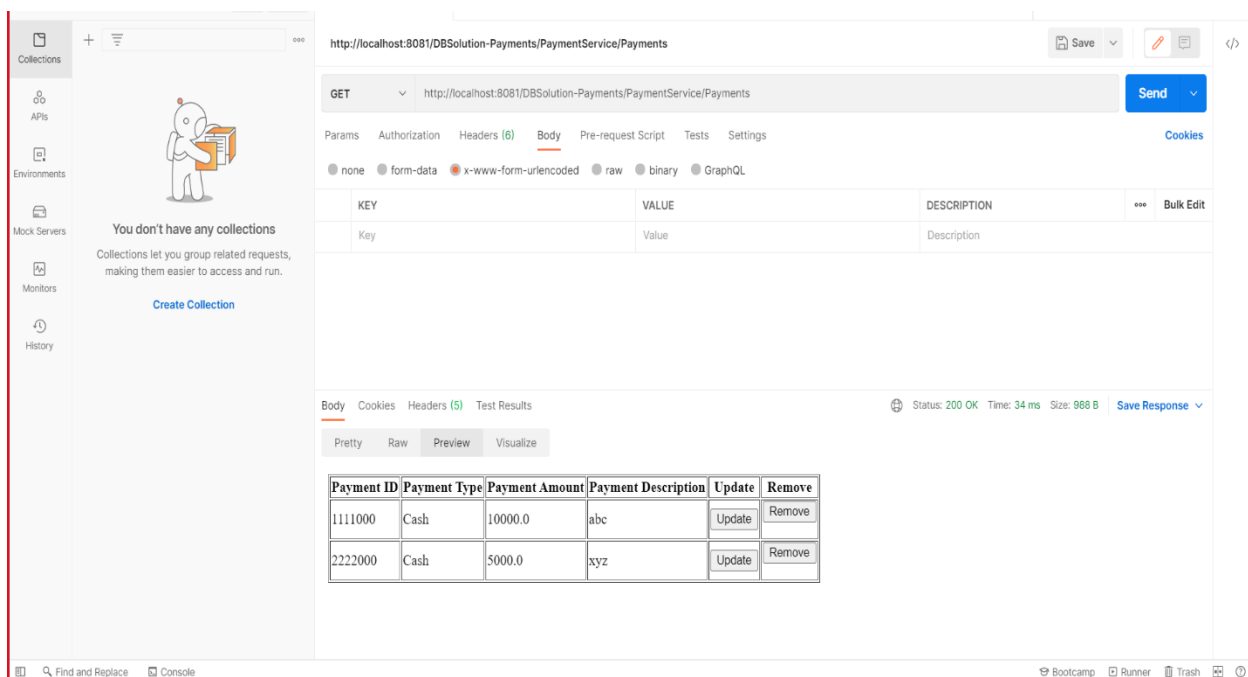


Figure 3

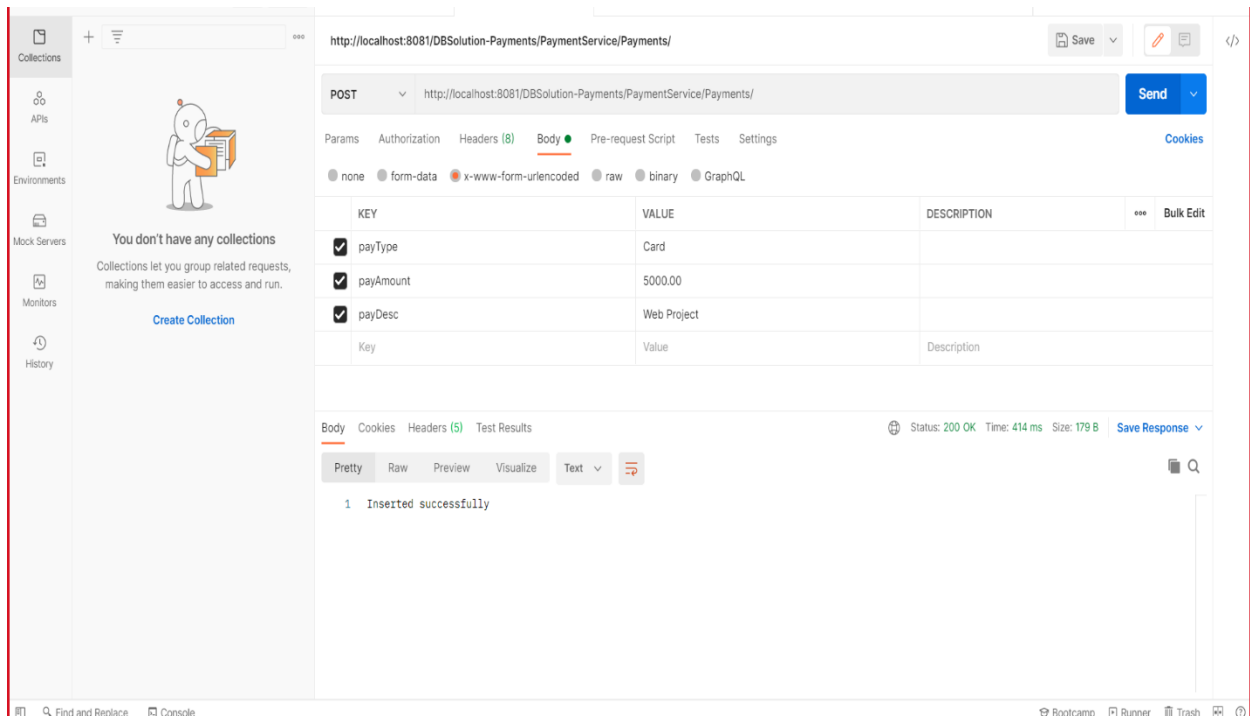


Figure 4

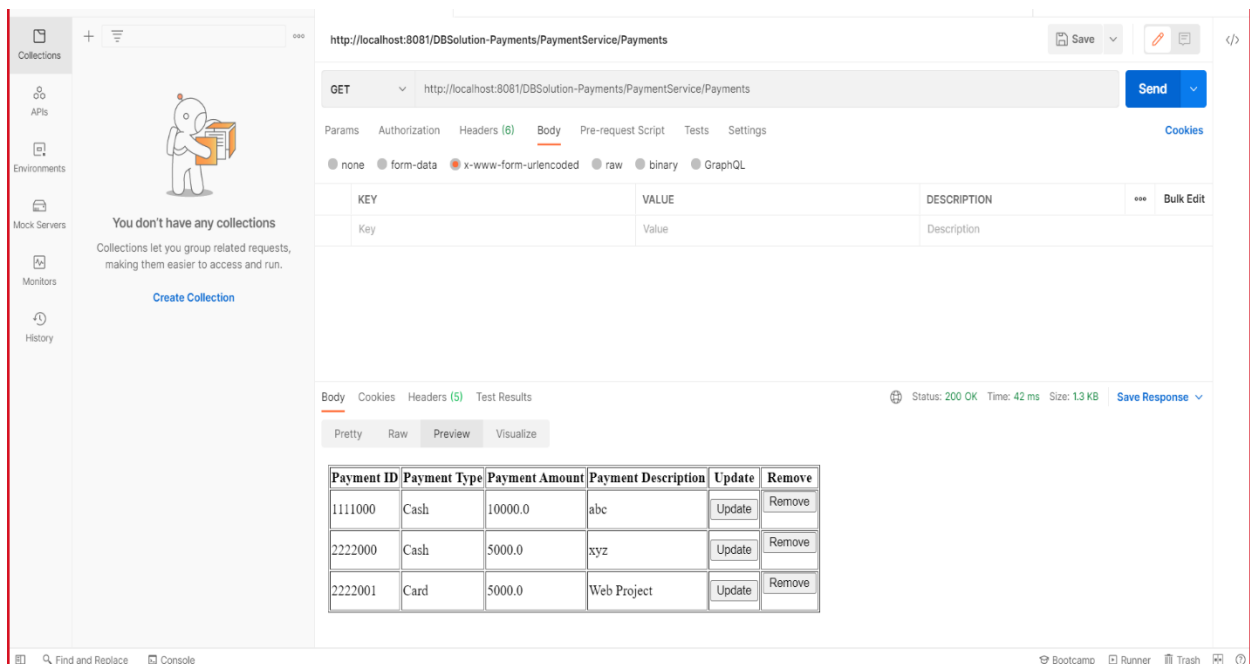


Figure 5

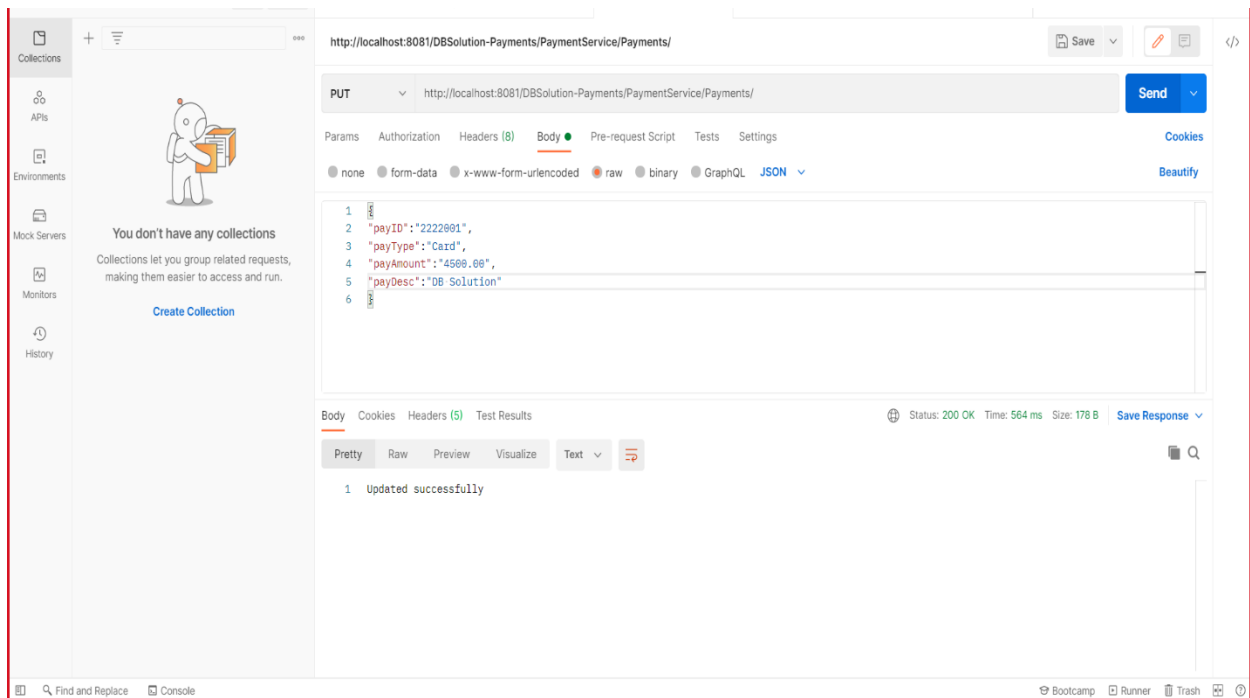


Figure 6

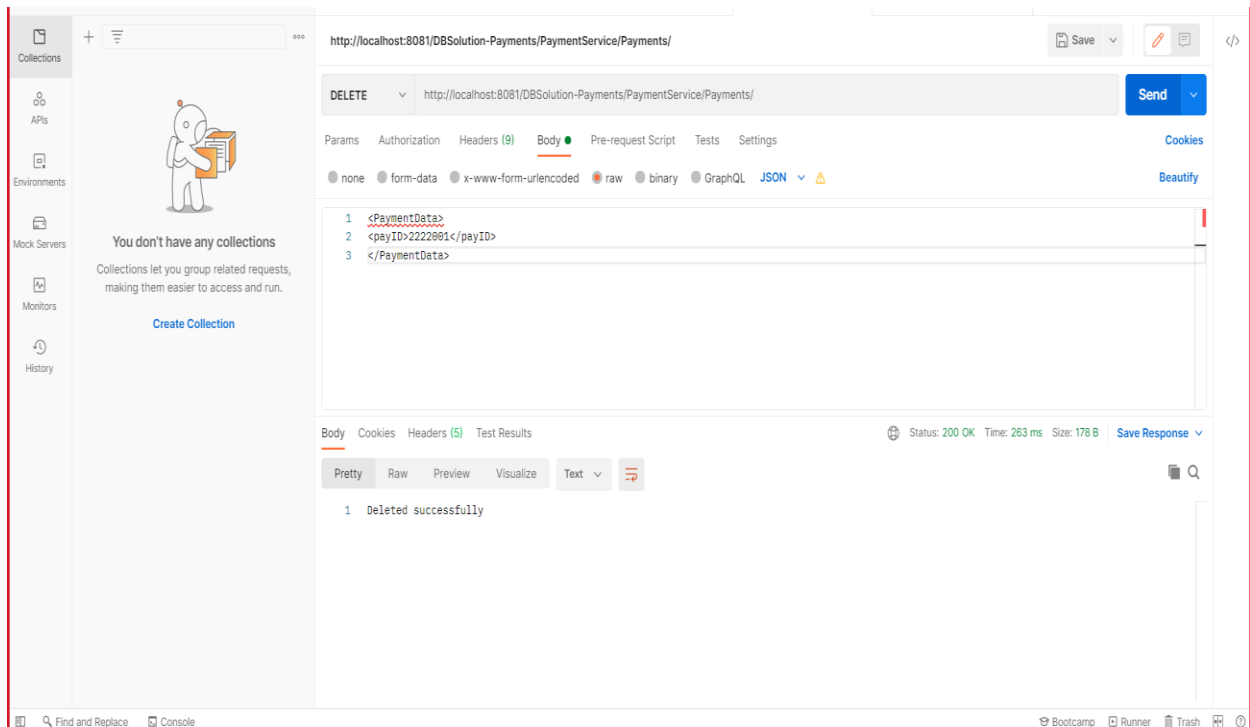


Figure 7

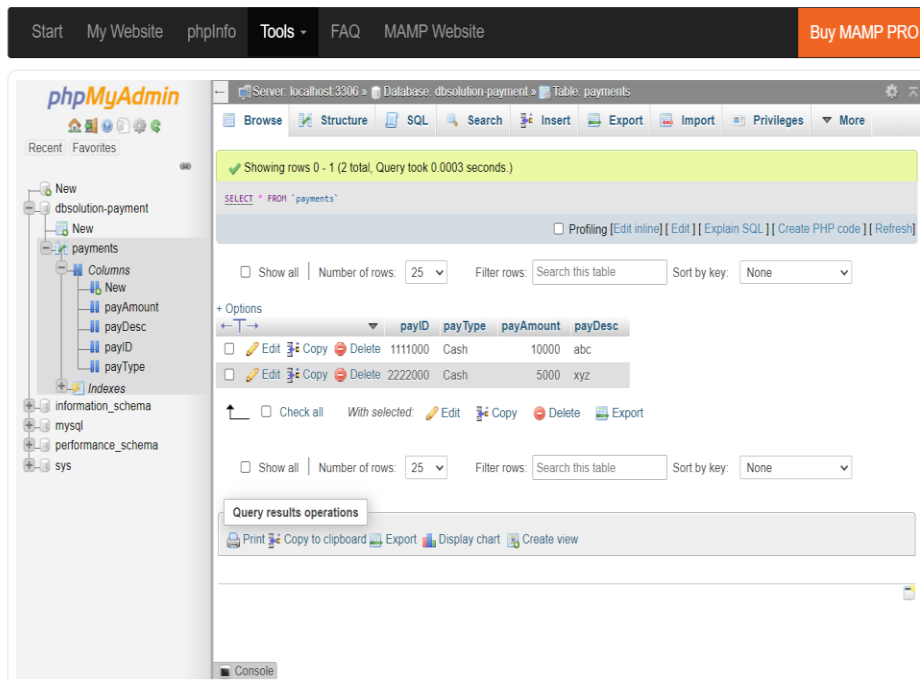


Figure 8