

CSI 5112: Software Engineering

API GUIDE

Shoppers-Marketplace-Group 13

The API can be tested using our **Postman Collection**.

Initial Steps to test the API:

- 1) Download “[marketplace_services_CSI5112.postman_collection.json](#)” and “[TESTS-marketplace_services_CSI5112.postman_collection.json](#)” from GitHub /QA folder and import it to Postman.
- 2) Add a new environment named [env](#).
- 3) Change the Environment variable as

Key:	url
Value:	https://localhost:7136/api or http://3.93.177.49/api

- 4) Change the Environment to [env](#) for running the Test

List of Endpoints

1. Users:

<https://localhost:7136/api/User>

2. Product:

<https://localhost:7136/api/Product>

3. Category:

<https://localhost:7136/api/Category>

4. Order:

<https://localhost:7136/api/Order>

5. Question:

<https://localhost:7136/api/Question>

6. Answer:

<https://localhost:7136/api/Answer>

Note: All API Parameters are Path Parameters.

Detailed Description of Endpoint API

1) Users

The screenshot shows the Postman interface with the 'User' collection selected. It contains three API endpoints:

- GET /api/User
- POST /api/User
- POST /api/User/{email}/{password}

1.1) Get Users:

<https://localhost:7136/api/User>

Whenever Users are found, postman will get the results and test will be passed with status code 200

The screenshot shows the Postman interface executing the 'GET User Details' request. The response status is 200 OK, and the response body is a JSON array of three user records:

```
1  [
2    {
3  "id": 1,
4  "name": "Vasu Mistry",
5  "email": "vmistry089@uottawa.ca",
6  "password": "123456",
7  "isMerchant": false
8  },
9  {
10 "id": 2,
11 "name": "Rushi Patel",
12 "email": "rpatel159@uottawa.ca",
13 "password": "654321",
14 "isMerchant": false
15 },
16 {
17 "id": 3,
18 "name": "Test user",
19 "email": "test@uottawa.ca",
20 "password": "123456",
21 }
22 ]
```

Whenever Users are not found, postman will not be able to get the results and Status code 404 (Not Found) will be generated.

1.2) Get Users by Email and Password:

For Testing this API, User need to enter email and password in order to fetch the particular User.

<https://localhost:7136/api/User/test%40uottawa.ca/123456>

If the user if found: Status Code 200 will be generated along with their details.

The screenshot shows the Postman interface with the following details:

- Collection:** TESTS-marketplace_services_CSIS112
- Request Type:** POST
- URL:** `(url)/User/test%40uottawa.ca/123456`
- Body (JSON):**

```
1
2
```
- Response Body (Pretty JSON):**

```
1
2
3
4
5
6
7
```

```
{
  "id": 3,
  "name": "Test user",
  "email": "test@uottawa.ca",
  "password": "123456",
  "isMerchant": false
}
```
- Status:** 200 OK

Status Code 404 (Not Found) is generated if any field is null.

The screenshot shows the Postman interface with the following details:

- Collection:** TESTS-marketplace_services_CSIS112
- Request Type:** POST
- URL:** `(url)/User/test%40uottawa.ca/`
- Query Params:**

KEY	VALUE	DESCRIPTION
Key	Value	Description
- Status:** 404 Not Found

If the email is in incorrect format status code 400-bad request is generated.

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CS15112'. A POST request is made to 'Users / Get User by email and password' with the URL parameter set to '({url})/User/test%40out/123456'. The response status is 400 Bad Request, indicating an incorrect email format. The response body contains the message '1 Incorrect email format'.

If the User Does not exist, status code 404 Not found is generated.

The screenshot shows the Postman interface with the same collection. A POST request is made to 'Users / Get User by email and password' with the URL parameter set to '({url})/User/abc@gr.com/123456'. The response status is 404 Not Found, indicating the user was not found. The response body contains the message '1 User not found with these credentials'.

1.3) Post/Add User Details:

For this API, we need to add the user details in the “Body”, which will get added Successfully with status code 200.

<https://localhost:7136/api/User>

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CSI5112'. A 'Users' folder contains three requests: 'GET Get User Details', 'POST Add User', and 'POST Get User by email and pass...'. The 'POST Add User' request is selected. The 'Body' tab shows a JSON payload:

```
1  {
2   ... "id": 16,
3   ... "name": "Dharmin",
4   ... "email": "dhrz@dr.com",
5   ... "password": "gvbjhjhj",
6   ... "isMerchant": false
7 }
```

The response section shows a status of 200 OK, time 6 ms, and size 152 B. The response body is 'true'.

User Added (Verified through GET API)

The screenshot shows the Postman interface with the same collection and 'Users' folder. The 'GET Get User Details' request is selected. The 'Body' tab indicates 'This request does not have a body'. The response section shows a status of 200 OK, time 43 ms, and size 631 B. The response body is a JSON object identical to the one in the previous screenshot:

```
29  {
30   ...
31   "id": 16,
32   "name": "Dharmin",
33   "email": "dhrz@jdbfd.com",
34   "password": "gvbjhjhj",
35   "isMerchant": false
36 }
37 }
```

If any field is null, then Status Code 400 (Bad Request) is generated:

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CSIS112'. A POST request is made to the endpoint '/Users/AddUser'. The request body contains the following JSON:

```
1 "id": null
2 "name": "Dhamin",
3 "email": "dhr@dhrt.com",
4 "password": "gvbjhjhj",
5 "isMerchant": false
```

The response status is 400 Bad Request, with the following error message in the body:

```
1 {"type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
2 "title": "One or more validation errors occurred.",
3 "status": 400,
4 "traceId": "00-4dcc65c9fdf1f742f2702f0997c8d20c5-5d804514eb94258b-00",
5 "errors": {
6     "newUser": [
7         "The newUser field is required."
8     ],
9     "$.id": [
10         "The id field is required."
11     ]
12 }
```

If email is in wrong format, Status Code 404 (Not Found) is generated:

The screenshot shows the Postman interface with the same collection and endpoint. The request body is identical to the previous one, except for the email address:

```
1 "id": 16,
2 "name": "Dhamin",
3 "email": "dhr",
4 "password": "gvbjhjhj",
5 "isMerchant": false
```

The response status is 404 Not Found, with the following message in the body:

```
1 Incorrect email format
```

If User Exist with same email or password, then Status Code 404 (Not Found) is generated:

The screenshot shows the Postman interface with the following details:

- Collection:** TESTS-marketplace_services_CSI5112 / Users
- Request Type:** POST
- URL:** {{url}}/User
- Body (JSON):**

```

1 "id": 16,
2 "name": "Dharmik",
3 "email": "dhz@dbfd.com",
4 "password": "gbjhjn",
5 "isMerchant": false
    
```

- Test Results (1/1):** User may already exists with same email or id
- Status:** 404 Not Found

2) Product

The screenshot shows the Postman interface with the following details:

- Collection:** Product
- Operations:**
 - GET /api/Product
 - POST /api/Product
 - PUT /api/Product
 - GET /api/Product/{id}
 - GET /api/Product/search/{productName}
 - GET /api/Product/search-cat/{categoryName}
 - DELETE /api/Product/{Id}

2.1) Get Product

<https://localhost:7136/api/Product>

If the products are found, status code 200 will be generated, and products will be fetched.

The screenshot shows the Postman interface with the following details:

- Collection:** TESTS-marketplace_services_CSI5112 / Product
- Request Type:** GET
- URL:** {{url}}/Product
- Body (JSON):**

```

1 pn.expect(response.category).not.eq(null);
2 pn.expect(response.price).not.eq(null);
3 pn.expect(response.quantity).not.eq(null);
        
```

- Test Scripts (JavaScript):**

```

Test scripts are written in JavaScript, and are run after the response is received.
Learn more about test scripts
        
```

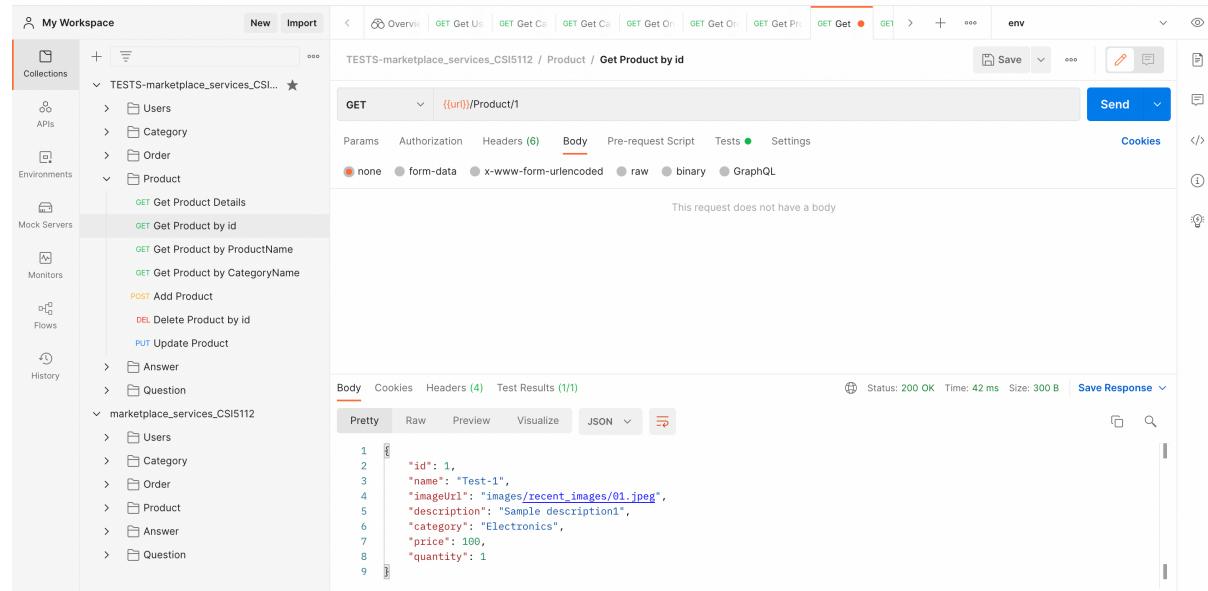
- Test Results (1/1):** Status: 200 OK Time: 99 ms Size: 1.04 KB

If no products are found, Status Code 404 (Not Found) will be generated.

2.2) Get Product by Product id:

<https://localhost:7136/api/Product/1>

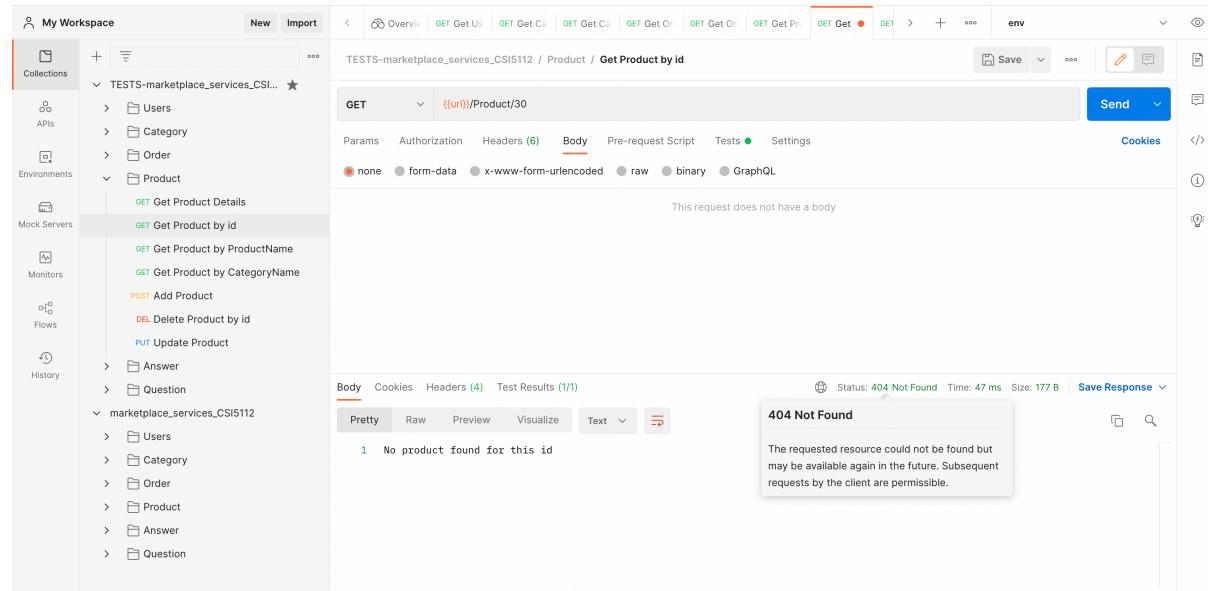
If the product id entered is correct, that particular product will be fetched and status code is 200.



The screenshot shows the Postman interface with a successful API call. The left sidebar lists collections, APIs, environments, mock servers, monitors, flows, and history. The main area shows a collection named 'TESTS-marketplace_services_CSIS112' with a 'Product' folder containing several requests. A specific GET request for '/Product/1' is selected, showing its details: method (GET), URL template ({url}/Product/1), headers (Content-Type: application/json), body (empty), and settings (Cookies). The response tab shows a status of 200 OK, time 42 ms, size 300 B, and a JSON response body:

```
1  "id": 1,
2  "name": "Test-1",
3  "imageUrl": "images/recent_images/01.jpeg",
4  "description": "Sample description1",
5  "category": "Electronics",
6  "price": 100,
7  "quantity": 1
```

If the product id is wrong Status code 404(Not found) is generated.



The screenshot shows the Postman interface with a failed API call. The left sidebar lists collections, APIs, environments, mock servers, monitors, flows, and history. The main area shows the same collection and request structure as the previous screenshot, but with a different URL template: {url}/Product/30. The response tab shows a status of 404 Not Found, time 47 ms, size 177 B, and an error message: 'No product found for this id'. The response body is empty.

If the Product Id is null, Status Code 400(Bad Request) is generated

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CS15112'. A test step titled 'Get Product by id' is selected. The request method is 'GET' with the URL path being '({url})/Product/null'. The 'Body' tab is active, showing the message 'This request does not have a body'. The response status is 400 Bad Request, with the error message: "type": "https://tools.ietf.org/html/rfc2311#section-6.5.1", "title": "One or more validation errors occurred.", "status": 400, "traceId": "00-e6f19009efdd1ebac42eac3abc79bfae-c86689a599cf527-00", "errors": { "id": ["The value 'null' is not valid."] }.

2.3) Get Product by Product Name:

<https://localhost:7136/api/Product/search/Test-1>

If the Product name is correct, it will generate list of products with status code 200.

The screenshot shows the Postman interface with the same collection. A test step titled 'Get Product by ProductName' is selected. The request method is 'GET' with the URL path being '({url})/Product/search/Test-1'. The 'Tests' tab is active, containing a snippet of JavaScript: pm.test("Status Code 404:No Product with given name exist", function () { pm.expect(pm.response.code).to.eql(404); }). The response status is 200 OK, with the JSON data: { "id": 1, "name": "Test-1", "imageUrl": "images/recent_images/01.jpeg", "description": "Sample description1", "category": "Electronics", "price": 100, "quantity": 1 }.

If the product is not found, Status Code 404(Not Found) will be generated).

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CS15112'. A GET request is made to 'Product / Get Product by ProductName' with the URL {{url}}/Product/search/Test-123. The response status is 404 Not Found, and the body contains the message 'No products found for this product name'.

```
pm.test("Status Code 404:No Product with given name exist", function () {
  pm.expect(pm.response.code).to.eq(404);
});
```

If the Product Name is null, Status Code 400(Bad Request) will be generated

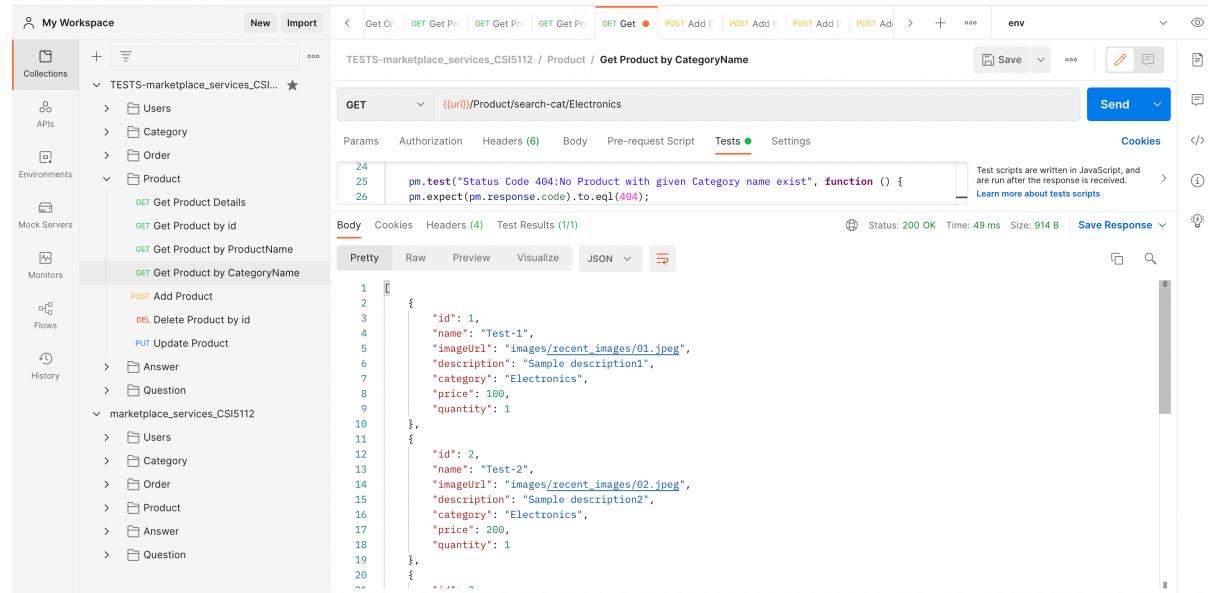
The screenshot shows the Postman interface with the same collection. A GET request is made to 'Product / Get Product by ProductName' with the URL {{url}}/Product/search/. The response status is 400 Bad Request, and the body is a JSON object indicating validation errors.

```
"type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
"title": "One or more validation errors occurred.",
"status": 400,
"traceId": "00-bb72746dda7d927a9869f4dcfb47c71-900e716e476b128b-00",
"errors": [
  {
    "id": [
      "The value 'search' is not valid."
    ]
  }
]
```

2.4) Get Product by Category Name:

<https://localhost:7136/api/Product/search-cat/Electronics>

If the Category Name is correct, it will return list of products with corresponding categories with status code 200.

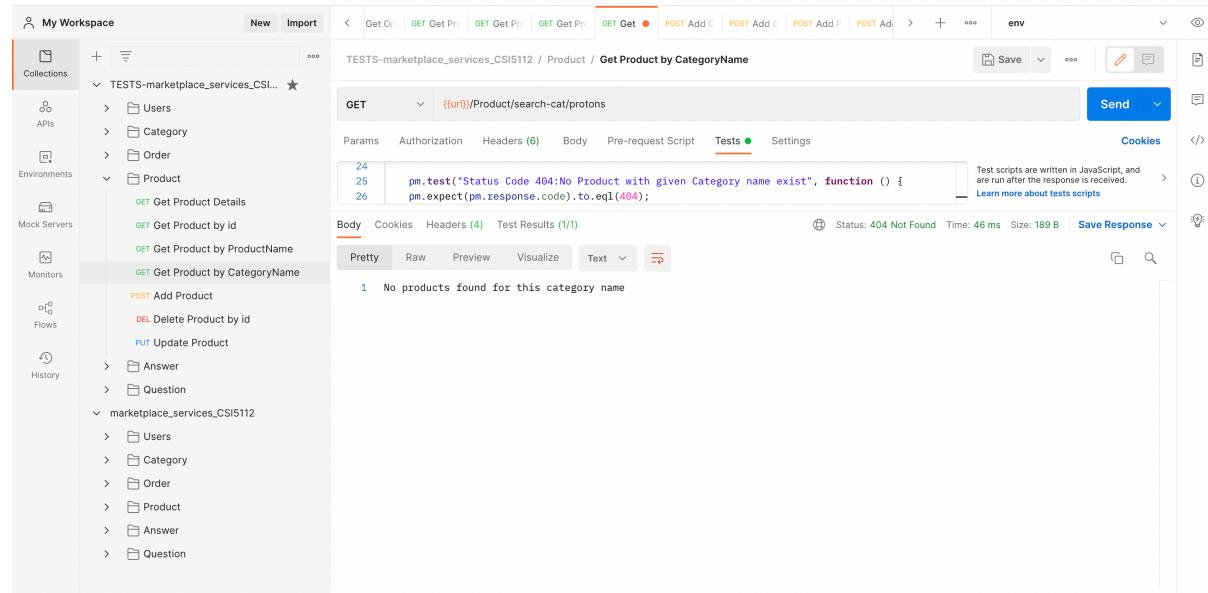


The screenshot shows a Postman collection named 'TESTS-marketplace_services_CSIS112'. Under the 'Product' section, there is a 'GET Get Product by CategoryName' request. The URL is `({url})/Product/search-cat/Electronics`. The response body is a JSON array containing two products:

```
1  [
2   {
3     "id": 1,
4     "name": "test-1",
5     "imageUrl": "images/recent_images/01.jpeg",
6     "description": "Sample description1",
7     "category": "Electronics",
8     "price": 100,
9     "quantity": 1
10    },
11   {
12     "id": 2,
13     "name": "test-2",
14     "imageUrl": "images/recent_images/02.jpeg",
15     "description": "Sample description2",
16     "category": "Electronics",
17     "price": 200,
18     "quantity": 1
19   }
20 ]
```

The status bar at the bottom indicates: Status: 200 OK Time: 49 ms Size: 914 B Save Response.

If the Category name is wrong, It will generate Status Code 404(Not Found).



The screenshot shows the same Postman collection and request setup as the previous one, but with a different URL: `({url})/Product/search-cat/protons`. The response body is a single line of text: 'No products found for this category name'. The status bar at the bottom indicates: Status: 404 Not Found Time: 46 ms Size: 189 B Save Response.

If the Category Name is null, Status Code 400 (Bad Request) will be generated.

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CSI5112'. A test script in the 'Tests' tab of a GET request to '/Product/search-cat/' contains a condition to check for a 400 status code. The response body shows a validation error message indicating that the category name is required.

```
pm.test("Status Code 404:No Product with given Category name exist", function () {
    pm.expect(pm.response.code).to.eql(404);
})
```

```
1
2
3
4
5
6
7
8
9
10
11
```

Response Body:

```
{
  "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
  "title": "One or more validation errors occurred.",
  "status": 400,
  "traceId": "00-8134843b8df6a3062c7a0a436bcab629-85d5bf866ac3dbac-00",
  "errors": [
    {
      "id": [
        "The value 'search-cat' is not valid."
      ]
    }
  ]
}
```

2.5) Add product

<https://localhost:7136/api/Product>

If the product body is passed without any duplication or null value, then the product is added with status code 200.

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CSI5112'. A POST request to '/Product' is made with a JSON body containing product details. The response status is 200 OK, and the response body is a simple boolean value 'true'.

```
POST https://localhost:7136/api/Product
```

```
Body (8)
```

```
{
  "id": 11,
  "name": "Test-11",
  "imageUrl": "images/recent_images/07.jpeg",
  "description": "Sample description8",
  "category": "Electronics",
  "price": 500,
  "quantity": 10
}
```

```
1
2
3
4
5
6
7
8
9
```

```
true
```

Product Added (Verified by GET product by Product Name API)

The screenshot shows a Postman collection named "TESTS-marketplace_services_CSI5112". The "Product" section contains a "GET Get Product by ProductName" endpoint. A test script in the "Tests" tab checks if the status code is 404 and the response matches. The response body is:

```

1
2
3
4
5
6
7
8
9
10
11
{
  "id": 11,
  "name": "Test-11",
  "imageUrl": "images/recent_images/07.jpeg",
  "description": "Sample description8",
  "category": "Electronics",
  "price": 500,
  "quantity": 10
}

```

If the Product body is null, Status Code 400 (Bad Request) will be generated.

The screenshot shows a Postman collection named "TESTS-marketplace_services_CSI5112". The "Product" section contains a "POST Add Product" endpoint. The body is set to "null". The response status is 400 Bad Request, with the following error message:

```

1
2
3
4
5
6
7
8
9
{
  "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
  "title": "One or more validation errors occurred.",
  "status": 400,
  "traceId": "00-c4192ed3295264db38312f321af2790c-cd573d60b28f2a4e-00",
  "errors": {
    "id": [
      "Value is required."
    ],
    "name": [
      "Value is required."
    ]
  }
}

```

If the Product body contains duplicate items, Status Code 400 (Bad request) will be generated.

The screenshot shows a Postman collection named "TESTS-marketplace_services_CSI5112". The "Product" section contains a "POST Add Product" endpoint. The body is identical to the previous successful one. The response status is 400 Bad Request, with the following error message:

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
1   Product with same id or name exists

```

2.6) Edit Product:

<https://localhost:7136/api/Product>

If the Body Parameter passed is validated, Product is edited, and Status Code 200 is generated.

The screenshot shows the Postman interface with the following details:

- Collection:** TESTS-marketplace_services_CSI5112
- Request Method:** PUT
- URL:** {{url}}/Product
- Body (JSON):**

```
1  {
2     "id": 5,
3     "name": "Test-111",
4     "imageUrl": "images/recent_images/01.jpeg",
5     "description": "Sample description1",
6     "category": "Electronics",
7     "price": 100,
8     "quantity": 1
9 }
```
- Response Status:** 200 OK
- Response Body:** true

Product edited (Verified using Get API)

The screenshot shows the Postman interface with the following details:

- Collection:** TESTS-marketplace_services_CSI5112
- Request Method:** GET
- URL:** {{url}}/Product
- Tests (JavaScript):**

```
15 pm.expect(response.category).not.eql(null),
14 pm.expect(response.price).not.edl(null);
15 pm.expect(response.quantity).not.eql(null);
```
- Response Status:** 200 OK
- Response Body:**

```
29   {
30     "id": 4,
31     "name": "Test-4",
32     "imageUrl": "images/recent_images/04.jpeg",
33     "description": "Sample description4",
34     "category": "Electronics",
35     "price": 400,
36     "quantity": 1
37   },
38   {
39     "id": 5,
40     "name": "Test-111",
41     "imageUrl": "images/recent_images/01.jpeg",
42     "description": "Sample description1",
43     "category": "Electronics",
44     "price": 100,
45     "quantity": 1
46   },
47   {
48     "id": 6,
```

If the Body Parameter is null, Status Code 400 (Bad Request) is generated.

The screenshot shows the Postman application interface. On the left sidebar, there are sections for Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. The main area displays a collection named "TESTS-marketplace_services_CSI5112". Under this collection, there are sub-folders "Users", "Category", "Order", and "Product". The "Product" folder is expanded, showing several GET and POST methods. A "PUT Update Product" method is selected. The "Body" tab is active, showing a JSON payload:

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
69
70
71
72
73
74
75
76
77
78
79
79
80
81
82
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
99
100
101
102
103
104
105
106
107
108
109
109
110
111
112
113
114
115
116
117
118
119
119
120
121
122
123
124
125
126
127
128
129
129
130
131
132
133
134
135
136
137
138
139
139
140
141
142
143
144
145
146
147
148
149
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
169
170
171
172
173
174
175
176
177
178
179
179
180
181
182
183
184
185
186
187
188
189
189
190
191
192
193
194
195
196
197
198
199
199
200
201
202
203
204
205
206
207
208
209
209
210
211
212
213
214
215
216
217
218
219
219
220
221
222
223
224
225
226
227
228
229
229
230
231
232
233
234
235
236
237
238
239
239
240
241
242
243
244
245
246
247
248
249
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
698
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2397
2397
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2429
2430
2431
```

If the Product is not found, Status Code 404 (Not Found) is generated.

The screenshot shows the Postman application interface. On the left, the sidebar displays collections, APIs, environments, mock servers, monitors, flows, and history. The 'Products' collection under 'TESTS-marketplace_services_CSI5112' is selected. The main workspace shows a 'PUT' request to '({uri})/Product'. The 'Body' tab is active, showing a JSON payload:

```
1  "id": 15,
2  "name": "Test-111",
3  "imageUrl": "images/recent_images/01.jpeg",
4  "description": "Sample description1",
5  "category": "Electronics",
6  "price": 100,
7  "quantity": 1
```

The response status is 404 Not Found, with a message: 'Product does not exist so cannot update'.

If Category is edited, it will also show edited category in product list.

Before:

The screenshot shows the Postman application interface. On the left sidebar, there are sections for Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. The main area displays a collection named "TESTS-marketplace_services_CSI5112". Under the "Category" section, a "GET Get Category Details" endpoint is selected. The "Body" tab of the request editor shows the following JSON response:

```
[{"id": 2, "imageURL": "images/category_images/02.png", "name": "Electronics"}, {"id": 3, "imageURL": "images/category_images/03.png", "name": "Food"}, {"id": 4, "imageURL": "category_images/04.png", "name": "Sports"}, {"id": 5, "imageURL": "images/category_images/05.png", "name": "Books"}, {"id": 6, "imageURL": "images/category_images/06.png", "name": "Clothing"}]
```

Editing Category:

The screenshot shows the Postman interface with the following details:

- Collection:** TESTS-marketplace_services_CSI5112
- Request Type:** PUT
- URL:** {{url}}/Category
- Body (JSON):**

```

1  ...
2  ...
3  ...
4  ...
5  ...
6  ...
7  ...
8  ...
9  ...
10 ...
11 ...
12 ...
13 ...
14 ...
15 ...
16 ...
17 ...
18 ...
19 ...
20 ...
21 ...
22 ...
  
```

```

1 "id": 3,
2 "imageURL": "images/category_images/03.png",
3 "name": "Food Healthy"
  
```
- Response Status:** 200 OK
- Test Results:** true

Edited category shows in Category list:

The screenshot shows the Postman interface with the following details:

- Collection:** TESTS-marketplace_services_CSI5112
- Request Type:** GET
- URL:** {{url}}/Category
- Body:** This request does not have a body
- Response Status:** 200 OK
- Test Results:** true
- Sample Response (JSON):**

```

8 ...
9 ...
10 ...
11 ...
12 ...
13 ...
14 ...
15 ...
16 ...
17 ...
18 ...
19 ...
20 ...
21 ...
22 ...
  
```

```

8 ...
9 ...
10 ...
11 ...
12 ...
13 ...
14 ...
15 ...
16 ...
17 ...
18 ...
19 ...
20 ...
21 ...
22 ...
  
```

```

8 ...
9 ...
10 ...
11 ...
12 ...
13 ...
14 ...
15 ...
16 ...
17 ...
18 ...
19 ...
20 ...
21 ...
22 ...
  
```

Edited Category shown in product list:

The screenshot shows the Postman interface with the following details:

- Collection:** TESTS-marketplace_services_CSI5112
- Request Type:** GET
- URL:** {{url}}/Product
- Body:** This request does not have a body
- Response Status:** 200 OK
- Test Results:** true
- Sample Response (JSON):**

```

32 ...
33 ...
34 ...
35 ...
36 ...
37 ...
38 ...
39 ...
40 ...
41 ...
42 ...
43 ...
44 ...
45 ...
46 ...
47 ...
48 ...
49 ...
50 ...
51 ...
  
```

```

32 ...
33 ...
34 ...
35 ...
36 ...
37 ...
38 ...
39 ...
40 ...
41 ...
42 ...
43 ...
44 ...
45 ...
46 ...
47 ...
48 ...
49 ...
50 ...
51 ...
  
```

If a Category is deleted, it will also delete that category from the product list:
Before:

The screenshot shows two API requests in Postman:

- TESTS-marketplace_services_CS15112 / Product / Get Product Details**: A GET request to `((url))/Product`. The response body contains a list of products, each with a category field. One product has a category named "Food Healthy".
- TESTS-marketplace_services_CS15112 / Category / Get Category Details**: A GET request to `((url))/Category`. The response body contains a list of categories.

Deleting category:

The screenshot shows a DELETE request in Postman:

- TESTS-marketplace_services_CS15112 / Category / Delete Category**: A DELETE request to `((url))/Category/3`.

The response body is `true`, indicating the category was successfully deleted.

Category Deleted from category list:

The screenshot shows the Postman interface with two API requests for the 'Category' collection.

Request 1: GET /Category

```
TESTS-marketplace_services_CS15112 / Category / Get Category Details
GET {{(url)}/Category
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Body
This request does not have a body
Body Cookies Headers (4) Test Results (1/1)
Pretty Raw Preview Visualize JSON
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
"imageURL": "images/category_images/01.png",
"name": "cloth"
},
{
"id": 2,
"imageURL": "images/category_images/02.png",
"name": "Electronics"
},
{
"id": 4,
"imageURL": "images/category_images/04.png",
"name": "Sports"
},
{
"id": 5,
```

Status: 200 OK Time: 4 ms Size: 641 B Save Response

Request 2: GET /Category/3

```
TESTS-marketplace_services_CS15112 / Category / Get Category by id
GET {{(url)}/Category/3
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Body
This request does not have a body
Body Cookies Headers (4) Test Results (1/1)
Pretty Raw Preview Visualize Text
1 Category does not exist with this id
404 Not Found
The requested resource could not be found but may be available again in the future. Subsequent requests by the client are permissible.
Status: 404 Not Found Time: 13 ms Size: 185 B Save Response
```

Category deleted from product list as well:

The screenshot shows the Postman interface with a single API request for the 'Product' collection.

Request: GET /Product

```
TESTS-marketplace_services_CS15112 / Product / Get Product Details
GET {{(url)}/Product
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Tests
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
pm.expect(response.category).not.eql(null);
pm.expect(response.price).not.eql(null);
pm.expect(response.quantity).not.eql(null);
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts

Status: 200 OK Time: 14 ms Size: 914 B Save Response

Response Body:

```
Pretty Raw Preview Visualize JSON
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
{
  "name": "Test-4",
  "imageURL": "images/recent_images/04.jpeg",
  "description": "Sample description4",
  "category": "Electronics",
  "price": 400,
  "quantity": 1
},
```

The screenshot shows the Postman interface with a collection named "TESTS-marketplace_services_CS15112". On the left sidebar, under the "Product" category, there is a "Product" item which contains a "GET Get Product by id" method. This method is selected in the main panel. The request URL is set to `((url))/Product/5`. The "Tests" tab is active, containing the following test script:

```
1 No product found for this id
```

The response status is 404 Not Found, and the response body is "No product found for this id".

2.7) Delete Product:

<https://localhost:7136/api/Product/5>

If the input id is true, product will be deleted, and Status Code 200 will be generated.

The screenshot shows the Postman interface with a collection named "TESTS-marketplace_services_CS15112". On the left sidebar, under the "Product" category, there is a "Product" item which contains a "DELETE Delete Product by id" method. This method is selected in the main panel. The request URL is set to `((url))/Product/5`. The "Tests" tab is active, containing the following test script:

```
1 if(pm.response.code == 200)
2 {
3     pm.test("Status Code 200:Product Successfully Deleted", function () {
4         pm.expect(pm.response.code).to.eql(200);
5         const response = pm.response.json();
6         pm.expect(response).to.eql(true);
7     });
8 }
9
10
11
12
13
14
```

The response status is 200 OK, and the response body is "true". A tooltip for the "Tests" tab indicates that test scripts are run after the response is received.

Product is deleted (Verified using Get product by id API)

The screenshot shows the Postman interface with a collection named "TESTS-marketplace_services_CSIS112". A GET request is made to `((url))/Product/5`. The response status is 404 Not Found, and the body contains the message "No product found for this id".

If the product does not exist, Status Code 400 (Bad request) is generated.

The screenshot shows the Postman interface with a collection named "TESTS-marketplace_services_CSIS112". A DELETE request is made to `((url))/Product/50`. The response status is 400 Bad Request, and the body contains the message "Cannot delete product because it does not exist".

If the Product id is null, Status Code 400 (Bad Request) is generated.

The screenshot shows the Postman interface with a collection named "TESTS-marketplace_services_CSIS112". A DELETE request is made to `((url))/Product/null`. The response status is 400 Bad Request, and the body is a JSON object containing validation errors:

```

1  {
2      "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
3      "title": "One or more validation errors occurred.",
4      "status": 400,
5      "traceId": "00-5076a647f0a74617b08e9cfdd99d67-c772b219c1209067-00",
6      "errors": [
7          {
8              "id": [
9                  "The value 'null' is not valid."
10         ]
11     }
12   ];
13 }
  
```

3) Category:

The screenshot shows the Postman interface with a collection named 'Category'. It lists six endpoints:

- GET /api/Category
- POST /api/Category
- PUT /api/Category
- GET /api/Category/{id}
- DELETE /api/Category/{Id}

The last endpoint, DELETE /api/Category/{Id}, is highlighted with a red border.

3.1) Get Category:

<https://localhost:7136/api/Category>

If the Category exist, Status Code 200 will be generated, and the list of categories will be fetched.

The screenshot shows a successful GET request to `((url))/Category`. The response status is 200 OK, and the response body is a JSON array of five category objects:

```
[{"id": 1, "imageURL": "images/category_images/01.png", "name": "Cloth"}, {"id": 2, "imageURL": "images/category_images/02.png", "name": "Electronics"}, {"id": 3, "imageURL": "images/category_images/03.png", "name": "Food"}, {"id": 4, "imageURL": "images/category_images/04.png", "name": "Sports"}, {"id": 5, "imageURL": "images/category_images/05.png", "name": "Toys"}]
```

If the category is not found, Status Code 404 (Not Found) error will be generated.

3.2) Add Category:

<https://localhost:7136/api/Category>

If the Body is validated, the Category will be successfully added, and Status Code 200 will be generated.

The screenshot shows the Postman interface with a successful API call. The left sidebar lists collections, APIs, environments, mock servers, monitors, flows, and history. The main area shows a POST request to `(url)/Category`. The body contains JSON data: `{ "id": 9, "imageURL": "images/category_images/09.png", "name": "Category-3"}`. The response status is 200 OK, time is 70 ms, size is 152 B, and the body is `true`.

Category Added (Verified through GET API)

The screenshot shows the Postman interface with a successful API call. The left sidebar lists collections, APIs, environments, mock servers, monitors, flows, and history. The main area shows a GET request to `(url)/Category`. The body contains JSON data representing multiple categories, including `{ "id": 9, "imageURL": "images/category_images/09.png", "name": "Category-3"}`. The response status is 200 OK, time is 46 ms, size is 710 B, and the body is a detailed JSON array of categories.

If the Category is null, Status Code 400(Bad Request) will be generated.

The screenshot shows the Postman interface with a collection named "TESTS-marketplace_services_CSI5112". The "Category" section contains a "POST Add Category" request. The request body is JSON:

```
1 id
2 ...
3 ...
4 "name": "Category-3"
```

The response status is 400 Bad Request, with the error message:

```
1 type: "https://tools.ietf.org/html/rfc7231#section-6.5.1"
2 title: "One or more validation errors occurred."
3 status: 400
4 traceId: "00-aa4b8f9b0518261d899fd9e2b8fd6bf8-d7e8c4501852086e-00"
5 errors: {
6   "category": [
7     {
8       "error": "The category field is required."
9     }
10   ],
11   "$.id": [
12     "The 'id' value could not be converted to marketplace.services.CS15112. Path: $ .id | LineNumber: 1"
13   ]
14 }
```

If the Category id already exist, Status Code 400 (Bad Request) will be generated.

The screenshot shows the Postman interface with the same collection and request setup as the previous screenshot, but with a different body:

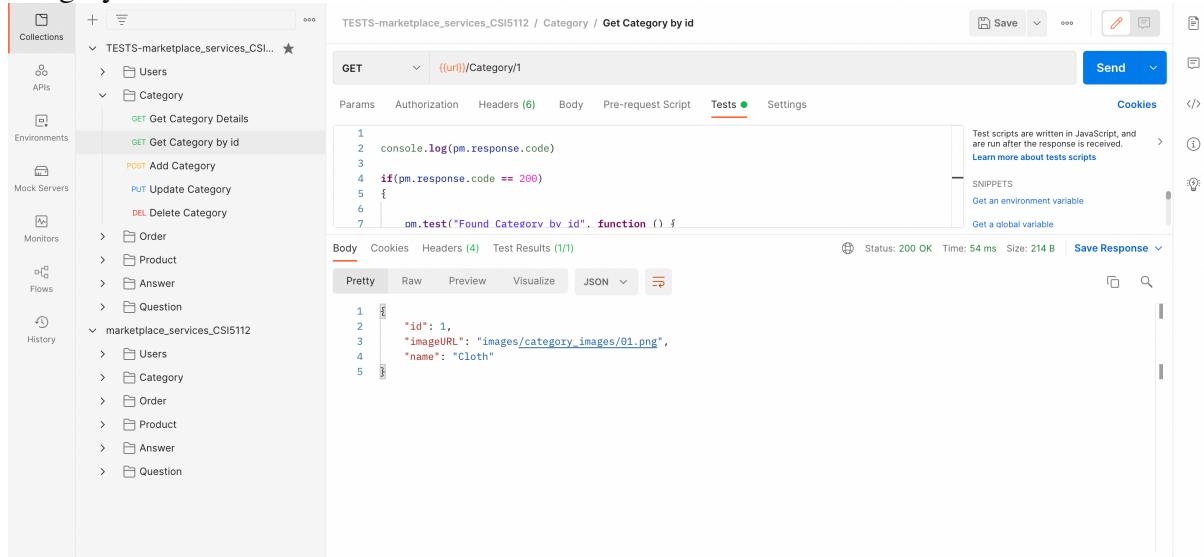
```
1 id: 1
2 ...
3 ...
4 "name": "Category-3"
```

The response status is 400 Bad Request, with the error message:

```
1 Category with same id or name already exists
```

3.3) Get Category by id:

If the Category id exist, Status Code 200 will be generated along with details of the requested category id

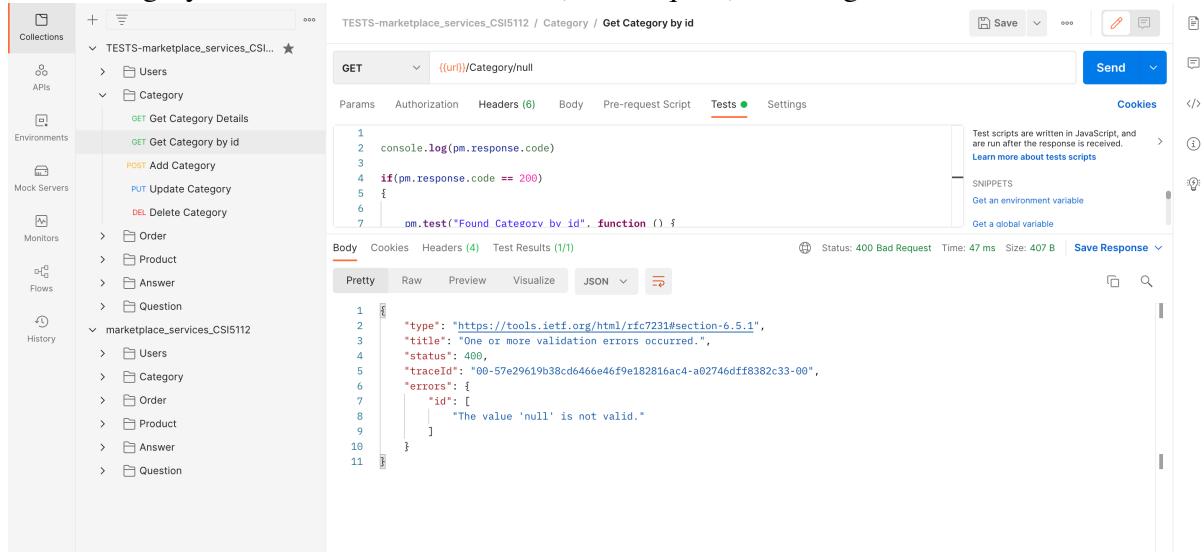


The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CSIS112'. A test script is present in the 'Tests' tab of the 'Get Category by id' request, which logs the response code and checks if it's 200. The response body is a JSON object with fields: id, imageURL, and name. The status bar indicates a 200 OK response.

```
1 console.log(pm.response.code)
2
3 if(pm.response.code == 200)
4 {
5   pm.test("Found Category by id", function () {
6     pm.expect(pm.response.json().id).to.be.a("number");
7     pm.expect(pm.response.json().imageURL).to.be.a("string");
8     pm.expect(pm.response.json().name).to.be.a("string");
9   })
10 })
11 }
```

Status: 200 OK Time: 54 ms Size: 214 B Save Response

If the Category id is null, Status Code 400 (Bad Request) will be generated.



The screenshot shows the Postman interface with the same collection. The test script in the 'Tests' tab of the 'Get Category by id' request checks if the response code is 200. The response body is a JSON object indicating validation errors, with one error message: 'The value 'null' is not valid.' The status bar indicates a 400 Bad Request response.

```
1 console.log(pm.response.code)
2
3 if(pm.response.code == 200)
4 {
5   pm.test("Found Category by id", function () {
6     pm.expect(pm.response.json().errors[0].id).to.be.a("string");
7     pm.expect(pm.response.json().errors[0].message).to.be.a("string");
8   })
9 })
10 }
```

Status: 400 Bad Request Time: 47 ms Size: 407 B Save Response

If the category id is not found, Status code 404 (Not Found) will be generated

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CSI5112'. A test script is present in the 'Tests' tab of a 'GET' request for '/Category/{id}'. The script logs the response code and checks if it's 200. If not, it asserts that the category does not exist. The response status is 404 Not Found.

```
1 console.log(pm.response.code)
2
3 if(pm.response.code == 200)
4 {
5
6   pm.test("Found Category by id", function () {
7     pm.expect(pm.response.code).to.equal(200)
8     pm.expect(pm.response.json().name).to.equal("Clothing")
9   })
10 })
11
12 pm.expect(pm.response.error).to.be.undefined
```

Status: 404 Not Found Time: 51 ms Size: 185 B Save Response

3.4) Edit Category:

<https://localhost:7136/api/Category>

If the body is validated, the Status Code 200 will be generated, and category will be edited.

Before:

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CSI5112'. A test script is present in the 'Tests' tab of a 'GET' request for '/Category/1'. The script logs the response code and checks if it's 200. If not, it asserts that the category does not exist. The response status is 200 OK.

```
1 console.log(pm.response.code)
2
3 if(pm.response.code == 200)
4 {
5
6   pm.test("Found Category by id", function () {
7     pm.expect(pm.response.code).to.equal(200)
8     pm.expect(pm.response.json().name).to.equal("Clothing")
9   })
10 })
11
12 pm.expect(pm.response.error).to.be.undefined
```

Status: 200 OK Time: 583 ms Size: 214 B Save Response

After Editing:

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** Shows collections, environments, mock servers, monitors, flows, and history.
- Request URL:** TESTS-marketplace_services_CSI5112 / Category / Update Category
- Method:** PUT
- Body:** JSON (raw) content:

```
1  ...
2  ...
3  ...
4  ...
5  ...
```

Content of the raw JSON body:

```
1 "id": 1,
2 ...
3 ...
4 ...
5 ...
```
- Response Status:** 200 OK
- Response Body:** true

Category Edited (Verified by Get Category by id API)

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** Shows collections, environments, mock servers, monitors, flows, and history.
- Request URL:** TESTS-marketplace_services_CSI5112 / Category / Get Category by id
- Method:** GET
- Body:** This request does not have a body
- Response Status:** 200 OK
- Response Body:** JSON (raw) content:

```
1 ...
2 ...
3 ...
4 ...
5 ...
```

Content of the raw JSON body:

```
1 ...
2 ...
3 ...
4 ...
5 ...
```

If the body parameter is null, Status Code 400 (Bad Request) will be generated.

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CSI5112'. The 'Category' folder contains a 'PUT Update Category' request. The request URL is `({url})/Category`. The 'Body' tab is selected, showing the following JSON:

```
1  ...
2  ...
3  ...
4  ...
5  ...
```

The response status is 400 Bad Request, with the following error message in the body:

```
1   "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
2   "title": "One or more validation errors occurred.",
3   "status": 400
4   "traceId": "00-4fe7e9aea27e2b269016e39e23d6fa57-f5ea93dc2734d51f-00",
5   "errors": {
6     "editedCategory": [
7       "The editedCategory field is required."
8     ],
9     "$.id": [
10       "The JSON value could not be converted to marketplace.services.CS15112.Category. Path: $ .id | LineNumber: 1"
11     ]
12   }
```

If no existing category is found to be edited, Status Code 400 (Bad Request) will be generated.

The screenshot shows the Postman interface with the same collection and request setup as the previous screenshot. The 'Body' tab is selected, showing the following JSON:

```
1  ...
2  ...
3  ...
4  ...
5  ...
```

The response status is 400 Bad Request, with the following error message in the body:

```
1   No existing category found with the passed category id value
```

3.5) Delete Category by id:

<https://localhost:7136/api/Category/1>

If the id is validated, the Status Code 200 will be generated, and category will be deleted.
Before:

The screenshot shows the Postman interface with the 'TESTS-marketplace_services_CSIS112' collection selected. Under the 'Category' folder, there is a 'GET Get Category Details' endpoint. A test result for this endpoint is shown, indicating a 200 OK status code with a response time of 146 ms and a response size of 707 B. The response body is a JSON array of five categories, each with an id, name, and imageUrl.

```
1 [ { "id": 1, "name": "Cloth", "imageUrl": "images/category_images/01.png"}, { "id": 2, "name": "Electronics", "imageUrl": "images/category_images/02.png"}, { "id": 3, "name": "Food", "imageUrl": "images/category_images/03.png"}, { "id": 4, "name": "Sports", "imageUrl": "images/category_images/04.png"}, { "id": 5, "name": "Books", "imageUrl": "images/category_images/05.png"} ]
```

After Deleting:

The screenshot shows the Postman interface with the 'TESTS-marketplace_services_CSIS112' collection selected. Under the 'Category' folder, there is a 'DEL Delete Category' endpoint. A test script is attached to this endpoint, checking if the response code is 200 and the response body is true. A test result for this endpoint is shown, indicating a 200 OK status code with a response time of 57 ms and a response size of 152 B. The response body is true.

```
1 if(pm.response.code == 200) { 2   pm.test("Status Code 200:Category Successfully Deleted", function () { 3     pm.expect(pm.response.code).to.eql(200); 4     const response = pm.response.json(); 5     pm.expect(response).to.eql(true); 6   }); 7 } 8 9 pm.response.json(); 10 11 12 };
```

```
1 true
```

Category Deleted (verified using Get Category by id API)

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CSI5112'. A 'Category' folder contains a 'GET Get Category by id' request. The request URL is `(url)/Category/1`. The response status is 404 Not Found, and the body contains the message 'Category does not exist with this id'.

If the id parameter is null, Status Code 400 (Bad Request) will be generated.

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CSI5112'. A 'Category' folder contains a 'DEL Delete Category' request. The request URL is `(url)/Category/null`. The response status is 400 Bad Request, and the body contains a JSON object with an error message: "The value 'null' is not valid."

If no existing category id is found to be deleted, Status Code 404 (Not Found) will be generated.

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CSI5112'. A 'Category' folder contains a 'DEL Delete Category' request. The request URL is `(url)/Category/100`. The response status is 404 Not Found, and the body contains the message 'No category found with passed Id param'.

4) Order:

Order	
GET	/api/Order
POST	/api/Order
GET	/api/Order/byUser/{UserId}
GET	/api/Order/byOrder/{OrderId}

4.1) Get All order:

<https://localhost:7136/api/Order>

If the order exists, then status code 200 will be generated, and list of orders will be displayed.

The screenshot shows the Postman application interface. On the left sidebar, there are sections for Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. The main area displays a collection named 'TESTS-marketplace_services_CSI5112'. A specific API endpoint, 'GET Order Details', is selected under the 'Order' category. The 'Body' tab is active, showing a JSON response structure. The response body contains an array of three items, each representing a product with fields like id, name, imageUrl, description, category, price, and quantity. The first item is for an iPhone 123, the second for an iPhone 3, and the third for an iPhone 10.

```
1 *111-222*: [
2   {
3     "id": 1,
4     "name": "iPhone 123",
5     "imageUrl": "images/product_images/iphone.jpg",
6     "description": "Sample description1",
7     "category": "Electronics",
8     "price": 100,
9     "quantity": 1
10   },
11   {
12     "id": 2,
13     "name": "iPhone 3",
14     "imageUrl": "images/product_images/iphone.jpg",
15     "description": "Sample description2",
16     "category": "Electronics",
17     "price": 200,
18     "quantity": 1
19   },
20   {
21     "id": 3,
22     "name": "iPhone 10",
23     "imageUrl": "images/product_images/iphone.jpg",
24 }
```

If the Order does not exist, Status Code 404 (Not Found) will be generated.

4.2) Get Order by User id:

https://localhost:7136/api/Order/byUser/111

If the User id exist corresponding the order, the particular order will be fetched.

The screenshot shows the Postman application interface. On the left sidebar, there are sections for Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. The main area displays a collection named "TESTS-marketplace_services_CS15112". Under the "Order" section, a GET request is selected with the URL {{url}}/Order/byUser/111. The response status is 200 OK, time is 58 ms, size is 1.11 KB, and the response body is as follows:

```
1 *111-222*: [
2   {
3     "id": 1,
4     "name": "iPhone 123",
5     "imageUrl": "images/product_images/iphone.jpg",
6     "description": "Sample description1",
7     "category": "Electronics",
8     "price": 100,
9     "quantity": 1
10 },
11   {
12     "id": 2,
13     "name": "iPhone 3",
14     "imageUrl": "images/product_images/iphone.jpg",
15     "description": "Sample description2",
16     "category": "Electronics",
17     "price": 200,
18     "quantity": 1
19 },
20   {
21     "id": 3,
22     "name": "iPhone 10",
23     "imageUrl": "images/product_images/iphone.jpg",
24 }
```

If the Order does not exist by particular user id, Status Code 404 (Not Found) will be generated.

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CSIS112'. A GET request is made to the endpoint `((url))/Order/byUser/11`. The response status is 404 Not Found, and the body of the response is "1 No Products found for this user".

If the User id is null, Status Code 404 (Not Found) will be generated.

The screenshot shows the Postman interface with the same collection. A GET request is made to the endpoint `((url))/Order/byUser/null`. The response status is 404 Not Found, and the body of the response is "1 No Products found for this user".

4.3) Get Order by Order id:

<https://localhost:7136/api/Order/byOrder/222>

If the Order id exist corresponding the order, the order will be fetched.

The screenshot shows the Postman interface with a successful API call. The URL is `https://localhost:7136/api/Order/byOrder/222`. The response status is 200 OK, time 61ms, size 629B. The response body is a JSON array of three items, each representing an order for an iPhone:

```
[{"id": 1, "name": "iPhone 123", "imageUrl": "images/product_images/iphone.jpg", "description": "Sample description1", "category": "Electronics", "price": 100, "quantity": 1}, {"id": 2, "name": "iPhone 3", "imageUrl": "images/product_images/iphone.jpg", "description": "Sample description2", "category": "Electronics", "price": 200, "quantity": 1}, {"id": 3, "name": "iPhone 10", "imageUrl": "images/product_images/iphone.jpg", "description": "Sample description3", "category": "Electronics", "price": 300, "quantity": 1}]
```

If the Order does not exist by Order id, Status Code 404 (Not Found) will be generated.

The screenshot shows the Postman interface with a failed API call. The URL is `https://localhost:7136/api/Order/byOrder/2`. The response status is 404 Not Found, time 55ms, size 181B. The response body is a single line: "No orders found for this orderid".

If the Order id is null, Status Code 404 (Not Found) will be generated.

The screenshot shows the Postman interface with a collection named "TESTS-marketplace_services_CSI5112". A GET request is made to the endpoint `({url})/Order/byOrder/null`. The response status is 404 Not Found, and the body contains the message "No orders found for this orderid".

4.4) Add Order:

<https://localhost:7136/api/Order?id=111-22>

If the Format of parameters (a-b) are true and duplicate order does not exist, then order will be added, and status code 200 will be generated.

The screenshot shows the Postman interface with a POST request to the endpoint `({url})/Order?id=111-22`. The request body is a JSON object representing an order. The response status is 200 OK, and the body contains the value "true".

```
1 {  
2   "id": 1,  
3   "name": "iPhone 123",  
4   "imageurl": "images/product_images/iphone.jpg",  
5   "description": "Sample description1",  
6   "category": "Electronics",  
7   "price": 100,  
8   "quantity": 1  
9 },  
10 }  
11 
```

Product Added (Verified through Get Api)

```

    [
      {
        "id": 1,
        "name": "iPhone 123",
        "imageurl": "images/product_images/iphone.jpg",
        "description": "Sample description1",
        "category": "Electronics",
        "price": 100,
        "quantity": 1
      },
      {
        "id": 2,
        "name": "iPhone 3",
        "imageurl": "images/product_images/iphone.jpg",
        "description": "Sample description2",
        "category": "Electronics",
        "price": 200,
        "quantity": 1
      },
      {
        "id": 3,
        "name": "iPhone 10"
      }
    ]
  
```

If the parameters are null, Status Code 400 (Bad Request) will be generated.

```

    [
      {
        "id": null,
        "name": "iPhone 123",
        "imageurl": "images/product_images/iphone.jpg",
        "description": "Sample description1",
        "category": "Electronics",
        "price": 100,
        "quantity": 1
      }
    ]
  
```

```

  {
    "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
    "title": "One or more validation errors occurred.",
    "status": 400,
    "traceId": "00-2b778287192adff41211a70de18f4ac-24c7b678c9a45092-00",
    "errors": {
      "products": [
        {
          "id": null
        }
      ],
      "$[0].id": [
        "The 'id' value could not be converted to marketplace.coremodel.PTEntityModel. Product 'id' is mandatory."
      ]
    }
  }
  
```

If the Parameters are not formatted well or they do not exist, Status Code 400 will be generated.

```

    [
      {}
    ]
  
```

```

  {
    "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
    "title": "One or more validation errors occurred.",
    "status": 400,
    "traceId": "00-2c9a0019af6d6122f56c75c69ca3ca34-8985549a5d7bc564-00",
    "errors": {
      "products": [
        {
          "id": null
        }
      ],
      "$[0]": [
        "The 'id' object contains a trailing comma at the end which is not supported in this node. Change the reader."
      ]
    }
  }
  
```

5) Question:

Question

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CSI5112'. Under the 'Question' folder, there are four API endpoints listed:

- GET /api/Question
- POST /api/Question
- GET /api/Question/{id}
- DELETE /api/Question/{Id}

5.1) Get All Question:

<https://localhost:7136/api/Question>

If the Question are listed, they will be fetched, and Status code 200 will be generated.

The screenshot shows a successful GET request to `((url))/Question`. The response body is a JSON array containing four question objects:

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
{
  "id": 1,
  "title": "First Question",
  "description": "A suit of armor provides excellent sun protection on hot days.",
  "userName": "Esra Ersan"
},
{
  "id": 2,
  "title": "Second Question",
  "description": "A suit of armor provides excellent sun protection on hot days.",
  "userName": "Bhavesh Bisth"
},
{
  "id": 3,
  "title": "Third Question",
  "description": "A suit of armor provides excellent sun protection on hot days.",
  "userName": "Vasu Mistry"
},
{
  "id": 4,
  "title": "Fourth Question",
  "description": "A suit of armor provides excellent sun protection on hot days.",
  "userName": "Rushi Patel"
}
```

If the Questions are not found, Status code 404 (Not Found) will be generated.

5.2) Get Question by id:

<https://localhost:7136/api/Question/1>

If question exist by particular id, it will be fetched.

The screenshot shows a successful GET request to `((url))/Question/1`. The response body is a single question object:

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
{
  "id": 1,
  "title": "First Question",
  "description": "A suit of armor provides excellent sun protection on hot days.",
  "userName": "Esra Ersan"
}
```

If the Question id is wrong, Status Code 404(Not Found) will be generated.

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CSIS112'. A test script is present in the 'Tests' tab of the 'Get Question by id' request. The response status is 404 Not Found, and the body contains the message 'No Question found for this id'.

```
1 console.log(pm.response.code)
2 if(pm.response.code == 200)
3 {
4     pm.test("Found Question by id", function () {
5         pm.expect(pm.response.code).to.eql(200);
6         const response = pm.response.json();
7         console.log(response.name);
8         pm.expect(response.name).not.eql(null);
9         pm.expect(response.imageUrl).not.eql(null);
10        pm.expect(response.description).not.eql(null);
11    })
12 }
13
```

Status: 404 Not Found Time: 69 ms Size: 178 B Save Response

If the question id is null, Status Code 400 (Bad Request) will be generated.

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CSIS112'. A test script is present in the 'Tests' tab of the 'Get Question by id' request. The response status is 400 Bad Request, and the body contains validation errors indicating that the 'id' field cannot be null.

```
1 console.log(pm.response.code)
2 if(pm.response.code == 200)
3 {
4     pm.test("Found Question by id", function () {
5         pm.expect(pm.response.code).to.eql(200);
6         const response = pm.response.json();
7         console.log(response.name);
8         pm.expect(response.name).not.eql(null);
9         pm.expect(response.imageUrl).not.eql(null);
10        pm.expect(response.description).not.eql(null);
11    })
12 }
13
```

Status: 400 Bad Request Time: 44 ms Size: 407 B Save Response

5.3) Post/Add Question:

<https://localhost:7136/api/Question>

If the body parameters are correctly passed, the status code 200 will be generated and question will be added.

The screenshot shows the Postman interface with the following details:

- Collection:** TESTS-marketplace_services_CSI5112 / Question / Add Question
- Method:** POST
- URL:** {{url}}/Question
- Body:** (JSON)
{"id": 13, "title": "tenth Question", "description": "A suit of armor provides excellent sun protection on hot days.", "userName": "Esra Ersan"}
- Test Results:** Status: 200 OK, Time: 57 ms, Size: 152 B
- Response Body:** true

Question Added (Verified by Get by id API)

The screenshot shows the Postman interface with the following details:

- Collection:** TESTS-marketplace_services_CSI5112 / Question / Get Question by id
- Method:** GET
- URL:** {{url}}/Question/13
- Tests:** (JavaScript)
```  
if(pm.response.code == 200){  
 pm.test("Found Question by id", function () {  
 pm.expect(pm.response.code).to.eql(200);  
 const response = pm.response.json();  
 console.log(response.name);  
 pm.expect(response.name).not.eql(null);  
 pm.expect(response.imageUrl).not.eql(null);  
 pm.expect(response.description).not.eql(null);  
 });  
}  
```
- Test Results:** Status: 200 OK, Time: 49 ms, Size: 285 B
- Response Body:** {"id": 13, "title": "tenth Question", "description": "A suit of armor provides excellent sun protection on hot days.", "userName": "Esra Ersan"}

If the body parameters are null, Status Code 400 (Bad Request) will be generated.

The screenshot shows the Postman application interface. The left sidebar contains collections, APIs, environments, mock servers, monitors, flows, and history. The main area shows a collection named "TESTS-marketplace_services_CS15112". A sub-collection "Question" is selected, displaying four methods: GET Question, POST Add Question, GET Get Question by id, and DEL Delete Question by id. The "POST Add Question" method is currently selected, showing its details. The "Body" tab is active, containing the following JSON payload:

```
1
2     ...
3     ...
4     ...
5     ...
6
```

The "Body" tab also includes tabs for Cookies, Headers (4), Test Results (1/1), Pretty, Raw, Preview, Visualize, and JSON. Below the preview area, the status bar indicates: Status: 400 Bad Request, Time: 60 ms, Size: 565 B, and Save Response.

The response body is displayed as:

```
1
2     ...
3     ...
4     ...
5     ...
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
229
230
231
232
233
234
235
236
237
238
239
239
240
241
242
243
244
245
246
247
248
249
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2397
2397
239
```

If the body parameters already exist, Status Code 400 (Bad Request) will be generated

The screenshot shows the Postman application interface. On the left, the sidebar lists collections, APIs, environments, mock servers, monitors, flows, and history. The 'TESTS-marketplace_services_CSI5112' collection is expanded, showing 'Users', 'Category', 'Order', 'Product', 'Answer', and 'Question' endpoints. The 'Question' endpoint has two options: 'Get Question' (GET) and 'Add Question' (POST). The 'Add Question' option is selected, and its details are shown in the main panel.

POST `((url))/Question`

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1
2   ...
3     ...
4       ...
5         ...
6             ...
```

Body Cookies Headers (4) Test Results (1/1)

Status: 400 Bad Request Time: 58 ms Size: 182 B Save Response

Pretty Raw Preview Visualizer Text

```
1 Question with id already exists
```

5.4) Delete Question by id:

<https://localhost:7136/api/Question/1>

If the Question id is correct, the question will be deleted with status code 200.

The screenshot shows the Postman application interface. On the left, the sidebar contains sections for Collections, APIs, Environments, Mock Servers, Monitors, and History. The main workspace displays a collection named 'TESTS-marketplace_services_CSIS112'. A specific test case titled 'Delete Question by id' is selected, showing a 'DELETE' request to '((url))/Question/13'. The 'Tests' tab is active, containing the following JavaScript code:

```
12    );
13  }
14
15
16
17
18  if(pm.response.code == 400)
19  {
20    console.log(pm.response)
21    pm.test("Status Code 400:Please check Body tab for more information", function () {
22      pm.expect(pm.response.code).to.eql(400);
23    });
24 }
```

To the right of the code, a status bar indicates 'Status: 200 OK', 'Time: 57 ms', 'Size: 152 B', and a 'Save Response' button. A context menu is open over the code area, listing options like 'Test scripts are written in JavaScript, and are run after the response is received.', 'Learn more about tests scripts', 'SNIPPETS', 'Get an environment variable', 'Get a global variable', 'Get a variable', 'Get a collection variable', and 'Set an environment variable'. At the bottom, there are tabs for 'Body', 'Cookies', 'Headers (4)', 'Test Results (1/1)', and buttons for 'Pretty', 'Raw', 'Preview', 'Visualize', and 'JSON'.

Question is deleted (Verified using Get by Id API)

The screenshot shows the Postman interface with a collection named "TESTS-marketplace_services_CSIS112". A GET request is made to the endpoint `(url)/Question/13`. The response status is 404 Not Found, and the body contains the message "No Question found for this id".

If the question id is wrong, Status Code 400 (Bad Request) will be generated.

The screenshot shows the Postman interface with a collection named "TESTS-marketplace_services_CSIS112". A DELETE request is made to the endpoint `(url)/Question/100`. The response status is 400 Bad Request, and the body contains the message "Question with id does not exists".

If the Question id is null, Status Code 400 (Bad Request) is generated.

The screenshot shows the Postman interface with a collection named "TESTS-marketplace_services_CSIS112". A DELETE request is made to the endpoint `(url)/Question/null`. The response status is 400 Bad Request, and the body is a JSON object containing validation errors:

```

1   {
2     "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
3     "title": "One or more validation errors occurred.",
4     "status": 400,
5     "traceId": "00-3a06a5927dee7adc3093f7b5b80ebc74-9d61bcc04401d63f-00",
6     "errors": [
7       "Id": [
8         "The value 'null' is not valid."
9       ]
10    ]
11  }
  
```

6) Answer:

Answer

- GET /api/Answer
- POST /api/Answer
- GET /api/Answer/{questionId}

6.1) Get All Answers

<https://localhost:7136/api/Answer>

If the Answers are listed then status code 200 will be generated and the answers will be fetched.

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CSI5112'. Under the 'Answer' folder, there is a 'GET Get Answer' request. The request URL is `({url})/Answer`. The response status is 200 OK, time 105 ms, size 1.19 KB. The response body is a JSON array of four objects, each representing an answer with fields: id, questionId, description, and userName. The answers are: 1. id: 1, questionId: 1, description: "Yes I agree, same applies for Winter too.", userName: "Esra Ersan"; 2. id: 2, questionId: 1, description: "Yes I agree, same applies for Winter too.", userName: "Bhavesh Bisth"; 3. id: 3, questionId: 1, description: "Yes I agree, same applies for Winter too.", userName: "Vasu Mistry"; 4. id: 4, questionId: 1, description: "Yes I agree, same applies for Winter too.", userName: "Esra Ersan".

If the Answers are not listed then Status Code 404 (Not Found) will be generated.

6.2) Get Answer by Question id

<https://localhost:7136/api/Answer/1>

If the Question id Exist, then the answer will be fetched, and the status code 200 will be generated.

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CSI5112'. Under the 'Answer' folder, there is a 'GET Get Answer by Question id' request. The request URL is `({url})/Answer/1`. The response status is 200 OK, time 102 ms, size 574 B. The response body is a JSON array of four objects, each representing an answer with fields: id, questionId, description, and userName. The answers are: 1. id: 1, questionId: 1, description: "Yes I agree, same applies for Winter too.", userName: "Esra Ersan"; 2. id: 2, questionId: 1, description: "Yes I agree, same applies for Winter too.", userName: "Bhavesh Bisth"; 3. id: 3, questionId: 1, description: "Yes I agree, same applies for Winter too.", userName: "Vasu Mistry"; 4. id: 4, questionId: 1, description: "Yes I agree, same applies for Winter too.", userName: "Esra Ersan".

If the Question id does not exist, then status code 404(Not Found) will be generated.

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CSI5112'. A test script is present in the 'Tests' tab of the request configuration:

```
if(pm.response.code == 200)
```

The response body shows the expected error message:

```
No answers found for this questionId
```

Test results indicate a status of 404 Not Found.

If the Question id is null, then Status Code 400 (Bad Request) will be generated.

The screenshot shows the Postman interface with the same collection. A test script is present in the 'Tests' tab of the request configuration:

```
if(pm.response.code == 200)
```

The response body shows the validation error message:

```
"type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",  
"title": "One or more validation errors occurred.",  
"status": 400,  
"traceId": "00-7ec1a12f28a6e8e4f3f34d4d7cc3fb-f086c9f2add1d287-00",  
"errors": {  
    "questionId": [  
        "The value 'null' is not valid."  
    ]  
}
```

Test results indicate a status of 400 Bad Request.

6.3) Post/Add Answer

<https://localhost:7136/api/Answer>

If the Body parameters are validated, The Answer will be posted an status code is 200.

The screenshot shows the Postman interface with a collection named 'TESTS-marketplace_services_CS15112'. A POST request is made to the '/Answer' endpoint with the URL <https://localhost:7136/api/Answer>. The 'Body' tab contains the following JSON payload:

```
1 "id": 11,
2 "questionId": 3,
3 "description": "Yes I agree, same applies for Winter too.",
4 "userName": "Esra Ersan"
```

The response status is 200 OK, and the body of the response is true.

Answer is added (Verified by Get by question id API)

The screenshot shows the Postman interface with the same collection. A GET request is made to the '/Get Answer by Question id' endpoint with the URL <https://localhost:7136/api/Answer/3>. The 'Tests' tab contains a JavaScript test script:

```
3 if(pm.response.code == 200)
4 {
5 }
```

The response status is 200 OK, and the body of the response is a JSON array containing two answer objects.

If the parameters are null, Status Code 400 (Bad request) will be generated.

The screenshot shows the Postman application interface. On the left sidebar, there are sections for Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. The main area displays a collection named "TESTS-marketplace_services_CSI5112". Under this collection, there is a "Answers" folder which contains a "POST Add Answer" request. This request has a "Body" tab selected, showing the following JSON payload:

```
1 id: null
2 questionId: 3
3 description: "Yes I agree, same applies for Winter too."
4 userName: "Esra Ersan"
```

Below the request, the response status is shown as "Status: 400 Bad Request" with a "Save Response" button. The response body is displayed in "Pretty" format, showing validation errors:

```
1 type: "https://tools.ietf.org/html/rfc7231#section-6.5.1"
2 title: "One or more validation errors occurred."
3 status: 400
4 traceId: "00-54bad8aalc3affbded2a0960392ec958-e7cf3d9ab7864885-00"
5 errors: [
6   answer: [
7     "The answer field is required."
8   ],
9   "$.id": [
10     "The JSON value could not be converted to marketplace.services.CSI5112.Models.Answer. Path: $ .id" ]]
```

If the Question id is duplicate, Status Code 400 (Bad Request) will be generated.

The screenshot shows the Postman application interface. On the left, the sidebar contains sections for Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. The 'Collections' section is expanded, showing a collection named 'TESTS-marketplace_services_CSI112'. This collection contains several items: 'Users', 'Category', 'Order', 'Product', 'Answer' (which is expanded to show 'Get Answer', 'POST Add Answer', and 'GET Get Answer by Question id'), 'Question', and two other collections: 'marketplace_services_CSI112' and 'TESTS-marketplace_services_CS1112'. The 'Answer' item under 'TESTS-marketplace_services_CSI112' is currently selected.

The main workspace shows a 'POST' request to '((url))/Answer'. The 'Body' tab is selected, showing the following JSON payload:

```
1 "id": 11,
2 ...
3 ...
4 ...
5 ...
6 ...
```

The response pane at the bottom shows a status of '400 Bad Request'. The response body contains the message: '1 Answer with same id exists in db'.