

## 1. Add Question (POST)

GET http://localhost:8080/api/ POST http://localhost:8080/ap POST http://localhost:8080/ap + ...

HTTP http://localhost:8080/api/v1/question/add

POST http://localhost:8080/api/v1/question/add

Params Authorization Headers (9) Body Pre-request Script Tests Settings

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

```
3  "optionA": "5Java",
4  "optionB": "Java5",
5  "optionC": "23Java",
6  "optionD": "Error",
7  "correctAnswer": "A",
8  "category": "JAVA",
9  "difficultyLevel": "EASY"
10 }
11
```

Body Cookies Headers (5) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1  {
2    "id": 1,
3    "title": "What is the output of System.out.println(2 + 3 + \"Java\")?",
4    "optionA": "5Java",
5    "optionB": "Java5",
6    "optionC": "23Java",
7    "optionD": "Error",
8    "correctAnswer": "A",
9    "category": "JAVA",
10   "difficultyLevel": "EASY"
11 }
```

## 2. Get Paginated Questions (GET)

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/api/v1/question?page=0&size=10`. The request is sent, and the response is displayed in JSON format. The response includes a list of questions and pagination details.

```
1 {
2   "title": "What is the output of System.out.println(2 + 3 + \"Java\")?",
3   "optionA": "5Java",
4   "optionB": "Java5",
5   "optionC": "23Java",
6   "optionD": "Error",
7   "correctAnswer": "A",
8   "category": "JAVA",
9 }
10
11 {
12   "pageable": {
13     "pageNumber": 0,
14     "pageSize": 20,
15     "sort": {
16       "sorted": false,
17       "unsorted": true,
18       "empty": true
19     },
20     "offset": 0,
21     "unpaged": false,
22     "paged": true
23   }
24 }
```

## 3. Get Questions by Category (GET)

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/api/v1/question/category/JAVA`. The request is sent, and the response is displayed in JSON format. The response is a list of questions belonging to the 'JAVA' category.

```
1 {
2   "id": 1,
3   "title": "What is the output of System.out.println(2 + 3 + \"Java\")?",
4   "optionA": "5Java",
5   "optionB": "Java5",
6   "optionC": "23Java",
7   "optionD": "Error",
8   "correctAnswer": "A",
9   "category": "JAVA",
10  "difficultyLevel": "EASY"
11 },
12 {
13   "id": 2,
14   "title": "What is the size of int in Java?",
15   "optionA": "2 bytes",
16   "optionB": "4 bytes",
17   "optionC": "8 bytes",
18   "optionD": "Depends on OS",
19   "correctAnswer": "B",
20   "category": "JAVA",
21   "difficultyLevel": "EASY"
22 },
23 }
```

## 4. Create Quiz (POST)

The screenshot shows a Postman interface with a POST request to `http://localhost:8080/api/v1/quiz/create?category=JAVA&difficulty=EASY`. The request body is a JSON object: `{ "quizId": 1, "message": "Quiz created successfully" }`. The response status is 200 OK, with a time of 543 ms and a size of 214 B. The response body is also a JSON object: `{ "quizId": 1, "message": "Quiz created successfully" }`.

```
1 {
2   "quizId": 1,
3   "message": "Quiz created successfully"
4 }
```

## Get quiz questions (GET):

The screenshot shows a Postman interface with a GET request to `http://localhost:8080/api/v1/quiz/1/questions`. The response status is 200 OK, with a time of 99 ms and a size of 592 B. The response body is a JSON array of two quiz questions.

```
1 [
2   {
3     "id": 2,
4     "title": "Which keyword is used to inherit a class in Java?",
5     "optionA": "super",
6     "optionB": "this",
7     "optionC": "extends",
8     "optionD": "implements"
9   },
10  {
11    "id": 1,
12    "title": "What is the size of int in Java?",
13    "optionA": "2 bytes",
14    "optionB": "4 bytes",
15    "optionC": "8 bytes",
16    "optionD": "Depends on OS"
17  }
18 ]
```

## Submit quiz (POST)

The screenshot shows the Postman interface for a POST request to `http://localhost:8080/api/v1/quiz/1/submit`. The request is in the "Body" tab, using the "raw" format with JSON content. The response is visible at the bottom, showing a status of 200 OK.

**Request:**

```
POST http://localhost:8080/api/v1/quiz/1/submit
```

**Body (raw):**

```
[{"questionId": 1, "selectedOption": "A"}, {"questionId": 2, "selectedOption": "C"}, {"questionId": 3, "selectedOption": "B"}]
```

**Response:**

Status: 200 OK Time: 13 ms Size: 165 B

## Exception Testing in Postman for QuizApp

The screenshot shows the Postman interface for a GET request to `http://localhost:8080/api/quizzes/999`. The request is in the "Query Params" tab. The response is visible at the bottom, showing a status of 404 Not Found.

**Request:**

```
GET http://localhost:8080/api/quizzes/999
```

**Query Params:**

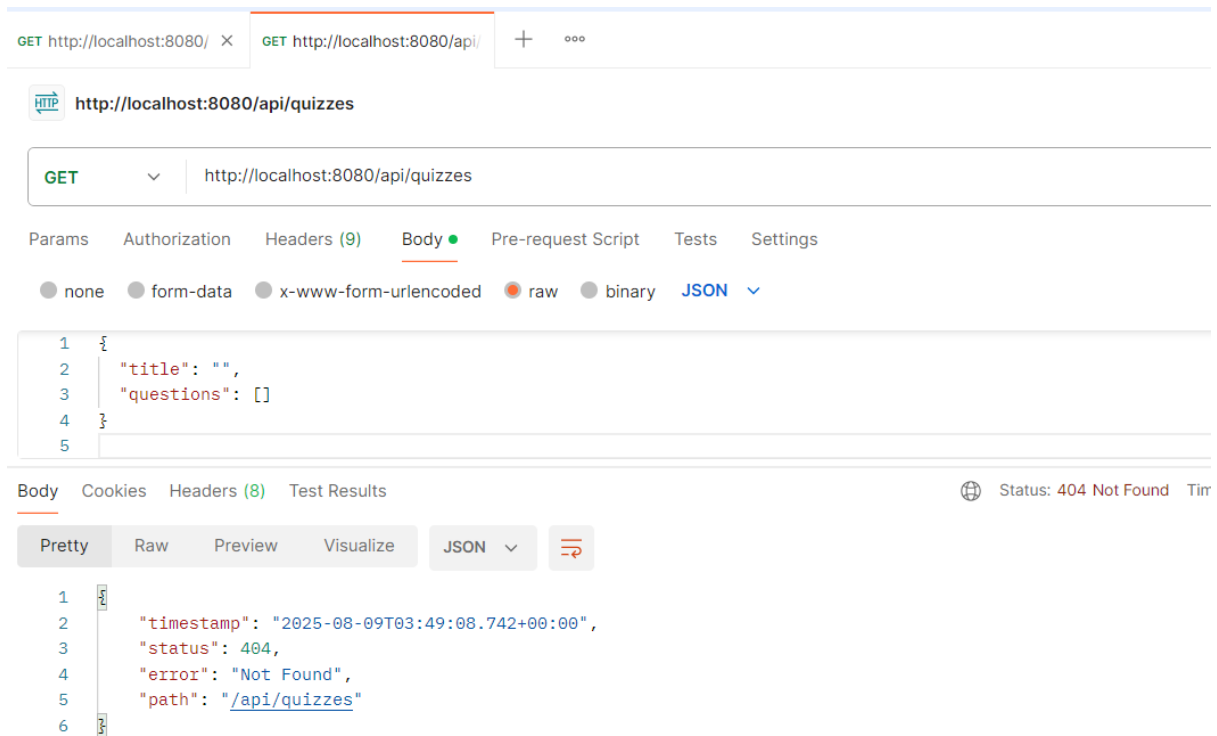
Key	Value
Key	Value

**Response:**

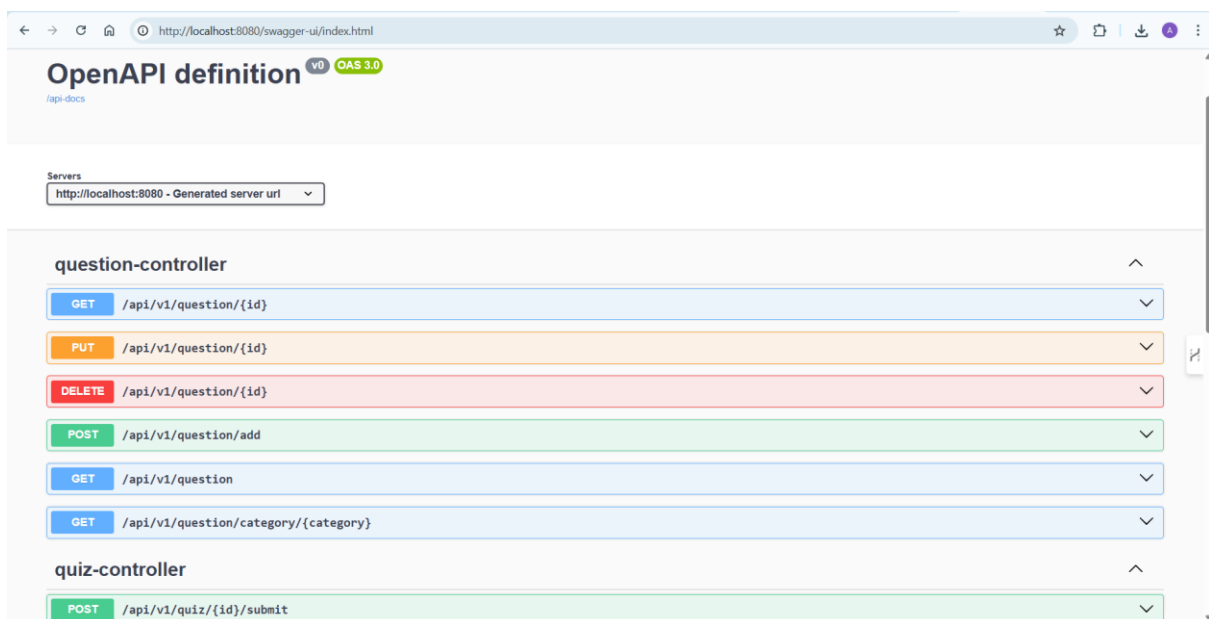
Status: 404 Not Found Time: 1178 ms Size: 364 B

**Body (JSON):**

```
{  "timestamp": "2025-08-09T03:45:35.318+00:00",  "status": 404,  "error": "Not Found",  "path": "/api/quizzes/999"}
```



**/swagger-ui** – An interactive web interface that lets you **view and test your API endpoints** directly in the browser.



[illegible]