Quiz Mono Repo API

1.1 Add Question

Endpoint:
POST /question

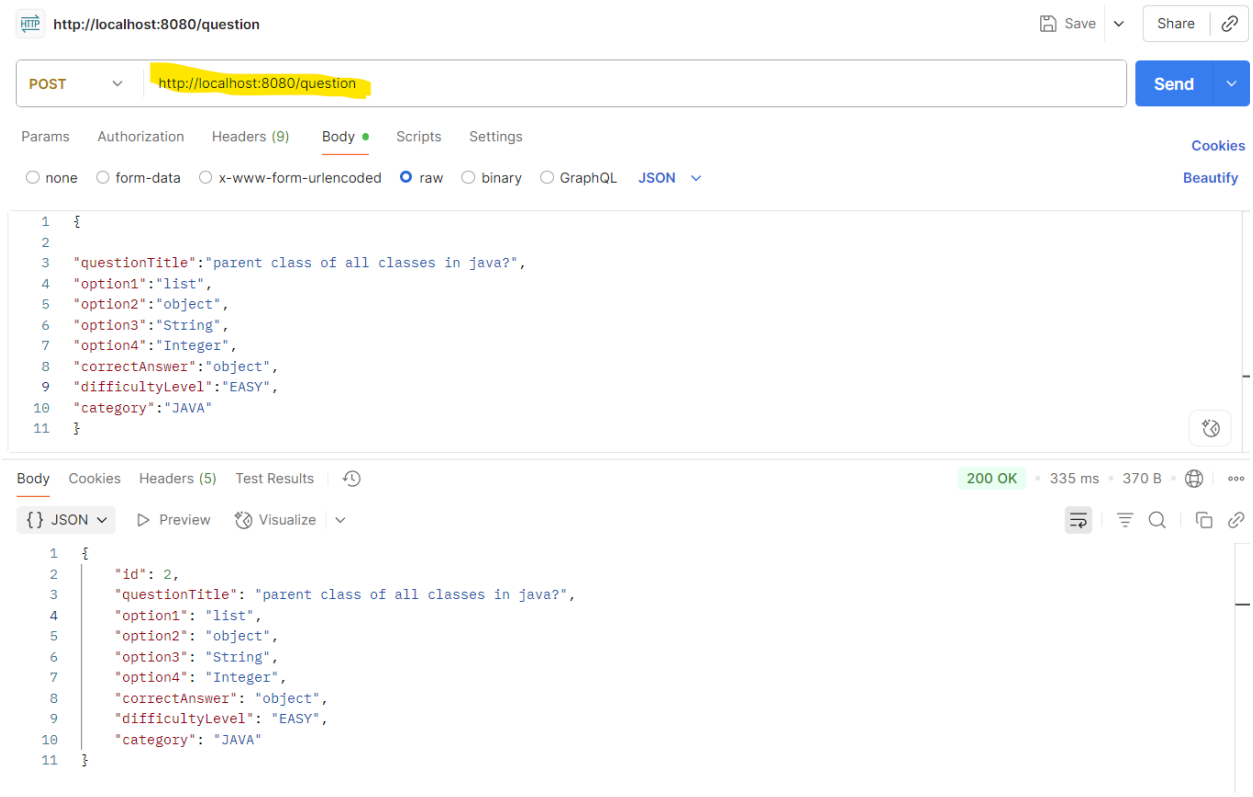
Description:
Creates a new question in the system.

Request Body (JSON):

json

```
{
  "questionTitle": "What is JVM?",
  "option1": "Java Virtual Machine",
  "option2": "Java Very Much",
  "option3": "Just Virtual Memory",
  "option4": "Java Virtual Memory",
  "correctAnswer": "Java Virtual Machine",
  "difficultyLevel": "EASY",
  "category": "JAVA"
}
```

Response (201 Created):



1.2 Get All Questions (Paginated)

Endpoint:

GET /question?page={page}&size={size}

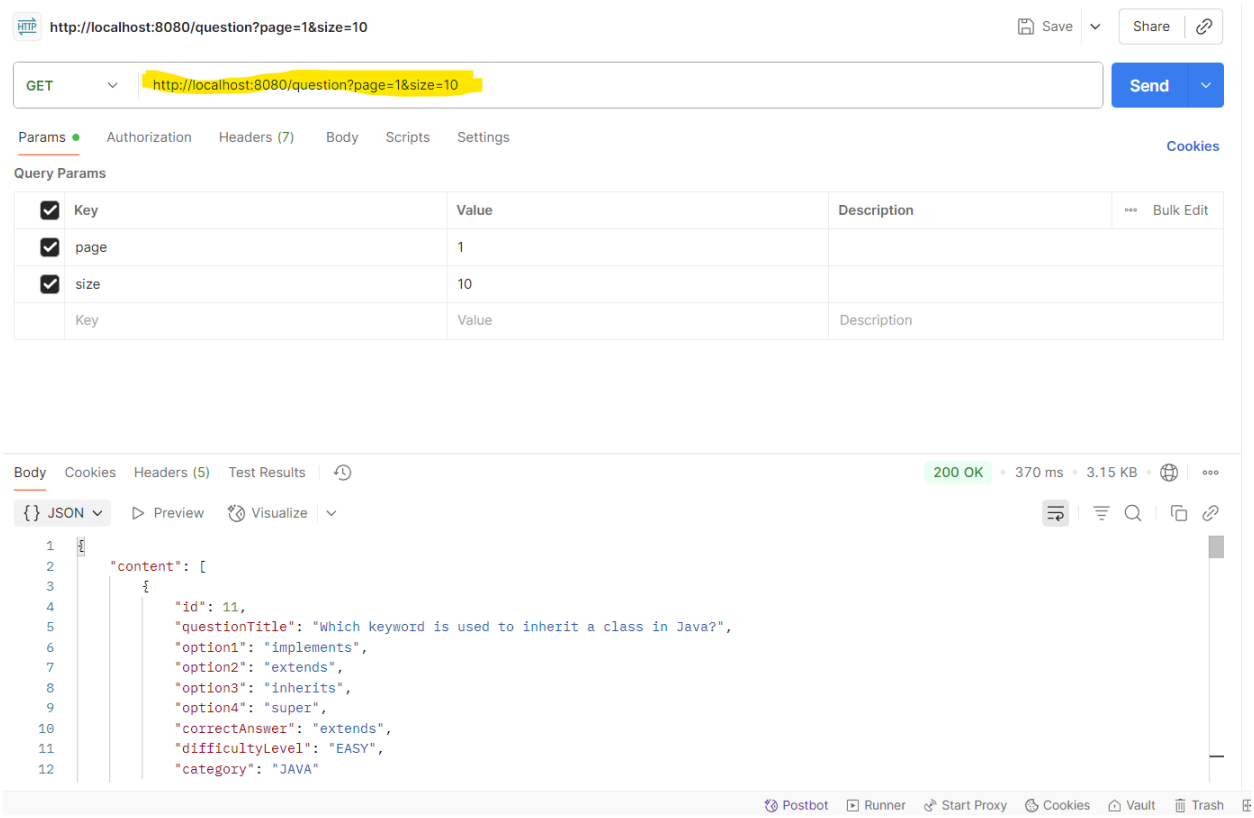
Description:

Retrieves all questions with pagination.

Example Request:

GET /question?page=0&size=5

Response:



1.3 Filter Questions by Category & Difficulty

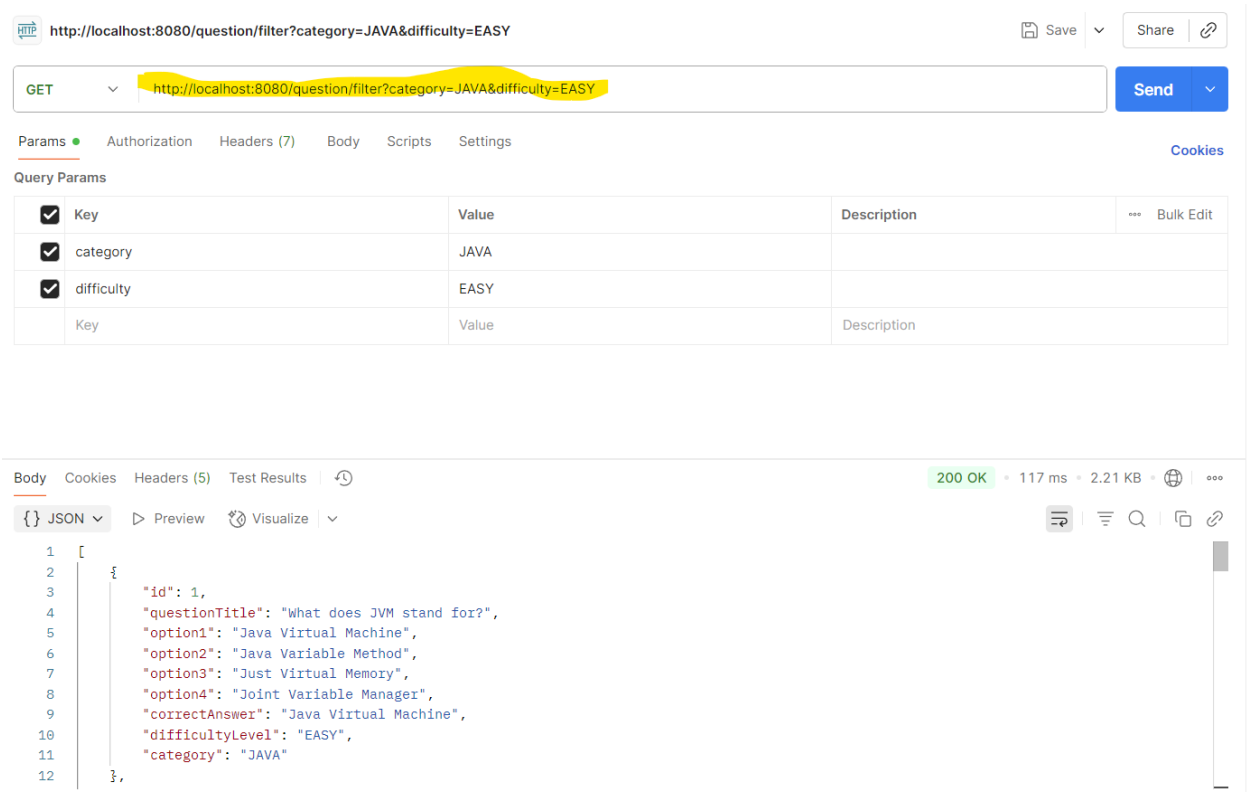
Endpoint:

GET /question/filter?category={category}&difficulty={difficulty}

Example Request:

GET /question/filter?category=JAVA&difficulty=EASY

Response:



1.4 Update Question Partially

Endpoint:
PATCH /question/{id}

Description:
Partially updates question details.

Request Body Example:

Json:

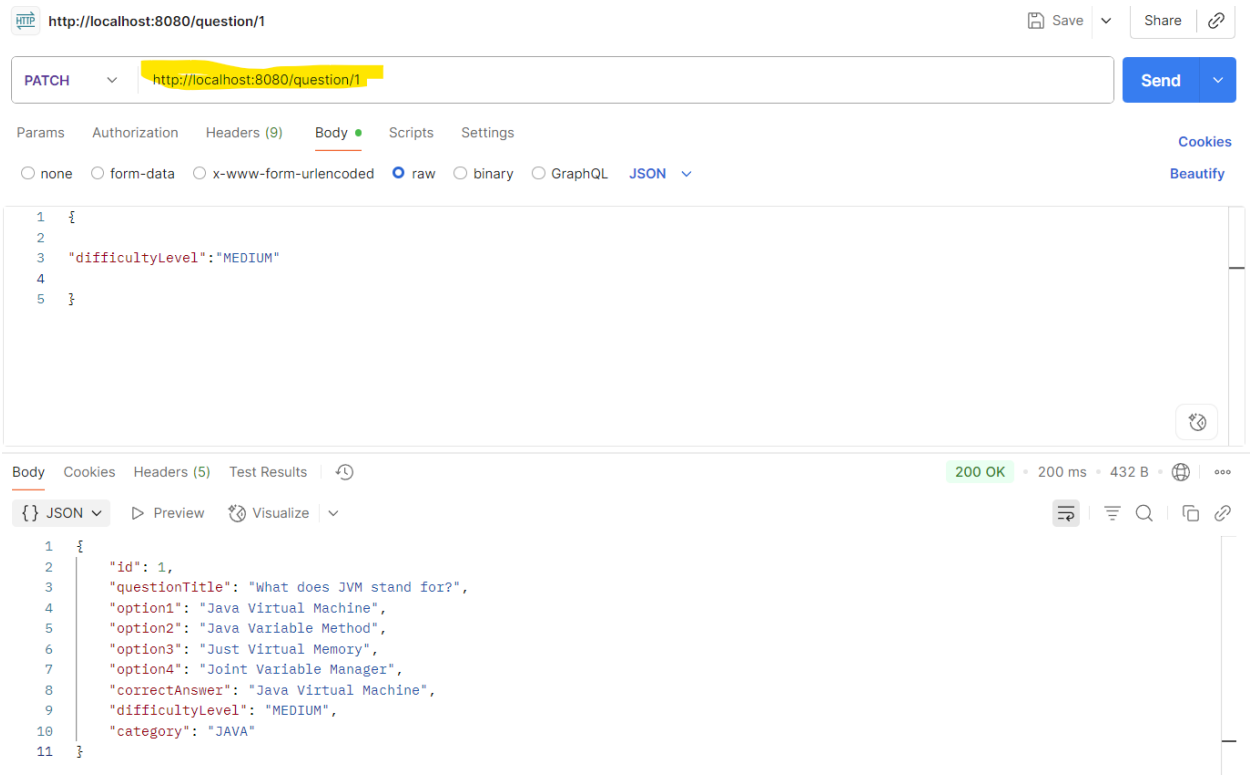
{

"questionTitle": "Explain JVM in Java",

"difficultyLevel": "MEDIUM"

}

Response:



1.5 Delete Question

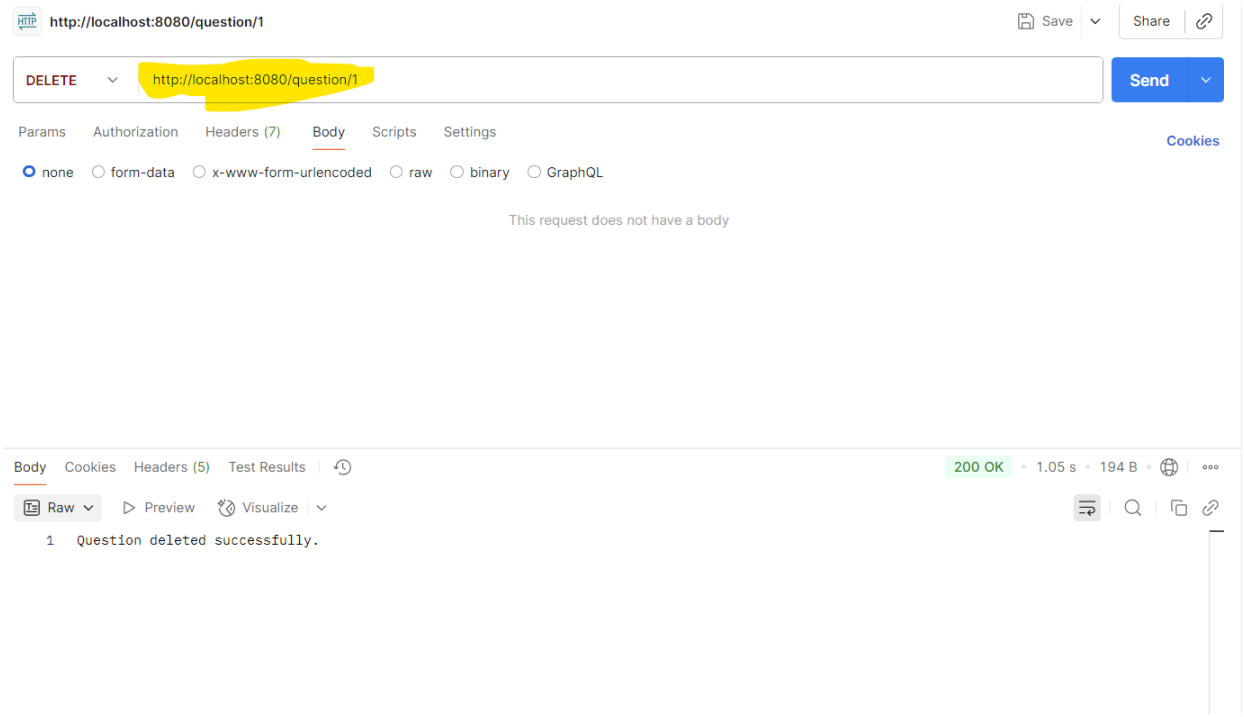
Endpoint:

DELETE /question/{id}

Example:

DELETE /question/1

Response:



Quiz APIs:

2.1 Create Quiz

Endpoint:
POST /quiz

Request Body:

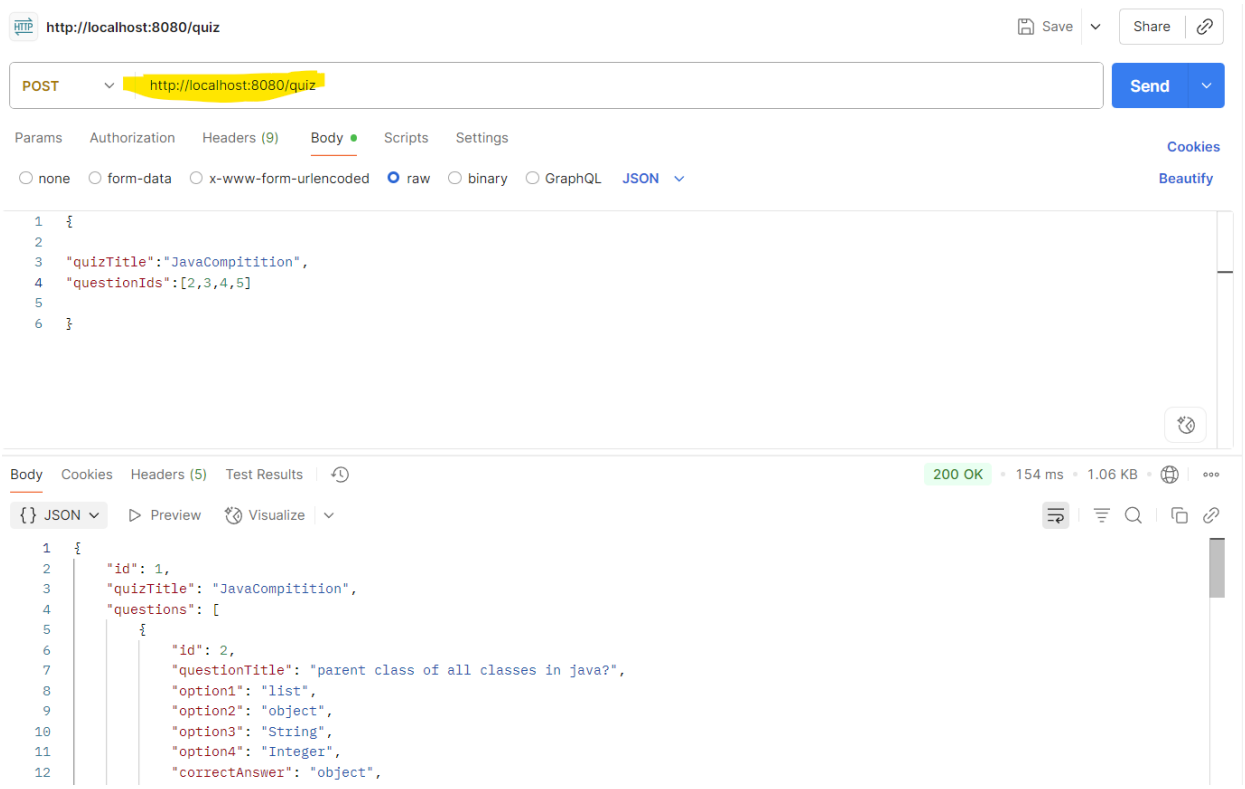
Json

{

"quizTitle": "Java Basics Quiz",

"questionIds": [1, 2, 3]

}



2.2 Get Quiz by ID

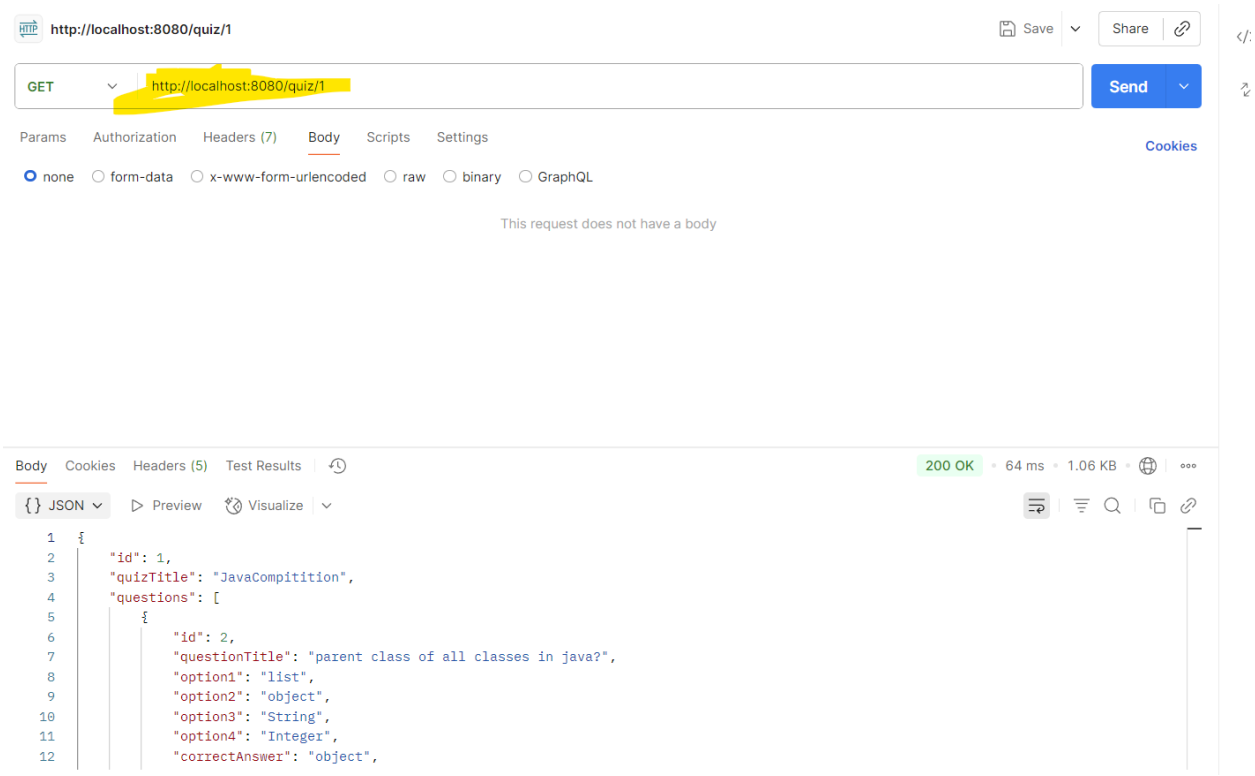
Endpoint:

GET /quiz/{id}

Example:

GET /quiz/1

Response:



2.3 Submit Quiz

Endpoint:

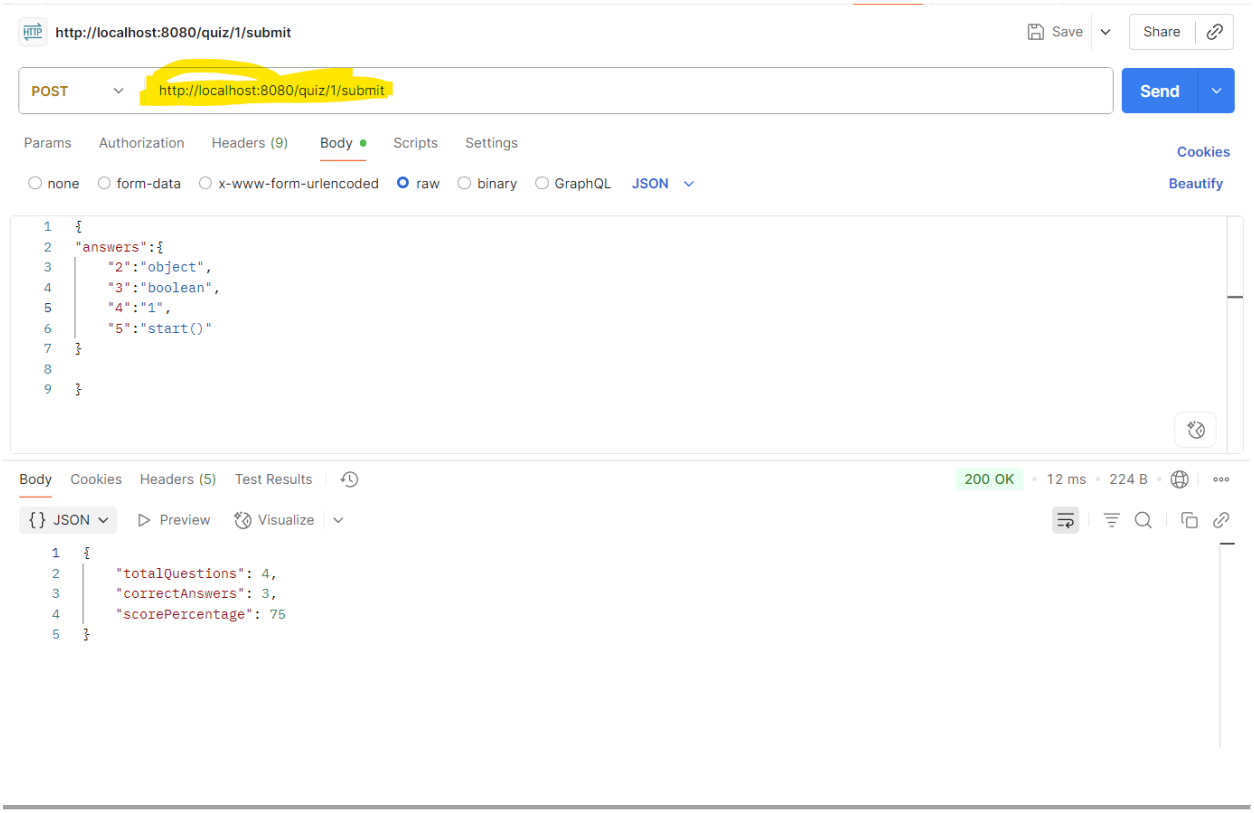
POST /quiz/{id}/submit

Request Body Example:

Json:

```
{
  "answers": {
    "1": "Java Virtual Machine",
    "2": "Spring Framework"
  }
}
```

Response:

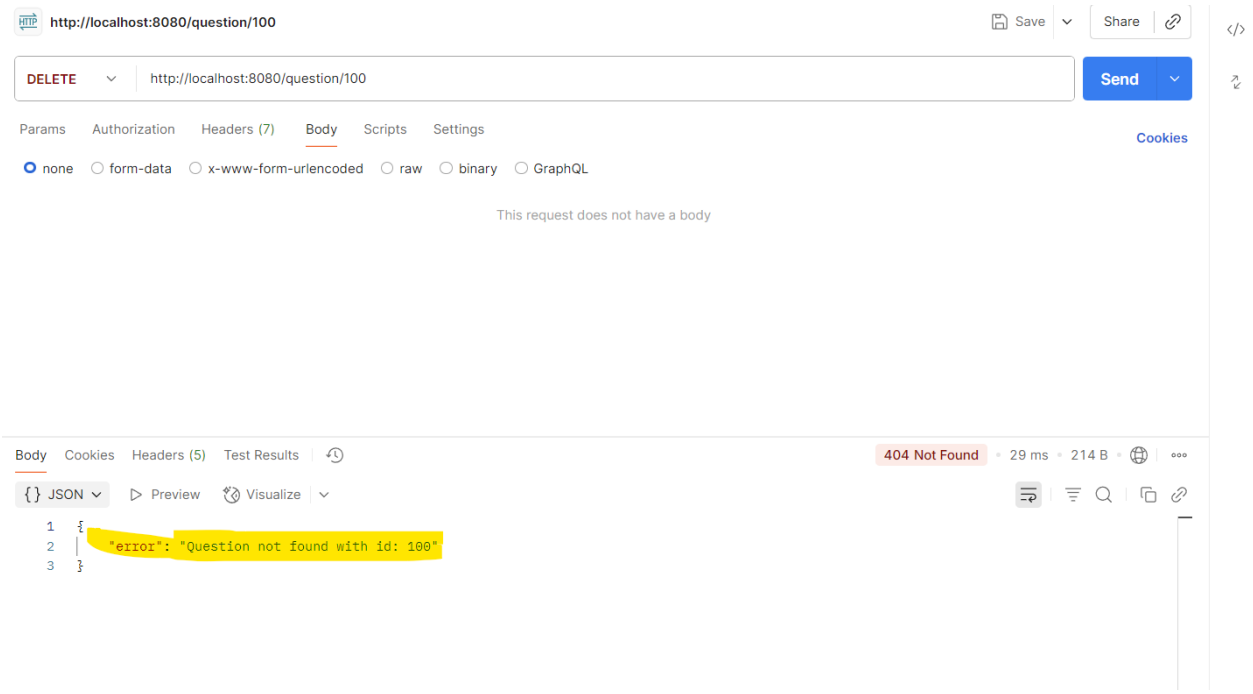


Extra Features:

1) Custom Exception Handling

1. Implemented **@ControllerAdvice** to centralize exception handling for all controllers.
2. Created **custom exceptions** such as ResourceNotFoundException and InvalidRequestException to return meaningful error messages.
3. All errors are returned in **consistent JSON format** with message, **proper HTTP status codes** (404, 400, etc.).

Example Error Response:



---

2) Aspect-Oriented Programming (AOP)

1. Added **logging aspect** using @Aspect and @Around to log API execution time, method entry/exit, and parameters.
2. Helps in **monitoring performance** and debugging without modifying controller/service logic.
3. Can be extended for **security checks, transaction handling, or auditing** in the future.

Example Log Output:

---

3) Swagger API Documentation

1. Integrated **Springdoc OpenAPI (Swagger UI)** to generate interactive API documentation automatically.

2. Available at /swagger-ui.html or /swagger-ui/index.html for easy testing of endpoints.
3. Includes **request/response schemas**, **enum values**, and example bodies for each API.