

Implementing questions:-

1.addNewQuestion:-

The screenshot shows a REST client interface for a POST request to `http://localhost:8080/addNewquestion`. The request body is a JSON object with the following structure:

```
{  "opt4": "super",  "correctAnswer": "extends",  "level": "EASY",  "category": "java"}
```

The response is a 200 OK status with a response time of 297 ms and a body size of 368 B. The response body is a JSON object:

```
{  "id": 1,  "questionText": "Which keyword is used to inherit a class in Java?",  "opt1": "extends",  "opt2": "implements",  "opt3": "inherits",  "opt4": "super"}
```

2.getAllQuestions:-

The screenshot shows a REST client interface for a GET request to `http://localhost:8080/getAllQuestions`. The response is a 200 OK status with a response time of 1341 ms and a body size of 684 B. The response body is a JSON array containing one question object:

```
{  "content": [    {      "id": 1,      "questionText": "Which keyword is used to inherit a class in Java?",      "opt1": "extends",      "opt2": "implements",      "opt3": "inherits",      "opt4": "super"    }  ]}
```

3.getById implementing with custom exception:--

The screenshot shows a Postman interface for a GET request to `http://localhost:8080/getById/2`. The request is sent, and the response is a 404 Not Found status with a response time of 140 ms and a body size of 245 B. The response body is displayed in JSON format:

```
{  "message": "question with this id not found2",  "status": "questionNotFound"}
```

4.getById:;

The screenshot shows a Postman interface for a GET request to `http://localhost:8080/getById/1`. The request is sent, and the response is a 200 OK status with a response time of 137 ms and a body size of 368 B. The response body is displayed in JSON format:

```
{  "id": 1,  "questionText": "Which keyword is used to inherit a class in Java?",  "opt1": "extends",  "opt2": "implements",  "opt3": "inherits"}
```

5.updateQuestion:-

HTTP <http://localhost:8080/updateQuestion/1> Save </>

PUT <http://localhost:8080/updateQuestion/1> Send ⌵

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

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary **JSON** ⌵ Beautify

```

5      "opt3": "inherits",
6      "opt4": "super",
7      "correctAnswer": "extends",
8      "level": "EASY",
9      "category": "java"
10
11

```

Body Cookies Headers (5) Test Results 200 OK 496 ms 368 B Save Response ⌵

Pretty Raw Preview Visualize **JSON** ⌵ ⌵

```

1
2      "id": 1,
3      "questionText": "Which keyword is used to inherit a class in Java?",
4      "opt1": "extends",
5      "opt2": "implements",
6      "opt3": "inherits",
7      "opt4": "super",
8      "correctAnswer": "extends",
9      "level": "EASY",
10     "category": "java"
11

```

6.deleteQuestion:-

HTTP <http://localhost:8080/deleteQuestion/1> Save </>

DELETE <http://localhost:8080/deleteQuestion/1> Send ⌵

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

Headers 8 hidden

Key	Value	Bulk Edit
Key	Value	

Body Cookies Headers (4) Test Results 200 OK 749 ms 123 B Save Response ⌵

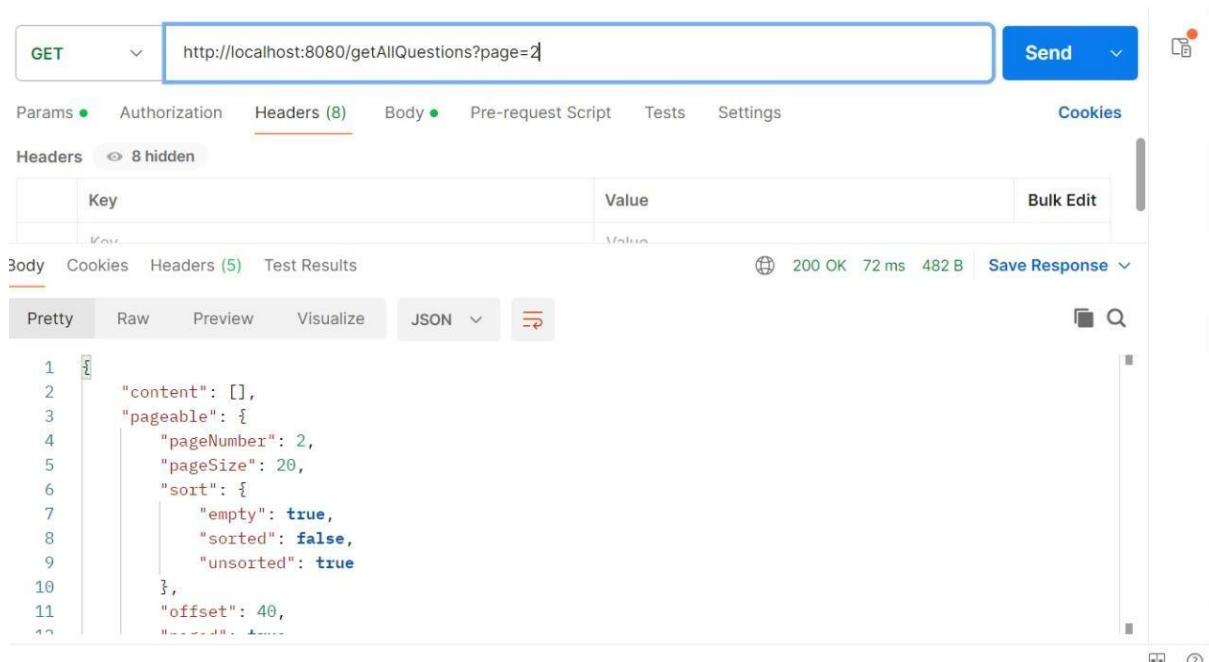
Pretty Raw Preview Visualize **Text** ⌵ ⌵

```

1

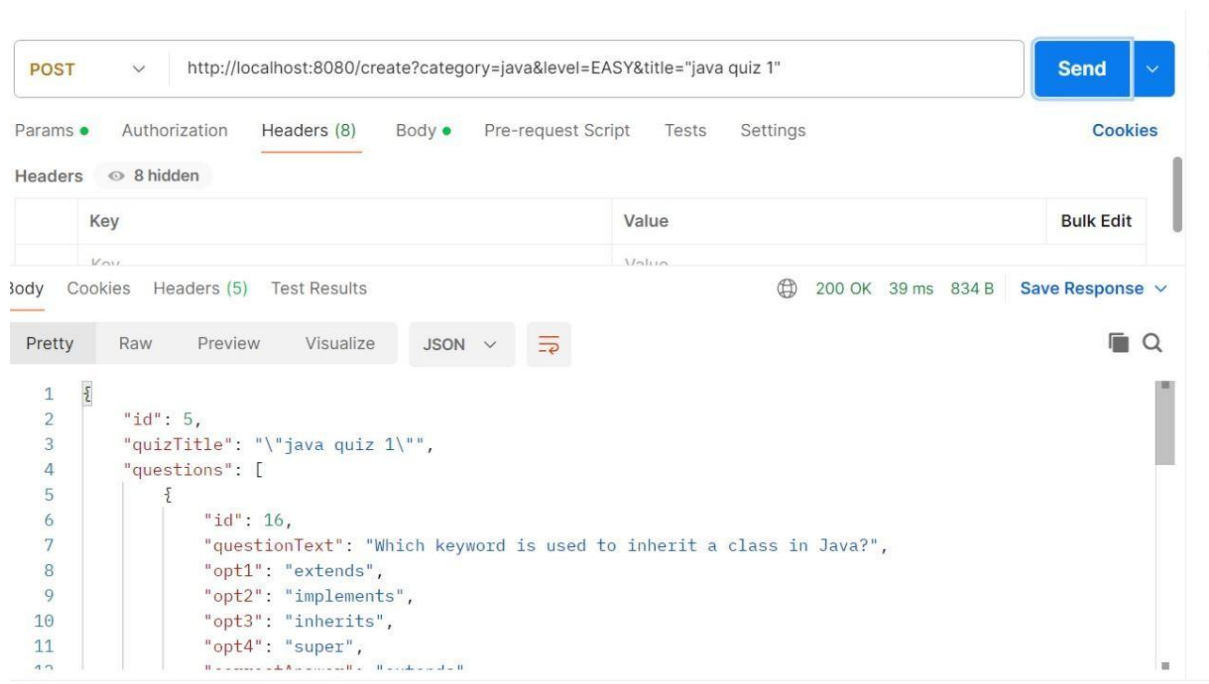
```

7.getAll question with implementing pagination:--



Quiz implementation

1.Create quiz:-



2.Get Quiz BY id:-

GET http://localhost:8080/getQuizById/2 Send

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

Key	Value	Bulk Edit
Key	Value	

Body Cookies Headers (5) Test Results 200 OK 74 ms 602 B Save Response

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 14,
4     "opt2": "implements",
5     "op3": "inherits",
6     "op4": "super",
7     "questionTitle": "Which keyword is used to inherit a class in Java?",
8     "opt1": "extends"
9   },
10  {
11    "id": 13,
```

3.Submitting quiz with getting score on console:-

HTTP http://localhost:8080/submitQuiz/1 Save

POST http://localhost:8080/submitQuiz/1 Send

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

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

```
1 {
2   "id": 15,
3   "userAnswer": "extends"
4 },
5 {
6   "id": 8,
7   "userAnswer": "Paris"
8 },
9 }
```

Body Cookies Headers (5) Test Results 200 OK 26 ms 165 B Save Response

Pretty Raw Preview Visualize JSON

1 2

4.Implementing Aop:--

```

1 package com.example.demo.aop;
2
3 import org.aspectj.lang.JoinPoint;
4 import org.aspectj.lang.annotation.Aspect;
5 import org.aspectj.lang.annotation.Before;
6 import org.springframework.stereotype.Component;
7
8 @Aspect
9 @Component
10 public class LoggingAspect {
11
12     @Before("execution(* com.example.demo.service.*(..))")
13     public void logBefore(JoinPoint joinPoint) {
14         System.out.println("AOP BEFORE: Calling method - " + joinPoint.getSignature().getName());
15     }
16 }

```

Output of Aop:-

```

2025-08-09T09:09:30.077+05:30 INFO 19392 --- [quizSpringBoot-App] [nio-8080-exec-1] o.s.web
2025-08-09T09:09:31.208+05:30 INFO 19392 --- [quizSpringBoot-App] [nio-8080-exec-8] o.s.prin
AOP BEFORE: Calling method - createQuiz
Hibernate:
  SELECT
    *
  FROM
    question q
  WHERE
    q.category = ?

```

5.Implemented logging:-

```

@GetMapping("/getQuizByID/{id}")
public List<QuestionWrapper> getQuizQuestions(@PathVariable Integer id){
    log.info("quiz starting...");

    return quizService.getQuizQuestions(id);
}

```

6.Implementing validations:-

```

@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
@NotNull(message="question not be null")
private String questionText;

private String opt1;

```

7.Swagger ui:-

question-con		^
PUT	/updateQuestion/{id}	▼
POST	/addNewQuestion	▼
GET	/getId/{id}	▼
GET	/getAllQuestions	▼
DELETE	/deleteQuestion/{id}	▼
quiz-controller		^
POST	/submitQuiz/{id}	▼
POST	/create	▼
GET	/getQuizByID/{id}	▼

8.DataBase:-

The screenshot shows a database management interface. On the left, a tree view displays the database structure for 'quizquestion_db', including tables 'question' and 'quiz'. The main area shows a 'Result Grid' for the 'question' table. The table has columns: id, category, correct_answer, level, opt1, opt2, opt3, opt4, and question_text. The data rows show questions about Java keywords (super, extends, implements, inherits).

id	category	correct_answer	level	opt1	opt2	opt3	opt4	question_text
2	java	extends	EASY	extends	implements	inherits	super	Which keyword is used to inherit a class in Java?
3	java	extends	EASY	extends	implements	inherits	super	Which keyword is used to inherit a class in Java?
4	java	extends	EASY	extends	implements	inherits	super	Which keyword is used to inherit a class in Java?
5	java	extends	EASY	extends	implements	inherits	super	Which keyword is used to inherit a class in Java?
6	java	extends	EASY	extends	implements	inherits	super	Which keyword is used to inherit a class in Java?
7	java	extends	EASY	extends	implements	inherits	super	Which keyword is used to inherit a class in Java?

9.docs

Pretty-print ☒

```
{
  "openapi": "3.1.0",
  "info": {
    "title": "OpenAPI definition",
    "version": "v0"
  },
  "servers": [
    {
      "url": "http://localhost:8080",
      "description": "Generated server url"
    }
  ],
  "paths": {
    "/updateQuestion/{id}": {
      "put": {
        "tags": [
          "question-con"
        ],
        "operationId": "updateQuestion",
        "parameters": [
          {
            "name": "id",
            "in": "path",
            "required": true,
            "schema": {
              "type": "integer",
              "format": "int64"
            }
          }
        ],
        "requestBody": {
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/Question"
              }
            }
          }
        }
      }
    }
  }
}
```