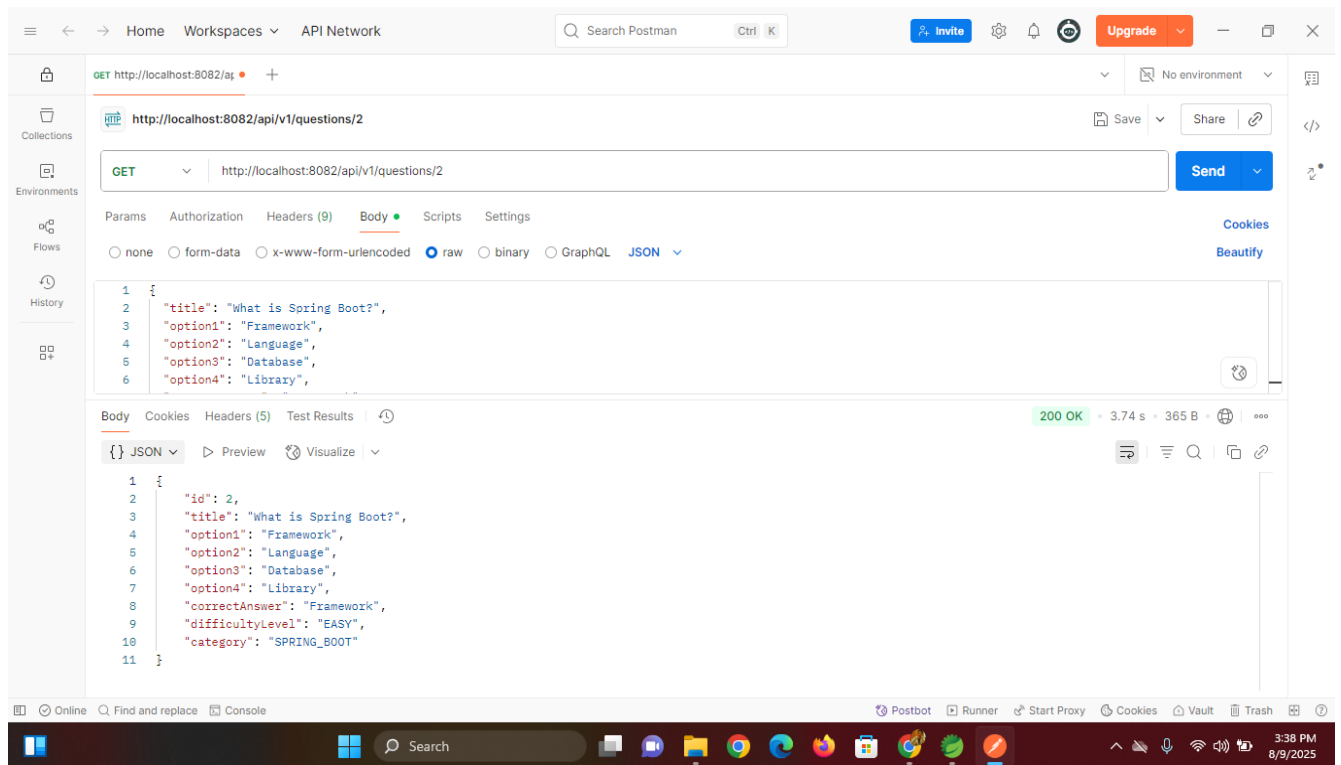


POSTMAN : GET

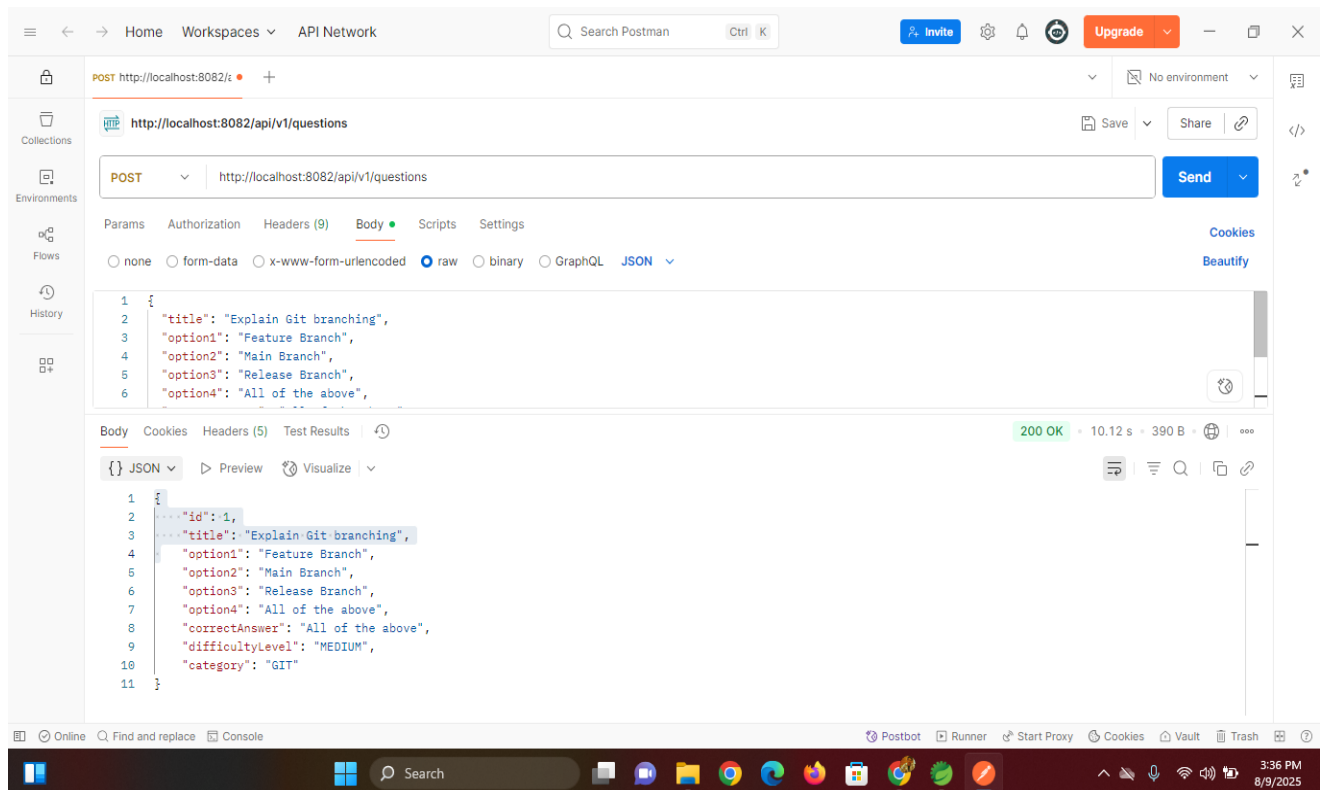


A screenshot of the Postman application showing a GET request to `http://localhost:8082/api/v1/questions/2`. The request is configured with the method 'GET' and the URL. The 'Body' tab is selected, showing a JSON payload with a title and four options. The response is displayed below, showing a 200 OK status and a JSON object with an id, title, options, correct answer, difficulty level, and category.

```
1 {
2   "title": "What is Spring Boot?",
3   "option1": "Framework",
4   "option2": "Language",
5   "option3": "Database",
6   "option4": "Library",
7 }
8
9
10
11
```

```
1 {
2   "id": 2,
3   "title": "What is Spring Boot?",
4   "option1": "Framework",
5   "option2": "Language",
6   "option3": "Database",
7   "option4": "Library",
8   "correctAnswer": "Framework",
9   "difficultyLevel": "EASY",
10  "category": "SPRING_BOOT"
11 }
```

POSTMAN :POST

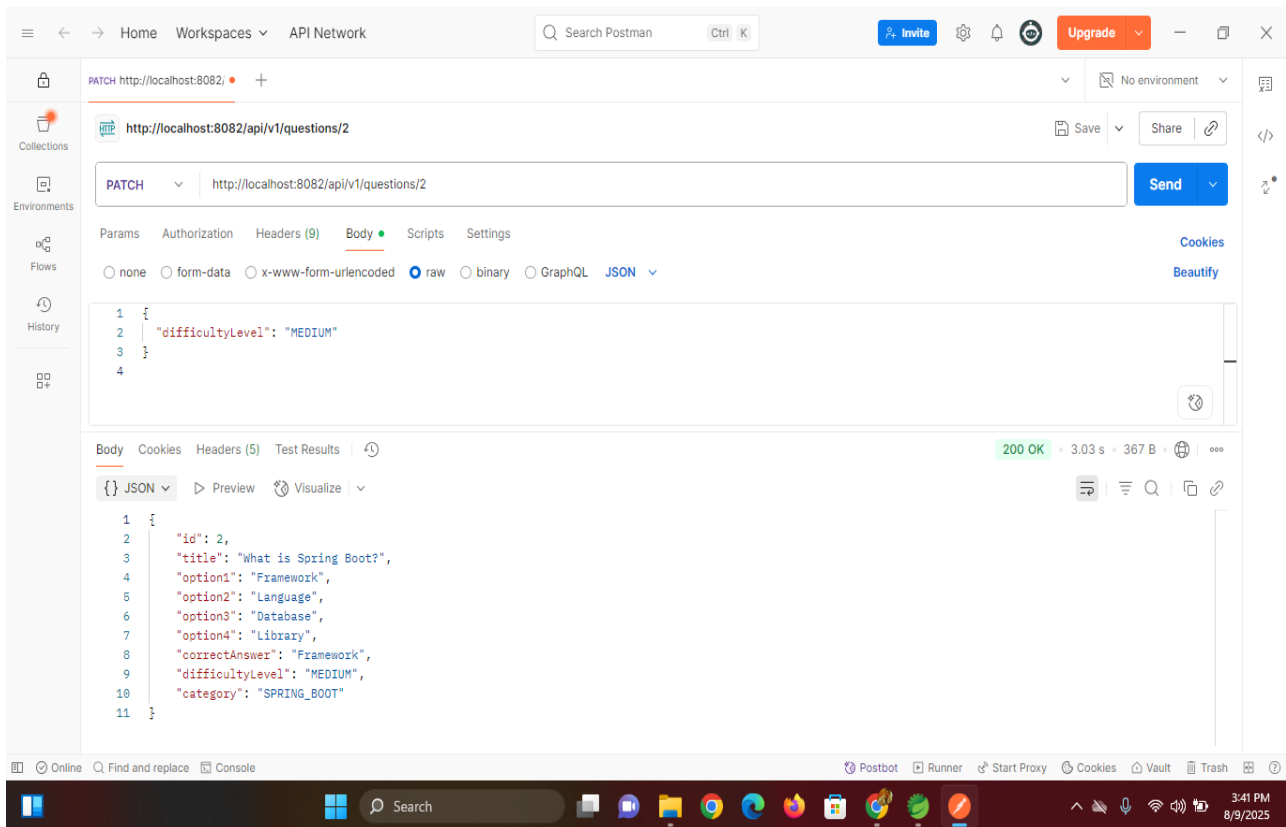


A screenshot of the Postman application showing a POST request to `http://localhost:8082/api/v1/questions`. The request is configured with the method 'POST' and the URL. The 'Body' tab is selected, showing a JSON payload with a title and four options. The response is displayed below, showing a 200 OK status and a JSON object with an id, title, options, correct answer, difficulty level, and category.

```
1 {
2   "title": "Explain Git branching",
3   "option1": "Feature Branch",
4   "option2": "Main Branch",
5   "option3": "Release Branch",
6   "option4": "All of the above",
7 }
8
9
10
11
```

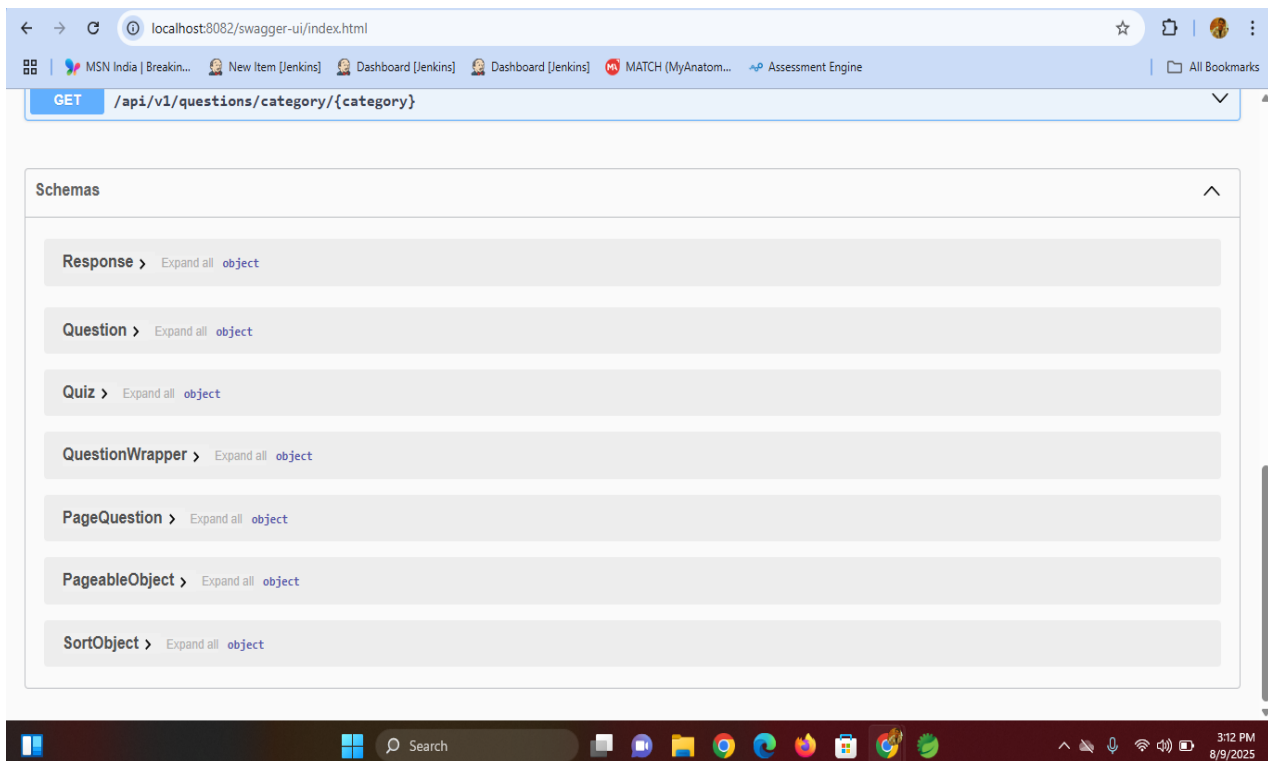
```
1 {
2   "id": 1,
3   "title": "Explain Git branching",
4   "option1": "Feature Branch",
5   "option2": "Main Branch",
6   "option3": "Release Branch",
7   "option4": "All of the above",
8   "correctAnswer": "All of the above",
9   "difficultyLevel": "MEDIUM",
10  "category": "GIT"
11 }
```

POSTMAN : PATCH



A screenshot of the Postman application interface. The top bar shows the URL `http://localhost:8082/` and the method `PATCH`. The request body is a JSON object: `{ "difficultyLevel": "MEDIUM" }`. The response is a `200 OK` status with a response time of `3.03 s` and a body size of `367 B`. The response body is a JSON object: `{ "id": 2, "title": "What is Spring Boot?", "option1": "Framework", "option2": "Language", "option3": "Database", "option4": "Library", "correctAnswer": "Framework", "difficultyLevel": "MEDIUM", "category": "SPRING_BOOT" }`. The interface includes a sidebar with Collections, Environments, Flows, and History. The bottom status bar shows the Postman logo, search bar, and system tray icons.

SWAGGER



A screenshot of the Swagger UI interface. The top bar shows the URL `localhost:8082/swagger-ui/index.html`. The interface displays a list of schemas under the heading "Schemas". The schemas are: `Response`, `Question`, `Quiz`, `QuestionWrapper`, `PageQuestion`, `PageableObject`, and `SortObject`. Each schema is expandable and is of type `object`. The interface includes a sidebar with a search bar and a list of schemas. The bottom status bar shows the Swagger UI logo, search bar, and system tray icons.

SWAGGER

The screenshot displays the Swagger UI interface for a REST API. The browser address bar shows the URL `localhost:8082/swagger-ui/index.html`. The top navigation bar includes links to MSN India, Jenkins, and other services. The main content area is titled **question-controller** and lists several API endpoints:

- GET** `/api/v1/quiz/getQuizByID/{id}`
- GET** `/api/v1/questions`
- POST** `/api/v1/questions`
- GET** `/api/v1/questions/{id}`
- DELETE** `/api/v1/questions/{id}`
- PATCH** `/api/v1/questions/{id}`
- GET** `/api/v1/questions/category/{category}`

Below the endpoints, there is a section for **Schemas** and a **Response** section with a dropdown menu set to `Expand all object`.

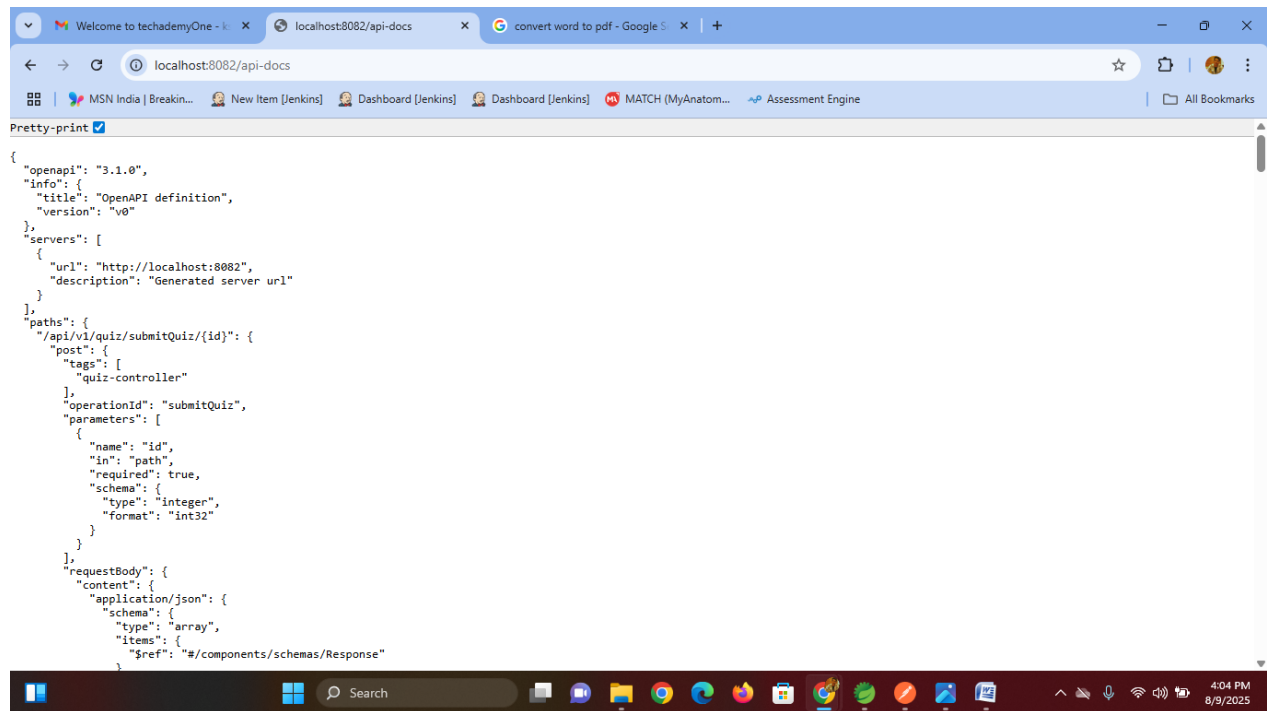
SWAGGER

The screenshot displays the Swagger UI interface for a REST API. The browser address bar shows the URL `localhost:8082/swagger-ui/index.html`. The top navigation bar includes links to MSN India, Jenkins, and other services. The main content area is titled **OpenAPI definition** with a version indicator `v0` and a `OAS 3.1` label. Below the title, there is a **Servers** section with a dropdown menu set to `http://localhost:8082 - Generated server url`. The main content area lists several API endpoints:

- POST** `/api/v1/quiz/submitQuiz/{id}`
- POST** `/api/v1/quiz/create`
- GET** `/api/v1/quiz/getQuizByID/{id}`
- question-controller**
 - GET** `/api/v1/questions`
 - POST** `/api/v1/questions`

The bottom of the screenshot shows the Windows taskbar with the time `3:12 PM` and date `8/9/2025`.

API-DOCS



The screenshot shows a web browser window with the address bar at `localhost:8082/api-docs`. The page displays a JSON OpenAPI definition. The definition includes a title "OpenAPI definition", a version "v0", and a server URL `http://localhost:8082`. A single path `/api/v1/quiz/submitQuiz/{id}` is defined with a POST method. The request body is an array of integers, and the response is a reference to `#/components/schemas/Response`.

```
{
  "openapi": "3.1.0",
  "info": {
    "title": "OpenAPI definition",
    "version": "v0"
  },
  "servers": [
    {
      "url": "http://localhost:8082",
      "description": "Generated server url"
    }
  ],
  "paths": {
    "/api/v1/quiz/submitQuiz/{id}": {
      "post": {
        "tags": [
          "quiz-controller"
        ],
        "operationId": "submitQuiz",
        "parameters": [
          {
            "name": "id",
            "in": "path",
            "required": true,
            "schema": {
              "type": "integer",
              "format": "int32"
            }
          }
        ],
        "requestBody": {
          "content": {
            "application/json": {
              "schema": {
                "type": "array",
                "items": {
                  "$ref": "#/components/schemas/Response"
                }
              }
            }
          }
        }
      }
    }
  }
}
```

POSTMAN : DELETE

